Computers and Mathematics with Applications 60 (2010) 294–306

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

An adaptive learning routing protocol for the prevention of distributed denial of service attacks in wireless mesh networks

Sudip Misra [a], P. Venkata Krishna [b,*], Kiran Isaac Abraham [b], Navin Sasikumar [b], S. Fredun [b]

[a] School of Information Technology, Indian Institute of Technology, Kharagpur, West Bengal, India

[b] School of Computing Sciences and Engineering, VIT University, Vellore, Tamil Nadu, India

ARTICLE INFO

Keywords:
WMN
Distributed denial of service
Optimized link state routing protocol
Learning automata

ABSTRACT

Wireless Mesh Networks (WMNs) have potentially unlimited applications in the future. Therefore, establishing a viable and secure wireless network routing protocol for these networks is essential. Currently, these networks are being used in connecting large sections of cities by setting up wireless routers at strategic points all around the city. These networks can also support connecting remote areas of the country, instead of having to lay a cable all the way. The nature of applications mentioned above make these networks prone to different attacks. Thus, security of these networks is a serious concern. In this paper, we study the impact of *Distributed Denial of Service* (DDoS) attacks on WMNs. We base our work on the existing Optimized Link State Routing protocol (OLSR) and we weave in concepts of Learning Automata (LA) to protect the network from this kind of attack. The simulation results for the proposed scheme show that the proposed protocol is effective in the prevention of DDoS attacks in WMNs.

1. Introduction

Security is one of the primary concerns that arise due to the open nature of WMNs. Security flaws in WMNs make them prone to attacks. These attacks are intentionally undertaken to cripple a network. The objective of such attacks may be to gain administrative control over the network or may be simply to disrupt the services provided by it.

WMNs aim at guaranteeing connectivity [1]. These networks are used to build a multihop wireless backbone to allow interconnectivity of isolated Local Area Networks (LANs). The backbone routers remain predominantly stationary, while the mobile nodes are allowed to maintain mobility of varying degrees amongst them.

Due to the open nature of WMNs, these networks are susceptible to various kinds of attacks. Physical threats [2] can arise due to the improper placing of the routers; resource depletion attack, tempering, wormhole attack [3], 'black hole' attack [4], rushing attack [5,6], and SRP (Secure Routing Protocol) attack [7] are examples of other attacks that may be faced.

The objective of our work is to defend a server from a DDoS attack. In this section, we first explain the characteristics of a DDoS attack. Subsequently, we explain techniques that may be used to combat this unwanted situation.

1.1. DDoS attacks

The main goal of a Denial of Service (DoS) attack is to flood the network with service requests to the server. This can lead to the server being unable to service all the requests, thereby denying offering service to legitimate requests. A DoS attack

* Corresponding author. Tel.: +91 99943215749.
E-mail addresses: sudipm@iitkgp.ac.in (S. Misra), pvenkatakrishna@vit.ac.in (P. Venkata Krishna), kiranisaac@gmail.com (K. Isaac Abraham), navinsasikumar@gmail.com (N. Sasikumar), freddysrinivasan@gmail.com (S. Fredun).

0898-1221/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2009.12.035

from multiple agents attacking the network from various locations constitutes a DDoS. The objective of the routing protocol presented in this paper is to, initially, detect a DoS attack and, if one is detected, take the necessary actions to minimize the number of service requests to the server coming from the attacking hosts, thereby preventing a DoS.

These attacks cripple the server because the server can handle only a fixed volume of requests at a particular time. When the number of requests received by the server exceeds the server's threshold capacity, its response time increases. As the intensity of the attack increases, the server becomes incapable of servicing any requests. Further, any request being serviced is mostly likely to be a rogue one. This leads to legitimate users being denied access to the server.

1.2. Related works

There are many works that introduce novel ways to tackle this potentially network crashing problem. Wellons et al. [8] proposed an oblivious routing pattern for considering traffic dynamics and uncertainty in the mesh network routing optimization. Mao et al. [9] described how the community understands DDoS attacks through backscatter measurement techniques, which have limitations in terms of applicability in the present day Internet. They proposed a swift and seamless shift to direct techniques of attack measurement.

Another approach for securing WMNs, proposed by Li et al. [10], adopts a stochastic security method by using a saddle routing policy. The saddle routing policy-based approach demonstrates a way to increase the throughput by choosing a multipath routing mechanism for known source and destination nodes. They model the problem as a two-player game, between the routing policy maker and the attacker, by proposing a multipath routing protocol that reacts differently to different scenarios.

Lue et al. [11] used an unsupervised learning approach to tackle the DDoS problem. The packet flow is analyzed by a procedure called feature analysis, thereby producing a traffic-based metric; further an outlier algorithm detects the noisy data and the intrusion decision is taken on the basis of the results of the earlier phase.

Beitollahi et al. [12] categorized the kinds of DDoS attacks, split into two techniques — reactive and proactive. Reactive techniques monitor traffic at a particular target location, and use filtering techniques such as ingress, egress and distributed packet filtering. Ingress filtering checks all the received packets' source addresses and blocks those packets that lie outside the router's ingress address range. Proactive techniques include the use of rate control, location hiding, and heuristic techniques. Location hiding is regarded as one of the ways that could be used to develop a complete solution to DDoS [13]. It hides the application's IP address so that malicious users cannot target it in case of a DDoS attack.

Mopari et al. [14] proposed a method for detecting a DDoS attack and defending against a particular type of attack, namely IP spoofing, by maintaining an IP–hop count table and analyzing it (a hop count filter) to detect malicious packets and drop them. Tupakula et al. [15] discuss the effect of TCP SYN and reflection DDoS attacks by preventing the attacker traffic from passing through the ingress router that is nearest to the attack source.

Vrizlynn et al. [16] describe a non-intrusive IP traceback scheme using non-attack conditions to build and maintain a database of valid source addresses traversing the edge routers.

Islam et al. [17] in their work mention the importance of not only the technique of defending against an attack, but also the spatial deployment of detection systems. They show that the ideal way of completely eliminating a DDoS attack is to sample at every node—but that would be impractical—thereby depending on the apt arrangement of detecting systems such that no packet reaches the server without being sampled at least once.

Gupta et al. [18] proposed a method in which propagation of sudden traffic changes inside the public domain is monitored to combat various DDoS attack methods. This scheme realizes how precious the memory and the processing that is involved in single-point sampling are, and thus delegates the work to nodes towards the edges of the network, thereby facilitating in the quick removal of rogue packets.

Goldstein et al. [19] illustrated that DDoS attack defense systems usually generate filter rules to block malicious packets. They propose a method involving traffic shaping wherein the amount of bandwidth allotted to a user is defined by the probability of the user being legal.

The objective of the work reported in this paper was to develop a new routing protocol called DLSR, which prevents DDoS attack using concepts based on intelligent learning using automata. The main contributions of the paper include the following:

- Development of a routing protocol to prevent DDoS attacks in WMNs.
- Introduction of LA-based components into the routing architecture for efficient detection of malicious service requests.
- Proposal of two new frame formats and development of the handling mechanism.
- Proposal of a new algorithm for determining the route to the destination in the case of a DDoS attack.
- Experimental evaluation of the proposed protocol.

2. Learning Automata

Learning Automata (LA) provides a mathematical model for an automated system whose choice of next action depends on the result of its previous actions. Learning automata have three major components — the environment, the set of actions and the system (see Fig. 1).

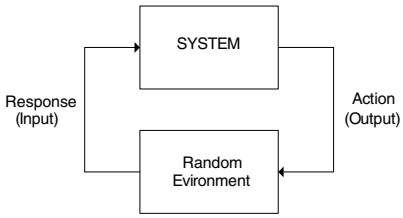

Fig. 1. The learning automaton.

In LA, the system performs without any knowledge of the environment. In other words, from the viewpoint of the system, the environment is random in nature. The system performs an action on this random environment. Depending on the response from the environment, the system decides its next action. This process of adapting its action using the 'knowledge' that it gains is known as 'learning'. The response from the network may be either positive or negative.

A comprehensive overview of LA can be found in the classic text by Narendra and Thathachar [20], and in the recent book chapter by Oommen and Misra [21]. Examples of applications of LA in the domain of networks include [22–27,30–36]. Before elaborating on our proposed solution, we feel it is necessary to provide an understanding of the mathematical formalisms in the definitions of the automaton and the environment in LA.

2.1. The automaton

- The learning automaton can be represented as a quintuple represented as $\{Q, A, B, F, H\}$, where [24]:
- Q is the finite set of internal states $Q = \{q_1, q_2, q_3 \ldots .q_n\}$ where q_n is the state of the automaton at instant n.
- A is a finite set of actions performed by the automaton. $A = \{\alpha_1, \alpha_2 \ldots .\alpha_n\}$ where α_n is the action performed by the automaton at instant n.
- B is a finite set of responses from the environment. $B = \{\beta_1, \beta_2, \beta_3 \ldots \beta_n\}$ where β_n is the response from the environment at an instant n.
- F is a mapping function. It maps the current state and input to the next state of the automaton. $Q \times B \to Q$.
- H is a mapping function. It maps the current state and response from the environment to determine the next action to be performed.

2.2. The environment

The environment corresponds to the medium in which the automaton functions. Mathematically, an environment can be abstracted as a triple $\{A, B, C\}$. A, B, and C are defined as follows [24]:

- $A = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ represents a finite input set;
- $B = \{\beta_1, \beta_2, \ldots, \beta_l\}$ is the output set of the environment; and
- $C = \{c_1, c_2, \ldots, c_r\}$ is a set of penalty probabilities, where element $c_i \in C$ corresponds to an input action α_i.

We now provide a few important definitions used in the field of LA. Given an action probability vector $\mathbf{P}(t)$ at time t, the *average penalty* is defined as [24]

$$\begin{aligned} M(t) &= E[\beta(t)|P(t)] = Pr[\beta(t) = 1|P(t)] \\ &= \sum_{i=1}^{r} Pr\left[\beta(t) = 1|\alpha(t) = \alpha_i\right] \times Pr[\alpha(t) = \alpha_i] \\ &= \sum_{i=1}^{r} c_i p_i(t). \end{aligned} \tag{1}$$

The average penalty for the "pure-chance" automaton is given by [24]

$$M_0 = \frac{1}{r}\sum_{i=1}^{r} c_i. \tag{2}$$

As $t \to \infty$, if the average penalty $M(t) < M_0$, at least asymptotically, the automaton is generally considered to be better than the pure-chance automaton. $E[M(t)]$ is given by [24]

$$E[M(t)] = E\{E[\beta(t)|P(t)]\} = E[\beta(t)]. \tag{3}$$

3. The system model

A wireless mesh network is formed of a collection of independent wireless radio transmitters. In this paper, we consider a wireless mesh scenario with multiple nodes. The server or service provider is connected to one of the nodes. Any of the other nodes may request services from the server (Fig. 2).

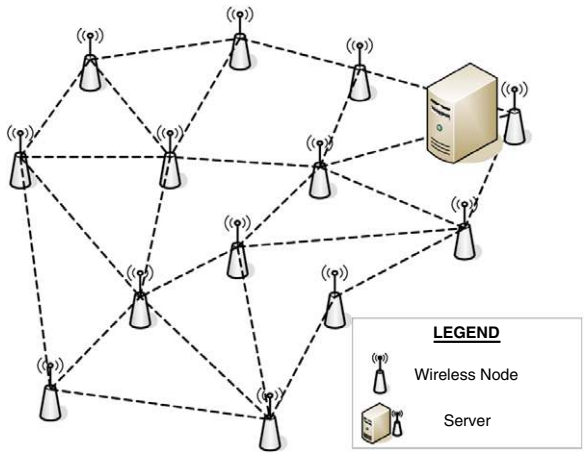

Fig. 2. A wireless mesh network.

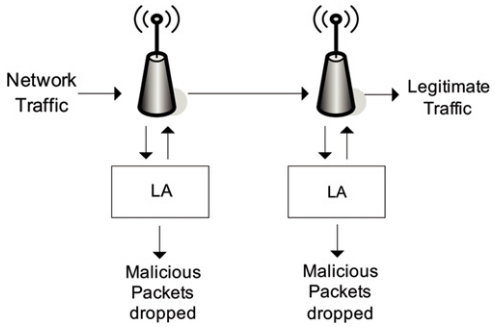

Fig. 3. LA system model.

Each wireless mesh node employs LA for the detection and removal of malicious packets passing through it. Our protocol associates a sampling budget with each node. The network traffic might contain malicious packets and this is passed to the LA system which samples the input based on the remaining budget. If a malicious packet is detected, it is dropped. Since the LA system only checks a portion of the packets being passed through it, some malicious packets may pass through this node undetected. These packets will be detected by subsequent nodes as they traverse the network (Fig. 3).

4. DLSR: A DDoS preventing optimized link state routing protocol

As mentioned earlier, the principal objective of the DLSR protocol is to prevent a DDoS attack from disrupting the services provided by the server. DLSR is a modification of the existing OLSR protocol [28]. The functioning of the DLSR protocol comprises the following three phases:

- DDoS detection.
- Attack identification
- DDoS defense mechanism.

4.1. DDoS detection

In order to service a client's request, the server will need to allocate resources to the client. The resources of the server are limited. When the server does not have enough resources to allocate to service a request, a DoS occurs.

To analyze a DDoS attack, it is required that we fix a maximum service capacity for the server. When the number of service requests exceeds this limit the server is incapable of providing the service and this thus leads to DoS. We consider the server capacity in terms of the number of service requests that the server is capable of servicing within a unit of time. We assume that all service requests are identical.

We also define a service threshold for the server as a percentage of the server's service capacity. If the number of service requests is above the service threshold then there is a possibility that DoS may ensue. So long as the number of service requests is below the service capacity of the server, a DoS will not occur.

In the DDoS detection phase, the server monitors incoming traffic. If the server finds that the service threshold is exceeded, the server sends an alert message (DALERT) to all the nodes in the network. Upon receiving this alert, the nodes go into the attack identification mode (phase 2), while the server continues to monitor and service all requests. The alert message frame format, contents and handling algorithms are discussed in Section 5.1

In essence, the DDoS detection phase is mainly performed at the server by analysis of incoming traffic. It should be duly noted that in this phase all other nodes simply function as OLSR nodes. No special actions are performed. It is also important to note that the start of the attack identification phase does not confirm an attack but merely indicates the possibility of one.

4.2. The attack identification phase

Upon receiving the alert message (DALERT), the nodes are aware that a server in the network is on the verge of DoS. From the DALERT message (Section 5.1), the node is able to identify the server's IP address. At such an instant, all nodes are only aware of the possibility of a DoS, but have no information of the attacker(s). In order to gain information about the attacker, the nodes begin to sample the incoming data and make a note of the hosts that are requesting services from the server. Any host trying to execute a DoS attack on the server will send significantly more service requests than any other host. From the information obtained during the sampling process, the nodes can determine the IP addresses of the malicious hosts.

If any node identifies that one or more hosts are trying to execute a DoS attack, then all nodes are alerted about this. The attacker's information is sent using an Attacker Information Packet (AIP). The AIP contains the IP addresses of all hosts that have been found trying to execute a DoS attack. Upon receiving this packet, all nodes go into the DDoS defense phase. This asserts that the server is under attack and DoS can ensue. The AIP is described in Section 5.2.

4.3. The DDoS defense phase

The server experiences high load, close to its service capacity. All the nodes are informed of this. Some node(s) assert(s) that the network is under attack. The IP address of the attacker has been identified and this information has been conveyed to all other nodes. If the attack is distributed in nature, multiple sources are identified.

The DDoS defense phase starts on a node when it receives an AIP. In this phase, the nodes continue to sample incoming traffic. The node discards packets coming from the identified host(s), continues to monitor traffic flow, and identify any new attackers. If any are found, then the node sends this information using the AIP.

The discarding of packets from identified attackers helps in reducing the number of service requests (that reach the server) and, thus, reducing the load on the server. This, in turn, reduces the possibility of a DoS attack.

It is possible that the attacker uses a false IP address, which might belong to a legitimate user. One can argue that the dropping of packets would result in the genuine user being denied access. However, keeping in mind that our goal is to prevent the server from crashing, we sacrifice the user whose IP address has been spoofed and keep the server functioning.

The sampling of packets by the nodes consumes energy, and causes a small amount of latency in the network. Therefore, it is mandatory that we associate a cost with this sampling procedure. We assume that each node has a sampling budget. This budget can be visualized as the maximum number of packets that a node can sample per unit time. Continuously sampling at maximum budget is not a very energy efficient solution. It is ideal to adjust the rate of sampling on the basis of the number of malicious service requests that pass through the node. Depending on the location of the attackers, the paths taken by the attacker's packets will be different. There is a need for an intelligent system that understands the pattern of the attack and samples the interface in an optimized manner. In this paper, we suggest the use of LA [20,21], briefly presented in Section 2, to model the intelligent sampling system.

It is important to note that the sampling of x packets at a node will not guarantee that all the packets from the attacking host will be identified and discarded. Only a certain fraction of the total DoS attack packets will be caught.

In order to increase the probability of detection and removal of malicious DoS attack packets, the proposed routing algorithm will route the packets such that they traverse a greater number of nodes before reaching the server. This helps to reduce the number of malicious service requests that reach the server. As each packet traverses through more nodes, the probability of detecting the malicious packets increases. There needs to be a limit by which the length of the path can be increased. This limit and the algorithm for routing the packets in this phase are discussed in Section 5.3.

The sketch of the system functionalities is presented in Fig. 4.

5. Frame formats

5.1. The DALERT packet

We propose a new routing message called DALERT (DDoS Alert). The frame format for DALERT is shown in Fig. 5.

The DALERT packet is transmitted by the server when it feels that the network is under threat. The purpose of this message is to alert all nodes in the network of the possibility of an attack. The DALERT message provides information about the server

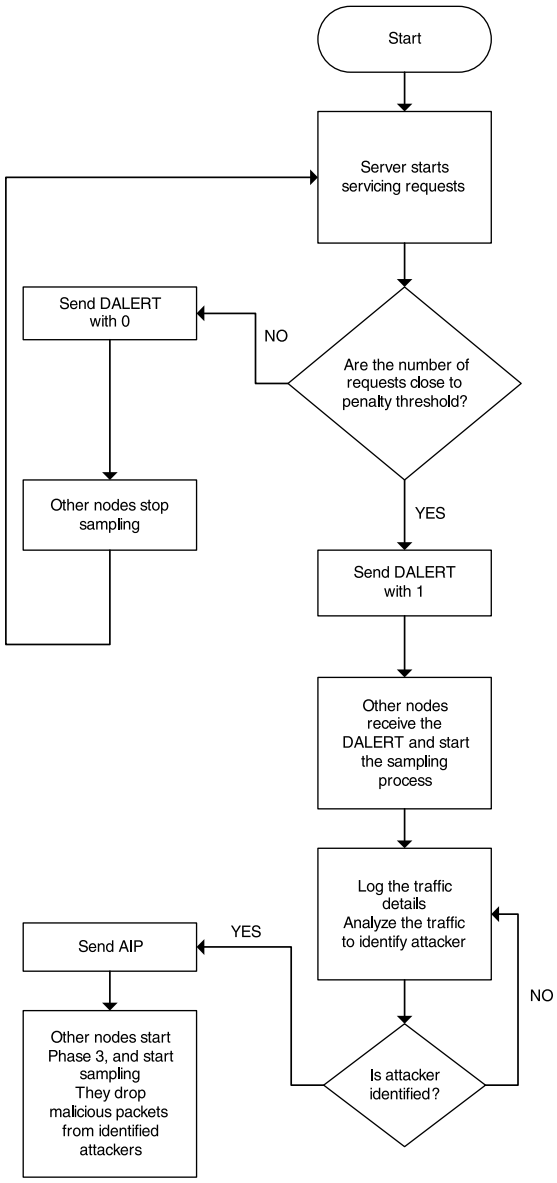

Fig. 4. Sketch of system functionalities.

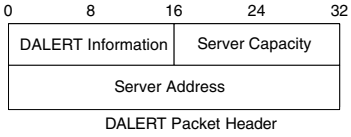

Fig. 5. DALERT frame format.

experiencing high load. It contains a field for DALERT information which can take two values: either 0 or 1, 1 signifying that there is an imminent threat and 0 meaning that there is no threat at the present time. The server capacity field contains an integer that signifies the amount of requests that a server can handle at a given period of time. The server address field contains the server's IP address.

The 'PROCESS DALERT' algorithm is followed by the nodes when a DALERT packet is received.

PROCESS DALERT

A DALERT message can be of two types. A value of 1 indicates that a large volume of packets is being sent to the server and the server is on the verge of DoS. A value of 0 indicates that the server has recovered. Whenever a DALERT message is received by a node:
1. Check the value present in the DALERT message.
2. If the value is 1:
   (i) Set the DALERT information value of the node to 1.
   (ii) Read the server's IP address and capacity from the DALERT message and set the node's server IP and server capacity to the respective values.
3. Otherwise, reset the DALERT value to 0.

The 'SEND DALERT' algorithm is followed by the server when a DALERT packet needs to be sent.

SEND DALERT

A node needs to send a DALERT message if its DALERT value is set to 1.
1. If DALERT value is 1:
   (i) Set the DALERT information in the DALERT message to 1.
   (ii) Copy the server IP and the server capacity to the DALERT message.
2. Otherwise, set the DALERT value to 0.
3. Propagate the message to its neighbouring nodes.

5.2. The AIP

The message format for sending an identified Attacker IP (AIP) to all nodes is shown in Fig. 6. The AIP field contains the potential attacker's IP address.

The 'PROCESS AIP' algorithm is followed by the nodes when an AIP is received.

PROCESS AIP

Whenever an attacker is detected by a node, an AIP is propagated through the network. When a node receives such a packet:
1. Retrieve the attackers IP address from the AIP.
2. Add the attackers IP address to a list containing all the attackers' IP addresses. This serves to prevent a distributed DoS attack by keeping track of multiple attackers attacking at the same time.

The 'SEND AIP' algorithm is used to by the server to send an AIP.

SEND AIP

1. When a node identifies that a certain IP address is sending more packets than a pre-defined percentage of the server's capacity, it recognizes this node as a potential attacker and adds the IP address to a list containing the IP addresses of attackers.
2. Whenever a new attacker has been identified, this information is propagated to all other nodes using an AIP message.
3. Add the new IP address of the attacker to an AIP message and send the message to its neighboring nodes.

5.3. The REQUEST ROUTE algorithm

REQUEST ROUTE

For every packet received at a node:
1. Find the routing table entry for the packets intended for that destination.
2. Find the routing table entry which specifies the next hop for forwarding the packet.
3. If the network is under a threat of a DDoS attack and the packet has not exhausted the number of times that it can be
   re-routed:
   (i) Find a neighbour that is common to both the current node and the next hop node.
   (ii) Change the next hop address of the packet to that of the node identified in the previous step.
4. Forward the packet to the next hop node.

Table 1
Simulation parameters used in NS3.

Parameters	Value
Channel helper	YansWifiChannelHelper
Channel propagation delay	ns3::ConstantSpeedPropagationDelayModel
Physical medium helper	YansWifiPhyHelper::Default
Mac type	AdhocWifiMac
Position allocator	ns3::GridPositionAllocator
Min *X*	0
Min *Y*	0
Delta *X*	250
Delta *Y*	250
Grid width	5
Layout type	RowFirst
Mobility model	ns3::StaticMobilityModel

Table 2
Experimental parameters for Experiment 1.

Parameters	Value
Number of service requests sent by attacker(s)	258
Rate of attack	Constant one packet per second
Simulation time	500 s
LA reward parameter	0.1
LA penalization parameter	0.7
No. of mesh nodes	50
Server threshold	70%
Attack hops	10

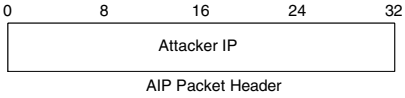

Fig. 6. AIP frame format.

6. Experimental evaluation

The proposed protocol was simulated using the NS3 simulator [29]. Four sets of experiments were conducted to assess the performance of DLSR. The performance of DLSR was compared with respect to OLSR [28]. The general parameters of the simulation that were used in the experiments are presented in Table 1.

6.1. Experiment 1: Varying the server's service capacity

In this experiment, we studied the performance of the network while varying the server's service capacity. The experimental parameters for this experiment are listed in Table 2.

Fig. 7 shows the number of service requests that were denied due to the successful execution of a DDoS attack. It should be noted that the server capacity was set to low values compared to the rate of attack. The aim of this experiment was to assess the performance of the DLSR protocol when the network is faced with an overwhelming attack.

The rate of attack in this experiment is set such that it will push the server into a denial of service attack long before the DLSR nodes can identify the possible attackers. This will surely result in a DDoS attack.

It is important to note from Fig. 7 that as the server's service capacity increases, the number of service requests that are denied decreases. For example, when the server's service capacity was 20, the OLSR protocol denied 158 service requests and the DLSR protocol denied 60 service requests. On the other hand, when the server's service capacity was increased to 50, the OLSR protocol denied only 33 service requests and DLSR denied only 14 service requests. Also, from Fig. 7, it is evident that even in the case of an overwhelming attack, DLSR is capable of significantly reducing its effect on the server. If the server's service capacity is increased to higher values, then the DDoS can be completely prevented.

As the server's service capacity increases, it is capable of servicing more requests. But if this increased service capacity is used to service the attacker's requests, then the server's resources are wasted and this may also lead to a DoS.

In Fig. 8, we see that dropping of malicious service requests in DLSR helps in conserving the server's resources. We see that at a service capacity of 20, when using the OLSR protocol, the server serviced a total of 100 packets from the attacker, while the DLSR protocol only serviced 60 requests. The remaining 40 service requests were removed by intermediate nodes. Depending on the sampling budget of the intermediate nodes, a greater number of service requests from the attacker can be removed from the network before it reaches the server.

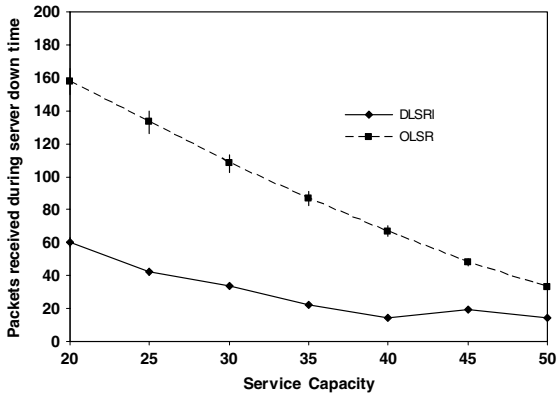

Fig. 7. Denied service requests.

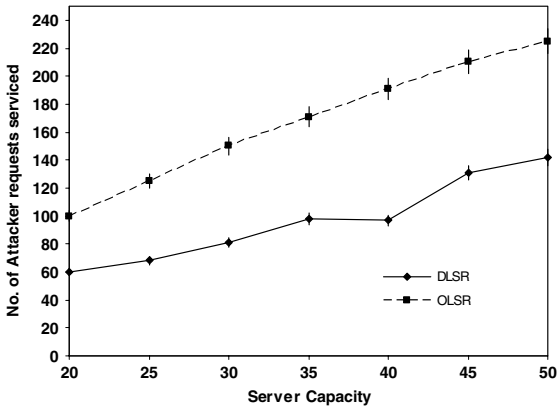

Fig. 8. Serviced attacker requests versus server capacity.

Table 3
Experimental parameters for Experiment 2.

Parameters	Value
Rate of attack	Constant one packet per second
Simulation time	500 s
LA reward parameter	0.1
LA penalization parameter	0.7
Service capacity	30
Server threshold	70%
Attack hops	10

6.2. Experiment 2: The impact of the network size

The aim of this experiment was to study the effect of network size on the performance of DLSR protocol. The experimental parameters for this experiment are presented in Table 3.

In Fig. 9, it can be seen that OLSR services the same number of nodes irrespective of the size of the network. This is due to the fact that no matter how many nodes there are in the network, all the nodes simply send the packet along to the next hop on its way to the destination. But in the case of DLSR, the nodes do not simply forward the packet. Each node checks the originating IP address of the packet and discards it, if it belongs to an attacker. Since each node can sample only a fixed number of requests, all the attacker's packets cannot be discovered by one node. Each node along the route discovers and discards some packets, thereby reducing the number of attacker requests that the server will be servicing. If the number of nodes in a network is greater, the number of hops to the destination will also be greater. Hence, the discovery rate is greater, thereby decreasing the number of attacker requests reaching the server. This is evident from Fig. 9. When the number of nodes in the network was only 25, 100 requests were serviced by the server. When the number of nodes increased to 50, only 81 packets were being serviced. This is due to the fact that there were more nodes in the path to the destination for discovering and discarding the packets.

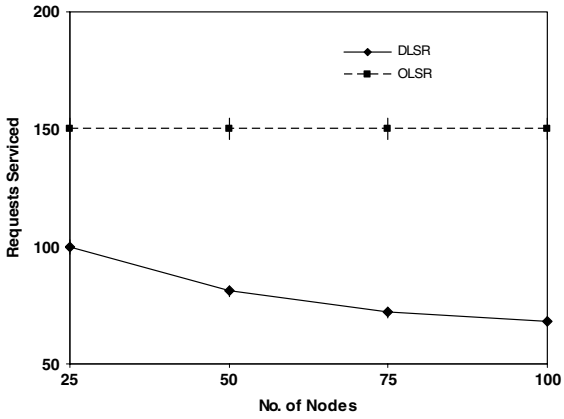

Fig. 9. Service requests versus size of the network.

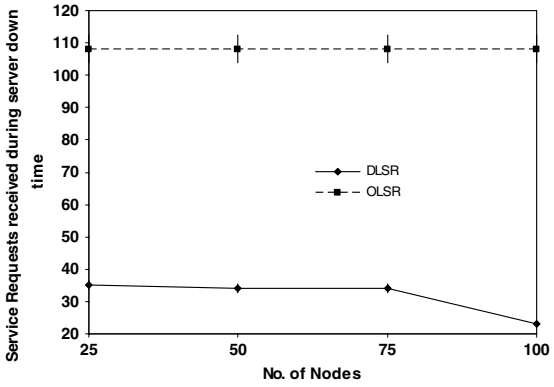

Fig. 10. Denied service requests versus size of network.

Table 4
Experimental parameters for Experiment 3.

Parameters	Value
Number of service requests sent by attacker(s)	258
Rate of attack	Constant one packet per second
No. of nodes	50
Simulation time	500 s
LA reward parameter	0.1
LA penalization parameter	0.7
Service capacity	30
Server threshold	70%
Attack hops	10

It can be inferred from Fig. 10 that the server downtime is less in the case of DLSR than for OLSR. In the case of OLSR, since the size of the network has no effect on the number of packets discovered and discarded, the number of packets reaching the server when it is down is always a constant despite the network having more intermediate nodes.

DLSR, on the other hand, discards more packets as the size of the network increases. Since those packets do not reach the server while it is close to its threshold or while it is down, the server is able to quickly recover and resume normal operation, thereby decreasing the amount of time for which the server is down.

Fig. 10 shows that when there are 75 nodes in the network and the server goes down, 34 packets are received at the server. But when there are 100 nodes in the network, only 23 packets are received when the server is down. This means that the server is facing less downtime when there are 100 nodes as compared to 75 nodes in the network.

6.3. Experiment 3: Analysis of the packet dropping behavior of nodes

Experiment 3 was conducted to study the packet dropping behavior of the nodes in the network. In this experiment, the attacker was installed on Node 1 and the server on Node 50 (see Table 4).

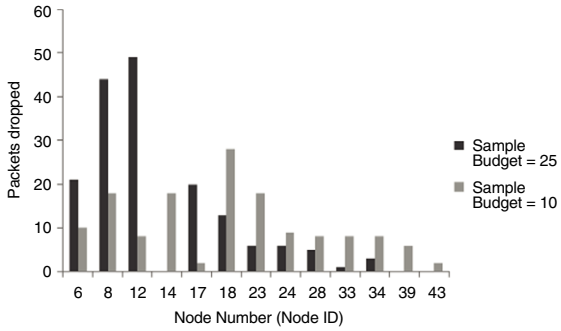

Fig. 11. Packet dropping behavior of nodes.

Table 5
Experimental parameters for Experiment 4.

Parameters	Value
Number of service requests sent by attacker(s)	258
Rate of attack	Constant one packet per second
No. of nodes	50
Simulation time	500 s
LA reward parameter	0.1
LA penalization parameter	0.7
Service capacity	30
Server threshold	70%
Attack hops	10

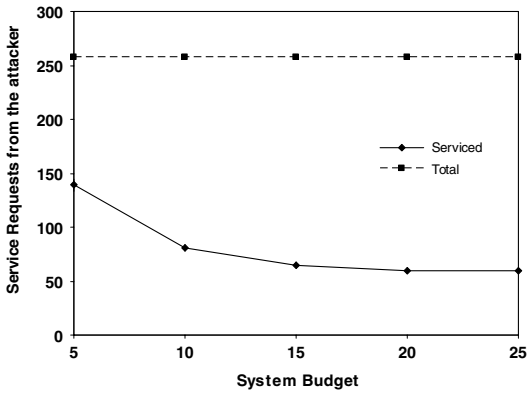

Fig. 12. Packet dropping behavior of nodes.

As the service requests travel through the network, they can get detected and discarded by the intermediate nodes. Fig. 11 shows the packet dropping pattern for the intermediate nodes in the network. The figure shows the behavior when the sample budget is set to 25 and also when it is set to 10.

The initial nodes in the path to the server have a higher probability of detecting malicious service requests. As an increasing number of packets get dropped, the probability of detection of the rest of the service requests decreases.

The same can be noted from Fig. 11 for the case when the sample budget is 25. The high sampling budget allows the earlier nodes in the path to detect more packets. For example, in Fig. 12, when the sample budget is 25, Nodes 6, 8 and 12 drop 21, 24 and 48 packets respectively. On the other hand, Nodes 24, 28 and 33 respectively drop only 6, 5 and 1 packet(s).

When the sample budget was reduced to 10, the detection and discarding of malicious service requests became more evenly distributed. This is because now each node has a smaller sample budget and is not capable of detecting all malicious service requests that pass through it.

6.4. Experiment 4: Varying the sampling budget

In Experiment 4 we studied the effect of the sampling budget on the network. Instead of considering the best case scenario, we consider an overwhelming attack on the network, similar to that executed in Experiment 1. The experiment has been modeled in such a way that the server will go into DoS at least once.

The experimental parameters for this experiment are listed in Table 5.

Fig. 12 shows the total number of service requests that were sent by the attacker, and the fraction of those that were serviced by the server. We see that as the sampling budget increased, the number of malicious requests serviced decreased. When the sampling budget was 5, 139 out of the 258 requests were serviced. On the other hand, when the sampling budget was increased to 25, only 60 out of the 258 packets were serviced. This shows that the higher the sample budget, the smaller the amount of the server's resources that are wasted.

7. Conclusions

In this paper, we have proposed a protocol to prevent DDoS attacks in a wireless mesh network. Our proposed protocol intertwines LA concepts, thereby optimizing the packet sampling mechanism. We suggested two new frame formats. We also evaluated the performance of the system using the NS3 simulator. It was observed that the proposed protocol outperformed OLSR with respect to DDoS type attacks.

In the future, we intend to compare DLSR with other protocol developed for WMNs. We also intend to study different attack scenarios in the future and test our solution in real-life environments. We are particularly keen to study the performance of the proposed protocol when the number of nodes in the network is large.

References

[1] I.F. Akyildiz, X. Wang, A survey on wireless mesh networks, IEEE Communications Magazine 43 (9) (2005) S23–S30.
[2] P. Yi, Y. Jiang, Y. Zhong, S. Zhang, Security for mobile ad hoc networks, Acta Electronica Sinica 33 (5) (2005) 893–899.
[3] Y.-C. Hu, A. Perrig, D.B. Johnson, Wormhole detection in wireless ad hoc networks, Technical Report TR01-384, Department of Computer Science, Rice University, December 2001.
[4] H. Deng, W. Li, D.P. Agrawal, Routing security in wireless ad hoc networks, IEEE Communications Magazine (October) (2002) 70–75.
[5] Y.-C. Hu, A. Perrig, D. Johnson, Rushing Attacks and defense in wireless ad hoc network routing protocols, in: Proceedings of the ACM Workshop on Wireless Security, WiSe 2003, September 19, 2003, Westin Horton Plaza Hotel, San Diego, California, USA.
[6] Y.-C. Hu, A. Perrig, D.B. Johnson, Rushing attacks and defense in wireless ad hoc network routing protocols, in: Proceedings of the 2003 ACM Workshop on Wireless Security, ACM Press, 2003, pp. 30–40.
[7] P. Papadimitratos, Z. Haas, Secure routing for mobile ad hoc networks, in: Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference, San Antonio, TX, January 27–31, 2002.
[8] J. Wellons, Y. Xue, Oblivious routing for wireless mesh networks, in: IEEE International Conference on Communications, ICC, Beijing, China, 2008, pp. 2969–2973.
[9] Z.M. Mao, V. Sekar, O. Spatscheck, J. van der Merwe, R. Vasudevan, Analyzing large DDoS attacks using multiple data sources, in: Proceedings of the SIGCOMM'06 Workshops, September 11–15, 2006, Pisa, Italy.
[10] X.-Y. Li, Y. Wu, W. Wang, Stochastic security in wireless mesh networks via saddle routing policy, in: Proceedings of the International Conference on Wireless Algorithms, Systems and Applications, 2007, pp. 121–128.
[11] W. Lue, I. Traore, An unsupervised approach to detecting DDoS attacks based on traffic based metrics, in: Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, August 2005, pp. 462–465.
[12] H. Beitollahi, G. Deconinck, An overlay protection layer against Denial-of-Service attacks, in: Proceedings of the IEEE International Symposium on Parallel and Distributed Processing, 2008, April 2008, pp. 1–8.
[13] A. Boukerche, L. Guardalben, J.B.M. Sobral, M.S.M.A. Notare, A Self-X approach to OLSR routing protocol in large-scale wireless mesh networks, Proceedings of IEEE GLOBECOM (2008) 778–783.
[14] I.B. Mopari, S.G. Pukale, M.L. Dhore, Detection of DDoS attack and defense against IP spoofing, in: Proceedings of the International Conference on Advances in Computing, Communication and Control, ICAC3'09, January 23–24, 2009, Mumbai, Maharashtra, India, pp. 489–493.
[15] U.K. Tupakula, V. Varadharajan, S.R. Pandalaneni, DoSTRACK: A system for defending against DoS attacks, in: Proceedings of the ACM Symposium on Applied Computing, SAC'09, March 8–12, 2009, Honolulu, Hawaii, USA, pp. 47–53.
[16] V.L.L. Thing, M. Sloman, N. Dulay, Non-intrusive IP traceback for DDoS attacks, in: Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security, ASIACCS'07, March 20–22, 2007, Singapore, pp. 371–373.
[17] M.H. Islam, K. Nadeem, S.A Khan, Efficient placement of sensors for detection against distributed denial of service attack, in: Proceedings of the International Conference on Innovations in Information Technology, IIT, 2008 December 2008, pp. 653–657.
[18] B.B. Gupta, R.C. Joshi, M. Misra, An efficient analytical solution to Thwart DDoS attacks in public domain, in: Proceedings of the International Conference on Advances in Computing, Communication and Control, ICAC3'09, January 23–24, 2009, Mumbai, Maharashtra, India, pp. 503–509.
[19] M. Goldstein, M. Reif, A. Stahl, T. Breuel, High performance traffic shaping for DDoS mitigation, in: Proceedings of the ACM CoNEXT Conference, Student Workshop, Madrid, Spain, December 9, 2008, Article 41.
[20] K.S. Narendra, M.A.L. Thathachar, Learning Automata, Prentice-Hall, 1989.
[21] B.J. Oommen, S. Misra, Cybernetics and learning automata, in: S. Nof (Ed.), Handbook of Automation, Springer, 2009 (Chapter 12).
[22] P. Nicopolitidis, G.I. Papadimitriou, A.S. Pomportsis, Using learning automata for adaptive push-based data broadcasting in asymmetric wireless environments, IEEE Transactions on Vehicular Technology (November) (2002) 1652–1660.
[23] H. Beigy, M.R. Meybodi, Learning automata-based dynamic guard channel algorithms, Journal of High Speed Networks (2009).
[24] M. Esnaashari, M.R. Meybodi, Data aggregation in sensor networks using learning automata, Wireless Networks (October) (2009).
[25] P. Nicopolitidis, G.I. Papadimitriou, A.S. Pomportsis, Learning automata-based polling protocols for wireless LANs, IEEE Transactions on Communications 51 (3) (2003) 453–463.
[26] G.I. Papadimitriou, A.S. Pomportsis, Self-adaptive TDMA protocols for WDM star networks: A learning-automata-based approach, IEEE Photonics Technology Letters 11 (10) (1999) 1322–1324.
[27] G.I. Papadimitriou, D.G. Maritsas, Learning automata-based receiver conflict avoidance algorithms for WDM broadcast-and-select star networks, IEEE/ACM Transactions on Networking 4 (3) (1996) 407–412.
[28] Optimized Link State Routing Protocol (OLSR), RFC 3626. http://hipercom.inria.fr/olsr/rfc3626.txt.
[29] NS3 Simulator, http://www.nsnam.org/.
[30] S. Misra, B.J. Oommen, GPSPA: A new adaptive algorithm for maintaining shortest path routing trees in stochastic networks, International Journal of Communication Systems (Wiley) 17 (10) (2004) 963–984.
[31] S. Misra, B.J. Oommen, An efficient dynamic algorithm for maintaining all-pairs shortest paths in stochastic networks, IEEE Transactions on Computers 55 (6) (2006) 686–702.
[32] S. Misra, B.J. Oommen, Dynamic algorithms for the shortest path routing problem: Learning automata-based solutions, IEEE Transactions on Systems, Man, and Cybernetics, Part B 35 (6) (2005) 1179–1192.

[33] B.J. Oommen, S. Misra, O.-C. Granmo, Routing bandwidth guaranteed paths in MPLS traffic engineering: A multiple race track learning approach, IEEE Transactions on Computers 56 (7) (2007) 959–976.
[34] S. Misra, K.I. Abraham, M.S. Obaidat, P.V. Krishna, LAID: A learning automata-based scheme for intrusion detection in wireless sensor networks, Security and Communication Networks (Wiley) 2 (2) (2009) 105–115.
[35] S. Misra, V. Tiwari, M.S. Obaidat, LACAS: Learning automata-based congestion avoidance scheme for healthcare wireless sensor networks, IEEE Journal on Selected Areas in Communications 27 (4) (2009) 466–479.
[36] S. Misra, B.J. Oommen, S. Yanamandra, M.S. Obaidat, Random early detection for congestion avoidance in wired networks: The discretized pursuit learning-automata-like solution, IEEE Transactions on Systems, Man and Cybernetics, Part B (in press).