Tsinghua Science and Technology

Volume 22 | Issue 4 Article 8

2017

Detecting Isolate Safe Areas in Wireless Sensor Monitoring Systems

Chunyu Ai
Division of Mathematics & Computer Science, University of South Carolina Upstate, Spartanburg 29303, USA.

Frank Haizhon Li
Division of Mathematics & Computer Science, University of South Carolina Upstate, Spartanburg 29303, USA.

Kejia Zhang
College of Computer Science & Technology, Harbin Engineering University, Harbin 150001, China.

TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 08/09 pp427–436
Volume 22, Number 4, August 2017

Detecting Isolate Safe Areas in Wireless Sensor Monitoring Systems

Chunyu Ai, Frank Haizhon Li, and Kejia Zhang*

Abstract: Wireless sensors are deployed widely to monitor space, emergent events, and disasters. Collected real-time sensory data are precious for completing rescue missions quickly and efficiently. Detecting isolate safe areas is significant for various applications of event and disaster monitoring since valuable real-time information can be provided for the rescue crew to save persons who are trapped in isolate safe areas. We propose a centralized method to detect isolate safe areas via discovering holes in event areas. In order to shorten the detection delay, a distributed isolate safe area detection method is studied. The distributed method detects isolate safe areas during the process of event detection. Moreover, detecting isolate safe areas in a building is addressed particularly since the regular detecting method is not applicable. Our simulation results show that the distributed method can detect all isolate safe areas in an acceptable short delay.

Key words: sensor networks; isolate safe area; area detection; rescue

1 Introduction

Wireless sensors play an important role for Internet of Things to make collecting data and information of physical environment and objects possible[1, 2]. Nowadays, there are various cyber-physical systems being used for disaster preparedness, rescue, and emergency management[3–5]. This benefits from the advanced development of communication technology, embedded computing technology, and sensing technology. Sensors that have the capabilities of sensing, computation, and communication are emerging all over the world. Sensor networks consisting of different kinds of sensors are capable of gathering a large amount of information such as temperature, humidity, and light intensity, at any time, any place, and under any circumstances[6]. Therefore, sensors are important elements for cyber-physical systems to create connections between physical and virtual worlds. Sensor networks are widely deployed to monitor surrounding environment and detect various events in military fields, national security, traffic control, health, environmental monitoring, industry, and disaster prevention and recovery. In recent years, researchers have proposed various data collection, query processing, event detection, and monitoring systems since wireless sensor monitoring systems are significant for various fields[7–14]. However, most proposed systems in the literature focus only on how to detect various events and delivery alarms. Our focus centers on providing real-time and accurate information for rescue missions when a dangerous event happens.

An Intelligent Monitoring System (IMS) can be widely used for various monitored environments and monitoring missions such as coal mine monitoring[15, 16], underground structure monitoring[17], and fire detection and rescue[18, 19]. For instance, an IMS can be used for monitoring fires in a building as shown in Fig. 1. The system can detect a fire according to abnormal temperature and smoke intensity reports sensed by sensors. Then, a fire alarm is triggered automatically to inform people in the building to

• Chunyu Ai and Frank Haizhon Li are with Division of Mathematics & Computer Science, University of South Carolina Upstate, Spartanburg 29303, USA. E-mail: aic@uscupstate.edu; fli@uscupstate.edu.
• Kejia Zhang is with College of Computer Science & Technology, Harbin Engineering University, Harbin 150001, China. E-mail: kejiazhang@hrbeu.edu.cn.
∗ To whom correspondence should be addressed.
Manuscript received: 2016-11-26; revised: 2017-01-25; accepted: 2017-02-12

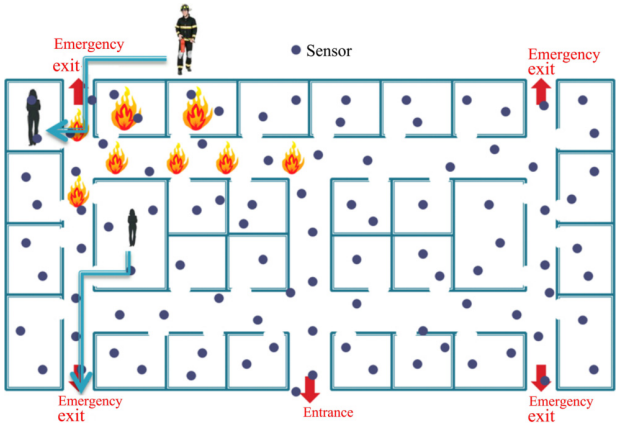


Fig. 1 Fire detection and rescue.

evacuate and notify firefighters to come to the scene. Since sensors can sense environmental changes and activities of human beings, an IMS can collect and integrate useful information from sensor networks in a real-time manner and generate a real-time fire map which can reflect the real circumstances in the building. The people in the building can use a smart phone, tablet, or computer to ask for a safe escape route. Through accessing IMS, the fire department can find the locations of trapped persons and the most efficient route to implement rescue. An IMS can be used for various fields to prevent and reduce loss of life and property when dangerous events happen. For example, safety is always a big concern in the coal industry. Gas leak and collapse accidents have caused so many worker injures and fatalities. An IMS can help rescue crew locate trapped workers and accomplish rescue missions efficiently.

After analyzing the common features of events, we conclude that finding the locations of trapped persons is the key of rescue. An isolate safe area is a safe area which is contained by a dangerous area. Detecting isolate safe areas is significant for rescue missions since there is a big probability that people are trapped in these areas when a dangerous event happens. Also, these isolate safe areas are just temporarily safe because dangerous areas usually spread quickly such as fire. Therefore, detecting isolate safe areas and informing rescue crew of the locations in a short time are significant.

In this paper, we address the isolate safe area detection issue. The rest of the paper is organized as follows. Section 2 introduces related work. Our proposed centralized and distributed detection methods are addressed in Section 3. Section 4 shows our simulation results, and Section 5 concludes the paper.

2 Related Work

Event detection of wireless sensor networks has been widely studied by researchers. In Ref. [20], a framework is designed to dynamically monitor building fires based on Internet of Things. In Ref. [21], events are defined by spatial-temporal patterns. In the scheme, events are effectively detected via matching contour maps of sensed data to event patterns. In Ref. [7], algorithms are proposed to detect event boundary and faulty sensors. The proposed algorithms can be used to the outlier detection and regional data analysis. A distributed grid-based event detection scheme is proposed in Ref. [11]. The whole network is partitioned into grids, and sensor nodes in each grid form a cluster. Event detection of a cluster is based on the readings of its member sensor nodes. When a cluster head receives an alarm from its members, event detection process is initiated. A cluster head cooperates with its neighboring clusters to detect events. In Ref. [9], a distributed method of dynamic event region detection is studied. Dynamic Markov random fields are used to model the spatio-temporal relationship of the evolving regions. To improve the performance loss of conventional quickest change detection methods, this work uses the system dynamics to predict the field evolution and proposes a distributed event region detection algorithm using mean field approximation. A hybrid detection scheme is also introduced to improve the performance for rarely occurred events. Reference [22] proposes an area query processing scheme which can retrieve not only sensing values but also specific geographical information compared to traditional queries. Area Query is more practicable and useful than traditional queries for some applications which require geographical information. Area query also can be used to detect events via writing event conditions in a query.

In Ref. [18], a cellular-automata-like algorithm and an averaging consensus algorithm are proposed specifically for fire detection and localization with sensor networks. When fire is detected in the network, the algorithm notifies all sensor nodes rapidly. Then, the algorithm predicts the parameters of the circle surrounding the fire location. The proposed method can detect fire rapidly and monitor the extension of the fire in real-time manner. Also, every live sensor holds the information of the fire outbreak and extension. In Ref. [19], a fire rescue architecture,

FireNet, is proposed. System requirements including accountability of firefighters, real-time monitoring, intelligent scheduling and resource allocation, and Web-enabled service and integration are discussed. Implementation challenges of this system are also addressed in the work.

Above mentioned event detection methods can detect user predefined events efficiently and notify users in a short delay; however, these methods are not applicable for isolate safe area detection.

3 Detecting Isolate Safe Area

In many applications, users expect information about areas of their interests instead of sensed values of individual sensor nodes. Users can write a query to describe desired area results by specifying area size and conditions of sensing attributes. Only an area which has every sensor node in it satisfied all attribute conditions can be returned as results.

3.1 Preliminaries

Our previous work[22] proposed an energy-efficient in-network area query processing scheme which can answer area queries with low energy consumption. In order to process area queries efficiently, the monitored area is partitioned into grids, and each grid is assigned a unique gray code number as its Grid ID (GID) as shown in Fig. 2. A GID is formatted as (X, Y) where X represents a gray code number in horizontal direction, and Y represents a gray code number in vertical direction. The GID of the grid at the top-right corner is $(100, 000)$. The grid at the bottom-right corner is $(100, 10x)$, where x means either 0 or 1 indicating the union of $(100, 101)$ and $(100, 100)$. Gray code based area description can significantly reduce the size of query results compared to other description methods since one gray code can represent multiple grids, thus saving energy consumption during transmission of results or partial results. To perform in-network query processing, a reporting tree is constructed by the base station. In Fig. 2, a reporting tree example is shown. An area query is processed in bottom-up manner along the reporting tree. An internal node receives the reports from its children and merges the reports to generate partial results, then sends to its parent.

This scheme can detect dangerous areas when an event happens. Intuitively, we can use this scheme to detect dangerous areas, then detect isolate safe areas by judging whether there are holes in a dangerous area.

3.2 Centralized isolate safe area detection

In order to discover isolate safe areas, an $m \times n$ matrix $\boldsymbol{A}$ (m rows and n columns) is created to help the processing. Each element of the matrix $\boldsymbol{A}$ represents a grid of the monitored area. The base station of wireless sensor networks can send an event query to find all grids which have the event happened in it. As shown in Fig. 3, the shaded grids are affected by the event. According

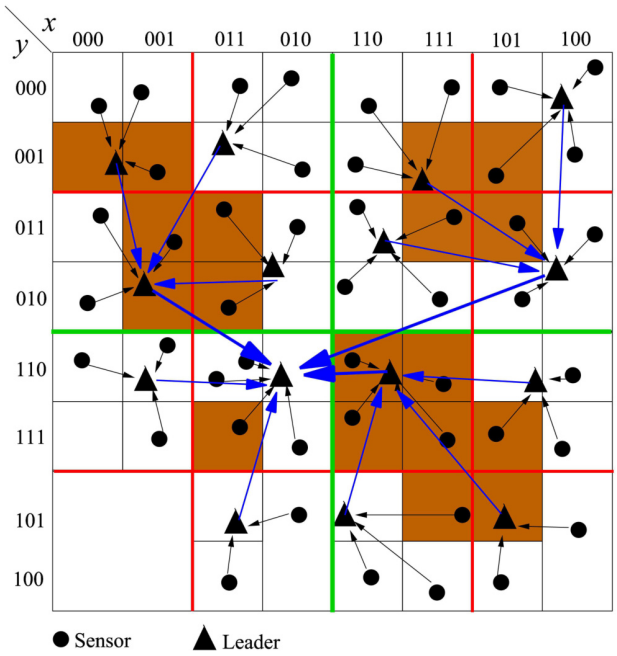


Fig. 2 Area query processing.

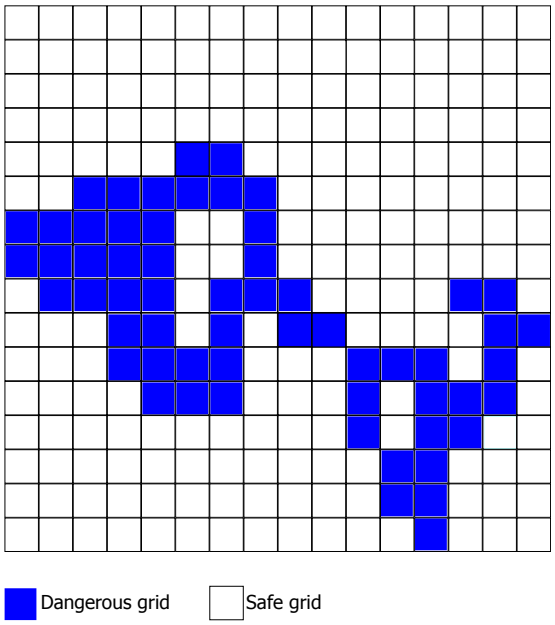


Fig. 3 Centralized isolate safe area detection example.

to the event query results, the matrix element of a safe grid is set to 0, and the matrix element of a dangerous grid (affected by the event) is set to -1. Neighbors of a grid are defined as the adjacent grids located at above, below, left, and right of the current one. As shown in Fig. 4, the shaded grids are neighboring grids of the stripped grid. Only neighboring grids with the same status (either safe or dangerous) can be merged to generate a bigger area.

Definition 1 Isolate Safe Area An isolate safe area is a continuous area which is surrounded by a dangerous area.

In the example of Fig. 3, there are two isolate safe areas. We assume safe grids located at boundary of the whole monitored area cannot be a grid of an isolate safe area. This assumption is applicable to a lot of applications.

Algorithm 1 is proposed to detect isolate safe areas. The basic idea is to mark each safe grid with an either safe or safe_isolate area_id. The area_id of a grid is stored in the corresponding element in matrix $\boldsymbol{A}$. Since safe grids on the boundary must be a part of a safe area, we start to process all grids on the boundary, then move to the inner grids next to boundary grids until the center is reached. The order of processing grids is shown in Fig. 5. Initially, two empty lists, $List_{safe}$ and $List_{isolate\_safe}$, are created to store area_ids (starting at 1) of safe areas and isolate safe areas, respectively. First, we mark each safe grid on the boundary with an area_ids, and add the area_id to $List_{safe}$ if it is new. If the current processing safe grid has a processed safe neighboring grid, the neighbor's area_id is used to mark the current grid; otherwise, use a new area_id to mark it. For inner safe grids, if the current processing grid is a dangerous grid, we do not change its value; otherwise, we use one of its processed safe neighboring grid's area_id to mark it. If it has no processed safe

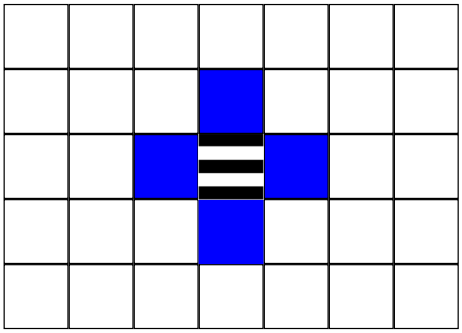


Fig. 4 Neighbors of a grid.

Algorithm 1 Centralized Isolate Safe Area Discovery

Input: Matrix $\boldsymbol{A}$
Output: Isolate safe areas

$id_{area} = 0$
set $List_{safe}$ to empty
set $List_{isolate\_safe}$ to empty
set $pre\_id = 1$
add $\{1\}$ into $List_{safe}$
for every element in Matrix $\boldsymbol{A}$, scan it from boundary to center as the order shown in Fig. 5 **do**
  if the current scanning element $A[i, j]$ is not -1 where i is the row number and j is the column number **then**
    if $A[i, j]$ is on the boundary, that is, (i is 1 or m) and (j is 1 or n) **then**
      if pre_id is not -1 **then**
        $A[i, j] = pre\_id$
      else
        $id_{area}++$
        $A[i, j] = id_{area}$
        $pre\_id = A[i, j]$
        add $\{id_{area}\}$ into $List_{safe}$
      end if
    else
      if pre_id is in $List_{safe}$ **then**
        $A[i, j] = pre\_id$
      else
        if there exists a processed neighbor is marked with a safe area_id id **then**
          $A[i, j] = id$
        else if pre_id is not -1 **then**
          $A[i, j] = pre\_id$
        else if $A[i, j]$ has a processed neighbor is marked with an isolate safe area_id id **then**
          $A[i, j] = id$
        else
          $id_{area}++$
          $A[i, j] = id_{area}$
          add $\{id_{area}\}$ into $List_{isolate\_safe}$
        end if
      end if
      $pre\_id = A[i, j]$
      if $A[i, j]$ is in $List_{safe}$ **then**
        for each value id of any processed neighbor of $A[i, j]$ is in $List_{isolate\_safe}$ **do**
          remove the sublist which id belongs to from $List_{isolate\_safe}$ and add it to $List_{safe}$
        end for
      end if
    end if
  end if
  merge area_ids of $A[i, j]$ and its processed non-dangerous neighbor grids in either $List_{safe}$ or $List_{isolate\_safe}$
end for
return $List_{isolate\_safe}$, each sublist in it represents an isolate area

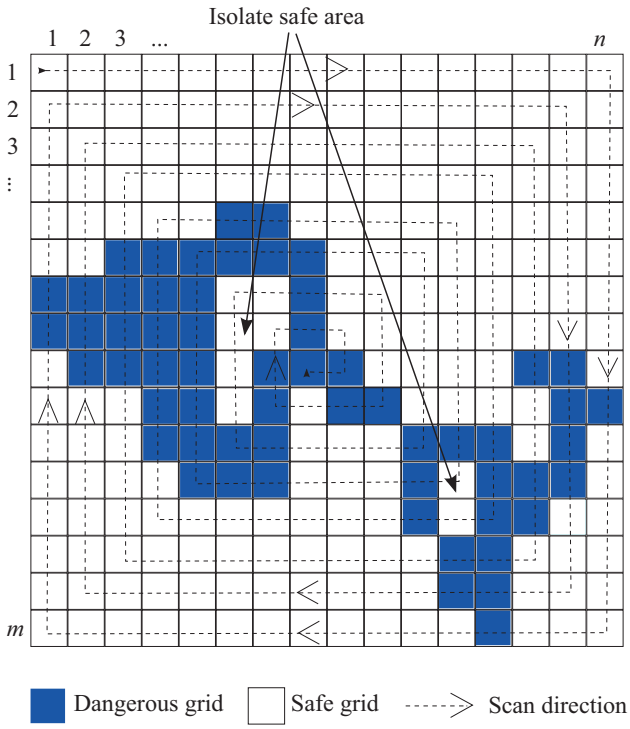


Fig. 5 Centralized isolate safe area detection example.

neighboring grid, we use its processed isolate safe neighboring grid's area_id to mark it. If all of its processed grids are dangerous, we mark it with a new area_id, and add this new area_id to $List_{isolate\_safe}$.

Possibly, neighboring safe grids are marked with different area_ids even though they belong to the same safe or isolate safe area; so after we marked a safe grid, if the current processing grid's area_id is different from area_ids of its neighboring safe grids, we unite the sublists which these area_ids belong to as one. If this union is between a sublist in $List_{safe}$ and a sublist in $List_{isolate\_safe}$, remove the sublist from $List_{isolate\_safe}$ and add it to $List_{safe}$ first, then union them. Thus, after marking every safe grid, each sublist in $List_{isolate\_safe}$ can be used to identify one isolate safe area.

Centralized isolate safe area detecting is easy to be implemented on the base station, and its time complexity is $O(m \times n)$; however, the base station has to wait for the sensor nodes to finish processing the event query, then run Algorithm 1 to discover isolate safe areas. Therefore, the time delay of centralized method might not be acceptable for a serve time sensitive rescue mission. One way to shorten the time delay is to design an in-network detecting method, that is, discovering isolate safe areas during processing the event query.

3.3 Distributed isolate safe area detection

In order to shorten the response delay of isolate safe area detection, we design a distributed method to discover isolate safe areas during the in-network processing of event queries. The motivation of distributed design is to discover isolate safe areas as early as possible and send the locations of these areas to the rescue crew. We use the query scheme in our previous work introduced briefly in Section 3.1. As shown in Fig. 2, when a sensor node senses a value satisfied the predefined condition of an event, it sends the report to its parent in the reporting tree. An internal node in the reporting tree receives the reports from its children, then merges the reports to generate a partial result for the subarea it covers (the subarea includes all grids represented by sensor nodes in the subtree rooted at the current internal node); then, it sends the subarea merging result to its parent. The whole query is processed from the bottom to the top of the reporting tree.

In order to discover isolate safe areas as early as possible, we also process along the reporting tree from the bottom to the top. For each internal node, its responsibility is to discover isolate safe areas within the subarea it covers. Here, an isolate safe area within a subarea means that any grid of this isolate area cannot be located at the boundary of the subarea. The grids at the boundary have a chance to be merged with grids in its adjacent subareas, thus whether it is an isolate safe area or not and its range cannot be determined at the current stage; so these will be saved for the current node's ancestor to judge. When an internal node discovers an isolate safe area within its subarea, it will send the report of the isolate safe area to the base station directly instead of sending along the reporting tree, thus guaranteeing the isolate safe areas can be reported immediately.

The method of detecting an isolate safe area in a subarea is described in Algorithm 2. Algorithm 2 is run on each internal node of the reporting tree. Each internal node also uses a matrix $\boldsymbol{A}$ to record status of each grid within its subarea. The element $A[i, j]$ is -1 if the grid is dangerous; otherwise, $A[i, j]$ is 0. The current internal node updates matrix $\boldsymbol{A}$ according to the reports sent by its children.

As shown in Fig. 6, the smallest subarea covered by the direct parent of leaf nodes has only 4 grids. It is not necessary to run detecting algorithm since there is

Algorithm 2 Distributed Isolate Safe Area Discovery

Input: Matrix A
Output: Isolate safe areas

$id_{area} = 0$
create an array $List_{isolate\_safe}$ with four lists in it and initialize all lists to empty
$i = 0$
for each subarea **do**
  $next\_round = true$
  for each round scanning **do**
    if *continue* is not *true* **then**
      break
    else
      if the current processing element $A[i, j]$ is a potential isolate grid and $A[i, j]$=0 (safe) **then**
        if $A[i, j]$ has no processed safe neighboring grids in its subarea **then**
          $id_{area}++$
          $A[i, j] = id_{area}$
          add $\{id_{area}\}$ to $List_{isolate\_safe}[i]$
        else
          use its processed safe neighboring gird's value to set $A[i, j]$
        end if
      else if $List_{isolate\_safe}[i]$ is empty **then**
        break
      else if $A[i, j] = 0$ and $A[i, j]$ is a boundary grid **then**
        if any of its processed neighboring grid value $id >= 1$ **then**
          remove the sublist contains id from $List_{isolate\_safe}[i]$
        end if
      else if $A[i, j] = 0$ **then**
        if the processed neighboring grid value $id >= 1$ and id is in a sublist of $List_{isolate\_safe}[i]$ **then**
          $A[i, j] = id$
          $next\_round = true$
          merge area_ids of $A[i, j]$ and its processed non-dangerous neighbor grids in $List_{isolate\_safe}[i]$
        end if
      end if
    end if
  end for
  $i++$
end for
if there are non-empty list in $List_{isolate\_safe}$ **then**
  merge discovered isolate areas if possible
  use the method in Ref. [22] to generate a GID lists description for each isolate area and send to the base station
end if

no isolate safe area to be found. The rationale is that an isolate safe area must be surrounded by dangerous grids in all above, below, left, and right directions. The

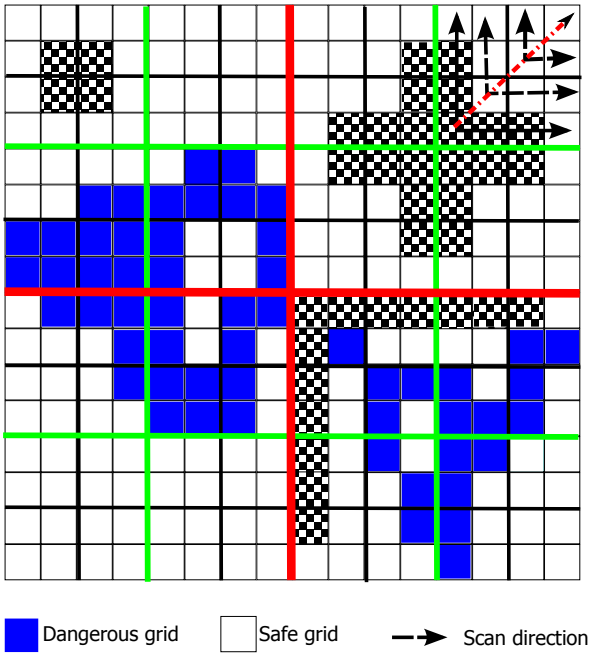


Fig. 6 Distributed isolate safe area detection.

next level internal nodes have 16 grids in its covered area including 4 subareas covered by its 4 children respectively. Figure 6 shows an example at the top-left corner. Only 4 grids shaded with checkerboard can be part of an isolate area detected in this phase, thus we just need to start scanning these 4 grids. For the next level internal nodes, they have 64 grids in their covered area. An example is shown at the top-right corner in Fig. 6. If there is an isolate area, one of its grid must be in the grids shaded with checkerboard. So, if we start scanning from these shaded grids, it is guaranteed that all isolate safe areas can be discovered. The isolate areas (which do not include these shaded grids) do not span two subareas, thus they must be found and reported already in its child node. For each internal node, we define these grids with shaded checkerboard as potential isolate grids. The potential isolate grids of an area are these grids which are adjacent to vertical and horizontal center lines but except these grids also located at the boundary. In Fig. 6, for the whole area, the grids shaded with checkerboard in the bottom-right corner are potential grids in bottom-right subarea.

According to analysis results, for each internal node, we can start scanning from potential isolate grids to discover all isolate safe areas. In each processing area, we divide the area into four subareas, that is, four subareas in which its four children covered. In each subarea, we start scanning grids adjacent to the vertical

and horizontal center lines, then scan grids adjacent to the grids scanned in the previous round, and so on until scanning all grids in the subarea. An example of the scanning order is shown in Fig. 6 at top-right corner.

In order to reduce the running time, whenever the current scanning round has no grid added to any candidate isolate areas, isolate safe areas are discovered and the process is terminated. Also, if all candidate isolate areas are excluded since they are directly or indirectly merged with boundary grids, the process is done, and no isolate safe areas are found. After scanning all four subareas and discovering all candidate isolate areas, merge these candidates if possible. Then, use the gray code based method in Ref. [22] to generate an isolate safe area information report and send to the base station. The reason we use the gray code based description is that the report size is smaller than other description methods, thus saving transmission energy consumption and also achieving a shorter transmission delay. For each internal node, the time complexity of detecting algorithm is $O(m \times n)$ where $m \times n$ is the number of grids the current internal node covers.

3.4 Incremental detection

When an event occurs, we always want the system report real-time information of events. For rescue, we need the size and location of isolate safe areas frequently. Therefore, the system needs to report to the user frequently. In most applications, areas with events will gradually spread or shrink. In other words, isolate safe areas surrounded by event areas will shrink or extend gradually too. Due to the feature that isolate safe area spreads or shrinks gradually, we proposed an incremental detection method to improve performance of detection.

We now present an incremental updating scheme to detect isolate safe areas on non-leaf nodes of the reporting tree after the initial results are calculated. Each sensor node stores its previous report, and non-leaf nodes store the previous matrix result. For each grid, we compare its current and previous values: if the current and the previous values are the same, we do not need to do any special processing; if the previous value is safe and the current value is dangerous, we call this negative grid; if the previous value is dangerous and the current value is safe, we call this positive grid.

For a non-leaf node in the reporting tree, negative grids might generate new isolate safe areas. If so, we just need to scan safe neighboring grids of each negative grid to detect a potential new generated isolate safe area. If these safe neighboring grids do not connect to a safe area directly or indirectly, a new isolate safe area needs to be reported. Positive grids might make a previous isolate safe area become safe; for this situation, we need to see whether a positive grid connects a safe and an isolate area, if so, that isolate can be removed from the report. Positive grids can also generate a new isolate safe area which just includes itself. We can judge by checking whether the positive grid has a safe neighboring node, if not, a new isolate safe area is generated and reported.

Increment detection can process the positive and negative grids locally in a small area to generate results based on the previous results; therefore, the detection time can be reduced efficiently.

3.5 Isolate safe area detection in a building

The method introduced above cannot be directly used to detect isolate safe areas in a building since walls and dangerous areas together can generate an isolate safe area too. For instance, the room at top-left corner in Fig. 1 is an isolate safe area because its only exit is next to a fire area. In order to detect isolate safe areas in a building, we modify the proposed algorithm as follows:

(1) The safe grids located at the boundary of the whole monitored area cannot be used to mark its adjacent safe area non-isolate since the boundary is inside walls of the building.

(2) Only the safe grid located at an exit can claim that the safe area it belongs to is non-isolate.

(3) An safe area in a room is not isolate only if it is adjacent to a non-isolate safe area which includes at least one of the room's exits. For grids in each room, we will assign a unique room number and maintain a list of the room exit grids. A grid with a room number can only merge with other grids with the same room number or the grids in the exit list. Thus, we can avoid merging areas separated by walls.

The above three rules guarantee that isolate safe areas in a building are detected accurately.

The method we proposed is applicable in two dimensional monitored areas. However, for buildings with multiple floors, our method cannot be applied directly. We can apply the proposed method on each floor separately. After generating results of each floor, we can merge results according to the stairs and

elevators which connect exits of adjacent floors.

3.6 Improving reliability

Some events and disasters such as fire can destroy sensors. Therefore, in order to improve reliability of the system, we design an urgent mode for sensor nodes. If a sensor node does not receive acknowledgement of sent messages constantly or it cannot hear its neighbors anymore, the urgent mode is triggered. In the urgent mode, the sensor node keeps increasing its power level of sending messages until it can receives acknowledgement normally. For an internal node in the reporting tree, if it does not receive reports from its child for a period of time longer than a predefined duration threshold, it marks the grid covered by that child as dangerous grid. Losing some sensor nodes can also make the reporting tree disconnected. For any inner node of the reporting tree, if it loses a child, it will probe its grandchildren to find a replacement. If no replacement can be found, then that subarea is marked as dangerous area permanently.

4 Simulation Results

In this section, we evaluate the performance of our proposed centralized and distributed isolate safe area detection methods. When a severe event happens, the time delay is the biggest concern, thus we focus on evaluating the time delay of those two proposed methods.

We simulate the event query and isolate safe area detection of a coal mine. The monitored area size is 64×64 grids. The payload size limit of a packet is 29 bytes which are the standard payload sizes provided by the TinyOS[23]. The transmission range of sensor nodes is 30 m, and the sensing range is 15 m. DSDV[24] is used as the routing protocol. We randomly generate 10 event and isolate safe area distribution scenarios to test the performance for each parameter value, and the results is the average of those 10 tests.

First, we evaluate the time delay with different percentages of the event area. The event area includes both dangerous grids and grids in isolate safe areas. The time delay with different percentages of the event area is shown in Fig. 7. The response delay is the average detection time of all isolate safe areas. As seen in the figure, the distributed detection method has an obvious shorter average delay than the centralized method. The reason is that some isolate safe areas can be detected at the lower levels of the reporting tree and results are sent to the base station directly. Also, the time delay increases with the increasing of the percent of event area because more event grids mean more message transmissions.

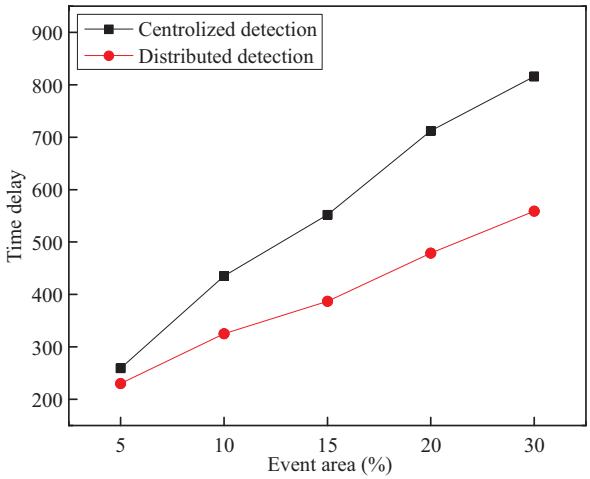


Fig. 7 Time delay with different percentages of the event area.

Figure 8 shows the time delay with different numbers of isolate safe areas. The percent of the event area is 20%. As shown in the figure, the delay of the centralized method increases with the number of isolate safe areas because more isolate safe areas inside event areas make the size of the description of event areas bigger thus causing more transmissions. The distributed method is affected by this too but not as obvious. The reason is that when the number of isolate safe areas is bigger, more of them can be detected at the lower levels of the reporting tree.

Overall, the distributed detection method performs significantly better than the centralized method in terms

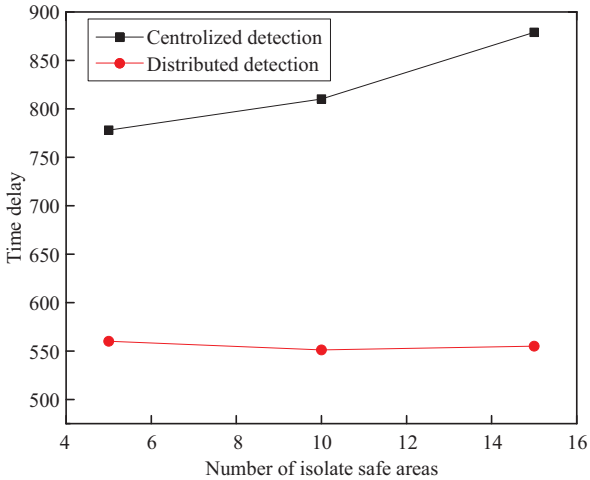


Fig. 8 Time delay with different numbers of isolate safe areas.

of detection delay. However, the distributed method causes some internal nodes of the reporting tree to spend more energy for detection compared to the centralized method. Consuming more energy is worth while to gain a shorter delay, thus more people and properties can be saved.

5 Conclusion

In order to detect isolate safe areas to save trapped persons, we proposed a centralized detection method and a distributed detection method, respectively. The centralized version is easy to be implemented; however, its response delay is longer than the distributed method. This is verified by our simulation results. Moreover, detecting isolate safe areas in a building is a special case, and we addressed this issue in our work as well. We also defined an urgent mode for sensor nodes to improve reliability during emergent events or disaster.

References

[1] D. Christin, A. Reinhardt, P. S. Mogre, and R. Steinmetz, Wireless sensor networks and the internet of things: Selected challenges, in *Proceedings of the 8th GI/ITG KuVS Fachgespräch Drahtlose Sensornetze*, 2009, pp. 31–34.

[2] The role of sensor fusion in the internet of things. http://www.mouser.com/applications/sensor-fusion-iot/, Accessed on Nov. 2, 2016.

[3] E. Gelenbe and F.-J. Wu, Future research on cyber-physical emergency management systems, *Future Internet*, vol. 5, no. 3, pp. 336–354, 2013.

[4] J. Liu, C.-S. Shih, and E.-H. Chu, Cyberphysical elements of disaster-prepared smart environments, *Computer*, vol. 46, no. 2, pp. 69–75, 2013.

[5] F.-J. Wu, Y.-F. Kao, and Y.-C. Tseng, From wireless sensor networks towards cyber physical systems, *Pervasive and Mobile Computing*, vol. 7, no. 4, pp. 397–413, 2011.

[6] T. Arampatzis, J. Lygeros, and S. Manesis, A survey of applications of wireless sensors and wireless sensor networks, in *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*, 2005, pp. 719–724.

[7] M. Ding, D. Chen, K. Xing, and X. Cheng, Localized fault-tolerant event boundary detection in sensor networks, in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 2005, pp. 902–913.

[8] Z. Cai, S. Ji, and J. Li, Data caching-based query processing in multi-sink wireless sensor networks, *IJSNet*, vol. 11, no. 2, pp. 109–125, 2012.

[9] T. Wu and Q. Cheng, Distributed dynamic event region detection in wireless sensor networks, in *Prognostics and Health Management (PHM), 2011 IEEE Conference on*, 2011, pp. 1–8.

[10] J. Li, S. Cheng, H. Gao, and Z. Cai, Approximate physical world reconstruction algorithms in sensor networks, *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3099–3110, 2014.

[11] Y.-H. C. Ja Won Ko, A grid-based distributed event detection scheme for wireless sensor networks, *Sensors*, vol. 11, no. 11, pp. 10048–10062, 2011.

[12] S. Cheng, Z. Cai, and J. Li, Curve query processing in wireless sensor networks, *IEEE Transactions on Vehicular Technology*, vol. 64, no. 11, pp. 5198–5209, 2015.

[13] S. Cheng, Z. Cai, J. Li, and H. Gao, Extracting kernel dataset from big sensory data in wireless sensor networks, *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 4, pp. 813–827, 2017.

[14] X. Zheng, Z. Cai, J. Li, and H. Gao, A study on application-aware scheduling in wireless networks, *IEEE Transactions on Mobile Computing*, doi: 10.1109/TMC.2016.2613529.

[15] X. Wang, X. Zhao, Z. Liang, and M. Tan, Deploying a wireless sensor network on the coal mines, in *Networking, Sensing and Control, 2007 IEEE International Conference on*, 2007, pp. 324–328.

[16] W. Yang and Y. Huang, Wireless sensor network based coal mine wireless and integrated security monitoring information system, in *Proceedings of the Sixth International Conference on Networking*, Washington, DC, USA, 2007, pp. 22–28.

[17] Z. Li and Y. Liu, Underground structure monitoring with wireless sensor networks, in *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, 2007, pp. 69–78.

[18] A. Khadivi and M. Hasler, Fire detection and localization using wireless sensor networks, in *Sensor Applications, Experimentation, and Logistics*, N. Komninos, ed. Springer Berlin Heidelberg, 2010, vol. 29, pp. 16–26.

[19] K. Sha, W. Shi, and O. Watkins, Using wireless sensor networks for fire rescue applications: Requirements and challenges, in *Electro/information Technology, 2006 IEEE International Conference on*, 2006, pp. 239–244.

[20] X. Ma, Q. Huang, Q. Liu, X. Shu, and Q. Zhao, Dynamic monitoring of building fires based on internet of things (in Chinese), *Journal of Tsinghua University (Science and Technology)*, vol. 52, no. 11, pp. 1584–1590, 2012.

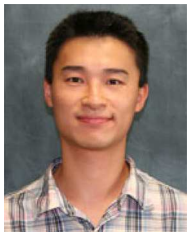
[21] W. Xue, Q. Luo, L. Chen, and Y. Liu, Contour map matching for event detection in sensor networks, in *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2006, pp. 145–156.

[22] C. Ai, L. Guo, Z. Cai, and Y. Li, Processing area queries in wireless sensor networks, in *Mobile Ad-hoc and Sensor Networks, 2009. MSN '09. 5th International Conference on*, 2009, pp. 1–8.

[23] Tinyos faq, Available: http://tinyos.stanford.edu/tinyos-wiki/index.php/FAQ, Accessed on Nov. 2, 2016.

[24] C. E. Perkins and P. Bhagwat, Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers, *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 4, pp. 234–244, 1994.

Chunyu Ai received the bachelor and master degrees from Heilongjiang University in 2001 and 2004, respectively. She received the PhD degree in computer science from Georgia State University in 2010. Dr. Ai is currently an assistant professor of Computer Science at University of South Carolina Upstate. Her research interests include wireless sensor networks, data management, and social networks.



Frank Haizhon Li received the MS and PhD degrees from University of Memphis in 2002 and 2005, respectively. Prior to pursuing the career in computer science, he has worked as a chemical engineer for a number of years. Dr. Li is an associate professor of Computer Science at University of South Carolina Upstate. His research interests include wireless networks and mobile computing, computer security, and artificial intelligence applications in education and smart home.

Kejia Zhang received the BS degree in information management from Beijing Normal University, China, in 2004, and MS and PhD degrees in computer science & technology from Harbin Institute of Technology, China, in 2007 and 2012, respectively. He is currently an assistant professor of Computer Science & Technology at Harbin Engineering University, China. His current research interests include data privacy, wireless sensor networks, and Internet of things.