Title: 6MoNPlus:Geographically distributed Dual Stack network monitoring

Filippo Lauria, Claudio Porta, Andrea De Vita, Abraham Gebrehiwot, Alessandro Mancini

CNR - Istituto di Informatica e Telematica, Via G. Moruzzi 1, 56124, Pisa - Italy. Division of "Telematic Network of CNR Pisa".

e-mail:[filippo.lauria, claudio.porta, andrea.devita, abraham.gebrehiwot, alessandro.mancini]@iit.cnr.it

Paper type

Technical paper

Abstract

Monitoring and controlling geographically distributed Dual Stack networks on the present Internet architecture is a complex task. The diffused use of Network Address Translation (NAT) and issues caused by border firewalls make remote network monitoring difficult. It is also necessary to physically be connected to the remote networks to sniff packets. There are several situations in which it is convenient to have an easy to use tool, accessible from every location, for monitoring and managing various networks, distributed in different locations, using a single management interface. This article is proposing a geographically distributed, scalable and extensible open tool for monitoring and controlling geographically distributed Dual Stack (IPv4/Ipv6) networks using a single management interface by solving the NAT traversal and firewall issues.

Keywords

Dual Stack (IPv4/IPv6), Geographically distributed Network Monitoring, Private Cloud, Network Address Translation traversal, real-time signaling protocol.

1. Introduction

Monitoring and controlling geographically distributed Dual Stack networks on the present Internet architecture is a complex task. It is necessary to physically be connected to remote networks to sniff packets. The diffused use of Network Address Translation (NAT [1]) and the use of border firewalls limits the possibility to access networks from remote. By proper configuration of NAT servers and firewalls, hosts placed behind NATs or firewalls can be reachable from the rest of the Internet but this has a scalability issue. There are several situations in which it is convenient to have an easy to use plug and play tool for monitoring, controlling and managing various networks, distributed in different locations, by using a single management interface. This article proposes a modular, scalable and extensible open tool for monitoring and controlling geographically distributed Dual Stack (IPv4/Ipv6) networks, which also implement a plug and play NAT and firewall traversal.

The proposed architecture is based on distributed remote components called Probes, used for physically sniffing and monitoring packets from remote networks, and a single central component called CORE that collects the remote monitored packets from the Probes. The network monitoring results are presented using a user-friendly WEB based front-end. The communication between the CORE and the Probes is implemented using two separate channels; a Control Plane and a Data Plane. The Control Plane uses a persistent TCP [2] connection. It is used to manage and control the distributed Probes, to configure the network monitoring modules and for event notification exchanges between the Probes and the CORE. The data plane uses UDP [3] protocol and is used to send the monitored traffic from the Probes to the CORE.

Using 6MoNPlus network administrators are able to:

- Detect, mitigate and notify rogue IPv6 router advertisements

- Monitor the network address utilization by finding the associations between IPv4/IPv6/MAC/DUID/Username in a given range of time interval
- Detect and notify the presence of unofficial DHCP [4] servers
- L2 Loop detection
- Control and manage the distributed remote probes and the installed modules.

In this article we discuss about the distributed architecture of 6MoNPlus and we show how it helps to monitor geographically distributed Dual Stack networks. We also show various measurement results and graphs to demonstrate the scalability and the efficiency of 6MoNPlus by monitoring approximately 4000 Dual Stack active nodes using a single Probe installed on a low cost, credit card-sized single board Raspberry Pi B+ computer.
As a future work, the proposed architecture will also be extended for developing a framework for monitoring and controlling geographically distributed processes and objects. We are convinced that this communication framework could be useful for developing applications in the field of *Internet of Things* and *Smart Cities*.

6MoNPlus is developed by Division of "Telematic Network of CNR Pisa" of *Institute of Informatics and Telematics of CNR*, Pisa - Italy.

2. Comparison of 6MoNPlus with other solutions

2.1 Rogue Router Advertisement mitigation

Rogue Router Advertisement [5] problems may be solved using several approaches: manual address configuration, RA-guard [6] which is a layer 2 filtering of rogue router advertisements on all network devices, SEcure Neighbor Discovery (SEND) [7] and host based packet filtering are some of the possible approaches. Layer 2 filtering might be the best solution, but is still unsupported on most low cost switches. The other methods have serious scalability problems and, in particular, destroy the "plug-and-play" nature of IPv6 [8]. An alternative approach to the above-mentioned solutions is to use an intelligent network-monitoring tool that can automatically detect and mitigate rogue router advertisements. Our previous version 6MoN [9] may also be used to solve the same problems, allowing a network administrator to discover, monitor and mitigate rogue router advertisements automatically. The inspection of router advertisements does not need any configuration and it operates as a plug and play tool while the rogue router advertisement mitigation function needs a minimal and basic configuration, as simple as enabling and disabling a checkbox in a Web form. However, our previous 6MoN version lacks the ability to monitor distributed networks from a single management interface. In addition, the network monitoring code of our previous tool is written using *scapy*[1] python library, while 6MoNPlus is written using more efficient and a lower level C++ programming language. Therefore the previous version requires a powerful hardware with respect to 6MoNPlus to run the tool. There is also a product called NDPMon [10] that allows to monitor Dual Stack networks but the distributed network monitoring plugin is released only as an experimental version and not stable for production use.

2.2 DHCP monitoring

Hosts receiving unofficial IPv4 [11] addresses from Rogue DHCP [4] servers cannot access the network. The problem of rogue DHCP servers may be solved by using DHCP snooping [12], which is a Layer 2 filtering of untrusted DHCP servers. Unfortunately this solution is still unsupported on most low cost L2 switches, therefore it is necessary to have an automatic tool that allow to immediately find the location of the rouge DHCP servers and disconnect it from the network. Using 6MoNPlus is possible to immediately detect and notify the presence of a rogue DHCP server. It functions as follows: each Probe sends a DISCOVER message on every VLAN at a regular interval of time. All DHCP servers, present in each VLAN, respond with an OFFER message. The Probe sends the received OFFER messages including the source MAC addresses and the source IP addresses of all the DHCP servers to the CORE. Each DHCP OFFER message is processed by the CORE which then sends an email notification message to the network administrator notifying the presence of a Rogue DHCP server. The CORE stores the received information in a DBMS for later use. Successive notification

[1] http://www.secdev.org/projects/scapy/

messages will not be sent to the network administrator until the time specified in a configuration variable has not passed. Default time is set to 3600 seconds so, with this configuration, for each rogue DHCP server notifications are sent every hour. For proper operation, the network administrator should selectively flag the OFFER messages stored in the DBMS, relative to the official DHCP servers as legal in order to stop email notifications. This configuration is as simple as enabling and disabling a checkbox in a Web form. From that moment only Rogue DHCP servers will be notified to the administrator. A typical Rogue DHCP notification message, indicating also the Probe ID, looks as follows.

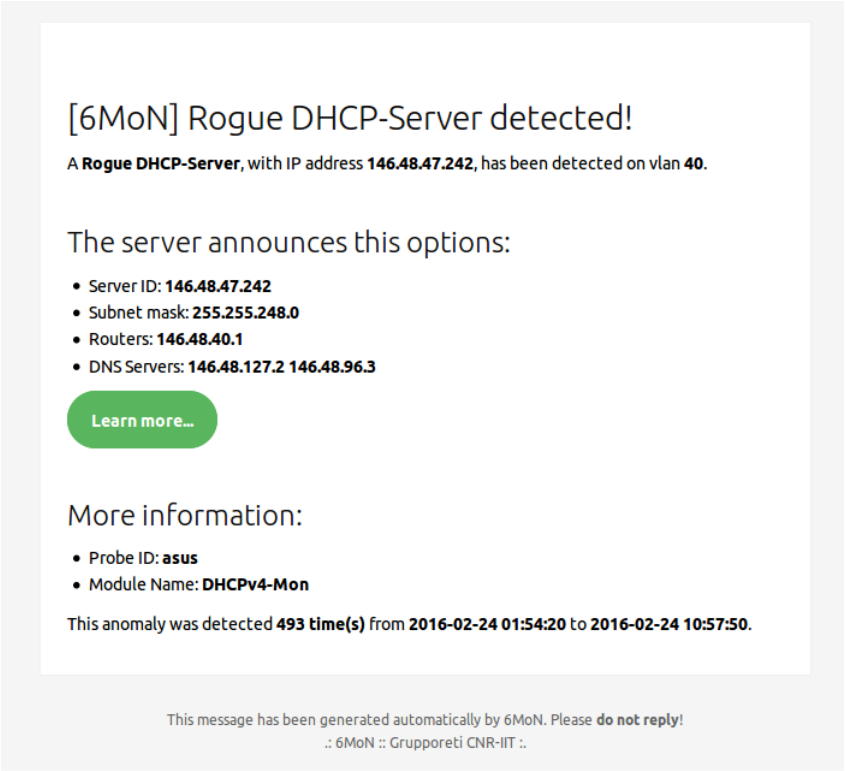


Fig. 1 Rogue DHCP notification message

2.3 NAT and firewall traversal

As we said before, 6MoNPlus uses a distributed architecture and for this reason some Probes might be located behind a NAT server or a firewall. *Network Address Translation* [13] is primarily used to alleviate IPv4 address exhaustion by utilizing the reserved private IPv4 address spaces. It is well known that the presence of NAT breaks the mechanisms by which packets may flow end-to-end from source to destination maintaining the state of the connection on the end systems. The internal network devices, using private IP addresses, can initiate new sessions towards hosts on the external networks by using the NAT server. The NAT device changes the source IP address of the outgoing packet using its own public IP address and relaying replies back to the originating device. On the contrary, without a proper configuration on the NAT server, new sessions cannot be initiated from the external network towards the internal devices. Configuring every NAT server to allow the initiation of new sessions from devices located on external network has serious scalability issue, therefore we propose the following NAT traversal solution.
There is no one single solution for NAT and firewall traversal. Our solution has been to use an autonomous NAT and firewall traversal technique and is based on the following assumptions;

- The CORE should be on a public Internet.
- The Probes are located behind a NAT server or a firewall.
- The Probes should be configured to know the global IPv4 address of the CORE.

The functionalities of our NAT traversal algorithm is as follows; as in the premise, the Probe should be configured to know the public IPv4 address of the CORE. When the Probe process starts, it initiates a new persistent TCP connection and registers to the CORE using the Control Plane. After the registration, the CORE sends the initial configuration message to the Probe. From that moment the Probe sends continuous TCP Keep Alive messages (via TCP) to the CORE, using the control plane, at a regular interval of 30 seconds to maintain the NAT port translation alive on the NAT device. If for some reason the Probe gets disconnected, it will re-register to the CORE using a new TCP session repeating the whole process again. These TCP sessions are used for bidirectional communication and notification of messages between the CORE and the Probes. Typical configuration messages that may be sent from the CORE to the Probes using our NAT traversal are:

- Start sniffing on a Probe
- Stop sniffing on a Probe
- Start a single module (example, Start ARP-Watch, Start NS-Mon, …)
- Stop a single module (example, Stop ARP-Watch, Stop NS-Mon, …)

Configuration messages, specific to a single module, may also be send from the CORE to the Probes and looks as follows:

```
ARP-Watch: {
 module.enabled (true), cache.enabled (true),
 cache.last_seen_timestamp (30), cache.packet_threshold (100)
};
NS-Mon: {
 module.enabled (true), cache.enabled (true),
 cache.last_seen_timestamp (30), cache.packet_threshold (100)
};

Neighbor-Cache-Inspector: {
 module.enabled (false), sending.interval (600)
};

DHCP-Server-Discover: {
 module.enabled (false), probing.interval (60)
}
```

3. Innovative features of 6MoNPlus

All the features supported on our previous version of 6MoN [9] (IPv6-NS, IPv6-RA, IPv4 ARPWatch, DHCPv4 and Radius-Acc) are still supported. Additional and innovative new features have also been added and are described in this section.

3.1 Efficient code and better algorithms

6MoNPlus is an extension of our previous work [9] to which we have added many new features, with better algorithms to minimize both the necessary processing power and the exchanged traffic between the distributed processes. Details of the 6MoNPlus architecture will be shown on the section 4 *(6MoNPlus architecture)***.** The backend application is composed of the CORE and the distributed Probe modules. The tool has been developed using an efficient and low level C++ programing language, in order to be able to run the various processes, particularly the Probes, on low power consumption and low cost computers such as Raspberry Pi B+. The overall libraries used for the development of 6MoNPlus are available for almost every Linux distribution. In this way it is very easy to compile the source code on almost every Linux distribution.
The following figure shows the hardware performance of a Raspberry Pi B+ computer monitoring 34 VLANS and a total of 1074 hosts in the CNR Research Area of Pisa during a range of 24 hours of time.

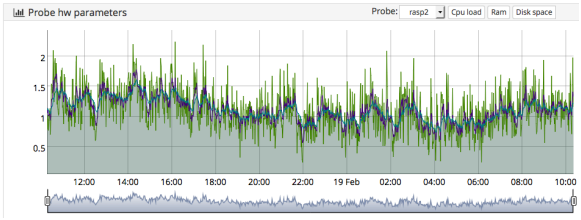


Fig.2 CPU usage of a Raspberry Pi B+ computer

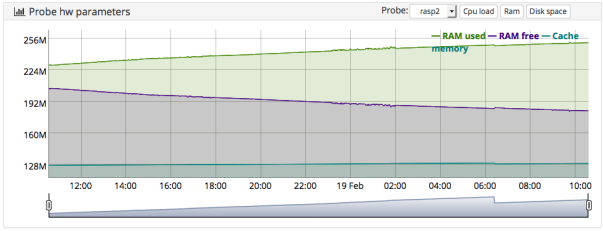


Fig.3 RAM usage of a Raspberry Pi B+ computer

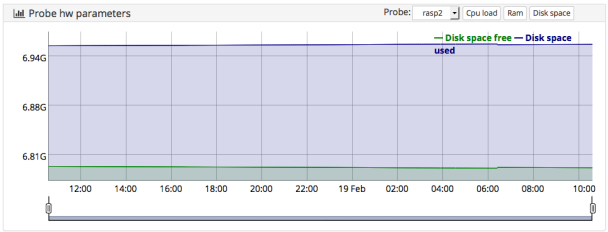


Fig.4 Hard DISK usage of a Raspberry Pi B+ computer

3.2 DHCPv6 DUID based address correlation

Whether a client uses stateless or stateful DHCPv6 [14] auto-configuration or both is controlled by the M and O flags contained in the Router Advertisement (RA) packets [5]. To enable stateful DHCPv6 auto-configuration (e.g., clients get addresses from the DHCPv6 server), the M flag in the RA should be set. To enable stateless DHCPv6 auto-configuration (e.g., clients get configuration data, such as DNS search domains, from the DHCPv6 server), the O flag in the RA should be set. The M and O flags says nothing about whether stateless address auto-configuration (SLAAC) may also be used, which depends only by the flag A contained in the Prefix Information option of the RA. It should be well known that a host might use both SLAAC and DHCPv6 auto-configuration simultaneously having multiple global unicast addresses at the same time. A Router Advertisement packet with M and O bits set captured by Wireshark[2] looks as depicted in the following figure.

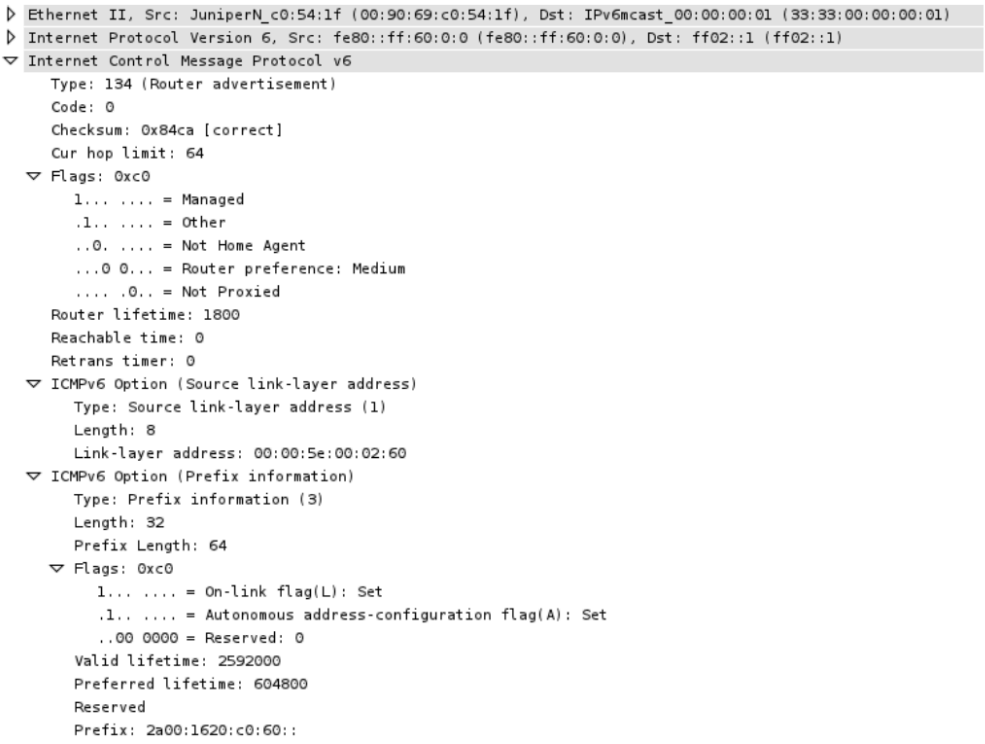


Fig. 5 A router-advertisement message with M and O bits set

When a host having a DHCPv6 client gets connected to a LAN, it sends a *Router Solicitation message* [15]. If the received router-advertisement contains the M and/or O bits set, the host is obliged to send a *DHCPv6 Solicit message*.

A complete DHCPv6 interaction consists of a solicit-advertise followed by a request-reply messages [14].

- The DHCPv6 client sends a *multicast Solicit message* to the following address [ff02::1:2]:547.
- The DHCPv6 server replies with a unicast Advertise message to the client.
- DHCPv6 the client sends a multicast Request message to the following address [ff02::1:2]:547.
- DHCPv6 server finishes with a unicast Reply message containing assigned address and configuration parameters.

A typical DHCP Solicit message looks as follows:

[2] https://www.wireshark.org/

```
▷ Frame 2092: 114 bytes on wire (912 bits), 114 bytes captured (912 bits)
▷ Ethernet II, Src: b8:8d:12:20:f3:6c (b8:8d:12:20:f3:6c), Dst: IPv6mcast_00:01:00:02 (33:33:00:01:00:02)
▷ Internet Protocol Version 6, Src: fe80::ba8d:12ff:fe20:f36c (fe80::ba8d:12ff:fe20:f36c), Dst: ff02::1:2 (ff02::1:2)
▷ User Datagram Protocol, Src Port: dhcpv6-client (546), Dst Port: dhcpv6-server (547)
▽ DHCPv6
    Message type: Solicit (1)
    Transaction ID: 0x625b95
  ▽ Client Identifier: 0001000116280972 3c075434db4d
      Option: Client Identifier (1)
      Length: 14
      Value: 0001000116280972 3c075434db4d
      DUID type: link-layer address plus time (1)
      Hardware type: Ethernet (1)
      Time: Oct 12, 2011 10:16:50 CEST
      Link-layer address: 3c:07:54:34:db:4d
  ▽ Option Request
      Option: Option Request (6)
      Length: 4
      Value: 00170018
      Requested Option code: DNS recursive name server (23)
      Requested Option code: Domain Search List (24)
  ▷ Elapsed time
  ▷ Identity Association for Non-temporary Address
```

Fig. 6 DHCPv6 Solicit multicast message

Each DHCPv6 client and server has a DUID [14]. The DHCPv6 Solicit message is sent to a multicast address ff02::1:2 and contains the DUID of the client host. The DUID is designed to be unique across all DHCPv6 clients and servers, and stable for any specific client or server - that is, the DUID used by a client or server SHOULD NOT change over time if at all possible; for example, a device's DUID should not change as a result of a change in the device's network hardware [14].

For the purpose of our tool it is sufficient to collect the DUID of every host. This is done by listening the *DHCPv6 Solicit multicast messages*, sent by the DHCPv6 client hosts, and associating them with the device's network hardware addresses. For example, if a host having a certain DUID has 2 MAC addresses, wireless and wired, it is possible to correlate each of the MAC addresses to the host's DUID. In this way we are able to tell the list of all the MAC addresses belonging to a single node.

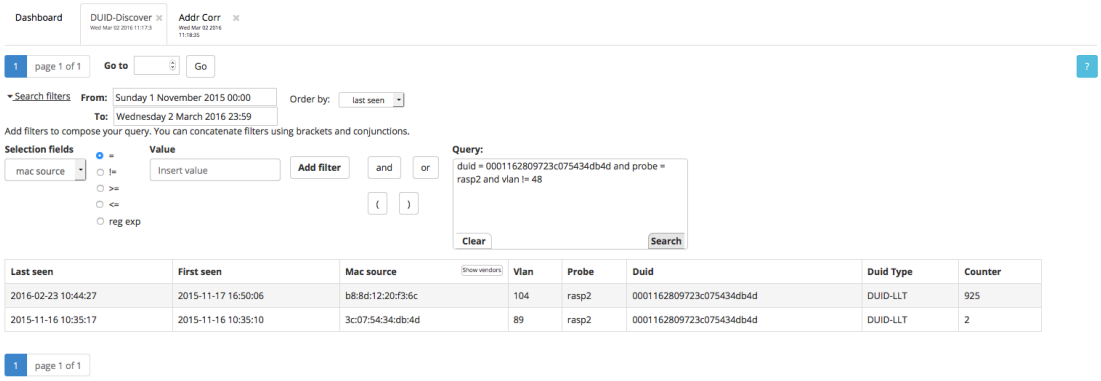


Fig. 7 A host having the DUID "0001162809723c075434db4d" with two MAC addresses wireless "b8:8d:12:20:f3:6c" and wired "3c:07:54:34:db:4d".

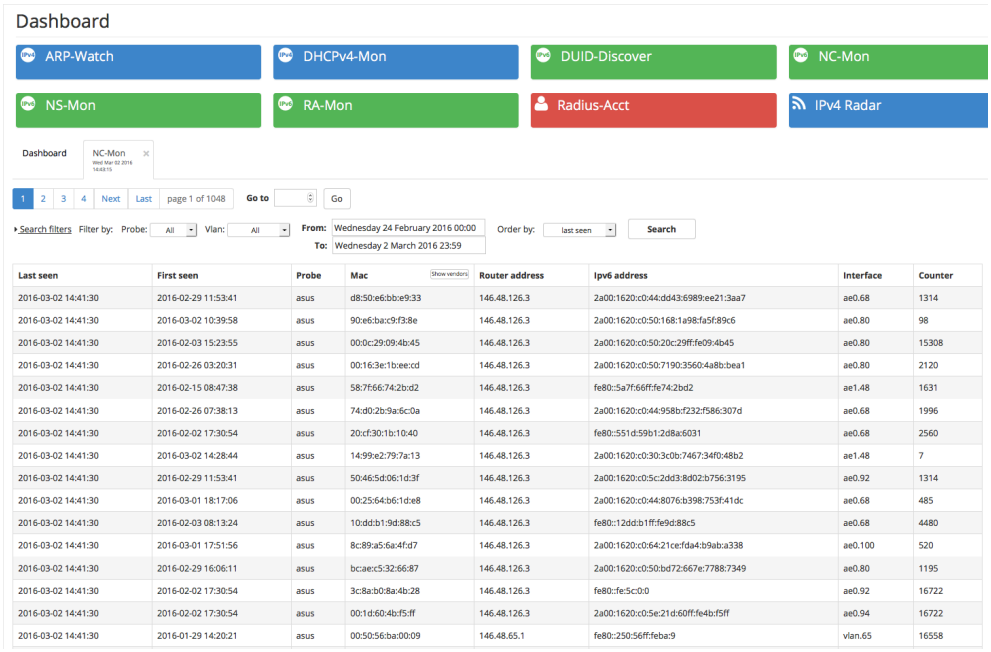
Once we know the MAC addresses of the host, it is easy to find all the correlations between the various IPv4 and IPv6 addresses corresponding to all the interfaces, wireless and wired, belonging to the host. In the front-end section we will describe the details of the DUID correlation function (see section 4.2.3).

3.3 Neighbor Cache

Nodes (hosts and routers) use *Neighbor Discovery Protocol* (NDP [16]) to determine the link-layer addresses for neighbors known to reside on attached links and to quickly purge cached values that become invalid. Hosts also use NDP to find neighboring routers that are willing to forward packets on their behalf. Finally, nodes use the protocol to actively keep track of which IPv6 neighbors are

reachable and which are not, and to detect changed link-layer addresses. When a router or the path to a router fails, a host actively searches for functioning alternates [17].

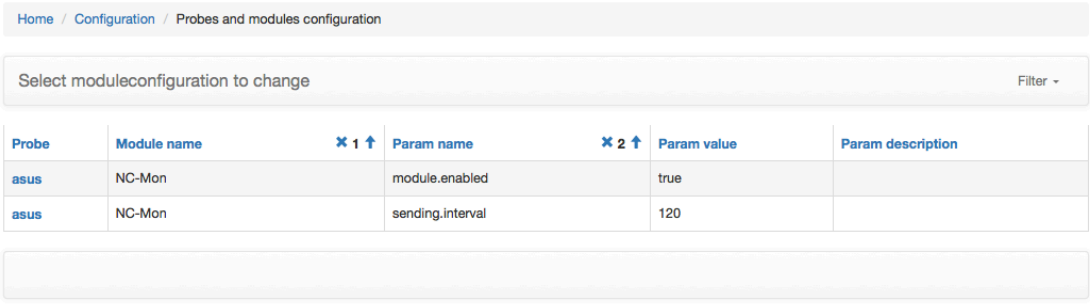
This "*Neighbor Cache*" module of 6MoNPlus enables the Probes to request the Neighbor Cache tables from the IPv6 routers using SNMP protocol [18]. The following figure shows the Neighbor Cache table downloaded from the dual stack router. This SNMP query may also be done using either IPv4 or IPv6 protocol.



Last seen	First seen	Probe	Mac	Router address	Ipv6 address	Interface	Counter
2016-03-02 14:41:30	2016-02-29 11:53:41	asus	d8:50:e6:bb:e9:33	146.48.126.3	2a00:1620:c0:44:dd43:6989:ee21:3aa7	ae0.68	1314
2016-03-02 14:41:30	2016-03-02 10:39:58	asus	90:e6:ba:c9:f3:8e	146.48.126.3	2a00:1620:c0:50:168:1a98:fa5f:89c6	ae0.80	98
2016-03-02 14:41:30	2016-02-03 15:23:55	asus	00:0c:29:09:4b:45	146.48.126.3	2a00:1620:c0:50:20c:29ff:fe09:4b45	ae0.80	15308
2016-03-02 14:41:30	2016-02-26 03:20:31	asus	00:16:3e:1b:ee:cd	146.48.126.3	2a00:1620:c0:50:7190:3560:4a8b:bea1	ae0.80	2120
2016-03-02 14:41:30	2016-02-15 08:47:38	asus	58:7f:66:74:2b:d2	146.48.126.3	fe80::5a7f:66ff:fe74:2bd2	ae1.48	1631
2016-03-02 14:41:30	2016-02-26 07:38:13	asus	74:d0:2b:9a:6c:0a	146.48.126.3	2a00:1620:c0:44:958b:f232:f586:307d	ae0.68	1996
2016-03-02 14:41:30	2016-02-02 17:30:54	asus	20:cf:30:1b:10:40	146.48.126.3	fe80::551d:59b1:2d8a:6031	ae0.68	2560
2016-03-02 14:41:30	2016-03-02 14:28:44	asus	14:99:e2:79:7a:13	146.48.126.3	2a00:1620:c0:30:3c0b:7467:34f0:48b2	ae1.48	7
2016-03-02 14:41:30	2016-02-29 11:53:41	asus	50:46:5d:06:1d:3f	146.48.126.3	2a00:1620:c0:5c:2dd3:8d02:b756:3195	ae0.92	1314
2016-03-02 14:41:30	2016-03-01 18:17:06	asus	00:25:64:b6:1d:e8	146.48.126.3	2a00:1620:c0:44:8076:b398:753f:41dc	ae0.68	485
2016-03-02 14:41:30	2016-02-03 08:13:24	asus	10:dd:b1:9d:88:c5	146.48.126.3	fe80::12dd:b1ff:fe9d:88c5	ae0.68	4480
2016-03-02 14:41:30	2016-03-01 17:51:56	asus	8c:89:a5:6a:4f:d7	146.48.126.3	2a00:1620:c0:64:21ce:fda4:b9ab:a338	ae0.100	520
2016-03-02 14:41:30	2016-02-29 16:06:11	asus	bc:ae:c5:32:66:87	146.48.126.3	2a00:1620:c0:50:bd72:667e:7788:7349	ae0.80	1195
2016-03-02 14:41:30	2016-02-02 17:30:54	asus	3c:8a:b0:8a:4b:28	146.48.126.3	fe80::fe:5c:0:0	ae0.92	16722
2016-03-02 14:41:30	2016-02-02 17:30:54	asus	00:1d:60:4b:f5:ff	146.48.126.3	2a00:1620:c0:5e:21d:60ff:fe4b:f5ff	ae0.94	16722
2016-03-02 14:41:30	2016-01-29 14:20:21	asus	00:50:56:ba:00:09	146.48.65.1	fe80::250:56ff:feba:9	vlan.65	16558

Fig. 8 A Neighbor Cache table

This feature needs minimal configuration like enabling the module on the specific Probe and setting the "sending.interval" variable, the IP address of the router and the SNMP community.



Probe	Module name	Param name	Param value	Param description
asus	NC-Mon	module.enabled	true	
asus	NC-Mon	sending.interval	120	

Fig. 9 A Neighbor Cache module configuration page

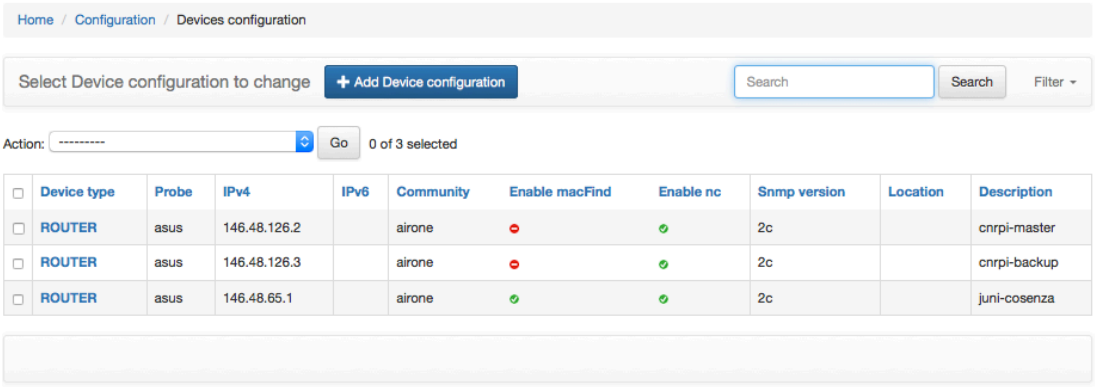


Fig. 10 List of devices to be queried via SNMP

3.4 IPv4 Radar

This application is based on the ARP-Watch module. It allows to visualize the IPv4 address utilization of a defined IP subnet (Network address/netmask) monitored by a Probe. It is necessary to select the date starting from which to start the query, by default it visualizes the IP addresses utilization of the last week. The IP addresses marked with green color are the ones which has been used while the white once specify those addresses not used within the selected range of time. The following figure shows the output of a query.

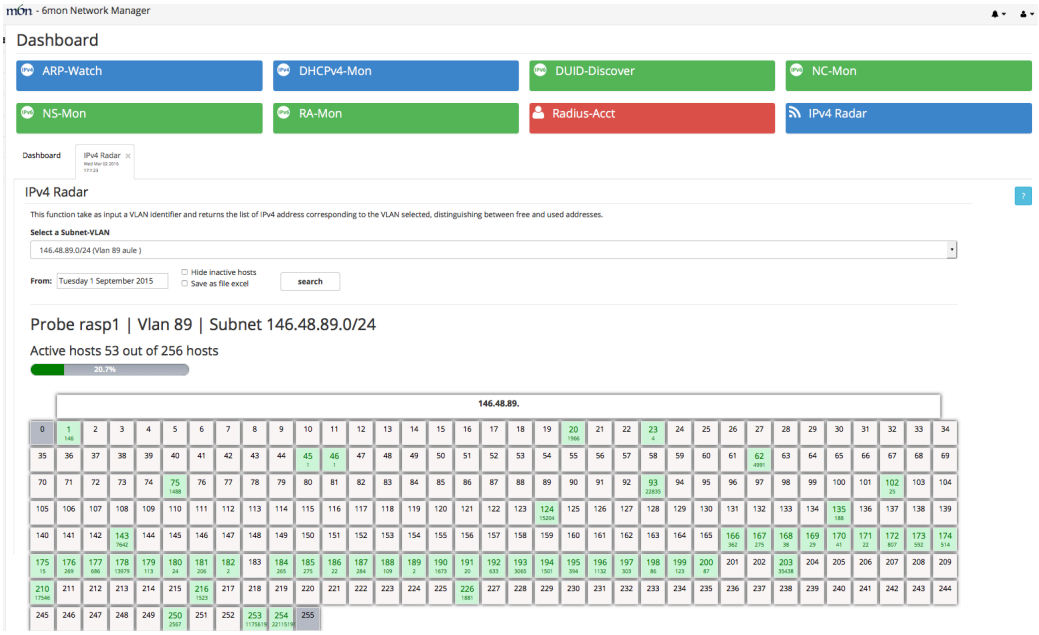
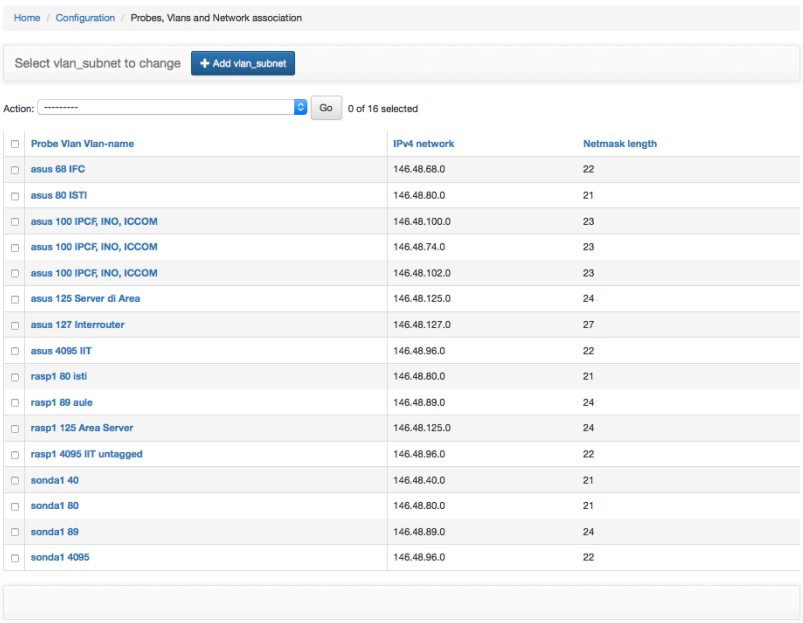


Fig. 11 IPv4 Radar view for a selected subnet

This module needs minimal configuration, for every Probe-VLAN it is necessary to define the corresponding network address and netmask length as shown in the following figure.



Probe Vlan Vlan-name	IPv4 network	Netmask length
asus 68 IFC	146.48.68.0	22
asus 80 ISTI	146.48.80.0	21
asus 100 IPCF, INO, ICCOM	146.48.100.0	23
asus 100 IPCF, INO, ICCOM	146.48.74.0	23
asus 100 IPCF, INO, ICCOM	146.48.102.0	23
asus 125 Server di Area	146.48.125.0	24
asus 127 Interrouter	146.48.127.0	27
asus 4095 IIT	146.48.96.0	22
rasp1 80 isti	146.48.80.0	21
rasp1 89 aule	146.48.89.0	24
rasp1 125 Area Server	146.48.125.0	24
rasp1 4095 IIT untagged	146.48.96.0	22
sonda1 40	146.48.40.0	21
sonda1 80	146.48.80.0	21
sonda1 89	146.48.89.0	24
sonda1 4095	146.48.96.0	22

Fig. 12 IPv4 Radar module configuration page

4. 6MoNPlus architecture

As previously stated, 6MoNPlus uses a modular and distributed architecture and it is based on a *back-end*, a *front-end* and a *DBMS*. The *back-end* is composed of a single central process called CORE and a number of geographically distributed processes called Probes. The *front-end* is a WEB application and it is used to present the elaborated network monitoring data. The *front-end* is also used to configure and manage the overall functionalities of 6MoNPlus. This means that the CORE interacts directly both with the DBMS and the front-end.

The following figure shows a typical installation of the 6MoNPlus tool. The CORE and the geographically distributed Probes are Dual Stack applications and may communicate with each other using either IPv4 or IPv6 protocol.

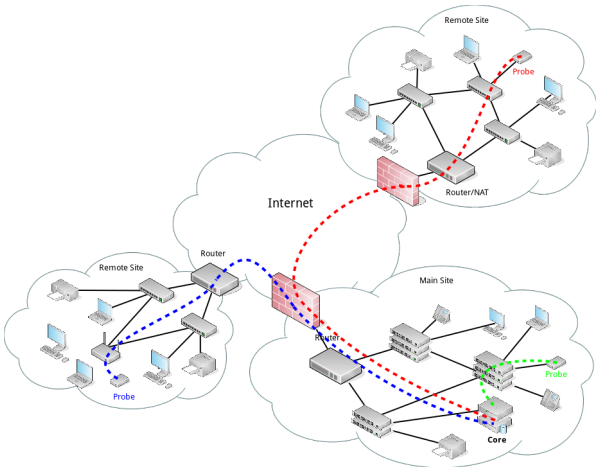


Fig. 13 A typical installation of a 6MoNPlus tool

Being a distributed and modular architecture, the CORE, the DBMS, the WEB server and the Probes can run on separate systems. This modularity of the architecture permits to monitor networks belonging to different administrative domains. In addition the architecture is scalable and is able to grow easily allowing to monitor very large networks. Once the tool has been installed, the addition of a new Probe is as simple as compiling the Probe process and making a minimal configuration. A schematic diagram of the distributed architecture of 6MonPlus is shown on figure 13.

To install the tool in a Private Cloud, accessible from every location, the CORE and the WEB server should be installed using public IP addresses, and may run on Virtual Machines. The Probes can indifferently be located on public Internet, located behind a NAT server or behind a firewall. It is advisable to run the Probes on hardware-based machines in order to be able to physically sniff the network traffic. For a single and small scale network monitoring, the overall system may be installed on a single hardware based machine. As the number of VLANs and the corresponding hosts to be monitored grows, new Probes may easily be added by distributing the number of monitored VLANs on the various Probes. If the above shown graphs indicating the CPU, RAM and HD utilization indicate that a specific Probe is suffering a lack of hardware resource it is important to upgrade the Hardware or to distribute the number of monitored VLANs by adding new Probes.

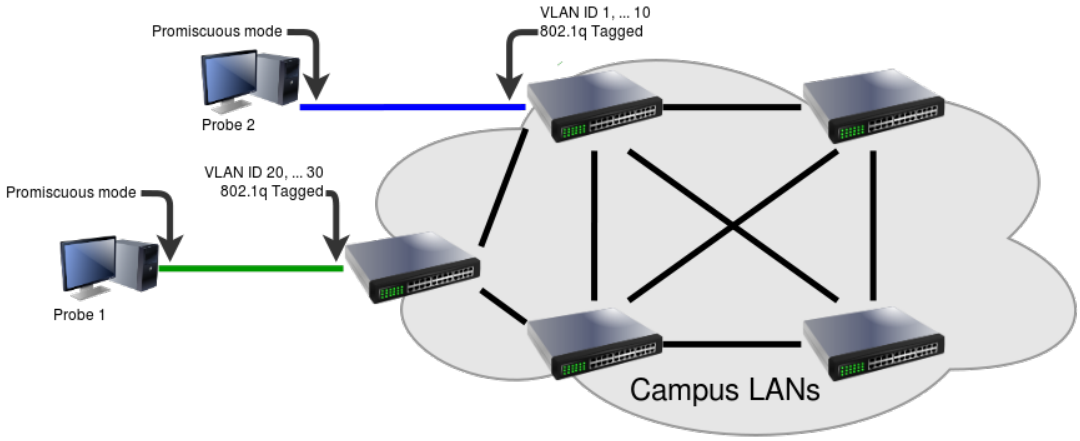


Fig. 14 A typical installation of the Probe tool

The Probes operate mainly in a passive mode by putting the network interface in promiscuous mode and by sniffing broadcast and multicast packets such as *ARP requests* [19]*, ICMPv6 multicast datagrams and DHCPv6 Solicit multicast messages* [14] contained in 802.1q tagged and non-tagged frames, etc. The Probes are also able to actively collect network-monitoring data using unicast communication. For example, it collects IPv6 Neighbor cache from routers via SNMP and RADIUS accounting messages from network devices. The collected network monitoring data is then sent to the CORE using the data plane.

Each Probe is totally controlled and managed by the CORE. The overall 6MoNPlus configuration and visualization of data is done from a user friendly and easy to use Web interface, which is composed of a *Dashboard* and an *Administration page*, which has been greatly improved compared to our previous version of 6MoN.

4.1 The back-end (Core and Probes)

In this section are described the details of the back-end. As stated previously, the communication between the CORE and the Probes uses a *control plane* and a *data plane.*

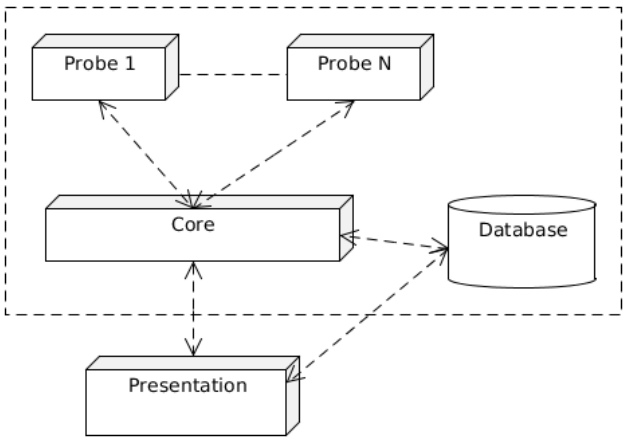


Figure 15: A schematic diagram of the distributed architecture of 6MonPlus

The *control plane* is based on a finite state machine. As described in *section 2.3 (NAT and firewall traversal),* the control plane uses a persistent TCP connection, which is used for bidirectional signaling communication, with a firewall and NAT traversal capabilities. This channel is used for different purposes:

- managing and controlling the overall functionalities of the Probes,
- configuring and managing each single module running on a Probe
- event notification exchanges, for example a rogue DHCP server notification

The CORE is also able to send event notification messages to network administrators via e-mail. This process needs a minimal configuration that can be done using the front-end.
The data plane is based on UDP protocol and is used to send Dual Stack network monitoring data from the Probes to the CORE.
The Probes uses specific modules to collect network-monitoring data. Each monitoring module running on a Probe can be turned on/off separately by sending configuration commands from the front-end. The front-end communicates with the CORE, which in turn sends the configuration commands to the Probes using the control plane.
To avoid hard disk I/O bottlenecks and to improve the processing power performance on the Probes, the collected network monitoring data is directly processed, using efficient and simple algorithms, in RAM memory. Hard disk access is only necessary for debugging and logging the functionalities of the Probe process and can be turned off during normal operation. The captured data is then sent to the Core.

4.1.1 Data information flow on the Probe:

The following figure describes, a simplified version of the data information flow from the Network Interface Card of a Probe until it is sent to the CORE.

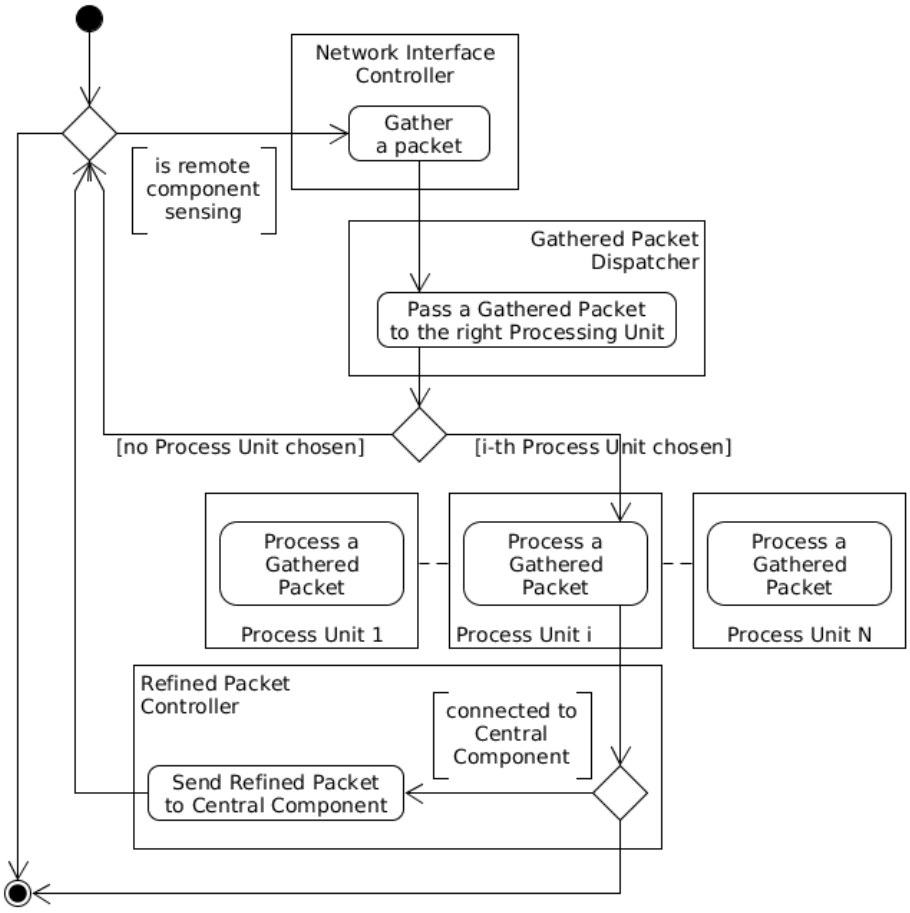


Figure 16: Simplified version of the data information flow from the Network Interface Card of a Probe to the Core

On this diagram, there are various *sub-components* that explain how the information is handled :

- *Gathered Packet Dispatcher* – this *sub-component* is necessary to classify and dispatch to the right *Process Unit* the different types of gathered packets *(ARP, IPv6 RA, DHCP, ...)* coming from the *Network Interface Controller.*
- *Process Unit* – this *sub-component* must be fast, in order to process the *Gathered Packets* (already classified by the previous sub-component) and shrink the content by maintaining only the essential information according to the Processing Unit's internal logic (e.g. as stated above, we can have *IPv6 RA Processing Unit*, *ARP Processing Unit, DHCP Processing Unit,* or *Broadcast Processing Unit*);
- *Refined Packet Controller* – this *sub-component* expects to receive the already processed pieces of information and send them to the *Central Component.*

The modules, which correspond to the process unit, implemented at the time of this publication are:

- RA-Mon: Pv6 Router Advertisements detection and mitigation
- DUID-Discover: DUIDs collected from client machines
- ARP-Watch: IPv4 Watch ARP
- DHCPv4-Mon: Rogue DHCPv4 server detection
- NC-Mon: IPv6 Neighbor cache collected from routers via SNMP [18]
- NS-Mon: IPv6 Duplicate Address detection
- IPv4-Radar: IPv4 Address utilization

In the above figure representing the data information flow, the presence of a *"Threshold based Caching Mechanism"* has been omitted to give a simplified description of the data flow process. This

"Threshold based Caching Mechanism" is used to reduce both the processed information on the Probe and also to drastically reduce the exchanged network monitoring data between the Probe and the Core. It has been possible to reduce the processed information up to 95% of the originally gathered information. It is out of the scope of this article going into the details of this abstraction.

The *Core* receives the redefined packets sent by the Probes and starts processing them. The flow of data information on the *Core* with its *sub-components* is shown in the following diagram*:*

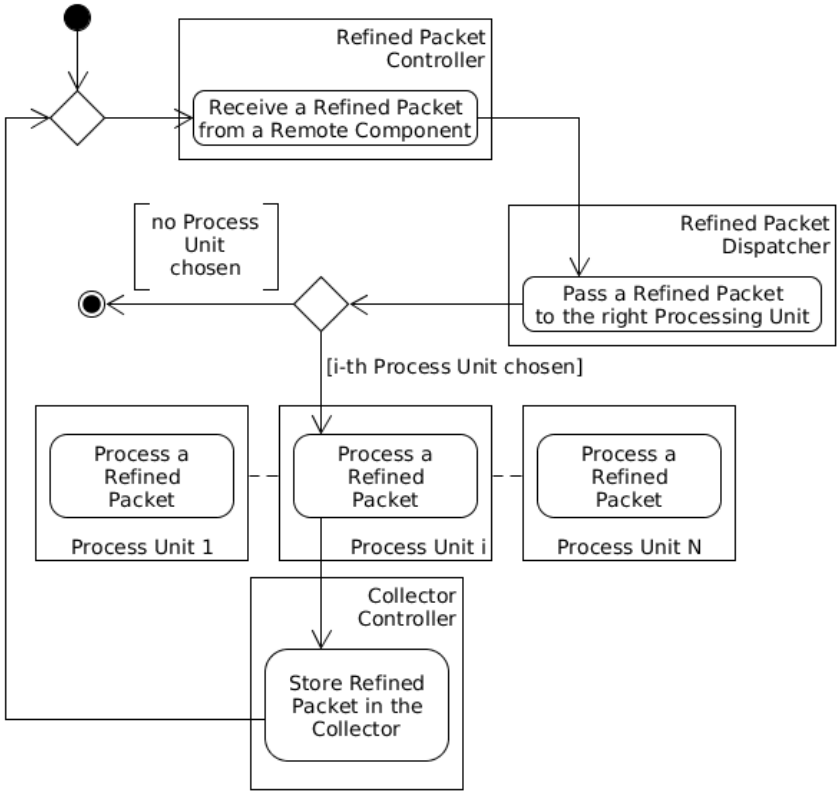


Figure 17: The data information flow from the Network Interface Card of the Core until the storage

- *Refined Packet Controller* – this *sub-component* has to receive all of the *Refined Packet*s incoming from all of the *Probes* connected to *the Core;*
- *Refined Packet Dispatcher* – this *sub-component* is responsible of passing the received *Refined Packet* to the right *Process Unit* (if any);
- *Process Unit* – like its dual on the *Probe's* side, this *sub-component* has to deal with protocol specific *Refined Packet*s and pass it to the Collector Controller*;*
- *Collector Controller* – this last sub-component stores all of the pieces of information that it receives into a database.

4.2 The front-end

It is composed of a *Dashboard* and an *Administration* page. The former is used to present the elaborated network monitoring data and the later is used to configure and manage the overall functionalities of 6MoNPlus.

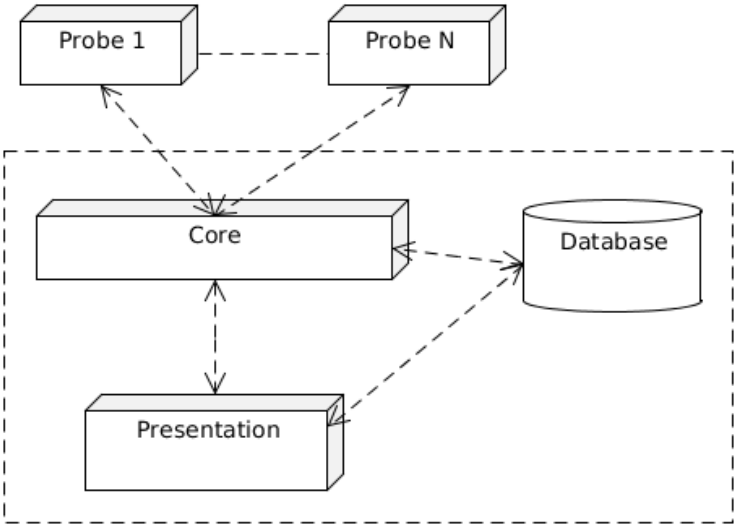


Fig. 18 A schematic diagram of the front-end of 6MonPlus

4.2.1 The Dashboard:

At first login the Dashboard appears as the following figure:

Fig. 19 The Dashboard

The Dashboard provides several graphs allowing to visualize;

- The broadcast and multicast messages monitored in every VLAN by the Probes,
- The hardware performance of each Probe (RAM, CPU and Hard disk usage),
- The current status of each Probe (if it is running, stopped or not registered) ,
- The status of each module on each Probe (if it is running or stopped),

The Dashboard also provides several data presentation features for each of the implemented modules. Within all of the above-mentioned modules, a rich correlation functions and advanced search functions are implemented. These functions allow to easily find the correlation/associations between IPv4/IPv6/MAC/DUID/Username in a given range of time interval. The following figure shows the presentation interface of the ARP-Watch module with the personalized optional filters to perform custom searches. It allows to do custom searches for address utilization and address correlation functions to find the association between MAC/IPv4/IPv6/DUID/Username, DNS name lookup, MAC

vendor lookup and much more. The front-end is developed using both Django[3] and Bootstrap WEB frameworks[4].

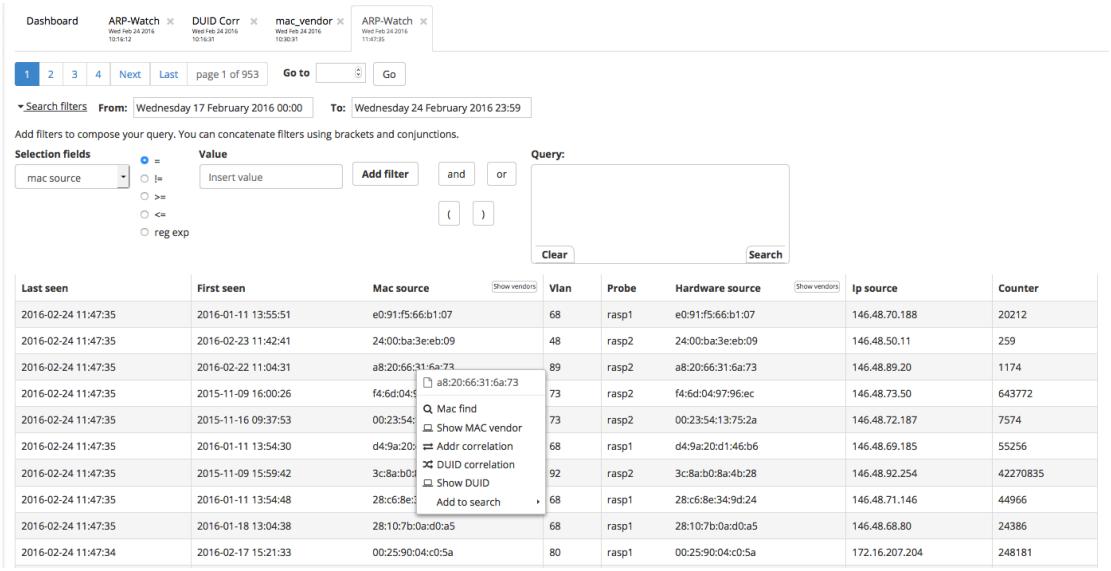


Fig. 20 The advanced search and correlation functions present in each module

4.2.2 The Administration page:

The Administration page allows to configure and manage the overall functionalities of the CORE and the Probes. A single module of a Probe can selectively be turned on/off using the Administration page. As shown previously, for example, it allows to send a configuration message to start/stop a specific module to a Probe. The Administration page also provides the possibility to configure generic parameters that are global to the whole system, such as the behavior of the *"Notifier", used for email based notification of events,* shown in the following figure.

[3] Django. https://www.djangoproject.com
[4] Bootstrap. http://getbootstrap.com/

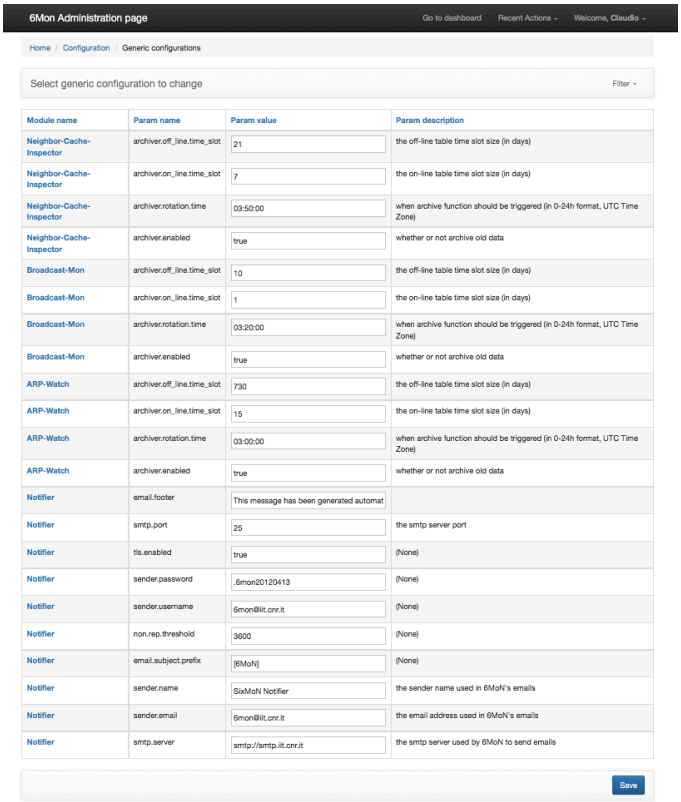


Fig. 21 The Administration page of 6MoNPlus

4.2.3 DUID Correlation

As stated in the section 3.2 (*innovative features of 6MoNPlus*) this section describes the details of the DUID correlation function. It is a very useful tool for a network administrator to identify and correlate the following information in an easy manner:

- All the network interfaces and the corresponding MAC addresses belonging to a specific host
- All IPv6 addresses used by all the network interfaces used in a certain time
- All the IPv4 addresses used by all the network interfaces used in a certain time
- To whom belongs the machine (the username if RADIUS [20] accounting is being used)

The overall process of the DUID correlation function operates as follows:

- Starting from the given MAC the DUID is obtained
- Using the obtained DUID, all the corresponding MAC addresses having that same DUID are obtained
- The data stored in the separate modules (IPv4 Watch ARP / IPv6 NS / Radius-Acc Username / IPv6 NC) are correlated using each of the MAC addresses obtained during the previous steps

The following figure 22 shows the output of a DUID correlation function starting from a single MAC address. As seen in the output, two different MAC addresses corresponding the Wireless and Wired network cards are obtained and the corresponding IPv4/IPv6 addresses are also shown.

Fig. 22 The DUID correlation function output of 6MoNPlus

5. Future work

As a future work, the proposed architecture will also be extended for developing a framework for monitoring and controlling geographically distributed processes and objects for sensor networks. We are convinced that, this communication framework could be useful for developing applications in the field of *Internet of Things* and *Smart Cities*. In addition, an extension for Cloud Network monitoring module will be developed.

6. Conclusions

In this paper we have described the overall architecture of 6MoNPlus introducing the innovative aspects and functionalities compared to the previous version (6MoN). We also provided a comparison between our tool and other similar tools used for monitor networking, showing how it is able to manage the same issues in a simpler and better way.
We also have described in details the distributed architecture by showing how it allows to easily build a scalable network monitoring tool, ideal to monitor very large and distributed dual-stack networks. Some details on the implementation and on the operation of the distributed architecture have also been provided.
Finally we also have described the rich front-end used for presenting the monitoring data and for managing the overall architecture of it.
The developed tool will soon be available and released for free as public domain software.

7. References:

1. Srisuresh, P. and Holdrege, M., 1999. RFC 2663 - IP Network Address Translator (NAT) Terminology and Considerations.
   Retrieved from https://tools.ietf.org/html/rfc2663
2. Information Sciences Institute - University of Southern California, 1981. RFC 793 - Transmission Control Protocol Specification.
   Retrieved from https://www.ietf.org/rfc/rfc793.txt
3. Postel, J., 1980. RFC 768 - User Datagram Protocol.
   Retrieved from https://www.ietf.org/rfc/rfc768.txt
4. Droms, R., 1997. RFC 2131 - Dynamic Host Configuration Protocol.
   Retrieved from https://www.ietf.org/rfc/rfc2131.txt
5. Chown, T. and Venaas, S.,2011. RFC 6104 Rogue IPv6 Router Advertisement Problem Statement. Retrieved from https://tools.ietf.org/html/rfc6104
6. RA-guard http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6_fhsec/configuration/xe-3s/ip6f-xe-3s-book/ip6-ra-guard.html
7. Kempf, J., Arkko, J., Nikander, P. and Zill, B., 2005. RFC 3971 SEcure Neighbor Discovery (SEND). Retrieved from https://tools.ietf.org/html/rfc3971
8. Deering S., Hinden R., 1998. RFC 2460 - Internet Protocol, Version 6 (IPv6) Specification. Retrieved from https://www.ietf.org/rfc/rfc2460.txt
9. *"6MON: Rogue IPV6 router advertisement detection and mitigation and IPV6 address utilization network monitoring tool"* presented on TERENA 2012 Networking Conference
10. NDPMon http://ndpmon.sourceforge.net/
11. Information Sciences Institute - University of Southern California, 1981. RFC 791 - Internet Protocol. Retrieved from https://tools.ietf.org/pdf/rfc791.pdf
12. DHCP snooping http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book/snoodhcp.html
13. Rekhter, Y. Moskowitz, B. Karrenberg, D. de Groot, J. G. and Lear, E., 1996. RFC 1918 - Address Allocation for Private Internets. Retrieved from https://tools.ietf.org/html/rfc1918
14. R. Droms, Ed. Bound, J. Volz, B. Lemon, T. Perkins, C. Carney, M., July 2003. RFC 3315 - Dynamic Host Configuration Protocol for IPv6 (DHCPv6). Retrieved from https://www.ietf.org/rfc/rfc3315.txt
15. Deering, S., 1991. RFC 1256 - ICMP Router Discovery Messages. Retrieved from https://tools.ietf.org/html/rfc1256

16. Narten, T. Nordmark, E. Simpson, W. and Soliman, H., 2007. RFC 4861 - Neighbor Discovery for IP version 6 (IPv6). Retrieved from https://tools.ietf.org/html/rfc4861
17. Narten, T. Nordmark, E. Simpson, W., 1998. RFC 2461 - Neighbor Discovery for IP Version 6 (IPv6). Retrieved from https://www.ietf.org/rfc/rfc2461.txt
18. Case, J. Fedor, M. Schoffstall, M. Davin, J., 1990. RFC 1157 - A Simple Network Management Protocol (SNMP). Retrieved from https://www.ietf.org/rfc/rfc1157.txt
19. Plummer, D., 1982. RFC 826 - Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. Retrieved from https://tools.ietf.org/html/rfc826
20. Rigney, C. Willens, S. Rubens, A. and Simpson, W., 2000. RFC 2865 - Remote Authentication Dial In User Service (RADIUS). Retrieved from https://tools.ietf.org/html/rfc2865

8. Author Biographies:

1. Filippo Lauria is a research fellow at the Institute of Informatics and Telematics of CNR in Pisa, Italy. He currently develops the back-end of 6MonPlus
2. Claudio Porta is a research fellow at the Institute of Informatics and Telematics of CNR in Pisa, Italy. He currently develops the front-end of 6MonPlus
3. Andrea De Vita is technician at the Institute of Informatics and Telematics of CNR in Pisa, Italy. He currently develops network monitoring tools and operates on multimedia cooperation applications.
4. Abraham Gebrehiwot is a technologist at the Institute of Informatics and Telematics of CNR in Pisa, Italy. He is the ***inventor of 6MoN*** and head of the division of "Telematics Network of CNR Pisa" and currently works on network management and developing of IPv6 network monitoring tools.
5. Alessandro Mancini is a technologist at the Institute of Informatics and Telematics of CNR in Pisa, Italy. He is currently involved in developing VoIP services and network management.