EPS
UdG Escola Politècnica Superior

Incoming Exchange Student - Final Degree Project

Erasmus ■ Techno □ Other (specify):

Degree course: Enginyeria Informàtica Pla 1997

Title: Quality of Service driver for VITAM

Document: Final Degree Project

Student (Name & Surname): HNICHE Yassine

EPS Advisor: Pr. Jose Luis MARZO
Department: Informàtica i Matemàtica Aplicada

Delivered on (month/year): 01/2015

Project Work: Quality Of Service Driver for VITAM

Directed by:

- **Yassine HNICHE**

Supervised by:

Pr. Jose Luis MARZO

Thanks

- *I would like to take this opportunity to thank all the people who have contributed in some way to this report particularly Professor Jose Luis MARZO, who initiated the project.*

- *My thanks also go to Mr Lluís Fàbrega Soler who helped me with his appreciated remarks, useful advices and for all the time he awarded to me.*

- *I also thank all the members of laboratory who were always ready to help me.*

- *Then, I would like to thank all the members of "EPS" administration for their patience and the time they spent to explain the procedures at my arrival to Girona.*

- *Finally, I would like to thank Doctor Eric SALVAT my supervisor in IMERIR for his time and attention.*

Preface

Under the Erasmus program, I chose to work on a project in the
“Escola Politècnica Superior “in Girona during the first semester of 2014/2015.

When I first arrived in Girona, and after having a meeting with the VITAM team, my supervisor asked me to look for a test tool and adapt it to perform the quality of communication service of VITAM, this module will be developed at the same time as my classmates will make other functionalities based on HTML5 and WebRTC that are the multi-video player where the user can play recorded videos from the old sessions instantly, and the mailbox that should let text, audio or video message when the user is absent or offline.

To add more features to the tool, it have to be stable and should know how to deal with problems like latency or packet errors, that’s why my project is important for the pursuit of development.

This semester in EPS laboratory was also the opportunity for me to offer my knowledge to this project that is mainly devised to eHealth and remote assistance.

This Project assembles several technologies like Webrtc and HTML5, for the features, i needed to use my knowledge in virtualization, computer networks and systems administration, i also faced some problems and worked with my classmates for some common parts of the project.

Table of contents

The main project 6

VITAM 6

Virtualization 8

What is it? 8

Types of virtualization 8

- Type 1 hypervisor or bare metal: 8
- Type 2 hypervisor: 9

WebRTC & HTML5 9

- What is WebRTC 9
- What is HTML5 10

QOS 13

The QOS driver project 16

QOS Driver 16

QOS Open source algorithms: 16

Virtualization environment 18

Networking configuration 21

Advantages 21

Route tables: 22

NETEM 25

How NetEM works 26

About traffic shaping 26

Terminology 27

WireShark 28

Features: 28

Using Wireshark: 29

How does our solution works? 37

Tests 38

Ping: 38

Video test 40

Wireshark tests: 42

- 1st scenario 42
- 2nd scenario 43
- 3rd scenario 44

Project management 45

Risks management 45

Methodology 46

Conclusion 48

Assessment of the situation 48

- Issue and problems before starting the project:.. 48
- Solutions proposed and realized: 48

Personal assessment 48

Bibliography 50

Annex..51

Install Wirshark... 51

Install Wirshark... 54

Glossary.. 55

Parts of the Project

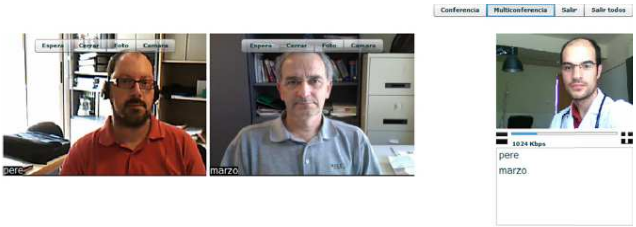
This project is divided in several parts, The first part is to know VITAM, how it works and for which use was it created, the second part is about to work on virtualization to know the best environment that can simulate our Client-server configuration, then , and for a best understanding of VITAM's functioning, i have to expose the two technologies that contributed to success of this tool, and to conclude this chapter, the final part is an explanation of the QOS.

VITAM

VITAM is a project of BCDS (Broadband Communications and Distributed Systems), Universitat de Girona. It's a set of management and remote control tools for videoconferencing mainly devised to eHealth and remote assistance; it offers Simple and Usable Interfaces extremely simplified to facilitate at most the use of technology by non-experts.

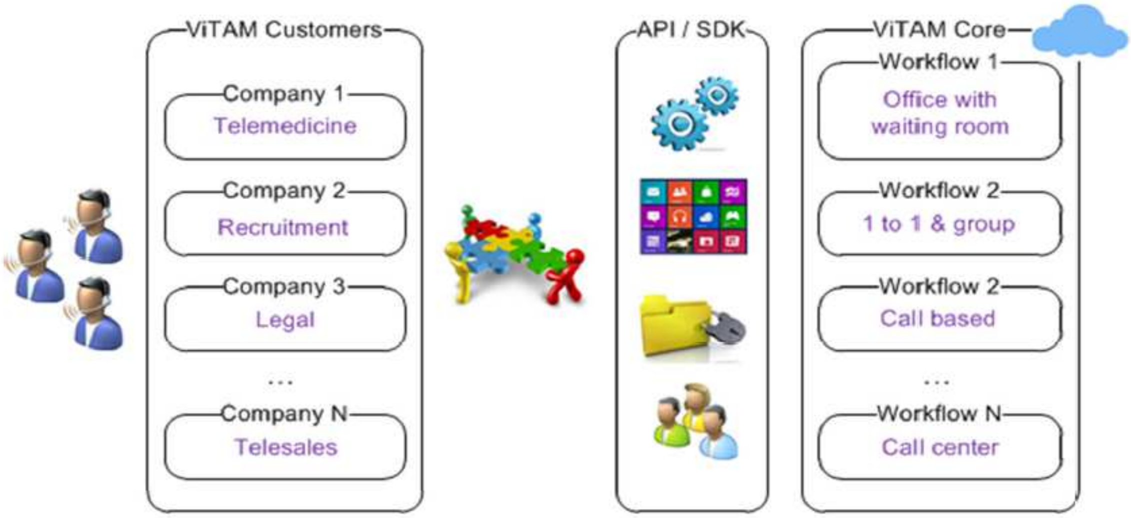
This tool can be used under several operating systems (Windows, Mac, Android and iOS) and in all the smart devices like PC, laptop, tablet and of course smartphones. It can be easily integrated in or with other applications through an API or SDK, as well as connect with external services like electronic health record, calendar …

The principal service offered by this tool is the Multi-party videoconferencing that is 1 to 1 or multi video communication and in a single and unified interface. User interfaces are asymmetrical and simplified, differing from the user and the specialist (administrator).



Despite the fact that this software opens a channel between users, Privacy and Data Protection is an important point, VITAM ensures privacy protection by providing communications and data protection and enables its deployment on certified medical data management servers. Data in VITAM are encrypted using SSL.

Currently, this software implements two innovative operation modes not available in the market: the online consultation with waiting room and the videoconferencing call centre, it also offers customised development and product personalization to cover any specific customer needs.



Virtualization

What is it?

Virtualization operated simultaneously tightly (or not) multiple operating systems on a single physical machine or more.

A virtual machine (VM) is an operating system OS or application environment that is installed on software which imitates dedicated hardware. The end user has the same experience on a virtual machine as they would have on dedicated hardware.

Specialized software called a hypervisor emulates the PC client or server's CPU, memory, hard disk, network and other hardware resources completely, enabling virtual machines to share the resources. The hypervisor can emulate multiple virtual hardware platforms that are isolated from each other, allowing virtual machines to run Linux and Windows server operating systems on the same underlying physical host.

Virtualization saves costs by reducing the need for physical hardware systems. Virtual machines more efficiently use hardware, which lowers the quantities of hardware and associated maintenance costs, and reduces power and cooling demand.

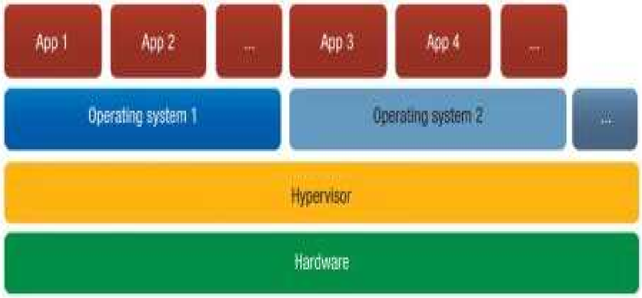
They also ease management because virtual hardware does not fail. Administrators can take advantage of virtual environments to simplify backups, disaster recovery, new deployments and basic system administration tasks.

Types of virtualization

- **Type 1 hypervisor or bare metal:**

Type 1 hypervisor is a tool that stands between the hardware layer and software layer:

- It access to machine components and has its own core, it is above this core, and the OS will be installed.
- It therefore controls the OS from the hardware layer.
- It is administered via an interface for managing virtual machines and it is much more powerful than Type 2 hypervisors

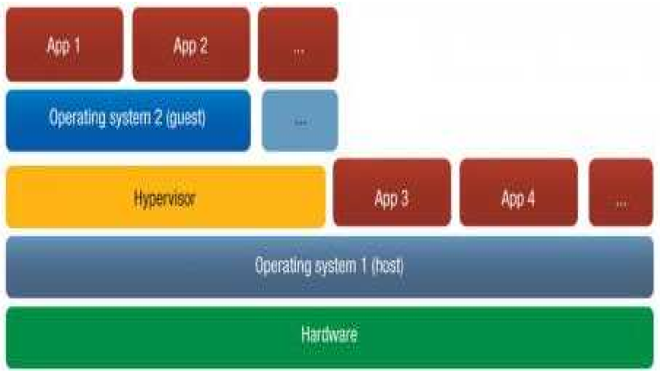


Example type 1 hypervisor:

- ***VmWarevSphere***
- ***Microsoft Hyper-V***
- ***XEN***
- ***KVM (open source)***

- **Type 2 hypervisor:**

The type 2 hypervisor is an application installed on an operating system; the application depends on the operating system.
The performance is reduced in comparison with type 1 hypervisors because access to the hardware (CPU, RAM ...) is via an intermediate layer.
It offers a perfect seal between the systems installed farms.



Example type 2 hypervisor:

- ***VmWare Workstation, Fusion, Player***
- ***Oracle VirtualBox***
- ***Microsoft Virtual PC***
- ***QEMU (open source)***

We chose type 2 because unlike the type 1 hypervisor, a type 2 hypervisor requires you to first install an OS.

These hypervisors are basically like applications that install on a guest OS. This approach provides better hardware compatibility than type 1, because the OS is responsible for the hardware drivers instead of the hypervisor.

Finally type 2 hypervisor are common for desktops, because they allow you to run multiple operating systems.

WebRTC & HTML5

- **What is WebRTC**

One of the last major challenges for the web is to enable human communication via voice and video: Real Time Communication, RTC for short. RTC should be as natural in a web application as entering text in a text input. Without it, we're limited in our ability to innovate and develop new ways for people to interact.

Historically, RTC has been corporate and complex, requiring expensive audio and video technologies to be licensed or developed in house. Integrating RTC technology with existing content, data and services has been difficult and time consuming, particularly on the web.

Gmail video chat became popular in 2008, and in 2011 Google introduced Hangouts, which use the Google Talk service (as does Gmail). Google bought GIPS, a company which had developed many components required for RTC, such as codecs and echo cancellation techniques. Google open sourced the technologies developed by GIPS and engaged with relevant standards bodies at the IETF and W3C to ensure industry consensus. In May 2011, Ericsson built the first implementation of WebRTC.

WebRTC has now implemented open standards for real-time, plug-in-free video, audio and data communication. The need is real:

- Many web services already use RTC, but need downloads, native apps or plug-ins. These include Skype, Facebook (which uses Skype) and Google Hangouts (which use the Google Talk plug-in).
- Downloading, installing and updating plug-ins can be complex, error prone and annoying.
- Plug-ins can be difficult to deploy, debug, troubleshoot, test and maintain—and may require licensing and integration with complex, expensive technology. It's often difficult to persuade people to install plug-ins in the first place!

- **What is HTML5**

Is a core technology mark-up language of the Internet used for structuring and presenting content for the World Wide Web. As of October 2014 this is the final and complete fifth revision of the HTML standard of the World Wide Web Consortium (W3C). The previous version, HTML 4, was standardized in 1997.

The architecture of the WebRTC API is based on a triangular construction involving a server and two peers. Both browsers download from a server a JavaScript application to their local context. The server is used as a meeting point to coordinate communication between browsers until the direct connection between browsers is established.

The downloaded application uses the WebRTC API to communicate with the local context. The goal is to have a client application in JavaScript and HTML5 interacting with the browser through the WebRTC API.

Flow exchanges between browsers may pass upon various servers that take care to modify, translate, or manage the signal required, allowing for example crossing firewalls, proxies and NAT.

To establish a connection using WebRTC standard , A and B browsers must be connected simultaneously to the service page and upload the HTML and JavaScript code to keep the connection open or HTTPS socket. When the browser A wants to connect with B, API instantiates an object Peer Connection which, once created, allows establishing the media or data streams. It is also necessary for a videoconference for example; the users A and B agree to share their webcam and / or their microphone.

Once this PeerConnection object created by A, the browser sends the server a packet containing information on shared media and a signal linking the connection to A. The server will decode the packet and identify that it is a destinated to B and therefore will send a signal to B. B is notified of the wish of A to connect and accepts or not the request. If accepted, the same process takes place between B and A, this time to establish a bidirectional connection. Once it has been established, or the media data stream may be added to the free connection. Once it has been established, or the media data stream may be added to the connection freely.

For example in the context of a video streaming peer-to -peer between browsers, the user downloads from a server the metadata of the video he wants to watch as well as a list of peers and having available all or part of video. Establishing a connection with peers allows, by the data stream, to download pieces of the video that are reassembled after checking their integrity and launch the video in an HTML5 player.

The WebRTC API builds on existing standards such as STUN, ICE, TURN, DTLS or SRTP, technologies parts from the LIBJINGLE project.

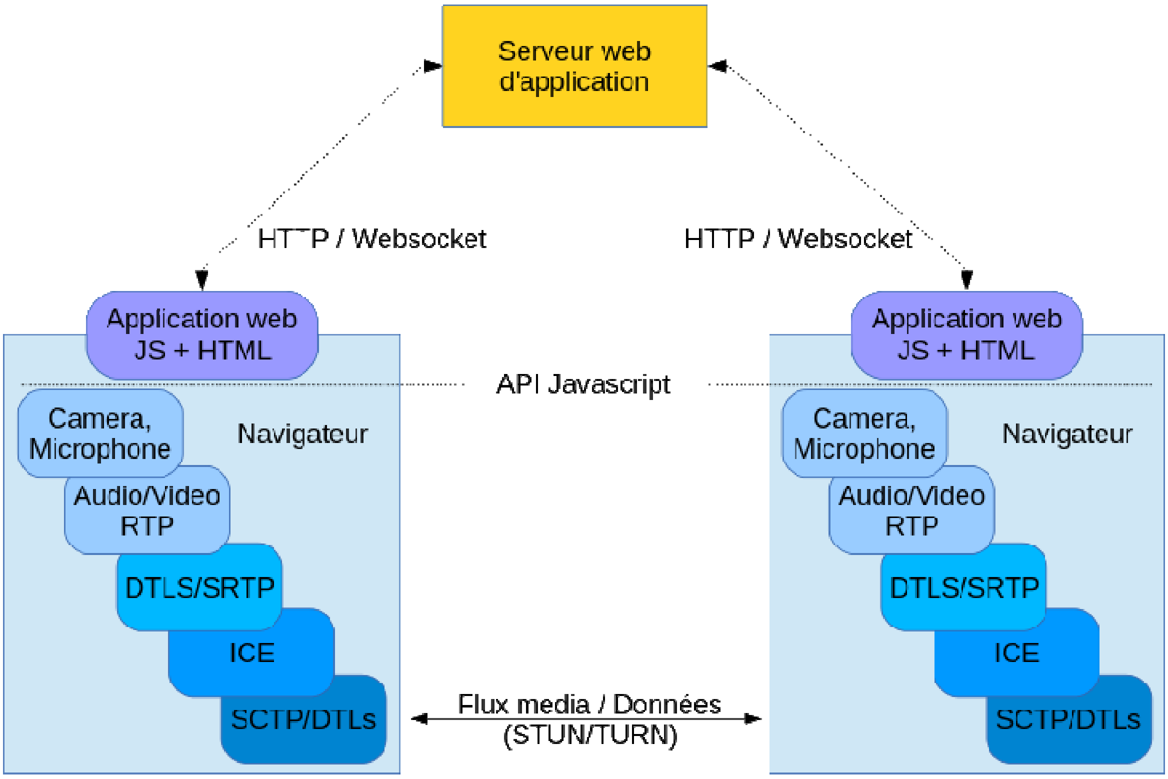


Figure 1: Architecture of WebRTC Application

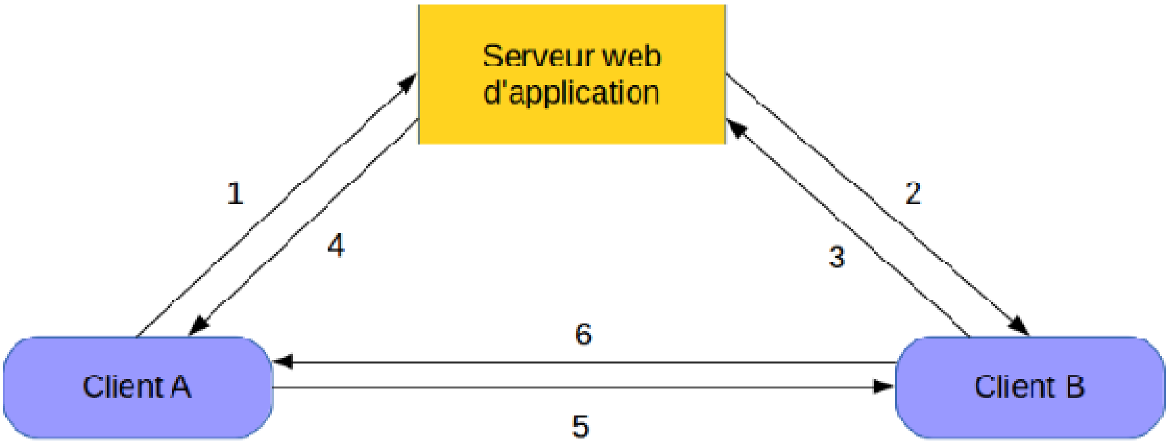


Figure 2: Establishing a connection between two clients using WebRTC

- 1: A request to the server a connection with B.
- 2: The server relays the request of A.
- 3: If B accepts, it sends a connection request to A.
- 4: The server relays the request to A.
- 5 and 6: Bidirectional PeerConnection is established.

QOS

Quality of service (QoS) is the overall performance of telephony or computer networks particularly the performance seen by the users of the network. In our case we want to perform the ability of the VITAM system to resolve problems such as error rates, bandwidth, throughput, transmission delay, availability, jitter, etc.

Quality of service is particularly important for the transport of traffic with special requirements. In particular, much technology has been developed to allow computer networks to become as useful as telephone networks for audio conversations, as well as supporting new applications with even stricter service demands.

In the field of telephony, QoS was defined by the ITU in 1994 comprising requirements on all the aspects of a connection, such as service response time, loss, signal-to-noise ratio, crosstalk, echo, interrupts, frequency response, loudness levels, and so on. A subset of telephony QoS is grade of service (GoS) requirements, which comprises aspects of a connection relating to capacity and coverage of a network, for example guaranteed maximum blocking probability and outage probability.

In the field of computer networking and other packet-switched telecommunication networks, the traffic engineering term refers to resource reservation control mechanisms rather than the achieved service quality. Quality of service is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow. For example, a required bit rate, delay, jitter, packet dropping probability and/or bit error rate may be guaranteed. Quality of service guarantees are important if the network capacity is insufficient, especially for real-time streaming multimedia applications such as voice over IP, online games and IP-TV, since these often require fixed bit rate and are delay sensitive, and in networks where the capacity is a limited resource, for example in cellular data communication.

A network or protocol that supports QoS may agree on a traffic contract with the application software and reserve capacity in the network nodes, for example during a session establishment phase. During the session it may monitor the achieved level of performance, for example the data rate and delay, and dynamically control scheduling priorities in the network nodes. It may release the reserved capacity during a tear down phase.

A best-effort network or service does not support quality of service. An alternative to complex QoS control mechanisms is to provide high quality communication over a best-effort network by over-provisioning the capacity so that it is sufficient for the expected peak traffic load. The resulting absence of network congestion eliminates the need for QoS mechanisms.

QoS is sometimes used as a quality measure, with many alternative definitions, rather than referring to the ability to reserve resources. QoS sometimes refers to the level of quality of service, i.e. the guaranteed service quality. High QoS is often confused with a high level of performance or achieved service quality, for example high bit rate, Low latency and low bit error probability.

In packet-switched networks, quality of service is affected by various factors, which can be divided into “human” and “technical” factors. Human factors include: stability of service, availability of service, delays, user information. Technical factors include: reliability, scalability, effectiveness, maintainability, grade of service, etc.

Many things can happen to packets as they travel from origin to destination, resulting in the following problems as seen from the point of view of the sender and receiver:

Low throughput

Due to varying load from disparate users sharing the same network resources, the bit rate (the maximum throughput) that can be provided to a certain data stream may be too low for real-time multimedia services if all data streams get the same scheduling priority.

Dropped packets

the routers might fail to deliver (drop) some packets if their data loads are corrupted, or the packets arrive when the router buffers are already full. The receiving application may ask for this information to be retransmitted, possibly causing severe delays in the overall transmission.

Errors

Sometimes packets are corrupted due to bit errors caused by noise and interference, especially in wireless communications and long copper wires. The receiver has to detect this and, just as if the packet was dropped, may ask for this information to be retransmitted.

Latency

it might take a long time for each packet to reach its destination, because it gets held up in long queues, or it takes a less direct route to avoid congestion. This is different from throughput, as the delay can build up over time, even if the throughput is almost normal. In some cases, excessive latency can render an application such as VoIP or online gaming unusable.

Jitter

Packets from the source will reach the destination with different delays. A packet's delay varies with its position in the queues of the routers along the path between source and destination and this position can vary unpredictably. This variation in delay is known as jitter and can seriously affect the quality of streaming audio and/or video.

Out-of-order delivery

when a collection of related packets is routed through a network, different packets may take different routes, each resulting in a different delay. The result is that the packets arrive in a different order than they were sent. This problem requires special additional protocols responsible for rearranging out-of-order packets to an isochronous state once they reach their destination. This is especially important for video and VoIP streams where quality is dramatically affected by both latency and lack of sequence.

An alternative and disputable definition of QoS, used especially in application layer services such as telephony and streaming video, is requirements on a metric that reflects or predicts the subjectively experienced quality. In this context, QoS is the acceptable cumulative effect on subscriber satisfaction of all imperfections affecting the service. Other terms with similar meaning are the quality of experience (QoE) subjective business concept, the required “user perceived performance”, the required “degree of satisfaction of the user” or the targeted “number of happy customers”.

THE PROJECT

QOS Driver

QOS driver is a module that the project VITAM needs to enhance the communication between the users of the service. The drivers have to be able to eliminate some packets and add delays for the deliverance. The CODECS of VITAM must be able to resolve the problems and guarantee more stability for the system.

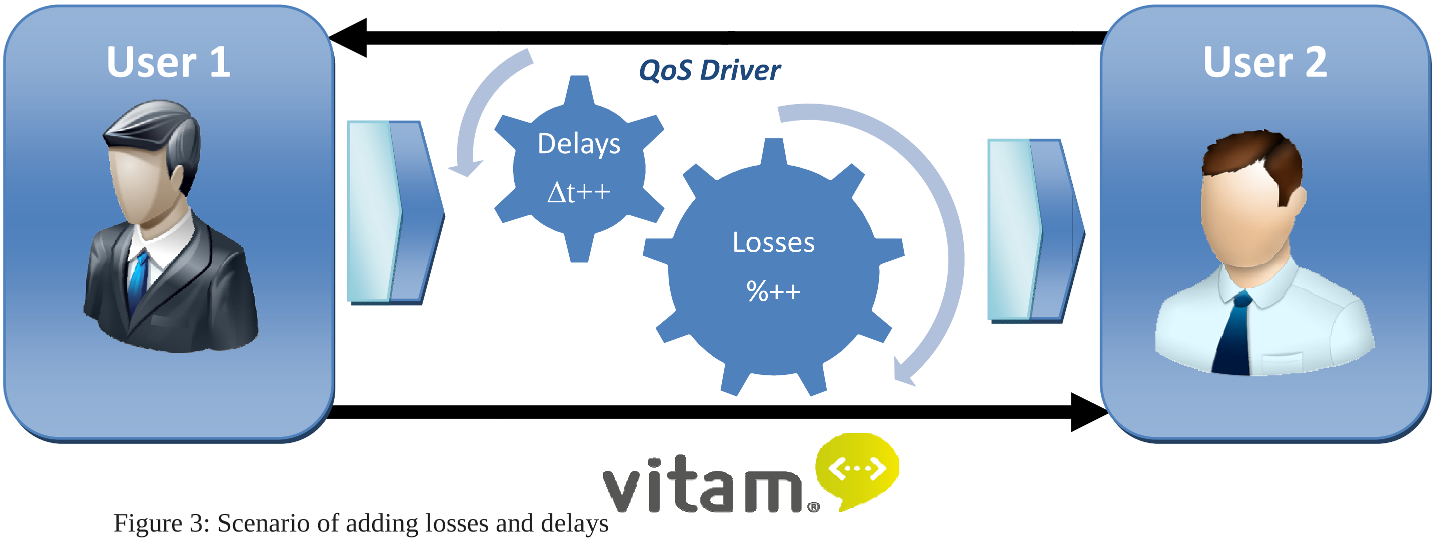


Figure 3: Scenario of adding losses and delays

So this project focuses on finding a solution to test the ability of VITAM to handle network errors in order to improve the quality of service.

To realise the project, knowledge of networking, systems and open source software are required. It's important to do some researches about the existing solutions, and then try to define the best candidate that can satisfy all the needs of this project.

QOS Open source algorithms:

- **TC**

On Linux, the quality of service (QoS: Quality of Service) is controlled by the kernel via the TC tool. The configuration is independent of Netfilter, but a system of packet marking allows TC to identify the packages that were intercepted by certain rules and avoids duplication.

- **PRIO**

Called PREQ in BSD, is a basic algorithm based on QoS priorities. Each stream has a certain priority and when a packet has to be sent by the KERNEL, it first chooses the one with the highest priority.

- **HFSC**

HFSC QoS algorithm is to simultaneously satisfy two requirements:

-The transmission rate

-The transmission delay

Indeed, some protocols require both a CIR data, and low transmission delay is the case for example of VOIP.

Unlike PRIO HFSC it is possible with much more finely control the flow.

Virtualization environment

Using the type 2 of virtualization, we choose that our configuration will be the following:

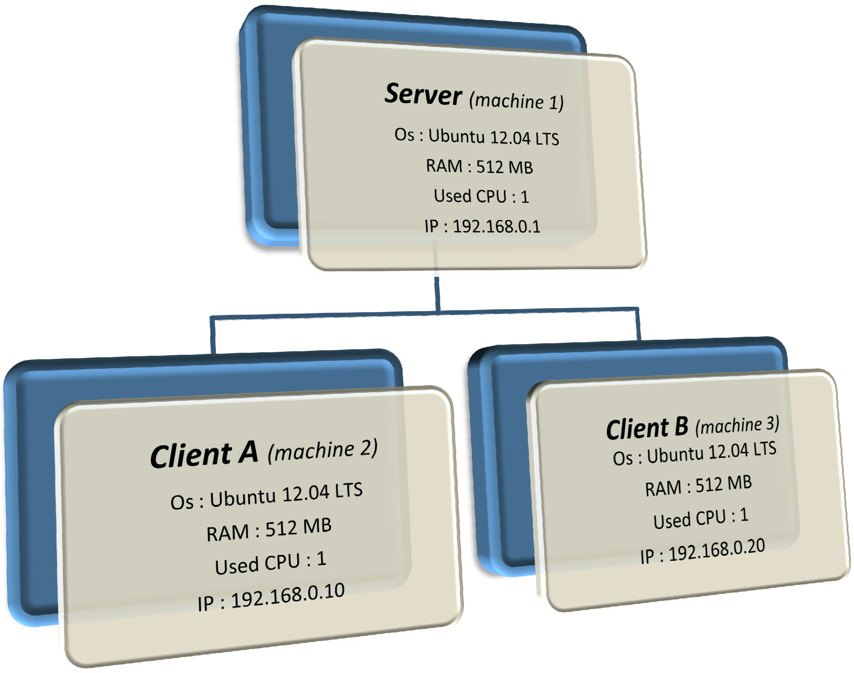


Figure 4: Configuration before the distribution of the topics

Operating system:

Using Oracle Virtualbox, we have installed 3 machines with Ubuntu 12.04 LTS; this version is known for its stability and its huge community of users.

Networking:

All machines were attached to internal. Some changes were necessary and are explained in the networking chapter.

Services and protocols:

Some services and protocols were important to be installed before Mr marzo gives us the topics, those services are attached to VITAM functioning and we may need them:

- Apache: Apache supports a variety of features, many implemented as compiled modules which extend the core functionality.
- Mysql: is a popular choice of database for use in web applications, we will certainly need it for our development.

- **ICE:** Interactive Connectivity Establishment (ICE) is a framework to allow your web browser to connect with peers. We'll need to bypass firewalls that would prevent opening connections, so ICE will give us a unique address and relay data through a server if our router doesn't allow you to directly connect with peers.
- **STUN:** Session Traversal Utilities for NAT (STUN) is a protocol to discover our public address and determine any restrictions in the router that would prevent a direct connection with a peer

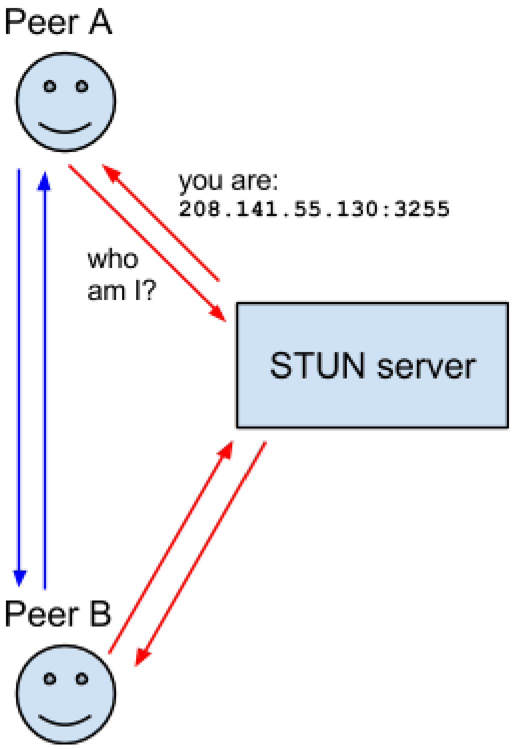


Figure 5: STUN functioning

- **NAT:** Network Address Translation (NAT) is used to give our device a public IP address. A router will have a public IP address and every device connected to the router will have a private IP address. Then Requests will be translated from the device's private IP to the router's public IP with a unique port.
- **TURN:** Traversal Using Relays around NAT (TURN) is meant to bypass the Symmetric NAT restriction by opening a connection with a TURN

server and relaying all information through that server.

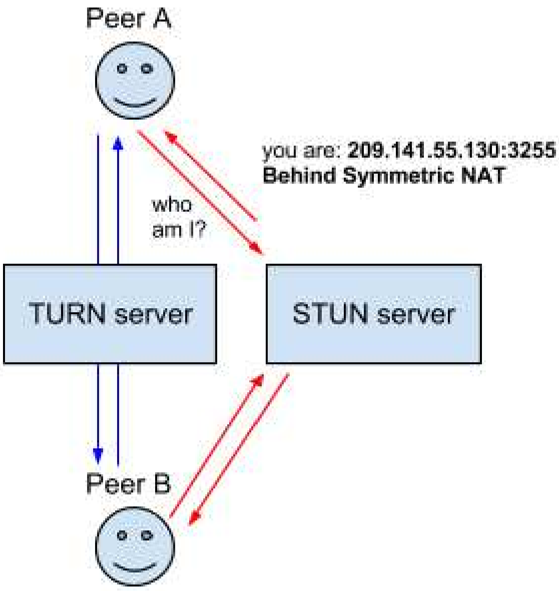


Figure 6: TURN Functioning

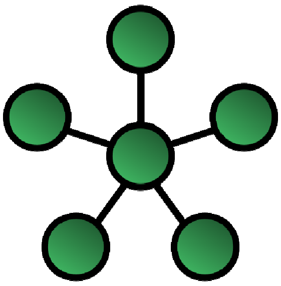
Ports:

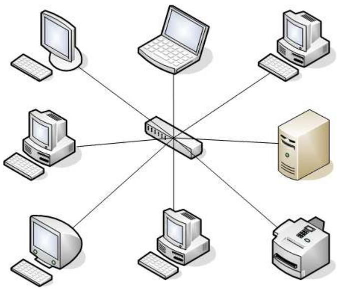
This is a list of some ports that could be used:

Service	Reserved port
MYSQL	3306
APACHE	80/443
STUN/TURN	80(can be changed)

Networking configuration

Star networks are one of the most common computer network topologies. In its simplest form, a star network consists of one central switch, hub or computer, which acts as a conduit to transmit messages. This consists of a central node, to which all other nodes are connected; this central node provides a common connection point for all nodes through a hub. In star topology, every node (computer workstation or any other peripheral) is connected to a central node called a hub or switch. The switch is the server and the peripherals are the clients. Thus, the hub and leaf nodes, and the transmission lines between them, form a graph with the topology of a star. If the central node is passive, the originating node must be able to tolerate the reception of an echo of its own transmission, delayed by the two-way transmission time (i.e. to and from the central node) plus any delay generated in the central node. An active star network has an active central node that usually has the means to prevent echo-related problems.





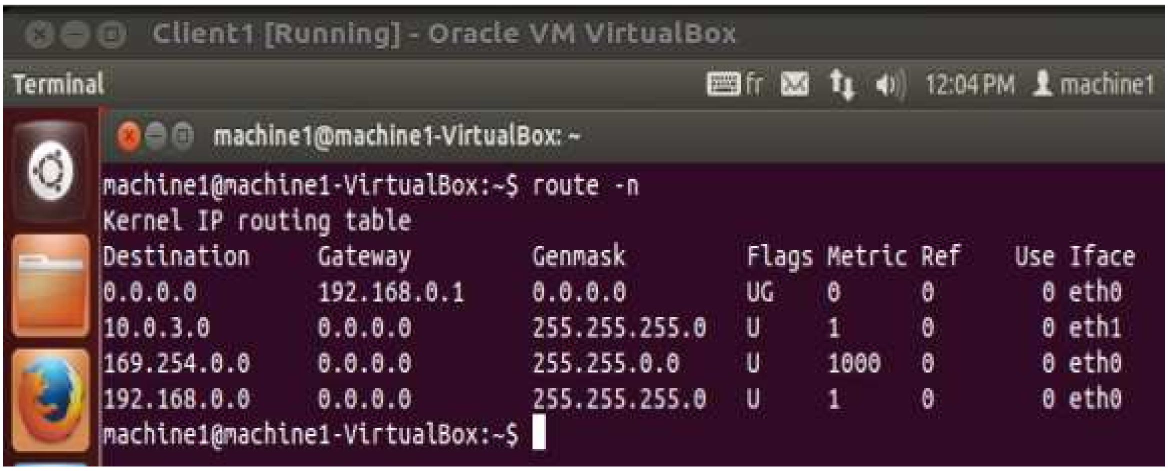
Advantages

- **Better performance:** Star topology prevents the passing of data packets through an excessive number of nodes. At most, 3 devices and 2 links are involved in any communication between any two devices. Although this topology places a huge overhead on the central hub, with adequate capacity, the hub can handle very high utilization by one device without affecting others.
- **Isolation of devices:** Each device is inherently isolated by the link that connects it to the hub. This makes the isolation of individual devices straightforward and amounts to disconnecting each device from the others. This isolation also prevents any non-centralized failure from affecting the network.
- **Benefits from centralization:** As the central hub is the bottleneck, increasing its capacity, or connecting additional devices to it, increases the size of the network very easily. Centralization also allows the inspection of traffic through the network. This facilitates analysis of the traffic and detection of suspicious behaviour.
- Easy to detect faults and to remove parts.
- No disruptions to the network when connecting or removing devices.
- Installation and configuration is easy since every one device only requires a link and one input/output port to connect it to any other device(s).

In our case, we choose this topology to trouble the central router interfaces and to have 3 separated networks that make us able to perform this tool in different scenarios.

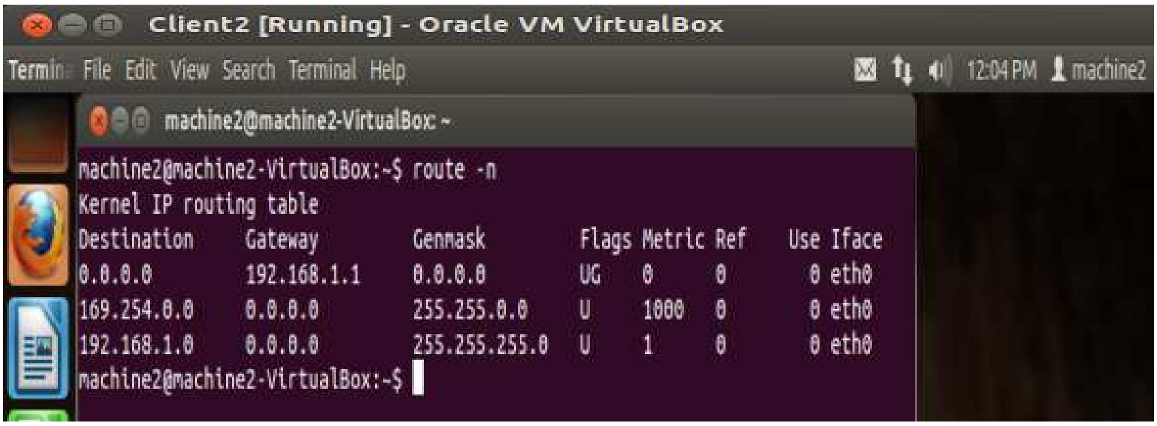
Route tables:

- **Client 1:**


Client1 [Running] - Oracle VM VirtualBox
Terminal
fr
12:04 PM
machine1
machine1@machine1-VirtualBox: ~

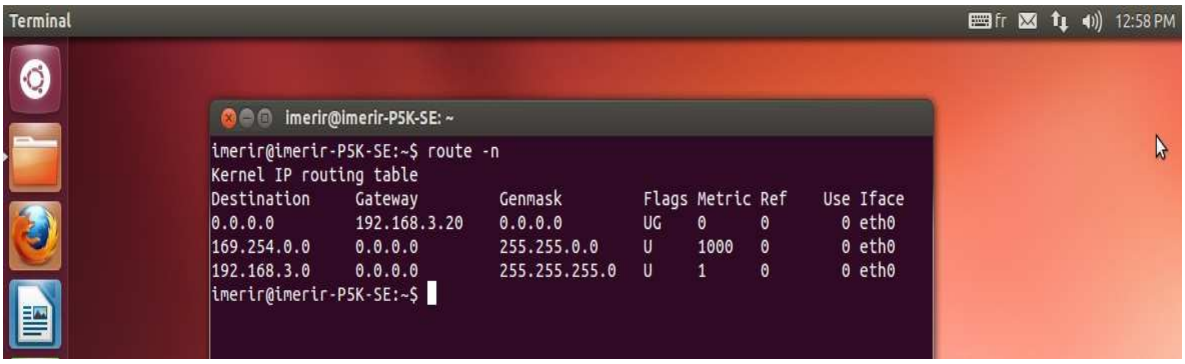
```
machine1@machine1-VirtualBox:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.0.1     0.0.0.0         UG    0      0        0 eth0
10.0.3.0        0.0.0.0         255.255.255.0   U     1      0        0 eth1
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eth0
192.168.0.0     0.0.0.0         255.255.255.0   U     1      0        0 eth0
machine1@machine1-VirtualBox:~$
```

- **Client 2:**


Client2 [Running] - Oracle VM VirtualBox
Termin
File
Edit
View
Search
Terminal
Help
12:04 PM
machine2
machine2@machine2-VirtualBox: ~

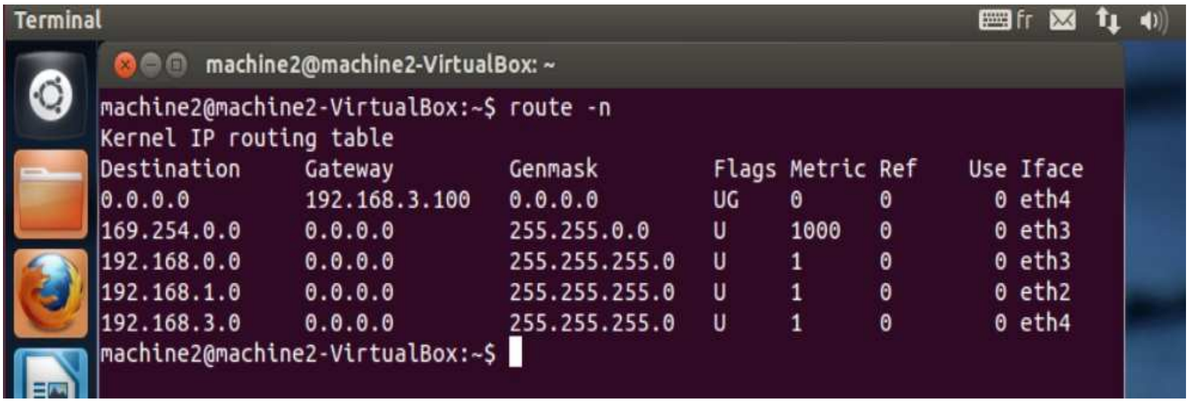
```
machine2@machine2-VirtualBox:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG    0      0        0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     1      0        0 eth0
machine2@machine2-VirtualBox:~$
```

- **Client 3:**



```
imerir@imerir-P5K-SE:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.3.20    0.0.0.0         UG    0      0        0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eth0
192.168.3.0     0.0.0.0         255.255.255.0   U     1      0        0 eth0
imerir@imerir-P5K-SE:~$
```

- **Linux-router:**



```
machine2@machine2-VirtualBox:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.3.100   0.0.0.0         UG    0      0        0 eth4
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eth3
192.168.0.0     0.0.0.0         255.255.255.0   U     1      0        0 eth3
192.168.1.0     0.0.0.0         255.255.255.0   U     1      0        0 eth2
192.168.3.0     0.0.0.0         255.255.255.0   U     1      0        0 eth4
machine2@machine2-VirtualBox:~$ 
```

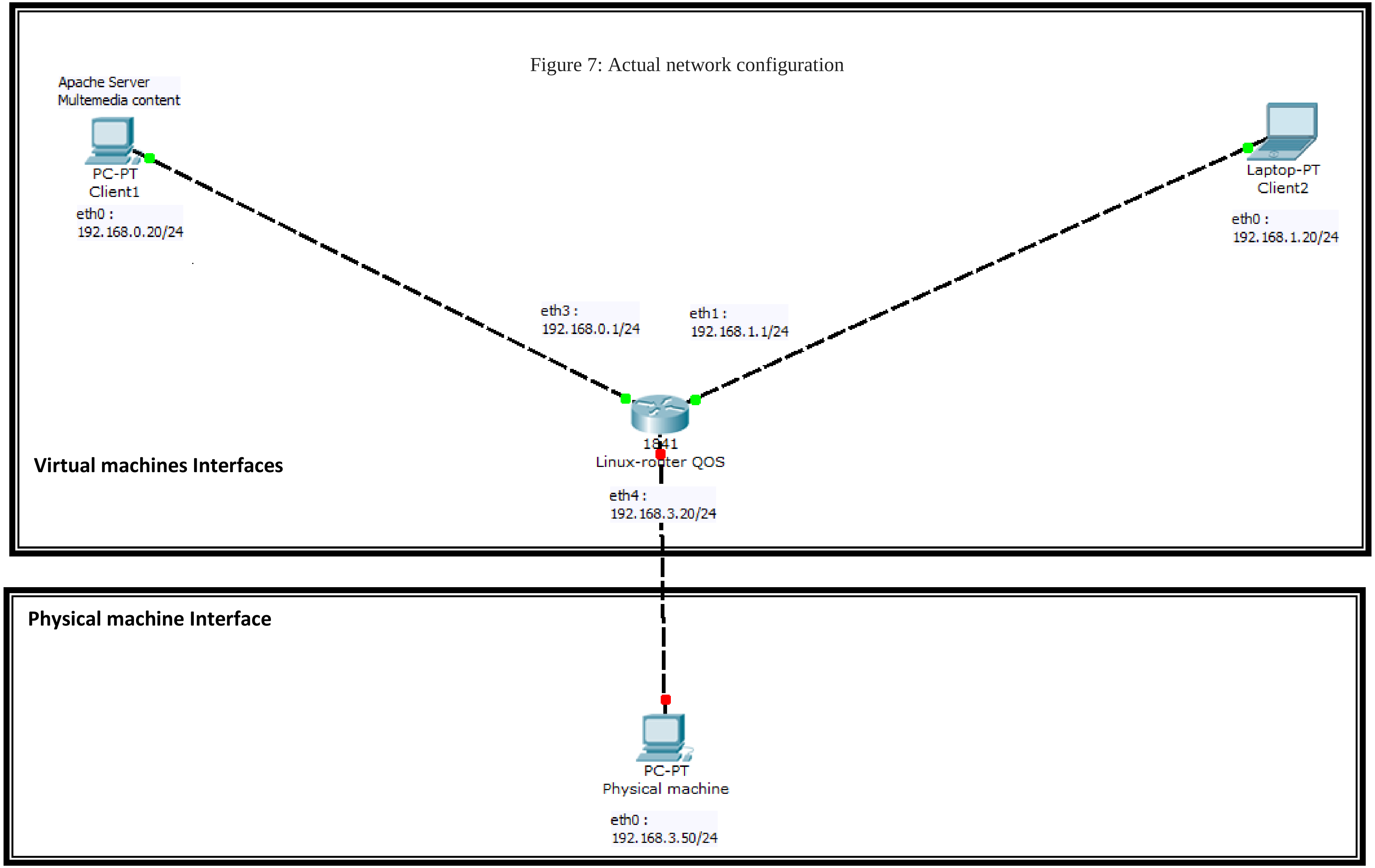


Figure 7: Actual network configuration

NETEM

NetEM (Network Emulation) is an enhancement of the Linux traffic control facilities that allow adding several characteristics to packets outgoing from a selected network interface. NetEm is built using the existing Quality Of Service (QOS) and Differentiated Services (diffserv) facilities in the Linux kernel.

NetEM capabilities include delay (it delays each packet); loss (it drops some packets); duplication (it duplicates some packets); and corruption (it introduces a single bit error at a random offset in a packet).

The configuration of this module is done via the command line tc .

The delay is the transit time of an IP packet network. It depends on a lot of parameters (crossing equipment, buffer size and physical distance between two points on the network). We will use the delay command that will simulate a transit time X ms on all outgoing IP packets at the network interface. We will use the command "ping" to make sure everything works as expected.

The classic configuration could be one of the following:

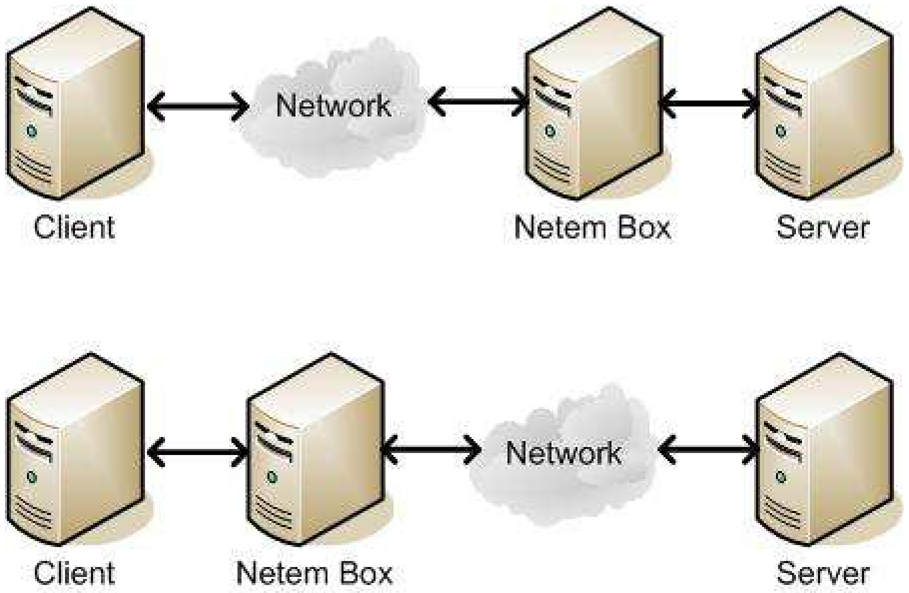


Figure 8: NETEM configuration

In our environment (see Figure 7), the NetEM box also plays the role of a Linux router. Recent Linux kernels have built-in network traffic shaping capabilities. Those capabilities, in combination with the command-line tool tc (a part of the iproute2 package) can be used to set a bandwidth limit on one of your network interfaces, and even on incoming traffic on a specific port.

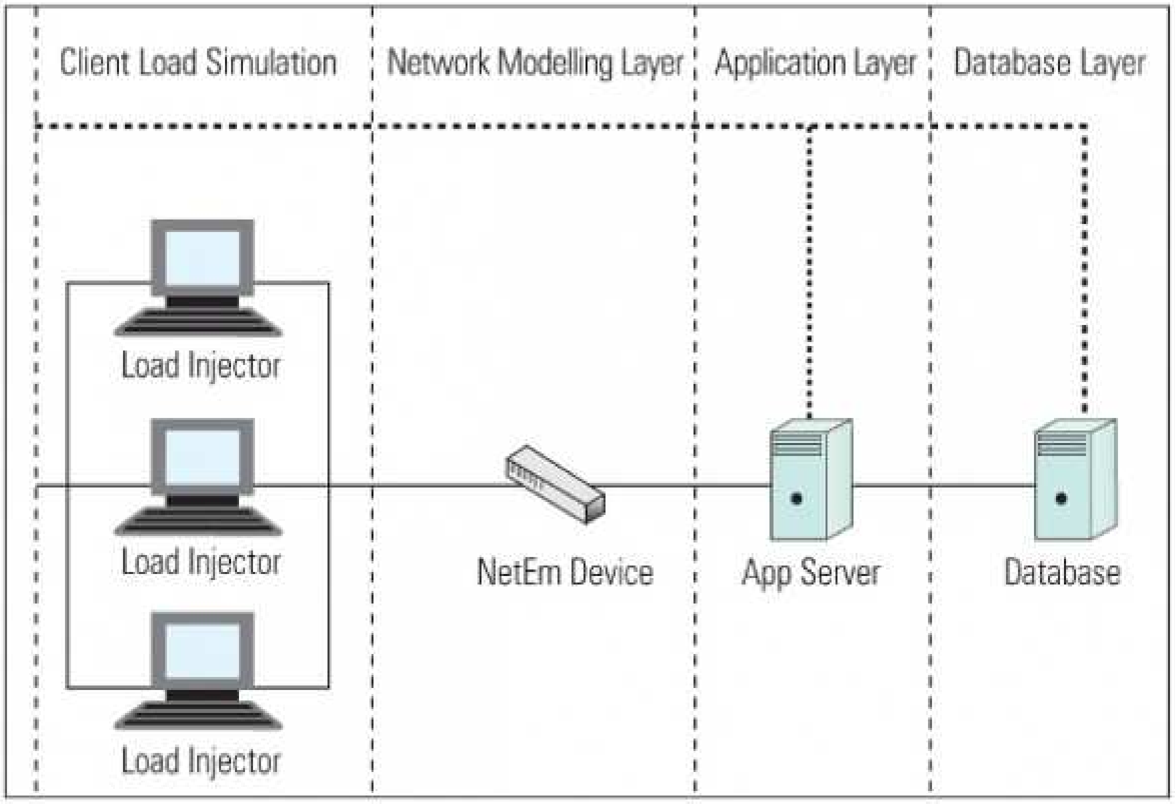


Figure 9: NETEM BOX in a Client-Server configuration

How NetEM works

NetEM consists of two components — a tiny kernel module for a queuing discipline, and a command-line utility to configure it. The kernel module has been integrated in 2.6.8 (and 2.4.28) or later, and the command is part of the iproute2 package. The command-line utility communicates with the kernel via the netlink socket interface. It encodes its requests into a standard message format, which the kernel decodes.

The queuing layer exists between the network device and the protocol output. The default queuing discipline is a simple FIFO packet queue. Queuing discipline consists of two key interfaces; one queue packets to be sent, and the other releases packets to the network device for transmission. The queuing discipline makes the policy decision of which packets to send, based on the current settings.

About traffic shaping

Traffic shaping is an attempt to control network traffic by prioritizing network resources. It guarantees certain bandwidth, based on predefined policy rules. Traffic shaping uses traffic classification, policy rules, queue disciplines and quality of service (QoS).

The need for traffic shaping arises because network bandwidth is an expensive resource that is shared among many parties in an organization, and some applications require guaranteed bandwidth and priority. Traffic shaping lets you: (1) control network services, (2) limit bandwidths, and (3) guarantee Quality of Service

(QoS). Intelligently managed traffic shaping improves network latency, service availability and bandwidth utilization.

Terminology

qdisc — A queue discipline (qdisc) is a set of rules that determine the order in which arrivals are serviced. It is a packet queue with an algorithm that decides when to send each packet.

Classless qdisc — A qdisc with no configurable internal subdivision.

Classful qdisc — A qdisc that may contain classes; classful qdiscs allow packet classification (Class-Based Queueing and others)

Root qdisc — the root qdisc is attached to each network interface — either classful or classless.

Egress qdisc — Works on outgoing traffic only.

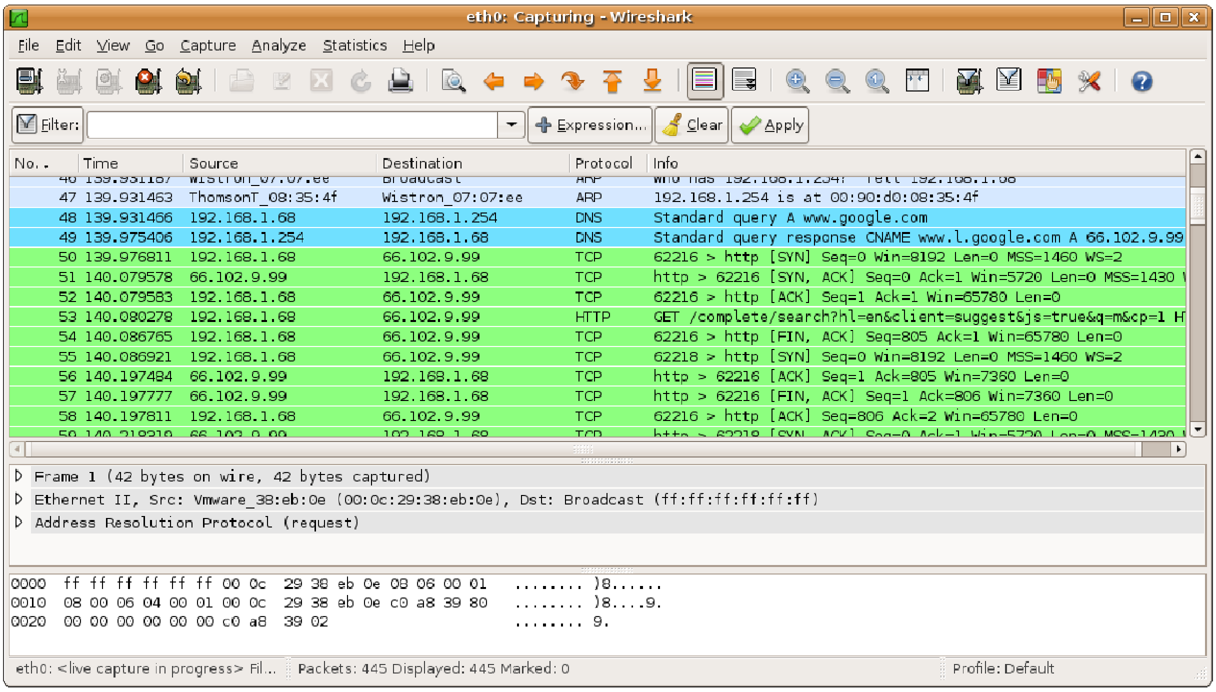
Ingress qdisc — Works on incoming traffic.

Filter — Classification can be performed using filters

WireShark

Wireshark Is a free packet analyzer used in the troubleshooting and analysis of computer networks , Development of protocols education and reverse engineering. Its origin(Ethereal) is amended in May 2006 paid parental Questions to trademark law . Wireshark using the GTK + library Software pour the User interface and Implementation son for the pcap packet capture ; It Works on Many compatible UNIX Environments As GNU / Linux, FreeBSD, NetBSD, OpenBSD ou Mac OSX , but aussi Microsoft Windows.

Today, Wireshark now recognizes 1,515 protocols.


eth0: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply

```
No. .  Time        Source            Destination       Protocol  Info
  46 139.931187  Wistron_07:07:ee  Broadcast         ARP       Who has 192.168.1.254?  Tell 192.168.1.68
  47 139.931463  ThomsonT_08:35:4f Wistron_07:07:ee  ARP       192.168.1.254 is at 00:90:d0:08:35:4f
  48 139.931466  192.168.1.68      192.168.1.254     DNS       Standard query A www.google.com
  49 139.975406  192.168.1.254     192.168.1.68      DNS       Standard query response CNAME www.l.google.com A 66.102.9.99
  50 139.976811  192.168.1.68      66.102.9.99       TCP       62216 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
  51 140.079578  66.102.9.99       192.168.1.68      TCP       http > 62216 [SYN, ACK] Seq=0 Ack=1 Win=5720 Len=0 MSS=1430 W
  52 140.079583  192.168.1.68      66.102.9.99       TCP       62216 > http [ACK] Seq=1 Ack=1 Win=65780 Len=0
  53 140.080278  192.168.1.68      66.102.9.99       HTTP      GET /complete/search?hl=en&client=suggest&js=true&q=m&cp=1 H
  54 140.086765  192.168.1.68      66.102.9.99       TCP       62216 > http [FIN, ACK] Seq=805 Ack=1 Win=65780 Len=0
  55 140.086921  192.168.1.68      66.102.9.99       TCP       62218 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
  56 140.197484  66.102.9.99       192.168.1.68      TCP       http > 62216 [ACK] Seq=1 Ack=805 Win=7360 Len=0
  57 140.197777  66.102.9.99       192.168.1.68      TCP       http > 62216 [FIN, ACK] Seq=1 Ack=806 Win=7360 Len=0
  58 140.197811  192.168.1.68      66.102.9.99       TCP       62216 > http [ACK] Seq=806 Ack=2 Win=65780 Len=0
  59 140.218318  66.102.9.99       192.168.1.68      TCP       http > 62218 [SYN, ACK] Seq=0 Ack=1 Win=5720 Len=0 MSS=1430 W
```

```
▷ Frame 1 (42 bytes on wire, 42 bytes captured)
▷ Ethernet II, Src: Vmware_38:eb:0e (00:0c:29:38:eb:0e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▷ Address Resolution Protocol (request)
```

```
0000  ff ff ff ff ff ff 00 0c  29 38 eb 0e 08 06 00 01   ........ )8......
0010  08 00 06 04 00 01 00 0c  29 38 eb 0e c0 a8 39 80   ........ )8....9.
0020  00 00 00 00 00 00 c0 a8  39 02                     ........ 9.
```

eth0: <live capture in progress> Fil... Packets: 445 Displayed: 445 Marked: 0 Profile: Default

Features:

This useful tool has a rich feature set which includes the following:

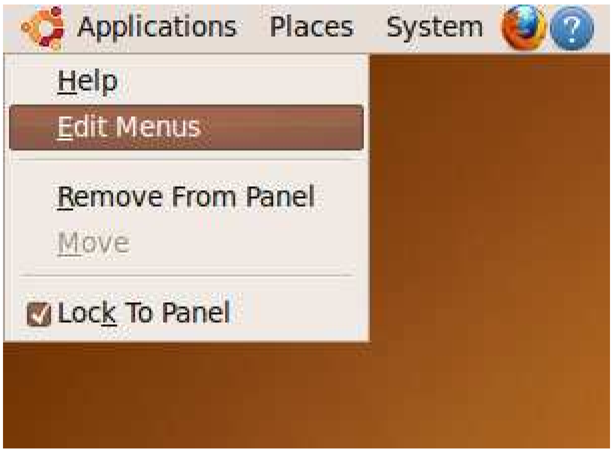
- Deep inspection of hundreds of protocols, with more being added all the time
- Live capture and offline analysis
- Standard three-pane packet browser
- Multi-platform: Runs on Windows, Linux, OS X, Solaris, FreeBSD, NetBSD, and many others
- Captured network data can be browsed via a GUI, or via the TTY-mode TShark utility
- The most powerful display filters in the industry
- Rich VoIP analysis

- Read/write many different capture file formats: tcpdump (libpcap), Pcap NG, Catapult DCT2000, Cisco Secure IDS iplog, Microsoft Network Monitor, Network General Sniffer® (compressed and uncompressed), Sniffer® Pro, and NetXray®, Network Instruments Observer, NetScreen snoop, Novell LANalyzer, RADCOM WAN/LAN Analyzer, Shomiti/Finisar Surveyor, Tektronix K12xx, Visual Networks Visual UpTime, WildPackets EtherPeek/TokenPeek/AiroPeek, and many others
- Capture files compressed with gzip can be decompressed on the fly
- Live data can be read from Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI, and others (depending on your platform)
- Decryption support for many protocols, including IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, and WPA/WPA2
- Coloring rules can be applied to the packet list for quick, intuitive analysis
- Output can be exported to XML, PostScript®, CSV, or plain text

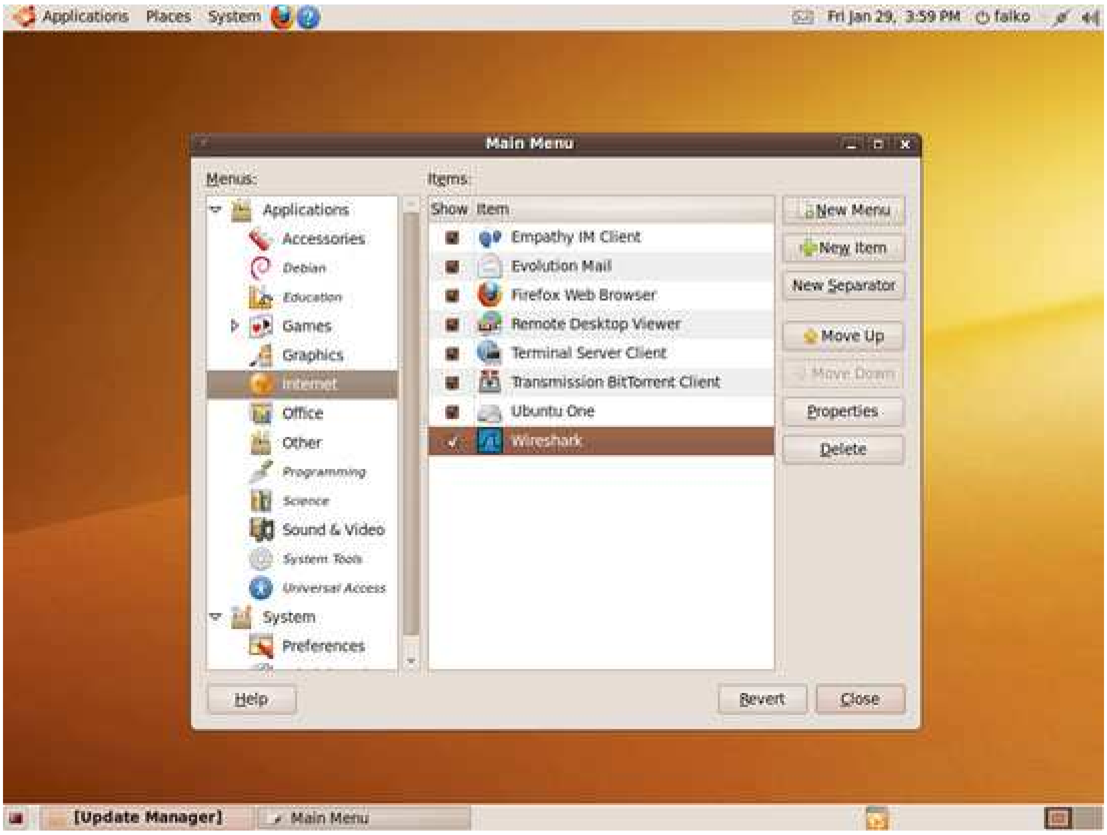
In our project, we will use Wirshark to capture the traffic and see the rejected and delayed requests.

Using Wireshark:

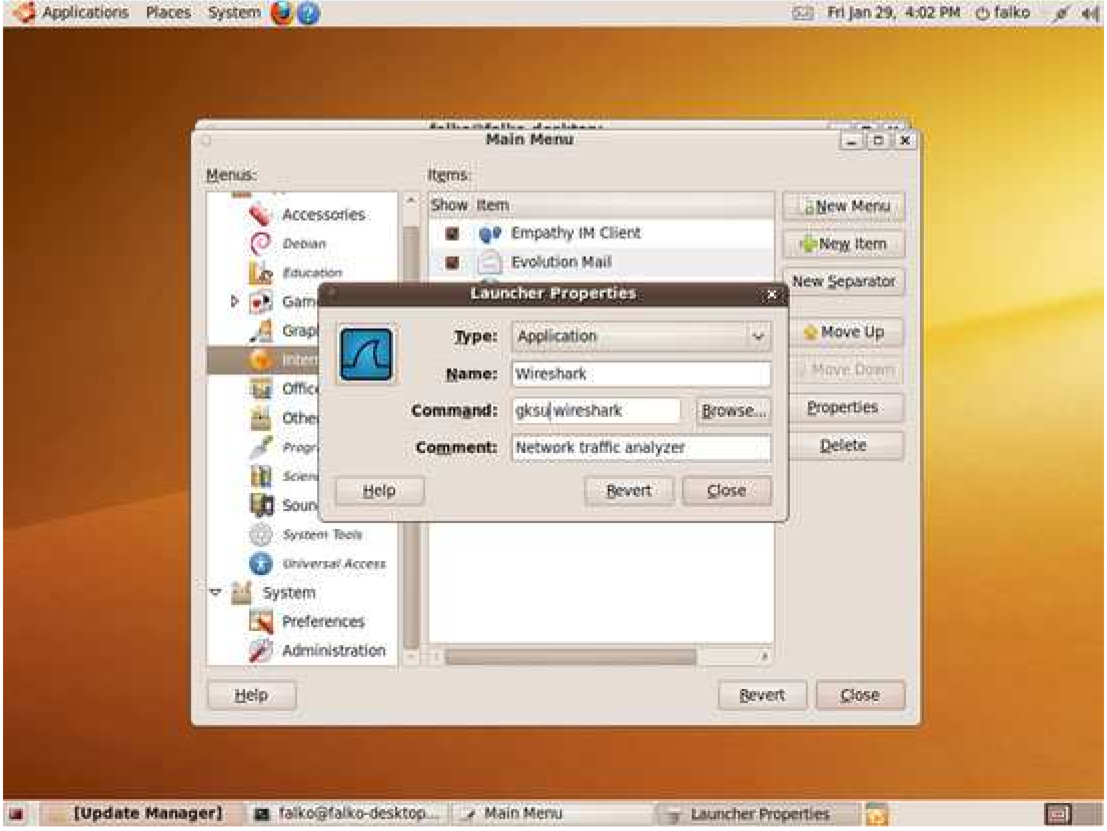
We must run Wireshark with root privileges so that it has enough permission to monitor the network interfaces. Because the default Wireshark launcher starts Wireshark with normal user privileges, we have to modify the launcher now. Right-click Applications and select Edit Menus:



In the Menu Editor, go to Internet > Wireshark and click the Properties button:



In the Launcher Properties window, add gksu in the Command field so that the command readsgksu wireshark. Click Close afterwards and leave the Menu Editor:



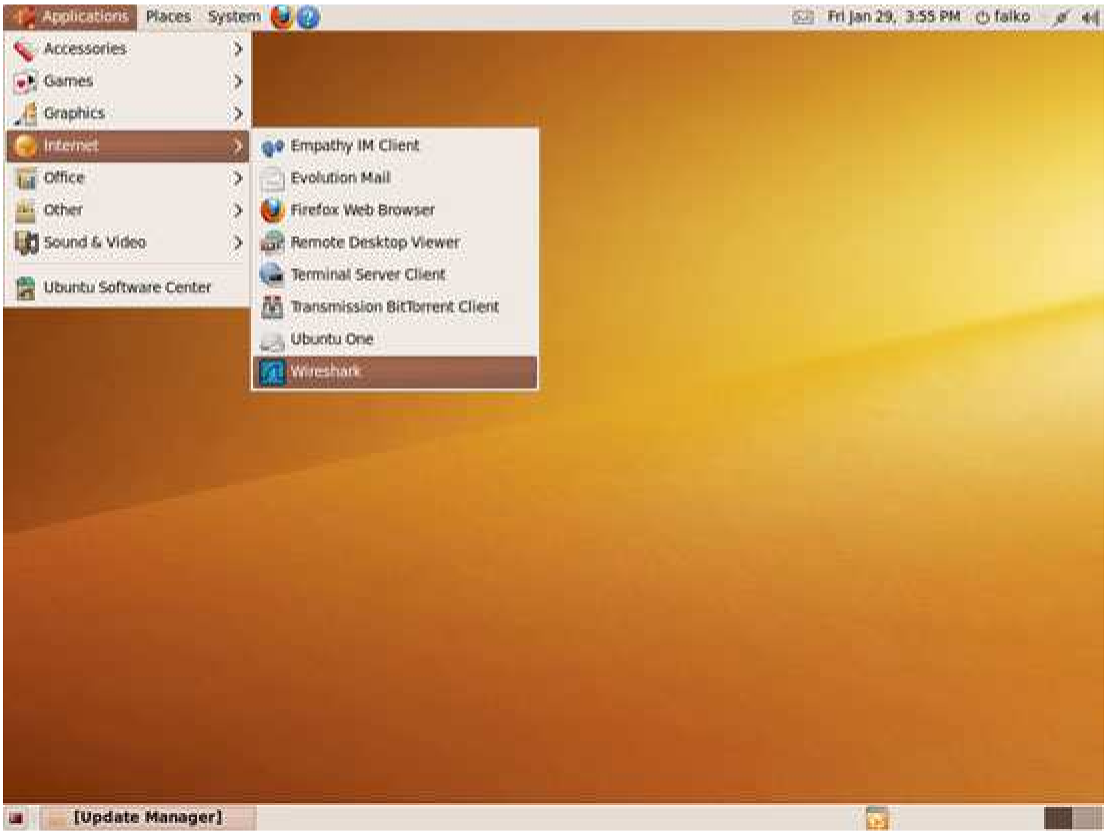
Type: Application

Name: Wireshark

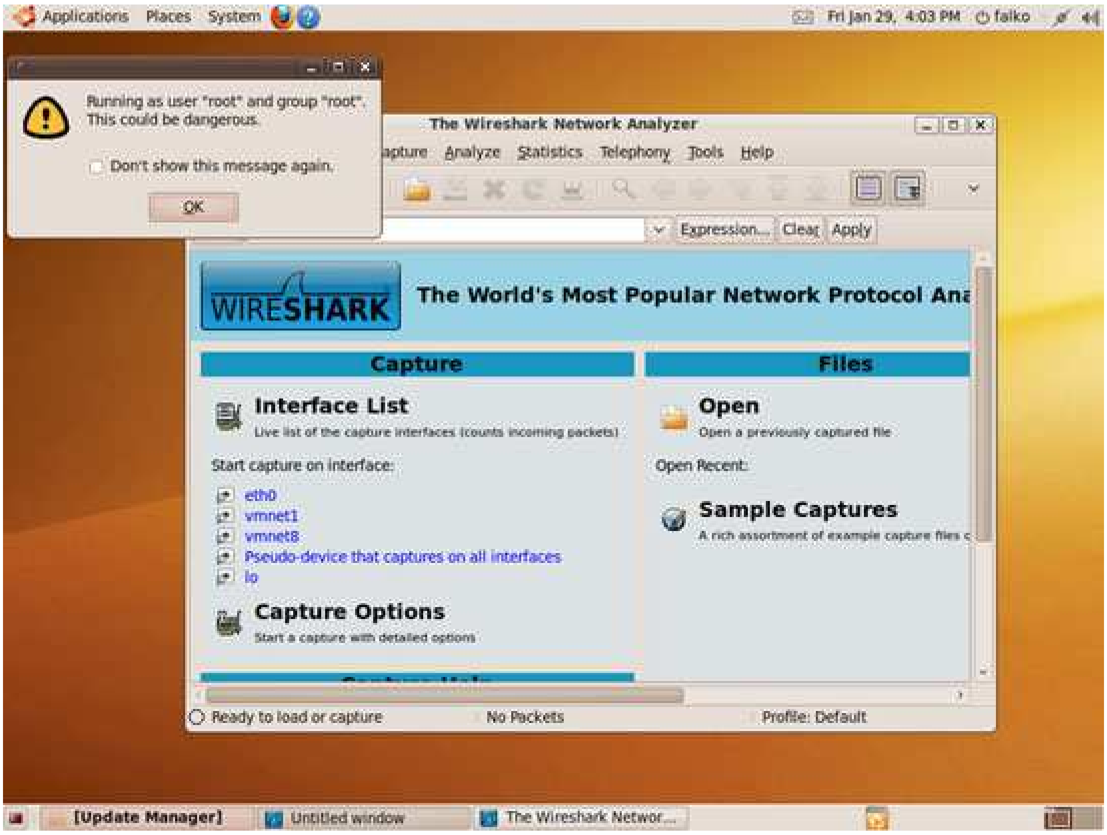
Command: gksu wireshark

Comment: Network traffic analyzer

Open the Wireshark application (Applications > Internet > Wireshark):



Because we are running Wireshark with root privileges, you will see the following warning (Running as user "root" and group "root". This could be dangerous.). Click OK:

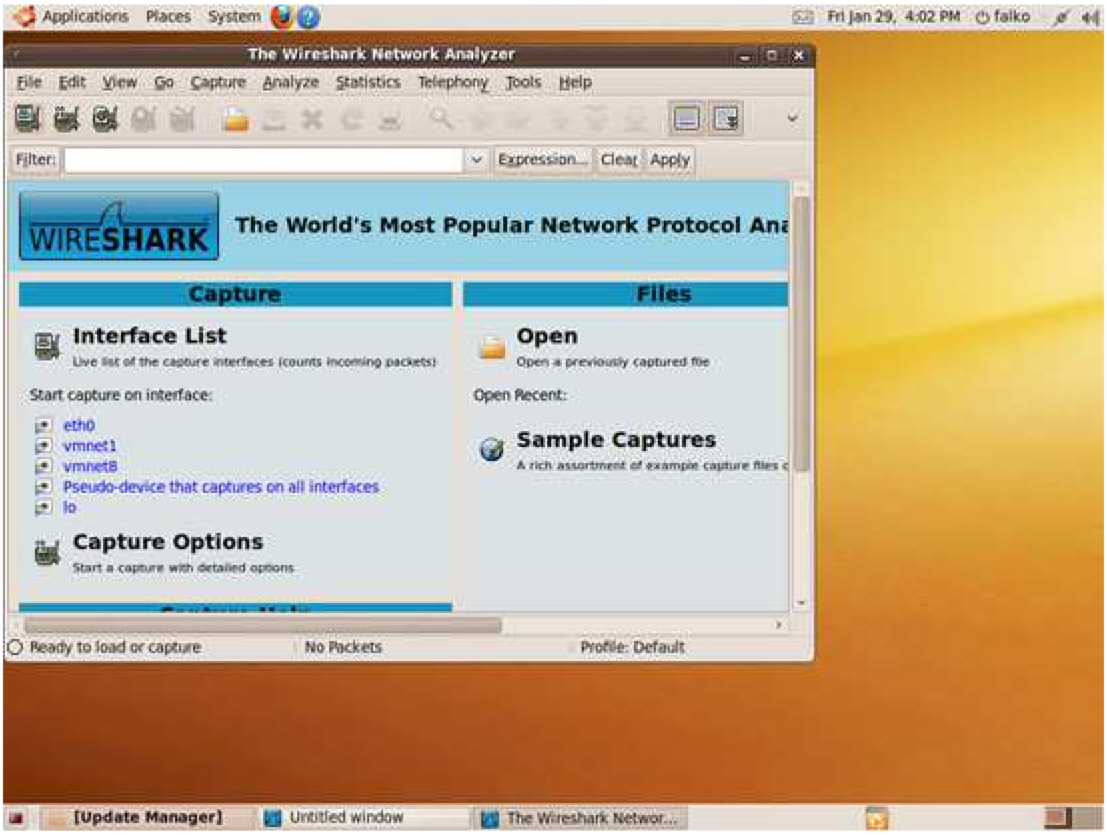


Running as user "root" and group "root". This could be dangerous.

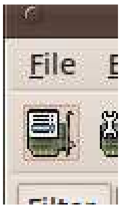
□ Don't show this message again.

Applications
Places
System
Fri Jan 29, 4:03 PM
falko
OK
The Wireshark Network Analyzer
apture
Analyze
Statistics
Telephony
Tools
Help
Expression...
Clear
Apply
WIRESHARK
The World's Most Popular Network Protocol Ana
Capture
Files
Interface List
Live list of the capture interfaces (counts incoming packets)
Start capture on interface:
eth0
vmnet1
vmnet8
Pseudo-device that captures on all interfaces
lo
Capture Options
Start a capture with detailed options
Open
Open a previously captured file
Open Recent:
Sample Captures
A rich assortment of example capture files c
Ready to load or capture
No Packets
Profile: Default
[Update Manager]
Untitled window
The Wireshark Networ...

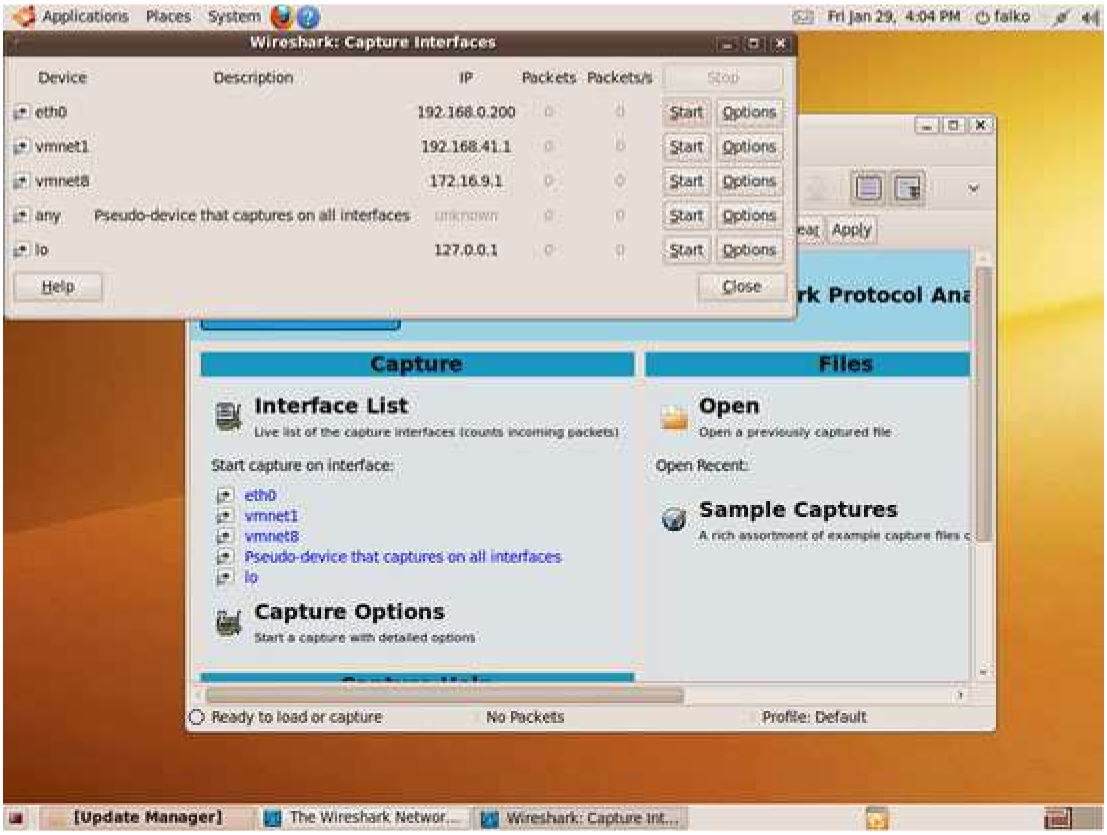
This is how Wireshark looks when you first start it:


Applications Places System
Fri Jan 29, 4:02 PM falko
The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter: Expression... Clear Apply
WIRESHARK
The World's Most Popular Network Protocol Ana
Capture
Files
Interface List
Live list of the capture interfaces (counts incoming packets)
Start capture on interface:
eth0
vmnet1
vmnet8
Pseudo-device that captures on all interfaces
lo
Capture Options
Start a capture with detailed options
Open
Open a previously captured file
Open Recent:
Sample Captures
A rich assortment of example capture files o
Ready to load or capture
No Packets
Profile: Default
[Update Manager]
Untitled window
The Wireshark Networ...

Click the List the available capture interfaces... button:

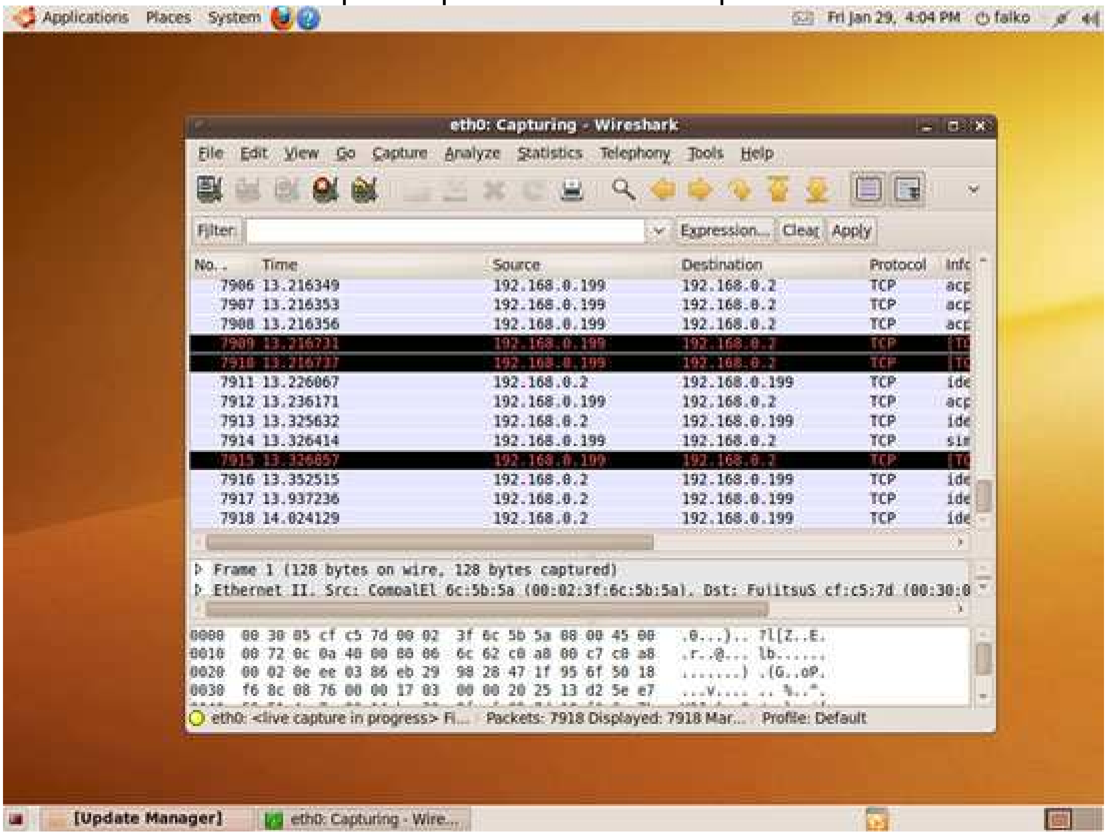


A new window opens with a list of available network interfaces on your system. Normally you want to capture the traffic on your primary network device (eth0 in this example), so you click the Start button in the eth0 row to start an analysis of the traffic on that interface:


Applications Places System
Fri Jan 29, 4:04 PM falko
Wireshark: Capture Interfaces
Stop
Help
Close
rk Protocol Ana
Capture
Files
Interface List
Live list of the capture interfaces (counts incoming packets)
Start capture on interface:
eth0
vmnet1
vmnet8
Pseudo-device that captures on all interfaces
lo
Capture Options
Start a capture with detailed options
Open
Open a previously captured file
Open Recent:
Sample Captures
A rich assortment of example capture files c
Ready to load or capture
No Packets
Profile: Default
[Update Manager]
The Wireshark Networ...
Wireshark: Capture Int...

Device	Description	IP	Packets	Packets/s		
eth0		192.168.0.200	0	0	Start	Options
vmnet1		192.168.41.1	0	0	Start	Options
vmnet8		172.16.9.1	0	0	Start	Options
any	Pseudo-device that captures on all interfaces	unknown	0	0	Start	Options
lo		127.0.0.1	0	0	Start	Options

You can now see the captured packets for various protocols in the main window.


Applications Places System
Fri Jan 29, 4:04 PM falko
eth0: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression...
Clear
Apply
[Update Manager]
eth0: Capturing - Wire...

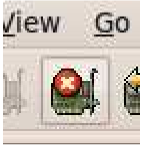
No..	Time	Source	Destination	Protocol	Info
7906	13.216349	192.168.0.199	192.168.0.2	TCP	acp
7907	13.216353	192.168.0.199	192.168.0.2	TCP	acp
7908	13.216356	192.168.0.199	192.168.0.2	TCP	acp
7909	13.216731	192.168.0.199	192.168.0.2	TCP	[TC
7910	13.216737	192.168.0.199	192.168.0.2	TCP	[TC
7911	13.226067	192.168.0.2	192.168.0.199	TCP	ide
7912	13.236171	192.168.0.199	192.168.0.2	TCP	acp
7913	13.325632	192.168.0.2	192.168.0.199	TCP	ide
7914	13.326414	192.168.0.199	192.168.0.2	TCP	sim
7915	13.326857	192.168.0.199	192.168.0.2	TCP	[TC
7916	13.352515	192.168.0.2	192.168.0.199	TCP	ide
7917	13.937236	192.168.0.2	192.168.0.199	TCP	ide
7918	14.024129	192.168.0.2	192.168.0.199	TCP	ide

```
▷ Frame 1 (128 bytes on wire, 128 bytes captured)
▷ Ethernet II, Src: CompalEl 6c:5b:5a (00:02:3f:6c:5b:5a), Dst: FujitsuS cf:c5:7d (00:30:0
```

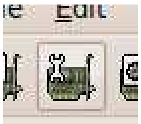
```
0000  00 30 05 cf c5 7d 00 02  3f 6c 5b 5a 08 00 45 00   .0...}.. ?l[Z..E.
0010  00 72 0c 0a 40 00 80 06  6c 62 c0 a8 00 c7 c0 a8   .r..@... lb......
0020  00 02 0e ee 03 86 eb 29  98 28 47 1f 95 6f 50 18   .......) .(G..oP.
0030  f6 8c 08 76 00 00 17 03  00 00 20 25 13 d2 5e e7   ...v.... .. %..^.
```

eth0: <live capture in progress> Fi... Packets: 7918 Displayed: 7918 Mar... Profile: Default

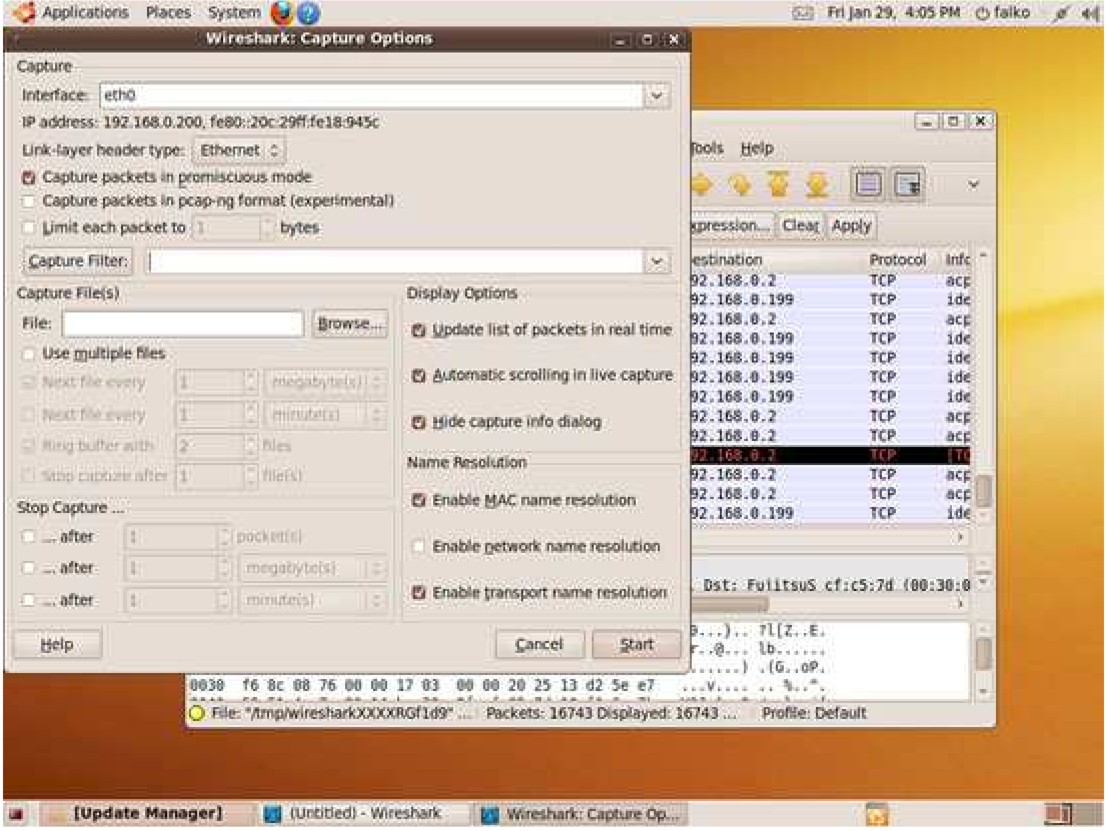
The capture goes on until you click the Stop button:



You can now browse the results, apply filters, find problems, etc.
To fine-tune future captures, you can click the Show the capture options... button:



A new window opens where you can set parameters for the next capture.
Click Start afterwards to start the capture:



Wireshark: Capture Options

Capture
Interface: eth0
IP address: 192.168.0.200, fe80::20c:29ff:fe18:945c
Link-layer header type: Ethernet
☑ Capture packets in promiscuous mode
□ Capture packets in pcap-ng format (experimental)
□ Limit each packet to 1 bytes
Capture Filter:

Capture File(s)
File: Browse...
□ Use multiple files
□ Next file every 1 megabyte(s)
□ Next file every 1 minute(s)
□ Ring buffer with 2 files
□ Stop capture after 1 file(s)

Stop Capture ...
□ ... after 1 packet(s)
□ ... after 1 megabyte(s)
□ ... after 1 minute(s)

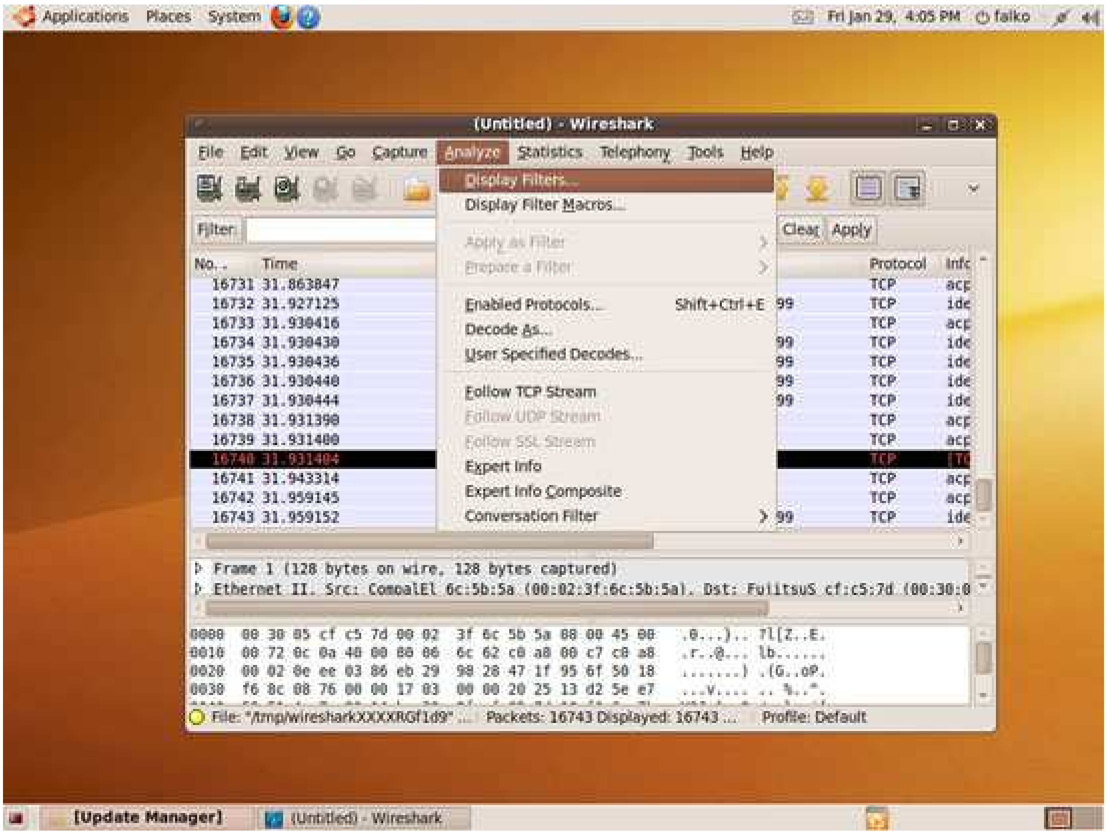
Display Options
☑ Update list of packets in real time
☑ Automatic scrolling in live capture
☑ Hide capture info dialog

Name Resolution
☑ Enable MAC name resolution
□ Enable network name resolution
☑ Enable transport name resolution

Help Cancel Start

File: "/tmp/wiresharkXXXXRGf1d9" ... Packets: 16743 Displayed: 16743 ... Profile: Default

The result of a capture lists all found protocols by default. If you'd like to concentrate on a certain protocol (for example), you can apply a filter to the result. Go to Analyze > Display Filters...:



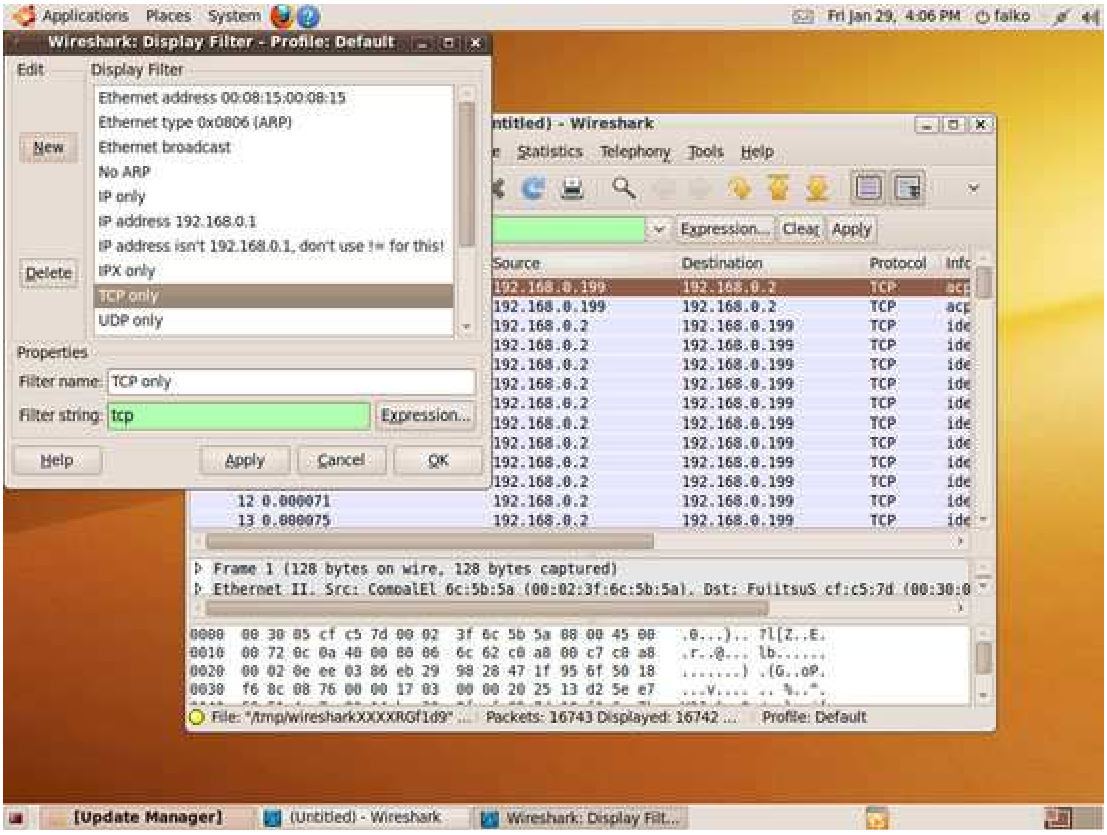
```
No. .  Time                                  Protocol  Info
 16731 31.863847                             TCP       acp
 16732 31.927125                         99  TCP       ide
 16733 31.930416                             TCP       acp
 16734 31.930430                         99  TCP       ide
 16735 31.930436                         99  TCP       ide
 16736 31.930440                         99  TCP       ide
 16737 31.930444                         99  TCP       ide
 16738 31.931390                             TCP       acp
 16739 31.931400                             TCP       acp
 16740 31.931404                             TCP       [TC
 16741 31.943314                             TCP       acp
 16742 31.959145                             TCP       acp
 16743 31.959152                         99  TCP       ide
```

```
▷ Frame 1 (128 bytes on wire, 128 bytes captured)
▷ Ethernet II, Src: CompalEl_6c:5b:5a (00:02:3f:6c:5b:5a), Dst: FujitsuS_cf:c5:7d (00:30:0
```

```
0000  00 30 05 cf c5 7d 00 02  3f 6c 5b 5a 08 00 45 00   .0...}.. ?l[Z..E.
0010  00 72 0c 0a 40 00 80 06  6c 62 c0 a8 00 c7 c0 a8   .r..@... lb......
0020  00 02 0e ee 03 86 eb 29  98 28 47 1f 95 6f 50 18   .......) .(G..oP.
0030  f6 8c 08 76 00 00 17 03  00 00 20 25 13 d2 5e e7   ...v.... .. %..^.
```

File: "/tmp/wiresharkXXXXRGf1d9" ... Packets: 16743 Displayed: 16743 ... Profile: Default

A new window opens where you can select your desired protocol (TCP for example). Click OKafterwards:


Applications Places System
Fri Jan 29, 4:06 PM falko
Wireshark: Display Filter - Profile: Default
Edit
Display Filter
New
Delete
Properties
Help Apply Cancel OK
(Untitled) - Wireshark
Statistics Telephony Tools Help
Expression... Clear Apply
[Update Manager]
(Untitled) - Wireshark
Wireshark: Display Filt...

Ethernet address 00:08:15:00:08:15
Ethernet type 0x0806 (ARP)
Ethernet broadcast
No ARP
IP only
IP address 192.168.0.1
IP address isn't 192.168.0.1, don't use != for this!
IPX only
TCP only
UDP only

Filter name: TCP only
Filter string: tcp

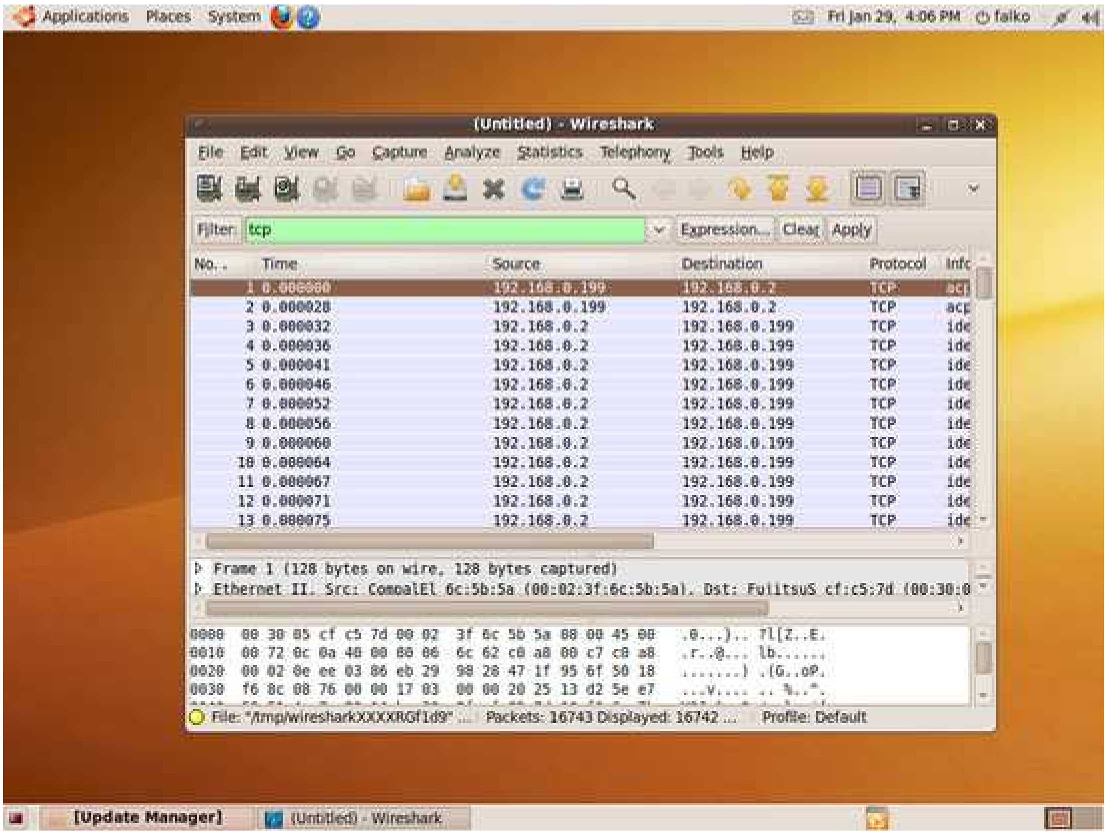
```
      Source          Destination     Protocol  Info
      192.168.0.199   192.168.0.2     TCP       acp
      192.168.0.199   192.168.0.2     TCP       acp
      192.168.0.2     192.168.0.199   TCP       ide
      192.168.0.2     192.168.0.199   TCP       ide
      192.168.0.2     192.168.0.199   TCP       ide
      192.168.0.2     192.168.0.199   TCP       ide
      192.168.0.2     192.168.0.199   TCP       ide
      192.168.0.2     192.168.0.199   TCP       ide
      192.168.0.2     192.168.0.199   TCP       ide
      192.168.0.2     192.168.0.199   TCP       ide
      192.168.0.2     192.168.0.199   TCP       ide
   12 0.000071        192.168.0.2     192.168.0.199   TCP       ide
   13 0.000075        192.168.0.2     192.168.0.199   TCP       ide
```

```
▷ Frame 1 (128 bytes on wire, 128 bytes captured)
▷ Ethernet II, Src: CompalEl_6c:5b:5a (00:02:3f:6c:5b:5a), Dst: FujitsuS_cf:c5:7d (00:30:0
```

```
0000  00 30 05 cf c5 7d 00 02  3f 6c 5b 5a 08 00 45 00   .0...}.. ?l[Z..E.
0010  00 72 0c 0a 40 00 80 06  6c 62 c0 a8 00 c7 c0 a8   .r..@... lb......
0020  00 02 0e ee 03 86 eb 29  98 28 47 1f 95 6f 50 18   .......) .(G..oP.
0030  f6 8c 08 76 00 00 17 03  00 00 20 25 13 d2 5e e7   ...v.... .. %..^.
```

File: "/tmp/wiresharkXXXXRGf1d9" ... Packets: 16743 Displayed: 16742 ... Profile: Default

In the result window, you should now find TCP traffic only - all other protocols have been filtered out:



Filter: tcp

No. .	Time	Source	Destination	Protocol	Infc
1	0.000000	192.168.0.199	192.168.0.2	TCP	acp
2	0.000028	192.168.0.199	192.168.0.2	TCP	acp
3	0.000032	192.168.0.2	192.168.0.199	TCP	ide
4	0.000036	192.168.0.2	192.168.0.199	TCP	ide
5	0.000041	192.168.0.2	192.168.0.199	TCP	ide
6	0.000046	192.168.0.2	192.168.0.199	TCP	ide
7	0.000052	192.168.0.2	192.168.0.199	TCP	ide
8	0.000056	192.168.0.2	192.168.0.199	TCP	ide
9	0.000060	192.168.0.2	192.168.0.199	TCP	ide
10	0.000064	192.168.0.2	192.168.0.199	TCP	ide
11	0.000067	192.168.0.2	192.168.0.199	TCP	ide
12	0.000071	192.168.0.2	192.168.0.199	TCP	ide
13	0.000075	192.168.0.2	192.168.0.199	TCP	ide

```
▷ Frame 1 (128 bytes on wire, 128 bytes captured)
▷ Ethernet II, Src: CompalEl 6c:5b:5a (00:02:3f:6c:5b:5a), Dst: FujitsuS cf:c5:7d (00:30:0
```

```
0000  00 30 05 cf c5 7d 00 02  3f 6c 5b 5a 08 00 45 00   .0...}.. ?l[Z..E.
0010  00 72 0c 0a 40 00 80 06  6c 62 c0 a8 00 c7 c0 a8   .r..@... lb......
0020  00 02 0e ee 03 86 eb 29  98 28 47 1f 95 6f 50 18   .......) .(G..oP.
0030  f6 8c 08 76 00 00 17 03  00 00 20 25 13 d2 5e e7   ...v.... .. %..^.
```

File: "/tmp/wiresharkXXXXRGf1d9" ... Packets: 16743 Displayed: 16742 ... Profile: Default

How does our solution works?

To add constant delay to every packet going out through a specific interface, use the following command:

```
# tc qdisc add dev eth1 root netem delay 80ms
```

Now a ping test to this host should show an increase of 80ms in the delay to replies.

To add random variance, use the command below:

```
# tc qdisc change dev eth1 root netem delay 80ms 10ms
```

We can also add variable delay (jitter)/Random Variance too. Most wide-area networks like the Internet have some jitter associated with them. The following command will add +/- 10 ms of jitter to the 80 ms of delay.

```
# tc qdisc add dev eth1 root netem delay 80ms 10ms
```

To see what queueing discipline (qdisc) has been applied to an interface, use:

```
# tc qdisc show dev eth1
```

To turn off/delete the qdisc from a specific interface (in this case, eth1), execute the command given below:

```
# tc qdisc del dev eth1 root
```

Typically, the delay in a network is not uniform. It is more common to use something like a normal distribution to describe the variation in delay. NetEM can accept a non-uniform distribution:

```
# tc qdisc change dev eth1 root netem delay 100ms 20ms distribution normal
```

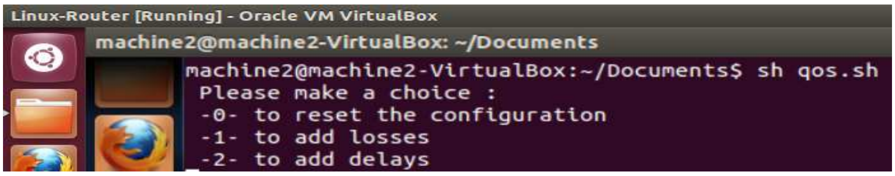
Packet loss can be replicated:

```
# tc qdisc change dev eth1 root netem loss 0.1%
```

Packet duplication/corruption can also be configured:

```
# tc qdisc change dev eth1 root netem duplicate/corrupt 1%
```

I have written a shell script that gives me the choice of adding **losses**, **delays** or **reset the configuration.**


Linux-Router [Running] - Oracle VM VirtualBox
machine2@machine2-VirtualBox: ~/Documents

```
machine2@machine2-VirtualBox:~/Documents$ sh qos.sh
 Please make a choice :
 -0- to reset the configuration
 -1- to add losses
 -2- to add delays
```

The values are already specified (30% losses and 2s delay) and could be changed directly in the file located in Documents folder on machine “Linux router”

Tests

Ping:

- **Losses**

	NETEM		first			Second			Third		
Protocol	% Losses	Transmitted	Recieved	Real loss	Errors	Recieved	Real loss	Errors	Recieved	Real loss	Errors
ICMP	10%	20	19	5%	None	18	10%	None	17	15%	None
ICMP	20%	20	15	25%	None	16	20%	None	15	25%	None
ICMP	30%	20	11	45%	None	15	25%	None	13	35%	None
ICMP	40%	20	12	40%	None	16	20%	None	8	60%	None
ICMP	50%	20	9	55%	None	12	40%	None	11	45%	None
ICMP	60%	20	6	70%	None	9	55%	None	8	60%	None
ICMP	70%	20	8	60%	None	6	70%	None	1	95%	None
ICMP	80%	20	4	80%	None	4	80%	None	2	90%	None
ICMP	90%	20	0	100%	None	0	100%	12	0	100%	3
ICMP	100%	20	0	100%	12	0	100%	20	0	100%	20

This table shows that we can have the losses already but not exactly the rate requested. This is normal because netem can’t fix the number of losses per Ping, so it applies this on all the requests passing through the interface.

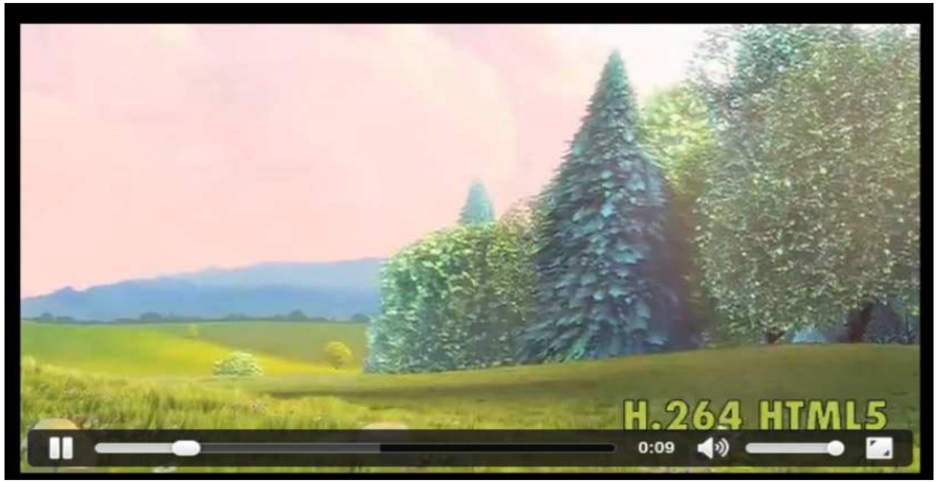
- **Delays**

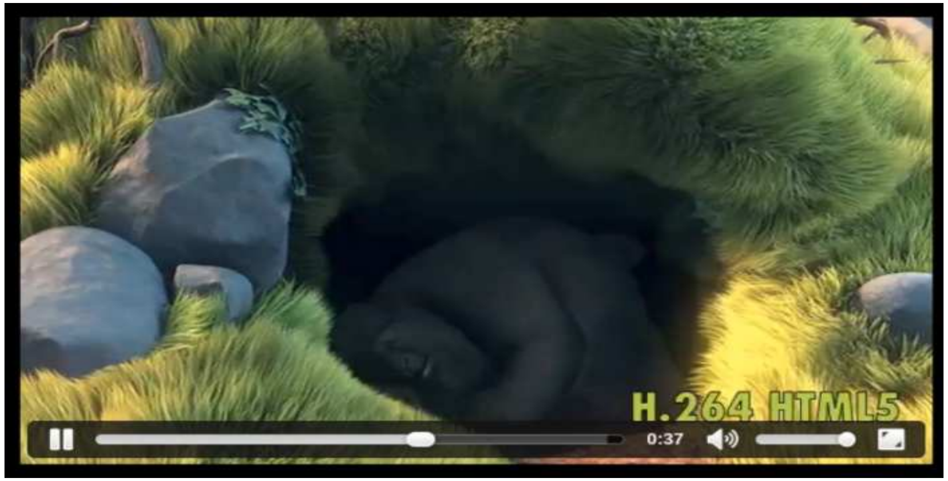
Protocol	NETEM		first			Second			Third			
ICMP	Delays	Transmitted	Recieved	Lost	Errors	Recieved	Lost	Errors	Recieved	Lost	Errors	
ICMP	1s	20	20	0%	None	20	0%	None	20	0%	None	pipe3
ICMP	2s	20	20	0%	None	20	0%	None	20	0%	None	pipe2
ICMP	3s	20	20	0%	None	20	0%	None	20	0%	None	pipe3
ICMP	4s	20	17	15%	None	20	0%	None	16	20%	None	pipe5
ICMP	5s	20	18	10%	None	20	0%	None	20	0%	None	pipe7
ICMP	6s	20	19	5%	None	20	0%	None	20	0%	None	pipe9
ICMP	7s	20	17	15%	3	17	15%	None	17	15%	3	pipe8
ICMP	8s	20	17	15%	None	15	25%	None	15	25%	None	pipe8
ICMP	9s	20	17	15%	3	20	0%	None	20	0%	None	pipe9
ICMP	10s	20	14	30%	6	18	10%	None	13	35%	None	pipe10

Delays are better controlled than losses because they are directly injected to the requests.

Video test

- **Before**



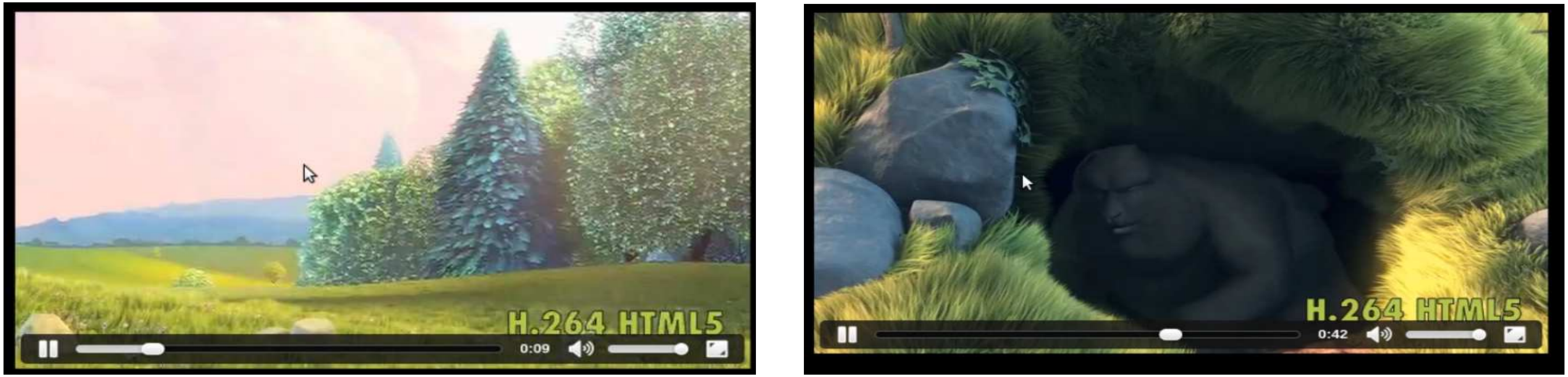


The Client1 (192.168.1.20) is requesting the video from the Apache Server in the Client2 (192.168.0.20).

The video start automatically when the page is requested.

The video is downloaded normally without any interruption and the user can watch it like it was located in the localhost (LAN simulation).

- **<u>After</u>**



The Client1 (192.168.1.20) is requesting the video from the Apache Server in the Client2 (192.168.0.20).

<u>LOSSES</u>

The video start automatically when the page is requested.

The video is not played normally, we can remark several interruptions.

The video is downloaded slowly.

<u>DELAYS</u>

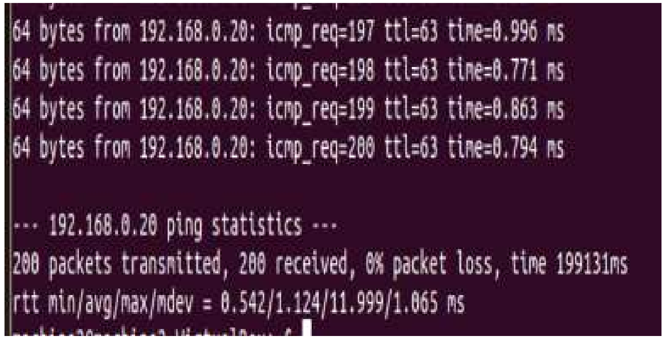
The video doesn’t start automatically when the page is requested (2000ms delay).

The video isn’t played normally, we can remark several interruptions.

The video is downloaded slowly with regular delays.

Wireshark tests:

- **<u>First scenario: 200 ping without any troubles</u>**

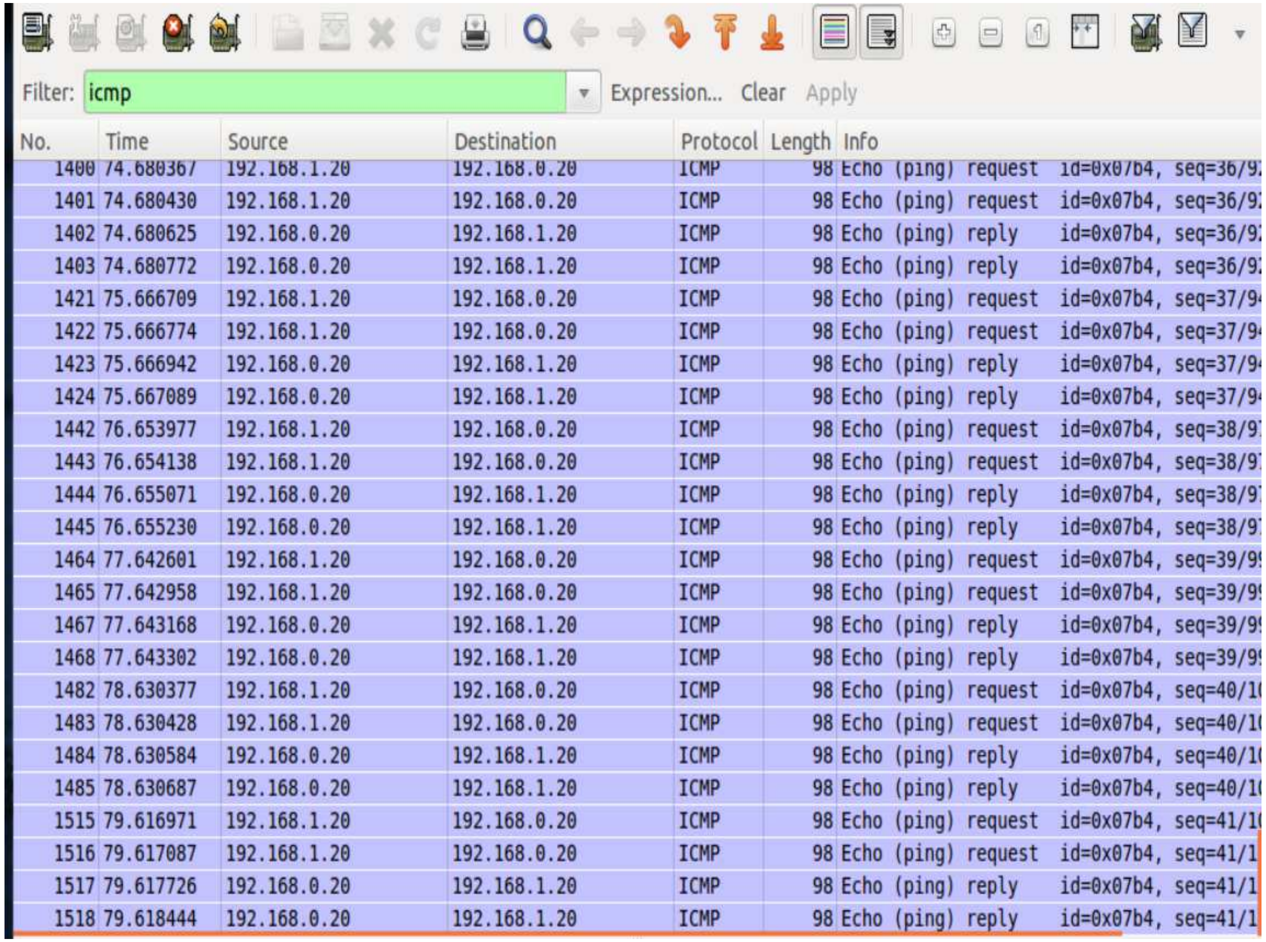


```
64 bytes from 192.168.0.20: icmp_req=197 ttl=63 time=0.996 ms
64 bytes from 192.168.0.20: icmp_req=198 ttl=63 time=0.771 ms
64 bytes from 192.168.0.20: icmp_req=199 ttl=63 time=0.863 ms
64 bytes from 192.168.0.20: icmp_req=200 ttl=63 time=0.794 ms

--- 192.168.0.20 ping statistics ---
200 packets transmitted, 200 received, 0% packet loss, time 199131ms
rtt min/avg/max/mdev = 0.542/1.124/11.999/1.065 ms
```

<u>Result</u>

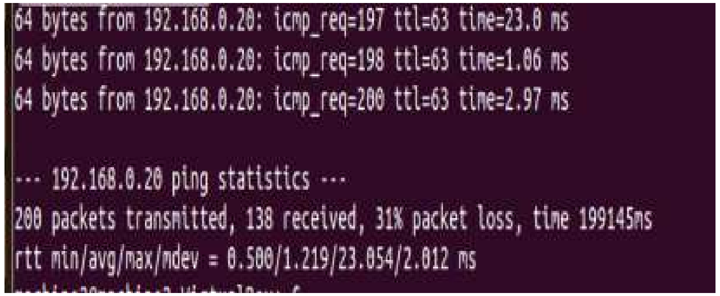
- ➢ Packets are sent normally
- ➢ No losses
- ➢ Normal delays
- ➢ 2 requests + 2 replays (2 hops)



Filter: icmp

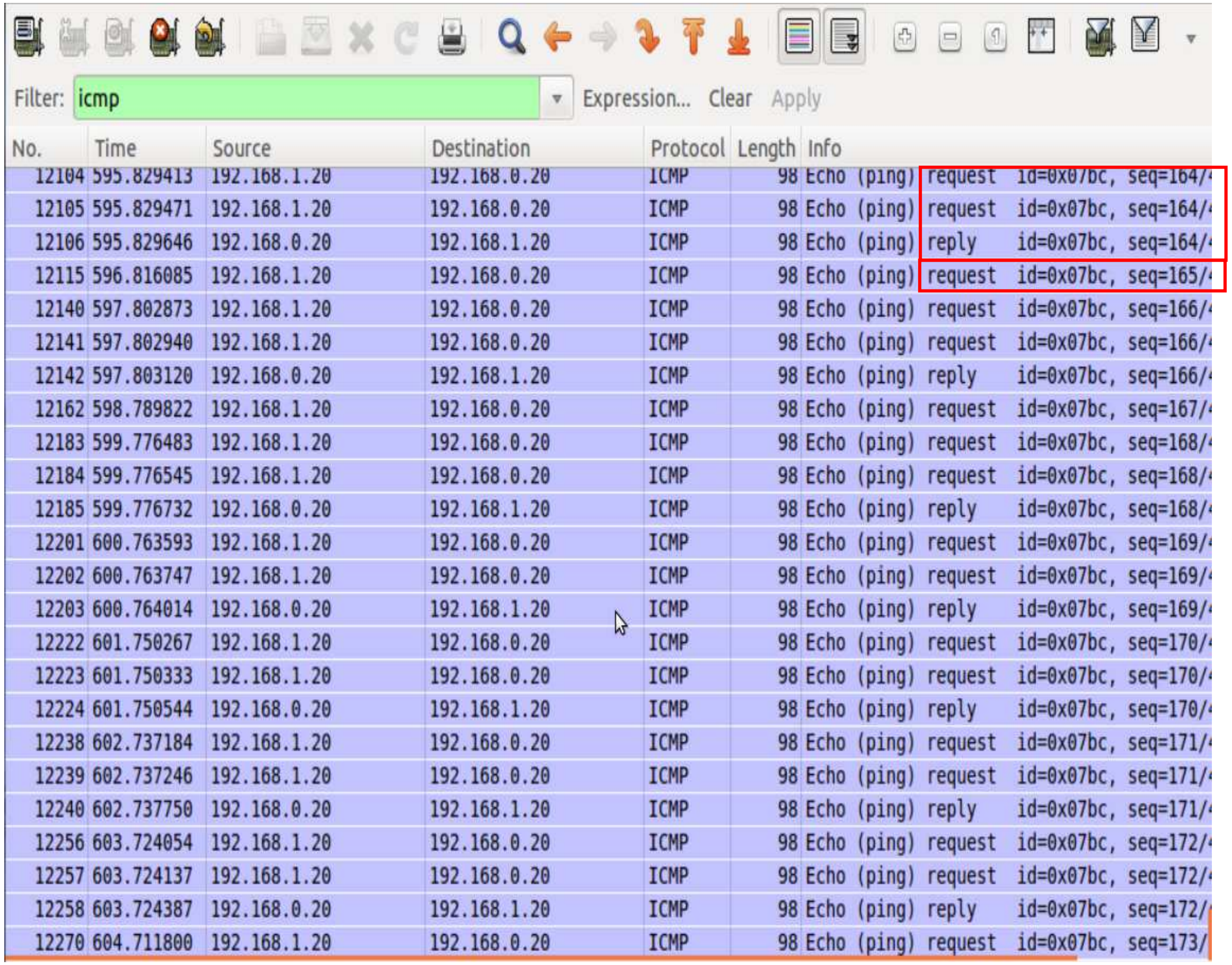
No.	Time	Source	Destination	Protocol	Length	Info
1400	74.680367	192.168.1.20	192.168.0.20	ICMP	98	Echo (ping) request id=0x07b4, seq=36/9
1401	74.680430	192.168.1.20	192.168.0.20	ICMP	98	Echo (ping) request id=0x07b4, seq=36/9
1402	74.680625	192.168.0.20	192.168.1.20	ICMP	98	Echo (ping) reply id=0x07b4, seq=36/9
1403	74.680772	192.168.0.20	192.168.1.20	ICMP	98	Echo (ping) reply id=0x07b4, seq=36/9
1421	75.666709	192.168.1.20	192.168.0.20	ICMP	98	Echo (ping) request id=0x07b4, seq=37/9
1422	75.666774	192.168.1.20	192.168.0.20	ICMP	98	Echo (ping) request id=0x07b4, seq=37/9
1423	75.666942	192.168.0.20	192.168.1.20	ICMP	98	Echo (ping) reply id=0x07b4, seq=37/9
1424	75.667089	192.168.0.20	192.168.1.20	ICMP	98	Echo (ping) reply id=0x07b4, seq=37/9
1442	76.653977	192.168.1.20	192.168.0.20	ICMP	98	Echo (ping) request id=0x07b4, seq=38/9
1443	76.654138	192.168.1.20	192.168.0.20	ICMP	98	Echo (ping) request id=0x07b4, seq=38/9
1444	76.655071	192.168.0.20	192.168.1.20	ICMP	98	Echo (ping) reply id=0x07b4, seq=38/9
1445	76.655230	192.168.0.20	192.168.1.20	ICMP	98	Echo (ping) reply id=0x07b4, seq=38/9
1464	77.642601	192.168.1.20	192.168.0.20	ICMP	98	Echo (ping) request id=0x07b4, seq=39/9
1465	77.642958	192.168.1.20	192.168.0.20	ICMP	98	Echo (ping) request id=0x07b4, seq=39/9
1467	77.643168	192.168.0.20	192.168.1.20	ICMP	98	Echo (ping) reply id=0x07b4, seq=39/9
1468	77.643302	192.168.0.20	192.168.1.20	ICMP	98	Echo (ping) reply id=0x07b4, seq=39/9
1482	78.630377	192.168.1.20	192.168.0.20	ICMP	98	Echo (ping) request id=0x07b4, seq=40/1
1483	78.630428	192.168.1.20	192.168.0.20	ICMP	98	Echo (ping) request id=0x07b4, seq=40/1
1484	78.630584	192.168.0.20	192.168.1.20	ICMP	98	Echo (ping) reply id=0x07b4, seq=40/1
1485	78.630687	192.168.0.20	192.168.1.20	ICMP	98	Echo (ping) reply id=0x07b4, seq=40/1
1515	79.616971	192.168.1.20	192.168.0.20	ICMP	98	Echo (ping) request id=0x07b4, seq=41/1
1516	79.617087	192.168.1.20	192.168.0.20	ICMP	98	Echo (ping) request id=0x07b4, seq=41/1
1517	79.617726	192.168.0.20	192.168.1.20	ICMP	98	Echo (ping) reply id=0x07b4, seq=41/1
1518	79.618444	192.168.0.20	192.168.1.20	ICMP	98	Echo (ping) reply id=0x07b4, seq=41/1

- **2nd scenario: 30% Losses**



```
64 bytes from 192.168.0.20: icmp_req=197 ttl=63 time=23.0 ms
64 bytes from 192.168.0.20: icmp_req=198 ttl=63 time=1.06 ms
64 bytes from 192.168.0.20: icmp_req=200 ttl=63 time=2.97 ms

--- 192.168.0.20 ping statistics ---
200 packets transmitted, 138 received, 31% packet loss, time 199145ms
rtt min/avg/max/mdev = 0.500/1.219/23.054/2.012 ms
```

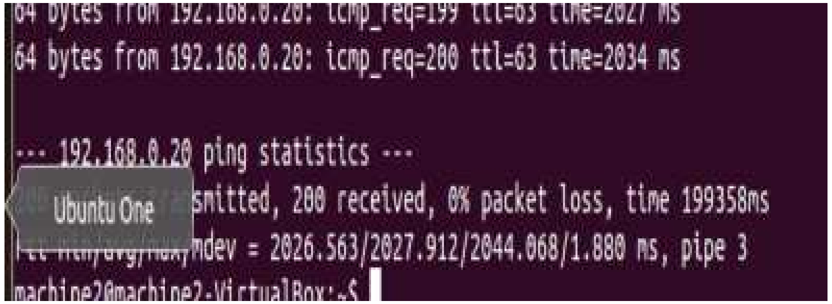


```
No.   Time       Source        Destination   Protocol Length Info
12104 595.829413 192.168.1.20  192.168.0.20  ICMP         98 Echo (ping) request  id=0x07bc, seq=164/
12105 595.829471 192.168.1.20  192.168.0.20  ICMP         98 Echo (ping) request  id=0x07bc, seq=164/
12106 595.829646 192.168.0.20  192.168.1.20  ICMP         98 Echo (ping) reply    id=0x07bc, seq=164/
12115 596.816085 192.168.1.20  192.168.0.20  ICMP         98 Echo (ping) request  id=0x07bc, seq=165/
12140 597.802873 192.168.1.20  192.168.0.20  ICMP         98 Echo (ping) request  id=0x07bc, seq=166/
12141 597.802940 192.168.1.20  192.168.0.20  ICMP         98 Echo (ping) request  id=0x07bc, seq=166/
12142 597.803120 192.168.0.20  192.168.1.20  ICMP         98 Echo (ping) reply    id=0x07bc, seq=166/
12162 598.789822 192.168.1.20  192.168.0.20  ICMP         98 Echo (ping) request  id=0x07bc, seq=167/
12183 599.776483 192.168.1.20  192.168.0.20  ICMP         98 Echo (ping) request  id=0x07bc, seq=168/
12184 599.776545 192.168.1.20  192.168.0.20  ICMP         98 Echo (ping) request  id=0x07bc, seq=168/
12185 599.776732 192.168.0.20  192.168.1.20  ICMP         98 Echo (ping) reply    id=0x07bc, seq=168/
12201 600.763593 192.168.1.20  192.168.0.20  ICMP         98 Echo (ping) request  id=0x07bc, seq=169/
12202 600.763747 192.168.1.20  192.168.0.20  ICMP         98 Echo (ping) request  id=0x07bc, seq=169/
12203 600.764014 192.168.0.20  192.168.1.20  ICMP         98 Echo (ping) reply    id=0x07bc, seq=169/
12222 601.750267 192.168.1.20  192.168.0.20  ICMP         98 Echo (ping) request  id=0x07bc, seq=170/
12223 601.750333 192.168.1.20  192.168.0.20  ICMP         98 Echo (ping) request  id=0x07bc, seq=170/
12224 601.750544 192.168.0.20  192.168.1.20  ICMP         98 Echo (ping) reply    id=0x07bc, seq=170/
12238 602.737184 192.168.1.20  192.168.0.20  ICMP         98 Echo (ping) request  id=0x07bc, seq=171/
12239 602.737246 192.168.1.20  192.168.0.20  ICMP         98 Echo (ping) request  id=0x07bc, seq=171/
12240 602.737750 192.168.0.20  192.168.1.20  ICMP         98 Echo (ping) reply    id=0x07bc, seq=171/
12256 603.724054 192.168.1.20  192.168.0.20  ICMP         98 Echo (ping) request  id=0x07bc, seq=172/
12257 603.724137 192.168.1.20  192.168.0.20  ICMP         98 Echo (ping) request  id=0x07bc, seq=172/
12258 603.724387 192.168.0.20  192.168.1.20  ICMP         98 Echo (ping) reply    id=0x07bc, seq=172/
12270 604.711800 192.168.1.20  192.168.0.20  ICMP         98 Echo (ping) request  id=0x07bc, seq=173/
```

Result

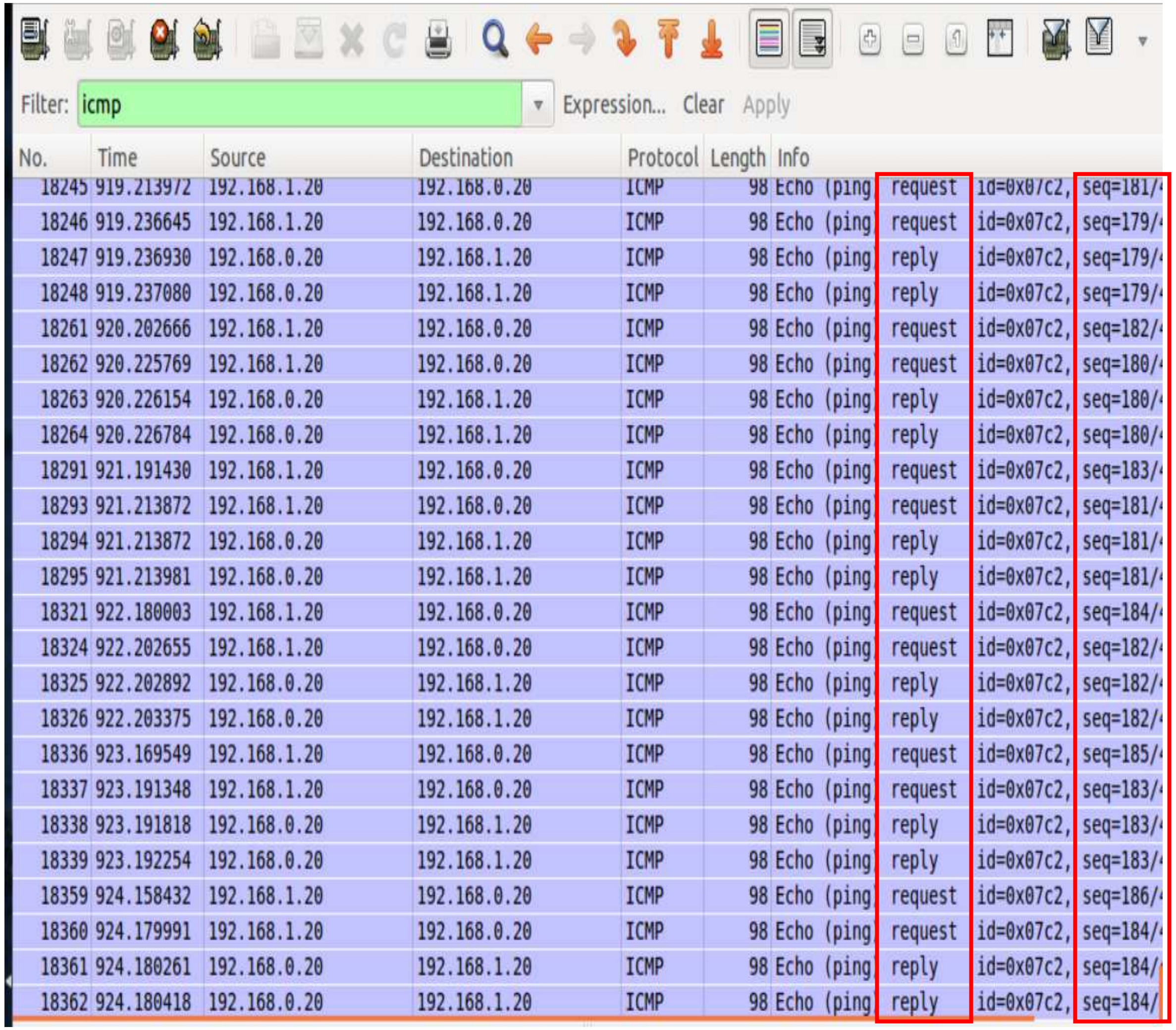
- Packets are not sent normally
- 31% of packets are lost
- Normal delays
- Problem: Some packets are destroyed so every combination is possible (1 request and no reply ex : seq=165 / 2 requests and no reply / 2 requests and 1 reply ex: seq=164 / 2 requests and 2 replies)

- **<u>3rd scenario: 2000ms Delay</u>**



```
64 bytes from 192.168.0.20: icmp_req=199 ttl=63 time=2027 ms
64 bytes from 192.168.0.20: icmp_req=200 ttl=63 time=2034 ms

--- 192.168.0.20 ping statistics ---
Ubuntu One  smitted, 200 received, 0% packet loss, time 199358ms
rtt min/avg/max/mdev = 2026.563/2027.912/2044.068/1.880 ms, pipe 3
machine2@machine2-VirtualBox:~$
```



```
No.   Time       Source        Destination   Protocol Length Info
18245 919.213972 192.168.1.20  192.168.0.20  ICMP     98 Echo (ping) request id=0x07c2, seq=181/
18246 919.236645 192.168.1.20  192.168.0.20  ICMP     98 Echo (ping) request id=0x07c2, seq=179/
18247 919.236930 192.168.0.20  192.168.1.20  ICMP     98 Echo (ping) reply   id=0x07c2, seq=179/
18248 919.237080 192.168.0.20  192.168.1.20  ICMP     98 Echo (ping) reply   id=0x07c2, seq=179/
18261 920.202666 192.168.1.20  192.168.0.20  ICMP     98 Echo (ping) request id=0x07c2, seq=182/
18262 920.225769 192.168.1.20  192.168.0.20  ICMP     98 Echo (ping) request id=0x07c2, seq=180/
18263 920.226154 192.168.0.20  192.168.1.20  ICMP     98 Echo (ping) reply   id=0x07c2, seq=180/
18264 920.226784 192.168.0.20  192.168.1.20  ICMP     98 Echo (ping) reply   id=0x07c2, seq=180/
18291 921.191430 192.168.1.20  192.168.0.20  ICMP     98 Echo (ping) request id=0x07c2, seq=183/
18293 921.213872 192.168.1.20  192.168.0.20  ICMP     98 Echo (ping) request id=0x07c2, seq=181/
18294 921.213872 192.168.0.20  192.168.1.20  ICMP     98 Echo (ping) reply   id=0x07c2, seq=181/
18295 921.213981 192.168.0.20  192.168.1.20  ICMP     98 Echo (ping) reply   id=0x07c2, seq=181/
18321 922.180003 192.168.1.20  192.168.0.20  ICMP     98 Echo (ping) request id=0x07c2, seq=184/
18324 922.202655 192.168.1.20  192.168.0.20  ICMP     98 Echo (ping) request id=0x07c2, seq=182/
18325 922.202892 192.168.0.20  192.168.1.20  ICMP     98 Echo (ping) reply   id=0x07c2, seq=182/
18326 922.203375 192.168.0.20  192.168.1.20  ICMP     98 Echo (ping) reply   id=0x07c2, seq=182/
18336 923.169549 192.168.1.20  192.168.0.20  ICMP     98 Echo (ping) request id=0x07c2, seq=185/
18337 923.191348 192.168.1.20  192.168.0.20  ICMP     98 Echo (ping) request id=0x07c2, seq=183/
18338 923.191818 192.168.0.20  192.168.1.20  ICMP     98 Echo (ping) reply   id=0x07c2, seq=183/
18339 923.192254 192.168.0.20  192.168.1.20  ICMP     98 Echo (ping) reply   id=0x07c2, seq=183/
18359 924.158432 192.168.1.20  192.168.0.20  ICMP     98 Echo (ping) request id=0x07c2, seq=186/
18360 924.179991 192.168.1.20  192.168.0.20  ICMP     98 Echo (ping) request id=0x07c2, seq=184/
18361 924.180261 192.168.0.20  192.168.1.20  ICMP     98 Echo (ping) reply   id=0x07c2, seq=184/
18362 924.180418 192.168.0.20  192.168.1.20  ICMP     98 Echo (ping) reply   id=0x07c2, seq=184/
```

<u>Result</u>

- Packets are not sent normally
- No losses remarked
- Every packet took almost 2000ms (delay is added to every packet)
- Problem: Replies are delivred with a lag, so we can remark using the capture of wireshark that the packets are not ordered because of the delay applied, but this solution guarantee a traffic without losses.

PROJECT MANAGEMENT

Risks management

Risk management seeks to identify potential and quantifiable losses inherent in our activity, the risk analysis has allowed us to move faster and avoid scenarios that can bring down our project to failure.

N°	Risk	Probability	severity	Impact	Nature of risk	Action
1	Absence or illness of human resources (Me)	1	2	Project on stand by	Delay	Provide an additional delay for this kind of situation
2	Failure of equipments (The server for example)	1	3	Loss of data/ virtual machines/ configuration and also time	Quality Delay Costs	Make regular saves
3	Problem with the virtual machines	2	1	Project on standby and loss of time	Delay	Save regularly a copy of the VM
4	Connectivity	1	1	Researches on standby	Quality Delay	Save the necessary documents
5	software's incompatibility with the OS	3	2	Can add delays to the project	Delay	Make sure that the software are compatible before starting the project
6	Solution impact on the OS functioning	2	3	Instability of the OS	Quality Delay Costs	Make sure that the software have no impact on the OS before starting the project
7	GThe server is	2	1	The solution can't	Cost	Request

	unable to support the solution			be tested	Quality Delay	another server with best configuration

Methodology

The method followed is cutting activities (WBS), which is to conduct a static division into activities with inputs and outcomes identified and a responsibility given to a named person. This approach was based on the mastery of the duration of each activity, the knowledge of the resources required and the cost of each activity. Here is the Gantt of my project, explaining the distribution of the activities per periods.

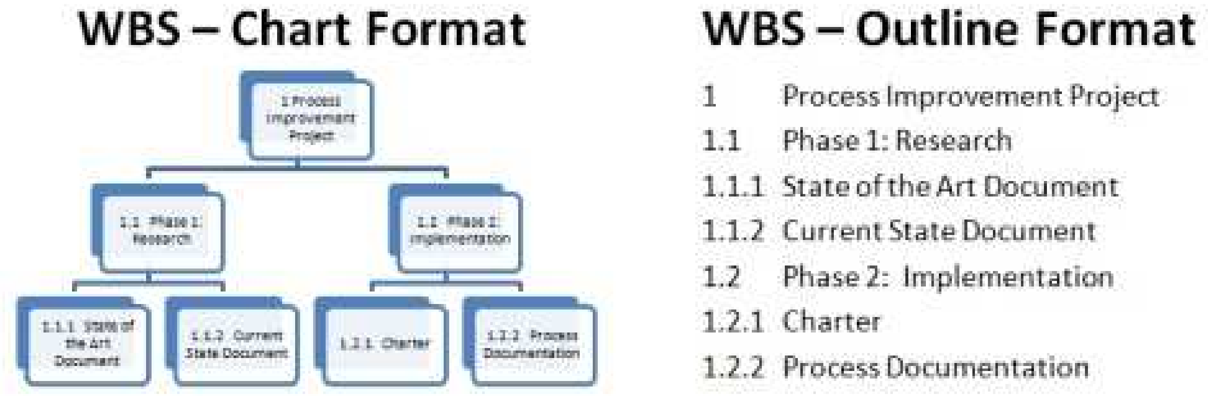
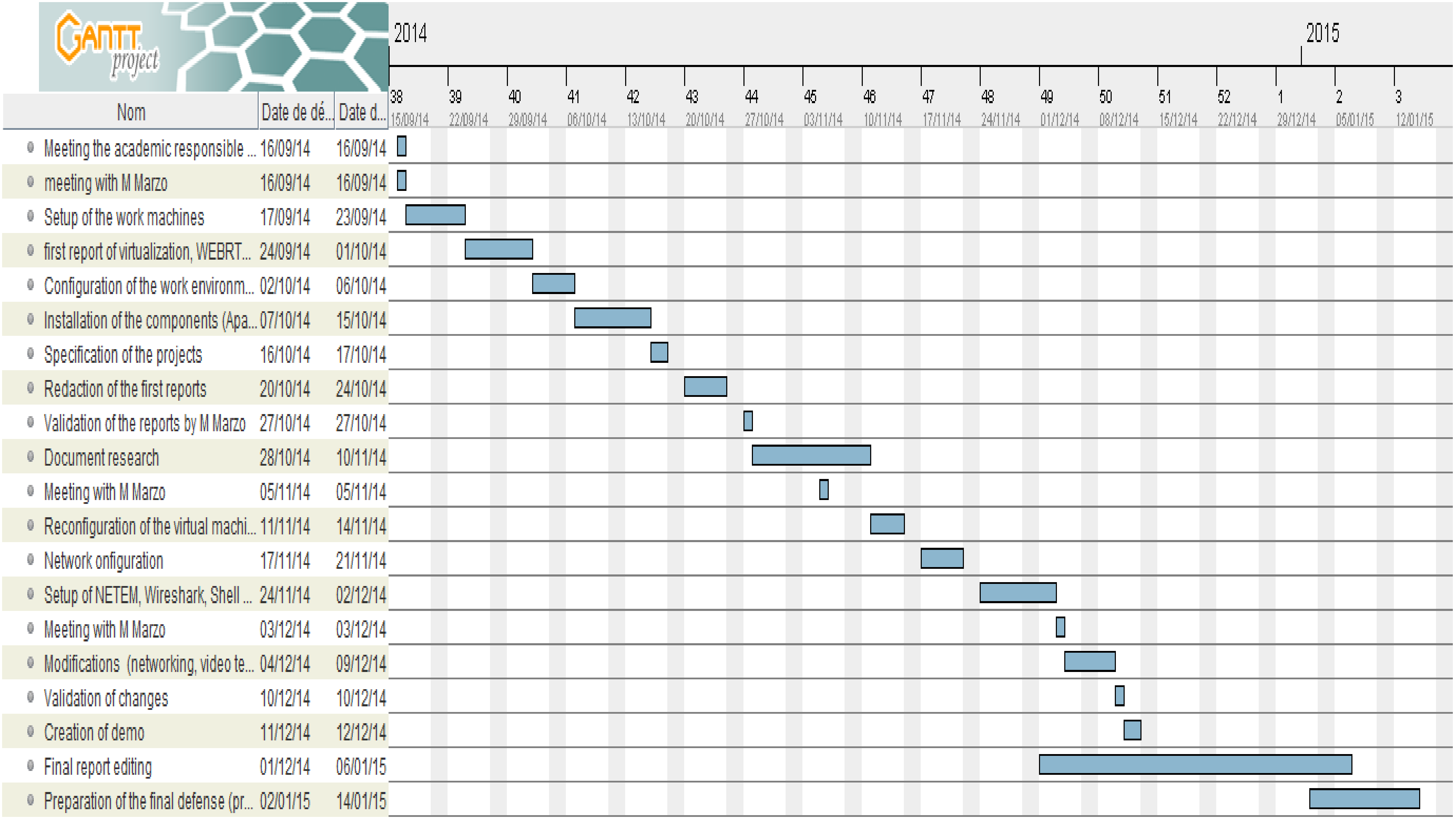

WBS – Chart Format
1 Process Improvement Project
1.1 Phase 1: Research
1.2 Phase 2: Implementation
1.1.1 State of the Art Document
1.1.2 Current State Document
1.2.1 Charter
1.2.2 Process Documentation
WBS – Outline Format
1 Process Improvement Project
1.1 Phase 1: Research
1.1.1 State of the Art Document
1.1.2 Current State Document
1.2 Phase 2: Implementation
1.2.1 Charter
1.2.2 Process Documentation

Figure 10: WBS chart format



Nom	Date de dé...	Date d...
Meeting the academic responsible ...	16/09/14	16/09/14
meeting with M Marzo	16/09/14	16/09/14
Setup of the work machines	17/09/14	23/09/14
first report of virtualization, WEBRT...	24/09/14	01/10/14
Configuration of the work environm...	02/10/14	06/10/14
Installation of the components (Apa...	07/10/14	15/10/14
Specification of the projects	16/10/14	17/10/14
Redaction of the first reports	20/10/14	24/10/14
Validation of the reports by M Marzo	27/10/14	27/10/14
Document research	28/10/14	10/11/14
Meeting with M Marzo	05/11/14	05/11/14
Reconfiguration of the virtual machi...	11/11/14	14/11/14
Network onfiguration	17/11/14	21/11/14
Setup of NETEM, Wireshark, Shell ...	24/11/14	02/12/14
Meeting with M Marzo	03/12/14	03/12/14
Modifications (networking, video te...	04/12/14	09/12/14
Validation of changes	10/12/14	10/12/14
Creation of demo	11/12/14	12/12/14
Final report editing	01/12/14	06/01/15
Preparation of the final defense (pr...	02/01/15	14/01/15

CONCLUSION

Assessment of the situation

NetEM is vast, and we can't cover everything in this project.

The solution is operational and will be integrated to VITAM, The tests are pleasant and the team seems to understand the working of the tool.

The Delays and Losses can be added as requested, and we can also add corruption and other problems to test the QOS of VITAM, this test tool is so simple and can be used in different situations and different areas.

The report can be a good help for the person who want to continue the researches about the quality of service, not only for VITAM but also for other modules related to this tool.

- **Issue and problems before starting the project:**
  - **Memory of the server:** The server has insufficient memory to support many virtual machines (only 2GB).
  - **Lack of addresses in laboratory:** We have not enough addresses to connect all the machines with Ethernet connection.
  - **Wi-Fi access :** Wi-Fi access in laboratory via BCDS207 turns off and we have to reboot the router every day, otherwise, we haven't receive our student cards yet so we cannot access to the UDG Wireless network.

- **Solutions proposed and realized:**
  - **Memory of the server:** We have added more memory to the server and now it supports the 3 Virtal machines working simultaneously.
  - **Lack of addresses in laboratory:** We didn't need to connect all the virtual machines to internet, we have worked only with private addresses on private networks, and have used NAT interfaces on Virtualbox to use internet when it was necessary.
  - **Wi-Fi access:** We have received our student cards and the problem was resolved.

Personal assessment

Finally, the realization of this project was an opportunity to take my role as a project manager and take care of everything from understanding the needs to issuance of the solution while respecting the instructions and the given deadlines. Thus this work allowed me to learn the methodology of working in UDG laboratory and enhance my knowledge on networking and systems administration allowing me to be able to apply it to future projects in my career as a project manager.

Another positive point is the relational aspect and teamwork. Indeed the collaboration with the other members of VITAM team has been a huge source of information about the functioning of everything in the laboratory. Also the fact of working in a team with my classmates has developed more my communication skills and improved a lot my English with – of course- some Catalan words.

This semester in ERASMUS have been very beneficial and has allowed me to get to know the requirements of another university outside of FRANCE, I gained practical experience that will serve me throughout my professional career.

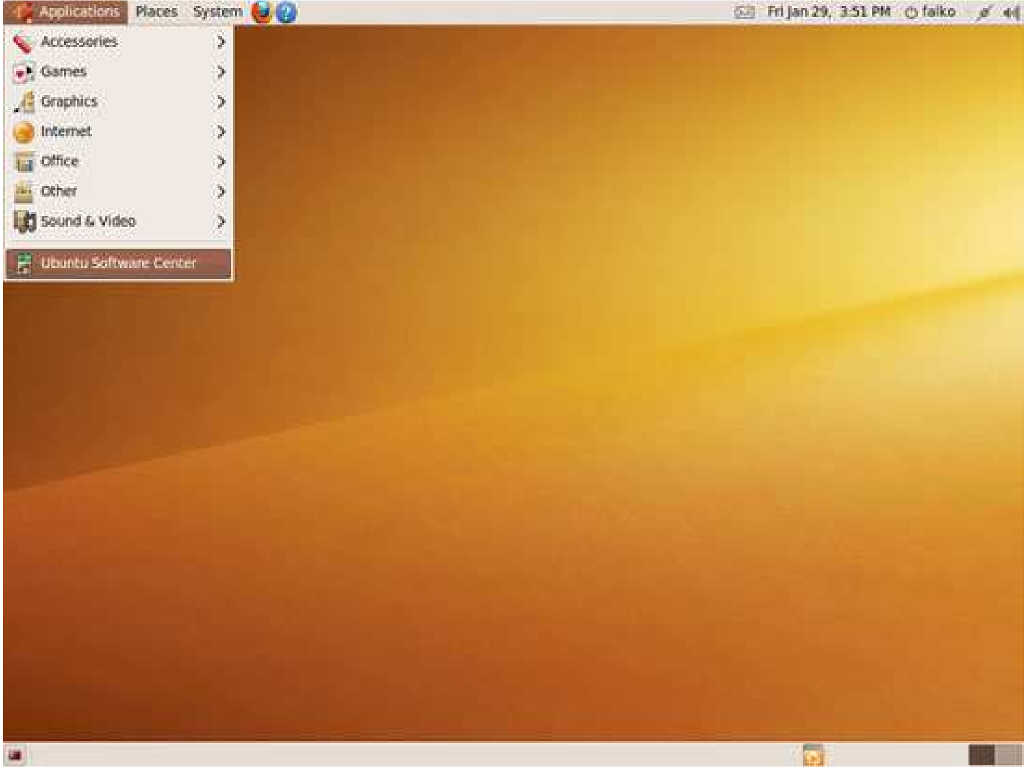
BIBLIOGRAPHY

- http://en.wikipedia.org/wiki/Quality_of_service
- http://fr.wikipedia.org/wiki/Qualit%C3%A9_de_service
- http://fr.wikipedia.org/wiki/WebRTC
- http://en.wikipedia.org/wiki/Virtualization
- http://www.html5rocks.com/en/tutorials/webrtc/basics/
- http://searchservervirtualization.techtarget.com/definition/virtualization
- http://man7.org/linux/man-pages/man8/tc-netem.8.html
- http://www.linuxfoundation.org/collaborate/workgroups/networking/netem
- http://tdistler.com/2011/06/10/netem-wan-emulation-how-to-setup-a-netem-box
- http://fr.scribd.com/doc/240355395/Wireshark-Ubuntu-Installation
- http://packetlife.net/blog/2010/mar/19/sniffing-wireshark-non-root-user/
- http://www.webrtc.org/
- http://www.rtcquickstart.org/ICE-STUN-TURN-server-installation
- http://fr.wikipedia.org/wiki/WebRTC
- http://www.linux-france.org/
- http://www.networkworld.com/
- https://www.wireshark.org/about.html
- https://www.howtoforge.com/network-analysis-with-wireshark-on-ubuntu-9.10

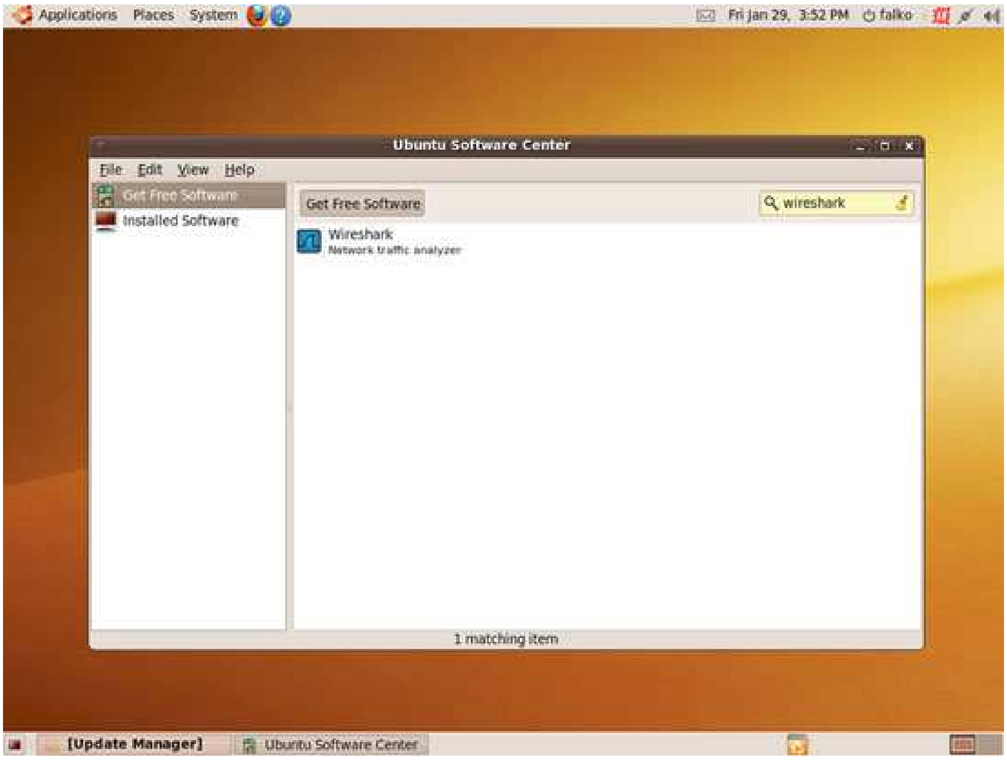
ANNEX

Install Wirshark

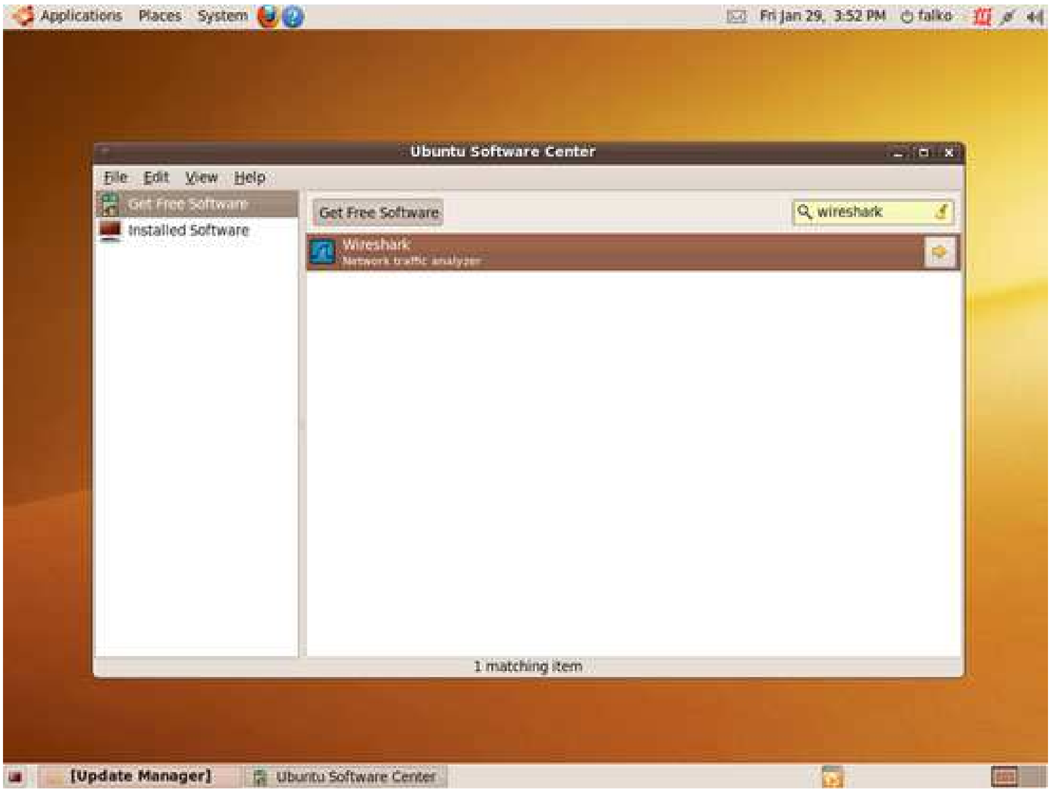
Go to Applications > Ubuntu Software Center...



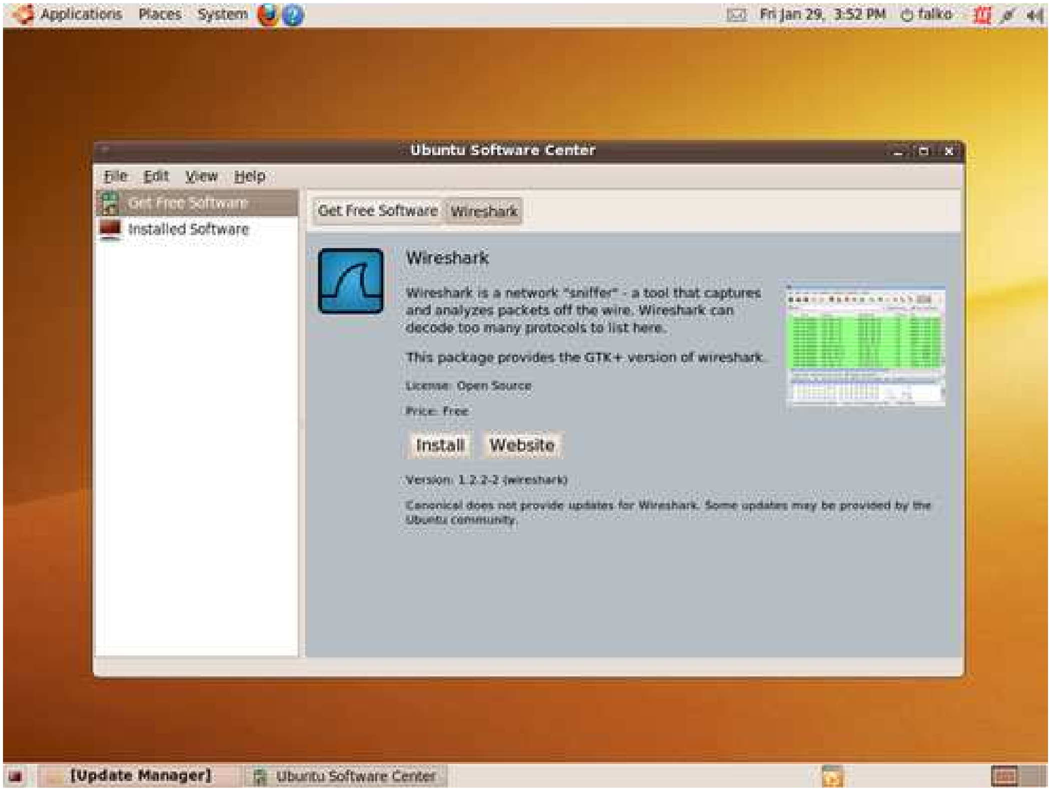
... and search for wireshark:



Mark the Wireshark package and click on the arrow on the right:



On the next screen, click the Install button:



Wireshark

Wireshark is a network "sniffer" - a tool that captures and analyzes packets off the wire. Wireshark can decode too many protocols to list here.

This package provides the GTK+ version of wireshark.

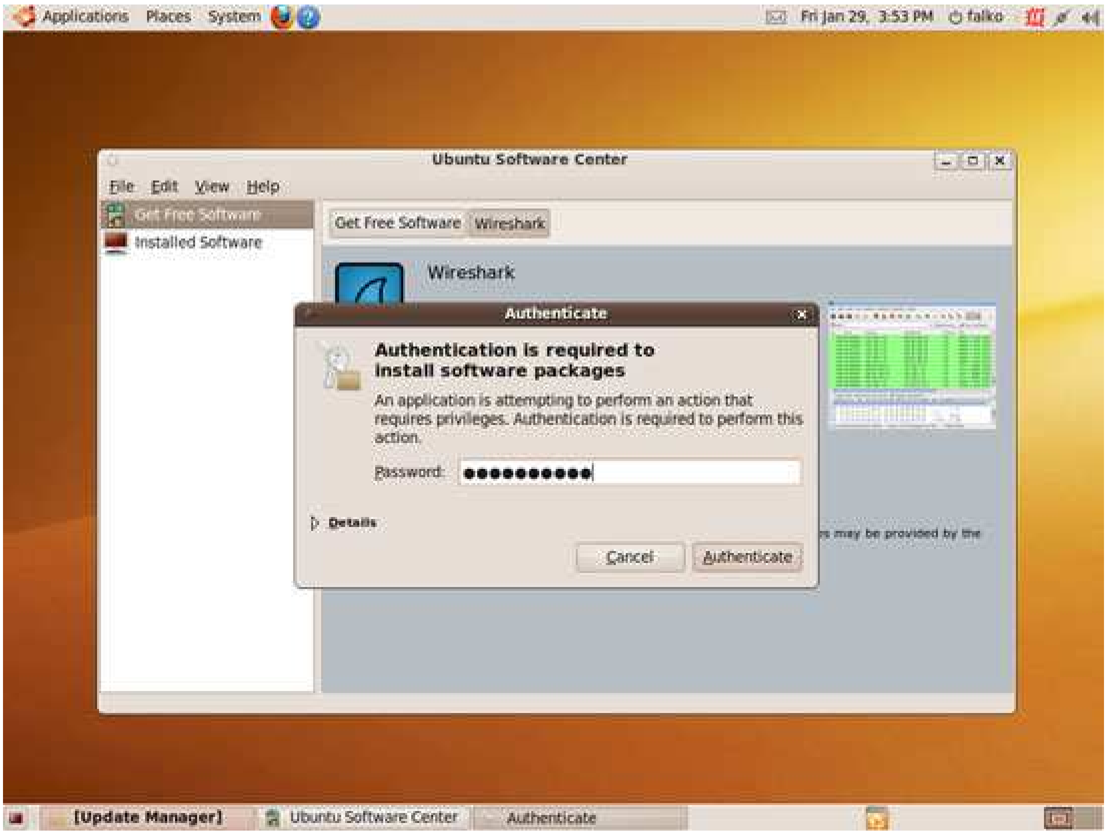
License: Open Source

Price: Free

Version: 1.2.2-2 (wireshark)

Canonical does not provide updates for Wireshark. Some updates may be provided by the Ubuntu community.

Type in your password:

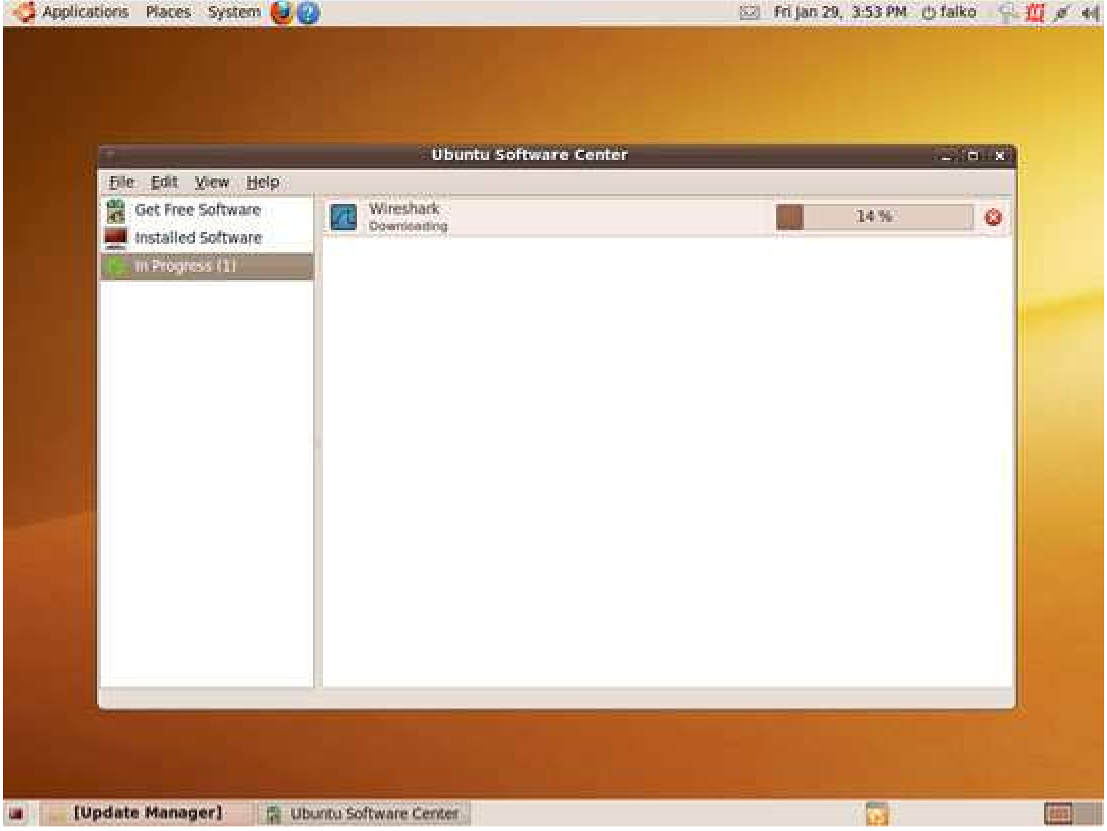


Authentication is required to install software packages

An application is attempting to perform an action that requires privileges. Authentication is required to perform this action.

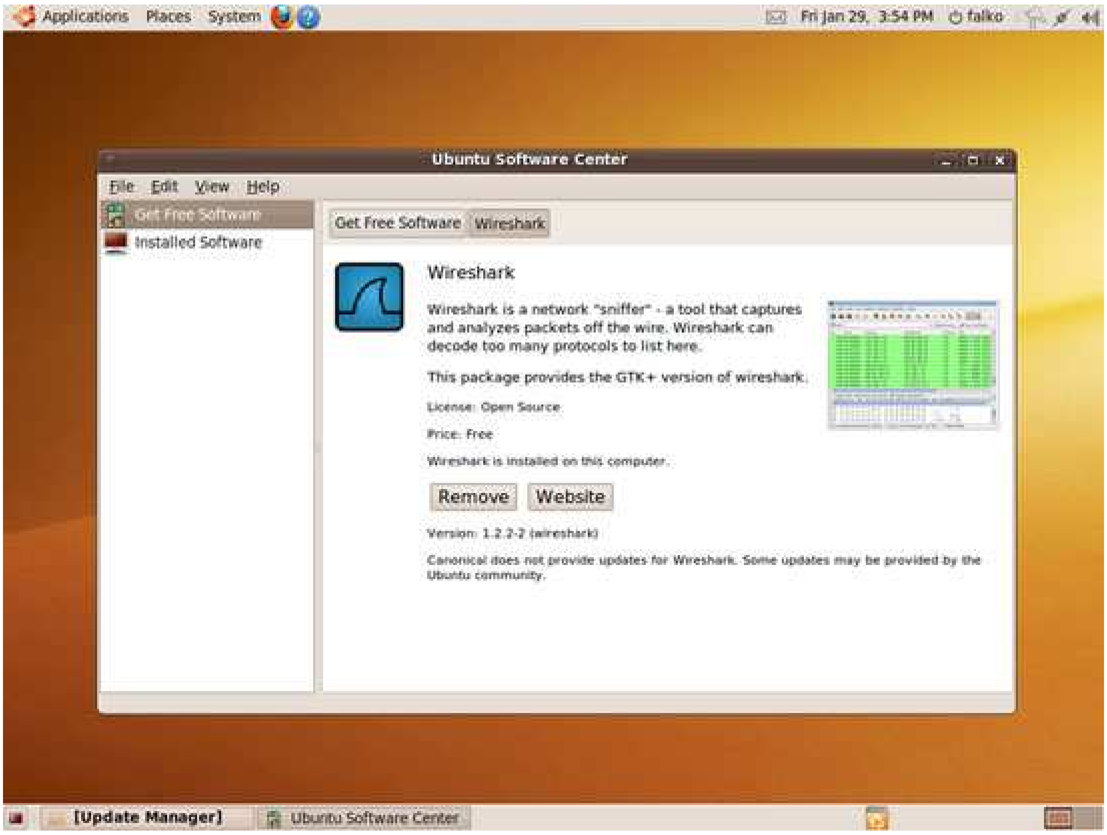
Password: ●●●●●●●●●●●

Wireshark is now being downloaded and installed:



Wireshark
Downloading
14 %

You can close the Ubuntu Software Center window afterwards:



Wireshark

Wireshark is a network "sniffer" - a tool that captures and analyzes packets off the wire. Wireshark can decode too many protocols to list here.

This package provides the GTK+ version of wireshark.

License: Open Source

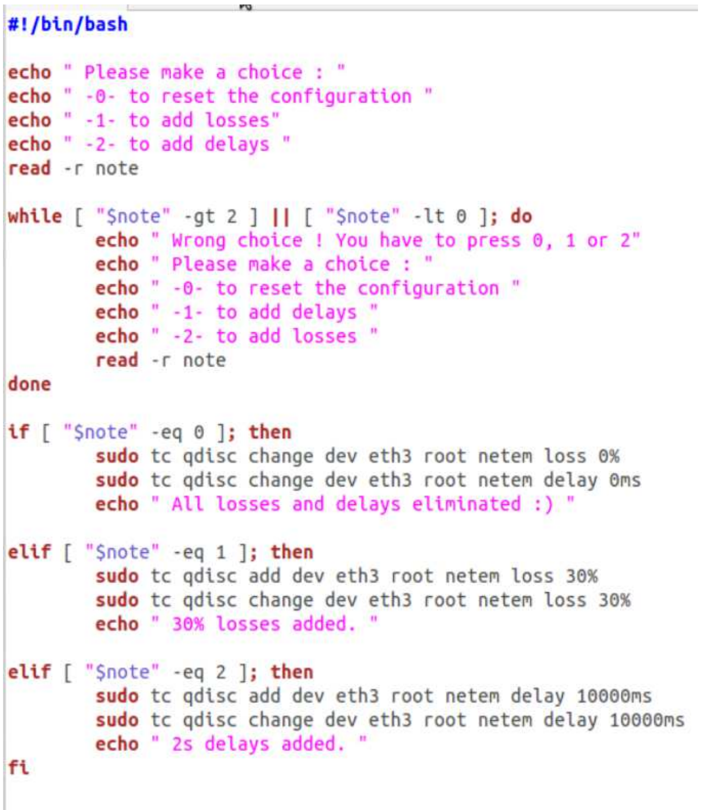
Price: Free

Wireshark is installed on this computer.

Version: 1.2.2-2 (wireshark)

Canonical does not provide updates for Wireshark. Some updates may be provided by the Ubuntu community.

NETEM script:



```
#!/bin/bash

echo " Please make a choice : "
echo " -0- to reset the configuration "
echo " -1- to add losses"
echo " -2- to add delays "
read -r note

while [ "$note" -gt 2 ] || [ "$note" -lt 0 ]; do
        echo " Wrong choice ! You have to press 0, 1 or 2"
        echo " Please make a choice : "
        echo " -0- to reset the configuration "
        echo " -1- to add delays "
        echo " -2- to add losses "
        read -r note
done

if [ "$note" -eq 0 ]; then
        sudo tc qdisc change dev eth3 root netem loss 0%
        sudo tc qdisc change dev eth3 root netem delay 0ms
        echo " All losses and delays eliminated :) "

elif [ "$note" -eq 1 ]; then
        sudo tc qdisc add dev eth3 root netem loss 30%
        sudo tc qdisc change dev eth3 root netem loss 30%
        echo " 30% losses added. "

elif [ "$note" -eq 2 ]; then
        sudo tc qdisc add dev eth3 root netem delay 10000ms
        sudo tc qdisc change dev eth3 root netem delay 10000ms
        echo " 2s delays added. "
fi
```

This script contains the two important NETEM commands to add delays and losses to the traffic.

First of all, the script asks the user to make a choice, and read its answer; if the answer is not in the interval [0, 2] the message is showed again until the answer is correct.

One the answer is good; the script use the “if condition” to know which command is good to apply.

Each choice has two commands; the first is to add when no value already exists, and the second one change the value when it already exists.

GLOSSARY

.

VM: virtual machine
CPU: Central Processing Unit
OS: Operating system
JS: Java Script
HTML: Hypertext Markup Language
WebRTC: Web Real-Time Communication
NAT: Network Address Translation
STUN: Session Traversal Utilities for NAT
TURN: Traversal Using Relay NAT
OSMF: Open Source Media Framework
HTTP: HyperText Transfert Protocol
BCDS: Broadband Communications and Distributed Systems
NETEM: Network Emulation
LAN: Local Area Network
WAN: Wide Area Network
QOS: Quality Of Service
IP: Internet Protocol
TC: Traffic Control
FIFO: First In First Out
TCP: Transmission Control Protocol
UDP: User Datagram Protocol
SSL: Secure Socket Layer
TLS: Transport Layer Security
CSV: Comma-Separated Values
XML: Extensible Markup Language
ICMP: Internet Control Message Protocol