Network Tools

Reti di Elaboratori - 18/19
Corso di Laurea in Informatica
Università degli Studi di Roma “La Sapienza”

Mauro Piva - Andrea Coletta
{coletta | piva}@di.uniroma1.it

Tools

- Ifconfig
- Ping
- Traceroute
- Nslookup
- Netstat
- Netcat
- Postman
- Wireshark

Disclaimer

You are free to use your favourite operating system, but during this and the following practical lectures, **we will only refer to GNU/Linux.**

- Other operating systems may have slightly different behaviours or tool implementations we won't discuss (although there might be some exception to this rule)
- It is strongly recommended to run the examples at home (and also in class)
- For Windows/OSX users:
  - You can run Linux on a virtual machine
  - VirtualBox is free and easy to use
  - You can download the image of a XUbuntu distribution from: http://virtualboxes.org/images/xubuntu/
    - it's very lightweight, should run on older computers too
- Another possibility would be to use a XUbuntu as a Live distribution http://xubuntu.org/getxubuntu/ (does not require to install software)

Installation (Ubuntu)

- *sudo apt update*
- *sudo apt install net-tools*
- *sudo apt install traceroute*
- *Wireshark:*
  - *sudo add-apt-repository ppa:wireshark-dev/stable*
  - *sudo apt-get update*
  - *sudo apt-get install wireshark*

Why network tools?

- They are useful in networking **Teaching/Research** and also in **“real world”** (e.g. **debugging/ monitoring/ …)**
- **Measurement/monitoring tools** and tools for **handle complex tasks (**e.g. **opening connection/ creating HTTP request/ port scanning/** …)
- **Active** and **Passive Monitoring Network Tools:**
    - **Active monitoring tools** entail **injecting test traffic** onto a network and monitoring the flow of that traffic. Usually, with active monitoring we can find information about delay/packet loss, topology/routing, bandwidth/throughput.
    - **Passive monitoring tools passively** monitor existing traffic in the network. Passive monitoring requires a device on the network to capture network packets for analysis.

Ifconfig

Command: **“ifconfig”**

What is it?

Why is it useful?

Ifconfig

Command: **“ifconfig”**

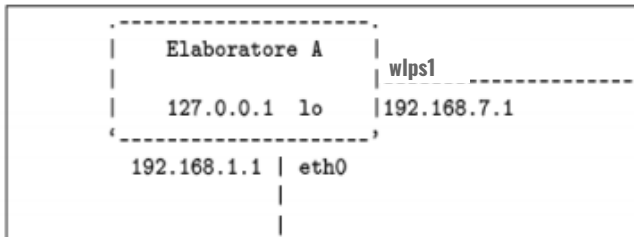
```
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Loopback locale)
        RX packets 8825  bytes 810929 (810.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8825  bytes 810929 (810.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.137  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::5df9:e9b3:9109:7df  prefixlen 64  scopeid 0x20<link>
        ether 18:1d:ea:b1:91:3a  txqueuelen 1000  (Ethernet)
        RX packets 535048  bytes 687010893 (687.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 189755  bytes 39604163 (39.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

ifconfig

- It helps to to **configure** and **show** the network interfaces.
  - It is used at boot time to **set up** interfaces as necessary.
  - After the boot time it is usually only needed for **debugging or system tuning.**

Do you remember **network interfaces?**



*How many **interfaces? Which are?***

Network interfaces:

- **Loopback**
- **Ethernet**
- **Wireless**

ifconfig

“ifconfig” : If no arguments are given, ifconfig **displays the status** of the all currently **active interfaces**.

“ifconfig eht0” : If a single interface argument is given, it **displays the status of the given interface only;**

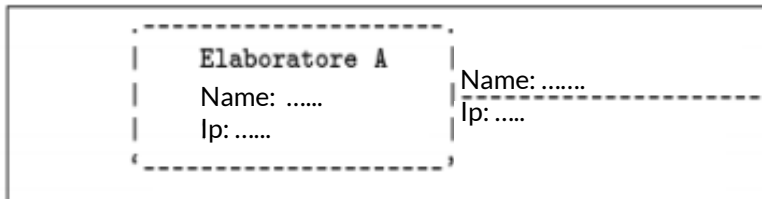
“ifconfig -a” : if a single -a argument is given, it **displays** the status of all interfaces, **even those that are down.**

Otherwise, it configures an interface:

- “**ifconfig eth0 down”** : it **disables** the given interface (no traffic is sent or received on a disabled interface).
- **“ifconfig eth0 up” :** it **enables** the given interface.

ifconfig

Understand output:



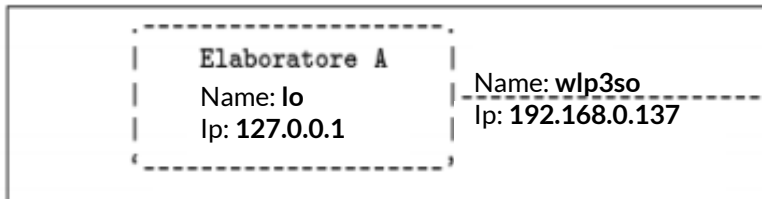

```
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Loopback locale)
        RX packets 8825  bytes 810929 (810.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8825  bytes 810929 (810.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.137  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::5df9:e9b3:9109:7df  prefixlen 64  scopeid 0x20<link>
        ether 18:1d:ea:b1:91:3a  txqueuelen 1000  (Ethernet)
        RX packets 535048  bytes 687010893 (687.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 189755  bytes 39604163 (39.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

ifconfig

Understand output:

Description of the network interface




```
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Loopback locale)
        RX packets 8825  bytes 810929 (810.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8825  bytes 810929 (810.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.137  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::5df9:e9b3:9109:7df  prefixlen 64  scopeid 0x20<link>
        ether 18:1d:ea:b1:91:3a  txqueuelen 1000  (Ethernet)
        RX packets 535048  bytes 687010893 (687.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 189755  bytes 39604163 (39.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Mask di rete

Indirizzo ip

Transmission info (mac_address, len queue, packet type)

ping

Command: “**ping www.google.com**”

What is it?

Why is it useful?

ping

Command: “**ping www.google.com**”

```
PING www.google.com (216.58.205.68) 56(84) bytes of data
64 bytes from mil04s25-in-f4.1e100.net (216.58.205.68): icmp_seq=1 ttl=55 time=11.6 ms
64 bytes from mil04s25-in-f4.1e100.net (216.58.205.68): icmp_seq=2 ttl=55 time=11.8 ms
64 bytes from mil04s25-in-f4.1e100.net (216.58.205.68): icmp_seq=3 ttl=55 time=13.4 ms
64 bytes from mil04s25-in-f4.1e100.net (216.58.205.68): icmp_seq=4 ttl=55 time=11.9 ms
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 11.731/11.924/12.310/0.240 ms
```

ping

Command: “**ping www.google.com**”

Find ip address

Payload (Size of packet)

```
PING www.google.com (216.58.205.68) 56(84) bytes of data
64 bytes from mil04s25-in-f4.1e100.net (216.58.205.68): icmp_seq=1 ttl=55 time=11.6 ms
64 bytes from mil04s25-in-f4.1e100.net (216.58.205.68): icmp_seq=2 ttl=55 time=11.8 ms
64 bytes from mil04s25-in-f4.1e100.net (216.58.205.68): icmp_seq=3 ttl=55 time=13.4 ms
64 bytes from mil04s25-in-f4.1e100.net (216.58.205.68): icmp_seq=4 ttl=55 time=11.9 ms

4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 11.731/11.924/12.310/0.240 ms
```

Overall statistics

Response time shows the round trip time (**not one-way)**.

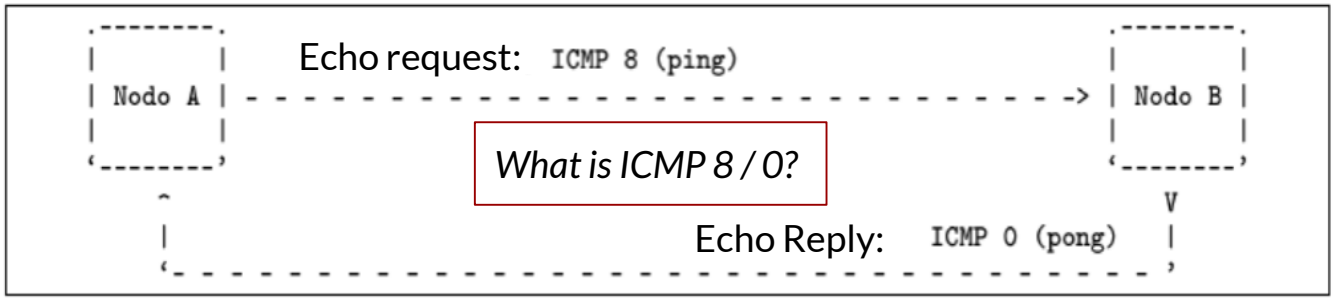
Time-to-live of packet

ping

- **Ping** is used to test the **reachability of a host** on an Internet Protocol (IP) network or to **discovery hosts**. It also measures the **round-trip time** for messages from the host to a destination computer.
- It use **ICMP protocol. Port: 1** and **58**

What is ICMP?

- **Source host creates** an **ICMP packet** and forwards it.
  - **Echo Request.**
- If **destination host receives** the packet, it creates a new **ICMP packet** and send it back to the source.
  - **Echo Reply.**



ICMP

- ICMP, specified in [RFC 792], is used by hosts and routers to **communicate network-layer information to each other**. The most typical use of ICMP is for **control, diagnostic** or **error reporting.**
  - **For example:** in HTTP session, you may have encountered an error message such as "**Destination host unreachable.**" This message had its origins in ICMP.

Who creates this error ICMP packet?

- ICMP messages are carried as IP payload, just as TCP or UDP segments are carried as IP payload

```
+-----------+-------------+----------
| Header IP | Header ICMP | payload....
+-----------+-------------+----------
```

- **Time-To-Live**, **Source Address** and **Destination Address**, from IP header, are of significant importance for ICMP.

Why?

ICMP

ICMP Type	Code	Description
0	0	echo reply (to ping)
3	0	destination network unreachable
3	1	destination host unreachable
.....		
8	0	echo request
.....		
11	0	TTL expired
12	0	IP header bad

- ICMP Header:
  - **Type** of ICMP message (8 bits)
  - **Code** (8 bits)
  - **Checksum** (16 bits)
  - **Header Data** (32 bits) field. In this case (ICMP echo request and replies) they will be:
    - **Identifier** (16 bits)
    - **Sequence number** (16 bits)
  - **ICMP Payload.** It can be an arbitrary length.
    - The payload may include a **timestamp** indicating the time of transmission. This allows ping to compute the **round trip time**.

Why are useful?

ping

Command: **“ping address”**

- To run ping open a terminal
- Ping command: “**ping www.google.com”**
- Find out about ping options, some are interesting : “**man ping**”
- Try to **send specific number of packets**

traceroute

Command: **“traceroute google.com”**

What is it?

Why is it useful?

traceroute

Command: “**traceroute www.google.com**”

```
traceroute to google.com (216.58.205.110), 30 hops max, 60 byte packets
 1  _gateway (192.168.43.1)  5.561 ms  5.812 ms  6.324 ms
 2  * * *
 3  172.31.9.101 (172.31.9.101)  31.753 ms  34.202 ms 32.044 ms
 4  172.30.32.132 (172.30.32.132)  33.687 ms  33.668 ms  33.638 ms
 5  172.19.202.2 (172.19.202.2)  31.470 ms  34.650 ms  32.370 ms
 6  172.19.202.17 (172.19.202.17)  33.815 ms  23.462 ms  27.474 ms
 7  172.19.202.36 (172.19.202.36)  28.521 ms 172.19.202.32 (172.19.202.32)  32.609 ms  32.802 ms
 8  172.17.54.158 (172.17.54.158)  38.374 ms 39.416 ms 39.643 ms
 9  172.19.177.42 (172.19.177.42)  38.987 ms  38.289 ms *
10  * * *
11  etrunk49.milano1.mil.seabone.net (195.22.205.98)  33.981 ms  35.888 ms 34.874 ms
12  74.125.51.148 (74.125.51.148)  36.745 ms 27.963 ms  34.215 ms
13  108.170.245.81 (108.170.245.81)  33.119 ms 45.432 ms  43.940 ms
14  216.239.50.241 (216.239.50.241)  41.238 ms 38.945 ms  34.009 ms
15  mil04s26-in-f110.1e100.net (216.58.205.110) 52.120 ms 48.032 ms  44.321 ms
```

traceroute

Command: “**traceroute www.google.com**”

```
traceroute to google.com (216.58.205.110), 30 hops max, 60 byte packets
 1  _gateway (192.168.43.1)  5.561 ms  5.812 ms  6.324 ms
 2  * * *
 3  172.31.9.101 (172.31.9.101)  31.753 ms  34.202 ms 32.044 ms
 4  172.30.32.132 (172.30.32.132)  33.687 ms  33.668 ms  33.638 ms
 5  172.19.202.2 (172.19.202.2)  31.470 ms  34.650 ms  32.370 ms
 6  172.19.202.17 (172.19.202.17)  33.815 ms  23.462 ms  27.474 ms
 7  172.19.202.36 (172.19.202.36)  28.521 ms 172.19.202.32 (172.19.202.32)  32.609 ms  32.802 ms
 8  172.17.54.158 (172.17.54.158)  38.374 ms 39.416 ms 39.643 ms
 9  172.19.177.42 (172.19.177.42)  38.987 ms  38.289 ms *
10  * * *
11  etrunk49.milano1.mil.seabone.net (195.22.205.98)  33.981 ms  35.888 ms 34.874 ms
.....
```

Router index	Router name	IP router	RTT (1° pkt)	RTT (2° pkt)	RTT (3° pkt)
11	**etrunk49.milano1.mil.seabone.net**	**195.22.205.98**	**33.981 ms**	**35.888 ms**	**34.874 ms**

traceroute

Command: "**traceroute www.google.com**"

```
traceroute to google.com (216.58.205.110), 30 hops max, 60 byte packets
 1  _gateway (192.168.43.1)  5.561 ms  5.812 ms  6.324 ms
 2  * * *
 3  172.31.9.101 (172.31.9.101)  31.753 ms  34.202 ms 32.044 ms
 4  172.30.32.132 (172.30.32.132)  33.687 ms  33.668 ms  33.638 ms
 5  172.19.202.2 (172.19.202.2)  31.470 ms  34.650 ms  32.370 ms
 6  172.19.202.17 (172.19.202.17)  33.815 ms  23.462 ms  27.474 ms
 7  172.19.202.36 (172.19.202.36)  28.521 ms 172.19.202.32 (172.19.202.32)  32.609 ms  32.802 ms
 8  172.17.54.158 (172.17.54.158)  38.374 ms 39.416 ms 39.643 ms
 9  172.19.177.42 (172.19.177.42)  38.987 ms  38.289 ms *
10  * * *
11  etrunk49.milano1.mil.seabone.net (195.22.205.98)  33.981 ms  35.888 ms 34.874 ms
12  74.125.51.148 (74.125.51.148)  36.745 ms 27.963 ms  34.215 ms
13  108.170.245.81 (108.170.245.81)  33.119 ms 45.432 ms  43.940 ms
14  216.239.50.241 (216.239.50.241)  41.238 ms 38.945 ms  34.009 ms
15  mil04s26-in-f110.1e100.net (216.58.205.110) 52.120 ms 48.032 ms  44.321 ms
```

Local Network

Why two IPs?

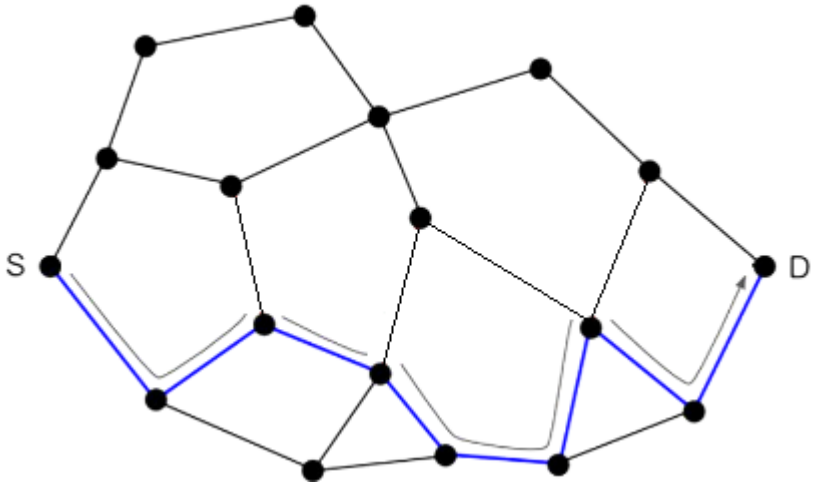
The packet was not returned within the expected timeframe.

Request Timed out (the router doesn't answer or answer is lost)

Destination

traceroute

- The **traceroute** is a network diagnostic tool and it is used to **discover the routes** that packets actually take when traveling from the host to the destination.
- It helps to **measuring transit delays** of packets across the network.
- It use **IP protocol.**
- It uses **UDP packets** and **ICMP packets**.
    - UDP high port number usually 33434 (ubuntu).



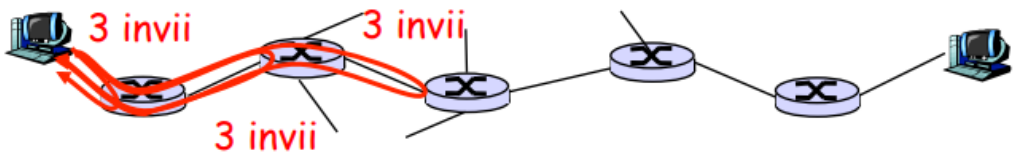
Why?

traceroute

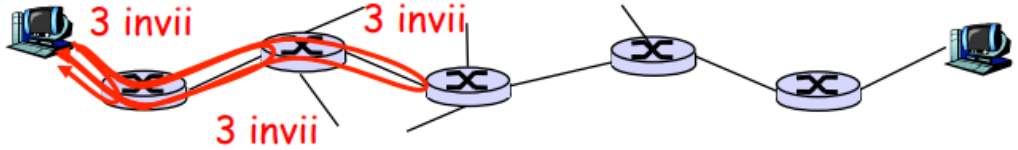
How it works?

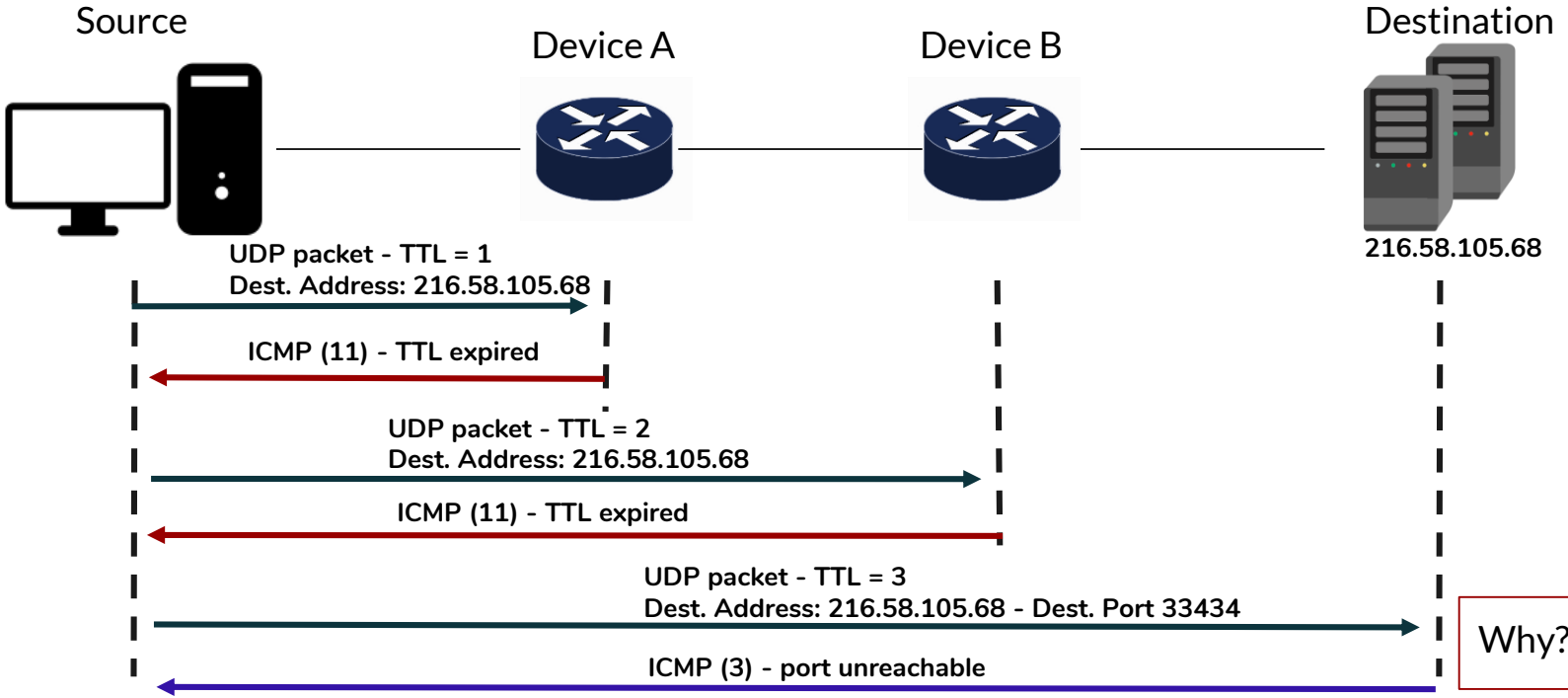
The device (for example, our laptop) **sends out a sequence of User Datagram Protocol (UDP)** datagrams to an invalid (high) port address at the remote host (destination).

1. It send groups of **3** (default) **UDP packets with incremental TTL (from 1 to n)** along the network to reach the destination. (Note that **n** is the maximum number of **hop** to reach the destination and **by default n=30**.)
2. A packet group with TTL = i will reach the router at i-position on the path to the destination.
3. This router send back a response that helps the host to calculates the RTT.



traceroute


3 invii
3 invii
3 invii


Source
Device A
Device B
Destination
216.58.105.68
UDP packet - TTL = 1
Dest. Address: 216.58.105.68
ICMP (11) - TTL expired
UDP packet - TTL = 2
Dest. Address: 216.58.105.68
ICMP (11) - TTL expired
UDP packet - TTL = 3
Dest. Address: 216.58.105.68 - Dest. Port 33434
ICMP (3) - port unreachable
Why?

traceroute

“traceroute [options] destination”

Options:

- **-q** *nqueries* **-** Sets the number of probe packets per hop. The default is 3.
- **-m** *max_ttl* -Specifies the maximum number of hops (max time-to-live value) *traceroute* will probe. The default is 30.
- **-p** *port* - For UDP tracing, specifies the destination port base *traceroute* will use (the destination port number will be incremented by each probe).

nslookup

Command: **“nslookup www.google.com”**

What is it?

Why is it useful?

nslookup

Command: **“nslookup www.google.com”**

```
Server:  8.8.8.8
Address: 8.8.8.8#53

Non-authoritative answer:
Name:    google.it
Address: 216.58.205.35
Name:    www.google.it
Address: 2a00:1450:4002:807::2003
```

DNS Server

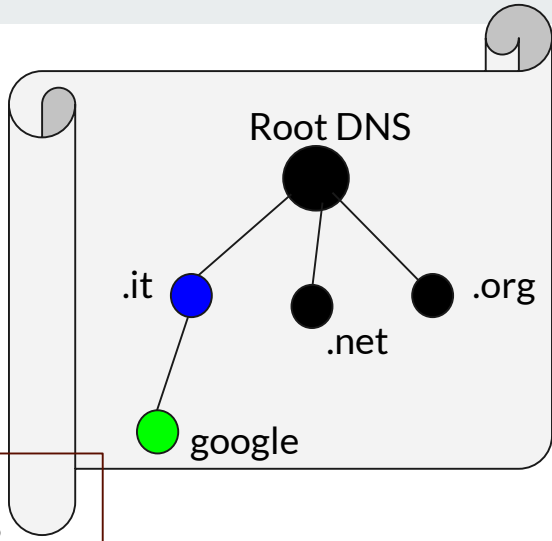
Answer_1

Answer_2

nslookup

- **nslookup** is a network command-line tool for querying the **Domain Name System (DNS).**
- **Syntax: nslookup [-option] [name] [server]**
- **By default:**
    - **Use current default DNS Server** (/etc/resolv.conf on Linux).
    - **It tells the DNS server to perform a recursive query.**
    - **Default DNS query type=A**
- A query DNS use **UDP and port 53**

DNS



The main task of **Internet's domain name system (DNS)** is to **translates hostnames to IP addresses.**

The DNS is (1) **a distributed database** implemented in a hierarchy of DNS servers, and (2) **an application-layer protocol** that allows hosts to query the distributed database.

Why not one DNS server that contains a map with all the data ?

To deal with the issue of scale, the DNS uses a large number of servers, organized in a hierarchical fashion and distributed around the world. Three classes o DNS server:

1. root DNS servers
2. top-level domain (TLD) DNS servers
3. Authoritative DNS servers

Furthermore, there is also the **Local DNS server** (default name server)**.** It handles the DNS queries of the user by forwarding them to the DNS server hierarchy.

DNS

The DNS servers store the **resource records (RRs), i.e. mapping between two resources**

Each DNS reply message carries one or more resource records.

A resource record is a four-tuple that contains the following fields:

(Name, Value, Type, TTL)

The meaning of Name and Value depend on Type.

With nslookup we can retrieve different RRs:
nslookup [-type=TYPE] name [server]

DNS

Types.

Type **A** ---> Hostname - IPv4 **A**ddress

Type **AAAA** ---> Hostname - IPv6 **A**ddress

Type **CNAME** --->Alias - **C**anonical **Name**

Type **NS ---> D**omain name - **N**ame **S**erver

Type **MX ---> A**lias - **M**ail Server

RR : (Name, Value, Type, TTL)

{host_name, IPv4_addr, **A**, TTL}

{host_name, IPv6_addr, **AAAA**, TTL}

{alias, canonical_name, **CNAME**, TTL}

{domain, authoritative_hostname, **NS**, TTL}

{alias, canonical_name, **MX**, TTL}

nslookup

Example 0.

```
nslookup uniroma1.it
Server:           8.8.8.8
Address:          8.8.8.8#53

Non-authoritative answer:
Name:    uniroma1.it
Address: 151.100.101.140
```

Means: give me the address of the domain uniroma1.it

the answer is not authorative. These values are coming from 8.8.8.8's cache

nslookup

Means: give me the IPv6 address of the www.google.com

Example 1.

```
nslookup -type=AAAA www.google.com

Server:           8.8.8.8
Address:          8.8.8.8#53


Non-authoritative answer:
Name:    www.google.com
Address: 2a00:1450:4002:80a::2004
```

IPv6

nslookup

Means: give me the name server responsible (authoritative) of the domain uniroma1.it

Example 2.

```
nslookup -type=NS uniroma1.it
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
uniroma1.it     nameserver = ns1.garr.net.
uniroma1.it     nameserver = desiree.cics.uniroma1.it.
uniroma1.it     nameserver = risc-ns.cics.uniroma1.it.
```

nslookup

Example 3.

Ip address of the desider authoritative DNS **`desiree.cics.uniroma1.it`**.

```
nslookup uniroma1.it 151.100.4.13
Server:         151.100.4.13
Address:        151.100.4.13#53

Name:   uniroma1.it
Address: 151.100.101.140
```

Finally! Next question is: who is responsible for the root of the tree?

nslookup

Means: give me the name server responsible for the **root domain '.'**

Example 4.

```
nslookup -type=NS .
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
.       nameserver = a.root-servers.net.
.       nameserver = b.root-servers.net.
.....
.       nameserver = l.root-servers.net.
.       nameserver = m.root-servers.net.
```

Root DNS servers.
In the Internet there are 13 root DNS servers (labeled A through M).

nslookup

Laundry service in Dallas (TX)

Example 5.

```
time nslookup dfwlls.com
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   dfwlls.com
Address: 23.236.62.147
```

8 seconds!!!

```
real    0m 8,136s
user    0m 0,014s
```

```
time nslookup bing.com
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   bing.com
Address: 204.79.197.200
```

```
real    0m 0,721s
user    0m 0,014s
```

nslookup

Laundry service in Dallas (TX)

Example 6.

```
time nslookup dfwlls.com
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   dfwlls.com
Address: 23.236.62.147
```

8 seconds!!!

```
real    0m 8,136s
user    0m 0,014s
```

Example 6 - **If I repeat the query?**

```
time nslookup dfwlls.com
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   dfwlls.com
Address: 23.236.62.147
```

```
real    0m 0,106s
user    0m 0,014s
```

nslookup

Get IP of “phd.di.uniroma1.it”

Example 7.

```
nslookup phd.di.uniroma1.it
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
phd.di.uniroma1.it      canonical name = ccalcolo.di.uniroma1.it.
Name:   ccalcolo.di.uniroma1.it
Address: 151.100.17.39
```

ccalcolo is the canonical (true) name of the **phd** host.

nslookup

Get canonical name of "phd.di.uniroma1.it"

Example 8.

```
nslookup -type=CNAME phd.di.uniroma1.it

Server:  8.8.8.8
Address: 8.8.8.8#53

Non-authoritative answer:
phd.di.uniroma1.it        canonical name = ccalcolo.di.uniroma1.it.
```

ccalcolo is the canonical (true) name of the **phd** host.

nslookup

Get ip of “ccalcolo.di.uniroma1.it”

Example 9.

```
nslookup ccalcolo.di.uniroma1.it
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   ccalcolo.di.uniroma1.it
Address: 151.100.17.39
```

Same IP of “phd.di.uniroma1.it”

Why we want to use an alias and CNAME?

nslookup

Exercises:

use nslookup to find out what are the name servers responsible for the domains:

- . (root)
- it
- uniroma1.it.
- di.uniroma1.it.
- redi.uniroma1.it.

netstat

Command line tool able to display:

- Network Connections
- Routing Tables
- Interface statistics
- etc...

Netstat: network connections

$netstat Shows only established connections

Shows TCP/UDP/TCPv6/UDPv6/UNIX sockets

UNIX sockets are like TCP/UDP connections, but used only for local inter-process communication

$netstat -a Shows also connection in **LISTEN** state, which tipically belongs to a server waiting for clients

Netstat: network connections

$netstat -a

Netstat: network connections

$netstat -a

```
ga0:~ mauropiva$ netstat -a
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  ga0.di.uniroma1..61154 ec2-52-201-150-1.https ESTABLISHED
tcp4       0      0  ga0.di.uniroma1..61153 mil04s28-in-f14..https ESTABLISHED
tcp4       0      0  ga0.di.uniroma1..61151 ec2-52-200-53-11.https ESTABLISHED
tcp4       0      0  ga0.di.uniroma1..61150 ec2-34-227-251-4.https ESTABLISHED
tcp4       0      0  ga0.di.uniroma1..61149 whatsapp-cdn-shv.https ESTABLISHED
tcp4       0      0  ga0.di.uniroma1..61148 ec2-54-208-222-3.https ESTABLISHED
```

Protocol Shows the protocol used by the connection

Netstat: network connections

$netstat -a

```
ga0:~ mauropiva$ netstat -a
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  ga0.di.uniroma1..61154 ec2-52-201-150-1.https ESTABLISHED
tcp4       0      0  ga0.di.uniroma1..61153 mil04s28-in-f14..https ESTABLISHED
tcp4       0      0  ga0.di.uniroma1..61151 ec2-52-200-53-11.https ESTABLISHED
tcp4       0      0  ga0.di.uniroma1..61150 ec2-34-227-251-4.https ESTABLISHED
tcp4       0      0  ga0.di.uniroma1..61149 whatsapp-cdn-shv.https ESTABLISHED
tcp4       0      0  ga0.di.uniroma1..61148 ec2-54-208-222-3.https ESTABLISHED
```

Local & Foreign Address connection endpoints in form addr:port

Wht https instead of 22? Try **$netstat -an**

Netstat: network connections

$netstat -a

```
ga0:~ mauropiva$ netstat -an
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  *.22                   *.*                    LISTEN
tcp4       0      0  151.100.17.107.61302   151.100.17.83.2869     SYN_SENT
tcp4       0      0  151.100.17.107.61300   143.204.15.29.80       ESTABLISHED
tcp4       0      0  *.80                   *.*                    LISTEN
```

Local & Foreign Address

*.22

.

* means “any”

On any interface on port 22

From any client from any port

Netstat: network connections

$netstat -a

```
ga0:~ mauropiva$ netstat -an
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  *.22                   *.*                    LISTEN
tcp4       0      0  151.100.17.107.61302   151.100.17.83.2869     SYN_SENT
tcp4       0      0  151.100.17.107.61300   143.204.15.29.80       ESTABLISHED
tcp4       0      0  *.80                   *.*                    LISTEN
```

State	Example of connection states:
LISTEN	waiting for connections
SYN_SENT	starting a connection
ENSTABLISHED	connection enstablished
CLOSE_WAIT	connection is about to be closed

Netstat: options

$ man netstat

-p (-v on MacOS) shows the name/PID of the process that opened the connection
-t TCP only
-l listening connection only
-4 IPv4 connections only
-n do not resolve ip or port address
-c shows output continuously

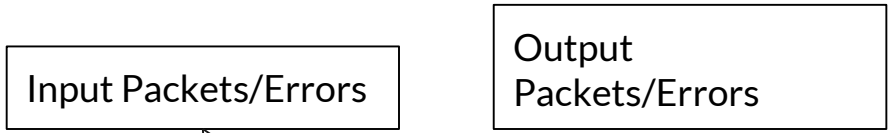
$netstat -ptn4c ?

$netstat -r

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         salaria-gw.di.u 0.0.0.0         UG    100    0        0 enp2s0
151.100.17.0    0.0.0.0         255.255.255.0   U     100    0        0 enp2s0
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 enp2s0
```

Netstat: options

$netstat -i




```
ga0:~ mauropiva$ netstat -in
Name  Mtu   Network       Address            Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16384 <Link#1>                       377285     0   377285     0     0
lo0   16384 127           127.0.0.1        377285     -   377285     -     -
lo0   16384 ::1/128     ::1                377285     -   377285     -     -
lo0   16384 fe80::1%lo0 fe80:1::1          377285     -   377285     -     -
gif0* 1280  <Link#2>                            0     0        0     0     0
stf0* 1280  <Link#3>                            0     0        0     0     0
XHC20 0     <Link#4>                            0     0        0     0     0
en0   1500  <Link#5>    98:e0:d9:96:5b:c5 1567353     0  1309080     0     0
p2p0  2304  <Link#6>    0a:e0:d9:96:5b:c5       0     0        0     0     0
awdl0 1484  <Link#7>    d6:6e:03:02:4c:a0       0     0      537     0     0
awdl0 1484  fe80::d46e: fe80:7::d46e:3ff:       0     -      537     -     -
en1   1500  <Link#8>    9a:00:05:3a:cf:10       0     0        0     0     0
bridg 1500  <Link#9>    9a:00:05:3a:cf:10       0     0        1     0     0
utun0 2000  <Link#10>                           0     0        2     0     0
utun0 2000  fe80::2787: fe80:a::2787:89d5       0     -        2     -     -
utun1 1380  <Link#11>                       1416     0     1575     0     0
```

```
utun1  1380   fe80::4d5:5  fe80:b::4d5:5aa7:     1416     -     1575     -     -
en4    1500   <Link#12>    00:50:b6:22:0b:c1   576535     1   179626     0     0
```

Netstat

```
ga0:~ mauropiva$ netstat -v
Active Internet connections
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)     rhiwat shiwat    pid   epid
tcp4       0      0  ga0.di.uniroma1..60472 178-85-118-227.d.37914 SYN_SENT    131072 131072  43150      0
tcp4       0      0  ga0.di.uniroma1..60470 softbank12616306.41181 SYN_SENT    131072 131072  43150      0
tcp4       0     74  ga0.di.uniroma1..60467 h174.68.184.173..33868 ESTABLISHED 1575000 132132  43150      0
tcp4       0      0  ga0.di.uniroma1..60466 d173-180-240-184.46563 SYN_SENT    131072 131072  43150      0
tcp4       0      0  ga0.di.uniroma1..60465 mobile-166-175-6.29256 SYN_SENT    131072 131072  43150      0
tcp4       0      0  ga0.di.uniroma1..60464 165.56.53.48.49066     SYN_SENT    131072 131072  43150      0
tcp4       0      0  ga0.di.uniroma1..60462 static-145.130.2.27135 SYN_SENT    131072 131072  43150      0
tcp4       0      0  ga0.di.uniroma1..60442 thisis.feralhost.50294 FIN_WAIT_2  250000 131400  43150      0
tcp4       0      0  ga0.di.uniroma1..60430 47.75.67.248.jacobus-l SYN_SENT    131072 131072  43150      0
tcp4       0      0  ga0.di.uniroma1..60423 185.217.0.78.http      SYN_SENT    131072 131072  43150      0
tcp4       0      0  ga0.di.uniroma1..60422 172.ip-51-68-122.https SYN_SENT    131072 131072  43150      0
tcp4       0    401  ga0.di.uniroma1..60421 acg.loli.http-alt      ESTABLISHED 131860 131860  43150      0
```

Differences?

Netstat: BitTorrent

```
ga0:~ mauropiva$ netstat -v
Active Internet connections
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)     rhiwat shiwat    pid   epid
tcp4       0      0  ga0.di.uniroma1..60472 178-85-118-227.d.37914 SYN_SENT    131072 131072  43150      0
tcp4       0      0  ga0.di.uniroma1..60470 softbank12616306.41181 SYN_SENT    131072 131072  43150      0
tcp4       0     74  ga0.di.uniroma1..60467 h174.68.184.173..33868 ESTABLISHED 1575000 132132  43150      0
tcp4       0      0  ga0.di.uniroma1..60466 d173-180-240-184.46563 SYN_SENT    131072 131072  43150      0
tcp4       0      0  ga0.di.uniroma1..60465 mobile-166-175-6.29256 SYN_SENT    131072 131072  43150      0
tcp4       0      0  ga0.di.uniroma1..60464 165.56.53.48.49066     SYN_SENT    131072 131072  43150      0
tcp4       0      0  ga0.di.uniroma1..60462 static-145.130.2.27135 SYN_SENT    131072 131072  43150      0
tcp4       0      0  ga0.di.uniroma1..60442 thisis.feralhost.50294 FIN_WAIT_2  250000 131400  43150      0
tcp4       0      0  ga0.di.uniroma1..60430 47.75.67.248.jacobus-l SYN_SENT    131072 131072  43150      0
tcp4       0      0  ga0.di.uniroma1..60423 185.217.0.78.http      SYN_SENT    131072 131072  43150      0
tcp4       0      0  ga0.di.uniroma1..60422 172.ip-51-68-122.https SYN_SENT    131072 131072  43150      0
tcp4       0    401  ga0.di.uniroma1..60421 acg.loli.http-alt      ESTABLISHED 131860 131860  43150      0
```

$netstat -v shows the process name/pid

Netstat: BitTorrent

```
ga0:~ mauropiva$ netstat -v
Active Internet connections
Proto Recv-Q Send-Q  Local Address          Foreign Address
udp4       0      0  *.plysrv-https         *.*
udp4       0      0  *.ssdp                 *.*
udp4       0      0  ga0.di.uniroma1..62594 *.*
udp4       0      0  localhost.51878        *.*
udp6       0      0  *.41822
udp4       0      0  *.41822                *.*
udp4       0      0  *.*                    *.*
```

All connections with the same PID of BitTorrent

Netstat: BitTorrent

```
ga0:~ mauropiva$ netstat -v
Active Internet connections
Proto Recv-Q Send-Q  Local Address          Foreign Address
udp4       0      0  *.plysrv-https         *.*
udp4       0      0  *.ssdp                 *.*
udp4       0      0  ga0.di.uniroma1..62594 *.*
udp4       0      0  localhost.51878        *.*
udp6       0      0  *.41822
udp4       0      0  *.41822                *.*
```

BitTorrent is listening on a number of different ports
Line #3 and #4 accepts only connections from localhost

netcat

The easiest tool for networking

Allows to read and write data directly from a socket, both in TCP and UDP

Helpful for better understanding how networks work and debug

Open two terminals:
T1: **$nc -l -p 12345**
T2: **$nc localhost 12345**
Write something
What's happening?

netcat

With netcat is also possible to transfer files:

T1 : **$nc -l -p 12345**

Activate netcat acting as a server (-l, listening) on port(-p) 12345

T2: **$nc a.b.c.d 12345**

Activate netcat as a client, connect to a.b.c.d (the IP address) on port 12345, and send data

netcat

With netcat is also possible to transfer files:

Receiver : **$nc -l -p 12345 > file.txt**

Activate netcat acting as a server (-l, listening) on port(-p) 12345 and write (> is a unix command which redirects the standard output to something, in this case file.txt)

Sender: **$nc a.b.c.d 12345 < file.txt**

Activate netcat as a client, connect to a.b.c.d (the IP address) on port 12345, and send the data contained in file.txt

netcat

Try:

1) **$sudo nc -l 80**

2) Open your browser and visit localhost:80

Netcat (listen)

Try:

1) **$sudo nc -l 80**

2) Open your browser and visit localhost:80 (or localhost?)

Your browser is sending a GET request to localhost:80, and netcat is attached to such port

```
ga0:tmp mauropiva$ sudo nc -l 80
Password:
GET / HTTP/1.1
Host: localhost
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 1(
Accept: text/html,application/xhtml+xml,application/)
Accept-Encoding: gzip, deflate, br
Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7
```

Netcat (listen)

A simple web server for bash:

While true; do { echo “HTTP/1.1 200 OK”; echo ; echo “<html> Hello world </html>”;} | nc -l -p 8080; done

Even if it is not following the HTTP protocol (e.g. no header in the response) the browser is able to display this page

Netcat (client)

Try to connect to a server with nc

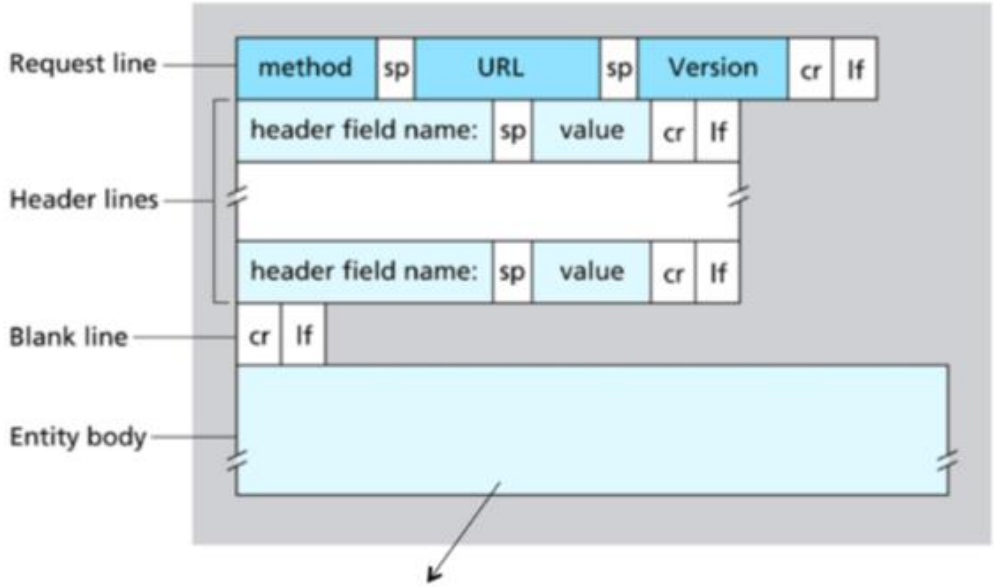
1) **$ nc google.it 80**
2) Send ????

Netcat (client)

Try to connect to a server with nc

1) **$ nc google.it 8**
2) Send ????

Formato generale dei messaggi di richiesta HTTP:



Netcat (client)

Try to connect to a server with nc

1) **$ nc google.it 80**

2) Send

GET / HTTP/1.1

two new line

Why the two new line are required?

Netcat (client)

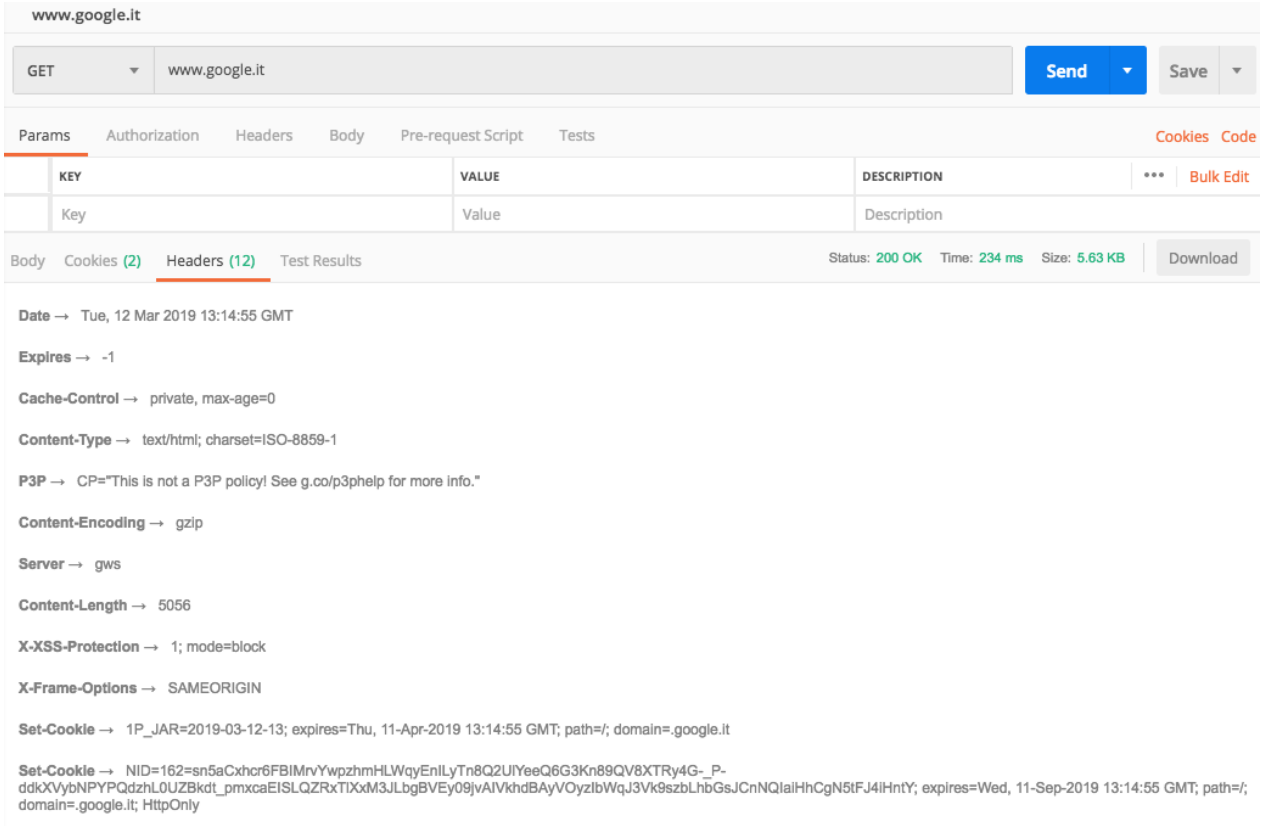
In the same way, is possible to analyze the behaviour of a number of things:

Mail Server
DNS
FTP
SSH

It may seem easy, but protocols should always follows all the requirements described in their RFC.

postman

POSTman is a graphical
tool useful for the creation and debug of HTTP requests



What is it?

Why is it useful?

Wireshark

"Tell me and I forget, teach me and I may remember, involve me and I learn."

— Benjamin Franklin

Wireshark

- Wireshark is a software **(packet analyzer)** that allows to **monitor** the **incoming/ outgoing network frames.**
  - It captures **a copy** of the frames
  - Does not inject traffic
- It can expose the **whole** content of each frame (i.e., the whole protocol stack)
- very useful for:
  - learning how TCP/IP works
  - network administrators
- it is **not** a security tool
- Wireshark is a rather complex and powerful tool, whose complete set of functionalities cannot be discussed with a single lecture
  - we will cover its basics only
- other packet analyzers:
  - **tcpdump**
  - **tshark**

Wireshark

Again:

- to **install** Wireshark on Windows or OSX, go to *http://www.wireshark.org*

On a Debian-based GNU/Linux distribution (e.g., Ubuntu, Linux Mint.. and Debian), just open a terminal window and type:

- ***sudo add-apt-repository ppa:wireshark-dev/stable***
- ***sudo apt-get update***
- ***sudo apt-get install wireshark***

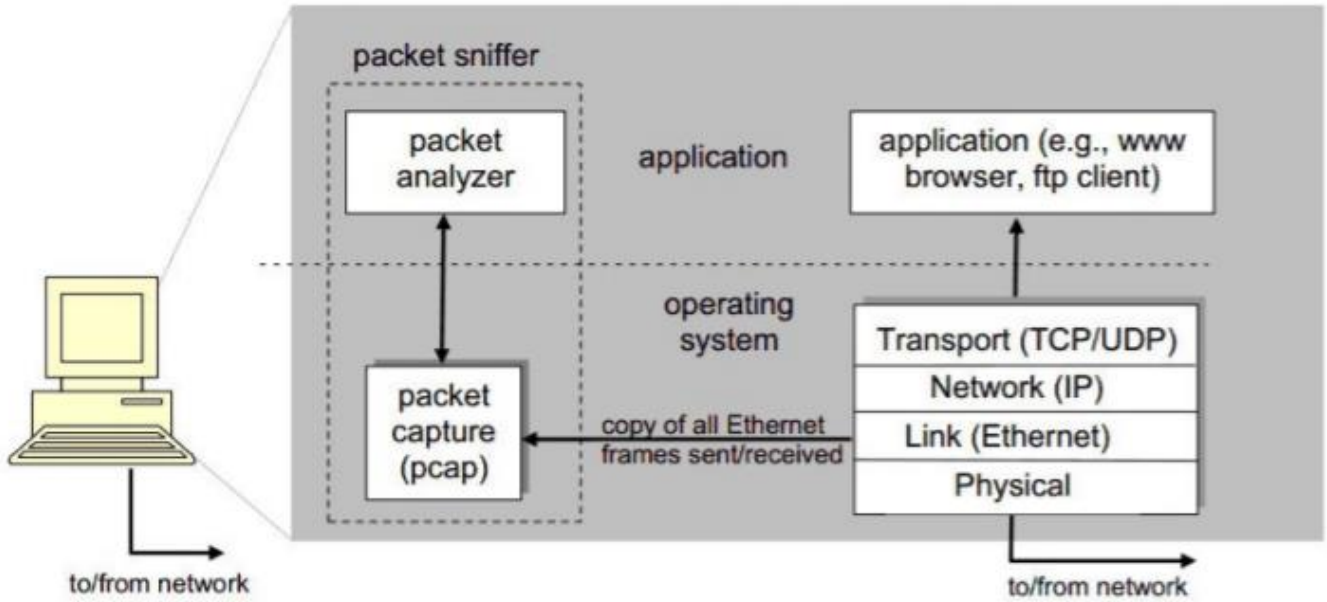
When the installation is complete, just type on a terminal:

- **wireshark** (you may need **sudo)**

Or run it from the applications menu.

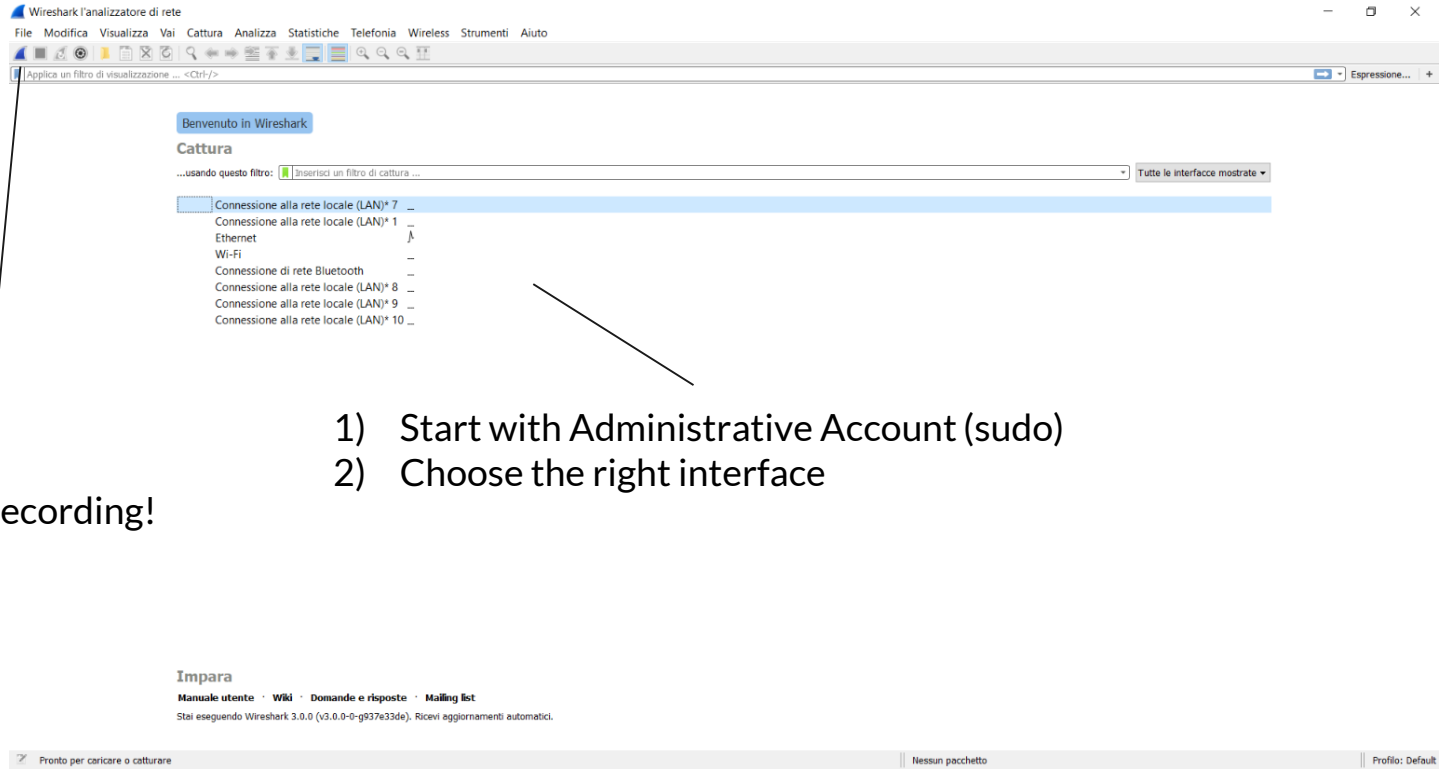
Useful links: http://wiki.wireshark.org/CaptureSetup
https://www.wireshark.org/docs/wsug_html_chunked/
http://wiki.wireshark.org/SampleCaptures

Wireshark



pcap: Packet capture library

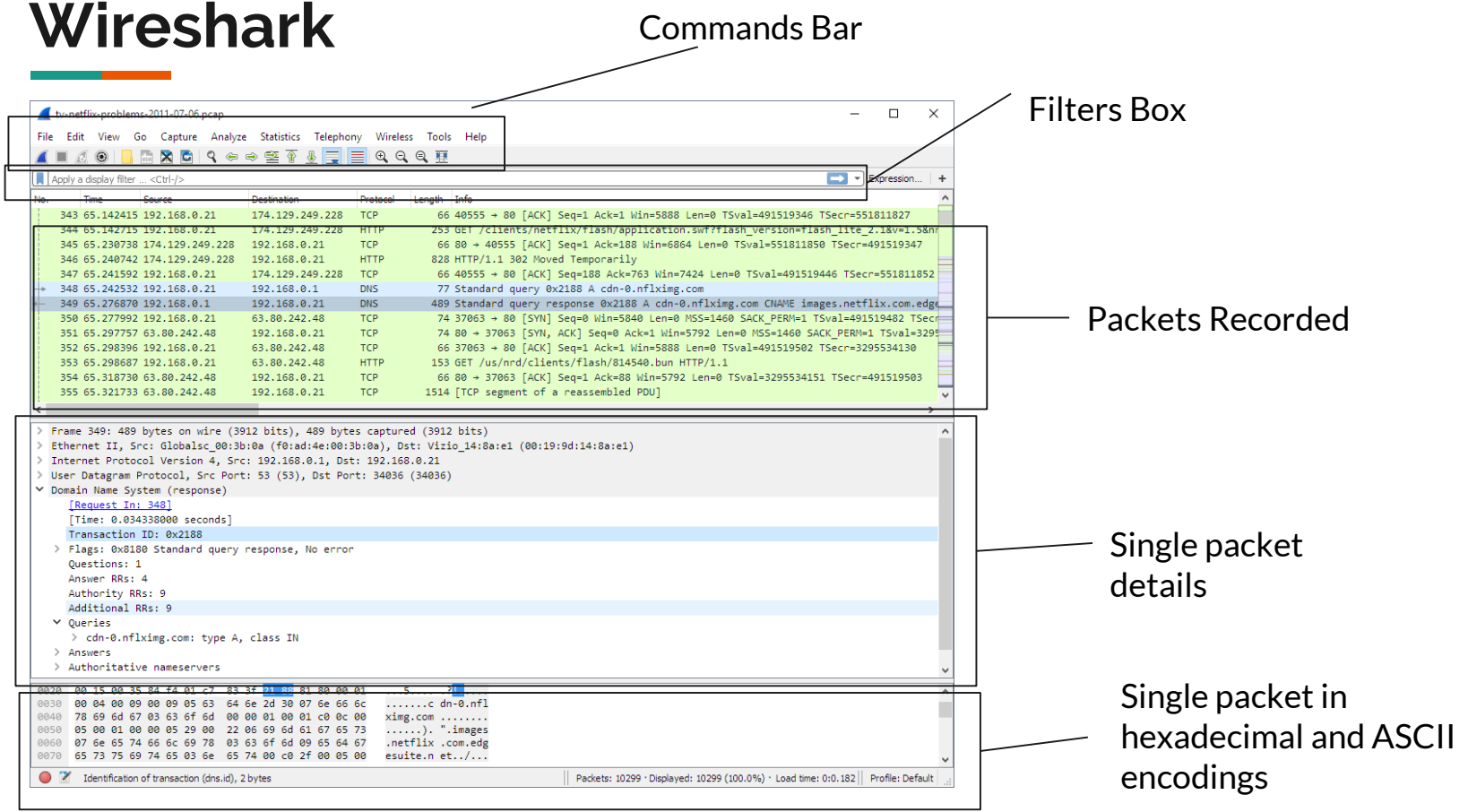
Wireshark



1) Start with Administrative Account (sudo)
2) Choose the right interface

3) Start recording!

Wireshark


Commands Bar
Filters Box
Packets Recorded
Single packet details
Single packet in hexadecimal and ASCII encodings

Wireshark: SSH

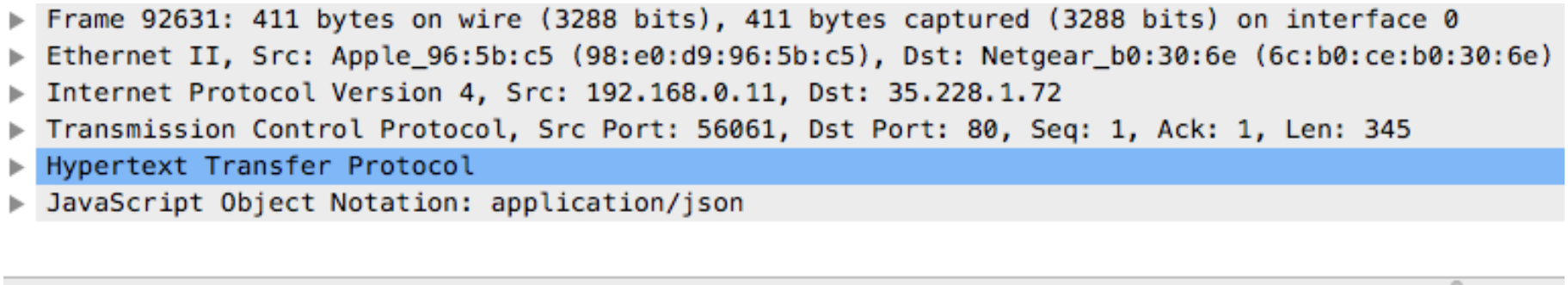
263…	9…	192.168.0.11	35.228.1.72	TCP	78	56029 → 22 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=862878559 TSecr=0 SACK_PERM=1
264…	9…	35.228.1.72	192.168.0.11	TCP	74	22 → 56029 [SYN, ACK] Seq=0 Ack=1 Win=28160 Len=0 MSS=1418 SACK_PERM=1 TSval=4076312853 TSec
264…	9…	192.168.0.11	35.228.1.72	TCP	66	56029 → 22 [ACK] Seq=1 Ack=1 Win=132160 Len=0 TSval=862878641 TSecr=4076312853
264…	9…	192.168.0.11	35.228.1.72	SSHv2	87	Client: Protocol (SSH-2.0-OpenSSH_7.6)
265…	9…	35.228.1.72	192.168.0.11	TCP	66	22 → 56029 [ACK] Seq=1 Ack=22 Win=28160 Len=0 TSval=4076312957 TSecr=862878645
265…	9…	35.228.1.72	192.168.0.11	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3)
265…	9…	192.168.0.11	35.228.1.72	TCP	66	56029 → 22 [ACK] Seq=22 Ack=42 Win=132096 Len=0 TSval=862878734 TSecr=4076312960
265…	9…	192.168.0.11	35.228.1.72	SSHv2	1426	Client: Key Exchange Init
265…	9…	35.228.1.72	192.168.0.11	SSHv2	1146	Server: Key Exchange Init
265…	9…	192.168.0.11	35.228.1.72	TCP	66	56029 → 22 [ACK] Seq=1382 Ack=1122 Win=131040 Len=0 TSval=862878822 TSecr=4076313060
265…	9…	35.228.1.72	192.168.0.11	TCP	66	22 → 56029 [ACK] Seq=1122 Ack=1382 Win=30976 Len=0 TSval=4076313137 TSecr=862878745
265…	9…	192.168.0.11	35.228.1.72	SSHv2	114	Client: Diffie-Hellman Key Exchange Init
266…	9…	35.228.1.72	192.168.0.11	TCP	66	22 → 56029 [ACK] Seq=1122 Ack=1430 Win=30976 Len=0 TSval=4076313238 TSecr=862878892
266…	9…	35.228.1.72	192.168.0.11	SSHv2	518	Server: Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=172)
266…	9…	192.168.0.11	35.228.1.72	TCP	66	56029 → 22 [ACK] Seq=1430 Ack=1574 Win=130592 Len=0 TSval=862878991 TSecr=4076313242
266…	9…	192.168.0.11	35.228.1.72	SSHv2	82	Client: New Keys
267…	9…	35.228.1.72	192.168.0.11	TCP	66	22 → 56029 [ACK] Seq=1574 Ack=1446 Win=30976 Len=0 TSval=4076313421 TSecr=862879002
267…	9…	192.168.0.11	35.228.1.72	SSHv2	110	Client: Encrypted packet (len=44)
267…	9…	35.228.1.72	192.168.0.11	TCP	66	22 → 56029 [ACK] Seq=1574 Ack=1490 Win=30976 Len=0 TSval=4076313522 TSecr=862879134
267…	9…	35.228.1.72	192.168.0.11	SSHv2	110	Server: Encrypted packet (len=44)

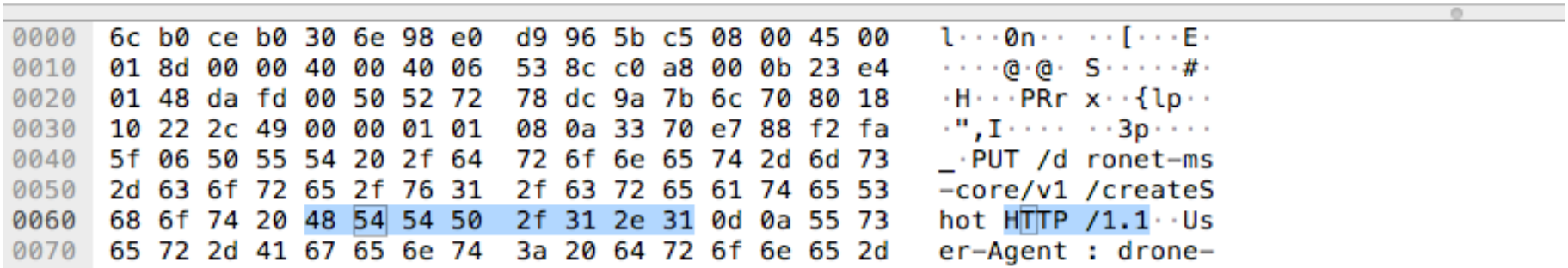
Connessione tra un Client (192.168.0.11) ed un server (35.228.1.72) mediante SSH

Wireshark: Postman

926…	2…	35.228.1.72	192.168.0.11	TCP	74	80 → 56061 [SYN, ACK] Seq=0 Ack=1 Win=28160 Len=0 MSS=1418 SACK_P
926…	2…	192.168.0.11	35.228.1.72	TCP	66	56061 → 80 [ACK] Seq=1 Ack=1 Win=132160 Len=0 TSval=863037319 TSe
926…	2…	192.168.0.11	35.228.1.72	HTTP	411	PUT /dronet-ms-core/v1/createShot HTTP/1.1 (application/json)
926…	2…	35.228.1.72	192.168.0.11	TCP	66	80 → 56061 [ACK] Seq=1 Ack=346 Win=29312 Len=0 TSval=4076494701 T

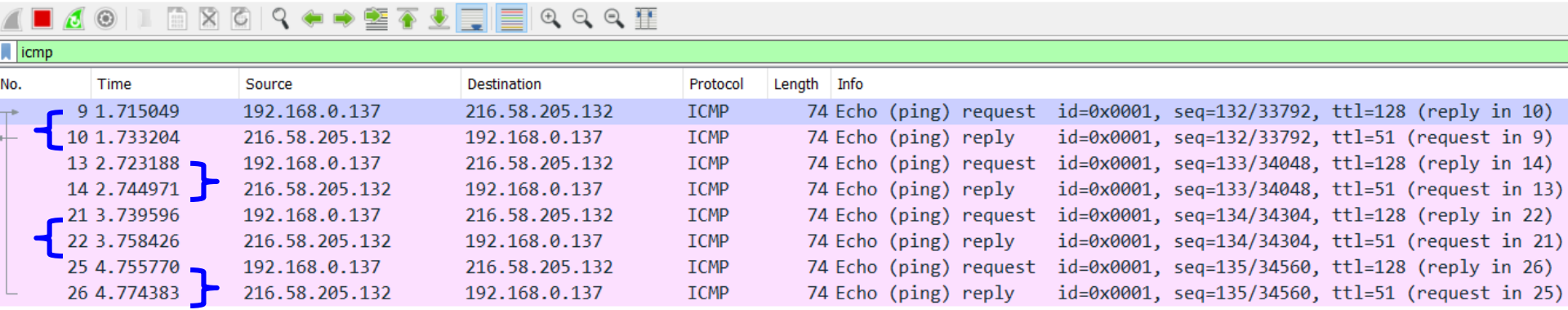
Wireshark: Postman





Wireshark: ping

ping www.google.com (4 packets - Ip (216.58.205.132))

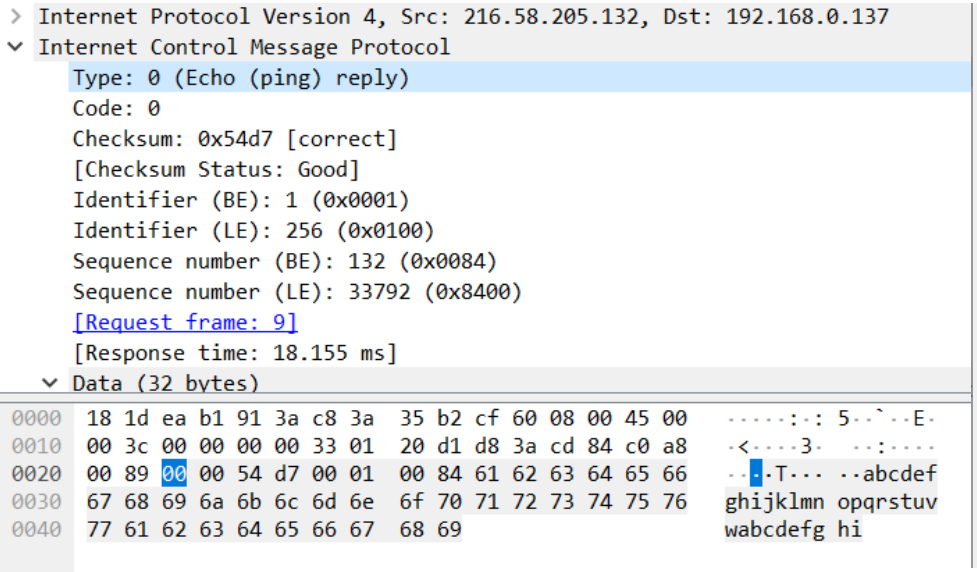


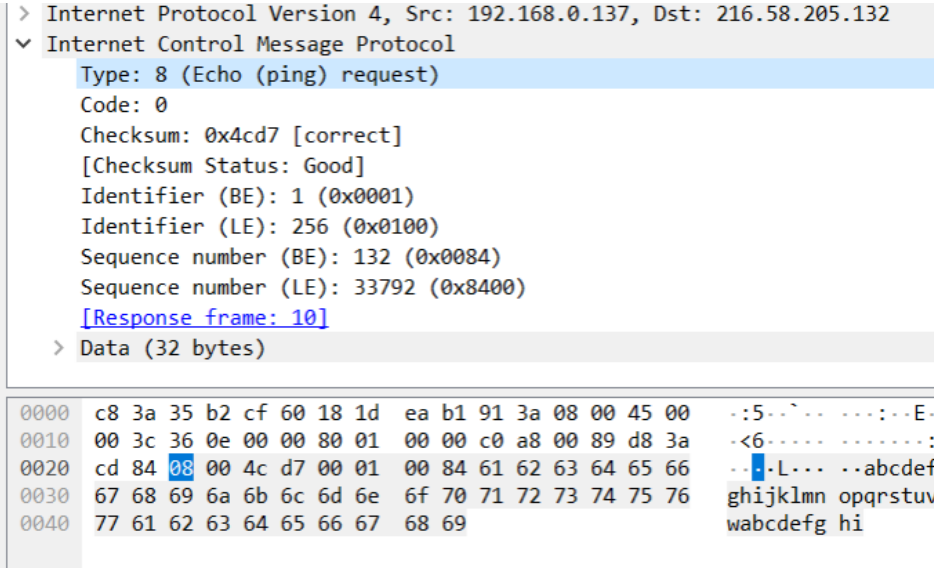
icmp

No.	Time	Source	Destination	Protocol	Length	Info
9	1.715049	192.168.0.137	216.58.205.132	ICMP	74	Echo (ping) request id=0x0001, seq=132/33792, ttl=128 (reply in 10)
10	1.733204	216.58.205.132	192.168.0.137	ICMP	74	Echo (ping) reply id=0x0001, seq=132/33792, ttl=51 (request in 9)
13	2.723188	192.168.0.137	216.58.205.132	ICMP	74	Echo (ping) request id=0x0001, seq=133/34048, ttl=128 (reply in 14)
14	2.744971	216.58.205.132	192.168.0.137	ICMP	74	Echo (ping) reply id=0x0001, seq=133/34048, ttl=51 (request in 13)
21	3.739596	192.168.0.137	216.58.205.132	ICMP	74	Echo (ping) request id=0x0001, seq=134/34304, ttl=128 (reply in 22)
22	3.758426	216.58.205.132	192.168.0.137	ICMP	74	Echo (ping) reply id=0x0001, seq=134/34304, ttl=51 (request in 21)
25	4.755770	192.168.0.137	216.58.205.132	ICMP	74	Echo (ping) request id=0x0001, seq=135/34560, ttl=128 (reply in 26)
26	4.774383	216.58.205.132	192.168.0.137	ICMP	74	Echo (ping) reply id=0x0001, seq=135/34560, ttl=51 (request in 25)

Wireshark: ping

ping www.google.com (4 packets - Ip (216.58.205.132))





Wireshark: nslookup

nslookup www.google.com

```
> Ethernet II, Src: IntelCor_b1:91:3a (18:1d:ea:b1:91:3a), Dst: HuaweiTe_dd
> Internet Protocol Version 4, Src: 192.168.43.71, Dst: 192.168.43.1
> User Datagram Protocol, Src Port: 57351, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0x0002
  v Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    > www.google.com: type A, class IN
    [Response In: 902]
```

```
0000  10 44 00 dd 54 80 18 1d  ea b1 91 3a 08 00 45 00   ·D··T··· ···:··E·
0010  00 3c 3f 01 00 00 80 11  00 00 c0 a8 2b 47 c0 a8   ·<?····· ····+G··
0020  2b 01 e0 07 00 35 00 28  d7 d2 00 02 01 00 00 01   +····5·( ········
0030  00 00 00 00 00 00 03 77  77 77 06 67 6f 6f 67 6c   ·······w ww·googl
0040  65 03 63 6f 6d 00 00 01  00 01                     e·com··· ··
```

```
> Frame 902: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface
> Ethernet II, Src: HuaweiTe_dd:54:80 (10:44:00:dd:54:80), Dst: IntelCor_b1:91:3a (1
> Internet Protocol Version 4, Src: 192.168.43.1, Dst: 192.168.43.71
> User Datagram Protocol, Src Port: 53, Dst Port: 57351
v Domain Name System (response)
    Transaction ID: 0x0002
  v Flags: 0x8180 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    > www.google.com: type A, class IN
  v Answers
    > www.google.com: type A, class IN, addr 216.58.205.196
    [Request In: 901]
    [Time: 0.004356000 seconds]
```

```
0000  18 1d ea b1 91 3a 10 44  00 dd 54 80 08 00 45 00   ·····:·D ··T···E·
0010  00 4c 8f ab 40 00 40 11  d3 5c c0 a8 2b 01 c0 a8   ·L··@·@· ·\··+···
0020  2b 47 00 35 e0 07 00 38  d2 0e 00 02 81 80 00 01   +G·5···8 ········
0030  00 01 00 00 00 00 03 77  77 77 06 67 6f 6f 67 6c   ·······w ww·googl
0040  65 03 63 6f 6d 00 00 01  00 01 c0 0c 00 01 00 01   e·com··· ········
0050  00 00 00 5c 00 04 d8 3a  cd c4                     ···\···: ··
```

Wireshark: traceroute

```
traceroute to www.facebook.com (31.13.86.36), 30 hops max, 60 byte packets
 1  _gateway (192.168.43.1)  5.120 ms  6.659 ms  6.673 ms
 2  * * *
 3  172.31.9.101 (172.31.9.101)  53.067 ms  53.226 ms 172.31.9.69 (172.31.9.69)  54.160 ms
 4  172.30.32.131 (172.30.32.131)  58.303 ms  58.444 ms  60.853 ms
```

3952	144.515…	192.168.43.71	31.13.86.36	UDP	74	45479 → 33446 Len=32
3953	144.515…	192.168.43.71	31.13.86.36	UDP	74	41937 → 33447 Len=32
3954	144.515…	192.168.43.71	31.13.86.36	UDP	74	55257 → 33448 Len=32
3955	144.515…	192.168.43.71	31.13.86.36	UDP	74	35392 → 33449 Len=32
3956	144.520…	192.168.43.1	192.168.43.71	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
3958	144.522…	192.168.43.1	192.168.43.71	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
3959	144.522…	192.168.43.1	192.168.43.71	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
3961	144.544…	192.168.43.71	31.13.86.36	UDP	74	34766 → 33450 Len=32
3962	144.544…	192.168.43.71	31.13.86.36	UDP	74	38087 → 33451 Len=32
3963	144.544…	192.168.43.71	31.13.86.36	UDP	74	34260 → 33452 Len=32
3964	144.568…	172.31.9.101	192.168.43.71	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
3965	144.568…	172.31.9.101	192.168.43.71	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
3966	144.568…	192.168.43.71	31.13.86.36	UDP	74	48904 → 33453 Len=32
3967	144.569…	192.168.43.71	31.13.86.36	UDP	74	45587 → 33454 Len=32
3968	144.569…	172.31.9.69	192.168.43.71	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
3969	144.570…	192.168.43.71	31.13.86.36	UDP	74	35128 → 33455 Len=32
3970	144.571…	172.19.202.0	192.168.43.71	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
3971	144.572…	192.168.43.71	31.13.86.36	UDP	74	52255 → 33456 Len=32
3972	144.572…	172.19.202.0	192.168.43.71	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)

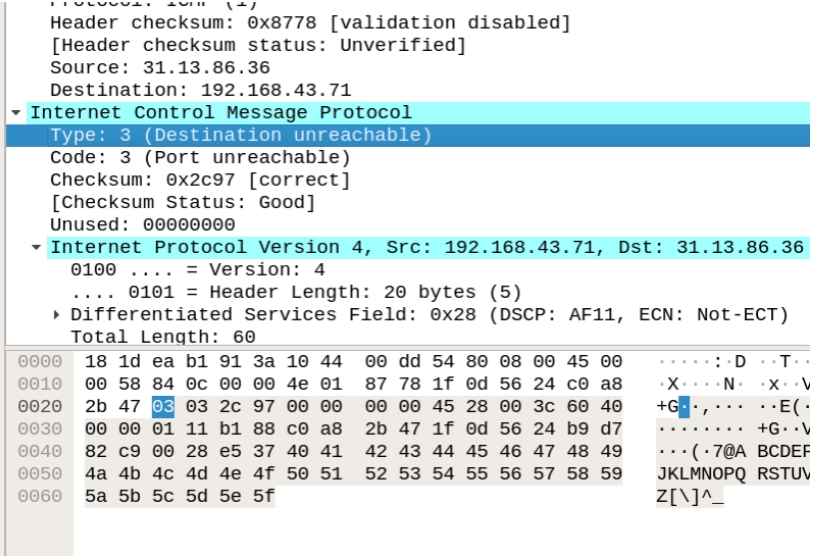
Wireshark: traceroute

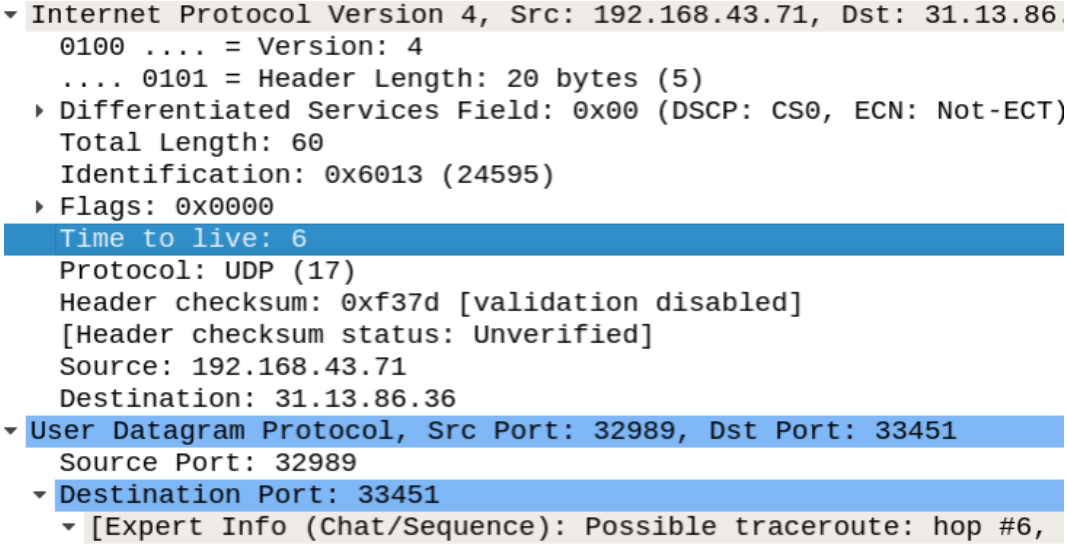
```
traceroute to www.facebook.com (31.13.86.36), 30 hops max, 60 byte packets
 1  _gateway (192.168.43.1)  5.120 ms  6.659 ms  6.673 ms
 ..........................
16  edge-star-mini-shv-01-mxp1.facebook.com (31.13.86.36)  40.198 ms  42.462 ms  43.656 ms
```

102	0.235814146	192.168.43.71	31.13.86.36	UDP	74	57532 → 33489 Len=32
103	0.244241650	31.13.86.36	192.168.43.71	ICMP	102	Destination unreachable (Port unreachable)
104	0.245282816	31.13.86.36	192.168.43.71	ICMP	102	Destination unreachable (Port unreachable)
105	0.246370923	31.13.86.36	192.168.43.71	ICMP	102	Destination unreachable (Port unreachable)
106	0.247397224	31.13.86.36	192.168.43.71	ICMP	102	Destination unreachable (Port unreachable)

Wireshark: traceroute

```
traceroute to www.facebook.com (31.13.86.36), 30 hops max, 60 byte packets
 1  _gateway (192.168.43.1)  5.120 ms  6.659 ms  6.673 ms
 .....................
16  edge-star-mini-shv-01-mxp1.facebook.com (31.13.86.36)  40.198 ms  42.462 ms  43.656 ms
```





Wireshark - Exercise

Close your web-browser

Open Wireshark and start it

Open a browser and a web-page (http://testreti.ddns.net/)

Pause Wireshark

Identify and analyze DNS packets (tip: **filter dns**) -> Find the IP of testireti.ddns.net

Identify and analyze HTTP packets (tip: **filter http**) -> Find the Secret Header

Thanks!