



System Administration and IP Services

Performance Definitions and Measurement



These materials are licensed under the Creative Commons *Attribution-Noncommercial 3.0 Unported* license
(<http://creativecommons.org/licenses/by-nc/3.0/>)

Metrics

Network performance metrics

- Channel capacity, nominal & effective
- Channel utilization
- Delay and *jitter*
- Packet loss and errors

Nominal Channel Capacity

The maximum number of bits that can be transmitted on a unit of time (eg: bits per second)

Depends on:

- Bandwidth of the physical medium
 - Cable
 - Electromagnetic waves
- Processing capacity for each transmission element
- Efficiency of algorithms in use to access medium
- Channel encoding and compression

Effective Channel Capacity

Always a fraction of the nominal channel capacity

Dependent on:

- Additional overhead of protocols in each layer
- Device limitations on both ends
 - Flow control algorithm efficiency, etc.
 - For example: TCP

Channel Utilization

What fraction of the nominal channel capacity is actually in use

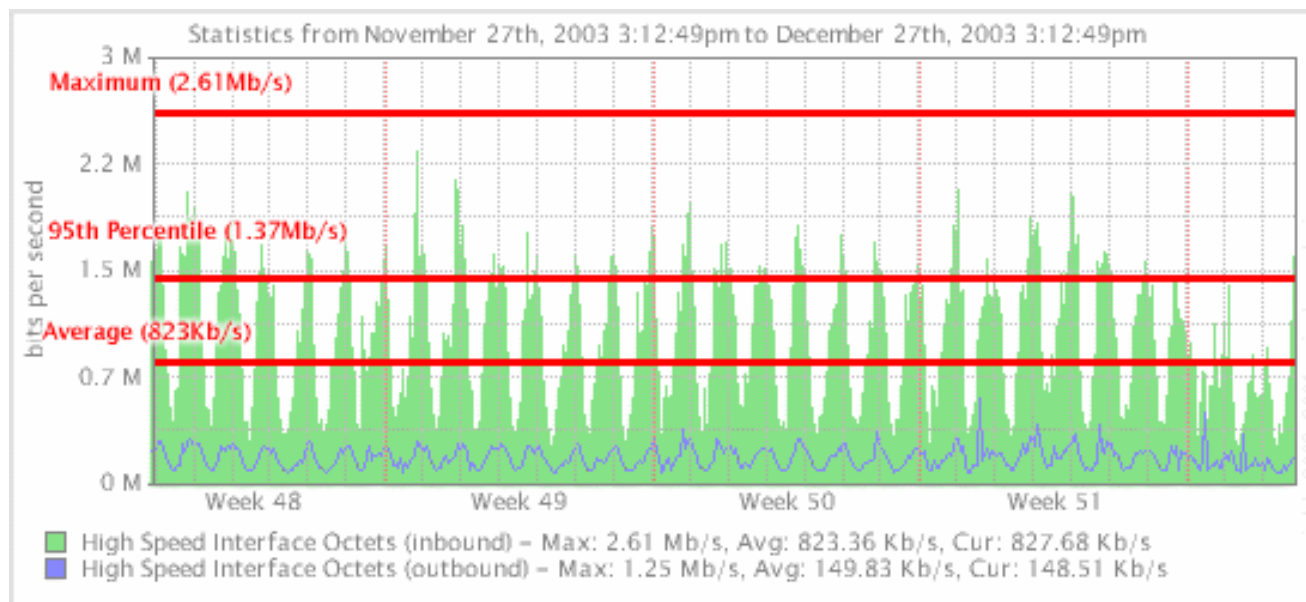
Important!

- Future planning
 - What utilization growth rate am I seeing?
 - For when should I plan on buying additional capacity?
 - Where should I invest for my updates?
- Problem resolution
 - Where are my bottlenecks, etc.

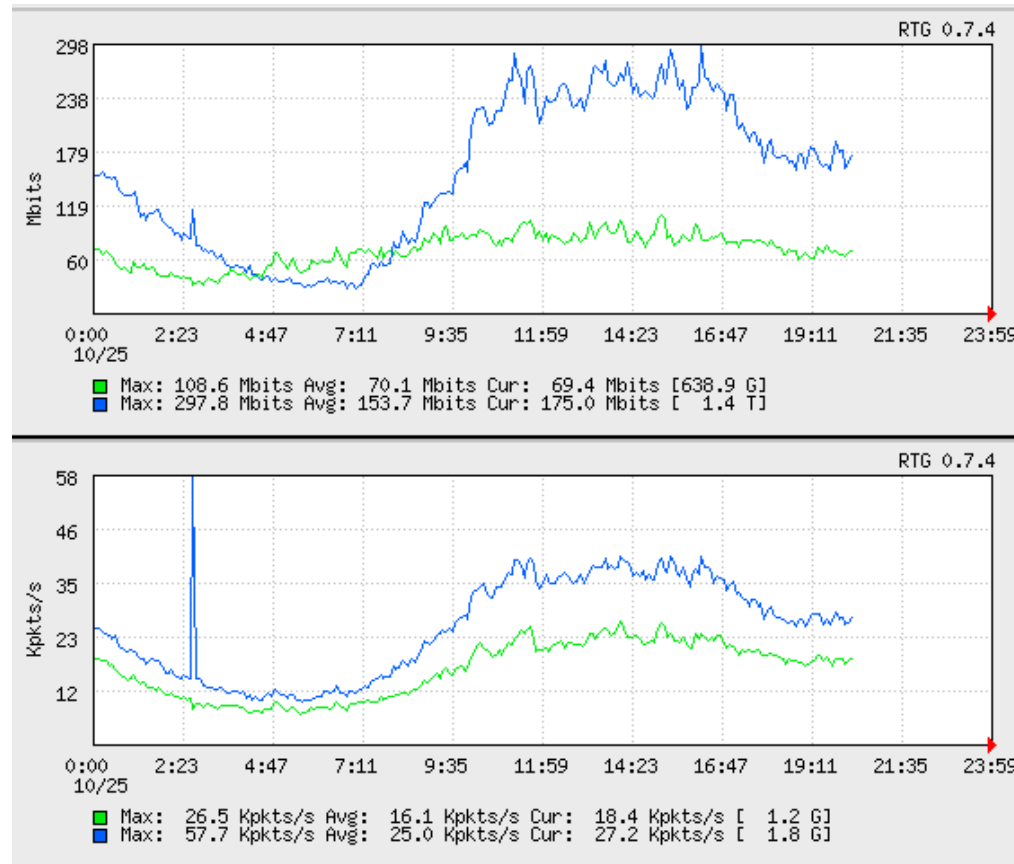
95th Percentile

- The smallest value that is larger than 95% of the values in a given sample
- This means that 95% of the time the channel utilization is equal to or *less* than this value
 - Or rather, the peaks are discarded from consideration
- Why is this important in networks?
 - Gives you an idea of the standard, sustained channel utilization.
 - ISPs use this measure to bill customers with “larger” connections.

95th Percentile



Bits per second vs Packets p.s.

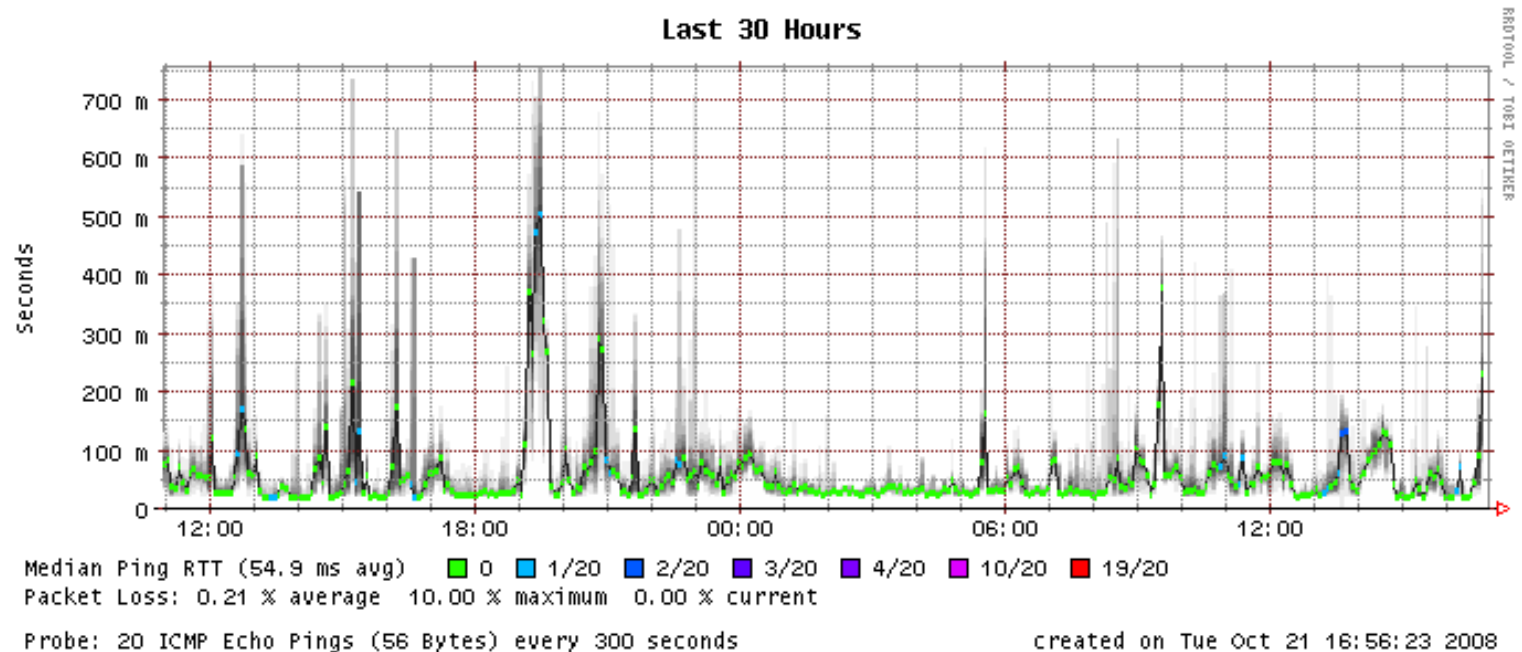


End-to-end Delay

The time required to transmit a packet along its entire path

- *Created by an application, handed over to the OS, passed to a network card (NIC), encoded, transmitted over a physical medium (copper, fibre, air), received by an intermediate device (switch, router), analyzed, retransmitted over another medium, etc.*
- The most common measurement uses *ping* for total round-trip-time (RTT).

Historical Measurement of Delay



Types of Delay

Causes of end-to-end delay:

- Processing delays
- Buffer delays
- Transmission delays
- Propagation delays

Processing Delay

Required time to analyze a packet header and decide where to send the packet (e.g. a routing decision)

Inside a router this depends on the number of entries in the routing table, the implementation of data structures, hardware in use, etc.

This can include error verification, such as IPv4, IPv6 header checksum calculations.

Queuing Delay

- The time a packet is enqueued until it is transmitted
- The number of packets waiting in the queue will depend on traffic intensity and of the type of traffic (bursty or sustained)
- Router queue algorithms try to adapt delays to specific preferences, or impose equal delay on all traffic.

Transmission Delay

The time required to push all the bits in a packet on the transmission medium in use

For N =Number of bits, S =Size of packet, d =delay

$$d = S/N$$

For example, to transmit 1024 bits using Fast Ethernet (100Mbps):

$$d = 1024/1 \times 10^8 = 10.24 \text{ micro seconds}$$

Propagation Delay

- Once a bit is 'pushed' on to the transmission medium, the time required for the bit to propagate to the end of its physical trajectory
- The velocity of propagation of the circuit depends mainly on the actual distance of the physical circuit
- In the majority of cases this is close to the speed of light.

For d = distance, s = propagation velocity

$$PD = d/s$$

Transmission vs. Propagation

Can be confusing at first

Consider this example:

Two 100 Mbps circuits

- 1 km of optic fiber
- Via satellite with a distance of 30 km between the base and the satellite

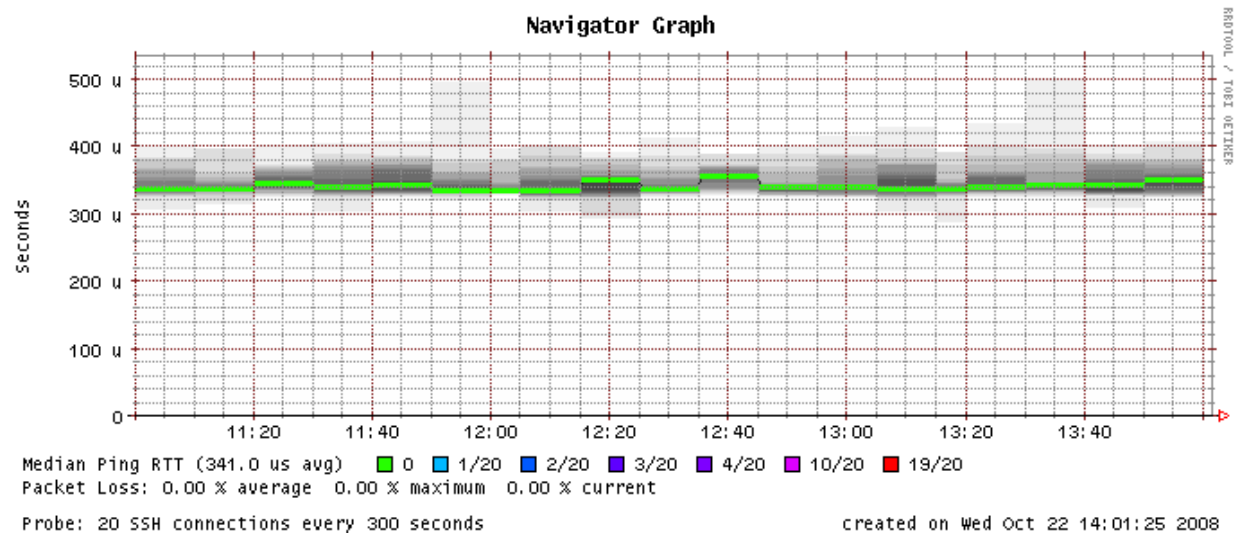
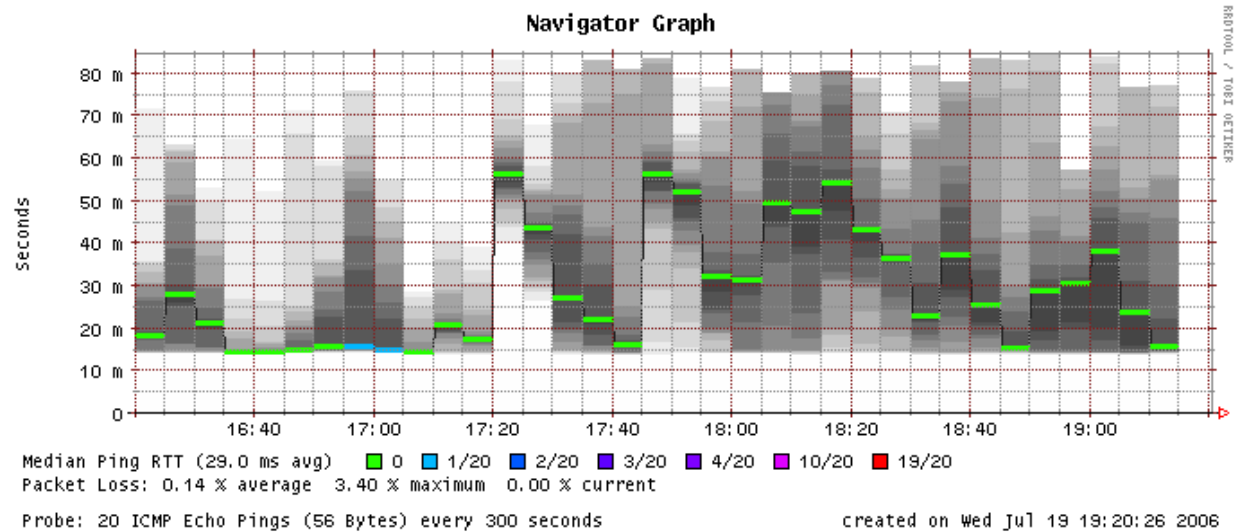
For two packets of the same size which will have the larger transmission delay?
Propagation delay?

Packet Loss

Occurs due to the fact that buffers are not infinite in size

- When a packet arrives to a buffer that is full the packet is discarded.
- Packet loss, if it must be corrected, is resolved at higher levels in the network stack (transport or application layers)
- Loss correction using retransmission of packets can cause yet more congestion if some type of (flow) control is not used (to inform the source that it's pointless to keep sending more packets at the present time)

Jitter



Questions

?

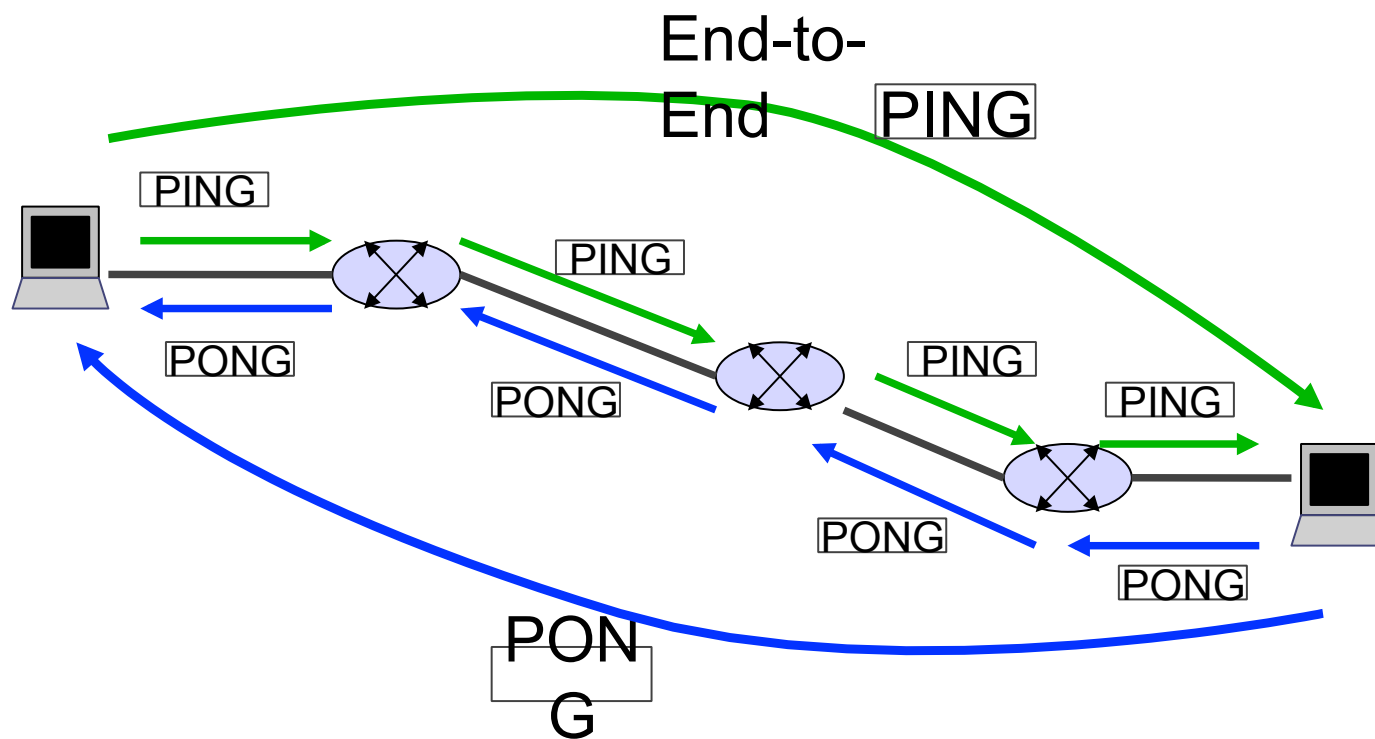
IP End-to-End principle

- IP is an end-to-end protocol
- The network doesn't keep track of connections
- The host takes a decision on where to send *each packet*
- The network equipment takes a decision on where to forward packets *every time*

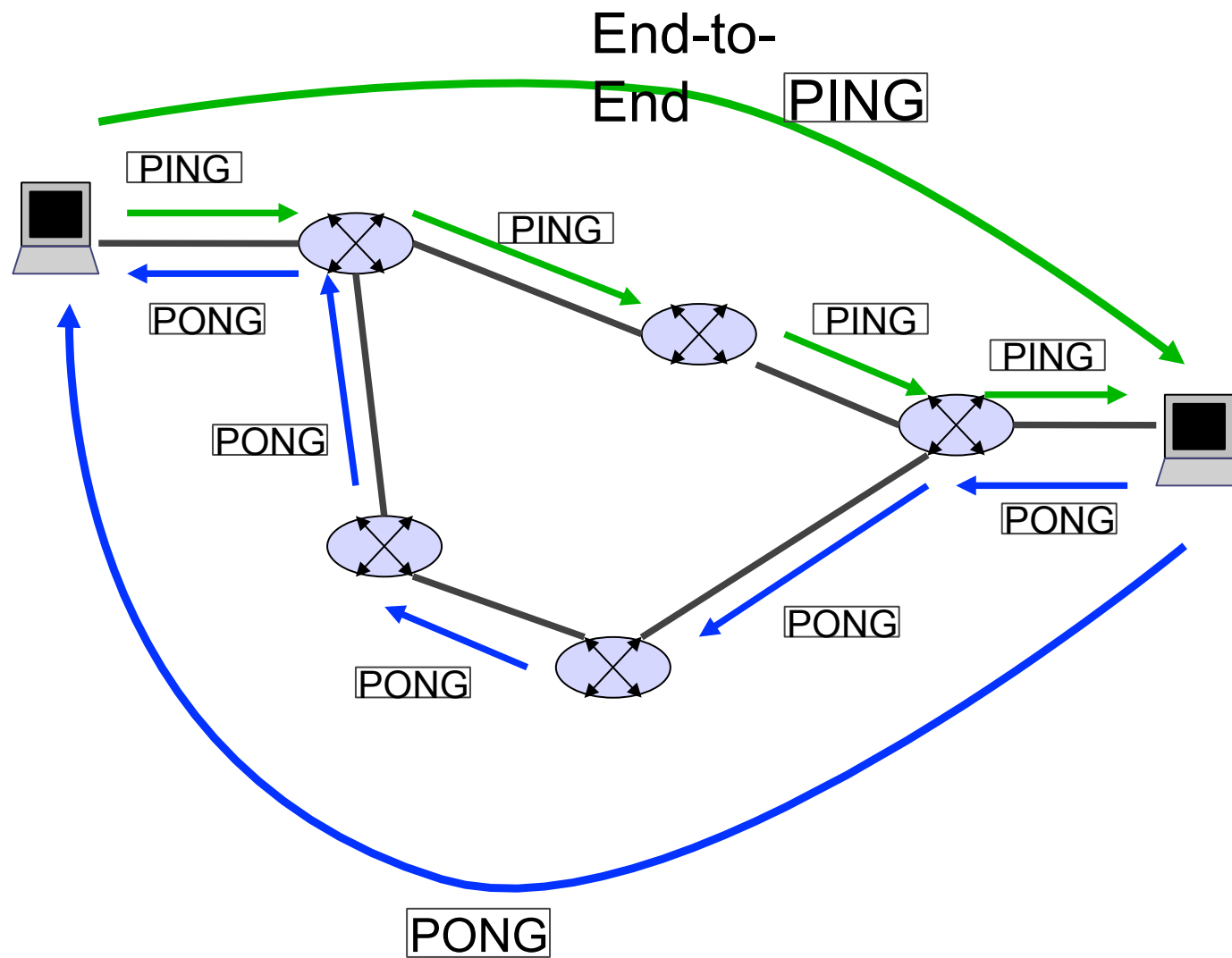
The path is not necessarily symmetric

Cost constraints, reconfiguration of the network, network failures can make the IP packets travel different routes going out and coming back.

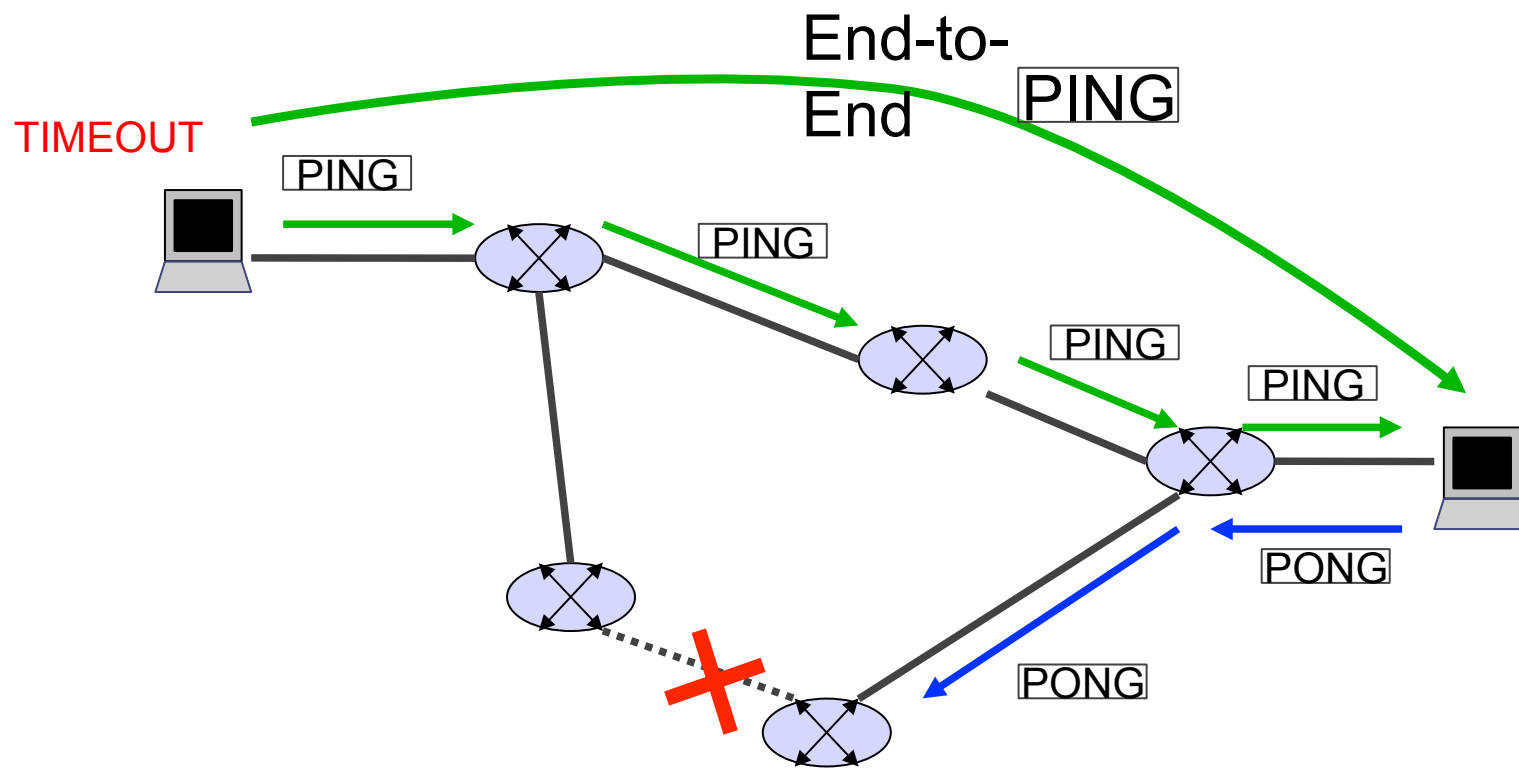
IP path



IP path



IP path



Network configuration

- Done in the file: **/etc/network/interfaces**
- Either static or dynamic (DHCP)

Static

```
# The primary network interface
# nsr.org
auto eth0
iface eth0 inet static
    address 128.223.157.19
    netmask 255.255.255.128
    network 128.223.157.0
    broadcast 128.223.157.127
    gateway 128.223.157.1
# IPv6 address. This resolves to nsr.org
iface eth0 inet6 static
    address 2001:0468:0d01:0103:0000:0000:80df:9d13
    netmask 64
    network 2001:468:D01:103::/64
    gateway 2001:468:D01:103::1
```


Network configuration

Dynamic

```
# The primary network interface
# nsrc.org
auto eth0
iface eth0 inet dhcp
```

- Addressing is received from a DHCP server.
- Classroom DHCP server is on noc.
- Can still be “static” if DHCP server knows MAC address

top

- Basic performance tool for Unix/Linux environments
- Periodically show a list of system performance statistics:
 - CPU use
 - RAM and SWAP memory usage
 - Load average (cpu utilization)
 - Information by process

top cont.

- **Information by process (most relevant columns shown):**
 - PID: Process ID
 - USER: user running (owner) of the process
 - %CPU: Percentage of CPU utilization by the process since the last sample
 - %MEM: Percentage of physical memory (RAM) used by the process
 - TIME: Total CPU time used by the process since it was started

top

```
top - 19:45:25 up 4:05, 1 user, load average: 1.18, 0.94, 0.88
Tasks: 122 total, 2 running, 116 sleeping, 0 stopped, 4 zombie
Cpu(s): 27.7%us, 13.3%sy, 0.2%ni, 58.7%id, 0.0%wa, 0.0%hi, 0.2%si, 0.0%st
Mem: 2052028k total, 612020k used, 1440008k free, 86784k buffers
Swap: 2588668k total, 0k used, 2588668k free, 172348k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
22005	www-data	20	0	39796	13m	5512	S	68	0.7	0:15.72	php
1033	mysql	20	0	315m	44m	6416	S	15	2.2	4:23.46	mysqld
21964	root	20	0	2836	1132	876	R	1	0.1	0:00.83	top
1012	dhcpd	20	0	5088	3128	1264	S	0	0.2	0:04.64	dhcpd
1282	nobody	20	0	5028	3000	2680	S	0	0.1	0:24.16	softflowd
20984	root	20	0	11820	3728	2932	S	0	0.2	0:00.21	sshd
22011	www-data	20	0	40404	13m	5716	S	0	0.7	0:00.58	php
1	root	20	0	3640	2044	1348	S	0	0.1	0:03.08	init
2	root	20	0	0	0	0	S	0	0.0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0	0.0	0:00.90	ksoftirqd/0
4	root	20	0	0	0	0	S	0	0.0	0:05.41	kworker/0:0
6	root	RT	0	0	0	0	S	0	0.0	0:00.00	migration/0

Load Average

Average number of active processes in the last 1, 5 and 15 minutes

- A simple yet useful measurement
- Depending on the machine the acceptable range considered to be normal can vary:
 - Multi-processor machines can handle more active processes per unit of time (than single processor machines)

netstat

Show us information about:

- Network connections
- Routing tables
- Interface (NIC) statistics
- Multicast group members

netstat

Some useful options

- n**: Show addresses, ports and userids in numeric form
- r**: Routing table
- s**: Statistics by protocol
- i**: Status of interfaces
- l**: Listening sockets
- tcp**, **--udp**: Specify the protocol
- A**: Address family [inet | inet6 | unix | etc.]
- p**: Show the name of each process for each port
- c**: Show output/results continuously

netstat

Examples (follow along):

```
# netstat -anr
```

```
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	MSS	Window	irrt	Iface
192.168.5.128	0.0.0.0	255.255.255.128	U	0	0	0	eth0
0.0.0.0	192.168.5.129	0.0.0.0	UG	0	0	0	eth0

```
# netstat -o -t
```

```
Active Internet connections (w/o servers)
```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	Timer
tcp	0	0	192.168.5.135:ssh	192.168.3.124:34155	ESTABLISHED	
keepalive (6754.95/0/0)						

```
# netstat -atv
```

```
Active Internet connections (servers and established)
```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	*:ssh	*:*	LISTEN
tcp	0	0	192.168.5.135:ssh	192.168.3.124:34155	ESTABLISHED
tcp6	0	0	:::ssh	:::*	LISTEN

netstat cont.

Examples:

```
# netstat -tcp -listening --program
```

Active Internet connections (only servers)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp	0	0	*:5001	*:*	LISTEN	13598/iperf
tcp	0	0	localhost:mysql	*:*	LISTEN	5586/mysqld
tcp	0	0	*:www	*:*	LISTEN	7246/apache2
tcp	0	0	t60-2.local:domain	*:*	LISTEN	5378/named
tcp	0	0	t60-2.local:domain	*:*	LISTEN	5378/named
tcp	0	0	t60-2.local:domain	*:*	LISTEN	5378/named
tcp	0	0	localhost:domain	*:*	LISTEN	5378/named
tcp	0	0	localhost:ipp	*:*	LISTEN	5522/cupsd
tcp	0	0	localhost:smtp	*:*	LISTEN	6772/exim4
tcp	0	0	localhost:953	*:*	LISTEN	5378/named
tcp	0	0	*:https	*:*	LISTEN	7246/apache2
tcp6	0	0	[::]:ftp	[::]:*	LISTEN	7185/proftpd
tcp6	0	0	[::]:domain	[::]:*	LISTEN	5378/named
tcp6	0	0	[::]:ssh	[::]:*	LISTEN	5427/sshd
tcp6	0	0	[::]:3000	[::]:*	LISTEN	17644/ntop
tcp6	0	0	ip6-localhost:953	[::]:*	LISTEN	5378/named
tcp6	0	0	[::]:3005	[::]:*	LISTEN	17644/ntop

lsOf (LiSt of Open Files)

`lsOf` is particularly useful because in Unix everything is a file: unix sockets, ip sockets, directories, etc.

Allows you to associate open files by:

- p**: PID (Process ID)
- i** : A network address (protocol:port)
- u**: A user

Isof

Example:

- First, using *netstat -ln -tcp* determine that port 6010 is open and waiting for a connection (LISTEN)

netstat -ln --tcp

Active Internet connections (only servers)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	127.0.0.1:6010	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:6011	0.0.0.0:*	LISTEN

Isof cont.

What network services am I running?

```
# lsof -i
```

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE	NODE	NAME
firefox	4429	hervey	50u	IPv4	1875852		TCP	192.168.179.139:56890->128.223.60.21:www (ESTABLISHED)
named	5378	bind	20u	IPv6	13264		TCP	*:domain (LISTEN)
named	5378	bind	21u	IPv4	13267		TCP	localhost:domain (LISTEN)
sshd	5427	root	3u	IPv6	13302		TCP	*:ssh (LISTEN)
cupsd	5522	root	3u	IPv4	1983466		TCP	localhost:ipp (LISTEN)
mysqld	5586	mysql	10u	IPv4	13548		TCP	localhost:mysql (LISTEN)
snmpd	6477	snmp	8u	IPv4	14633		UDP	localhost:snmp
exim4	6772	Debian-exim	3u	IPv4	14675		TCP	localhost:smtp (LISTEN)
ntpd	6859	ntp	16u	IPv4	14743		UDP	*:ntp
ntpd	6859	ntp	17u	IPv6	14744		UDP	*:ntp
ntpd	6859	ntp	18u	IPv6	14746		UDP	[fe80::250:56ff:fec0:8]:ntp
ntpd	6859	ntp	19u	IPv6	14747		UDP	ip6-localhost:ntp
proftpd	7185	proftpd	1u	IPv6	15718		TCP	*:ftp (LISTEN)
apache2	7246	www-data	3u	IPv4	15915		TCP	*:www (LISTEN)
apache2	7246	www-data	4u	IPv4	15917		TCP	*:https (LISTEN)
...								
iperf	13598	root	3u	IPv4	1996053		TCP	*:5001 (LISTEN)
apache2	27088	www-data	3u	IPv4	15915		TCP	*:www (LISTEN)
apache2	27088	www-data	4u	IPv4	15917		TCP	*:https (LISTEN)

tcpdump

- Show received packet headers by a given interface. Optionally filter using boolean expressions.
- Allows you to write information to a file for later analysis.
- Requires administrator (root) privileges to use since you must configure network interfaces (NICs) to be in “promiscuous” mode.

tcpdump

Some useful options:

- i** : Specify the interface (ex: -i eth0)
- l** : Make stdout line buffered (view as you capture)
- v, -vv, -vvv**: Display more information
- n** : Don't convert addresses to names (avoid DNS)
- nn** : Don't translate port numbers
- w** : Write raw packets to a file
- r** : Read packets from a file created by '-w'

tcpdump

Boolean expressions:

- Using the 'AND', 'OR', 'NOT' operators
- Expressions consist of one, or more, primitives, which consist of a qualifier and an ID (name or number):

Expression ::= [NOT] <primitive> [AND | OR | NOT <primitive> ...]

<primitive> ::= <qualifier> <name|number>

<qualifier> ::= <type> | <address> | <protocol>

<type> ::= host | net | port | port range

<address> ::= src | dst

<protocol> ::= ether | fddi | tr | wlan | ip | ip6 | arp | rarp | decnet | tcp | udp

tcpdump

Examples:

- Show all HTTP traffic that originates from 10.10.0.250

```
# tcpdump -lnXvvv port 80 and src host 10.10.0.250
```

- Show all traffic originating from 10.10.0.250 *except* SSH

```
# tcpdump -lnXvvv src host 10.10.0.250 and not port 22
```

Bibliography

- *Monitoring Virtual Memory with vmstat*
<http://www.linuxjournal.com/article/8178>
- *How to use TCPDump*
<http://www.erg.abdn.ac.uk/users/alastair/tcpdump.html>
- *linux command tcpdump example*
<http://smartproteam.com/linux-tutorials/linux-command-tcpdump/>
- *simple usage of tcpdump*
<http://linux.byexamples.com/archives/283/simple-usage-of-tcpdump/>
- *TCPDUMP Command man page with examples*
<http://www.cyberciti.biz/howto/question/man/tcpdump-man-page-with-examples.php>
- *TCPDump Tutorial*
<http://inst.eecs.berkeley.edu/~ee122/fa06/projects/tcpdump-6up.pdf>