Consiglio Nazionale delle Ricerche

A distributed architecture for monitoring and controlling geo-distributed computer networks

F. Lauria, C. Porta, A. De Vita

IIT TR-03/2016

Technical Report

Marzo 2016

Istituto di Informatica e Telematica

A distributed architecture for monitoring and controlling geo-distributed computer networks

Filippo Lauria, Claudio Porta*, Andrea De Vita**

** Istituto di Informatica e Telematica*
Consiglio Nazionale delle Ricerche
via G. Moruzzi, 1 - 56124 Pisa, Italy
email: name.surname@iit.cnr.it

Abstract

Geo-distributed dual-stack (IPv4/IPv6) [1], [2] computer networks are systems which allow computers to share information and software/hardware resources. Monitoring and controlling this kind of systems is not a trivial task for a network administrator of a generic organization, who might want to use a distributed network monitoring software tool to efficiently perform it.

In order to monitor the real-time geo-distributed network traffic, by constantly sensing the carrier with a *network interface card* working in *promiscuous mode,* the tool's architecture has to be distributed too: i.e. a *Remote Component* must be physically connected in each of the subnetworks and *VLANs* [15], forming the organization's network.

In addition, all of the information gathered by the latter has to be processed and sent to a *Central Collector Component.* All of this *Components* have to be properly designed in order to accomplish the overall task in an efficient a reliable way.

This report will describe how to address the latter issue, presenting a distributed architecture for real-time monitoring and controlling geo-distributed dual-stack network, detecting eventual anomalies in it. The proposed architecture has been implemented and it is fully functionally monitoring the campus network of *"The CNR Research Area of Pisa".* The tool's name is *6MoN.* [3]

Keywords: network monitoring, geo-distributed network, dual-stack network, distributed architecture

General Index

1. *Introduction*

Monitoring and controlling geo-distributed dual-stack networks is not a trivial task for network administrators. The main reason for this is that the present Internet architecture is highly heterogeneous, being formed by many interconnected networks, the majority of which includes a large variety of devices (e.g. end users' smart-phones and computers, printers, servers, etc.) that may be, also, interconnected with each other through routers, switches, firewalls, NATs [4], etc. In addition, all of these devices may use various communications protocols to interact with each other, depending on the types of applications they are going to use.

This concept can be extended assuming that, a user wants to open a web-page using a WEB browser. After editing the URL and starting the navigation, a series of events will be triggered:

a) Firstly, a *DNS query* [5] (generally this protocol is used on top of the *UDP protocol* [6], port 53) will be performed to resolve the host-name typed in the browser's address bar and getting back the *IP address* of the web-server (generally an IPv4 address, but sometimes this could be an IPv6 address, too). If the host uses an IPv4 protocol, it needs to make an ARP Request [7] to obtain the *MAC address* of the *DNS server* and this request is done in broadcast. Instead, if the selected protocol is IPv6, the host needs to make a multicast *Neighbor Solicitation message* [7] to obtain the MAC address of the DNS server and this query is sent using the *Solicited Node multicast address* of the DNS server.

b) using the obtained IP address, the host will be able to establish a connection with the web-server, for requesting the desired web-page;

c) Finally, the web-page will be downloaded by the browser typically using HTTP protocol, or sometimes using HTTPS protocol.

This whole chain of events will bring the user on the desired web-site.

Of course this brief introduction wants to stress the concept of how much all the interactions around the Internet are complex. Having an exhaustive view on how the whole Internet works is out of the scope of this report, but it is necessary, for the reader, to understand how difficult monitoring and controlling a network is.

Another aspect to consider, which is relative to the traffic flowing through all of the networks forming the Internet, is the traffic type.

The reader may know that in a generic network, according to one of the protocol aforementioned, both unicast and multicast/broadcast traffic can be found. It is however necessary to note that in a switched network, monitoring broadcast and multicast traffic is much more easier than monitoring unicast traffic. Fortunately many important protocols useful for the operation in a LAN are mainly based on the use of broadcast, in the case of IPv4 and multicast traffic in the case of IPv6. Since the nature of this traffic categories is different, a monitoring tool might adapt its behavior while processing one, another or even both of them. According to the purpose of the monitoring tool, a *punctual approach* (i.e. tracking each single piece of gathered information) or a *statistical approach* (i.e. using statistics to synthesize the whole gathered information and maintaining only the essential part of it with no loss) might be followed.

This report will be focused on a generic tool architecture, for monitoring and controlling a geo-distributed dual-stack network that leverages on statistics to synthesize the gathered information. In the next section a geo-distributed dual-stack network architecture will be introduced to go deeper and understand how this kind of system may be monitored and controlled.

2. Model of a geo-distributed and dual-stack network architecture

Consider an organization with multiple branches geographically distributed in different places of a territory. Assume that all of the subnetworks (marked with A, B and C and represented in Figure 1), are forming a unique geo-distributed dual-stack network, which allows devices (of the organization) to exchange information, share software and hardware resources.

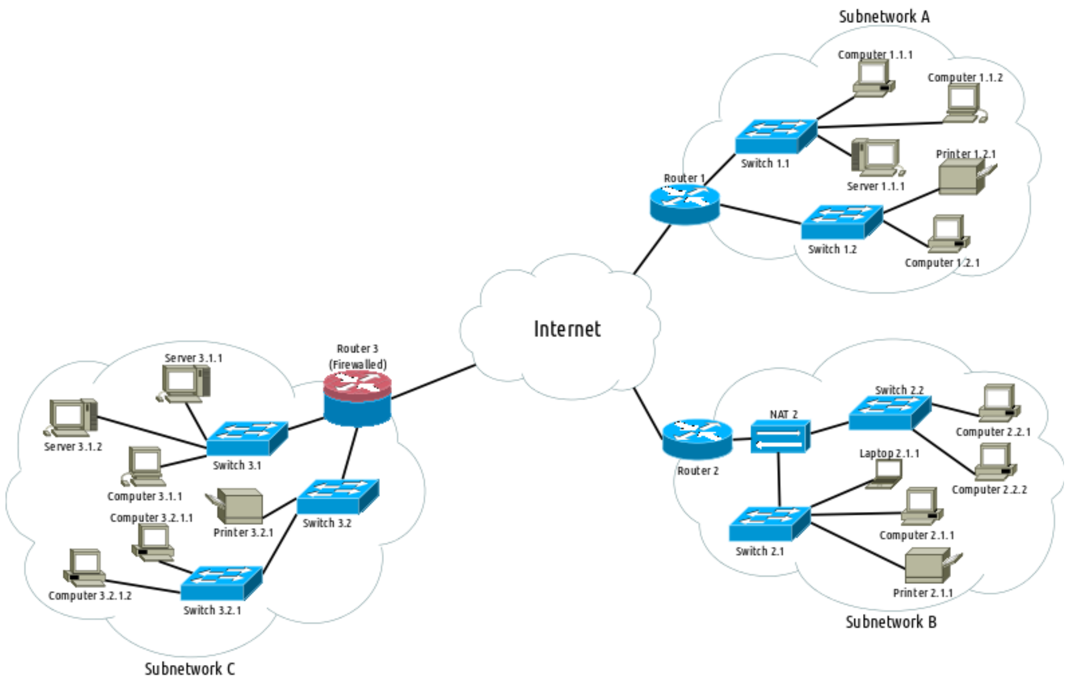


Figure 1: a model for a geo-distributed dual-stack network of a generic organization

Each of those subnetworks is different from the others:

- *Subnetwork A* – in this subnetwork many different components can be identified: several end users' computers, a server, a printer, two switches and a router to allow accesses to the Internet. It is also possible that inside this network both IPv4 and IPv6 protocol stacks are used, in other word this is a dual-stack subnetwork;

- *Subnetwork B* – in this subnetwork there are no servers, but the substantial difference, from the previous one, is the presence of a NAT. Only IPv4 stack is used (single-stack subnetwork) using private address space [12];
- *Subnetwork C* – this last subnetwork contains several servers and end users' computers, a printer, few switches and a particular device that acts, at the same time, both as a firewall and as a router. Of course both the stacks (IPv4 and IPv6) are supposed to be used inside this subnetwork, but the latter fact suggests an important thing: several UDP/TCP ports might be filtered or blocked by the firewall for someone who is accessing the network from the Internet.

This architecture represents a simplified real world scenario of an organization's network. In the next section a distributed architecture for monitoring and controlling geo-distributed networks will be introduced.

3. The proposed generic distributed network monitoring architecture

For what has been said so far, it is mandatory to use a distributed architecture formed by three distinguished types of components:

- *Various Remote Components* – each of those peripheral components has to reside in one of the subnetworks to be monitored. In fact, in order to detect the subnetwork traffic, a generic *Remote Component* must be physically connected to the subnetwork to be monitored. More in details, the *Network Interface Controller* of a generic *Remote Component* has to sense the carrier and should be set in *promiscuous mode,* in order to be able to gather all of the network traffic flowing through the carrier;
- *A Central Component* – this component is central to the whole distributed architecture. It is used to save all the information retrieved by the *Remote Component*s, it is also used to configure and manage the *Remote Components.* From a high-level perspective this component can be split into *two main components*:
  - *Central Processor Component* – that will interact with all of the *Remote Component*s connected to it;
  - *Central Collector Component* – this component is used by the

Processor Component to store the monitored data incoming from the *Remote Component*s.

- A *Presentation Component* – it is the component with which users interact. It elaborates the data stored in the *Central Collector Component* with the aim of retrieving other additional information, by correlating the gathered data of subnetwork traffic, and provides a graphical interface in order to present the elaborated data. It also provides a control panel used to configure the various *Remote Component*s. It sends eventual configuration parameters to the *Central Component* which will take care of properly configuring the *Remote Components*.

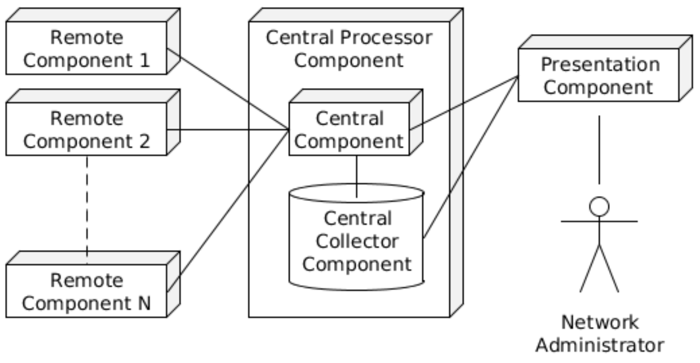


Diagram 1: proposed macro-architecture for a distributed monitoring tool

A block diagram of the proposed architecture is shown in Diagram 1.

4. Communications among the distributed components of the proposed architecture

There are significant differences between types of information exchanged among components. In particular, in this proposed architecture, two different *communication planes* have been defined:

- *Data Plane* – this plane is used for sending network monitoring information, gathered by the *Remote Components,* to the *Central Component.* It could be considered as a *"unidirectional channel"*, since within it the information has to flow only in one direction. In an architecture that follows a statistical approach of network monitoring (as explained in the Introduction) this *"channel"* could be connection-less since loosing a piece of information is complained by the chosen approach;
- *Control Plane* – this plane is used for sending *bidirectional control information* (i.e. the status of the *Remote Component*s) or commands from the *Central Component* to a generic *Remote Component* (i.e. start/stop signal, load a specific configuration, etc.) and messaging information (i.e. notification of an occurred event to a network administrator, for example the detection of a *Rogue IPv6 Router Advertisement).* It must be bidirectional since the information flows both ways from *Central Component* to *Remote Component*s and vice-versa.

At least three types of information or packet formats can be found in this particular architecture:

- Remotely *Gathered Packet* – by constantly sensing the carrier, *Network Interface Controller* gathers a lot of network traffic, that is a lot of *Packet Data Unit*s (*PDU*s). For the purpose of this report a *Gathered Packet* is a *generic PDU;*
- *Refined Packet* – since a single remotely *Gathered Packet*, generally contains more than what is really needed to extract the essential statistical network monitoring information from it. This particular information format has to be considered as a shrunk version of the remotely *Gathered Packet* format;

- *Control Message* – this packet format slightly differs from the previous two. It must be used for infrastructural information exchanges and/or event notification between the geo-distributed *components.*

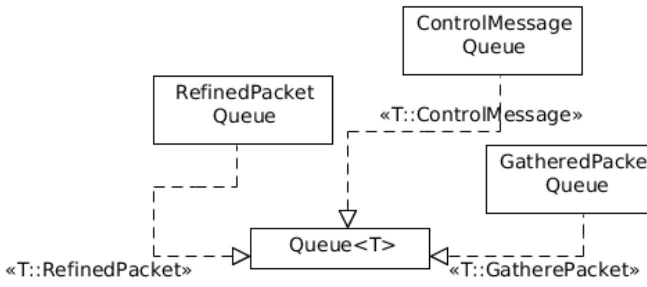


Diagram 2: class diagram for representing all of types of information used in the communications of the architecture

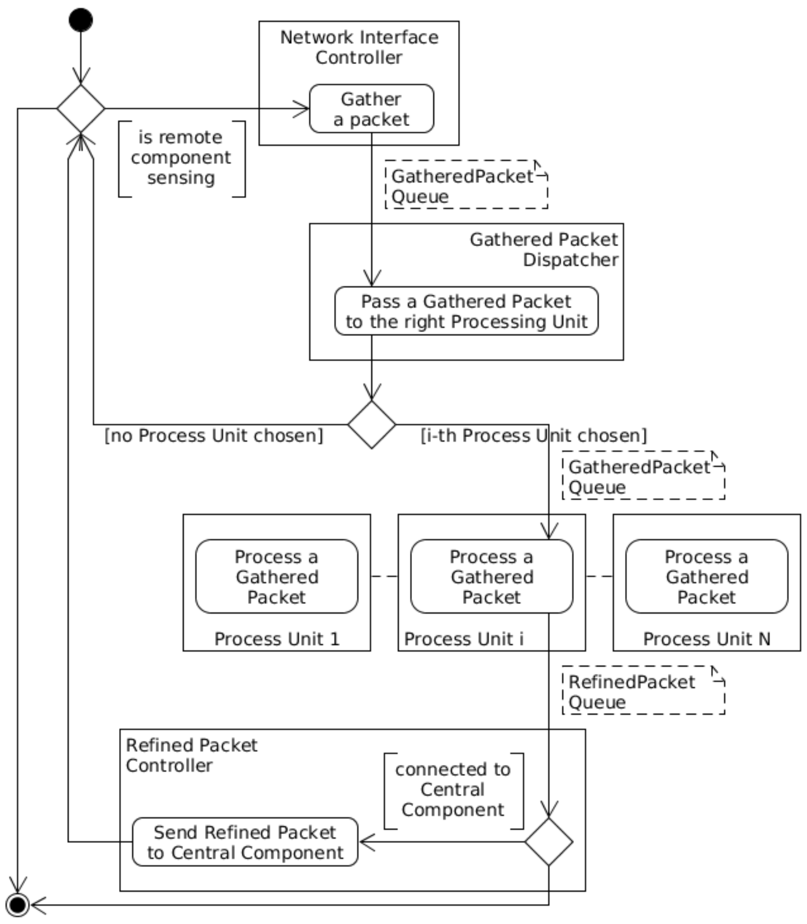
A simple class diagram is shown in Diagram 2 where it is also represented a (Template) *Queue* for a better organization of the information inside the single component. Depending on the type of information to be processed, the generic template class *Queue* is specialized for processing the specific data type (*Gathered Packet, Refined Packet or ControlMessage*). How and when this data structure is used will be implicitly described in the next section.

5. Data information flow

This section presents several activity diagrams to explain in details how data information *(Gathered* and *Refined Packets)* flows from a *Network Interface Controller* of a generic *Remote Controller* to the *Central Collector Component.*

The data information flow described in this chapter has been used for the development of the geo-distributed architecture of *6MoN* back-end (*Central Component* and *Remote Components).* For this reason its detailed description is considered as the main focus of this technical report.

Diagram 3 represents the whole set of activities done by the *Remote Components* to gather a packet from the carrier, then prepare it and finally send it to the *Central Component*.



Network Interface Controller
Gather a packet
is remote component sensing
GatheredPacket Queue
Gathered Packet Dispatcher
Pass a Gathered Packet to the right Processing Unit
[no Process Unit chosen]
[i-th Process Unit chosen]
GatheredPacket Queue
Process a Gathered Packet
Process Unit 1
Process a Gathered Packet
Process Unit i
Process a Gathered Packet
Process Unit N
RefinedPacket Queue
Refined Packet Controller
connected to Central Component
Send Refined Packet to Central Component

Diagram 3: activity diagram describing the information flow on the Remote Component

In Diagram 3, 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 traffic coming from the *Network Interface Controller;*
- *Process Unit* – this sub-component must be fast, in order to process the *Gathered Packet* (already classified by the previous sub-component) and shrink it according to the *Processing Unit's* internal logic (e.g. there would be *IPv6 RA* Processing Unit [13], *ARP* Processing Unit, *DHCP* Processing Unit [14], or *Broadcast* Processing Unit, etc.);
- *Refined Packet Controller* – this sub-component expects to receive already processed pieces of information and send them to the *Central Component.*

The *Central Component* receives *Refined Packets* and starts processing them. The flow of data information on the *Central Component* with its sub-components is shown in Diagram 4*:*

- *Refined Packet Controller* – this sub-component has to receive *Refined Packet*s incoming from all of the *Remote Component*s connected to *Central Component;*
- *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 *Remote Component* side, this *sub-component* has to deal with protocol specific *Refined Packet*s and let them proceeding to the *Collector Controller;*
- *Collector Controller* – this last sub-component stores all of the pieces of information that it receives into a database.

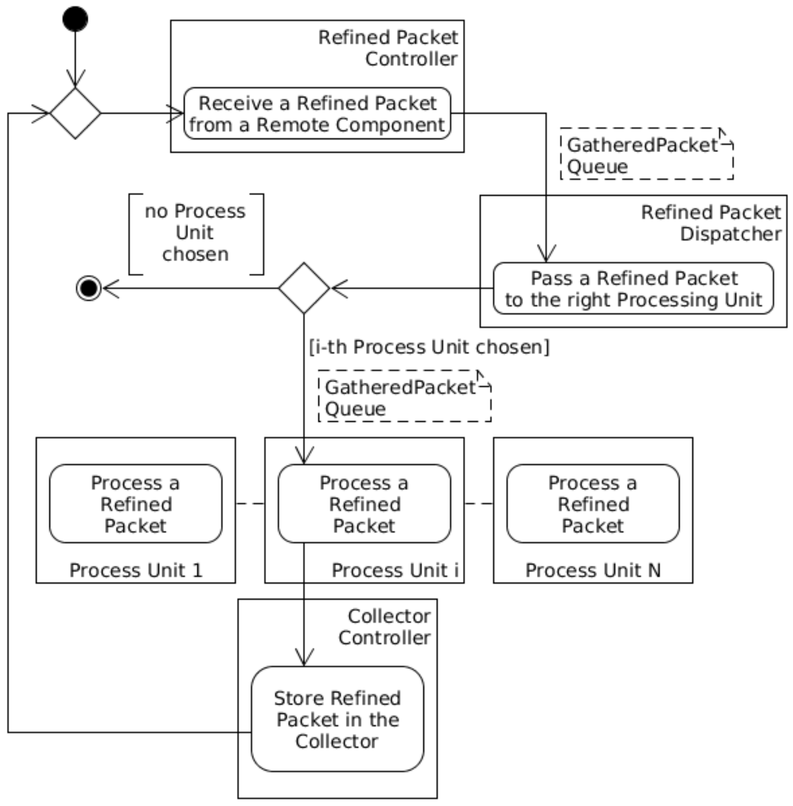

Refined Packet Controller
Receive a Refined Packet from a Remote Component
GatheredPacket Queue
Refined Packet Dispatcher
no Process Unit chosen
Pass a Refined Packet to the right Processing Unit
[i-th Process Unit chosen]
GatheredPacket Queue
Process a Refined Packet
Process Unit 1
Process a Refined Packet
Process Unit i
Process a Refined Packet
Process Unit N
Collector Controller
Store Refined Packet in the Collector

Diagram 4: activity diagram that shows the process of storing a Refined Packet received from the Central Component

6. Interaction of the Presentation Component with the rest of the architecture

The *Presentation Component* communicates with the *Central Component* as described in Diagram 1.

There are two different types of interaction with the *Central Component*:

1. *Presentation* – of the data retrieved from the *Remote Component* and stored in the *Central Collector Component;*
2. *Configuration* – of the whole system which may be the configuration of a *Remote Component*, the starting/stopping of a module, or of a *Remote Component* by directly interacting with the *Central Processor*

Component. All of the above interaction is provided with a common web browser.

7. Implementation and test

An implementation of the discussed architecture has been developed. It has been called *6MoN* and it is now monitoring and controlling the entire campus network of *the CNR Research Area of Pisa,* which consists of 41 subnets/VLANs and approximately 4000 hosts/day.

All of the components described on this report have been implemented and are currently running. The technologies used for developing each of the *components* are the following:

- *Remote Component*s and *Central Processor Component* – those components have been developed using *C++ programming language* and in particular using *STL*[1], *Boost*[2] *and OpenSSL*[3] *libraries.* Few *Remote Component*s have been installed and tested on *Raspberry Pi, model B+*[4], monitoring efficiently 34 VLANs containing approximately 1070 hosts;
- *Central Collector Component* – this component uses a *MySQL*[5] *database* to provide data persistence;

1 Standard Template Library C++. https://it.wikipedia.org/wiki/Standard_Template_Library

2 Boost C++. http://www.boost.org/

3 OpenSSL. https://www.openssl.org/

4 Raspberry pi Model B+ - Hardware specification. https://www.raspberrypi.org/products/model-b-plus/

5 MySQL database. https://www.mysql.it/

- *Presentation Component* – this component has been built using *HTML, CSS, and JavaScript* for the development of the *GUI* and *Python* for the interaction with the back-end development. In details *bootstrap*[6] *framework, Jquery*[7] *and D3*[8] *libraries* have been used to develop the GUI module, while mainly *Django*[9] *framework* has been used to develop the server module with the back-end interaction.

8. Conclusions

After briefly introducing all the problems that can arise when monitoring networks, this report has shown a general software architecture for the realization of a tool for monitoring and controlling geo-distributed and dual-stack computer networks. It has also discussed some implementation details concerning each *Component* of the proposed architecture and finally it has given references to a real and fully functional implementation of it.

6 Bootstrap. http://getbootstrap.com/
7 Jquery. https://jquery.com/
8 D3 - Data Driven documents. https://d3js.org/
9 Django. https://www.djangoproject.com/

9. Bibliography

[1] Information Sciences Institute - University of Southern California. (1981). RFC 791 - Internet Protocol.
Retrieved from https://tools.ietf.org/pdf/rfc791.pdf

[2] S. Deering, R. Hinden (1998). RFC 2460 - Internet Protocol, Version 6 (IPv6) Specification.
Retrieved from https://www.ietf.org/rfc/rfc2460.txt

[3] A. Gebrehiwot, M. Sommani, A. De Vita & A. Mancini (2012).
6Mon : Rogue IPv6 Router Advertisement Detection and Mitigation and IPv6 Address Utilization Network Monitoring Tool.

[4] P. Srisuresh & M. Holdrege (1999). RFC 2663 - IP Network Address Translator (NAT) Terminology and Considerations.
Retrieved from https://tools.ietf.org/html/rfc2663

[5] P. Mockapetris (1987).
RFC 1035 - Domain Names - Implementation and Specification.
Retrieved from https://www.ietf.org/rfc/rfc1035.txt

[6] J. Postel (1980). RFC 768 - User Datagram Protocol.
Retrieved from https://www.ietf.org/rfc/rfc768.txt

[7] D. Plummer (1982). RFC 826 - Ethernet Address Resolution Protocol.
Retrieved from https://tools.ietf.org/html/rfc826

[8] T. Narten, W. A. Simpson, E. Nordmark & H. Soliman (2007).
RFC 4861 - Neighbor Discovery for IP version 6 (IPv6).
Retrieved from https://tools.ietf.org/html/rfc4861

[9] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach & T. Berners-Lee (1999). RFC 2616 - Hypertext Transfer Protocol – HTTP1.1.
Retrieved from https://tools.ietf.org/pdf/rfc2616.pdf

[10] Information Sciences Institute - University of Southern California. (1981). RFC 793 - Transmission Control Protocol Specification.
Retrieved from https://www.ietf.org/rfc/rfc793.txt

[11] E. Rescorla (2000). RFC 2818 - HTTP Over TLS.
Retrieved from https://tools.ietf.org/html/rfc2818

[12] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot & E. Lear (1996). RFC 1918 - Address Allocation for Private Internets.
Retrieved from https://tools.ietf.org/html/rfc1918

[13] B. Haberman, R. Hinden (2008).
RFC 5175 - IPv6 Router Advertisement Flags Option.
Retrieved from https://tools.ietf.org/html/rfc5175

[14] R. Droms (1997). RFC 2131 - Dynamic Host Configuration Protocol.
Retrieved from https://www.ietf.org/rfc/rfc2131.txt

[15] D. McPherson, B. Dykes (2001).
VLAN Aggregation for Efficient IP Address Allocation
Retrieved from https://www.ietf.org/rfc/rfc3069.txt