Wireless mesh networks filtering, from a firewall to a waterwall*

Leonardo Maccari, Renato Lo Cigno
DISI – University of Trento, Italy
leonardo.maccari@unitn.it, renato.locigno@unitn.it

Abstract

Filtering has always been intended as a function to be applied to the border routers, in order to separate segments of networks. In a wireless mesh network, however, there is no usable definition of border, so the function of a firewall must be distributed among all the nodes in the network. But a firewall is generally installed on a complex, powerful machine, which is able to filter/analyze/log traffic based on rule-sets that can be made of thousands of rules, without any obvious structure to suggest an efficient organization to reduce search costs. A low-cost mesh node that need to implement filtering can be realized with a 166 MHz CPU and 4 Mbytes of disk space. Nevertheless, mesh networks like community networks can be made of thousands of nodes, and service tens of thousands of end customers, just like any big traditional network. and consequently they can be configured with thousands of specific rules. The firewall concept must be changed: from a powerful single host to a coalition of nodes that realize the same function by distributing among a subset of them portions of the rule-set. The goal becomes reaching a given global accuracy with the minimum cost, and the means is no more a single unbeatable barrier, but a fluid shifting and changing functionality whose goal is washing the network clean of unwanted traffic: a waterwall!

In this contribution we explore two different possibilities. The first one is based on the estimation of the filtering node position along the source-destination path, while the second is based on the estimation of the node centrality with respect to the network routing/traffic.

*Financed by Provincia di Trento under *The Trentino programme of 360 research, training and mobility of post-doctoral researchers, incoming Post-docs 361 2010* CALL 1, PCOFUND-GA-2008-226070

1 Introduction

Protecting a network from unsolicited, often malicious traffic is one of the constant concerns of any network administrator. Apart from standard networking devices as switches and routers, normally middleboxes as NATs (Network Address Translations) and firewalls are installed on the network boundary to separate trusted portions of the network form the global Internet or in general from less trusted ones.

In some cases however, even the separation between the internal and the external network is not straightforward, and identifying boundaries and points of interconnection is even more difficult. A typical example is a wireless mesh network, where a collection of subnets are interconnected through a backbone of mesh nodes, but each subnet is only loosely coupled with the others, and many points of access to the global Internet may exist (see figure 1 for a pictorial representation). Mesh networks are often used with this configuration in order to bring connectivity in a cost-effective way to areas where other technologies would be too expensive. As a concrete example, community networks use this approach to share network resources between hundreds or even

thousands of users and represent one of the most successful application of mesh networking. Projects like Guifi or Awmn (see `Guifi.net` and `awmn.gr`) represent an example of how this technology can be competitive with the infrastructured communication networks and how successful this approach can be.

In this report we tackle the problem of firewalling in large mesh networks. In such networks each mesh node applies a specific firewall rule-set to the traffic directed to itself (or to subnets attached to it). The firewall will be used in order to defend the local network from attacks, to shape the access to the Internet across its connection, or to forbid the access to certain logical resources. If all the nodes share their rule-sets and enforce them also on the outgoing traffic, the traffic would not be filtered at the destination but directly at the source, reducing the waste of network resources. But in this case each node would be filtering with a global rule-set made of thousands of rules, which is not practical for most of low-power Linux-based mesh routers. We propose to split the global rule-set in pieces and enforce only a portion of it at every hop with the goal of filtering the packets as close as possible (given the constraints on the rule-set dimension to be implemented) to their source node in order to save network resources.

We explore two different alternative approaches. In the first one, we propose to partition the rule-set among the nodes that are along the source-destination pair, trying to filter the packet as close as possible to the source (to reduce resource waste in the network), while maintaining the filtering burden for nodes below a given threshold. In the second one, we propose to enforce the full rule-sets only in a portion of the nodes of the network. We show that using the OLSR protocol we can easily identify the most *central* group of nodes that will route a large portion of the whole traffic, and apply the full rule-set only on that set of nodes. We will recall and use the notion of *shortest path betweenness* SPBof a node, in order to identify the nodes that will carry a larger amount of traffic.

A traditional firewall does not introduce false positives (packets that should be dropped but are instead forwarded) when it is correctly configured. Indeed, with our approach each node singularly introduces some false positives, but as a global network function the firewall will work with a parametrizable accuracy. To stress the difference from a typical firewall we chose the term *waterwall* to indicate a distributed and homogeneous filtering function spread on all the nodes in the network. Note that we use filtering as our target application for the sake of simple explanation and by way of example, but with the same logic we could enforce some other traffic analysis functions such as intrusion detection.

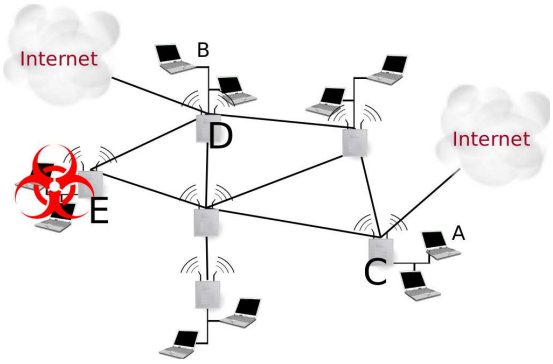

Figure 1: A common mesh-network structure

2 Motivation

Figure 1 shows a widely-used configuration for a wireless mesh network where a set of mesh routers interconnect separated LANs. Each LAN has its own IP addressing and the routing protocol running on the mesh routers allows the clients of distinct LANs to communicate. In some cases nodes may physically roam from a LAN to another, depending on the kind of routing protocol they may or may not maintain their initial IP address to keep their sessions alive. Finally, some of the LANs have a direct access to Internet and share it with the other users that are not equipped with it.

In this scenario the owner of a mesh node is generally also the manager of the corresponding LAN and is interested in protecting it. We take into consideration three use cases applicable to the simple network in figure 1:

1. The manager of network C wants to protect its network from unwanted traffic coming from the outside. For instance, he does want to block connections to remote shell protocols coming from the mesh network to host A in LAN C;

2. With a more finer grain he may want to limit access only to some logical resources, for instance, host A may have some folders that are shared only on the LAN while some other are shared with the whole mesh network. The access to these resources can be denied or simply limited to a maximum bit-rate;

3. The manager of network C wants to forbid some kinds of traffic that come from the mesh network and are directed to the Internet using its connection. This is normally due to the commercial agreements that the manager has with his networks service provider. Again, traffic can be forbidden or it can be limited to a certain maximum bitrate.

Now imagine that a node in the network labeled E starts an attack against, let's say, a host in network C. This may be due to a malicious user or to a virus that took control of a host in the network and starts a DoS, or a brute force attack. We then add a fourth use case:

4. The manager of network C detects an attack and reactively enforces network filters to protect its resources.

The issues described in the use cases can be partially resolved configuring a firewall on each mesh node in order to filter the traffic directed to its LAN. The first three use-cases can be approached setting up a mixture of layer-4 and application layer firewall rules on the mesh-router in C that will drop or shape some traffic. The fourth one can be approached with dynamical rules that are activated when the firewall detects an anomaly in the usage of the resources, for instance, an abnormal number of ICMP packets. Linux-based firewalls support all these features. What remains unsolved are the consequences for the rest of the mesh network and for the other LANs. The malicious traffic coming from network E will still traverse the mesh network and subtract useful resources to the allowed communications. Considering that in a mesh network the available bandwidth is shared between upload and download, this can severely impact the victim LAN but also the other networks on the way from the attacker to the victim.

This example shows how the concept of border firewall does not correctly apply to the mesh network scenario, where the *border* of the network is hard to define.

To solve this problem the mesh routers can share their rule-sets in order to apply them directly on the other mesh routers. The rule-set of mesh router C applies only to the traffic directed to LAN C, to some logical resources it controls or to the Internet traffic flowing across its connection. Each mesh router will collect all the rule-sets in a global rule-set and enforce it directly on the multi-hop path so that the traffic is filtered as close as possible to the source. It's not the goal of this paper to investigate how the rule-sets are securely distributed, in the most simple case rule-sets can be known in advance and every node just sponsors the ID of one or more predefined rule-set in routing messages. This approach indeed protects not only the resources of each LAN but also the shared resources of the mesh network.

Now imagine that this model is applied to a large mesh network. As an extreme but realistic use-case, imagine that this model is applied to a community wireless mesh network like the Guifi network. Guifi is made of thousands of nodes[1] and used by tens of thousands of users that daily access the network from various places (see [3] for a characterization of its topology features). What happens if even only 10% of the mesh routers start distributing a rule-set made of, let's say, 30 rules each? That the global rule-set will be made by tens of thousands of rules. Corporate firewalls can handle large rule-sets up to tens of thousands of rules, but this is not the case for wireless routers that are generally low-cost devices with minimal energy consumption. The most used products

[1] At the time of writing, Guifi network is made of about 19.000 nodes and growing at a pace of a hundred nodes per week. The network is divided in zones, each one can be formed by hundreds of nodes.

are commercial devices that embed a low-power processor (e.g., a 133 MHz Intel or AMD low end device), one or more IEEE 802.11b/g/a/n wireless cards and run a customized Linux kernel. The whole hardware is enclosed in an outdoor shell powered over LAN and costs no more than 100€. A 133 MHz processor can not easily handle a rule-set made of thousands of rules organized in a linear list, it will introduce processing delays and packet dropping.

To improve filtering performance, rule-sets can be pre-processed with various approaches, none of which is easy to port in this context. For instance, once the whole rule-set has been created, wildcards and numeric ranges can be used to group rules and reduce their total number. This involves a complex and costly pre-processing of the rule-set but speeds up the look-up time during the routing decision. It is convenient when the rule-set is mostly static and when the hardware is powerful enough for the pre-processing. In the case we consider rules can be dynamically generated, nodes can be added or removed to the network and links may be temporally unavailable. Each of these events will change the single rule-sets or the whole topology (and consequently add/remove rule-sets associated with nodes). Assuming that the nodes are powerful enough to perform the pre-processing, they would spend most of their CPU time repeating this task.

Complex data structures can be used instead of a linear list such as trees or graphs. The more complex is the data structure, the more memory and pre-processing are needed. The less complex the data structure, the less flexible and performing is the technique. For instance, rules can be grouped using their target netmask, but this is meaningless for application layer rules, for multicast rules or when a node that has a certain resource to be filtered roams to a new network. Moreover with a mesh network made of thousands of nodes, there are thousands of netmasks, so that filtering is still cumbersome.

Both these approaches are hardly applicable when the rules do not match IP addresses and TCP ports but layer-7 data inside a packet. In this work we take a different direction trying to exploit the distributed nature of mesh networks instead of being doomed by it.

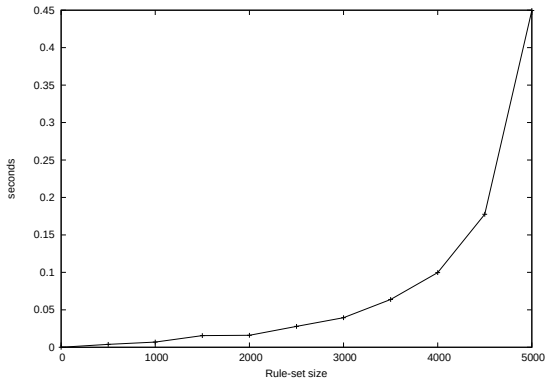

Figure 2: Delay introduced by a growing rule-set size when the host forwards approximately 1 Mbit/s of traffic.

2.1 Firewalls with large rule-sets

Before we detail the proposed approach, we further investigate the consequences of large rule-sets on the performance of the network. Figure 2 reports the increment in the processing time of a single packet when the rule-set size grows. The data have been measured using an embedded system equipped with a 400 MHz processor and 128 Mbytes of RAM over a wired network. 50% of rules match network and transport layer fields, the rest match the packet contents at layer-7. Contrarily to the results we obtained for a previous work [13] where the tests were carried without traffic, the measures have been taken when the node is under a load of 1 Mbit/s.

Up to 3000 rules the delay grows almost linearly, meaning that the system is able to handle the load as expected. After that threshold the delay grows at a faster pace and arrives close to 0.5 s with 5000 rules. Since this delay is introduced by every node for every hop, the total round-trip-time in a mesh network using this rule-set makes the network unusable. This explains that filtering is not a function that can be introduced "for free" when the rule-sets get large.

3 Route based filtering

Consider a network like the one in figure 1 made of N nodes where a proactive routing protocol is running, (from now on, for simplicity we will refer to mesh-nodes simply as "nodes"). Each node j is connected to a subnet and for each node j exists a rule-set r_j that is used to filter the traffic directed to its own subnet, to itself or to the Internet across the connection attached to its subnet. Node j will sponsor its own rule-set to the rest of the nodes, so that every node is aware of a global rule-set R that is the union of r_j for every j. The routing table of a node i contains the next hop and the distance in terms of hops to reach j and all the nodes in the subnet of j. This is the usual setup of a mesh network configured, for instance, with the OLSR protocol.

Now consider a packet p that comes from the subnet of node k, is forwarded by node i, and is destined to the subnet of node j, for this packet there exists a rule in R that will drop it when it arrives to j. The aim of the *waterwall* is to drop the packet as close as possible to the source node k. The most simple solution is to enforce the whole R directly in k. This solution has two drawbacks: it is impossible if R is made of thousands of rules for the considerations introduced in section 2.1, and it would be extremely easy to circumvent, since when the packet leaves its own subnet it is not filtered anymore. A node k that behaves in a malicious way can start an attack against a node j and all the traffic will arrive at destination. To tackle the second issue, more nodes on the path from k to j will have to apply the filter, thus aggravating the first issue. The strategy we propose is to filter at each hop with only a subset of the global rule-set that is dynamically chosen for each packet and for each hop. We use larger rule-sets for nodes close to the source and smaller rule-sets for nodes far from the source. Table 1 contains symbols and notation used in this contribution.

How can a node i estimate the distance from the source node k of the packet p destined to j? The most simple measure is to look at the time-to-live field in the IP header, but, again, it is also the easiest to circumvent. The attacker we take into consideration is able to mangle the contents of packets, but we imagine that the routing protocol implements some security measures to avoid, or at least identify, attacks on network routing. This kind of attacker could simply change the TTL value in the source packet and avoid the *waterwall* to be effective. Another way is to use the distance from the source node k to i. The attacker could set the source IP to the address of another node w and, contrarily to what happens on the Internet, it would still be able to intercept the replies provided it is in the shortest path between w and i. Summing up, node i can not trust the contents of a packet coming from a node that is possibly an attacker so the distance from the source must be estimated with other means.

$sp(i,j)$	a set of nodes that form the shortest path between node i and node j
$sp_l(i,j)$	the length of $sp_l(i,j)$
δ	a parameter that determines the maximum size of a rule-set enforced on a node
$m(i)$	the average distance of node i from all the other nodes in N
r	the average size or a rule-set used by a node in N
R	the global rule-set, i.e. the union of all the rule-sets
$t(i)$	$sp_l(i,j)$ for a packet with source node k, destination j, averaged for every couple (k,j) for which $i \in sp(k,j)$

Table 1: Notation and symbols used

We propose to increase the size of the rule-set depending on the ratio between the distance from the destination and $m(i)$, the average distance of node i to any node in the network. In practice, node i compares the length of the remaining path to the destination with the average length of the path of packets generated by i itself. We define:

$$Pf(k,i,j) \triangleq \begin{cases} \frac{sp_l(i,j)}{m(i)}\delta & \text{if } sp_l(i,j) \leq m(i) \\ 1\delta & \text{if } sp_l(i,j) > m(i) \end{cases}$$

δ is a parameter that can be used to limit the maximum number of rules enforced in a single node. Node i will use a random subset R_i of R of size $Pf(k,i,j) * ||R||$, so $Pf(k,i,j)$ represents the probability of p to be filtered on i. If R is organized as a linear list, this can be implemented starting to scan the list from a random point for a portion of the list of size $||R_i||$. When i is close to j the fraction $\frac{sp_l(i,j)}{m(i)}$ decreases, contrarily when i is close to k the value of $Pf(k,i,j)$ is close to $1 * \delta$. We define also $t(i)$ as the value of $sp_l(i,j)$ averaged on all routes passing through i between any couple (k,j).

$$t(i) \triangleq \sum_{k,j \in N, k!=j, j!=i, i \in sp(k,j)} \frac{sp_l(i,j)}{(N-1)(N-2)}$$

To understand how our approach scales with the size and shape of the network graph we have to understand the behaviour of $t(i)$. Let's define m and t as the average $m(i)$ and $t(i)$ computed on every node. m is the expected number of hops that a packet p generated by any node k will perform in the network, t is the expected remaining path of p when it passes across a node, averaged on all the nodes. Intuitively we expect t to be smaller than m, but how do $t(i)$ and $m(i)$ change depending on the position of i in the network? In the next sections we will first present the results based on an example linear topology, then we will analyze a more complex 2D topology.

3.1 1D linear topology

As a clarifying example, we take a linear topology with 10 nodes and we report the average values of $t(i)$, $m(i)$ and $t(i)/m(i)$ in figure 3.

It can be noticed that the values of $m(i)$ are influenced by the position of i in the topology. In particular, nodes that are close to the border will have larger values compared to nodes that are in the center of the topology. This can be explained noting that when i is in the periphery of the network its distance from the other nodes is in average larger than when i is in the center, so $m(i)$ is higher on the borders. It can also be easily shown that in this simple topology, if we compute $t(i)$ excluding the packets that are generated by i itself, $t(i)$ is constant. This would

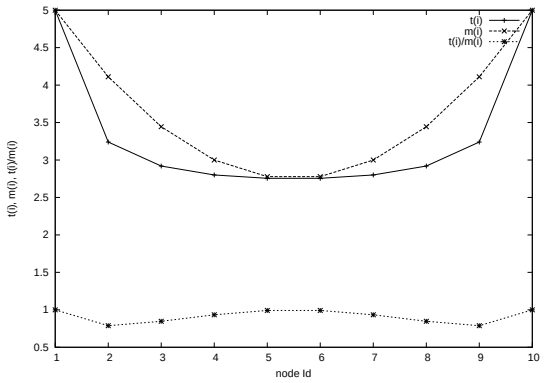

Figure 3: The values of t(i), m(i) and their ratio on a sample linear topology with 10 nodes

make the ratio $t(i)/m(i)$ decrease for nodes close to the extremes of the network. In the figure, instead, we plotted $t(i)$ including also the packets generated by node i, which increases the values of $t(i)$ on the borders. This is to take into account that in our network scenario we consider each node as the gateway of its own subnet, so the first hop is generated in the subnet. Even in this case, $t(i)/m(i)$ is still larger for nodes that are central in the topology.

We expect the central nodes of the network to be more congested than the nodes in the borders, since the number of shortest paths that pass across them is higher. Considering this, the shape of $t(i)/m(i)$ introduces a positive effect: the more p gets close to the center of the network, the higher is the chance of being filtered. The practical consequence is that when p is moving from the periphery to the center of the network, that is more congested, its chances of being filtered are increased. When p has already passed the central region of the network, the chances of being filtered decrease. If we look it from a different perspective, we impose a larger filtering effort for packets that are going towards the most loaded area of the network because we want to save resources in that area where they are more precious. When the packets have passed the central area we spend less effort to filter them, since they are directed to the periphery of the network, which is going to be less

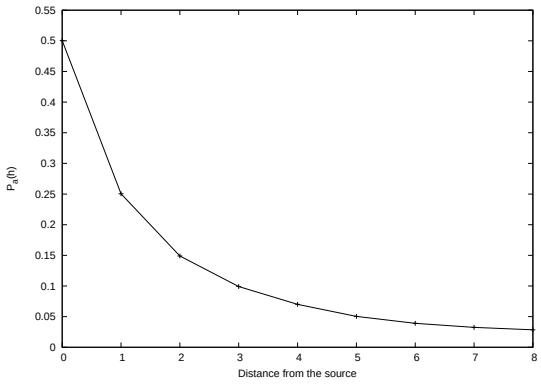

Figure 4: The values of Pf(h) on the linear topology with 10 nodes

congestioned. Moreover, packets will be filtered at destination anyway.

We now define the probability of a packet p to be filtered after h hops from the source node k when it is destined to node j.

$$Pf_h(k,h,j) = Pf(k,i,j) \text{ where } sp_l(k,i) = h$$

$Pf_h()$ moves the dependency of $Pf()$ from the node i where the packet is filtered to the position of i in the route from k to j. $Pf_h()$ can be averaged for all the couples k, j in order to keep only the dependency on h. Once defined $Pf_h()$ we can compute the inverse probability that p arrives at h hops from the source node, that we call $P_a(h)$ (arrival probability):

$$P_a(h) = \prod_{i=0}^{i=h-1} (1 - Pf_h(i))$$

In figure 4 we report P_a for the network depicted in figure 3 when δ as been set to 0.5

The diameter of the network is equal to 9 hops, when the packet arrives at destination it is filtered with a different rule-set then when it is filtered on the path, so we do not include the last hop in the curve. We have numbered them from 0 to 8 indicating that the first chance of being filtered is on node k itself. From figure 4 we can see that in average we obtain the desired effect, the chances of a packet to be filtered are higher in the neighborhood of the source node and decrease when it gets close to destination.

3.2 2D Topologies

In the linear topology described so far the distance between two nodes is given by the modulus of the difference between their node Id so the results are obtained by means of simple computations. When the network topology is defined on a 2-dimensions plane more complex instruments must be used. The most suitable instrument to study the behaviour of a mesh network with a 2D topology is computer simulations, nevertheless, we want to test our technique against network that may grow up to hundreds of hosts. Network simulators can not handle scenarios of such size, so we will use Python networkX library to evaluate the characteristics associated with large topologies. For some applications approximating a wireless mesh network with an abstract graph may be a simplification that is too far from reality. In our case instead, we rely on the existence of a proactive routing protocol running in the mesh network. We are not interested in physical layer and MAC layer performances (that are more sensitive to the simplifications introduced by graph analysis), we operate directly on the graph that the routing protocol generates, assuming that it is able to find neighbor nodes, to identify and use only symmetric links and to build the routing table from any source k to any destination j. This is perfectly compatible with, for instance, the widely used OLSR protocol. Note also that we assume the routing protocol uses a shortest-path metric. It is out of the scope of this paper to show it, but we believe that the same approach can be applied even when the routing is not a simple distance-vector. In this case the graph will be a weighted one where it is still possible to compute $m(i)$ and $t(i)$ taking into account the weights of each graph edge.

To test the performance of the *waterwall* we will use two metrics introduced in previous works [13], [15] and defined as follows:

- $M1(k,j)$**:** counts each false positives on the route from k to j, that is, it is incremented each time an unwanted packet is forwarded on the

path from the sender to the destination. It is normalized on the route length from k to j so it expresses the fraction of the path that p is able to reach before being filtered.

- $M2(k,j)$**:** counts each false positives end-to-end, that is, it is incremented each time an unwanted packet arrives to j. It is normalized to 1 so it represents the probability of arriving to destination j.

When averaged on every couple (k,j) $M1$ gives an estimation of the impact of false positives on the whole network traffic. For instance, when a node that has been infected by a worm starts a DoS attack against any host, M1 tells how much the *water-wall* is able to mitigate this attack in terms of wasted network resources.

$M2$ instead measures the inefficiency in filtering traffic directed against a certain host. In our scenario we imagine that the destination node j applies its own rule-set, so that $M2$ always goes to zero one step before the destination. We consider it since it is useful in other scenarios (for instance for intrusion detection , or when some traffic is forbidden by a network administrator but not all nodes support filtering) and it is used to compute $M1$.

$M2(k,j)$ can be defined as the probability of not being filtered on the whole path from k to j, that is equal to $P_a(k,j)$: we can then define $M(1)$ as the average number of hops that p is able to reach:

$$M1(k,j) = \Big(\sum_{i=0}^{sp_l(k,j)-1} sp_l(k,i) * M2(k,i) * P(k,i,j) + sp_l(k,j) * M2(k,j) \Big) \frac{1}{sp_l(k,j)}$$

The first term of the equation takes into account packets that are filtered before they arrive to destination (including on node k) while the second includes packets that arrive to the destination.

One more evaluation parameter we consider is the average end-to-end delay for every route in the network. For a network in which every node j has a rule-set of size $r_j = 30$, for each route we compute the average end-to-end delay introducing at each hop a delay d that depends on $Pf(k,i,j)$. The value of d is taken directly by the data measured on a real platform and reported in figure 2. The delay thus depends on the total number of nodes and on the value of the δ parameter.

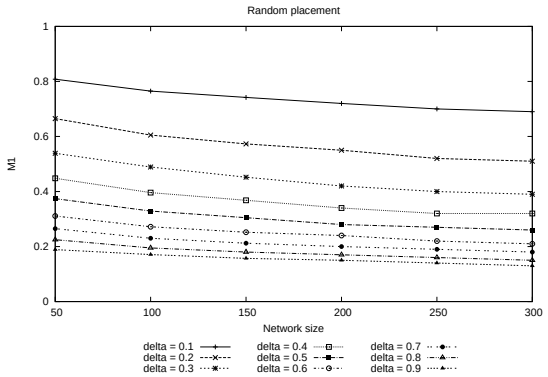

Figure 5: Metric M1 for increasing network size and δ ranging from 0.1 to 0.9.

The figures 5, 6, 7 report the value of the metrics M2, M1 and delay for a 2D topology with random placement of nodes increasing the network size and varying δ. The nodes are placed in a playground of growing size with constant spatial density of nodes and each node is connected to the neighbors that fall inside a radius of 70m.

We can see that, as expected, M1 and M2 decrease when δ is increased (recall that M1 and M2 measure false positives, so they are measures of badness). This is intuitive since a larger δ corresponds to less false positives. Less intuitive is the fact that given a certain δ a larger network has smaller values of M1 and M2. In the previous section we have shown that the values of $t(i)$ are smaller if i is close to the borders of the network, this is still true even in 2D topologies. As a consequence, the ratio $t(i)/m(i)$ is smaller in the periphery of the network as can be seen in the figure 3. In a 2D topology the border of the network is represented by nodes that are placed on the perimeter of the covered area and have fewer neighbors compared to the ones that are in center of the playground. If we keep the density constant

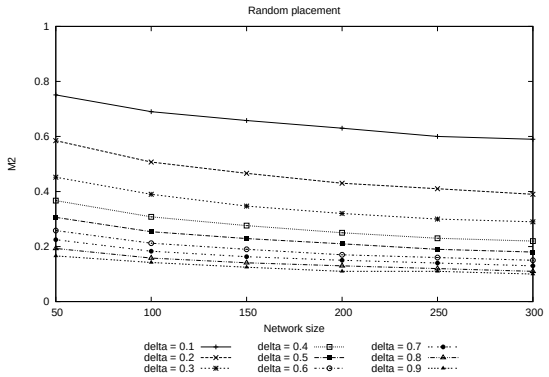

Figure 6: Metric M2 for increasing network size and δ ranging from 0.1 to 0.9.

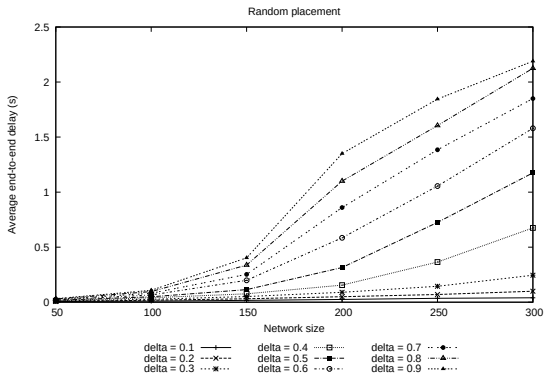

Figure 7: Estimated delay for increasing network size and δ ranging from 0.1 to 0.9.

and we increase the number of nodes we increase the covered area, and consequently its perimeter. But the perimeter of the network grows more slowly compared to the area, so the largest the network, the less relevant is the fraction of the nodes on the perimeter. As a consequence, a larger network will have a larger average t/m value and will filter more packets per hop. Figure 7 shows the expected delay for the networks under consideration. A larger δ corresponds to higher delays introduced at every hop.

To interpret these results, consider a network with 200 nodes and 30 rules per node, thus $||R|| = 6000$. With such a large rule-set, each hop would introduce a delay larger than 0.45 seconds, as can be seen in figure 2. If we consider that m in such a network has an average larger than 8, this would produce en average delay larger than 3.6 seconds, which makes the network unusable. Instead, with the *waterwall* approach we can configure the δ parameter in order to find the right equilibrium between latency and filtering efficiency, for instance, if $\delta = 0.4$ we obtain an average $M1$ lower than 50% and we keep the delay around 0.15 seconds. That is, we decrease the filtering efficiency to one half, but we reduce the delay of a factor of 24.

Still, if a higher performance of the firewall is needed with large network size, the delay introduced by the *waterwall* must be further reduced. In the next section we introduce a further optimization that, at the cost of a simple ordering function applied to the rule-set can further reduce the false positive rate.

3.3 Smart rule-set partition

When node i processes a packet p from k to j, it randomly chooses a position λ in R and uses a portion of the rule-set R_i starting from λ, of a size determined by $Pf(k, i, j)$. Packet p is tested against all the rules in R_i. The probability of evaluating the same rule twice in the path from source to destination is high since each choice of λ is independent at each hop. We can try to find a smarter way to chose λ in order to minimize the intersection between R_i at various steps. One way is to chose λ in function of specific network parameters of p and node i. A simple approach is to define λ as follows:

$$\lambda = (IP_{dest} \oplus IP_{src} \oplus (IP_{prot}|IP_{tos}|IP_{id}) \oplus \hat{IP_i})mod(||R||)$$

Where:

- IP_{dst} and IP_{src} are the destination and source IP of p
- IP_{prot}, IP_{tos} and IP_{id} are fields of the IP header that are immutable from source to destination

but their combination is unique for each packet (due to the identification field)

- $\hat{IP_i}$ is the IP address of node i with the bytes order inverted.

- $\oplus$ is the XOR operator, $|$ is the concatenation operator and *mod* is the modulo operation

The rationale of this choice is to produce a δ that changes from hop to hop depending on a unique parameter of node i, this way trying to spread the various choice of λ with a deterministic algorithm. Since the size of R_i is not predictable, we need to prevent that node i, when filtering two packets belonging to the same traffic flow chooses twice the same λ, or else, some portion of the R may never be covered. Thus, the choice of including the IP_{id} field that is unique for every IP packet.

For this approach to be applicable, every node must keep the rules in its rule-set in an ordered list. This ordering is not intended to speed-up the evaluation of the rule-set so it can be any ordering function, independent on the semantic contents of the rule. For instance, given the data structure that is used to store the rule in the operative system, an ordering based on a fingerprint on this data structure is sufficient.

If we are able to use minimum overlapping R_i sets than the M1 and M2 decrease more sharply with the distance from the source. The results obtained with this new constraint are reported in figure 8 and 9. Comparing 5 and 8 we can see that to obtain similar results a lower value of δ is sufficient, for instance to have M1 below 50% $\delta = 0.3$ is sufficient (even $\delta = 0.2$ is below 50% for network larger than 100 nodes) which corresponds in figure 7 to a tolerable delay even for a network with 300 nodes.

Note that in all the results we have shown so far M1 and M2 hardly reach values lower than 0.1. This is due to the fact that $P_f(k, i, j)$ at the first hop may not be equal to 1 even at the first hop. If larger values of δ were used (values larger than 1 are perfectly allowed) we can lower the false positive rate even more.

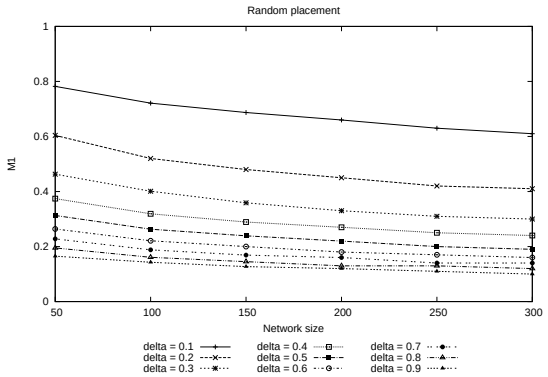

Figure 8: Metric M1 for increasing network size and δ ranging from 0.1 to 0.9, with ordered rule-set.

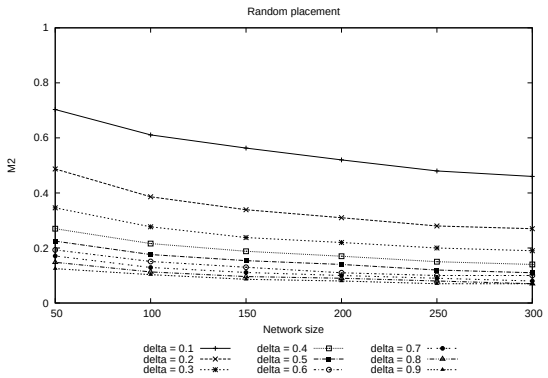

Figure 9: Metric M2 for increasing network size and δ ranging from 0.1 to 0.9, with ordered rule-set.

4 Centrality based filtering

In this section we present a different approach to the same problem. Instead of enforcing the *waterwall* on every node we identify the nodes that have a higher centrality in the network and enforce the whole global rule-set only on those nodes. This approach is imagined as a simpler alternative to the previous one for networks of smaller size. If we look at Fig. 2 we see that the performance of the filtering function dramatically decreases after a certain threshold. If the global

rule-set size is below that threshold, we can afford to enforce the whole rule-set in some nodes that will not introduce false positives. To reduce the overall delay we limit the numbr of nodes that participate to the *waterwall*. If we call F the set of filtering nodes we will try to reduce its size ($||F||$) selecting the nodes that are more *central* in the topology, while limiting the rate of false positives over the total traffic sent.

Estimating the centrality of a set of nodes in a network graph is a hard task for several reason we will explain, nevertheless we will show that using the OLSR protocol the centrality of a group of nodes can be approximated using the size of the MPR selector set of all the MPR nodes. Before we go deep in this subject we need to recall the basic notions about shortest path betweenness (the centrality measure we adopted) and the way the OLSR protocol works.

If $\sigma_{k,j}(i)$ is the set of all the shortest paths between nodes n_k and n_j passing across n_i and $\sigma_{k,j}$ is the set of all the shortest paths between n_k and n_j, the shortest path betweenness SPB($\cdot$) of n_i is defined as

$$\mathrm{SPB}(i) = \frac{1}{(N-1)(N-2)} \sum_{k \neq j, j \neq i} \frac{||\sigma_{k,j}(i)||}{||\sigma_{k,j}||} \quad (1)$$

Accordingly, the SPB($\cdot$) of a group of nodes F is

$$\mathrm{SPB}(F) = \frac{1}{(N-1)(N-2)} \sum_{k \neq j, j \neq i} \frac{||\bigcup_{i \in F} \sigma_{k,j}(i)||}{||\sigma_{k,j}||} \quad (2)$$

As in the previous approach the betweenness is defined excluding the end-points of a path. Using OLSR, node n_i has enough information to compute all the shortest paths to any other node n_j in the network. To identify a set of nodes F with a given group betweenness, n_i should compute its own betweenness, the betweenness of any other node n_j in the network and then solve a combinatorial problem to have one of the smallest sets possible. Once F has been identified n_i will be part of the *waterwall* only if it falls into F. Two issues make this approach inadequate, the first is that OLSR doesn't give enough information to node n_i to compute the shortest path between *any* couple of nodes (n_k, n_j) in the network (see the next section). The second is the complexity of such a computation, which must be repeated at every node at every modification of the network graph.

Before we describe the proposed solution it is necessary to highlight how OLSR works. As it is a very well treated subject in the literature, we recall here only the features important for our work.

4.1 OLSR Principles

In OLSR each node n_i periodically sends an HELLO message containing its symmetric one-hop neighbors. This is enough for every node to have the full knowledge of its two-hop neighborhood. Once the two-hop neighborhood is known, n_i will choose among its one-hop neighbors a subset of them that will be elected Multi Point Relays (MPR).

As defined in [9] the MPR set $M(n_i)$ is an arbitrary subset of its symmetric 1-hop neighborhood $N_1(n_i)$ that satisfies the following condition: every node in the 2-hop neighborhood $N_2(n_i)$ must have at least a symmetric link towards a node in $M(n_i)$. More intuitively, node n_i can reach any node in $N_2(n_i)$ through nodes in $M(n_i)$. It has been shown that identifying the minimal MPR set is NP-Hard, so OLSR introduces a heuristic to reduce the necessary computation that is effective in most cases [11].

Once n_i has selected its MPRs it communicates to each of them that it has become one of their *MPR selectors* setting the appropriate flag to its HELLO messages. Nodes that have been selected MPR behave as follows:

1. They periodically generate TC messages containing the list of their selectors;
2. They rebroadcast the TCs that are received from nodes that are their selectors.

The first point allows the construction of shortest path routing table and the second reduces the number of control messages compared to flooding. Moreover this procedure distributes globally another important information: the set of all the MPRs in the network and the size of the selector set for each MPR. In the OLSR RFC it is strongly suggested that MPR nodes are preferred over non MPR nodes as next-hop in routing tables, we expect the implementations to

follow this guideline. In OLSRv2 RFC [8] the choice of MPR nodes can be done with different algorithms but routing through MPRs is mandatory. If nodes n_i and n_j are direct neighbors they may talk directly even if none of them is an MPR.

Suppose that an MPR m has been selected by both nodes n_i and n_j. Node m sends TC messages containing both the IP addresses of its selectors. When node a n_k that is not in the two-hop neighborhood of m receives such a TC it will not be able to tell if n_i and n_j are direct neighbors since m does not propagate that information. Nodes n_k approximates the topology of the network other than its two-hop neighbors only with the links between MPRs and their selectors. For this reason a generic node n_k does not have enough information to compute the exact betweenness of any other node n_i in the network. Even if the deviation may be little, each node may compute a different F set.

4.2 Betweenness of groups of ranked nodes

We consider a network in which each node generates traffic flows to random destinations, each node is aware of the global rule-set but, contrarily to the previous approach only a subset made of the most central nodes will enforce the whole rule-set (and be part of the *waterwall*) while the others will simply forward the traffic. The key observation is that the selection of node n_i as MPR is an indicator of its local betweenness. Let S_i be the selector set of node n_i, and $||S_i||$ its size. If n_i is not an MPR then $S_i = \emptyset$ and $||S_i|| = 0$.

If we set F to the set of all the MPRs, then only the traffic that flows between two non-MPR neighbors will not be filtered. Moreover, the larger is the selector set of n_i, the more n_i is important as an intermediary in its neighborhood. This intuition is confirmed by the results of Fig. 10 where the betweenness of the MPRs of random networks with 100 nodes is plotted versus $||S_i||$. The betweenness grows with $||S_i||$. Topologies where some MPRs have large $||S_i||$ are rare, so for large $||S_i||$ the data is noisy, still the trend is clear.

If we desire to reduce F we can remove MPR nodes

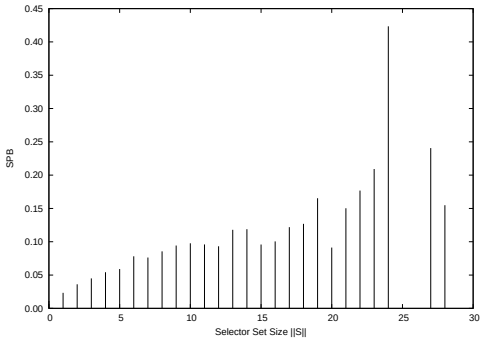

Figure 10: Betweenness centrality for MPR nodes ranked by $||S_i||$ on 20 runs for a 100-nodes topology

starting from the ones that have small $||S_i||$. In practice, we enforce the filters on the group of MPR nodes that have $||S_i|| > t$. But how this threshold relates to the rate of false positives? Can we find a relationship that is scalable on the network size and valid on different topologies?

In section 4.3 we identify a normalized threshold t' with the remarkable property of being almost invariant of the networks size and topology, offering a practical means for any node to independently decide if it is part of F or not.

4.3 Simulation Results

We consider an area covering 500×500 m with five different topologies:

Uni – Nodes are randomly placed in the area with a uniform distribution.

NonUni – The nodes follow a Poisson distribution centered on the center of the playground (density decreasing with distance).

Grid – The nodes are initially distributed on a regular grid so that each node will have at most 8 one-hop neighbors. A random noise on both X and Y axis is added for at most 20% of the inter-node distance.

Clust – Four clusters are created, each cluster contains one fourth of the nodes. The center of each

cluster is fixed and placed on the diagonals of the square area, within clusters nodes follow a Poissone distribution.

OBST – Three obstacles are inserted in the area. Nodes are randomly placed outside the obstacles and the penetration of the wireless signal in the obstacles is extremely limited.

For each topology we simulate 5 cases with increasing number of nodes: 36, 49, 64, 81 and 100. The number of nodes are perfect squares to generate coherent grids. In total 25 networks with distinct features are considered. Each node in the simulation uses an omnidirectional antenna and has an approximate maximum communication radius of 75 m with a dual-slope path-loss model. Ray-tracing is implemented in order to limit the penetration of the wireless signal in the obstacles to few meters.

Fig. 11 reports metric $M2(t)$, for the sake of readability we report only the curves relative to 36 and 100 nodes, as all other curves fall between these.

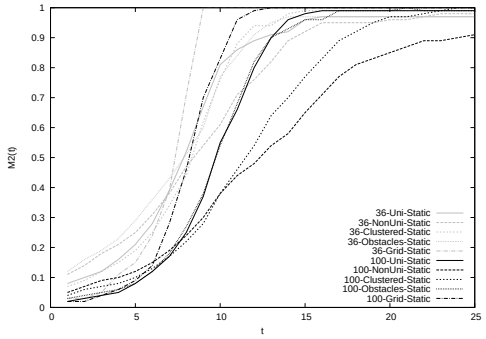

Figure 11: Metric $M2(t)$ in all topologies for 36 and 100 nodes

$M2(1)$ is due only to traffic generated by a non-MPR node with final destination a non-MPR node that is a neighbor of the generator. In a small network, the fraction of 1-hop neighbors of a node over the total is higher than compared to larger networks, so $M2(1)$ is higher for small networks. This is one-hop traffic that can be filtered only at the destination node, and represents in large networks a low percentage of the overall traffic. Since we are interested in the betweenness of nodes we will not consider this fraction of the overall traffic in the next graphs.

As t increases, the difference between the behaviour of a small and large network increases too and a stronger dependency on the topology arises.

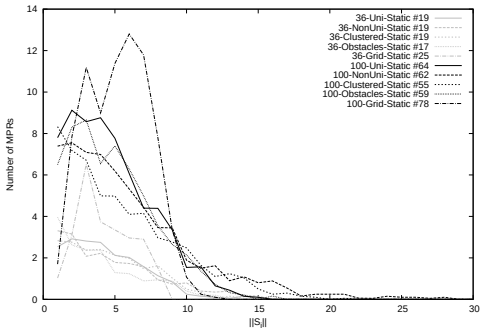

Figure 12: Number of MPR for each selector set size $||S_i||$

A straight explanation is that a larger network has more MPRs than a smaller one and a more dense network has S_i on average than a less dense one. This trend can be seen in Fig. 12 where for each scenario is reported the histogram of the number of MPRs versus S_i. It can be seen that the networks differ in the right limit (the highest selector set size), in the total number of MPRs (the integral of the curve) and in the shape of the curves. As a consequence, using the same threshold doesn't correspond to a similar behaviour: a threshold t set on S_i is not a scale and topology free parameter to select F.

The authors of [11] have shown that when the density of a network increases the number of MPR nodes grows slowly, while the average $||S_i||$ increases. This means that given a certain covered area, increasing the density will not increase the number of MPRs but will increase the selectors per MPR. As a consequence, the value of t to achieve a certain filtering rate strongly depends by the network size while the number of MPRs has a weaker dependency, if instead of making F depend on t we make it depend on the fraction of MPR nodes over the total that are included in $F(t)$ we expect the curves in 11 to be closer.

Let $N_{\mathrm{MPR}}(t)$ be the number of MPRnodes n_i that have $||S_i|| = t$ and normalize it to obtain a complementary cumulative distribution:

$$N'_{\mathrm{MPR}}(t) = \frac{\sum_{i=t}^{i=t_{max}} N_{\mathrm{MPR}}(i)}{\sum_{i=1}^{i=t_{max}} N_{\mathrm{MPR}}(i)}$$

where t_{max} is the largest $||S_i||$. $N'_{\mathrm{MPR}}(t)$ is the fraction of MPR nodes with $||S_i|| \geq t$. $N'_{\mathrm{MPR}}(t)$ is plotted in Fig. 13.

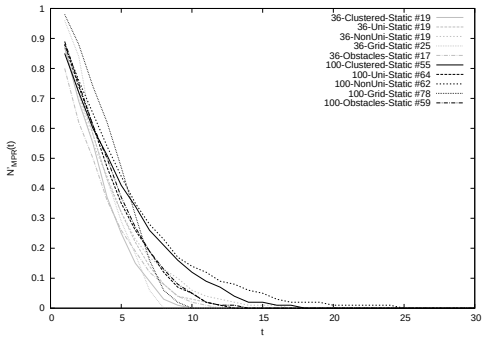

Figure 13: Rescaled number of MPRs for each selector set

We define $t'(t) = N'_{\mathrm{MPR}}(t)$. When $t = 1$ then $t'(1) = 1$ and when $t = t_{max}$ then $t'(t_{max}) = 0$. Furthermore we observe that the following two equivalences hold:

$$M2'(t'(t)) \equiv M2(t)$$

and we redefine the set F' as:

$$F'(t'(t)) \equiv F(t)$$

In practice, instead of using the dependence of $M2$ from t, that is too much scenario dependent we have formalized a dependence of $M2$ on t' which we have shown in Fig. 13 is less scenario-dependent. When $t' = 1$ F is made of all the MPR nodes, for other values, t' represents a fraction of MPR nodes in F. Note that every node n_i is able to compute $t'(t)$ without much effort since n_i knows all the MPRs and their selector sets so they can easily compute $t'(t)$.

The results are plotted in Fig. 14 which shows a much more regular behaviour and curves that are

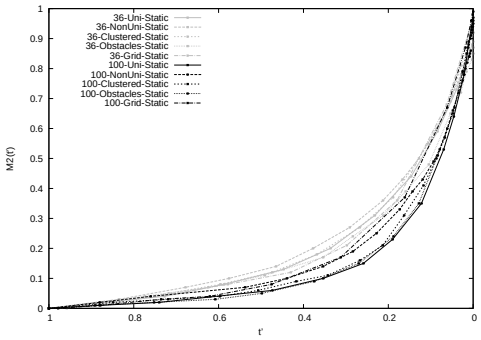

Figure 14: Rescaled metric M2'(t')

closer than before. The results for all the scenarios are summarized in figure Fig. 15 where the color (grey) area is limited by the curves that have the largest and the smallest area below them. It includes all points of all the simulations for every network size while the dotted curve is the spline generated using the average values of all the curves. It can be seen that using t' instead of t we can define an upper threshold that is valid for **anyone** of the considered scenarios (without having to discriminate on the topology or on the size) close to the real performance.

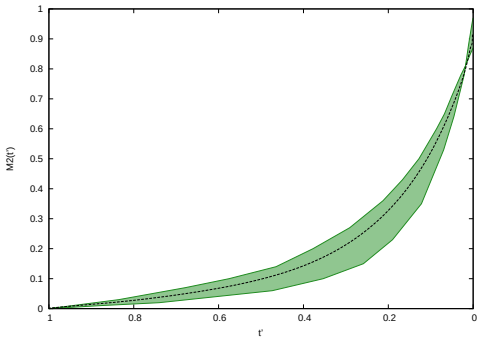

Figure 15: Average and envelope of M2'(t')

In Fig. 16 is reported also metric $M1'$, that is, metric $M1$ with the same rescaling of the x axis applied to have $M2'$. The metric has higher values than $M1$ and the trend of the curves is different from Fig. 14, for a large portion of the x axis the curves show a trend close to linear. This can be easily explained

observing that each traffic flow is composed of multiple hops. For each false positive in $M2$ several false positives are generated in $M1$, moreover, $M2 = 0$ does not imply $M1 = 0$. The value of $M1$ depends on how close to the source node a flow is filtered, which, compared to $M2$ is more influenced by the size of F than by the position of the nodes included in F.

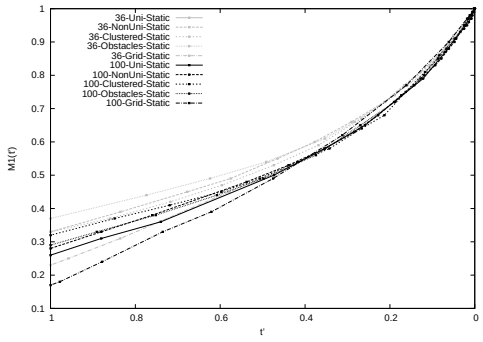

Figure 16: Rescaled metric M1'(t')

4.3.1 Practical implications

The practical implication of this analysis is that the average curve in figure 15 can be taken as a reference by the network manager of a wireless mesh network (or by any automated algorithm) to set the size of F in order to achieve the necessary betweenness and consequent precision for the firewall, network monitor or IDS. For instance, if the manager wants to apply a very fine grained filter achieving 90% of the precision, he can set $t' = 0.5$ independently of the size and topology of his network.

The lowest is t' the smallest is the size of F, the less resources are spent. To have an estimation of how much nodes are necessary to achieve the wanted group betweenness in table 2 we report the size of the MPR sets and the average hop-count corresponding of the results of fig. 14.

It can be seen that $t' = 0.5$ will correspond to only 25% of the nodes in the best case and to 40% the worst, with a significant resource saving at the cost of only 10% accuracy.

	36 nodes		100 nodes	
Scenario	MPS	hop-count	MPS	hop-count
uniform	19	2.7	64	4.4
nonuniform	19	2.2	62	3.3
clustered	19	2.8	55	3.9
obstacles	17	2.6	59	4.5
grid	25	3	78	3.9

Table 2: MPR set size (MPS) and average hop-count per simulation scenario

5 Related Works

Distributed firewalling has been explored in literature, an initial model has been proposed by Bellovin et al. in [10] where the firewall is moved from a bastion host to the endpoints of a still traditional centralized network. Recently, the subject has received more attention, Bellovin proposed a distributed policy enforcement platform [21],[19],[20],[22], as well as other authors [1]. Those works are not focused on the complexity introduced by large rule-sets. Other works focus on the application of hash functions to speed-up rule-matching [5] or on limiting the nodes that enforce the firewall [13]. None of these works focus on techniques to reduce single rule-sets. The work that has more in common with this one is [17], where a cache is used to filter only using a subset of the most recently matching rules. The cache is split in two halves, each one regulated with a different policy in order to ensure efficiency and fairness. This approach requires a feedback from the nodes that generate the rule-sets in order to organize the cache, moreover, as every caching strategy its performance depends on the characteristics of the underlying traffic.

In our work we assume that the default filtering policy is to forward a packet. Packets are denied only when there is a rule that matches them. This approach is more viable in a mesh networks than a deny-by-default one. For the last one to be usable the rule-sets must be perfectly synchronized and updated, otherwise there is the risk of dropping allowed traffic. Nevertheless, in networks where a high security level is required this approach can be used [18, 2], but it does not match our network scenario.

SPB has been used in social science since the late seventies to identify influential individuals in social networks [7]. Its adoption in the context of networking, together with other metrics derived by social studies is much more recent [12]. The extension of SPB to group betweenness has been shown to be a NP-Hard task in the general case [6], nevertheless, it is at the base of some works whose aim is to find a favourable subset of nodes in a network to perform network monitoring [4, 16]. Our approach differs from the ones in the literature since we do not take into consideration the traffic matrix, that is very hard to estimate, and we target a specific and widely used routing protocol for mesh networks, OLSR, whose features, we discovered, allow betweenness inference with no signalling or computation overhead. Moreover, as we explored in [14], OLSR MPR selection can be modified and improved to obtain global characteristics of the network, and the TC messages of OLSR can be easily extended to support signalling for *waterwall* coordination. The approach based on distributing the filtering function along the route from the source to the destination, has never been explored before.

6 Conclusions

In this paper we have introduced a new model to perform distributed firewalling in mesh networks that takes advantage of the multi-hop nature of those networks to share the load needed for the filtering function. To stress the difference with a traditional firewall, we chose the term *waterwall* indicating a fluid and distributed network function, instead of a single filtering host. We proposed two approaches to enforce the *waterwall* in a wireless mesh network, both can be used to greatly reduce the unwanted traffic in a mesh network.

We used two different ways to validate the proposed techniques, graph analysis for large networks and computer simulation for smaller networks that are more computationally tractable. In both cases we show significant improvements over a traditional firewalling approach, being able to scale up to thousands of rules. The source code used will be made available on the main website financing this project, `www.pervacy.eu`.

As future work we intend to port the first filtering approach to the network simulator, in order to test and optimize the enhancement described in 3.3. Moreover we will merge the two approaches in a single configurable platform and embed it in some widely used routing protocol implementation, such as OLSR in order to make tests on real networks.

References

[1] Alicherry, M., Keromytis, A., Stavrou, A.: Distributed firewall for manets. Tech. rep., Columbia University Computer Science Technical Report Series (2008)

[2] Alicherry, M., Keromytis, A.D., Stavrou, A.: Evaluating a collaborative defense architecture for manets. In: Proceedings of the 3rd IEEE international conference on Internet multimedia services architecture and applications, IMSAA'09, pp. 229–234. IEEE Press, Piscataway, NJ, USA (2009)

[3] Cerda-Alabern, L.: On the topology characterization of guifi. net. In: Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on, pp. 389–396. IEEE (2012)

[4] Dolev, S., Elovici, Y., Puzis, R.: "Routing betweenness centrality". *Journal of the ACM (JACM)"* **57**(4) (2010)

[5] Fantacci, R., Maccari, L., Ayuso, P., Gasca, R.: Efficient packet filtering in wireless ad hoc networks. Communications Magazine, IEEE **46**(2), 104 –110 (2008). DOI 10.1109/MCOM.2008.4473091

[6] Fink, M., Spoerhase, J.: "Maximum Betweenness Centrality: Approximability and Tractable Cases". In: *WALCOM: Algorithms and Computation*, *Lecture Notes in Computer Science*, vol. 6552. Springer Berlin Heidelberg (2011)

[7] Freeman, L.C.: "A Set of Measures of Centrality Based on Betweenness". in *Sociometry* **40**(1) (1977)

[8] Herberg, U., Clausen, T., Jacquet, P., Dearlove, C.: Draft Request for Comment, "The Optimized Link State Routing Protocol version 2", revision 17. URL http://tools.ietf.org/html/draft-ietf-manet-olsrv2-17

[9] IETF Network Working Group: Request for Comments: 3626, "Optimized Link State Routing Protocol (OLSR)" (2003). URL http://tools.ietf.org/html/rfc3626

[10] Ioannidis, S., Keromytis, A.D., Bellovin, S.M., Smith, J.M.: Implementing a distributed firewall. In: ACM Conference on Computer and Communications Security. Athens, Greece (2000)

[11] Jacquet, P., Laouiti, A., Minet, P., Viennot, L.: "Performance Analysis of OLSR Multipoint Relay Flooding in Two Ad Hoc Wireless Network Models". In: *The second IFIP-TC6 NETWORKING Conference* (May 19-24 2002, Pisa, Italy)

[12] Katsaros, D., Dimokas, N., Tassiulas, L.: "Social network analysis concepts in the design of wireless Ad Hoc network protocols". *IEEE Network* **24**(6) (2010)

[13] Maccari, L.: "A Collaborative Firewall for Wireless Ad-Hoc Social Networks". In: *Proceedings of the 9th International Conference on Security and Cryptography (Secrypt)* (2012)

[14] Maccari, L., Lo Cigno, R.: How to Reduce and Stabilize MPR sets in OLSR networks. In: 2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 381–388 (2012)

[15] Maccari, L., Lo Cigno, R.: Privacy in the pervasive era: A distributed firewall approach. In: 8th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2012), Poster Session (2012)

[16] Ou, P., Li, Z.: "A Variant betweenness Centrality Approach towards Distributed Network Monitoring". In: *Proceedings of the Fourth International Symposium on Parallel Architectures, Algorithms and Programming (PAAP).* IEEE Computer Society (December 9-11 2011, Tianjin, China)

[17] Taghizadeh, M., Khakpour, A., Liu, A., Biswas, S.: Collaborative firewalling in wireless networks. In: Proceedings of INFOCOM 2011. Joint Conference of the IEEE Computer and Communications Societies. (2011)

[18] Zhang, H., DeCleene, B., Kurose, J., Towsley, D.: Bootstrapping deny-by-default access control for mobile ad-hoc networks. In: Military Communications Conference, 2008. MILCOM 2008. IEEE, pp. 1 –7 (2008). DOI 10.1109/MILCOM.2008.4753174

[19] Zhao, H., Bellovin, S.M.: Source prefix filtering in ROFL. Tech. Rep. CUCS-033-09, Department of Computer Science, Columbia University (2009)

[20] Zhao, H., Bellovin, S.M.: High performance firewalls in MANETs. In: International Conference on Mobile Ad-hoc and Sensor Networks, pp. 154–160. IEEE Computer Society, Los Alamitos, CA, USA (2010)

[21] Zhao, H., Chau, C.K., Bellovin, S.M.: ROFL: Routing as the firewall layer. In: New Security Paradigms Workshop (2008). A version is available as Technical Report CUCS-026-08

[22] Zhao, H., Lobo, J., Roy, A., Bellovin, S.M.: Policy refinement of network services for MANETs. In: The 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011). Dublin, Ireland (2011). To appear