Integrated Intrusion Detection and Tolerance in Homogeneous Clustered Sensor Networks

HAMID AL-HAMADI, Kuwait University
ING RAY CHEN, Virginia Tech

In this article, we propose and analyze dynamic redundancy management of integrated intrusion detection and tolerance for lifetime maximization of homogeneous clustered wireless sensor networks (WSNs). We take a holistic approach of integrating multisource and multipath routing for intrusion tolerance with majority voting for intrusion detection in our redundancy management protocol design. By dynamically controlling the redundancy level for both multisource multipath routing and voting-based intrusion detection with energy consideration, we identify the optimal redundancy level to be applied to maximize the WSN lifetime in response to changing environment conditions including node density, radio range, and node capture rate. We demonstrate the effectiveness of our integrated redundancy management protocol by a comparative analysis with a multisource multipath routing algorithm called AFTQC that considers only fault/intrusion tolerance.

Categories and Subject Descriptors: C.2.2 [Computer-Communication Networks]: Network Protocols

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Wireless sensor networks, intrusion tolerance, intrusion detection, multisource multipath routing, security, reliability, timeliness

ACM Reference Format:

1. INTRODUCTION

Advances in wireless sensor networks (WSNs) have led to its wide deployment across many fields. Many WSN applications have quality of service (QoS) requirements in security, reliability, and timeliness. In addition, many autonomous WSNs are deployed in an unattended manner, so sensor nodes (SNs) are susceptible to capture attacks turning them into malicious inside attackers. SNs have limited resources in energy, computation, transmission range, and storage capability [Bravos and Kanatas 2005; Qun 2007]. Thus, the challenge is not only in providing designs satisfying the application-specific QoS requirements but also in a way that would consume minimum energy and prolong the lifetime. This issue is especially acute for homogeneous WSNs for which SNs of the same capability are typically deployed massively into the operational area by air drop, so SNs must consume energy at about the same rate for lifetime maximization.
Multipath routing is considered an effective way to improve data delivery success probability in WSNs [Stavrou and Pitsillides 2010]. The basic idea is that the probability of at least one path reaching the sink node or base station increases as we have more paths doing data delivery. Although most prior research focused on using multipath routing for fault tolerance to improve reliability [Chen et al. 2011; Felemban et al. 2006; Yang et al. 2010], some attention has been paid to using multipath routing to tolerate insider attacks [Deng et al. 2006; Kang et al. 2006; Lou and Kwon 2006]. Most studies, however, largely ignored the trade-off between QoS gain and energy consumption, which can adversely shorten the system lifetime.

The trade-off issue between energy consumption and QoS gain becomes much more complicated when inside attackers are present, as a path may be broken when a malicious node is on the path. In a hostile environment, it is necessary to apply intrusion detection system (IDS) design to complement passive intrusion tolerance to result in safety-critical WSNs. Monitoring-based intrusion detection with voting [Bo et al. 2007; Cho et al. 2010; Da Silva et al. 2005; Krontiris et al. 2007] to cope with collusion is a proven effective intrusion detection mechanism for detecting and evicting compromised nodes with a low false alarm probability. How often intrusion detection should be invoked to effectively remove potentially malicious nodes without excessive energy consumption for system lifetime maximization is still an open issue.

In this article, we develop a new design concept of dynamic redundancy management of integrated intrusion detection and tolerance. Intrusion tolerance is achieved by means of multisource multipath routing, whereas intrusion detection is achieved by means of majority voting. To the best of our knowledge, we are the first to address both intrusion detection and tolerance with energy consideration via dynamic redundancy management for WSNs. By means of a novel model-based analysis methodology based on probability theory, we model the trade-off between energy consumption and reliability, timeliness and security gain, and identify the optimal multisource multipath redundancy level and intrusion detection settings for maximizing the lifetime of the WSN while satisfying application QoS requirements. A main contribution of our article is that our dynamic redundancy management protocol design addresses the issue of how many paths to use in multisource multipath routing for intrusion tolerance. Another contribution is that we take an integrated approach combining intrusion detection and tolerance in the protocol design to address the issue of how much instruction detection is enough to prevent security failure and prolong the WSN lifetime.

2. RELATED WORK

Over the past few years, numerous protocols have been proposed to detect intrusion in WSNs [Bo et al. 2007]. In Da Silva et al. [2005], a decentralized rule-based IDS was proposed by which monitor nodes are responsible for monitoring neighboring nodes using promiscuous listening and monitoring the collisions for the messages they send to their neighbors. The monitor nodes apply predefined rules to collected messages and raise alarms if the number of failures exceeds a threshold value. For example, if an SN does not forward a packet within a time limit, if an SN forwards the same packet multiple times without suppression, or if a packet is received directly from a non-neighbor SN or from a neighbor SN who is not supposed to send a packet during a particular time interval, then the SN in question is suspected of maliciousness. When an SN’s “maliciousness count” exceeds a tolerance limit, the SN is diagnosed as compromised. Our host IDS essentially follows this strategy, with the flaws of the host IDS characterized by a false-positive probability ($H_{fp}$) and a false-negative probability ($H_{fn}$). In Da Silva et al. [2005], however, there was no consideration about bad-mouthing attacks by compromised monitor nodes themselves, so if a monitor node is malicious, it can quickly infect others. In Krontiris et al. [2007], a collaborative approach was...
proposed for intrusion detection where the decision is based on a majority voting of monitoring nodes. Their work, however, does not consider energy consumption issues. Our voting-based IDS approach extends from Cho et al. [2010] with considerations given to the trade-off between energy loss and security and reliability gain due to employment of voting-based IDS with the goal to prolong the WSN system lifetime.

In the area of redundancy management of intrusion detection, Misra et al. [2010] proposed a self-learning system used by SNs to detect malicious packets. An SN follows a sampling rate to sample packets in an attempt to detect malicious packets and drop them. The sampling rate is controlled such that it matches the level of compromise exhibited in the system, thus resulting in an energy-efficient system. Our work also aims to determine the best rate at which to invoke intrusion detection; however, we aim to find the best balance of reliability and security gain versus energy consumption to maximize the system lifetime.

In Chatzigiannakis and Strikos [2007], the authors proposed a hierarchal model for WSNs with cluster heads (CHs) responsible for reporting intrusion-related data to the base station, thus avoiding expensive transmission from SNs to the base station. However, this approach drains energy of CHs faster than SNs and thus may adversely shorten the lifetime span of CHs that are critical to the WSN. eHIP was proposed in Su et al. [2007] to combine intrusion prevention based on authentication with collaborative-based intrusion detection by which a CH monitors member SNs. A CH isolates SNs that behave suspiciously. CHs themselves are monitored by SN groups that accumulate suspicious activity up to a threshold, after which the CH is evicted. SN groups are formed on a rotating basis for energy conservation purposes.

In Hoang and Nam [2007], a protocol for optimal selection and activation of intrusion detection agents was proposed. Every node is assigned a trust value by its CH. Only agents with sufficient trust are allowed to activate the monitor agent and participate in the cooperative IDS by sending alerts to its CH. CHs enforce this rule and lower trust levels of misbehaving agents. Energy is conserved by limiting the number of active monitoring agents and avoiding duplicate alerts to CHs, which in turn enhances the system lifetime. Bao et al. [2011, 2012] also considered the use of trust in a hierarchically structured WSN such that a CH collects trust evaluations of SNs toward their peer SNs in the same cluster for intrusion detection in an effort to reduce energy consumption. Relative to the work cited earlier, our article takes a more holistic approach by considering energy consumption versus security gain not only for intrusion detection by means of adjusting IDS strength in response to attacker strength but also for intrusion tolerance by means of redundancy management of multisource multipath routing with the goal to maximize the WSN lifetime.

In the literature, many multipath routing protocols have been proposed for WSNs [Stavrou and Pitsillides 2010]. In Kang et al. [2006], multiple paths are used to route traffic to the destination using geographic routing, aiming to increase packet delivery ratio in the presence of packet dropping attacks (through black hole and selective forwarding). A trust-based approach is taken by which a sender uses overhearing to monitor if the next nodes forward its packets. Our work differs from theirs in that we concern not only multipath routing but also energy consumption issues to maximize the WSN system lifetime in the presence of malicious nodes performing bad-mouthing attacks and packet dropping attacks.

INSENS [Deng et al. 2006] is a disjoint multipath routing protocol that aims to tolerate intrusions by using multiple redundant paths to send a message to a destination. It aims to operate correctly in the presence of undetected intruders. However, it relies on the existence of a powerful base station to plan multipath routing, which is normally not available in WSNs, or otherwise would be a single point of failure. SEEM [Nasser and Chen 2007] is a multipath routing protocol that also relies on a powerful base
station to perform route discovery, maintenance, and route selection. However, it does not consider the existence of malicious nodes, and there is no consideration given to detect attacks. Our approach is totally distributed, with considerations given to the presence and detection of malicious nodes in the WSN.

In Shu et al. [2010], packets are sent over randomized dispersive multipath routes with the aim to avoid black holes resulting from compromised nodes performing packet dropping and/or denial of service attacks. A packet is split into \( n \) shares based on coding theory so that the packet can be reconstructed if \( k \) out of \( n \) shares are received. The randomized multipath routes generated are dispersive to avoid the black hole and to enhance the probability that at least \( k \) out of \( n \) shares can reach the receiver. However, the approach does not consider intrusion detection to detect compromised nodes. Our work considers multipath multisource routing as well to circumvent black hole attacks for intrusion tolerance. In addition, we consider intrusion detection to detect and evict compromised nodes as well as the best rate to invoke intrusion detection to best trade off energy consumption versus security and reliability gain for WSN lifetime maximization.

In the area of redundancy management of WSNs, Al-Hamadi [2013] and Chen et al. [2011] address the issue of how many paths should be used from a source node to a sink node in the context of multipath routing. In particular, in the context of homogeneous clustered WSNs, AFTQC [Chen et al. 2011] identified the best path redundancy to apply to best trade energy consumption for reliability gain to maximize the system lifetime. However, no presence of malicious nodes was considered. Relative to the work cited earlier, we consider dynamic redundancy for both fault/intrusion tolerance (via multisource multipath routing) as well as for intrusion detection (via voting) in response to changing environment conditions with the goal to maximize the WSN lifetime. In this article, we perform a comparative performance analysis of our approach against AFTQC [Chen et al. 2011].

3. SYSTEM MODEL

We consider a WSN with low-power SNs distributed in a geographic area through air drop. SNs are homogenous with the same initial energy (\( E_0 \)). The deployment area of the WSN is of size \( A^2 \). SNs are distributed according to a homogeneous spatial Poisson process with intensity \( \lambda \). We assume that the domain is relatively free of obstacles and that the WSN is dense enough so that the length of a path connecting two SNs can be approximated by the straight line distance divided by \( r \). The transmission power is kept to a minimum such that one-hop radio range (\( r \)) is used for transmission. Thus, any communication between two nodes with a distance greater than \( r \) between them would require a multihop. The one-hop radio range can be adjusted to maintain connectivity as the network becomes less dense because of node failures at the expense of more energy consumption.

Environment conditions that could cause an SN to fail with a certain probability include hardware failure (\( q \)) and transmission failure due to noise and interference (\( e \)). Moreover, the WSN is vulnerable to sensor captures—that is, SNs may be captured and compromised. Because of random deployment of SNs (e.g., air drop), we assume that all SNs have equal chances of capture, with the capture time characterized by a distribution function \( F_c(t) \) based on historical data and knowledge about the application environment.

The WSN is cluster based, where CHs are elected periodically and form clusters with non-CH nodes. The clustering algorithm ensures that the energy consumption due to the role of CH is distributed fairly evenly among nodes by performing a fair rotation of the CH role among SNs. The clustering algorithm follows closely with that developed in Heinzelman et al. [2002] and Younis and Fahmy [2004]. Specifically, an SN will decide
Fig. 1. A homogeneous clustered WSN with multisource multipath routing and voting-based IDS.

If it should announce itself to be a CH by measuring against a probability parameter, $p$, denoting the probability of being a CH. If yes, it will announce its intention to be a CH. All SNs listen to announcement messages and select one CH candidate that has the lowest cost in terms of distance to minimize CH–SN communication cost. An SN not covered by any CH will double the probability of being a CH (upper bounded by 1) and measure itself against this probability to see if should announce itself as a CH candidate in the next iteration. This iteration process continues until all SNs are covered.

Queries can be issued by a mobile user (while moving) and can be issued anywhere in the WSN through a nearby CH. A CH that takes a query to process is referred to as a query processing center (PC). Each query has a strict timeliness requirement ($T_{req}$). The query must be delivered within $T_{req}$ seconds; otherwise, the query fails.

Multisource multipath routing is achieved through two forms of redundancy: (1) source redundancy by which $m_s$ SNs sensing a physical phenomenon in the same feature zone are used to forward sensing data to their CH, referred to as a source CH, and (2) path redundancy by which $m_p$ paths are used to relay packets from the source CH to the PC.

Figure 1 (bottom middle) shows an example of multisource multipath routing with sensing data being relayed from three source SNs to the source CH ($m_s = 3$) and then from the source CH to the PC over two paths ($m_p = 2$). Each source SN forms a disjoint path to the source CH. In other words, one SN is chosen to relay the sensing data in each hop. Thus, there are a total of $m_s$ paths from $m_s$ source SNs. The source CH in turn creates $m_p$ disjoint paths between the source CH and the PC. The $m_p$ disjoint paths are formed by randomly choosing distinct $m_p$ SNs in the first hop and then randomly
choosing only one SN in each of the subsequent hops. Each query has a unique id, so an intermediate SN selected for packet forwarding can check the query id associated with the packet routing request to ensure that it is only committed once for a query. It has been reported that the number of edge-disjoint paths between nodes is equal to the average node degree with a very high probability [Deb et al. 2003]. Therefore, when the density is sufficiently high such that the average number of one-hop neighbors is sufficiently larger than \( m_p \) and \( m_s \), we can effectively result in \( m_p \) disjoint paths for path redundancy and \( m_s \) disjoint paths from \( m_s \) sensors for source redundancy.

Geographic forwarding is used to route the information between nodes; thus, no path information is maintained. Only the location of the destination SN needs to be known to correctly forward a packet. As part of clustering, a CH knows the locations of SNs within its cluster and vice versa.

We assume that the WSN executes a pairwise key establishment protocol (e.g., Haowen and Perrig [2005]) in a secure interval after deployment. Each node establishes pairwise keys with its \( k \)-hop neighbors, where \( k \) is large enough to cover a cluster area. Thus, upon electing a new CH, the CH will have pairwise keys with the SNs joining its cluster. Since every SN shares a pairwise key with its CH, an SN can encrypt data sent to the CH for confidentiality and authentication purposes.

We assume that SNs operate in power-saving mode (e.g., Bravos and Kanatas [2005] and Qun [2007]). Thus, an SN is either active (transmitting or receiving) or in sleep mode. For the transmission and reception energy consumption of sensors, we adopt the energy model in Younis and Fahmy [2004]. The energy spent by an SN to transmit an encrypted data packet of length \( n_b \) bits over a distance \( r \) is estimated as [Younis and Fahmy 2004]:

\[
E_T = n_b(E_{elec} + E_{amp} r^2).
\] (1)

Here, \( E_{elec} \) is the energy dissipated to run the transmitter and receiver circuitry, \( E_{amp} \) is the energy used by the transmit amplifier, and \( r \) is the transmission radio range. The energy spent by an SN to receive an encrypted message of length \( n_b \) bits is given by

\[
E_R = n_b E_{elec}.
\] (2)

A malicious node aims to break the WSN functionality. We assume that a malicious node can perform the following attacks:

(1) **Packet dropping attacks:** A malicious node can simply drop a packet as it is asked to forward a packet. The effect of a packet dropping attack is that a path is broken. Thus, a path cannot tolerate even a single packet dropping attack.

(2) **Data modification attacks:** A malicious node can perform a data modification attack to corrupt a data packet. If a source SN performs data modification attacks, then the single path originating from that source SN fails. If the source CH performs data modification attacks, then all \( m_p \) paths originating from the source CH fail, and consequently the query fails. Intermediate SNs routing a packet are prevented from performing data modification attacks by source authentication—that is, the packet is encrypted with the pairwise key between the source node (e.g., a source SN) and the destination node (e.g., a source CH) so that the destination node can discern if the data packet actually originates from the source node. If the data packet is detected as modified, then the destination node will simply discard the packet. Thus, a data modification attack performed by an intermediate SN has the same effect as a packet dropping attack.

(3) **Bad-mouthing attacks:** A malicious node can ruin the reputation of well-behaved SNs (by providing bad recommendations against good nodes) to decrease the chance of packets routing through good nodes.
(4) **Ballot stuffing**: A malicious node can boost the reputation of malicious SNs (by providing good recommendations for them) to increase the chance of packets routing through malicious nodes (and being dropped).

To detect and remove malicious nodes from the system, a voting-based system IDS is applied periodically in every $T_{IDS}$ time interval. How often $T_{IDS}$ should be applied is a design issue that we aim to identify in this article. Every node runs a simple rule-based host IDS to detect if its neighbors exhibit suspicious behavior or comply with the prescribed protocol design, including the packet forwarding protocol, the voting-based system IDS protocol, and the clustering algorithm. An example is that when A selects B to forward a packet, A can monitor if B actually forwards the packet or not through overhearing based on the packet broadcast by B. If a neighbor node selected for packet forwarding does not forward as intended, then it is counted as a negative experience. Another example is that when A and B are involved in a voting-based system IDS activity on C, if A is sure that C is good but B insists that C is bad, then A can view it as a negative experience against B. To conserve energy, a node does not promiscuously monitor all neighbors but just uses packets received or overheard during protocol execution according to the rules defined in the host IDS design (e.g., Bhuse and Gupta [2006] and Da Silva et al. [2005]). The flaw of the host IDS is characterized by a false-positive probability ($H_{fp}$) and a false-negative probability ($H_{fn}$), which are assumed known at deployment time. In each interval, $m$ neighbor nodes around a target node will be chosen randomly as voters to decide if the target node is still a good node. Figure 1 (upper left) shows an example of voting-based intrusion detection with $m = 3$ voters. The $m$ voters share their votes through secure transmission using their pairwise keys. How big $m$ should be is another design issue that we aim to identify in this article. When the majority of voters come to the conclusion that a target node is bad, then the target node is evicted. There is a system-level false-positive probability ($P_{fp}$) that the voters can incorrectly identify a good node as a bad node. There is also a system-level false-negative probability ($P_{fn}$) that the voters can incorrectly misidentify a bad node as a good node. These two system-level IDS probabilities will be derived based on the attack model in the work.

To provide a unifying metric that considers the preceding two design trade-offs, we define the total number of queries that the system can answer correctly until it fails as the **lifetime** or the **mean time to failure** (MTTF) of the system that can be translated into the actual system lifetime span based on the query arrival rate.

**4. PERFORMANCE MODEL**

In this section, we develop a probability model to estimate the MTTF of an autonomous WSN using multisource multipath data forwarding to answer queries issued from a mobile user roaming in the WSN area. Table I summarizes the notation used for model parameters for easy reference. There are four design variables: $m_p$, $m_s$, $m$, and $T_{IDS}$. There is only one output variable: MTTF. The basic idea of our MTTF formulation is that we first deduce the maximum number of queries, $N_q$, the system can possibly handle before running into energy exhaustion for the best case in which all queries are processed successfully. Because the system evolves dynamically, the amount of energy spent per query also varies dynamically. Given the query arrival rate $\lambda_q$ as input, we can reasonably estimate the amount of energy spent due to query processing and intrusion detection for query $j$ based on the query arrival time $t_{Q,j}$. Next we derive the corresponding query success probability $R_q(t_{Q,j})$—that is, the probability that the response to query $j$ arriving at time $t_{Q,j}$ is delivered successfully before the query deadline expires. Finally, we compute MTTF as the probability-weighted average of the number of queries that the system can handle without experiencing any deadline,
transmission, or security failure. More specifically, the MTTF is given by

\[
MTTF = \sum_{i=1}^{N_q-1} i \left( \prod_{j=1}^{i} R_q(t_{Q,j}) \right) (1 - R_q(t_{Q,i+1})) + N_q \prod_{j=1}^{N_q} R_q(t_{Q,j}). \tag{3}
\]

Here, \((\prod_{j=1}^{i} R_q(t_{Q,j}))(1 - R_q(t_{Q,i+1}))\) accounts for the probability of the system being able to successfully execute \(i\) consecutive queries but failing the \((i+1)^{th}\) query. The second term is for the best case in which all queries are processed successfully without experiencing any failure for which the system will have the longest lifetime.

ACM Transactions on Sensor Networks, Vol. 11, No. 3, Article 47, Publication date: March 2015.
4.1. Network Dynamics

Initially at deployment, all SNs are good nodes. Assume that the capture time of an SN follows a distribution function $F_c(t)$ that can be determined based on historical data and knowledge about the target application environment. Let $X$ be the capture time. Then, the probability that an SN is compromised at time $t$, given that it was a good node at time $t - T_{IDS}$, denoted by $P_c$, is given by

$$P_C = 1 - P\{X > t, X > t - T_{IDS}\} = 1 - \frac{P\{X > t, X > t - T_{IDS}\}}{P\{X > t - T_{IDS}\}} = \frac{1 - (1 - F_c(t))}{(1 - F_c(t)) - (t - T_{IDS})}.$$  \tag{4}

We note that $P_c$ is time dependent. For the special case in which the capture time is exponentially distributed with rate $\lambda_c$, $P_c = 1 - e^{-\lambda_c \times T_{IDS}}$. Recall that the voting-based system IDS executes periodically, with $T_{IDS}$ being the interval. At the $i_{th}$ IDS execution time (denoted by $t_i$), a good node may have been compromised with probability $P_c$ since the previous IDS execution time ($t_{i-1}$). Let $n_{good}(t)$ and $n_{bad}(t)$ denote the numbers of good and bad neighbor nodes at time $t$, respectively, with $n_{good}(t) + n_{bad}(t) = n(t)$. Then, the population of good and bad neighbor nodes at time $t_i$ just prior to IDS execution can be recursively estimated from the population of good and bad neighbor nodes at time $t_{i-1}$:

$$n_{good}(t_i) = n_{good}(t_{i-1}) - n_{good}(t_{i-1}) \times P_c,$$

$$n_{bad}(t_i) = n_{bad}(t_{i-1}) + n_{good}(t_{i-1}) \times P_c.$$  \tag{5}

With $n_{good}(t)$ and $n_{bad}(t)$ in hand, the system-level false-positive probability ($P_{fp}$) and false-negative probability ($P_{fn}$) as a result of executing voting-based IDS are as follows:

$$P_{fp} or P_{fn} = \sum_{i=0}^{m-n_{maj}} \frac{C(n_{bad} + i, n_{maj} + i) \times C(m - (n_{maj} + i), n_{good})}{C(m - (n_{maj} + i), m)}$$

$$+ \sum_{i=0}^{m-n_{maj}} \frac{C(n_{bad} + n_{good}, m) \times \sum_{j=m_{maj} - 1}^{m-i} C(n_{good} + j, m - i - j) \times (1 - \omega)^{n_{maj} - i - j}}{C(n_{bad} + n_{good}, m)}. \tag{6}$$

where $m_{maj}$ is the minimum majority of $m$; for example, 3 is the minimum majority of 5, and $\omega$ is $H_{pf}$ for calculating $P_{fp}$ and $H_{pn}$ for calculating $P_{fn}$. We explain how the false-positive probability at time $t$ is derived later. The explanation for the false-negative probability is similar. A false positive results when the majority of the voters vote against the target node (which is a good node) as compromised. The first term in Equation (6) accounts for the case in which more than half of the voters selected from the target node’s neighbors are bad sensors who, as a result of performing bad-mouthing attacks, will always vote a good node as a bad node to break the functionality of the WSN. Here, the denominator is the total number of combinations to select $m$ voters out of all neighbor nodes, and the numerator is the total number of combinations to select at least $m_{maj}$ bad voters out of $n_{bad}$ nodes and the remaining good voters out of $n_{good}$ nodes. The second term accounts for the case in which more than half of the voters selected from the neighbors are good nodes, but unfortunately some of these good nodes mistakenly misidentify the target nodes as a bad node with probability $H_{pn}$, resulting in more than half of the voters (some of those may be bad nodes) voting against the target node. Here, the denominator is again the total number of
combinations to select \( m \) voters out of all neighbor nodes, and the numerator is the total number of combinations to select \( i \) bad voters not exceeding the majority \( m_{\text{maj}} \), \( j \) good voters who diagnose incorrectly with \( i + j \geq m_{\text{maj}} \), and the remaining \( m - i - j \) good voters who diagnose correctly.

After the voting-based IDS is executed, some good nodes will be misidentified as bad nodes with probability \( P_{fp} \) and will be mistakenly removed from the WSN. Consequently, we need to adjust the population of good nodes after IDS execution. Let \( n_{\text{good}}(t) \) be the number of good neighbor nodes at time \( t \) right after IDS execution. Then,

\[
n_{\text{good}}(t_{I,i}) = n_{\text{good}}(t_{I,i}) - n_{\text{good}}(t_{I,i}) \times P_{fp}.
\]  

(7)

On the other hand, some bad nodes will remain in the system because the voting-based IDS fails to identify them with probability \( P_{fn} \). Let \( n_{\text{bad}}(t) \) be the number of bad neighbor nodes at time \( t \) right after IDS execution. Then,

\[
n_{\text{bad}}(t_{I,i}) = n_{\text{bad}}(t_{I,i}) - n_{\text{bad}}(t_{I,i}) \times (1 - P_{fn}).
\]  

(8)

As the capture attack is totally random, the probability that any neighbor node is a bad node at time \( t \), denoted by \( Q_{c,j}(t) \), thus is given by

\[
Q_{c,j}(t_{I,i}) = \frac{n_{\text{bad}}(t_{I,i})}{n_{\text{bad}}(t_{I,i}) + n_{\text{good}}(t_{I,i})}.
\]  

(9)

\( Q_{c,j}(t) \) derived earlier provides critical information, as a bad node can perform packet dropping or data modification attacks if it is on a path from source SNs to the PC. Here we note that the node population density is evolving because of some nodes being compromised and some being detected and evicted by the IDS dynamically. The node population remains the same until the next IDS execution (after \( T_{IDS} \) seconds) because the IDS only detects and evicts nodes periodically (as typically node hardware/software failure happens less frequently than security failure). Denote the node population density at time \( t \) by \( \lambda(t) \) with \( \lambda(0) = \lambda \). Then, \( \lambda(t) \) can be computed by

\[
n(t_{I,i}) = n_{\text{bad}}(t_{I,i}) + n_{\text{good}}(t_{I,i})
\]  

(10)

\[
\lambda(t_{I,i}) = \frac{n(t_{I,i})}{\pi r^2}.
\]  

(11)

4.2. Query Success Probability

There are three ways by which data forwarding from SN\(_j\) to SN\(_k\) could fail: (1) transmission speed violation, (2) sensor/channel failures, and (3) SN\(_j\) compromise.

The first source of failure, transmission speed violation, accounts for query deadline violation. To know the failure probability due to transmission speed violation, we first derive the minimum hop-by-hop transmission speed required to satisfy the query deadline \( T_{req} \). Let \( d_{SN-CH} \) be the expected distance between an SN (selected to report sensor readings) and its CH, and let \( d_{CH-PC} \) be the expected distance between the source CH and the PC accepting the query result. Given a query deadline \( T_{req} \) as input, a data packet from an SN through its CH to the PC must reach the PC within \( T_{req} \). Thus, the minimum hop-by-hop transmission speed denoted by \( S_{req} \) is given by

\[
S_{req} = \frac{d_{SN-CH} + d_{CH-PC}}{T_{req}}.
\]  

(12)

Since an SN becomes a CH with probability \( p \) and all sensors are distributed in the area in accordance with a spatial Poisson process with intensity \( \lambda \), CHs and non-CH
SNs will also be distributed in accordance with a spatial Poisson process with rates $p\lambda$ and $(1-p)\lambda$, respectively. Non-CH SNs thus would join the closest CH to form a Voronoi cell [Heinzelman et al. 2002] corresponding to a cluster in the WSN. It can be shown that the average number of non-CH SNs in each Voronoi cell is $(1-p)/p$, and the expected distance from an SN to its CH is given by $d_{SN-CH} = 1/2(1/p\lambda)^{1/2}$. On the other hand, since a query may be issued from anywhere by the mobile user to a CH (which serves as the PC) and the source CH requested by the query also can be anywhere in the WSN, $d_{CH-PC}$ essentially is the average distance between any two CHs in the WSN. For derivation convenience without loss of generality, let the PC be located in the center of the sensor area with the coordinate at $(0, 0)$ and the source CH be randomly located at $(X, Y)$ in the square sensor area with $-A/2 \leq X \leq A/2$ and $-A/2 \leq Y \leq A/2$. Then, $d_{CH-PC}$ is given by

$$d_{CH-PC} = \int_{-A/2}^{A/2} \int_{-A/2}^{A/2} \sqrt{(X^2 + Y^2)} \left( \frac{1}{A} \right)^2 \, dX \, dY = 0.382A. \quad (13)$$

The same final expression for $d_{CH-PC}$ will result if we take the coordinate of the PC to be $(X, Y)$ and put two more integrals, one for $X$ and the other for $Y$, with $-A/2 \leq X \leq A/2$ and $-A/2 \leq Y \leq A/2$, because of symmetric properties. With the knowledge of $d_{SN-CH}$ and $d_{CH-PC}$, we can also estimate the average numbers of hops to forward data from an SN to the source CH, denoted by $N_{SC}$, and the average numbers of hops to forward data from the source CH to the PC, denoted by $N_{CP}$, by $N_{HC} = [d_{SN-CH}]/r$ and $N_{CP} = [d_{CH-PC}]/r$, where $r$ is radio range and $[x]$ is the ceiling of $x$.

Let $Q_{t, jk}$ denote the probability that the forwarding speed from SN$_j$ to SN$_k$ would violate the minimum speed requirement, thus leading to a query deadline violation failure. To calculate $Q_{t, jk}$, we need to know the transmission speed $S_{jk}$ from SN$_j$ to SN$_k$. This can be dynamically measured by SN$_j$ following the approach described in Felemban et al. [2006]. If $S_{jk}$ is above $S_{req}$, then $Q_{t, jk} = 0$; otherwise, $Q_{t, jk} = 1$. In general, $S_{jk}$ is not known until runtime. If $S_{jk}$ is uniformly distributed within a range $[a, b]$, then $Q_{t, jk}$ can be computed as

$$Q_{t, jk} = cdf(S_{jk} \leq S_{req}) = \frac{S_{req} - a}{b - a}. \quad (14)$$

The second source of failure is due to sensor failure or channel failure. Let $Q_{c, j}$ denote the probability of failure due to these types of failure. Since $q$ is the hardware failure probability and $e_j$ is transmission failure probability of node $j$, given as input, $Q_{c, j}$ can be estimated by

$$Q_{c, j} = 1 - (1-q)(1-e_j). \quad (15)$$

The third source of failure is due to node $j$ being compromised, and thus the packet is dropped. We make use of $Q_{c, j}(t)$ derived earlier in Equation (9). By combining these three failure probabilities, we obtain $Q_{rtc, jk}$, the probability of SN$_j$ failing to relay a data packet to a one-hop neighbor SN$_k$ because of either speed violation, sensor/channel failure, or SN$_j$ being compromised, as

$$Q_{rtc, jk} = 1 - [(1 - Q_{t, j})(1 - Q_{c, jk})(1 - Q_{c, j})]. \quad (16)$$

By using this one-hop failure probability, we next compute the success probability for SN$_j$ to transmit a packet to at least one next-hop SN neighbor along the direction...
of the destination node as
\[ \theta_j = 1 - \prod_{h=1}^{f \times n} Q_{\text{rtc}, jh}, \]  
where \( f = 1/4 \) to account for the fact that only neighbor SNs in the quadrant toward the destination node can perform geographic forwarding; \( n \) is the number of neighbor SNs of node \( j \) as given in Equation (10).

Since on average there will be \( N_{hCP}^h \) hops on a path from the source CH to the PC, a data packet transmitted along the path is successfully delivered only if it is delivered successful hop by hop without experiencing any speed violation failure, hardware/channel failure, packet dropping failure, or data modification failure for \( N_{hCP}^h \) hops. Consequently, the probability of a single path between the source CH and the PC being able to deliver data successfully is given by
\[ \Theta(N_{hCP}^h) = \left( \prod_{j=1}^{N_{hCP}^h-1} \theta_j \right) \times (1 - Q_{\text{rtc}, N_{hCP}^h(N_{hCP}^h+1)}). \]  
(18)

For path redundancy, we create \( m_p \) paths between the source CH and the PC. The \( m_p \) paths are formed by randomly choosing \( m_p \) SNs satisfying the minimum speed requirement in the first hop and then randomly choosing only one SN satisfying the minimum speed requirement in each of the subsequent hops. The source CH will fail to deliver data to the PC if one of the following happens: (1) none of the SNs in the first hop receives the message; (2) in the first hop, \( i \) (\( 1 \leq i < m_p \)) SNs receive the message and each of them attempts to form a path for data delivery, yet all \( i \) paths fail to deliver the message because the subsequent hops fail to receive the broadcast message; and (3) in the first hop, at least \( m_p \) SNs receive the message from the source CH from which \( m_p \) SNs are selected to forward data, but all \( m_p \) paths fail to deliver the message because the subsequent hops fail to receive the message. To summarize, the probability of the source CH failing to deliver data to the PC is given by
\[
Q_{fp}^{m_p} = 1 - \theta_1 + \sum_{|I|<m_p} \left\{ \prod_{i \in I} (1 - Q_{\text{rtc}, i}) \right\} \left\{ \prod_{i \not\in I} Q_{\text{rtc}, i} \right\} \left\{ \prod_{i \in I} (1 - \Theta_i(N_{hCP}^h - 1)) \right\} 
+ \sum_{|I|\geq m_p} \left\{ \prod_{i \in I} (1 - Q_{\text{rtc}, i}) \right\} \left\{ \prod_{i \not\in I} Q_{\text{rtc}, i} \right\} \prod_{i \in M, M \subseteq I, |M|=m_p} \left[ 1 - \Theta_i(N_{hCP}^h - 1) \right]. \]  
(19)

Following the same derivation to Equation (18), the success probability of a single path from an SN to its CH is given by
\[ \Theta(N_{hSC}^h) = \left( \prod_{j=1}^{N_{hSC}^h-1} \theta_j \right) \times (1 - Q_{\text{rt}, N_{hSC}^h(N_{hSC}^h+1)}). \]  
(20)
For source redundancy, we use $m_s$ SNs to report query responses to their source CH. The probability that all $m_s$ SNs fail to deliver data to their CH is given by

$$Q_{fs}^m = \prod_{i=1}^{m_s} [1 - \Theta_i (N_{SC}^h)].$$  \hfill (21)

Consequently, the failure probability of data delivery from $m_s$ SNs to the CH, and subsequently using $m_p$ paths to relay data from CH to PC, is given by

$$Q_f = 1 - (1 - Q_{fs}^m)(1 - Q_{fp}^m).$$  \hfill (22)

Therefore, the query success probability is given by

$$R_q = 1 - Q_f.$$  \hfill (23)

Note that in the preceding derivation, we omit time for brevity. More precisely, $R_q$ derived earlier should be $R_q(t_{Q,i})$, as the query success probability is a function of time, depending on the node count (Equation (10)) and population density (Equation (11)) at the $i^{th}$ query’s execution time (i.e., at time $t_{Q,i}$).

### 4.3. Energy Consumption

We estimate the amounts of energy spent during a query interval $[t_{Q,i}, t_{Q,i+1}]$, an IDS interval $[t_{I,i}, t_{I,i+1}]$, and a clustering interval $[t_{C,i}, t_{C,i+1}]$ to estimate $N_q$, the maximum number of queries that the system can possible handle before running into energy exhaustion. Our energy model follows Younis and Fahmy [2004]. Because of the randomness introduced in our protocol in CH election, IDS detection, and query processing, all SNs consume energy at about the same rate. Hence, it suffices to consider the overall system energy level instead of individual SN energy levels for calculating the amount of time that it takes for the system to exhaust energy. To normalize energy consumption over $N_q$ queries, let $\alpha$ be the ratio of the IDS execution rate to the query arrival rate and let $\beta$ be the ratio of the clustering rate to the query arrival rate so that $\alpha N_q$ and $\beta N_q$ are the numbers of IDS cycles and clustering cycles, respectively, before system energy exhaustion. Then, we can estimate $N_q$ by the fact that the total energy consumed due to intrusion detection, clustering, and query processing is equal to the system energy as follows:

$$E_{init} = \sum_{i=1}^{\alpha N_q} E_{IDS}(t_{I,i}) + \sum_{i=1}^{\beta N_q} E_{clustering}(t_{C,i}) + \sum_{i=1}^{N_q} E_q(t_{Q,i}).$$  \hfill (24)

Next, we outline how to obtain $E_{IDS}(t_{I,i})$, $E_{clustering}(t_{C,i})$, and $E_q(t_{Q,i})$. The energy consumed for processing the $i^{th}$ query, $E_q(t_{Q,i})$, is the sum of the energy consumed through $m_p$ paths for the communication between CH and PC, denoted by $E_{CP}(t_{Q,i})$, and the energy consumed for the communication between $m_s$ source SNs and the CH, denoted by $E_{SC}(t_{Q,i})$—that is,

$$E_q(t_{Q,i}) = E_{CP}(t_{Q,i}) + E_{SC}(t_{Q,i}).$$  \hfill (25)

The energy consumed for the communication between CH and PC is due to setting up $m_p$ paths in the first hop and subsequently transmitting data over the $m_p$ paths—that is,

$$E_{CP}(t_{Q,i}) = [E_T + n(t_{Q,i})E_R] + \{m_p(N_{cp}^h - 1)[E_T + n(t_{Q,i})E_R]\}.$$  \hfill (26)

Here, the first term accounts for the transmission energy consumed by the source CH and the reception energy consumed by its one-hop SNs for setting up the $m_p$ paths, and the second term accounts for the energy consumed for data transmission over the
$m_p$ paths in the remaining $N_{SC}^{th} - 1$ hops. We note that the number of neighbor SNs at time $t$, $n(t) = \lambda(t) \times \pi r^2$ by Equation (11), depends on the SN population density at time $t$—that is, $\lambda(t)$. The energy consumed for the communication between source SNs and the CH is due to transmitting data over the $m_n$ paths each with $N_{SC}^{th}$ hops—that is,

$$E_{SC}(t_{Q,i}) = m_nN_{SC}^{th}[E_T + n(t_{Q,i})E_R].$$ (27)

For clustering, the system would consume energy for broadcasting the announcement message and for the cluster-join process. Since $p$ is the probability of an SN becoming a CH, there will be $p \times n(t_{c,i})$ SNs that would be broadcasting the announcement message where $n(t_{c,i}) = \lambda(t_{c,i}) \times A^2$ is the number of SNs in the WSN at time $t_{c,i}$. This announcement message will be received and retransmitted by each SN to the next hop until the TTL of the message reaches the value 0 (i.e., the number of hops is equal to $N_{SC}^{th}$). Thus, the energy required for broadcasting is

$$pn(t_{c,i})[N_{SC}^{th}n(t_{c,i})(E_T + E_R)].$$ (28)

The cluster-join process will require an SN to send a message to the CH informing that it will join the cluster and the CH to send an acknowledgement to the SN. Since there are $p n(t_{c,i})$ CHs and $(1 - p) n(t_{c,i})$ SNs in the system, the energy for this is $n(t_{c,i})(E_T + E_R)$. Let $N_{iteration}$ be the number of iterations required to execute the clustering algorithm. Then, the energy required for executing the clustering algorithm at time $t_{c,i}$, $E_{clustering}(t_{c,i})$, is given by

$$E_{clustering}(t_{c,i}) = pn(t_{c,i})N_{iteration}[N_{SC}^{th}n(t_{c,i})(E_T + E_R)] + n(t_{c,i})(E_T + E_R).$$ (29)

Last, for intrusion detection, every node is evaluated by $m$ voters in an IDS cycle, and each voter sends its vote to the other $m - 1$ voters. Hence, the energy spent in each voting-based IDS cycle is given by

$$E_{IDS}(t_{l,i}) = n(t_{l,i-1})[m(m - 1)](E_T + n(t_{l,i-1})E_R].$$ (30)

Once we obtain $E_{IDS}(t_{l,i})$, $E_{clustering}(t_{c,i})$, and $E_{q}(t_{Q,i})$, we calculate $N_q$ from Equation (24). The knowledge of $N_q$ along with $R_q(t_{Q,i})$ in Equation (23) allows us to calculate the system MTTF given by Equation (3).

4.4. Computational Procedure for Determining Optimal ($m_p$, $m_n$) for Maximizing MTTF

Next we present a high-level description of the computational procedure to determine the optimal redundancy level ($m_p$, $m_n$) for maximizing MTTF. The MTTF equation (Equation (3)) is embedded on lines 14 through 20, 30, and 31. The accumulation of queries is shown on line 12. The value of $N_q$ is computed on line 32. The computational procedure essentially has a complexity of $O(m_p \times m_n)$ as it exhaustively searches for the best ($m_p$, $m_n$) pair, given a set of input parameter values characterizing the operational environment of a query-based WSN, as well as instance values of $m$ (the number of voters for intrusion detection) and $T_{IDS}$ (the intrusion detection interval) characterizing the voting-based IDS as input.

**Input:** Table II input parameters

**Output:** optimal MTTF, optimal($m_p$, $m_n$)

1: \textbf{for} $m_n \leftarrow 1$ to $\max m_n \textbf{do}$
2: \textbf{for} $m_p \leftarrow 1$ to $\max m_p \textbf{do}$
3: \textbf{num}_q \leftarrow 0 \textbf{where num}_q \textbf{is the query counter}$\lambda(t) \times E_w$ at $t = 0$
4: $E_{clustering}, E_{q}, E_{IDS}$ at $t = 0$
5: \textbf{//Processing clustering, query, and IDS events}

ACM Transactions on Sensor Networks, Vol. 11, No. 3, Article 47, Publication date: March 2015.
5. ALGORITHM FOR DYNAMIC REDUNDANCY MANAGEMENT OF INTEGRATED INTRUSION DETECTION AND TOLERANCE

The objective of dynamic redundancy management is to dynamically identify and apply the best redundancy level in terms of path redundancy \((m_p)\) and source redundancy \((m_s)\) for multisource multipath routing, as well as the best intrusion detection settings in terms of the number of voters \((m)\) and the intrusion invocation interval \((T_{IDS})\) to maximize MTTF in response to environmental changes including node density \((\lambda)\), radio range \((r)\), and node capture rate \((\lambda_c)\).

Our algorithm is distributed in nature. The algorithm specifies control actions taken by individual SNs and CHs in response to dynamically changing environments as follows:

1: \textbf{CH Execution:}
2: Get next event
3: if event is \(T_D\) timer then
4: determine radio range to maintain connectivity
5: determine optimal \(T_{IDS}, m, m_s, m_p\) by table lookup based on the current estimated density, CH radio range and compromise rate
6: notify SNs with in the cluster of the new optimal settings of \(T_{IDS}\) and \(m\)
else if event is query arrival then
  trigger multipath routing using \( m_s \) and \( m_p \)
else if event is \( T_{\text{clustering}} \) timer then
  perform clustering
else if event is \( T_{\text{IDS}} \) timer then
  For each neighbor node
    if selected as a voter then
      execute voting – based intrusion detection
else // event is data packet arrival
  follow multipath routing protocol design to route
  the data packet

SN Execution:
Get next event
if event is \( T_D \) timer then
  determine radio range to maintain connectivity
  within a cluster
else if event is control packet arrival from CH then
  change the optimal settings of \( T_{\text{IDS}}, m \)
else if event is \( T_{\text{clustering}} \) timer then
  perform clustering
else if event is \( T_{\text{IDS}} \) timer then
  For each neighbor node
    if selected as a voter then
      execute voting based intrusion detection
else // event is data packet arrival
  follow multipath routing protocol design to route
  the data packet

All nodes in the system act periodically to a “\( T_D \) timer” event to adjust the optimal parameter setting in response to changing environments. This is indicated on line 3 for a CH and on line 20 for an SN. The optimal design settings in terms of optimal \( T_{\text{IDS}}, m, m_s, \) and \( m_p \) are determined at static design time (described in Section 6) and pre-stored in a table over perceivable ranges of input parameter values. As there is no base station in the system, the duty of performing a table lookup operation with interpolation and/or extrapolation techniques applied to determine the optimal design parameter settings will be assumed by CHs. The action performed by a CH upon a \( T_D \) timer event includes (1) adjusting radio range to maintain connectivity (line 4); (2) determining \( T_{\text{IDS}}, m, m_s, \) and \( m_p \) (line 5) based on the sensed environmental conditions at runtime; and (3) notifying SNs within the cluster of the new \( T_{\text{IDS}} \) and \( m \) settings. The action performed by an SN upon this \( T_D \) timer event is to adjust its radio range to maintain connectivity within a cluster (line 21). The action taken upon receiving the control packet from its CH is to update the new \( T_{\text{IDS}} \) and \( m \) settings (line 23) for intrusion detection. When the \( T_{\text{IDS}} \) timer event happens, each node in the system uses its current \( T_{\text{IDS}} \) and \( m \) settings to perform intrusion detection. The \( T_{\text{IDS}} \) timer event and the action taken are specified on lines 11 through 14 for a CH and on lines 26 through 29 for an SN.

When a CH acting as a query PC receives a query from a user, it triggers multisource multipath routing for intrusion tolerance using the current optimal \( m_s \) and \( m_p \) settings to prolong the system useful lifetime. This query arrival event and the action taken are specified on lines 7 and 8. When a data packet arrival event occurs, each node simply follows the prescribed multipath routing protocol to route the packet (lines 15 and
Table II. Input Parameter Values Characterizing a Homogeneous Clustered WSN

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>1,500 nodes</td>
</tr>
<tr>
<td>$p$</td>
<td>0.01</td>
</tr>
<tr>
<td>$q$</td>
<td>$10^{-6}$</td>
</tr>
<tr>
<td>$\epsilon_j$</td>
<td>[0.0001–0.1]</td>
</tr>
<tr>
<td>$r$</td>
<td>40m</td>
</tr>
<tr>
<td>$f$</td>
<td>0.25</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>15 nodes/(40 \times 40 m^2)</td>
</tr>
<tr>
<td>$\lambda_q$</td>
<td>1 query/sec</td>
</tr>
<tr>
<td>$\lambda_c$</td>
<td>Once per 4 days to 28 days</td>
</tr>
<tr>
<td>$A \times A$</td>
<td>400 \times 400 m</td>
</tr>
<tr>
<td>$n_b$</td>
<td>50 bits</td>
</tr>
<tr>
<td>$E_{elec}$</td>
<td>50 nJ/bit</td>
</tr>
<tr>
<td>$E_{amp}$</td>
<td>10 pJ/bit/m^2</td>
</tr>
<tr>
<td>$E_o$</td>
<td>10 Joule</td>
</tr>
<tr>
<td>$N_{iteration}$</td>
<td>3 iterations</td>
</tr>
<tr>
<td>$T_{clustering}$</td>
<td>60 sec</td>
</tr>
<tr>
<td>$T_{req}$</td>
<td>[0.3–1.0] sec</td>
</tr>
<tr>
<td>$H_{fpf}$, $H_{fn}$</td>
<td>[0.01–0.05]</td>
</tr>
</tbody>
</table>

16 for a CH and lines 30 and 31 for an SN). Finally, each node periodically performs clustering as prescribed by the cluster algorithm—in other words, when a $T_{clustering}$ timer event occurs, each node executes clustering (lines 9 and 10 for a CH and lines 24 and 25 for an SN) for periodic cluster formation.

6. PERFORMANCE EVALUATION

In this section, we present numerical results obtained from the evaluation of our probability model given in Section 4. We use AFTQC [Chen et al. 2011], a multisource multipath routing algorithm for fault/intrusion tolerance, as a baseline case for performance comparison to demonstrate the effectiveness of our dynamic resource management algorithm.

For fair comparison, we set up a query-based clustered WSN as in Chen et al. [2011], characterized by a set of input parameter values as listed in Table II. The WSN comprises 1,500 SNs deployed in a square area of $A^2$ (400 \times 400 m). Nodes are distributed in the area following a Poisson process with density $\lambda = 15$ nodes/(40 \times 40 m^2) at deployment time. The radio range $r$ is 40m. Thus, initially, an SN has $n = \lambda \times \pi r^2 = 15$ neighbor SNs. The probability of an SN becoming a CH is $p = 1\%$, as in Chen et al. [2011]. This means that every SN has the same chance (1\%) to serve as a CH so that all nodes rotate fairly for the role of CH and therefore consume energy at about the same rate. This leads to the result that there are about $1/p = 100$ nodes per cluster, and there are about 15 clusters in the system [Chen et al. 2011]. Each SN has an initial energy level $E_o = 10$ Joules. The energy parameters used by the radio module are adopted from Heinzelman et al. [2002] and Younis and Fahmy [2004]. The energy dissipation $E_{elec}$ to run the transmitter and receiver circuitry is 50 nJ/bit. The energy used by the transmit amplifier to achieve an acceptable signal to noise ratio ($\epsilon_{amp}$) is 10 pJ/bit/m^2. The query arrival rate $\lambda_q$ is a variable and is set to 1 query/sec to reveal points of interest. The query deadline $T_{req}$ is strict and set to between 0.3 and 1 sec.

On top of the WSN environment described earlier, we consider parameters related to node compromising events and intrusion detection events. Specifically, the capture rate ($\lambda_c$) is in the range of once per 4 days to once per 28 days, and the host IDS false-positive
probability and false-negative probability ($H_{pfp}$ and $H_{pfn}$) vary between 1% and 5% to reflect the host intrusion detection accuracy as in Da Silva et al. [2005].

6.1. Identifying Protocol Setting for Optimal Redundancy Management

We utilize the performance model developed to help us identify optimal redundancy management for integrated intrusion tolerance and intrusion detection proposed in the article for maximizing the WSN lifetime. The best parameter setting is defined in terms of $(m_p, m_s)$ for multipath routing, as well as $m$ (the number of voters) and $T_{IDS}$ (the intrusion detection interval) for intrusion detection.

We run the computational procedure described in Section 4 to analyze the effect of $\lambda_c$, $m$, and $T_{IDS}$ on optimal $(m_p, m_s)$. Figures 2 and 3 show MTTF (in seconds) versus $(m_p, m_s)$ under low and high $\lambda_c$ values, respectively. By comparing these two graphs,
we observe a trend that as the capture rate increases (i.e., going from Figure 2 to Figure 3), the optimal \((m_p, m_s)\) redundancy level increases. For instance, when the capture rate increases from once in 3 weeks to once a week, the optimal \((m_p, m_s)\) redundancy level changes from \((3, 3)\) to \((4, 4)\). The reason behind this trend is that as more nodes are compromised in the system, a higher multisource multipath redundancy must be used to cope with packet dropping or data modification attacks. Although increasing \((m_p, m_s)\) consumes more energy, the gain toward increasing the query success probability (and thus toward increasing MTTF) outweighs the loss of lifetime due to energy consumption. It is worth noting that although the trend is intuitive, we identify the exact optimal \((m_p, m_s)\) redundancy level to maximize the WSN lifetime.

Table III summarizes the effect of \(\lambda_c\) and \(m\) on optimal \((m_p, m_s)\) values under which MTTF is maximized. As the number of voters in intrusion detection \((m)\) increases, the optimal \((m_p, m_s)\) redundancy level decreases. This is because increasing \(m\) has the effect of detecting and evicting bad nodes more effectively, thus requiring a lower level of redundancy in \((m_p, m_s)\) to cope with packet dropping or data modification attacks by bad nodes.

On the other hand, for a given \(\lambda_c\), there exists an optimal \(m\) value that will maximize MTTF. Table IV summarizes the effect of \(\lambda_c\) on the optimal \(m\) value at which MTTF is maximized. We first observe that there exists an optimal \(m\) value goes from 3 to 7. The reason is that as the capture rate increases, there are more and more malicious nodes in the system, so using more voters (e.g., \(m = 7\)) can help identify and evict malicious nodes, thus increasing the query success probability and consequently increasing the MTTF value. Again, the system is better off this way to cope with increasing malicious node population for lifetime maximization even though more energy is consumed due to more voters being used.

Next we analyze the effect of \(T_{IDS}\) on MTTF. Figures 4 and 5 show MTTF versus \(T_{IDS}\) with varying \(m\) under low capture rate (once per 3 weeks) and high capture rate (once per week), respectively. We first observe that there exists an optimal \(T_{IDS}\) value under which MTTF is maximized. Furthermore, the optimal \(T_{IDS}\) value increases as \(m\) increases. For example, in Figure 4, as \(m\) increases from 3, 5, to 7, we see that correspondingly the optimal \(T_{IDS}\) at which MTTF is maximized increases from 15, 32, to 46 hours. The reason is that as the number of voters increases, the intrusion detection capability increases per invocation; there is no need to invoke intrusion detection too often, which would waste energy and adversely shorten the system lifetime. We also observe two general trends. One trend is that as \(T_{IDS}\) increases, the optimal \(m\) value increases. The reason is that when \(T_{IDS}\) is small, intrusion detection is invoked frequently; we do not need many voters per invocation, which would waste energy
unnecessarily and adversely shorten the system lifetime. The second trend shown in Figures 4 and 5 is that as the node capture rate increases, the optimal $m$ value increases to cope with more compromised nodes in the system. These two trends correlate well those summarized earlier in Table IV.

Last, $T_{IDS}$ is also a tunable parameter to maximize MTTF. Figure 6 shows MTTF versus $T_{IDS}$ with varying $\lambda_c$ values. It exhibits the trend that as the capture rate increases, the optimal $T_{IDS}$ at which MTTF is maximized must decrease to cope with malicious attacks. For example, in Figure 6, the optimal $T_{IDS}$ is 20 hours when $\lambda_c$ is once per 4 weeks and reduces to 6 hours when $\lambda_c$ is once per 4 days. Furthermore, the optimal $T_{IDS}$ value increases as $m$ increases. The reason is that as the number of voters ($m$) increases, the intrusion detection capability increases per invocation; there is no need to invoke intrusion detection too often, which would waste energy and adversely shorten the system lifetime.

Table V summarizes the effect of $\lambda_c$ and $m$ on the optimal $T_{IDS}$ value at which MTTF is maximized.

6.2. Runtime Dynamic Redundancy Management

It should be noted that Tables III, IV, and V are numerical solutions generated from evaluating the analytical equations derived in Section 4, given node density $\lambda$, radio range $r$, and node capture rate $\lambda_c$ as input. As the system evolves, all of these input
parameter values may change—that is, $\lambda$ will decrease as described by Equation (11), radio range $r$ will increase to maintain connectivity as more nodes fail or are evicted from the system, and $\lambda_c$ may evolve depending on the instantaneous attacker strength. Lookup tables such as Tables III, IV, and V are built at static time, covering a wide range of $(\lambda, r, \lambda_c)$ values as input. Our dynamic multisource multipath routing algorithm described in Section 5 then utilizes these lookup tables built at static time to perform a simple lookup operation to decide the optimal settings of $(m_p, m_s, m, T_{IDS})$ to maximize the WSN lifetime at runtime.

6.3. Comparative Performance Analysis

We perform a comparative analysis of our dynamic redundancy management algorithm against AFTQC [Chen et al. 2011]. AFTQC is capable of dynamically identifying and applying the best $(m_p, m_s)$ setting for query processing to maximize the WSN lifetime. However, it is designed for fault/intrusion tolerance only with no consideration given to intrusion detection of compromised nodes. For fair comparison, we tailor the probability model developed in Section 3 without IDS for AFTQC to identify the optimal $(m_p, m_s)$.

In Figure 7, we show MTTF versus $\lambda_c$ under our dynamic redundancy management algorithm against AFTQC, both operating at the optimal setting. The optimal setting in terms of the optimal $(m_p, m_s)$ for each data point is labeled in the graph.

Analyzing the optimal $(m_p, m_s)$ settings reveals insight as to why our algorithm performs better than AFTQC. In other words, the optimal $(m_p, m_s)$ value under our algorithm is consistently lower than that under AFTQC given the same capture rate. The intrusion detection activity in our algorithm consumes extra energy. However,
the energy consumption is minimized, as it operates under the identified best setting in terms of the detection interval ($T_{IDS}$) and the number of voters ($m$). Moreover, because bad nodes are removed effectively, the system does not need to use excessive redundancy in terms of $(m_p, m_s)$ to cope with bad nodes performing packet dropping or data modification attacks. Consequently, the MTTF obtainable by our algorithm is substantially higher than that of AFTQC.

7. CONCLUSION

The issue of energy-aware redundancy management in WSNs for satisfying QoS requirements without excessive energy consumption requires a holistic approach of exploiting the trade-off between intrusion detection/tolerance strength and energy consumption. In this article, we provided a solution to the issue of dynamic redundancy management of multisource multipath routing for intrusion/fault tolerance, integrated with majority voting for intrusion detection, for lifetime maximization of clustered WSNs. We developed a novel probability model to analyze the best multisource multipath redundancy level in terms of path redundancy ($m_p$) and source redundancy ($m_s$), as well as the best intrusion detection settings in terms of the number of voters ($m$) and the intrusion invocation interval ($T_{IDS}$) under which the lifetime of a query-based WSN may be maximized in the presence of unreliable wireless communication and malicious nodes. Our dynamic multisource multipath routing algorithm utilizes the analysis result to determine the optimal system settings for redundancy and intrusion detection based on the sensed environmental conditions at runtime, thus resulting in the system achieving its maximum lifetime. We demonstrated its effectiveness against a multisource multipath routing algorithm called AFTQC that considers only fault/intrusion tolerance.

There are several future research directions that can be extended from this work. First, we can generalize the analysis methodology to handle both homogeneous and heterogeneous WSNs [Al-Hamadi 2013]. We plan to consider more sophisticated attacker models [Mitchell and Chen 2013a, 2013b], such as an adversary that can perform more targeted attacks, capture certain strategic nodes with higher probability, alternate between benign and malicious behavior, and collude with other attackers to avoid intrusion detection. This work considered a strict system failure criterion—that is, if a query fails, the system fails. In the future, we plan to investigate the effect of fuzzy failure criteria [Bastani et al. 1994] under which the system fails after it experiences
a number of query failures. This article addresses the best redundancy level for multi-source multipath routing—that is, how many sources and how many paths one should use for multipath routing to maximize the system lifetime. In the future, we plan to explore trust management [Bao et al. 2011, 2012; Chen et al. 2010; Cho et al. 2009, 2012] to address the issue of what paths one should use to avoid untrustworthy, malicious nodes to further enhance WSN survivability. In situations where concurrent query traffic is heavy, we plan to explore trust-based admission control [Chen and Hsi 1998; Cheng et al. 2000, 2003; Yilmaz and Chen 2009] utilizing Petri net modeling techniques [Chen and Wang 1996a, 1996b; Li and Chen 2011; Trivedi 1999] to optimize application performance.

REFERENCES


Received June 2013; revised November 2014; accepted December 2014