Maccari and Lo Cigno *EURASIP Journal on Wireless Communications and Networking* 2013, **2013**:225
http://jwcn.eurasipjournals.com/content/2013/1/225

EURASIP Journal on
Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

Waterwall: a cooperative, distributed firewall for wireless mesh networks

Leonardo Maccari* and Renato Lo Cigno

Abstract

Firewalls are network devices dedicated to analyzing and filtering the traffic in order to separate network segments with different levels of trust. Generally, they are placed on the network perimeter and are used to separate the intranet from the Internet. Firewalls are used to forbid some protocols, to shape the bandwidth resources, and to perform deep packet inspection in order to spot malicious or unauthorized contents passing through the network. In a wireless multihop network, the concept of perimeter is hard to identify and the firewall function must be implemented on every node together with routing. But when the network size grows, the rule-set used to configure the firewall may grow accordingly and introduce latencies and instabilities for the low-power mesh nodes. We propose a novel concept of firewall in which every node filters the traffic only with a portion of the whole rule-set in order to reduce its computational burden. Even if at each hop we commit some errors, we show that the filtering efficiency measured for the whole network can achieve the desired precision, with a positive effect on the available network resources. This approach is different from the protection of a space behind a wall: we use the term *waterwall* to indicate a distributed and homogeneous filtering function spread among all the nodes in the network.

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, middleboxes as NATs and firewalls are normally installed on the network boundary to separate trusted portions of the network from the global Internet and 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, in which a collection of subnets are interconnected through a backbone of mesh nodes, but each subnet is only loosely coupled with the others. Moreover, 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 [1]. As a concrete example, community networks [2,3] 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 http://guifi.net and http://awmn.gr) are examples of how this technology integrates with standard networks and how successful this approach can be. Future advances will open new possibilities for this technology [4].

In this paper, 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 is used to defend the local network from attacks, to shape the access to the Internet across a connection, or to forbid the access to certain logical resources. If all nodes share their rule-sets and enforce them also on the outgoing traffic, the traffic is not filtered at the destination but directly at the source. This reduces the waste of network resources but forces each node to filter 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 to their source node in order to save network resources.

*Correspondence: maccari@disi.unitn.it
Department of Information Engineering and Computer Science (DISI), University of Trento, Via Sommarive 5, Povo, Trento 38123, Italy

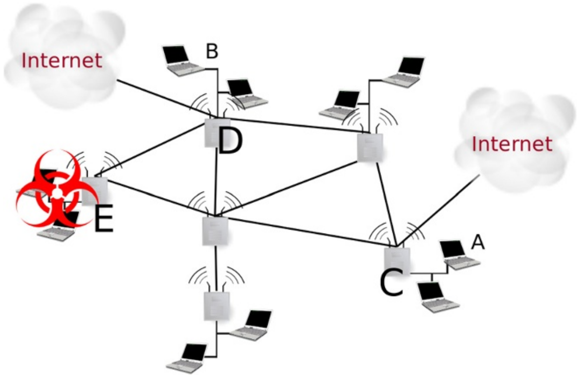

Figure 1 A common mesh network structure.

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

2 Related works and motivation

Distributed firewalling has not received much attention in the literature, but an initial model has been proposed by Bellovin et al. in [5], where the firewall was moved from a bastion host to endpoints in a traditional architecture network. Recently, the subject has been investigated with more attention. Bellovin again, followed by other authors, proposed a distributed policy enforcement platform [6-10]. These 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 [11-13] or on limiting the nodes that enforce the firewall [14]. None of these works focus on techniques to reduce the rule-sets on single nodes.

The work whose idea comes closer to the contribution of this paper is [15], where the most recently matching rules are stored in a cache that is used to enforce filtering, thus using only a subset of the entire rule-set. The cache is split in two halves, with 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 with every caching strategy, its performance depends on the characteristics of the underlying traffic.

Our work assumes that the default filtering policy is to forward a packet. Packets are dropped only when there is a rule that matches them. This approach is more viable in a mesh network 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 legitimate traffic. In networks where a high security level is required, also deny-by-default rule-set can be used as in [16,17], but it does not match our network scenario.

Additional similarities can be found in the field of intrusion detection since filtering with large rule-sets presents the same difficulties of traffic inspection with a large database of fingerprints. An approach to distributed intrusion detection systems (IDS) like [18] could benefit from the solution we propose. More affinities can be found in [19] that, as our work, exploits the distributed nature of an *ad hoc* network to spread the IDS function over the entire network.

2.1 Motivation

Figure 1 shows a widely used configuration for a wireless mesh network where a set of mesh routers interconnects separated local area networks (LANs). Each LAN has its own internet protocol (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 one LAN to another, depending on the kind of routing protocol they may or may not maintain their initial IP addresses to keep their sessions alive. Finally, some of the LANs have a direct access to the Internet and share it with 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 he 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 finer granularity, 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 others 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 traffic types 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 network service provider. Again, traffic can be forbidden or it can be limited to a certain maximum bit rate.

Now imagine that a node in the network labeled *E* starts an attack against, let us say, host *A*. This may be due to a malicious user or to a virus that took control of a host in the network and starts a denial of service (DoS) or a brute force attack. We can 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 by configuring a firewall on each mesh node in order to filter the traffic directed to its LAN. The first three use cases can be tackled by setting up a mixture of layer-4 and application layer firewall rules on the mesh router in *C*, which will drop or shape some traffic. The fourth one can be approached with dynamic rules that are activated when the firewall detects an anomaly in the usage of the resources, for instance, an abnormal number of internet control message protocol packets. Modern 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. Clearly, the malicious traffic coming from network *E* will still traverse the mesh network and subtract useful resources to the other allowed communications. Considering that in a mesh network the available bandwidth is shared between upload and download, this can severely impact not only 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 itself of the network is very 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 publish its rule-set and collect all the other rule-sets in a global rule-set. Then it will enforce it directly on the packets that it is forwarding so that the traffic is filtered as close as possible to the source. This approach indeed protects not only the resources of each LAN but also the shared resources of the mesh network. It's not the goal of this paper to investigate how the rule-sets are securely distributed, in the simplest case rule-sets can be known in advance and every node just sponsors the identifier (ID) of one or more predefined rule-set in routing messages.

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 up of thousands of nodes[a] and used by tens of thousands of users that daily access the network from various places (see [20] 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 us say, 30 rules each? The result will be a global rule-set with 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 designed for minimal energy consumption. The most used products 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 cannot easily handle a rule-set made up 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 preprocessed 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 preprocessing of the rule-set but speeds up the lookup time during the routing decision. It is convenient when the rule-set is mostly static and when the hardware is powerful enough for the preprocessing. In the case we consider, rules can be dynamically generated, nodes can be added to or removed from the network, and links may be temporarily unavailable. Each of these events will change the rule-sets or the network topology (and consequently add/remove rule-sets associated with nodes). Assuming that the nodes are powerful enough to perform the preprocessing, they would spend most of their CPU time repeating this task.

Techniques based on complex data structures, such as trees or graphs, can be used instead of using a linear list. The more complex the data structure, the more memory and preprocessing are needed. The less complex the data structure, the less the technique will be flexible and high performing. 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 filtering is still cumbersome. This gets even worse with networks based on IPv6 addresses. Both these approaches are hardly applicable when the rules do not match IP addresses and TCP/UDP ports but layer-7 data inside a packet.

In this work, we take a different direction. We keep the simplicity of linear lists, but we exploit the cooperative nature of mesh networks to reduce the overhead for each single node.

2.2 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. Fifty percent of the rules matches the network and transport layer fields; the rest matches the packet contents at layer 7. Contrary to the results we obtained in a previous work [14], where the tests were carried without traffic, the measures have been taken when the node is under a load of 1 Mbit/s.

With up to 3,000 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 5,000 rules. Since this delay is introduced by every node for every hop, the total round-trip time in a mesh network using large rule-sets makes the network unusable. Filtering is simply not a function that can be introduced 'for free' when the rule-sets get large.

3 Filtering based on route length

Consider a network N like the one in Figure 1 where a proactive routing protocol is running (from now on, we refer to mesh nodes simply as 'nodes'). Each node j is connected to a subnet, and for each node j, there 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 $R = \bigcup_j r_j;\ \forall j \in N$. The routing table of any 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 configuration of a mesh network configured, for instance, with optimized link state routing (OLSR) protocol [21].

Now consider a packet p coming from the subnet of node k that is forwarded by node i and is destined to the subnet of node j. Assume that 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 simplest solution is to enforce the whole R directly in k. This solution has two drawbacks: it is impossible if R is made up of thousands of rules for the considerations introduced in Section 2.2, 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 the destination[b]. 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 aim to use larger rule-sets for nodes close to the source and smaller rule-sets for nodes far from the source. The definition of

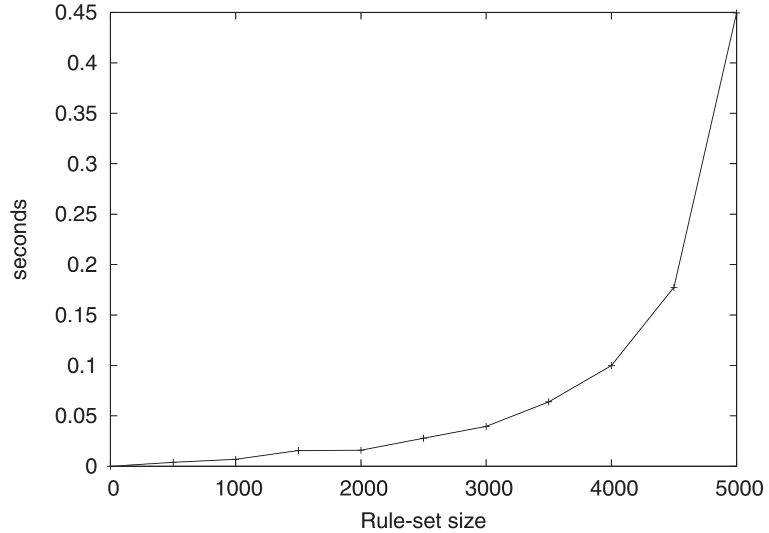

Figure 2 Delay introduced by a growing rule-set size when the host forwards approximately 1 Mbit/s of traffic.

the strategy behind this intuition, however, requires some more discussion and formalization. Table 1 contains a set of definitions that are used (and detailed) in the rest of the paper.

First of all, how should a node i estimate its distance from the source node k of a packet p with destination to j? The simplest way is to look at the time-to-live (TTL) field in the IP header, but it is also the easiest to circumvent. The attacker could simply forge packets with a low TTL and avoid the waterwall to be effective.

Another way is to use the distance from the source node k to i, but the attacker can set the source IP to the address of another node w and, contrary to what happens in 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 cannot 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.

What we propose is that each node uses a subset of rules R_i whose size depends 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 P^f as the probability that node i filters a packet going from node k to node j:

$$P^f(k,i,j) \triangleq \begin{cases} \frac{\mathrm{sp}_l(i,j)}{m(i)}\delta & \text{if } \mathrm{sp}_l(i,j) \le m(i) \\ 1\cdot\delta & \text{if } \mathrm{sp}_l(i,j) > m(i) \end{cases} \qquad (1)$$

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 $P^f(k,i,j) \times ||R||$, ensuring that $P^f(k,i,j)$ is the probability that i filters p. If R is organized as a linear list, this can be implemented as 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 $\frac{\mathrm{sp}_l(i,j)}{m(i)}$ decreases; in contrary, when i is close to k, the value of $P^f(k,i,j)$ is close to $1\cdot\delta$. We also define $t(i)$ as the value of $\mathrm{sp}_l(i,j)$ averaged on all routes passing through i between any couple (k,j). If we call $C(i)$ the set of all the couples (k,j) for which $i \in \mathrm{sp}(k,j)$, then

$$t(i) \triangleq \sum_{(k,j)\in C(i)} \frac{\mathrm{sp}_l(i,j)}{||C(i)||}. \qquad (2)$$

Table 1 Definitions and formal notation

Notation	Definition
N	The set of nodes in the network
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 the shortest path sp(*i*, *j*)
δ	A parameter that determines the maximum size of a rule-set enforced on a single node
m(*i*)	The average distance of node *i* from all the other nodes in *N*
r	The average size of a rule-set used by a node in *N*
R	The global rule-set, i.e., the union of all rule-sets
t(*i*)	The average sp$_l$(*i*, *j*) computed on node *i* for every packet with source *k* and destination *j* and for every (*k*,*j*) for which *i* ∈ sp(*k*, *j*) (see Equation 2)

To understand how our approach scales with the size and shape of the network graph, we have to understand the behavior of $t(i)$. Let us define m and t as the average $m(i)$ and $t(i)$ computed on every node, respectively; m is the average number of hops in the network, and t is the average number of hops remaining after a packet is forwarded by any node, averaged on all the nodes. Intuitively, t must 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 of linear topology, then we will analyze a more complex two-dimensional (2D) topology.

3.1 1D Linear topology

As a clarifying example, we take a linear topology with 10 nodes and 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 periphery 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 average distance from the other nodes is larger than when i is in the center, so $m(i)$ is higher on the periphery. It is also easy to see 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 make the ratio $t(i)/m(i)$ decrease for nodes close to the extreme ends of the network. In the figure, instead, we plot $t(i)$ including also the packets generated by node i, which increases the values of $t(i)$ on the periphery. This takes into account that in our scenario, each node is a gateway for its own subnet, so the first hop is counted in its own 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

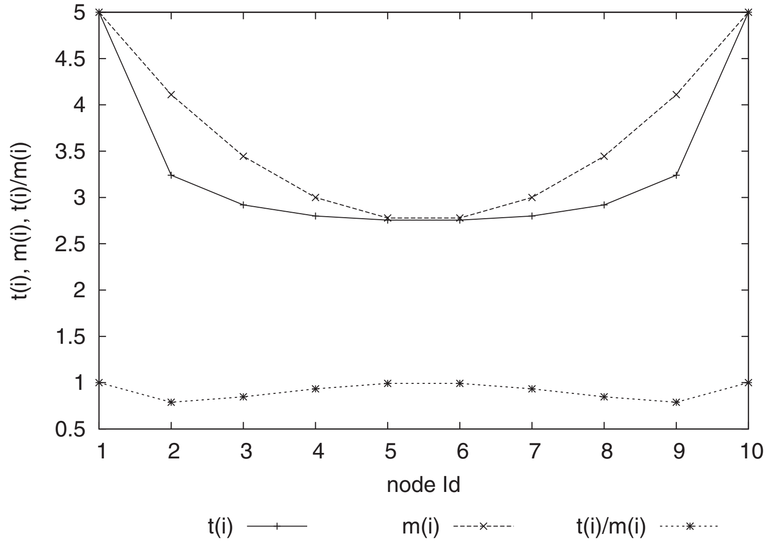

Figure 3 Values of $t(i)$, $t(m)$, and their ratio on the sample linear topology with 10 nodes. Each node is identified by an ID corresponding to its position in the line.

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 less congested; in any case, packets will be filtered at the destination.

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

$$P_h^f(k, h, j) = P^f(k, i, j) \text{ where } \text{sp}_l(k, i) = h. \tag{3}$$

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

$$P_a(h) = \prod_{i=0}^{h-1}(1 - P_h^f(i)) \tag{4}$$

In Figure 4, we report P_a for the same network considered in Figure 3 when $\delta = 0.5$.

The diameter of the network is equal to nine hops. When the packet arrives at the destination, it is filtered with a destination-specific rule-set, 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. Figure 4 shows that we obtain indeed the desired effect: the chances of a packet to be filtered are higher close to the source 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 IDs, so the results are obtained by means of simple algebra. When the network topology is defined on a 2D plane, more complex instruments must be used. The most suitable instrument to study the behavior of a mesh network with a 2D topology is computer simulations; nevertheless, we want to test our technique against networks that may grow up to hundreds of nodes. Network simulators cannot handle scenarios of such size; thus, we 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, 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, and we use NetworkX functions

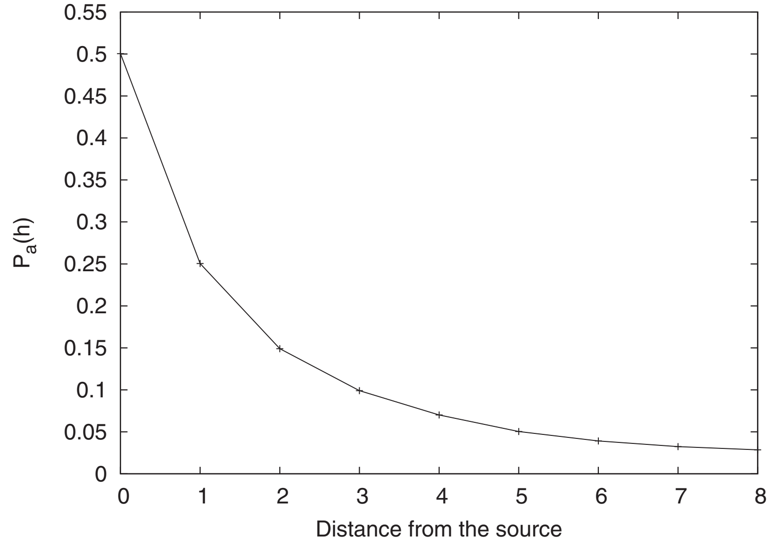

Figure 4 The values of $P_a(h)$ on the linear topology with 10 nodes.

in order to compute the values of $\mathrm{sp}_l(i,j)$ directly on the graph. 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 minimum hop. 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 [14,22] and defined as follows:

- $M_1(k,j)$. It 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.
- $M_2(k,j)$. It 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 unwanted traffic to arrive to destination j.

When averaged on every couple (k,j), M_1 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 another host, M_1 tells how much the waterwall fails to mitigate this attack in terms of wasted network resources.

M_2 instead measures the inefficiency in filtering traffic directed against a specific host. In our scenario, the destination node j applies its own rule-set so that M_2 always goes to zero when p arrives to its destination. But 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).

$M_2(k,j) = P_a(k,j)$ as it is the probability of not being filtered on the whole path from k to j. M_1 is defined as the average number of hops that p makes before being discarded:

$$M_1(k,j) = \left(\sum_{i=0}^{\mathrm{sp}_l(k,j)-1} \mathrm{sp}_l(k,i) \times M_2(k,i) \times P^f(k,i,j) + \mathrm{sp}_l(k,j) \times M_2(k,j) \right) \frac{1}{\mathrm{sp}_l(k,j)} \tag{5}$$

The first term of the equation takes into account packets that are filtered before they arrive to the destination (including node k). It is the sum of the path length from k to i, multiplied by the probability of reaching i and multiplied again by the probability of being filtered on node i. The second terms takes into consideration the packets that arrive to the destination j.

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 every node a processing delay d that depends on $P^f(k,i,j)$. The value of d is taken directly from the data measured on a real platform and

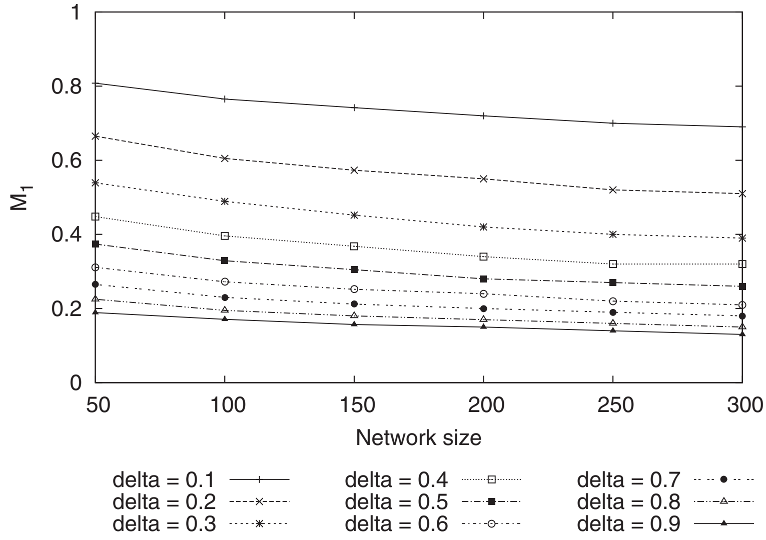

Figure 5 Metric M_1 for increasing network size and δ ranging from 0.1 to 0.9.

reported in Figure 2. The delay thus depends on the total number of nodes and on the value of the δ parameter.

Figures 5, 6, and 7 report the value of the metrics M_1, M_2, and delay for a 2D topology with random placement of nodes, increasing the network size and varying δ. The nodes are placed in an area of growing size with constant spatial density of nodes, and each node is connected to the neighbors that fall inside a radius of 70 m. Using NetworkX primitives, we are able to compute the shortest paths on the considered graphs and compute the equations we have defined so far.

We can see that, as expected, M_1 and M_2 decrease when δ is increased (recall that M_1 and M_2 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 M_1 and M_2. In the previous section, we have shown that the values of $t(i)$ are smaller if i is close to the periphery of the network; this is true also 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 Figure 3. In a 2D topology, the periphery

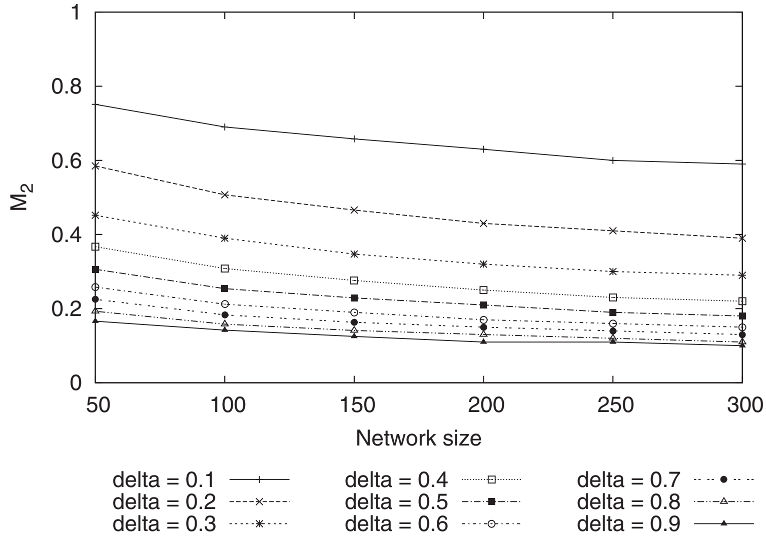

Figure 6 Metric M_2 for increasing network size and δ ranging from 0.1 to 0.9.

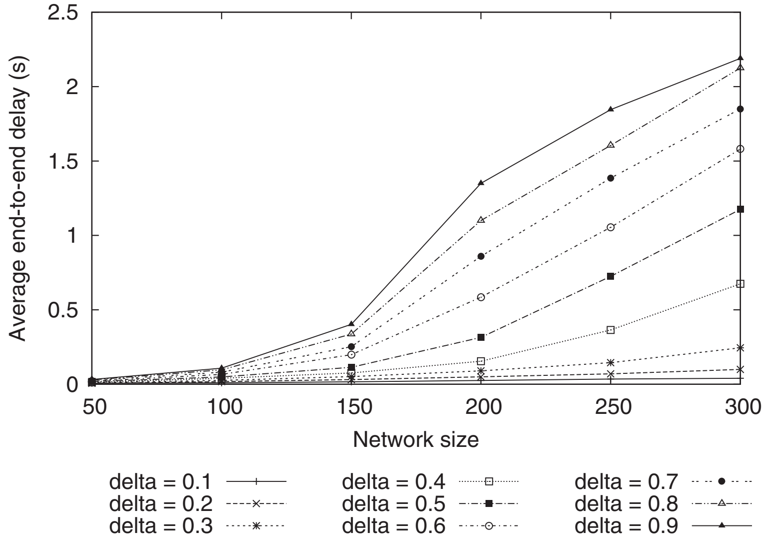

Figure 7 Estimated delay for increasing network size and δ ranging from 0.1 to 0.9.

of the network is represented by nodes that are placed on the perimeter of the covered area and that have fewer neighbors compared to the ones that are at the center of the area. If we keep the density constant and 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 in larger networks the fraction of the nodes on the perimeter becomes less relevant. As a consequence, a larger network will have a larger average value of the t/m ratio and will filter more packets per hop. Figure 7 shows the average end-to-end delay for the networks under consideration. A larger δ corresponds to higher processing delays introduced at every hop.

To interpret these results, consider a network with 200 nodes and 30 rules per node, thus $||R|| = 6,000$. With such a large rule-set, each hop would introduce a delay larger than 0.45 s, 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 an average delay larger than 3.6 s, which would make 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 M_1 lower than 50% and keep the delay around 0.15 s. That is, we decrease the filtering efficiency to one half, but we reduce the delay by a factor of 24.

Still, if a higher performance of the firewall is needed with a large network size, the delay introduced by the waterwall must be further reduced. In the next section, we introduce an 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 R_i of the rule-set of size $P^f(k,i,j) \times ||R||$ starting from position λ. 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 given that each choice of λ is independent at each hop is high due to the so-called *birthday paradox*. If we are able to use minimum overlapping R_i sets, then we can expect that M_1 and M_2 decrease faster with the distance from the source. The results obtained with disjoint rule-sets are reported in Figures 8 and 9 and represent an upper bound of the gain reachable with this improvement. Comparing Figures 5 and 8, we can see that to obtain similar results, a lower value of δ is sufficient; for instance, to have M_1 below 50%, $\delta = 0.3$ is sufficient (even if $\delta = 0.2$ is below 50% for a network larger than 100 nodes) which corresponds in Figure 7 to a tolerable delay even for a network with 300 nodes.

We can thus try to find a smarter way to choose λ in order to minimize the intersection between different R_i along the path. In this paper, we introduce two proposals to be further evaluated in future works.

The first is to choose λ as a function of specific network parameters of p and node i:

$$x = (\mathrm{IP}_{\mathrm{dest}} \oplus \mathrm{IP}_{\mathrm{src}} \oplus (\mathrm{IP}_{\mathrm{chk}}|\mathrm{IP}_{\mathrm{id}}) \oplus \widehat{\mathrm{IP}_i}) \qquad (6)$$

$$\lambda = x \ (\mathrm{mod}||R||) \qquad (7)$$

where

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

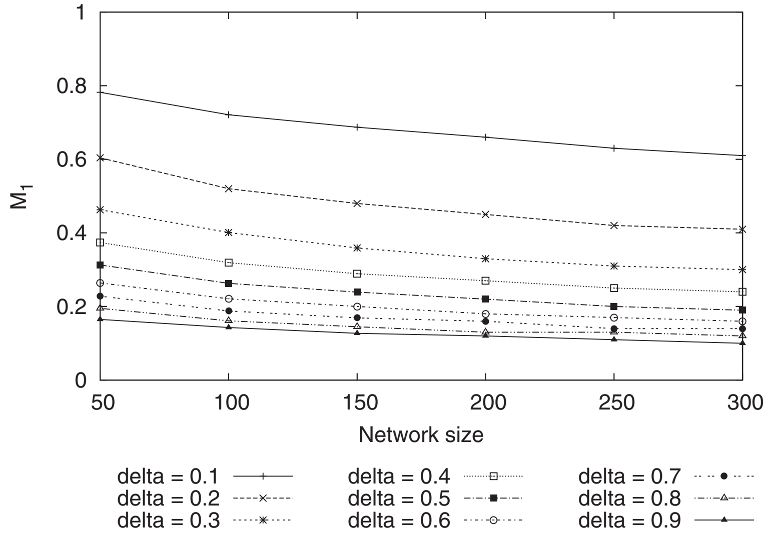

Figure 8 Metric M_1 for increasing network size and δ ranging from 0.1 to 0.9 with ordered rule-set.

- IP_{dest} and IP_{src} are the destination and source IP addresses of p, respectively
- IP_{chk} and IP_{id} are checksum and identification field of the IP header, respectively. Those fields are immutable from source to destination, and their combination is unique for each packet. They are concatenated since their size is just half of the size of an IP address.
- $\widehat{IP_i}$ is the IP address of node i with inverted byte order. Bytes are swapped since we want the host identifier of the IP that has a larger variability to be the most significant byte. Otherwise, the modulo operation may just return the same value for each host.

The rationale of this choice is to produce a λ that changes from hop to hop depending on a unique parameter of node i. In this way, we spread the choices of λ with a deterministic algorithm and try to get a better coverage of R. Nevertheless, we have to avoid that a node i deterministically selects the same λ for all the packets belonging to the same flow (identified by IPs and ports). If this condition does not hold and a node always chooses the same rule-set to filter the packets, then there is a chance that portions of R are never covered.

If this condition does not hold, then an attacker may try to precompute the behavior of the nodes in between the attacker and the destination and choose the route with the

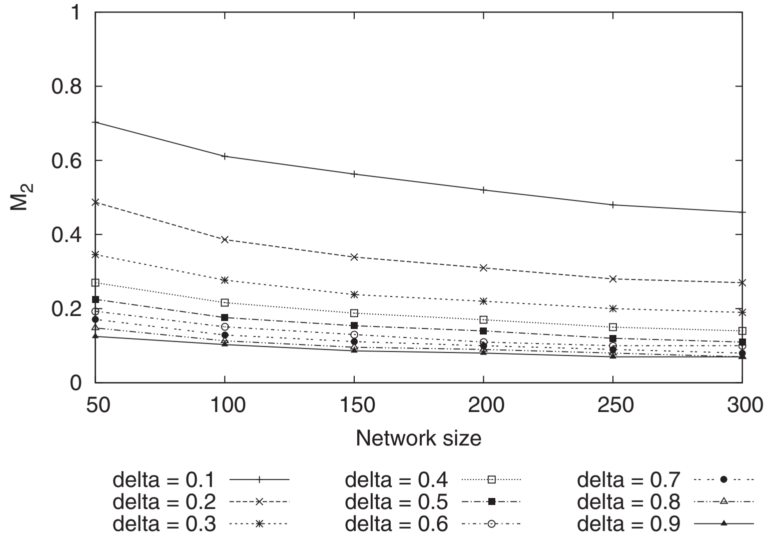

Figure 9 Metric M_2 for increasing network size and δ ranging from 0.1 to 0.9 with ordered rule-set.

highest probability of not being filtered. For this reason, we introduce in Equation 6 the unique identifier $\mathrm{IP}_{\mathrm{id}}$ and the checksum $\mathrm{IP}_{\mathrm{chk}}$ in order to make λ hard to predict along the evolution of the traffic flow.

As an alternative approach, we could use $\frac{\mathrm{sp}_l(i,j)}{m(i)}$ to determine not only the size of R_i, but also its position in R. This way, λ would not depend on some identifier of the node that is performing the filtering (IP_i) but on the estimation of its position in the path from source to destination. This proposal, as the previous one, is an initial design that needs further analysis.

For both these approaches to be applicable, every node must keep the rules in its rule-set in an ordered list. Nevertheless, we do not lose the generality of the approach since the ordering is independent on the semantics of the rule, so it can be applied to rules of any kind. For instance, given the data structure that is used to store the rule in the operating system, an ordering based on a fingerprint on this data structure is sufficient.

Note that in all the results we have shown so far, M_1 and M_2 hardly reach values lower than 0.1. This is due to the fact that $P^f(k,i,j)$ in Equation 1 may not be equal to 1 even if $\delta = 1$ at the first hop for the case $\mathrm{sp}_l(i,j) > m(i)$. To have values of $P^f(k,i,j)$ closer to 1, we can use $\delta > 1$. In this case, Equation 1 must be modified in order to make the value of $P^f()$ bounded by 1, as follows:

$$P^f(k,i,j) \triangleq \begin{cases} \frac{\mathrm{sp}_l(i,j)}{m(i)}\delta & \text{if } \mathrm{sp}_l(i,j) \le m(i) \text{ and } \frac{\mathrm{sp}_l(i,j)}{m(i)} < 1/\delta \\ 1 & \text{if } \mathrm{sp}_l(i,j) \le m(i) \text{ and } \frac{\mathrm{sp}_l(i,j)}{m(i)} > 1/\delta \\ 1 & \text{if } \mathrm{sp}_l(i,j) > m(i) \end{cases} \cdot \tag{8}$$

The rest of the equations do not change. In Figure 10, we report M_1 and M_2 when δ is larger than 1; for the sake of clarity, we report only the smallest scenario (50 nodes). It can be noticed that the metrics follow the same trend observed for values lower than 1 and reach values lower than 0.1.

4 Conclusions

In this paper, we have introduced a new model to perform distributed firewalling in mesh networks that take advantage of the multihop 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 have shown that the waterwall can be used to greatly reduce the unwanted traffic in a mesh network. To quantify the cost of the filtering function, we used the delay measured on an embedded processor by large rule-sets and have shown that our approach scales well up to mesh networks of hundreds of nodes. The source code used to realize the test is available on the website of the main project financing this work (www.pervacy.eu).

As future work, we intend to implement the filtering strategy on a network simulator in order to test and optimize the enhancement described in Section 3.3. Afterwards, we plan to embed this technique in some widely used routing protocol implementation, such as OLSR, in order to test on real networks.

Endnotes

[a] At the time of writing, the Guifi network is made up of about 22,000 nodes and growing at a pace of a hundred

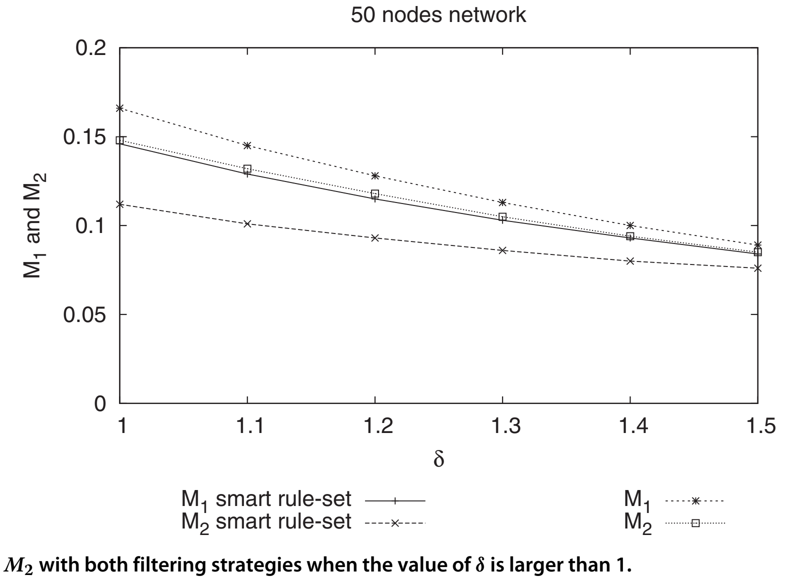

Figure 10 Metric M_1 and M_2 with both filtering strategies when the value of δ is larger than 1.

nodes per week. The network is divided in zones, each one can be formed by hundreds of nodes.

[b] 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.

Competing interests
Both authors declare that they have no competing interests.

Acknowledgements
This work has been 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. Renato Lo Cigno has been partially funded by the European Commission under grant agreement no. FP7-288535 'CONFINE': Open Call 1, *Open Source P2P Streaming for Community Networks –OSPS–*.

Received: 31 January 2013 Accepted: 9 August 2013
Published: 8 September 2013

References

1. IF Akyildiz, X Wang, W Wang, Wireless mesh networks: a survey. Elsevier Comput. Netw. **47**(4), 445–487 (2005)
2. P Frangoudis, G Polyzos, V Kemerlis, Wireless community networks: an alternative approach for nomadic broadband network access. IEEE Commun. Mag. **49**(5), 206–213 (2011)
3. D Vega, L Cerda-Alabern, L Navarro, R Meseguer, in *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. Topology patterns of a community network: Guifi.net (Barcelona, 8–10 Oct 2012)
4. G Min, Y Wu, AY Al-Dubai, Performance modelling and analysis of cognitive mesh networks. IEEE Transactions on Communications. **60**(6), 1474–1478 (2012)
5. S Ioannidis, AD Keromytis, SM Bellovin, JM Smith, in *7th ACM Conference on Computer and Communications Security (CCS)*. Implementing a distributed firewall (Athens, 01–04 Nov 2000)
6. H Zhao, CK Chau, SM Bellovin, in *Workshop on New security paradigms (NSPW)*. ROFL: routing as the firewall layer (Lake Tahoe, 22–25 Sept 2008)
7. H Zhao, SM Bellovin, Source prefix filtering in ROFL. Technical report CUCS-033-09, Columbia University, 2009
8. H Zhao, SM Bellovin, in *IEEE International Conference on Mobile Ad-hoc and Sensor Networks*. High performance firewalls in MANETs (Hangzhou, 20–22 Dec 2010)
9. H Zhao, J Lobo, A Roy, SM Bellovin, in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*. Policy refinement of network services for MANETs (Dublin, 23–27 May 2011)
10. M Alicherry, A Keromytis, A Stavrou, Distributed firewall for MANETs. Technical report. Columbia University (Computer Science Technical Report Series), 2008
11. R Fantacci, L Maccari, P Ayuso, R Gasca, Efficient packet filtering in wireless ad hoc networks. IEEE Commun. Mag. **46**(2), 104–110 (2008)
12. L Maccari, R Fantacci, P Neira, R Gasca, in *IEEE International Conference on Communications (ICC)*. Mesh network firewalling with bloom filters (Glasgow, 24–28 June 2007)
13. P Neira, R Gasca, L Maccari, L Lefevre, in *International Conference on New Technologies, Mobility and Security, 2008 (NTMS)*. Stateful firewalling for wireless mesh networks (Tangier, 5–7 Nov 2008)
14. L Maccari, in *International Conference on Security and Cryptography (SECRYPT)*. A collaborative firewall for wireless ad-hoc social networks (Rome, 24–27 July 2012)
15. M Taghizadeh, A Khakpour, A Liu, S Biswas, in *IEEE International Conference on Computer Communications (INFOCOM)*. Collaborative firewalling in wireless networks (Shanghai, 10–15 Apr 2011)
16. H Zhang, B DeCleene, J Kurose, D Towsley, in *IEEE Military Communications Conference (MILCOM)*. Bootstrapping deny-by-default access control for mobile ad-hoc networks (San Diego, 17–19 Nov 2008)
17. M Alicherry, AD Keromytis, A Stavrou, in *IEEE International Conference on Internet Multimedia Services Architecture and Applications (IMSAA)*. Evaluating a collaborative defense architecture for MANETs (Bangalore, 9–11 Dec 2009)
18. M Esposito, C Mazzariello, F Oliviero, L Peluso, SP Romano, C Sansone, in *Intrusion Detection Systems*, ed. by R di Pietro, LV Mancini. Advances in Information Security, vol. 38 172–210, 2008. Intrusion detection and reaction: an integrated approach to network security (Springer, New York)
19. C Panos, C Xenakis, I Stavrakakis, in *International Conference on Security and Cryptography (SECRYPT)*. A novel intrusion detection system for MANETs (Athens, 26–28 July 2010)
20. L Cerda-Alabern, in *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. On the topology characterization of Guifi.net (Barcelona, 8–10 Oct 2012)
21. T Clausen, P Jacquet (eds.), *3626 - Optimized Link State Routing Protocol (OLSR)* (The Internet Society, Geneva, 2003)
22. L Maccari, R Lo Cigno, in *IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS), Poster Session*. Privacy in the pervasive era: a distributed firewall approach (Courmayeur, 23 Dec 2012)

doi:10.1186/1687-1499-2013-225
Cite this article as: Maccari and Lo Cigno: **Waterwall: a cooperative, distributed firewall for wireless mesh networks.** *EURASIP Journal on Wireless Communications and Networking* 2013 **2013**:225.