2013

Identifying packet droppers in sensor networks via report analysis

Jin-Sook Kim
Iowa State University

Identifying packet droppers in sensor networks via report analysis

by

Jin-Sook Kim

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:

Wensheng Zhang, Major Professor

Daji Qiao

Lu Ruan

Iowa State University

Ames, Iowa

2013

TABLE OF CONTENTS

LIST OF FIGURES iv

ACKNOWLEDGEMENTS vi

ABSTRACT vii

CHAPTER 1. OVERVIEW 1

1.1 Motivation and contribution 1

1.2 Organization of this thesis 2

CHAPTER 2. REVIEW OF LITERATURE 3

CHAPTER 3. METHODS AND PROCEDURES 5

3.1 Preliminaries 5

3.1.1 Network model 5

3.1.2 Security assumptions and attack model 5

3.2 Proposed scheme 6

3.2.1 Information of tree topology 7

3.2.2 Packet encryption and decryption 7

3.2.3 Monitoring report transmission and analysis 10

3.3 Catching packet droppers and modifiers in wireless sensor networks 23

CHAPTER 4. RESULTS 35

4.1 Impact of network density on failed-detection rate 35

4.2 Impact of natural dropping rate and mis-overhearing on failed-detection rate . 35

4.3 Impact of percentage of droppers on failed-detection rate 37

4.4 Impact of detection rounds on failed-detection rate 37

CHAPTER 5. SUMMARY AND DISCUSSION 39

REFERENCES 40

LIST OF FIGURES

Figure 3.1 Diverse packet forwarding paths 6

Figure 3.2 Random numbers for parents 7

Figure 3.3 Topology for a source 8

Figure 3.4 Encryption by a source 9

Figure 3.5 Encryption in packet forwarding 9

Figure 3.6 A Path for packet generation and forwarding 9

Figure 3.7 Packet at the base station 11

Figure 3.8 A Path for travel of a packet 11

Figure 3.9 Example of EF, GG, R, F and SF 12

Figure 3.10 Example of correct R 13

Figure 3.11 Example of correct R and F 14

Figure 3.12 Example of correct F and SF 15

Figure 3.13 Example of wrong F and SF 15

Figure 3.14 Example of correct F, SF and GG 16

Figure 3.15 A child and its parent do not drop packets 16

Figure 3.16 A child drops packets but its parent does not 17

Figure 3.17 A parent drops packets but its child does not 17

Figure 3.18 Example of correct R, GG and F 18

Figure 3.19 Example of correct R of a parent and F for its children 19

Figure 3.20 Relation for R of a parent, F and GG of its children 20

Figure 3.21 A child has correct F but a parent has wrong R 20

Figure 3.22 Grand-relation for R, F and SF 21

Figure 3.23 Example of correct F, SF, R and GG 22

Figure 3.24 Example of correct R, F and SF between a node and its grandchildren 22

Figure 3.25 Example of EF and correct F 23

Figure 3.26 Example of compromised F 23

Figure 4.1 Failed-detection rate vs. network density 36

Figure 4.2 Failed-detection rate vs. natural dropping rate and mis-overhearing rate 36

Figure 4.3 Failed-detection rate vs. percentage of droppers 37

Figure 4.4 Failed-detection rate vs. number of detection rounds 38

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Zhang for his guidance, patience and support throughout this research and the writing of this thesis. His insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education. I would also like to thank my committee members for their efforts and contributions to this work: Dr. Daji Qiao and Dr. Lu Ruan.

ABSTRACT

In an unattended sensor network, sensor nodes can be compromised. Based on compromised nodes, the adversary can launch various attacks and packet dropping is one of the easiest attacks. Many schemes have been proposed to mitigate the packet dropping attack, but few can effectively and efficiently identify the droppers. In this thesis, we propose a simple yet effective scheme to identify packet droppers. This scheme only requires sending and forwarding nodes to report their observations to the base station, and the base station can analyze the reports, identify inconsistencies in the reports, and then locate the droppers. All nodes are organized into a tree rooted at the base station and each node is required to report the number of packets it has received as well as the number and the composition of packets forwarded by its parent node on the tree, which it has overheard. Using rules we propose, the base station can analyze the received reports to check if there is inconsistency among the reports; if some inconsistencies are found, the base station can further infer the identities of packet droppers. The scheme can also tolerate erroneous reports, natural packet dropping and so on. A mark-based scheme is also proposed to identify packet dropper or modifier. Extensive simulations have been conducted to demonstrate the effectiveness of the scheme.

CHAPTER 1. OVERVIEW

1.1 Motivation and contribution

In wireless sensor networks, each sensor node monitors the environment, detects events of interest, produces data and collaborates in forwarding the data toward a base station, storage node, or querying user. Sensor networks are often deployed in unattended and hostile environments. Due to the lack of physical protection, sensor nodes are subject to node compromise. After compromising one or multiple sensor nodes, the adversary may launch various attacks to disrupt the in-network communication. Among these attacks, the easiest one may be *dropping packets*, i.e., compromised nodes drop the packets which are supposed to be forwarded.

To deal with packet dropping, a widely adopted countermeasure is multi-path forwarding [Budhaditya Deb et al. (2003); Thiemo Voigt et al. (2005); Suk-Bok Lee et al. (2006)], in which each packet is forwarded along multiple redundant paths and hence packet dropping in some but not all of these paths can be tolerated. This scheme introduces high extra communication overhead. Moreover, without identifying packet droppers, this countermeasure can not fully solve the problem because the compromised nodes can continue attacking the network without being caught.

Some local monitoring schemes [Issa Khalil et al. (2005); Saurabh et al. (2004)] have also been proposed for identifying packet droppers. In these schemes, all or some of nodes monitor their neighboring nodes and then these monitoring nodes make decision together without intervention of the base station. The advantage of this kind of schemes is that compromised nodes can be found locally. However, this local decision can lead to other attacks such as bad mouth attack [Saurabh et al. (2004)]. Moreover, the monitoring nodes should spend much more resources than any other nodes.

In this thesis, we propose a new scheme in which each node runs a simple monitoring module and the base station can find out the compromised nodes without incurring other attacks as described above. Specially, the scheme organizes all nodes into a tree rooted at the base station. Each node is required to report the number of packets it has received as well as the number and the composition of packets forwarded by its parent node on the tree, which it has overheard. Using rules we propose, the base station can analyze the received reports to check if there is inconsistency among the reports; if some inconsistencies are found, the base station can further infer the identities of packet droppers. For example, our proposed mark-based scheme may be used to identify packet droppers or modifiers. Extensive simulations have been conducted using ns2 simulator. The results verify that the proposed scheme is effective, and can tolerate erroneous reports, natural packet dropping and so on.

1.2 Organization of this thesis

The rest of the thesis is organized as follows: Related works are reviewed in the chapter 2 and the proposed scheme is explained in the chapter 3. The section 3.1 describes the network model, security assumptions and attack model. The way the scheme works is explained in the section 3.2 and and the section 3.3 explains briefly our proposed mark-based scheme that can be used together with the scheme proposed in this thesis work to identify packet droppers/modifiers. The simulation results are shown in the chapter 4 and finally we conclude in the chapter 5.

CHAPTER 2. REVIEW OF LITERATURE

Dropping packets may be the easiest attackt that can be launched by compromised nodes and unfortunately it affects the perfmance of sensor network significantly. To deal with the packet dropping attacks launched by passive attackers who do not participate in forwarding packets to save their battery selfishly, Afrand Agah et al. (2006) propose a scheme based on game theory. An IDS module running on a base station plays an important role in managing a reputation value for each node. The nodes which forward packets get high reputation values and the nodes not doing so get low reputation values. As total reputation values accumulate, the nodes which get negative reputation values are considered as passive attackers and a path consisting of less number of attackers is chosen to be a winning path. As the major communication overhead of this scheme, a base station should be informed by each destination of packet arrivals and the reputation value of a node should be periodically broadcast to the neighbors of the node to keep their knowledge fresh.

To maintain data reliability in the presence of dropping attacks, the scheme proposed by Budhaditya Deb et al. (2003) sends multiple copies of each packet along multiple paths from source. The degree of redundancy introduced, is controlled according to the desired reliability, the local channel error conditions, neighborhood information available and so on. Thiemo Voigt et al. (2005) also present a simple routing algorithm for on-demand construction of multiple non-interfering paths in a wireless sensor network using geographic routing. Its main idea is that a node chooses a node as next hop only if the node's distance from the straight line between the source and the sink of the data transfer is at least the transmission range. Besides mitigating data unreliability using multiple paths, the schemes with multiple paths are also used to detect and isolate the compromised nodes which try to inject inconsistent routing information from the network using a neighbor report system proposed by Suk-Bok Lee et

al. (2006). In the neighbor report system, the route advertisement of a node is verified by its surrounding neighbor nodes so that the suspected node is reported to a base station and is excluded from the network. These multi-path schemes can improve the arrival rate of packets at the destinations and make sensor networks more secure. However, it can also incur high extra communication overhead.

To isolate compromised nodes that have been identified, there is watchdog mechanism [Saurabh et al. (2004)] in which every node monitors its neighbors and decides whether they are good or bad based on the information obtained from monitoring. Here, the decision from only one node is not trustful and thus multiple monitoring nodes are required to exchange their monitoring information or their opinions on whether a node is good or bad, with each other. However, this opinion exchange incurs the so-called bad mouth attack. That is, compromised nodes may speak ill of a victim to make other nodes believe that the victim is compromised. Additionally, this monitoring mechanism may cause extra overhead in terms of storage and communication. Fortunately, our proposed scheme does not make each node spend much storage - a base station needs only three kinds of monitoring reports from sensor nodes, the communication overhead is low and monitoring reports can be piggybacked in sensory data reports.

Unlike the aforementioned schemes in which every node monitors its neighbors, only some nodes take the responsibility to monitor other nodes as proposed by Issa Khalil et al. (2005) and A. P. et al. (2005). In this work, Issa Khalil et al. (2005) uses local monitoring to detect control traffic misbehavior and local response to diagnose and isolate the suspect nodes. For a link (i, j), the sender i is a guard node for node j. Information for each packet sent from X to A is saved in a $watch\ buffer$ at each guard for a time τ. The work proposed by A. P. et al. (2005) is based on the inference of the network behavior obtained from the analysis of events detected by a monitor node, i.e, the node that implements the IDS system. The monitoring node keeps its radio in a promiscuous mode, storing relevant information and processing it according to selected rules. However, such schemes are still vulnerable to bad mouth attacks and introduce the communication overhead for announcing compromised nodes.

CHAPTER 3. METHODS AND PROCEDURES

3.1 Preliminaries

3.1.1 Network model

We consider a sensor network composed of a sink (base station) and regular sensor nodes. The network is deployed in an unattended environment for monitoring purpose. All the regular sensor nodes and a base station form a tree rooted at the base station. Each node in this tree can have multiple parents if it has a number of neighbors. Whenever a sensor node detects an event, it samples its environment at a certain rate and reports sensory data at the same rate. The rate is denoted as G. The sensory data is first sent to the parent node on the tree, which further forwards the data to its own parent, and so on and so forth. If a sensor node does not detect any event, it does not report any data. We assume that all sensor nodes and the base station are time synchronized, and thus they share the same timer reference. The time is divided into rounds, each of fixed length. As the round change, each node changes the parent for packet forwarding to give packet forwarding information on more paths in the tree to the base station. For example, Fig. 3.1 shows diverse packet forwarding paths. The packets from a source ***9*** can travel on the path which is constructed by the nodes ***7***, ***3*** and ***1*** in the round RO_i and in the next round RO_{i+1} the packets from the the source ***9*** can travel through the nodes ***8***, ***5*** and ***2***.

3.1.2 Security assumptions and attack model

We assume that every sensor node shares a unique pairwise key with the base station, which is known only by the node and the base station. With this key a source encrypts the packets which the source generates and the forwarding nodes for the packets also encrypts

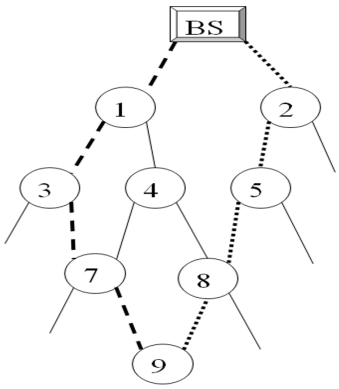

Figure 3.1 Diverse packet forwarding paths

the packets forwarded by the previous forwarding node or the source for the packets. This encryption prevents a selective dropping by compromised nodes for specific sources. For the decryption, each node reports its topology information after the construction of tree rooted at the base station. The base station cannot be compromised, but any regular sensor node can be compromised. Using compromised sensor nodes, the adversary can launch various kinds of attacks. In this thesis, we focus on the simplest dropping attack, i.e., a compromised node may drop some packets which are supposed to be forwarded. Besides dropping by the compromised nodes, we consider dropping by environment such as interference whose frequency is much less than that of dropping by compromised nodes.

3.2 Proposed scheme

The proposed dropper identification scheme includes two major components, the transmission/forwarding of packets, and the transmission/forwarding and analysis of monitoring reports. The first component ensures that packets cannot be dropped selectively based on their sources, and the second component enables collaborative monitoring of node behaviors and identification of packet droppers. In this section, these two components are described in detail.

3.2.1 Information of tree topology

After deployment of sensor nodes, a multi-parents tree is established. Each node comes to know its parents and children if it has numerous neighboring nodes and to give monitoring data in more diverse situations, each node changes its parent in each round. Here, we add one more data to lessen memory consumption for security. This is a random number r_{ij} for a node i to its parent, a node j. When a node i knows which nodes are its parents, it assigns a random number to each parent. In Fig. 3.2, a node ***4*** has the parents, a node ***1*** and a node ***2*** and it assigns a random number r_{41} to a node ***1*** and r_{42} to a node ***2***. With the real identities of the parents and children, the random numbers for the parents, the depth and the order of the parents for rounds are sent to the base station by each node.

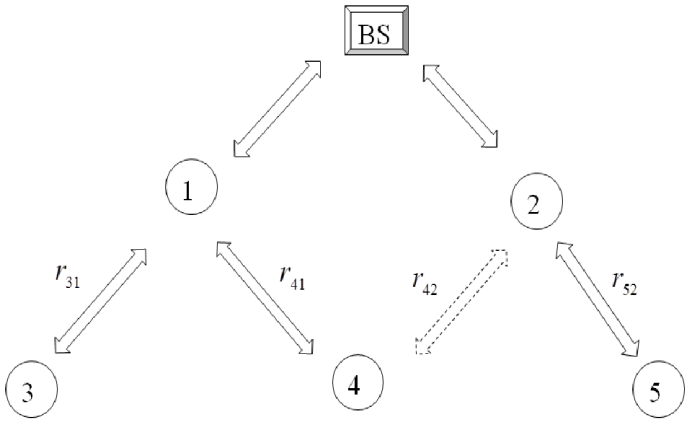

Figure 3.2 Random numbers for parents

3.2.2 Packet encryption and decryption

As described in the network model (Section 3.1.1), a sensor node reports sensory data at rate G if the node detects some event. The packet containing sensory data should be encrypted to hide the source of the packet, so as to prevent an adversary en-route node from selectively dropping packets from particular sources. Hence, in the following we introduce the approaches to encrypt packets by sensor nodes and to decrypt packets by the base station.

3.2.2.1 Packet encryption

Using the key which is installed before node deployment, a packet is encrypted in generation as well as forwarding to mitigate the infiltration of false data [Sencun Zhu et al. (2004); FanYe et al. (2007)]. However, the multiple encryption leads to more memory consumption because of multiple identifier addition. Our approach to lessen memory consumption is a random number which each node assigns to its parent. The random number requires less bits than the identifier of nodes since the maximum number of parents for each node, P_N is smaller than the total number of nodes.

Encryption in packet generation

When a source generates a packet, the identifier of the source goes with the sensory data and is encrypted to prevent selective dropping. Additionally, a packet contains a random number which the source assigns to its parent. Eq. (3.1) is the packet format which sources use.

$$\langle P_i, \langle r_{ij}, i, D \rangle_{k_i}, pad \rangle \tag{3.1}$$

Here, P_i is the identifier of source's parent which is the next hop destination. r_{ij} is a random number which a source i assigns to its parent, a node j and i is the identifier of a source. k_i is the pairwise key which a source i shares with the base station and $\langle A \rangle_{k_i}$ means that A is encrypted by a key k_i. pad is an arbitrary number which keeps packet size same and hence prevents packet dropping based on packet size. It is chopped away when a forwarding node adds its random number. In Fig. 3.3 where a circle is a node and the number on an arrow is the random number which the node in the tail of the arrow assigns to the node in the head of the arrow, a source ***3*** assigns a random number r_{34} to a node ***4*** and a source ***3*** generates a packet like in Fig. 3.4

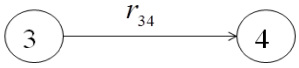

Figure 3.3 Topology for a source

Encryption in packet forwarding

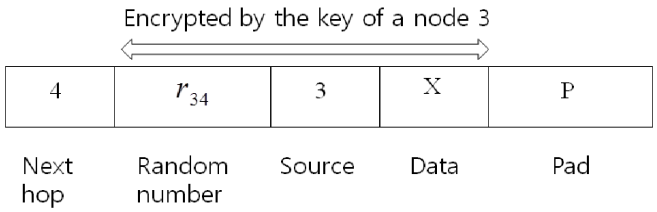

Figure 3.4 Encryption by a source

When a node v receives a packet with the format of Eq. (3.1)(described as $\langle v, m \rangle$ more briefly), v shifts m to the right by $log(P_N)$ bits and its random number r_{vu} is added. The result is encrypted by v's pairwise key. For example, Fig. 3.4 shows a generated packet by a source ***3*** and Fig. 3.5 shows encryption in packet forwarding. In Fig. 3.6, there is a path through which a generated packet travels. A source ***3*** assigns a random number r_{34} to a node ***4*** and a node ***4*** also assigns a random number r_{42} to a node ***2***. After a node ***4*** receives a packet from a source ***3***, it shifts the packet to the right by $log(P_N)$ bits and a random number r_{42} is added. Then the packet is encrypted again with the pairwise key of a node ***4*** and the real identity of a node ***2*** is added in the head of the packet.

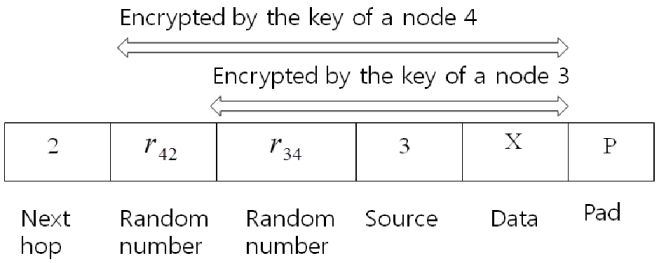

Figure 3.5 Encryption in packet forwarding

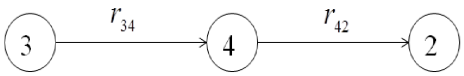

Figure 3.6 A Path for packet generation and forwarding

With a changed packet, v sends the received one for monitoring by its children like in Eq. (3.2). If v forwards only changed one, its children can not know whether v forwards the packet which they sent to v or not. The next forwarding node discards the first part like $\langle v, m \rangle$

in Eq. (3.2) and changes the second part.

$$\langle v, m \rangle, \langle P_v, m' \rangle \tag{3.2}$$

For the overhead from the addition of random numbers, it depends on the maximum number of parents, P_N which requires at most $log(P_N)$ bits and the number of hops. If P_N is 8 and the tree height is 20 hops (allowing the sensor network to cover about 2000m $\times$ 2000m area if using MicaZ motes [Z. Yu et al. (2006)]), the number of extra bits introduced is 60 bits or 8 bytes, which is not a big overhead especially when considering that, the IEEE 802.15.4 standard (followed by most of current sensor node products) allows packet length to be up to 127 bytes [Chuang Wang et al. (2009)].

3.2.2.2 Packet decryption

Due to multiple encryption, decryption also has several steps. In each decryption step, only pairwise keys from the children of a current node become the candidate keys for decryption. If any of the candidate keys brings the corresponding random number, the decryption process continues until the sensory data is decrypted. If not, the packet is rejected as false injected data. For instance, Fig. 3.7 shows one packet example which arrives at the base station whose id is ***0***. The path which the packet has traveled is shown in Fig. 3.8. When the base station receives the packet, it needs to decrypt only $\langle r_{kl}, r_{jk}, r_{ij}, i, D \rangle$. At first, if the base station decrypts it with the pairwise key of a node ***2***, finds the random numbver r_{20} which a node ***2*** assigns to the base station. Then, r_{20} is trimmed and the next decryption part is $\langle r_{42}, r_{34}, 3, X \rangle$. r_{42} can be decrypted by the pairwise key of a node ***4*** and this is the random number which is assigned to a node ***2*** by a node ***4***. Finally, a pairwise key of a node ***3*** decrypts $\langle r_{34}, 3, X \rangle$. The random number r_{34} matches the random number which a node ***3*** assigns to a node ***4***.

3.2.3 Monitoring report transmission and analysis

In our scheme, the additional role of each node is only to monitor its parent's behavior and report the monitoring results. The simple monitoring reports have the relation with each

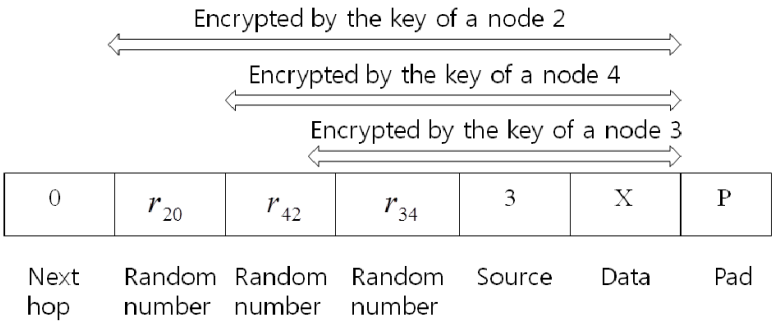

Figure 3.7 Packet at the base station

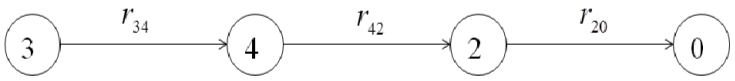

Figure 3.8 A Path for travel of a packet

other and the report of the base station for its received packets. This relation contributes to the development of rules to find compromised nodes.

3.2.3.1 Constitution of report

For the report of the base station, it counts the number of packets that a source i generates (denoted as SG_i) among the received packets and with receiving of any packet from a source i, the base station considers that the source i generates actually GG_i packets which a source should generate during a round according to the sampling rate G. The base station also counts the number of packets that a sensor node i forwards (denoted as EF_i). In Fig. 3.9, $EF_3 = 150$.

For the report of each sensor node, a sensor node i counts the number of packets that it has received during a round (denoted as R_i). In Fig. 3.9, $R_3 = 150$. In addition, a sensor node i monitors its parent, a sensor node j, during a round and tells how many packets its parent has forwarded for its own packet(denoted as F_{ij}). In Fig. 3.9, $F_{13} = 50$ and $F_{23} = 100$. For the packets about which it can not recognize the source and forwarding nodes (denoted as SF_{ij}), SF_{ij} includes the packets from the siblings and the parent. In Fig. 3.9, $SF_{13} = 150$ and $SF_{23} = 100$. To summarize, every node i should report the monitoring result as $\langle R_i, F_{ij}, SF_{ij} \rangle$ to the base station after every round. These reports should also be encrypted, and the procedures

for encrypting, sending, forwarding and decrypting these reports are same as those for sensory packets, as described in the above section.

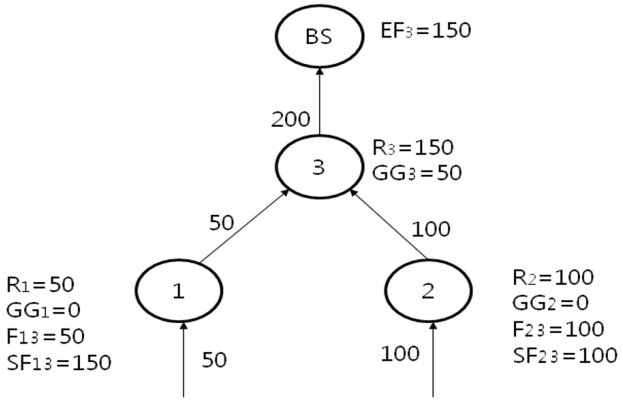

Figure 3.9 Example of EF, GG, R, F and SF

3.2.3.2 Development of rules

The relation between the report of the base station and each node contributes to the development of the following rules to find compromised nodes. They are organized into basic rules and extended rules.

We use $\doteq$ instead of $=$ since we consider the dropping by environment such as interference and weakness of monitoring system.

Basic Rules

The basic rules hold regardless of the number of compromised neighboring nodes since they are based on consistent relation for the report of the base station and each node.

Rule 1: This rule is based on the relation between a node and its children for R. R of a node is the number of packets which the node has received during a round.

Since packets which are received as well as generated by a node must reach the parent of the node, the parent's R is correct if its children have the correct R and the summation of (R + GG) for its children is close to parent's R like in Fig. 3.10.

The formal expression of this is Eq. 3.3 where CS_j is the set for the children of a node j and the function symbol, $Correct(A, B, ...)$ means that the arguments A, B, ... are correct.

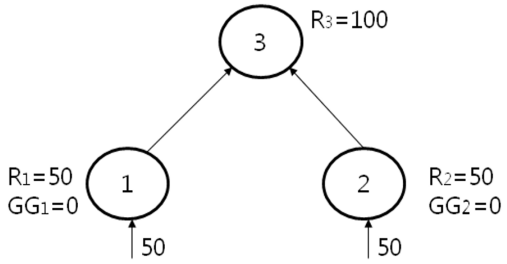

Figure 3.10 Example of correct R

$$[\forall i \in CS_j \quad Correct(R_i)] \wedge \sum_{\forall i \in CS_j} (R_i + GG_i) \doteq R_j \Rightarrow Correct(R_j) \tag{3.3}$$

To use this rule more efficiently, it is applied from the bottom of a tree to top; that is, from sources to the base station. This is because R and GG of sources are already correct. R of sources is zero and GG is calculated with sampling rate.

Rule 2: This rule is based on the relation between R and F for the nodes on a path through which most packets arrive at the base station. If the base station receives most packets from a source, the nodes on the path for the source can be considered as non-droppers. However, they can falsify their report, R or F. For R, it must be close to EF for the nodes on the path since before being forwarded by a node (corresponds to EF), packets should be received by the node (corresponds to R). Eq. 3.4 expresses this where P_s is a path whose source is a node s and $Drop\_Path(A)$ means that a path A includes at least one dropper. $\neg f()$ represents the negation of $f()$

With the correct R, we can find correct F. F is the number of packets which a node sends to its parent and its parent forwards. To understand it more clearly, we show an example in Fig. 3.11 where $F_{12} = 50$ since a node ***1*** sends 50 packets and a node ***2*** forwards all of them. Since a node sends the generated and received packets by itself, it is true that F of a node is close to $(R + GG)$ of the node. In Fig. 3.11, $F_{12} = R_1 + GG_1$. Eq. 3.5 expresses this.

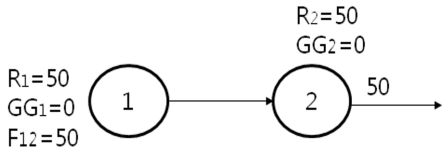

Figure 3.11 Example of correct R and F

$$[\neg Drop\_Path(P_s)] \wedge [i \in P_s \quad R_i \doteq EF_i] \Rightarrow Correct(R_i) \tag{3.4}$$

$$\begin{aligned}[\neg Drop\_Path(P_s)] \wedge [i \in P_s, i \in CS_j \quad & Correct(R_i) \\ \wedge \quad F_{ij} \doteq R_i + GG_i] & \Rightarrow Correct(F_{ij})\end{aligned} \tag{3.5}$$

Rule 3: This rule is based on the relation among R, F and SF. SF of a node is the number of packets which its siblings send to its parent and the parent forwards. It also includes the packets which the parent of the node generates. To understand it more clearly, we show an example in Fig. 3.12. The number of packets on the link from a node ***3*** is 150. 50 of them are from a node ***1***, 100 of them are from a node ***2*** and a node ***3*** does not generate any packet. Hence, $SF_{13} = 100$ and $SF_{23} = 50$. Additionally, $F_{13} + SF_{13} = R_3 + GG_3$ and $F_{23} + SF_{23} = R_3 + GG_3$, which means that the sum of F and SF of a node is close to the number of packets which its parent receives and generates. The reason is that the packets which the parent of a node receives and generates (corresponds to R+GG) should be forwarded and they are monitored by its children (corresponds to F+SF) if the parent of the node does not drop packets. However, there is one case we must consider. A compromised node can falsify its F and SF such that the sum of them is still close to (R+GG) of its parent. For instance, a node ***1*** is compromised in Fig. 3.13 and changes F_{13} to 25 and SF_{13} to 125. However, (F_{13}+SF_{13}) is still equal to (R_3+GG_3). Hence, we use this rule when either F or SF is correct. Eq. 3.6 is the formal expression of this rule where $Drop(j)$ means that a node j drops packets.

$$\begin{aligned}\exists i \in CS_j \quad & [\neg Drop(j) \wedge Correct(R_j)] \\ & \wedge [R_j + GG_j \doteq F_{ij} + SF_{ij}]\end{aligned}$$

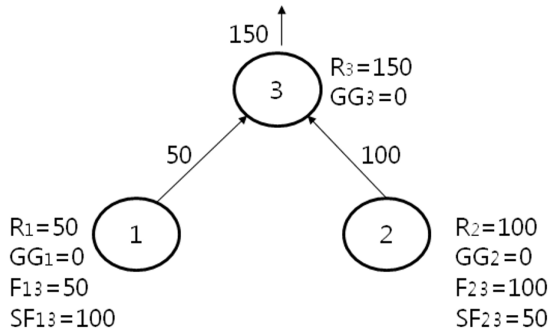

Figure 3.12 Example of correct F and SF

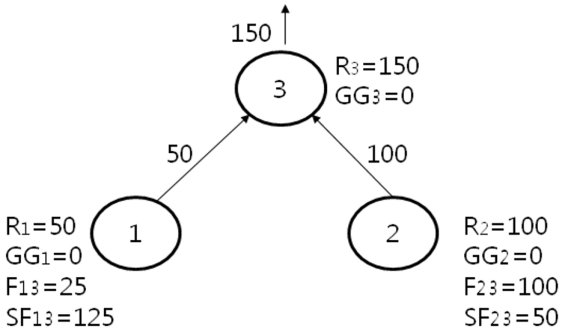

Figure 3.13 Example of wrong F and SF

$$\begin{aligned} &\wedge[Correct(F_{ij}) \vee Correct(SF_{ij})] \\ &\Rightarrow [Correct(F_{ij}) \Rightarrow Correct(SF_{ij}) \\ &\vee Correct(SF_{ij}) \Rightarrow Correct(F_{ij})] \end{aligned} \tag{3.6}$$

Rule 4: This rule is based on the relation between F and SF. SF is the monitoring result for the packets from the siblings of a node and its parent and F is the monitoring result for the packet from itself. According to this definition, SF of a node includes the packets of its siblings (corresponds to F of its siblings) and its parent (corresponds to GG of its parent). Let's look at Fig. 3.14 where $SF_{13} \doteq F_{23} + GG_3$ and $SF_{23} \doteq F_{13} + GG_3$. Thus, if F for the siblings of a node is correct and SF of the node is close to the sum of F for its siblings and GG of its parent, SF of the node is correct. Eq. 3.7 expresses this rule where SS_i is the set for the siblings of a node i.

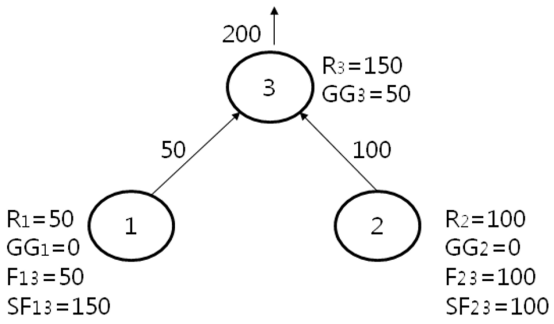

Figure 3.14 Example of correct F, SF and GG

$$\exists i \in CS_j \quad [\forall k \in SS_i \quad Correct(F_{kj})] \\ \wedge [SF_{ij} \doteq \sum_{\forall k \in SS_i} F_{kj} + GG_j] \Rightarrow Correct(SF_{ij}) \tag{3.7}$$

Rule 5: Based on the relation between R and F, a node can be verified as a non-dropper. If a node does not drop packets, it forwards all the received packet and this forwarding is monitored by its child. If the child also does not drop packets, (R+GG) of the child should be close to F of the child since the child sends its generated and received packets to the node and the node forwards them with being monitored by the child. For example, in Fig. 3.15, a node ***1*** and ***2*** do not drop any packet and hence $R_1 + GG_1 = F_{12}$. However, if a node does not drop packets but its child does, this does not hold. In Fig. 3.16, a node ***2*** does not drop any packet but a node ***1*** drops 50 packets. Therefore, $R_1 + GG_1 \neq F_{12}$ even if R_1 and F_{12} are correct. In sum, if a node does not drop packets and $(R + GG)$ for the child of the node is close to F for the child of the node, the child of the node dose not drop packets. Eq. 3.8 expresses this.

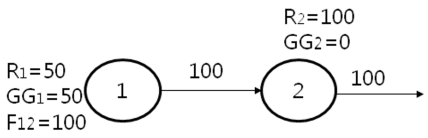

Figure 3.15 A child and its parent do not drop packets

$$\exists i \in CS_j \quad \neg Drop(j) \wedge Correct(R_i, F_{ij})$$

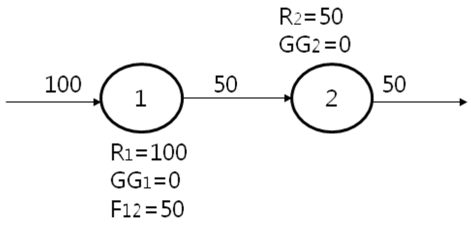

Figure 3.16 A child drops packets but its parent does not

$$\wedge[R_i + GG_i \doteq F_{ij}] \Rightarrow \neg Drop(i) \tag{3.8}$$

Rule 6: Like in Rule 5, this rule is also based on the relation between R and F to verify a node as a non-dropper. If a node and its child do not drop packets, (R+GG) of the child is close to F of the child for the correct R and F. However, if the node drops packets, (R+GG) of the child is not close to F of the child. For instance, in Fig. 3.17, a node ***2*** drops 50 packets but a node ***1*** does not drop any packet. Hence, $R_1 + GG_1 \neq F_{12}$.

In sum, if the child of a node does not drop packets and (R+GG) of the child is close to F of the child for correct R and F, the node does not drop packets. Eq. 3.9 is the formal expression of this.

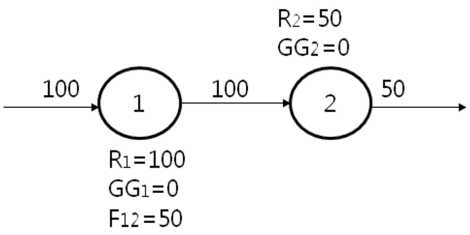

Figure 3.17 A parent drops packets but its child does not

$$\begin{aligned} \exists i \in CS_j \quad & \neg Drop(i) \wedge Correct(R_i, F_{ij}) \\ & \wedge[R_i + GG_i \doteq F_{ij}] \Rightarrow \neg Drop(j) \end{aligned} \tag{3.9}$$

Rule 7: Based on the report from all children, it is verified whether a parent drops packets or not. If the children of a node have correct R, the sum of (R+GG) for its children means

the number of packets which the children of the node receive and generate. Similarly, if the children of a node have correct F, the sum of F for its children means the number of packets which the node forwards. These two sums must be close to each other if the node does not drop packets. For example, in Fig. 3.18, $R_1 + GG_1 + R_2 + GG_2 = F_{13} + F_{23}$. To summarize, with correct R and F for the children of a node, if the sum of $(R + GG)$ is close to the sum of F for the children of the node, it is true that the node does not drop packets. Eq 3.10 is the formal expression of this rule.

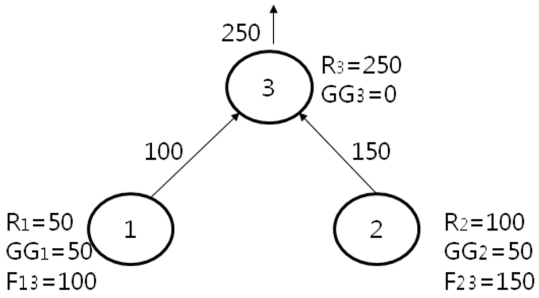

Figure 3.18 Example of correct R, GG and F

$$\wedge \sum_{\forall i \in CS_j} (R_i + GG_i) \doteq \sum_{\forall i \in CS_j} F_{ij} \Rightarrow \neg Drop(j) \qquad \forall i \in CS_j \quad Correct(R_i, F_{ij}) \tag{3.10}$$

Rule 8: This rule is based on the relation between R of a node and F of its children for the verification of a node as a non-dropper. As explained in Rule 7, if the children of a node have correct F, the sum of them means the number of packets which the node forwards. Here, before being forwarded by a node, the packets should be received by the node. Thus, correct R of a node must be close to the sum of F for its children if the node does not drop packets. For example, in Fig. 3.19, $R_3 = F_{13} + F_{23}$. In sum, with correct R of a node and correct F for its children, if R of the node is close to the sum of F for its children, it is true that the node does not drop packets. Eq. 3.11 expresses this.

$$[\forall i \in CS_j \quad Correct(F_{ij})] \wedge Correct(R_j)$$

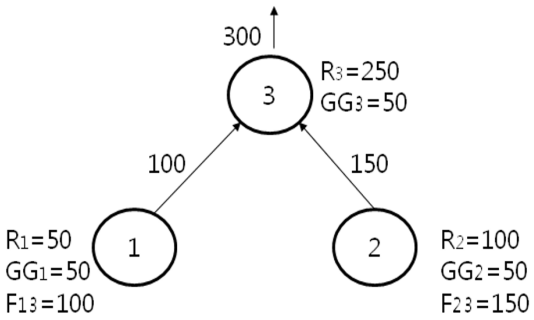

Figure 3.19 Example of correct R of a parent and F for its children

$$\wedge[R_j \doteq \sum_{\forall i \in CS_j} F_{ij}] \Rightarrow \neg Drop(j) \tag{3.11}$$

Extended rules

Until now, we explain the rules which always hold even if all the neighborings of a node are compromised. However, the ability to find compromised nodes decreases as the number of compromised nodes increases with these rules due to strict relation checking. Thus, we extend the basic rules to the rules which hold unless a node and all its neighboring nodes are compromised at the same time, which is a rare case.

Extended rule 1: As explained in Rule 7 and 8, the summation of F for the children of a node represents the number of packets which the node forwards and it is close to R of the node. Hence, if F for the children of a node is correct and its sum is close to R of the node, R of the node might be correct like in Fig. 3.20. However, we still have to consider the case that R of the node is falsified like in Fig. 3.21. A node ***2*** receives 100 packets but forwards only 50 packets. F_{12} is correct and $F_{12} = R_2$. However, R_2 is not correct.

To overcome this case, we consider R of a node as well as the children of the node. If R of a node is also close to the sum of (R+GG) for the children of the node, R of the node is correct unless all the children of the node are compromised to hold this. Eq. 3.12 is the formal expression of this.

$$[\forall i \in CS_j \quad Correct(F_{ij})] \wedge [R_j \doteq \sum_{\forall i \in CS_j} F_{ij}]$$

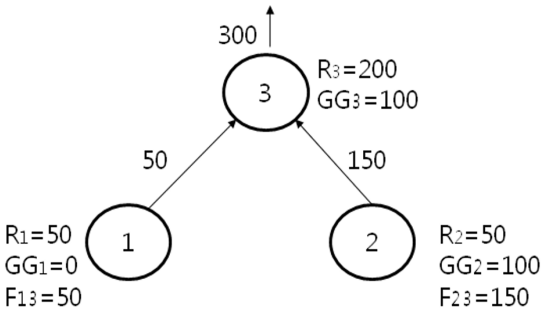

Figure 3.20 Relation for R of a parent, F and GG of its children

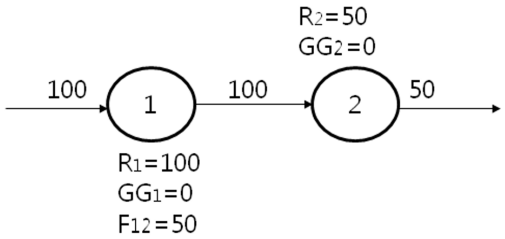

Figure 3.21 A child has correct F but a parent has wrong R

$$\wedge[R_j \doteq \sum_{\forall i \in CS_j} (R_i + GG_i)] \Rightarrow Correct(R_j) \tag{3.12}$$

Extended rule 2: This rule is based on the relation between a node and its grandchildren. F of a node means the number of packets which its parent receives from the node and forwards. They are also received by its grandparent. Therefore, R of a node includes the packets of F for its grandchildren as well as GG of its children (corresponds to the generated packets by its children). For example, in Fig. 3.22, $F_{15} + F_{25} + GG_5 + F_{36} + F_{46} + GG_6 = R_7$. Hence, if R of a node is close to the sum of F for its grandchildren and GG of its children together, there is some possibility that R of the node and F of its grandchildren are correct unless the node and all its grandchildren are compromised. Additionally, if R for each child of a node is close to the sum of F of its grandchildren for each child, the possibility becomes higher since this means that each child of the node is more likely to forward all the packets which it receives. In Fig. 3.22, $F_{15} + F_{25} = R_5$ and $F_{36} + F_{46} = R_6$. To summarize, if the sum of F for its grandchildren and GG of its children are close together to R of the node and the sum of F of

its grandchildren for each child is close to R of each child, R of the node, R of its children and F of its grandchildren are correct unless the node, all its children and all its grandchildren are compromised. Eq. 3.13 expresses this rule.

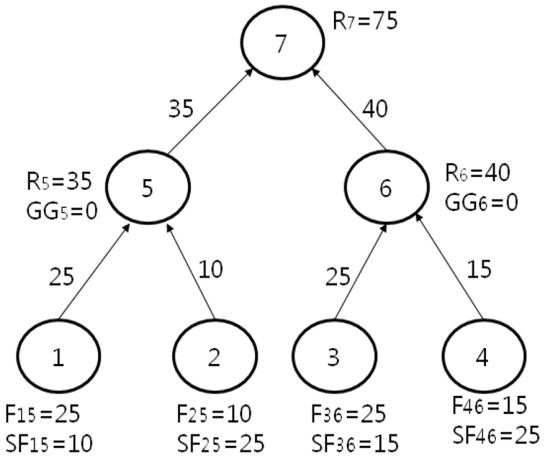

Figure 3.22 Grand-relation for R, F and SF

$$\begin{aligned} i \in CS_j, j \in CS_k \quad & [R_k \doteq \sum_{\forall i,j} (F_{ij} + GG_j)] \\ \wedge [\forall j \in CS_k \quad & R_j \doteq \sum_{\forall i} F_{ij}] \\ \Rightarrow \forall i,j,k \quad & Correct(R_k, R_j, F_{ij}) \end{aligned} \tag{3.13}$$

Extended rule 3: This rule also uses grand-relation. As explained in Rule 3, $(R + GG)$ of a node is close to $(F+SF)$ of its child. In Fig. 3.23, $F_{12} + SF_{12} = R_2 + GG_2$. This can be extended to between a node and some of its grandchildren. R of a node is close to the sum of $(F+SF)$ for a grandchild from each child. For example, in Fig. 3.24, $R_3 = F_{12} + SF_{12}$. Hence, if these two relations hold among a node, its children and its grandchildren, we can say that R of the node, R of its children and F or SF for some of its grandchildren are correct. Eq. 3.14 is the formal expression of this.

$$\begin{aligned} i \in CS_j, j \in CS_k \quad & [R_k \doteq \sum_{\exists i, \forall j} (F_{ij} + SF_{ij})] \\ \wedge [\forall j \in CS_k \quad & (R_j + GG_j) \doteq \sum_{\exists i} (F_{ij} + SF_{ij})] \end{aligned}$$

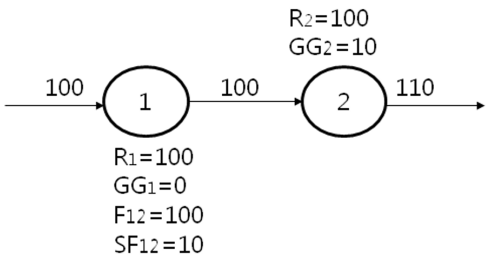

Figure 3.23 Example of correct F, SF, R and GG

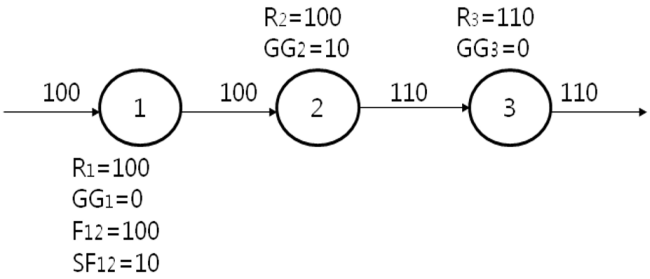

Figure 3.24 Example of correct R, F and SF between a node and its grandchildren

$$\Rightarrow \exists i, \forall j, k \quad Correct(R_k, R_j)$$
$$\wedge[Correct(F_{ij}) \Rightarrow Correct(SF_{ij}) \vee Correct(SF_{ij}) \Rightarrow Correct(F_{ij})] \tag{3.14}$$

Extended rule 4: This rule is based on the relation between EF, F and R. EF and F commonly count the number of packets which are forwarded by a node; EF by a node and F by a parent of a node. Thus, if EF of a node is close to the sum of F for its children, we can claim that F might be correct. For example, in Fig. 3.25, $EF_2 = 100$ and $EF_2 = F_{12}$.

However, there is one case that F is not correct even if EF of a node is close to the sum of F for its children. The children of a node are compromised to hold this. For example, in Fig. 3.26, $F_{13} = 25$ but a node ***1*** changes it into 15 and a node ***2*** also changes F_{23} into 20. To deal with this case, we use the relation between R of a node and the sum of F for its children. If R of the node is close to the sum of F for its children, F is more likely correct. These two hold together unless the node and all its children are compromised. Eq. 3.15 expresses this rule.

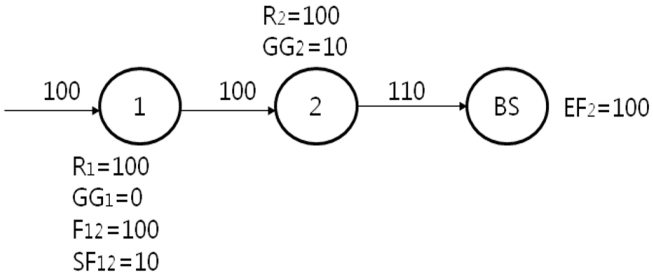

Figure 3.25 Example of EF and correct F

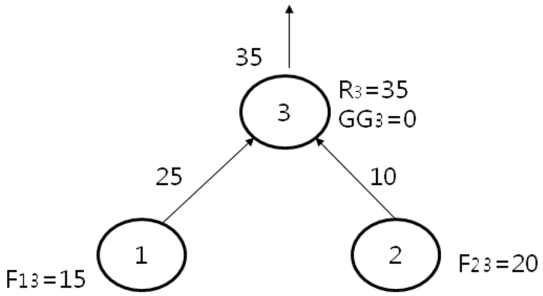

Figure 3.26 Example of compromised F

$$[EF_j \doteq \sum_{\forall i \in CS_j} F_{ij}] \wedge [R_j \doteq \sum_{\forall i \in CS_j} F_{ij}] \Rightarrow \forall i \in CS_j \quad Correct(F_{ij}) \tag{3.15}$$

In the following, the algorithm 1 shows how to apply the rules to find compromised nodes and the algorithms from 2 to 13 show how to use each rule as a function for each rule.

3.3 Catching packet droppers and modifiers in wireless sensor networks

Like the work presented in the previous subsections, this work is also to catch packet droppers and modifiers. In the scheme, the sensory data is transmitted along the tree rooted at a sink while each packet sender or forwarder adds a small number of extra bits, which is called packet marks, to the packet. Based on the packet marks, the sink can figure out the dropping rate associated with every sensor node, and then run our proposed node categorization algorithm to identify nodes that are droppers/modifiers for sure or suspicious droppers/modifiers.

Algorithm 1 Algorithm to find compromised nodes

Topology Input for $\forall i$

- CS, *the set of the children*
- SS, *the set of the siblings*
- $DEPTH$, *the depth*

Topology Input for a routing tree

- MAX_D, *the maximum depth of a routing tree*
- P, *the set of the nodes on the path for each source*

Report Input for $\forall i$

- SG, GG, EF, R, F, SF

Output

- *Identities of the nodes which have wrong* R, F, SF *values or are identified as droppers*

1: **repeat**
2: **repeat** {/*This loop is to identify wrong or correct reports. Note: only reports identified as correct can be used in identifying droppers/non-droppers*/}
3: Rule1(CS, $DEPTH$, MAX_D, GG, R)
4: ExtenedRule2(CS ,GG, R, F)
5: ExtenedRule1(CS ,GG, R, F)
6: ExtenedRule4(CS, EF, R, F, SF)
7: ExtenedRule3(CS ,GG, R, F, SF)
8: Rule4(CS, SS, GG, F, SF)
9: Rule2(CS, P, SG, GG, EF, R, F)
10: **until** Can not find more correct or wrong R, F and SF
11: /*The following rules are applied to identify droppers and non-droppers. Note: there could be nodes that cannot be determined as dropper or non-dropper, the set of which can be reduced with the following rules*/
12: Rule7(CS, R, F)
13: Rule8(CS, R, F)
14: Rule5(CS, R, F, ND)
15: Rule6(CS, R, F, ND)
16: Rule3(CS, GG, R, F, SF)
17: **until** Can not find more droppers or non-droppers

Algorithm 2 Rule1(CS, $DEPTH$, MAX_D, GG, R)

Paramenters

- *CS, the set of the children for each node*
- *$DEPTH$, the depth for each node*
- *MAX_D, the maximum depth of a routing tree*
- *GG for each node*
- *R for each node*

Return values

- *The identities of the nods which have correct or wrong R*

```
1: for every leaf node i do
2:    if R_i = 0 then
3:       R_i is correct
4:    else
5:       R_i is not correct
6:       R_i ← 0
7:    end if
8: end for
9: for D = MAX_D-1 to 1 do
10:   if DEPTH_j = D then
11:      if R_i is correct for ∀i ∈ CS_j then
12:         if R_j ≐ Σ_{∀i∈CS_j}(R_i + GG_i) then
13:            R_j is correct
14:         end if
15:      end if
16:   end if
17: end for
```

Algorithm 3 Rule2(CS, P, SG, GG, EF, R, F)

Parameters

- *CS, the set of the children for each node*
- *P, the set of the nodes on the path for each source*
- *SG for each node*
- *GG for each node*
- *EF for each node*
- *R for each node*
- *F for each node*

Return values

- *The identities of the nods which have correct or wrong F*

```
1: for every source s do
2:   if SG_s > 0 and SG_s ≐ GG_s then
3:     i does not drop packets for ∀i ∈ P_s
4:     for all i ∈ P_s do
5:       if R_i ≐ EF_i then
6:         R_i is correct
7:         if F_ij ≐ R_i + GG_i then
8:           F_ij are correct
9:         end if
10:      end if
11:    end for
12:  end if
13: end for
```

Algorithm 4 Rule3(CS, GG, R, F, SF, ND)

Parameters

- CS *the set of the children for each node*
- GG *for each node*
- R *for each node*
- F *for each node*
- SF *for each node*
- ND, *the identities of the nodes which are found as non-droppers currently*

Return values

- *The identities of the nods which have correct or wrong* F, SF

```
1: for every node j do
2:   if j does not drop packets and R_j is correct then
3:     if R_j + GG_j ≐ F_ij + SF_ij then
4:       if F_ij is correct then
5:         SF_ij is correct
6:       else if SF_ij is correct then
7:         F_ij is correct
8:       end if
9:     end if
10:  end if
11: end for
```

Algorithm 5 Rule4(CS, SS, GG, F, SF)

Parameters

- *CS, the set of the children for each node*
- *SS, the set of the siblings for each node*
- *GG for each node*
- *F for each node*
- *SF for each node*

Return values

- *The identities of the nods which have correct or wrong SF*

```
1: for every node j do
2:    for all i ∈ CS_j do
3:       if F_kj is correct for ∀k ∈ SS_i then
4:          if SF_ij ≐ Σ_{∀k∈SS_i} F_kj + GG_j then
5:             SF_ij is correct
6:          end if
7:       end if
8:    end for
9: end for
```

Algorithm 6 Rule5(CS, R, F, ND)

Parameters

- *CS, the set of the children for each node*
- *R for each node*
- *F for each node*
- *ND, the identities of the nodes which are found as non-droppers currently*

Return values

- *The identities of the nodes which are newly found as droppers or non-droppers*

```
1: for every node j do
2:   if j does not drop packets then
3:     for all i ∈ CS_j do
4:       if R_i and F_ij are correct then
5:         if R_i + GG_i ≐ F_ij then
6:           i does not drop packets
7:         end if
8:       end if
9:     end for
10:  end if
11: end for
```

Algorithm 7 Rule6(CS, R, F, ND)

Parameters

- *CS, the set of the children for each node*
- *R for each node*
- *F for each node*
- *ND, the identities of the nodes which are found as non-droppers currently*

Return values

- *The identities of the nodes which are newly found as droppers or non-droppers*

for every node j **do**
  if i does not drop packets for $\exists i \in CS_j$ **then**
    if R_i and F_{ij} are correct **then**
      if $R_i + GG_i \doteq F_{ij}$ **then**
        j does not drop packets
      end if
    end if
  end if
end for

Algorithm 8 Rule7(CS, R, F)

Parameters

- *CS, the set of the children for each node*
- *R for each node*
- *F for each node*

Return values

- *The identities of the nodes which are newly found as droppers or non-droppers*

for every node j **do**
  if R_i is correct for $\forall i \in CS_j$ **then**
    if F_{ij} is correct for $\forall i \in CS_j$ **then**
      if $\sum_{\forall i \in CS_j} R_i \doteq \sum_{\forall i \in CS_j} F_{ij}$ **then**
        j does not drop packets
      end if
    end if
  end if
end for

Algorithm 9 Rule8(CS, R, F)

Parameters

- *CS, the set of the children for each node*
- *R for each node*
- *F for each node*

Return values

- *The identities of the nodes which are newly found as droppers or non-droppers*

```
1: for every node j do
2:   if F_ij is correct for ∀i ∈ CS_j then
3:     if R_j is correct then
4:       if R_j ≐ Σ_{∀i∈CS_j} F_ij then
5:         j does not drop packets
6:       end if
7:     end if
8:   end if
9: end for
```

Algorithm 10 ExtenedRule1(CS, GG, R, F)

Parameters

- *CS, the set of the children for each node*
- *GG for each node*
- *R for each node*
- *F for each node*

Return values

- *The identities of the nods which have correct R*

```
1: for every node j do
2:   if F_ij is correct for ∀i ∈ CS_j then
3:     if R_j ≐ Σ_{∀i∈CS_j} F_ij then
4:       if R_j ≐ Σ_{∀i∈CS_j} (R_i + GG_i) then
5:         R_j is correct
6:       end if
7:     end if
8:   end if
9: end for
```

Algorithm 11 ExtenedRule2(CS ,GG, R, F)

Parameters

- *CS, the set of the children for each node*
- *GG for each node*
- *R for each node*
- *F for each node*

Return values

- *The identities of the nods which have correct R, F*

1: **for** every node k **do**
2: **if** $R_k \doteq \sum_{\forall i \in CS_j \& \forall j \in CS_k} F_{ij} + GG_j$ **then**
3: **if** $R_j \doteq \sum_{\forall i \in CS_j} F_{ij}$ **then**
4: R_k, R_j and F_{ij} are correct for $\forall i \in CS_j$ and $\forall j \in CS_k$
5: **end if**
6: **end if**
7: **end for**

As the tree structure dynamically changes every certain time interval, behaviors of sensor nodes can be observed in a large variety of scenarios. In addition, as the information of node behaviors has been accumulated, the sink periodically run our proposed heuristic ranking algorithms to identify most likely bad nodes from suspiciously bad nodes whose number are potentially large. Global Ranking-Based (GR) Method is based on the heuristic that, the more times a node is identified as suspiciously bad, the more likely it is a bad node. Stepwise Ranking-Based (SR) method reduces the value of node v's accused account by the times that u and v have been suspected together once a bad node u is identified, for any other node v that has been suspected together with node u since the GR method will misaccuse innocent nodes that have frequently been parents or children of bad nodes. Hybrid Ranking-Based (HR) Method concerns the possibility that an innocent node being framed by bad nodes is also considered by not choosing the nodes who have always being suspected together with already-identified bad nodes and after selecting a most likely bad node, choses the one which has the highest accused account value among the rest if the node has not always been accused together with the bad nodes that have been identified already. The simulation results show that the hybrid ranking is the best ranking algorithm among the three for its high detection rate and low false positive

Algorithm 12 ExtenedRule3(CS ,GG, R, F, SF)

Parameters

- CS, *the set of the children for each node*
- GG *for each node*
- R *for each node*
- F *for each node*
- SF *for each node*

Return values

- *The identities of the nods which have correct* F, SF

1: **for** every node k **do**
2: **if** $R_k \doteq \sum_{\forall j \in CS_k \& \exists i \in CS_j} (F_{ij} + SF_{ij})$ **then**
3: **if** $(R_j + GG_j) \doteq (F_{ij} + SF_{ij})$ for $\exists i \in CS_j$, $\forall j \in CS_k$ **then**
4: R_k, R_j are correct
5: **if** F_{ij} is correct **then**
6: SF_{ij} is correct
7: **else if** SF_{ij} is correct **then**
8: F_{ij} is correct
9: **end if**
10: **end if**
11: **end if**
12: **end for**

Algorithm 13 ExtenedRule4(CS, EF, R, F, SF)

Parameters

- *CS, the set of the children for each node*
- *R for each node*
- *EF for each node*
- *R for each node*
- *F for each node*
- *SF for each node*

Return values

- *The identities of the nods which have correct F*

```
1: for every node j do
2:    if EF_j ≐ ∑_{∀i∈CS_j} F_ij then
3:       if R_j ≐ ∑_{∀i∈CS_j} F_ij then
4:          F_ij are correct for ∀i ∈ CS_j
5:       end if
6:    end if
7: end for
```

and almost all bad nodes can be identified after 8 rounds regardless of the attack model. It is also shown that the less the number of bad nodes, the easier to identify these nodes and as the threshold to categorize nodes is higher, the detection rate becomes lower. The collusion among nodes also makes the detection rate lower.

CHAPTER 4. RESULTS

Using ns2, we conduct extensive simulation to evaluate the performance of our proposed scheme. We study the *failed-detection rate* (i.e., the percentage of packet droppers that are not identified) as the *density of networks* and the *percentage of droppers* vary. We simulate three different network settings: 101 nodes (including the base station) deployed to $300 \times 300m^2$ area (denoted as 300×300), 101 nodes deployed to $400 \times 400m^2$ area (denoted as 400×400), and 101 nodes deployed to $500 \times 500m^2$ (denoted as 500×500). In all simulation results, each packet dropper drops 50% of the packets which are supposed to be forwarded.

4.1 Impact of network density on failed-detection rate

In Fig. 4.1, we show the failed-detection rate as the network density varies. Here, the number of regular sensor nodes is 100, natural dropping rate of packets (due to channel noises, interference, and other environmental factors) is 1%, mis-overhearing rate is 3%, and the number of detection rounds is 5. We show the result in two cases: when 10% nodes are droppers and when 20% nodes are droppers. As can be seen, as the network density decreases, the failed-detection rate increases. This is because, as the network density decreases, each dropper has less neighbors that can monitor its behavior.

4.2 Impact of natural dropping rate and mis-overhearing on failed-detection rate

In reality, some packets are dropped not only by the attackers but also by environment factors such as interference. Additionally, a node can not overhear the communication of its neighboring nodes perfectly. Thus, we study how these factors affect the performance of our

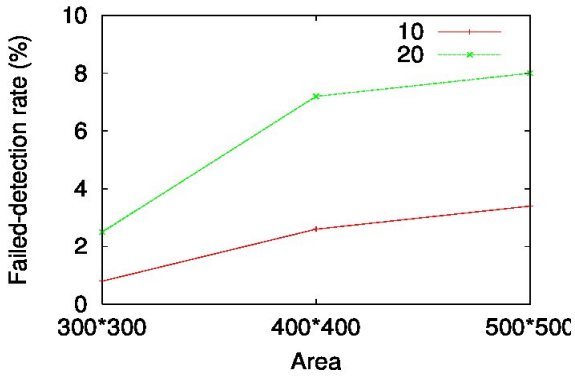

Figure 4.1 Failed-detection rate vs. network density

scheme. Let the dropping rate affected by environmental reasons vary from 1% to 6% and mis-overhearing rate vary from 3% to 8%. In Fig. 4.2, the horizontal axis presents three different combinations of the natural dropping rates and the mis-overhearing rates, and the vertical axis represents the failed-detection rates with these three combinations. In addition, two different settings of network density are considered: 100 nodes deployed to $300 \times 300m^2$ and 100 nodes deployed to $400 \times 400m^2$.

As can be seen, the larger are the natural dropping/mis-overhearing rates, the higher is the failed-detection rate. The reason is that the allowed error range for R, F and SF increases as the natural dropping/mis-overhearing rates increase. Consequently, even though some droppers drop some of its received packets, they could be protected from being detected.

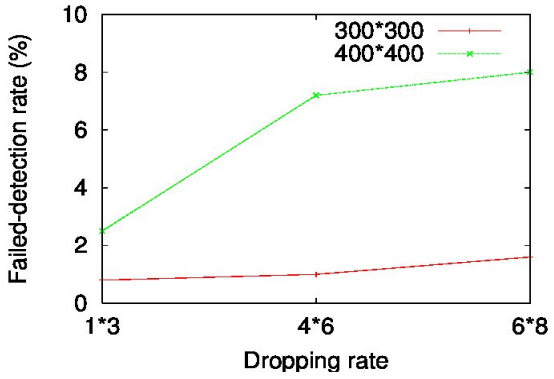

Figure 4.2 Failed-detection rate vs. natural dropping rate and mis-overhearing rate

4.3 Impact of percentage of droppers on failed-detection rate

In Fig. 4.3, the horizontal axis represents the percentage of all droppers and the vertical axis represents the corresponding failed-detection rates. The natural dropping rate is 1% and mis-overhearing rate is 3%. The number of detection rounds is 5. The results were shown in two network settings: 100 nodes deployed to $300 \times 300m^2$ and 100 nodes deployed to $400 \times 400m^2$.

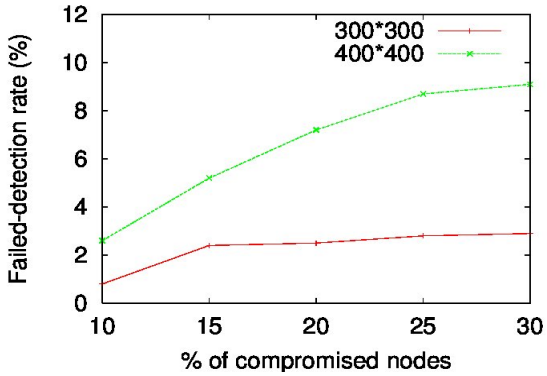

Figure 4.3 Failed-detection rate vs. percentage of droppers

As the number of droppers increases, the failed-detection rate also increase. This is because the number of neighboring nodes which make correct monitoring reports on a path decreases. This becomes more clear when we compare the setting of $300 \times 300m^2$ to the setting of $400 \times 400m^2$. In the $400 \times 400m^2$ setting, the failed-detection rate increases faster than in the $300 \times 300m^2$ setting as the number of compromised nodes increases. Let's think about the number of neighboring nodes for each node in two areas. To find droppers, we need at least one innocent neighboring node for each dropper. The network in the $400 \times 400m^2$ setting has lower density and so the percentage that each dropper has no innocent neighboring node is higher than that in the $300 \times 300m^2$ setting.

4.4 Impact of detection rounds on failed-detection rate

In Fig. 4.4, the horizontal axis represents the number of detection rounds, and the vertical axis represents the corresponding failed-detection rates. The number of nodes is 100, the natural dropping rate is 1% and the mis-overhearing rate is 3%. The area is $400 \times 400m^2$. As

can be seen, as the number of rounds increases, the failed-detection rate drops. This is because more paths are used to forward packets from a source and the number of paths where any node does not drop packets increases. This increase helps the base station to find innocent nodes. However, the number of paths where some nodes drop packets also increases but these paths cross the paths where any node does not drop packets. Thus, the innocent nodes on the paths without packets dropped can accuse the droppers on the paths with packets dropped.

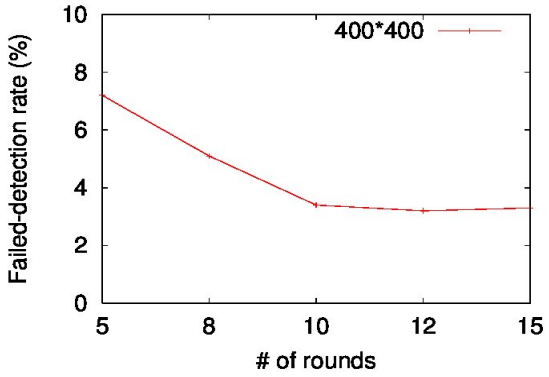

Figure 4.4 Failed-detection rate vs. number of detection rounds

CHAPTER 5. SUMMARY AND DISCUSSION

In this thesis, we propose a simple yet effective scheme to identify packet droppers. This scheme only requires sending and forwarding nodes to report their observations to a base station, and the base station can analyze the reports, identify inconsistencies in the reports, and then locate the intruders using two kinds of the rules we propose. The basic rules hold regardless of the number of compromised neighboring nodes since they are based on consistent relation for the report of the base station and each node and the extended rules hold unless a node and all its neighboring nodes are compromised at the same time, which is a rare case. Extensive simulations have been conducted to demonstrate the effectiveness of the scheme. In the simulations, the failed-detection rate increases as the network density decreases. This is because, as the network density decreases, each dropper has less neighbors that can monitor its behavior. It is also shown that the larger are the natural dropping/mis-overhearing rates, the higher is the failed-detection rate. The reason is that the allowed error range for R, F and SF increases as the natural dropping/mis-overhearing rates increase. As the number of droppers increases, the failed-detection rate also increase. The main cause for this is that the number of neighboring nodes which make correct monitoring reports on a path decreases. As the number of rounds increases, the failed-detection rate also drops. This is because more paths are used to forward packets from a source and the number of paths where any node does not drop packets increases. The proposed scheme can also tolerate malicious reports, natural packet dropping and so on.

As a future work, the distinction of dropping by environment factor such as interference from dropping by compromised nodes might give better performance. The range for the valid value of R, F and SF can be also adjusted as the packet reception rate of the base station for over all sensor nodes changes.

REFERENCES

Budhaditya Deb and Sudeept Bhatnagar Badri Nath (2003). ReInForM: Reliable Information Forwarding using Multiple Paths in Sensor Networks. *IEEE conference on Local Computer Networks (IEEE-LCN).*

Thiemo Voigt, Adam Dunkels and Torsten Braun (2005). On-demand construction of non-interfering multiple paths in wireless sensor networks. *The 2nd Workshop on Sensor Networks at Informatik.*

Suk-Bok Lee and Yoon-Hwa Choi (2006). A secure alternate path routing in sensor networks. *Computer Communications. 30*(1), 153–165.

Issa Khalil, Saurabh Bagchi and Cristina Nina-Rotaru (2005). DICAS: Detection, Diagnosis and Isolation of Control Attacks in Sensor Networks. *IEEE Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm).*

Saurabh Ganeriwal, Laura K. Balzano and Mani B. Srivastava (2004). Reputation-based framework for high integrity sensor network. *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks.*

FanYe, Hao Yang and Zhen Liu (2007). Caching "Moles" in Sensor Networks. *ICDCS '07 Proceedings of the 27th International Conference on Distributed Computing Systems.*

Sencun Zhu, Sanjeev Setia, Sushil Jajodia and Peng Ning (2004). An Interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected False Data in Sensor Networks. *IEEE Symposium on Security and Privacy.*

Z. Yu and Y. Guan (2006). A Dynamic En-route Scheme for Filtering False Data in Wireless Sensor Networks. *IEEE Infocom.*

Chuang Wang, Taiming Feng, Jinsook Kim, Guiling Wang and Wensheng Zhang (2009). Catching Packet Droppers and Modifiers in Wireless Sensor Networks. *IEEE SECON.*

Afrand Agah, Mehran Asadi and Sajal K. Das (2006). Prevention of DoS Attack in Sensor Networks using Repeated Game Theory. *Proceedings of ICWN.*

A. P. da Silva, M. Martins, B. Rocha, A. Loureiro, L. Ruiz and H. C. Wong (2005). Decentralized intrusion detection in wireless sensor networks. *Q2SWinet.*