

# DPDSN: Detection of packet-dropping attacks for wireless sensor networks

Vijay Bhuse, *Student Member, IEEE*, Ajay Gupta, *Senior Member, IEEE*, and Leszek Lilien, *Senior Member, IEEE*

**Abstract-** Denial-of-service (DoS) attacks on wireless sensor networks (WSNs) can deplete network resources and energy without much effort on the part of an adversary. Packet-dropping attacks are one category of DoS attacks. Lightweight solutions to detect such attacks on WSNs are needed. Current techniques for detecting such attacks in ad hoc networks need to monitor every node in the network. Once they detect malicious nodes that drop packets, a new path has to be found that does not include them. In this paper, we propose a lightweight solution called DPDSN. It identifies paths that drop packets by using alternate paths that WSN finds earlier during route discovery. Responding to a packet-dropping attack incurs no additional cost because one of the alternate paths is utilized for all subsequent communication. DPDSN does not require monitoring individual nodes, making it feasible for WSNs. We formulate the probability of success and failure of DPDSN in the presence of malicious nodes that drop packets. We compare our approach with existing techniques. Our analysis found that the overhead of DPDSN is at most  $O(\sqrt{N})$  for a two-dimensional grid network of  $N$  nodes. Our simulations show that the overhead of DPDSN for a WSN with 100 nodes is less than 3% of energy consumed on route discovery when using DSR or Directed Diffusion routing protocols.

**Keywords-** Denial-of-service, intrusion detection, wireless sensor networks.

## I INTRODUCTION

Wireless sensor networks (WSNs) consist of small devices—called sensor nodes—with a radio, a processor, a memory, a battery and sensor hardware. With a widespread deployment of these devices, one can precisely monitor the environment. Sensor nodes are resource-constrained in terms of the radio range, processor speed, memory size and power. The resource-constrained nature forces designers to design application-specific systems. This leads to specific communication patterns in WSNs. Traffic is not as random as in ad hoc networks. Karlof and Wagner [KW03] classify WSN traffic into one of three categories:

1. *Many-to-one*: Many sensor nodes send readings to a base station or aggregation point in the network.
2. *One-to-many*: A single node (typically a base station or an aggregator) floods several sensor nodes with query or control information.
3. *Local communication*: Neighboring nodes send localized messages to discover and coordinate tasks.

Apart from this, sensor nodes are generally static and the traffic rate in WSNs is very low. Traffic is periodic as well. There may be long idle periods during which sensor nodes turn off their radio and go to sleep to save energy consumed by idle listening. MAC protocols like S-MAC and TDMA-MAC [YH02, CK04] have been introduced to harness this WSN property to save energy. Sensor nodes use batteries, so energy is a precious resource. Recharging or replacing batteries is expensive and may not even be possible in some situations. Therefore, WSN applications need to be extremely energy-aware.

WSNs are mostly unguarded and the wireless medium is inherently broadcast in nature. This makes WSNs vulnerable to all kinds of denial-of-service (DoS) attacks. Without proper security measures, an adversary can launch various kinds of attacks in hostile environments. These attacks can disrupt the normal working of WSNs and can even defeat the purpose of their deployment. An adversary can launch some attacks without even cracking keys used for cryptography-based solutions. DoS attacks (like packet dropping, false route request, or flooding) can deplete the network of energy without much effort on the part of an adversary. Therefore, intrusion detection mechanisms to detect DoS attacks are needed. To be practical for implementing on WSNs, solutions for detecting intrusions should be lightweight.

In this paper, we address the problem of detecting packet-dropping attacks in WSNs. Apart from malicious intent, there can be other reasons of packet dropping like collisions, buffer overflows, congestion, etc. It is important to find solutions that take these factors into account, for example, to prevent false alarms.

Existing solutions for detecting packet dropping in ad hoc networks work by monitoring individual nodes. Sleep-wakeup schedules followed by nodes in a WSN [YH02] make continuous monitoring impractical. Also, monitoring individual nodes is too expensive for WSNs.

Our approach, called DPDSN (Detection of Packet-Dropping attacks for wireless Sensor Networks), uses the observation that alternate routing paths are readily available in WSNs, which are typically dense. DPDSN monitors paths and detects whether any node on a path drops packets. Once we detect such an event, we switch to an alternate path for communication. We always keep an alternate path ready to minimize the switching delay. The cost of finding an alternate path is minimized by having it embedded in route discovery of source-initiated and receiver-initiated routing protocols such as the ones proposed in [JM96, IG00].

Keeping alternate paths readily available is justified even if no packet-dropping attacks are detected. First, the alternate paths can be used for load-balancing transmissions. Second, uneven consumption of energy is a biggest threat to lifetime of a WSN because it can partition the network. Use of alternate paths for transmission can protect nodes on the original path from expending all their energy too soon.

DPDSN can be extended to detect individual nodes that drop packets. We do so only if there is a real need, because finding such nodes is costly for resource-constrained WSNs.

## II RELATED WORK

Marti *et al.* [M01] discussed two techniques that detect compromised nodes that agree to forward packets but fail to do so. The authors use *watchdogs* that identify misbehaving nodes and a *pathrater* that helps routing protocols avoid these nodes. When a node forwards a packet, the node's watchdog verifies that the next node in the path also forwards the packet. The watchdog does this by listening promiscuously to the next node's broadcast transmissions. If the next node does not broadcast the packet, it is misbehaving and the watchdog detects it. Every time a node fails to forward a packet, the watchdog increments the failure-tally. If the tally exceeds a certain threshold, it is determined that the node is misbehaving; this node is then avoided with the help of the *pathrater*. The *pathrater* combines knowledge of misbehaving nodes with link reliability data to pick the route most likely to be reliable. Each node maintains a rating for every other node it knows about in the network. It calculates a path metric by averaging the node ratings in the path. The overhead of passive continuous passive listening is formidable for WSNs.

Buchegger *et al.* [BB02] proposed a mechanism that detects misbehaving nodes by means of observations or reports about several types of attacks. This allows nodes to find routes around misbehaving nodes and to isolate them from the network. Nodes have a *monitor* for observations, *reputation records* for first-hand observations and trusted second-hand reports, *trust records* to control trust given to received warnings, and a *path manager* to adapt their behavior according to reputation of other nodes. This approach involves continuous monitoring similar to Marti's approach and collecting information about intrusion detections at other places in the network. The overhead is prohibitive for WSNs.

Michiardi *et al.* [MM02] proposed a collaborative reputation mechanism that has a *watchdog* component. However, it is complemented by a reputation mechanism that differentiates between subjective reputation (observations), indirect reputation (positive reports by others), and functional reputation (task specific behavior). They are weighted for a combined reputation value used to make decisions about cooperation with or gradual isolation of a node. This approach involves continuous monitoring and collecting information about intrusion detections at other places in the network for specific functions. The overhead is too high for WSNs.

Huang *et al.* [HL03] proposed a mechanism that needs separate monitoring nodes, specifically one monitor per cluster (nodes that are in one-hop range form a cluster). The approach requires monitors to be active. If there is one

monitor per cluster, the monitor does most of the work. In WSNs, there is a risk that monitor nodes run out of energy before the network does or before the network gets partitioned. This contradicts one of the main goals of prolonging WSN lifetime and keeping WSN connected as much as possible (since battery replacement is a very costly or unavailable alternative).

All the above approaches monitor individual nodes all the time. Continuous monitoring of each and every node is not feasible for resource-constrained WSNs especially when extending lifetime is the main goal in the design of WSNs. Our proposed solution, DPDSN avoids continuous monitoring of every node.

## III THE DPDSN APPROACH

This section describes the DPDSN approach, including detection of compromised paths, embedding alternate path discovery in route discovery, overhead analysis, and discussion of malicious node discovery.

### A Detection of Compromised Paths

Our detection mechanism uses an alternate path between a source and a destination. We propose that the process of finding an alternate path be embedded in the route discovery phase of routing protocols like DSR [JM96] and Directed Diffusion [IG00]. Ideally, the alternate path does not have any node in common with the original path. We assume that the source and the destination are not malicious or compromised. Since our goal is to detect packet-dropping attacks, we do not consider masquerading or forging packets. For counteracting masquerades and packet forging, techniques such as the ones proposed in [BG05, LP05] can be used.

Packet loss can be caused by congestions due to heavy traffic, collisions at link layer, buffer overflows, etc. In WSNs, congestions and buffer overflows seem unlikely to happen because of low traffic rates. Reliable MAC protocol rules out collisions as a reason for packet dropping. Assuming reliable MAC protocol and assuming low traffic rates, the main possible reason for packet losses in WSNs is malicious non-forwarding or packet dropping by an adversary or compromised nodes. We focus on this to detect paths that drop packets.

**Detect\_compromised\_path(s, d)**

**begin**

    Get  $n_s, n_r$ .

**while** (TRUE)

**if**  $n_s - n_r > 0$  **then**

            Guess that packets are being dropped by malicious nodes on the source-to-destination path.

**return** TRUE

**else**

**return** FALSE

        Wait till next verification cycle.

**end**

Fig. 1: Detection of a packet-dropping path.

Compromised paths can be detected as follows. Let  $n_s$  be the number of packets sent by the source in a given period of time. Using the alternate path found during route discovery, the source periodically requests the destination to send the number of packets received,  $n_r$ . In DPDSN, the source sends query to the destination using an alternate path, requesting it to send  $n_r$ . Algorithm in Fig. 1 outlines the detection approach for the currently used path.

The alternate path is used not only to verify whether  $n_s=n_r$ . If we find that packets are dropped by the original path, it can also be used for all subsequent communication.

**B Embedding Alternate Path Discovery in Route Discovery**

Finding an alternate path during route discovery is a challenging problem and finding an optimal alternate path is beyond the scope of this paper. We address the embedding problem heuristically and show possible approaches for DSR and Directed Diffusion.

*Embedding in DSR* Let us consider an example shown in Fig. 2a and Fig. 2b for DSR. Node 1 is the source whereas Node 8 is the destination. Node 1 sends out a route request packet, which floods the network as shown by broken-line arrows in Fig. 2a. Node 8 responds by sending route reply packet as shown by single-line arrows in Fig. 2b. Double-line arrows in Fig. 2b indicate an alternate path that can be used later to verify the number of actual packets received by the destination. In this case, Node 8 determines if there is an alternate path to Node 1.

The destination node finds an alternate path that does not have any node in common with the nodes that are on the original path. Next, we need to discuss the difficulties one may face while addressing this problem.

In some cases it may not be possible to find an alternate path. For example, in Fig. 2c, the original path from Node 8 to Node 1 is through Nodes 5 and 2. The alternate path from Node 8 to Node 1 includes Nodes 4, 5 and 2. In this case, detecting whether packets are dropped on a path from Node 8 to Node 1 will not work if Node 2 drops packets. Note that even if we detect that Node 2 drops packets, there is no alternative path to use to avoid packet dropping on a path from Node 1 to Node 8.

There can be cases for which even if there is an alternate path from the destination to the source, the destination cannot find it from the route request packets it receives. It happens because some information is lost when intermediate nodes forward a route request packet after receiving multiple route request packets from different neighbors. In Fig. 2a, Node 7 receives route request packets from Nodes 4 and 6 but it forwards only one of the packets to Node 8. In general suppose  $s$  is the source,  $d$  the destination and  $j$  the only neighbor of  $d$ . Also, assume that  $j$  receives multiple route request packets from its predecessors. Node  $j$  will forward only one route request packet to  $d$ . In this case there is no alternate path from  $d$  to  $s$  because  $j$  is the only neighbor of  $d$  that can forward packets from  $s$  to  $d$  or  $d$  to  $s$ . But there may be an alternate path from  $j$  to  $s$ . After receiving only one route

request packet (of course from  $j$ ), node  $d$  may ask node  $j$  to find an alternate path from  $j$  to  $s$ , if one exists. If  $j$  receives only one route request packet, then it can ask its predecessor  $k$  to find an alternate path from  $k$  to  $s$ .

There can be a case in which node  $d$  receives multiple route request packets but it cannot find *node-disjoint* alternate path to reach  $s$ . It may find an alternate path to reach some intermediate node  $m$ .

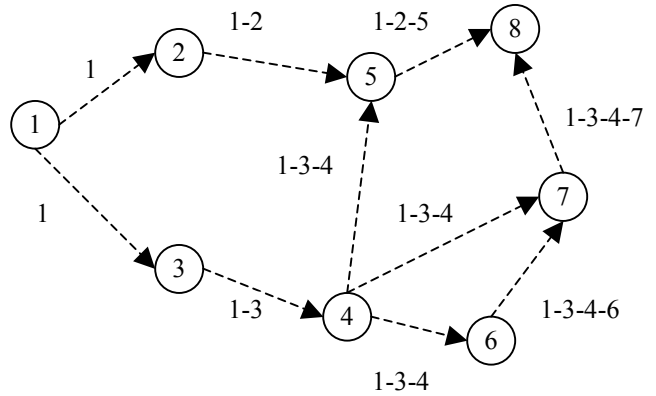


Fig. 2a: Source (Node 1) floods the network with DSR route request packet. Node 8 is the destination.

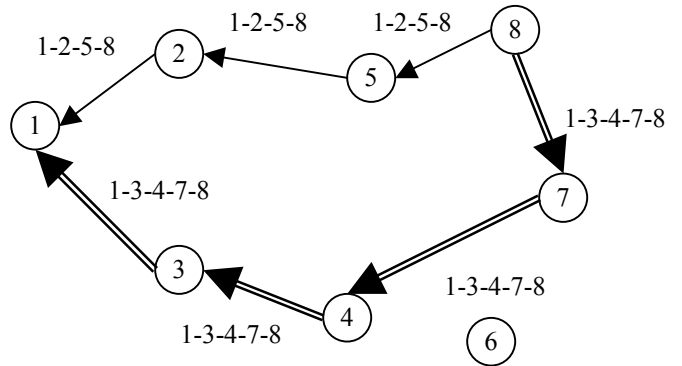


Fig. 2b: Destination (Node 8) sends route reply packet (single-line arrows). Destination finds alternate path (double-line arrows).

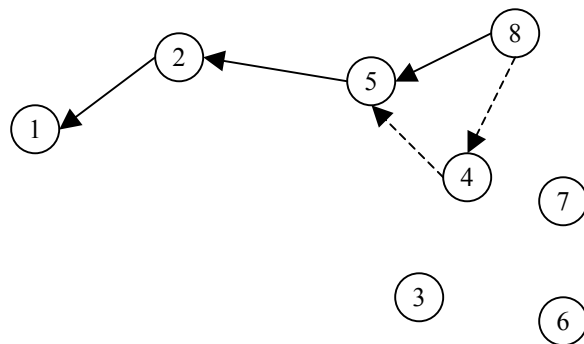


Fig. 2c: Alternate path from Node 8 to Node 1 does not exist.

Algorithm in Fig. 2d outlines a heuristic process of finding an alternate path when DSR is used for route discovery. A straightforward solution is to perform route discovery using DSR and mark the edges of the original path. It incurs significant cost due to flooding. A better heuristic approach would be to keep two route requests at every node when a node receives multiple route requests. One of the route requests is used for establishing the path and second one will be used for alternate path. The algorithm is outlined in Fig. 2d. Obviously this algorithm is just one approach and my not result in an efficient alternate path. Finding optimal alternate path is a challenging problem as mentioned above and our future work involves designing efficient lightweight solutions. The described algorithm suffices for this paper to address packet-dropping attacks for WSNs.

#### Find\_alternate\_path(s, d, AP)

```

begin
  // Let N be the number of nodes. Each node stores 2
  //route requests. Node j stores its route replies in
  //R[j, 0] and R[j, 1]. predecessor[x] is a function that
  //defines the set of nodes that can send route request
  //to node x. AP is initialized to {d}.
  if (d=s) then return.
  if (|predecessor [d]|=1) then //No alternate path to s exists.
    j= predecessor [d]
  else
    if(j ∈ R[d, 1])
      j= predecessor [d] //note j not in R[d, 0]
      AP=AP ∪ {i}
  Find_alternate_path(s, d, AP).
end

```

Fig. 2d: A heuristic algorithm to determine alternate path.

*Embedding in Directed Diffusion* Let us consider the alternate path discovery problem for Directed Diffusion, a receiver-initiated protocol. In Directed Diffusion, the destination/sink (Node 8) floods the network in search of some data (called an *interest*) as shown by broken-line arrows in Fig. 3a. Whenever that message reaches the source (Node 1), it floods the network back as shown by solid line arrows. Then, the source reinforces only one path from the source to the sink, which is shown by single-line arrows in Fig. 3b. This path is used for all future communication. We modify the reinforcement step similar to the ideas used for DSR to identify an alternate path. The double line arrows in Fig. 3b show an alternate path discovered during reinforcement step.

#### C Analysis of Overhead for Finding Alternate Paths

We analyze the overhead for finding alternate paths for DSR and Directed Diffusion. Let the cost of sending a packet be  $T_x$  and the cost of receiving a packet be  $R_x$ . Let  $N$  be the number of nodes,  $m$  be the average number of node neighbors (*neighbor* of a node is any other node within its broadcast range). Let  $p$  be the length of the path.

*Analysis for DSR* For DSR, the approximate cost of route request and route reply is  $N(T_x + mR_x)$  and  $p(T_x + R_x)$ , respectively. The cost of finding an alternate path is  $p(T_x +$

$R_x)$ . Therefore, the ratio of the cost of finding an alternate path to the cost of DSR path discovery is  $p(T_x + R_x) / (N(T_x + mR_x) + p(T_x + R_x))$ .

The overhead ratio is plotted against  $p$  and  $R_x/T_x$  in Fig. 4.  $R_x/T_x$  is the ratio of the energy consumed for receiving a packet to the energy consumed for sending a packet. We assume  $N=100$  and  $m=6$ . The reason for this value of  $m$  is as follows. The maximum coverage and the maximum number of neighbors for each sensor are provided by beehive configuration, in which there are 6 neighbors per node (except nodes on the boundary) [JW04].

Sending a packet consumes much more energy than receiving it. For MICA2 mote sensors [MM04], sending a packet consumes 81 mW energy, whereas receiving it consumes 30 mW energy [MV05], that is  $R_x/T_x$  is 0.37. To cover the  $R_x/T_x$  range well, we chose values of  $R_x/T_x$  to be from 0.25 to 0.5 as shown in Fig. 4. Overhead increases as path length or  $R_x/T_x$  increases. Still it is no higher than 6% for a path of length 13

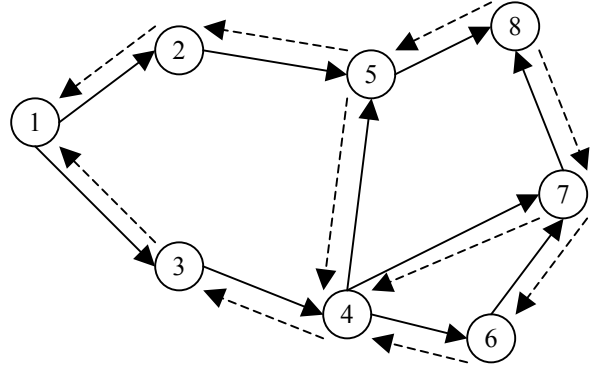


Fig. 3a: Source (Node 1) sends out interest (solid lines). Sink (Node 8) replies back (broken lines).

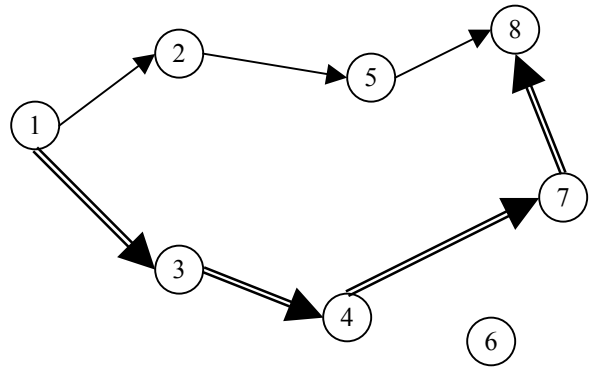


Fig. 3b: Source (Node 1) reinforces a path (single-line arrows) to reach sink (Node 8) and finds alternate path (double-line arrows).

*Analysis for Directed Diffusion* For Directed Diffusion, the approximate cost of path discovery is  $2N(T_x + mR_x) + p(T_x + R_x)$ . The cost of finding an additional path is  $p(T_x + R_x)$ . Therefore, the ratio of overhead of finding an alternate path to the cost of Directed Diffusion path discovery is  $p(T_x + R_x) / (2N(T_x + mR_x) + p(T_x + R_x))$ .

The overhead ratio is plotted against  $p$  and  $R_x/T_x$  in Fig. 5. We assume  $N=100$  and  $m=6$ . For reasons explained above, we again chose values of  $R_x/T_x$  from 0.25 to 0.5 as shown in Fig. 5. Overhead increases as path length or  $R_x/T_x$  increases. Still it is no higher than 3% for a path of length 13.

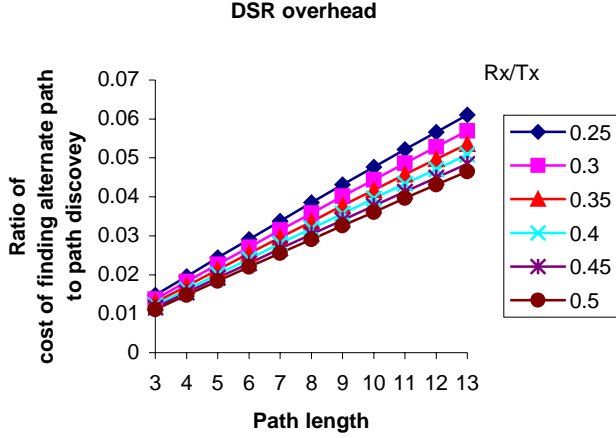


Fig. 4: Ratio of overhead of DPDSN to DSR path discovery as a function of path length and  $R_x/T_x$ .

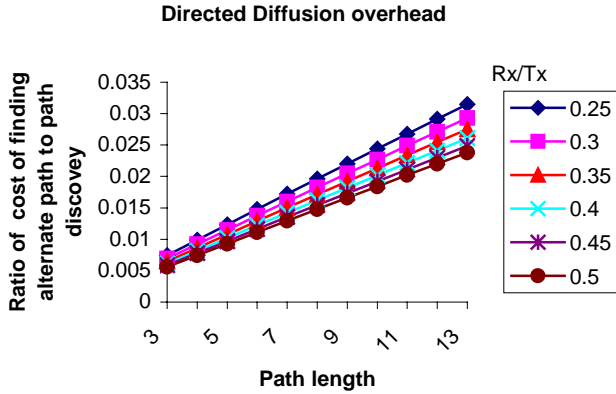


Fig. 5: Ratio of overhead of DPDSN to Directed Diffusion route discovery as a function of path length and  $R_x/T_x$ .

#### D Probability of Success for DPDSN

DPDSN works when we can correctly determine whether the original path is dropping packets. DPDSN *succeeds* whenever an alternate path does not have any malicious nodes that drop packets. In the following lemma we assume that  $N$  nodes are placed randomly, and  $M$  of them are malicious and drop packets.

**Lemma 1:** Assuming that we always find an alternate path, the probability of success for DPDSN is  $\binom{N-M}{p} \div \binom{N}{p}$  where  $N$  is the number of nodes in the network,  $M$  is the number of

malicious nodes that drop packets, and  $p$  is the length of the alternate path.

**Proof:** Let  $P(k)$  be the probability of finding  $k$  malicious nodes on a path of length  $p$ . Note that  $P(0)$  is the probability of success for DPDSN.

For  $k \geq 0$ ,  $P(k) =$

$$\frac{(\text{No. of ways of selecting } k \text{ malicious nodes}) * (\text{No. of ways of selecting } p-k \text{ non-malicious nodes})}{(\text{No. of ways of selecting } p \text{ nodes from } N \text{ nodes})}$$

There are  $\binom{N}{p}$  ways of selecting  $p$  nodes from  $N$  nodes,

$\binom{M}{k}$  ways of selecting  $k$  malicious nodes from  $M$  malicious

nodes and  $\binom{N-M}{p-k}$  ways of selecting  $p-k$  non-malicious nodes from  $N-M$  nodes.

$$\text{Hence, } P(k) = \frac{\binom{M}{k} \binom{N-M}{p-k}}{\binom{N}{p}} \text{ for } k \geq 0$$

The probability  $P(0)$  of finding no malicious node on the alternate path is:

$$P(0) = \frac{\binom{N-M}{p}}{\binom{N}{p}}. \blacksquare$$

Obviously, the probability of failure of DPDSN is  $1-P(0)$ .

#### Detect\_compromised\_nodes(s, d, V)

**begin**

// Let  $s, s+1, s+2, \dots, d-1, d$  be the nodes that are

// on the path from source  $s$  to destination  $d$ .

//  $V$  is the set of nodes that drop packets.

**if**  $d-s=2$  **and** Detect\_compromised\_path( $s, d$ )

**= TRUE then**

$V=V \cup \{s+1\}$

**elseif** Detect\_compromised\_path( $s, d$ ) = TRUE

**then**

**if** Detect\_compromised\_path( $s, [s+(d-s)/2]$ )

**= TRUE then**

**Detect\_compromised\_nodes**( $s, [s+(d-s)/2], V$ )

**if** Detect\_compromised\_path( $[s+(d-s)/2] + 1, d$ ) = TRUE **then**

**Detect\_compromised\_nodes**( $[s+(d-s)/2] + 1, d, V$ )

**end**

Fig. 6: Algorithm detecting compromised nodes.

#### E Finding a Specific Node Dropping Packets

DPDSN can be used to detect specific nodes that drop packets. Assume, without a loss of generality, that the nodes

on the path from source  $s$  to destination  $d$  are labeled  $s, s+1, s+2, \dots, d-1, d$ .

We can detect whether packets are being dropped on this path using the mechanism discussed in Section IIIA. We describe that algorithm as the procedure `Detect_compromised_path` ( $s, d$ ) shown in Fig. 1. It returns TRUE if packet dropping by a malicious node is detected on the currently used path from  $s$  to  $d$ .

Let `Detect_compromised_nodes`( $s, d, V$ ) denote a procedure that performs a binary search for packet-dropping nodes on a path from  $s$  to  $d$  (Fig. 6). It stores detected packet-dropping nodes in set  $V$ .  $[x]$  represents integer portion of a real number  $x$  (the *floor* function).

#### IV COMPARISON WITH EXISTING APPROACHES AND SIMULATION

In this section, we estimate the overhead of other approaches and compare them with DPDSN. We calculate the overhead in terms of the number of messages sent and received. We compute the overhead for all the approaches for just the detection of packet dropping by a path.

The other existing approaches incur separate cost for the *response*. Response includes finding alternate path avoiding nodes dropping packets. It is important to note that for DPDSN the cost for response is *included* in the cost of bad path detection. The reason is that the *same* alternate path can be used later for transmitting packets.

##### A Analytical Results on DPDSN Overhead

The destination is queried periodically by the source verifying the number of packets received by the destination. The following lemma considers overhead for one query period  $T$ , that is for a single verification cycle.

**Lemma 2:** For a single verification cycle, the overhead for DPDSN is  $O(p)$  messages for a network of  $N$  nodes where  $p$  is the path length.

**Proof:** Overhead of DPDSN is the cost of establishing an alternate path, and sending and receiving a query (verifying the number of packets received by the destination) that uses this path. For any path, one message (from the destination to the source) has to be sent during the route reply phase and two messages (one from the source to the destination and another from the destination to the source) have to be sent for verifying the number of packets received. Every node on the path sends and receives these 3 messages. Therefore, for a path length of  $p$  hops, the cost is  $3p$  send messages and  $3p$  receive messages. This cost is for a single verification cycle. Hence, the overhead is  $O(p)$ . ■

**Corollary:** For a single verification cycle, the overhead for DPDSN for a two-dimensional grid sensor network of  $N$  nodes is  $O(\sqrt{N})$  messages where  $N$  is the number of nodes.

**Proof:** Overhead of DPDSN, for a path length of  $p$  hops, is  $3p$  send messages and  $3p$  receive messages. This cost is for a single verification cycle. For a two-dimensional grid sensor network of  $N$  nodes, maximum value of  $p$  can be  $2\sqrt{N}$ . Therefore, the maximum cost of DPDSN is  $6\sqrt{N}$  send messages and  $6\sqrt{N}$  receive messages. Hence, the overhead is  $O(\sqrt{N})$ . ■

##### B Analysis of DPDSN Overhead for Finding Specific Node Dropping Packets

For a path of length  $p$ , determining whether packets are dropped on the path costs  $3p$  send messages and  $3p$  receive messages. Detecting a specific node that drops packets will cost  $6p$  send messages and  $6p$  receive messages. In the worst case, all nodes on the path except the source and the destination can be malicious and detecting them all one after another will cost  $3p \cdot \log p$  send messages and  $3p \cdot \log p$  receive messages. Detecting individual nodes that drop packets incurs significant cost for WSNs, so `Detect_compromised_nodes()` should be used only if there is a real need of such fine detection.

##### C Comparison with Other Approaches

The following analysis uses  $j$  as the number of packets sent, and  $\delta$  as the number of packet-dropping instances detected in time period  $T$ .

For the same time interval and path of length  $p$ , Huang and Lee's approach [HL03] needs  $p/2$  dedicated monitor nodes placed on a path alternately, such that each of the  $p/2$  monitors nodes guards the remaining  $p/2$  nodes. The monitors actively participate in the communication and they have to take into account each and every message sent in a period of time  $T$  for detection of packet dropping. If  $j$  messages are sent in a period of time  $T$ , the cost is  $jp$  send messages and  $jp$  receive messages.

If  $j$  messages are sent over a path of length  $p$  in a period of time  $T$ , Marti's approach incurs a cost of  $jp$  receive messages.

The approach taken by Buchegger *et al.* uses Marti's watchdog mechanism apart from the trust manager, the reputation system and the path manager. Reporting detected attack to other watchdogs will cost  $p$  send messages and  $p$  receive messages. If  $\delta$  instances are detected in a period of time  $T$ , reporting local detections to other watchdogs will cost  $\delta p$  send messages and  $\delta p$  receive messages.

The technique proposed by Michiardi *et al.* also uses Marti's watchdog mechanism apart from the distributed cooperative reputation mechanism. Therefore, the cost is even higher. These kinds of approaches are too expensive to be suitable for resource-constrained WSNs.

Fig. 7 summarizes the costs for different techniques in terms of the number of messages sent and received for

detecting packet dropping over a path of length  $p$  in a time period  $T$ .

Techniques for detecting packet-dropping attacks	Overhead (in time period $T$ )
DPDSN	$3p$ send and $3p$ receive messages
Huang and Lee [HL03]	$jp$ send and $jp$ receive messages
Marti <i>et al.</i> [M01]	$jp$ receive messages.
Buchegger <i>et al.</i> [BB02]	$(jp + \delta p)$ receive and $\delta p$ send messages
Michiardi <i>et al.</i> [MM02]	$(jp + \delta p)$ receive and $\delta p$ send messages + the cost for reputation mechanism

Fig. 7: Comparison of techniques for detecting packet dropping.

*D Experimental Results on DPDSN Overhead*

In order to further validate our approach, we wrote a simulation program in C program to simulate the following communication scenario. The simulated WSN consisted of 100 nodes that were placed randomly in an area 100m by 100m. Base station is situated in one corner. The base station is the message source in all cases.

We set the radio range of nodes to 12m because this assures that each node has approximately 6 neighbors (based on random placement of nodes), which is beneficial for communication as discussed in Section III.C. For MICA2 mote sensors [MM04], sending a packet consumes 81 mW energy, whereas receiving a packet consumes 30 mW energy [MV05].

We varied the path length from 3 to 13 hops. We randomly selected malicious nodes. We assumed that only malicious nodes drop packets. The alternate path was used to verify the number of packets sent over the original path.

Based on simulation results, we calculated the ratio of overhead of DPDSN to the cost of discovering path using DSR and Directed Diffusion. Fig. 8 and Fig. 9 show the overhead for DSR and Directed Diffusion. We found that the overhead for DPDSN is proportional to the path length. This observation is consistent with the analytical results shown in Fig. 7 and Lemma 2. This observation is also validated by Fig. 4 and Fig. 5, which plot the overhead for different values of  $R_x/T_x$ . The overhead is independent of the number of packets sent in a given period of time. Figure 10 shows the probability of success of DPDSN.

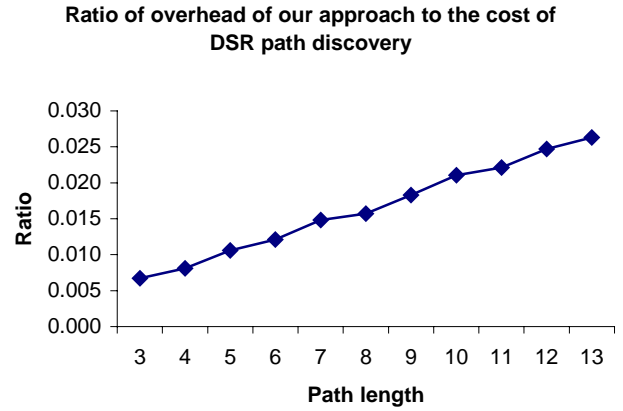


Fig. 8: Overhead for DSR.

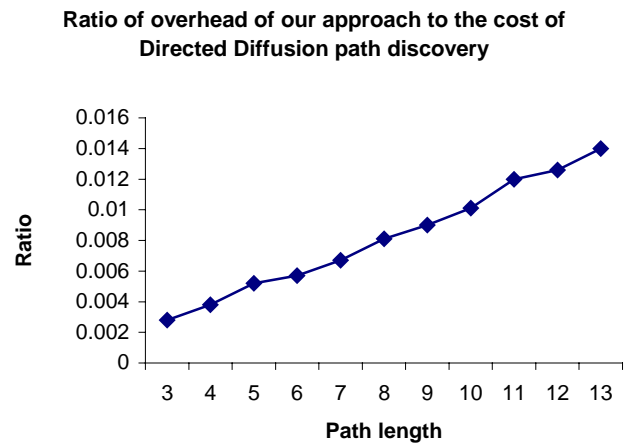


Fig. 9: Overhead for Directed Diffusion.

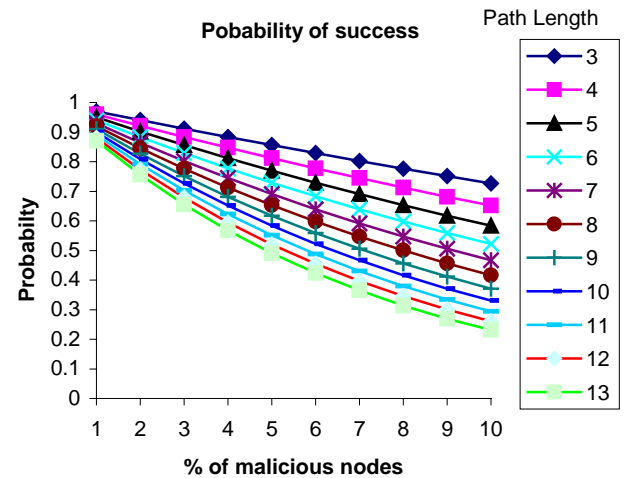


Fig. 10: Probability of success of DPDSN (for a path length 3 to 13 for a network of 100 nodes in the presence of 1 to 10 percent of packet-dropping nodes).

## V ALTERNATIVE APPROACHES TO DETECTION OF PACKET-DROPPING ATTACKS

We considered two other approaches for detecting packet-dropping attacks.

The first approach requires the destination being informed beforehand how many packets the source will send to it next. If an adversary drops *all* packets including the initial packet (with the count of packets to be transmitted), then the adversary will not only succeed in dropping all the packets but will go undetected as well. Including sequence numbers in packets or guarantees of authenticity and integrity of packets will not help. Even if assuring authentication and integrity could help, use of computationally intensive security primitives (such as message authentication codes, digital signatures, etc.) is not feasible for resource-constrained WSNs.

The following second approach can be used for detecting packet dropping but only for bi-directional communication. A source that does not receive a reply from its destination can guess that packets are being dropped by an attacker (unless it knows of other known reasons—like collisions, congestion, buffer overflow etc.—for packets being dropped). The basic idea is as follows. An intermediate node that forwards a packet to the next node on the path but does not receive a reply within a timeout period guesses that its neighbor is dropping packets. It informs the source about misbehavior of the neighbor. (Under our assumption of no packet infusion, no false “reply” is sent by the neighbor. We have a technique to make this assumption a reality [BG05].) The node that detects an adversary neighbor can choose another neighbor to reach the destination. This approach will not work for a unidirectional communication, since there are no replies. In contrast, DPDSN works for both uni- and bidirectional communication.

We plan to investigate the second approach in the future.

## VI CONCLUSION

Recent techniques for detection of packet-dropping nodes in ad hoc networks incur heavy costs and are not suitable for resource-constrained WSNs. DPDSN detects whether a *path* is dropping packets. Overhead for a two-dimensional grid of  $N$  nodes is  $O(\sqrt{N})$  packets, where  $N$  is the number of nodes. The cost is independent of the number of packets transmitted and the number of packet-dropping attacks being detected.

DPDSN incurs no cost for the response following the detection of a packet-dropping attack because the alternate path, established during route discovery, is ready for the response. The cost of finding an alternate path is the overhead paid during the route discovery, as shown in Fig. 8 and Fig. 9. We have shown that DPDSN works for DSR and Directed Diffusion.

Our simulations show that the overhead of DPDSN for WSN employing of MICA2 motes, for a path of length 3 to 13, is approximately 0.6% to 2.6% of the energy the network consumes on DSR path discovery if the DSR protocol is used for routing. Similar overhead for Directed Diffusion is approximately 0.3% to 1.4%.

## FUTURE WORK

Our future work will involve extending DPDSN by efficiently finding multiple alternate paths to increase the probability of success, and investigating other approaches to detect packet dropping (as proposed in Section V). We also propose to find solutions for detecting other DoS attack, including as Sybil attacks and masquerading.

## ACKNOWLEDGEMENT

We thank anonymous reviewers for their very helpful comments.

This research is supported in part by the National Science Foundation, under grants ACI-0000442, ACI-0203776, and MRI-0215356; by the Department of Education grant R215K020362; and by the Fund for the Improvement of Education, a Congressional Award administered by the US Department of Education. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies or institutions.

## REFERENCES

- [BB02] Buchegger, S. and Le Boudec, J., “Performance Analysis of the CONFIDANT Protocol: Cooperation of Nodes - Fairness in Dynamic Ad-hoc Networks”, Proc. 13th IEEE/ACM Symp. on Mobile Ad Hoc Networking and Computing (MobiHoc), Lausanne, Switzerland, June 2002.
- [BG05] Bhuse, V. and Gupta, A., “Anomaly intrusion detection in Wireless Sensor networks”, accepted for publication in the *Journal of High Speed Networks*, 2005.
- [CK04] Chen, Z. and Khokhar, A., “Self Organization and Energy Efficient TDMA MAC Protocol by Wake Up For Wireless Sensor Networks”, Proc. First Annual IEEE Intl Conf. on Sensor and Ad Hoc Communications and Networks (SECON 2004), Santa Clara, CA, October 2004.
- [HL03] Huang, Y. and Lee, W., “A Cooperative Intrusion Detection System for Ad Hoc Networks,” ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '03), Fairfax, VA, October 2003.
- [IG00] Intanagonwivat, C., Govindan, R., and Estrin, D., “Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks”, Proc. 6th Annual Intl. Conf. on Mobile Computing and Networking (MobiCom'00), Boston, Massachusetts, August 2000, pp. 56-67.
- [JM96] Johnson, D. and Maltz, D., “Dynamic source routing in ad hoc wireless networks,” in *Mobile Computing*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.
- [JW04] Jourdan, D.B. and de Weck, O.L., “Layout Optimization for a Wireless Sensor Network using a Multi-Objective Genetic Algorithm”, IEEE Semiannual Vehicular Technology Conference, Milan, Italy, May 2004.
- [KW03] Karlof, C. and Wagner, D., “Secure Routing in Sensor Networks: Attacks and Countermeasures,” in *Ad Hoc Networks*, volume 1, issues 2--3 (Special Issue on Sensor Network Applications and Protocols), Elsevier, September 2003, pp. 293-315.
- [LP05] Lu, B. and Pooch U.W., “A Lightweight Authentication Protocol for Mobile Ad Hoc Networks,” Proc. Intl. Conf. on Information Technology: Coding and Computing

(ITCC'05) - Volume II, Las Vegas, Nevada April 2005, pp. 546-551.

- [M01] Marti, S., Giuli, T. J., Lai, K., and Baker, M., "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," Proc. 6th Annual Intl. Conf. on Mobile Computing and Networking (MobiCom'00), Boston, Massachusetts, August 2000, pp. 255-265.
- [MM02] Michiardi, P. and Molva, R., "CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks", Proc. IFIP 6<sup>th</sup> Joint Working Conference on Communications and Multimedia Security (CMS'02), Portorož, Slovenia, September 2002, pp. 107-122.
- [MM04] MICA2 Mote Datasheet, Crossbow Technology, San Jose, CA. Available at: [http://www.xbow.com/Products/Product\\_pdf\\_files/Wireless\\_pdf/6020-0042-05\\_A\\_MICA2.pdf](http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0042-05_A_MICA2.pdf). Last accessed in October 2005.
- [MV05] Miller, M. and Vaidya, N., "A MAC Protocol to Reduce Sensor Network Energy Consumption Using a Wakeup Radio", *IEEE Transactions on Mobile Computing*, May/June 2005, pp. 228-242.
- [YH02] Ye, W., Heidemann, J., and Estrin, D., "An Energy Efficient MAC Protocol for Wireless Sensor Networks", in Proc. IEEE INFOCOM 2002: The Conference on Computer Communications, New York, NY, June 2002, pp. 1567-1576.