Chapter 3

Discrete Event System Based IDS for Power Aware AODV

3.1 Introduction

Wireless Sensor Network (WSN) is basically a wireless network, comprised of a large number of sensor nodes which are densely deployed, small in size, lightweight and portable [113]. When two nodes want to communicate beyond the limit of their transmission range, then routing protocols are essential [114]. This is because in that case some critical decisions have to be made, such as, which is the optimal route from the source to the destination. Such decisions are important because the mobile nodes operate on some kind of battery power. Thus, it becomes necessary to transfer data with the minimal delay so as to waste less power. WSNs lack fixed infrastructure and so no dedicated routes exist. In WSNs every mobile node behaves as a router and is accountable for identifying and taking care of the routes. In addition, WSNs can be called autonomous without centralized administration. To keep up this kind of autonomy the routing protocol needs automatically to adjust to regular changes in the environment. The key goal of routing protocols is to discover enhanced routes and establish communication between arbitrary nodes within
Routing protocols can be classified into three categories namely, proactive, reactive and hybrid, depending on how the source finds a route to the destination [17]. In proactive protocols, all routes are computed before they are really needed, while in reactive protocols, routes are computed on demand. Hybrid protocols use a combination of these two ideas.

Ad-hoc On-demand Distance Vector (AODV) is a well-known reactive routing protocol used in WSNs [115]. The baseline or traditional routing protocols e.g., AODV were designed to meet the basic requirements i.e., discovering source destination routes, route maintenance etc. [1]. Performance of a protocol is measured in terms of parameters like throughput, Packet Delivery Ratio (PDR), end-to-end delay, power etc. [116, 117]. With the advent of better resources in the sensor nodes, like processing speed, memory etc., several enhancements to these basic protocols have been proposed [118], which achieve better PDR, throughput, battery life time, low latency, tolerance to route failures etc. Lots of work have been done on enhancement of the basic AODV [119] to improve the quality of service in terms of these parameters [2]. Power Aware AODV (PAW-AODV) [22], one of the most prominent enhancement of ADOV, improves the WSNs from the perspective of lifetime of nodes.

AODV is subject to several attacks like blackhole, wormhole, man-in-the-middle etc. [102, 43, 38]. Several Intrusion Detection Systems (IDSs) have been proposed which successfully detect these simple and existing attacks [120]. In this chapter we show that PAW-AODV is more venerable compared to AODV, in the effort to reduce power. Such attacks reduce the life time of the nodes instead of increasing them. To the best of our knowledge, we have shown for the first time that power related enhancement on AODV has improved its performance but at the same time made it more vulnerable. Simple blackhole attack like “vampire blackhole” [9] can be launched against PAW-AODV and existing IDSs cannot detect the same. Following that we propose a Discrete Event System
(DES) based IDS for this advanced protocol. Network Simulator (NS-2) based simulation results illustrate that PAW-AODV augmented with the IDS achieves better performance compared to both the traditional AODV and PAW-AODV without the IDS. Thus, it could be shown that the use of appropriate IDS along with PAW-AODV can help to retain the improved performance achieved due to the enhancement of AODV.

The chapter is organized as follows. Section 3.2 presents the enhancements of AODV routing protocol for power aware routing i.e., PAW-AODV. In the same section we illustrate the vulnerabilities arising out of the enhancement. The preliminaries of DESs are discussed in Section 3.3. Section 3.4 describes the proposed DES based IDS which can detect power aware vampire attack on the power aware AODV. Section 3.5 describes the working of the IDS using an example. Section 3.6 presents the performance evaluation of PAW-AODV and compares it with that of traditional AODV. Also, efficacy of using the proposed IDS with PAW-AODV is justified using experimental results. The chapter is concluded in Section 3.7.

3.2 Preliminaries and background

3.2.1 AODV

AODV is a well-known reactive and stateless routing protocol used in WSNs. Reactive routing protocols create and maintain routes only on demand. That is, routes between the nodes are built only when the source node desires. Reactive protocols usually use distance-vector routing algorithms. One unique feature of AODV is its use of a destination sequence number ($\text{Dst}_SN$) for each route entry. The $\text{Dst}_SN$ is created by the destination node. This $\text{Dst}_SN$ and the routing table determine the freshness of the routes. An important feature of AODV is that a routing entry not recently used is expired. But route table information must be kept even for short-lived routes. Only active routes are allowed
to forward data packets. AODV uses mainly three control packets:

1. Route Request Message (RREQ) is broadcasted by a node requiring a route to another node,

2. Route Reply Message (RREP) is unicasted back to the source of RREQ,

3. Route Error Message (RERR) is sent to notify other nodes of the loss of a link.

![Figure 3.1: Example to illustrate AODV scenario](image)

AODV uses periodic HELLO messages to inform the neighbors that the link is still alive. In AODV, when any originator or source node tries to establish a path to a destination node, it broadcasts a RREQ message with a unique request ID to all its neighbor nodes. Broadcasting means transmitting to the IP limited broadcast address, i.e., 255.255.255.255. After receiving the RREQ a node may unicast a RREP with information regarding the path to the destination. This node may be the destination node or may be an intermediate node. If it is an intermediate node then it has a path to the destination. On the other hand if the intermediate node does not have a path to the destination, it rebroadcasts the RREQ message. If a node receives a RREQ which it has already processed, it does not forward it again and discards the packet.

In AODV, the Sequence Number (SN) plays an important role to create a loop free route. $Dst_{SN}$ is the latest sequence number received in the past by the originator for
any route towards the destination. If the source node is unaware about this \( Dst_{SN} \), then it sets zero (unknown) as its value. Similarly, source sequence number (\( Src_{SN} \)) or originator sequence number is the current sequence number, used in the route discovery using RREQ packet. Source (Destination) sequence number is increased when the source node initiates RREQ (when the destination node replies with RREP).

**Table 3.1:** Values of RREQ packets

<table>
<thead>
<tr>
<th>Attributes of Packet</th>
<th>RREQ</th>
<th>RREP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Im_Src_IP</td>
<td>S</td>
<td>I_1</td>
</tr>
<tr>
<td>Im_Dst_IP</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>HC</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dst_IP</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>Dst_SN</td>
<td>0 (Unknown)</td>
<td></td>
</tr>
<tr>
<td>Src_IP</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>Src_SN</td>
<td>289</td>
<td></td>
</tr>
<tr>
<td>RREQ_ID</td>
<td>31</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.1 illustrates the flow of the RREQ and RREP messages in a scenario wherein a node S wants to find a route to a node D. (Initially, nodes S, I\_1, I\_2, I\_3 and D do not have routes to each other). In the figure, S is a source node or originator and it wants to establish a route to the destination node D. Node S broadcasts a RREQ message (a1), which reaches node I\_1 (and all one hop neighbors of node S). Node I\_1 then re-broadcasts the request packet (a2). Another intermediate node I\_2 receives the message and again broadcasts the message (a3), which arrives at the destination node D through another intermediate node I\_3. After receiving the RREQ packet (a4), destination node D unicasts back the RREP message to S through the intermediate nodes I\_3, I\_2 and I\_1. We call these
RREQ and RREP packets as a request-reply flow.

Table 3.1 illustrates the various parameters for the set of RREQ packets for the example shown in Figure 3.1. First, we consider the RREQ packet a1. The parameter *Src_IP* (i.e., IP address of source node) is S. As the packet a1 originates from S, *Im_Src_IP* (i.e., IP address of immediate source) is also S. As the RREQ is a broadcast packet, so its *Im_Dst_IP* (i.e., IP address of immediate destination) is 255. As the a1 has originated from S, so Hop Count (HC) is 0.

<table>
<thead>
<tr>
<th>Attributes of Packet</th>
<th>RREQ</th>
<th>RREP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet</td>
<td>a1</td>
<td>a2</td>
</tr>
<tr>
<td>Im_Src_IP</td>
<td>A</td>
<td>I₁</td>
</tr>
<tr>
<td>Im_Dst_IP</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>HC</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dst_IP</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>Dst_SN</td>
<td>0 (Unknown)</td>
<td>303</td>
</tr>
<tr>
<td>Src_IP</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Src_SN</td>
<td>289</td>
<td></td>
</tr>
<tr>
<td>RREQ_ID</td>
<td>31</td>
<td></td>
</tr>
</tbody>
</table>

The destination being searched is D; so *Dst_IP* (IP address of destination node) is D. As we assume that in the past no path to D has been established so, *Dst_SN* is 0 (Unknown). *Src_SN* (source sequence number) and *RREQ_ID* (request ID) for all RREQ packets should be the same and the value of *Src_SN* and *RREQ_ID* are assumed to be 289 and 31, respectively. Next RREQ packet is a2. For the packet a2, values of *Src_IP*, *Dst_IP*, *Src_SN*, *Dst_SN* and *RREQ_ID* are same as the values of its previous RREQ packet a1. Packet a1 is broadcasted from originator S and received by one of its neighbor node I₁. As
I_1 does not have information about the destination D, I_1 re-broadcasts a1 as a2, making the 
HC of a2 as 1. Im_Src_IP of the packet a2 is I_1, as a2 is initiated by the intermediate node 
I_1. As in the case of a1, Im_Dst_IP of a2 is 255. The packet a2 is received by I_2, which is 
the one hop neighbor of I_1. Also I_2 does not have any route to the destination. Therefore, 
node I_2 again broadcasts the request packet. In this manner request packet a4 is received 
by the destination node D from the originator S through the intermediate nodes I_1, I_2 and 
I_3. After receiving the request packet, destination node D generates a RREP packet b1. As 
the packet b1 originates from D, Im_Src_IP of RREP packet b1 is D. RREP packet (b1) is 
unicast by node D and b1 is received by the next intermediate node I_3, so its Im_Dst_IP is 
I_3. Hop count of b1 is definitely 0, because it just starts from the destination node D. I_3 is 
not that node which originated the RREQ packet, so I_3 forwards this reply packet b1 with 
a new name b2. Now HC of b2 is 1 and it is received by I_2. So Im_Src_IP and Im_Dst_IP of 
packet b2 are I_3 and I_2, respectively. The node I_2 sends this reply packet as b3 to the next 
node I_1 and I_1 again unicasts this RREP packet to the source node S, thereby establishing 
the path from S to D. We assume that the Destination SN initiated by D for this sequence 
of RREP packets is 303. As intermediate nodes do not change the value of Dst_SN, the 
value of Dst_SN is fixed (i.e., 303) for b1, b2, b3, b4. Src_IP and Dst_IP for this particular 
RREP flow are S and D, respectively. Finally, a path is established between node S and 
D. Now S can send its data packets to node D through the intermediates nodes I_1, I_2 and 
I_3. Table 3.2 presents the complete details of the request-reply flow in the example being 
considered.

3.2.2 Power aware AODV (PAW-AODV)

In the previous section, we have described the traditional AODV protocol. Now we 
present PAW-AODV [22], which is an extension of the AODV routing protocol from the 
context of balancing overall power of the WSN. In addition with the basic parameters
of AODV described in Sub-section 3.2.1, two new parameters are added in PAW-AODV: \textit{Prev\_Src\_Ip} and \textit{Min\_Pow}. \textit{Prev\_Src\_Ip} represents the IP address of the previous node of immediate source node of a RREQ packet and \textit{Min\_Pow} indicates the minimum remaining battery power of the route.

![Figure 3.2: Example to illustrate PAW-AODV](image)

PAW-AODV is an extended version of AODV, which considers a cost function based on the availability of the battery power. Here, the cost function of the overall route is the sum of the cost functions of the individual nodes along the route. The aim of PAW-AODV is to apply an algorithm on AODV that can find a route with the least cost.

Next, we explain PAW-AODV with an example as shown in Figure 3.2. In the figure, the number beside each node denotes the remaining battery power; e.g., node S has remaining battery power as 100 unit. S, I_1, I_2, I_3, I_4, D are some nodes of the WSN. The originator S searches for a route to the node D. Node S broadcasts an RREQ packet as a1, a3 and a6. First, we consider the RREQ packet a1. The value of the parameter \textit{Src\_IP} of a1 is S. As the packet a1 originates from S, \textit{Im\_Src\_IP} of this packet is also S. As the RREQ is a broadcast packet, its \textit{Im\_Dst\_IP} is 255. a1 has originated from S, so HC is 0. Since S
is the originator, a1 displays the remaining power of node S. So, the value of $Min_{Pow}$ is 100. After receiving that RREQ packet, the intermediate node $I_1$ re-broadcasts the RREQ packet as a2.

<table>
<thead>
<tr>
<th>Attributes of Packet</th>
<th>RREQ of Packet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet</td>
<td>a1</td>
</tr>
<tr>
<td>Im_Src_IP</td>
<td>S</td>
</tr>
<tr>
<td>Prev_Src_IP</td>
<td>-</td>
</tr>
<tr>
<td>Im_Dst_IP</td>
<td>255</td>
</tr>
<tr>
<td>HC</td>
<td>0</td>
</tr>
<tr>
<td>Min_Pow</td>
<td>100</td>
</tr>
<tr>
<td>Dst_IP</td>
<td>D</td>
</tr>
<tr>
<td>Dst_SN</td>
<td>0 (Unknown)</td>
</tr>
<tr>
<td>Src_IP</td>
<td>S</td>
</tr>
<tr>
<td>Src_SN</td>
<td>952</td>
</tr>
<tr>
<td>RREQ_ID</td>
<td>45</td>
</tr>
</tbody>
</table>

For the packet a2, values of $Src_{IP}$, $Dst_{IP}$, $Src_{SN}$, $Dst_{SN}$ and $RREQ_{ID}$ are same as the values of the previous RREQ packet a1. As $I_1$ does not have information about the destination D, $I_1$ re-broadcasts a1 as a2, making the $HC$ of a2 is 1. $Im_{Src_{IP}}$ of the packet a2 is $I_1$, as a2 is initiated by the intermediate node $I_1$ and $Prev_{Src_{IP}}$ of a2 is S. Like a1, $Im_{Dst_{IP}}$ of a2 is 255. The remaining battery power of node $I_1$ is 80. Remaining battery power of node $I_1$ is less than that of node S. So, the value of $Min_{Pow}$ of a2 is 80. Packet a2 is received by the destination D. Table 3.3 illustrates the various parameters for the set of RREQ packets (i.e., a1,a2) for the route SI1D.

It may be noted that, unlike traditional AODV, in PAW-AODV, more than one path
from S to D is discovered because the intermediate nodes do not drop multiple RREQs with same source IP and Request ID.

<table>
<thead>
<tr>
<th>Attributes of Packet</th>
<th>RREQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet</td>
<td>a3</td>
</tr>
<tr>
<td></td>
<td>a4</td>
</tr>
<tr>
<td></td>
<td>a5</td>
</tr>
<tr>
<td>Im_Src_IP</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>I2</td>
</tr>
<tr>
<td></td>
<td>I4</td>
</tr>
<tr>
<td>Prev_Src_IP</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>I2</td>
</tr>
<tr>
<td>Im_Dst_IP</td>
<td>255</td>
</tr>
<tr>
<td></td>
<td>255</td>
</tr>
<tr>
<td></td>
<td>255</td>
</tr>
<tr>
<td>HC</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Min_Pow</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>80</td>
</tr>
<tr>
<td>Dst_IP</td>
<td>D</td>
</tr>
<tr>
<td>Dst_SN</td>
<td>0 (Unknown)</td>
</tr>
<tr>
<td>Src_IP</td>
<td>S</td>
</tr>
<tr>
<td>Src_SN</td>
<td>952</td>
</tr>
<tr>
<td>RREQ_ID</td>
<td>45</td>
</tr>
</tbody>
</table>

The paths that are generated for this example are as follows:

1. SI₁D
2. SI₂I₄D
3. SI₃D

When the originator S broadcasts the RREQ packet, intermediate nodes I₂ and I₃ also receive the packet. Since I₂ does not have any active path to the destination node, I₂ accepts the RREQ packet a3 and re-transmits it as a4. I₂ also receives another RREQ packet (a8) from node I₁. But I₂ observes that the values of Src_IP, Dst_IP and RREQ_ID are same in both these packets a3 and a8; so, I₂ discards a8. As explained for the route SI₁D, in a similar manner we can complete the explanation for the route SI₂I₄D. Table 3.4 illustrates
3.2. Preliminaries and background

the various parameters for the set of RREQ packets (i.e., a3, a4, a5) for the route SI2I4D. Table 3.5 illustrates the various parameters for the set of RREQ packets (i.e., a6, a7) for the route SI3D.

Table 3.5: Values of RREQ for packets a6, a7-path SI3D

<table>
<thead>
<tr>
<th>Attributes of Packet</th>
<th>RREQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet</td>
<td>a6</td>
</tr>
<tr>
<td>Im_Src_IP</td>
<td>S</td>
</tr>
<tr>
<td>Prev_Src_IP</td>
<td>-</td>
</tr>
<tr>
<td>Im_Dst_IP</td>
<td>255</td>
</tr>
<tr>
<td>HC</td>
<td>0</td>
</tr>
<tr>
<td>Min_Pow</td>
<td>100</td>
</tr>
<tr>
<td>Dst_IP</td>
<td>D</td>
</tr>
<tr>
<td>Dst_SN</td>
<td>0 (Unknown)</td>
</tr>
<tr>
<td>Src_IP</td>
<td>S</td>
</tr>
<tr>
<td>Src_SN</td>
<td>952</td>
</tr>
<tr>
<td>RREQ_ID</td>
<td>45</td>
</tr>
</tbody>
</table>

Remaining battery power of originator S is 100. So the packet a1 starts with the value 100 for Min_Pow. But the remaining power of intermediate node I2 is 80. RREQ packets of PAW-AODV always exhibit the minimum remaining power of the route. Therefore, the packet broadcasted by I2 (i.e., a2) carries 80 as the value of Min_Pow. So the Min_Pow of the route SI1D is 80 and is noted by the destination D. In a similar way the Min_Pow of the other two routes can be calculated and are presented below.

1. SI2I4D – 20
2. SI3D – 70

As the Min_Pow of the route SI1D is the maximum, the destination node D selects it for
rout. After selecting the route, the destination D sends a RREP packet to the originator S.

Table 3.6: Values of RREQ and RREP for SI₁D

<table>
<thead>
<tr>
<th>Attributes of Packet</th>
<th>RREQ</th>
<th>RREP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet a₁ a₂ b₁ b₂</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PreV_Src_IP S I₁ D I₁</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Im_Dst_IP 255 255 I₁ S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HC 0 1 0 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dst_IP D D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dst_SN 0 (Unknown) 1011</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Src_IP S S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Src_SN 952</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RREQ_ID 45</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Packet b₁ is unicasted from the destination D to the intermediate node I₁. Src_IP and Dst_IP of the packet b₁ is S and D, respectively. I₁ receives packet b₁ and unicasts it to the originator node S as b₂. Therefore, a path is established between the node S and D. Now, the node S can send its data packets to D through the intermediate node I₁. This request reply flow along the path SI₁D is described in Table 3.6.

3.2.3 Power aware vampire attack on PAW-AODV

PAW-AODV enhances WSNs from the perspective of lifetime of nodes (in terms of power). However, PAW-AODV can be easily affected by the attackers using the reverse of the basic philosophy used to select nodes having highest remaining power, i.e., selecting paths having nodes with least remaining battery power, thereby reducing the lifetime of
3.2. Preliminaries and background

some nodes. Power aware vampire attack on PAW-AODV [22] is a prominent attack that works on this philosophy. Figure 3.3 illustrates an example of the power aware vampire attack on PAW-AODV. In Figure 3.3, we again consider the same situation as of Figure 3.1. However, in this case there is a malicious node M between the node I₃ and D. Details of RREQ and RREP packets of Figure 3.3 are shown in Table 3.7.

![Diagram](https://via.placeholder.com/150)

**Figure 3.3:** Illustration of an attack against PAW-AODV

In this example, RREQ packet reaches D from the originator S, via I₃ and M. The value of cost function should be 70 due to low remaining power of I₃, however, M makes it 200 maliciously. This erroneously shows D that the path SI₃MD can be taken in the perspective of remaining battery life time. It may be easily noted that, this path would lead to depletion of power of node I₃ much faster than expected. To address this issue, we propose a Discrete Event System (DES) based IDS which can detect attacks that work by maliciously elevating the value of availability of the battery power of the nodes. Before we discuss the IDS, we present the basics of DES in the next section.
Table 3.7: Details of RREQ and RREP packets of Figure 3.3

<table>
<thead>
<tr>
<th>Attributes of Packet</th>
<th>RREQ 1</th>
<th>RREQ 2</th>
<th>RREQ 3</th>
<th>RREQ 4</th>
<th>RREP via M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet</td>
<td>a3</td>
<td>a4</td>
<td>a5</td>
<td>a1</td>
<td>a2</td>
</tr>
<tr>
<td>Im_Src_IP</td>
<td>S</td>
<td>I_2</td>
<td>I_3</td>
<td>S</td>
<td>I_1</td>
</tr>
<tr>
<td>Im_Dst_IP</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>HC</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Min_Pow</td>
<td>100</td>
<td>80</td>
<td>80</td>
<td>100</td>
<td>70</td>
</tr>
<tr>
<td>Dst_IP</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>Dst_SN</td>
<td>0 (Unknown)</td>
<td>0 (Unknown)</td>
<td>0 (Unknown)</td>
<td>0 (Unknown)</td>
<td>1011</td>
</tr>
<tr>
<td>Src_IP</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Src_SN</td>
<td>952</td>
<td>952</td>
<td>952</td>
<td>952</td>
<td>952</td>
</tr>
<tr>
<td>RREQ_ID</td>
<td>45</td>
<td>45</td>
<td>45</td>
<td>45</td>
<td>45</td>
</tr>
</tbody>
</table>

3.3 Preliminaries on DES modeling and failure diagnosis

In this section, we reproduce the formalism of DES framework [121, 122] that would be used to detect the power aware vampire attack on PAW-AODV. The DES model $G$ is defined as:

$$G = \langle X, X_0, \Sigma, \mathcal{I}, V \rangle$$  \hspace{1cm} (3.1)

- $X$ is the finite set of states.
- $X_0$ is the initial state.
- $\Sigma$ is the set of events. Note that an event can be measurable or unmeasurable.
- $\mathcal{I}$ is the set of transitions.
- $V$ is the set of model variables. Each element $v_i$ of $V$ can take values from a domain.
A transition $\tau \in \mathcal{I}$ is defined as a five-valued tuple $\langle x, x^+, \sigma, check(v), assign(v) \rangle$, where

- $x$ is the initial state of the transition denoted as $initial(\tau)$.
- $x^+$ is the final state of the transition denoted as $final(\tau)$.
- $\sigma \in \Sigma$ is an event on which the transition $\tau$ is enabled.
- $check(V)$ represents the conditions on a subset of the model variables. For firing $\tau$, along with the enabling event $\sigma$, $check(V)$ should hold true.
- $assign(V)$ represents a subset of the model variables and assignment of the values from their corresponding domain, when $\tau$ fires.

A trace of the model $G$ is a sequence of transitions generated by $G$ denoted as $s = \langle \tau_1, \tau_2, \cdots, \tau_f \rangle$, where $initial(\tau_i+1) = final(\tau_i)$, for $i = 1$ to $(f - 1)$. We denote $initial(\tau_1)$ as $initial(s)$ and $final(\tau_f)$ as $final(s)$. The set of all traces generated by $G$ is the language of $G$, denoted as $L(G)$. Naturally, $L(G)$ is a subset of $\mathcal{I}^w$, where $\mathcal{I}^w$ is the set of all infinite sequences of $\mathcal{I}$. Any finite member of $L(G)$ is in $\mathcal{I}^*$, the Kleene closure of $\mathcal{I}$. The post language of $G$ after a trace $s$ is denoted as $L(G)/s = \{ t \in \mathcal{I}^* | st \in L(G) \}$.

### 3.3.1 Model under measurability

As mentioned, transition from a state in the model occurs on enabling of an event and holding of $check(V)$. It may be noted that, events may or may not be measurable. For example, event corresponding to “launching of an attack” is not measurable, while the event for “receipt of RREQ packet” is measurable. The model variables are stored in the system memory (i.e., IDS), so their values are always measurable. Hence, the notion of measurability is associated only with the events (i.e., $\Sigma$) of a transition and not with
the model variables. The events can be classified as measurable or unmeasurable. The event set \( \Sigma \) is divided as \( \Sigma = \Sigma_m \cup \Sigma_{um} \), where \( \Sigma_m \) is the set of measurable events and \( \Sigma_{um} \) is a set of unmeasurable events. Given such a partition of events, the transitions can also be partitioned into two sets, \( \mathcal{I}_m \) and \( \mathcal{I}_{um} \), of measurable and unmeasurable transitions, respectively, based on the corresponding events being measurable or unmeasurable. We now define measurement equivalent transitions and states as follows:

**Definition 3.1 (Measurement Equivalent Transitions and States):** Two transitions \( \tau_1 = (x_1, x_1^+, \sigma_1, \text{check}_1(V), \text{assign}_1(V)) \) and \( \tau_2 = (x_2, x_2^+, \sigma_2, \text{check}_2(V), \text{assign}_2(V)) \) are measurement equivalent, if \( \sigma_1 = \sigma_2 \) (same event), \( \text{check}_1(V) \equiv \text{check}_2(V) \) (same check over the same subset of variables in \( V \)) and \( \text{assign}_1(V) \equiv \text{assign}_2(V) \) (same subset of model variables with same assignment). If \( \tau_1 \equiv \tau_2 \), then the source states of the transitions are equivalent and so are the destination states, i.e., \( x_1 \equiv x_2 \) and \( x_1^+ \equiv x_2^+ \).

**Definition 3.2 Projection Operator:** A projection operator \( P : \mathcal{I}^* \rightarrow \mathcal{I}_m^* \) is defined as:
\[
P(\epsilon) = \epsilon, \text{ the null string, } P(\tau) = \tau \text{ if } \tau \in \mathcal{I}_m, \text{ } P(\tau) = \epsilon \text{ if } \tau \in \mathcal{I}_{um}, \text{ } P(s\tau) = P(s)P(\tau), \text{ where } s \in L(G), \tau \in \mathcal{I}. \text{ } P(s) \text{ is termed as the measurable trace corresponding to the trace } s.
\]

**Definition 3.3 (Measurement equivalence of traces):** Two traces \( s \) and \( s' \) are equivalent, denoted as \( sEs' \), if
\[
P(s) = (\tau_1, \tau_2, ..., \tau_f), P(s') = (\tau'_1, \tau'_2, ..., \tau'_f) \text{ and } (3.2)
\]
\[
\forall i(1 \leq i \leq f), \tau_i E \tau'_i.
\]

We loosely use the same symbol \( E \) to represent all these equivalence relations.
3.3. Preliminaries on DES modeling and failure diagnosis

3.3.2 Failure modeling

Each state $x$ is assigned a failure label, defined by a function:

$$C : x \rightarrow \{N\} \cup 2^{\{F_1, F_2, \ldots, F_k\}}, 1 \leq i \leq k$$ (3.3)

where $F$ stands for permanent failure status and $N$ stands for normal status.

**Definition 3.4 (Normal G-state):** A G-state $x$ is normal if $C(x) = \{N\}$. The set of all normal states is denoted as $X_N$.

**Definition 3.5 ($F_i$-G-state):** A G-state $x$ is failure state, or synonymously, a $F_i$-state, if $F_i \in C(x)$. The set of all $F_i$-states is denoted as $X_{F_i}$.

**Definition 3.6 (Normal-G-transition and trace):** A G-transition $\tau$ is called a normal G-transition if $x, x^+ \in X_N$, where $x = \text{initial}(\tau)$ and $x^+ = \text{final}(\tau)$. A G-trace $s$ is called normal-G-trace if all transitions in $s$ are normal-G-transitions.

**Definition 3.7 ($F_i$-G-transition and trace):** A G-transition $\tau$ is called a $F_i$-G-transition if $x, x^+ \in X_{F_i}$, where $x = \text{initial}(\tau)$ and $x^+ = \text{final}(\tau)$. A G-trace $s$ is called $F_i$-G-trace if all transitions in $s$ are $F_i$-G-transitions.

A transition $\tau = \langle x, x^+, \sigma, \text{check}(V), \text{assign}(V) \rangle$, where $C(x) \neq C(x^+)$, is called a failure causing transition indicating the first occurrence of some failure in the set $C(x^+) - C(x)$. For example, if a transition $\tau = \langle x, x^+, \sigma, \text{check}(V), \text{assign}(V) \rangle$ occurs where $C(x) = \{N\}$ and $C(x^+) = \{F_i\}$ then it implies that the fault $F_i$ has occurred. A failure $F_i$ causing transition is represented as $\tau_{F_i}$ and the set of all such transitions are represented as $\mathcal{I}_{F_i}$.

The objective of the failure diagnosis problem is to determine the occurrence of a failure $F_i$. If the event ($\sigma$) corresponding to $\tau_{F_i}$ is measurable, failure diagnosis is trivial. So the failure causing transitions are assumed to be unmeasurable. For such failure causing
transitions, \( \sigma \) is unmeasurable. As \( \sigma \) is unmeasurable, there are no checks or assignment for model variables.

Network attacks are similar to faults in a DES. The similarity is in the sense that they indicate deviation from the normal activity of the system and its occurrence is not directly measurable. As failures are assumed to be permanent, there is no transition from any state in \( x_F \) to any state in \( x_N \). The event related to causing \( F_i \) is denoted as \( \sigma_{F_i} \). Informally, a DES \( G \) is diagnosable if it is always possible to determine the fault status of the states beyond a certain point along all the possible traces of \( G \) after the occurrence of a fault, using the sequence of measurements. A few terms are introduced before formally defining diagnosability.

Let \( \psi(X_{F_i}) = \{s|s \in L_f(G) \text{ and } \text{final}(s) \in X_{F_i} \text{ and } s \text{ ends in a measurable transition}\} \).

**Definition 3.8** \( F_i \)-Diagnosability: A DES \( G \) is said to be diagnosable under a measurement limitation for fault \( F_i \) iff the following holds.

\[
(\exists n_j \in \mathbb{N})[(\forall s \in \psi(X_{F_i}))(\forall t \in L_f(G)/s)[|t| \geq n_j \Rightarrow D] \quad (3.4)
\]

where the condition \( D \) is

\[
\forall y \in \{P^{-1}[P(st)]\}, \text{final}(y) \in X_{F_i} \quad (3.5)
\]

The above definition means the following. Let \( s \) be any finite prefix of a trace of \( G \) that ends in a \( F_i \)-state and let \( t \) be any sufficiently long continuation of \( s \). Condition \( D \) then requires that every sequence of transitions, measurement indistinguishable with \( st \), shall end into a \( F_i \)-state. This implies that, along every continuation \( t \) of \( s \), one can detect the occurrence of fault \( F_i \) within a finite steps, specifically in atmost \( n_j \) transitions of the system after \( s \).

In the next subsection, we discuss the procedure to construct the diagnoser from the model and the condition to be checked for diagnosability analysis.
3.3.3 Diagnoser

The diagnoser is represented as a directed graph

\[ D = \langle Z, A \rangle \]  

where \( Z \) is the set of diagnoser states, called \( D\)-states, and \( A \) is the set of diagnoser transitions, called \( D\)-transitions. Henceforth, terms like transitions, states etc. of the DES are termed as model transitions, model states etc. to differentiate them from those of the diagnoser. Each \( D\)-state \( z \in Z \) is an ordered set comprising a subset of equivalent model states representing the uncertainty about the actual state. Each \( D\)-transition \( a \in A \) is a set of equivalent model transitions representing the uncertainty about the actual transition that occurs. \( initial(a) \) (\( final(a) \)) represents the source (destination) \( D\)-state of \( a \).

**Definition 3.9** Unmeasurable successor of a \( D\)-state \( z \) is defined as \( U(z) = \bigcup_{x \in z} \{ x^+ | \tau = \langle x, x^+ \rangle \in \mathcal{I}_{um} \} \).

The unmeasurable reach of \( z \) is the transitive closure (Kleene closure) of unmeasurable successors of \( z \) and is denoted as \( U^*(z) \).

The diagnoser is constructed starting from the initial state(s) of the system model \( G \). The initial \( D\)-state \( (z_0) \) is obtained first as \( U^*(X_0) \), where \( X_0 \) is the set of initial states of \( G \). Given any \( D\)-state \( z \), the transitions emanating from \( z \) are obtained as follows. Let \( \mathcal{I}_{mz} \) denote the set of measurable transitions defined from \( z \). Let \( A_z \) be the set of all equivalence classes of \( \mathcal{I}_{mz} \) under \( E \). For each \( a \in A_z \), a successor \( D\)-state \( z^+ \) of \( z \) such that \( z^+ = final(a) \) can be created in the following way. Let \( z^+_0 = \{ final(\tau) | \tau \in a \} \); then \( z^+ = U^*(z^+_0) \) and \( a = \langle z, z^+ \rangle \). So \( z^+ \) comprises of \( G\)-states that can be reached from \( G\)-states in \( z \) via measurable equivalent transitions comprised in \( a \). Further, new \( G\)-states are added to \( z^+ \) which can be reached from the already existing \( G\)-states in \( z^+ \) using unmeasurable transition sequence. The transition set of the diagnoser is augmented as \( A \leftarrow A \cup \{ a \} \) and
the state set is augmented as $Z \leftarrow Z \cup \{z^+\}$. The algorithm for diagnoser construction is given below.

**Algorithm 1**: Algorithm for construction of diagnoser $O$ for an DES model $G$

**Input**: DES model $G$

**Output**: DES Diagnoser

Partition $X_0$ into equivalent subsets $X_{01}, X_{02}, \ldots, X_{0m}$

for all $i, 1 \leq i \leq m$ do

$z_{0i} = \mathcal{U}^r(X_{0i})$

end

$Z_0 \leftarrow z_{01} \cup \cdots \cup z_{0m}$

$Z \leftarrow Z_0$

$A \leftarrow \emptyset$

for all $z \in Z$ do

/* Find the set of measurable $G$-transitions ($\mathcal{G}_{mz}$) outgoing from $z$ */

$\mathcal{G}_{mz} \leftarrow \{\tau|\tau \in \mathcal{G}_m \land initial(\tau) \in z\}$

/* Find the set of all measurement equivalent classes $A_z$, of $\mathcal{G}_{mz}$ */

for all $a \in A_z$ do

$z_a^+ = \{final(\tau)|\tau \in a\}$

$z_a^+ = \mathcal{U}^r(z_a^+)$

$Z = Z \cup \{z_a^+\}$

$A = A \cup \{a\}$

end

end

3.3.3.1 Diagnosability analysis

Before discussing the condition to be checked for diagnosability analysis on the diagnoser, certain definitions and properties are introduced.

**Definition 3.10 (Embedding of $G$-traces in $D$-traces):** Given a $D$-trace $\gamma = \langle a_1, a_2, \ldots, a_k \rangle$, a $G$-trace $s$, where $P(s) = \langle \tau_1, \tau_2, \ldots, \tau_k \rangle$, is said to be embedded in $\gamma$, if $\tau_i \in a_i, 1 \leq i \leq k$. The set of all $G$-traces embedded in a $D$-trace $\gamma$ is represented as $A_D(\gamma)$.

**Property 3.1** If two traces $t, y \in A_D(\gamma)$, where $t$ is $F_I$-G-trace and $y$ is non-$F_I$-G-trace, then the $D$-states traversed by $\gamma$ are $F_I$-uncertain.
3.3. Preliminaries on DES modeling and failure diagnosis

**Proof 3.1** The property also follows from the diagnoser construction. As any D-transition $a \in \gamma$ has a non-$F_i$-G-transition and a $F_i$-G-transition (which are equivalent), so source and destination D-states of $a$ are $F_i$-uncertain.

The fault label of any D-state $z = (x_1, x_2, ..., x_i, ....)$ is defined as $C(z) = \bigcup_{x \in z} C(x)$.

**Definition 3.11 (Normal D-state):** A D-state $z$ is called normal and denoted as $z_N$, if $C(z) = \{N\}$; the set of all normal D-states is denoted as $Z_N$.

**Definition 3.12 ($F_i$-D-state):** A D-state $z$ is called a $F_i$-D-state and denoted as $z_{F_i}$, if $F_i \in C(z)$. The set of all $F_i$ D-states is denoted as $Z_{F_i}$.

**Definition 3.13 ($F_i$-certain D-state):** A $F_i$-D-state $z$ is called a $F_i$-certain D-state if $z \subseteq X_{F_i}$.

**Definition 3.14 ($F_i$-uncertain D-state):** A $F_i$-D-state which is not $F_i$-certain is called $F_i$-uncertain.

In words, $F_i$-certain D-state comprises only $F_i$-G states, while $F_i$-uncertain D-state comprises some $F_i$-G states and some non $F_i$-G-states. So, if the diagnoser reaches any $F_i$-certain D-state failure is diagnosed. By Property 3.1 and Definition 3.10, if there is a D-trace $\gamma$ which moves in $F_i$-uncertain D-states, there is a $F_i$-G-trace $t$ which is embedded in $\gamma$. After failure, diagnoser moves in $\gamma$ by virtue of $t$. However, as there is another non-$F_i$-G-trace $y$ say, equivalent to $t$, fault cannot be diagnosed till $\gamma$ is exited.

**Definition 3.15 $F_i$-indeterminate cycle:** A $F_i$-uncertain cycle $\gamma$ is called a $F_i$-indeterminate cycle if the $F_i$-states contained in the D-states of $\gamma$ form a cycle in $G$ comprising transitions from $\gamma$.

Thus, a $F_i$-indeterminate cycle is a $F_i$-uncertain cycle with the special property as stated below:

“There is also a cycle involving $F_i$-states of the system model $G$ that corresponds to the..."
In simple words, the existence of a $F_i$-indeterminate cycle in the diagnoser implies that there are at least two measurement equivalent syntactic cycles in $G$, one comprising only non-$F_i$-states and the other comprising $F_i$-states. This implies that if the system moves in a $F_i$-indeterminate cycle, then the measurable transitions are observed to be similar in both non-$F_i$ and $F_i$ conditions. Thus, if the diagnostic estimate moves along such a $F_i$-indeterminate cycle, then the fault $F_i$ cannot be diagnosed because, at each point in the cycle there exists uncertainty regarding the occurrence of $F_i$ and as faults are assumed to be permanent, the system may not exit from such a cycle. The existence of a $F_i$-indeterminate cycle thwarts diagnosability.

In the next section, we propose an IDS for the detection of power aware vampire attack on PAW-AODV which is based on the DES framework just discussed.

### 3.4 Proposed scheme: Discrete event system based IDS for power aware vampire attack on PAW-AODV

In this section, we first look into the architecture of the proposed DES based IDS. Following that, we present the construction of normal and fault (attack) model. DES diagnoser, which is instrumental for detection of the attack, is described next. Following that, we illustrate the detection of the attack using the DES diagnoser.

#### 3.4.1 Architecture of the IDS for detecting the attack on PAW-AODV

A sensor network is composed of a large number of sensor nodes. We divide the nodes into clusters and assign a node as Sensor Monitor (SM) for each cluster. The DES based IDS is executed in the SM. So all the incoming and outgoing network packets (AODV in this case) for the nodes falling in the cluster must reach its SM. In other words, the SM runs in promiscuous mode and it must be able to receive the transmission of all the
3.4. Proposed scheme: Discrete event system based IDS for power aware vampire attack on PAW-AODV packets to and from the nodes lying in its cluster. Thus, the size of a cluster depends on the range of its SM. It may be noted that clusters may be overlapping (and some nodes may be monitored by more than one SMs), however, it is ascertained that all nodes are monitored by some SM.

The block diagram for the DES based IDS of the attack on PAW-AODV is shown in Figure 3.4.

![Figure 3.4: Basic block diagram of proposed IDS](image)

The IDS is executed in the SMs of all the clusters. The components are described as follows:

- **Wi-Fi sniffer**: The Wi-Fi sniffer works in promiscuous mode and captures all network packets traveling in between the nodes in the cluster. Only when an AODV related packet is observed, it awakes the “event generator” module. The other packets are dropped and the event generator and DES diagnoser are in sleep mode. This scheme helps to minimize the energy consumption of the SM, where most of the time only the Wi-Fi sniffer is powered ON. Only when a packet relevant to the attack is received, the other two modules are put ON.

- **Event generator**: The event generator is responsible for extracting packet related information like source IP address, destination IP address, source node, destination node, remaining power etc. from the AODV traffic. The event generator component
is also responsible for the generation of events like `RREQ,RREP, Wake, Time Out` etc. The generated event(s) and the packet related information are sent to the DES diagnoser. The DES diagnoser determines whether the attack has occurred or not.

- **DES diagnoser**: The DES diagnoser is actually implemented as a software module and used as attack detector. The DES diagnoser forms the crux of the detection methodology and is constructed from the DES model under normal and attack conditions. The construction and uses of the diagnoser are described in Subsection 3.4.2.

### 3.4.2 DES based model of the normal and attack condition

The DES model $G$ used to represent PAW-AODV under normal and attack scenarios is shown in Figure 3.5. For readability purposes, Figure 3.5 is annotated with transition number $\tau_i$.

![DES model for normal and PAW-AODV under attack situation](image)

*Figure 3.5: DES model for normal and PAW-AODV under attack situation*
The transitions are explained in Table 3.8 and Table 3.9. States and transitions belonging to normal (attack) model are denoted by the non-primed (primed) notations. The various components of $G$ are as follows:

$X = \{s_0, s_1, s_2, s_3, s'_0, s'_1, s'_2, s'_3\}$

$X_0 = \{s_0\}$

$\Sigma = \{\text{Wake}, \text{RREQ}, \text{RREP}, \text{Time out}\}$

$\mathcal{I} = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7, \tau_{1F}, \tau'_1, \tau'_2, \tau'_3, \tau'_5, \tau'_6, \tau'_7, \tau'_8\}$.

$V = \{\text{Im\_Src\_IP}, \text{Im\_Dst\_IP}, \text{Src\_IP}, \text{Dst\_IP}, \text{Prev\_Src\_IP}, \text{RREQ\_ID}, \text{Min\_Pow}\}$ is the set of model variables. The domain of $\text{Im\_Src\_IP}, \text{Im\_Dst\_IP}, \text{Src\_IP}, \text{Dst\_IP}$ is $\{x.x.x.x|x \in [0-255]\}$. $\text{Im\_Src\_IP}$ and $\text{Im\_Dst\_IP}$ contain immediate source and destination IP addresses, respectively, of the RREP or RREQ packets. $\text{Src\_IP}$ holds source IP address of a route while $\text{Dst\_IP}$ holds the destination IP address of the route. $\text{Prev\_Src\_IP}$ contains the IP address of the previous node of immediate source node of a RREQ/RREP packet. The domain of $\text{RREQ\_ID}$ and $\text{Min\_Pow}$ is $\mathbb{P}$ (positive integer). The former represents the route request ID and the latter represents the remaining power of a node, sent through RREQ packet. The values of these fields are present in the RREP and RREQ packets.

- **Behavior under normal conditions:**

States $\{s_0, s_1, s_2, s_3\}$ and transitions $\{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7, \tau_8\}$ represent the system under normal condition.

1. ($s_0$) The model starts at state $s_0$. Event generator and DES diagnoser are in sleep mode. Wi-Fi Sniffer is monitoring packets and to determine if any AODV related packet is captured.

2. $\tau_1 : (s_0 \rightarrow s_1)$ The Wi-Fi sniffer receives a RREQ packet and so the event generator module and the DES diagnoser need to be started. The event generator generates the event $\text{Wake}$. There is no check or assignment of model variables in this transition. Therefore, $\text{check}(V)$ and $\text{assign}(V)$ are "-", "-" in the transition. The initial $(\tau_1) \equiv s_0$,
3. \( \tau_2 : (s_1 \rightarrow s_0) \) After a certain interval of time, if the Wi-Fi sniffer does not receive any AODV packets, it puts the event generator and the DES diagnoser to sleep mode. Here, the initial \( (\tau_2) \equiv s_1, \sigma \equiv \text{Wake} \). Here, \( \text{check}(V) = \{ - \} \) and \( \text{assign}(V) = \{ - \} \).

4. \( \tau_3 : (s_1 \rightarrow s_2) \) In state \( s_1 \), RREQ packet is received by the Wi-Fi sniffer and the event generator generates the event “RREQ”. Also, the values related to \( \text{Im} \text{Src}_{\text{IP}}, \text{Prev} \text{Src}_{\text{IP}}, \text{Src}_{\text{IP}}, \text{Dst}_{\text{IP}}, \text{RREQ}_{\text{ID}} \) and \( \text{Min} \text{Pow} \) are generated. The values of these fields as generated by the event generator for the RREQ packet are designated as \( \text{Im} \text{Src}_{\text{IP}}^{\text{RREQ}}, \text{Prev} \text{Src}_{\text{IP}}^{\text{RREQ}}, \text{Src}_{\text{IP}}^{\text{RREQ}}, \text{Dst}_{\text{IP}}^{\text{RREQ}}, \text{RREQ}_{\text{ID}}^{\text{RREQ}} \) and \( \text{Min} \text{Pow}^{\text{RREQ}} \). Transition \( \tau_3 \) assigns the values of immediate source IP \( (\text{Im} \text{Src}_{\text{IP}}^{\text{RREQ}}) \), previous source IP \( (\text{Prev} \text{Src}_{\text{IP}}^{\text{RREQ}}) \), source IP \( (\text{Src}_{\text{IP}}^{\text{RREQ}}) \), request ID \( (\text{RREQ}_{\text{ID}}^{\text{RREQ}}) \) and minimum remaining power \( (\text{Min} \text{Pow}^{\text{RREQ}}) \) of the recent RREQ packet to the variables \( \text{Im} \text{Src}_{\text{IP}}, \text{Prev} \text{Src}_{\text{IP}}, \text{Src}_{\text{IP}}, \text{RREQ}_{\text{ID}} \) and \( \text{Min} \text{Pow} \), respectively. The model then reaches state \( s_2 \). So for \( \tau_3 \), initial state is \( s_1 \), final state is \( s_2 \), \( \sigma \equiv \text{RREQ}, \text{check}(V) = \{ - \} \), and \( \text{assign}(V) = \{ \text{Src}_{\text{IP}} \leftarrow \text{Src}_{\text{IP}}^{\text{RREQ}}, \text{Dst}_{\text{IP}} \leftarrow \text{Dst}_{\text{IP}}^{\text{RREQ}}, \text{RREQ}_{\text{ID}}^{\text{RREQ}} \leftarrow \text{RREQ}_{\text{ID}}, \text{Im} \text{Src}_{\text{IP}} \leftarrow \text{Im} \text{Src}_{\text{IP}}^{\text{RREQ}}, \text{Min} \text{Pow}^{\text{RREQ}} \leftarrow \text{Min} \text{Pow} \} \).

5. \( \tau_4 : (s_2 \rightarrow s_2) \) When an intermediate sensor node receives a RREQ packet, it matches

- Previous source IP of the RREQ \( (\text{Prev} \text{Src}_{\text{IP}}^{\text{RREQ}}) \) with value stored in the variable immediate source IP \( (\text{Im} \text{Src}_{\text{IP}}) \).
- Source IP of the RREQ \( (\text{Src}_{\text{IP}}^{\text{RREQ}}) \) with value stored in the variable source IP \( (\text{Src}_{\text{IP}}) \).
- Destination IP of the RREQ \( (\text{Dst}_{\text{IP}}^{\text{RREQ}}) \) with value stored in the variable destination IP \( (\text{Dst}_{\text{IP}}) \).
Table 3.8: Transition table for normal situation

<table>
<thead>
<tr>
<th>Transition (τ)</th>
<th>Initial State (s)</th>
<th>Final State (s')</th>
<th>Event</th>
<th>check(V)</th>
<th>assign(V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₁</td>
<td>s₀</td>
<td>s₁</td>
<td>Wake</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>τ₂</td>
<td>s₁</td>
<td>s₀</td>
<td>Time Out</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>τ₃</td>
<td>s₁</td>
<td>s₂</td>
<td>RREQ</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Src_IP ← Src_IP&lt;sub&gt;RREQ&lt;/sub&gt; Dst_IP ← Dst_IP&lt;sub&gt;RREQ&lt;/sub&gt; RREQ_ID ← RREQ_ID&lt;sub&gt;RREQ&lt;/sub&gt; Im_SRC_IP ← Im_SRC_IP&lt;sub&gt;RREQ&lt;/sub&gt; Min_Pow ← Min_Pow&lt;sub&gt;RREQ&lt;/sub&gt;</td>
</tr>
<tr>
<td>τ₄</td>
<td>s₂</td>
<td>s₂</td>
<td>RREQ</td>
<td>-</td>
<td>Min_Pow ← Min_Pow&lt;sub&gt;RREQ&lt;/sub&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Im_SRC_IP ← Prev_SRC_IP&lt;sub&gt;RREQ&lt;/sub&gt; Min_Pow ≥ Min_Pow&lt;sub&gt;RREQ&lt;/sub&gt;</td>
</tr>
<tr>
<td>τ₅</td>
<td>s₂</td>
<td>s₀</td>
<td>Time Out</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>τ₆</td>
<td>s₂</td>
<td>s₃</td>
<td>RREP</td>
<td>-</td>
<td>Im_SRC_IP ← Im_SRC_IP&lt;sub&gt;RREP&lt;/sub&gt; Im_Dst_IP ← Im_Dst_IP&lt;sub&gt;RREP&lt;/sub&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Dst_IP ← Dst_IP&lt;sub&gt;RREP&lt;/sub&gt; Im_Dst_IP ← Im_Dst_IP&lt;sub&gt;RREP&lt;/sub&gt;</td>
</tr>
<tr>
<td>τ₇</td>
<td>s₃</td>
<td>s₃</td>
<td>RREP</td>
<td>-</td>
<td>Im_SRC_IP ← Im_SRC_IP&lt;sub&gt;RREP&lt;/sub&gt; Im_Dst_IP ← Im_Dst_IP&lt;sub&gt;RREP&lt;/sub&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Src_IP ← Src_IP&lt;sub&gt;RREP&lt;/sub&gt; Im_SRC_IP ← Im_SRC_IP&lt;sub&gt;RREP&lt;/sub&gt;</td>
</tr>
<tr>
<td>τ₈</td>
<td>s₃</td>
<td>s₀</td>
<td>Time Out</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- Request ID of the RREQ (RREQ_ID<sub>RREQ</sub>) with value stored in the variable request ID (RREQ_ID).

The above points check if the current RREQ being broadcasted corresponds to the RREQ of transition τ₃. This transition is indicated by τ₄, where initial (τ₂) ≡ s₂, final (τ₃) ≡ s₂. Here, the event is RREQ and check(V) = \{Im_SRC_IP ≡ Prev_SRC_IP<sub>RREQ</sub>, Src_IP ≡ Src_IP<sub>RREQ</sub>, Dst_IP ≡ Dst_IP<sub>RREQ</sub>, RREQ_ID ≡ RREQ_ID<sub>RREQ</sub> \}. Along with the checks mentioned above, one of the very important checks that holds in this transition, is the key to detect the attack – Min_Pow ≥ Min_Pow<sub>RREQ</sub>. This verifies that, minimum remaining battery power reported by the current RREQ packet is less or equal to the value reported by the previous RREQ packet. τ₃ (last transition) records the value of remaining battery power reported by the last RREQ packet in the variable Min_Pow (← Min_Pow<sub>RREQ</sub>). Now, the value of minimum remaining battery power reported by the current RREQ packet must be lower than or equal
to $Min_{Pow}$; this is modeled by $\tau_4$ as $Min_{Pow} \geq Min_{Pow}^{RREQ}$. This modeling is to be repeated for all the RREQ packet sequences. So, in $\tau_4$ we overwrite the variable $Min_{Pow}$ with the minimum remaining battery power reported by the current RREQ; 

$$assign(V) = \{Min_{Pow} \leftarrow Min_{Pow}^{RREQ}\}.$$ 

Now, in the next iteration of the self loop, $\tau_4$ will compare the minimum remaining battery power reported by the next RREQ with $Min_{Pow}$ (minimum remaining battery power reported by the current RREQ). This goes on till the final RREQ packet reaches the source node.

6. $\tau_5 : (s_2 \rightarrow s_0)$ After reviving a RREQ packet, the Wi-fi sniffer waits for a certain interval of time and if it does not receive any RREP packet corresponding to the previous RREQ, it sends a time out frame. This is similar to $\tau_2$.

7. $\tau_6 : (s_2 \rightarrow s_3)$ The destination node (or some intermediate node having path to the destination node) responds to the RREQ packet with an unicast RREP packet to the source node. This is modeled by transition $\tau_6$, implying that Wi-fi sniffer has received a RREP packet, which is the response to the RREQ packet under consideration. Source IP address and destination IP address of the RREP packet remain same as that of the RREQ packet. So, to verify that the current RREP is corresponding to the RREQ sequence (i.e., transitions $\tau_1, \tau_3, \tau_4$), the conditions to be checked are given as $check(V)$ for $\tau_6$—$Src_{IP} \equiv Src_{IP}^{RREP}$ and $Dst_{IP} \equiv Dst_{IP}^{RREP}$. So $\tau_6$ can be represented as—initial state of $\tau_6 \equiv s_2$, final state of $\tau_6 \equiv s_3$, $\sigma \equiv RREP$, $check(V) = \{Src_{IP} \equiv Src_{IP}^{RREP}, Dst_{IP} \equiv Dst_{IP}^{RREP}\}$. Immediate source IP address and immediate destination IP address of the first RREQ packet are stored in the corresponding variables by $assign(V) = \{Im_{Src_{IP}} \leftarrow Im_{Src_{IP}}^{RREP}, Im_{Dst_{IP}} \leftarrow Im_{Dst_{IP}}^{RREP}\}$; these assignments would be used for correlation of the future RREQs in the sequence as explained in $\tau_7$.

8. $\tau_7 : (s_3 \rightarrow s_3)$ The RREP packet is now unicasted by the intermediate nodes to their
next hop neighbors, till it reaches the source node. This is modeled by self loop
transition $\tau_7$ from state $s_3$. Here $\text{check}(V) = \{ \text{Src}\_IP \equiv \text{Src}\_IP^{RREP}, \text{Dst}\_IP \equiv \text{Dst}\_IP^{RREP}, \text{Im}\_Dst\_Node \equiv \text{Im}\_Src\_Node^{RREP} \}$. The first two checks verify that the RREP packet
received by the sniffer belongs to the same RREQ sequence under consideration. The
third check verifies that the immediate source IP of the current RREP is same as that
of the immediate destination IP of the last RREP; this checks the transitive property
of two consecutive RREPs. To repeat this consecutive check we have $\text{assign}(V) = \{ \text{Im}\_Src\_IP \leftarrow \text{Im}\_Src\_IP^{RREP}, \text{Im}\_Dst\_IP \leftarrow \text{Im}\_Dst\_IP^{RREP} \}$. 

9. $\tau_8 : (s_3 \rightarrow s_0)$: This is a time out transition similar to $\tau_5$ and $\tau_2$.

**Behavior under attack conditions:**
States $\{s'_0, s'_1, s'_2, s'_3\}$ and transitions $\{ \tau'_1, \tau'_2, \tau'_3, \tau'_4, \tau'_5, \tau'_6, \tau'_7, \tau'_8 \}$ represent the system under
attack condition. For the attack scenario we will explain only those transition which are
different from the normal situation.

1. $\tau'_5 : (s'_2 \rightarrow s'_2)$: This transition is indicated by $\tau'_5$, where initial ($\tau'_5$) $\equiv s'_2$, final ($\tau'_5$) $\equiv s'_2$. Here, the event is RREQ and $\text{check}(V) = \{ \text{Im}\_Src\_IP \equiv \text{Prev}\_Src\_IP^{RREQ}, \text{Src}\_IP \equiv \text{Src}\_IP^{RREQ}, \text{Dst}\_IP \equiv \text{Dst}\_IP^{RREQ}, \text{RREQ}\_ID \equiv \text{RREQ}\_ID^{RREQ} \}$. 

   Along with the checks mentioned above, one of the very important checks that holds in this
   transition, that is the key to attack detection is $-\text{Min}\_Pow < \text{Min}\_Pow^{RREQ}$. This verifies
   that, minimum remaining battery power reported by the current RREQ packet is
greater than the value reported by the previous RREQ packet. $\tau'_3/\tau'_4$ (last transition)
records the value of remaining battery power reported by the last RREQ packet in
the variable $\text{Min}\_Pow$ ($\leftarrow \text{Min}\_Pow^{RREQ}$). Now, the value of minimum remaining
battery power reported by the current RREQ packet is greater than $\text{Min}\_Pow$ of the
malicious node; this is modeled by $\tau'_5$ as $\text{Min}\_Pow < \text{Min}\_Pow^{RREQ}$. This modeling
is to be repeated for all the RREP packet sequences where malicious node reports
inflated remaining battery power. So, in $\tau'_4$ we overwrite the variable $\text{Min}\_Pow$ with
the minimum remaining battery power reported by the current RREQ; \(assign(V) = \{Min_{Pow} \leftarrow Min_{Pow}^{RREQ}\}\). This procedure goes on till the final RREP packet reaches the source node.

### Table 3.9: Transition table for attack situation

<table>
<thead>
<tr>
<th>Transition ((\tau'_k))</th>
<th>Initial State ((x))</th>
<th>Final State ((x'))</th>
<th>Event ((o))</th>
<th>check((V))</th>
<th>assign((V))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\tau'_1)</td>
<td>(s'_0)</td>
<td>(s'_1)</td>
<td>Wake</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>(\tau'_2)</td>
<td>(s'_1)</td>
<td>(s'_0)</td>
<td>Time Out</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>(\tau'_3)</td>
<td>(s'_1)</td>
<td>(s'_2)</td>
<td>RREQ</td>
<td>–</td>
<td>(Min_{Pow} \leftarrow Min_{Pow}^{RREQ})</td>
</tr>
<tr>
<td>(\tau'_4)</td>
<td>(s'_2)</td>
<td>(s'_2)</td>
<td>RREQ</td>
<td>(Min_{Pow} \leftarrow Min_{Pow}^{RREQ})</td>
<td></td>
</tr>
<tr>
<td>(\tau'_5)</td>
<td>(s'_2)</td>
<td>(s'_2)</td>
<td>RREQ</td>
<td>(Min_{Pow} \leftarrow Min_{Pow}^{RREQ})</td>
<td></td>
</tr>
<tr>
<td>(\tau'_6)</td>
<td>(s'_2)</td>
<td>(s'_0)</td>
<td>Time Out</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>(\tau'_7)</td>
<td>(s'_2)</td>
<td>(s'_3)</td>
<td>RREP</td>
<td>(Min_{Pow} \leftarrow Min_{Pow}^{RREP})</td>
<td></td>
</tr>
<tr>
<td>(\tau'_8)</td>
<td>(s'_3)</td>
<td>(s'_3)</td>
<td>RREP</td>
<td>(Min_{Pow} \leftarrow Min_{Pow}^{RREP})</td>
<td></td>
</tr>
<tr>
<td>(\tau'_9)</td>
<td>(s'_3)</td>
<td>(s'_0)</td>
<td>Time Out</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

### Diagnoser construction for attack on PAW-AODV

Figure 3.6 is the diagnoser for the DES model G shown in Figure 3.5. The diagnoser is built following the steps listed in Algorithm 1. Some of the initial steps for this example are as follows.

1. The initial state of the diagnoser, i.e., \(Z_0\) is obtained as follows. \(X_0\) is partitioned into measurement equivalent subsets of G-states, which is one in our case, i.e., \(X_0 = \{X_{01}\}\);
there is only a single initial $G$-state $s_0$ (as shown in Figure 3.5) and $X_{01} = \{s_0\}$. Since $X_0$ can be partitioned into only one subset of measurement equivalent initial $G$-states, $Z_0 = z_{01}$ and $z_{01} = \mathcal{U}(X_{01}) = \mathcal{U}(s_0) = \{s_0, s_0'\}$. Thus, there is only one initial $O$-node $z_0$ (as shown in Figure 3.6) and $z_0 = \{s_0, s_0'\}$. As $z_0 = \{s_0, s_0'\}$ is formed using measurement equivalent states $\{s_0, s_0'\}$, the state of the system is uncertain as $s_0$ corresponds to a normal state while $s_0'$ corresponds to attack state. So, given the diagnoser $O$-node $z_0$, it cannot be predicted whether the attack has occurred or not. Thus $z_0$ is an $F_i$-uncertain $O$-node.

![Figure 3.6: Diagnoser for DES model of PAW-AODV under attack (Figure 3.5)](image)

3. The outgoing $O$-transitions from $z_0$ are obtained as follows. Here, $\mathcal{J}_{z_0} = \{\tau_1, \tau_1'\}$, which are the outgoing measurable transitions from $O$-node in $z_0$. Now, $A_{z_{01}} = \{\tau_1, \tau_1'\}$, as $\{\tau_1, \tau_1'\}$ forms a set of measurement equivalent transitions. Corresponding to $\{\tau_1, \tau_1'\}$, there is $O$-transition $a_1$. 


4. The destination O-node corresponding to \( a_1 \) is obtained as follows. \( z_{0a_1}^+ = \{s_1, s'_1\} \) as \( a_1 \) comprises G-transitions \( \{\tau_1, \tau'_1\} \) and \( \text{final}(\tau_1) = s_1 \) and \( \text{final}(\tau'_1) = s'_1 \). Further, \( z_1^+ = \{s_1, s'_1\} \) as \( \mathcal{U}'(s_1) = \{s_1\} \) and \( \mathcal{U}'(s'_1) = \{s'_1\} \), since there is no unmeasurable transition emanating from either of the states \( s_1 \) or \( s'_1 \). Thus, the destination O-node of the O-transition \( a_1 \) is \( z_1 : \{s_1, s'_1\} \).

In order for the diagnoser to detect attack in PAW-AODV, the diagnoser should reach the O-node \( z_4 \), which is an F\(_i\)-certain O-node (attack certain O-node). However, if the diagnoser gets stuck in an F\(_i\)-indeterminate cycle before reaching \( z_4 \), it leads to non-diagnosability. The DES diagnoser for the attack, shown in Figure 3.6, does not have any F\(_i\)-indeterminate cycles. The cycle comprising of the O-transition sequence \( \langle a_2, a_3, a_4, a_7 \rangle \) is an uncertain cycle and is not an indeterminate cycle. Thus, the DES framework successfully detects the attack in PAW-AODV.

### 3.5 Illustrative example

In this section, we will illustrate the DES based diagnoser construction for power aware vampire attack against PAW-AODV, using a simple example shown in Figure 3.7. First, we will illustrate the DES based modeling of normal and attack condition.

As shown in the figure, the entire region can be divided into two clusters, which are monitored by two SMs, SM\(_1\) and SM\(_2\). The IDS is executed in both these SMs. The sequence of RREQ packets is \( a_1, a_2, a_3 \) and the RREP packet sequence is \( b_1, b_2, b_3 \). The numbers (e.g., 150 for \( S \)) near each node represent the remaining battery power of that node.
3.5. Illustrative example

3.5.1 DES modeling for normal and attack condition for the example network

3.5.1.1 DES modeling for normal condition

In the example under consideration, we have two clusters and hence two SMs. We first illustrate the DES modeling at $SM_1$ under normal conditions.

- Node $S$ broadcasts a RREQ message to discover a path to destination $D$; this packet is marked as $a_1$. The details of the DES modeling for this packet as seen in $SM_1$ are as follows.
  
  - The model starts at state $s_0$. Event generator module and DES diagnoser are in sleep mode. $SM_1$ is monitoring packets and finds an RREQ.
  
  - $\tau_1 : (s_0 \rightarrow s_1)$—The event generator generates the event $\textit{Wake}$. There is no check or assignment of model variables in this transition.
  
  - $\tau_3 : (s_1 \rightarrow s_2)$—In state $s_1$, the event generator generates the event \textit{“RREQ”}. Also, the values related to $\textit{Im\_Src\_IP}$, $\textit{Prev\_Src\_IP}$, $\textit{Src\_IP}$, $\textit{Dst\_IP}$, $\textit{RREQ\_ID}$ and $\textit{Min\_Pow}$ are generated as $\textit{Im\_Src\_IP}^{\textit{RREQ}} = S$, $\textit{Prev\_Src\_IP}^{\textit{RREQ}} = \text{NULL}$, $\textit{Src\_IP}^{\textit{RREQ}} = S$, $\textit{Dst\_IP}^{\textit{RREQ}} = D$, $\textit{RREQ\_ID}^{\textit{RREQ}} = 1234$ (assumed) and $\textit{Min\_Pow}^{\textit{RREQ}} = 150$.

- Node $I_1$ then re-broadcasts the request packet (as $a_2$). The details of the DES modeling for this packet as seen in $SM_1$ are as follows.

  - $\tau_4 : (s_2 \rightarrow s_2)$

  $SM_1$ receives the RREQ packet $a_2$ and it matches

  - Previous source IP of the RREQ ($\textit{Prev\_Src\_IP}^{\textit{RREQ}} = S$) with the value stored in the variable \textit{immediate source IP} ($\textit{Im\_Src\_IP} = S$).
- Source IP of the RREQ ($Src_{IP}^{RREQ} = S$) with the value stored in the variable source IP ($Src_{IP} = S$).
- Destination IP of the RREQ ($Dst_{IP}^{RREQ} = D$) with the value stored in the variable destination IP ($Dst_{IP} = D$).
- Request ID of the RREQ ($RREQ_{ID}^{RREQ} = 1234$) with the value stored in the variable request ID ($RREQ_{ID} = 1234$).

As the above points match the current RREQ, $a_2$ corresponds to the RREQ of transition $\tau_3$. Along with the checks mentioned above, the important check regarding minimum power also holds $-Min_{Pow} = 150 \geq Min_{Pow}^{RREQ} = 90$. This verifies that minimum remaining battery power reported by the current RREQ packet is less or equal to the value reported by the previous RREQ packet. As the value of minimum remaining battery power reported by the current RREP packet is lower than or equal to $Min_{Pow}$, the system is normal.

$assign(V) = [Min_{Pow} \leftarrow Min_{Pow}^{RREQ}]$; now $Min_{Pow}$ becomes 90.

Node $I_2$ then re-broadcasts the request packet as $a_3$, however, it is not captured by $SM_1$ and hence not modeled in the current DES. Packer $a_3$ reaches $D$, which is the destination node. Then $D$ sends an unicast RREP packet $b_1$ to node $I_2$. As $SM_1$ does not capture this packet it is not modeled at the DES of $SM_1$.

- Next, intermediate node $I_2$ sends an unicast RREP packet $b_2$ to node $I_1$. The details of the DES modeling for this packet as seen in $SM_1$ are as follows.

$\tau_6 : (s_2 \rightarrow s_3)$ The Wi-fi sniffer in $SM_1$ has received an RREP packet which is the response to the RREQ packet under consideration. Source IP address and destination IP address of the RREP packet remain same as that of the RREQ packet. So, to verify that the current RREP corresponds to the RREQ sequence (i.e., transitions $\tau_1, \tau_3, \tau_4$), the conditions checked are ($check(V)$ for $\tau_6$)—$Src_{IP} = S \equiv Src_{IP}^{RREP} = S$ and $Dst_{IP} = D \equiv Dst_{IP}^{RREP} = D$. Immediate source IP address and immediate destination
IP address of the first RREQ packet are stored in the corresponding variables by
\( assign(V) = \{ \text{Im} \_ \text{Src} \_ \text{IP} \leftarrow \text{Im} \_ \text{Src} \_ \text{IP}^{\text{RREP}} = I_2, \text{Im} \_ \text{Dst} \_ \text{IP} \leftarrow \text{Im} \_ \text{Dst} \_ \text{IP}^{\text{RREP}} = I_1 \} \).

\[ \tau_7 : (s_3 \rightarrow s_3) : \text{Here check}(V) = \{ \text{Src} \_ \text{IP} = S \equiv \text{Src} \_ \text{IP}^{\text{RREP}} = S, \text{Dst} \_ \text{IP} = D \equiv \text{Dst} \_ \text{IP}^{\text{RREP}} = D, \text{Im} \_ \text{Dst} \_ \text{Node} = I_1 \equiv \text{Im} \_ \text{Src} \_ \text{Node}^{\text{RREP}} = I_1 \}. \] The first two checks verify that the RREP packet received by the sniffer belongs to the same RREQ sequence under consideration. The third check verifies that the immediate source IP of the current RREP is same as that of the immediate destination IP of the last RREP; this checks the transitive property of two consecutive RREPs. To repeat this consecutive check, we have \( assign(V) = \{ \text{Im} \_ \text{Src} \_ \text{IP} \leftarrow \text{Im} \_ \text{Src} \_ \text{IP}^{\text{RREP}} = I_1, \text{Im} \_ \text{Dst} \_ \text{IP} \leftarrow \text{Im} \_ \text{Dst} \_ \text{IP}^{\text{RREP}} = S \} \).

Figure 3.7: Example of network scenario with network monitor

- Now, intermediate node I_1 sends an unicast RREP packet b3 to node S. The details of the DES modeling for this packet as seen in SM_1 are as follows.
• After the source node S receives the RREP, the flow of RREP-RREQ packet for source S and destination D stops. There is a time out and is modeled by $\tau_8$.

Now we illustrate the DES modeling at $SM_2$.

• First, node S broadcasts a RREQ message a1 to discover a path to destination D. It may be noted that, $SM_2$ does not receive this packet and hence not modeled.

Node I_1 then re-broadcasts the request packet (as a2). The details of the DES modeling for this packet as seen in $SM_2$ are as follows.

- The model starts at state $s_0$. Event generator and DES diagnoser are in sleep mode. $SM_1$ is monitoring packets and finds an RREQ.

- $\tau_1 : (s_0 \rightarrow s_1)$–The event generator generates the event $Wake$. There is no check or assignment of model variables in this transition.

- $\tau_3 : (s_1 \rightarrow s_2)$–In state $s_1$, the event generator generates the event “RREQ”. Also, the values related to $Im._Src.IP, Prev._Src.IP, Src.IP, Dst.IP, RREQ.ID$ and $Min.Pow$ are generated as $Im._Src.IP^{RREQ} = I_1, Prev._Src.IP^{RREQ} = S, Src.IP^{RREQ} = S, Dst.IP^{RREQ} = D, RREQ.ID^{RREQ} = 1234$ (assumed) and $Min.Pow^{RREQ} = 90$.

• Node I_2 then re-broadcasts the request packet (as a3). The details of the DES modeling for this packet as seen in $SM_2$ are as follows.

$\tau_4 : (s_2 \rightarrow s_2)$

$SM_2$ receives the RREQ packet a3 and it matches

- Previous source IP of the RREQ ($Prev._Src.IP^{RREQ} = I_1$) with value stored in the variable immediate source IP ($Im._Src.IP = I_1$).

- Source IP of the RREQ ($Src.IP^{RREQ} = S$) with value stored in the variable source IP ($Src.IP = S$).
3.5. Illustrative example

- Destination IP of the RREQ ($D_{st \_IP^{RREQ}} = D$) with value stored in the variable destination IP ($D_{st \_IP} = D$).

- Request ID of the RREQ ($RREQ \_ID^{RREQ} = 1234$) with value stored in the variable request ID ($RREQ \_ID = 1234$).

As the above points match, the current RREQ $a_3$ corresponds to the RREQ of the transition $\tau_3$. Along with the checks mentioned above, the important check regarding minimum power also holds $\text{Min\_Pow} = 90 \geq \text{Min\_Pow}^{RREQ} = 75$. This verifies that minimum remaining battery power reported by the current RREQ packet is less or equal to the value reported by the previous RREQ packet. As the value of minimum remaining battery power, reported by the current RREQ packet is lower than or equal to $\text{Min\_Pow}$, the system is normal.

Assign ($V$) = {$\text{Min\_Pow} \leftarrow \text{Min\_Pow}^{RREQ}$}; now $\text{Min\_Pow}$ becomes 75.

- Now, as D is the destination node, it sends an unicast RREP packet $b_1$ to node $I_2$. The details of the DES modeling for this packet (as seen in $SM_2$) are as follows.

  $\tau_6 : (s_2 \rightarrow s_3)$ The Wi-fi sniffer in $SM_2$ has received an RREP packet, which is the response to the RREQ packet under consideration. Source IP address and destination IP address of the RREP packet remain same as that of the RREQ packet. So, to verify that the current RREP corresponds to the RREQ sequence (i.e., transitions $\tau_1, \tau_3, \tau_4$), the conditions checked are (check ($V$) for $\tau_6$)—$\text{Src\_IP} = S \equiv \text{Src\_IP^{RREP}} = S$ and $\text{Dst\_IP} = D \equiv \text{Dst\_IP^{RREP}} = D$. Immediate source IP address and immediate destination IP address of the first RREP packet are stored in the corresponding variables by assign ($V$) = {$\text{Im\_Src\_IP} \leftarrow \text{Im\_Src\_IP^{RREP}} = D, \text{Im\_Dst\_IP} \leftarrow \text{Im\_Dst\_IP^{RREP}} = I_2$}.

- Now intermediate node $I_2$ sends an unicast RREP packet $b_2$ to node $I_1$. The details of the DES modeling for this packet (as seen in $SM_2$) are as follows.

  $\tau_7 : (s_3 \rightarrow s_3)$: Here check ($V$) = {$\text{Src\_IP} = S \equiv \text{Src\_IP^{RREP}} = S, \text{Dst\_IP} = D \equiv \text{Dst\_IP^{RREP}} = D$.
Chapter 3. Discrete Event System Based IDS for Power Aware AODV

\[ D, \text{Im}_\text{Dst}_\text{Node} = I_2 \equiv \text{Im}_\text{Src}_\text{IP}^\text{RREP} = I_2 \}. \] The first two checks verify that the RREP packet received by the sniffer belongs to the same RREQ sequence under consideration. The third check verifies that the immediate source IP of the current RREP is same as that of the immediate destination IP of the last RREP; this checks the transitive property of two consecutive RREPs. To repeat this consecutive check we have \( \text{assign}(V) = \{ \text{Im}_\text{Src}_\text{IP} \leftarrow \text{Im}_\text{Src}_\text{IP}^\text{RREP} = I_2, \text{Im}_\text{Dst}_\text{IP} \leftarrow \text{Im}_\text{Dst}_\text{IP}^\text{RREP} = I_1 \}. \)

- Following that, node \( I_1 \) sends an unicast packet to \( S \). However, as \( SM_2 \) does not receive this packet, it is not modeled. There is no more RREPs for this case; there is a time out and is modeled by \( \tau_8 \).

### 3.5.2 DES modeling for PAW-AODV under attack condition

Now we consider the DES modeling for the network under attack. Figure 3.8 illustrates the same network scenario as shown in Figure 3.7. In Figure 3.8, there is an attacker, designed as "M", in the the second cluster (monitored by \( SM_2 \)).

As shown in the figure, two RREQ packets reach \( D \) viz. (i) \( a_3 \) from \( I_2 \) and (ii) \( a_5 \) from \( M \). It may be noted that as \( I_1 \) and \( I_2 \) are genuine nodes the information obtained via path \( S,I_1,I_2,D \) (through RREPs \( a_1, a_2 \) and \( a_3 \)) is correct; in this case, hop count is reported as 2 and remaining power as 75. It may be noted that, as M is a malicious node that induces the PAW-AODV attack, information obtained via path \( S,I_1,I_2,M,D \) (through RREPs \( a_1,a_2,a_4 \) and \( a_3 \)) is falsified. Specifically, hop count is reported as 1 and remaining power as 200. This erroneously convinces \( D \) that path \( S,I_1,I_2,M,D \) should be taken instead of \( S,I_1,I_2,D \) considering remaining battery life time perspective, leading to PAW-AODV attack.

As seen in Figure 3.7, the attacker “M” is present in the the second cluster and monitored by \( SM_2 \). So the DES framework illustrating the modeling of PAW-AODV attack can be visualized from the model developed for \( SM_2 \) only. It many be noted that the DES modeling for \( SM_1 \) does not capture the attack situation in this example. Now we illustrate
the DES modeling of PAW-AODV attack as seen from SM$_2$.

![Diagram of network scenario with malicious node]

**Figure 3.8:** Example of network scenario with malicious node

- First, node S broadcasts a RREQ message a1 to discover a path to destination D. It may be noted that SM$_2$ does not receive this packet and hence not modeled.

  Node I$_1$ then re-broadcasts the request packet (as a2). The details of the DES modeling for this packet as seen in SM$_2$ are as follows.

  - The model starts at state $s_0$. Event generator and DES diagnoser are in sleep mode. SM$_1$ is monitoring packets and finds an RREQ.

  - $\tau_1: (s_0 \rightarrow s_1)$—The event generator generates the event *Wake*. There is no check or assignment of model variables in this transition.

  - $\tau_3: (s_1 \rightarrow s_2)$—In state $s_1$, the event generator generates the event “RREQ”. Also, the values related to $Im\_Src\_IP$, $Prev\_Src\_IP$, $Src\_IP$, $Dst\_IP$, $RREQ\_ID$ and
\( Min_{Pow} \) are generated as \( Im_{Src\_IP}^{RREQ} = I_1, Prev_{Src\_IP}^{RREQ} = S, Src\_IP^{RREQ} = S, Dst\_IP^{RREQ} = D, RREQ\_ID^{RREQ} = 1234 \) (assumed) and \( Min_{Pow}^{RREQ} = 90 \).

- Node I_2 then re-broadcasts the request packet (as a3 and a4). We assume that, \( SM_2 \) first receives a3 and then a4. The details of DES modeling for a3 as seen in \( SM_2 \) are as follows.

\[ \tau_4 : (s_2 \rightarrow s_2) \]

\( SM_2 \) receives the RREQ packet a3 and it matches

- Previous source IP of the RREQ \( (Prev_{Src\_IP}^{RREQ} = I_1) \) with the value stored in the variable immediate source IP \( (Im_{Src\_IP} = I_1) \).

- Source IP of the RREQ \( (Src\_IP^{RREQ} = S) \) with the value stored in the variable source IP \( (Src\_IP = S) \).

- Destination IP of the RREQ \( (Dst\_IP^{RREQ} = D) \) with the value stored in the variable destination IP \( (Dst\_IP = D) \).

- Request ID of the RREQ \( (RREQ\_ID^{RREQ} = 1234) \) with the value stored in the variable Request ID \( (RREQ\_ID = 1234) \).

The above points match the current RREQ a3, so it corresponds to the RREQ of the transition \( \tau_3 \). Along with the checks mentioned above, the important check regarding minimum power also holds \( Min_{Pow} = 90 \geq Min_{Pow}^{RREP} = 75 \). This verifies that the minimum remaining battery power reported by the current RREP packet is less or equal to the value reported by the previous RREP packet. As the value of minimum remaining battery power reported by the current RREP packet is lower than or equal to \( Min_{Pow} \), the system is normal.

\[ assign(V) = \{ Min_{Pow} \leftarrow Min_{Pow}^{RREP} \} \]; now \( Min_{Pow} \) becomes 75.

The details of DES modeling for a4 as seen in \( SM_2 \) are as follows.
3.5. Illustrative example

\( \tau_4 : (s_2 \rightarrow s_2) \)

\( SM_2 \) receives the RREQ packet a4 and it matches

- Previous source IP of the RREQ (\( \text{Prev} \_\text{Src} \_\text{IP}^{RREQ} = I_1 \)) with the value stored in the variable immediate source IP (\( \text{Im} \_\text{Src} \_\text{IP} = I_1 \)).

- Source IP of the RREQ (\( \text{Src} \_\text{IP}^{RREQ} = S \)) with the value stored in the variable source IP (\( \text{Src} \_\text{IP} = S \)).

- Destination IP of the RREQ (\( \text{Dst} \_\text{IP}^{RREQ} = D \)) with the value stored in the variable destination IP (\( \text{Dst} \_\text{IP} = D \)).

- Request ID of the RREQ (\( \text{RREQ} \_\text{ID}^{RREQ} = 1234 \)) with the value stored in the variable request ID (\( \text{RREQ} \_\text{ID} = 1234 \)).

As the above points match the current RREQ a4, so it corresponds to the previous RREQ. Along with the checks mentioned above, the important check regarding minimum power also holds – \( \text{Min} \_\text{Pow} = 90 \geq \text{Min} \_\text{Pow}^{RREQ} = 75 \). This verifies that the minimum remaining battery power reported by the current RREQ packet is less or equal to the value reported by the previous RREQ packet. As the value of minimum remaining battery power reported by the current RREQ packet is lower than or equal to \( \text{Min} \_\text{Pow} \), the system is normal.

\[ \text{assign}(V) = \{ \text{Min} \_\text{Pow} \leftarrow \text{Min} \_\text{Pow}^{RREQ} \}; \text{now Min} \_\text{Pow} \text{ becomes 75.} \]

- Node M then re-broadcasts the request packet (as a5). The details of the DES modeling for a5 as seen in \( SM_2 \) are as follows.

\( \tau_4 : (s_2 \rightarrow s_2) \)

\( SM_2 \) receives the RREQ packet a5 and it matches

- Previous source IP of the RREQ (\( \text{Prev} \_\text{Src} \_\text{IP}^{RREQ} = I_2 \)) with the value stored in the variable immediate source IP (\( \text{Im} \_\text{Src} \_\text{IP} = I_2 \)).
- Source IP of the RREQ \( (\text{Src\_IP}^{\text{RREQ}} = S) \) with the value stored in the variable source IP \( (\text{Src\_IP} = S) \).
- Destination IP of the RREQ \( (\text{Dst\_IP}^{\text{RREQ}} = D) \) with the value stored in the variable destination IP \( (\text{Dst\_IP} = D) \).
- Request ID of the RREQ \( (\text{RREQ\_ID}^{\text{RREQ}} = 1234) \) with the value stored in the variable request ID \( (\text{RREQ\_ID} = 1234) \).

As the above points match the current RREQ a5, so it corresponds to the RREQ of previous transition \( \tau_4 \). However, the check regarding minimum power **fails** – \( \text{Min\_Pow} = 90 \geq \text{Min\_Pow}^{\text{RREQ}} = 200 \). The check that “minimum remaining battery power reported by the current RREQ packet (200) is less or equal to the value reported by the previous RREQ packet (75)” fails; the system is under attack.

\[
\text{assign}(V) = \{ \text{Min\_Pow} \leftarrow \text{Min\_Pow}^{\text{RREQ}} \}; \text{ now Min\_Pow becomes 200.}
\]

- Now, as D is the destination node, it sends an unicast RREP packet b1 to the node M. The details of the DES modeling for this packet (as seen in \( SM_2 \)) are as follows.

\( \tau_6: (s_2 \rightarrow s_3) \): The Wi-fi sniffer in \( SM_2 \) has received a RREP packet which is the response to the RREQ packet a5. Source IP address and destination IP address of the RREP packet remain the same as that of the RREQ packet. So to verify that the current RREP is corresponding to the RREQ sequence (i.e., transitions \( \tau_1, \tau_3, \tau_4, \tau_4, \tau_4 \)), the conditions checked are \( \text{check}(V) \) for \( \tau_6 \)— \( \text{Src\_IP} = S \equiv \text{Src\_IP}^{\text{RREP}} = S \) and \( \text{Dst\_IP} = D \equiv \text{Dst\_IP}^{\text{RREP}} = D \). Immediate source IP address and immediate destination IP address of the first RREP packet are stored in the corresponding variables by

\[
\text{assign}(V) = \{ \text{Im\_Src\_IP} \leftarrow \text{Im\_Src\_IP}^{\text{RREP}} = D, \text{Im\_Dst\_IP} \leftarrow \text{Im\_Dst\_IP}^{\text{RREP}} = D \}.
\]

- Now intermediate node M sends an unicast RREP packet b2 to node I\(_2\). The details of the DES modeling for this packet (as seen in \( SM_2 \)) are as follows.

\( \tau_7: (s_3 \rightarrow s_3) \): Here \( \text{check}(V) = \{ \text{Src\_IP} = S \equiv \text{Src\_IP}^{\text{RREP}} = S, \text{Dst\_IP} = D \equiv \text{Dst\_IP}^{\text{RREP}} = D \} \).
3.5. Illustrative example

\( D, \text{Im}_\text{Dst}_\text{Node} = M \equiv \text{Im}_\text{Src}_\text{Node}^{\text{RREP}} = M \). The first two checks verify that the RREP packet received by the sniffer belongs to the same RREQ sequence under consideration. The third check verifies that the immediate source IP of the current RREP is same as that of the immediate destination IP of the last RREP; this checks the transitive property of two consecutive RREPs. To repeat this consecutive check we have assign(\( V \)) = { \text{Im}_\text{Src}_\text{IP} \leftarrow \text{Im}_\text{Src}_\text{IP}^{\text{RREP}} = I_2, \text{Im}_\text{Dst}_\text{IP} \leftarrow \text{Im}_\text{Dst}_\text{IP}^{\text{RREP}} = I_2 }.

- Now intermediate node \( I_2 \) sends an unicast RREP packet \( b_3 \) to node \( I_1 \). The details of the DES modeling for this packet (as seen in \( SM_2 \)) are as follows.

\( \tau_7 : (s_3 \rightarrow s_3) \): Here check(\( V \)) = { \text{Src}_\text{IP} = S \equiv \text{Src}_\text{IP}^{\text{RREP}} = S, \text{Dst}_\text{IP} = D \equiv \text{Dst}_\text{IP}^{\text{RREP}} = D, \text{Im}_\text{Dst}_\text{Node} = I_2 \equiv \text{Im}_\text{Src}_\text{Node}^{\text{RREP}} = I_2 }.

The first two checks verify that the RREP packet received by the sniffer belongs to the same RREQ sequence under consideration. The third check verifies that the immediate source IP of the current RREP is the same as that of the immediate destination IP of the last RREP; this checks the transitive property of two consecutive RREPs. To repeat this consecutive check we have assign(\( V \)) = { \text{Im}_\text{Src}_\text{IP} \leftarrow \text{Im}_\text{Src}_\text{IP}^{\text{RREP}} = I_2, \text{Im}_\text{Dst}_\text{IP} \leftarrow \text{Im}_\text{Dst}_\text{IP}^{\text{RREP}} = I_1 }.

- Following that node \( I_1 \) sends unicast packet to \( S \). However as \( SM_2 \) does not receive this packet it is not modeled. There is no more RREPs for this case; there is a time out and is modeled by \( \tau_8 \).

3.5.3 DES diagnoser for PAW-AODV attack

In this sub-section, we discuss how the power aware vampire attack on PAW-AODV attack is detected using the DES diagnoser. We have already discussed diagnoser construction in Algorithm 1 and illustrated the working of the diagnoser for detecting the attack in
Figure 3.6.

In the network taken as the example (as shown in Figure 3.7), the attacker “M” is present in the the second cluster and monitored by SM₂. So the DES diagnoser illustrating the detection of the attack can be explained from the diagnoser for SM₂ only. Now we illustrate step-wise how the DES diagnoser detects this PAW-AODV attack.

- First, node S broadcasts a RREQ message a₁ to discover a path to destination D. It may be noted that SM₂ does not receive this packet and hence the diagnoser is not initialized.

Node I₁ then re-broadcasts the request packet (as a₂). As per the DES model shown in Figure 3.5, it corresponds to τ₁ i.e., event \textit{Wake}. However, the attacker may also be present in the network and in that case, in the DES model, first the failure causing transition “Failure/Attack” (unmeasurable) takes place followed by τ'₁. As transitions τ₁ and τ'₁ are equivalent, the occurrence of τ₁ (or τ'₁) is captured by the DES diagnoser using the transition \( a₁ : z₀ \rightarrow z₁ \).

Then the event generator module generates the event “REEQ”; this is captured in the DES model by the transition τ₃ (or τ'₃, if there is an attacker). Equivalent transitions τ₃ and τ'₃ are captured by the DES diagnoser using the transition \( a₃ : z₁ \rightarrow z₂ \).

- Node I₂ then re-broadcasts the request packet (as a₃ and a₄). We assume that SM₂ first receives a₃ and then a₄. The details of the diagnoser reacting to RREQ a₃ is as follows.

SM₂ receives the RREQ packet a₃ and it finds that a₃ corresponds to the RREQ of transition τ₃ (or τ'₃). Also, the check regarding minimum power holds – \( Min \text{Pow} = 90 \geq Min \text{Pow}^{RREQ} = 75 \). So the diagnoser cannot determine which of the measurement equivalent transitions τ₃ or τ'₃ has occurred. The diagnoser takes the transition \( a₄ : z₂ \rightarrow z₂ \).

The details of diagnoser reacting to RREQ a₄ is as follows.
3.6. Performance evaluation

SM₂ receives the RREQ packet a₄ and it finds that a₃ corresponds to the RREQ of transition τ₃ (or τ₃'). Also, the check regarding minimum power holds - \( \text{Min}_\text{Pow} = 90 \geq \text{Min}_\text{Pow}^{\text{RREQ}} = 75 \). So the diagnoser cannot determine which of the measurement equivalent transitions τ₃ or τ₃' has occurred. The diagnoser takes the self transition \( a₄ : z₂ \rightarrow z₂ \). It may be noted that, diagnoser reacts to transitions a₃ and a₄ in a similar way.

- Node M then re-broadcasts the request packet (as a₅). SM₂ receives the RREQ packet a₅ and it finds that a₅ corresponds to the RREQ of transition τ₄ (or τ₄'). However, the check regarding minimum power Fails - \( \text{Min}_\text{Pow} = 90 \geq \text{Min}_\text{Pow}^{\text{RREQ}} = 200 \). The check that “minimum remaining battery power reported by the current RREP packet (200) is less or equal to the value reported by the previous RREP packet (75)” fails; now the diagnoser is able to determine that in the DES model the transition τ₆' has occurred and not τ₆. It may be noted that τ₆ is similar to τ₆' except that in case of τ₆, the check regarding minimum power holds. So the diagnoser takes transition \( a₇ : z₂ \rightarrow z₄ \), which comprises only the \( F_i\)-G-transition τ₆' and leads to \( F_i\)-certain \( D \)-state \( z₄ \). In the \( F_i\)-certain \( D \)-state \( z₄ \), attack is detected by the diagnoser.

3.6 Performance evaluation

The performance evaluation of the proposed DES based IDS has been studied in this subsection based on Network Simulator (NS-2) simulation using the parameters shown in Table 3.10. As shown in the table, for simulation we consider an area of 1000m * 1000 m and 5-100 nodes randomly deployed in this area. The initial battery power of each node is set to 20 Joules. Here, the rate of packet generation is set to 1 packet/s. Performance of protocols, IDSs, etc. in WSNs are generally measured in terms of the following parameters.
- **Packet Delivery Ratio (PDR):** PDR is the ratio of the number of packets delivered to the number of packets generated by the source.

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Simulator</td>
<td>NS-2 (Version 2.34)</td>
</tr>
<tr>
<td>2</td>
<td>Operating System</td>
<td>Linux (Redhat 5)</td>
</tr>
<tr>
<td>3</td>
<td>Channel Type</td>
<td>Channel/Wireless Channel</td>
</tr>
<tr>
<td>4</td>
<td>Traffic Model</td>
<td>Constant Bit Rate (CBR)</td>
</tr>
<tr>
<td>5</td>
<td>Source type</td>
<td>UDP</td>
</tr>
<tr>
<td>6</td>
<td>Area (m* m)</td>
<td>1000 * 1000 (initially)</td>
</tr>
<tr>
<td>7</td>
<td>Number of sensor node</td>
<td>5-100</td>
</tr>
<tr>
<td>8</td>
<td>Simulation Time</td>
<td>500 s</td>
</tr>
<tr>
<td>9</td>
<td>Routing Protocol</td>
<td>AODV</td>
</tr>
<tr>
<td>10</td>
<td>Antenna</td>
<td>Omni-directional</td>
</tr>
</tbody>
</table>

- **End-to-End Delay:** End-to-End delay is defined as the average transit time of a packet, i.e., the time taken for a packet to reach the destination from the source.

- **Throughput:** Throughput is computed as the amount of data transferred divided by the data transfer time.

Along with these basic parameters, some other additional parameters are also used based on the context of the protocol of the WSN. For example, in case of PAW-AODV, life time of the nodes is used as an important parameter to evaluate the proposed IDS. PAW-AODV works by selecting nodes with higher remaining battery power targeting improved lifetime of the network, while the power aware vampire attack intends to lower the lifetime.
First, we report our experimental results for PDR. Figure 3.9 illustrates the PDR of AODV and PAW-AODV versus the number of nodes under normal condition. It may be noted that, PDR is improved by PAW-AODV compared to AODV because nodes remain alive for more time in case of the former. However, the reverse trend is observed in case of PDR comparison between PAW-AODV and PAW-AODV under power aware vampire attack;
this is shown in the graph of Figure 3.10.

![Comparison of Packet Delivery Ratio](image)

**Figure 3.11:** Comparison of PDR under attack condition with and without IDS: PAW-AODV

As discussed in Section 3.2, PAW-AODV improves the performance of AODV, however, the attacker nodes can exploit the philosophy of PAW-AODV in a reverse manner leading to selection of nodes have lower battery lifetime. So under attack, PAW-AODV selects
nodes with less battery life thus, lowering the network lifetime.

![Comparison of End to End Delay](image1)

**Figure 3.13:** Comparison of delay under normal condition: AODV and PAW-AODV

![Comparison of End to End Delay](image2)

**Figure 3.14:** Comparison of delay of PAW-AODV under normal condition and attack

So, as shown in Figure 3.10 (and comparing with Figure 3.9), it can be noted that, attacker can make PDR of PAW-AODV much lower even compared to (traditional) AODV.

Following that, we study the performance of the proposed IDS based on PDR.
Figure 3.15: Comparison of delay under attack condition with and without IDS: PAW-AODV

Figure 3.16: Comparison of delay under attack condition on PAW-AODV with IDS and PAW-AODV

Figure 3.11 illustrates the PDR of PAW-AODV under attack, with and without the IDS in execution. Results show that when IDS is executed the PDR of PAW-AODV under attack is improved substantially. In other words, the IDS nullifies the effect of the attack on the PAW-AODV in terms of PDR.
Finally, we compare the PDR of PAW-AODV and PAW-AODV with attack augmented with the IDS in Figure 3.12. Results show that, PAW-AODV under attack with IDS in execution provides near equivalent performance compared with with PAW-AODV under normal condition.
So, we can conclude that the proposed IDS can handle the power aware vampire attack on PAW-AODV and retain the benefits of the enhancement with respect to PDR.

Similar trends were found for the other two parameters i.e., end-to-end delay and throughput. Figures 3.13 through 3.20 illustrate this fact.
Finally, we report our experimental results for average lifetime of the nodes by studying the number alive nodes versus time.

Figure 3.21 illustrates the number alive nodes versus time in case of AODV and PAW-AODV under normal condition. It may be noted that the more number of nodes are alive at any given time in case of PAW-AODV compared to AODV. So, as expected, PAW-AODV
improves the overall lifetime of the network.

As in the case of PDR, throughput and end-to-end delay, the number alive nodes versus time is lowered for PAW-AODV compared to PAW-AODV under attack condition; this is shown in the graph of Figure 3.22. PAW-AODV improves the lifetime of nodes by selecting only the nodes with high remaining battery lifetime, however, the attacker
nodes can exploit the philosophy in a reverse manner leading to lowering of the average network lifetime. So, as shown in Figure 3.22 (and comparing with Figure 3.21), it can be noted that, attacker can make average lifetime of PAW-AODV much lower even compared to (traditional) AODV.

Following that, we study the performance of the proposed IDS based on lifetime of the network. Figure 3.23 illustrates the lifetime of PAW-AODV under attack, with and without the IDS in execution. Figure 3.24 shows the lifetime of PAW-AODV and PAW-AODV with attack augmented with the IDS in execution. Both these results show that the proposed IDS can handle the power aware attacks on PAW-AODV and retain the improved lifetime of network achieved by PAW-AODV.

3.7 Conclusion

In this chapter, we have demonstrated that the enhancements over the basic routing protocols can improve the quality of service. However, some of the improvements make the protocols more vulnerable. Here, the well accepted reactive routing protocol, AODV was considered as the basic protocol and its power based improvement PAW-AODV was considered as the enhancement. Simulation results illustrated that PAW-AODV improves the basic AODV with respect to all the metrics to measure the quality namely, PDR, end-to-end delay, throughput and node lifetime. However, under a simple attack “power aware vampire attack” the performance of PAW-AODV may go down even below the standard AODV. To handle this situation, we have designed a DES based IDS, which can successfully detect the attack on PAW-AODV. Simulation results have proved that if the IDS is applied over PAW-AODV, then it outperforms the standard AODV. To summarise, enhanced routing protocols can outperform their basic versions, however, there is a need for an appropriate IDS to circumvent the vulnerabilities that arise due to the enhancements.

In the next chapter, we will make another similar study, however, on the proactive
protocol “OLSR”. Also, we will design an appropriate IDS to detect attacks that exploit the vulnerabilities arising due to the enhancements over OLSR.