2009 28th IEEE International Symposium on Reliable Distributed Systems

The Blocking Option in Routing Protocols

Yan Li
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712
Email: yanli@cs.utexas.edu

Mohamed G. Gouda
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712
Email: gouda@cs.utexas.edu

***Abstract*—Routing protocols are designed under the assumption that each node in a network should be able to reach (i. e. send or forward packets to) every other node in the network. Unfortunately, adopting this assumption in a routing protocol does allow adversary nodes to launch spam or DoS attacks against the other nodes in the network. In this paper, we introduce the "blocking option" in routing protocols; this option allows a node *u* to block a specified set of nodes {v,..,w} and prevent each of them from reaching node *u*. It turns out that if node *u* blocks a large number of nodes, then *u* may end up blocking other nodes as well. We refer to these unintentionally blocked nodes as blind_to *u* nodes. Clearly, a node *u* cannot communicate with its blind nodes in a regular manner. Thus, we extend the routing protocol to allow each node *u* to communicate with its blind nodes via some special node, called the joint node. To perform its intended function, the joint node needs to be neither blocked by any node nor blind to any node in the network. We give an algorithm for identifying the node that is best suited to be the joint node in a network. Finally, we show, through extensive simulation, that the average number of blind nodes is close to zero when the average number of blocked nodes is small (< 20) and that the probability of a joint node being blind is quite small, on the order of 10^{-3}. The path length of using a joint node for communication between a node *u* and any one of its blind nodes *v* is around 1.5 the shortest path between *u* and *v*.**

***Keywords*-Routing Protocols; Distance Vector; Blocking Option; Security**

I. Introduction

Routing protocols are usually designed under the assumption that each node should acquire enough routing information to enable the node to reach (i.e. send or forward packets to) each other node in the network. Unfortunately, this assumption does not allow a node *u*, who suspects that another node *v* has been compromised and has become a source of DoS or spam attacks, to "block" node *v* and prevent it from reaching node *u* for some period of time. In this paper, we revisit this assumption and introduce a blocking option that can be supported by routing protocols. This option allows any node *u* to block any set of nodes and prevent each of them from reaching *u*. We also discuss how a routing protocol can be modified to support this blocking option.

There are two main types of routing protocols in the Internet: distance vector routing protocols and link state routing protocols [20]. In distance vector protocols like Routing Information Protocol (RIP) [17], [25], Enhanced Interior Gateway Routing Protocol (EIGRP) [2], Destination-Sequenced Distance Vector (DSDV) [30], a node periodically informs each of its neighbors of its shortest distance to every other node in the network. And then each node computes and updates its own shortest distance to every other node using Bellman-Ford algorithm. Link state routing protocols like Open Shortest Path First (OSPF) [26], on the contrary, requires each node to obtain a global snapshot of the entire network connectivity. And then each node uses Dijkstra's algorithm to compute a shortest path to every other node in the network. Both types of routing protocols can be modified to support our blocking option. Nevertheless, for convenience, we focus in this paper on how to modify a distance vector routing protocol to support the blocking option.

One may argue that the blocking option is not necessary in routing protocols, since this option can always be replaced by local filtering. For instance, instead of node *u* blocking the set of nodes {v,..,w}, node *u* can let the packets, whose original source is one of the nodes {v,..,w} and whose ultimate destination is *u*, arrive at node *u* then be filtered out and discarded by *u*. However, this is not the case! Attacking packets whose "real" original source is one of the nodes {v,..,w} can carry in their IP headers "spoofed" original sources and so they can evade immediate detection and filtering by *u*.

This scenario cannot occur when the blocking option is supported by the underlying routing protocol. Using our blocking option, node *u* first advertises its intention to block the set of nodes {v,..,w} to all the nodes, other than *v*,..,*w*, in the network. Henceforth, whenever any of the nodes in the blocked set {v,..,w} attempts to forward a packet, whose ultimate destination is node *u*, to any neighboring node, the neighboring node discards the packet, regardless of whether the original source in the IP header of the packet is spoofed or not. Note that the blocking option discards the violating packets near their original sources rather than near their ultimate destinations which tends to save the network bandwidth and avoid congestion. Consequently, the blocking option can force the network nodes to form a distributed "firewall" for the destination node against unwanted traffic from large botnets. We compare our approach with other related work in Section VII.

It turns out that if a node *u* blocks a large number of nodes, then *u* may end up unintentionally blocking other nodes as well. We refer to these unintentionally blocked nodes as blind_to *u* nodes. We give a necessary and sufficient condition

1060-9857/09 $26.00 © 2009 IEEE
DOI 10.1109/SRDS.2009.10

for a node to become blind to node u. Based on this condition, we present an algorithm that uses the blocked set of node u to compute the set of all blind_to u nodes. Furthermore, we describe restrictions that can be imposed on the blocked set of node u to ensure that a given node can never become blind to u.

Clearly, a node u cannot communicate with its blind nodes in a regular manner (since the blind nodes do not know how to send packets to node u). Thus, we extend the routing protocol to allow each node u to communicate with its blind nodes, via some special node in the network called the joint node. The joint node needs to be neither blocked by any node nor blind to any node in the network. We give a necessary and sufficient condition to ensure that a network can have a joint node.

We also present an algorithm for identifying the node that is best suited to be the joint node in a network. That is, a joint node that allows the maximum number of blocked nodes in the network.

Finally, we show, through extensive simulation over power-law and random networks of 1000 nodes each, that the average number of blind nodes is close to zero when the average number of blocked nodes is small (< 20). We also demonstrate that the probability of a joint node being blind to any node is quite small, on the order of 10^{-3}. We also show that on average the length of the path used for communication between a node u and any one of its blind nodes v is around 1.5 the shortest path between u and v.

II. The Blocking Option

In this section, we describe how to extend the distance vector routing protocol in order to support the *blocking option*. This option allows each node u in the network to specify a set of nodes and prevent each of these nodes from reaching node u, i.e., prevent each of the specified nodes from sending or forwarding packets whose ultimate destination is node u. (Following the descriptions in this section, other routing protocols, such as the link state routing protocol [20], can also be extended to support the blocking option.)

In the distance vector routing protocol, the routing table in each node in the network has three columns, named:

(destination, distance, next hop)

Thus, an entry (d, 2, a) in the routing table of a node c indicates that the shortest path from node c to node d consists of two edges and that the first edge on this path is the edge connecting node c with its neighboring node a.

As an example, Figure 1 shows a network N_1 that has nine nodes, named a through i. The routing table of node c in network N_1 is exhibited in Table I. Note that since node i is not reachable from node c due to the failure of edges connecting node i to the rest of N_1, the routing table of node c has the tuple (i, ∞, -).

Periodically, each node sends the two columns (destination, distance) of its own routing table to everyone of its neighboring nodes. Each neighboring node uses the received two columns to compute the new shortest distance to every node and update its own routing table, and the cycle repeats.

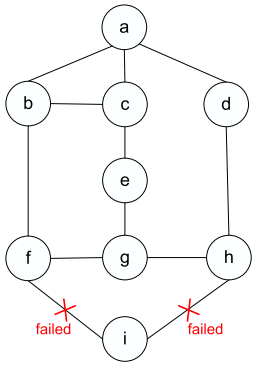

Fig. 1. A network example N_1

TABLE I
ORIGINAL ROUTING TABLE OF NODE c IN NETWORK N_1

destination	distance	next hop
a	1	a
b	1	b
c	0	-
d	2	a
e	1	e
f	2	b
g	2	e
h	3	a
i	∞	-

Three modifications need to be made to the distance vector routing protocol in order to make this protocol support the blocking option. These three modifications are as follows.

1. The routing table in each node is extended to have a fourth column. Thus the extended routing table has the following four columns:

   (destination, distance, next hop, blocked set)

   Thus, an entry (d, 2, a, B.d) in the routing table of node c indicates that node d has decided to block every node in its blocked set B.d and prevent each of them from reaching node d. For example, if B.d = {b, f}, then from now on, if node b or node f ever attempts to send or forward a packet whose ultimate destination is node d, the packet will be discarded by b or f's neighboring nodes before reaching node d. Note that the blocked set B.d of any node d can not include d itself nor anyone of its neighboring nodes. For example, referring to network N_1 in Figure 1, the blocked set B.d of node d cannot include any of the nodes d, a, and h.

   As an example, an extended routing table of node c in network N_1 is exhibited in Table II. According to this routing table, nodes a and b have decided not to block any nodes, whereas node c has decided to block only

node *g*, and node *h* has decided to block the two nodes *a* and *e*. Note that because node *h* blocked both *a* and *e*, node *c* cannot use either of these two nodes to reach node *h*. Thus, the shortest distance from node *c* to node *h* has increased from 3 to 4, and the next hop from node *c* to node *h* has changed from *a* to *b*. The entry (f, ∞, -, -) in the routing table of node *c* can mean one of two things (and node *c* cannot tell for sure which one it is). First, it can mean some some links in the network have failed preventing node *c* from reaching node *f*. Second, it can mean that node *f* has blocked node *c* and so node *c* can no longer reach node *f*.

TABLE II
EXTENDED ROUTING TABLE OF NODE *c* IN NETWORK N_1

Destination	Distance	next hop	blocked set
a	1	a	{}
b	1	b	{}
c	0	-	{g}
d	2	a	{b, f}
e	1	e	{a, h}
f	∞	-	-
g	2	e	{}
h	4	b	{a, e}
i	∞	-	-

2. Periodically each node sends the three columns

   (destination, distance, blocked set)

   of its routing table to each one of its neighboring nodes, possibly after some modifications. In particular, before a node *u* sends the three columns from its routing table to a neighboring node *v*, *u* first checks if *v* is in the blocked set B.w of any node *w*, and if so, node *u* replaces the entry (w, distance, B.w) by the entry (w, ∞, -) in the three columns before forwarding them to node *v*. As a result, node *v* does not know how to reach node *w* via node *u*.

   For example, in network N_1, before *c* sends its three columns to its neighbor *a*, *c* first checks that if *a* is in the blocked set of any node. Since *a* is in the blocked sets of nodes *e* and *h*, *c* will change the two entries of *e* and *h* to (e, ∞, -) and (h, ∞, -) before sending its three columns to node *a*. Consequently, node *a* does not know how to reach *e* or *h* and can not distinguish whether this is because of some link failure or because node *a* is being blocked by *e* and *h*.
3. If a node *u* receives a packet from a neighboring node *v*, and the ultimate destination of this packet is a node *w*, and if *u* checks from its own routing table that *v* is in the blocked set B.w of node *w*, then node *u* discards the packet. (Otherwise, node *u* forwards the packet to the next hop according to its routing table.)

III. THE BLIND NODES

When a node *u* blocks several nodes, it may unintentionally block one or more other nodes. We refer to these nodes, that are unintentionally blocked by node *u*, as the blind_to *u* nodes. As an example, consider the entry (e, 1, e, {a, h}) in Table II. According to this entry, node *e* has decided to block the two nodes *a* and *h*. This means that neither of these two nodes can send or forward packets to node *e* in network N_1 in Figure 1. But because each packet sent or forwarded by node *d* to node *e* has to pass through node *a* or node *h* (before reaching node *e*), node *e* has unintentionally blocked node *d* by blocking node *a* and *h*. In other words, node *d* has become blind to node *e*.

In this section, we define formally what it means for a node *v* to be blind to another node *u*. We then give, in Theorem 1 below, a necessary and sufficient condition for *v* to be blind to *u*. And based on this condition we develop an algorithm, Algorithm 1 below, for computing the set of all nodes that are blind to node *u*, given the blocked set of *u*. Finally we state, in Theorems 2 and 4 and Corollaries 3 and 5 below, several restrictions on the blocked set of node *u* to ensure that a given node *v* is not blind to *u*.

A node *v* is said to be *blind* to a node *u* in network N if *v* is not blocked by *u* but the distance to *u* in the routing table of *v* is ∞. Let D.u denote the set of all nodes that are blind to node *u*. Set D.u is called the *blind_to set* of *u*.

A node *v* is said to *reach* to a node *u* in a network N iff the distance to *u* in the routing table of *v* is finite. Let R.u denote the set of all nodes that reach to node *u*. Set R.u is called the *reach_to* set of u.

Theorem 1.
A node *v* is blind to a node *u* in network N iff each path between *u* and *v* in N has at least one node that is blocked by *u*. □

Theorem 1 states a necessary and sufficient condition for a node *v* to be blind to another node *u* in network N. Next, we use this condition in Algorithm 1 below to compute the set D.u of all blind nodes to node *u*, given network N and the blocked set B.u of node *u*. The basic idea of Algorithm 1 is straightforward: set D.u is computed as the set of all nodes in N that cannot be reached from node u after removing all the blocked nodes in set B.u from N.

The time complexity of Algorithm 1 is O(E+V), where E is the number of edges in network N, and V is the number of nodes in N.

Next, we describe several restrictions that can be imposed on the blocked set of a node *u* in network N in order to ensure that a given node *v* is not blind to node *u* in N. These restrictions will prove useful in the next section.

Two distinct nodes *u* and *v* are said to be *k-connected* in a network N, for some positive integer k, iff either *u* and *v* are neighbors in N or N has at least k node-disjoint paths between u and v. Network N is *k-connected*, for some positive integer k, iff every two distinct nodes in N are k-connected.

Algorithm 1: Compute blind_to set of a node

```
input  : a network N,
         a node u in N,
         blocked set B.u of node u
output: blind_to set D.u of node u
variable: visited: array, R.u: set

1  begin
2      for each node v in N do
3          visited[v] := 0;
4      end
5      R.u := {};
6      visit(u);
7      D.u := {all nodes in N} - R.u - B.u;
8  end

9  function visit(v)
10 begin
11     visited[v] := 1;
12     R.u := R.u ∪ {v};
13     for each neighbor w of v do
14         if visited[w] = 0 and w ∉ B.u then
15             visit(w);
16         end
17     end
18 end
```

Theorem 2.
If two distinct nodes u and v are k-connected in network N, and if node u blocks at most k-1 nodes in N, then node v is not blind to node u in N. □

Corollary 3.
If network N is k-connected and if each node in N blocks at most k-1 nodes, then network N has no blind nodes. □

Theorem 4.
If the shortest distance between any two distinct nodes u and v in network N is less than the shortest distance between node u and each node that is blocked by u in N, then v is not blind to u. □

Corollary 5.
If the shortest distance between two distinct nodes in a network is at most 2, then neither node is blind to the other. □

IV. The Joint Node

Consider the case where a node v is blind to a node u, but node u is not blind to node v in a network N. In this case, u can send or forward packets to v, but v cannot send or forward packets to u since the routing table of node v has the entry (u, ∞, -, -). Thus, nodes u and v cannot carry out a two-way communication in the regular manner. In this section, we describe a routing scheme that can be employed in network N to allow nodes u and v to carry out two-way communications in this case. This routing scheme proceeds as follows:

i. First, designate one node j in network N to be the *joint node* in N. Each node in N knows that j is the joint node in N and should not include it in its blocked set. For now we assume that node j is not blind to any node in N.

ii. Now, consider the case where u is not blind to v but v is blind to u in N. And assume that u sends a packet to v. When v receives this packet from u and discovers that it cannot send back to u, v concludes that "v is blind to u", and so v sends a special message blind(v,u) to node j which forwards the message to u. When u receives the blind(v,u) message from j, it adds v to its own blind_to set D.u.

iii. The next time, u needs to send a packet to v, u sends the packet, using loose source routing, to reach node j before it reaches node v.

iv. When node j receives the packet, it performs a Network Address Translation (NAT) on the packet, indicating that the original source of the packet is j instead of u, before j forwards the packet to v.

v. When node v receives the packet and decides to reply to it, node v sends the reply packets to node j which performs reverse NAT on the packets before forwarding them to node u.

Note that although this routing scheme allows node u to initiate a two-way communication with node v, which is blind to node u, it does not allow the blind node v to initiate a two-way communication with node u. To explain this point, assume that the blind node v uses the above routing scheme to initiate a two-way communication with node u. In this case, node v uses loose source routing to send a packet that is intended to reach the joint node j before reaching node u. However, because node v is blind to node u, this packet, prior to reaching node j, will reach some node b that has been blocked by u. When node b attempts to forward this packet to one of its neighbors, the neighbor will determine that the ultimate destination of the packet is node u and that node u has blocked node b and so the neighbor will discard the packet, according to the third modification in Section II. This also implies that even if a blocked node of node u spoofs an IP address of a blind node to attempt to send packets to node u via the joint node, the packets would be discarded by the neighbors of the blocked node.

Although the blind node v cannot send a packet to initiate a two-way communication between u and v (as explaind in the previous paragraph), node v can still send a reply packet that replies to some earlier packet sent from u to v. This is because the reply packet, as it travels from node v to the joint node j, does not indicate that its ultimate destination is node u. (Remember that when the joint node j performs a NAT operation on the packet from u to the blind node v, node j puts itself as the source of the packet. Thus the blind node v does not even know that the packet actually comes from node u.) Thus, any node b that is blocked by node u can still forward this packet to anyone of its neighbors without the neighbor discarding the packet. Only after the packet reaches node j, does the ultimate destination of the packet become u

(by reverse NAT). But then this packet, as it travels from node *j* to node *u*, is guaranteed not to reach any node that is blocked by node *u*.

It follows from this discussion that IP tunneling cannot be used, instead of NAT and reverse NAT, to send the reply packets from the blind node *v* to node *u*. Had IP tunnelling been used instead of NAT, then the reply packets from the blind node *v* would have node *u* as the ultimate destination in the inner IP header. Since the packet will reach some node that is blocked by node *u*, and when the blocked node attempts to forward the packet to one of its neighbors, the neighbor will discard the packet.

Later we will show that since the number of blind nodes is close to zero when a node only blocks a small number of nodes (<20), it is reasonable to only allow node *u* to initiate a two-way communication with its blind nodes under these attacking conditions.

In order that the joint node *j* can support a two-way communication between any two nodes in network N, when one of the two nodes is blind to the other, node *j* should not be blind to any node in N. The following theorem states a necessary and sufficient condition for network N to have a node that is not blind to any node in N.

Theorem 6.
A network N can have a joint node that is not blind to any node in N iff the following condition holds:

$$(\bigcap_{B.u \neq \emptyset} R.u) \neq \emptyset$$

where B.u is the blocked set of node u and R.u is the reach_to set of node u. □

According to Theorem 6, if any node in the nonempty set $(\bigcap_{B.u \neq \emptyset} R.u)$ is selected to be the joint node *j* of network N, then node *j* is guaranteed not to be blind to any node in network N, as required.

V. How to Choose The Joint Node

The problem of using Theorem 6 in selecting the joint node of network N is that Theorem 6 implicitly assumes that the blocked set B.u for every node *u* in N is completely defined and is guaranteed to remain fixed forever, before the joint node of N is selected. Clearly this assumption is not valid in practice.

It is reasonable to assume that in practice the joint node *j* is selected first, then the maximum possible size for the blocked set B.u of each other node *u* is computed, to ensure that *j* is not blind to *u*, according to the following two rules:

1. If the smallest distance between *j* and *u* is at most 2 then, according to Corollary 5 above, *j* is not blind to *u* no matter how large B.u is. In this case, B.u can contain every node in the network other than *j*, *u*, and every neighboring node of *u*. Thus, the maximum possible size for B.u in this case is (V - 2 - deg.u), where *deg.u* is the number of neighboring nodes of *u*.
2. If the smallest distance between *j* and *u* is more than 2 and if the maximum number of node-disjoint paths between *j* and *u* is dp(j,u) and the size of B.u is less than dp(j,u) then, according to Theorem 2 above, *j* is not blind to *u*. Thus, the maximum possible size for B.u in this case is (dp(j,u) - 1).

This discussion suggests the following algorithm, Algorithm 2.

Algorithm 2: Compute the size of the largest blocked set of a given node *u* in a network N such that the given joint node for N is not blind to *u*

input : a network N,
the joint node *j* of N,
a node *u*, other than *j*, in N
output: size sz of the largest blocked set of *u* such that *j* is not blind to *u*

```
1  begin
2      compute the shortest distance dist(u,j) between u and j
3      if dist(u,j) ≤ 2 then
4          sz := V - deg.u - 2
5      else
6          compute max number dp(u,j) of node-disjoint
7          paths between u and j;
8          sz := dp(u,j) -1;
9      end
10 end
```

The time complexity of Algorithm 2 is dominated by the time complexity to compute dp(u,j), the maximum number of node-disjoint paths between nodes *j* and *u* in network N. Using the well-known push-relabel algorithm [13], one can compute dp(u,j) in O(V^3) time, where *V* is the number of nodes in network N. Thus, the time complexity of Algorithm 2 is O(V^3).

After selecting the joint node *j* of a network N, one can use Algorithm 2 to compute the maximum possible size for the blocked set of every other node in the network such that *j* is not blind to any node in the network. But then one can select the joint node of a network N to be the node that yields the maximum number of blocked nodes in all the blocked sets in the network. This observation leads to the following algorithm.

Since the time complexity of Algorithm 2 is O(V^3), where *V* is the number of nodes in the given network N, the time complexity of Algorithm 3 is O(V^5).

Applying Algorithm 3 to the example network N_1 in Figure 1, one concludes that either of the two nodes *b* or *g* yields the maximum number, namely 33, of blocked nodes in all the blocked sets in network N_1. Thus, either of the two nodes *b* or *g* can be selected as the joint node in network N_1. From this example, one notices that nodes with higher degrees tend to allow more blocked nodes in all the blocked sets in the network. Therefore, one can simply select any node with the highest degree in a network to be the joint node in this network.

VI. Simulation Results

In this section, we present our simulation results intended to answer several questions: 1) What is the relationship between

Algorithm 3: Compute a joint node that allows maximum number of blocked nodes in a network N

input : a network N
output: a joint node "jnode" of N that allows maximum number of blocked nodes in Network N
variable: maxnum: **integer**, blocked: **array**

```
1  begin
2      maxnum := 0;
3      for each node j in N do
4          blocked[j] := 0;
5          for each node u other than j in N do
6              use Algorithm 2 to compute the size sz
7              of the largest blocked set of node u
8              such that the joint node j is not blind to u;
9              blocked[j] := blocked[j] + sz;
10             if maxnum < blocked[j] then
11                 maxnum := blocked[j];
12                 jnode := j;
13             end
14         end
15     end
16 end
```

the average number of blocked nodes and the average number of blind nodes? 2) What is the probability of a joint node being blind to any node in the network? 3) What is the overhead of using a path through a joint node to communicate with a blind node instead of using the original shortest path?

We conducted our simulations using two general networks, generated using the BRITE tool [1]. The first network satisfies the power law distribution based on the Barabasi-Albert model with parameter m = 2. The second is a random network based on the Waxman model with parameters α = 0.15 and β = 0.2. In both networks, the total number of nodes in the network is 1000.

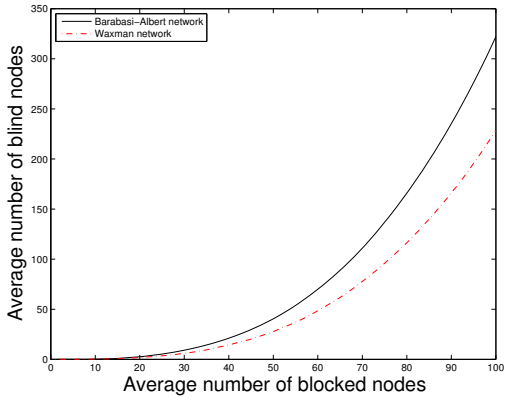

Fig. 2. Average number of blind nodes versus average number of blocked nodes

In the first set of experiments, we evaluate the relationship between average size of the blind_to set and the average size of the blocked set. The result is shown in Figure 2 for both the Barabasi-Albert network and the Waxman network. We can see that for both networks, when a node blocks less than 20 nodes, it will end up with almost zero blind nodes. In this case, the joint node will be barely used. If a node blocks 100 nodes, which is tenth of all the nodes in the network, the average number of blind nodes would be about 320 in the Barabasi-Albert network and about 220 in the Waxman network.

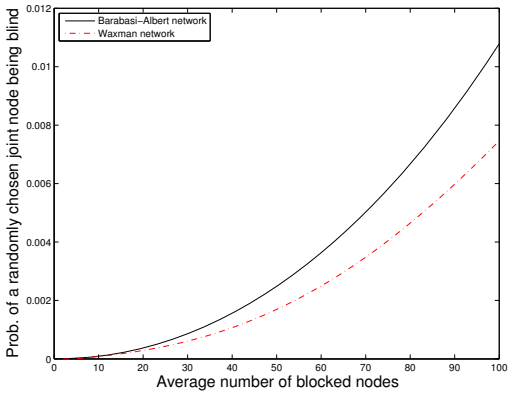

Fig. 3. Probability of a randomly chosen joint node being blind versus average number of blocked nodes

In the second set of experiments, we evaluate the probability of a joint node being blind when the number of blocked nodes varies from 1 to 100. First, we randomly choose a joint node in each of the two networks, and plot the results in Figure 3. In both networks, the randomly chosen joint node has a very small probability of being blind, less than 1.1%, to any node in the network even when the average number of blocked nodes is tenth of the network size.

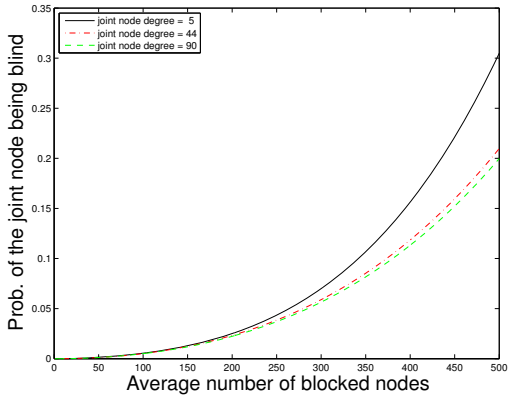

Fig. 4. Probability of a joint node with different degrees being blind versus average number of blocked nodes for the Barabasi-Albert network

Second, we randomly choose a joint node with different degrees: highest degree (90 in the Barabasi-Albert network and 18 in the Waxman network), medium degree (44 in the in the Barabasi-Albert network and 8 in the Waxman network) and a

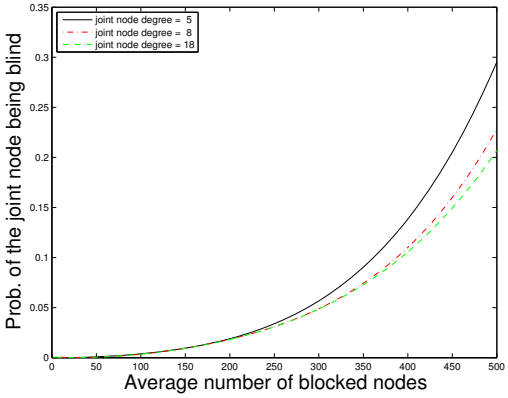

Fig. 5. Probability of a joint node with different degrees being blind versus average number of blocked nodes for the Waxman network

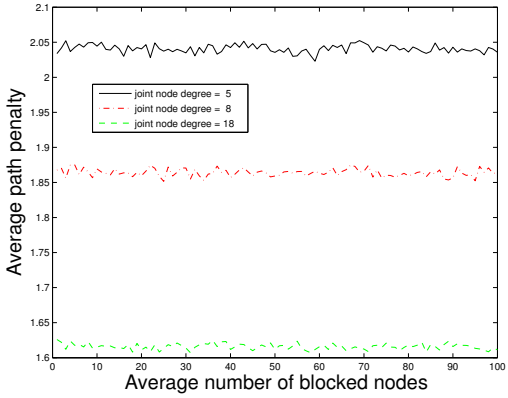

Fig. 7. Average path penalty versus average number of blocked nodes for the Waxman network

small degree 5 in both networks, and evaluate the probability of the chosen joint node being blind. The results are shown in Figure 6 for the Barabasi-Albert network and in Figure 7 for the Waxman network. When the number of blocked nodes is smaller than 150 nodes, there is not much difference to choose a joint node with different degrees. However, if in the extreme case that a node wants to block a large number of nodes, for example half of the nodes in the network, a joint node with a larger degree will have a lower probability of being blind (about 10% lower when choosing the joint node with the largest degree compared to choosing the joint node with a small degree 5 in both networks). Also, the probability of the joint node being blind when choosing a joint node with medium degree is close to that when choosing a joint node with the largest degree. It indicates that we do not need to choose a joint node with the largest degree, instead, we can choose a joint node with a medium degree in the network.

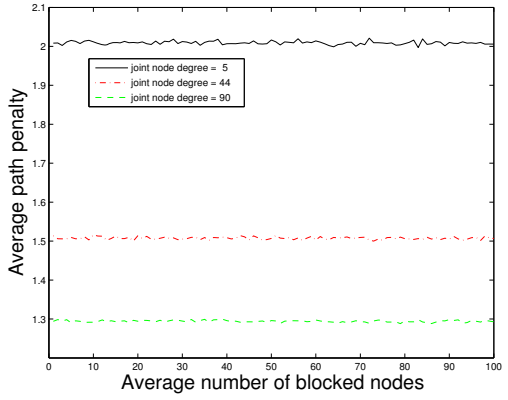

Fig. 6. Average path penalty versus average number of blocked nodes for the Barabasi-Albert network

Next, we evaluate the overhead of using a joint node to communicate between a node and its blind nodes. We define *path penalty* as the path distance of using a joint node to communicate from a node u to u's blind node v divided by the shortest path distance between u and v. We then using the average path penalty to measure the overhead of using a joint node in communication instead of the shortest path. Figure 6 and Figure 7 show the average path penalty versus the average number of blocked nodes for joint nodes with different degrees, for the Barabasi-Albert network and Waxman network respectively. Not surprisingly, a joint node with larger degree will have a smaller average path penalty. But with the increase of blocked nodes, the average path penalty stays quite stable.

VII. Related Work

There have been many studies of non-shortest path routing to achieve various goals such as security [6], [9], [19], [21], [40]–[42], reliability [7], [10], [11], [16], [27], [33], [37] and load balancing [8], [12].

Zlatokrilov and Levy studied how to achieve node avoidance routing in distance vector networks in [41], [42]. In [42], they propose to retrieve distance vector information from some reference nodes and then use loose source routing to avoid certain area. In [41], a node determines a sequence of virtual positions to avoid certain area and the routing algorithm forwards packets along a path that is closest to the virtual path. In [19], Khurana et al. addressed a similar problem to route around misbehaving nodes in ad hoc networks to minimize the impact of attacks.

In [6], [9], [40], multi-path routing is used to prevent session eavesdropping with different models. Lee et al. [21] proposed to use multi-path routing to minimize the damage incurred by DoS attacks to a single link.

In this paper, we focus on a new routing paradigm with blocking options to block unwanted traffic from unwanted nodes such as nodes within a botnet.

Besides using multi-path routing to counter DoS attacks, many other schemes have been proposed to battle unwanted traffic, such as spams or DDoS attacks from botnets. Spammer behavior and spamming botnets have been studies in [22], [31], [36] and some anti-spam strategies have been proposed such

as the filtering scheme in [32]. Also, many solutions have been proposed to defend against DoS attacks in the Internet such as filter-based approaches [5], [15], [23], [24], [28], capability-based approaches [4], [38], [39], overlay-based approaches [3], [14], [18] and other approaches [34], [35].

Existing filtering schemes are targeting interdomain attacks. Both ingress filtering [15] and Distributed Packet Filtering (DPF) [28] are targeting how to proactively filter out spoofed packets and thus mitigating DDoS attacks, but they can do nothing to protect against flooding attacks which originate from valid IP prefixes. dFence [24] proposes to dynamically introduce a filtering middlebox on the path to a destination within a domain and ensure all traffic to that destination will pass through that middlebox. This filtering function is centralized and thus is not scalable to large botnets. Both AITF [5] and StopIt [23] first try to filter attack traffic at the source node, if not successful, they will try to filter attack traffic at the destination node. These schemes are well designed and suitable for interdomain filtering.

Our blocking option focuses on blocking unwanted traffic from within a domain. Within a domain, relying on the source node to filter its own traffic can be precarious. This is because hosts in a local network tend to have same configurations and consequently same vulnerabilities. if an attacker can comprimise one host within the local network, it is easy for an attacker to compromise other hosts within the same network. Thus, when one compromised host is discovered and filtered, the attacker can switch to other hosts in the same network, which makes it a waste of time to figure out which hosts should be filtered. Second, hijacking and controlling a node may allow an attacker to conduct various attacks like DoS attacks using traffic routed through that node. Consequently, when the destination node wants to block all traffic from the source node, of course we can not ask the source node to achieve this. Third, if many hosts are demanded to be filtered at the source node, the source node may not want to filter so much of its own traffic. So it may not be safe to ask the source node itself to execute the filtering. Based on these considerations, we introduce the blocking option to be a complementary choice for a node to achieve different filtering goals. This option can provide the destination node a quick and easy way to block unwanted traffic under some severe conditions.

On the other hand, relying on the destination node to filter unwanted traffic is less desired since it will waste bandwidth and may cause congestion. Instead, our blocking option relies on neighboring nodes of the source to block unwanted traffic sent or forwarded by the source node.

In capability-based approaches [4], [38], [39], the receiver explicitly authorizes the traffic it wants to receive, so every packet needs to carry some certificate to prove its identity. Still, the unwanted traffic will waste network bandwidth to reach the receiver and be discarded at the receiver. We want to discard the unwanted packets as early as possible to avoid congestion and any harm to the intermediate nodes in the network. Also, the overhead involved in the cryptographic operations makes it suitable for end-hosts but not for routers.

In the overlay-based approaches, SOS [18] and Mayday [3] use a set of overlay nodes to authenticate clients and filter unwanted traffic to protect a server from DDoS attacks. Phalanx [14] leverages a large set of overlay nodes to relay traffic for a protected destination. These approaches concern how to protect some specific destinations via the rich connectivity in the overlay network.

Our work differs from the above work in the sense that we focus on the routing level to block nodes that are unwanted from the destination node and we block the unwanted traffic as early as possible and in a scalable way. Thus, we introduce flexibility to traditional routing protocols to block unwanted traffic in the first place.

The blocking option presented in this paper is also applicable in DSDV [30] or AODV [29] based ad-hoc networks: when the destination node believes that the source node is suspicious (such as compromised or caught by hostile parties), the destination node can apply the blocking option to block the source node as early as possible.

VIII. Concluding Remarks

We presented, in this paper, the blocking option that can be adopted by any routing protocol. This option allows any node *u* to block a specified set of nodes and prevent each of them from reaching (i.e. sending or forwarding packets to) node *u* as early as possible and in a distributed manner. The blocked set of node *u* can change over time to reflect two concerns of node *u*. First, if at some point *u* believes that it will not need to communicate with another node *v* for an extended time period, then *u* can add node *v* to its blocked set during that period. Second, if at some point *u* becomes the target of DoS or Spam attacks and believes that a node *v* is the source of this attack, then *u* can add node *v* to its blocked set for some time period.

We noticed that if node *u* adds a relatively large number of nodes to its blocked set, then *u* may end up blocking other nodes that *u* did not intend to block in the first place. We refer to these unintentionally blocked nodes as blind_to u nodes. To solve the problem of the blind_to *u* nodes, we extended the routing protocol to allows node *u* to communicate, via a special node in the network called the joint node, with each one of its blind nodes. To perform its intended function, the joint node should not be blocked by any node or blind to any node in the network. We gave an algorithm to identify the node that is best suited to be the joint node in a network.

Through simulation, we found out that the average number of blind nodes is close to zero when the average number of blocked nodes is small (< 20), and that the probability of a joint node being blind is quite small, on the order of 10^{-3}. We also measured the overhead of using a joint node in the communication between a node and its blind nodes. The length of the communication path, that passes through the joint node, between a node *u* and its blind node *v* is around 1.5 the shortest distance between *u* and *v*.

As mentioned before, to perform its intended function, the joint node should not block any other node and should not be blind to any other node. This means that the joint node cannot rely on the blocking option to defend itself against DoS or Spam attacks. On the other hand, the function of the joint node is not critical to the routing of packets in the network. For instance, if the joint node is attacked and is no longer able to perform its intended function, only the exchanged packets between each node and each one of its blind nodes will cease. This does not represent a great loss since the number of blind nodes is usually close to zero.

REFERENCES

[1] BRITE: Boston univeristy Representative Internet Topology gEnerator. http://www.cs.bu.edu/BRITE/.

[2] Enhanced interior gateway routing protocol. http://www.cisco.com/en/US/tech/tk365/technologies_white _paper09186a0080094cb7.shtml.

[3] D. Andersen. Mayday: Distributed filtering for internet services. In *3rd Usenix USITS*, 2003.

[4] T. Anderson, T. Roscoe, and D. Wetherall. Preventing internet denial of service with capabilities. In *ACM HotNets-II*, 2003.

[5] K. Argyraki and D. R. Cheriton. Scalable network-layer defense against internet bandwidth-flooding attacks. In *to appear in ACM/IEEE Transaction on Networking*.

[6] A. Bagchi, A. Chaudhary, M. T. Goodrich, and S. Xu. Constructing disjoint paths for secure communication. In *17th International Symposium on Distributed Computing(DISC'03)*, Sorrento, Italy, October 2003.

[7] S. Bahk and M. E. Zarki. Dynamic multi-path routing and how it compares with other dynamic routing algorithms for high speed wide area networks. In *ACM Sigcomm'92*, Baltimore, Maryland, August 1992.

[8] S. Bak and J. Cobb. Randomized distance-vector routing protocol. In *Proceedings of the 1999 ACM Symposium on Applied Computing*, 1999.

[9] S. Bohacek, J. P. Hespanha, K. Obraczka, J. Lee, and C. Lim. Enhancing security via stochastic routing. In *The 11th International Conference on Computer Communications AND Networks(IEEE ICCCN'02)*, Miami, Florida, October 2002.

[10] M. Cha, S. Moon, C.-D. Parkz, and A. Shaikh. Placing relay nodes for intra-domain path diversity. In *IEEE Infocom'06*, Barcelona, Spain, April 2006.

[11] J. Chen, P. Druschel, and D. Subramanian. An efficient multipath forwarding method. In *IEEE Infocom'98*, San Francisco, April 1998.

[12] R. Cohen and G. Nakibli. On the computational complexity and effectiveness of N-hub shortest-path routing. In *IEEE Infocom'04*, Hong Kong, March 2004.

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press/McGraw Hill, 2nd edition, 2001.

[14] C. Dixon, A. Krishnamurthy, and T. Anderson. Phalanx: Withstanding multimillion-node botnets. In *USENIX/ACM NSDI*, 2008.

[15] P. Ferguson and D. Senie. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. In *RFC 2827*, 2000.

[16] Y. Guo, F. Kuipers, and P. V. Mieghem. Link disjoint paths for reliable QoS. *International Journal of Communication Systems*, pages 779–798, November 2003.

[17] C. Hedrick. RFC 1058 - Routing Information Protocol, June 1988.

[18] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure overlay services. In *ACM Sigcomm'02*, Pittsburgh, Pennsylvania, August 2002.

[19] S. Khurana, N. Gupta, and N. Aneja. Minimum exposed path to the attack (mepa) in mobile ad hoc networks. In *IEEE International Conference on Networking (ICN 2007)*, Sainte-Luce, Martinique, April 2007.

[20] J. F. Kurose and K. W. Ross. *Computer Networking: A Top-Down Approach*. Addison Wesley, 4th edition, 2008.

[21] P. Lee, V. Misra, and D. Rubenstein. Distributed algorithms for secure multipath routing. In *IEEE Infocom'05*, Miami, FL, March 2005.

[22] F. Li and M.-H. Hsieh. An empirical study of clustering behavior of spammers and group-based anti-spam strategies. In *Proceedings of the 3rd Conference on Email and Anti-Spam*, 2006.

[23] X. Liu, X. Yang, and Y. Lu. To filter or to authorize: Network-layer DoS defense against multimillion-node botnets. In *ACM Sigcomm'08*, Seattle, Washington, August 2008.

[24] A. Mahimkar, J. Dange, V. Shmatikov, H. Vin, and Y. Zhang. dFence: transparent network-based denial of service mitigation. In *USENIX/ACM NSDI*, 2007.

[25] G. Malkin. RFC 2453 - RIP Version 2, November 1998.

[26] J. Moy. RFC 2328 - OSPF Version 2, April 1998.

[27] A. Orda and A. Spronston. Efficient algorithms for computing disjoint QoS paths. In *IEEE Infocom'04*, Hong Kong, March 2004.

[28] K. Park and H. Lee. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. In *ACM Sigcomm'01*, San Diego, California, August 2001.

[29] C. Perkins, E. Belding-Royer, and S. Das. RFC 3561 - Ad hoc On-Demand Distance Vector (AODV) Routing, July 2003.

[30] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *ACM Sigcomm'94*, London, UK, August 1994.

[31] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *ACM Sigcomm'06*, 2006.

[32] A. Ranmachandran, N. Feamster, and S. Vempala. Filtering spam with behavior blacklisting. In *Proceedings of the 14th ACM Conference on Computers and Communications Security*, 2007.

[33] D. Sidhuand, R. Nairand, and S. Abdallah. Finding disjoint paths in networks. In *ACM Sigcomm'91*, 1991.

[34] A. Snoeren, C. Patridge, L. Sanchez, C. Jones, F. Tchakountio, S. Kent, and W. Strayer. Hash-based IP traceback. In *ACM Sigcomm'01*, 2001.

[35] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker. DDoS defense by offense. In *ACM Sigcomm'06*, 2006.

[36] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov. Spamming botnets: Signatures and characteristics. In *ACM Sigcomm'08*, Seattle, WA, August 2008.

[37] D. Xu, Y. Chen, Y. Xiong, C. Qiao, and X. He. On finding disjoint paths in single and dual link cost network. In *IEEE Infocom'04*, Hong Kong, March 2004.

[38] A. Yaar, A. Perrig, and D. Song. SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks. In *IEEE Symposium on Security and Privacy*, 2004.

[39] X. Yang, D. Wetherall, and T. Anderson. TVA: A DoS-limiting network architecture. *IEEE/ACM Transactions on Networking*(to appear), 2009.

[40] H. Zlatokrilov and H. Levy. Privacy and reliability by dispersive routing. In *IEEE Infocom'06*, Barcelona, Spain, April 2006.

[41] H. Zlatokrilov and H. Levy. Navigation in distance vector spaces and its use for node avoidance routing. In *IEEE Infocom'07*, Anchorage, Alaska, May 2007.

[42] H. Zlatokrilov and H. Levy. Area avoidance routing in distance-vector networks. In *IEEE Infocom'08*, Phoenix, Arizona, April 2008.