Underground Coal Mine Monitoring with Wireless Sensor Networks

MO LI and YUNHAO LIU
Hong Kong University of Science and Technology

Environment monitoring in coal mines is an important application of wireless sensor networks (WSNs) that has commercial potential. We discuss the design of a Structure-Aware Self-Adaptive WSN system, SASA. By regulating the mesh sensor network deployment and formulating a collaborative mechanism based on a regular beacon strategy, SASA is able to rapidly detect structure variations caused by underground collapses. We further develop a sound and robust mechanism for efficiently handling queries under instable circumstances. A prototype is deployed with 27 mica2 motes in a real coal mine. We present our implementation experiences as well as the experimental results. To better evaluate the scalability and reliability of SASA, we also conduct a large-scale trace-driven simulation based on real data collected from the experiments.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Distributed networks; Network communications*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

General Terms: Algorithms, Design, Measurement

Additional Key Words and Phrases: Wireless sensor networks, structure monitoring, underground, coal mine

ACM Reference Format:

Li, M. and Liu, Y. 2009. Underground coal mine monitoring with wireless sensor networks. ACM Trans. Sensor Netw., 5, 2, Article 10 (March 2009), 29 pages. DOI = 10.1145/1498915.1498916 http://doi.acm.org/10.1145/1498915.1498916

1. INTRODUCTION

A Wireless Sensor Network (WSN) is a self-organized wireless network composed of a large number of sensor nodes that interact with the physical world

This work is supported in part by the Hong Kong RGC grant HKUST6169/07E, the National Basic Research Program of China (973 Program) under grant No. 2006CB303000, the National High Technology Research and Development Program of China (863 Program) under grant No. 2007AA01Z180, and Key Project Grant No. 60533110.
Authors' address: Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong; email: {limo,liu}@cse.ust.hk.

DOI 10.1145/1498915.1498916 http://doi.acm.org/10.1145/1498915.1498916

[Akyildiz et al. 2002]. Various low-power and cost-effective sensor platforms have been developed based upon recent advances in wireless communication and microsystem technologies. The increasing study of WSNs [Chakrabari et al. 2004, Vural and Ekici 2005, Wan et al. 2005] aims to enable computers to better serve people by automatically monitoring and interacting with physical environments.

Environment monitoring in underground tunnels, which are usually long and narrow, with lengths of tens of kilometers and widths of several meters, has been a crucial task to ensure safe working conditions in coal mines where many environmental factors, including the amount of gas, water, and dust, need to be monitored. To obtain a full-scale monitoring of the tunnel environment, sample data need to be collected at many different places. A precise environment overview requires a high sampling density, which involves a large number of sensing devices. Current methods of coal mine environment monitoring are typically conducted in a sparse and manual way, due to the lack of techniques for constructing an automatic large-scale sensing system.

Utilizing wires to connect sensing points to the processing server requires a large amount of wire deployment, which is difficult because of poor working conditions and high maintenance costs underground. Moreover, the wired communication method makes the system less scalable; as the tunnel advances, more sensing devices need to be deployed. In this situation a wireless system takes advantage of convenient deployment and flexible adjustment. Due to the unpredictable interference caused by the proximity of working machines and miners, however, it is often impossible to maintain direct wireless communication channels between sensing devices and the processing server. The long and narrow tunnels also make direct wireless communications unfeasible.

WSNs employ multi-hop routing to implement data gathering. Each sensor node plays the role of data collector as well as message forwarder in the network. The utilization of a WSN to implement the monitoring system benefits from rapid and flexible deployment. Additionally, the multi-hop transmitting method conforms to the tunnel structure and provides more scalability for system construction.

The unstable nature of geological construction in coal mines makes underground tunnels prone to structural changes. This instability, which could result in collapses caused by mine quakes or coasts, renders previous WSN monitoring solutions unfeasible. Among the 480 coal mine fatalities reported in the past 10 years in the U.S., collapses account for more than 50%. Most fatalities are the result of small collapses caused by falling roof or walls. Hence, it is of great importance to quickly detect collapse hole regions and accurately provide location references for workers. Since a collapse may destroy part of a monitoring system, maintaining the validity of the network in extreme situations is a challenge, which is rarely encountered in other WSN applications.

In this article, we present a **S**tructure-**A**ware **S**elf-**A**daptive sensor system, SASA, which aims to address the challenges and provide a feasible framework for underground monitoring in coal mines. The design objectives of SASA include: (1) to rapidly detect the collapse area and report to the sink node; (2) to maintain the system integrity when the sensor network structure is altered;

and (3) provide a sound and robust mechanism for efficiently handling queries over the sensor network under unstable circumstances.

SASA employs a hole-detection algorithm to monitor the inner surface of tunnels by utilizing the radio signals among sensor nodes to model the structure of the sensor network. With an appropriate arrangement of sensor nodes and a collaborative mechanism, SASA is able to accurately report locations of collapses, to detect and to reconfigure displaced nodes, thus maintaining the system integrity. SASA adopts a multi-path routing scheme for data collection; and by signature-file-based data aggregation SASA is able to accurately and efficiently route back information even under the influence of collapse holes.

We conducted field studies in the D. L. coal mine and deployed a prototype system, which included 27 Crossbow Mica2 motes [Hill and Culler 2002]. Due to resource and environment constraints, our prototype is limited in size. To better evaluate its scalability, we launched a large-scale trace-driven simulation with the real data collected from the prototype implementation.

The rest of this article is organized as follows. Section 2 discusses related work. Section 3 introduces the underground coal mine environment. Section 4 presents design details of SASA. Section 5 presents the performance evaluation through both trace-driven simulation and experimental results. Section 6 concludes this work.

2. RELATED WORK

Many WSN systems have been developed to support environment monitoring [Mainwaring et al. 2002], object tracking [Gui and Mohapatra 2004, He and Hou 2005], scientific observation [Xu et al. 2004], and so on. The underground environment of our system differs from most other systems in its varying geologic structures and conditions. Trying to capture and adapt to geologic structure changes, such as collapses, requires non-trivial solutions embedded in a sensor network system.

There have been several works on tunnel monitoring [Cheekiralla 2005]. Cheekiralla proposes utilizing electrolevel systems to measure the structural variations in London underground tunnels. The “smart infrastructure” proposed by Cambridge University explores the usage of fiber optics to monitor possible deformations on tunnel structures. Their system benefits from the advantages of fiber optics, including the insensitivity to electromagnetic interference, durability, and so on. Our work faces more challenging environments in the underground coal mine, where the monitoring system is expected to work against larger structure variations possibly caused by collapses, mine quakes or explosions, and persist even under such emergent accidents. The monitoring system is also expected to be more flexible and easy to be deployed and removed, as the progress of coal mine digging requires frequent movement of the system. This makes previous utilization of large and integrated systems infeasible.

Hole problems in WSNs have been surveyed by Ahmed et al. [2005], who divide holes into four categories: coverage holes, routing holes, jamming holes, and sink/black/worm holes. None of the works cited correlate the sensor holes to physical structure variations, or discuss the holes caused by topology changes.

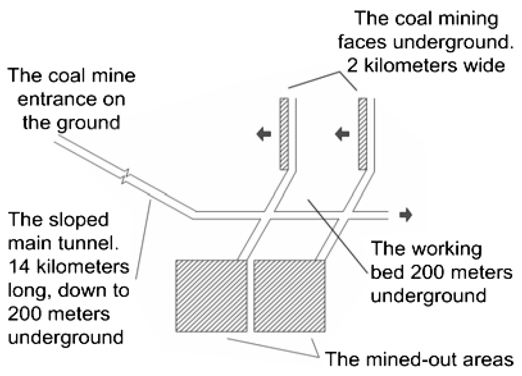

The coal mining faces underground. 2 kilometers wide
The coal mine entrance on the ground
The sloped main tunnel. 14 kilometers long, down to 200 meters underground
The working bed 200 meters underground
The mined-out areas

Fig. 1. An illustration of the D. L. coal mine.

Karp and Kung [2000] propose the Greedy Perimeter Stateless Routing (GPSR) protocol, which aims to utilize nodes' location information to provide efficient routing in WSNs. It employs perimeter mode routing to forward packets around holes. Douglas et al. [2001] propose the intermediate node forwarding (INF) scheme, which allows nodes originating packets destined to different intermediate nodes to route around holes. Aiming at efficient routing, these works do not localize the holes. Fang et al. [2004] define stuck nodes and propose BOUNDHOLE to find the sensor holes, utilizing strong stuck nodes. However, it is a theoretical work with strong assumptions and simplifications on the network model.

Traditional query aggregation techniques generally assume that sensing data are collected through routing trees. However, the routing tree scheme is not robust to the hostile environment in the underground coal mine. Recently, Considine et al [2004] and Nath et al [2004] concurrently proposed approximated approaches to answer aggregative queries using sketch. Their approaches approximately answer aggregative queries in multi-path routing schemes to achieve system robustness. The duplicate-insensitive sketch is used for carrying the *SUM* and *COUNT* information. Their approaches achieve approximate results even under frequent network variations, including node failures, link breakage, packet loss, and so on. However, their approaches only work on aggregative queries and there is no control on the error rate of aggregation results due to the property of random hash functions that sketch uses. The accuracy of the sketch approach is based on a large number of nodes being located in the query zone. If the number of nodes is small, the possible excessive bias on the sketch inserting may result in very low accuracy.

3. APPLICATION SCENARIO

We cooperated with the S. H. Coal Corporation and selected the D. L. coal mine as our experimental environment. It is one of the most automated coal mines, yielding the second largest production of coal worldwide. The D. L. coal mine is a typical slope mine, as illustrated in Figure 1. A slightly sloped 14-kilometer long main tunnel starts from the entrance above the ground surface and goes

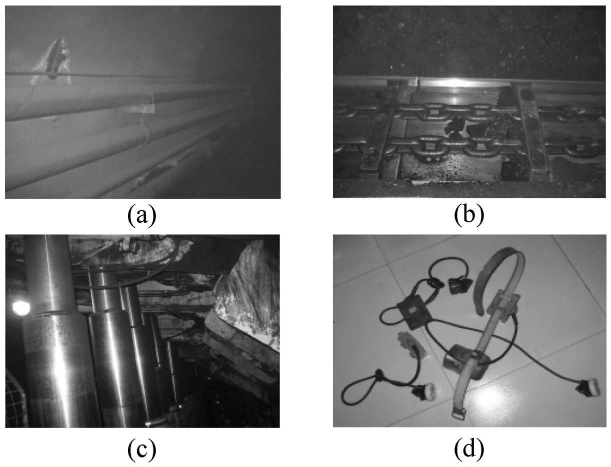


Fig. 2. (a) Main tunnel; (b) Conveyor belt transporting coal; (c) Coal mining face; (d) Devices carried by a miner.

200 meters deep underground to the working bed, as shown in Figure 2(a). The main tunnel is the primary passage for miners and equipment.

The state of the art of underground mining—longwall mining technology—was adopted in the D. L. coal mine. Today, longwall mining accounts for about one third of all underground coal tonnage. In a continuous, smooth motion, a rotating shear on the mining machine moves back and forth across the face of a block of coal, cutting the coal. Coal drops onto a conveyor and is removed from the mine, as shown in Figure 2(b).

Each longwall mining machine has a hydraulically operated steel canopy that supports the upper strata and protects miners at the face, as shown in Figure 2(c). There are currently two 2–3 kilometer wide faces being mined.

To monitor the underground environment in a coal mine, we designed and implemented the SASA system along the main tunnel and working spaces to fulfill the following requirements.

Remote management. Since it is preferable to remotely maintain and manage the entire monitoring system, efficient and robust communication and routing mechanisms are required under all conditions.

In-situ interactions. Providing geographical references could greatly facilitate locating miners underground. In addition to stationary sensors deployed on the walls, poles, and floors, miners carry mobile sensors. Figure 2(d) shows the devices carried by the miners of the D. L. coal mine.

Query answering. Providing the capability to answer various types of queries based on sensory data collection through the network; aggregation may be utilized to increase the efficiency of answering such queries.

Awareness of structural variations. One major goal of SASA is to instantly and accurately detect the collapse region. SASA aims to provide an infrastructural framework for underground monitoring with various environment sensors. Although we can detect collapses by equipping each node with acceleration

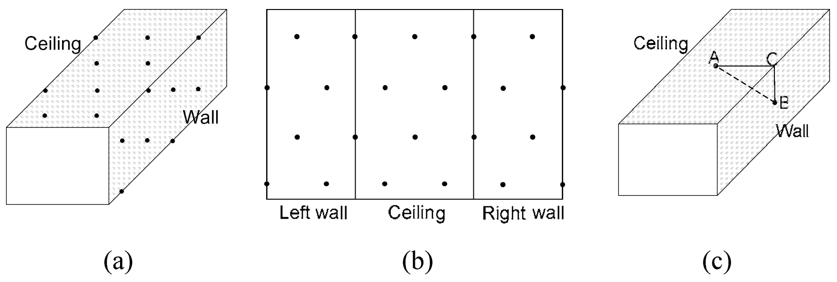


Fig. 3. Sensor node deployment.

sensors, this tends to make the system cost-inefficient. SASA achieves this goal through developing a sound mechanism of node collaborating.

Maintenance of system validity. A collapse may change the system structure. Maintaining the validity of the monitoring system in extreme situations is necessary; robust service is expected; an efficient recovery mechanism is required.

Efficiency and accuracy maintenance. Since these are collapse holes that may break the original sensor network structures, we need to design a sound and robust data collection mechanism to efficiently and accurately collect data. The mechanism should be robust to sensor network variations and efficient in communication cost.

4. SASA SYSTEM DESIGN

In this section, we present the design of the Structure-Aware Self-Adaptive sensor network, SASA.

4.1 Overview

In SASA, stationary sensor nodes are deployed on the walls and ceiling of tunnels to form a mesh network, as illustrated in Figure 3(a). To facilitate hole detection, SASA unfolds the two walls of the tunnel and builds a 2-D representation of the 3-D deployment on the inner surface of the tunnel, as depicted in Figure 3(b). The location preconfigured in each node is a 2-D location coordinate on the 2-D surface.

Nodes placed in the 3-D real environment are configured with 2-D coordinates on the unfolded 2-D surface. SASA conducts a transformation between the two locations with the knowledge of the longitudinal section of the tunnel such that the 2-D location uniquely corresponds to the 3-D location. In practice, the relationships between neighboring nodes in the 3-D real environment are the same as in the 2-D representation, except for a small area in corners where ceilings meet walls. As Figure 3(c) shows, the distance between any two nodes in the 3-D real environment is less than or equal to the distance between the pair in the unfolded 2-D view. Thus, the real connectivity of our sensor network is no less than shown in the 2-D representation. Later we will show that the neighbor set defined in our system in the 2-D representation is preserved in the 3-D real environment, and the correctness of the hole-detecting algorithm is preserved.

In a real application, the sensors deployed in different tunnels are differentiated by being marked with different tunnel numbers. This way, holes in different tunnels can be identified. In the following discussions, we focus on the design rationale of SASA deployed in one tunnel.

We also require each miner to carry two sensors together with the devices shown in Figure 2(d). As the miners are moving, these mobile sensors are utilized to calculate miners' locations based on the stationary mesh nodes. This is crucial to the rescue operation when an underground accident happens. When any exceptional situation is detected, alarm messages are created and transmitted to the sink, triggering an external safety system to inform operators outside the tunnel.

The hardware layer for our system is built on the widely used Mica2 mote platform [Hill and Culler 2002], developed at UC Berkeley. The MPR400 radio board employed has a 7.3MHz microprocessor, with 128K bytes of program flash memory and 512K bytes of measurement flash memory. An 868/916 MHz tunable Chipcon CC1000 multichannel transceiver with a 38.4 Kbps transmission rate is employed for wireless communication with a 500-foot outdoor range. A sensor board is connected to the Mica2 mote performing environmental data collection. The collected data is delivered to the Mica2 mote for further processing.

In this work, SASA focuses mainly on the construction and maintenance of the sensor network for monitoring collapse holes. According to statistics in coal mines, such a collapse may occur at any time and any place. SASA uses robust fault recovery and data collection mechanisms, during collapses. The functions of regular monitoring such as gas and water monitoring are also supported by SASA but are not emphasized in this article. The main functions of SASA include:

Detecting and locating the collapse hole. This is the primary function of SASA. Successfully locating the hole region after collapse assists instant rescue and subsequent repairs.

Accident reporting. The accident reporting messages need to be rapidly and reliably routed from the collapse region back to the sink. SASA aims to provide a systematic solution for that.

Displaced node detection and reconfiguration. After the collapse, the original sensor nodes in the hole region may be relocated. The original location configurations of these nodes then become outdated, which may lead to incorrect location references, and improper routing actions, thus reducing the stability and reliability of the SASA system. Consequently, it is necessary to rapidly detect these nodes and reconfigure them with correct locations in order to maintain system validity.

Maintaining robust query handling. The collapse holes might break the original routing structures in the sensor network, leading to the inaccuracy of data for query answering. SASA aims to avoid this problem by developing a multipath routing scheme, where the desired data are copied at each hop and delivered through multiple paths towards the sink. This scheme provides robust data collection even during possible collapses. SASA further utilizes data aggregation to increase the efficiency of query handling, which enables us to answer both aggregative and range queries [Lian et al. 2005] accurately and efficiently.

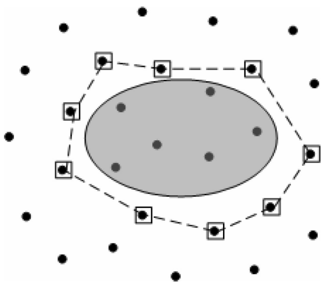


Fig. 4. The sensor hole and its outline nodes.

4.2 Design Rationale

In SASA, to get rapid and accurate detection of the collapse hole, we exploit the relation between sensors within and outside of the collapse region. SASA does not rely on any additional devices for achieving this task. Although equipping accelerometers for the sensor nodes might help, it incurs excessive cost for the system. Each accelerometer costs $50+ and is much more expensive for more tolerance (10g+) on impulse. Adding accelerometers in sensor nodes also complicates the design of hardware. The SASA system aims to provide a framework for general monitoring applications. System efficiency will drop with any add-in block.

When a collapse occurs, the sensor nodes in the accident region are moved, and a hole of sensor nodes emerges. For a reasonable density of sensor node deployment, the sensor node hole should reflect the actual collapse hole to a certain degree. When the sensor hole emerges, as shown in Figure 4, the nodes on the hole edge will have a loss of neighbor nodes, and these nodes outline the hole.

The basic idea in detecting a hole is to let sensor nodes maintain a set of their neighbors. When a node suddenly finds that a subset of its neighbors has disappeared, it should be aware that it is now likely to be an edge node of a hole. A straightforward method of maintaining neighbor sets is to require that nodes periodically probe their neighbors. However, this approach is costly in terms of traffic overhead. To address this issue, we propose a beacon mechanism, which requires each node to actively report its existence. By carefully deploying the sensor nodes into a regular mesh network and determining a criterion for hole detection by neighbor losses, our algorithm can provide approximation of the collapse hole region through the *edge nodes* around the hole region. The hole region approximation is calculated in the sink. A data aggregation strategy is employed to reduce the instant traffic.

The node reconfiguration process is then divided into two phases: displaced node detection, and node reconfiguration. In the displaced node detection phase, both centralized and decentralized mechanisms are employed to achieve short detection latency. In the node reconfiguration phase, a displaced node estimates its new location based on surrounding normal nodes. Iterative calculation is conducted to get an accurate estimation.

Besides these structure-aware behaviors, SASA delivers data through our predeployed mesh sensor network. Many data gathering algorithms for WSNs have been proposed [Karp and Kung 2000, Melodia et al. 2004, Subramanian and Shakkottai 2005], but most of them undergo inefficiency because of the possible variations of the sensor network during the collapses. SASA aims to provide robust data delivery through a multipath routing strategy [Nath et al. 2004]. SASA further develops a sound data aggregation scheme on top of the multipath routing strategy, which provides accurate and efficient query handling for both aggregative and range queries.

Many key issues have been examined in SASA design and implementation. Our discussion in this article will focus on node beaconing mechanisms hole detection, accident reporting, displaced nodes detection, and reconfiguration, as well as robust query handling as follows.

4.3 Node Beaconing Mechanism

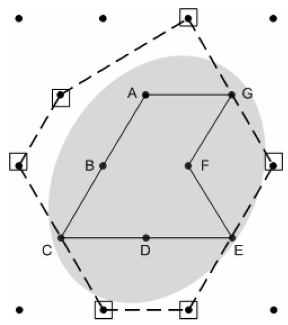
In order to monitor structural change, each node is responsible for inspecting its surrounding nodes. Intuitively, to require each node to dynamically probe its neighbor nodes is simple but inefficient. In the sensor network, a transmission between two nodes can only be achieved by the node broadcasting locally. The broadcast creates a collision domain where all other nodes in this domain must remain silent in order to avoid collisions. If we consider the message broadcasting manipulation as the cost unit, the active probing strategy has a traffic cost of $O(nk)$, where n represents the network size, and k is the average number of neighbors per node. Replies from the neighbors are $O(k)$.

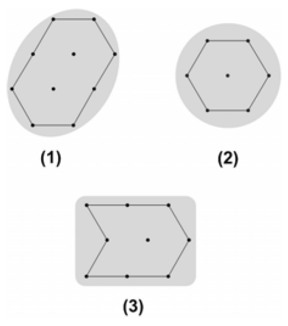
SASA adopts a beacon mechanism, in which nodes passively listen to their neighbors: each node periodically broadcasts beacon messages that include its location. This beacon mechanism benefits from the "wireless multicast advantage" (WMA) [Wieselthier et al. 2000] in WSNs and could effectively reduce the traffic cost down to $O(n)$. To avoid collisions, we set a small random variation for the beacon interval, which prevents multiple nodes from broadcasting beacon messages simultaneously.

4.4 Hole Detection

A node maintains a neighbor list in its memory. Upon receiving a beacon message, it updates the corresponding entry. A timer T_1 is then set to determine the entry expiration: an entry not updated by the time it expires represents the loss of the neighbor. In our experiment T_1 is set to be 3 times the beacon interval. Upon a collapse, nodes beside a hole become edge nodes. They are able to rapidly detect loss of neighbors.

However, regulating the neighbor set of a node is challenging because the RSS (Radio Signal Strength) value between nodes is highly dynamic in the coal mine environment, making it hard to function as an indicator. Consequently, a naïve method, in which all the nodes whose beacon messages could be received were taken as neighbors, failed in our prototype implementation experiment. It was observed that the neighbor set of a node is highly unstable, even if all the nodes work normally. Also, nodes often had different sizes of neighbor sets, if initially the nodes were not regularly spaced. All of these factors made it hard to determine a criterion for detecting the hole via neighbor loss detection.





(a) (b)

Fig. 5. (a) Hole and edge nodes; (b) Hole polygon examples.

To address this issue, SASA deploys sensor nodes in a cellular hexagonal placement such that the node distribution is uniform, as illustrated in Figure 3(b). In the 2-D representation, every pair of adjacent nodes is separated by the same interval, which can be varied from several meters to tens of meters, as determined by the size of the detection area, required precision, and the signal range of the sensor nodes. Every node (excluding boundary nodes), if taken as the center of a regular hexagon, has 6 adjacent nodes on the 6 vertices of the hexagon. In our experiment, we selected a 3-meter interval deployment. Keeping effective radio signals at 3 meters might result in maximum radio ranges of 4 to 5 meters with interspaces [Feyerstein et al. 1994], due to the individual differences in nodes. With this setting, a sensor node may receive beacon messages from nodes other than the 6 adjacent ones. However, in the neighbor list, we limit each node's neighbor set to the 6 adjacent nodes, therefore the nodes other than those 6 will not be maintained in neighbor entries although their beacons may be received. This is achieved by each sensor locally examining the locations of the beacon sender. By calculating the distance between the sender and itself, each sensor discovers and maintains its neighborhood. Such a scheme of neighbor maintenance provides a firm set of neighbors for each node and thus a regular method to determine the edge nodes.

Definition. A node defines itself as an *edge node* if the two adjacent neighbor nodes are detected as lost during a time period T_2.

Another timer, T_2, is set for determining the edge nodes. Timer T_2 is slightly larger than timer T_1 for detecting neighbor loss. Upon a collapse, this criterion generates a set of edge nodes. These edge nodes act as landmarks to display the hole region.

Definition. hole polygon is defined as the largest polygon outlined by the collapsed sensor nodes with every edge ending at two adjacent nodes.

For example, the polygon *ACEFG* in Figure 5(a) forms a hole polygon. A hole polygon functions as a geometric representation for the hole region. We provide more examples of hole polygons in Figure 5(b). Since every edge node has at least two neighbors in the hole polygon, each is at most 2.6 meters away from

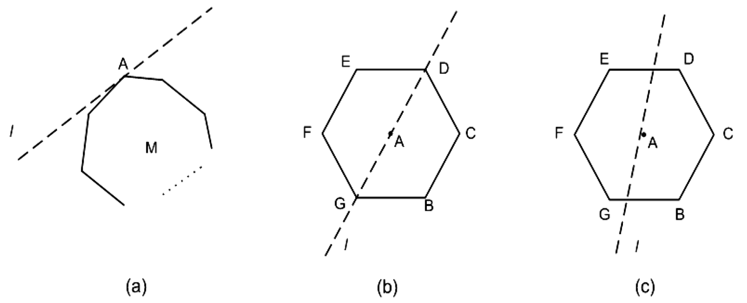


Fig. 6. (a) The convex hull of edge points and hole points; (b) and (c) Two cases of the relationship between 1 and M.

the hole region, and the outline drawn by these edge nodes is at most 2.6 meters away from the hole polygon. This gives an upper bound. We give a proof that the convex hull of the edge nodes (see Figure 5(a)) encloses the hole polygon, which is the lower bound of the outline drawn by the edge nodes.

THEOREM. *The convex hull of edge nodes in SASA encloses the hole polygon.*

PROOF. We prove it by contradiction. For a geometric abstraction, we refer to all the edge nodes as edge points, and all the vertices of the hole polygon as hole points.

Suppose there is at least one hole point outside of the convex hull of edge points. We draw a convex hull, M, of both the edge points and hole points, as shown in Figure 6(a). There must be one hole point that is the hull point. Without loss of generality, suppose the point is A.

As shown in Figure 6(a), we can draw a line l across A such that all other points of M are on one side of l. This is guaranteed by the characteristic of a convex hull. Point A has two adjacent hole points on the hole polygon out of its 6 adjacent neighbor points. According to the relationship between line l and the 6 adjacent neighbor points of A, there are two cases, as shown in Figures 6(b) and (c).

Case 1. In Figure 6(b), line l crosses two neighbor points. If M is bounded on the right side of l the same applies when M is bounded on the left side of l), the two adjacent hole points of A can only be B and C. In this case, points D and G must be edge nodes since they both have two hole points as neighbors. This contradicts the assumption that point A is a hull point of convex hull M.

Case 2. In Figure 6(c), line l crosses no neighbor points. If we suppose M is bounded on the right side of l, the two adjacent hole points of A can only be either B and C or D and C. In both cases, either B or D is a hole point adjacent to point A, which makes either point E or G an edge node. Since both points E and G are outside of M, a contradiction is formed. Therefore, there is no hole point outside of the convex hull of the edge points, and thus the theorem. □

Since the algorithm for calculating the convex hull of n points has a computational complexity of $O(n \cdot logn)$, it is a light-weight method for the sink to achieve this bound.

In practice, multiple nodes breaking down in a region at the same time can be considered the result of a collapse, whereas a single node failure in a certain region is likely the result of a power off or node failure. Our hole detecting algorithm is tolerant of the interference from single node failures since the failure of at least two adjacent nodes is necessary to define an edge node. Nevertheless, if two adjacent nodes fail simultaneously, the algorithm fails. As a marginal effect, a small hole affecting only one sensor node cannot be detected by this algorithm. This sets the threshold of the size of detectable holes. However, this threshold can be lowered by increasing the density of deployed sensors. A lager density of sensor deployment helps to provide a more precise detection ratio for small holes, while a smaller density of sensor deployment helps to reduce the number of sensors needed. We can achieve a balance between the system requirements and cost.

The Mica2 motes adopted in SASA employ a CSMA transmitting protocol for multiple accesses in wireless communication channels. Although this protocol is effective for collision avoidance, collisions are still a major problem in a densely deployed sensor network due to the hidden terminal problem, especially when the communication density is high. Such collisions waste network bandwidth and greatly increase the packet loss rate.

To reduce collisions, SASA tries to maintain a comparatively low communication density, which is highly dependent on the beacon mechanism. A lower beacon density helps keep communication density lower while leading to a longer detecting latency. So how to balance this tradeoff is important and will be examined in the experiment section.

4.5 Accident Reporting

When edge nodes detect a hole, they report to the sink with the locations of the edge nodes so that the hole can be outlined by calculating the convex hull. A relatively effective but expensive approach is to deliver messages by flooding. When a collapse occurs, however, all the edge nodes might flood report messages at the same time, creating a traffic peak and increasing the collision probability. To reduce the collisions [Rajendran et al. 2003], we introduce (1) a randomized forwarding latency, and (2) a data aggregation strategy.

We insert a flag into the beacon messages that indicates whether the beaconing node is an edge node. The edge node waits a short time before sending out its report. Upon receiving other edge nodes' beacon messages, an edge node records them locally. When this edge node sends out its report message, it aggregates all the recorded locations of its nearby edge nodes in one report message. If an edge node receives a report message containing its own location, it is aware of the fact that another edge node has already aggregated its location. This node will simply forward this message instead of generating a new one. The total amount of traffic is thus reduced.

The sink reply is employed to maintain the reliable transmission of report messages. An aggregated reply message including all the received locations of edge nodes is flooded out from the sink. The edge nodes not included retransmit report messages. SASA limits the number of retransmissions so that the edge

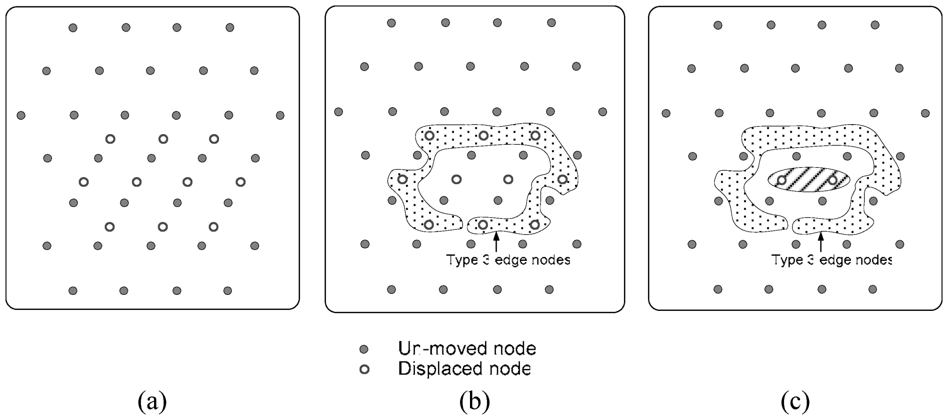


Fig. 7. An example of the distributed detection algorithm. (a) Several nodes fall into a place together with all of their neighbors; (b) Type3 edge nodes stop beaconing and displaced inner ones find neighbor loss; (c) Inner nodes define themselves as edge nodes and indicate displacement.

node will not keep transmitting report messages if it has been isolated from the sink. Such isolation is possible as the network can be disconnected by a large collapse. In our system implementation, this phase is simplified and merged into the *node reconfiguration* process.

4.6 Displaced Nodes Detection and Reconfiguration

During a collapse, the sensor nodes in the hole region are displaced with new nodes surrounding them. When a node becomes an edge node, we also need to determine whether or not it has been moved. The other challenge is that not all the displaced nodes become edge nodes immediately after a collapse. For example, a node and all its neighbors may fall into one place together, as shown in Figure 7(a). Since the inner-displaced nodes do not find any neighbor loss, they will not define themselves as edge nodes. In this application, we need to detect displaced nodes and reconfigure their locations.

The basic idea will be trivial if we utilize the global information. When the sink receives report messages with the edge nodes' locations and approximates the hole region, it broadcasts the convex hull area, informing the nodes in the hole region of their displacement. Every node within the convex hull will start detecting its surroundings and check its location from beacon messages. An average location can be calculated and compared with its own configured location. If the two locations differ beyond some threshold, it knows its displacement.

To shorten the message length and save power, SASA uses a rectangular enclosure to approximate the convex area, which costs 16 bytes to represent the 4 vertices and simplifies the calculation of each node, as illustrated in Figure 8. Though the approximation is less accurate, it is adequate to describe the hole.

The major issue of such a centralized approach is that it often suffers long latency and low accuracy due to the high link loss rate in coal mines, especially when a collapse area in a long tunnel is far from the sink. In extreme cases, the

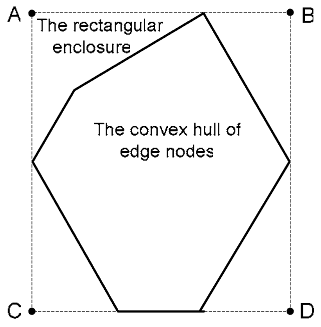


Fig. 8. The rectangular enclosure.

network could be disconnected by a large collapse, although such large collapses are rare according to the past 20-year history of the D. L. coal mine. Indeed, since small scale collapses frequently precede the more dangerous and more easily located large scale collapses, we can use the detection of small collapses as an early warning, alert or indication of the possibility of a large collapse in order to evacuate or repair the dangerous area/structure. It is already too late when large collapses occur, so rapidly detecting and reporting small collapse locations are significant for coal mine safety. Hence, the primary focus of SASA is on locating small scale collapses.

In order to further reduce detection latency and improve accuracy, we propose a distributed algorithm. Recall that the definition of *edge node* is a node that has lost at least two contiguous neighbors. There are three types of edge nodes: (1) the edge nodes that lose neighbors but themselves do not move; (2) edge nodes that fall into an area where no normal node exists; (3) edge nodes that fall into another normal node range. For type 1 nodes, their locations are correct, so they do not need any reconfiguration. For type 2 edge nodes, they have no impact on normal nodes, so they do not need any action as well. Indeed, a node cannot easily recognize whether it belongs to type 1 or to type 2.

So our focus is on type 3 edge nodes. A node defines itself as a type 3 edge node if and only if: (1) it is an edge node and (2) it detects newly emerged neighbors. A type 3 edge node stops beaconing immediately, as illustrated in Figure 7(b). This operation will lead the neighboring displaced nodes to become edge nodes, if they are not yet, as shown in Figure 7(c). In a recursive manner, all the nodes removed from hole region will become edge nodes and detect their location variations.

The recovery latency is correlated with the recursive process, which may have several phases, so it is longer than that of the centralized algorithm when the collapse area is close to the sink. However, since it is a local algorithm, the recovery latency is independent of the distance to the sink. Combining the two detecting algorithms provides efficient and reliable recovery for various situations. SASA employs both mechanisms.

When the displaced nodes are discovered, we can simply turn them off or reconfigure their locations to conform to their new positions. The SASA adopts node reconfiguration to conserve as many working nodes as possible to maintain

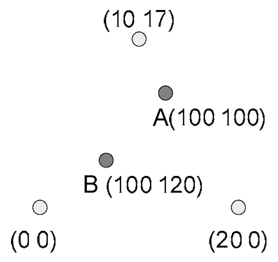


Fig. 9. Reconfiguration of A and B.

an adequate node density. Although many schemes [Lazos et al. 2005, Ray et al. 2003, Savvides et al. 2001] have been proposed for localization in general WSNs, we find most of them infeasible in our context, since the highly dynamic radio signal strength in the underground environment makes it extremely difficult for the ranging operations of those schemes. We try to explore simple but effective solutions.

If we let the nodes calculate average locations from surrounding nodes, as some of the surrounding nodes may also come from a hole, the calculation could lead to an inaccurate result. Therefore, we design an iterative method for location calculation. Suppose two nodes, A and B, drop into a new area surrounded by 3 resident nodes as shown in Figure 9. Initially they have their original location. When node A first detects the surrounding four nodes, it calculates a new location as (32.5, 34.25) and replaces the original location. Then when node B detects its surroundings, it utilizes the new location of node A and calculates a new location as (15.63, 12.81). Thus, when node A iteratively calculates its new location, it will get a more accurate result of (11.41, 7.45). This iterative process continues and the calculated locations of nodes A and B tend to the center of the three original resident nodes, which is a close approximation for their new locations. This process is accompanied by the process of displaced node detection, so as previously described, the displaced nodes will stop beaconing once their identities are confirmed. This prevents those displaced nodes from sending out too many misleading messages and helps to speed up the iterative process of location updating.

4.7 Robust Query Handling

Query handling is an everlasting job running over the deployed sensor network in the coal mine. In our coal mine monitoring scenarios, we mainly need to handle external information queries, including aggregative queries, like *Min*, *Max*, and *Count*, as well as range queries, which collect the IDs of sensors of specified sensory values. The main challenge in answering queries in our SASA system is twofold. First, SASA aims to provide a robust mechanism, which reliably delivers sensory data even with unstable network structures caused by collapses, and accurately handles query answering. Second, SASA aims to provide accurate and efficient query processing with small in-network communication overhead since the underground environment is not friendly for wireless communications. Also the sensor nodes are normally battery powered and changing batteries is often very difficult in an underground coal mine [Madden et al. 2002].

The first goal leads SASA to adopt the multi-path routing strategy. Compared with the traditional tree based routing strategy (TAG [Madden et al. 2002]), in the multi-path strategy, each node delivers and relays data to multiple parents, greatly increasing the robustness of the system. By this strategy, even with variations in structure, the sensor network can achieve maximum reliability in data delivery, as long as there exists a connected path to the sink. However, this scheme introduces another problem, the possibility of the same data being counted more than once (*duplicate sensitive*). When doing in-network aggregation on queries like *Count* or *Ave*, this might lead to failure in achieving the second goal. The sensory value of one node might be copied and delivered to several parent nodes, leading to double counting in the in-network aggregation process.

We address these problems by encoding collected sensor readings using signature files, reducing the cost of transmitting data, and removing possible duplicates by a simple bitwise "OR" operation . Signature files were first introduced as an indexing method for text retrieval [Faloutsos 1985]. A fixed-width signature (bitstring) of m bits (length) is assigned to represent each key word (distinct value) with w bits (weight) being set to 1. The m bits are set with a number of hashing functions. One advantage of signature files is duplicate-insensitivity after superimposed coding. Furthermore, the overall false drop (alarm rate) can be controlled by carefully setting w and m. Instead of sending the real values, we let sensors send the signature files to the upper level node and carry out the partial aggregation by superimposing (ORing) with the received signature files. All the duplications can be removed because of the "OR" operation. At the sink, we will get the superimposed bitstring (result signature), and we compare (ANDed) the signature of each distinct value with the result signature to check whether a distinct value exists in the final result. If it does, the value will be used to compute the final aggregation result. Due to the "OR" operation, a value that is not sensed by the sensors within the sensor network may be identified as *existence*, which is named as *false drop*. The false drop rate can be controlled by carefully setting the signature weight (w) and length (m). The equation for computing w and m is proved by Davis and Kamamohanaro [1983]. The false drop rate of a signature file P^f is $P(N, w)$, where N is the number of distinct values. In order to minimize P^f, w and m can be set at the following:

$$w = \left(\frac{1}{\ln 2}\right) \times \ln\left(\frac{1}{P^f}\right) \tag{1}$$

$$m = \left(\frac{1}{\ln 2}\right)^2 \times N \times \ln\left(\frac{1}{P^f}\right). \tag{2}$$

In SASA, each sensor has a unique ID and knows its position. We adopt the multi-path topology for data transmission in SASA [Madden et al. 2002]. The multi-path topology is created in the sensor network according to the node hops to the sink. Nodes are divided into different levels according to their hop count. The hop count of each node from the sink indicates its level. This could be achieved in the system initialization phase through advertising the node hop count from the sink. In the data collection (query reply) phase, each node reports its aggregated result by local broadcasting. All of its neighbors, which are one level lower, capture the report and aggregate it. Nodes at different levels aggregate data in different time epochs during the data collection phase, as in the TAG approach [Madden et al. 2002].

To answer aggregative queries, SASA encodes distinct sensing values into signature files. The number of distinct values, N, can be easily computed with a use- or application-specified precision, ε,

$$N = \left\lceil \frac{(V_{\max} - V_{\min})}{\varepsilon} + 0.5 \right\rceil. \tag{3}$$

SASA divides the range into N buckets and represents each bucket with its mean value. When the sink floods the aggregative queries, it also sends the information with respect to bucket size and hash functions. Each sensor will determine whether it needs to send its sensing data by checking its position against the query specification. If it is required to send the data, its sensing value is compared with the bucket information and a bucket that is closest to its value is selected. Then, the selected bucket value is encoded into signature files using the hashing functions.

This approach divides the data range into buckets with the same size, which may introduce large errors when data distribution does not follow a uniform distribution. For example, the oxygen density data that we collected from readings in the coal mine show that the data come from a Gaussian normal distribution. If we adopt the same size bucket for each data value in the data space, the probability that values appear in some bucket ranges will be higher than that of other ranges. Thus, for the buckets with a higher probability that data will fall in, multiple sensor readings may compete for the same value bucket. Once the different values from different sensors share the same bucket, they are treated as a single value, which leads to inaccuracy in the final aggregation result. To address this problem, we exploit the *dynamic bucket allocation* method. In this bucket allocation method, instead of segmenting the value space into N equal sized buckets, we segment the value space into various sized buckets that conform to the data distribution curve. The boundary of the ith bucket is computed according to the following formula:

$$\int_{V_i}^{V_{(i+1)}} P(x)dx = \frac{1}{N} \int_{V_{\min}}^{V_{\max}} P(x)dx, \tag{4}$$

where $P(x)$ is the normal probability distribution function, $1 \leq i \leq N$, $V_1 = V_{min}$, and $V_{N+1} = V_{max}$. With this type of segmentation, we assign equal probability to each bucket the data falls into.

Answering range queries is similar to aggregative queries. We assume the knowledge of all the sensor IDs, and that they can be represented by positive integers. A range query needs to collect a set of sensor IDs whose sensing values are within the query-specified value range. Since an ID is unique to a sensor and values of IDs are uniformly distributed, we can uniformly divide the value range of IDs into N buckets, where each bucket corresponds to one unique ID.

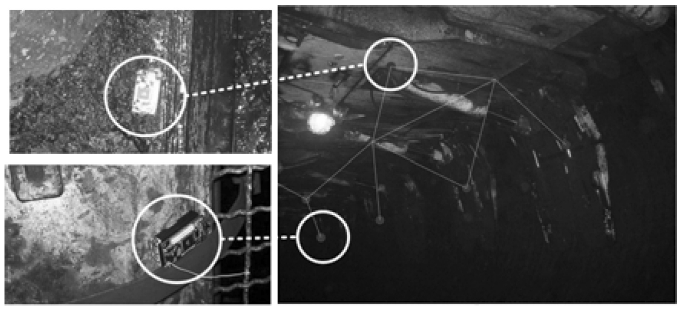


Fig. 10. SASA deployment.

We encode the IDs as signature files from the sensors that are at the highest level (furthest from the sink). Superimposing is used on lower level sensor nodes to aggregate signature files of IDs. At the sink, we then check the existence of IDs through an ANDed operation. Compared to the simple approach that collects and transmits a list of IDs directly (*LIST* approach), whose message length increases along the path from the sensors to the sink, our signature file-based approach significantly reduces the message communication cost.

As indicated in Equations 1 and 2, in order to make signature files more accurate (to lessen the error rate), the length of the signature file is usually very long. Directly transmitting raw signature files may lead to a high communication cost, which is exactly the cost we want to avoid. By observing the generated signature files, we found that even when they are very long, the number of “1”s are few. Also, these “1”s sparsely locate in the bit string and are separated by many continuous “0”s. To compress the example signature file, we can use a modified Run-Length Encoding (RLE). The original RLE uses block representation for both “1” and “0” appearing in the string. To shorten the string representation, a k bit block is used to represent a continuously appearing “1” or “0.” We modify the original RLE according to the feature of the signature file that generally has long substrings of continuous “0”s but discrete “1”s. For compressing the signature files, we merely substitute the continuous “0”s with the block representations (a bit “0” $+ k$ bit representation of N). The block representation size, k, can be adjusted in the same way that RLE does. This enables us to significantly reduce the length of the transmitted data.

5. EXPERIMENT AND PERFORMANCE

5.1 Prototype Implementation

A prototype system with 27 Mica2 motes is implemented and deployed in the D. L. Coal mine as illustrated in Figure 10. The system is distributed on a tunnel wall about 8 meters wide and 4 meters high. The motes are preconfigured with their location coordinates and manually placed at surveyed points with an interval of 3 meters, as specified in our proposed hexagon mesh regulation. The *CC1000control* component of each mica2 mote is adjusted so that when the motes broadcast beaconing messages, the maximum signal range is minimized

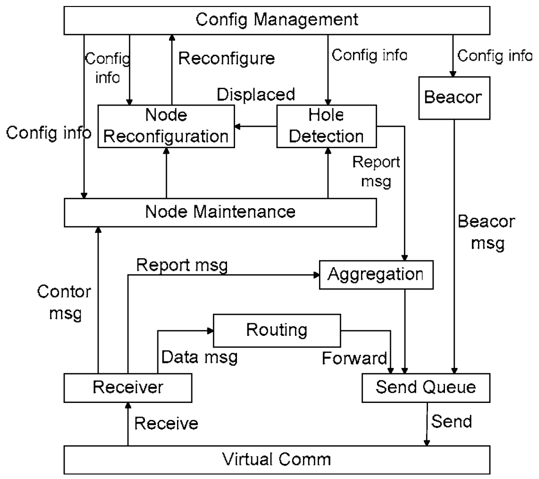


Fig. 11. Block diagram of the system architecture.

in order to reduce collisions, while guaranteeing the desired 4-meter signal coverage. The signal range is increased for flooding or forwarding messages to maintain transmitting efficiency. Figure 11 shows the block diagram of SASA architecture implemented in TinyOS on the Mica2 motes. The "Config Management" component manages the configuration information of the node, including the node ID and its configured location, which act as the fundamental elements for each node. The "Hole Detection" and "Node Reconfiguration" components are constructed on the "Node Maintenance" component, which deals with control information from surrounding node beacon messages and centralized control messages.

An indicator "*nodeStatus*" is used to switch the system between the two working statuses: normally working (for hole detection) or in reconfiguration. The "Beacon" component periodically broadcasts the current config information of the sensor node. It is taken as application data payload in the TinyOS RF message with the destination of local broadcast *TOS_BCAST_ADDR* and the specified handler ID *AM_BEACONMSG* = 131.

For the analysis of the hole detecting performance in this experiment, 20 different sensor holes are selected from collapses recorded in S. H. Coal Corporation history. Their sizes range from 48m^2 to 132m^2. For each instance, we randomly redistribute the displaced sensor nodes from the hole region in the tunnel 10 times.

Table I presents the statistics of our system performance in the 200 testing samples. The metrics are defined as following.

The *hole detection percentage* reflects the effectiveness of the system in detecting the hole. A hole is counted undetected if less than 3 edge node reports are received by sink.

The *hole detection error* is measured by the distance between the real and detected positions of the hole region. The position of the hole is represented by the geometric center of the hole region.

Table I. System Performance

Hole detection percentage (%)	100%
Average hole detection error (m)	0.73
Average reconfiguration 2D error (m)	0.87
Average reconfiguration 3D error (m)	2.62

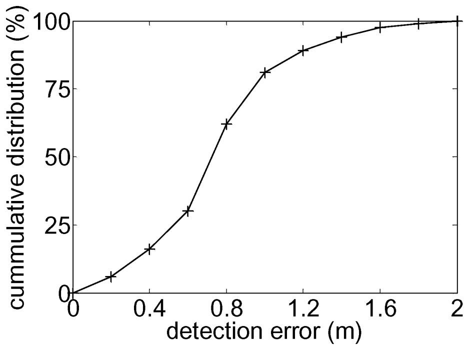


Fig. 12. Hole detection accuracy.

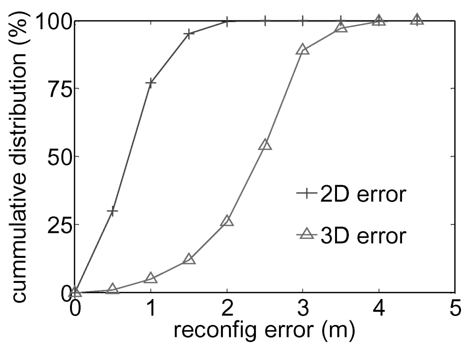


Fig. 13. Reconfiguration accuracy.

The *reconfiguration error* is the localization error in the reconfiguration process. The 2D error is the error of the reconfigured node position in the 2D representation of the tunnel surface, and the 3D error is the error of the reconfigured node location in the 3D real space. Though both kinds of errors affect the system performance, the 2D error exerts a dominating effect on the system validity, while the 3D error degrades the accuracy for mobile node localization. A more precise reconfiguration process achieves better system resilience.

Figure 12 plots the hole detection error where over 80% of the detected holes are located within 1 meter from its real position, and 99 + % are less than 2 meters. The detection error comes mainly from the mismatch between the outlined hole region and the real hole region. The loss of report messages due to collisions also introduces error. Figure 13 plots the cumulative distributions of the 2D and 3D errors of node reconfiguration. We can see that all of the 2D errors and over 80% of the 3D errors are less than 3 meters.

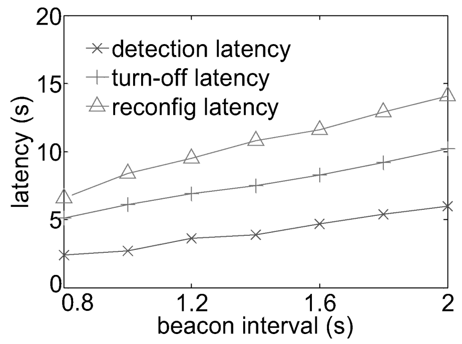


Fig. 14. Processing latencies vs. beacon interval.

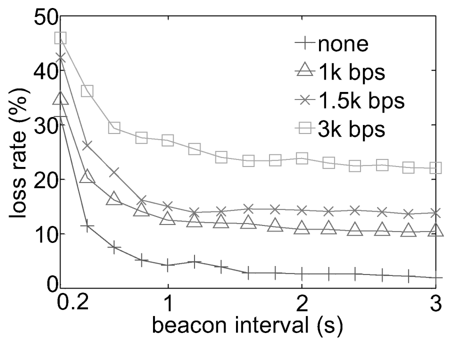


Fig. 15. Packet loss rate vs. beacon interval.

A short beacon interval leads to short processing latency for both hole detection and node reconfiguration. Figure 14 plots three kinds of processing latencies against the beacon interval. The *detection latency* represents the time from when the hole emerges until it is detected. The *turn-off latency* represents the latency when we choose to simply turn off the detected displaced nodes, and the *reconfig latency* represents the latency when we choose to reconfigure the displaced nodes according to the normal nodes surrounding them. Each of the three types of latencies is proportionally increased as the beacon interval increases. We observe that for each beacon interval, the *reconfig latency* is longer than the turn-off latency. This difference is caused due to the time needed for nodes to recursively calculate their new locations.

Figure 14 suggests a short beacon interval for pursuing short processing latencies. However, frequent beaconing brings large overhead, leading to heavy collisions and increased packet losses. In experiments, the communication quality between two neighboring nodes is tested for various beacon intervals under different traffic pressures. As shown in Figure 15, the packet loss rate rapidly drops as the beacon interval increases for beacon intervals of less than 0.8s, then becomes stable around a fixed level. The loss rate is increased as the exerted traffic overhead increases.

Based on these observations, we are able to carefully select a proper beacon interval for a specific application workload to balance communication

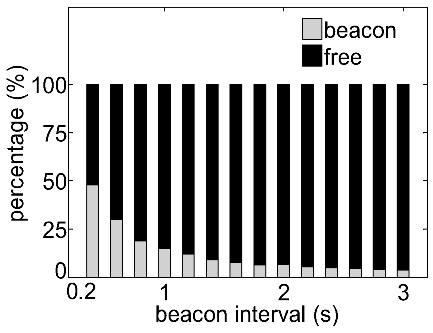


Fig. 16. Bandwidth vs. beacon interval.

quality and processing latency. We can shorten the beacon interval to reduce the processing latency if the application workload is light or lengthen the beacon interval to reduce the packet loss rate if the application workload is heavy, while the application is tolerant to the processing latency. Note that while more communication overhead is introduced with the increase of beacon frequency, less free bandwidth for data transfer is preserved. To further exploit this relationship, we code the motes to generate artificial traffic and test the maximum data transfer bandwidth under various beacon frequencies.

Figure 16 shows the tradeoff between the bandwidth for beacon and other data transfer. As the beacon interval increases, more available bandwidth can be provided for other practical applications. The available bandwidth is important because as we mentioned, SASA also provides the function of monitoring other environmental factors such as gas, water leakage, and oxygen density, and so on.

In SASA, once the stationary mesh sensor network is established and maintained, well-configured nodes provide accurate location references for mobile node localization.

The mobile nodes on miners and tramcars determine their own locations by detecting surrounding mesh nodes as location references. The detecting operation can be achieved by mobile nodes listening to the beacon messages of the stationary mesh nodes. Here we discuss the impact of node reconfiguration on localization accuracy. We first test the localization error in a normally working system, then in a reconfigured system after a hole emerges. The *localization error* is defined as the distance between the real and the computed positions. As plotted in Figure 17, we can see that because the reconfiguration process introduces error, the localization accuracy is also affected. Nevertheless, the error is relatively small according to the experimental results.

5.2 Simulation

The experiments on the SASA prototype present a partial image of our system performance, with some basic phenomena observed. In order to have a more extensive picture of the performance of SASA with thousands of sensor nodes, and to evaluate its scalability, we conduct a large-scale simulation based on the data collected from our prototype experiment.

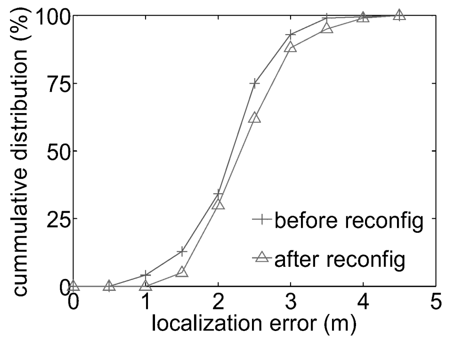


Fig. 17. Mobile node localization accuracy.

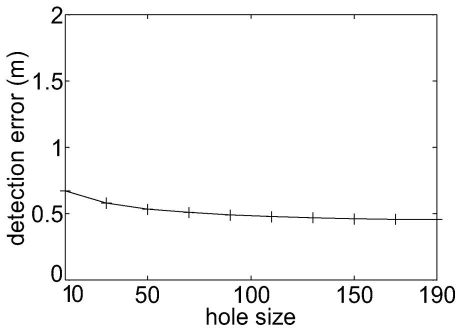


Fig. 18. Hole detection error vs. hole size.

In this trace-driven simulation, 2000 nodes were simulated on a 1000 m $\times$ 20 m plane with a 3 meter interval in the hexagon mesh regulation. A transmission rate of 16 packets/s is used in the simulation for the nodes' communication channels. This transmission rate was selected based on data from our experiment on the mica2 motes in the coal mine. The sizes of beacon messages and report messages are both 14 bytes, including the headers. Each node is assumed to have a 4-meter transmission range of beaconing, and 20-meter maximum communication radius when needed. The hole detection accuracy is tested for various hole sizes. Figure 18 exhibits the detection error as the hole size varies. The detection error is stable and decreases slightly as the hole size increases. A larger hole includes more edge nodes, giving a more accurate outline of the hole region.

We define another metric, *hole detection precision* $p = D^2/H \cdot G \times 100\%$, where H and G represent the area of the convex hull of the hole nodes and the area of the outlined hole region by the edge nodes, respectively. D is the area of the overlaps of H and G. This metric describes the tightness of the outlined hole region. A tighter outline requires a more precise shape and size, suiting the real hole region. Figure 19 plots the detection precision against the hole size. As we discussed in Section 4.2, the outline drawn from the edge nodes is bounded within one hop from the hole nodes. When the hole size increases, the

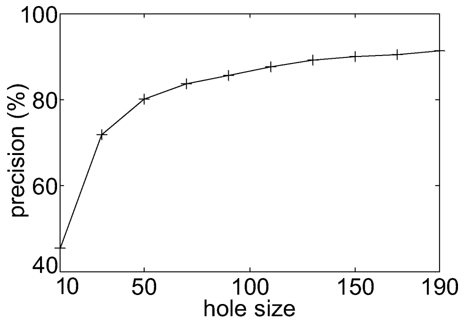


Fig. 19. Hole detection precision vs. hole size.

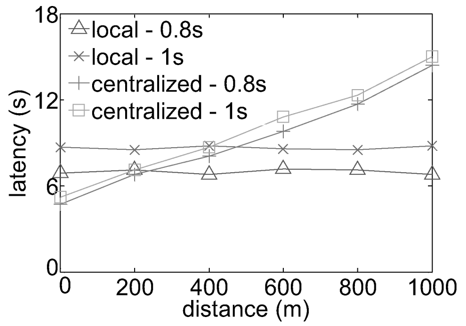


Fig. 20. Reconfiguration latency vs. hole distance.

outline of the edge nodes actually becomes tighter, and the detection precision is dramatically increased.

In our next experiment, we compare the reconfiguration latencies of the local algorithm and the centralized algorithm. A hole containing 30 nodes is assumed, located at different distances from the sink. The nodes in the hole are displaced to other places but kept unseparated, creating the worst case scenario for the local recovery algorithm in terms of convergence time. Two beacon intervals are tested (0.8s and 1s).

Figure 20 plots the results. Clearly, when the hole is close to the sink, the centralized algorithm benefits from rapid information collection and reaction from the sink, and has a shorter latency. When the distance of the hole increases, the reconfiguration latency increases linearly in the centralized algorithm, due to the increase of the round trip time from the sink. The local algorithm is not affected, and its latency is determined by the beacon interval. The combination of the two algorithms provides good reconfiguration latency for the whole distance axis.

Here we must mention that for these three tests, the communication channel is assumed to have a packet loss rate of 15%, which comes from our prototype experiment. Apparently, such a constant communication quality is not always realistic in the real environment where the traffic distribution is imbalanced, especially in the edge node reporting phase where the report messages are triggered and congregated almost simultaneously. However, while the traffic

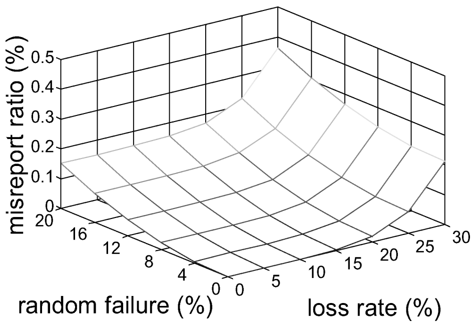


Fig. 21. Misreport ratio vs. packet loss rate and node failures.

model and interference relationship are obscure and hard to determine, we choose to simplify this factor and hope to gain an elementary knowledge of the characteristics of our scaled system.

The system stability is also investigated by varying the wireless channel loss rate and artificially introducing random node failure into the system. In Figure 21, the loss rate is the packet loss rate between any two communicating nodes, and the random failure rate is the ratio of artificially-introduced node failures per simulated minute. The misreport ratio increases as the two parameters increase. Based on these observations, we are able to carefully select a proper beacon interval for a specific application workload to balance communication quality and processing latency. We can make a shorter beacon interval to reduce the processing latency if the application workload is light, or make a longer beacon interval to reduce the packet loss rate if the application workload is heavy, while the application is tolerant to the processing latency.

We further test the performance of the proposed signature file-based approach in SASA for robust query handling. As we discussed in Section 4.7, the compression ratio of our compression method is affected by the number of “1” bits and the block representation size. In this simulation, we test the effects of block representation size k on the compression ratio. Indeed, the block representation size should be determined based on the number of “1” bits in the signature files. With the increase of the number of “1” s in the signature file, the block size should be reduced to avoid using more information to represent separated “0”s in each block. Figure 22 plots the compression ratio with a different percentage of “1” bits in the signature files. The number above each data point denotes the optimal k value selected. Figure 22 confirms our expectation that k will decrease with the increase of the percentage of “1” bits in the signature files, in order to maintain an optimal compression ratio. In our simulation, more than 80% of the signature files in the transmission achieve up to 70% compression, as most of the signature files during the transmission contain less than 5% “1” bits.

In Figure 23, we contrast the performance of signature files with and without (original) a dynamic bucket allocation technique. In the range queries, sensor IDs naturally form a uniform distribution so that no bucket collision will occur.

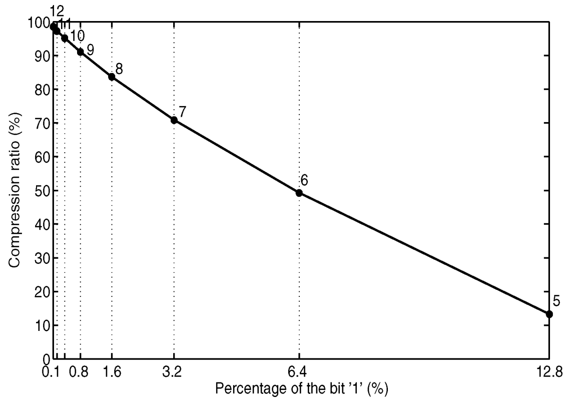


Fig. 22. Compression ratio of signature files.

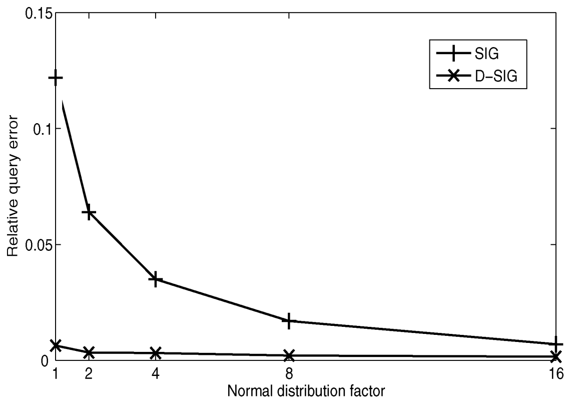


Fig. 23. SIG vs. D-SIG approach.

Therefore, we run the comparison test over aggregative queries. The results of aggregative queries over sensing values, which follow a normal distribution, are reported in Figure 23. The relative query error of the original signature file approach (SIG) and the approach with dynamic bucket allocation (D-SIG) are shown with different distribution factors, δ. Compared to SIG, D-SIG significantly reduces query error when δ is small. Since small δ indicates a concentrated distribution of the sensing values, the results confirm that the dynamic bucket allocation technique indeed avoids bucket collisions by dynamically allocating the data buckets based on the data distribution.

We compare the traffic overhead of the four approaches for answering aggregative queries over a 1000 node network: (1) TAG [Madden et al. 2002], in which the network is organized into a tree structure; (2) LIST, a multi-path approach in which each sensor node simply aggregates all received items in a list and removes duplicates. (3) SKETCH [Considine et al. 2004, Nath et al. 2004], also a multi-path approach in which each sensor node aggregates the statistical data into sketches. The sink extracts the results from the final aggregated sketches; (4) SIG: Our signature file-based multi-path approach. The sensing values for each node are selected to be uniformly random from [0, 10000]. The traffic overhead is measured by the number of packets, because as observed in our experiments as well as previous experiences, in terms of power consumption, the traffic overhead is largely determined by the number of packets transmitted rather than the actual number of bytes during the communication. A TinyDB packet (up to 48 bytes) is assumed as the basic transmitting carrier. For the TAG approach, an aggregated data value contained in one packet is transmitted from each sensor node to its parent. The LIST approach aggregates (ID, value) items into a list, thus its message length is 8 bytes long for a single item (4 bytes each). The SKETCH approach brings 20 sketches together to improve the estimation accuracy. The length of each sketch is set to 4 bytes as specified in Considine et al. [2004]. In our SIG approach, a compressed signature file is included in the transmitted packet. The total number of transmitted packets of TAG, SIG, SKETCH, and LIST are 1000, 1886, 2000, and 7534, respectively. As we expected, the TAG approach achieves the lowest traffic. The tree topology communication strategy of TAG incurs no extra cost while TAG has poor performance for query accuracy. The LIST approach's performance is the worst because it blindly combines all items in the transmission. The SKETCH and SIG approaches lie in between, due to their aggregation approaches.

6. CONCLUSION AND FUTURE WORK

In this article, we discuss SASA, a Structure-Aware Self-Adaptive wireless sensor network system, for underground monitoring in coal mines. By regulating the mesh sensor network deployment and formulating a collaborative mechanism based on the regular beacon strategy, SASA is able to rapidly detect structural variations caused by underground collapses. The collapse holes can be located and outlined, and the detection accuracy is bounded. We provide a set of mechanisms to discover the relocated sensor nodes in the hole region. We further provide a robust query handling approach for unstable network conditions. The proposed signature file-based approach explores the multi-path effect in the network and performs accurately and efficiently.

We deployed a prototype in the coal mine to test system validity. System error was measured during both the detection and reconfiguration processes. The detection latency, packet loss rate, and network bandwidth were also measured. Based on the data we collected in experiments, we conducted a large-scale simulation to evaluate the system scalability and reliability.

Several issues remain to be addressed further. First, when a collapse occurs, the stationary mesh network could be ruined and become unreliable, then the mobile nodes carried on miners or tramcars could be utilized as intermediate supporters. How to organize mobile nodes to form efficient collaborative groups is a challenging issue. Second, the proposed mechanism only detects single holes. Since multi-collapses and aftershocks are possible and have happened in underground tunnels, extending this work beyond single-hole detection is of great importance. These efforts are currently in progress in our lab.

REFERENCES

AHMED, N., KANHERE, S. S., AND JHA, S. 2005. The holes problem in wireless sensor networks: A survey. *ACM SIGMOBILE Mobile Comput. Commun. Rev. 9*, 4–18.

AKYILDIZ, I. F., SU, W., SANKARASUBRAMANIAM, Y., AND CAYIRCI, E. 2002. A survey on sensor networks. *IEEE Commun. Mag. 40*, 102–114.

CHAKRABARI, A., SABHARWAL, A., AND AAZHANG, B. 2004. Multi-hop communication is order-optimal for homogeneous sensor networks. In *Proceedings of the 3rd IEEE/ACM International Conference on Information Processing in Sensor Networks*.

CHEEKIRALLA, S. 2005. Wireless sensor network-based tunnel monitoring. In *Proceedings of the Workhop on Real-World Wireless Sensor Networks*.

CONSIDINE, J., LI, F., KOLLIOS, G., AND BYERS, J. 2004. Approximate aggregation techniques for sensor databases. In *Proceedings of the 20th International Conference on Data Engineering.*

DAVIS, R. S. AND KAMAMOHANARAO, K. 1983. A two-level superimposed coding scheme for partial match retrieval. *Inform. Syst. 8*, 4, 273–280.

DOUGLAS, S., COUTO, D., AND MORRIS, R. 2001. Location proxies and intermediate node forwarding for practical geographic forwarding. Tech. Rep. MIT Laboratory for Computer Science MIT-LCS-TR-824.

FALOUTSOS, C. 1985. Access methods for text. *ACM Comput. Surv. 17*, 1, 49–74.

FANG, Q., GAO, J., AND GUIBAS, L. 2004. Locating and bypassing routing holes in sensor networks. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*.

FEYERSTEIN, M. J., BLACKARD, K. L., RAPPAPORT, T. S., SEIDEL, S. Y., AND XIA, H. H. 1994. Path loss, delay spread, and outage models as functions of antenna height for microcellular system design. *IEEE Trans. Vehic. Tech. 43*, 487–498.

GUI, C. AND MOHAPATRA, P. 2004. Power conservation and quality of surveillance in target tracking sensor networks. In *Proceedings of the 10th ACM Annual International Conference on Mobile Computing and Networking*.

HE, G. AND HOU, J. C. 2005. Tracking targets with quality in wireless sensor networks. In *Proceedings of the 13th IEEE International Conference on Network Protocols*.

HILL, J. AND CULLER, D. 2002. Mica: A wireless platform for deeply embedded networks. *IEEE Micro. 22*, 12–24.

KARP, B. AND KUNG, H. T. 2000. Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th ACM Annual International Conference on Mobile Computing and Networking*.

LAZOS, L., POOVENDRAN, R., AND CAPKUN, S. 2005. Robust position estimation in wireless sensor networks. In *Proceedings of the 4th IEEE/ACM International Conference on Information Processing in Sensor Networks*.

LIAN, J., CHEN, L., NAIK, K., OZSU, M. T., AND AGNEW, G. 2005. Localized routing trees for query processing in sensor networks. In *Proceedings of the 14th ACM Conference on Information and Knowledge Management*

MADDEN, S., FRANKLIN, M. J., AND HELLERSTEIN, J. M. 2002. TAG: A tiny aggregation service for ad hoc sensor networks. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (USENIX)*.

MAINWARING, A., POLASTRE, J., SZEWCZYK, R., CULLER, D., AND ANDERSON, J. 2002. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM Workshop on Wireless Sensor Networks and Applications*.

MELODIA, T., POMPILI, D., AND AKYILDIZ, I. F. 2004. Optimal local topology knowledge for energy efficient geographical routing in sensor networks. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*.

NATH, S., GIBBONS, P. B., SESHAN, S., AND ANDERSON, Z. R. 2004. Synopsis diffusion for robust aggregation in sensor networks. In *Proceedings of the 2nd ACM International Conference on Embedded Networked Sensor Systems*.

RAJENDRAN, V., OBRACZKA, K., AND GARCIA-LUNA-ACEVES, J. J. 2003. Energy-efficient collision-free medium access control for wireless sensor networks. In *Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems*.

RAY, S., UNGRANGSI, R., PELLEGRINI, F. D., TRACHTENBERG, A., AND STAROBINSKI, D. 2003. Robust location detection in emergency sensor networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*.

SAVVIDES, A., HAN, C., AND SRIVASTAVA, M. B. 2001. Dynamic fine-grained localization in ad hoc networks of sensors. In *Proceedings of the 7th ACM Annual International Conference on Mobile Computing and Networking*.

SUBRAMANIAN, S. AND SHAKKOTTAI, S. 2005. Geographic routing with limited information in sensor networks. In *Proceedings of the 4th IEEE/ACM International Conference on Information Processing in Sensor Networks*.

VURAL, S. AND EKICI, E. 2005. Analysis of hop-distance relationship in spatially random sensor networks. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*.

WAN, C. Y., EISENMAN, S. B., CAMPBELL, A. T., AND CROWCROFT, J. 2005. Siphon: Overload traffic management using multi-radio virtual sinks in sensor networks. In *Proceedings of the 3rd ACM International Conference on Embedded Networked Sensor Systems*.

WIESELTHIER, J. E., NGUYEN, G. D., AND EPHREMIDES, A. 2000. On the construction of energy-efficient broadcast and multicast trees in wireless networks. In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies*.

XU, N., RANGWALA, S., CHINTALAPUDI, K. K., GANESAN, D., BROAD, A., GOVINDAN, P., AND ESTRIN, D. 2004. A wireless sensor network for structural monitoring. In *Proceedings of the 2nd ACM International Conference on Embedded Networked Sensor Systems*.

Received June 2007; revised November 2007 and January 2008; accepted February 2008