Network Resilience in Low-Resource Mobile Wireless Sensor Networks*

Bai Li, Lynn Margaret Batten, and Robin Doss

Deakin University,Victoria 3125, Australia
{libai,lmbatten,robin.doss}@deakin.edu.au

Abstract. Wireless sensor networks (WSNs) are deployed in numerous mission critical applications in which the network needs to remain active for as long as possible while delivering quality information to a base station. However, WSNs suffer from a wide range of attacks due to their limited processing and energy capabilities. Their resiliency, however, depends on fast recovery from such attacks being achieved. In recent work, the authors developed and implemented clustering, reprogramming and authentication protocols involved in recovering stationary WSNs with low resources. In this paper, we determine the additional resources required in implementing these protocols in a mobile WSN.

We present recovery protocols on TinyOS motes for a low-resourced, mobile deployment. We describe the issues we encountered in the implementation. We present times, RAM and ROM needed to run the recovery protocols and compare these with the stationary case, demonstrating that the additional cost of reprogramming in a mobile WSN is less than 25% of that in a stationary WSN and the additional cost of re-clustering in a mobile WSN is less than 9% of that in a stationary WSN. Authentication has an insignificant cost increase.

Keywords: Wireless sensor network, resilience, recovery, implementation, reprogramming, re-clustering, authentication.

1 Introduction

Wireless sensor networks (WSNs) are deployed in many mission critical applications and for monitoring of critical areas of national defence. With their development, various security attacks have appeared, usually with the aim of taking over nodes in the network, destroying nodes or disrupting data flow. Detection, response to and recovery from these attacks have become major challenges in protecting sensor networks.

We define WSN resiliency as the ability of the network to:

$R1$. Restore damaged network functions to a target level established by the base station.

$R2$. Maintain the network sensor operations for as long as possible.

* Suported by NSST grant 070030.

A.U. Schmidt et al. (Eds.): MobiSec 2010, LNICST 47, pp. 127–138, 2010.

While stationary WSNs have been studied from a recovery viewpoint [1], a more difficult problem is recovering WSNs which are mobile. There are numerous scenarios in which mobility can occur. One such is the deployment of a WSN in a coal mine subject to collapses studied in [2]; in this case, motes fall into holes or move towards a hole and transmission paths must be reconnected. Other mobile situations see motes deployed in vehicles or on people or animals. The main problem distinguishing this situation from the stationary one is finding and maintaining transmission paths.

In this paper, we focus on recovery in a mobile WSN after an attack has been detected and after preliminary assessment (response) has been completed. We assume that the network, comprising a fixed Base Station (BS) and mobile sensor nodes, is low resourced but has the capability to detect an attack on the nodes and determine which nodes have been compromised (with high probability). Thus, we may assume that the BS can set target operational levels for the WSN and determine what steps need to be taken to return to these target levels.

We also assume that the range of mobility of a node is limited in some respect (we use the waypoint method described in [3,4]). We then introduce protocols which assist the BS in securely reprogramming compromised nodes and assist the WSN to securely re-determine data transmission paths and self-organize (cluster and re-cluster).

We implemented our recovery techniques on the TelosB platform based on TinyOS using the standards-based protocols ZigBee and IEEE 802.15.4 along with the off-the-shelf software Deluge [5] for reprogramming. We discuss the challenges we faced in detail. We also present results on times, RAM and ROM used in the implementation and compare these with those needed in a comparable stationary WSN. Finally, we offer suggestions for those implementing recovery on low-resource mobile WSN platforms.

The rest of the paper is outlined as follows. In Section 2, we discuss related work. In Section 3, we summarize our reprogramming, re-clustering and authentication protocols. Section 4 discusses the implementation issues and provides data comparing the mobile and stationary situations. In Section 5, we summarize and draw conclusions as well as mentioning possible future work.

2 Related Work

Motion of sensors in a WSN can have various goals. It may be used in order to position sensors for optimal data recovery [6], as in detection of leaking gas: sensors attempt to locate the positions from which the gas is leaking by searching for maximum emission readings. Motion may be used to allow sensors to track a target as in deployment of robots in a battle-field to locate a strategic bridge [7]. Moving sensors may also be used to detect anomalies in a changing environment as in searching for potholes in city road surfaces [8].

When sensors are in motion, there are important issues which must be dealt with. One is: should sensors be able to move at random or should there be constraints on the paths they can follow? A second is: when sensors are moving,

how does the network remain connected so that information can be passed from the sensors back to the base station via a suitable route?

2.1 The Random Waypoint Mobility Model

The issues above are connected. If sensors are allowed to move in a random fashion, then they run the risk of collecting in a small area of the coverage space and not sampling correctly. They even run the risk of collision and damage. Thus it is critical to guide sensor motion in a mobile WSN. A well-known and much studied method for doing this is the random waypoint mobility model introduced in [4]. In this model, several uniformly distributed fixed target points, referred to as waypoints, are selected in the sensor domain. The domain is assumed to be convex so that any straight line segment between two waypoints remains inside it. Nodes then move within the domain in straight-line segments from one waypoint to another. The waypoints are chosen so that information from anywhere in the domain can be retrieved by sensors as they move along the designated paths at pre-set velocities [9,10]. In some scenarios, for example [11], in order to ensure optimal coverage of the sensor domain, some sensors may remain stationary while others move.

2.2 Connectivity

Maintaining connectivity of the WSN so that sensor data can be returned regularly from the nodes to the BS in order to maintain routine operation is of critical importance. There are several approaches to connectivity of a mobile WSN in the literature. Ma and Yang [12] divide the sensor domain into triangles and assume that nodes always move within a small range of some triangle vertex. While ensuring that any sensor node is always in range of several others, this restricts the motion of sensors more severely than that in the random waypoint model.

Wu [13,14] has developed a method of determining a connected set of nodes within a WSN by having the nodes keep track of 1- and 2-hop neighbors. In a mobile situation, the tables must be re-determined periodically according to how the network is changing. If nodes are mostly stationary with only a few moving from time to time, the BS can trigger re-determination of the neighbor tables as soon as it knows that movement has taken place. In case nodes are constantly moving, it is better to set up time periods within which nodes regularly re-establish neighbor tables.

2.3 Route Discovery

Determining a route through a WSN from a node to the BS so that information collected can be delivered in a timely fashion is an important issue for stationary WSNs, but a critical one for mobile WSNs.

Dynamic Source Routing (DSR) is a widely used routing protocol [3] in which the source node specifies the complete ordered route in the packet header before

sending the packet data. It consists of two mechanisms: Route Discovery and Route Maintenance. Route Discovery is the mechanism whereby a source wishing to send a packet to a destination node obtains a source route to the destination node; Route Maintenance is the mechanism whereby the source node is able to re-establish a route if the network topology has changed. DSR is commonly chosen because of its simplicity and performance [15].

Wu [13,14] solves the problem of connectivity and route discovery simultaneously using Connected Dominating Sets (CDS) based on the enhanced multi-point relay (EMPR) algorithm. This algorithm efficiently partitions a WSN with a flat topology into a hierarchical network consisting of a set of small-sized clusters of nodes around an aggregator. It can be shown that the selected aggregators are connected among each other and also that every node is either an aggregator or a neighbor of an aggregator; ***such a set of aggregators is called a CDS***. The method determines in a distributed, localized fashion a set of aggregators by firstly collecting 2-hop neighborhood information in each node of the WSN, then selecting by iteratively searching for the best-suited set of multi-point relays, and finally associating cluster members to an appropriate aggregator.

3 Our Approach

3.1 Assumptions

We assume a single, secure and trustworthy BS along with nodes capable of operating as either member nodes or aggregator (AG) nodes. The WSN is clustered into groups of nodes each monitored by an AG. An AG node stores identifying information about those nodes in its cluster. All messages transmitted in the WSN identify the message source. Member nodes gather data which is then sent to aggregator nodes. Aggregator nodes both gather data and aggregate collected data before sending it to the BS. The BS analyses and stores data and keeps logs of this process. We assume that an essentially infinite timer is available to each of the nodes and the BS. (In practice, timers on nodes may overflow and re-use times which invalidates our protocol.)

We assume the existence of a globally unique ID for each sensor node. The base station keeps track of all IDs. In addition, we assume that the BS shares with each WSN node a common secret, known to no other nodes, which is allocated when the WSN is set up. These secrets are used to implement authentication when messages are sent. We also assume that this secret is stored in a tamper-resistant section of the node and that calculations involving it are executed in this tamper-proof section. (Note that the use of a tamper-proof or secure area for storing secrets and executing computations with them is a standard solution to key management.)

We assume that the same reprogramming, re-clustering and hash function code is programmed into each node at set-up and does not have to be transmitted during network operation. We pre-store all necessary code in the tamper-proof section of the mote.

The sensors are permitted to move, in some orderly pre-defined way, around a convex area which we refer to as the sensor domain. We assume that, at any time, some node is within range of the BS; however, many nodes may not be in range of the BS.

It is also necessary to assume that at any time, at least one node has the BS within range. This is critical for the construction of a CDS in the following sub-section.

3.2 Constructing Connected Dominating Sets

We adopt Wu's method of determining a CDS [13] and, in order to support the resiliency requirement $R2$, we employ both the battery energy level and the number of neighbors of each node in deciding which should be aggregators. This adaptation of Wu's algorithm was presented in [1] and is given again below for the reader.

A 1-hop neighbor of a node v is the set of nodes within range of v. A 2-hop neighbor of v is the set of nodes within range of the 1-hop neighbors of v.

After collecting the neighborhood information, each node v selects a set of nodes that can be viewed as candidate AGs. This set comprises a small subset of nodes $C(v)$ from the 1-hop neighbor set $N1(v)$ of node v that fully covers the 2-hop neighbor set $N2(v)$ of node v. (A set S *fully covers* $N2(v)$ if every node of $N2(v)$ is in range of some node of S.) $C(v)$ is thus also called the *coverage set* of node v, and it can be shown that the $C(v) \cup v$ forms a CDS for $N2(v)$. The coverage set $C(v)$ is obtained by executing the modified EMPR algorithm given below that takes into account a known metric value $M(v)$ associated with each node v. $M(v)$ is a function of the energy level of the node and of the number of 1-hop neighbors of the node. The higher each of these values is, the greater the chance of the node being chosen as an AG.

Enhanced Multi-point Relaying Algorithm

1. Add all free neighbors of $N1(v)$ to the coverage set $C(v)$. Node u is a free neighbor of v if v is not the highest metric neighbor of u.
2. Add node $u \in N1(v)$ to the coverage set $C(v)$ if there is an uncovered node in $N2(v)$ that is only covered by u. Any node in $N2(v)$ that is not covered by $C(v)$ is called an uncovered node.
3. Add node $u \in N1(v)$ to the coverage set $C(v)$ if u covers the largest number of uncovered nodes in $N2(v)$. Use the node metrics to break a tie when two nodes cover the same number of uncovered nodes.
4. Repeat step 3 until all nodes in $N2(v)$ are covered.

After the multi-point relays have been selected, each node broadcasts its coverage set $C(v)$ to its 1-hop neighbors at a random time instant in the next time interval. At this point, a node v decides to act as an AG if -

1. it has a larger metric $M(v)$ than all its 1-hop neighbors and has at least two unconnected neighbors, or
2. it is in the coverage set formed by its neighbor with the largest metric.

The set of all such AGs forms a CDS [13,14] for the WSN.

It is important for the recovery protocols that at least one member of the CDS be able to communicate directly with the BS; that is, the BS is within its range. If this is not the case, one additional mote which can reach the BS is added to the CDS. This additional mote is chosen to have maximum metric from among all motes which can reach the BS. The CDS properties are still held by this new set. ***In what follows, when we refer to a CDS, we assume that this additional property is satisfied.***

3.3 Recovery

A major part of network resiliency deals with recovery after accidents or attacks. In this sub-section, we present our protocols to reprogram compromised nodes and to re-cluster when routes between nodes and to the BS are broken or may become broken.

By ensuring that the BS is always in range of at least one member of the CDS constructed in subsection 3.2, network operation is maintained while the recovery procedures of re-clustering and node reprogramming are carried out. In the mobile setting, neighbor tables are updated while the nodes are in motion. The accuracy of these tables in this case, is not as high as in the stationary case. However, our experiments show that the proposed approach is robust enough to minimise the impact of both AG and member node mobility on the recovery process.

In each case, a node with ID n contains secret Sn (shared with the BS), R represents a random value but in practice is the local time obtained from the LocalTime.get() command in TinyOS, M represents a message to reprogram or re-cluster or a request that another node be reprogrammed. All messages transmitted include node ID of both sender and receiver, including that of the BS. A hash function check achieves authentication of a message. The two hash functions SHA-1 [16] and Rabin [17,18] were chosen and both used in each protocol to compare their performance.

The first protocol is used when the BS determines that a node must be reprogrammed. Because the node may not be in range of the BS, in part (c), the BS can use the connected set of AGs, at least one of which is in its range. In (f), the node can transmit confirmation back to the BS through the CDS. Reprogramming and confirmation messages are authenticated to protect against denial of service attacks in which an attacker sends reprogramming messages into the network.

3.3.1 Mobile Reprogramming Protocol from the Base Station to a Compromised Node

(a) *BS XORs the secret Sn, the reprogramming message M and the local time R to obtain* $Sn \otimes M \otimes R = m$.
(b) *BS computes* $h(m) = c$.
(c) *BS transmits c, M and R to n through the CDS formed by the AGs.*
(d) *Node n re-computes* $h(Sn \otimes M \otimes R)$ *in its tamper-proof section, and checks that it is c.*

(e) *If the check is 'true' AND the time R has not been used in a reprogramming request earlier, the node initiates reprogramming procedures.*
(f) *The node confirms to the BS that it has reprogrammed by computing* $h(Sn \otimes M) = c'$ *and sending* c' *to the BS through the CDS formed by the AGs.*

In the next protocol, an AG determines that a node needs reprogramming and sends this request to the BS. The BS then initiates reprogramming. Again, the CDS is used for transmitting messages ((c) and (e)), and messages are again authenticated.

3.3.2 Mobile Reprogramming Protocol from an Aggregator Node to the Base Station on Behalf of a Compromised Node

(a) *The AG retrieves the ID n of the node to be reprogrammed.*
(b) *The AG computes* $h(n \otimes SA \otimes M \otimes R) = c$ *in its tamper-proof section.*
(c) *The AG transmits c, n, M and R to the BS through the CDS of AGs.*
(d) *The BS retrieves SA, re-computes the hash and checks if it is c.*
(e) *If the check is 'true' AND the time R has not been used in a reprogramming request earlier, BS initiates the mobile reprogramming protocol through the CDS formed by the AGs.*

In Protocol 3.3.3, the BS re-clusters the WSN by transmitting an authenticated re-clustering message to the network through the CDS in (b).

3.3.3 Mobile Authentication of a Re-clustering Command from the Base Station to the Network

(a) *The BS retrieves the secrets* $S1, \ldots Sc$ *of each node and for each node ID n, computes* $h(Sn \otimes M \otimes R) = cn$.
(b) *The BS then transmits cn with M and R to the corresponding node n through the CDS formed by the AGs.*
(c) *Each node n computes* $h(Sn \otimes M \otimes R)$ *in its tamper-proof area to confirm that the message is authentic.*
(d) *If there is a match AND if no such message with time R has been used in a re-clustering request earlier, n accepts this as a valid re-clustering message.*
(e) *Once all nodes have received and verified such a message, re-clustering commences.*

In the next section, we describe implementation of these protocols based on a waypoint mobility model with TelosB motes. We compare times, RAM and ROM with the use of the same protocols in a stationary TelosB network. In the stationary situation, one CDS is initially constructed and used throughout the testing. In the mobile case, the BS has the knowledge of node movement and either re-clusters the WSN after each node movement or at regularly spaced time intervals.

4 Experimental Results

We tested Protocols 3.3.1, 3.3.2 and 3.3.3 on TinyOS Version 2.1 using the Crossbow TelosB research mote as our experimental testbed platform. The results are presented in the three tables below. We tested using stationary motes and then mobile motes, in order to demonstrate the time differences between the two cases. Identical code was used. In the mobile case, motes were mounted on top of remote-controlled toy car models, As the code remains constant in both situations, there is no difference in RAM and ROM size between the stationary and mobile scenarios.

For Protocols 3.3.1 and 3.3.2, we used a rectangular area of size 10 metres by 8 metres with, as an initial set-up, 20 sensor nodes distributed in a grid pattern inside it uniformly at 2 metres apart horizontally and 2 metres apart vertically. In the mobile case, we moved the sensors using the waypoint model discussed in Section 2.1. See Figure 1.

In our tests, waypoints were also established in a uniform grid pattern along and within the boundary of the area at 2 metre intervals both horizontally and vertically. (In general, waypoints might be distributed in any uniform configuration across the sensor domain.) Each node has a radio power range of 2.5 meters and so, in the initial set-up, is within range of several nodes in the grid.

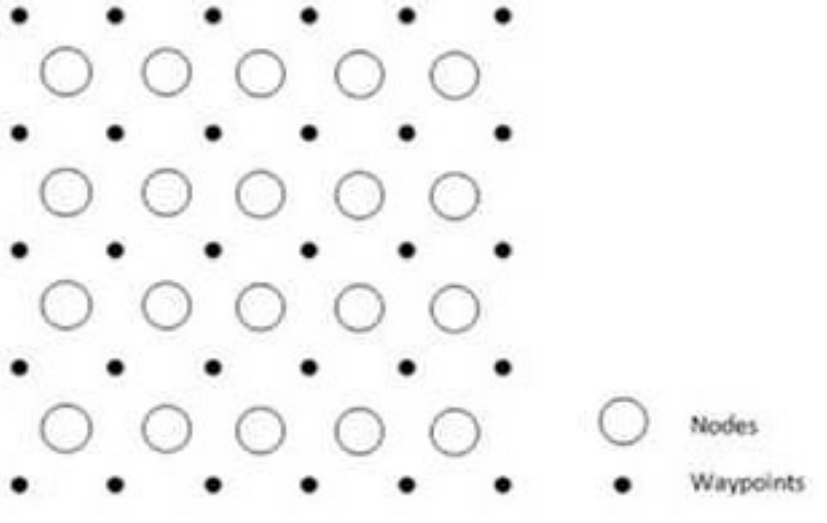

Fig. 1. Description of mote and waypoint location

Remote control devices such as toy cars are restricted by IEEE standards to the 40 MHz and 72MHz frequency bands. Thus, in order to avoid frequency interference, we were able to move at most two cars at any given time. Different motes were chosen to start moving in each experiment. When in motion, nodes moved towards waypoints at an average velocity of 0.5 metres per sec. When a

waypoint was reached, the mote stopped for a short interval of between 0.5 and 3 secs in order to determine its next direction before moving on to the next randomly chosen waypoint. Mote direction was based on the grid pattern in Figure 1. Motes moved towards nearest waypoints up to 2 metres away. This allowed up to 4 directions of movement for a given mote. Randomness was introduced by blind choice of one of 4 counters indicating the next move. If the move chosen was not possible (as for a mote on a side of the grid for instance), it was discarded and a second choice made.

Table 1 shows the differences in execution time of implementing Protocol 3.3.1 in both the stationary and mobile situations using several versions of hash function. Neighbor tables were not used here, so not computed. The results are averages over twenty executions. Time starts when the BS in Protocol 3.3.1 or AG in Protocol 3.3.2 issues its first message and stops when the compromised node finishes the reprogramming process after the hash verification.

Table 1. Comparative performance of the authenticated mobile reprogramming Protocol 3.1 from the base station to a compromised node

Primitives	Execution Time (secs)		Program Size (bytes)	
	Stationary WSNs	Mobile WSNs	RAM	ROM
Reprogramming based on Rabin 32 bit	12.79	15.87	2122	40506
Reprogramming based on Rabin 64 bit	12.85	15.93	2182	41100
Reprogramming based on SHA-1 32 bit	12.94	16.06	2344	43330
Reprogramming based on SHA-1 64 bit	12.97	16.09	2416	44196

Average execution time in the stationary case is 12.89 and in the mobile case is 15.99, each rounded to two decimal places. In the former, maximum variation is 0.18; in the latter it is 0.22. The additional percentage performance for the mobile case over the stationary case is 24.05, rounded to two decimal places.

The hash function Rabin is consistently (slightly) faster than SHA-1. The next table gives similar data for the Protocol 3.3.2. Again, Rabin is slightly faster.

Table 2. Comparative performance of the authenticated mobile reprogramming Protocol 3.2 from an aggregator node to the base station on behalf of a compromised node

Primitives	Execution Time (secs)		Program Size (bytes)	
	Stationary WSNs	Mobile WSNs	RAM	ROM
Reprogramming based on Rabin 32 bit	15.92	18.39	2164	41300
Reprogramming based on Rabin 64 bit	15.94	18.48	2224	41892
Reprogramming based on SHA-1 32 bit	15.95	18.69	2378	44036
Reprogramming based on SHA-1 64 bit	15.98	18.74	2450	44880

In this case, average execution time in the stationary case is 15.95 and in the mobile case is 18.58. The maximum variations are 0.06 and 0.35 respectively. The additional percentage performance for the mobile case over the stationary case is 16.49, rounded to two decimal places.

For Protocol 3.3.3, we continued to apply the random waypoint model and again ran twenty tests. One or two nodes were in motion at any time when authentication was used when the re-clustering process started.

Table 3 presents the average results of running Protocol 3.3.3 with Rabin twenty times with sets of 10, 15 and 20 TelosB motes. For authentication, time starts when the BS issues its first message and stops when the last node verifies the hash message. The authentication time increased as the network size increased.

In each case, nodes were arranged in a grid, 2 metres horizontally and 2 metres vertically from neighbors. For 20 nodes, the configuration is as in Figure 1. For 15 nodes, a $3 * 5$ grid was used; for 10 nodes, a $2 * 5$ grid was used. For each of these last cases, waypoints were distributed as in the top four and three layers respectively of waypoints in Figure 1. (18 waypoints with 10 nodes and 24 waypoints with 15 nodes.)

Table 3. Comparative performance of Rabin authentication and re-clustering from the base station to the network

Primitives	Execution Time for Authentication (secs)		Execution Time for Re-clustering (secs)		Program Size (bytes)	
	Stationary WSNs	Mobile WSNs	Stationary WSNs	Mobile WSNs	RAM	ROM
10 Nodes	58.13	63.23	87.67	87.63	9904	46118
15 Nodes	79.20	85.37	87.79	87.82	9904	46118
20 Nodes	93.46	99.58	87.21	87.57	9904	46118

For re-clustering, time starts when the BS issues a re-clustering command and stops when AGs have been chosen and every node belongs to a cluster. From Table 3, we see that the network size has an impact on the authentication component of the protocol, but not on the actual re-clustering. This is likely because authentication occurs in series through the BS while much of re-clustering occurs at a local level running in parallel.

The additional percentage performance for the mobile case over the stationary case is 8.77 with 10 nodes, 7.79 with 15 nodes and 6.55 with 20 nodes. One might speculate that the difference becomes unobservable for very large WSNs.

Testing this set-up for Protocol 3.3.3 twenty times resulted in the following CDS sizes. For 10 nodes, average CDS size is 5 AGs; the smallest CDS achieved was 4 and the largest was 5. For 15 nodes, average CDS size is 6 AGs; the smallest CDS achieved was 5 and the largest was 7. For 20 nodes, average CDS size is 6 AGs; the smallest CDS achieved was 5 and the largest was 8.

All the experiments related to mobility produced results slightly slower than in the stationary case, as evidenced in the three tables. Some of this is likely due to a combination of radio interference along with the dynamic transmission distance in the mobile case. All in all, the results in the mobile case remain acceptably close to those in the stationary situation.

5 Conclusions

In distinguishing between the ability of stationary versus mobile networks to execute recovery protocols to support network resiliency, we have demonstrated on TelosB motes that the additional cost of reprogramming in a mobile WSN is less

than 25% of that in a stationary WSN and the additional cost of re-clustering in a mobile WSN is less than 9% of that in a stationary WSN. Authentication has a marginal cost increase. In addition, it appears that as the network grows, authentication costs become indistinguishable in the two scenarios. Since both reprogramming and re-clustering are expected to occur in WSN resiliency maintenance, the average cost over a long period is expected to be under 15% greater in the mobile case than in the stationary case.

Issues to consider when developing such an implementation include appropriate motion control for motes and consideration for correct assessment of neighbors in a moving system. In further work, deployment of larger mobile WSNs in a variety of configurations would be of interest; also, a waypoint model for non-convex domains would be useful for real applications. Appropriate means of providing tamper-resistance on the nodes is, in addition, an area which needs some attention.

References

1. Li, B., Doss, R., Batten, L., Schott, W.: Fast Recovery from Node Compromise in Wireless Sensor Networks. In: NTMS 2009, pp. 186–191 (2009)
2. Li, M., Liu, Y.: Underground structure monitoring with wireless sensor networks. In: IPSN 2007, Cambridge, Mass., pp. 69–78 (2007)
3. Johnson, D.: Routing in ad hoc networks of mobile hosts. In: Proceedings of Mobile Computing Systems and Applications, pp. 158–163. IEEE Computer Society, Los Alamitos (1994)
4. Johnson, D., Maltz, D.: Dynamic source routing in ad hoc wireless networks. In: Imielinski, T., Korth, H. (eds.) Mobile Computing, ch. 5, vol. 353, pp. 153–181. Kluwer Academic Publishers, Dordrecht (1996)
5. Wang, Q., Zhu, Y., Cheng, L.: Reprogramming wireless sensor networks: Challenges and approaches. IEEE Network Magazine 20(3), 48–55 (2006)
6. Kansal, A., Kaiser, W., Pottie, G., Srivastava, M., Sukhatme, G.: Reconfiguration methods for mobile sensor networks. ACM Transactions on Sensor networks 3, Article 22, 1–28 (2007)
7. Zou, Y., Chakrabarty, K.: Distributed mobility management for target tracking in mobile sensor networks. IEEE Transactions on Mobile Computing 6, 872–887 (2007)
8. Eriksson, K., Girod, L., Hull, B., Newton, R., Madden, S., Balakrishnan, H.: The pothole patrol: using a mobile sensor network for road surface monitoring. In: Proceedings of MobiSys 2008, Colorado, pp. 29–39 (2008)
9. Alparslan, D., Sohraby, K.: A generalized random mobility model for wireless and ad hoc networks and its analysis: one-dimensional case. IEEE/ACM Transactions on Networking 15, 602–615 (2007)
10. Hyytia, E., Virtamo, J.: Random waypoint mobility model in cellular networks. Wireless Networks 13, 177–188 (2007)
11. Hu, L., Evans, D.: Localization for mobile sensor networks. In: Proceedings of MobiCom 2004, Pennsylvania, pp. 45–57 (2004)
12. Ma, M., Yang, Y.: Adaptive triangular deployment algorithm for unattended mobile sensor networks. In: Prasanna, V.K., Iyengar, S.S., Spirakis, P.G., Welsh, M. (eds.) DCOSS 2005. LNCS, vol. 3560, pp. 20–34. Springer, Heidelberg (2005)

13. Wu, J.: An enhanced approach to determine a small forward node set based on multipoint relay. In: Proc. IEEE Conference VTC 2003, pp. 2774–2777 (2003)
14. Wu, J., Lou, W., Dai, F.: Extended multipoint relays to determine connected dominating sets in MANETs. In: Proceedings of IEEE SECON 2004, vol. 55, pp. 334–347 (2004)
15. Hu, L., Li, Y., Chen, Q., Liu, J., Long, K.: A New Energy-Aware Routing Protocol for Wireless Sensor Networks. In: Proceedings of WiCom 2007, pp. 2444–2447 (2007)
16. National Institute of Standards and Technology: 'Secure hash standard', FIPS Publication 180–2. 32 pages (2002)
17. Rabin, M.: Digitalized Signatures and Public-Key Functions as Intractable as Factorization. MIT Laboratory for Computer Science, 16 pages (January 1979)
18. Shamir, A.: SQUASH – A new MAC with provable security properties for highly constrained devices such as RFID tags. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 144–157. Springer, Heidelberg (2008)