Efficient Packet Filtering in Wireless Ad Hoc Networks

Romano Fantacci and Leonardo Maccari, University of Florence
Pablo Neira Ayuso and Rafael M. Gasca, University of Sevilla

ABSTRACT

Wireless ad hoc networks are an emerging technology. These networks are composed of mobile nodes and may adopt different topologies depending on the nature of the environment. Nevertheless, they are vulnerable to network layer attacks that cannot be neutralized easily. In wired networks, firewalls improve the level of security by means of packet filtering techniques that determine what traffic is allowed, thereby reducing the impact of such attacks. In this work, we overview the requirements to adapt firewalls to wireless ad hoc networks and highlight the advantages of the use of filtering techniques based on Bloom filters.

INTRODUCTION

Wireless mobile ad hoc networks are distributed networks composed of terminals connected with wireless communication links. Their dynamic nature makes them appealing for applications in multiple scenarios, from completely distributed ad hoc networks to quasi-static wireless mesh networks for access delivery. In this article, we discuss the possibility of applying firewall techniques to such networks. In particular, our attention is focused on:

- Wireless mesh networks (WMNs) composed of a set of access points (APs) that provide network access to clients that can be mobile. The APs form a wireless backbone to carry the traffic from the clients to one or more gateways in a distributed fashion. The backbone can be considered an ad hoc network, and the whole system can be dynamic. Clients can be mobile so the network must be able to support roaming. Backbone links can be affected by multipath fading, so the backbone must be able to dynamically reconfigure its routes. Even APs can be mobile. For example, imagine a network made up of APs placed on top of ambulances or police cars offering connectivity to pedestrians and medical or police personnel. Generally speaking, client topology is more susceptible to changes than APs are. Usually, the infrastructure also has a centralized authentication server to grant network access to clients, such as an authentication, authorization, and accounting (AAA) server using RADIUS protocol.
- Mobile ad hoc networks (MANET) are self-configuring networks composed of mobile nodes that adopt a completely arbitrary topology that can change rapidly and unpredictably. In a fully distributed set up, the infrastructure lacks any centralized authenticator and uses distributed algorithms to support access control. Note that in a more realistic scenario, this is performed using one or more authentication servers that are always reachable or using pre-loaded keys. Typical applications are rescue operations or tactical networks.

Depending on the adopted standard (e.g., IEEE 802.11 or 802.16), layer 2 security should guarantee that entrance to the network is limited to authorized terminals. Some attacks can be performed at higher layers with the aim of degrading network performance or to exploit services that should not be available. In wired networks, packet filtering solutions, also known as firewalls, reduce the impact of these attacks. Firewalls are network elements that implement any packet filtering technique that determines the allowed traffic. These perimeter security solutions are used to control access to specific services. In wireless ad hoc networks, every node might be offering connectivity to other terminals, so that every node can be seen as a router connecting the internal network with the outside. Because the concept of *perimeter* is not well defined, to be effective, a firewall must be enforced not only on the gateway, but also in the nodes of the ad hoc network. In the two scenarios outlined previously, this raises the following issues that must be resolved:

- Ruleset dimension — Subnetting is not possible in mobile networks. We will see that this implies the explosion of rulesets, with consequent performance problems. Nodes of an ad hoc network typically are embedded devices with limited capabilities. Thus, the solution must use as few resources as possible.

0163-6804/08/$25.00 © 2008 IEEE

- Ruleset distribution between wireless nodes — The number, frequency, and size of control messages that contain ruleset information must be minimized so as not to waste networking bandwidth.

In this article, we provide an overview of the problems related to firewalls that are applied to wireless ad hoc and mesh networks. Moreover, we give some guidelines for implementing ad hoc firewalls, and we evaluate an algorithm for efficient packet filtering based on Bloom filters.

Firewalls: An Overview

A firewall is a network element that controls the delivery of packets across different network segments [1]. In other words, it is a mechanism to enforce policies for traffic shaping and for the availability of services. A firewall access control policy is a list of linearly ordered filtering rules (a ruleset) that define the actions that are performed on packets that satisfy specific conditions. A rule is composed of a set of selectors (or filtering fields), such as source IP address, destination IP address, source and destination ports, protocol type, as well as an action field. Each selector might contain a specific value (e.g., a single IP address) or a wildcard expressing a range of values (e.g., 192.168.0.*); the action field maybe a Boolean value, such as *permit* or *deny* or a more elaborate action. An example rule is the following:

```
if tcp destination port = 80 and
source IP = 150.217.10.8, accept the
packet
```

When a packet arrives at a firewall interface, the header fields are checked linearly against the entire ruleset. If an exact match is found for a rule, the associated action is performed. If the packet is directed to the firewall itself, it is passed to the TCP/IP stack of the operating system; otherwise, if the packet is directed to another system, the firewall forwards the packet through the right route. If the packet does not match a rule, then the packet firewall executes the default policy. For most firewalls, the default policy is to deny a packet that has not matched a rule, so rulesets are expressed in the so called *positive logic*. This approach has several advantages over negative or mixed logic:

- The resulting ruleset has no conflicting rules
- The ruleset is independent by order or rules
- It is easier to design the policy and the resulting ruleset

For the sake of simplicity, in this article we consider only Boolean actions (accept/deny) applied to rulesets expressed in positive logic.

Although the deployment of firewalls is an important step in the course of securing networks, the complexity of designing and maintaining firewall rulesets might limit the effectiveness of firewall security, especially in a distributed scenario. Some of the problems that firewalls must face in current networks are:

- *Ruleset consistency*. When rules are expressed using wildcards, the rules may not be disjoint. In such cases, rule ordering is important and can introduce a consistency problem. Moreover, if on the route from the sender to the network gateway, multiple firewalls are crossed, a consistency problem can be introduced between firewall rulesets. Building a consistent inter-firewall and intra-firewall ruleset is a difficult task and is even more challenging if it has support for frequent dynamic updates (for more details, see [2]).
- *Computational complexity problems*. Because each packet must be checked against a list of rules, the time required for filtering grows linearly with the size of the ruleset. Several algorithms and data structures exist to perform packet matching (see [3]), but their requirements might not fit with our specific field of application.

In an ad hoc scenario, each node of the network may act as a client or as a server and most of all, each has the capability of forwarding traffic to its neighbor nodes. In this article, we focus on the forwarding part of the filtering process; we are not interested in the final destination of the packet, but in the process of forwarding. The problem we want to address is how to perform packet filtering in each node of the ad hoc/mesh network, providing a scalable solution that takes into consideration the problems introduced by the peculiar nature of these networks. In the next section, we will define the application scenario and outline the issues that make implementing firewalls a challenging task due to the specific nature of wireless ad hoc networks.

Although the deployment of firewalls is an important step in the course of securing networks, the complexity of designing and maintaining firewall rulesets might limit the effectiveness of firewall security, especially in a distributed scenario.

Firewalls in Wireless Ad Hoc Networks

Traditionally, a firewall is implemented into the gateway nodes that interconnect a network with the Internet or that divide different areas of the same network. In ad hoc networks, this approach is not suitable. Figure 1 shows a wireless mesh AP network, where an ad hoc network of APs composes a backbone to offer connectivity to wireless clients. In such a network, we take firewalls into consideration for two main reasons:

Network Layer Security: Even if unauthorized nodes are unable to access the network due to MAC layer authentication procedures, an authorized node could perform a denial of service attack to saturate the resources of the backbone. If client A in Fig. 1 starts transmitting a high amount of data to the Internet, the whole path toward the border gateway will be affected. Because bandwidth resources are scarce (especially, if the backbone uses a single IEEE 802.11 channel for every link, which is typical if the APs are equipped with only two network interfaces), this could make the network unusable. Obviously, if a firewall is implemented only at the border gateway, this kind of attack cannot be prevented. Even MAC layer quality of service (QoS) techniques (such as enhanced distributed channel access (EDCA) in the IEEE 802.11e standard), are at the moment targeted at traffic differentiation among distinct traffic classes, but do not take into account the limitation of user resources. Thus, a firewall should be implemented on the APs that route the messages in the backbone.

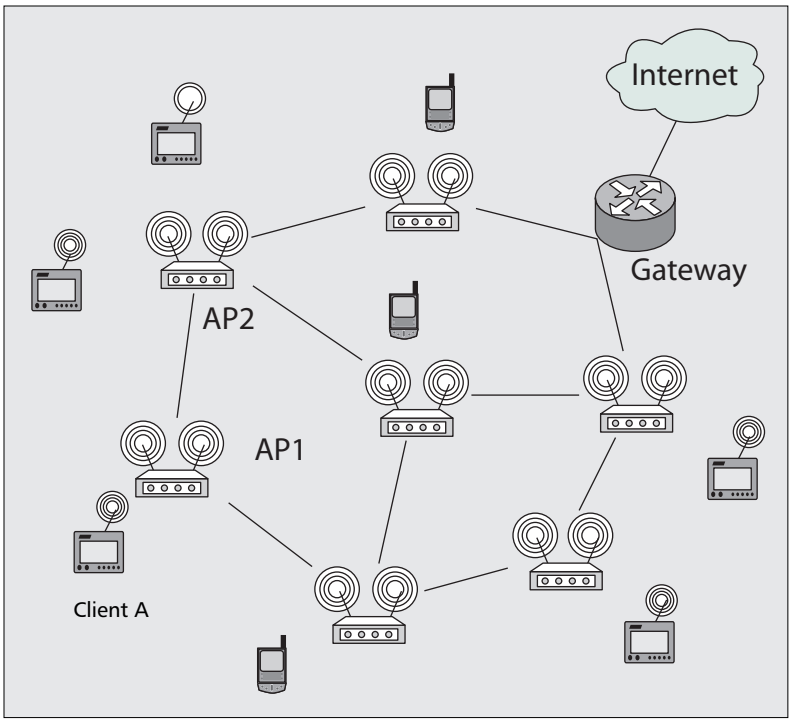

Figure 1. *An example ad hoc/mesh network.*

Traffic Shaping: In access networks with limited bandwidth, each user normally is allowed to access only a subset of the available services — both from nodes belonging to the network and from the Internet. Basically, this means that each client can use only a limited number of TCP/UDP ports when receiving or starting connections. Again, this issue cannot be completely addressed with a firewall on border gateways because it cannot prevent clients from communicating with each other, and it cannot prevent a client from starting multiple connections that will be dropped at the gateway but that also will reduce the network throughput.

In particular, the second issue outlined is of special interest, because the manager of the network must have full control over all the allowed connections for security, stability, or billing reasons.

A suitable approach to reach this goal, normally adopted in wired networking, is subnetting; that is, grouping terminals with similar capabilities in the same subnet (such as a demilitarized zone (DMZ) or an internal corporate network), and configuring the firewalls to use wildcards to filter whole subnets. Each firewall manages a certain number of terminals that are directly connected (or virtually routed) to itself and will drop packets that physically are coming from its network with a foreign IP address to prevent packet spoofing or theft of IP addresses. Subnetting and the use of wildcards in ad hoc networks may not be straightforward for multiple reasons:

- Wireless terminals can be mobile, so there is no possibility of physically grouping clients with similar requirements. Thus, even if the terminals are assigned IP addresses belonging to distinct classes based on their capabilities, APs can receive traffic from roaming clients that preserve their original IPs. This makes it impossible to logically partition the network. With a subnet-based firewall, a node could just change its IP address to a free address of a more valuable class and reach better services.
- If wildcards are used, rules may not be disjoint so that the order of rules in a single ruleset or the order in which firewalls are crossed is relevant. Because the path that a packet will follow to its destination is unpredictable, it is also hard to guarantee the consistency of firewalls with methods as suggested in [2] or previous works from the same authors. Moreover, automatically applied changes in a ruleset that is order dependant can lead to inconsistent rulesets.
- The network may be willing to accept clients coming from external networks, using mobile IPs to preserve original Internet addresses and ongoing connections. These clients do not belong to predetermined subnets; for these hosts, specific rules must be used for each of the allowed connection types.

If wildcards cannot be used, then each allowed connection requires a single rule, and we expect to have the explosion of ruleset dimensions. Different approaches have been proposed in the literature to reduce ruleset dimensions (e.g., [4]). However, due to their complexity, such approaches are not suitable to low-performance hardware with real-time frequent updates. The problem of firewalls in ad hoc networks can be divided into two main issues: finding an efficient and practical algorithm to parse a large ruleset and defining a policy for distribution of rulesets and updates.

Although the problems outlined so far are shared between pure ad hoc networks and so called mesh AP networks, such as the one depicted in Fig. 1, we now take into account that a mesh AP network has an implicit hierarchy. In fact, in a mesh AP, APs are under the control of the network manager, and clients are forced to have their first hop to an AP; they do not contribute to packet routing. Moreover, in a mesh AP, we can assume the presence of an authentication server (e.g., an AAA server, using protocols such as RADIUS) that is contacted at the entrance of every new client (possibly, even at every handover), and we can imagine that the APs might be trusted by each other, so they can exchange information between each other. In ad hoc networks, all this is not granted; the network might be purely distributed without any hierarchy. In our vision, a firewall is the enforcement of a management policy, so the presence of a manager entity is required. For simplicity, we think of it as a human manager able to push rulesets into the nodes and automatically update them whenever necessary; however in a reactive model, the network itself might be able to detect security problems and take countermeasures.

The following are a few more issues in pure ad hoc networks that are relevant to our analysis:

- The nodes are battery-powered embedded devices (mobile phones and cameras, PDAs, or laptops), so the problem of memory and CPU usage is extremely important.
- Because there is no hierarchy, for a firewall to be effective, it must be implemented in

every node of the network because there are no central points (APs) through which traffic is forced to pass.
- The nodes of the network may be mistrusted by each other (any node might be captured and reprogrammed, or it might be acting under the control of an enemy's malicious software), so the exchange of sensitive information, such as keys or filtering policies, between nodes should be discouraged.

In the next section, we will outline some possible solutions for implementing a firewall in a WMN that try to match the requirements of these networks.

Architecture of a Firewall for WMNs and MANETs

To adapt a firewall to WMN/MANET scenarios, two aspects are investigated further in this section: rules distribution and efficient packet classification.

Rules Distribution for WMNs

Rule distribution policies are distinct for the two models of a network under consideration. In a mesh AP network, a firewall is enforced only on the APs, and rules can be moved at the entrance or exit of client nodes. Referring to Fig. 1, in a simple design, each AP starts with an empty ruleset; whenever client A connects to AP1, it performs an authentication with an AAA server, normally wired to the gateway or embedded into it. The authentication is tunneled through the AP and is based on client credentials. After a successful authentication is performed, the AAA server might push the update of the ruleset into *all* the APs with a custom protocol relative to the client. This policy has the advantage of keeping all the APs always synchronized, so a client can freely roam from one to another while all the APs can implement the firewall on the path from source to destination, having more redundancy (we will see in a later section how this is useful). A main disadvantage is that if the client entrance or exit rate is high, the broadcast of the rules might represent a high overhead. A simpler solution would be to delegate the filtering process only to the APs at the first hop from the client: every AP filters only the traffic in output generated by its own clients. If the gateway filters the traffic coming from the Internet, then all the packets crossing the network are controlled. The obvious advantage is that there is no broadcast; a new RADIUS attribute could be used to push the rules into the AP as it normally happens with other parameters. A disadvantage is that if client A moves to AP2, then AP2 must be synchronized. This may happen in two ways: if there is no fast re-authentication protocol in use, then the AAA server will be contacted again, and there is no difference with the first authentication. Conversely, if pre-authentication or proactive key sharing is used [5], then AP2 will receive keying material from AP1 with the inter-access point protocol (IAPP) standard. Since the APs trust each other, the same channel could be used to move the ruleset update.

Since the entrance/exit ratio can be particularly high under certain situations due to movement patterns, such as the building layout or environmental factors, a so-called ping-pong movement phenomenon may result in a high number of requests from the APs to the AAA server or between APs. For that reason, we assume here that each AP caches the rulesets received from the AAA server for a certain interval, as it would do with authentication credentials. Note that even if each AP filters only its own clients, with high mobility and rule caching, the rulesets may be very large at certain moments. Therefore, an efficient rule distribution algorithm must be used together with a suitable algorithm for ruleset parsing.

The interaction with an AAA server is always started by the APs at the beginning of an authentication phase; if required, an asynchronous push algorithm could be enforced using a specific protocol. A possible application is the distribution and upgrade of generic network-wide rules on every AP.

Rules Distribution for Ad Hoc Networks

In contrast, in pure ad hoc networks, every node of the network should be able to do packet filtering together with routing; that said, not all the devices are able to enforce it. Obviously the lower the density of nodes that implement firewalls, the higher the impact of unwanted traffic. One possible option would be to store all the rules in every node, but then force the nodes to filter only a subset of the traffic, that is, the traffic generated by its neighbors. This strategy reminds us of the one described with mesh AP networks, but in pure ad hoc networks it has the following disadvantages:
- It is not always possible to have a list of neighbor nodes; for example, when using ad hoc on-demand distance vector (AODV) protocol without Hello messages, there is no knowledge of all the neighbors.
- There is no proof for the filtering node that the packet it receives is coming from a neighbor or not. A node might use just a spoofed IP or MAC address and time-to-live (TTL) in the IP header and pretend to be a few hops away.
- Upon detection of a new link, a node should reorganize its ruleset to filter out its new neighbor, which might be a difficult task to perform with thousands of rules, depending on the data structure used. With high mobility, this could happen quite frequently.

The problem of ruleset distribution also is less open to generalizing for ad hoc networks, being dedicated-purpose networks. Basically, the same entity that deals with access control should also manage ruleset updates (it might be a central authentication server or a distributed coalition of nodes). In any case, the policy of broadcast update distribution seems to be the most realistic one for limited size networks. Otherwise, in a decentralized model, the firewall implementation also could be enforced at a local level in a limited zone of the network if a group of terminals takes countermeasures upon detection of a certain event. This model resumes the

In pure ad hoc networks, every node of the network should be able to do packet filtering together with routing; that said, not all the devices are able to enforce it. Obviously the lower the density of nodes that implement firewalls, the higher the impact of unwanted traffic.

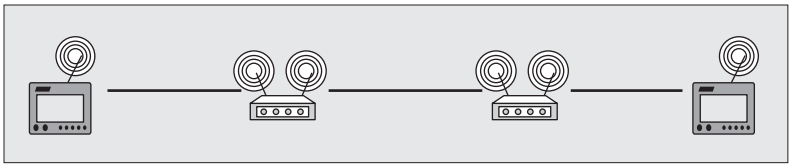

Figure 2. *Testbed composition. APs are integrated Intel XScale 533 Mhz processors with 64 Mbyte memory equipped with 2.6 Linux kernel.*

model of autonomic networks [6], which are beyond the scope of this article.

Packet Classification

In our model we assume that a single rule will be composed of a tuple of the kind $\{Address_{source}, Address_{dest}, Port_{source}, Port_{dest}\}$ where address follows the IPv4 scheme, and port descriptors refer to UDP or TCP protocols. In a network made of N clients, if all the clients are allowed to communicate with each other on a specific destination port, then the number of rules grows as $(N-1)(N-2)*2*p$, where p is the number of allowed ports, and the traffic is bidirectional. If $N = 50$, this yields 4704 rules for a single TCP port.

The most simple and common packet classification approaches check for matches through all entries in the ruleset; these solutions are so-called list-based, exhaustive search. However, such techniques suffer from scalability problems because lookups and memory consumption are $O(n)$. We will show at the end of this section that standard hardware cannot handle linear search over a ruleset composed of thousands of rules.

For this reason, during the last decade, several solutions have been proposed. Initially, the research community focused on algorithmic solutions. Nevertheless, the limitations of such approaches have opened the door to architectural solutions based on specific hardware to speed up packet matching. In [3], Taylor provides the following classification based on distinct designs:

Decision Trees: In this algorithmic approach, a rule is represented as an array of bits to be matched while the ruleset is mapped into a tree. A match is done exploring the tree and finding the match for each bit; the leaves contain the action of the filter. The use of prefixes, such as CIDR/24 for IP addresses, complicates the branching decisions so filters are converted to have single matches. In short, search is not $O(n)$, but the construction is quite complicated and memory consuming; moreover, dynamic incremental updates might require a considerable amount of time, depending on the variant used.

Decomposition: Multi-field lookups are decomposed into single field ones, so that parallel pipelined matching can be done. These solutions provide good look-up performance results when used with hardware supporting parallelization; however, they suffer from memory inefficiencies.

Tuple-Based: Tuples contain the number of unique bits in each rule, that is the bits that distinguish each rule from the others. This technique is based on the observation that the number of distinct tuples is much less than the number of filters in the ruleset. These solutions can be more efficient in terms of memory than linear list-based exhaustive search. However, the look-up performance is ruleset-dependent and performs better in rulesets with many wildcards, which is not the case of ad hoc networks.

Ternary content addressable memory (TCAM): These architectural solutions are based on devices that pipeline matching search, thus lookups are $O(n/p)$, p being the number of pipelines. These solutions are targeted at high performance routers because they require dedicated hardware.

None of these solutions seems directly applicable to software firewalls and ad hoc network models for various reasons. In the next section, we will describe a novel algorithm based on the use of Bloom filters that delivered good performance when applied to real networking devices. To support the issue described so far, first we present some experimental results measured over the network depicted in Fig. 2. All the nodes are equipped with a GNU/Linux operating system, and the two central machines implement firewalls with standard IP tables. Measurements of throughput and round trip delay between the two client terminals are shown in Fig. 3. We see that even under a low load (3 Mb/s unidirectional traffic), a linear search, as performed by IP tables, is a performance killer for both sets of data that are measured. Note that:

- Traffic was generated with UDP packets of size 1500 bytes, in applications such as voice over IP (VoIP); with the same bit rate, we expect to have much smaller frames (e.g., G.711 codec uses 64 kb/s with frame size of 160 bytes). With higher packet generation rates, we also expect higher losses, because processing time is independent of packet size.
- If filtering is done on a longer path, the chain would generate higher delays.

A Promising Approach: Results of a Bloom Filter

A Bloom filter (BF) is a space-efficient data structure for an approximate representation of a set. A query to a BF may produce a false positive with a certain probability, but never a false negative. This means that a query of the type is *element a part of the set S* will never produce a negative answer if $a \in S$, but may produce a positive answer if $a \notin S$.

Basically, a BF is an array of bits of size n. To build the filter, each element of the set is hashed with a number K of distinct hash functions, and the result of each hash $H(x)$ is used as an index to set the corresponding bit in the filter. Each hash function returns a value of size $ln(n)$; the corresponding subset of K bits is turned to 1 in the filter. Each time a new element is inserted in the filter, a new subset of bits is set to 1, with a certain probability of having two distinct elements generate two overlapping subsets. To check the presence of an element in the set, the same procedure is applied; it is hashed with the K hash functions and if all the bits of the subset

were set to1, the element belongs to the set. Because the subsets may be overlapping, there is a certain possibility of false positives, but no false negatives are admitted. For each query, at most, K hash functions should be calculated, so processing time is bounded. Bloom filters were introduced by Burton H. Bloom in 1970,but only in recent years have they received attention in various applications [7, 8].

In a previous article [9], we described the application of Bloom filters for mesh network firewalls. In this article, we summarize the idea, we show new results of its application in a real testbed, and we propose further possible developments to provide more functions.

To set up a ruleset, we populated a filter with elements of the type: {$Address_{source}$, $Address_{dest}$, $Port_{source}$, $Port_{dest}$}, for simplicity using only TCP protocol. The kind of BF is a counting BF that substitutes each bit with a counter made of four bits. A counting BF is a variant of a simple BF that supports deletion of elements, so that our solution is applicable in real world situations where updates are required. The filter size has been optimized to have a false positive rate of 0.1 percent. Since our rules are expressed in positive logic, a false positive means that a packet that should not have been allowed to pass is accepted, whereas there are no allowed packets randomly dropped, which would make the network unusable.

BFs have additional properties that make them appealing for filtering:

- Updates are easy; the only information that is to be transmitted is a mask with the bits to be switched to 0/1.
- The union of two filters is the OR operation of the two arrays, whereas the interception is the AND operation. Then, filters can be easily and completely moved or merged.

If over a multihop path more than one node is filtering, then each node could use a filter using different hash functions. Depending on the distribution strategy used to deliver filters, this can produce overhead, but it greatly lowers the probability of false positives over the whole path because the space of false positives of two different filters can be considered disjoint. As stated previously, having multiple firewalls over the same path increases redundancy and limits the impact of unwanted traffic over the network. In Table 1, we report the results measured on the same testbed using BFs for different configurations. We see that even when using the impractical number of 127 hash functions, the performance of the network is comparable with the performance without filters and has no dependency on the number of rules. In our tests, we stopped at 200,000 rules and 1.5 Mbytes of memory without any sign of instability.

As described in the previous section, the main differences between pure ad hoc networks and mesh networks are mainly due to the way rulesets are distributed and to the number of nodes that apply the firewall. In both scenarios, the rulesets increase to thousands of rules, so that an efficient algorithm must be applied instead of linear search. Consequently, our solution gives a performance improvement in both cases.

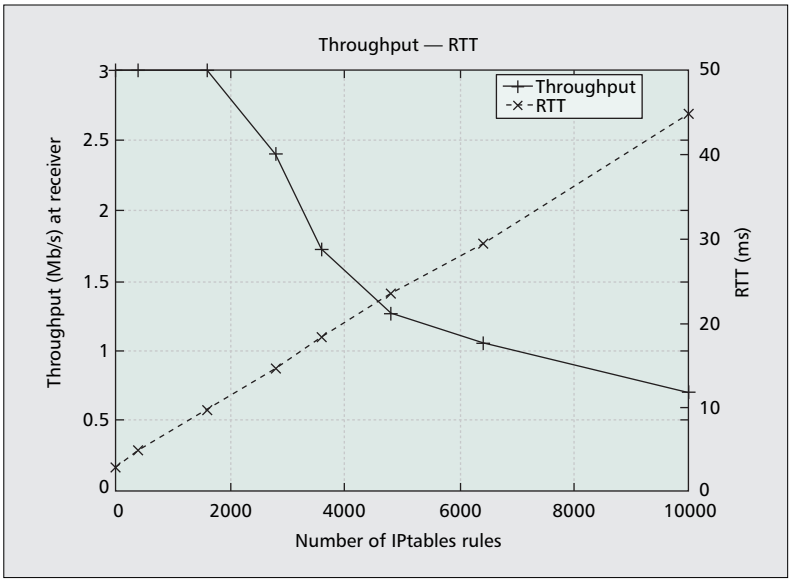

Figure 3. *On left y axis the throughput measured at the receiver for a monodirectional communication, on right y axis the ping round-trip time.*

Number of hashes	Filter size (kbytes)	Rules	Receiver data rate (Mb/s)	RTT
0	—	—	3	2.875
10	75	10,000	3	3.029
10	750	100,000	3	3.953
10	1500	200,000	3	3.043
127	75	10,000	3	4.061
127	1500	200,000	3	3.843

Table 1. *Performance of BF firewalling compared with no firewalling (first line). Data rate of sending terminal is 3 Mb/s.*

Conclusion

In this article, we introduced and analyzed the problem of adapting firewall techniques to reduce the impact of network layer attacks in wireless ad hoc networks, which is of interest for many applications. A promising approach based on Bloom filters also was discussed. Numerical results were provided to highlight the good performance of this approach. Possible research topics for future developments of the Bloom filter approach are:

- Implementation of stateful firewalls. Variants of BFs can be used to represent complex state machines and keep synchronization between two nodes in the ad hoc network.
- Evaluation of the use of more efficient BFs, such as d-left hash-based BFs [10] that can reduce memory occupation.

Acknowledgment

This work is partially supported by the MIUR-FIRB Integrated System for Emergency (InSyEme) project under grant RBIP063BPH.

References

[1] D. Chapman and E. Zwicky, *Building Internet Firewalls*, 2nd ed., O'Reilly & Associates, 2000.

[2] E. Al-Shaer and H. Hamed, "Taxonomy of Conflicts in Network Security Policies," *IEEE Commun. Mag.*, vol. 44, no. 3, 2006, pp. 134–41.

[3] D. E. Taylor, "Survey and Taxonomy of Packet Classification Techniques," *ACM Comp. Surveys*, vol. 37, no. 3, 2005, pp. 238–75.

[4] M. Gouda and X. Liu, "Firewall Design: Consistency, Completeness, and Compactness," *Proc. 24th Int'l Conf. Distrib. Comp. Sys.*, 2004, pp. 320–27.

[5] A. Mishra *et al.*, "Proactive Key Distribution Using Neighbor Graphs," *IEEE Wireless Commun.*, 2004, pp. 26–36.

[6] J. Branch *et al.*, "Autonomic 802.11 Wireless LAN Security Auditing," *IEEE Sec. and Privacy Mag.*, vol. 2, no. 3, 2004, pp. 56–65.

[7] B. H. Bloom, "Space/Time Trade-Offs in Hash Coding with Allowable Errors," *Commun. ACM*, vol. 13, no. 7, 1970, http://citeseer.ist.psu.edu/bloom70spacetime.html, pp. 422–26.

[8] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," 2002, http://citeseer.ist.psu.edu/broder02network.html

[9] L. Maccari *et al.*, "Mesh Network Firewalling with Bloom Filters," *Proc. IEEE ICC*, 2007.

[10] F. Bonomi *et al.*, "An Improved Construction for Counting Bloom Filters," *Proc. Euro. Symp. Algorithms*, 2006.

Biographies

ROMANO FANTACCI (fantacci@lart.det.unifi.it) graduated from the Engineering School of the Université di Firenze, Italy, with a degree in electronics in 1982. He received his Ph.D. degree in telecommunications in 1987. After joining the Dipartimento di Elettronica e Telecomunicazioni as an assistant professor, he was appointed associate professor in 1991 and full professor in 1999. His current research interests are digital communications, computer communications, queuing theory, satellite communication systems, wireless broadband communication networks, and ad hoc and sensor networks. He has been involved in several European Space Agency and INTELSAT advanced research projects. He is the author of numerous articles published in prestigious communication science journals. He has guest edited special issues of IEEE journals and magazines, and served as symposium chair of several IEEE conferences, including VTC, ICC, and GLOBECOM. He received the IEE IERE Benefactor premium in 1990 and the IEEE ComSoc Award for Distinguished Contributions to Satellite Communications in 2002. He is currently serving as an editor for *Telecommunication Systems*, *International Journal of Communications Systems*, *IEEE Transactions on Communications*, and *IEEE Transactions on Wireless Communications*.

RAFAEL M. GASCA (gasca@lsi.us.es) received his Ph.D. in computer science in 1998. He teaches computer security and programming at the University of Seville, Spain. Since 1999 he has been responsible for the Quivir Research Group at the University of Seville. Its research is focused on intelligent and security technology for information systems. He has also authored several articles on different topics about information security issues, such as mobile agent security, firewalls, and security policies.

LEONARDO MACCARI (maccari@lart.det.unifi.it) graduated from the Engineering School of Florence University with a degree in computer engineering in 2004. He joined the Dipartimento di Elettronica e Telecomunicazioni in 2005 as a research fellow. His current research interests are security aspects of wireless telecommunications with a special focus on mesh, sensor, and P2P networks.

PABLO NEIRA AYUSO (pneira@lsi.us.es) is currently working on his Ph.D. in computer science. He is a full-time teacher and researcher at the University of Seville, Spain. He has been working as a consultant for several companies in the IT security industry since 2003 with a focus on free software security solutions. His main research interests are dependable distributed systems and computer security.