Integrated Intrusion Detection and Tolerance in Homogeneous Clustered Sensor Networks

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

In this paper 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

1. INTRODUCTION

Advances in wireless sensor networks (WSNs) lead to its wide deployment across many fields. Many WSN applications have quality of service (QoS) requirements in security, reliability and timeliness. Also 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 [7,34]. 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 [36]. 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. While most prior research focused on using multipath routing for fault

This material is supported in part by the U. S. Army Research Office under contract number W911NF-12-1-0445.

Author's addresses: Hamid Al-Hamadi, Department of Computer Science, Kuwait University, University in Kuwait City, Kuwait, Email: hamid@cs.ku.edu.kw; Ing-Ray Chen, Department of Computer Science, Virginia Tech, Falls Church, VA 22043, Email: irchen@cs.vt.edu.

DOI:http://dx.doi.org/10.1145/0000000.0000000

tolerance to improve reliability [11,22,39], some attention has been paid to using multipath routing to tolerate insider attacks [21,26,29]. Most studies, however, largely ignored the tradeoff between QoS gain vs. energy consumption which can adversely shorten the system lifetime.

The tradeoff issue between energy consumption vs. 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 [6,16,19,27] to cope with collusion is proven to be an 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 paper, 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, while 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 tradeoff between energy consumption vs. 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 paper is that our dynamic redundancy management protocol design addresses the issues 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 time.

2. RELATED WORK

Over the past few years, numerous protocols have been proposed to detect intrusion in WSNs [6]. In [19], a decentralized rule-based intrusion detection system 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 a SN does not forward a packet within a time limit, if a 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 a 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_{pfp}) and a false negative probability (H_{pfn}). In [19], 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 [27], 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 [16] with considerations given to the

tradeoff between energy loss vs. 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, In [30], the authors proposed a self-learning system used by SNs to detect malicious packets. A 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 vs. energy consumption to maximize the system lifetime.

In [8], the authors proposed a hierarchal model for WSNs with cluster heads responsible for reporting intrusion related data to the BS, thus avoiding expensive transmission from SNs to the BS. This approach however drains energy of CHs faster than SNs, thus may adversely shorten the lifetime span of CHs which are critical to the WSN. eHIP was proposed in [37] 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 [25], 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. [2,3] 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 above, our paper takes a more holistic approach by considering energy consumption vs. 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 wireless sensor networks [36]. In [26] 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 [21] 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 [33] 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 [35], 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 of at least k out of n shares can reach the receiver. The approach, however, 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 tradeoff energy consumption vs. security and reliability gain for WSN lifetime maximization.

In the area of redundancy management of WSNs, in [1,11] the issue of how many paths should be used was addressed in the context of multipath routing from a source node to a sink node. In particular in the context of homogeneous clustered WSNs, AFTQC [11] 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 above, 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 paper we perform a comparative performance analysis of our approach against AFTQC [11].

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_o). The deployment area of the WSN is of size A^2. SNs are distributed according to a homogeneous spatial Poisson process with intensity λ. We assume the domain is relatively free of obstacles and 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 multi-hop. 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 which could cause a 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, i.e., SNs may be captured and compromised. Because of random deployment of SNs (e.g., air drop), we assume 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 [24,41]. Specifically, a SN will decide 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 which has the lowest cost in terms of distance to minimize CH-SN communication cost. A 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.

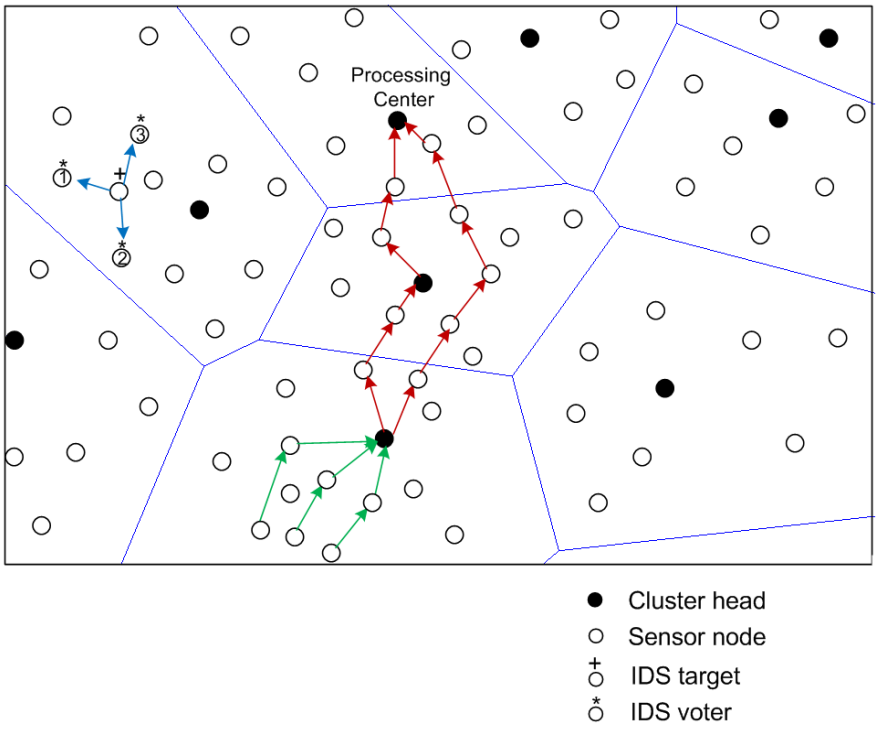

Fig. 1: A Homogeneous Clustered WSN with Multisource Multipath Routing and Voting-based IDS.

Queries can be issued by a mobile user (while moving) and can be issued anywhere in the WSN through a nearby CH. A CH which takes a query to process is called 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: (a) 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; (b) path redundancy by which m_p paths are used to relay packets from the source CH to the PC.

Fig. 1 (bottom middle) shows an example of multisource multipath routing with sensing data being relayed from 3 source SNs to the source CH (m_s=3) and then from the source CH to the PC over 2 paths (m_p=2). Each source SN forms a disjoint path to the source CH. That is, 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 [20]. 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., [23]) 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, a SN can encrypt data sent to the CH for confidentiality and authentication purposes.

We assume that SNs operate in power saving mode (e.g. [7,34]). Thus, a 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 [41]. The energy spent by a SN to transmit an encrypted data packet of length n_b bits over a distance r is estimated as [41]:

$$E_T = n_b(E_{elec} + E_{amp} r^2) \quad (1)$$

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 a SN to receive an encrypted message of length n_b bits is given by:

$$E_R = n_b E_{elec} \quad (2)$$

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 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, i.e., the packet is encrypted with the pair-wise key between the source node (e.g., a source SN) and the destination node (e.g., a source CH) so the destination node can discern if the data packet actually originates from the source node. If the data packet is detected modified, then the destination node will simply discard the packet. Thus, a data modification attack performed by an intermediate SN has the same effect of 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) so as 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) so as 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 should T_{IDS} be is a design issue which we aim to identify in this paper. 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 C is good but B insists on C being 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., [5,19]). The flaw of the host IDS is characterized by a false positive probability (H_{pfp}) and a false negative probability (H_{pfn}), 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.

Fig. 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 should m be is another design issue which we aim to identify in this paper. 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 paper.

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

TABLE I: NOTATION OF SYMBOLS.

Symbol	Meaning	Type
$A \times A$	WSN area (meter^2)	input
n_b	Size of a data packet (bit)	input
E_{elec}	Energy dissipation to run the transmitter and receiver circuitry (J/bit)	input
E_{amp}	Energy used by the transmit amplifier to achieve an acceptable signal to noise ratio (J/bit/m^2)	input
E_o	Initial energy per node (Joule)	input
E_{init}	Initial energy of the WSN (Joule)	input
$E_{clustering}(t)$	Energy consumed for executing the clustering algorithm at time t (Joule)	derived
$E_{IDS}(t)$	Energy consumed for executing the IDS algorithm at time t (Joule)	derived
$E_q(t)$	Energy consumed for executing a query at time t (Joule)	derived
$R_q(t)$	Probability that a query reply at time t is delivered successfully by the deadline	derived
r	Wireless radio communication range (meter)	input

q	node hardware failure probability	input
e_j	Transmission failure probability of node j	input
N	Number of SNs	input
$n(t)$	Number of neighbor nodes at time t	derived
$n_{good}(t)$	Number of good neighbor nodes at time t	derived
$n_{bad}(t)$	Number of bad neighbor nodes at time t	derived
N_q	Maximum number of queries before energy exhaustion	derived
f	Fraction of neighbor nodes that will forward data	input
λ_q	Query arrival rate (times/sec)	input
λ_c	Node capture rate	input
λ	Node population density at deployment time	input
$\lambda(t)$	Node population density (nodes/meter^2) at time t	derived
S_{jk}	Progressive transmission speed between node j and node k (meter/sec)	derived
$T_{clustering}$	Time interval for executing the clustering algorithm (sec)	input
T_{req}	Query deadline requirement (sec)	input
α	Ratio of IDS execution rate to query arrival rate	input
β	Ratio of clustering rate to query arrival rate	input
H_{pfp}	Probability of host IDS false positive	input
H_{pfn}	Probability of host IDS false negative	input
Pfp	Probability of system IDS false positive	derived
Pfn	Probability of system IDS false negative	derived
m	Number of voters selected for executing system IDS	design
T_{IDS}	IDS interval time (sec)	design
m_p	Path redundancy level: Number of paths from a source CH to the sink	design
m_s	Source redundancy level: Number of SNs per cluster in response to a query	design
MTTF	Lifetime of a WSN	output

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 4 design variables, namely, m_p, m_s, m, and T_{IDS}. There is only one output variable, namely, 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 λ_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 the system can handle without experiencing any deadline, transmission, or security failure. More specifically, the MTTF is given by:

$$MTTF = \sum_{i=1}^{N_q-1} i\left(\prod_{j=1}^{i} R_q(t_{Q,j})\right)\left(1 - R_q(t_{Q,i+1})\right) + N_q \prod_{j=1}^{N_q} R_q(t_{Q,j}) \qquad (3)$$

Here $\left(\prod_{j=1}^{i} R_q(t_{Q,j})\right)\left(1 - R_q(t_{Q,i+1})\right)$ accounts for the probability of the system being able to successfully execute i consecutive queries but failing the i+1th 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 span.

4.1 Network Dynamics

Initially at deployment all SNs are good nodes. Assume that the capture time of a SN follows a distribution function $F_c(t)$ which can be determined based on historical data and knowledge about the target application environment. Let X be the capture time. Then, the probability that a 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:

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

We note that P_c is time dependent. For the special case in which the capture time is exponential distributed with rate λ_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 ith IDS execution time (denoted by $t_{I,i}$), a good node may have been compromised with probability P_c since the previous IDS execution time $(t_{I,i-1})$. 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 $t_{I,i}$ just prior to IDS execution can be recursively estimated from the population of good and bad neighbor nodes at time $t_{I,i-1}$:

$$\begin{aligned} n_{good}(t_{I,i}) &= n_{good}(t_{I,i-1}) - n_{good}(t_{I,i-1}) \times P_c \\ n_{bad}(t_{I,i}) &= n_{bad}(t_{I,i-1}) + n_{good}(t_{I,i-1}) \times P_c \end{aligned} \quad (5)$$

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 resulting of executing voting-based IDS are as follows:

$$\begin{aligned} P_{fp}\ or\ P_{fn} &= \sum_{i=0}^{m-m_{maj}} \left[\frac{C\left(\begin{matrix} n_{bad} \\ m_{maj}+i \end{matrix}\right) \times C\left(\begin{matrix} n_{good} \\ m-(m_{maj}+i) \end{matrix}\right)}{C\left(\begin{matrix} n_{bad}+n_{good} \\ m \end{matrix}\right)} \right] \\ &+ \sum_{i=0}^{m-m_{maj}} \left[\frac{C\left(\begin{matrix} n_{bad} \\ i \end{matrix}\right) \times \sum_{j=m_{maj}-i}^{m-i} \left[C\left(\begin{matrix} n_{good} \\ j \end{matrix}\right) \times \omega^j \times C\left(\begin{matrix} n_{good-j} \\ m-i-j \end{matrix}\right) \times (1-\omega)^{m-i-j} \right]}{C\left(\begin{matrix} n_{bad}+n_{good} \\ m \end{matrix}\right)} \right] \end{aligned} \quad (6)$$

where m_{maj} is the minimum majority of m, e.g., 3 is the minimum majority of 5, and ω is H_{pfp} for calculating P_{fp} and H_{pfn} for calculating P_{fn}. We explain how the false positive probability at time t is derived below. The explanation to 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 1/2 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 1/2 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_{pfp}, resulting in more than 1/2 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_{maj}, j good voters who diagnose incorrectly with $i + j \geq m_{maj}$, 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 $\overline{n_{good}(t)}$ be the number of good neighbor nodes at time t right after IDS execution. Then,

$$\overline{n_{good}(t_{I,\iota})} = n_{good}(t_{I,i}) - n_{good}(t_{I,i}) \times P_{fp} \quad (7)$$

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 $\overline{n_{bad}(t)}$ be the number of bad neighbor nodes at time t right after IDS execution. Then,

$$\overline{n_{bad}(t_{I,\iota})} = n_{bad}(t_{I,i}) - n_{bad}(t_{I,i}) \times (1 - P_{fn}) \quad (8)$$

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:

$$Q_{c,j}(t_{I,i}) = \frac{\overline{n_{bad}(t_{I,\iota})}}{\overline{n_{bad}(t_{I,\iota})} + \overline{n_{good}(t_{I,\iota})}} \quad (9)$$

$Q_{c,j}(t)$ derived above 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:

$$n(t_{I,i}) = \overline{n_{bad}(t_{I,\iota})} + \overline{n_{good}(t_{I,\iota})} \quad (10)$$

$$\lambda(t_{I,i}) = \frac{n(t_{I,i})}{\pi r^2} \quad (11)$$

4.2 Query Success Probability

There are three ways by which data forwarding from SN_j to SN_k could fail: (a) transmission speed violation; (b) sensor/channel failures; and (c) SN_j is compromised.

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 a SN (selected to report sensor readings) and its CH and 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 a 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:

$$S_{req} = \frac{d_{SN-CH} + d_{CH-PC}}{T_{req}} \tag{12}$$

Since a SN becomes a CH with probability p and all the sensors are distributed in the area in accordance with a spatial Poisson process with intensity λ, CHs and non-CH SNs will also be distributed in accordance with a spatial Poisson process with rates $p\lambda$ and $(1\text{-}p)\lambda$, respectively. Non-CH SNs thus would join the closest CH to form a Voronoi cell [24] corresponding to a cluster in the WSN. It can be shown that the average number of non-CH SNs in each Voronoi cell is $(1\text{-}p)/p$ and the expected distance from a SN to its CH is given by $d_{SN-CH} = 1/2(p\lambda)^{1/2}$. 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_i, Y_i) in the square sensor area with $-A/2 \le X_i \le A/2$ and $-A/2 \le Y_i \le A/2$. Then, d_{CH-PC} is given by:

$$d_{CH-PC} = \int_{-A/2}^{A/2}\int_{-A/2}^{A/2}\sqrt{(X_i^2 + Y_i^2)}\left(\frac{1}{A}\right)\left(\frac{1}{A}\right)dX_i dY_i = 0.382A \tag{13}$$

The same final expression for d_{CH-PC} will result if we take the coordinate of the processing center to be (X_c, Y_c) and put two more integrals, one for X_c and the other for Y_c with $-A/2 \le X_c \le A/2$ and $-A/2 \le Y_c \le A/2$, 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 a SN to the source CH, denoted by N_{SC}^h, and the average numbers of hops to forward data from the source CH to the PC, denoted by N_{CP}^h, by $N_{SC}^h = \lceil d_{SN-CH} \rceil / r$ and $N_{cp}^h = \lceil d_{CH-PC} \rceil / r$ where r is radio range and $\lceil x \rceil$ 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 [22]. 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:

$$Q_{t,jk} = cdf(S_{jk} \le S_{req}) = \frac{S_{req} - a}{b - a} \tag{14}$$

The second source of failure is due to sensor failure or channel failure. Let

$Q_{r,j}$ denote the probability of failure due to sensor failure or channel failure. Since q is the hardware failure probability and e_j is transmission failure probability of node j, given as input, $Q_{r,j}$ can be estimated by:

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

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:

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

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:

$$\theta_j = 1 - \prod_{k=1}^{f \times n} Q_{rtc,jk} \quad (17)$$

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_{CP}^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_{CP}^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:

$$\Theta(N_{CP}^h) = \left(\prod_{j=1}^{N_{CP}^h - 1} \theta_j\right) \times (1 - Q_{rtc,\ N_{CP}^h (N_{CP}^h + 1)}) \quad (18)$$

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: (a) none of the SNs in the first hop receives the message; (b) 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; however, all i paths fail to deliver the message because the subsequent hops fail to receive the broadcast message; (c) 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. Summarizing above, the probability of the source CH failing to deliver data to the PC is given by:

$$Q_{fp}^{m_p} = 1-\theta_1 + \sum_{|I|<m_p} \{[\prod_{i\in I}(1-Q_{rtc,1i})](\prod_{i\notin I} Q_{rtc,1i})\}\{\prod_{i\in I}[1-\Theta_i(N_{CP}^h-1)]\} + \sum_{|I|\geq m_p} \{[\prod_{i\in I}(1-Q_{rtc,1i})](\prod_{i\notin I} Q_{rtc,1i})\}\{\prod_{\substack{i\in M,\\ M\subseteq I,\\ |M|=m_p}}[1-\Theta_i(N_{CP}^h-1)]\} \quad (19)$$

Following the same derivation to Equation 18, the success probability of a single path from a SN to its CH is given by:

$$\Theta(N_{SC}^h) = (\prod_{j=1}^{N_{SC}^h-1}\theta_j)\times(1-Q_{rt,\,N_{SC}^h(N_{SC}^h+1)}) \quad (20)$$

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:

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

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:

$$Q_f = 1-(1-Q_{fp}^{m_p})(1-Q_{fs}^{m_s}) \quad (22)$$

Therefore, the query success probability is given by:

$$R_q = 1-Q_f \quad (23)$$

Note that in the above derivation we omit time for brevity. More precisely, R_q derived above should be $R_q(t_{Q,i})$ since 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 $[t_{c,i}, t_{c,i+1}]$, so as to estimate N_q, the maximum number of queries the system can possible handle before running into energy exhaustion. Our energy model follows [41]. 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 it takes for the system to exhaust energy. To normalize energy consumption over N_q queries, let α be the ratio of the IDS execution rate to the query arrival rate and let β be the ratio of the clustering rate to the query arrival rate so that αN_q and βN_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:

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

Below 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 $E_{SC}(t_{Q,i})$, i.e.,

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

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, i.e.,

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

Here the first term accounts for the transmission energy consumed by the source CH and the reception energy consumed by its 1-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 $N_{CP}^h - 1$ 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, i.e., $\lambda(t)$. The energy consumed for the communication between source SNs and the CH is due to transmitting data over the m_s paths each with N_{SC}^h hops, i.e.,

$$E_{SC}(t_{Q,i}) = m_s N_{SC}^h [E_T + n(t_{Q,i})E_R] \quad (27)$$

For clustering, the system would consume energy for broadcasting the announcement message and for the cluster-join process. Since p is the probability of a 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}^h. Thus, the energy required for broadcasting is:

$$pn(t_{C,i})[N_{SC}^h\, n(t_{C,i})(E_T + E_R)] \quad (28)$$

The cluster-join process will require a 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 iteration 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:

$$E_{clustering}(t_{C,i}) = pn(t_{C,i})N_{iteration}\left[N_{SC}^h\, n(t_{C,i})(E_T + E_R)\right] + n(t_{C,i})(E_T + E_R) \quad (29)$$

Lastly, 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:

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

Once we obtain $E_{IDS}(t_{I,i})$, $E_{clustering}(t_{C,i})$ 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_s) for Maximizing MTTF

Below we present a high level description of the computational procedure to determine the optimal redundancy level (m_p, m_s) for maximizing MTTF. The MTTF Equation (Equation 3) is embedded on lines 14-20 and 30-31. 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($m_p \times m_s$) as it exhaustively searches for the best (m_p, m_s) 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_s)

```
1:  for m_s ← 1 to max_m_s do
2:    for m_p ← 1 to max_m_p do
3:      num_q ← 0      where num_q is the query counter
4:      E_init ← N(t) × E_o at t = 0
5:      Compute λ, R_q, E_clustering, E_q, E_IDS  at  t = 0
6:        //Processing clustering, query, and IDS events
7:      while E_init > E_threshold do
8:            ev ← next event
9:          if ev is clustering event then
10:           E_init ← E_init − E_clustering
11:         else if ev is query event then
12:           num_q ← num_q + 1
13:           E_init ← E_init − E_q
14:           if num_q = 1 then //first query
15:             rq_muls ← rq_muls × R_q
16:             temp ← num_q × rq_muls
17:           else //accumulate MTTF
18:             temp_MTTF ← temp_MTTF + temp ×
                                    (1 − R_q)
19:             rq_muls ← rq_muls × R_q
20:             temp ← num_q × rq_muls
21:         else // ev is an IDS event
22:           Update distribution of good and bad nodes
23:           Compute Pfp and Pfn
24:           Update E_IDS
25:           E_init ← E_init − E_IDS
26:           Remove bad caught and good
                        misidentified nodes
27:           Compute Q_c
28:           Update λ, N
29:           Update R_q, E_clustering, E_q
30:     temp_MTTF ← temp_MTTF + temp
```

31: $Mttf \leftarrow tempMttf$
32: $N_q \leftarrow num_q$
33: ***if*** $Mttf > optimalMttf$ ***then***
34: $optimalMttf \leftarrow Mttf$
35: $optimal\ (m_p, m_s) \leftarrow (m_p, m_s)$
36: ***return*** $optimalMttf\ and\ optimal\ (m_p, m_s)$

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 environment changes including node density (λ), radio range (r), and node capture rate (λ_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: ***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 within the cluster of the new optimal settings of* T_{IDS} *and* m
7: ***else if*** *event is query arrival* ***then***
8: *trigger multipath routing using* m_s *and* m_p
9: ***else if*** *event is* $T_{clustering}$ *timer* ***then***
10: *perform clustering*
11: ***else if*** *event is* T_{IDS} *timer* ***then***
12: ***For*** *each neighbor node*
13: ***if*** *selected as a voter* ***then***
14: *execute voting based intrusion detection*
15: ***else*** *// event is data packet arrival*
16: *follow multipath routing protocol design to route the data packet*
17:
18: ***SN Execution***:
19: *Get next event*
20: ***if*** *event is* T_D *timer* ***then***
21: *determine radio range to maintain connectivity within a cluster*
22: ***else if*** *event is control packet arrival from CH* ***then***
23: *change the optimal settings of* T_{IDS}, *and* m
24: ***else if*** *event is* $T_{clustering}$ *timer* ***then***
25: *perform clustering*
26: ***else if*** *event is* T_{IDS} *timer* ***then***
27: ***For*** *each neighbor node*
28: ***if*** *selected as a voter* ***then***

29: *execute votingbased intrusion detection*
30: ***else*** *// event is data packet arrival*
31: *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 line 20 for a SN. The optimal design settings in terms of optimal $T_{IDS}, m, m_s,$ 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 (a) adjusting radio range to maintain connectivity (line 4); (b) determining $T_{IDS}, m, m_s,$ and m_p (line 5) based on the sensed environmental conditions at runtime; and (c) notifying SNs within the cluster of the new T_{IDS} and m settings. The action performed by a 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_{IDS} and m settings (line 23) for intrusion detection. When the T_{IDS} timer event happens, each node in the system uses it current T_{IDS} and m settings to perform intrusion detection. The T_{IDS} timer event and the action taken are specified on lines 11-14 for a CH and lines 26-29 for a SN.

When a CH acting as a query processing center (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 7-8. When a data packet arrival event occurs, each node simply follows the prescribed multipath routing protocol to route the packet (lines 15-16 for a CH and lines 30-31 for a SN). Finally each node periodically performs clustering as prescribed by the cluster algorithm, i.e., when a $T_{clustering}$ timer event occurs, each node executes clustering (lines 9-10 for a CH and lines 24-25 for a 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 [11], 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 setup a query-based clustered WSN as in [11] characterized by a set of input parameter values as listed in Table 1. The WSN comprises 1500 SNs deployed in a square area of A^2 (400m×400m). Nodes are distributed in the area following a Poisson process with density λ =15 nodes/(40×40 m^2) at deployment time. The radio range r is 40m. So initially a SN has n= $\lambda \times \pi r^2$=15 neighbor SNs. The probability of a SN becoming a CH is p=1% as in [11]. This means that every SN has the same chance (1%) to server as a CH so that all nodes rotate fairly for the role of CH and thus 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 [11]. Each SN has an initial energy level E_o = 10 Joules. The

energy parameters used by the radio module are adopted from [24,41]. 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 (ε_{amp}) is 10 pJ/bit/m^2. The query arrival rate λ_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 above, we consider parameters related to node compromising events and intrusion detection events. Specifically, the capture rate (λ_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 [19].

TABLE II: INPUT PARAMETER VALUES CHARACTERIZING A HOMOGENEOUS CLUSTERED WSN.

Parameter	Default Value
N	1500 nodes
p	0.01
q	10^{-6}
e_j	[0.0001 – 0.1]
r	40 m
f	0.25
λ	15 nodes/(40 x 40 m^2)
λ_q	1 query/sec
λ_c	Once per 4 days to 28 days
$A \times A$	400m×400m
n_b	50 bits
E_{elec}	50 nJ/bit
E_{amp}	10 pJ/bit/m^2
E_o	10 Joule
$N_{iteration}$	3 iterations
$T_{clustering}$	60 sec
T_{req}	[0.3 – 1.0] sec
H_{pfp} , H_{pfn}	[0.01-0.05]

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 paper 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 λ_c, m and T_{IDS} on optimal (m_p, m_s). Figs. 2 and 3 show MTTF (in sec) vs. (m_p, m_s) under low and high λ_c values, respectively. By comparing these two graphs, we observe a trend that as the capture rate increases (i.e., going from Fig. 2 to Fig. 3), the optimal (m_p, m_s) redundancy level increases. For instance, when the capture rate increases from once in three 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. While increasing (m_p, m_s) consumes more energy, the gain towards increasing the query success probability (and thus towards increasing MTTF) outweighs the loss of lifetime due to energy consumption. It is worth noting that while the trend is intuitive, we identify the exact optimal (m_p, m_s) redundancy level to maximize the WSN lifetime.

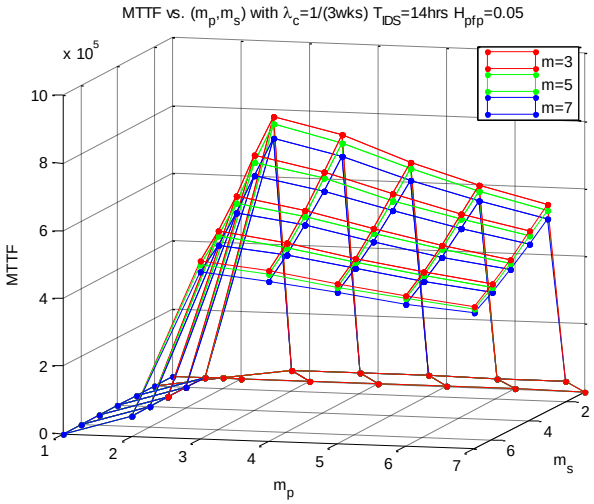

Fig. 2: MTTF vs. (m_p, m_s) under Low Capture Rate.

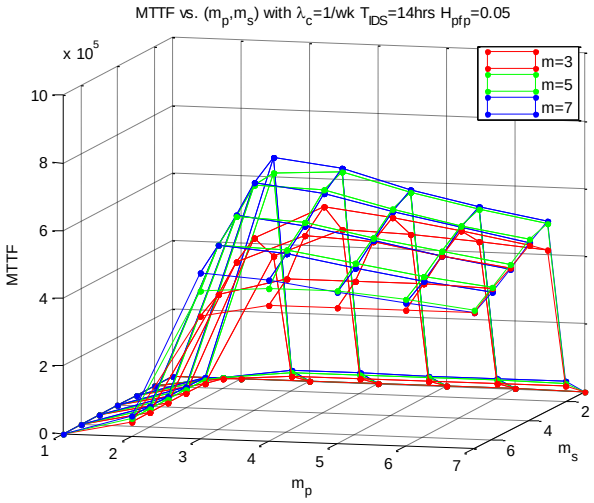

Fig. 3: MTTF vs. (m_p, m_s) under High Capture Rate.

Table III summarizes the effect of λ_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 λ_c there exists an optimal m value that will maximize MTTF. Table IV summarizes the effect of λ_c on the optimal m value at

which MTTF is maximized. When the node capture rate increases from once per 3 weeks to once a week, the 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.

TABLE III: OPTIMAL (m_p, m_s) WITH VARYING λ_c AND m.

	m=3	m=5	m=7
λ_c= 1/(4 days)	(m_p, m_s) =(5,7)	(4,6)	(4,5)
λ_c= 1/week	(4,4)	(3,4)	(3,3)
λ_c= 1/(3 weeks)	(3,3)	(3,3)	(3,3)

TABLE IV: OPTIMAL m WITH VARYING λ_c AND T_{IDS}.

	T_{IDS}=1hr	4hrs	14hrs	20hrs	24hrs
λ_c= 1/(4 days)	m=5	7	7	7	7
λ_c= 1/week	5	5	7	7	7
λ_c= 1/(2 weeks)	5	5	5	5	7
λ_c= 1/(3 weeks)	3	3	3	5	5

Next we analyze the effect of T_{IDS} on MTTF. Figs. 4 and 5 show MTTF vs. 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 Fig. 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, so the intrusion detection capability increases per invocation, there is no need to invoke intrusion detection too often so as not to 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 so intrusion detection is invoked frequently, we don't need many voters per invocation so as not to waste energy unnecessarily to adversely shorten the system lifetime. The second trend shown in Figs. 4 and 5 is that as the node capture rate increases, the optimal m value increases in order to cope with more compromised nodes in the system. These two trends correlate well those summarized in Table IV earlier.

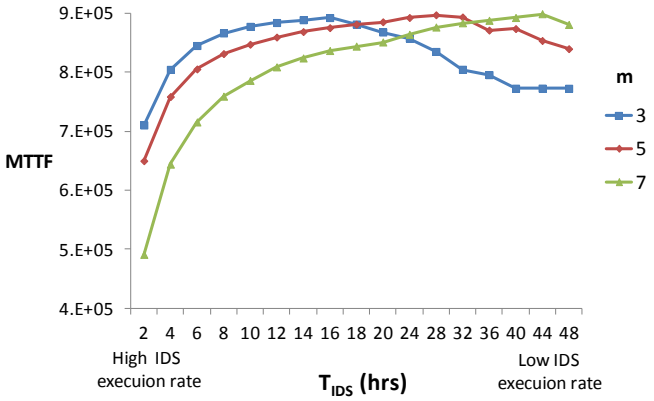

Fig. 4: Effect of T_{IDS} on MTTF under Low Capture Rate.

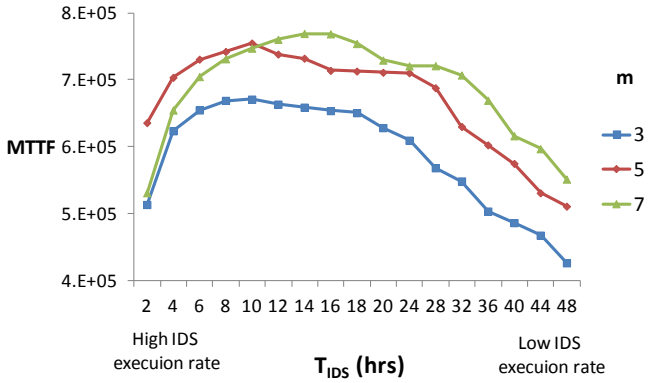

Fig. 5: Effect of T_{IDS} on MTTF under High Capture Rate.

Lastly T_{IDS} is also a tunable parameter to maximize MTTF. Fig. 6 shows MTTF vs. T_{IDS} with varying λ_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 Fig. 6 the optimal T_{IDS} is 20 hours when λ_c is once per 4 weeks and reduces to 6 hours when λ_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 so the intrusion detection capability increases per invocation, there is no need to invoke intrusion detection too often so as not to waste energy and adversely shorten the system lifetime.

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

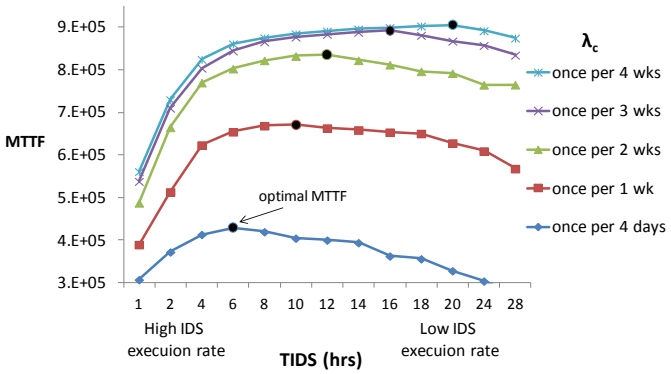

Fig. 6: Effect of Capture Rate on Optimal T_{IDS}.

TABLE V: OPTIMAL T_{IDS} WITH VARYING λ_c AND m.

	m =3	m =5	m =7
λ_c = 1/(4 days)	T_{IDS}=6 hours	6	10
λ_c = 1/week	10	10	14
λ_c= 1/(2 weeks)	12	20	28
λ_c = 1/(3 weeks)	16	28	44

6.2 Runtime Dynamic Redundancy Management

It should be noted that Tables III, IV and V presented above are numerical solutions generated from evaluating the analytical equations derived in Section 4, given node density λ, radio range r, and node capture rate λ_c as input. As the system evolves, all these input parameter values may change, that is, λ 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 λ_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 (λ, r, λ_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 [11]. 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 Fig. 7, we show MTTF vs. λ_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.

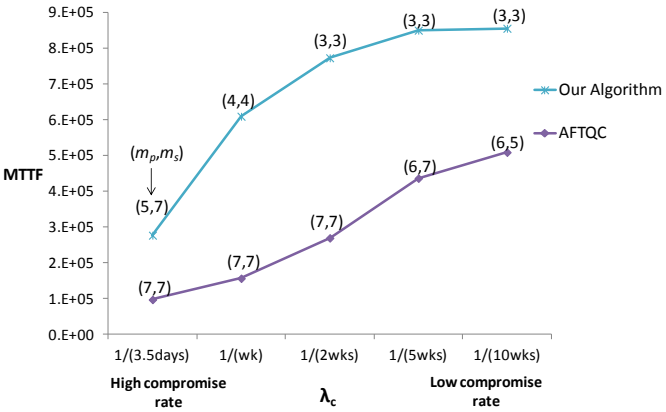

Fig. 7: Performance Comparison with AFTQC.

Our redundancy management algorithm significantly outperforms AFTQC in terms of MTTF obtainable under a wide range of capture rate values, the effect is especially pronounced when the capture rate is high. The result demonstrates the effectiveness of our dynamic redundancy management algorithm which considers both fault intrusion tolerance through multipath routing and intrusion detection through voting, as opposed to AFTQC which considers fault/intrusion tolerance only through multipath routing.

Analyzing the optimal (m_p, m_s) settings reveals insight why our algorithm performs better than AFTQC. That is, 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 tradeoff between intrusion detection/tolerance strength vs. energy consumption. In this paper 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 wireless sensor networks. 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 wireless sensor network 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 of all, we can generalize the analysis methodology to handle both homogeneous and heterogeneous wireless sensor networks [1]. We plan to consider more sophisticated attacker models [31,32], e.g., 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 [4] under which the system fails after it experiences a number of query failures. This paper addresses the best redundancy level for multisource multipath routing, i.e., how many sources and how many paths one should use for multisource multipath routing to maximize the system lifetime. In the future, we plan to explore trust management [2,3,9,17,18] 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 [10,14,15,40] utilizing Petri net modeling techniques [12,13,28,38] to optimize application performance.

ACKNOWLEDGEMENT

This work is supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under grant W911NF-12-1-0445.

REFERENCES

[1] Al-Hamadi, H. and Chen, I.R. Redundancy Management of Multipath Routing for Intrusion Tolerance in Heterogeneous Wireless Sensor Networks. *IEEE Trans. Network and Service Management*, 10, 2 (2013), 189-203.

[2] Bao, F., Chen, I.R., Chang, M. and Cho, J.H. Trust-Based Intrusion Detection in Wireless Sensor Networks. *IEEE Int. Conf. on Communications*. (2011).

[3] Bao, F., Chen, I.R., Chang, M. and Cho, J.H. Hierarchical Trust Management for Wireless Sensor Networks and its Applications to Trust-Based Routing and Intrusion Detection. *IEEE Trans. Netw. Service Manag.* 9, 2 (2012), 161-183.

[4] Bastani, F.B., Chen, I.R. and Tsao, T. Reliability of Systems with Fuzzy-Failure Criterion. *Annu. Reliability and Maintainability Symp.* (1994).

[5] Bhuse, V. and Gupta, A. Anomaly intrusion detection in wireless sensor networks. *J. High Speed Netw.* 15, 1 (2006), 33-51.

[6] Bo, S., Osborne, L., Yang, X. and Guizani, S. Intrusion detection techniques in mobile ad hoc and wireless sensor networks. *IEEE Wireless Commun.* 14, 5 (2007), 56-63.

[7] Bravos, G. and Kanatas, A.G. Energy consumption and trade-offs on wireless sensor networks. *16th IEEE Int. Symp. on Personal, Indoor and Mobile Radio Communications*. 2, (2005), 1279-1283.

[8] Chatzigiannakis, I. and Strikos, A. A decentralized intrusion detection system for increasing security of wireless sensor networks. *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*. (2007), 1408-1411.

[9] Chen, I.R., Bao, F., Chang, M. and Cho, J.H. Trust Management for Encounter-Based Routing in Delay Tolerant Networks. *IEEE Global Communications Conf.* (2010), 1-6.

[10] Chen, I.R. and Hsi, T.H. Performance analysis of admission control algorithms based on reward optimization for real-time multimedia servers. *Performance Evaluation.* 33, 2 (1998), 89-112.

[11] Chen, I.R., Speer, A.P. and Eltoweissy, M. Adaptive Fault-Tolerant QoS Control Algorithms for Maximizing System Lifetime of Query-Based Wireless Sensor Networks. *IEEE Trans. on Dependable and Secure Computing.* 8, 2 (2011), 161-176.

[12] Chen, I.R. and Wang, D.C. Analysis of replicated data with repair dependency. *The Computer Journal.* 39, 9 (1996), 767-779.

[13] Chen, I.R. and Wang, D.C. Analyzing dynamic voting using petri nets. *15th IEEE Symp. on Reliable Distributed Systems.* (1996), 44-53.

[14] Cheng, S.T., Chen, C.M. and Chen, I.R. Dynamic quota-based admission control with sub-rating in multimedia servers. *Multimedia systems*. 8, 2 (2000), 83-91.
[15] Cheng, S.T., Chen, C.M. and Chen, I.R. Performance evaluation of an admission control algorithm: dynamic threshold with negotiation. *Performance Evaluation*. 52, 1 (2003), 1-13.
[16] Cho, J.H., Chen, I.R. and Feng, P.G. Effect of Intrusion Detection on Reliability of Mission-Oriented Mobile Group Systems in Mobile Ad Hoc Networks. *IEEE Trans. Rel.* 59, 1 (2010), 231-241.
[17] Cho, J.H., Swami, A. and Chen, I.R. Modeling and analysis of trust management for cognitive mission-driven group communication systems in mobile ad hoc networks. *International Conference on Computational Science and Engineering*. (2009), 641-650.
[18] Cho, J.H., Swami, A. and Chen, I.R. Modeling and analysis of trust management with trust chain optimization in mobile ad hoc networks. *Journal of Network and Computer Applications*. 35, 3 (2012), 1001-1012.
[19] Da Silva, A.P.R., Martins, M.H.T., Rocha, B.P.S., Loureiro, A.F., Ruiz, L.B. and Wong, H.C. Decentralized intrusion detection in wireless sensor networks. *1st ACM Workshop on Quality of Service & Security in Wireless and Mobile Networks*. (2005), 16-23.
[20] Deb, B., Bhatnagar, S. and Nath, B. ReInForM: reliable information forwarding using multiple paths in sensor networks. *28th IEEE Local Computer Networks*. (2003), 406-415.
[21] Deng, J., Han, R. and Mishra, S. INSENS: Intrusion-tolerant routing for wireless sensor networks. *Computer Communications*. 29, 2 (2006), 216-230.
[22] Felemban, E., Chang-Gun, L. and Ekici, E. MMSPEED: multipath Multi-SPEED protocol for QoS guarantee of reliability and. Timeliness in wireless sensor networks. *IEEE Trans. Mobile Comput.* 5, 6 (2006), 738-754.
[23] Haowen, C. and Perrig, A. PIKE: peer intermediaries for key establishment in sensor networks. *24th Annu. Joint Conf. of the IEEE Computer and Communications Societies*. (2005), 524-535.
[24] Heinzelman, W.B., Chandrakasan, A.P. and Balakrishnan, H. An application-specific protocol architecture for wireless microsensor networks. *IEEE Trans. Wireless Commun.* 1, 4 (2002), 660-670.
[25] Hoang, H.T. and Nam, H.E. Optimal Selection and Activation of Intrusion Detection Agents for Wireless Sensor Networks. *Future Generation Communication and Networking*. 1, (2007), 350-355.
[26] Kang, K.D., Liu, K. and Abu-Ghazaleh, N. Securing Geographic Routing in Wireless Sensor Networks. *9th Annu. Cyber Security Conf. on Information Assurance*. (2006).
[27] Krontiris, I., Dimitriou, T. and Freiling, F.C. Towards intrusion detection in wireless sensor networks. *13th European Wireless Conference*. (2007).
[28] Li, Y. and Chen, I.R. Design and performance analysis of mobility management schemes based on pointer forwarding for wireless mesh networks. *IEEE Trans. Mobile Computing*. 10, 3 (2011), 349-361.
[29] Lou, W. and Kwon, Y.H. SPREAD: a hybrid multipath scheme for secure and reliable data collection in wireless sensor networks. *IEEE Trans. Veh. Technol.* 55, 4 (2006), 1320-1330.
[30] Misra, S., Krishna, P.V. and Abraham, K.I. Energy efficient learning solution for intrusion detection in Wireless Sensor Networks. *Second International Conference on Communication Systems and Networks*. (2010), 1-6.
[31] Mitchell, R. and Chen, I.R. Behavior-Rule Based Intrusion Detection Systems for Safety Critical Smart Grid Applications. *IEEE Transactions on Smart Grid*. 99 (2013), 1-10.
[32] Mitchell, R. and Chen, I.R. Effect of Intrusion Detection and Response on Reliability of Cyber Physical Systems. *IEEE Transactions on Reliability*. (2013).
[33] Nasser, N. and Chen, Y. SEEM: Secure and energy-efficient multipath routing protocol for wireless sensor networks. *Computer Communications*. 30, 11-12 (2007), 2401-2412.
[34] Qun, S. Power Management in Networked Sensor Radios A Network Energy Model. *IEEE Sensors Applications Symp.* (2007), 1-5.
[35] Shu, T., Krunz, M. and Liu, S. Secure Data Collection in Wireless Sensor Networks Using Randomized Dispersive Routes. *IEEE Trans. Mobile Comput.* 9, 7 (2010), 941-954.
[36] Stavrou, E. and Pitsillides, A. A survey on secure multipath routing protocols in WSNs. *Comput. Netw.* 54, 13 (2010), 2215-2238.
[37] Su, W.T., Chang, K.M. and Kuo, Y.H. eHIP: An energy-efficient hybrid intrusion prohibition system for cluster-based wireless sensor networks. *Computer Networks*. 51, 4 (2007), 1151-1168.
[38] Trivedi, K.S. Stochastic Petri Net Package. *Name*. ((1999).
[39] Yang, Y., Zhong, C., Sun, Y. and Yang, J. Network coding based reliable disjoint and braided multipath routing for sensor networks. *J. Netw. Comput. Appl.* 33, 4 (2010), 422-432.
[40] Yilmaz, O. and Chen, I.R. Utilizing call admission control for pricing optimization of multiple service classes in wireless cellular networks. *Computer Communications*. 32, 2 (2009), 317-323.
[41] Younis, O. and Fahmy, S. HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Trans. Mobile Comput.* 3, 4 (2004), 366-379.