Secure and Reliable Multisource Multipath Routing in Clustered Wireless Sensor Networks

Ing-Ray Chen, Member, IEEE, Hamid Al-Hamadi

Abstract—Multisource multipath data routing to a remote sink node is an effective way to cope with unreliable and/or malicious nodes in wireless sensor networks (WSNs). In this paper we analyze the optimal amount of redundancy in terms of the number of source sensors sensing the same physical phenomena and the number of paths through which data are routed to a remote sink node in the presence of unreliable and malicious nodes so that the query success probability is maximized while maximizing the sensor network lifetime. We consider this optimization problem in the case in which a voting-based distributed intrusion detection algorithm is applied to remove malicious nodes from the sensor network. Our model-based analysis results indicate that the optimal multisource multipath redundancy and intrusion detection settings for query-based clustered WSNs exist and such optimal settings evolve dynamically in response to changing environment conditions.

Index Terms—Wireless sensor networks, multisource multipath routing, redundancy engineering, intrusion detection, security, reliability, timeliness.

1 INTRODUCTION

Advances in wireless sensor networks (WSNs) lead to its wide deployment across many fields. Many WSN applications have high quality of service (QoS) requirements in security, reliability and timeliness. Also many applications 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. 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 of the WSN.

Multipath routing is considered an effective way to improve data delivery in WSNs. 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 to improve reliability [5], [21], [8], some attention has been paid to using multipath routing to tolerate insider attacks [4], [20], [11]. These studies (except [8]), however, largely ignored the tradeoff between QoS gain vs. energy consumption which can adversely shorten the system lifetime. Very recently Chen et al. [8] investigated this issue in the context of multisource multipath data forwarding in query-based clustered WSNs. They identified the best redundancy levels to be employed in multipath routing to trade energy off for reliability gain so that the lifetime of the query-based WSN is maximized. However, no consideration was given to the presence of inside attackers.

1 Ing-Ray Chen and Hamid Al-Hamadi are with the Department of Computer Science, Virginia Tech, 7054 Haycock Rd., Falls Church, VA 22043. E-mail: {irchen, hhamadi}@vt.edu.
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. Moreover very likely the system would employ an intrusion detection system (IDS) with the goal to detect and remove malicious nodes. While the literature is abundant in intrusion detection techniques for WSNs [2, 9-10, 19, 23], the issue of how often intrusion detection should be invoked to remove potentially malicious nodes so that the system lifetime is maximized (say to prevent a Byzantine failure [13]) is largely unexplored. The issue is especially critical for energy-constrained WSNs designed to stay alive for a long mission time.

In this paper we address the tradeoff between energy consumption vs. gain in reliability and security with the goal to maximize the lifetime of a query-based WSN. More specifically, we analyze the optimal amount of redundancy in terms of the number of source SNs sensing the same physical phenomena and the number of paths through which data are routed to a remote sink in the presence of malicious nodes so that so that the query success probability is maximized while maximizing the WSN lifetime. We consider this optimization problem in the case in which a voting-based distributed intrusion detection algorithm is applied to remove malicious nodes from the WSN. Our contribution is a model-based analysis methodology by which the optimal multisource multipath redundancy levels and intrusion detection settings may be identified for lifetime maximization of query-based WSNs.

The rest of the paper is organized as follows. In Section 2 we discuss related work and contrast our approach with existing work on multipath routing for intrusion tolerance and reliability enhancement. In Section 3, we define our system model with system assumptions given. In Section 4 we derive an analytical expression for the system lifetime, considering factors such as query rate, attacker behavior, node capture rate, link reliability, and energy consumption. In Section 5 we present numerical data and provide physical interpretations of the results. Finally in Section 6 we discuss applicability and outline some future research areas.

2 RELATED WORK

Over the past few years, numerous protocols have been proposed to detect intrusion in WSNs (e.g., [2, 23] provide excellent surveys of the subject). 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. 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 [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 [9], 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 associated with a distributed IDS, nor the issue of maximizing the WSN lifetime while satisfying QOS requirements in security, reliability and timeliness. Our voting-based IDS approach extends from [10] 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 literature, many multipath routing protocols have been proposed for wireless sensor networks. In [11] 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 blackhole 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 [4] 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. Our approach is totally distributed with hop-by-hop formation of multiple paths.

SEEM [14] is a multipath routing protocol that also relies on a powerful base station to perform route discovery, maintenance, and route selection. The base station takes into account the remaining energy of nodes when selecting a shortest path between source and sink. SEEM shows improvement over directed diffusion in certain performance metrics such as network throughput, communication overhead, and network lifetime. Furthermore, it has some resistance against false routing path attacks since only the base station can select the routing paths. 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 [18], 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 if \( k \) out of \( n \) shares are received then the packet can be reconstructed. 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 con-
sider 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 to maximize the system lifetime.

MMSPEED [5] is a multipath and multi-speed routing protocol that provides QOS guarantees in both timeliness and reliability domains. End-to-end requirements are guaranteed in a localized way without global network information or a priori path setup. It also adopts geographic forwarding for packet delivery. However there is no consideration given to the presence of compromised nodes. Our solution of satisfying timeliness and reliability requirements of a query is totally distributed, which in some sense follows the design principle of MMSPEED. Contrast to MMSPEED, our work considers not only timeliness and reliability but also security issues, with the multipath multisource routing problem, in the presence of malicious nodes and a voting based IDS, being formulated as a WSN lifetime optimization problem.

HEED [22] employs an energy-efficient distributed clustering approach for wireless sensor networks. The protocol aims to extend the lifetime of all the nodes in the network by distributing the energy consumption across the nodes. The role of cluster head is periodically changed based on residual energy and node proximity between sensor nodes such that energy consumption is distributed evenly among all sensors. We also consider cluster-based WSNs for energy reasons and we adopt an energy-efficient distributed clustering approach such as HEED to fairly share load among sensor nodes so that all sensor nodes consume energy fairly evenly.

3 SYSTEM MODEL

We consider a WSN with low-power SNs distributed in a geographic area through air-drop. The WSN is characterized by the following:

1. SNs are homogenous with the same initial energy ($E_0$).
2. The deployment area of the WSN is of size $A^2$.
3. SNs are distributed according to a homogeneous spatial Poisson process with intensity $\lambda$. 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.
4. 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 hostility to the WSN is characterized by a per-node capture rate of $\lambda_c$ which can be determined based on historical data and knowledge about the target applica-
tion environment. These probabilities are assumed to be constant and known at the time of deployment.

5. The WSN is cluster-based, where CHs are elected periodically using an energy-saving clustering algorithm (e.g., [7, 22]), and form clusters with non-CH nodes. The clustering algorithm ensures that the energy due to the role of CH is distributed fairly evenly among nodes by performing a fair rotation of the CH role among SNs.

6. 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.

7. 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.

8. 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.

9. We assume that SNs operate in power saving mode (e.g. [3, 15]). 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 [22].

10. We assume that the WSN executes a pairwise key establishment protocol (e.g., [6, 24]) 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.

11. We assume that when a node is compromised, it performs bad-mouthing attacks (recommending a good node as a bad node and a bad node as a good node when serving as a recommender) and packet dropping attacks [12] to disrupt the operation of the network.

12. To detect and remove malicious nodes from the system, a voting-based distributed 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 host IDS using overhearing and promiscuous monitoring techniques (e.g., [1, 16, 19]) to assess its neighbors. The flaws 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. 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 bad-mouthing attack model in the paper.

Fig. 1 shows a scenario where the percentage of compromised nodes is low relative to that of Fig. 2. In Fig. 1 we illustrate that we only need to choose a source redundancy of 2 ($m_s = 2$) and a path redundancy of 2 ($m_p = 2$) to maintain the source-SN-to-PC connectivity for query response delivery. As the bad node population increases as in Fig. 2, we need to use $m_s = 3$ and $m_p = 3$ to maintain the source-SN-to-PC connectivity.

Here we note that increasing source or path redundancy enhances reliability and security of source-SN-to-PC connectivity. However, it also increases the energy consumption, thus contributing to the decrease of the system lifetime. Thus, there is a tradeoff between reliability/security gain vs. energy consumption. The distributed IDS design attempts to detect and evict compromised nodes from the network without unnecessarily wasting energy so as to maximize the query success probability and the system lifetime. The effectiveness of the IDS depends on its parameters ($T_{IDS}$ and $m$). While a shorter $T_{IDS}$ or a higher $m$ can result in low $P_{fp}$ and $P_{fn}$, it also consumes more energy from the WSN nodes. Thus, this is another design tradeoff. 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. A failure occurs when no
response is received before the query deadline. The cause could be due to energy exhaustion, packet dropping by malicious nodes, channel/node failure, or insufficient transmission speed to meet the timeliness requirement. Our aim is to find both the optimal redundancy levels and IDS settings under which the MTTF is maximized, when given a set of parameters characterizing the operational and environment conditions.

4 PROBABILITY MODEL

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Length of each side of a square sensor area (meter)</td>
<td>input</td>
</tr>
<tr>
<td>ns</td>
<td>Size of a data packet (bit)</td>
<td>input</td>
</tr>
<tr>
<td>Edc</td>
<td>Energy dissipation to run the transmitter and receiver circuitry (J/bit)</td>
<td>input</td>
</tr>
<tr>
<td>Eamp</td>
<td>Energy used by the transmit amplifier to achieve an acceptable signal to noise ratio (J/bit/m²)</td>
<td>input</td>
</tr>
<tr>
<td>Er</td>
<td>Initial energy per SN (Joule)</td>
<td>input</td>
</tr>
<tr>
<td>Einit</td>
<td>Initial energy of the WSN (Joule)</td>
<td>derived</td>
</tr>
<tr>
<td>Eclustering(t)</td>
<td>Energy consumed for executing the clustering algorithm at time t (Joule)</td>
<td>derived</td>
</tr>
<tr>
<td>EIDS(t)</td>
<td>Energy consumed for executing an IDS algorithm at time t (Joule)</td>
<td>input</td>
</tr>
<tr>
<td>Rq(t)</td>
<td>Probability that a query reply at time t is delivered successfully by the deadline</td>
<td>derived</td>
</tr>
<tr>
<td>r</td>
<td>Wireless radio communication range (meter)</td>
<td>input</td>
</tr>
<tr>
<td>p</td>
<td>Probability of a SN becoming a CH</td>
<td>derived</td>
</tr>
<tr>
<td>q</td>
<td>SN hardware failure probability</td>
<td>input</td>
</tr>
<tr>
<td>ε</td>
<td>Transmission failure probability of SNj</td>
<td>input</td>
</tr>
<tr>
<td>N(t)</td>
<td>Number of SNs in the WSN at time t</td>
<td>input</td>
</tr>
<tr>
<td>n(t)</td>
<td>Number of neighbor SNs at time t</td>
<td>derived</td>
</tr>
<tr>
<td>ngood(t)</td>
<td>Number of good neighbor SNs at time t</td>
<td>derived</td>
</tr>
<tr>
<td>nbad(t)</td>
<td>Number of bad neighbor SNs at time t</td>
<td>derived</td>
</tr>
<tr>
<td>Nq</td>
<td>Maximum number of queries before energy exhaustion</td>
<td>derived</td>
</tr>
<tr>
<td>Nclustering</td>
<td>Number of iterations in clustering for CH election</td>
<td>derived</td>
</tr>
<tr>
<td>mp</td>
<td>Path redundancy level: Number of paths from a source CH to the sink</td>
<td>design</td>
</tr>
<tr>
<td>ms</td>
<td>Source redundancy level: Number of SNs per cluster in response to a query</td>
<td>design</td>
</tr>
<tr>
<td>f</td>
<td>Fraction of neighbor SNs that will forward data</td>
<td>input</td>
</tr>
<tr>
<td>λ(t)</td>
<td>SN population density (nodes/meter²) at time t</td>
<td>derived</td>
</tr>
<tr>
<td>λ</td>
<td>SN population density at deployment time</td>
<td>input</td>
</tr>
<tr>
<td>λq</td>
<td>Query arrival rate (times/sec)</td>
<td>input</td>
</tr>
<tr>
<td>Sk</td>
<td>Progressive transmission speed between SNi and SNj (meter/sec)</td>
<td>derived</td>
</tr>
<tr>
<td>Tclustering</td>
<td>Time interval for executing the clustering algorithm (sec)</td>
<td>input</td>
</tr>
<tr>
<td>Treq</td>
<td>Query deadline requirement (sec)</td>
<td>input</td>
</tr>
<tr>
<td>λc</td>
<td>Node capture rate</td>
<td>input</td>
</tr>
<tr>
<td>α</td>
<td>Ratio of IDS execution rate to query arrival rate</td>
<td>input</td>
</tr>
<tr>
<td>β</td>
<td>Ratio of clustering rate to query arrival rate</td>
<td>input</td>
</tr>
<tr>
<td>m</td>
<td>Number of voters selected for executing distributed IDS</td>
<td>design</td>
</tr>
<tr>
<td>Hpf</td>
<td>Probability of host IDS false positive</td>
<td>input</td>
</tr>
<tr>
<td>Hfn</td>
<td>Probability of host IDS false negative</td>
<td>input</td>
</tr>
<tr>
<td>Pfp</td>
<td>Probability of distributed IDS false positive</td>
<td>derived</td>
</tr>
<tr>
<td>Pfn</td>
<td>Probability of distributed IDS false negative</td>
<td>derived</td>
</tr>
<tr>
<td>TIDS</td>
<td>IDS interval time (sec)</td>
<td>design</td>
</tr>
<tr>
<td>MTTF</td>
<td>Lifetime of a query-based WSN</td>
<td>output</td>
</tr>
</tbody>
</table>
data forwarding to answer queries issued from a mobile user roaming in the WSN area. Table 1 provides the notation used for symbols and their physical meanings. A parameter is labeled as input, derived, design or output. In particular, \(m_p\) (path redundancy), \(m_s\) (source redundancy), \(m\) (the number of voters for intrusion detection) and \(T_{IDS}\) (the intrusion detection interval) are design parameters whose values may be fine-tuned to maximize MTTF, given a set of input parameter values charactering the operational and environmental conditions. Derived parameters are those deriving from input parameters. There is only one output parameter, namely, MTTF. Note that most derived parameters are dynamic, i.e., as a function of time. For example, the node density \(\lambda(t)\) decreases over time because of node failure/eviction as time progresses.

The basic idea of our MTTF formulation is that we first deduce the maximum number of queries, \(N_q\), the system can possible 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, the average interval between query arrivals is \(1/\lambda_q\). So 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 to the PC 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 computed by:

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

(1)

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

### 4.1 Intrusion Detection and Evolving Bad Node Population

Initially at deployment time all SNs are good nodes. Assume that the capture time of a SN is exponentially distributed with per node capture rate \(\lambda_c\) reflecting the hostility of the environment. Then, the probability of a good node being compromised over a time interval \(T_{IDS}\), denoted by \(P_c\), is given by:

\[
P_c = 1 - e^{-\lambda_c \times T_{IDS}}
\]

(2)

Recall that the voting-based distributed IDS is run periodically with \(T_{IDS}\) being the interval. At the \(i\)th IDS execution time (denoted by \(t_{i,1}\)), good nodes may have been compromised with probability \(P_c\) since the previous IDS execution
time \((t_{i,i-1})\). Let \(n_{\text{good}}(t)\) and \(n_{\text{bad}}(t)\) denote the numbers of good and bad neighbor nodes at time \(t\), respectively, with \(n_{\text{good}}(t) + n_{\text{bad}}(t) = n(t)\). 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 \(t_{i,i-1}\) as follows:

\[
\begin{align*}
n_{\text{good}}(t_{i,i}) & = n_{\text{good}}(t_{i,i-1}) - n_{\text{good}}(t_{i,i-1}) \times p_c \\
n_{\text{bad}}(t_{i,i}) & = n_{\text{bad}}(t_{i,i-1}) + n_{\text{good}}(t_{i,i-1}) \times p_c
\end{align*}
\]  

(3)

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

\[
P_{fp} \text{ or } P_{fn} = \sum_{i=0}^{m-m_{\text{maj}}} \left[ \frac{C \left( \binom{n_{\text{bad}}}{m_{\text{maj}} + i} \times C \left( \binom{m - (m_{\text{maj}} + i)}{n_{\text{good}}} \right) \right)}{C \left( \binom{n_{\text{bad}} + n_{\text{good}}}{m} \right)} \right] + \sum_{i=0}^{m-m_{\text{maj}}} \left[ \frac{C \left( \binom{n_{\text{bad}}}{i} \right) \times \sum_{j=m_{\text{maj}}-i}^{m-i} \left[ C \left( \binom{m_{\text{maj}} + j}{m_{\text{maj}} - i - j} \right) \times C \left( \binom{m - i - j}{n_{\text{good}} - j} \right) \times \left( 1 - \omega \right)^{m-i-j} \right]}{C \left( \binom{n_{\text{bad}} + n_{\text{good}}}{m} \right)} \right]
\]

(4)

where \(m_{\text{maj}}\) is the minimum majority of \(m\), e.g., 3 is the minimum majority of 5, and \(\omega\) is \(H_{pfp}\) for calculating \(P_{fp}\) and \(H_{pfn}\) for calculating \(P_{fn}\). We explain Equation 4 for the false positive probability at time \(t\) below. The explanation to the false negative probability is similar. A false positive results when the majority of the voters vote the target node (which is a good SN) as compromised. The first term in Equation 4 accounts for the case in which more than 1/2 of the voters selected from the target node’s neighbors are bad SNs 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 \(m_{\text{maj}}\) bad voters out of \(n_{\text{bad}}\) nodes and the remaining good voters out of \(n_{\text{good}}\) nodes. The second term accounts for the case in which more than 1/2 of the voters selected from the neighbors are good SNs but unfortunately some of these good nodes mistakenly misidentify the target SN as a bad node with probability \(H_{pfn}\), resulting in more than 1/2 of the voters (some of those may be bad SNs) 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 \(m_{\text{maj}}\) 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 \(\overline{n}_{\text{good}}(t)\) be the number of good neighbor nodes at time \(t\) right after IDS execution. Then,
\[ n_{\text{good}}(t_{i,i}) = n_{\text{good}}(t_{i,i}) - n_{\text{good}}(t_{i,i}) \times P_f \]  

(5)

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,

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

(6)

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:

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

(7)

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

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

(8)

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

(9)

### 4.2 Query Success Probability

To compute the query success probability \( R_q(t_{Q,i}) \) for query \( i \) which arrives at time \( t_{Q,i} \), we follow the formulation in [8] with adjustments made to take into consideration of query failures due to security attacks. We omit time in the derivation below for notational conveniences. 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 \( T_{\text{req}} \). Let \( d_{\text{SN-CH}} \) be the expected distance between a SN (selected to report sensor readings) and its CH and \( d_{\text{CH-PC}} \) be the expected distance between the source CH and the PC accepting the query result. Given a query dead-
line $T_{\text{req}}$ as input, a data packet from a SN through its CH to the PC must reach the PC within $T_{\text{req}}$. Thus, the minimum hop-by-hop transmission speed denoted by $S_{\text{req}}$ is given by:

$$S_{\text{req}} = \frac{d_{SN\text{-}CH} + d_{CH\text{-}PC}}{T_{\text{req}}} \quad (10)$$

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 $\lambda$, CHs and non-CH SNs will also be distributed in accordance with a spatial Poisson process with rates $p\lambda$ and $(1-p)p\lambda$ respectively. Non-CH SNs thus would join the closest CH to form a Voronoi cell [17] 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 a SN to its CH is given by $d_{SN\text{-}CH} = 1/2(p\lambda)^{1/2}$. 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, $d_{CH\text{-}PC}$ essentially is the average distance between any two CHs in the WSN. Given location randomness of CHs in the square area $A^2$, it can be shown geometrically that the average distance between any two CHs is $d_{CH\text{-}PC} = 0.382A$. With the knowledge of $d_{SN\text{-}CH}$ and $d_{CH\text{-}PC}$ 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 $N_{SC}^h = d_{SN\text{-}CH}/r$ and $N_{CP}^h = d_{CH\text{-}PC}/r$ where $r$ is radio range.

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 [5]. If $S_{jk}$ is above $S_{\text{req}}$ 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:

$$Q_{t,jk} = cdf (S_{jk} \leq S_{\text{req}}) = \frac{S_{\text{req}} - a}{b - a} \quad (11)$$

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:

$$Q_{r,j} = 1 - [(1-q)(1-e_j)] \quad (12)$$
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 7. 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:

$$Q_{rtc,jk} = 1 - [(1 - Q_{r,j})(1 - Q_{i,j})(1 - Q_{c,j})] \quad (13)$$

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:

$$\theta_j = 1 - \prod_{k=1}^{f_{sn}} Q_{rtc,jk} \quad (14)$$

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 8.

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, or packet dropping 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:

$$\Theta(N_{cp}^h) = \left( \prod_{j=1}^{N_{cp}^h-1} \theta_j \right) \times (1 - Q_{rtc, N_{cp}^h(N_{cp}^h+1)}) \quad (15)$$

For path redundancy, we create $m_p$ paths between the source CH and the PC. The $m_p$ paths are formed by choosing $m_p$ SNs in the first hop and then choosing only one SN 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 randomly 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:
Following the same derivation to Equation 15, the success probability of a single path from a SN to its CH is given by:

\[
\Theta(N_{SC}^h) = \left( \prod_{j=1}^{N_{SC}^h-1} \theta_j \right) \times (1 - Q_{rt,N_{SC}^h\max})
\]  

where

\[
Q_{rt,N_{SC}^h\max} = \prod_{i=1}^{m_2} \left[ 1 - \Theta_i (N_{CP}^h - 1) \right]
\]

For source redundancy we use \(m_2\) SNs to report query responses to their source CH. The probability that all \(m_2\) SNs fail to deliver data to their CH is given by:

\[
Q_{fs}^{m_2} = \prod_{i=1}^{m_2} \left[ 1 - \Theta_i (N_{SC}^h) \right]
\]  

Consequently, the failure probability of data delivery from \(m_2\) SNs to the CH, and subsequently using \(m_p\) paths to relay data from CH to PC, is given by:

\[
Q_f = 1 - (1 - Q_{fp}^{m_p})(1 - Q_{fs}^{m_2})
\]  

Therefore, the query success probability is given by:

\[
R_q = 1 - Q_f
\]  

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 8) and population density (Equation 9) at the \(i^{th}\) query’s execution time (i.e., at time \(t_{Q,i}\)).

### 4.3 Energy Consumption

Now 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 \(N_{sp}\), the maximum number of queries the system can possibly handle before running into energy exhaustion. Our energy model follows [22]. 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 \( \alpha \) 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:

\[
E_{\text{init}} = \sum_{i=1}^{\alpha N_q} E_{\text{IDS}}(t_{i,i}) + \sum_{i=1}^{\beta N_q} E_{\text{clustering}}(t_{c,i}) + \sum_{i=1}^{N_q} E_q(t_{q,i}) \tag{21}
\]

Below we outline how to calculate \( E_{\text{IDS}}(t_{i,i}) \), \( E_{\text{clustering}}(t_{c,i}) \) and \( E_q(t_{q,i}) \). We first estimate energy consumed by transmission and reception over wireless link. The energy spent by a SN to transmit an encrypted data packet of length \( n_b \) bits over a distance \( r \) is estimated as [22]:

\[
E_T = n_b(E_{\text{elec}} + E_{\text{amp}} r^2) \tag{22}
\]

Here \( E_{\text{elec}} \) is the energy dissipated to run the transmitter and receiver circuitry, \( E_{\text{amp}} \) 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:

\[
E_R = n_s E_{\text{elec}} \tag{23}
\]

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.,

\[
E_q(t_{q,i}) = E_{CP}(t_{q,i}) + E_{SC}(t_{q,i}) \tag{24}
\]

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.,

\[
E_{CP}(t_{q,i}) = \left\{ E_T + n(t_{q,i})E_R \right\} + m_p (N_{CP}^b - 1) \left\{ E_T + n(t_{q,i})E_R \right\} \tag{25}
\]

Here the first term accounts for the transmission energy consumed by the source CH and the reception energy con-
sumed 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 $M^h_{m_p} - 1$ hops. We note that the number of neighbor SNs at time $t$, $n(t) = \lambda(t) \times \pi r^2$ by Equation 9, 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 $m_p$ paths each with $N_{SC}^h$ hops, i.e.,

$$E_{SC}(t_{Q,i}) = m_i N_{SC}^h \left[ E_T + n(t_{Q,i})E_R \right]$$  \hspace{1cm} (26)

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 equals $N_{SC}^h$. Thus, the energy required for broadcasting is $pN(t_{c,i})[N_{SC}^h n(t_{c,i})(E_T + E_R)]$. 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 $pN(t_{c,i})$ 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:

$$E_{clustering}(t_{c,i}) = pN(t_{c,i})N_{iteration} \left[ N_{SC}^h n(t_{c,i}) (E_T + E_R) \right] + N(t_{c,i}) (E_T + E_R)$$  \hspace{1cm} (27)

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:

$$E_{IDS}(t_{i,l}) = N(t_{i,l-1})[m(m - 1)] [E_T + n(t_{i,l-1})E_R]$$  \hspace{1cm} (28)

Once we obtain $E_{IDS}(t_{i,l})$, $E_{clustering}(t_{c,i})$ and $E_q(t_{Q,i})$ from Equations 28, 27 and 24, respectively, we calculate $N_q$ from Equation 21. The knowledge of $N_q$ along with $R_q(t_{Q,i})$ in Equation 20 allows us to calculate the system MTTF given by Equation 1.

5 **Performance Evaluation**

In this section, we present numerical data obtained as a result of applying Equation 1. Table 2 lists the set of input parameter values characterizing a query-based clustered WSN. Our example WSN consists of 1500 nodes deployed in a
square area of $A^2$ (400m×400m). Nodes are distributed in the area following a Poisson process with density $\lambda = 15$ nodes/(40×40 m$^2$) at deployment time. The radio range $r$ is 40m. So initially a SN has $n = \lambda \times \pi r^2 = 15\pi$ neighbor SNs. The probability of a SN becoming a CH is $p = 1\%$. So initially a cluster has $1/p = 100$ nodes and there are 15 clusters in the system. Each SN has an initial energy level $E_0 = 10$ Joules. The energy parameters used by the radio module are adopted from [7, 22]. 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. The inter-arrival time in between attacks ($T_{comp}$) is between 4 and 28 days, corresponding to an attack rate ($\lambda_c$) of once per 4 days to once per 28 days. The host IDS false positive probability and false negative probability ($H_{pf}$ and $H_{pfn}$) vary between 1% and 5% to reflect the host intrusion detection strength as in [19].

Our objective is to identify the best setting in terms of $m_p$ (path redundancy), $m_s$ (source redundancy), $m$ (the number of voters for intrusion detection) and $T_{IDS}$ (the intrusion detection interval) to maximize MTTF, given a query-based clustered WSN characterized by a set of input parameter values as listed in Table 2.

**TABLE 2**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
<th>Parameter</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>1500</td>
<td>$A$</td>
<td>400m</td>
</tr>
<tr>
<td>$p$</td>
<td>0.01</td>
<td>$n_b$</td>
<td>50 bits</td>
</tr>
<tr>
<td>$q$</td>
<td>$10^{-6}$</td>
<td>$E_{elec}$</td>
<td>50 nJ/bit</td>
</tr>
<tr>
<td>$e_j$</td>
<td>[0.0001 – 0.1]</td>
<td>$\epsilon_{amp}$</td>
<td>10 pJ/bit/m$^2$</td>
</tr>
<tr>
<td>$r$</td>
<td>40 m</td>
<td>$E_o$</td>
<td>10 Joule</td>
</tr>
<tr>
<td>$f$</td>
<td>$1/4$</td>
<td>$N_{iteration}$</td>
<td>3</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>15 nodes/(40 x 40 m$^2$)</td>
<td>$T_{clustering}$</td>
<td>60 sec</td>
</tr>
<tr>
<td>$\lambda_q$</td>
<td>1 query/sec</td>
<td>$T_{req}$</td>
<td>[0.3 – 1.0] sec</td>
</tr>
<tr>
<td>$T_{comp}$ (or 1/$\lambda_c$)</td>
<td>[4-28] days</td>
<td>$H_{pf}$, $H_{pfn}$</td>
<td>[0.01-0.05]</td>
</tr>
</tbody>
</table>

Fig. 3 shows 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 1) is embedded on lines 14-20 and 30-31 in Fig. 3. 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 as listed in Table 2 (above) as well as instance values of $m$ (the number of voters for intrusion detection) and $T_{IDS}$ (the intrusion detection interval) characterizing a query-based WSN.
Input: Table 2 input parameters
Output: optimalMttf, optimal \((m_p, m_s)\)

1: \textbf{for} \(m_s \leftarrow 1 \) \textbf{to} maxMs \textbf{do}
2: \hspace{1em} \textbf{for} \(m_p \leftarrow 1 \) \textbf{to} maxMp \textbf{do}
3: \hspace{2em} \text{num}_{q} \leftarrow 0 \quad \text{where num}_{q} \text{is the query counter}
4: \hspace{2em} E_{\text{init}} \leftarrow N(t) \times E_{o} \quad \text{where} \ t = 0
5: \hspace{2em} \text{Compute} \ \lambda, R_{q}, E_{\text{clustering}}, E_{q}, E_{\text{IDS}} \ \text{at} \ t = 0
6: \hspace{2em} \text{Compute arrival time for next clustering, query, and IDS events}
7: \hspace{2em} \textbf{while} \ E_{\text{init}} > E_{\text{threshold}} \textbf{do}
8: \hspace{3em} \text{ev} \leftarrow \text{next event}
9: \hspace{3em} \text{if} \ ev \text{is clustering event then}
10: \hspace{4em} E_{\text{init}} \leftarrow E_{\text{init}} - E_{\text{clustering}}
11: \hspace{3em} \text{else if} \ ev \text{is query event then}
12: \hspace{4em} \text{num}_{q} \leftarrow \text{num}_{q} + 1
13: \hspace{4em} E_{\text{init}} \leftarrow E_{\text{init}} - E_{q}
14: \hspace{4em} \text{if} \ \text{num}_{q} = 1 \text{then} \ /\!\!/ \text{first query}
15: \hspace{5em} r_{q, \text{muls}} \leftarrow r_{q, \text{muls}} \times R_{q}
16: \hspace{5em} \text{temp} \leftarrow \text{num}_{q} \times r_{q, \text{muls}}
17: \hspace{4em} \text{else} \ /\!\!/ \text{terminate previous query in MTTF calculation}
18: \hspace{5em} \text{tempMttf} \leftarrow \text{tempMttf} + \text{temp} \times (1 - R_{q})
19: \hspace{5em} r_{q, \text{muls}} \leftarrow r_{q, \text{muls}} \times R_{q}
20: \hspace{5em} \text{temp} \leftarrow \text{num}_{q} \times r_{q, \text{muls}}
21: \hspace{3em} \text{else} \ /\!\!/ \ ev \text{is an IDS event}
22: \hspace{4em} \text{Update distribution of good and bad nodes}
23: \hspace{4em} \text{Compute} \ \lambda, P_{f} / P_{fa}
24: \hspace{4em} \text{Update} \ E_{\text{IDS}}
25: \hspace{4em} E_{\text{init}} \leftarrow E_{\text{init}} - E_{\text{IDS}}
26: \hspace{4em} \text{Remove Bad caught and Good misidentified nodes}
27: \hspace{4em} \text{Compute} \ Q, c
28: \hspace{4em} \text{Update} \ R_{q}, E_{\text{clustering}}, E_{q}
29: \hspace{4em} \text{tempMttf} \leftarrow \text{tempMttf} + \text{temp}
30: \hspace{4em} \text{Mttf} \leftarrow \text{tempMttf}
31: \hspace{4em} N_{q} \leftarrow \text{num}_{q}
32: \hspace{3em} \text{if} \ \text{Mttf} > \text{optimalMttf} \text{then}
33: \hspace{4em} \text{optimalMttf} \leftarrow \text{Mttf}
34: \hspace{4em} \text{optimal} \ (m_{p}, m_{s}) \leftarrow (m_{p}, m_{s})
35: \hspace{3em} \text{return} \ \text{optimalMttf and optimal} \ (m_{p}, m_{s})

Fig. 3: Computational Procedure to Determine Optimal \((m_{p}, m_{s})\) for Maximizing MTTF.

In Fig. 4, we show MTTF vs. \((m_{p}, m_{s})\) as a result of applying Equation 1 to the query-based WSN. We compare three cases: (a) there are no malicious nodes and no intrusion detection (the top curve); (b) there are malicious nodes but there is no intrusion detection (the bottom curve); (c) there are malicious nodes and there is intrusion detection (the middle two curves). First of all, in each case we observe the existence of an optimal \((m_{p}, m_{s})\) value under which MTTF is maximized. Secondly, for the special case in which there are no malicious nodes (the top curve), the optimal \((m_{p}, m_{s})\) is \((3, 3)\). When there are malicious nodes, however, the optimal \((m_{p}, m_{s})\) value becomes \((7, 7)\) because using higher redundancy in multi-source multipath routing is necessary to cope with malicious nodes that perform bad-mouthing and packet dropping at-
tacks. Third, when intrusion detection is used (middle curves), there exists an optimal $m$ value (the number of voters) to maximize MTTF. In Fig. 4, $m=5$ yields a higher MTTF value than $m=3$ because in this scenario the attack rate is relatively high (once a week), so a higher number of voters is needed to cope with and detect bad nodes more effectively, resulting in a higher query success rate and thus a higher MTTF. It is also worth noting that the optimal $(m_p, m_s)$ in this scenario equals (3, 4) and (4, 4) for $m=5$ and 3 respectively.

Next we run the computational procedure to analyze the effect of $T_{comp}$, $m$ and $T_{IDS}$ on optimal $(m_p, m_s)$. Fig. 5 shows MTTF vs. $(m_p, m_s)$ with varying $T_{comp}$ and $m$ values. The left graph is for the case in which $T_{comp}$ is large while the right graph is for the case in which $T_{comp}$ is small. By comparing these two graphs, we observe a trend that as the capture rate increases (i.e., going from the left graph to the right graph), the optimal $(m_p, m_s)$ redundancy level increases. For instance, when the capture rate increases from once in three weeks ($T_{comp} = 3$ weeks) to once a week ($T_{comp} = 1$ 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 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.
Another trend exhibited in Fig. 5 is that 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 attacks by bad nodes. Table 3 below summarizes the effect of \(T_{comp}\) and \(m\) on optimal \((m_p, m_s)\) values under which MTTF is maximized.

**TABLE 3**

<table>
<thead>
<tr>
<th>(T_{comp})</th>
<th>(m=3)</th>
<th>(m=5)</th>
<th>(m=7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 days</td>
<td>5,7</td>
<td>4,6</td>
<td>4,5</td>
</tr>
<tr>
<td>1 week</td>
<td>4,4</td>
<td>3,4</td>
<td>3,3</td>
</tr>
<tr>
<td>3 weeks</td>
<td>3,3</td>
<td>3,3</td>
<td>3,3</td>
</tr>
</tbody>
</table>

Fig. 5 also reveals that there exists an optimal \(m\) value under which MTTF is maximized and the optimal \(m\) value increases as the capture rate increases. When the capture rate increases from once in three weeks \((T_{comp} = 3\text{ weeks})\) to once a week \((T_{comp} = 1\text{ week})\), the optimal \(m\) value goes from 3 (the top curve in the left graph of Fig. 5) to \(m = 7\) (the top curve in the right graph of Fig. 5). 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 4 summarizes the effect of \(T_{comp}\) on the optimal \(m\) value at which MTTF is maximized.
Next we analyze the effect of $T_{IDS}$ on MTTF. Fig. 6 and 7 show MTTF vs. $T_{IDS}$ with varying $m$ under low capture rate ($T_{comp} = 3$ weeks) and high capture rate ($T_{comp} = 1$ 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. 6 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 Fig. 6 and 7 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 4 earlier.

Finally we examine the sensitivity of the optimal $T_{IDS}$ to the capture rate. Fig. 8 shows MTTF vs. $T_{IDS}$ with varying $T_{comp}$ values. It exhibits the trend that as the capture rate increases (a smaller $T_{comp}$ value), the optimal $T_{IDS}$ at which MTTF is maximized must decrease to cope with malicious attacks. For example, in Fig. 8 the optimal $T_{IDS}$ is 20 hours when $T_{comp} = 4$ weeks and reduces to 6 hours when $T_{comp} = 4$ days. The reason is that when the capture rate is low and hence the malicious node population is low, the negative effects of wasting energy for IDS execution (through evicting falsely identified nodes and executing the voting mechanism) outweighs the gain in the query success probability, so the system is better off by executing intrusion detection less often. On the other hand, when the capture rate is high and the malicious node population is high, the gain in the query success probability because of evicting malicious nodes often outweighs the energy wasted because of frequent IDS execution, so the system is better off by executing intrusion detection often. Table 5 below summarizes the effect of $T_{comp}$ and $m$ on the optimal $T_{IDS}$ value at which MTTF is maximized.
In this paper we provided a solution to the issue of secure and reliable multisource multipath routing in 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. To apply the results, a system can perform a table lookup choosing 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. A future research direction is to consider a more extensive list of malicious attacks each with different implications on energy, security and reliability, and investigate intrusion detection protocols to detect and react to these attacks. Another direction is to extend the analysis to heterogeneous WSN environments with distinct types of nodes (e.g., cluster heads vs. sensors) having vastly different energy, intrusion detection, and security and reliability capability. The challenge lies in an effective solution technique to solve the probability model describing the behaviors of a large number of nodes so we may know the status of the heterogeneous WSN in response to dynamic query and attack events.

### References


