SI5 – Administration des Réseaux

Troubleshooting Computer Networks

Dr. Quentin Jacquemart
quentin.jacquemart@unice.fr

http://www.qj.be/teaching/

Outline

Motivation I

- Situation:
  - You are a developer
  - There is an anomaly in your software
- What do you do?

Motivation II

1. Is it a bug?
   - Unit/module tests
   - Application log
   - $\Rightarrow$ Reproduce
2. It is a bug!
   - GDB, valgrind: step-by-step execution, controlled environment
   - strace, ptrace
   - printf
   - ...
3. Correct the bug (and introduce new ones :-)

Food for Thought

There are two ways of constructing a software design.

One way is to make it so simple that there are obviously no deficiencies.

The other way is to make it so complicated
that there are no obvious deficiencies.

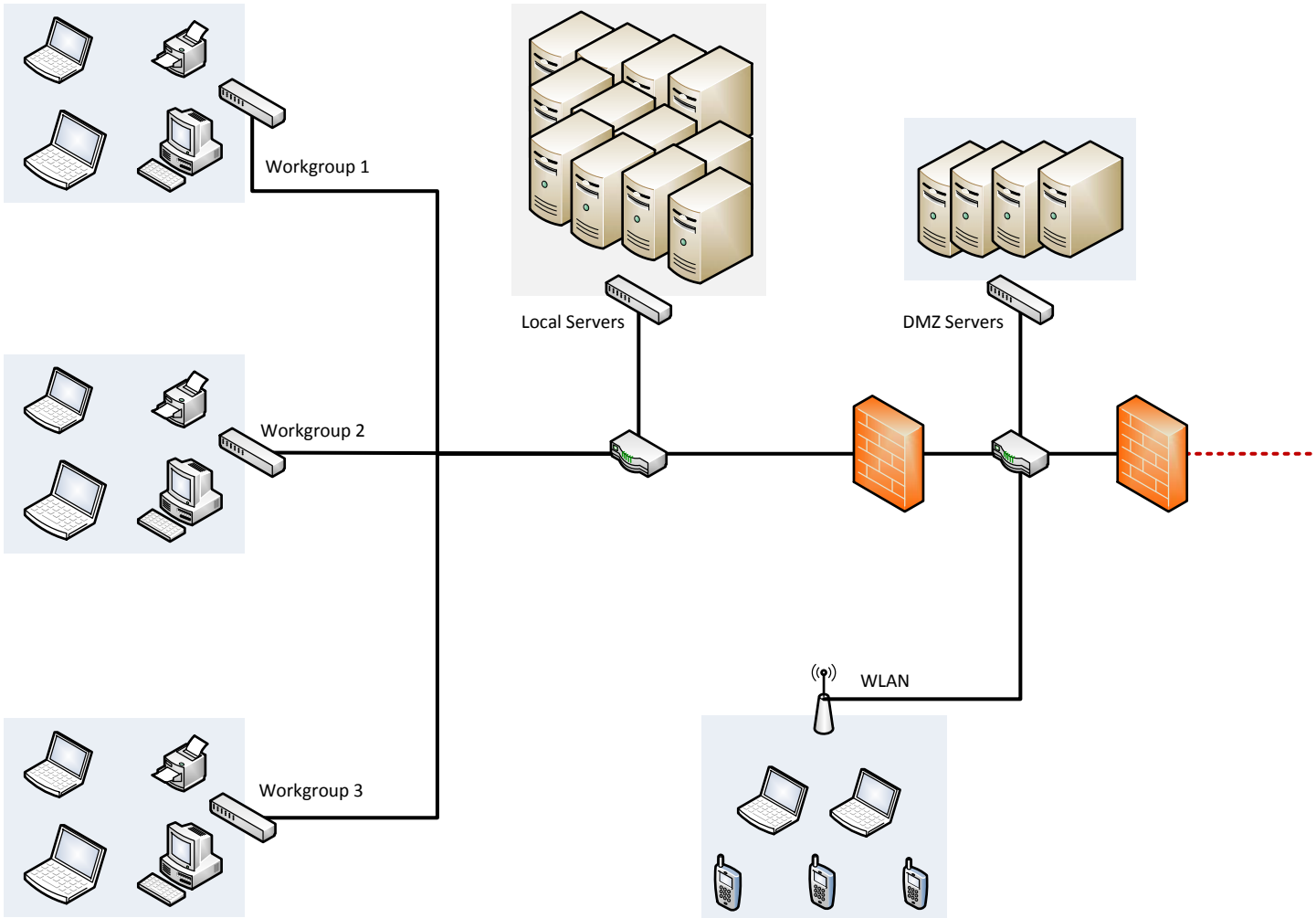
The first method is far more difficult.

Charles Antony Richard Hoare

Motivation III

- Situation:
  - You are (very) knowledgeable about networks
  - Someone says "The network/Internet does not work"
- What do you do?
  - What are the first-aid tools you can use?
  - What are good advanced tools?

Troubleshooting a Failing Network Connection. . . I



Troubleshooting a Failing Network Connection...II

A few ideas:

- Is the RJ45 properly inserted?
- Is the end-host correctly configured?
- Is the user authorized to access (external) services?
- Is a network service (e.g. DNS) down?
- Is a firewall blocking access?
- Is it a local issue? An ISP issue?
- ...

TCP/IP Stack: a Reminder I

TRANSPORT	*TCP*	*UDP*		
NETWORK	*ICMP* *IPv4*	*IPsec* *IPv6*		
DATA-LINK	*ATM*	*ARP*	*Ethernet*	*IS-IS*
PHYSICAL	*POTS*	*DSL*	*802.11*	

TCP/IP Stack: a Reminder II

[K&R]

1. **Physical** layer
   — bit streams/signals over the transmission medium
   (e.g. cable, air, fiber, . . .)
2. **Data-link** layer
   — data transfer between neighbouring network elements
3. **Network** layer
   — routing of datagrams from source to destination
4. **Transport** layer
   — process-to-process data transfer

- Layers 5-7 are part of OSI model, but not of "Internet stack"

Outline

- Motivation

⇒ **ICMP**

- Tools

ICMP: Introduction I

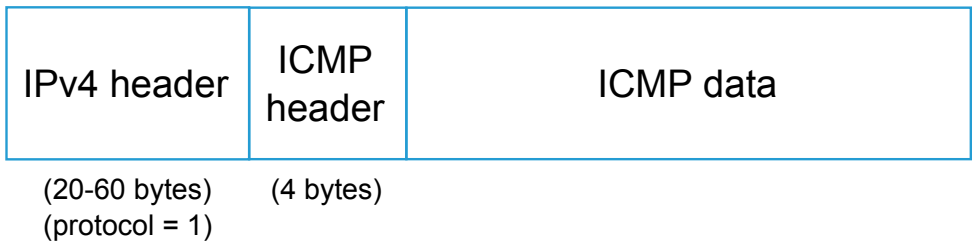
[Fall *et al.* 2011]

- IP is a *best-effort* protocol
  - No guarantee that a datagram will be delivered
- **ICMP**: *Internet Control Message Protocol*
  - Considered part of IP layer ⇒ mandatory
  - Relies on IP for transportation (i.e. between layer 2 and layer 3)
- ICMP enables *messages* related to **error** and **control**
  - Mostly handled directly by IP itself
  - Can be passed on to TCP/UDP and/or to applications for custom handling

ICMPv4

[Fall *et al.* 2011]

- Specific to IPv4
- *Defined* in [RFC792] (1981), [RFC1122] (1989), and [RFC1812] (1995)
- *Extensions* objects to ICMP defined in [RFC4884] (2007)
  (e.g. with MPLS [RFC4950])
- Behaviour of *ICMP through NAT* defined in [RFC5508] (2009)
- Used for **error reporting** and **signaling**
- **Encapsulation** of ICMPv4 within IPv4:

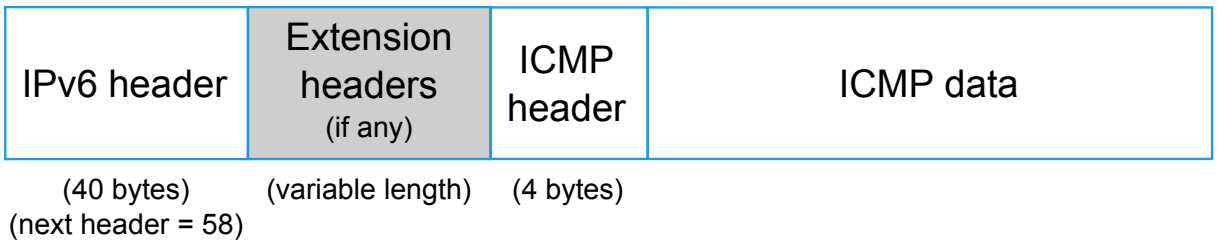


- The *ICMPv4 header* contains a *checksum* of the *ICMP data*

ICMPv6 I

[Fall *et al.* 2011]

- Specific to IPv6
- *Defined* in [RFC4443] (2006)
- Same specifications for extensions and NAT behaviour as ICMPv4
- **Encapsulation** of ICMPv6 within IPv6:



- The *ICMPv6 header* contains a *checksum* of the *ICMP data*, the *source* and *destination* IPv6 addresses, and some fields from the IPv6 header

ICMPv6 II

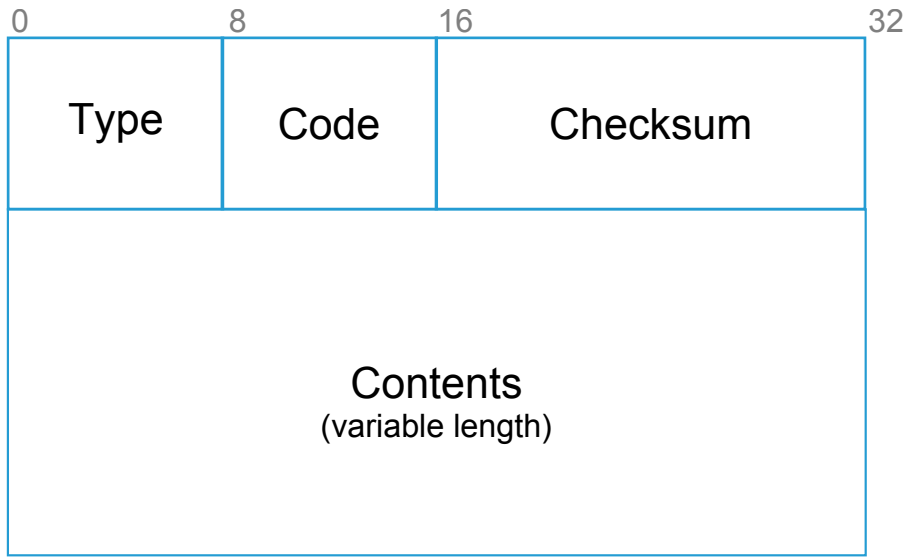
[Fall *et al.* 2011]

- *Broader* use than ICMPv4:
  - Error reporting and signaling (similar to ICMPv4)
  - **Neighbour Discovery** (IPv6 equivalent to ARP)
  - **Router Discovery**

ICMP Message Data Format

[Fall *et al.* 2011]

- ICMP message format



- The **type** corresponds to a *category* of message
- The **code** can be used to further specify the message
- The message **contents** depends on the message type

ICMP Error Messages

[Fall *et al.* 2011]

- **ICMP Error Messages** convey *information* about **IP delivery problems**
  - ICMPv4: types 3, 5, 11, 12
  - ICMPv6: types 0–127
- Contain a *copy* of the *full* IP header of the original IP datagram
  - plus any following data
    (up to 576 bytes (ICMPv4) or the minimum IPv6 MTU (ICMPv6))
  - receiver of ICMP message can identify the corresponding user process
    (e.g. using the `protocol` value)

ICMP Error Messages: Destination Unreachable I [Fall *et al.* 2011]

- *Error*: **Destination Unreachable**
  - A datagram could not be delivered to its destination
  - ICMPv4 type 3 ICMPv6 type 1
  - Error further specified by code value
- ICMPv4 type 3/code 1 **Host Unreachable** ICMPv6 type 1/code 3 **Address Unreachable**
  - Generated by a *router* or *host* upon **direct delivery failure**

    (e.g. ARP request from last hop fails)
- ICMPv4 type 3/code 1 **Host Unreachable** ICMPv6 type 1/code 0 **No Route to Destination**
  - ICMPv6 only
  - Used when no ad-hoc forwarding rule exists to the datagram destination

    (e.g. no installed route, no next-hop, . . .)

ICMP Error Messages: Destination Unreachable II [Fall *et al.* 2011]

- ICMPv4 type 3/code 13
  Communication Administratively Prohibited

  ICMPv6 type 1/code 1
  Communication with Destination Administratively Prohibited

  — An *administrative prohibition* prevents the successful communication with destination (e.g. firewall drops packets)

  — Administrators are usually *unwilling* to provide this information
  $\Rightarrow$ silent discard, or other ICMP error message generated
  (e.g. iptable's `--reject-with icmp-host-unreachable`)

- ICMPv4 type 3/code 3
  Port Unreachable

  ICMPv6 type 1/code 4
  Port Unreachable

  — Incoming datagram is destined for an application that is not ready to receive it (e.g. a UDP packet destined to a port without an active UDP server)

ICMP Error Messages: Destination Unreachable III [Fall *et al.* 2011]

- ICMPv4 type 3/code 4
  Packet Too Big (PTB)

  ICMPv6 type 2/code 0
  Packet Too Big (PTB)

  — Generated by a *router* if the datagram to be forwarded does not fit within the MTU on the selected outgoing interface

  — IPv4: only when the `don't fragment` flag is set (otherwise fragmented)

  — IPv6: used for *Path MTU Discovery* (fragmentation is done by sender)

- (Undefined in ICMPv4)

  ICMPv6 type 1/code 2
  Beyond Scope of Source Address

  — ICMPv6 only

  — Generated when the *source* and *destination* address of an IPv6 datagram are of different scopes

  (e.g. a datagram from a local-scope address sent to a global-scope address)

ICMP Error Messages: Destination Unreachable IV [Fall *et al.* 2011]

- (Undefined in ICMPv4)

  ICMPv6 type 1/code 5
  Source Address Failed Ingress/Egress Policy

  — ICMPv6 only

  — More refined than ICMPv6 type 1 code 5 "Communication with Destination Administratively Prohibited"

  — Specifies the reason for prohibiting the communication

- (Undefined in ICMPv4)

  ICMPv6 type 1/code 6
  Reject Route to Destination

  — ICMPv6 only

  — Generated by a *router* when a datagram matches a route marked as *blocked*
  (e.g. martian/bogon prefixes)

ICMP Error Messages: Redirect

[Fall *et al.* 2011]

- ICMPv4 type 5
  Redirect

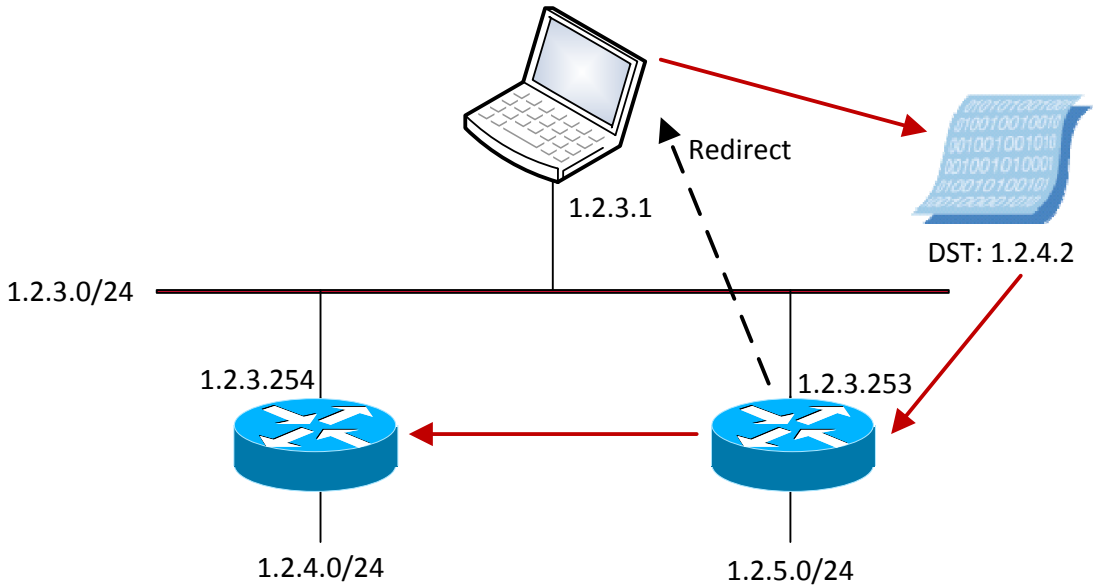
  ICMPv6 type 137
  Redirect

  — Generated by a *router* when it determines that it is not the correct next-hop to deliver the datagram

  — The message contains the IP address of the next-hop to be used
  (+ original IP header, etc. . .)

  — Hosts **update** their *routing table* (new entry to the destination IP)

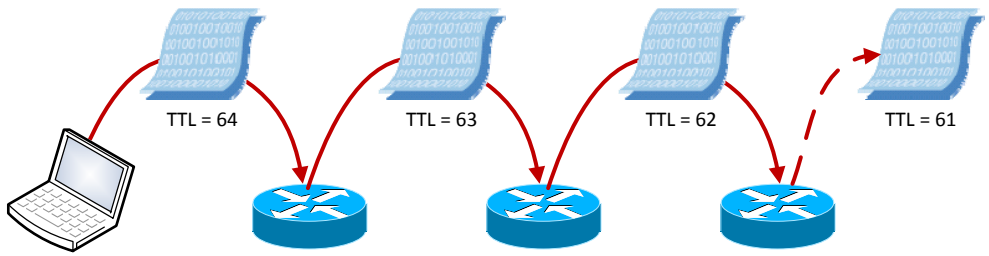
  — Routers do **not update** to avoid tampering with routing algorithms
  (e.g. OSPF)



ICMP Error Messages: Time Exceeded I

[Fall *et al.* 2011]

- IP Reminder
  - The TTL header field **limits** the *number of forwards* of an IP packet
    (IPv6 name: *Hop Limit*)
  - TTL value is *decreased* by each router *before* forwarding



  - If TTL == 0, packet is **dropped**
  - Observed default TTL values [Siby 2014]:

OS	TTL
*nix	64
Windows	128
Solaris	254

(you can easily check with `ping -4 localhost`)

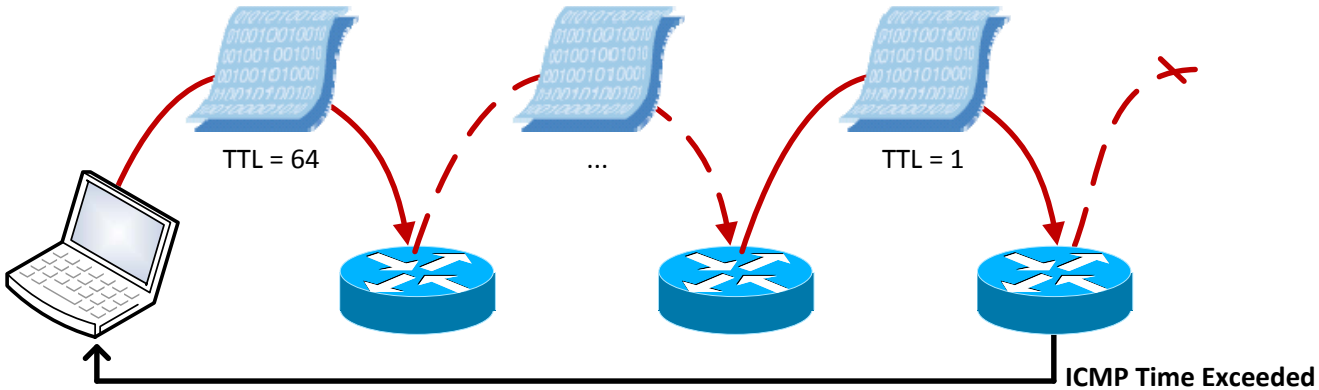
ICMP Error Messages: Time Exceeded II

[Fall *et al.* 2011]

- ICMPv4 type 11
  Time Exceeded

  ICMPv6 type 3
  Time Exceeded

  — Generated by a *router* when it discards a datagram because the TTL value is too low (i.e. arrives with value 0 or 1 and must be forwarded)



ICMP Error Messages: Parameter Problem

[Fall *et al.* 2011]

- ICMPv4 type 12
  Parameter Problem

  ICMPv6 type 4
  Parameter Problem

  — Generated by a *router* or *host* when receiving an IP datagram containing some problem in the IP header

  — In other words, when an IP datagram cannot be processed and no other ICMP message is adequate

  — The ICMP message contains a *pointer* to the byte where the error is located

  (+ copy of original IP datagram, etc., as previously)

Generation of ICMP Error Messages

[Fall *et al.* 2011]

- ICMP error messages could lead to **broadcast storms**
  - i.e. a small number of (maliciously generated) IP datagrams could generate a cascade of ICMP errors
- To avoid this, ICMP errors *are* **not** *sent* as replies to:
  - another ICMP error message
  - an IP datagram with bad header (e.g. bad checksum)
  - broadcast or multicast datagrams
  - encapsulated datagrams
  - datagrams with an invalid or zero source address
  - any fragment other than the first one

Rate Limiting of ICMP Error Messages

[Fall *et al.* 2011]

- [RFC4443] *recommends* a **token bucket** to limit the number of generated ICMP messages
  - Buckets holds B tokens, filled at rate N
  - 1 token is used per sent ICMP message

```
[q@genie ~]$ sudo sysctl -a | grep icmp
net.ipv4.icmp_echo_ignore_all = 0
net.ipv4.icmp_echo_ignore_broadcasts = 1
net.ipv4.icmp_errors_use_inbound_ifaddr = 0
net.ipv4.icmp_ignore_bogus_error_responses = 1
net.ipv4.icmp_msgs_burst = 50
net.ipv4.icmp_msgs_per_sec = 1000
net.ipv4.icmp_ratelimit = 1000
net.ipv4.icmp_ratemask = 6168
net.ipv6.icmp.ratelimit = 1000
```

ICMP Informational Messages

[Fall *et al.* 2011]

- **ICMP Informational Messages** convey *information* about **hosts** and/or **networks**
  - ICMPv4: types 0, 8, 9, 10, 13–18
  - ICMPv6: types 128–255
- Based on a query/response exchange
- The content of the ICMP response messages depends on the specific type

ICMP Informational Messages: Echo Request/Reply [Fall *et al.* 2011]

- ICMPv4 type 8
  Echo Request
  ICMPv4 type 0
  Reply

  ICMPv6 type 128
  Echo Request
  ICMPv6 type 129
  Reply

  — The *reply* is generated upon receipt of the *echo request*

  — Requests contain an *identifier*, a *sequence number*, and arbitrary data
  (size limited by IP)

  — Replies contain the same fields, and data
  (even if it leads to fragmentation)

ICMP Informational Messages: Other Uses

[Fall *et al.* 2011]

ICMP Queries/Informations are also used in other situations:

- Router Discovery: learn about all routers on network
  - Alternative to DHCPv4
  - **Mobile IP** (v4 and v6): ICMP with extensions
    - to discover home/foreign agents
    - to update agent addresses (ICMPv6)
    - to handle handovers (ICMPv6)
- Multicast management
- **IPv6 Neighbour Discovery** (equivalent and more powerful than IPv4's ARP)

ICMP: A few last Words

[Fall *et al.* 2011]

- It is considered *good practice* to handle ICMP correctly
- For **security** reasons, many network administrators **block** incoming/outgoing ICMP messages
  ... or **replace** an ICMP message with another one

$\Rightarrow$ ICMP messages can help you understand what's going on
but their content (or absence) is to be reviewed critically!

Outline

- Motivation
- ICMP

$\Rightarrow$ **Tools**

Packet Sniffers

- **Packet sniffers** (or *packet analyzers*) are a piece of *software* or *hardware* able to **dump** all packets transmitted on a medium
  - simplest form: a wire tap (from the telephone days. . .)
- Can be achieved
  - on a computer, e.g. with **tcpdump**
  - on a switch/router, e.g. using **port mirroring**
- Allows to
  - diagnose network problems
  - analyze implementation behaviour
  - monitor network usage (e.g. NetFlow)
  - . . .

tcpdump: Introduction

- *tcpdump* is (arguably) the most ubiquitous packet analyzer
- Originally developed in 1988 at Berkeley Lab
- **libpcap**
  - underlying raw packet capture library
  - can specify **capture filters** using the *pcap filter syntax* (`man pcap-filter`)
  - written in C
- **tcpdump**: CLI front-end to `libpcap`
  - able to do (limited) packet analysis (but good for basic/common protocols)
  - able to record `.pcap` dumps (many are available for teaching/research)
  - written in C

libpcap: Filter Syntax Basics I

- **Filters** out packets captured by libpcap
- All documentation in man pages ⇒ `man pcap-filter`
- Restrict per IP or hostname:

```
[dst|src] host <ip address|hostname>
```

  — restricts capture to IP packets sent/received by host vsignet:

```
host vsignet
```

  — restricts capture to IP packets sent by host 1.2.3.4:

```
src host 1.2.3.4
```

libpcap: Filter Syntax Basics II

- Restrict to port numbers:

```
[src|dst] port <number>
[src|dst] portrange port1-port2
```

  — restrict capture to *incoming* HTTP and HTTPs:

```
dst port 80 or dst port 443
```

  — restricts capture to FTP command and data:

```
port-range 20-21
```

- Restrict to specific L4 protocol:

```
ip protoco <icmp,icmp6,...,udp,tcp>
```

  — only show ICMP traffic: `ip proto icmp` or `icmp`

  — do not show UDP: `not udp` or `not ip proto udp`

tcpdump: Basics

- All documentation in man pages ⇒ `man tcpdump`
- Show list of available interfaces: `-D`
- Specify capture interface: `-i`
  e.g. `-i eth0` `-i any`
- Don't resolve hostnames: `-n`
- Don't resolve hostnames nor portnames: `-nn`
- Less verbose: `-q`
- More verbose: `-v` and `-vv` and `-vvv`

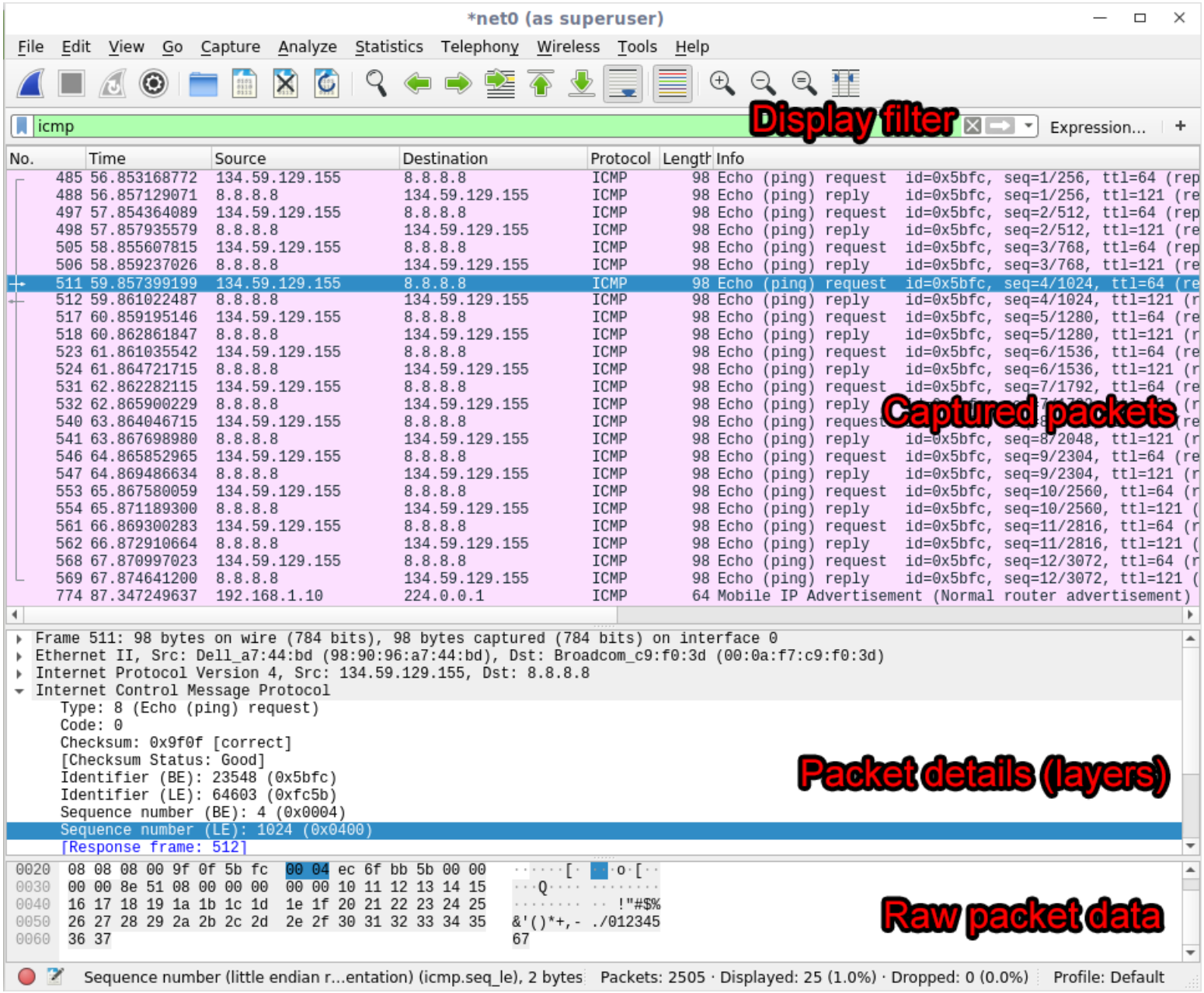
tcpdump: Output Example

```
q@steam:~$ sudo tcpdump -i eth0 -n not port 22
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
18:55:07.663366 ARP, Request who-has 134.59.130.159 tell 134.59.130.254, length 46
18:55:07.900640 ARP, Request who-has 134.59.130.147 tell 134.59.130.254, length 46
18:55:08.676714 ARP, Request who-has 134.59.130.159 tell 134.59.130.254, length 46
18:55:08.920013 ARP, Request who-has 134.59.130.147 tell 134.59.130.254, length 46
18:55:09.930051 ARP, Request who-has 134.59.130.147 tell 134.59.130.254, length 46
18:55:11.730418 IP 134.59.130.135.5353 > 224.0.0.251.5353: 0 [2q] [2n] ANY (QM)? 0.
18:55:11.730525 IP 134.59.130.135.5353 > 224.0.0.251.5353: 0*- [0q] 3/0/0 (Cache fl
18:55:11.731120 IP 134.59.130.162.5353 > 224.0.0.251.5353: 0*- [0q] 1/0/0 (Cache fl
18:55:11.733284 IP 134.59.130.135.5353 > 224.0.0.251.5353: 0*- [0q] 5/0/0 (Cache fl
18:55:11.853544 IP 134.59.130.135.5353 > 224.0.0.251.5353: 0 [3q] [5n] ANY (QM)? 4.
18:55:12.104183 IP 134.59.130.135.5353 > 224.0.0.251.5353: 0 [3q] [5n] ANY (QM)? 4.
18:55:12.354872 IP 134.59.130.135.5353 > 224.0.0.251.5353: 0 [3q] [5n] ANY (QM)? 4.
18:55:12.555247 IP 134.59.130.135.5353 > 224.0.0.251.5353: 0*- [0q] 5/0/0 (Cache fl
18:55:13.625045 IP 134.59.130.135.5353 > 224.0.0.251.5353: 0*- [0q] 5/0/0 (Cache fl
18:55:15.696490 IP 134.59.130.135.5353 > 224.0.0.251.5353: 0*- [0q] 5/0/0 (Cache fl
18:55:16.836748 ARP, Request who-has 134.59.130.200 tell 134.59.130.254, length 46
18:55:16.836762 ARP, Reply 134.59.130.200 is-at b0:83:fe:e6:82:e8, length 28
^C
17 packets captured
17 packets received by filter
0 packets dropped by kernel
```

Wireshark I

- **Wireshark** is a full-featured graphical packet analyzer
- Originally called *Ethereal*
- Uses `libpcap` as underlying capture library
- More protocol support than `tcpdump` (also less lightweight...)
- Many useful functions to analyze flows
  - Flow statistics
  - Display filters/colorization
  - Easy analysis of each layer independently
  - Following upper layer conversation through multiple packets
- Don't forget to run as superuser

Wireshark II



ping

- **ping** is the *best-known* network tool
- Written in December 1983 by Mike Muuss [Muuss 1997]
- Relies on *ICMP packets* exchanges
  - **Echo request** and **Reply**
  - Measures the *round-trip time* (**RTT**) between two network nodes

```
q@astrid:~$ ping www.unice.fr
PING sites.unice.fr (134.59.204.9) 56(84) bytes of data.
64 bytes from sites.unice.fr (134.59.204.9): icmp_seq=1 ttl=49 time=51.9ms
64 bytes from sites.unice.fr (134.59.204.9): icmp_seq=2 ttl=49 time=55.8ms
64 bytes from sites.unice.fr (134.59.204.9): icmp_seq=3 ttl=49 time=52.4ms
64 bytes from sites.unice.fr (134.59.204.9): icmp_seq=4 ttl=49 time=51.3ms
^C
--- sites.unice.fr ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 51.316/52.897/55.846/1.756 ms
```

ping: Some Options

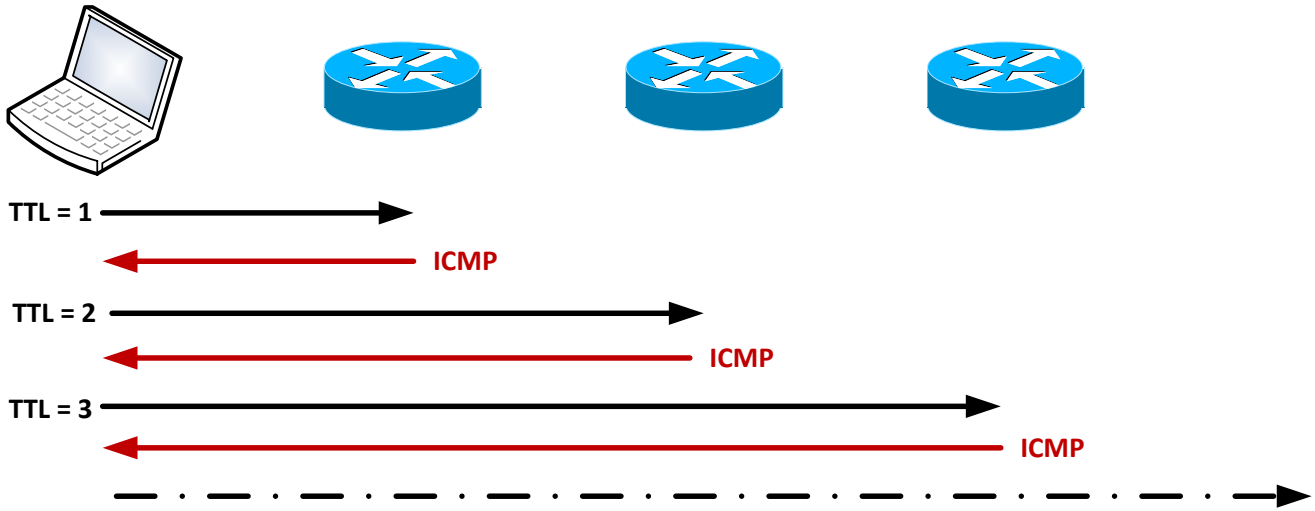
- All documentation in man pages $\Rightarrow$ `man ping`
- `-b`: ping to a broadcast address
- `-c` N: stop after N probes have been sent
- `-D`: print (a Unix) timestamp on each line
- `-i` t: wait t seconds between sending each Echo Request
- `-I interf`: specify interface to send the probes from
- `-n`: do not resolve hostnames
- `-W` t: specify a timeout of t seconds to receive the ICMP Reply

ping: ICMP Message Contents

```
[q@oufti ~]$ ping steam.i3s.unice.fr
PING steam.i3s.unice.fr (134.59.130.200) 56(84) bytes of data.
64 bytes from steam.i3s.unice.fr (134.59.130.200): icmp_seq=1 ttl=63 time=0.662 ms
64 bytes from steam.i3s.unice.fr (134.59.130.200): icmp_seq=2 ttl=63 time=0.694 ms
64 bytes from steam.i3s.unice.fr (134.59.130.200): icmp_seq=3 ttl=63 time=0.707 ms
64 bytes from steam.i3s.unice.fr (134.59.130.200): icmp_seq=4 ttl=63 time=0.743 ms
64 bytes from steam.i3s.unice.fr (134.59.130.200): icmp_seq=5 ttl=63 time=0.697 ms
64 bytes from steam.i3s.unice.fr (134.59.130.200): icmp_seq=6 ttl=63 time=0.708 ms
^C
--- steam.i3s.unice.fr ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5063ms
rtt min/avg/max/mdev = 0.662/0.701/0.743/0.041 ms



[q@oufti ~]$ ps -e | grep ping
23993 pts/7    00:00:00 ping



[q@oufti ~]$ sudo tcpdump -i net0 -n -v icmp
[sudo] password for q:
tcpdump: listening on net0, link-type EN10MB (Ethernet), capture size 262144 bytes
19:44:31.452249 IP (tos 0x0, ttl 64, id 60501, offset 0, flags [DF], proto ICMP (1), length 84)
    134.59.129.155 > 134.59.130.200: ICMP echo request, id 23993, seq 1, length 64
19:44:31.452902 IP (tos 0x0, ttl 63, id 3163, offset 0, flags [none], proto ICMP (1), length 84)
    134.59.130.200 > 134.59.129.155: ICMP echo reply, id 23993, seq 1, length 64
19:44:32.453753 IP (tos 0x0, ttl 64, id 60779, offset 0, flags [DF], proto ICMP (1), length 84)
    134.59.129.155 > 134.59.130.200: ICMP echo request, id 23993, seq 2, length 64
19:44:32.454418 IP (tos 0x0, ttl 63, id 3299, offset 0, flags [none], proto ICMP (1), length 84)
    134.59.130.200 > 134.59.129.155: ICMP echo reply, id 23993, seq 2, length 64
```

traceroute I

- **traceroute** is used to determine the routers used along a path
  - *Sends* IP packets with **increasing TTL**s
  - *Waits* for ICMP **Time Exceeded** responses



  - By default: 3 probes for each TTL value

traceroute II

- Termination condition:
  - ICMP Port Unreachable is received
  - TCP RST is received
  - Maximum TTL value is reached (by default: 30 hops)
- Network administrators *should* let their routers generate ICMP Time Exceeded errors
  - in practice...

traceroute: Output Example I

```
q@astrid:~$ traceroute -m 20 www.unice.fr
traceroute to www.unice.fr (134.59.204.9), 20 hops max, 60 byte packets
 1  192.168.1.1 (192.168.1.1)  0.990 ms  1.222 ms  1.494 ms
 2  1.196-4-62.wifi-dyn.isp.proximus.be (62.4.196.1)  25.328 ms  26.137 ms  35.273 ms
 3  * * *
 4  * * *
 5  brx-b1-link.telia.net (62.115.40.121)  38.200 ms  40.038 ms  40.667 ms
 6  adm-bb3-link.telia.net (62.115.117.246)  47.465 ms adm-bb3-link.telia.net (62.115.121.68) 29.469 ms
                                                      adm-bb4-link.telia.net (62.115.136.90)  30.146 ms
 7  adm-b5-link.telia.net (62.115.140.195)  32.385 ms adm-b5-link.telia.net (213.155.132.159)  32.864 ms
                                                      adm-b5-link.telia.net (62.115.116.199)  37.741 ms
 8  cogent-ic-143007-adm-b5.c.telia.net (80.239.160.14)  38.569 ms  38.947 ms  40.801 ms
 9  be2312.ccr42.ams03.atlas.cogentco.com (154.54.74.93)  43.784 ms  45.869 ms  46.144 ms
10  be12266.ccr42.par01.atlas.cogentco.com (154.54.56.174)  37.445 ms  37.435 ms  37.660 ms
11  be3093.ccr22.mrs01.atlas.cogentco.com (130.117.50.166)  49.808 ms  49.746 ms  52.504 ms
12  geant.demarc.cogentco.com (149.6.154.202)  51.351 ms  51.528 ms  55.750 ms
13  193.51.177.212 (193.51.177.212)  56.267 ms 193.51.177.184 (193.51.177.184)  51.350 ms  53.415 ms
14  193.51.177.21 (193.51.177.21)  52.001 ms 193.51.177.71 (193.51.177.71)  54.836 ms  54.750 ms
15  * * *
16  man-uns-vl979-gi8-5-nice-rtr-021.noc.renater.fr (193.51.191.9)  52.907 ms  53.043 ms  51.190 ms
17  * * *
18  * * *
19  * * *
20  * * *
```

traceroute: Output Example II

On this example:

- Each output line contains:
  - the TTL value
  - the intermediate hop's hostname (if available) and IP address
  - the probe's RTT to the intermediate hop
- A star ('*') is shown if no ICMP error is received
  - Default timeout value: 5 seconds
  - Q: how would you appropriately choose the timeout value?
    Clue: How long does it take for a packet to go and come back between 2 devices?
  - Q: Why wouldn't we receive ICMP errors?
- Load balancing on links 5–6 and 6–7

traceroute: Bursty Behaviour I

```
q@steam:~$ traceroute -m 10 109.130.242.114
traceroute to 109.130.242.114 (109.130.242.114), 10 hops max, 60 byte packets
 1  * * *
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
```

traceroute: Bursty Behaviour II

```
q@steam:~$ sudo tcpdump -i eth0 -n -v host 109.130.242.114
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
00:15:14.216052 IP (tos 0x0, ttl 1, id 22944, offset 0, flags [none], proto UDP (17), length 60)
    134.59.130.200.56783 > 109.130.242.114.33434: UDP, length 32
00:15:14.216076 IP (tos 0x0, ttl 1, id 22945, offset 0, flags [none], proto UDP (17), length 60)
    134.59.130.200.36367 > 109.130.242.114.33435: UDP, length 32
00:15:14.216097 IP (tos 0x0, ttl 1, id 22946, offset 0, flags [none], proto UDP (17), length 60)
    134.59.130.200.37047 > 109.130.242.114.33436: UDP, length 32
00:15:14.216106 IP (tos 0x0, ttl 2, id 22947, offset 0, flags [none], proto UDP (17), length 60)
    134.59.130.200.48889 > 109.130.242.114.33437: UDP, length 32
00:15:14.216121 IP (tos 0x0, ttl 2, id 22948, offset 0, flags [none], proto UDP (17), length 60)
    134.59.130.200.32852 > 109.130.242.114.33438: UDP, length 32
00:15:14.216134 IP (tos 0x0, ttl 2, id 22949, offset 0, flags [none], proto UDP (17), length 60)
    134.59.130.200.43417 > 109.130.242.114.33439: UDP, length 32
00:15:14.216147 IP (tos 0x0, ttl 3, id 22950, offset 0, flags [none], proto UDP (17), length 60)
    134.59.130.200.37927 > 109.130.242.114.33440: UDP, length 32
00:15:14.216217 IP (tos 0x0, ttl 4, id 22955, offset 0, flags [none], proto UDP (17), length 60)
    134.59.130.200.42632 > 109.130.242.114.33445: UDP, length 32
00:15:14.216246 IP (tos 0x0, ttl 5, id 22957, offset 0, flags [none], proto UDP (17), length 60)
    134.59.130.200.42769 > 109.130.242.114.33447: UDP, length 32
00:15:14.216275 IP (tos 0x0, ttl 6, id 22959, offset 0, flags [none], proto UDP (17), length 60)
    134.59.130.200.55042 > 109.130.242.114.33449: UDP, length 32
00:15:19.221527 IP (tos 0x0, ttl 6, id 23204, offset 0, flags [none], proto UDP (17), length 60)
    134.59.130.200.60271 > 109.130.242.114.33450: UDP, length 32
00:15:19.221547 IP (tos 0x0, ttl 6, id 23205, offset 0, flags [none], proto UDP (17), length 60)
    134.59.130.200.48949 > 109.130.242.114.33451: UDP, length 32
00:15:19.221561 IP (tos 0x0, ttl 7, id 23206, offset 0, flags [none], proto UDP (17), length 60)
    134.59.130.200.34344 > 109.130.242.114.33452: UDP, length 32
00:15:19.221575 IP (tos 0x0, ttl 7, id 23207, offset 0, flags [none], proto UDP (17), length 60)
    134.59.130.200.37132 > 109.130.242.114.33453: UDP, length 32
00:15:19.221588 IP (tos 0x0, ttl 7, id 23208, offset 0, flags [none], proto UDP (17), length 60)
    134.59.130.200.46167 > 109.130.242.114.33454: UDP, length 32
00:15:19.221600 IP (tos 0x0, ttl 8, id 23209, offset 0, flags [none], proto UDP (17), length 60)
```

traceroute: Bursty Behaviour III

`traceroute` sends packets in burst ⇒ generates ICMP errors in burst

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33435
3	0.010658	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33436
5	0.010995	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33437
7	0.011270	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33438
8	5.016777	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33439
9	10.022135	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33440
10	15.027686	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33441
11	20.032483	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33442
12	25.037641	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33443
13	30.043155	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33444
14	35.048657	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33445
15	40.054008	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33446
16	45.059907	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33447
17	50.063033	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33448
18	55.068490	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33449
19	60.074476	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33450
20	65.076497	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33451
21	70.085479	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33452
22	75.091457	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33453
23	80.097333	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33454
24	85.102285	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33455
25	90.108209	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33456
26	95.114198	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33457
27	100.120076	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33458
28	105.123853	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33459
29	110.129077	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33460
30	115.142311	192.168.96.128	134.59.204.9	UDP	Source port: 61321 Destination port: 33461

⇒ Easily triggers **rate throttling**

traceroute the Smart Way I

- Many factors impact the results obtained by traceroute
  - Uncooperative firewalls
  - Rate throttling

⇒ How to **positively impact** the topology discovery?

- Avoid rate throttling
  - option `-N squeries` specifies the number of probes sent simultaneously (default: 16)
  - option `-q nqueries` specifies the number of probes sent per TTL value (default: 3)
  - option `-z sendwait` specifies the minimal time interval between probes (default: 0)

traceroute the Smart Way II

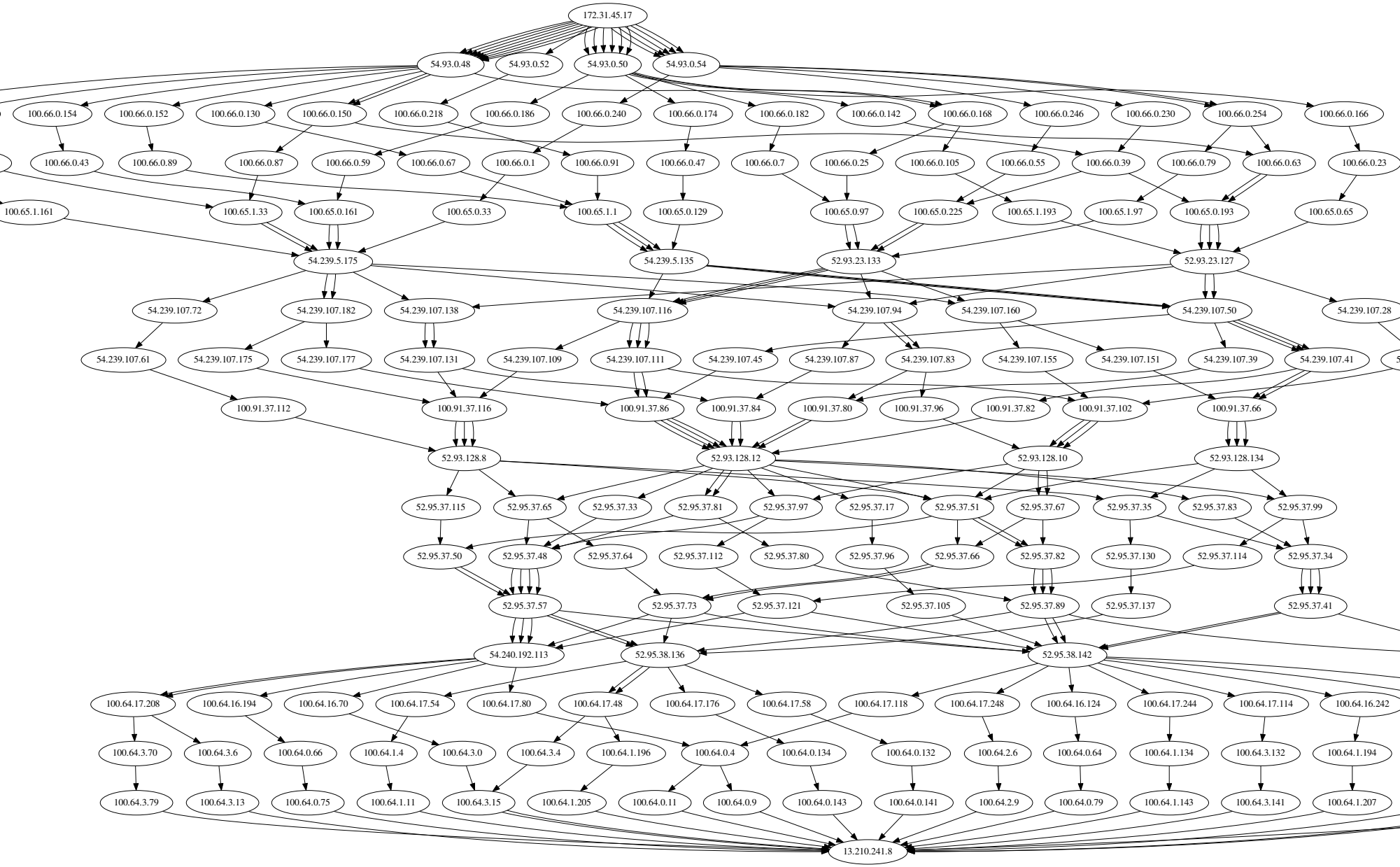
Which L4 packet to embed in IP?

- TTL expiration and ICMP generation depend on IP
- Should we probe using TCP, UDP, or ICMP?
- traceroute *implements* **all** of them
  - option `-U` uses UDP packets (default)
  - option `-T` uses TCP packets
  - option `-I` uses ICMP packets
- Why does it matter?
  - Firewalls treat TCP, UDP, and ICMP differently

traceroute the Smart Way III

- Many more options in the man page $\Rightarrow$ `man traceroute`

traceroute: A Real-World (Virtual) Topology



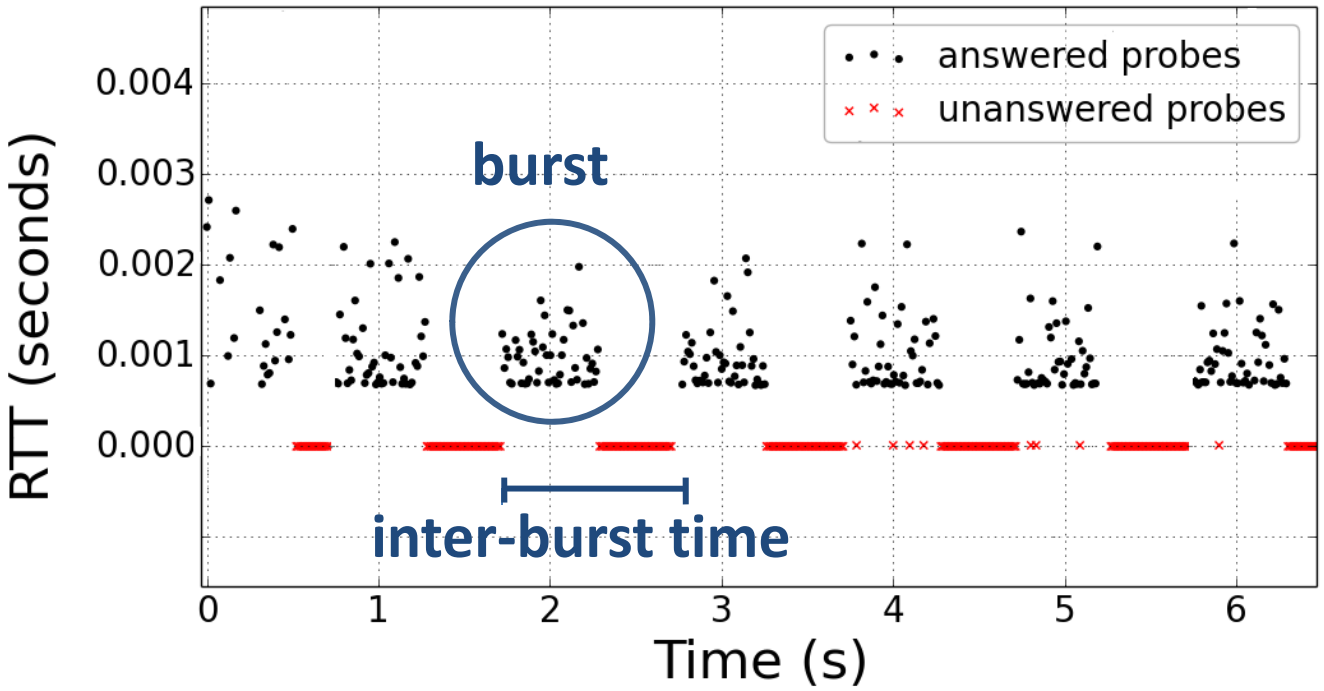
Impact of traceroute on routers

- traceroute relies on the generation of ICMP error packets
- These are generated by routers **slow path**
  - i.e. the general CPU
  - very slow compared to packet forwarding ASICs
- Question: knowing that traceroute is bursty,
  - what is the impact of probing routers with traceroute?
  - do the results depend on the probing rate?
  - $\Rightarrow$ See [Ravaioli *et al.* 2015]

Responsiveness to TTL-limited probes

- At any given probing rate, routers are
  - Unresponsive **(3.9 %)**
  - Rate-limited **(65.9%)**
  - Fully responsive **(30.2 %)**
- ICMP rate-limitation appears as:

Characterizing ICMP Rate Limitation on Routers, ICC '15

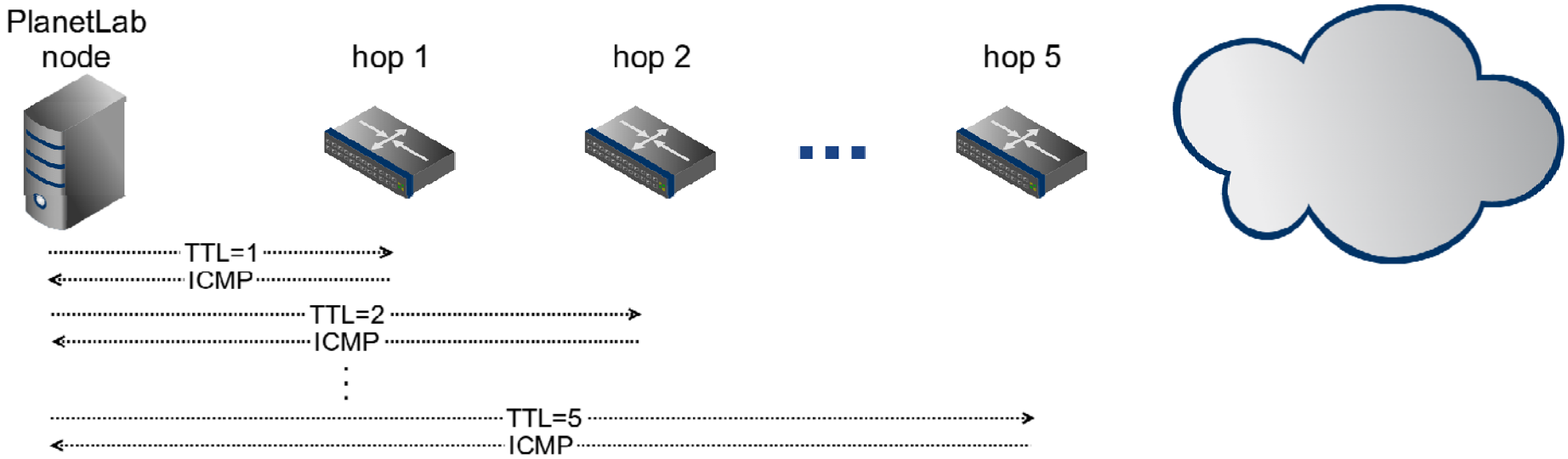


Large-scale campaign to characterize routers

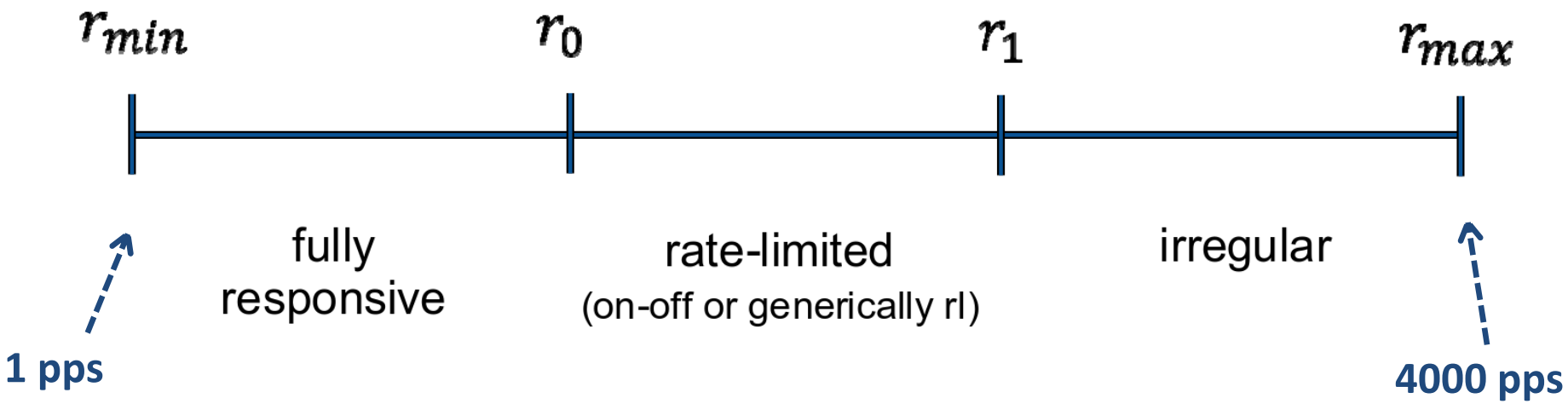
- Probed 850 routers at hops 1-5 from PL nodes

Measurement Campaign

- Tested 850 routers from 180 PlanetLab sites to a fixed IP destination
- Targeted hops 1 to 5
- 3 runs of 30s probing experiments at a constant rate to each router
- Exponentially-spaced rates in [1, 4000] pps



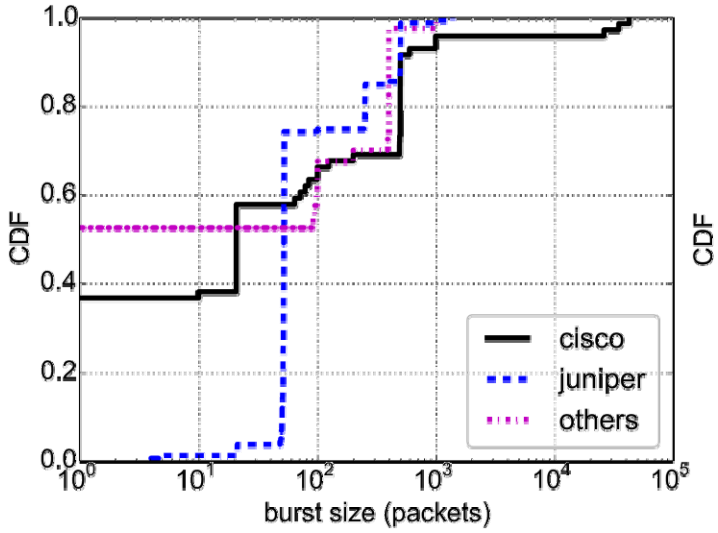
Router Responsiveness

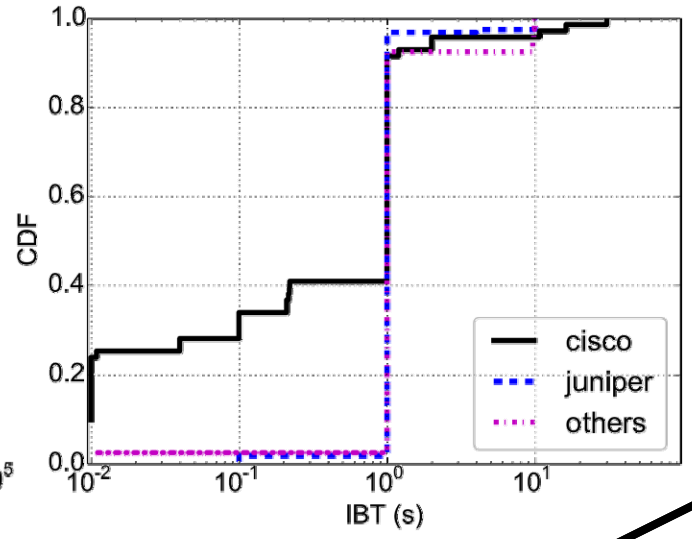


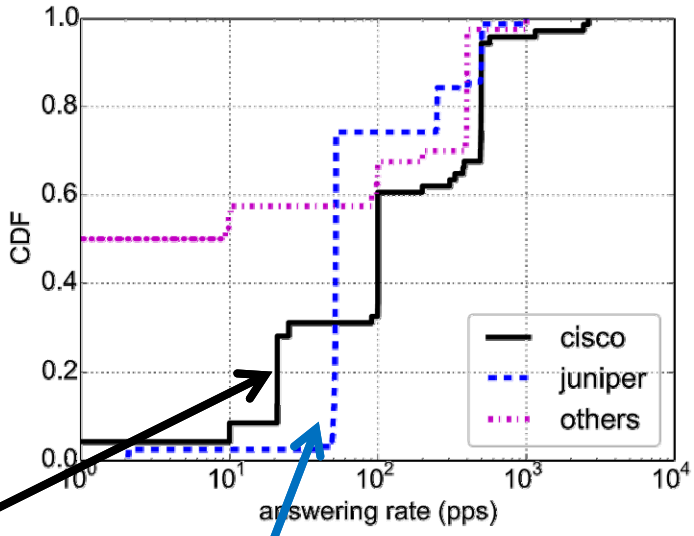
- **Rate-limited** routers [59.9 %]
  - Fully responsive - on off: 24.8 %
  - Fully responsive - on off - irregular: 21.2 %
  - Fully responsive - generically rate limited: 13.9 %
- **Non rate-limited** routers [36.2 %]
  - Fully responsive: 30.2 %
  - Fully responsive – irregular 6.0 %
- **Unresponsive** routers [3.9 %]

ICMP rate limitation: characterization

- TTL-fingerprinting on routers to get vendor name
  - Cisco (59%), Juniper (30.5%), others (10.5%)
- Analysis of rate-limitation parameters
  - Burst size, inter-burst time, answering rate





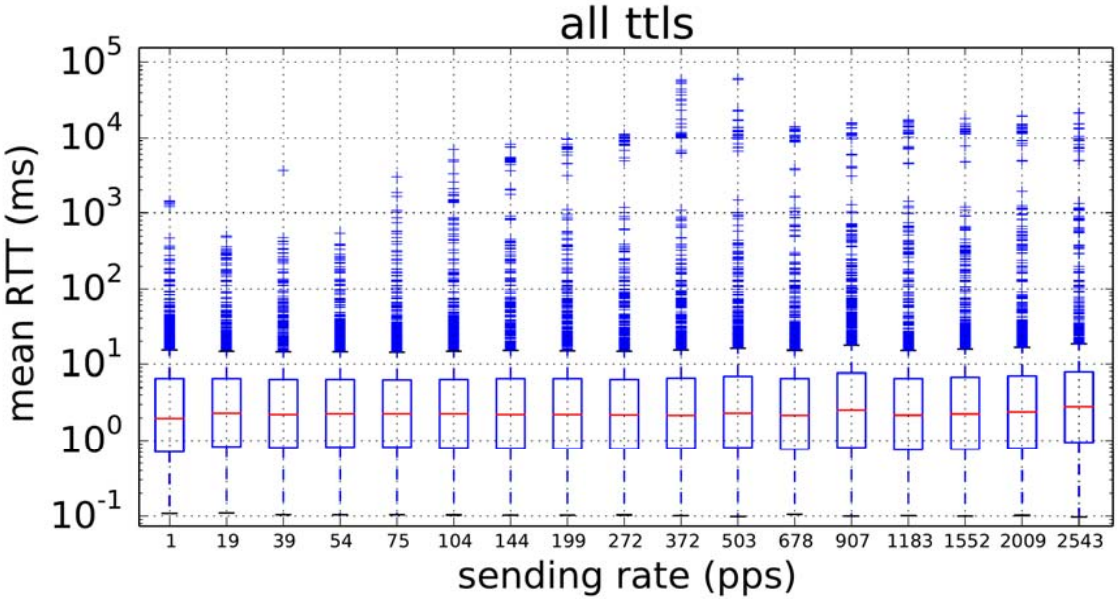


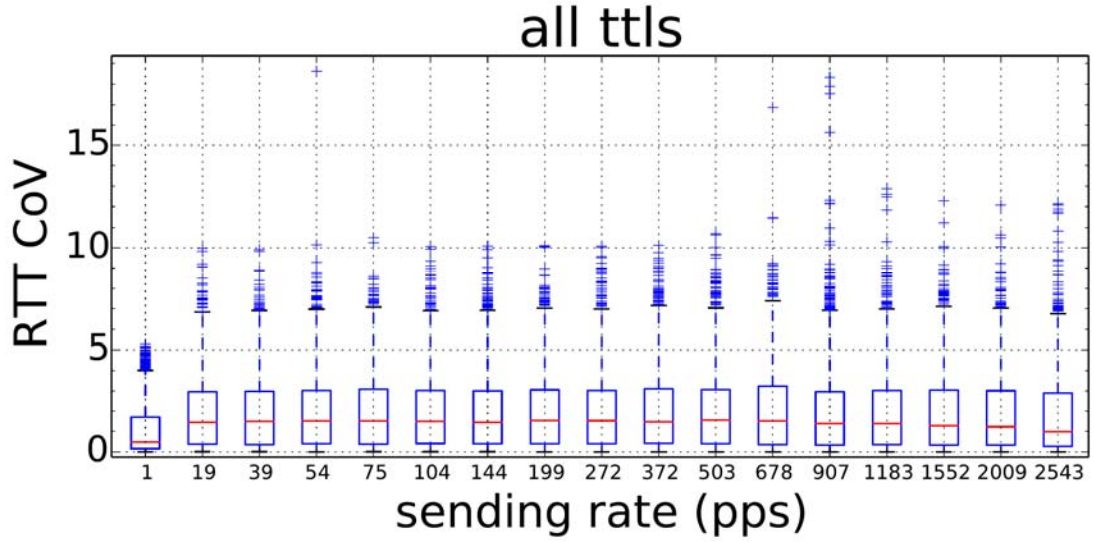
20, 100 and 500 pps for most cisco routers

50 pps for 75% of Juniper

ICMP rate limitation: delays

- Are RTTs biased by our probing rate?

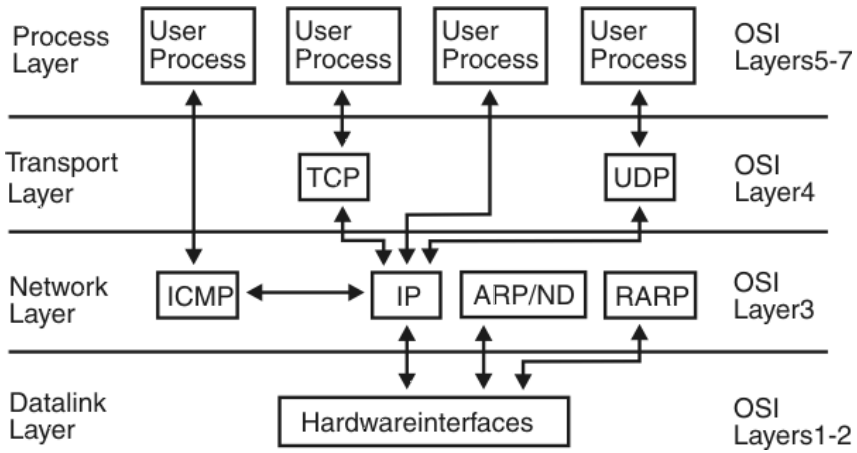




- No correlation between probing rate and resulting RTT
- Not hitting any capacity limits of routers

Going Further Upwards

- ping and traceroute probe machines on the IP-level (layer 3)
- What about other layers?



ping on Other Layers

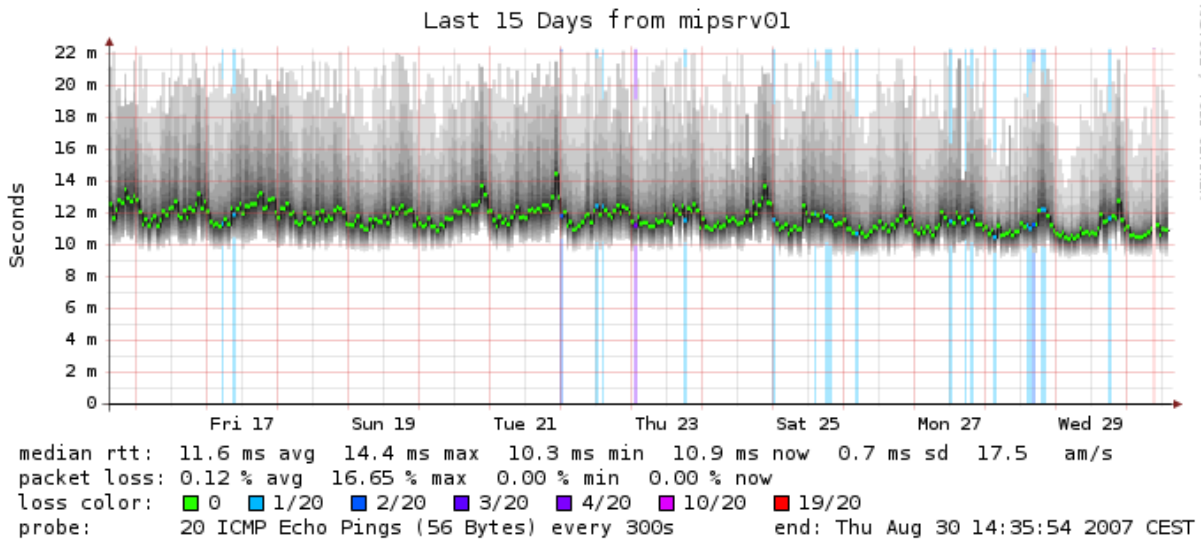
- arping: ping on layer 2
  - Sends an ARP REQUEST packet
  - Waits for the corresponding ARP REPLY packet
- httping: ping on application layer
  - Measures the time needed to connect to a HTTP server
  - Retrieves default page headers (but no page content)
- Question:
  - How do you think these measurements compare to the standard ICMP ping RTTs?

SmokePing I

- SmokePing **regularly** probes a target
- Stores results in a *Round-Robbin Database* (**RRD**)
  - Optimized to store temporal data
  - Data granularity varies: old data is less precise, new data is raw
  - Advantage: uses a constant storage space on disk
- SmokePign allows to
  - ping on HTTP level $\rightarrow$ curl on a web page
  - ping on TLS level $\rightarrow$ TLS handshake with a server
  - ping on DNS level $\rightarrow$ sends a DNS request
  - other applications/protocols: LDAP, NFS, ...

SmokePing II

- Output is typically a latency graph



Bibliography I

[Fall *et al.* 2011]
K. R. Fall and W. R. Stevens, *TCP/IP Illustrated – The Protocols*, 2nd ed. Addison-Wesley, 2011, vol. 1.

[K&R]
J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 6th. Addison-Wesley, 2012.

[Muuss 1997]
M. Muuss, *The story of the PING program*, http://ftp.arl.mil/mike/ping.html, 1997.

[Ravaioli *et al.* 2015]
R. Ravaioli, G. Urvoy-Keller, and C. Barakat, "Characterizing ICMP rate limitation on routers", in *2015 IEEE International Conference on Communications, ICC 2015*, 2015, pp. 6043–6049.

Bibliography II

[RFC792]
J. Postel, *Internet Control Message Protocol*, RFC 792, Sep. 1981.

[RFC1122]
R. Braden (Ed.), *Requirements for Internet Hosts - Communication Layers*, RFC 1122, Oct. 1989.

[RFC1812]
F. Baker (Ed.), *Requirements for IP Version 4 Routers*, RFC 1812, Jun. 1995.

[RFC4443]
A. Conta, S. Deering, and M. Gupta (Ed.), *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*, RFC 4443, Mar. 2006.

Bibliography III

[RFC4884]
R. Bonica, D. Gan, D. Tappan, and C. Pignataro, *Extended ICMP to Support Multi-Part Messages*, RFC 4884, Apr. 2007.

[RFC4950]
——, *ICMP Extensions for Multiprotocol Label Switching*, RFC 4950, RFC, Aug. 2007.

[RFC5508]
P. Srisuresh, B. Ford, S. Sivakumar, and S. Guha, *NAT Behavioral Requirements for ICMP*, RFC 5508, Apr. 2009.

[Siby 2014]
S. Siby, *Default TTL (Time To Live) values of different os*, https://subinsb.com/default-device-ttl-values/, Apr. 2014.