

LINUX NETWORK TOOLS

Let's see some real traffic...

```
File Edit View Terminal Tabs Help
17:13:21.395966 ip 10.10.2.17.36115 > 74.125.19.19.80: http http.method:POST htt
p.server:mail.google.com 1448
17:13:21.395982 ip 10.10.2.17.36115 > 74.125.19.19.80: http 204
17:13:21.396061 ip 10.10.2.17.36115 > 74.125.19.19.80: http http.mime_type:multi
part/form-data 1448
17:13:21.396636 ip 74.125.19.19.80 > 10.10.2.17.36115: http 0
17:13:21.396654 ip 74.125.19.19.80 > 10.10.2.17.36115: http 0
17:13:21.396662 ip 10.10.2.17.36115 > 74.125.19.19.80: http 404
17:13:21.396723 ip 74.125.19.19.80 > 10.10.2.17.36115: http 0
17:13:21.396993 ip 74.125.19.19.80 > 10.10.2.17.36115: http 0
17:13:22.159636 ip 74.125.19.19.80 > 10.10.2.17.36115: http http.mime_type:text/
html 1328
17:13:22.159664 ip 10.10.2.17.36115 > 74.125.19.19.80: http 0
17:13:37.903428 ip 10.10.2.17.36115 > 74.125.19.19.80: http http.method:POST htt
p.server:mail.google.com http.mime_type:application/x-www-form-urlencoded 1448
17:13:37.903445 ip 10.10.2.17.36115 > 74.125.19.19.80: http 241
17:13:37.904146 ip 74.125.19.19.80 > 10.10.2.17.36115: http 0
17:13:37.904172 ip 74.125.19.19.80 > 10.10.2.17.36115: http 0
17:13:37.904183 ip 10.10.2.17.36115 > 74.125.19.19.80: http 53
17:13:37.904478 ip 74.125.19.19.80 > 10.10.2.17.36115: http 0
17:13:38.265800 ip 74.125.19.19.80 > 10.10.2.17.36115: http http.mime_type:text/
html 349
17:13:38.265826 ip 10.10.2.17.36115 > 74.125.19.19.80: http 0
```

tcpdump

wireshark

The image shows the Wireshark network protocol analyzer interface. The main pane displays a list of captured packets with columns for No., Time, Source, Destination, Protocol, and Info. The selected packet (No. 1) is expanded to show its details, including Ethernet II, Internet Protocol Version 4, and Hypertext Transfer Protocol. The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Info
11	15.047027	208.67.222.222	192.168.1.101	DNS	Standard query response
12	15.647269	192.168.1.101	208.67.222.222	DNS	Standard query A www.
13	15.937059	208.67.222.222	192.168.1.101	DNS	Standard query response
14	15.937457	192.168.1.101	75.126.43.232	TCP	45861 > www [SYN] Seq
15	16.314591	75.126.43.232	192.168.1.101	TCP	www > 45861 [SYN, ACK]
16	16.314665	192.168.1.101	75.126.43.232	TCP	45861 > www [ACK] Seq
17	16.314984	192.168.1.101	75.126.43.232	TCP	[TCP segment of a rea
18	16.315020	192.168.1.101	75.126.43.232	TCP	[TCP segment of a rea
19	16.724366	75.126.43.232	192.168.1.101	TCP	www > 45861 [ACK] Seq
20	16.732070	75.126.43.232	192.168.1.101	TCP	www > 45861 [ACK] Seq
21	18.072290	192.168.1.101	208.67.222.222	DNS	Standard query A www.
22	18.360176	208.67.222.222	192.168.1.101	DNS	Standard query response
23	18.445066	192.168.1.101	208.67.222.222	DNS	Standard query AAAA w
24	18.448504	192.168.1.101	208.67.222.222	DNS	Standard query A www.

Frame 1 (42 bytes on wire (42 bytes captured) on interface eth0):

Ethernet II, Src: D-Link (08:00:0c:27:8c:44), Dest: Cisco-Li (08:00:0c:27:8c:44)

0000 00 18 39 6a c6 8b 00 17 9a 0a f6 44 08 06 00 01 ..9j.....D....

0010 08 00 06 04 00 01 00 17 9a 0a f6 44 c0 a8 01 65D...e

0020 00 00 00 00 00 00 c0 a8 01 01D....

Frame (frame), 42 bytes P: 582 D: 582 M: 0 Drops: 0

tcpdump

the command line network analyzer

For documentation:

- `man tcpdump` (program usage)
- <http://danielmiessler.com/study/tcpdump/> (nice tutorial)

Essentials:

- Capture all packets on all interfaces and dump the entire packet:
`tcpdump -i any -X`
- Capture all packets on all interfaces and don't convert addresses to names:
`tcpdump -i any -n`
- Capture all packets on eth0 and save the trace on file (the whole packets...):
`tcpdump -i eth0 -w file -s0`
- Capture 10 packets on eth0 to/from \$ADDR:
`tcpdump -i eth0 -c 10 host $ADDR`
- Capture all TCP packets to/from port 80 on eth0:
`tcpdump -i eth0 tcp port 80`
- Capture all packets with destination or source address != \$ADDR and port in the range [10000:20000]:
`tcpdump -i eth0 host not $ADDR portrange 10000-20000`

tcpdump output format

Normal output

```
0 packets dropped by kernel
root@marlon-vmxnb:/home/marlon/Src/netgroup# tcpdump -ni eth0 host 8.8.8.8
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
15:50:44.913843 IP 172.16.166.152 > 8.8.8.8: ICMP echo request, id 25220, seq 1, length 64
15:50:44.936668 IP 8.8.8.8 > 172.16.166.152: ICMP echo reply, id 25220, seq 1, length 64
```

Verbose output

```
0 packets dropped by kernel
root@marlon-vmxnb:/home/marlon/Src/netgroup# tcpdump -nvvvi eth0 host 8.8.8.8
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
15:51:05.529625 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)
    172.16.166.152 > 8.8.8.8: ICMP echo request, id 25250, seq 1, length 64
15:51:05.554011 IP (tos 0x0, ttl 128, id 745, offset 0, flags [none], proto ICMP (1), length 84)
    8.8.8.8 > 172.16.166.152: ICMP echo reply, id 25250, seq 1, length 64
```

tcpdump output format

Packet content in HEX and ASCII

```
root@marlon-vmxnb: /home/marlon/Src/netgroup# tcpdump -nXi eth0 host 8.8.8.8
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
15:51:28.311102 IP 172.16.166.152 > 8.8.8.8: ICMP echo request, id 25254, seq 1, length 64
    0x0000:  4500 0054 0000 4000 4001 d7f0 ac10 a698  E..T..@.@.....
    0x0010:  0808 0808 0800 57ba 62a6 0001 f08c 4f4f  .....W.b.....00
    0x0020:  0ebf 0400 0809 0a0b 0c0d 0e0f 1011 1213  .....
    0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223  .....!"#
    0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  $%&'()*+,-./0123
    0x0050:  3435 3637 4567
15:51:28.335982 IP 8.8.8.8 > 172.16.166.152: ICMP echo reply, id 25254, seq 1, length 64
    0x0000:  4500 0054 02eb 0000 8001 d505 0808 0808  E..T.....
    0x0010:  ac10 a698 0000 5fba 62a6 0001 f08c 4f4f  ....._b.....00
    0x0020:  0ebf 0400 0809 0a0b 0c0d 0e0f 1011 1213  .....
    0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223  .....!"#
    0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  $%&'()*+,-./0123
    0x0050:  3435 3637 4567
```

tcpdump advanced filtering

- `man pcap-filter` (filter syntax details)
- pcap filter primitives include
 - `host, dst host, src host`
 - `port, dst port, src port`
 - `ether host, ether dst, ether src`
 - `net, dst net, src net`
 - `portrange, dst portrange, src portrange`
 - `less, greater`
 - `ip proto, ip6 proto, ether proto`
 - `ip broadcast, ip multicast`
 - `ip, ip6, arp, tcp, udp, icmp`
 - `ifname`
 - `proto [expr : size]`
 - `ip[16:4] = 0xffffffff` → DEST BROADCAST IP PACKET
- Example:
 - `tcpdump -ni any "ip[12:4] = 0xac10a69c"`

NERD QUIZ



What do they mean?

- (1) `ether[0] & 1 != 0`
- (2) `ip[0] & 0xf != 5`
- (3) `ip[6:2] & 0x1fff = 0`

Are you sure?

Shall we light them?

Solutions



```
ether[0] & 1 != 0      (ethernet multicast/broadcast packet)
ip[0] & 0xf != 5      (ip packets with option)
ip[6:2] & 0x1fff = 0  (ip un-fragmented packets or frag0)
```


Wireshark

- Wireshark is a graphical packet analyzer
- Like `tcpdump` can analyze live streams or files
- It's compatible with `tcpdump` (pcap format) traces
- It provides additional features:
 - Better protocol parsing
 - Statistics tool
 - Exporting
 - Better Filtering (different syntax)
 - Can be extended to understand proprietary protocol

Wireshark

The screenshot shows the Wireshark 1.6.2 interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Tools, Internals, and Help. Below the menu is a toolbar with various icons for file operations, capture, and analysis. The main display area is divided into three panes:

- Filter:** A search bar with the text "Expression... Clear Apply".
- Packet List:** A table showing captured packets. The selected packet (No. 103) is highlighted in blue.
- Packet Details:** A tree view showing the structure of the selected packet, including Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol.
- Packet Bytes:** A hex dump of the packet data with its ASCII representation.

No.	Time	Source	Destination	Protocol	Length	Info
101	95.420197	172.16.166.147	173.194.35.24	TCP	54	35976 > http [ACK] Seq=1963 Ack=6954 Win=29200 Len=0
102	95.420219	173.194.35.24	172.16.166.147	TCP	1314	[TCP segment of a reassembled PDU]
103	95.420230	172.16.166.147	173.194.35.24	TCP	54	35976 > http [ACK] Seq=1963 Ack=8214 Win=32120 Len=0
104	95.420251	173.194.35.24	172.16.166.147	TCP	1514	[TCP segment of a reassembled PDU]
105	95.420258	172.16.166.147	173.194.35.24	TCP	54	35976 > http [ACK] Seq=1963 Ack=9674 Win=35040 Len=0
106	95.420281	173.194.35.24	172.16.166.147	TCP	1430	[TCP segment of a reassembled PDU]
107	95.420289	172.16.166.147	173.194.35.24	TCP	54	35976 > http [ACK] Seq=1963 Ack=11050 Win=37960 Len=0
108	95.420309	173.194.35.24	172.16.166.147	TCP	1314	[TCP segment of a reassembled PDU]
109	95.420316	172.16.166.147	173.194.35.24	TCP	54	35976 > http [ACK] Seq=1963 Ack=12210 Win=40880 Len=0

▶ Frame 103: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)

- ▶ Ethernet II, Src: Vmware_e2:37:0e (00:0c:29:e2:37:0e), Dst: Vmware_ef:4a:c8 (00:50:56:ef:4a:c8)
- ▶ Internet Protocol Version 4, Src: 172.16.166.147 (172.16.166.147), Dst: 173.194.35.24 (173.194.35.24)
- ▼ Transmission Control Protocol, Src Port: 35976 (35976), Dst Port: http (80), Seq: 1963, Ack: 8214, Len: 0
 - Source port: 35976 (35976)
 - Destination port: http (80)
 - [Stream index: 28]
 - Sequence number: 1963 (relative sequence number)
 - Acknowledgement number: 8214 (relative ack number)
 - Header length: 20 bytes
 - ▶ Flags: 0x10 (ACK)
 - Window size value: 32120
 - [Calculated window size: 32120]
 - [Window size scaling factor: 2 (no window scaling used)]

```
0000  00 50 56 ef 4a c8 00 0c 29 e2 37 0e 08 00 45 00  .PV.J... ).7...E.
0010  00 28 86 49 40 00 40 06 91 08 ac 10 a6 93 ad c2  .(.I@.@. ....
0020  23 18 8c 88 00 50 c6 a7 40 e4 5e d2 8f a4 50 10  #...P.. @.^...P.
0030  7d 78 23 99 00 00                                }x#...
```

Wireshark

Trace export

The screenshot shows the Wireshark File menu with the 'Export' option selected. The export options are:

- as "Plain Text" file...
- as "PostScript" file...
- as "CSV" (Comma Separated Values packet summary) file...
- as "C Arrays" (packet bytes) file...
- as XML - "P_SML" (packet summary) file...
- as XML - "P_DML" (packet details) file...
- Selected Packet Bytes...
- SSL Session Keys...
- Objects

Wireshark: Protocol Hierarchy Statistics

Display filter: none

Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
Frame	100.00%	13779	100.00%	9741057	0.484	0	0	0.00
Ethernet	100.00%	13779	100.00%	9741057	0.484	0	0	0.00
Internet Protocol Version 4	99.93%	13770	100.00%	9740607	0.484	0	0	0.00
User Datagram Protocol	6.35%	875	0.95%	92207	0.005	0	0	0.00
Bootstrap Protocol	0.02%	3	0.01%	1026	0.000	3	1026	0.00
Domain Name Service	6.32%	871	0.93%	90938	0.005	871	90938	0.00
NetBIOS Datagram Service	0.01%	1	0.00%	243	0.000	0	0	0.00
SMB (Server Message Block Protocol)	0.01%	1	0.00%	243	0.000	0	0	0.00
SMB MailSlot Protocol	0.01%	1	0.00%	243	0.000	0	0	0.00
Microsoft Windows Browser Protocol	0.01%	1	0.00%	243	0.000	1	243	0.00
Internet Control Message Protocol	0.03%	4	0.01%	757	0.000	4	757	0.00
Transmission Control Protocol	93.96%	12891	99.04%	9647643	0.479	11248	8452206	0.4
Hypertext Transfer Protocol	9.35%	1289	9.03%	879844	0.044	731	449668	0.00
Line-based text data	1.14%	157	1.25%	121401	0.006	157	121401	0.00
Portable Network Graphics	0.38%	52	0.43%	41425	0.002	52	41425	0.00
JPEG File Interchange Format	1.85%	255	2.07%	201631	0.010	255	201631	0.00
CompuServe GIF	0.30%	41	0.29%	28643	0.001	41	28643	0.00
Media Type	0.26%	36	0.26%	25098	0.001	36	25098	0.00
Online Certificate Status Protocol	0.09%	12	0.07%	6530	0.000	12	6530	0.00
Text item	0.03%	4	0.04%	4354	0.000	4	4354	0.00
eXtensible Markup Language	0.01%	1	0.01%	1094	0.000	1	1094	0.00
Secure Sockets Layer	2.57%	354	3.24%	315593	0.016	354	315593	0.00
Address Resolution Protocol	0.07%	9	0.00%	450	0.000	9	450	0.00

Protocol hierarchy

Wireshark: Filter Expression - Profile: Default

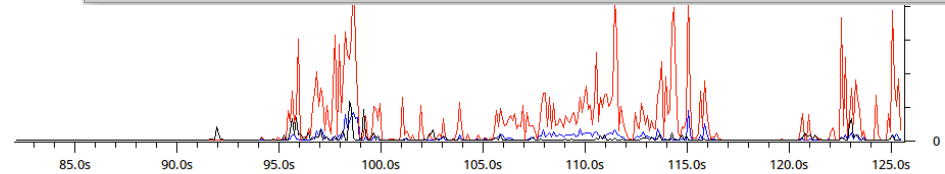
Field name: ip.version - Version

Relation: is present

Value (Boolean): 0

Predefined values: Low, Normal

Filtering



Graphs

Graph	Color	Filter	Style
Graph 1	Color	Filter: ip.addr == 172.16.166.2	Style: Line
Graph 2	Color	Filter: tcp	Style: Line
Graph 3	Color	Filter: udp	Style: Line
Graph 4	Color	Filter: http	Style: Line
Graph 5	Color	Filter:	Style: Line

X Axis: Tick interval: 0.1 sec

Y Axis: Pixels per tick: 2

Unit: Packets/Tick

Scale: Auto

I/O graphs

Wireshark and NETKIT

- Can I use wireshark to capture traffic on a NETKIT VM?
 - No! But I can use wireshark to open a trace captured with `tcpdump`
 - It's only a matter of copying the file from the VM to the HOST machine (let's use the `hosthome` directory)
 - Second option: copy the file with `nc`, `scp` or `rsync` (later on...)

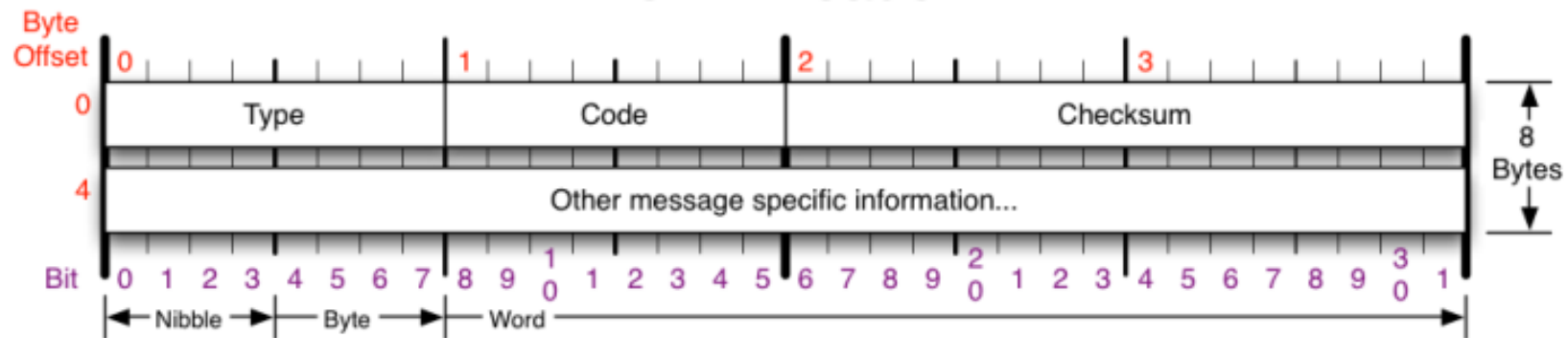
ping

- ping is one of the oldest IP utilities around
- ping asks another host if it is alive, and records the round-trip time between the request and the reply
- ping relies on ICMP echo-request and echo-reply packets (next slide..)
- **warning:** in some cases ICMP traffic is dropped by firewalls. We can not assume that all machines are down if they don't reply to a ping...

ICMP basics

- The **Internet Control Message Protocol** is one of the core protocols of the IP Suite
- ICMP packets are mainly used for diagnostic (ping, traceroute, timestamp request) and error notification (routing anomalies, unreachability, TTL expired, etc...)
- It goes directly on top of IP (but it can't be seen as a transport protocol)
 - IP.proto = 1
- We will focus on ICMP Echo Request/Reply. We'll see (and force the transmission) of other ICMP messages later on...

ICMP header



ICMP Message Types			Checksum
Type	Code/Name	Type	Code/Name
0	Echo Reply	3	Destination Unreachable (continued)
3	Destination Unreachable	12	Host Unreachable for TOS
0	Net Unreachable	13	Communication Administratively Prohibited
1	Host Unreachable	4	Source Quench
2	Protocol Unreachable	5	Redirect
3	Port Unreachable	0	Redirect Datagram for the Network
4	Fragmentation required, and DF set	1	Redirect Datagram for the Host
5	Source Route Failed	2	Redirect Datagram for the TOS & Network
6	Destination Network Unknown	3	Redirect Datagram for the TOS & Host
7	Destination Host Unknown	8	Echo
8	Source Host Isolated	9	Router Advertisement
9	Network Administratively Prohibited	10	Router Selection
10	Host Administratively Prohibited	11	Time Exceeded
11	Network Unreachable for TOS	0	TTL Exceeded
		1	Fragment Reassembly Time Exceeded
		12	Parameter Problem
		0	Pointer Problem
		1	Missing a Required Operand
		2	Bad Length
		13	Timestamp
		14	Timestamp Reply
		15	Information Request
		16	Information Reply
		17	Address Mask Request
		18	Address Mask Reply
		30	Traceroute

Checksum of ICMP header
RFC 792

Please refer to RFC 792 for the Internet Control Message protocol (ICMP) specification.

ping output and ICMP packets

The image shows a Wireshark capture of ICMP packets and a terminal window displaying the output of a ping command to 8.8.8.8.

Wireshark Filter: icmp

No.	Time	Source	Destination	Protocol	Length	Info
5	7.492344	172.16.166.147	8.8.8.8	ICMP	98	Echo (ping) request id=0x7c22, seq=1/256, ttl=64
6	7.517746	8.8.8.8	172.16.166.147	ICMP	98	Echo (ping) reply id=0x7c22, seq=1/256, ttl=128
7	8.494152	172.16.166.147	8.8.8.8	ICMP	98	Echo (ping) request id=0x7c22, seq=2/512, ttl=64
8	8.515512	8.8.8.8	172.16.166.147	ICMP	98	Echo (ping) reply id=0x7c22, seq=2/512, ttl=128
11	9.495708	172.16.166.147	8.8.8.8	ICMP	98	Echo (ping) request id=0x7c22, seq=3/768, ttl=64
12	9.520206	8.8.8.8	172.16.166.147	ICMP	98	Echo (ping) reply id=0x7c22, seq=3/768, ttl=128
13	10.497560	172.16.166.147	8.8.8.8	ICMP	98	Echo (ping) request id=0x7c22, seq=4/1024, ttl=64
14	10.521358	8.8.8.8	172.16.166.147	ICMP	98	Echo (ping) reply id=0x7c22, seq=4/1024, ttl=128

Frame 5: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)

- Ethernet II, Src: Vmware_e2:37:0e (00:0c:29:e2:37:0e), Dst: Vmware_ef:4a:c8 (00:0c:29:ef:4a:c8)
- Internet Protocol Version 4, Src: 172.16.166.147 (172.16.166.147), Dst: 8.8.8.8
- Internet Control Message Protocol
 - Type: 8 (Echo (ping) request)
 - Code: 0
 - Checksum: 0xa304 [correct]
 - Identifier (BE): 31778 (0x7c22)
 - Identifier (LE): 8828 (0x227c)
 - Sequence number (BE): 1 (0x0001)
 - Sequence number (LE): 256 (0x0100)

Data (56 bytes)

```
0000 00 50 56 ef 4a c8 00 0c 29 e2 37 0e 08 00 45 00 .PV.J... ).7...E.
0010 00 54 00 00 40 00 40 01 d7 f5 ac 10 a6 93 08 08 .T..@.@. ....
0020 08 08 08 00 a3 04 7c 22 00 01 03 78 50 4f 8e 0d .....| " ...xP0..
0030 0c 00 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 .....
0040 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 ..... !"#$$%
0050 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 &'()*+,- ./012345
0060 36 37
```

Terminal - marlon@marlon-vmxnb:~

```
marlon@marlon-vmxnb:~$ ping -c 10 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=128 time=25.4 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=128 time=21.3 ms
64 bytes from 8.8.8.8: icmp_req=3 ttl=128 time=24.5 ms
64 bytes from 8.8.8.8: icmp_req=4 ttl=128 time=23.8 ms
64 bytes from 8.8.8.8: icmp_req=5 ttl=128 time=24.5 ms
64 bytes from 8.8.8.8: icmp_req=6 ttl=128 time=23.9 ms
64 bytes from 8.8.8.8: icmp_req=7 ttl=128 time=24.0 ms
64 bytes from 8.8.8.8: icmp_req=8 ttl=128 time=25.2 ms
64 bytes from 8.8.8.8: icmp_req=9 ttl=128 time=26.5 ms
64 bytes from 8.8.8.8: icmp_req=10 ttl=128 time=24.1 ms

--- 8.8.8.8 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9014ms
rtt min/avg/max/mdev = 21.388/24.360/26.522/1.267 ms
marlon@marlon-vmxnb:~$
```

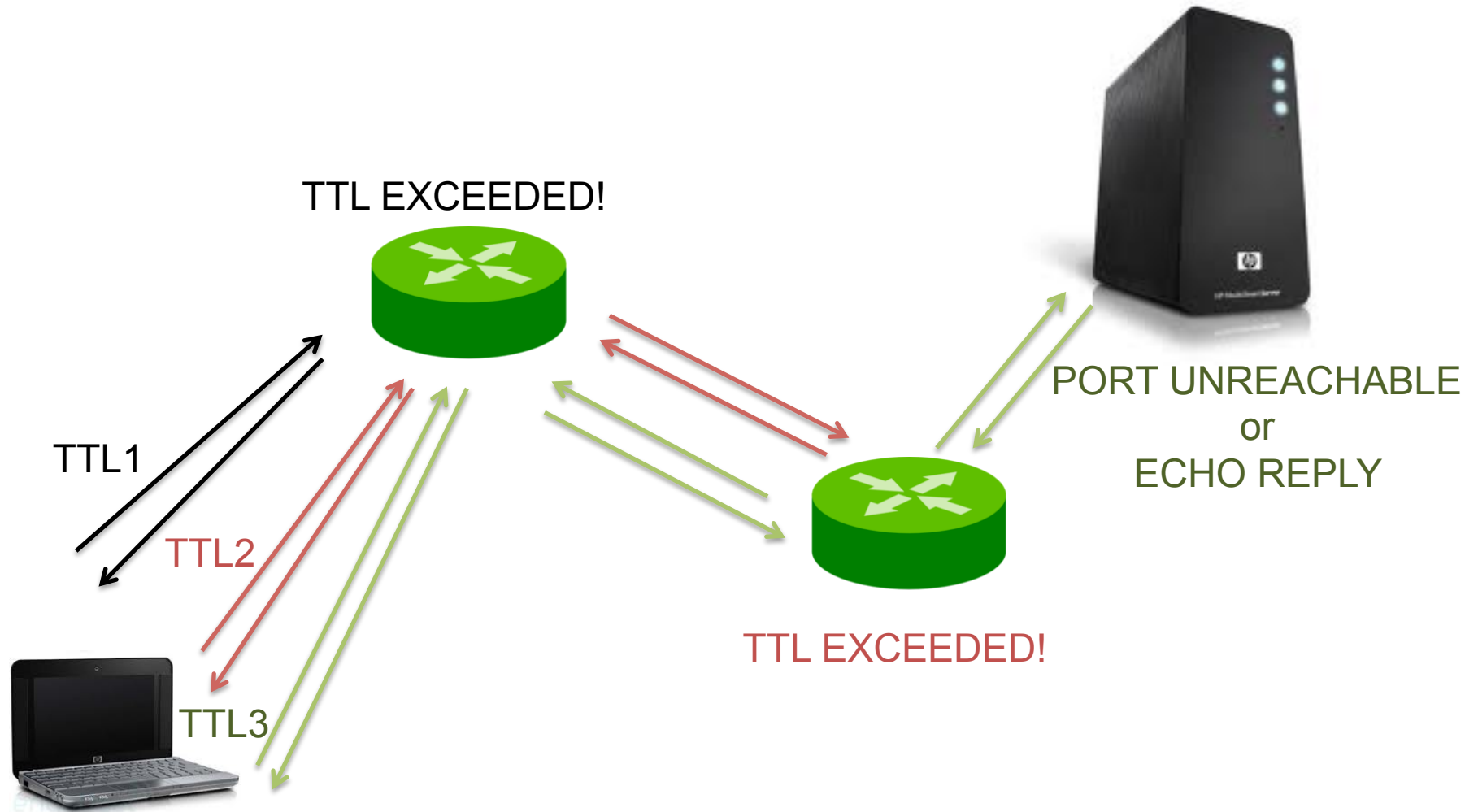

ping usage

- For a complete doc: `man ping`
- Essentials
 - Don't convert IP addresses to names (`-n`)
`ping -n 160.80.103.147`
 - Specify the number of packets (`-c`) and display only the summary line (`-q`)
`ping -q -c 10 160.80.103.147`
 - Specify the source address of the packets (`-I`)
`ping -I 10.0.0.12 160.80.103.147`
 - Stress the network (flood `-f`) and specify the size of the packet (`-s`)
`ping -c 5000 -s 512 -f 160.80.103.14`
 - Record the network route (many hosts ignore the ROUTE RECORD option. Let's use `traceroute` for that)
`ping -R 160.80.103.14`

traceroute

- A computer network diagnostic tool for displaying the route and measuring transit delays of packets across an IP network
- `traceroute` sends a sequence of packets to the destination
- `traceroute` works by increasing the TTL value of each successive (set of) packet(s)
- `traceroute` reconstructs the path to the destination by receiving the ICMP TTL Exceeded message by each router traversed by the packet
- Implementations on Unix-like OSs use UDP with ports from 33434 to 33534. Others use ICMP Echo Request
- For UDP version, `traceroute` ends when a port unreachable is received from the destination
- For ICMP version, `traceroute` ends when a ICMP Echo Reply is received for the destination

How does traceroute work?



traceroute

probe timeout

RTT

```
marlon@MarlonMAC:~$ traceroute -q 1 -v 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 64 hops max, 52 byte packets
 1  192.168.100.1 (192.168.100.1) 36 bytes to 192.168.100.63  6.710 ms
 2  10.192.0.1 (10.192.0.1) 36 bytes to 192.168.100.63  6.519 ms
 3  10.0.253.45 (10.0.253.45) 36 bytes to 192.168.100.63  5.579 ms
 4  10.0.253.30 (10.0.253.30) 36 bytes to 192.168.100.63  4.812 ms
 5  *
 6  rt-rm2-rt-mi2.mi2.garr.net (193.206.134.229) 36 bytes to 192.168.100.63  14.180 ms
 7  193.206.129.134 (193.206.129.134) 36 bytes to 192.168.100.63  11.496 ms
 8  216.239.47.128 (216.239.47.128) 36 bytes to 192.168.100.63  12.231 ms
 9  72.14.232.78 (72.14.232.78) 148 bytes to 192.168.100.63  22.366 ms
10  209.85.254.112 (209.85.254.112) 36 bytes to 192.168.100.63  21.627 ms
11  *
12  google-public-dns-a.google.com (8.8.8.8) 36 bytes to 192.168.100.63  24.035 ms
marlon@MarlonMAC:~$
```

Basis usage:

```
traceroute [options] $DEST_HOST
```

Useful options:

- q <num_queries>: number of queries
- i <iface_name>: source interface
- s <addr>: source address
- M <ttl>: initial TTL
- m <ttl>: maximum TTL
- w <time>: wait time for a probe response

netcat

- Utility that reads and writes data through IP transport session, either TCP or UDP
- It can create TCP or UDP socket in listening
 - `nc -l 9000` (open a TCP socket listening on port 9000)
 - `nc -lu 9000` (open a UDP socket listening on port 9000)
- It can connect a TCP socket
 - `nc 160.80.103.147 9000`
- It can create a UDP socket for sending packets
 - `nc -u 160.80.103.147 9000`
- **NOTE:** there are 2 versions of nc. One is the GNU version. The other one is the BSD porting. These 2 versions have a slightly different syntax and options. For example (that's the case of nc on the NETKIT VM), you might have to use the following syntax for listening sockets:
 - `# nc -l -p 9000`

Exercise: TCP connection

- Let's get back to Lab0
- On PC1 create a listening TCP socket on port 9999
- On PC2 connect a TCP socket to PC1:9999
- Write something and press CTRL+C to close
- Sniff the entire TCP flow on router
(connection, data, close – use `tcpdump` and write to a file)
- Display the trace with wireshark

Exercise: TCP connection

The image shows a network simulation environment. The top window is Wireshark 1.6.2, displaying a capture of a TCP connection. The packet list shows 10 packets, with packet 6 selected. The packet details pane shows the structure of the selected packet: Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The data field shows the raw bytes of the packet.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0a:ab:64:91:09:80	Broadcast	ARP	42	Who has 10.0.0.101? Tell 10.0.0.1
2	0.000176	6e:5f:98:37:0c:07	0a:ab:64:91:09:80	ARP	42	10.0.0.101 is at 6e:5f:98:37:0c:07
3	0.000186	10.0.1.101	10.0.0.101	TCP	74	32973 > 50010 [SYN] Seq=0 Win=5840 Len=0
4	0.000318	10.0.0.101	10.0.1.101	TCP	74	50010 > 32973 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0
5	0.000423	10.0.1.101	10.0.0.101	TCP	66	32973 > 50010 [ACK] Seq=1 Ack=1 Win=5840 Len=0
6	3.382944	10.0.1.101	10.0.0.101	TCP	71	32973 > 50010 [PSH, ACK] Seq=1 Ack=1 Win=5840 Len=5
7	3.383150	10.0.0.101	10.0.1.101	TCP	66	50010 > 32973 [ACK] Seq=1 Ack=6 Win=5840 Len=0
8	11.214624	10.0.1.101	10.0.0.101	TCP	66	32973 > 50010 [FIN, ACK] Seq=6 Ack=1 Win=0 Len=0
9	11.215401	10.0.0.101	10.0.1.101	TCP	66	50010 > 32973 [FIN, ACK] Seq=1 Ack=7 Win=0 Len=0
10	11.215500	10.0.1.101	10.0.0.101	TCP	66	32973 > 50010 [ACK] Seq=7 Ack=2 Win=5840 Len=0

The bottom window shows a terminal with Netkit logs and a shell prompt. The logs indicate the initialization of Netkit phase 2 and the capture of a TCP connection. The terminal output shows the following:

```
>>> Running pc1 specific startup script.
>>> End of pc1 specific startup script.

-----
Lab directory (host): /home/knoppix/Desktop
Version: <none>
Author: <none>
Email: <none>
Web: <none>
Description:
-----

router login: root (automatic login)
router:~# tcpdump -i eth0 -w /home/tcp.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
^C10 packets captured
10 packets received by filter
0 packets dropped by kernel
router:~#

-----
Netkit phase 2 initialization terminated

pc1 login: root (automatic login)
pc1:~# nc -lp 50010
ciao
^C
pc1:~#

pc2 login: root (automatic login)
Last login: Fri Jan 13 22:30:57 UTC 2012 on tty1
pc2:~# nc 10.0.0.101 50010
ciao
^C
pc2:~#
```

Advanced use of netcat

- We already saw nc as a chat 😊
- We can also transfer files:
 - server:# nc -l 9000 > received_file
 - client:# cat file_to_send | nc \$server 9000
- Get a web page (like wget)
 - client:# printf "GET / HTTP/1.0\\r\\n\\r\\n\\n"
| nc 160.80.103.147 80
- Remote shell (dangerous – removed from bsd porting)
 - server:# nc -l 9000 -e /bin/bash
 - client:# nc \$server 9000
- Perform a port scan (-z option)
 - client# nc -v -z \$target 7-1023

SS

- Utility to investigate sockets
- All TCP sockets, all UDP sockets, all established ssh / ftp / http / https connections, all local processes connected to X server, etc...
- Basic usage: `# ss [options] [filter]`
 - s: display summary
 - a: display both listening and non-listening
 - l: display listening socket
 - t: display TCP sockets
 - u: display UDP sockets
 - p: display processes using sockets

And many more...

- Documentation: `/usr/share/doc/iproute-doc/ss.html`

ss output

```
marlon@marlon-vmxnb:~$ ss -ap
State      Recv-Q  Send-Q           Local Address:Port           Peer Address:Port
LISTEN     0        4             127.0.0.1:5037                *:*
LISTEN     0       128            *:www                          *:*
LISTEN     0       128            *:webmin                       *:*
LISTEN     0        5             :::domain                      :::*
LISTEN     0        5             *:domain                       *:*
LISTEN     0       128            *:ssh                          *:*
LISTEN     0       128            :::ssh                         :::*
LISTEN     0       128            127.0.0.1:ipp                 *:*
LISTEN     0       128            :::1:ipp                       :::*
ESTAB      200      0             172.16.166.147:33756          172.16.166.1:netbios-s
sn users:(("gvfsd-smb-brows",29392,9))
ESTAB      0        0             172.16.166.147:43615          199.7.59.72:www
users:(("chromium-browse",2235,82))
ESTAB      0        0             172.16.166.147:42840          173.194.35.51:https
users:(("chromium-browse",2235,71))
```

```
marlon@marlon-vmxnb:~$ ss -s
Total: 512 (kernel 0)
TCP: 16 (estab 6, closed 1, orphaned 0, synrecv 0, timewait 0/0), ports 0

Transport Total      IP      IPv6
*          0
RAW        0
UDP        9
TCP        15
INET       24
FRAG       0
```

Remote access - telnet

- Telnet protocol provides a fairly general, bi-directional, eight-bit byte oriented communications facility
- A telnet connection is a Transmission Control Protocol (TCP – listening port 23) connection used to transmit data with interspersed telnet control information
 - Data: 1st bit 0 (ASCII character)
 - Commands: 1st bit 1
- Nice article describing the protocol:
 - <http://support.microsoft.com/kb/231866>
- Typical use: remote shell
- Example PCAP trace:
 - <http://stud.netgroup.uniroma2.it/cgri/traces/telnet.pcap>
- Client/server implementation for virtually all OSs!
 - On linux: `telnet/telnetd`
 - daemon usually not installed (`apt-get install telnetd`)
- Due to several security aspects it has been “abandoned” in favor of SSH

Remote Access - SSH

- Secure Shell (SSH) is a protocol for secure remote login and other secure network services over an insecure network
- RFCs define 3 major components:
 - The Transport Layer Protocol (RFC4252)
 - The User Authentication Protocol (RFC4253)
 - The Connection Protocol (RFC4254)
- OpenSSH (client/server implementation):
 - Encryption, Authentication, Data integrity
 - Secure file transfer (`scp`)
 - X session forwarding
 - Port forwarding
 - SOCKS4|5 proxy
 - Public Key authentication
- We won't take a look at the protocol, but we'll focus on some practical uses

OpenSSH installation and configuration (DEBIAN)

- `openssh-client` present in almost all Linux distribution (DEBIAN included)
- `openssh-server` usually not included
 - `apt-get install openssh-server`
- Configuration file in:
 - Server: `/etc/ssh/sshd_config`
 - Client: `/etc/ssh/ssh_config`
- Documentation:
 - `man (ssh_config|sshd_config)`
- Useful configuration parameters (server, except `ServerAliveInterval`):
 - `Protocol (1|2)`
 - `PermitRootLogin (yrs|no)`
 - `PasswordAuthentication (yes|no)`
 - `X11Forwarding (yes|no)`
 - `ServerAliveInterval <seconds>`
 - `DenyUsers <user list> and DenyGroups <group list>`
 - `UseDNS no`
- Remember to restart ssh to apply any changes in the configuration file
 - `/etc/init.d/ssh restart`

OpenSSH basic usage

To connect to a ssh server just type

```
ssh user@server
```

```
marlon@MarlonMAC:~$ ssh upmt@byron.netgroup.uniroma2.it
The authenticity of host 'byron.netgroup.uniroma2.it (160.80.103.147)' can't be established.
RSA key fingerprint is a8:74:39:b2:53:32:d5:18:f8:9a:eb:d9:bb:c3:62:c7.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'byron.netgroup.uniroma2.it,160.80.103.147' (RSA) to the list of known hosts.
upmt@byron.netgroup.uniroma2.it's password:
Linux byron 2.6.32.21-upmt #2 SMP Mon Mar 28 13:20:05 CEST 2011 x86_64 GNU/Linux
Ubuntu 10.04.4 LTS
Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/

28 packages can be updated.
24 updates are security updates.

Last login: Fri Jul 8 15:01:07 2011 from andrea-laptop.local
upmt@byron:~$
```

- The server send it's public key fingerprint
- The program asks you to verify the authenticity of the key
- Once the host is recognized, the server address is put in the file `~/.ssh/known_host`
 - To check fingerprints on server: `ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key`
- What if the key fingerprint doesn't match the one stored in `~/.ssh/known_host`? See the next slide...

SSH key authentication failure

```
marlon@MarlonMAC:~$ ssh 172.16.166.147 (ubuntu1) ...
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!    @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
9e:32:f0:94:09:84:6e:d9:6c:dd:01:f5:33:bb:82:88.
Please contact your system administrator.
Add correct host key in /Users/marlon/.ssh/known_hosts to get rid of this message.
Offending key in /Users/marlon/.ssh/known_hosts:3
RSA host key for 172.16.166.147 has changed and you have requested strict checking.
Host key verification failed.
marlon@MarlonMAC:~$
```

Not necessarily something nasty is happening!
E.g.: ssh has been reinstalled or a big update has
request the generation of a new key (pair)

SSH public key authentication

- It might happen that a sysadmin doesn't trust the strength of a user password
- Users' account violation can lead to apocalyptic scenarios (sudoers users...)
- Public key authentication is a stronger auth method
- Users are requested to generate a public/private key
- The public key is manually (and over a secure channel) installed on the server
- The user is not authenticated via user/password verification, but via a "safer" cryptographically challenge/response mechanism (later on...)

Public key authentication with OpenSSH

```
pippo@marlon-vmxnb:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pippo/.ssh/id_rsa):
Created directory '/home/pippo/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pippo/.ssh/id_rsa.
Your public key has been saved in /home/pippo/.ssh/id_rsa.pub.
The key fingerprint is:
3c:55:18:b3:fb:ce:b2:c2:c0:a9:4a:f9:9a:07:c8:63 pippo@marlon-vmxnb
The key's randomart image is:
+--[ RSA 2048 ]-----+
|           oo.       |
|          .+         |
|           o         |
|         . . . .     |
|      ..  .S .       |
| .E..  + . . .      |
| . +. . o . .       |
|  . oo  o . o       |
|   ++.   ..oo      |
+-----+
GN
```

Public key authentication with OpenSSH

- The client generates the key pair
`ssh-keygen -t (rsa|dsa)`
- By default, the public key is stored in:
`~/.ssh/id_rsa.pub`
or
`~/.ssh/id_dsa.pub`
- The public key has to be appended to the file `~/.ssh/authorized_keys` in the home of the authorized user
- 1st way, assuming that `id_rsa.pub` has been securely copied on the remote machine
`cat id_rsa.pub >> ~/.ssh/authorized_keys`
- 2nd way, with a tool provided by OpenSSH (from the client)
`ssh-copy-id user@server`

Exercise

- Back to Lab0-interfaces
- Install SSH server on router (if needed)
- Force public key authentication
- Configure public key authentication for user@router

Secure file transfer over SSH

- Basic usage

- `scp [-r] [[user@]host1:]file1 ... [[user@]host2:]file2`

- Examples

- 1) `scp file1 marlon@example.org:`
 - 2) `scp marlon@example.org:file2 /home/marlon/dir/`
 - 3) `scp -r dir/ marlon@example.org:/home/marlon/dir_target`

Where:

- 1) `file1` is copied in marlon's home on the remote host
 - 2) `file2` (in marlon's remote home) is copied in the specified local path with the same name
 - 3) The local directory `dir` is recursively copied into the specified remote path

OpenSSH advanced usage

- Running commands over ssh
 - `ssh username@server "command"`
- Forward X session
 - `ssh -X username@server`
- Local Port forward
 - `ssh -L lport:remote_addr:rport username@server`
- Remote port forward
 - `ssh -R rport:local_addr:lport username@server`
- Socks5 proxy
 - `ssh -ND 9999 username@server`
- Remote filesystem with sshfs
 - `sshfs user@host: mountpoint`
- Nice tutorials:
 - <http://www.subhashdasyam.com/2011/05/25-best-ssh-commands-tricks.html>

Local Port Forwarding example



Lab1-ssh

Problem: router1 doesn't have the route to 192.168.0.0/24 (as in real world topologies...)

(Note: router1 and router2 on the same lan is not a real topology... let's pretend they reach each other through the internet...)

Goal: connect pc to server:2024 with nc trough a "SSH tunnel"

Preliminaries:

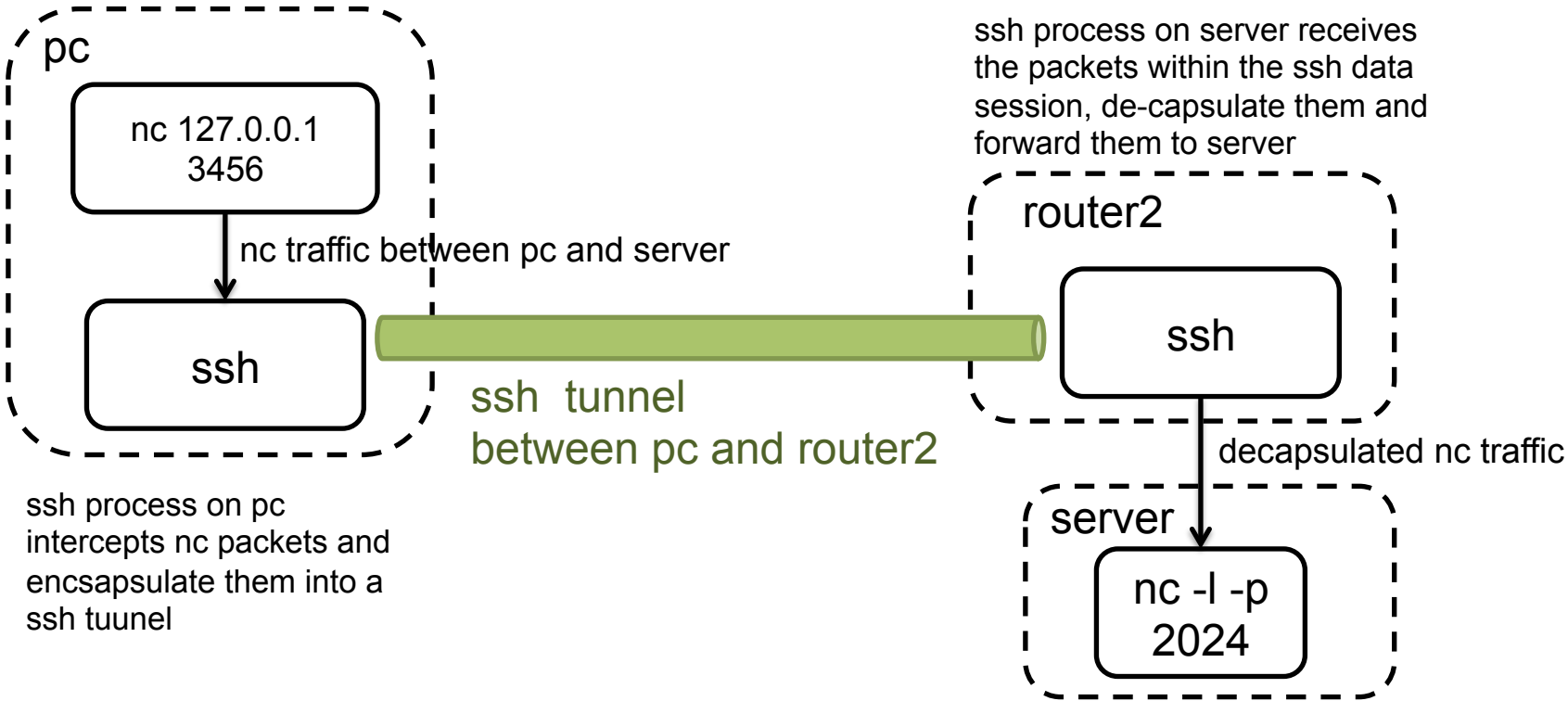
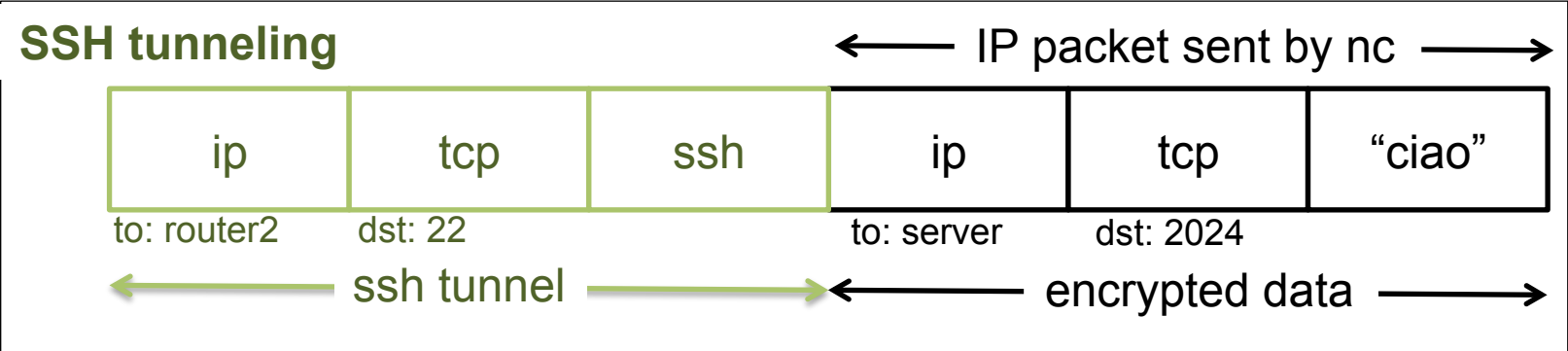
Install openssh-server on router2 (if not already installed)

Create a guest account (user) for ssh login on router2 (set the password for "user" account)

To reach server from pc:

- 1) Put server:2024 in listening on port 2024
`server# nc -l -p 2024`
- 2) Run ssh port forwarding command on pc
`pc# ssh -NL 3456:192.168.0.100:2024 user@8.0.0.2`
- 3) Connect nc to server
`pc# nc 127.0.0.1 3456`

Local Port Forwarding: how it works



SSH remote port forwarding

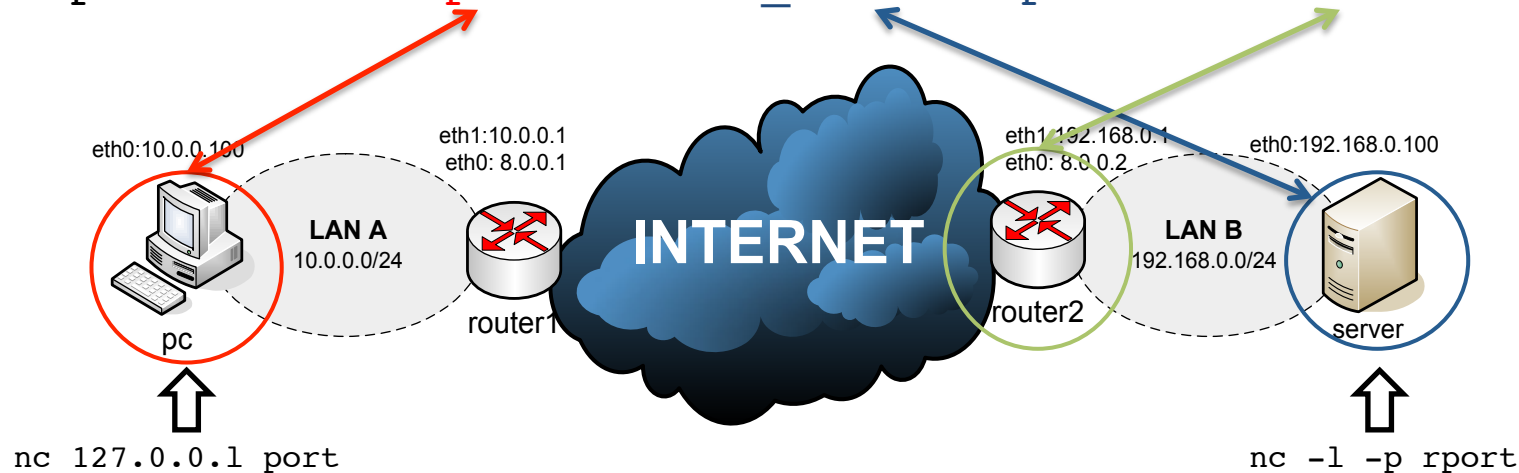
- Remote port forwarding
 - `ssh -NR r_port:local_addr:l_port user@server`
- In the previous example, we want to connect a tcp socket port 3000 from router2 to pc:2000
 - `pc# ssh -NR 3000:10.0.0.100:2000 user@8.0.0.2`
- We put nc in listening on pc
 - `pc# nc -l -p 2000`
- We connect nc from router2
 - `router2# nc 127.0.0.1 3000`

SSH port forwarding “for everyone”

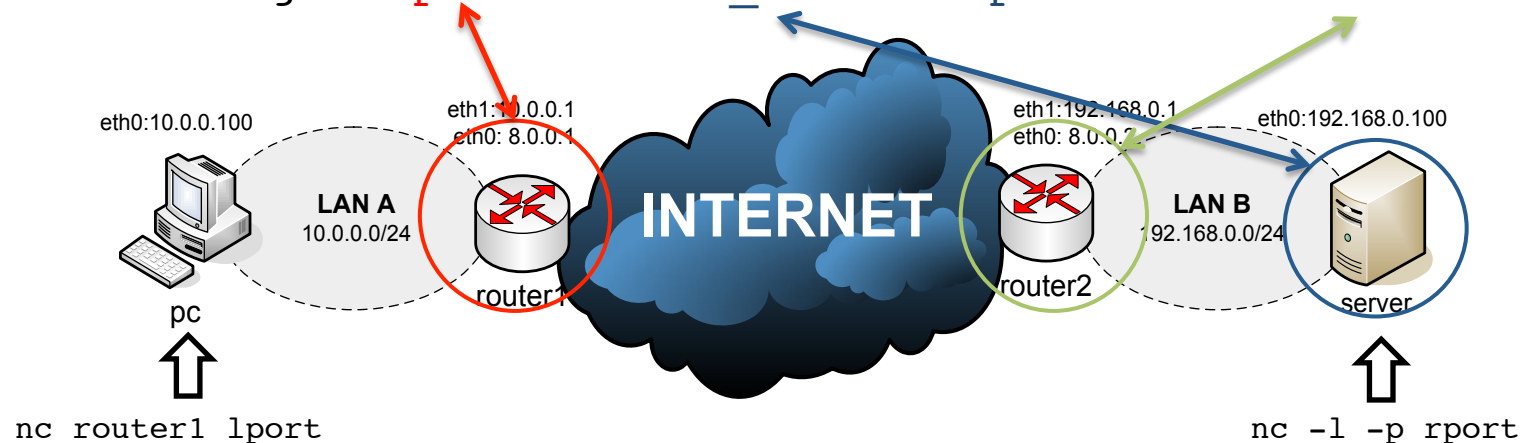
- We can also set up a gateway that forwards ports for all hosts in a LAN
- For example, we can run ssh local port forwarding on router1 for all hosts in LAN A
 - `router1# ssh -NL 3456:192.168.0.100:2024 user@router2 -g`
- For remote port forwarding there's no “-g” option
 - We have to set the following config option in `sshd_config`
 - `GatewayPorts yes`

SSH local port forwarding explained

```
pc# ssh -NL lport:remote_addr:rport user@server
```

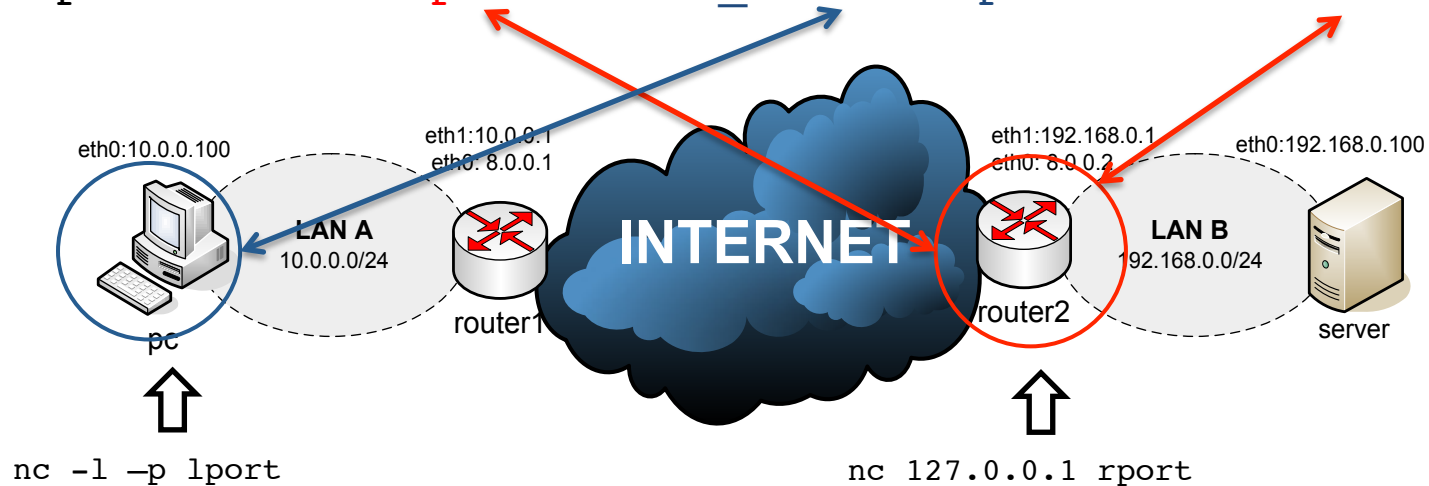


```
router1# ssh -gNL lport:remote_addr:rport user@server
```

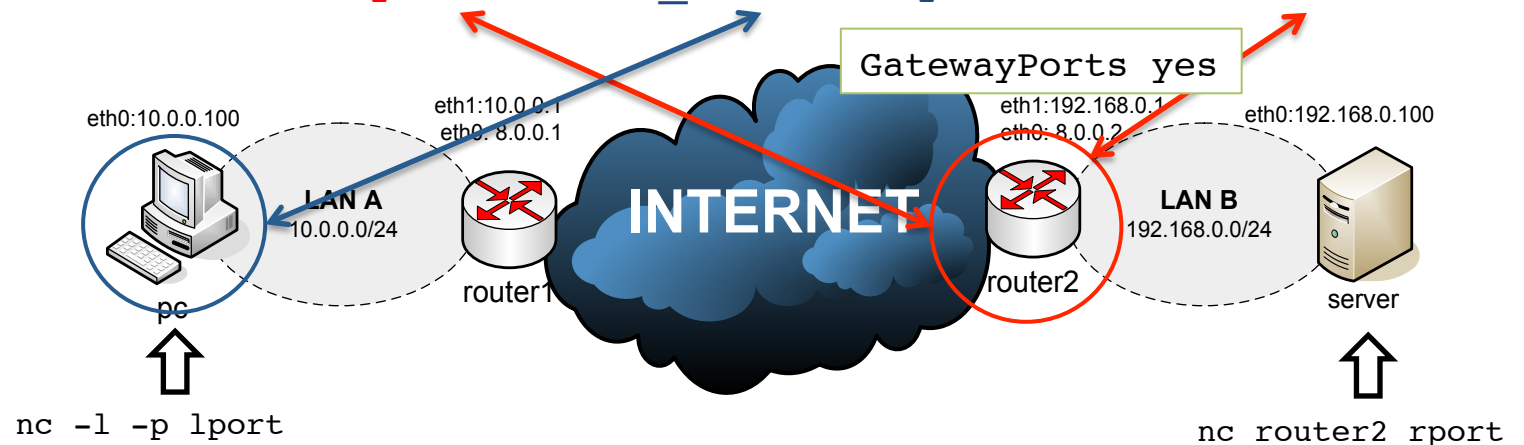


SSH remote port forwarding explained

```
pc# ssh -NR rport:local_address:lport user@server
```



```
router1# ssh -NR rport:local_address:lport user@server
```

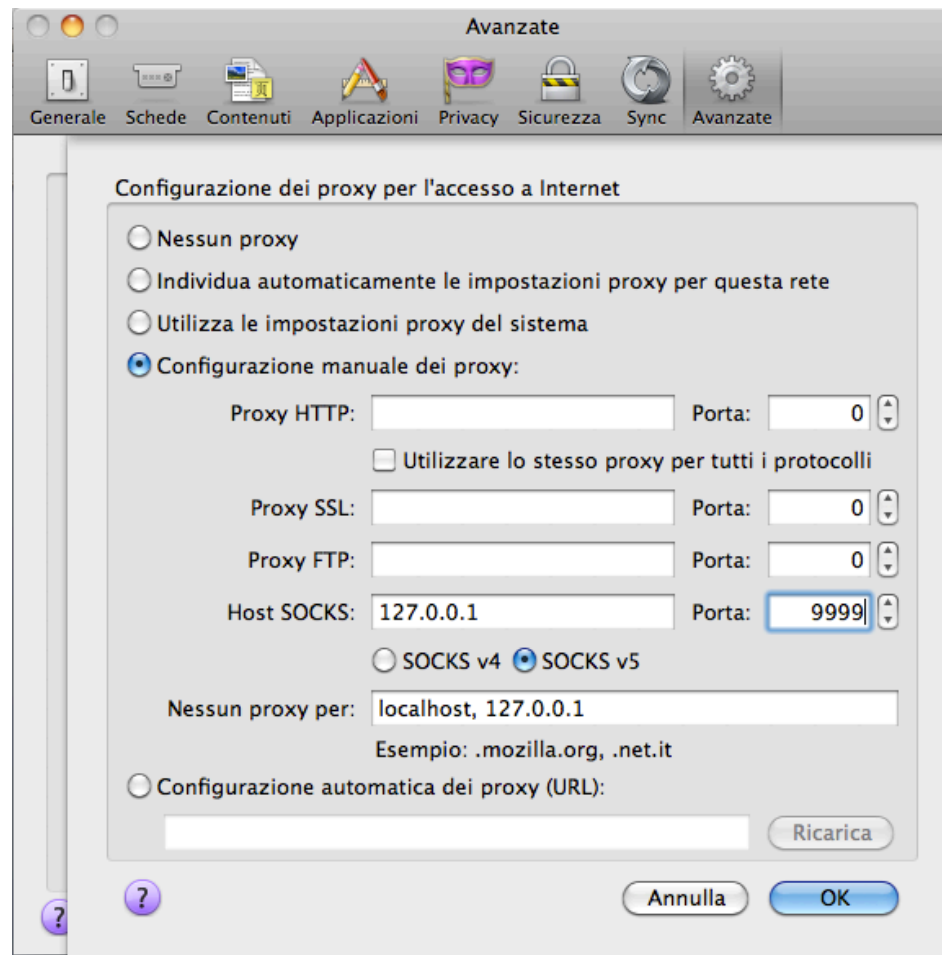


Shared screen and X forward with SSH

- Useful trick to share the same remote screen
 1. ssh to my machine marlonmac.local (or let's see what address I have now...) with the user "student" password "student"
 2. Attach to a already attached screen with "screen -r -x"
 3. Have fun!

- To run a graphic application on server
 - Set on server sshd_config:
`X11Forwarding yes`
 - Run ssh on client
`client# ssh -X user@server`
 - Run a graphic app on client
`client# xclock`

SSH SOCKS5 proxy



Example: `ssh -ND 9999 username@server`

SSH SOCKS5 test

- Copy Lab1-ssh/web_page_test/* into server:/var/www
- Configure firefox on the host machine to use a SOCKS5 local proxy
- Use router2 as relay to server
- Start apache in VM “server”
- Open the web page <http://192.168.0.100>, which is VM “server”

rsync

- Rsync is a fast and versatile file copying tool
- Rsync copies files either to or from a remote host, or locally on the current host
- **Delta-transfer Algorithm**
 - reduces the amount of data sent over the network by sending only the differences between the source files and the existing files in the destination
- Two modes:
 1. Through a secure shell (ssh, rsh)
 2. Contacting a remote rsync daemon directly via TCP
- Basic usages (man for the options..):

```
rsync -avz --progress foo:src/bar/ /data/tmp
rsync -av src/ dest/
rsync -av --delete host::src /dest
rsync -avd rsync://host:src /dest
rsync -ravz --exclude="*.o" foo:src/bar /data/tmp
```
- Nice tutorial
 - <http://www.thegeekstuff.com/2010/09/rsync-command-examples/>

Simple backup script with rsync in Lab1-ssh

```
#!/bin/sh
LOCAL=/root
REMOTE=/var/backup
HOST=8.0.0.1
LOG=/var/log/backup.log
SYNCLOG=/var/log/backup.synclog

#start log
echo $(date +"%d/%m/%Y") | cat >> $LOG
echo $(date +"%H:%M:%S") backup started... | cat >> $LOG

#Rsync
rsync --delete -azv -e ssh $LOCAL root@$HOST:$REMOTE | cat > $SYNCLOG

#end log
echo $(date +"%H:%M:%S") backup ended! | cat >> $LOG
```

1) Save the script in

```
/bin/rsyn_backup.sh
```

2) Make it executable

```
chmod +x /bin/rsyn_backup.sh
```

3) Add the cron job with the command

```
crontab -e
```

4) Put the following line

```
0 4 * * * /usr/local/bin/rsync_backup.sh
```


wget

- GNU Wget is a free utility for non-interactive download of files from the Web
- It supports HTTP, HTTPS, and FTP protocols, as well as retrieval through HTTP proxies
- Wget is non-interactive, meaning that it can work in the background, while the user is not logged on. This allows you to start a retrieval and disconnect from the system, letting wget finish the work
- Basic usage:
 - wget <http://www.example.com/>
- Recursive download (1 folder):
 - wget -l 1 -r byron.netgroup.uniroma2.it/
~marlon/RAT
 - (Change 1 → “n” for more levels...)

wget - mirroring

```
wget --recursive --no-clobber --page-requisites --adjust-  
extension --convert-links --restrict-file-names=windows --  
domains website.org --no-parent website.org
```

- --recursive: download the entire Web site
- --domains website.org: don't follow links outside website.org
- --no-parent: don't follow links outside the directory tutorials/html/
- --page-requisites: get all the elements that compose the page (images, CSS and so on)
- --adjust-extension: save files with the .html extension
- --convert-links: convert links so that they work locally, off-line
- --restrict-file-names=windows: modify filenames so that they will work in Windows as well
- --no-clobber: don't overwrite any existing files (used in case the download is interrupted and resumed)

source:

<http://www.linuxjournal.com/content/downloading-entire-web-site-wget>