Lesson Module Status

- Slides –
- Properties -
- Flashcards -
- 1st minute quiz –
- Web Calendar summary –
- Web book pages –
- Commands –
- Howtos –
- Lab tested –
- Depot (Opus) –
- Lab 02 template -
- Youtube Videos uploaded –

- VM (Classroom PC) –
- VMs (VLab) -

- Headset charged –

[ ] Has the phone bridge been added?

[ ] Is phone being used for voice input?

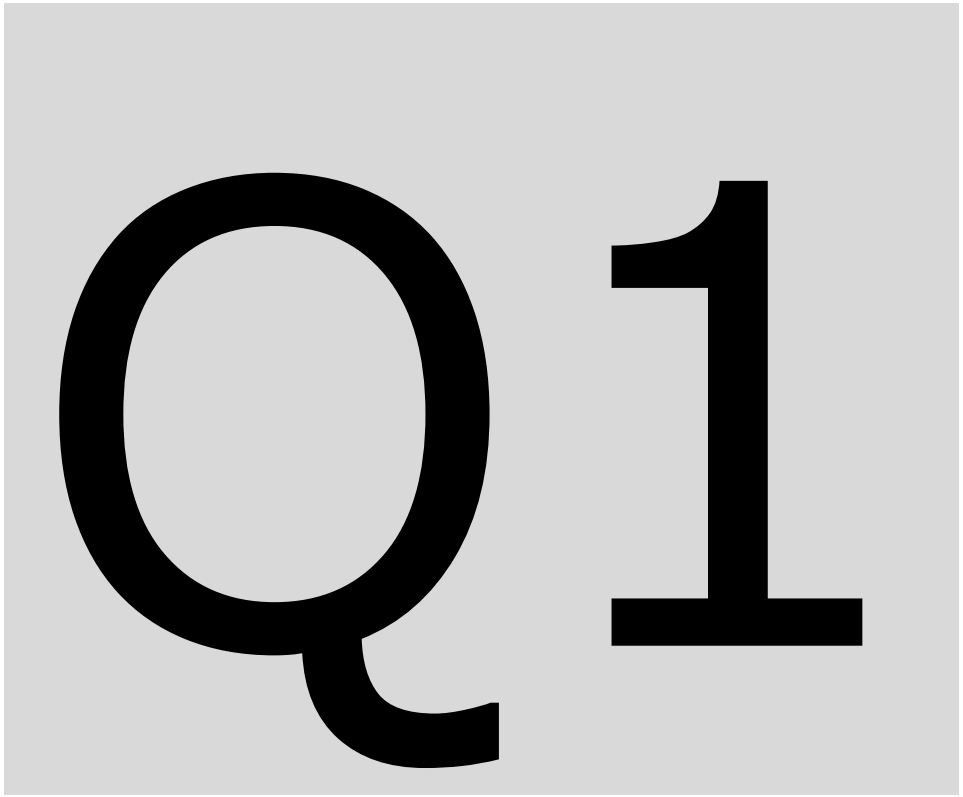
[ ] Is recording on?

[ ] Share slides, putty (rsimms, simben192), Chrome, vlab192.rdp, VMware Workstation, wireshark

[ ] Disable spelling on PowerPoint

[ ] Repeat all ?'s for remote students

[ ] Remote student proxy


Q1

Course history and credits

Jim Griffin

- Jim created the original version of this course
- Jim's site: http://cabrillo.edu/~jgriffin/

Rick Graziani

- Thanks to Rick Graziani for the use of some of his great network slides
- Rick's site: http://cabrillo.edu/~rgraziani/

First Minute Quiz

Please answer these questions **in the order** shown:

email answers to: `risimms@cabrillo.edu`
within the first few minutes of class

ARP and the Internet Layer

Related Course Objectives	Agenda
• Use basic network terminology to describe the five layers of the TCP/IP Reference Model, and describe at least one major function of each layer. • Use the arpwatch daemon to collect IP/hardware addresses, and manually add an address to the ARP table. • Install the device drivers and configure the network interface card (NIC) of a Linux system so that it may join a network. • Configure appropriate IP addresses, network and subnet masks, and broadcast addresses based on the size and number of network segments required. • Use a network sniffer to analyze network traffic between two hosts. • Identify, isolate, and correct malfunctions in a computer network.	• Quiz • Questions on previous material • Housekeeping • Cabling VMs • Joining a network (temp) • Joining a network (perm) • Aliases • ARP • arpwatch • Viewing packets • Internet Layer • IPv4 Addressing • NAT/PAT and IPv6 • Traversing VMs using SSH • Troubleshooting • Lab • Wrap

Questions and Review

Questions?

scp

scp command

Tip: The **scp** command can be very useful for adding file content or command output to your lab submittals.

```
cat /etc/sysconfig/network-scripts/ifcfg-eth* > notes
cat /etc/sysconfig/network >> notes
cat /etc/resolv.conf >> notes
ifconfig >> notes
route -n >> notes
scp notes xxxyyy192@opus.cabrillo.edu:
```

This example copies network configuration files and command output to the user's home directory on Opus.

Class Exercise

Transfer data between systems

Try it!

ping –c3 172.30.4.1

ping –c3 172.30.4.1 > labnotes

ifconfig

ifconfig >> labnotes

cat /etc/resolv.conf

cat /etc/resolv.conf >> labnotes

cat labnotes

scp labnotes xxxyyy192@opus.cabrillo.edu:

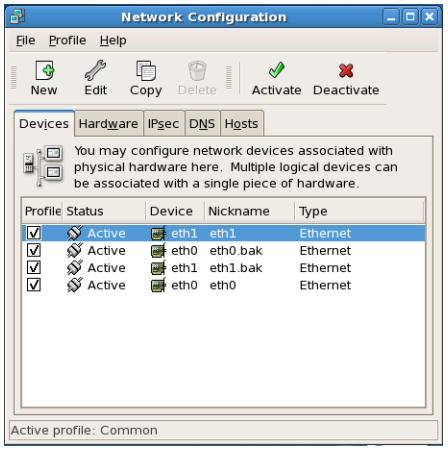
Network Configuration

(Joining a network)

GUI vs Command Line

*The **GUI** (Graphical User Interface) tools are easy to use but they are different with each distribution.*

CentOS 5.4

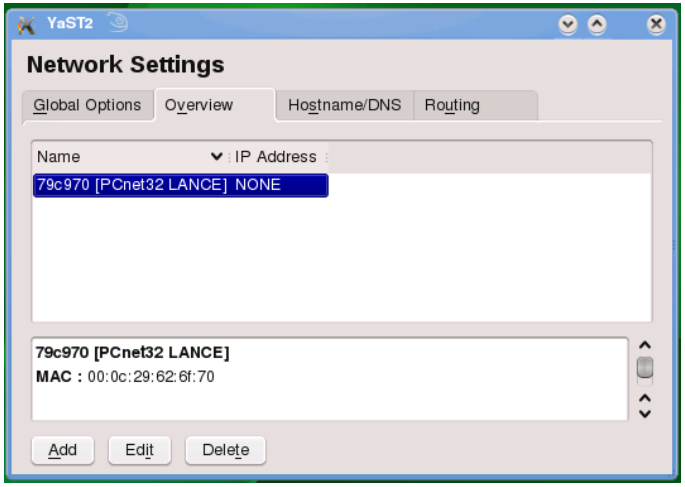


System
> Administration
> Network

Ubuntu 9.10

System
> Preferences
> Network Connections

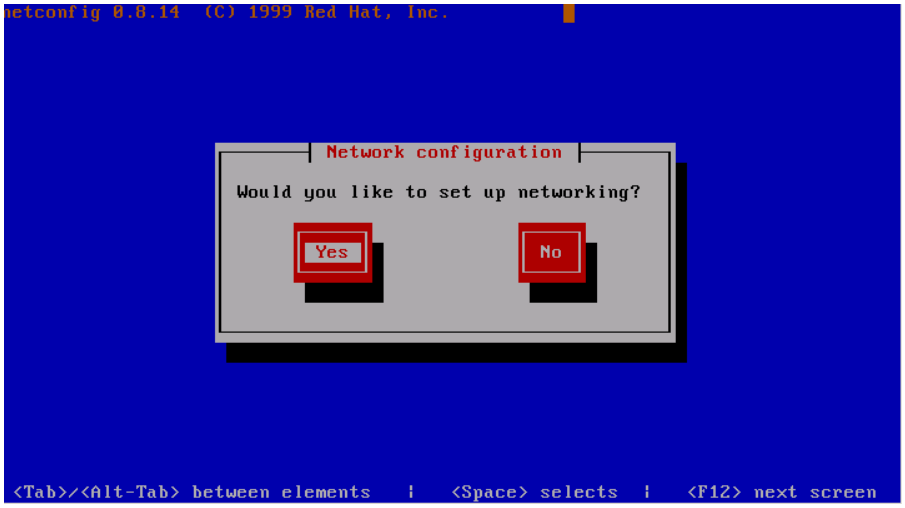
OpenSUSE 11.2



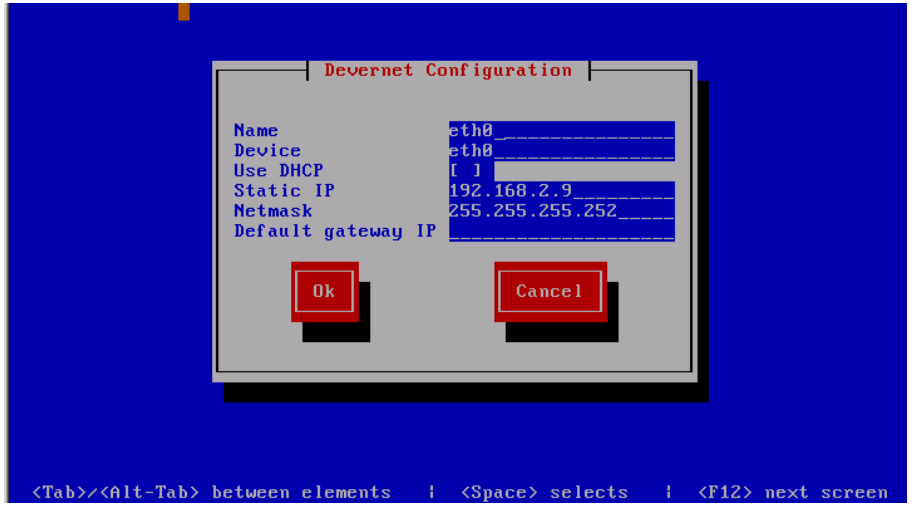
Application Launcher
> Computer
> YaST
> YaST Control Center
> Network Devices
> Network Settings

The UNIX/Linux customers first question was always: That a very pretty interface but I need to know exactly what commands you are calling underneath!

TUI (Red Hat Family)



*The **netconfig** command on Red Hat 9 provides a TUI interface to set the basic network settings.*



*The **system-config-network** command replaces **netconfig** on CentOS 5.4.*

Temporary vs Permanent

Commands and Configuration Files

*The **command line** tools are the same common across distributions plus they can be automated with scripts. Some of the **configuration files** differ by distribution family.*

Temporary (Commands)

- ifconfig
- route

Permanent (Configuration files)

- /etc/hosts
- /etc/resolv.conf
- Red Hat family:
  - /etc/sysconfig/network
  - /etc/sysconfig/network-scripts/ifcfg-eth*
  - **service network restart**
- Ubuntu family:
  - /etc/hostname
  - /etc/network/interfaces
  - **/etc/init.d/networking restart**
- OpenSUSE family
  - /etc/HOSTNAME
  - /etc/sysconfig/network/ifcfg-eth*
  - **rcnetwork restart**

Yes, there is no "e"!

*The commands are **temporary** and stay in effect only till the system (or the network service) is restarted.*

*The scripts are **permanent** but don't take effect until the system (or the network service) is restarted*

Joining a network (temporary)

Joining a Network
(Temporary - all families)

Settings kept in memory and are lost when system is restarted

Temporary	Dynamic	Static
IP and subnet mask	**dhclient eth***n*	**ifconfig eth***n xxx.xxx.xxx.xxx/pp*
Default gateway		**route add default gw** *xxx.xxx.xxx.xxx*
DNS		add nameservers to **/etc/resolv.conf** file

For older distributions use:

ifconfig eth*n xxx.xxx.xxx.xxx* ***netmask*** *xxx.xxx.xxx.xxx*

(to set static IP address and mask)

Configuring IP addresses and subnet mask (Temporary)

Set

• To set ip address and subnet mask:

ifconfig ethx xxx.xxx.xxx.xxx **netmask** xxx.xxx.xxx.xxx

or **ifconfig eth**x xxx.xxx.xxx.xxx/pp

Verify

• To show all interfaces (and to show your IP address):

ifconfig

• To show a single interface:

ifconfig ethx

Example

```
[root@elrond ~]# ifconfig eth1 192.168.2.107 netmask 255.255.255.0
[root@elrond ~]# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 00:0C:29:82:68:84
          inet addr:192.168.2.107  Bcast192.168.2.255 : Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe82:6884/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:33 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:8090 (7.9 KiB)
          Interrupt:185 Base address:0x1480
```

Configuring IP addresses and subnet mask
(Temporary)

Example: Set an IP address and subnet mask on Elrond in VLab Pod 6:

ifconfig eth0 172.30.4.250/24

```
[root@elrond ~]# ifconfig eth0 172.30.4.250/24
[root@elrond ~]#
[root@elrond ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:2A:57:17
          inet addr:172.30.4.250  Bcast:172.30.4.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe2a:5717/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1598 errors:0 dropped:0 overruns:0 frame:0
          TX packets:691 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:156414 (152.7 KiB)  TX bytes:86445 (84.4 KiB)

[root@elrond ~]# _
```

Remember to use the Static IP chart at:
http://simms-teach.com/docs/static-ip-addrs.pdf

when assigning IP addresses on the CIS classroom or lab networks.

Configuring the default gateway
(Temporary)

Set

- To set the default gateway
  route add default gw xxx.xxx.xxx.xxx

- To delete the default gateway
  route del default gw xxx.xxx.xxx.xxx

Verify

- To show the routing table (including gateway)
  route –n

Example

```
[root@elrond ~]# route add default gw 172.30.4.1
[root@elrond ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.30.4.0      0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.2.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
0.0.0.0         172.30.4.1      0.0.0.0         UG    0      0        0 eth0
[root@elrond ~]#
```

Routing table

Matches all addresses

G = Gateway

Configuring the default gateway
(Temporary)

Example: Set the default gateway to the Lab router

route add default gw 172.30.4.1

```
[root@elrond ~]# route add default gw 172.30.4.1
[root@elrond ~]#
[root@elrond ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.30.4.0      0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.2.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
169.254.0.0     0.0.0.0         255.255.0.0     U     1002   0        0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U     1003   0        0 eth1
0.0.0.0         172.30.4.1      0.0.0.0         UG    0      0        0 eth0
[root@elrond ~]# _
```

Configuring the DNS
(Permanent)

Set
- Add a line to **/etc/resolv.conf**
  nameserver xxx.xxx.xxx.xxx

Verify
- Show the file
  cat /etc/resolv.conf

Example

```
[root@elrond ~]# echo nameserver 10.240.1.2 > /etc/resolv.conf
[root@elrond ~]# cat /etc/resolv.conf
nameserver 10.240.1.2
[root@elrond ~]#
```

Note: Changes to /etc/resolv.conf take effect immediately

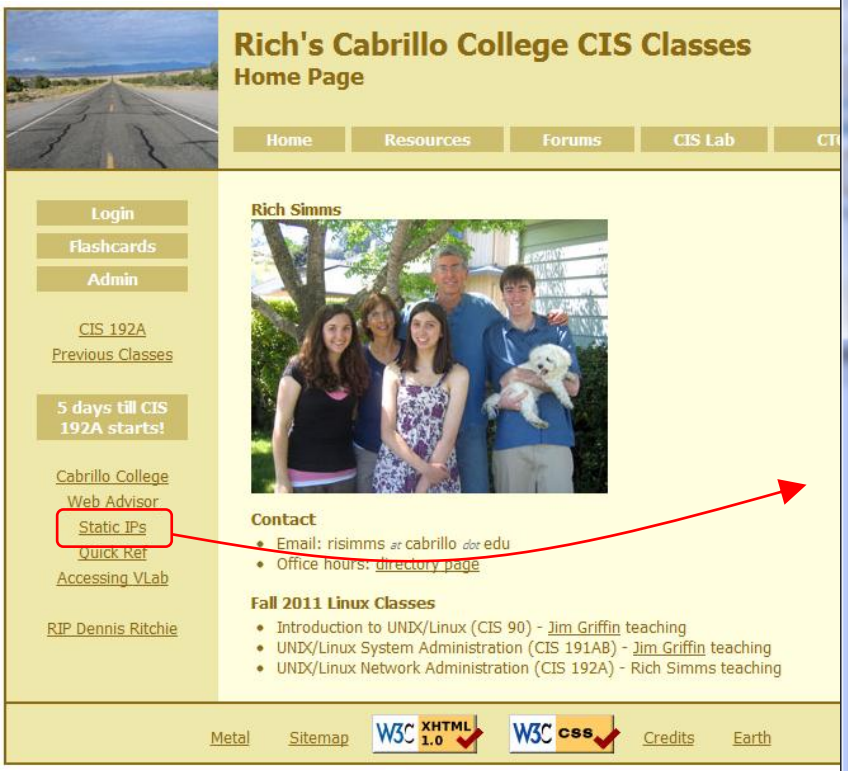
Configuring the DNS
(Permanent)

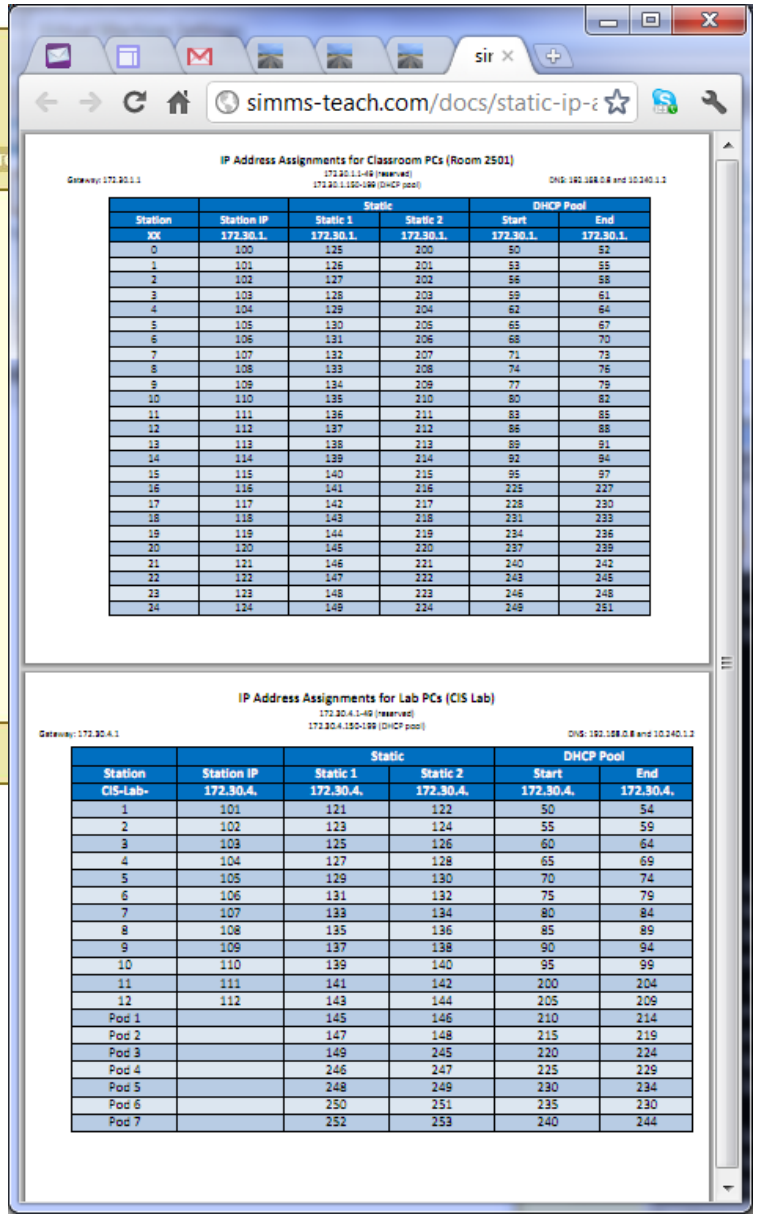
Example: Set the DNS server to the CIS Lab DNS server:

Add **nameserver 192.168.0.8** line to **/etc/resolv.conf**

```
[root@elrond ~]# echo nameserver 192.168.0.8 > /etc/resolv.conf
[root@elrond ~]#
[root@elrond ~]# cat /etc/resolv.conf
nameserver 192.168.0.8
[root@elrond ~]# _
```

Note: Changes to /etc/resolv.conf take effect immediately





IP Address Assignments for Classroom PCs (Room 2501)

Station	Station IP	Static		DHCP Pool	
		Static 1	Static 2	Start	End
XX	172.30.1.	172.30.1.	172.30.1.	172.30.1.	172.30.1.
0	100	125	200	50	52
1	101	126	201	53	55
2	102	127	202	56	58
3	103	128	203	59	61
4	104	129	204	62	64
5	105	130	205	65	67
6	106	131	206	68	70
7	107	132	207	71	73
8	108	133	208	74	76
9	109	134	209	77	79
10	110	135	210	80	82
11	111	136	211	83	85
12	112	137	212	86	88
13	113	138	213	89	91
14	114	139	214	92	94
15	115	140	215	95	97
16	116	141	216	225	227
17	117	142	217	228	230
18	118	143	218	231	233
19	119	144	219	234	236
20	120	145	220	237	239
21	121	146	221	240	242
22	122	147	222	243	245
23	123	148	223	246	248
24	124	149	224	248	251

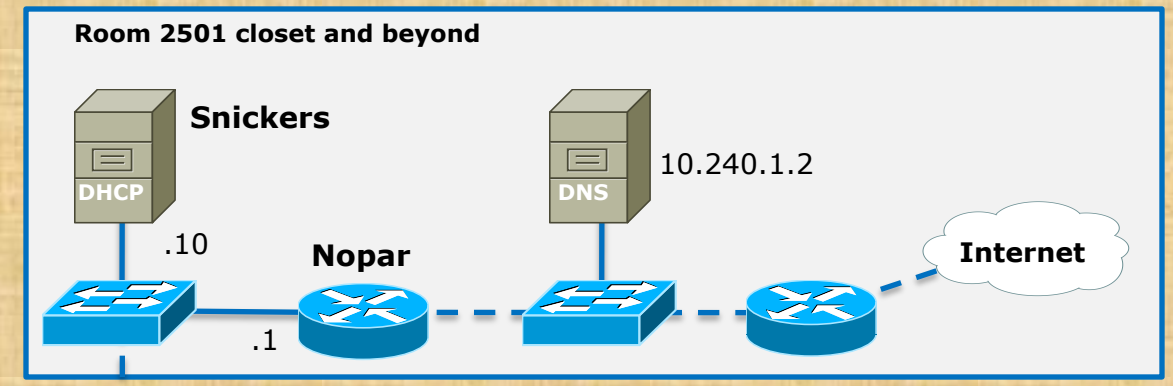
IP Address Assignments for Lab PCs (CIS Lab)

Station	Station IP	Static		DHCP Pool	
		Static 1	Static 2	Start	End
CIS-Lab-	172.30.4.	172.30.4.	172.30.4.	172.30.4.	172.30.4.
1	101	121	122	50	54
2	102	123	124	55	59
3	103	125	126	60	64
4	104	127	128	65	69
5	105	129	130	70	74
6	106	131	132	75	79
7	107	133	134	80	84
8	108	135	136	85	89
9	109	137	138	90	94
10	110	139	140	95	99
11	111	141	142	200	204
12	112	143	144	205	209
Pod 1		145	146	210	214
Pod 2		147	148	215	219
Pod 3		149	245	220	224
Pod 4		246	247	225	229
Pod 5		248	249	230	234
Pod 6		250	251	235	230
Pod 7		252	253	240	244

To avoid **TROUBLE**, use the Static IPs link on the web site to select IP addresses.

Only use static IPs assigned to the station you are using in the classroom or the lab!

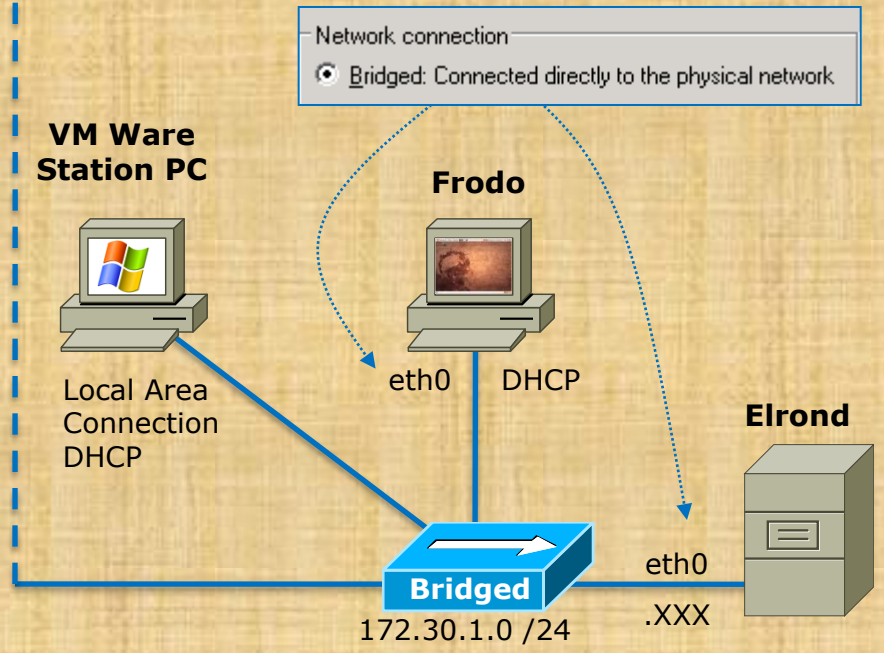
Class Exercise – Join Frodo and Elrond to classroom network



47 days till term ends!

Cabrillo College
Web Advisor
Static IPs
Quick Ref
Accessing VLab

No DUPS Please!



Frodo (dhcp)

- ifconfig eth0
- ping 172.30.1.1
- ping google.com

Elrond (static)

- ifconfig eth0 172.30.1.xxx/24
- route add default gw 172.30.1.1
- add nameserver 10.240.1.2 to /etc/resolv.conf
- ifconfig eth0
- ping 172.30.1.1
- ping google.com

Joining a network (permanent)

Joining a Network
(Permanent - Red Hat Family)

Settings kept in configuration files and used during the startup process

Area	Dynamic (permanent)
IP and subnet mask	**/etc/sysconfig/network-settings/ifcfg-eth*n*** **DEVICE="eth*n*"** **NM_CONTROLLED="no"** **ONBOOT="yes"** **BOOTPROTO=”dhcp”**
Default gateway	
DNS	

Use **service network restart** for changes to take effect

Joining a Network
(Permanent - Red Hat Family)

Settings kept in configuration files and used during the startup process

Area	Static (permanent)
IP and subnet mask	**/etc/sysconfig/network-scripts/ifcfg-eth*n*** **DEVICE="eth*n*"** **NM_CONTROLLED="no"** **ONBOOT="yes"** **BOOTPROTO="static"** **IPADDR=***xxx.xxx.xxx.xxx* **NETMASK=***xxx.xxx.xxx.xxx*
Default gateway	**/etc/sysconfig/network** **NETWORKING=yes** **HOSTNAME=***name.domain* **GATEWAY=***xxx.xxx.xxx.xxx*
DNS	**/etc/resolv.conf** **nameserver** *xxx.xxx.xxx.xxx* **nameserver** *xxx.xxx.xxx.xxx*

Use **service network restart** for changes to take effect

Permanent network configuration

Example: Permanently configure both interfaces on Elrond for Lab 02 (VLab Pod 6)

```
[root@elrond ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
NM_CONTROLLED="no"
ONBOOT="yes"
BOOTPROTO="static"
IPADDR=172.30.4.250
NETMASK=255.255.255.0
```

```
[root@elrond ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE="eth1"
NM_CONTROLLED="no"
ONBOOT="yes"
BOOTPROTO="static"
IPADDR=192.168.2.1
NETMASK=255.255.255.0
```

```
[root@elrond ~]# cat /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=elrond.localdomain
GATEWAY=172.30.4.1
```

```
[root@elrond ~]# cat /etc/resolv.conf
nameserver 192.168.0.8
```

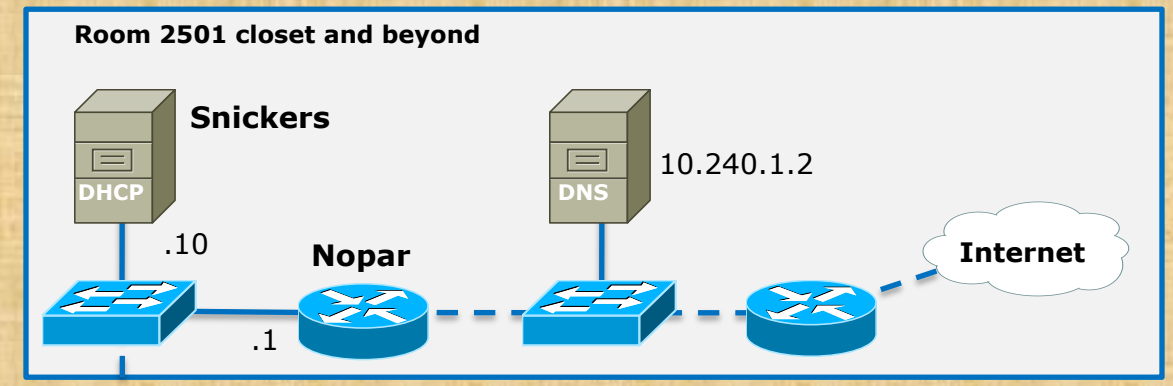
Permanent network configuration

Example: Restart the network services so any changes in the configuration take effect

service network restart

```
[root@elrond ~]# service network restart
Shutting down interface eth0:                              [  OK  ]
Shutting down interface eth1:                              [  OK  ]
Shutting down loopback interface:                          [  OK  ]
Bringing up loopback interface:                            [  OK  ]
Bringing up interface eth0:                                [  OK  ]
Bringing up interface eth1:                                [  OK  ]
[root@elrond ~]# _
```

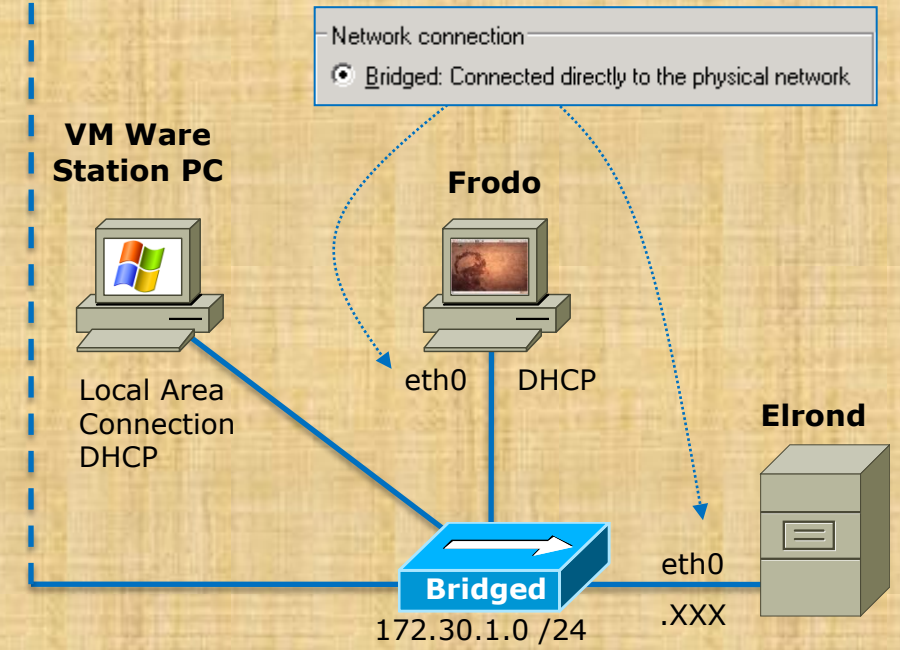
Class Exercise – Permanent Elrond network configuration



47 days till term ends!

Cabrillo College
Web Advisor
Static IPs
Quick Ref
Accessing VLab

No DUPS Please!



Elrond (static)

/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE="eth0"
NM_CONTROLLED="no"
ONBOOT="yes"
BOOTPROTO="static"
IPADDR=172.30.1.xxx
NETMASK=255.255.255.0
```

/etc/sysconfig/network

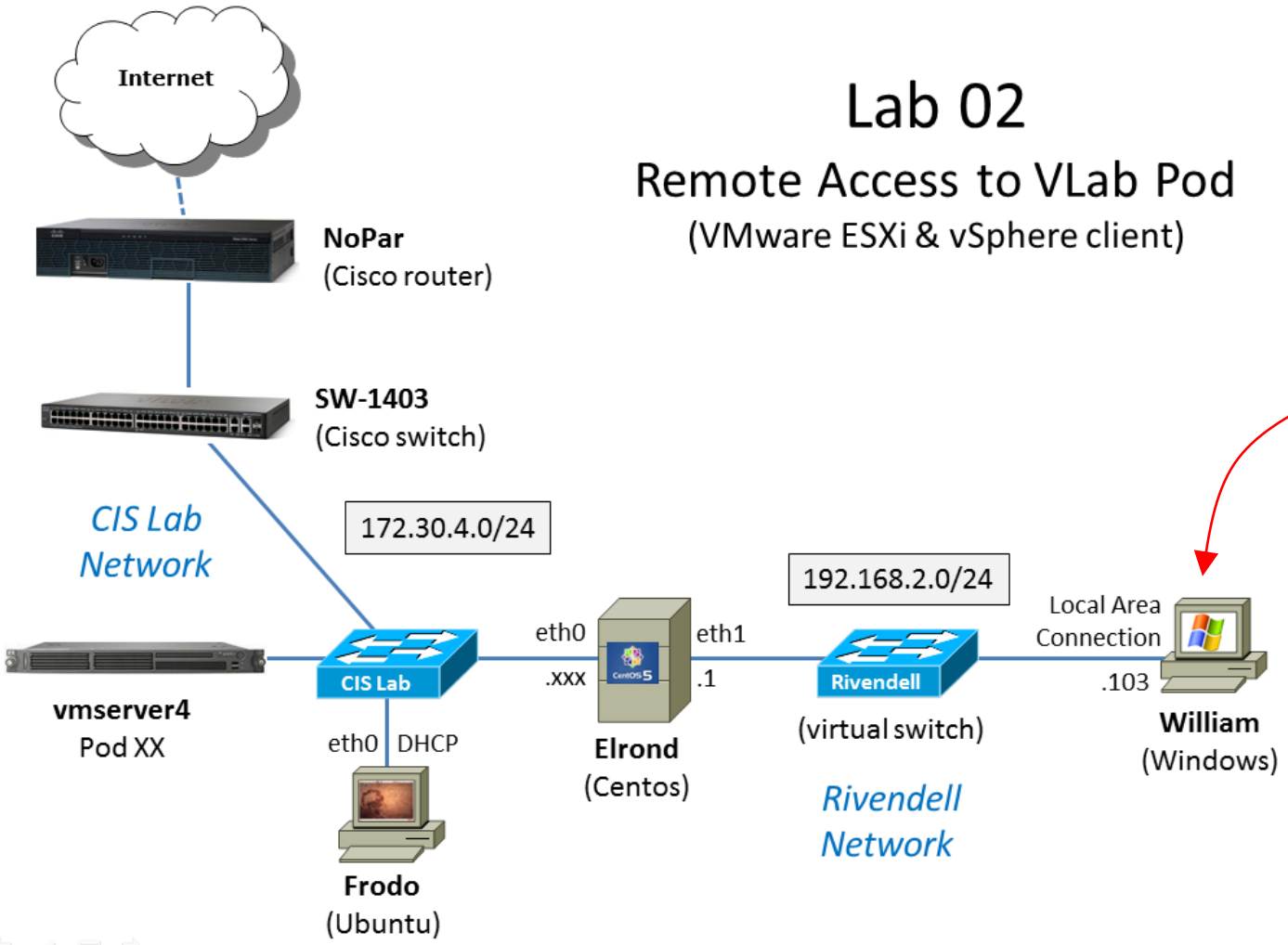
```
NETWORKING=yes
HOSTNAME=elrond.localdomain
GATEWAY=172.30.4.1
```

/etc/resolv.conf

```
nameserver 10.240.1.2
```

service network restart

Windows XP network configuration



Lab 02

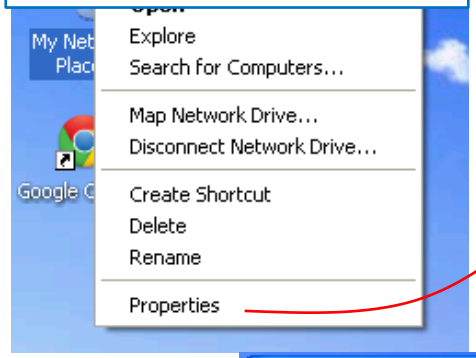
Remote Access to VLab Pod

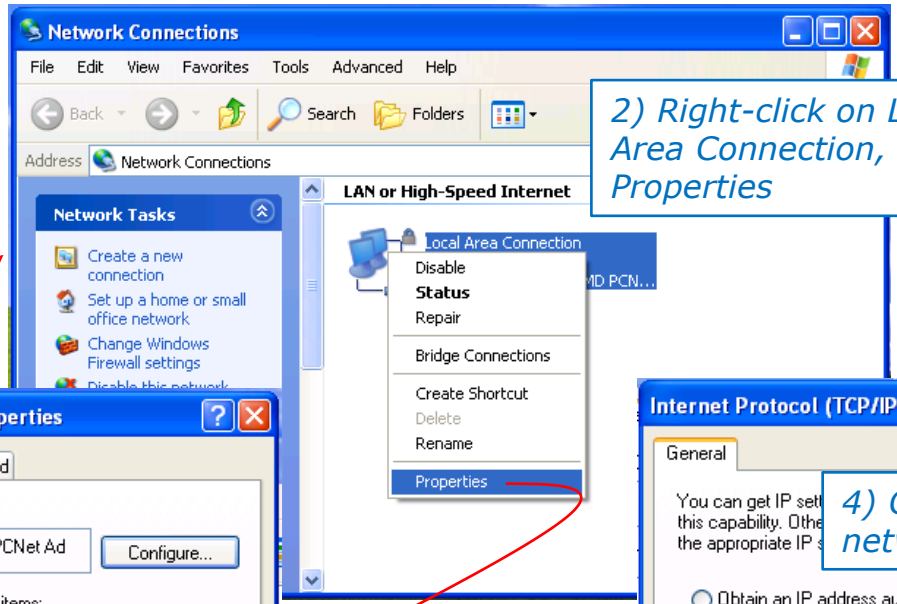
(VMware ESXi & vSphere client)

We will need to configure network settings on Windows in future labs

Network settings on Windows XP

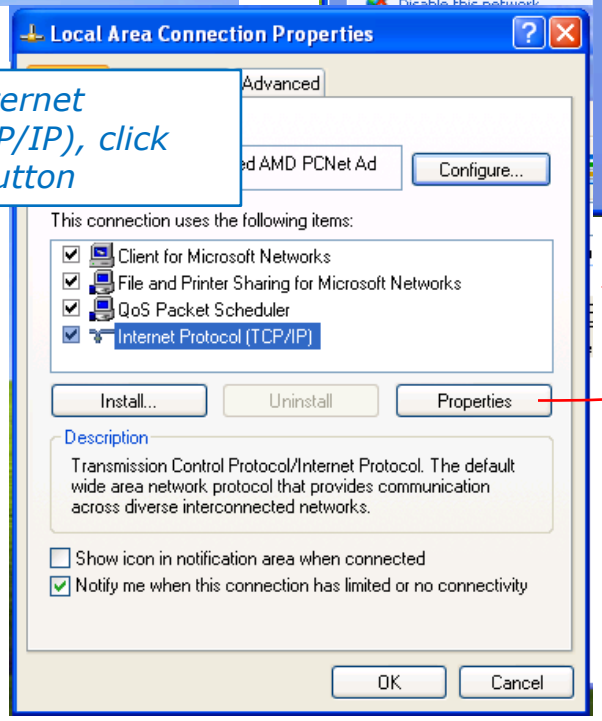
1) Right-click on My Network Places, select Properties

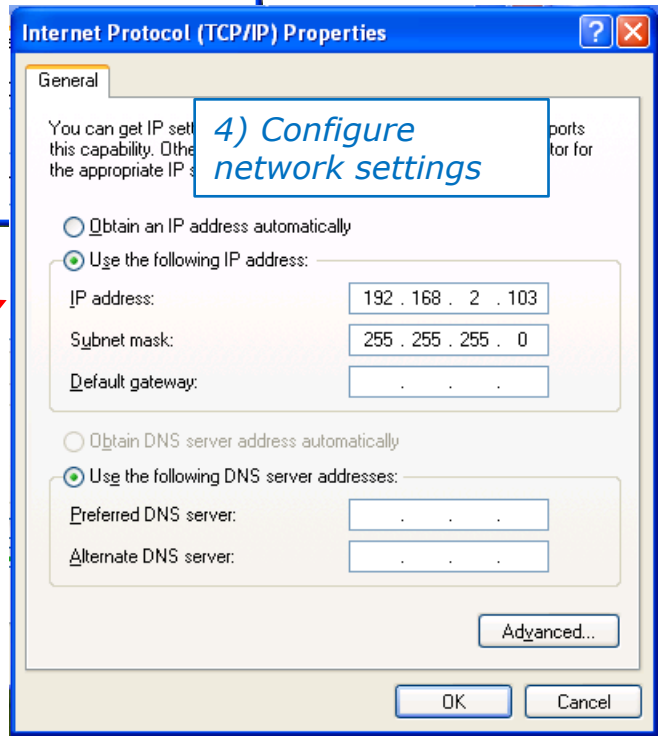




2) Right-click on Local Area Connection, select Properties

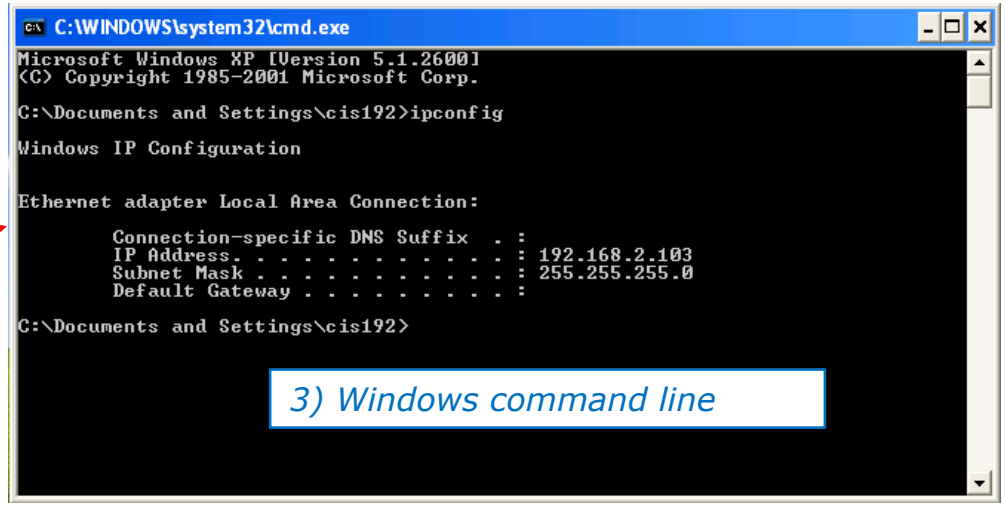
3) Select Internet Protocol (TCP/IP), click Properties button

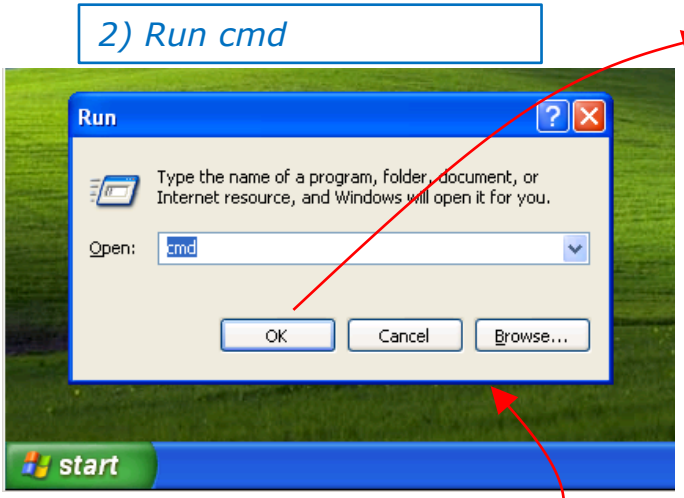


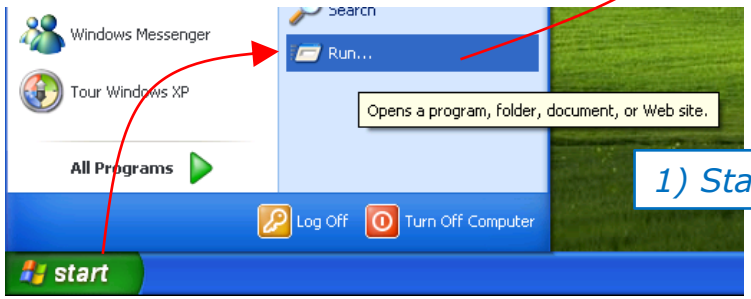


4) Configure network settings

Getting a command line on Windows XP







Class Exercise

Windows XP network settings and command line

Try it!

- *Cable William to Rivendell*
- *IP = 192.168.2.111*
- *Subnet mask = 255.255.255.0*
- *Run ipconfig to verify*

Housekeeping

- Lab 1 is due by midnight tonight (Opus time)
- Please use the sign in sheets in the lab which are used for tracking the TBA portion of the course
- Quick check on /home/rsimms/turnin on Opus

- Roll Call
- Adds – If you didn't use add code by 10/31, and want to stay in the course you will need to do a manual add with a Late Add Slip"

Temporarily turn off recording

CIS 192 – Lesson 2

Instructor: **Rich Simms**
Dial-in: **888-450-4821**
Passcode: **761867**

James Lars Daniel Elizabeth

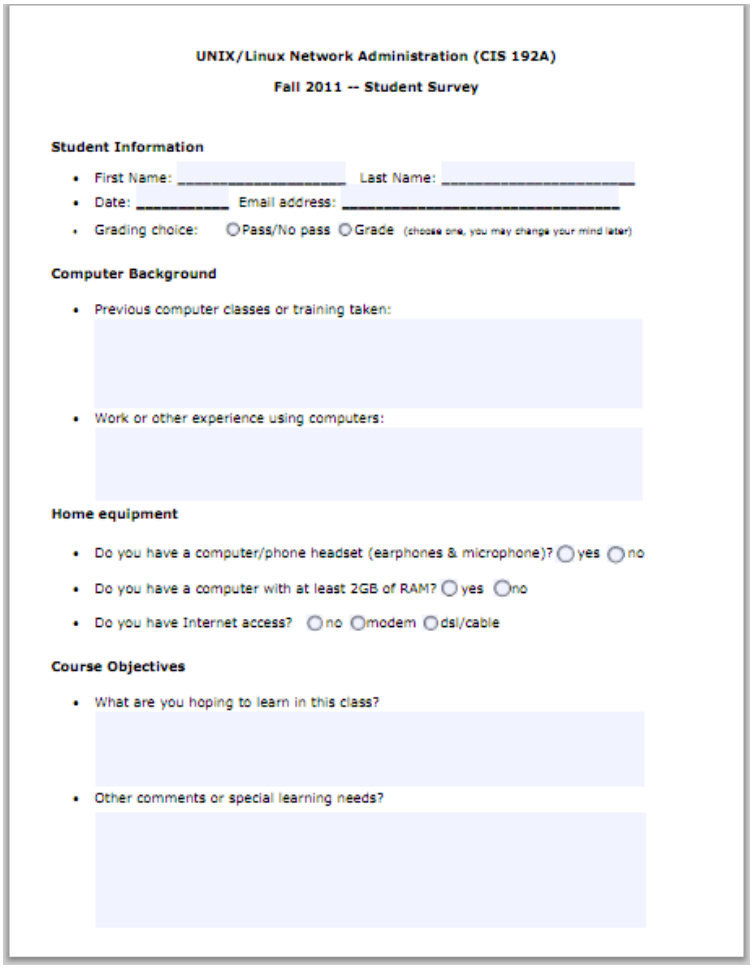
Carlos V Branden Chad Donovan Leopoldo Jacob G Jeff Timothy Jacob S Laura

Gabriel V Jason Thomas Josh Carlos R Geoffrey Ellison Mark David Leandro

Email me (risimms@cabrillo.edu) a relatively current photo of your face for 3 points extra credit

Student Survey

http://simms-teach.com/docs/cis192/cis192survey.pdf



UNIX/Linux Network Administration (CIS 192A)

Fall 2011 -- Student Survey

Student Information

- First Name: _____ Last Name: _____
- Date: _____ Email address: _____
- Grading choice: ◯Pass/No pass ◯Grade (choose one, you may change your mind later)

Computer Background

- Previous computer classes or training taken:
- Work or other experience using computers:

Home equipment

- Do you have a computer/phone headset (earphones & microphone)? ◯yes ◯no
- Do you have a computer with at least 2GB of RAM? ◯yes ◯no
- Do you have Internet access? ◯no ◯modem ◯dsl/cable

Course Objectives

- What are you hoping to learn in this class?
- Other comments or special learning needs?

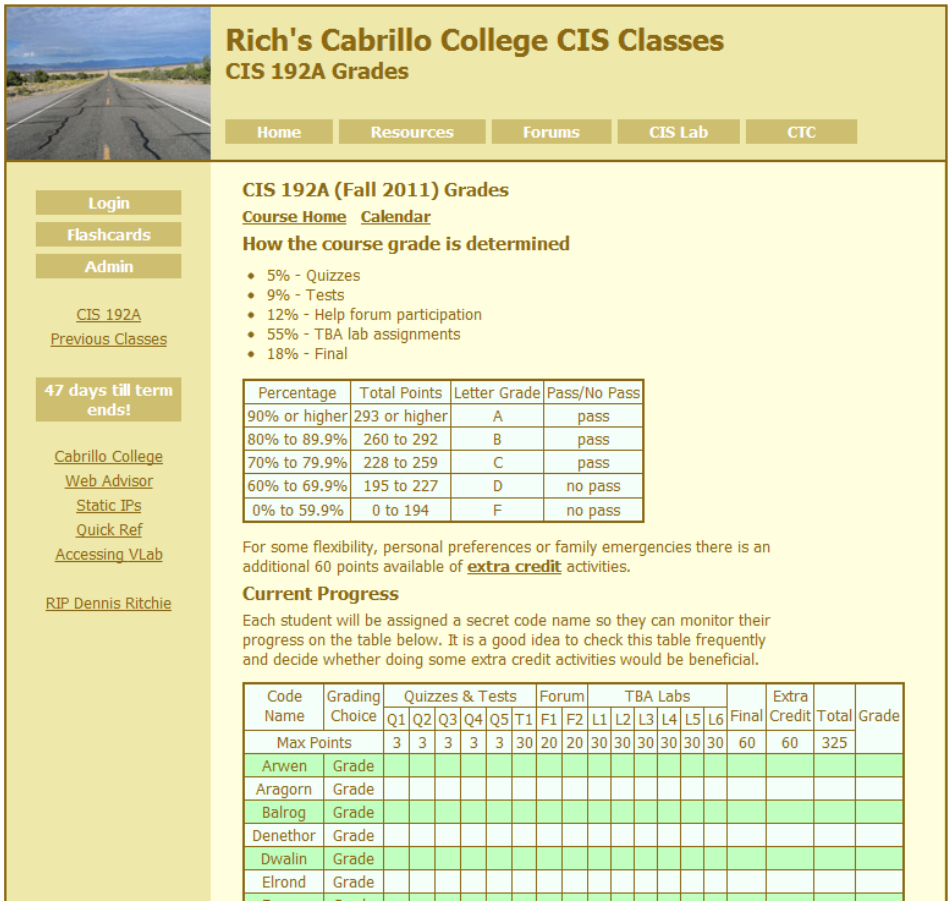
Surveys due midnight tonight!

Email them to me at: risimms@cabrillo.edu

CIS 192 – Code Names

Lord of the Rings Characters

http://simms-teach.com/cis192Agrades.php



Rich's Cabrillo College CIS Classes

CIS 192A Grades

Home | Resources | Forums | CIS Lab | CTC

Login | Flashcards | Admin

CIS 192A
Previous Classes

47 days till term ends!

Cabrillo College
Web Advisor
Static IPs
Quick Ref
Accessing VLab

RIP Dennis Ritchie

CIS 192A (Fall 2011) Grades

Course Home Calendar

How the course grade is determined

- 5% - Quizzes
- 9% - Tests
- 12% - Help forum participation
- 55% - TBA lab assignments
- 18% - Final

Percentage	Total Points	Letter Grade	Pass/No Pass
90% or higher	293 or higher	A	pass
80% to 89.9%	260 to 292	B	pass
70% to 79.9%	228 to 259	C	pass
60% to 69.9%	195 to 227	D	no pass
0% to 59.9%	0 to 194	F	no pass

For some flexibility, personal preferences or family emergencies there is an additional 60 points available of **extra credit** activities.

Current Progress

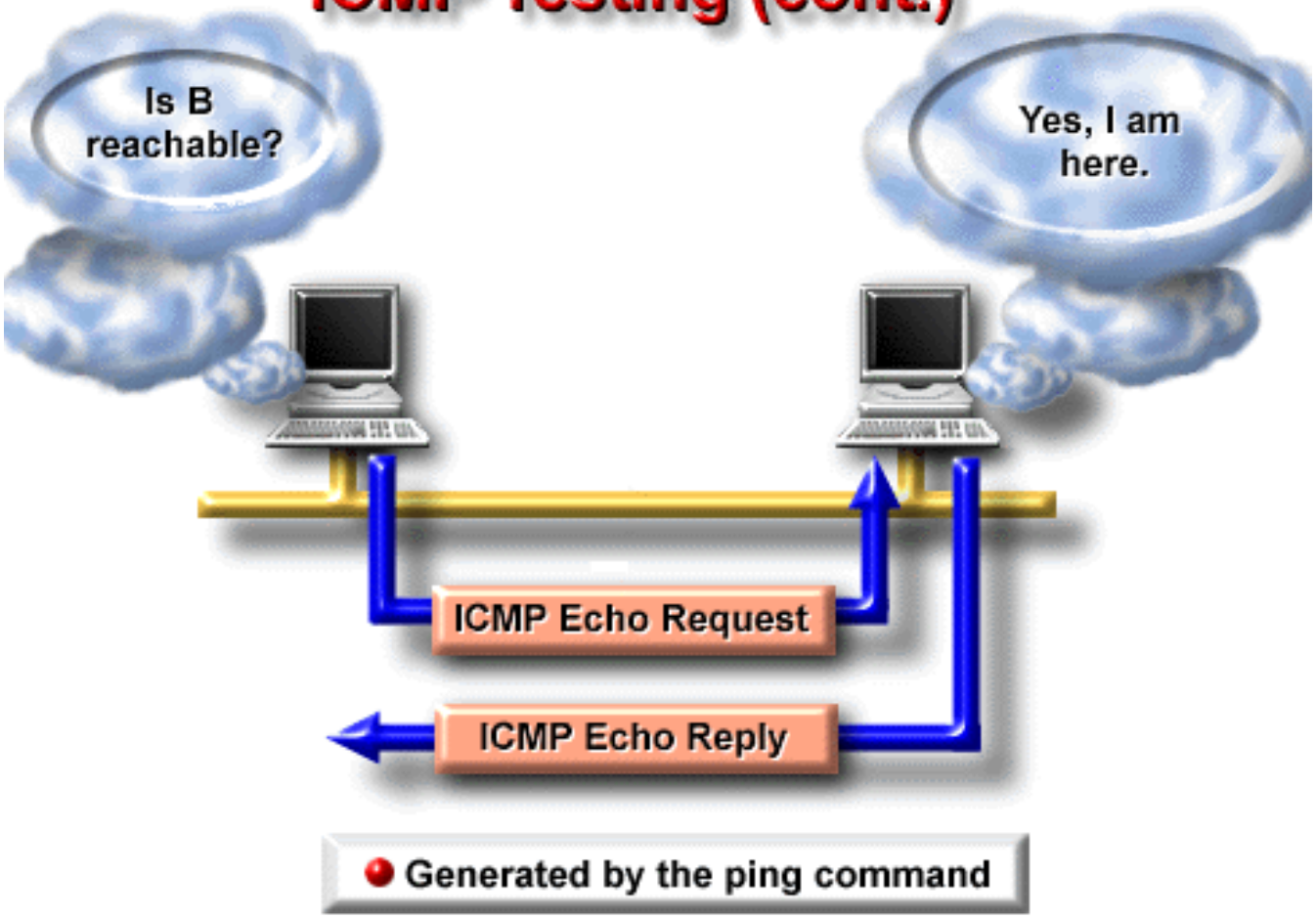
Each student will be assigned a secret code name so they can monitor their progress on the table below. It is a good idea to check this table frequently and decide whether doing some extra credit activities would be beneficial.

Code Name	Grading Choice	Quizzes & Tests						Forum		TBA Labs					Final	Extra Credit	Total	Grade
		Q1	Q2	Q3	Q4	Q5	T1	F1	F2	L1	L2	L3	L4	L5	L6				
Max Points		3	3	3	3	3	30	20	20	30	30	30	30	30	30	60	60	325	
Arwen	Grade																		
Aragorn	Grade																		
Balrog	Grade																		
Denethor	Grade																		
Dwalin	Grade																		
Elrond	Grade																		

Send me an email to get your code name

Trouble shooting

ICMP Testing (cont.)



Troubleshoot Network Connection

Follow these steps if your connection is not working:

1. Check cabling, IP and subnet mask settings by pinging another node on the same local network (which could be the router) using an IP address.

2. Check default gateway by pinging a node outside the local network using an IP address.

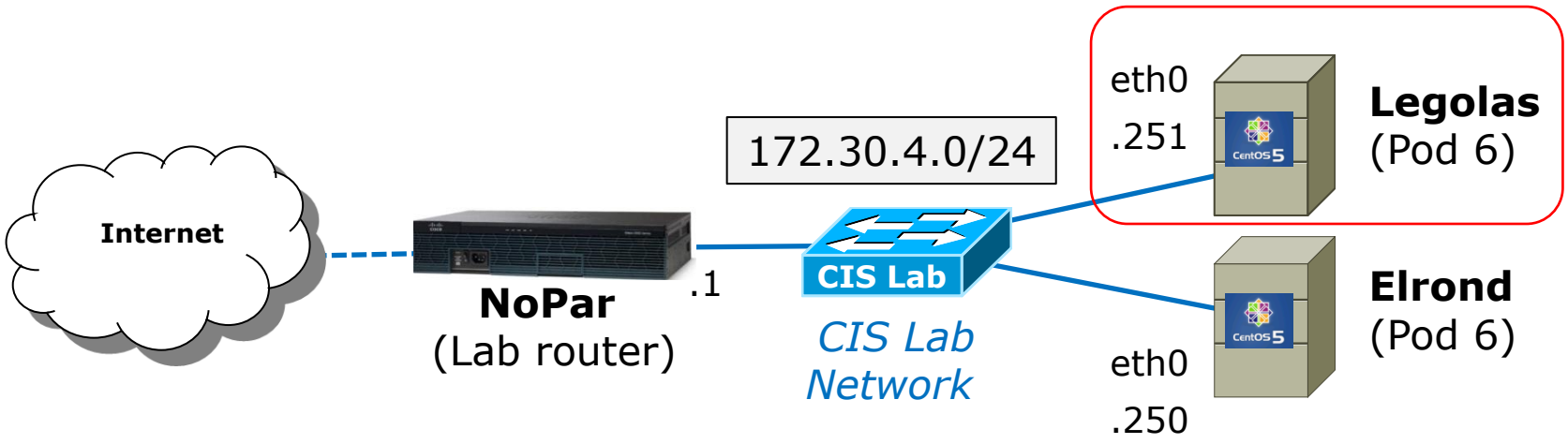
3. Check name resolution (DNS settings) by ping a node on the Internet by name.

Always work your way up the stack one layer at a time

Troubleshoot Network Connection

```
[root@legolas ~] ping –c4 172.30.4.1
PING 172.30.4.1 (172.30.4.1) 56(84) bytes of data.
64 bytes from 172.30.4.1: icmp_seq=1 ttl=255 time=3.90 ms
64 bytes from 172.30.4.1: icmp_seq=2 ttl=255 time=0.593 ms
64 bytes from 172.30.4.1: icmp_seq=3 ttl=255 time=0.596 ms
64 bytes from 172.30.4.1: icmp_seq=4 ttl=255 time=0.586 ms

--- 172.30.4.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.586/1.420/3.907/1.436 ms
```

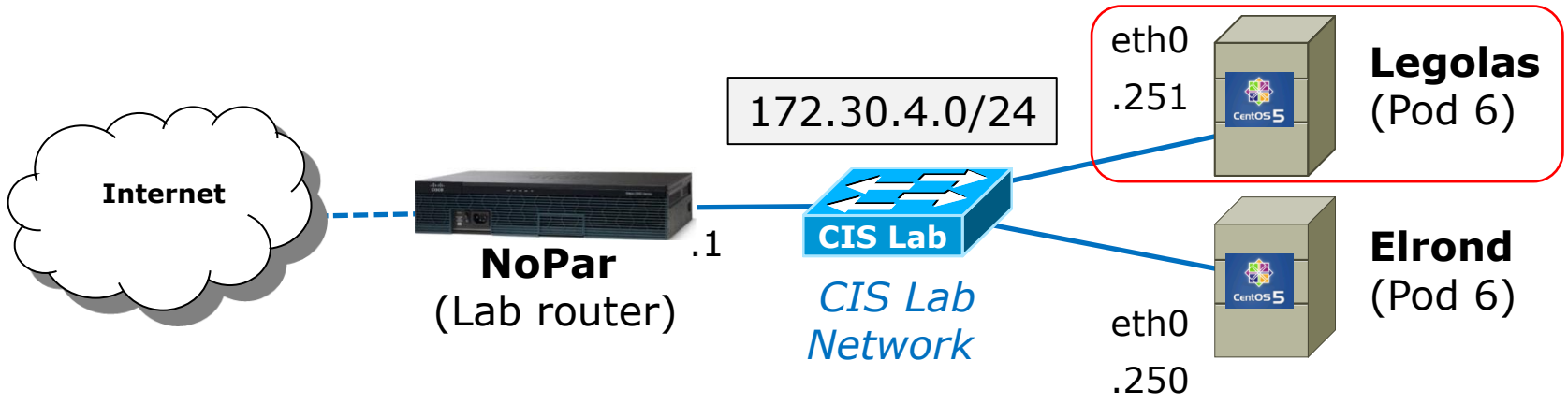
Success!



Troubleshoot Legolas connection (step 1)

Check cabling, IP and subnet mask settings by pinging another node on the same local network (which could be the router) using an IP address.

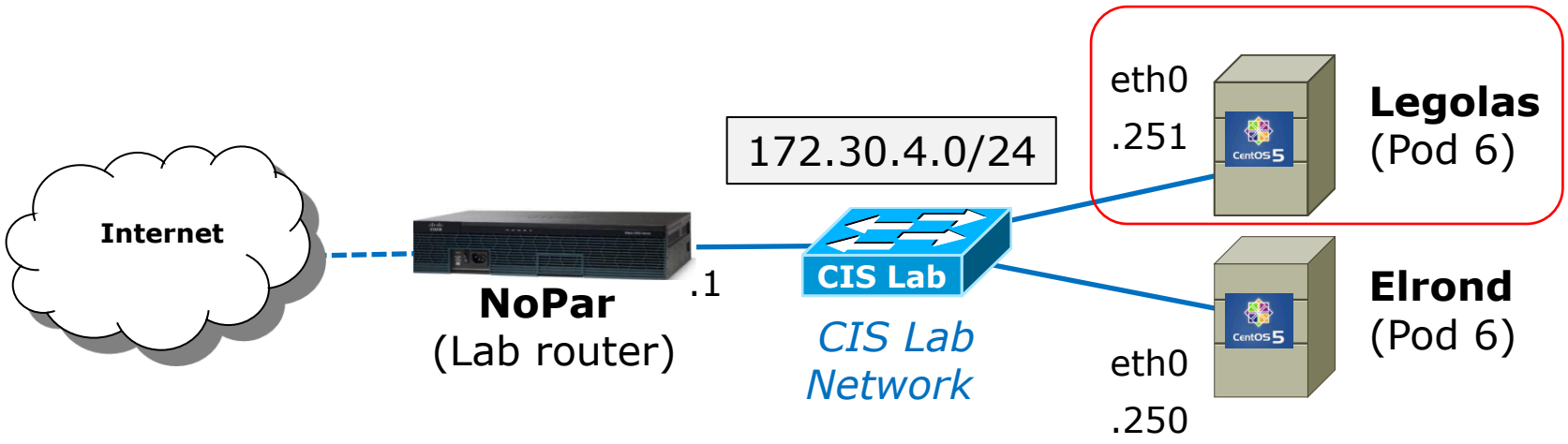
Ping test	Cabling	IP	Subnet mask	Default Gateway	DNS name servers	Ping results
ping 172.30.4.1	correct	correct	correct	correct	correct	Success
ping 172.30.4.1	Mordor (wrong network)	correct	correct	correct	correct	**Destination Host Unreachable**, 100% packet loss
ping 172.30.4.1	correct	172.30.4.250 (DUP)	correct	correct	correct	Variable amount of **packet loss**. More loss when other node, Elrond, is active.
ping 172.30.4.1	correct	192.30.4.251	correct	correct	correct	**Connect: Network is unreachable**



Troubleshoot Network Connection

```
[root@legolas ~] ping –c4 10.240.1.2
PING 10.240.1.2 (10.240.1.2) 56(84) bytes of data.
64 bytes from 10.240.1.2: icmp_seq=1 ttl=62 time=1.65 ms
64 bytes from 10.240.1.2: icmp_seq=2 ttl=62 time=1.67 ms
64 bytes from 10.240.1.2: icmp_seq=3 ttl=62 time=1.11 ms
64 bytes from 10.240.1.2: icmp_seq=4 ttl=62 time=1.15 ms

--- 10.240.1.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 1.118/1.401/1.678/0.270 ms
```

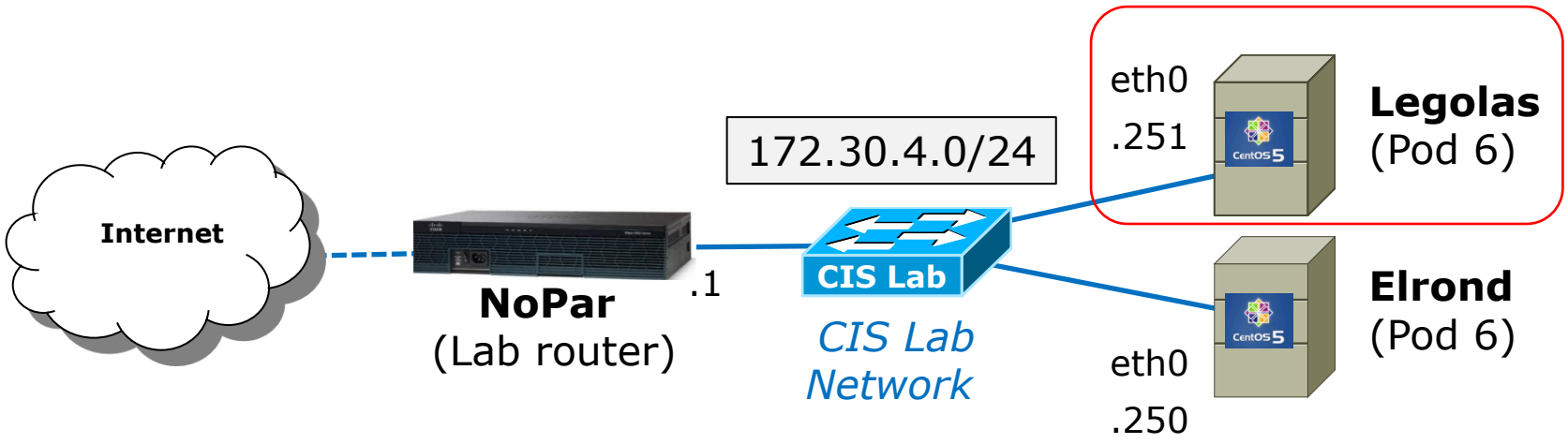
Success!



Troubleshoot Legolas connection (step 2)

Check default gateway by pinging a node outside the local network using an IP address.

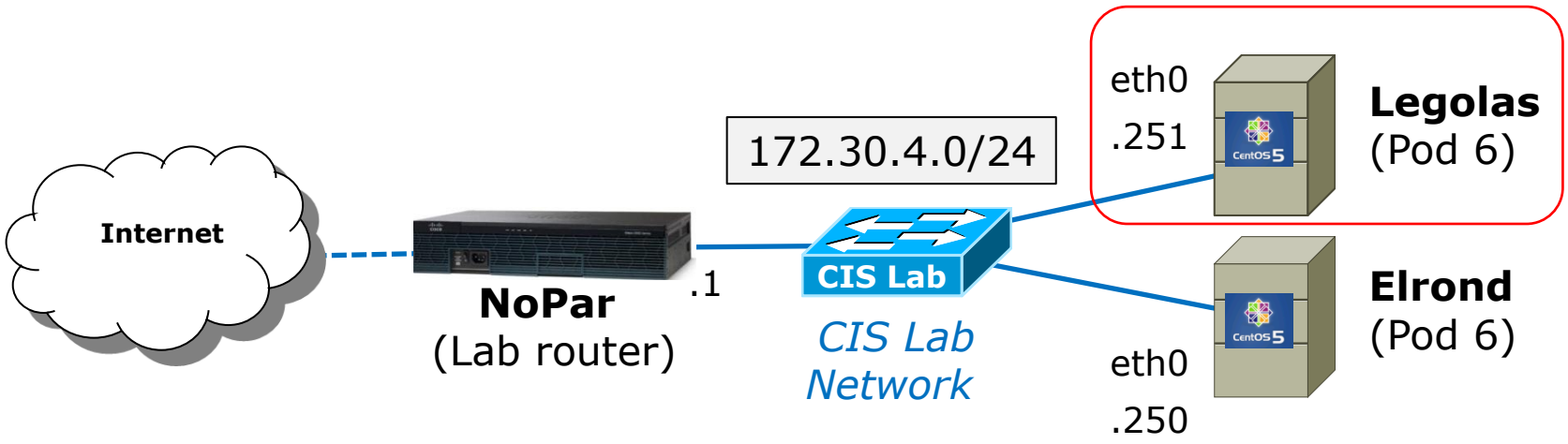
Ping test	Cabling	IP	Subnet mask	Default Gateway	DNS name servers	Ping results
ping 10.240.1.2	correct	correct	correct	correct	correct	Success
ping 10.240.1.2	correct	correct	correct	not added	correct	**connect: Network is unreachable**
ping 10.240.1.2	correct	correct	correct	non router specified	correct	no error message but **100% packet loss**



Troubleshoot Network Connection

```
[root@legolas ~] ping –c4 gogle.com
PING google.com (74.125.224.145) 56(84) bytes of data.
64 bytes from nuq04s09-in-f17.1e100.net (74.125.224.145): icmp_seq=1 ttl=54 time=6.87 ms
64 bytes from nuq04s09-in-f17.1e100.net (74.125.224.145): icmp_seq=2 ttl=54 time=6.62 ms
64 bytes from nuq04s09-in-f17.1e100.net (74.125.224.145): icmp_seq=3 ttl=54 time=6.64 ms
64 bytes from nuq04s09-in-f17.1e100.net (74.125.224.145): icmp_seq=4 ttl=54 time=6.59 ms

--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3012ms
rtt min/avg/max/mdev = 6.593/6.684/6.871/0.136 ms
```

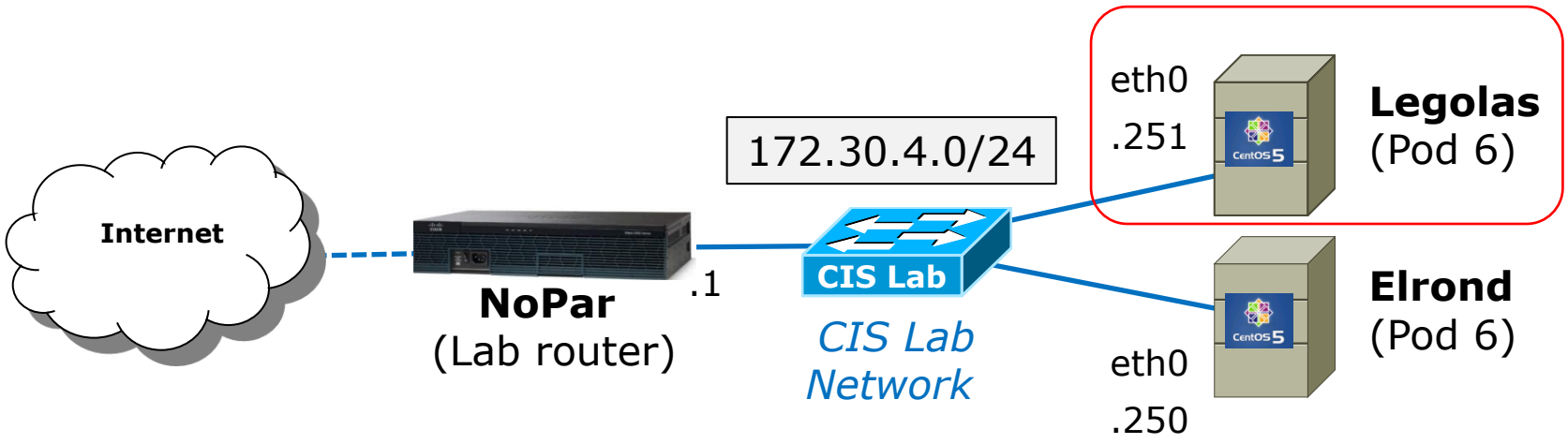
Success!



Troubleshoot Legolas connection (step 3)

Check name resolution (DNS settings) by ping a node on the Internet by name.

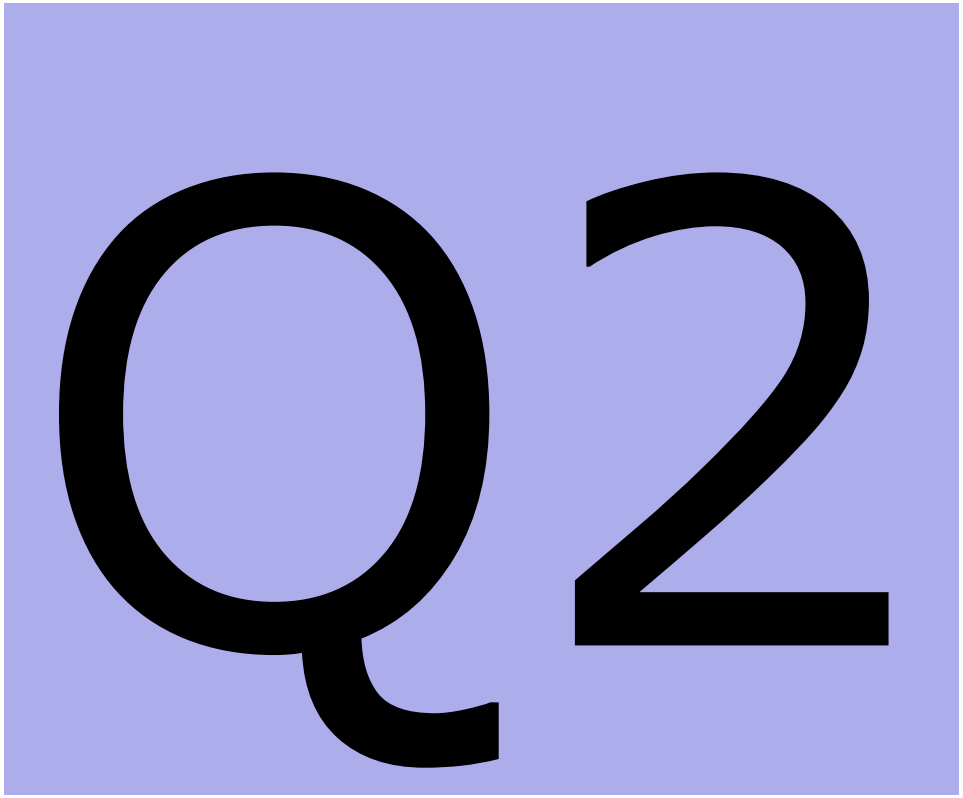
Ping test	Cabling	IP	Subnet mask	Default Gateway	DNS name servers	Ping results
ping google.com	correct	correct	correct	correct	correct	Success
ping google.com	correct	correct	correct	correct	none specified	**ping: unknown host google.com**



Class Activity
Troubleshooting Network Connections

Try it!

- Bad Cabling
- Missing default router
- Incorrect DNS server


Q2

ipv6 intro

Using IPv6 addresses in Linux

- IPv6 is a layer 3 protocol designed to replace IPv4
- The CentOS VMs for this course have the IPv6 module loaded into the kernel (**use lsmod | grep ipv6** to see it)
- IPv6 uses 128 bits to form an IP address as opposed to 32 bits in IPv4
- IPv4 IP address and mask do not need to be configured in order to use IPv6
- The loopback address for IPv6 is **::1**, for IPv4 it is **127.0.0.1**
- To ping yourself use **ping6 ::1**

Using IPv6 addresses in Linux – ping6

Elrond

lo

ping 127.0.0.1

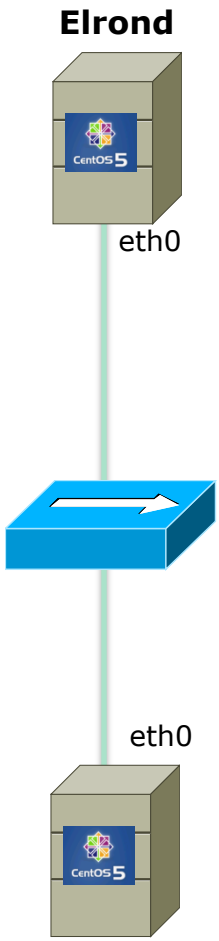
```
root@elrond ~]# ping 127.0.0.1
ING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
4 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.980 ms
4 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.095 ms

-- 127.0.0.1 ping statistics ---
 packets transmitted, 2 received, 0% packet loss, time 1000ms
tt min/avg/max/mdev = 0.095/0.537/0.980/0.443 ms
```

ping6 ::1

```
root@elrond ~]# ping6 ::1
ING ::1(::1) 56 data bytes
4 bytes from ::1: icmp_seq=0 ttl=64 time=0.330 ms
4 bytes from ::1: icmp_seq=1 ttl=64 time=0.265 ms

-- ::1 ping statistics ---
 packets transmitted, 2 received, 0% packet loss, time 1001ms
tt min/avg/max/mdev = 0.265/0.297/0.330/0.036 ms, pipe 2
```

Loopback address are used to make network connections to local services. Packets are not sent out the NIC to the network.

Using IPv6 addresses in Linux – ping6



ping6 –I eth0 fe80::20c:29ff:fe4b:f5ce

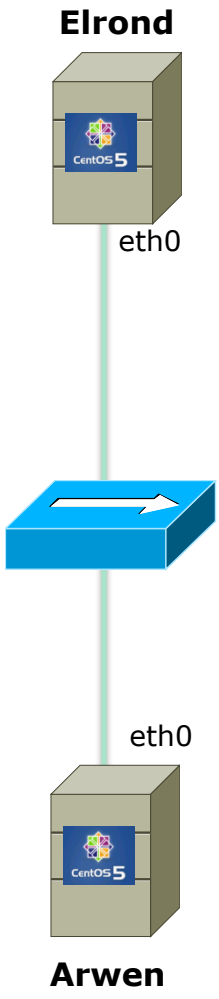
```
[root@elrond ~]# ping6 -I eth0 fe80::20c:29ff:fe4b:f5ce
PING fe80::20c:29ff:fe4b:f5ce(fe80::20c:29ff:fe4b:f5ce) from fe80::20c:29ff:fe68
:3687 eth0: 56 data bytes
64 bytes from fe80::20c:29ff:fe4b:f5ce: icmp_seq=0 ttl=64 time=2.30 ms
64 bytes from fe80::20c:29ff:fe4b:f5ce: icmp_seq=1 ttl=64 time=2.14 ms

--- fe80::20c:29ff:fe4b:f5ce ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 2.141/2.223/2.306/0.095 ms, pipe 2
[root@elrond ~]# _
```

Note: the interface must be specified on the ping6 command

```
eth0      Link encap:Ethernet  HWaddr 00:0C:29:4B:F5:CE
          inet6 addr: fe80::20c:29ff:fe4b:f5ce/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:713 errors:0 dropped:0 overruns:0 frame:0
          TX packets:605 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:557922 (544.8 KiB)  TX bytes:61674 (60.2 KiB)
          Interrupt:177 Base address:0x1400

[root@arwen ~]# _
```

*Use the **ifconfig** command to see what the ipV6 address is*

Using IPv6 addresses in Linux - ssh



ssh fe80::20c:29ff:fe4b:f5ce%eth0

```
[root@elrond ~]# ssh fe80::20c:29ff:fe4b:f5ce%eth0
root@fe80::20c:29ff:fe4b:f5ce%eth0's password:
Last login: Mon Jan 25 23:30:16 2010 from fe80::20c:29ff:fe68:3687%eth0
[root@arwen ~]# _
```

Note: the interface must be specified on the ssh command

```
eth0      Link encap:Ethernet  HWaddr 00:0C:29:4B:F5:CE
          inet6 addr: fe80::20c:29ff:fe4b:f5ce/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:713 errors:0 dropped:0 overruns:0 frame:0
          TX packets:605 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:557922 (544.8 KiB)  TX bytes:61674 (60.2 KiB)
          Interrupt:177 Base address:0x1400

[root@arwen ~]# _
```

*Use the **ifconfig** command to see what the ipV6 address is*

Class Activity
IPv6

Try it!

- [Elrond] ifconfig eth0
- [Frodo] ping6 -I eth0 fe80::20c:29ff:fed8:847f
- [Frodo] ssh cis192@fe80::20c:29ff:fed8:847f%eth0
- [Frodo] who

ifconfig and aliases

Alias IP Addresses

What is it

• It lets you assign more than one IP address to an interface

Why?

• It give you additional flexibility for customizing access to different groups of users for different services

It is possible to have more than one IP address on an interface using aliases. This is different than multi-homing which is having multiple interfaces on a computer.

Create an Alias IP Address (more than one IP address per interface)

Set

- To set an alias IP address and subnet mask:
  ifconfig eth*n:m* xxx.xxx.xxx.xxx **netmask** xxx.xxx.xxx.xxx

Verify

- To show all interfaces (and to show your IP address):
  ifconfig

- To show a single alias interface:
  ifconfig eth*n:m*

Create an Alias IP Address Example

ifconfig eth0 172.30.1.125/24
ifconfig eth0:1 172.30.1.200/24

```
[root@elrond ~]# ifconfig eth0 172.30.1.125/24
[root@elrond ~]# ifconfig eth0:1 172.30.1.200/24
```

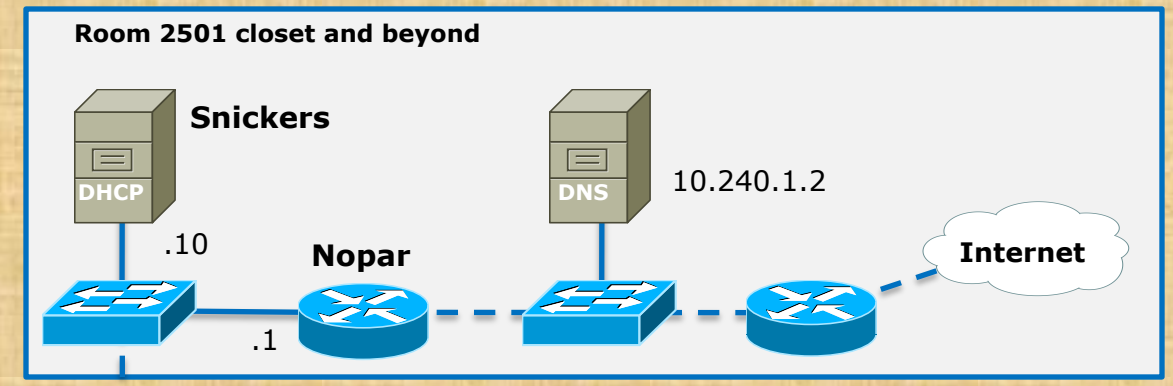
ifconfig eth0

```
[root@elrond ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:10:4F:D8
          inet addr:172.30.1.125  Bcast:172.30.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe10:4fd8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:36 errors:0 dropped:0 overruns:0 frame:0
          TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4567 (4.4 KiB)  TX bytes:1574 (1.5 KiB)
          Interrupt:19 Base address:0x2024
```

ifconfig eth0:1

```
[root@elrond ~]# ifconfig eth0:1
eth0:1    Link encap:Ethernet  HWaddr 00:0C:29:10:4F:D8
          inet addr:172.30.1.200  Bcast:172.30.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:19 Base address:0x2024
```

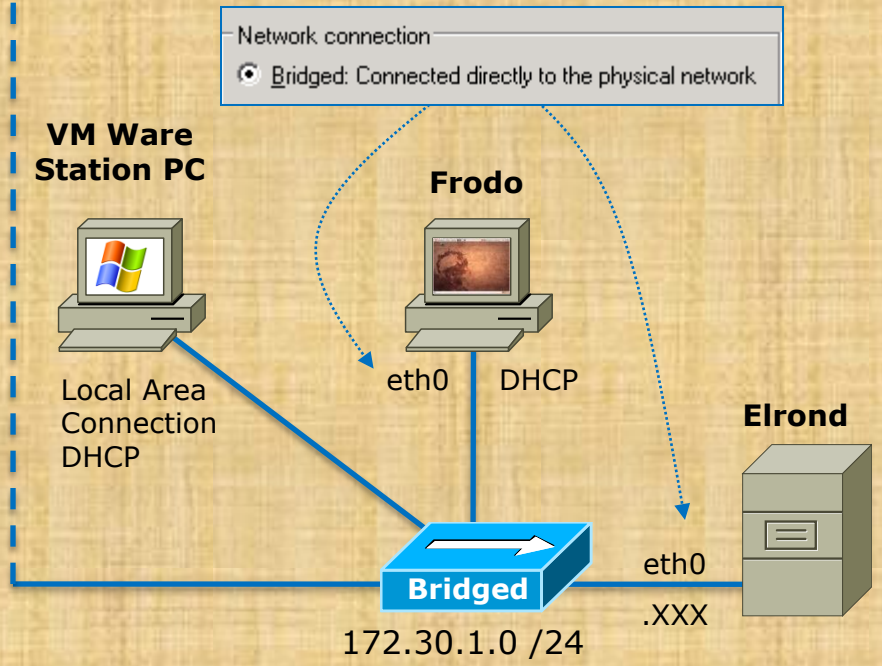
Class Exercise – Add an alias IP address



47 days till term ends!

Cabrillo College
Web Advisor
Static IPs
Quick Ref
Accessing VLab

No DUPS Please!



Elrond (static)
- ifconfig eth0:1 172.30.1.xxx/24
- ifconfig

Frodo (dhcp)
- ping both of Elrond's IP addresses

ARP

ARP – Address Resolution Protocol

Overview

The purpose of ARP is to provide the correct destination physical address given the destination IP address.

- RFC 826 (http://tools.ietf.org/html/rfc826)
- Part of IPv4 (IPv6 uses NDP, neighbor discovery protocol)
- The ARP request: generates and broadcasts its own request packet - "Who has this IP address?"
- The ARP reply: targeted to the requestor's address (unicast) – "I do and my MAC address is *xx:xx:xx:xx:xx:xx*"

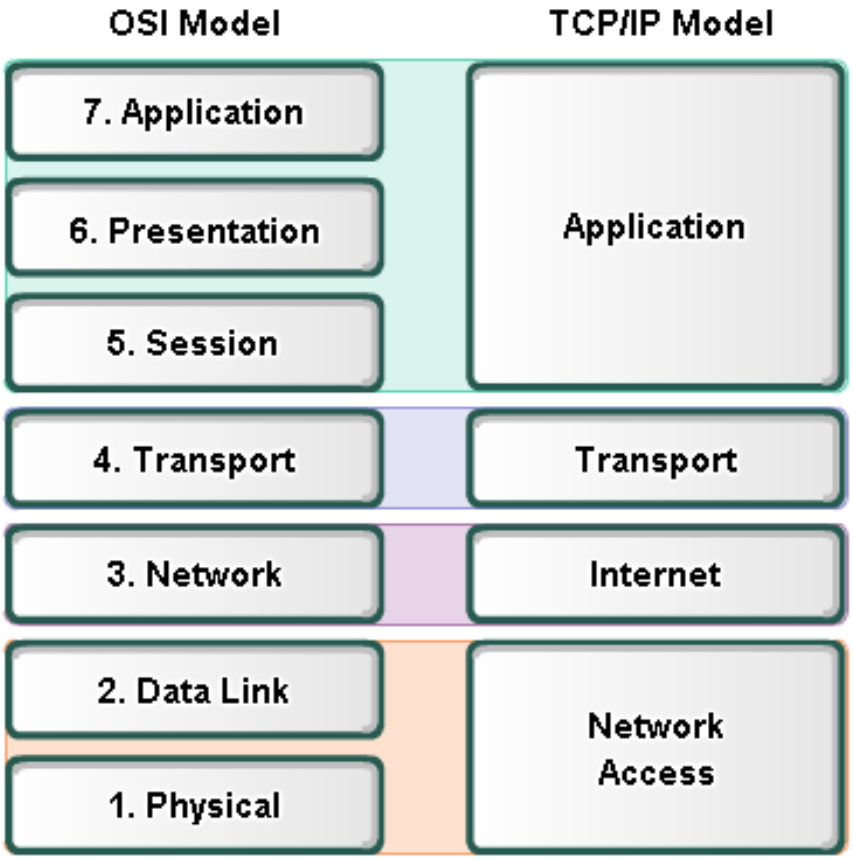
TCP/IP and ARP

The TCP/IP Suite of Protocols	
Application	File Transfer: FTP, TFTP, NFS, HTTP Email: SMTP Remote Login: Telnet, rlogin Network Management: SNMP, BootP Name Management: DNS, DHCP
Transport	TCP, UDP
Internet/Network	IP, ICMP, IGMP, **ARP, RARP**
Network Interface (Link Layer)	Not Specified: Ethernet, 802.3, Token Ring, 802.5, FDDI, ATM,

ARP is a layer 3 protocol, one of many protocols within the TCP/IP suite of protocols.

Protocol and Reference Models

ARP is a layer 3 protocol



- The **Open Systems Interconnection (OSI)** model is the *most widely known internetwork reference model*.

ARP – Address Resolution Protocol

Overview Example

Station04 wants to ping Station20

ARP – Address Resolution Protocol

Overview

Devices will remember pairings of IP addresses and MAC addresses which are kept in an ARP cache table

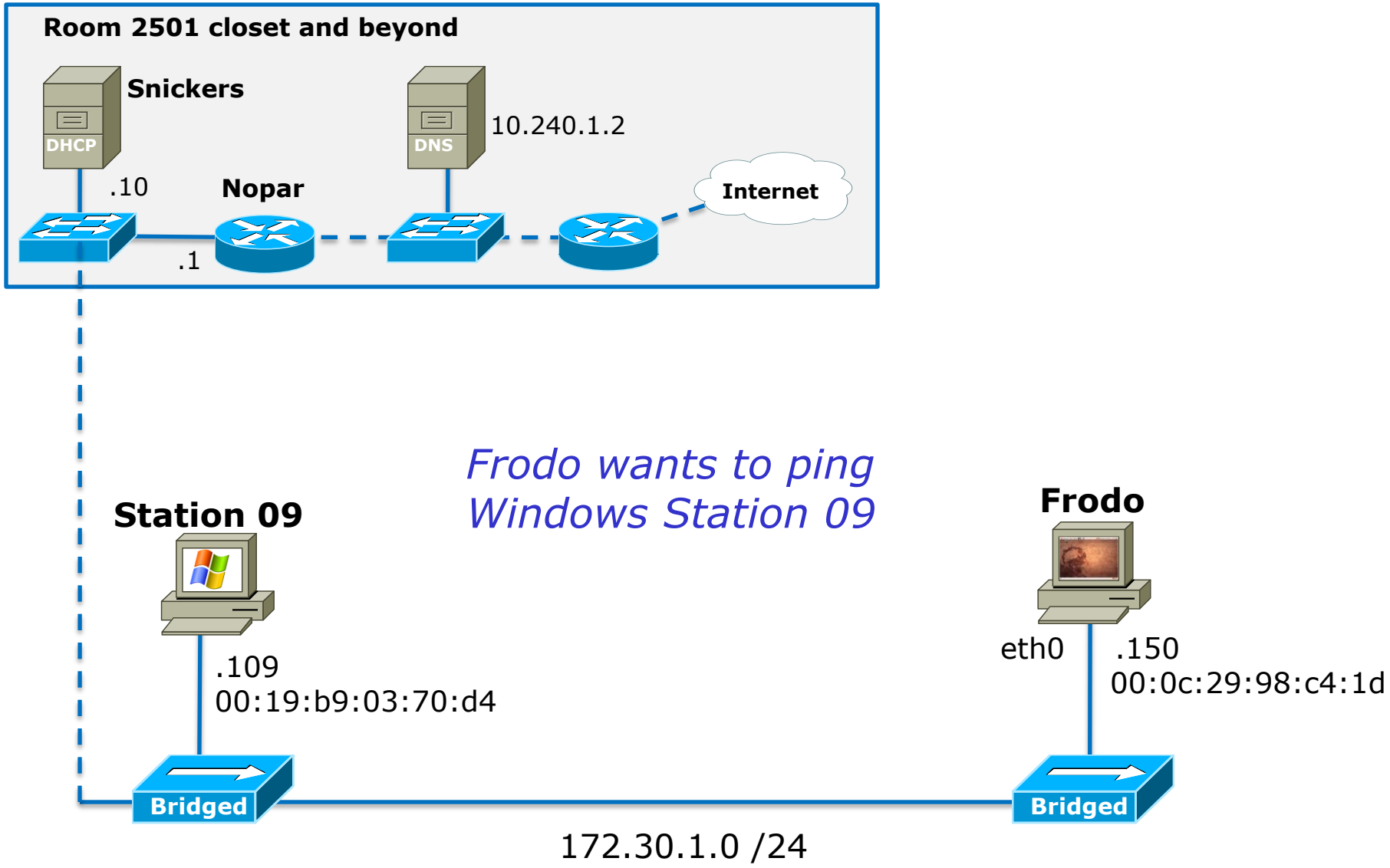
- In Linux, the **arp** command is used to show the ARP cache
- ARP cache entries will eventually timeout and be removed

ARP Poisoning

A NIC is gullible and will accept ARP replies even when not requested

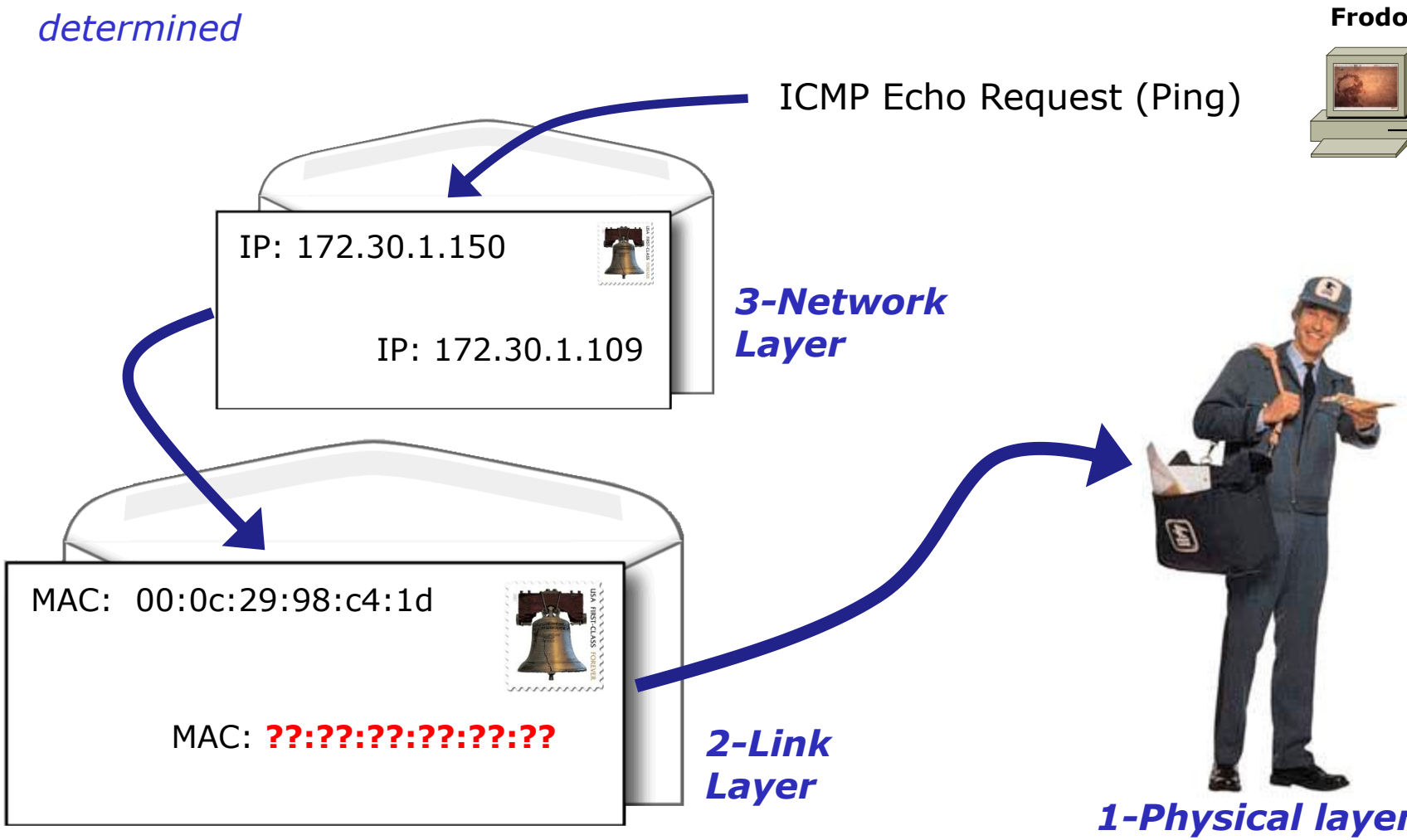
- An attacker can send arp replies (even as a broadcast) to populate arp caches with bogus MAC/IP pairs
  - Denial of service: pair a non-existing MAC address with the router's IP address. External destination packets can never leave the subnet.
  - Man-in-the-middle: pair an existing hosts IP address with attackers MAC address so attacker can snoop all packets for that host.
  - MAC flooding: overload a switch so it behaves like a hub allowing a sniffer to see all traffic.

ARP Example - Frodo pings Station09

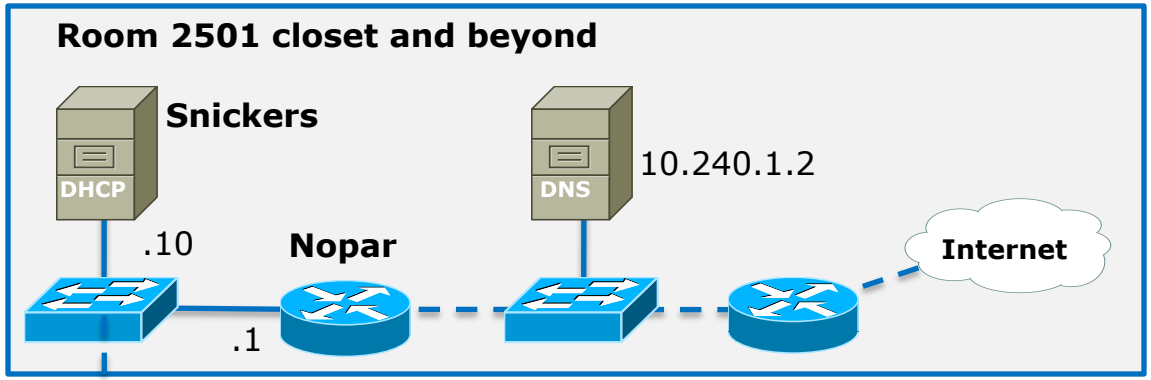


ARP Example - Frodo pings Station09

However, using encapsulation, the ping packet cannot be placed on the network until a destination MAC address for Station 09 can be determined



ARP Example - Frodo pings Station09



Step 1: *Who has IP Address 172.30.1.109? (broadcast to all) Tell 172.30.1.150 at 00:0c:29:98:c4:1d*

Step 2: *I do (unicast to 172.30.1.150) I'm at 00:19:b9:03:70:d4*

Station 09

.109
00:19:b9:03:70:d4

ARP is used to get the destination MAC address

Frodo

eth0 .150
00:0c:29:98:c4:1d

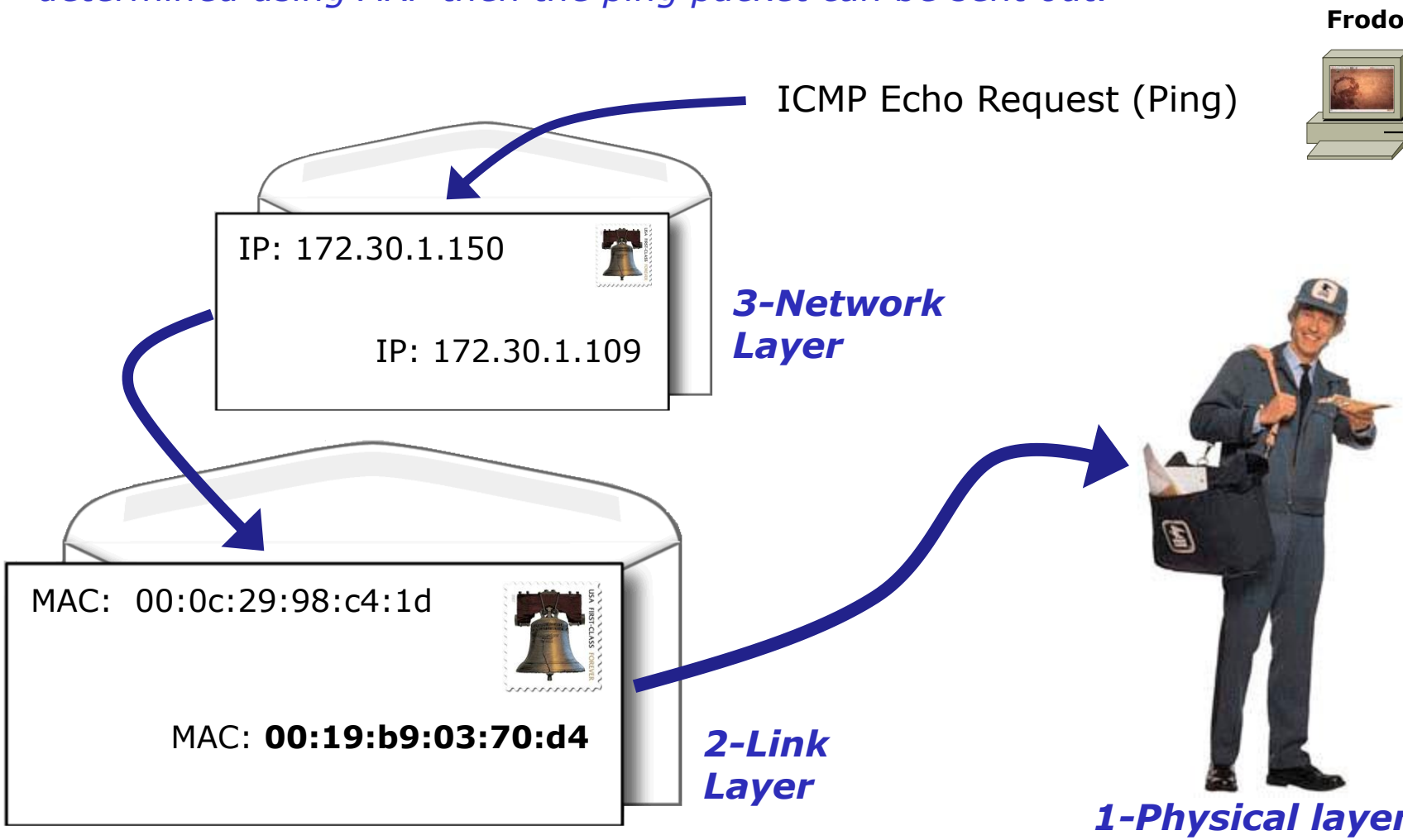
Bridged

Bridged

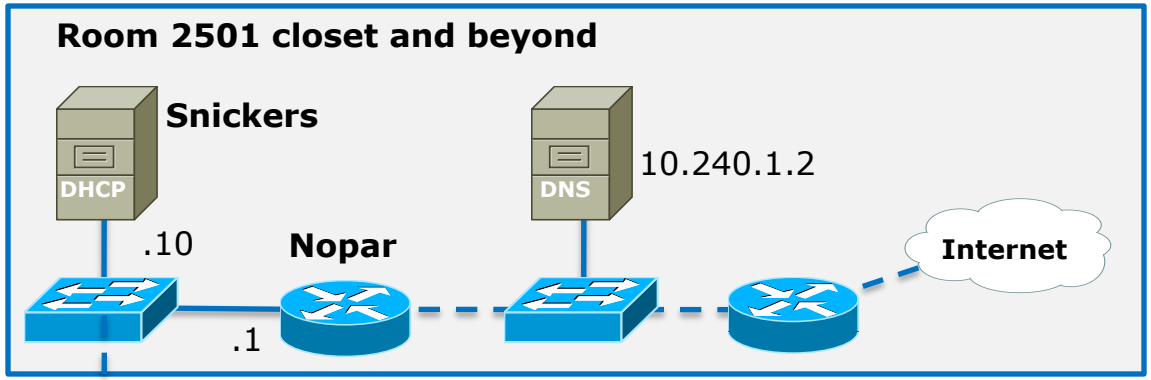
172.30.1.0 /24

ARP Example - Frodo pings Station09

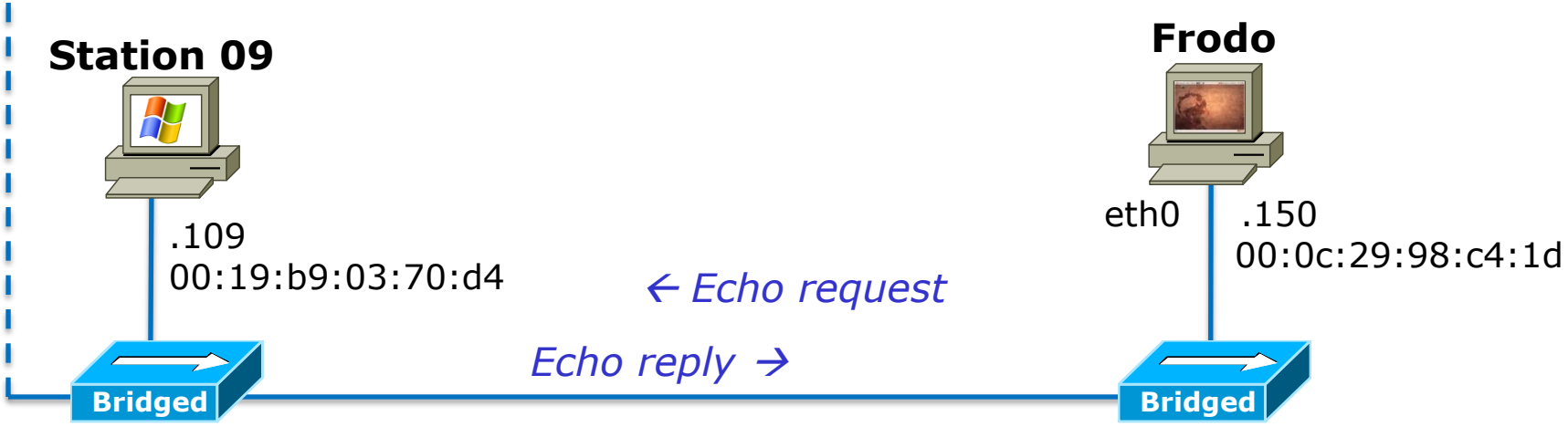
Once the destination MAC address for Station 09 has been determined using ARP then the ping packet can be sent out.



ARP Example - Frodo pings Station09



Once the destination MAC address for Station 09 has been determined using ARP then the ping packet can be sent out and the reply is sent back.



ARP Example - Frodo pings Station09

```
root@frodo:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0c:29:98:c4:1d
          inet addr:172.30.1.150  Bcast:172.30.1.255  Mask:255.255.255.0
< snipped>
```

Frodo's IP address is 172.30.1.150

```
root@frodo:~# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
172.30.1.1               ether   00:b0:64:53:42:01   C                     eth0
```

Frodo's ARP cache currently only has one entry and that is for the router

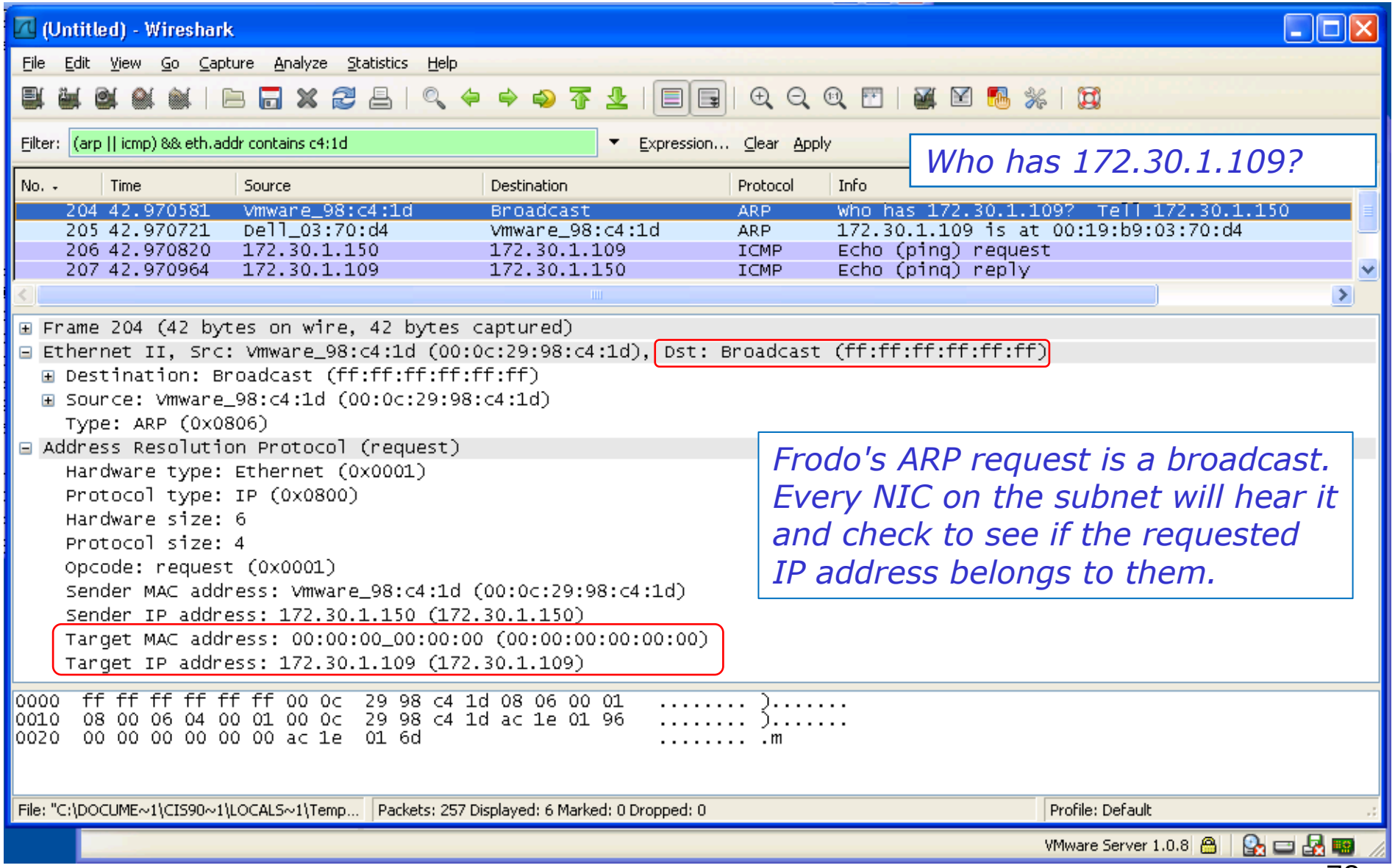
```
root@frodo:~# ping -c 1 172.30.1.109
PING 172.30.1.109 (172.30.1.109) 56(84) bytes of data.
64 bytes from 172.30.1.109: icmp_seq=1 ttl=128 time=3.71 ms
< snipped >
```

The ping command will result in an ARP request to get Station09 MAC address and this will be placed in the ARP cache

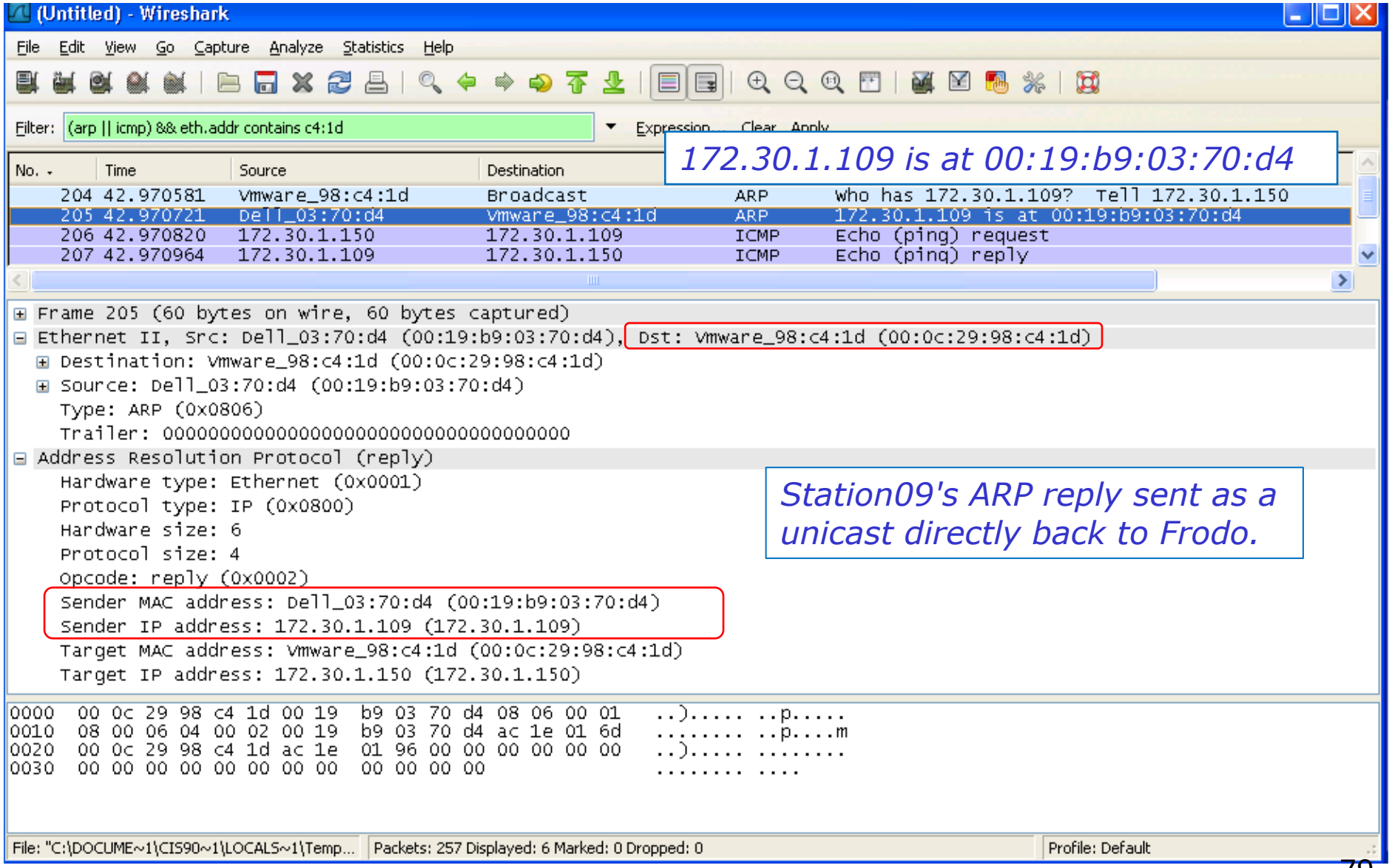
```
root@frodo:~# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
172.30.1.109             ether   00:19:b9:03:70:d4   C                     eth0
172.30.1.1               ether   00:b0:64:53:42:01   C                     eth0
```

The new MAC/IP pair for Station 09 has been added to the ARP cache

ARP Example - Frodo pings Station09



ARP Example - Frodo pings Station09



ARP Cache

Showing the ARP cache

- List ARP cache entries (IP/MAC pairs)

  arp

  arp –n *(no name resolution, faster)*

  arp –a *(uses BSD format for output)*

  ip neigh show *(shows more state information)*

- Delete ARP cache entries (IP/MAC pairs)

  ip neigh flush all

Showing the ARP cache

```
[root@elrond ~]# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
172.30.1.8                       (incomplete)                              eth0
172.30.1.196             ether   00:0C:29:BF:E4:F9   C                     eth0
172.30.1.108             ether   C8:00:0A:5C:00:00   C                     eth0
nosmo                    ether   00:0C:29:49:88:B8   C                     eth0

[root@elrond ~]# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
172.30.1.8                       (incomplete)                              eth0
172.30.1.196             ether   00:0C:29:BF:E4:F9   C                     eth0
172.30.1.108             ether   C8:00:0A:5C:00:00   C                     eth0
172.30.1.1               ether   00:0C:29:49:88:B8   C                     eth0

[root@elrond ~]# arp -a
? (172.30.1.8) at <incomplete> on eth0
? (172.30.1.196) at 00:0C:29:BF:E4:F9 [ether] on eth0
? (172.30.1.108) at C8:00:0A:5C:00:00 [ether] on eth0
nosmo (172.30.1.1) at 00:0C:29:49:88:B8 [ether] on eth0
```

*The **incomplete** entry resulted from pinging a non-existent device at 172.30.1.8*

C= complete

- -

```
[root@elrond ~]# ip neigh show
172.30.1.8 dev eth0  FAILED
172.30.1.196 dev eth0 lladdr 00:0c:29:bf:e4:f9 STALE
172.30.1.108 dev eth0 lladdr c8:00:0a:5c:00:00 STALE
172.30.1.1 dev eth0 lladdr 00:0c:29:49:88:b8 REACHABLE
```

Stale = getting old but should be still reachable

Showing the ARP cache

Flags shown on ARP command output:

- Complete (C) 0x02
- Permanent (M) 0x04
- Published (P) 0x08

Temporary ARP cache entries are aged out after several minutes.

Till next system restart

The system will act as a ARP server and respond to ARP requests for IP addresses that are not its own

*Note, there may be **incomplete** entries for failed ARP requests (pinging a non-existent or powered-off device) or entries that were manually deleted*

ARP commands on the different planets

[root@elrond ~]# arp -n

```
Address                  HWtype  HWaddress           Flags Mask            Iface
172.30.1.108             ether   C8:00:0A:5C:00:00   C                     eth0
172.30.1.1               ether   00:0C:29:49:88:B8   C                     eth0
```

R1#show arp

```
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  192.168.2.10           -   c800.0a5c.0001  ARPA   FastEthernet0/1
Internet  172.30.1.1             0   000c.2949.88b8  ARPA   FastEthernet0/0
Internet  172.30.1.107           8   000c.2968.3687  ARPA   FastEthernet0/0
Internet  172.30.1.108           -   c800.0a5c.0000  ARPA   FastEthernet0/0
```

C:\Users\Administrator>arp -a

```
Interface: 192.168.0.21 --- 0xe
  Internet Address      Physical Address      Type
  192.168.0.1           00-a0-c5-e1-c9-a8     dynamic
  192.168.0.2           00-0c-29-49-88-ae     dynamic
  192.168.0.12          00-14-38-9c-59-5f     dynamic
  192.168.0.18          00-24-8d-85-55-85     dynamic
  192.168.0.25          00-0c-6e-51-4c-2d     dynamic
  192.168.0.27          00-0c-f1-96-8e-68     dynamic
  192.168.0.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  224.0.0.253           01-00-5e-00-00-fd     static
  239.192.152.143       01-00-5e-40-98-8f     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

ARP command

Additional options and arguments

- List ARP cache entry for a host

  arp –a 172.30.1.1

- Add permanent ARP entries (lasts until next restart)

  arp –s 172.30.1.1 00:b0:64:53:42:01 *(add one IP/MAC entry)*

  arp –f /etc/ethers *(ASCII file of MAC/IP entries)*

- Delete ARP entry

  arp –d 172.30.1.1

arp command

Populate the arp cache via ping usage

Before

```
root@frodo:~# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
172.30.1.109             ether   00:19:b9:03:70:d4   C                     eth0
172.30.1.1               ether   00:b0:64:53:42:01   CM                    eth0

root@frodo:~# ping 172.30.1.110
PING 172.30.1.110 (172.30.1.110) 56(84) bytes of data.
64 bytes from 172.30.1.110: icmp_seq=1 ttl=128 time=0.741 ms
< snipped >
root@frodo:~# ping 172.30.1.111
PING 172.30.1.111 (172.30.1.111) 56(84) bytes of data.
64 bytes from 172.30.1.111: icmp_seq=1 ttl=128 time=2.01 ms
< snipped >
```

After

```
root@frodo:~# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
172.30.1.1               ether   00:b0:64:53:42:01   CM                    eth0
172.30.1.109             ether   00:19:b9:03:70:d4   C                     eth0
172.30.1.111             ether   00:18:8b:28:ac:ab   C                     eth0
172.30.1.110             ether   00:19:b9:03:71:00   C                     eth0
```

Note the new entries for 172.30.1.110 and 172.30.1.111 that were added because of the last two pings.

arp command

Populate the arp cache via manual entries

Before

```
root@frodo:~# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
172.30.1.109             ether   00:19:b9:03:70:d4   C                     eth0
172.30.1.1               ether   00:b0:64:53:42:01   C                     eth0
```

Add permanent entry for a node

```
root@frodo:~# arp -s 172.30.1.1 00:b0:64:53:42:01
```

After

```
root@frodo:~# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
172.30.1.109             ether   00:19:b9:03:70:d4   C                     eth0
172.30.1.1               ether   00:b0:64:53:42:01   CM                    eth0
```

CM flags = Complete and Permanent

arp cache

Populating the arp cache via a file option

Before

```
root@frodo:~# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
172.30.1.109             ether   00:19:b9:03:70:d4   C                     eth0
```

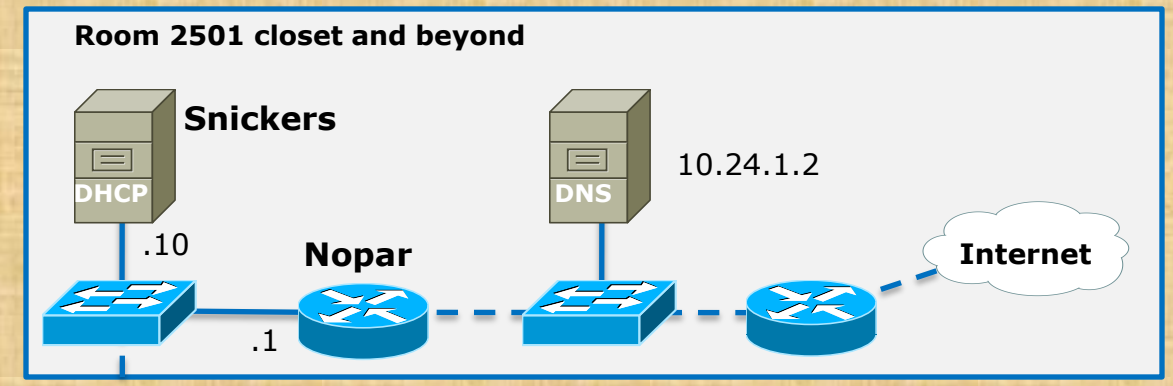
```
root@frodo:~# vi /etc/ethers
root@frodo:~# cat /etc/ethers
172.30.1.1       00:b0:64:53:42:01
172.30.1.10      00:90:27:76:97:ab

root@frodo:~# arp -f /etc/ethers
```

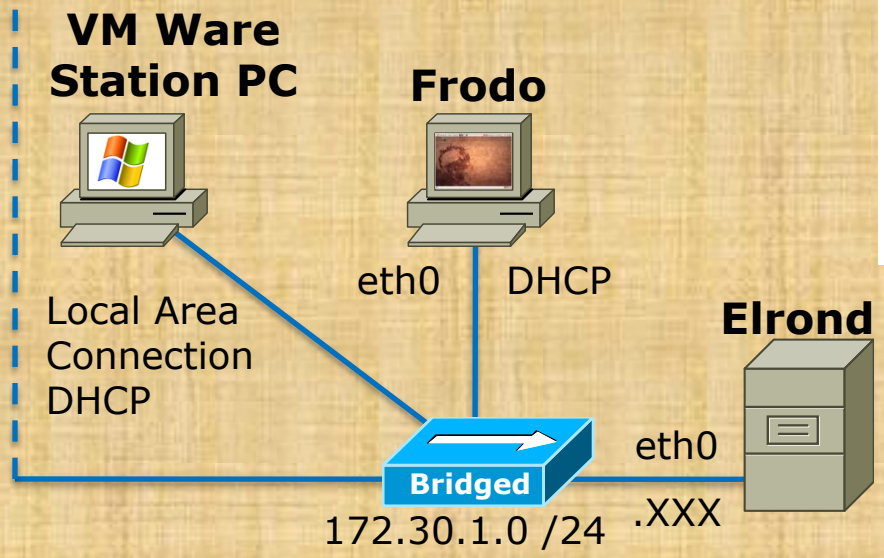
Permanent entries can also be added from a file using the –f option.

After

```
root@frodo:~# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
172.30.1.1               ether   00:b0:64:53:42:01   CM                    eth0
172.30.1.109             ether   00:19:b9:03:70:d4   C                     eth0
172.30.1.10              ether   00:90:27:76:97:ab   CM                    eth0
```

Class Exercise – populate and view the ARP cache





Frodo

- arp -n
- Ping router, VMware station, and Elrond
- arp -n

arpwatch

arpwatch

Track IP/MAC pairs

The arpwatch daemon

- Collects IP/MAC address pairs
- Save pairs in a log file: arp.dat
- Emails root as pairs are found
- Great way to inventory MAC addresses or monitor for fraudulent activity

arpwatch installation (Red Hat family)

```
--> Processing Dependency: libpcap.so.1 fo
686
--> Running transaction check
---> Package libpcap.i686 14:1.0.0-6.20091
--> Finished Dependency Resolution

Dependencies Resolved

================================================
 Package       Arch      Version

================================================
Installing:
 arpwatch      i686      14:2.1a15-14.el6
Installing for dependencies:
 libpcap       i686      14:1.0.0-6.200912

Transaction Summary
================================================
Install       2 Package(s)
Upgrade       0 Package(s)

Total download size: 292 k
Installed size: 766 k
Is this ok [y/N]: _
```

Install **arpwatch** if necessary:

- **rpm –qa | grep arpwatch**
- **yum install arpwatch**

Install **/bin/mail** if necessary:

- **rpm –qa | grep mailx**
- **yum install mailx**

arpwatch installation (Debian family)

```
root@frodo:~# dpkg -l | grep arpwatch
root@frodo:~# apt-get install arpwatch
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed
  arpwatch
0 upgraded, 1 newly installed, 0 to remove a
Need to get 185 kB of archives.
After this operation, 647 kB of additional d
Get:1 http://us.archive.ubuntu.com/ubuntu/ na
.1 [185 kB]
Fetched 185 kB in 2s (89.0 kB/s)
Selecting previously deselected package arpwa
(Reading database ... 132286 files and direct
Unpacking arpwatch (from .../arpwatch_2.1a15-
Processing triggers for man-db ...
Processing triggers for ureadahead ...
ureadahead will be reprofiled on next reboot
Setting up arpwatch (2.1a15-1.1) ...
Starting Ethernet/FDDI station monitor daemon
/arp.dat) arpwatch.
root@frodo:~#
```

Install **arpwatch** if necessary:

- **dpkg –l | grep arpwatch**
- **apt-get install arpwatch**

Install **/bin/mail** if necessary:

- **dpkg –l | grep sendmail**
- **apt-get install sendmail**

- **dpkg –l | grep heirloom-mail**
- **apt-get install heirloom-mail**

arpwatch

Collect MAC / IP pairs

[Red Hat family] **service arpwatch start**

or [Red Hat or Debian family] **/etc/init.d/arpwatch start**

The collection starts now. As new pairs are detected they get emailed. arp.dat file is not updated till arpwatch is restarted

[Red Hat family] **service arpwatch restart**

or [Red Hat or Debian family] **/etc/init.d/arpwatch restart**

```
[root@elrond ~]# cat /var/lib/arpwatch/arp.dat
0:b:fc:28:41:0      172.30.1.5      1234303973
0:c:29:a4:83:bc     172.30.1.126    1234303772
0:13:7f:55:f9:0     172.30.1.4      1234303973
0:3:e3:6c:77:80     172.30.1.3      1234303973
0:b0:64:53:42:1     172.30.1.1      1234303772
0:18:8b:28:ac:50    172.30.1.121    1234304404
0:19:b9:3:71:f5     172.30.1.120    1234304072
0:90:27:76:97:ab    172.30.1.10     1234304341
0:c:29:e4:be:d3     172.30.1.152    1234303463
0:19:b9:3:71:cc     172.30.1.103    1234303636
0:c:29:46:5:73      172.30.1.153    1234303945
```

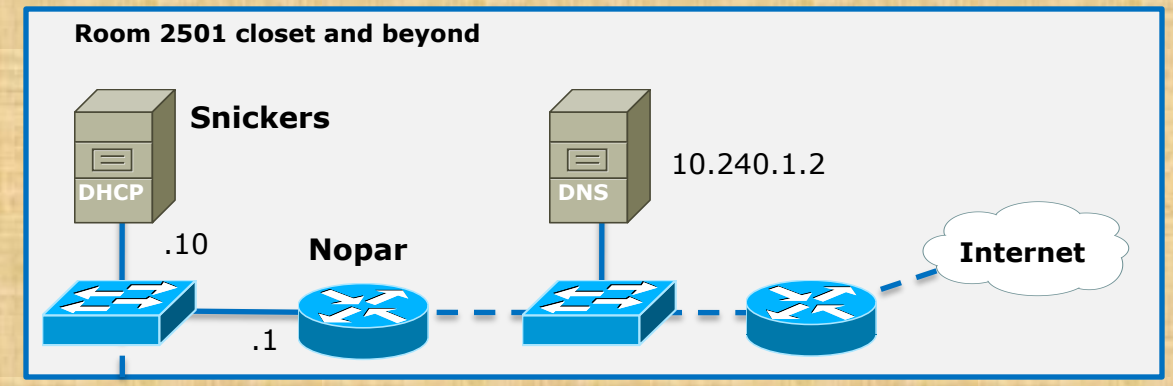
arpwatch

New pairs are emailed

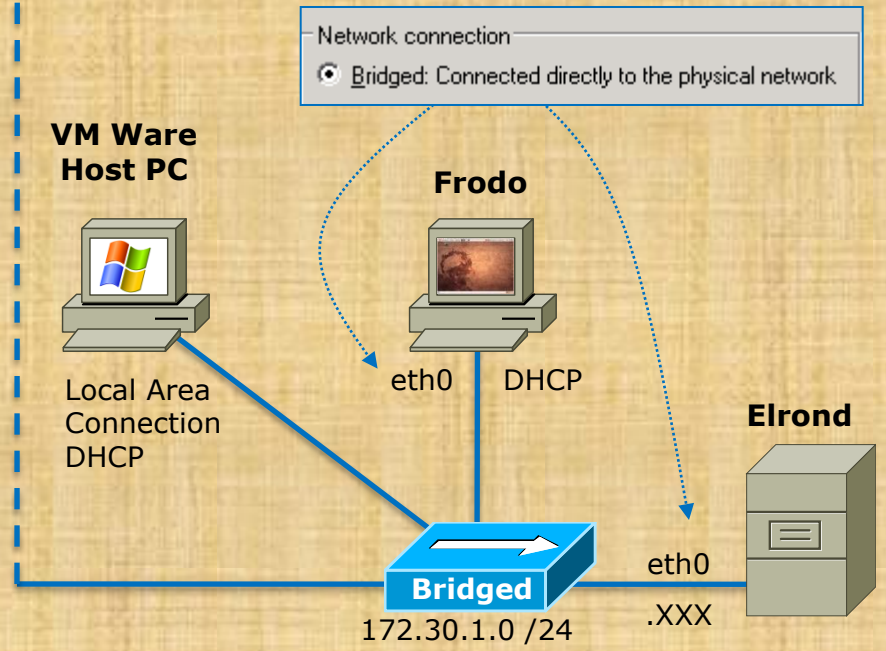
```
[root@elrond ~]# mail
Heirloom Mail version 12.4 7/29/08.  Type ? for help.
"/var/spool/mail/root": 4 messages 4 new
>N  1 Arpwatch              Tue Nov  1 07:15  18/667   "new station"
 N  2 Arpwat
 N  3 Arpwat
 N  4 Arpwat
&
```

```
Message  4:
From arpwatch@elrond.localdomain  Tue Nov  1 07:16:07 2011
Return-Path: <arpwatch@elrond.localdomain>
X-Original-To: root
Delivered-To: root@elrond.localdomain
From: root@elrond.localdomain (Arpwatch)
To: root@elrond.localdomain
Subject: new station
Date: Tue,  1 Nov 2011 07:16:07 -0700 (PDT)
Status: R

            hostname: <unknown>
          ip address: 172.30.1.151
    ethernet address: 0:c:29:db:1d:64
     ethernet vendor: VMware, Inc.
           timestamp: Tuesday, November 1, 2011 7:16:07 -0700

&
```

Class Exercise – Setting up arpwatch on Elrond

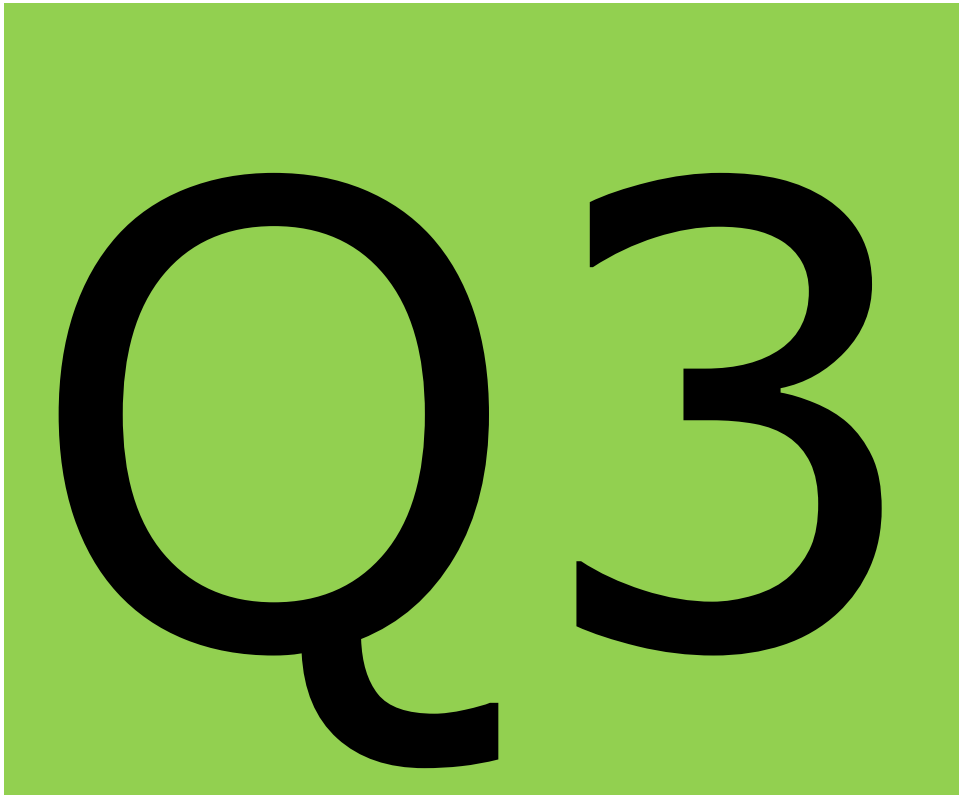


Try it!



Elrond

- Configure eth0 (previous exercise)
- **yum install arpwatch mailx**
- **service arpwatch start**
- Ping some other 172.30.1.xxx systems
- **service arpwatch restart**
- **cat /var/arpwatch/arp.dat**


Q3

Viewing Packets

Viewing Network Packets

Some sniffer options:

- Use tcpdump command on the Linux systems
  - [CentOS VMs] **yum install tcpdump**
  - [Ubuntu VMs] *already installed*
- Run Wireshark on the Classroom or Lab PCs
- Run Wireshark on the William VM (has Wireshark installed)
- Install and run Wireshark on Ubuntu VMs, use **apt-get install wireshark** and run as root

Sniffer software like Wireshark puts the NIC in promiscuous mode so it will see all the packets on the line rather than just its own.

Viewing Network Packets
tcpdump

```
[root@elrond ~]# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535
bytes
08:48:35.555899 IP 172.30.1.125.ssh > 172.30.1.100.49326: Flags [P.],
seq 1215753462:1215753658, ack 2360465031, win 317, length 196
08:48:35.556202 IP 172.30.1.100.49326 > 172.30.1.125.ssh: Flags [.],
ack 196, win 254, length 0
08:48:35.557680 IP 172.30.1.125.48727 > cisvdc1.cisvlab.net.domain:
6647+ PTR? 100.1.30.172.in-addr.arpa. (43)
08:48:35.558483 IP cisvdc1.cisvlab.net.domain > 172.30.1.125.48727:
6647 NXDomain* 0/1/0 (130)
08:48:35.558704 ARP, Request who-has snickers.cisvlab.net
(00:13:20:c6:a4:16 (oui Unknown)) tell 172.30.1.100, length 46
08:48:35.558768 ARP, Reply snickers.cisvlab.net is-at
00:13:20:c6:a4:16 (oui Unknown), length 46
<continues like this>
```

Ctrl-s *to pause* ***Ctrl-q*** *to continue* ***Ctrl-c*** *to end*

To install tcpdump use: ***yum install tcpdump***

Viewing Network Packets
tcpdump

```
[root@elrond ~]# tcpdump -c5 arp or icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
08:55:58.135729 IP snickers.cisvlab.net > 172.30.1.100: ICMP echo request, id
1280, seq 13402, length 80
08:55:58.135742 IP 172.30.1.100 > snickers.cisvlab.net: ICMP echo reply, id
1280, seq 13402, length 80
08:55:58.139540 ARP, Request who-has 172.30.1.1 tell 172.30.1.125, length 28
08:55:58.140088 ARP, Reply 172.30.1.1 is-at c8:9c:1d:4f:77:01 (oui Unknown),
length 46
08:55:58.359346 IP snickers.cisvlab.net > 172.30.1.100: ICMP echo request, id
1280, seq 13658, length 80
5 packets captured
8 packets received by filter
0 packets dropped by kernel
[root@elrond ~]#
```

Using the –c option to limit the capture to 5 packets and filter out anything but arp or icmp packets

Viewing Network Packets
tcpdump on Elrond

```
[root@elrond ~]# tcpdump -c5 arp or icmp and host 172.30.1.125
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
08:59:12.957730 ARP, Request who-has 172.30.1.125 tell 172.30.1.150, length 46
08:59:12.958153 ARP, Reply 172.30.1.125 is-at 00:0c:29:d8:84:7f (oui Unknown),
length 28
08:59:12.958444 IP 172.30.1.150 > 172.30.1.125: ICMP echo request, id 2428,
seq 1, length 64
08:59:12.958612 IP 172.30.1.125 > 172.30.1.150: ICMP echo reply, id 2428, seq
1, length 64
08:59:13.940973 IP 172.30.1.150 > 172.30.1.125: ICMP echo request, id 2428,
seq 2, length 64
5 packets captured
13 packets received by filter
0 packets dropped by kernel
[root@elrond ~]#
```

Using the –c option to limit the capture to 5 packets and filter out anything but arp or icmp packets for host 172.30.1.125

Viewing Network Packets
tcpdump

```
[root@elrond ~]# tcpdump -c5 arp or icmp and host 172.30.1.125 > capture
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
5 packets captured
6 packets received by filter
0 packets dropped by kernel

[root@elrond ~]# cat capture
09:01:01.943495 IP 172.30.1.150 > 172.30.1.125: ICMP echo request, id 2428,
seq 110, length 64
09:01:01.943564 IP 172.30.1.125 > 172.30.1.150: ICMP echo reply, id 2428, seq
110, length 64
09:01:02.943255 IP 172.30.1.150 > 172.30.1.125: ICMP echo request, id 2428,
seq 111, length 64
09:01:02.943332 IP 172.30.1.125 > 172.30.1.150: ICMP echo reply, id 2428, seq
111, length 64
09:01:03.943654 IP 172.30.1.150 > 172.30.1.125: ICMP echo request, id 2428,
seq 112, length 64
[root@elrond ~]#
```

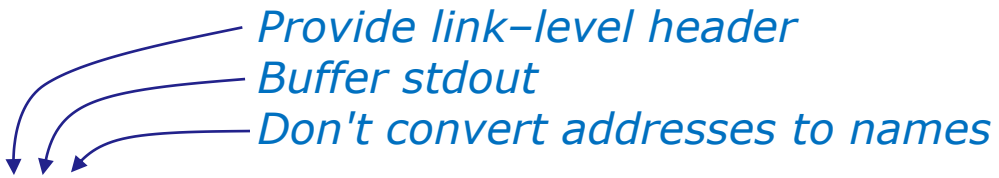
Same as before but saving the captured packets in a file

Viewing Network Packets
tcpdump

```
[root@elrond ~]# tcpdump src 172.30.1.150 or dst 172.30.1.150
tcpdump: verbose output suppressed, use -v or -vv for full protocol
  decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535
  bytes
09:05:35.763345 IP 172.30.1.150 > 172.30.1.125: ICMP echo request, id
  2469, seq 93, length 64
09:05:35.763413 IP 172.30.1.125 > 172.30.1.150: ICMP echo reply, id
  2469, seq 93, length 64
09:05:35.767609 IP 172.30.1.150.ssh > 172.30.1.100.49329: Flags [P.],
  seq 3250995165:3250995265, ack 256292814, win 591, length 100
09:05:35.972475 IP 172.30.1.100.49329 > 172.30.1.150.ssh: Flags [.],
  ack 100, win 255, length 0
^C
8 packets captured
9 packets received by filter
0 packets dropped by kernel
[root@elrond ~]#
```

View all packets coming or going from 172.30.4.125

Viewing Network Packets
tcpdump




```
[root@elrond ~]# tcpdump -eln src 172.30.1.105 or dst 172.30.1.105
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
11:23:35.938846 00:0c:29:a4:83:bc > 00:19:b9:03:70:b3, ethertype IPv4 (0x0800),
  length 98: 172.30.1.126 > 172.30.1.105: ICMP echo request, id 54547, seq 1,
  length 64
11:23:35.939741 00:19:b9:03:70:b3 > Broadcast, ethertype ARP (0x0806), length 60:
  arp who-has 172.30.1.126 tell 172.30.1.105
11:23:35.939769 00:0c:29:a4:83:bc > 00:19:b9:03:70:b3, ethertype ARP (0x0806),
  length 42: arp reply 172.30.1.126 is-at 00:0c:29:a4:83:bc
11:23:35.940051 00:19:b9:03:70:b3 > 00:0c:29:a4:83:bc, ethertype IPv4 (0x0800),
  length 98: 172.30.1.105 > 172.30.1.126: ICMP echo reply, id 54547, seq 1,
  length 64
```

Ctrl-C to end

```
4 packets captured
12 packets received by filter
0 packets dropped by kernel
[root@elrond ~]#
```

Show all packets with a source and destination IP address of 172.30.1.105

Class Activity
tcpdump

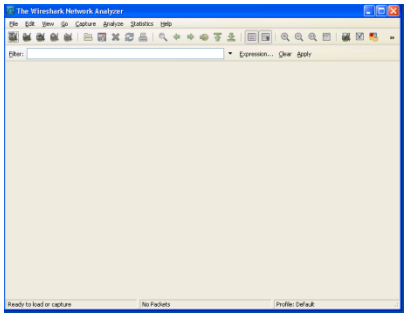
Try it!

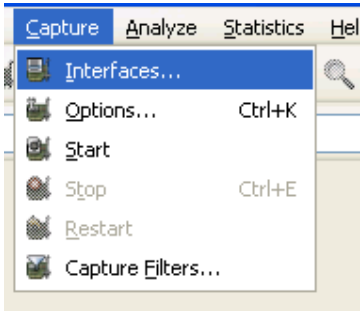
- [Elrond] **yum install tcpdump**
- [Elrond] **tcpdump**
- [Elrond] **tcpdump –c10 icmp or arp**

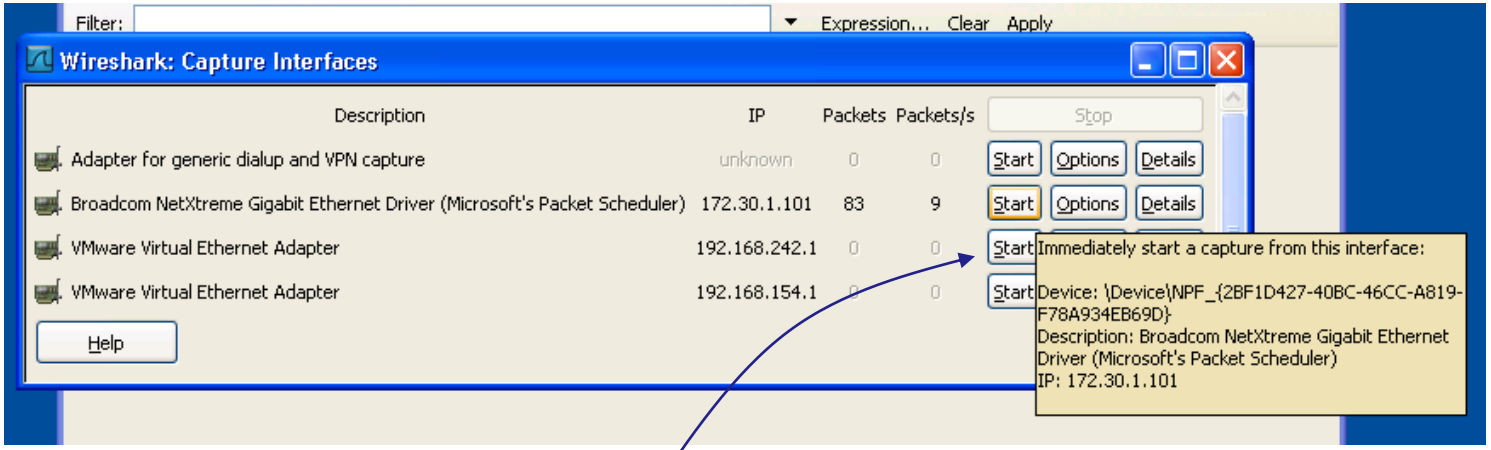
Ctrl-s to pause **Ctrl-q** to continue **Ctrl-c** to end

Viewing Network Packets
Wireshark on VMware station

Wireshark

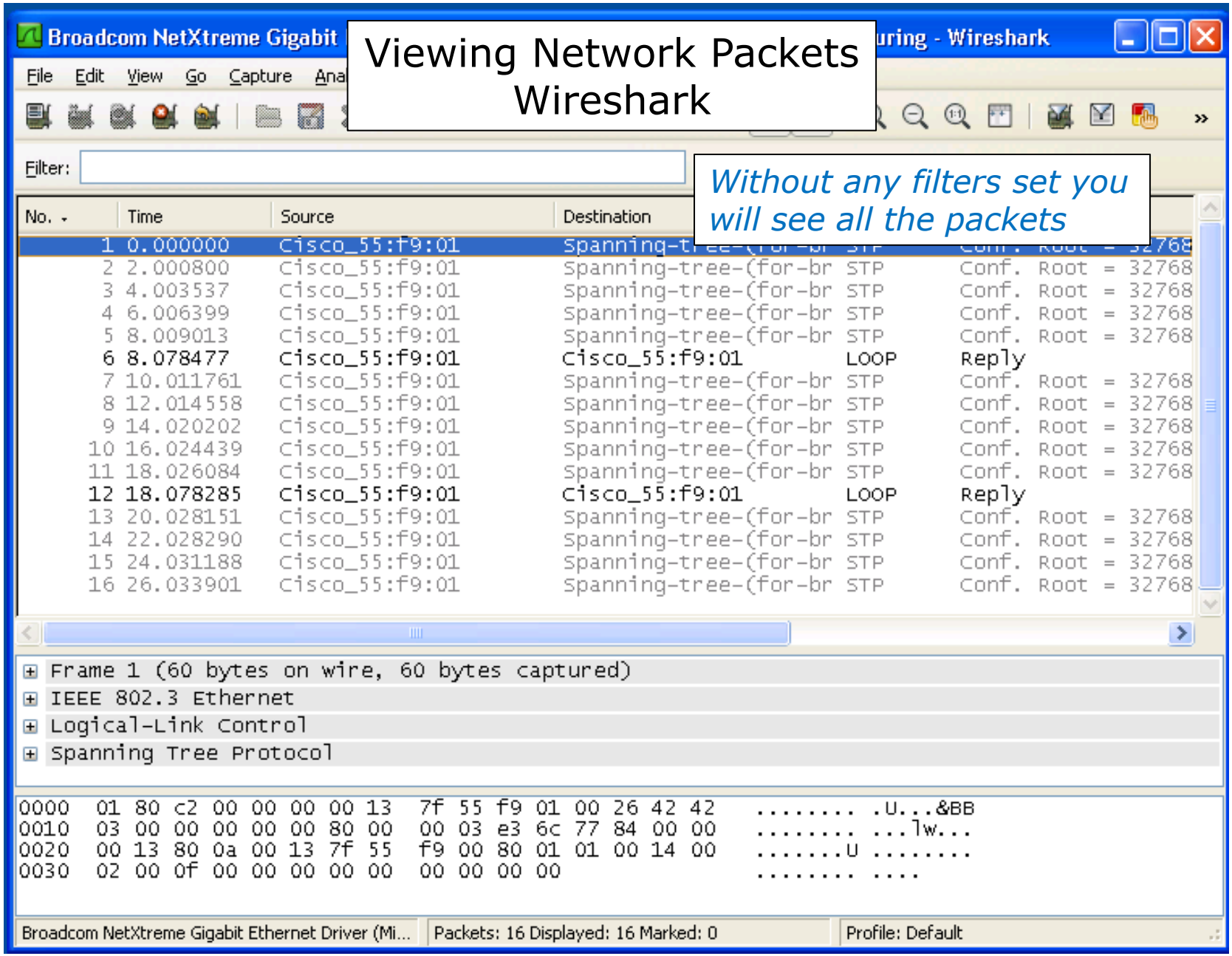






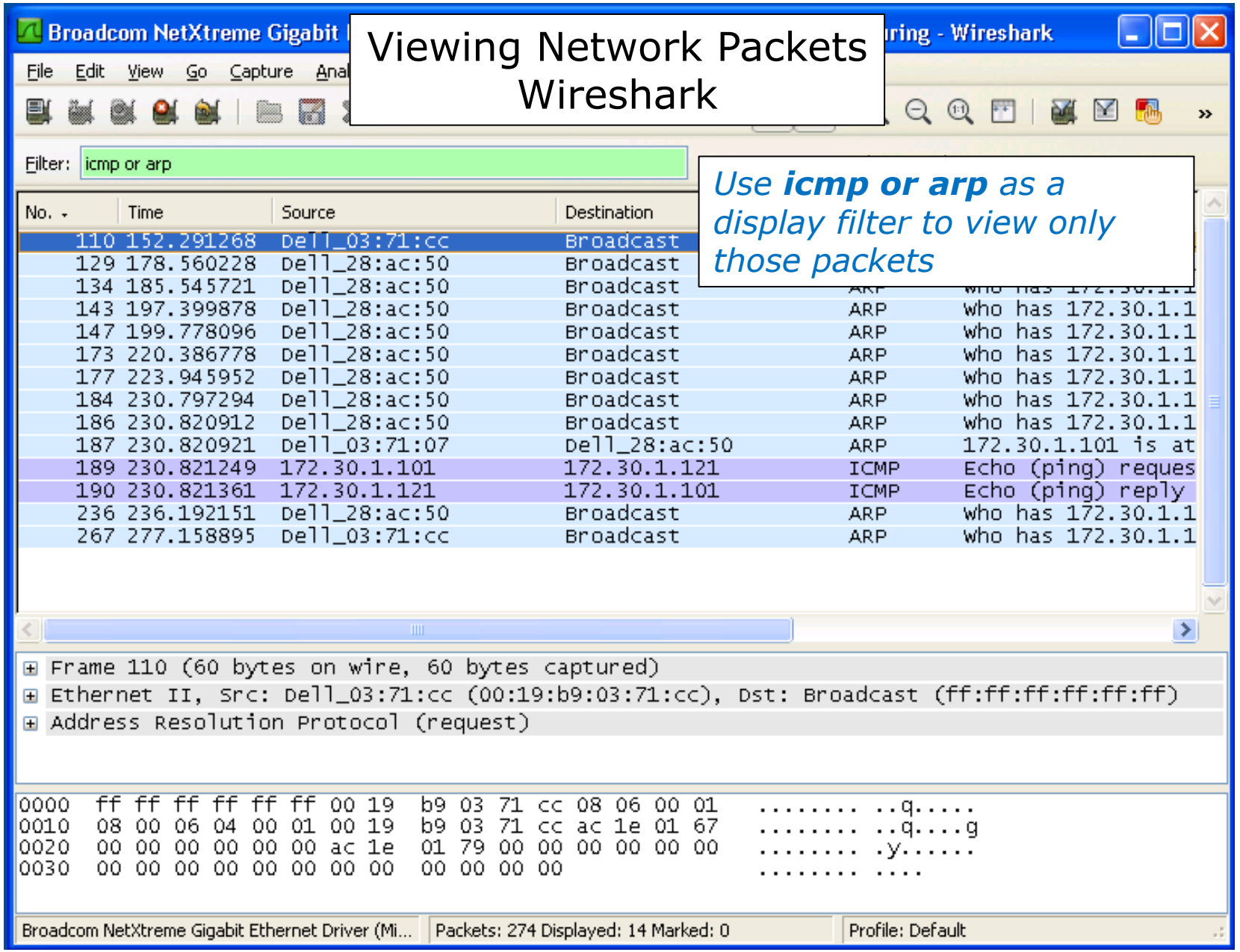
Click on the Start button for the Broadcom NIC interface

Viewing Network Packets Wireshark



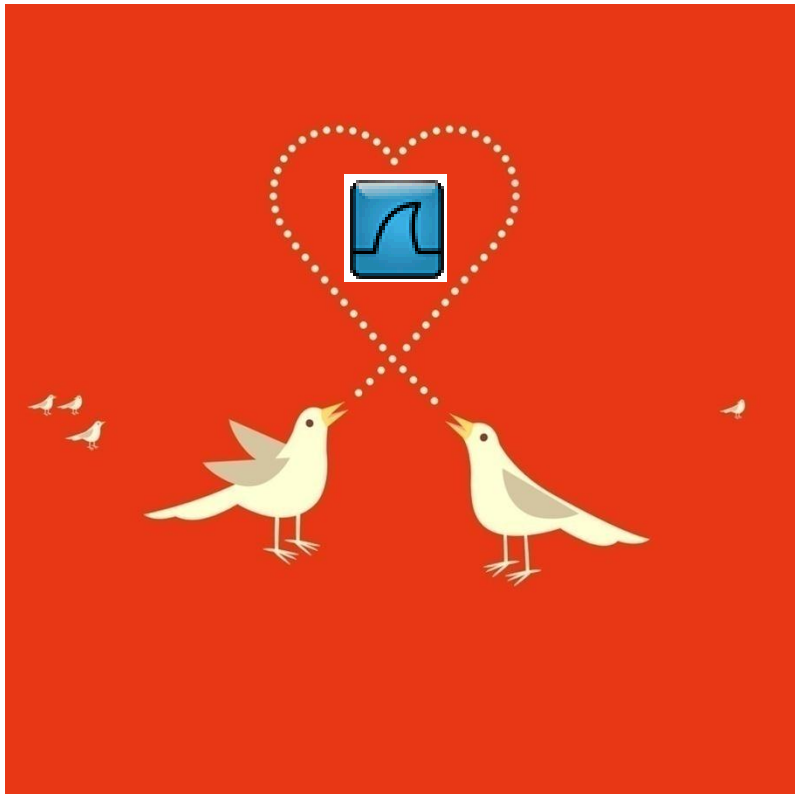
Without any filters set you will see all the packets

Viewing Network Packets Wireshark



*Use **icmp or arp** as a display filter to view only those packets*

Viewing Network Packets
Wireshark

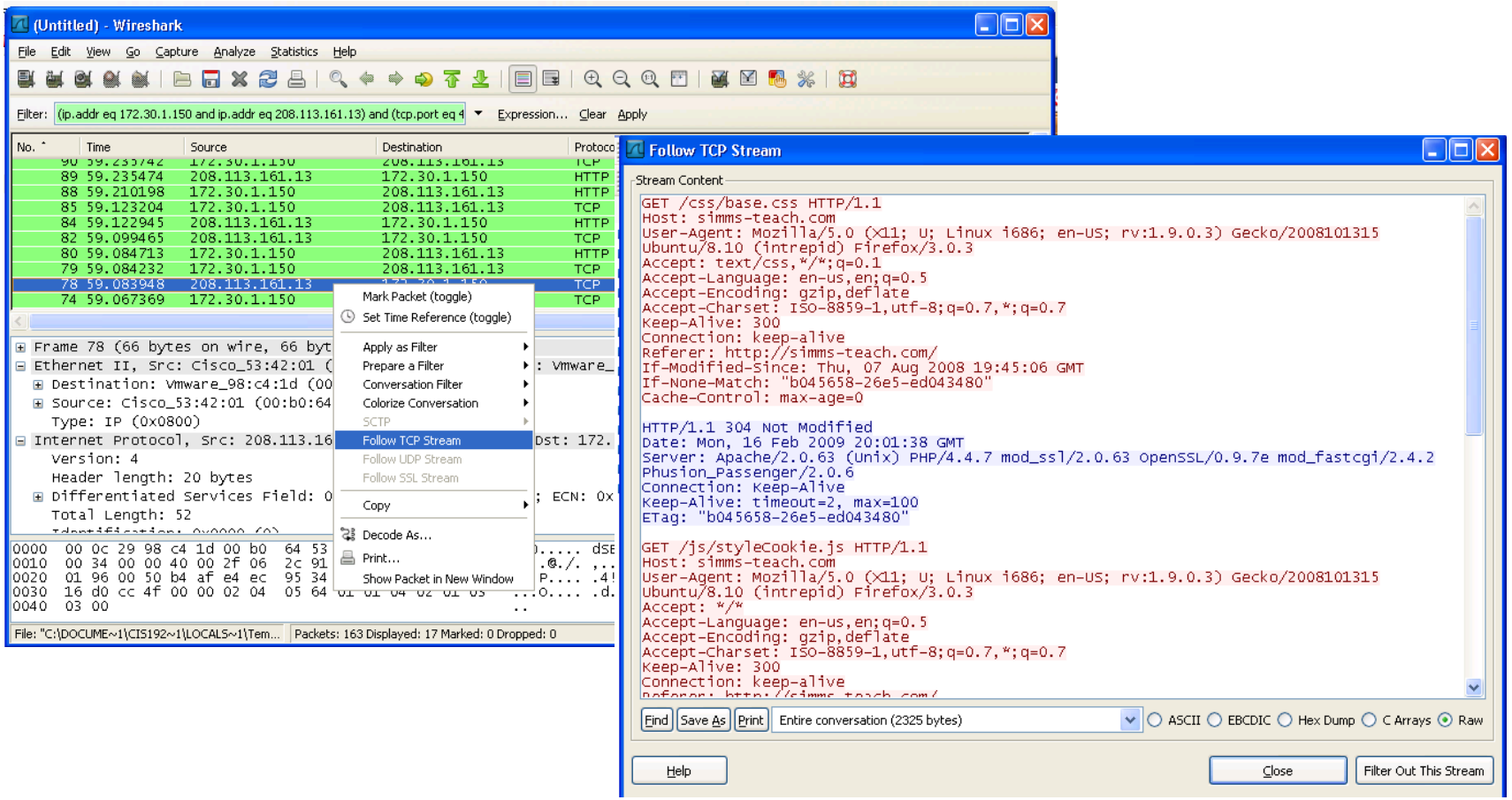


Some really nice options:

- Follow TCP stream
- Prepare a filter

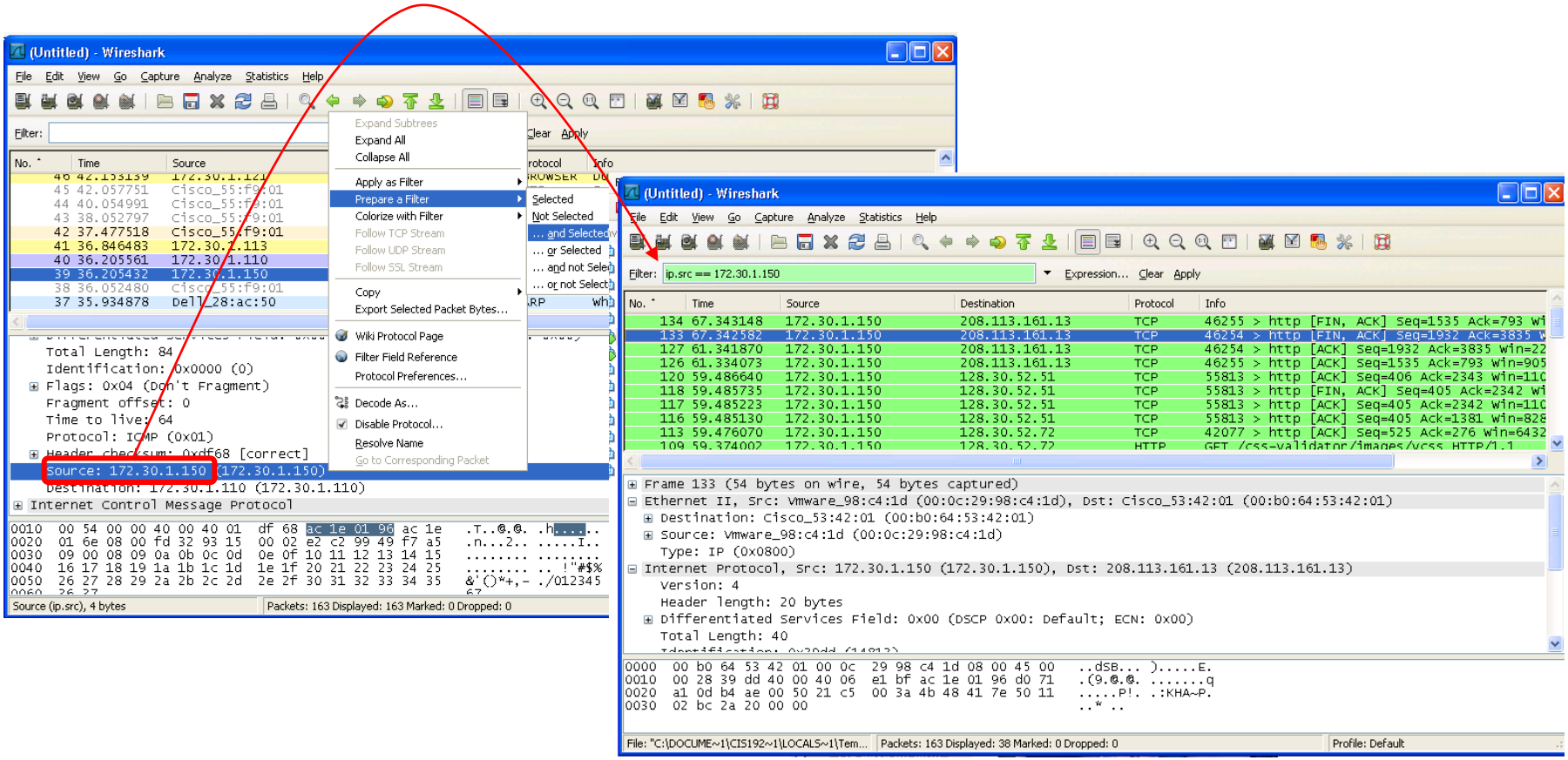
Use icmp or arp as a display filter to view only those packets

Viewing Network Packets
Wireshark – Follow TCP Stream



Following the TCP stream of viewing a web page

Viewing Network Packets
Wireshark – Prepare a filter



Select the source IP address of a packet and used it to make a display filter to only see packets from that IP address

Viewing Network Packets
Wireshark – example filters

- arp *will only show ARP packets*
- arp || icmp *will only show ARP and ICMP packets*
- http *will only show HTTP packets*
- bootp *will only show bootp and DHCP packets*
- (ip.src == 172.30.1.107 || ip.dst == 172.30.1.107) *will only show packets going to or from 172.30.1.107*
- icmp && (ip.src == 172.30.1.107 || ip.dst == 172.30.1.107) *will only show ARP packets going to or from 172.30.1.107*
- !ssh *will hide any SSH packets*
- ip.src == 172.30.1.0/24 *will only show packets with a source IP address in the 172.30.1.0/24 subnet*
- ip.host == 172.30.1.125

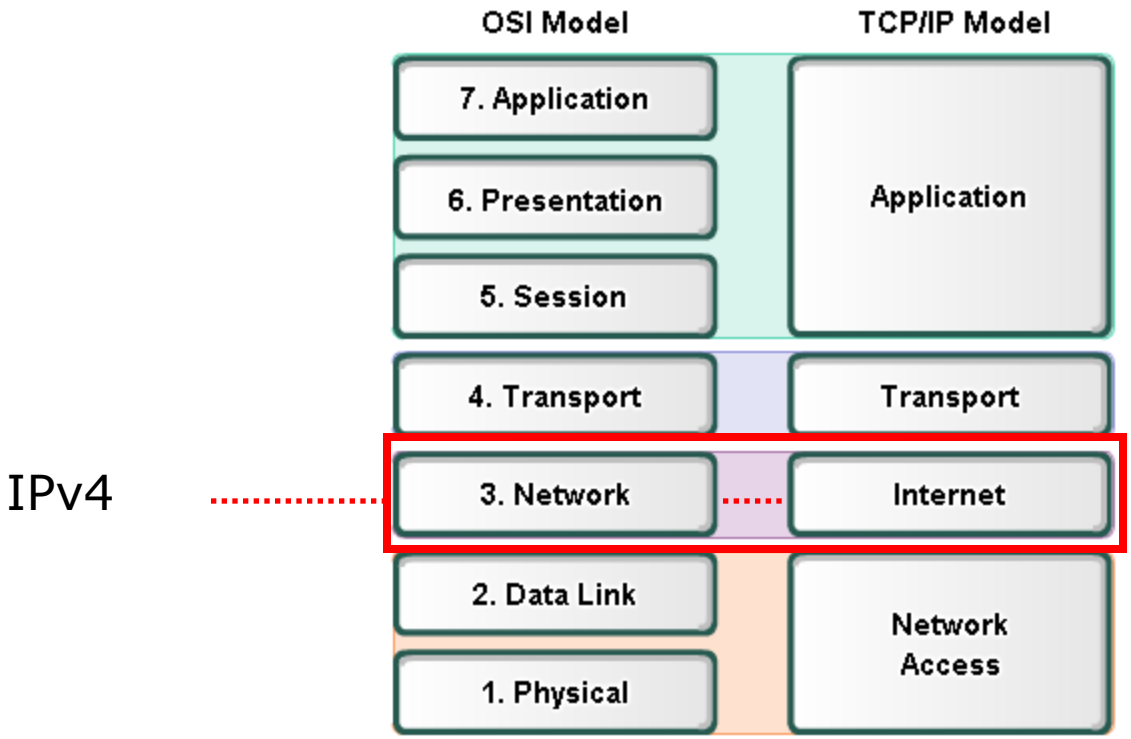
Class Activity
Wireshark

Try it!

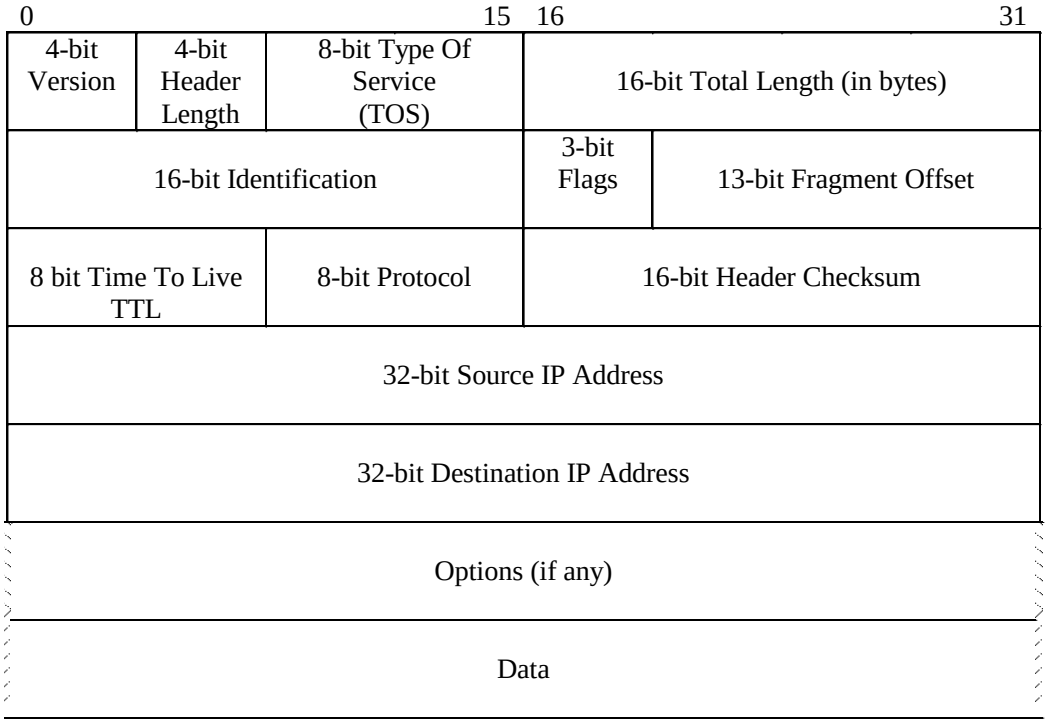
- [Classroom PC or William] run Wireshark
- [Classroom PC or William] "ip or arp" filter
- [Classroom PC or William] "ip or arp and ip.host == 172.30.1.125" filter

Layer 3

Network Layer

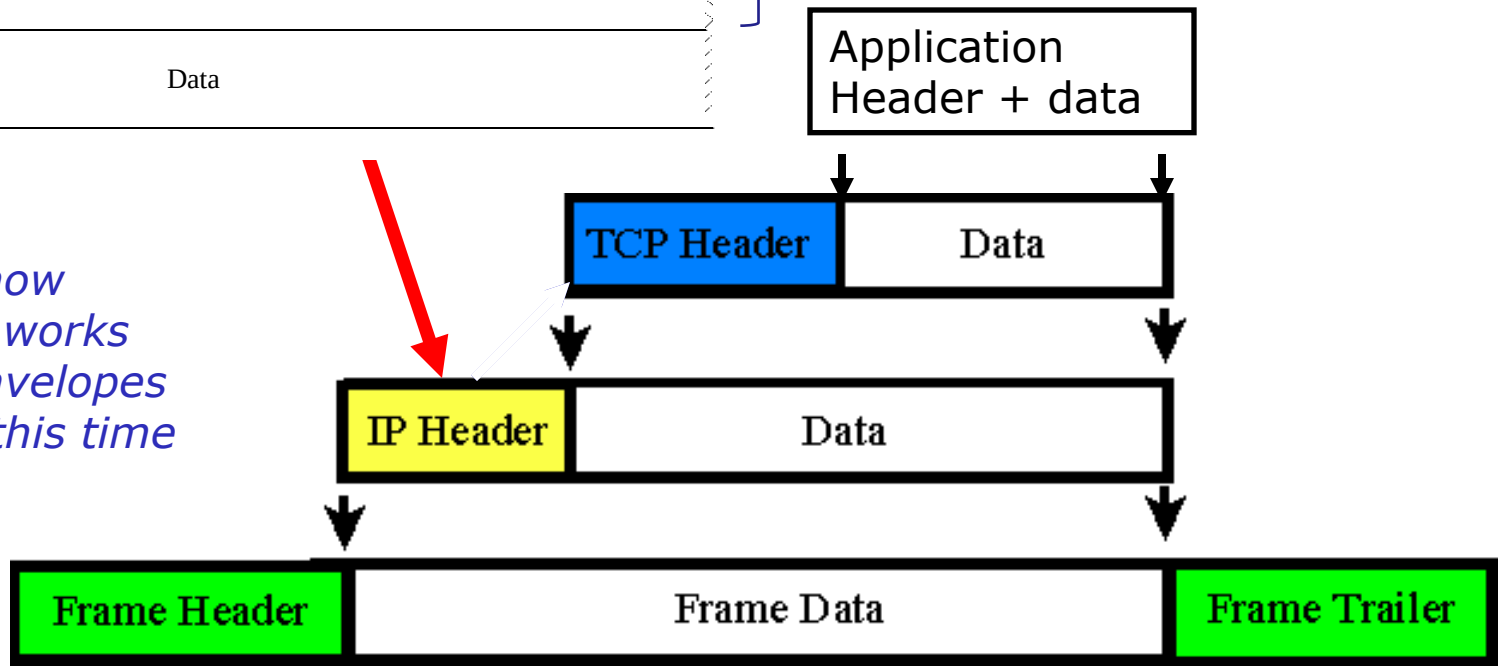


RS: More on Layer 3 tonight

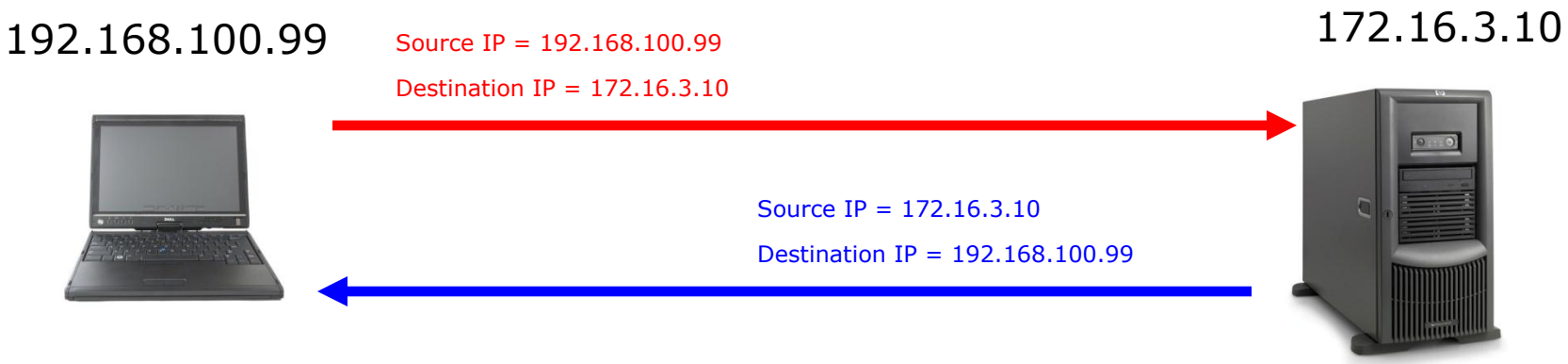


IP Header

RS: showing how encapsulation works without the envelopes and postman this time



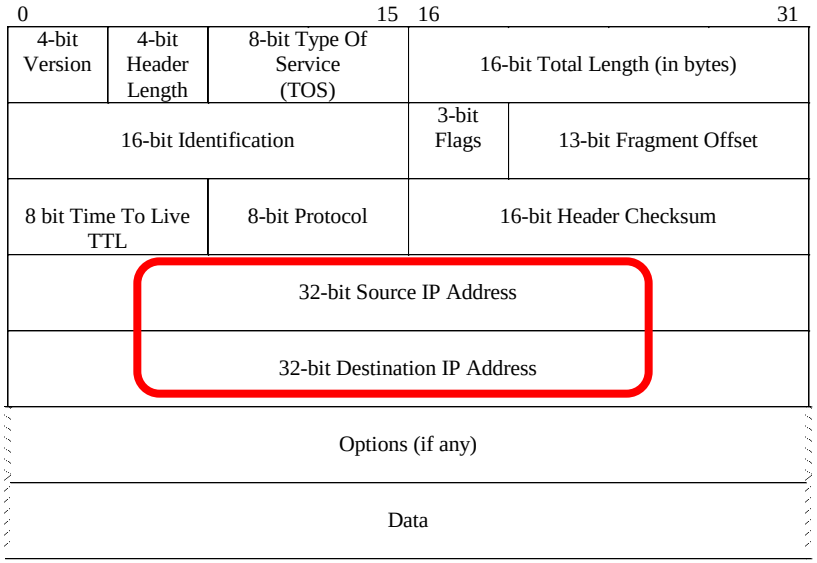
Addressing



- Source IP Address
- Destination IP Address

- More later!

RS: Layer 3 is where IP addresses are used. They are put in the header of the layer three packets.



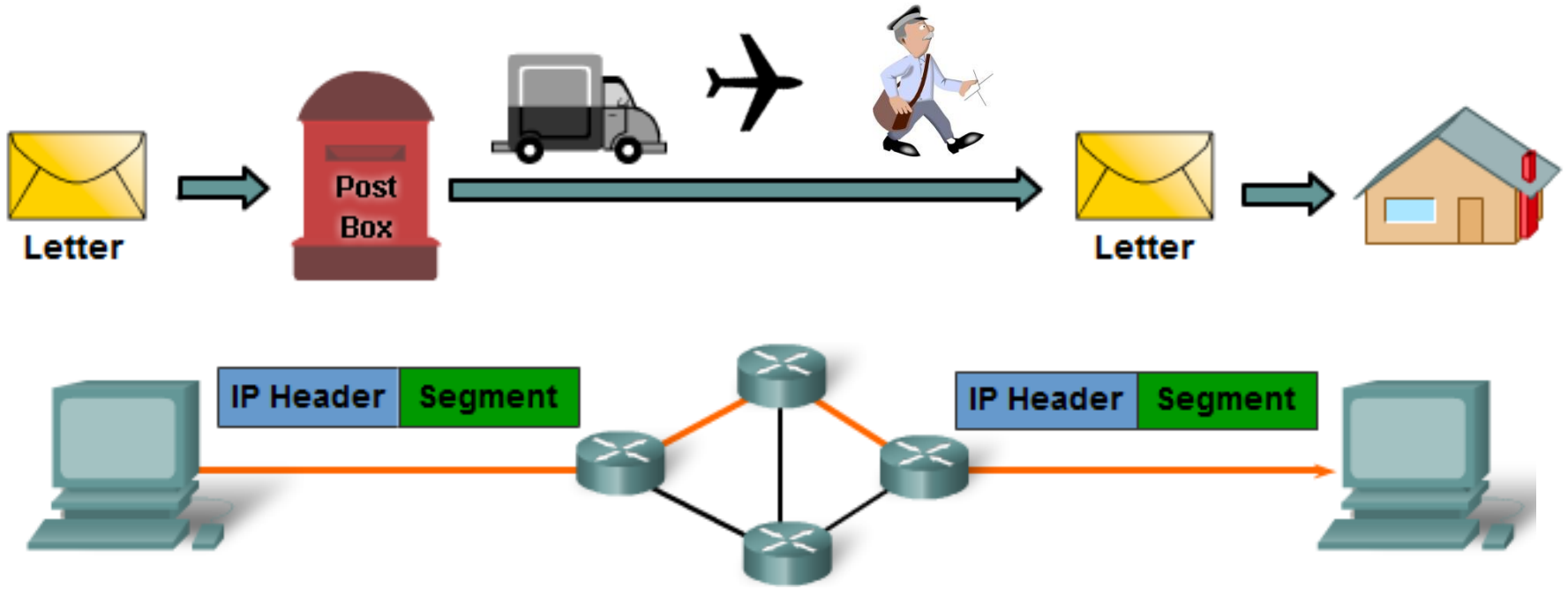
Network Layer Protocols

- Internet Protocol version 4 (IPv4)
- Internet Protocol version 6 (IPv6)
- Novell Internetwork Packet Exchange (IPX)
- AppleTalk
- Connectionless Network Service (CLNS/DECNet)

- The Internet Protocol (IPv4 and IPv6) is the most widely-used Layer 3 data carrying protocol and will be the focus of this course.

same goes for CIS 192!

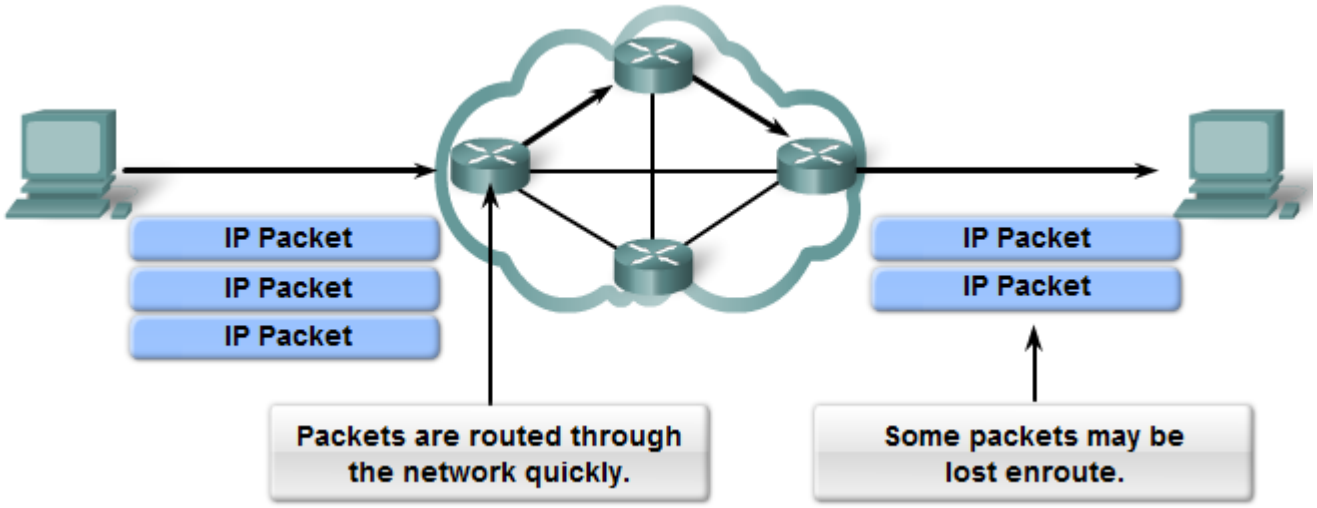
Connectionless



IP packets are sent without notifying the end host that they are coming. *(Layer 3)*

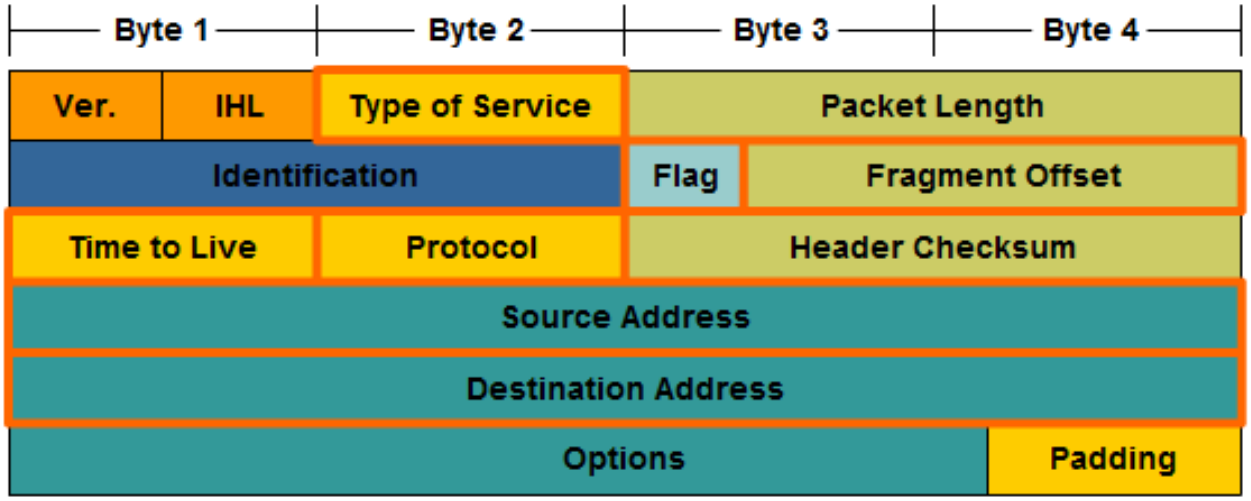
- **TCP**: A connection-oriented protocol does requires a connection to be established prior to sending TCP segments. *(Layer 4)*
- **UDP**: A connectionless protocol does not require a session to be established. *(Layer 4)*

Best Effort Service (unreliable)



- The mission of Layer 3 is to transport the packets between the hosts while placing as little burden on the network as possible.
  - Speed over reliability
- Layer 3 is not concerned with or even aware of the type of data contained inside of a packet.
  - This responsibility is the role of the upper layers as required.
- **Unreliable**: IP does not have the capability or responsibility to manage, and recover from, undelivered or corrupt packets.
  - TCP's responsibility at the end-to-end hosts

IP Header

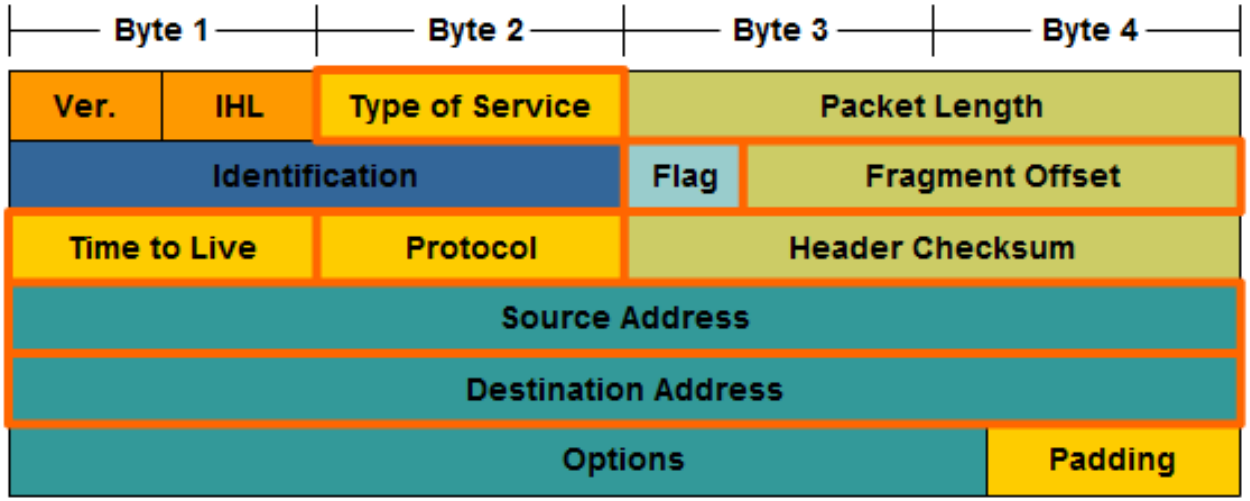


- **IP Destination Address**
  - 32-bit binary value that represents the packet destination Network layer host address.
- **IP Source Address**
  - 32-bit binary value that represents the packet source Network layer host address.

RS: IPv4 uses 32 bit addresses and there is always a source and destination address

IP's TTL – Time To Live field

Decrement by 1, if 0 drop the packet.

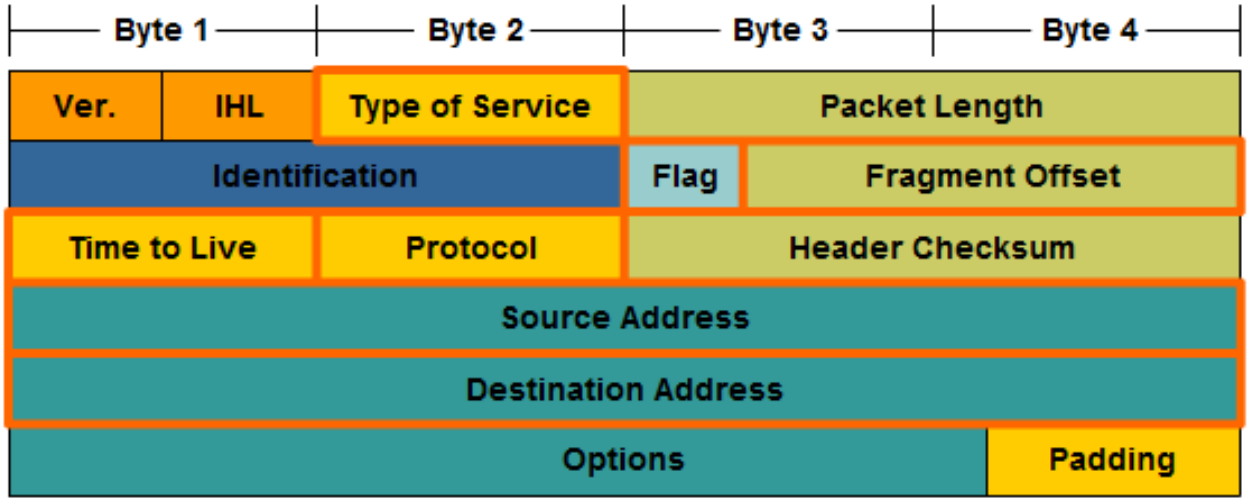


- If the router decrements the TTL field to 0, it will then drop the packet (unless the packet is destined specifically for the router, i.e. ping, telnet, etc.).
- Common operating system TTL values are:
  - UNIX: **255**
  - Linux: **64 or 255** depending upon vendor and version
  - Microsoft Windows 95: **32**
  - Other Microsoft Windows operating systems: **128**

RS: TTL keeps packets from endlessly wandering about the Internet forever. It is also used by traceroute and mtr commands

IP's TTL – Time To Live field

Decrement by 1, if 0 drop the packet.

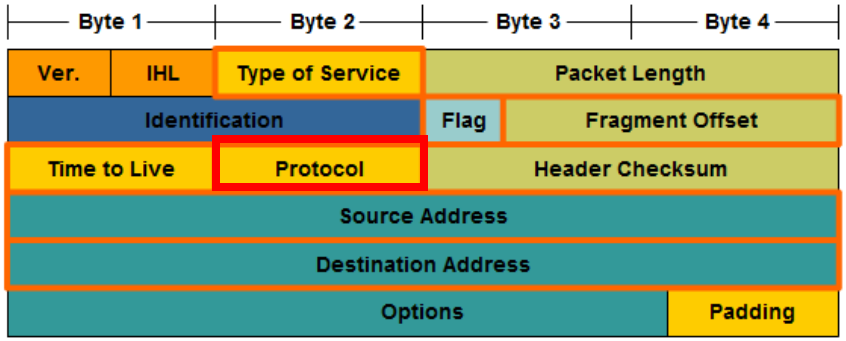


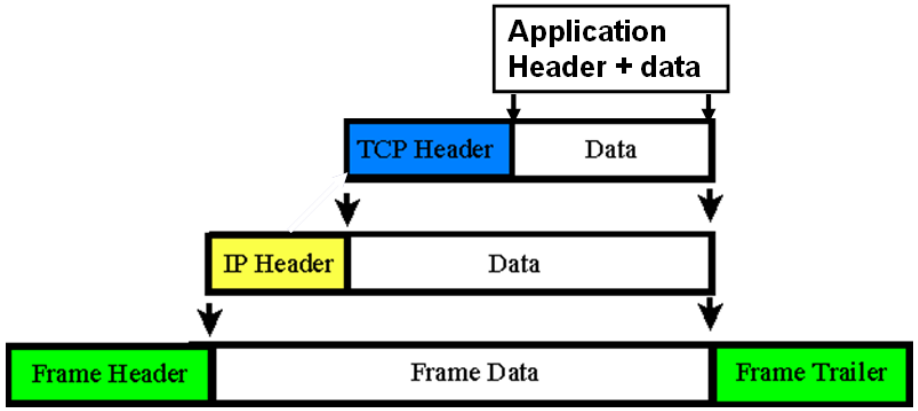
- The idea behind the TTL field is that IP packets can not travel around the Internet forever, from router to router.

- Eventually, the packet's TTL which reach 0 and be dropped by the router, even if there is a routing loop somewhere in the network.

RS: TTL errors are used by traceroute and mtr to discover the path a packet takes

IP's Protocol Field

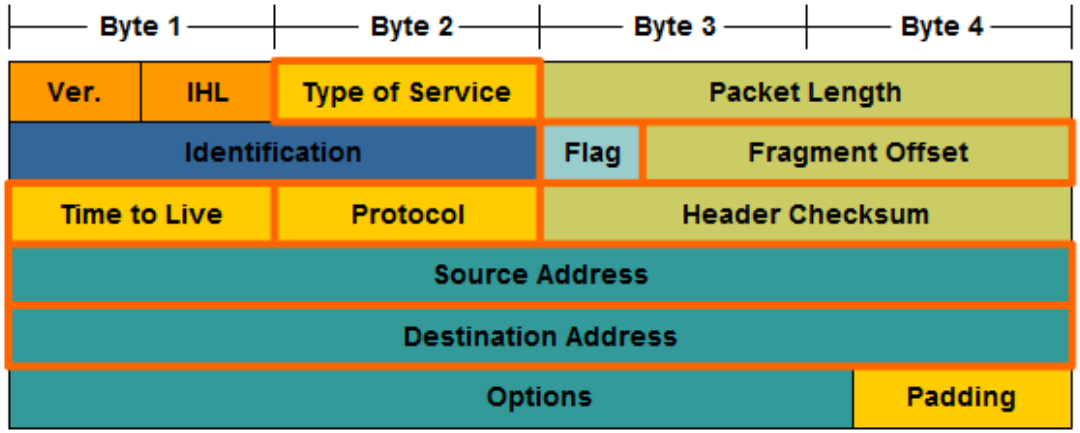




- **Protocol field** enables the Network layer to pass the data to the appropriate upper-layer protocol.
- Example values are:
  - 01 ICMP
  - 06 TCP
  - 17 UDP

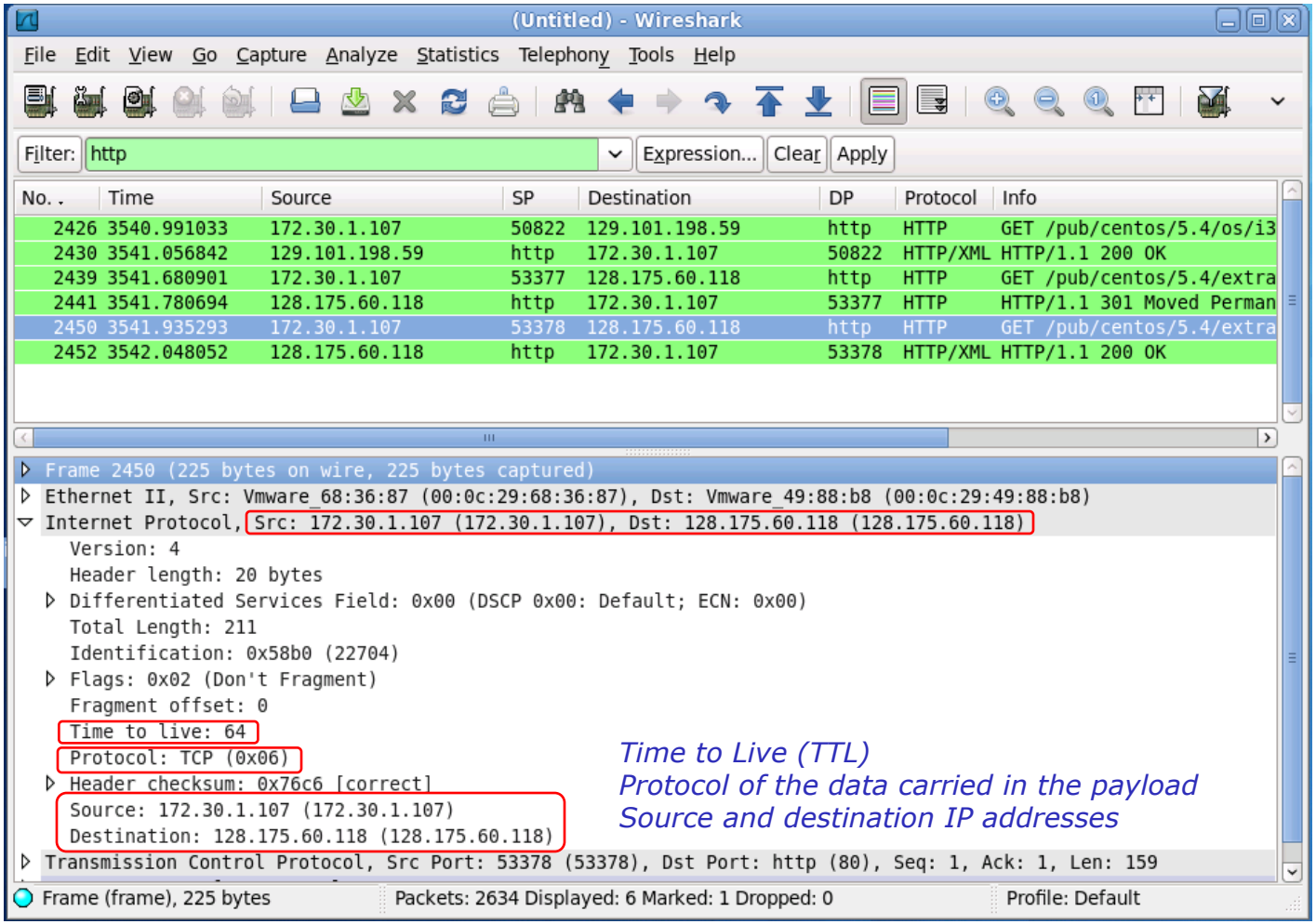
RS: The protocol is used to identify the format of the data payload

Other IPv4 fields



- **Version** - Contains the IP version number (4)
- **Header Length (IHL)** - Specifies the size of the packet header.
- Packet Length - This field gives the entire packet size, including header and data, in bytes.
- **Identification** - This field is primarily used for uniquely identifying fragments of an original IP packet
- **Header Checksum** - The checksum field is used for error checking the packet header.
- **Options** - There is provision for additional fields in the IPv4 header to provide other services but these are rarely used.

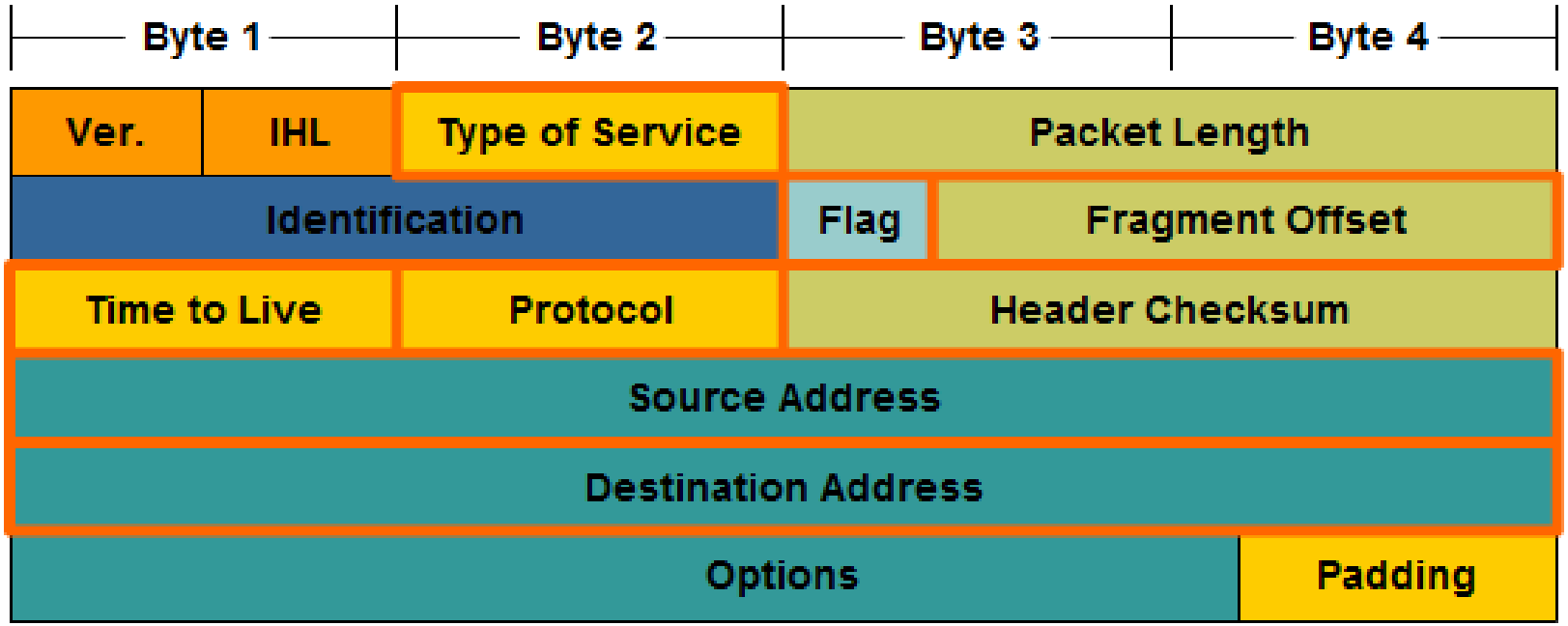
Viewing Layer 3 IP Packets with Wireshark



Frodo is browsing google.com

IPv4 addressing & subnetting

IPv4 Addresses



- IPv4 addresses are 32 bit addresses

RS: In this section we are going to take a deep dive into the IP addresses

IPv4 Addresses

- IPv4 Addresses are 32 bit addresses:

10101001110001110100011011000100

10101001 11000111 01000101 10001001

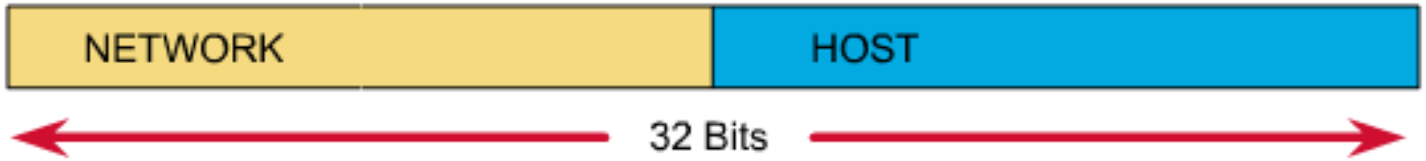
- We use dotted notation (or dotted decimal notation) to represent the value of each byte (octet) of the IP address in decimal.

10101001 11000111 01000101 10001001
169 . 199 . 69 . 137

IPv4 Addresses

An IP address has two parts:

- **network number**
- **host number**



Which bits refer to the network number?

Which bits refer to the host number?

IPv4 Addresses

Answer:

- Newer technology - **Classless IP Addressing**
  - The **subnet mask** determines the network portion and the host portion.
  - Value of first octet does NOT matter (older classful IP addressing)
  - Hosts and Classless Inter-Domain Routing (**CIDR**).
  - Classless IP Addressing is what is used within the Internet and in most internal networks.

- Older technology - **Classful IP Addressing**
  - **Value of first octet** determines the network portion and the host portion.
  - Used with classful routing protocols like RIPv1.
  - The Cisco IP Routing Table is structured in a classful manner (CIS 82)

RS: We will be using Classless IP Addressing in CIS 192 which means we will always be specifying network masks on interfaces and genmasks in routing tables

Network Addresses have all 0's in the host portion.

Types of Addresses

Network Address

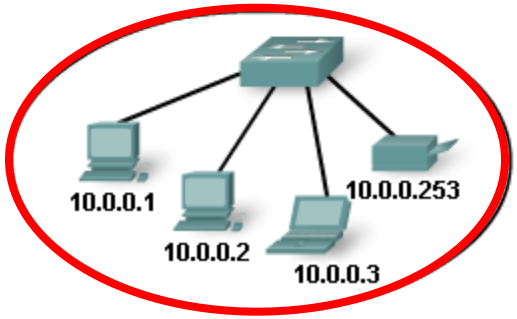
Broadcast Address

Host Address

Roll over to learn more.

	Network		Host
10	0	0	0
00001010	00000000	00000000	00000000
10	0	0	255
00001010	00000000	00000000	11111111
10	0	0	1
00001010	00000000	00000000	00000001

Subnet Mask: 255.255.255.0



- **Network address** - The address by which we refer to the network
- **Broadcast address** - A special address used to send data to all hosts in the network
- **Host addresses** - The addresses assigned to the end devices in the network

RS: Networks can be subnetted into smaller networks. The first address of the block is the network address (host portion is all zeros)

Types of Addresses

Network Address

Broadcast Address

Host Address

Roll over to learn more.

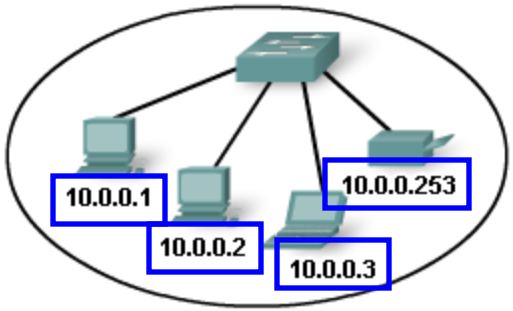
	Network		Host
10	0	0	0
00001010	00000000	00000000	00000000

Broadcast Addresses have all 1's in the host portion.

10	0	0	255
00001010	00000000	00000000	11111111

10	0	0	1
00001010	00000000	00000000	00000001

Subnet Mask: 255.255.255.0



- **Network address** - The address by which we refer to the network
- **Broadcast address** - A special address used to send data to all hosts in the network
- **Host addresses** - The addresses assigned to the end devices in the network

RS: Networks can be subnetted into smaller networks. The last address of the block is the broadcast address (host portion is all 1's)

Types of Addresses

Network Address

Broadcast Address

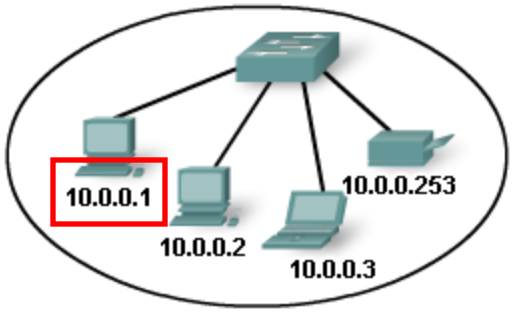
Host Address

Roll over to learn more.

	Network		Host
10	0	0	0
00001010	00000000	00000000	00000000
10	0	0	255
00001010	00000000	00000000	11111111
10	0	0	1
00001010	00000000	00000000	00000001

Host Addresses can not have all 0's or all 1's in the host portion.

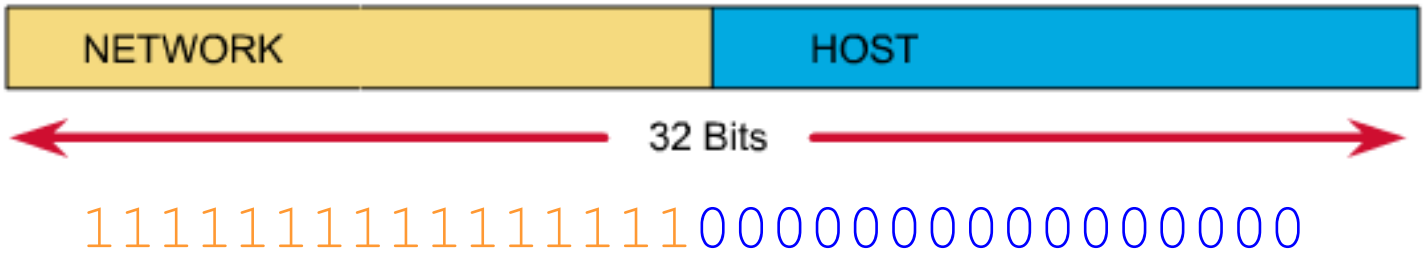
Subnet Mask: 255.255.255.0



- **Network address** - The address by which we refer to the network
- **Broadcast address** - A special address used to send data to all hosts in the network
- **Host addresses** - The addresses assigned to the end devices in the network

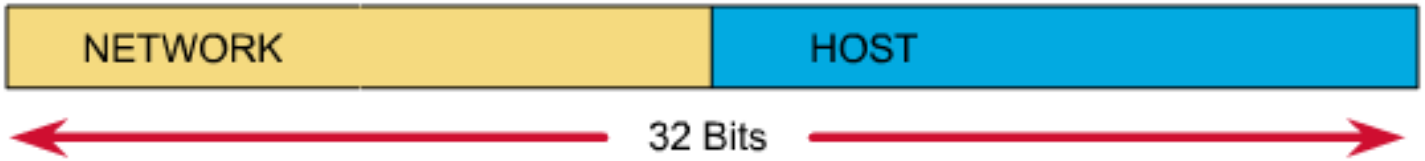
RS: Networks can be subnetted into smaller networks. The addresses between the network address and the broadcast address are for hosts.

Dividing the Network and Host Portions



- **Subnet Mask**
  - Used to define the:
    - Network portion
    - Host portion
  - 32 bits
  - Contiguous set of 1's followed by a contiguous set of 0's
    - 1's: Network portion
    - 0's: Host portion

Dividing the Network and Host Portions



11111111.11111111.00000000.00000000

Dotted decimal: 255 . 255 . 0 . 0

Slash notation: /16

- Subnet mask expressed as:
  - Dotted decimal
    - Ex: 255.255.0.0
  - Slash notation or prefix length
    - /16 (the number of one bits)

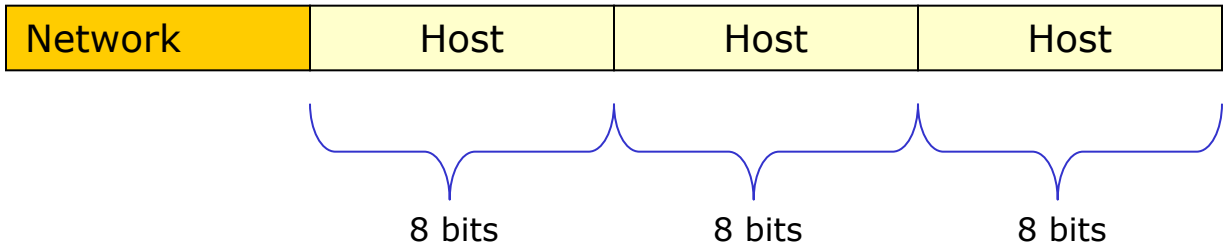
RS: We will use both dotted and slash notations in CIS 192

Why the mask matters: Number of hosts!

Subnet Mask:	1st octet	2nd octet	3rd octet	4th octet
255.0.0.0 or /8	Network	Host	Host	Host
255.255.0.0 or /16	Network	Network	Host	Host
255.255.255.0 or /24	Network	Network	Network	Host

- The more host bits in the subnet mask means the more hosts in the network.
- Subnet masks do not have to end on "natural octet boundaries"

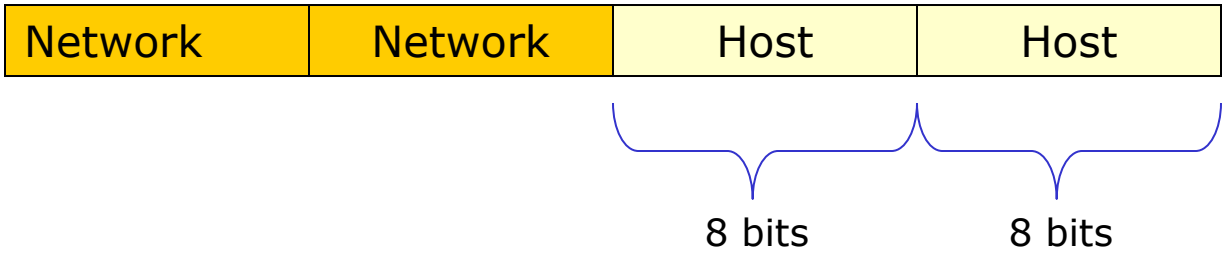
Subnet: 255.0.0.0 (/8)



With 24 bits available for hosts, there a 2^{24} possible addresses. That's 16,777,216 nodes!

- Only large organizations such as the military, government agencies, universities, and large corporations have networks with these many addresses.
- Example: A certain cable modem ISP has 24.0.0.0 and a DSL ISP has 63.0.0.0

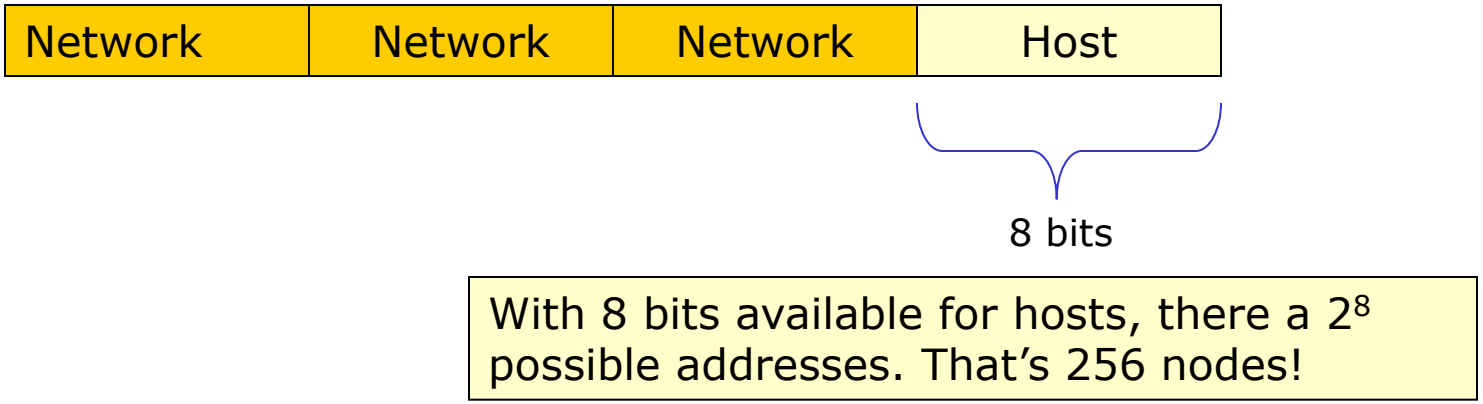
Subnet: 255.255.0.0 (/16)



With 16 bits available for hosts, there a 2^{16} possible addresses. That's 65,536 nodes!

- 65,534 host addresses, one for network address and one for broadcast address.

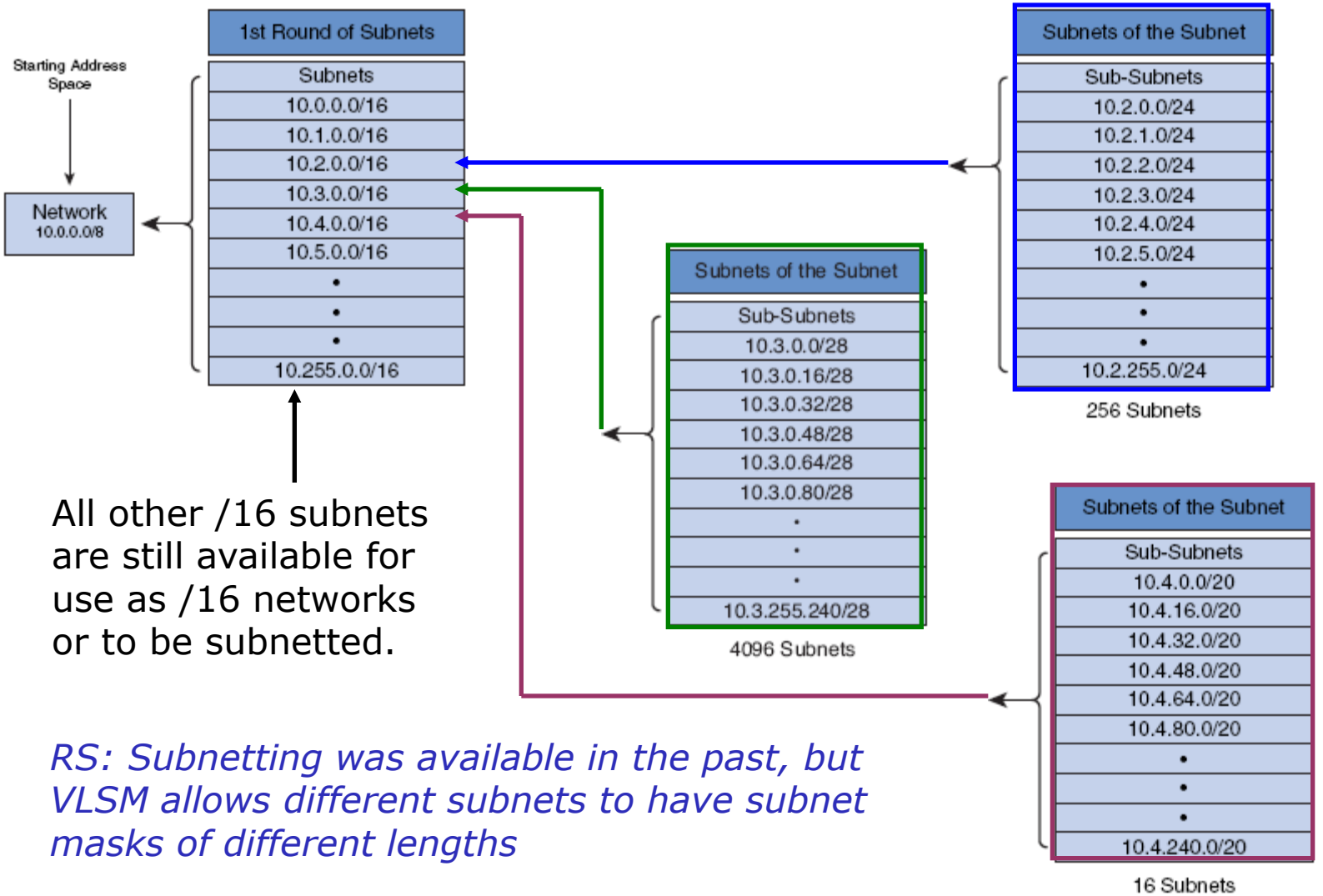
Subnet: 255.255.255.0 (/24)



- 254 host addresses, one for network address and one for broadcast address.

RS: We are using a /24 network in room 2501. That gives us 2^8 -2 (256 -2 = 254) host addresses. We drop by 2 because the first address (172.30.1.0) is the network address and the last address (172.30.1.255) is the broadcast address.

VLSM – Variable Length Subnet Masks
Subnet a subnet



All other /16 subnets are still available for use as /16 networks or to be subnetted.

RS: Subnetting was available in the past, but VLSM allows different subnets to have subnet masks of different lengths

Old Days: Classful IP Addressing

Class A	Network	Host		
Octet	1	2	3	4

Class B	Network		Host	
Octet	1	2	3	4

Class C	Network			Host
Octet	1	2	3	4

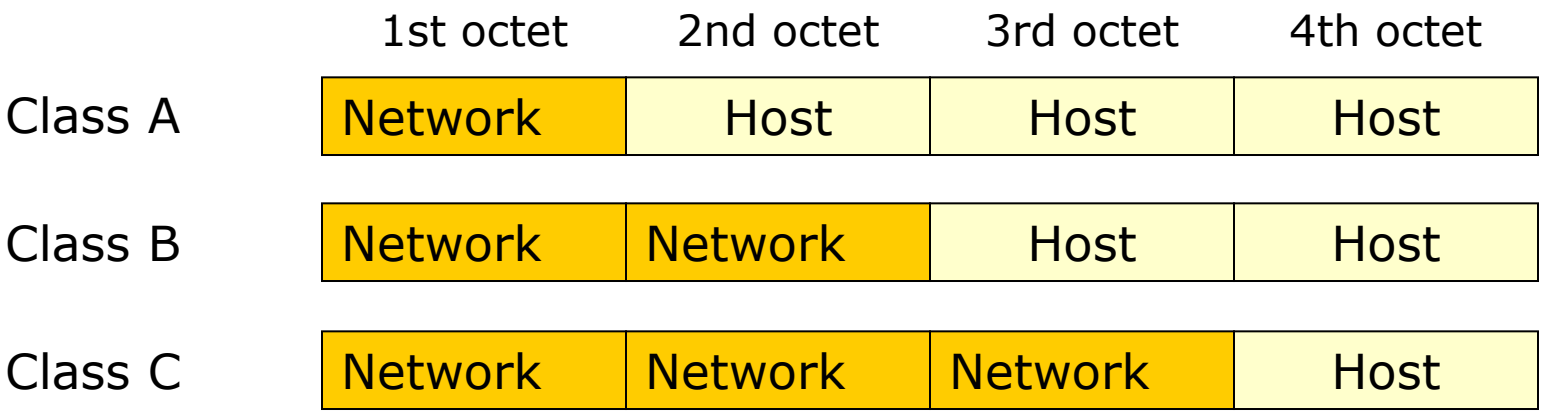
Class D	Host			
Octet	1	2	3	4

Address Class	First Octet Range	Number of Possible Networks	Number of Hosts per Network
Class A	0 to 127	128 (2 are reserved)	16,777,214
Class B	128 to 191	16,348	65,534
Class C	192 to 223	2,097,152	254

- In the early days of the Internet, IP addresses were allocated to organizations based on request rather than actual need.
- When an organization received an IP network address, that address was associated with a **"Class", A, B, or C.**
- This is known as **Classful IP Addressing**
- The **first octet** of the address determined what class the network belonged to and which bits were the network bits and which bits were the host bits.
- There were **no** subnet masks.
- It was not until 1992 when the IETF introduced CIDR (Classless Interdomain Routing), making the address class meaningless.
- This is known as **Classless IP Addressing**.

RS: We won't be using classful IP addressing in CIS 192

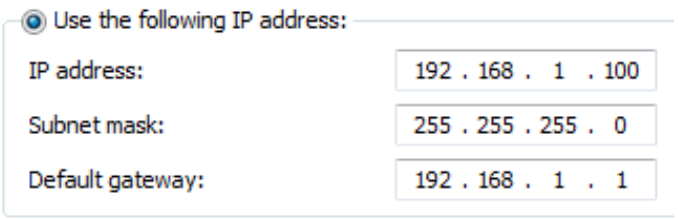
Old days: Address Classes



N = Network number assigned by ARIN (American Registry for Internet Numbers)
H = Host number assigned by administrator

RS: HP has the 15 and 16 networks (or they used to). They got the 15 net in the early days. After buying Compaq (which bought DEC) they had the 16 net as well!

Special Unicast IPv4 Addresses



- **Default Route**

- **Loopback Address**
  - Special address that hosts use to direct traffic to themselves.
  - 127.0.0.0 to 127.255.255.255

- **Link-Local Addresses (APIPA)**
  - 169.254.0.0 to 169.254.255.255 (169.254.0.0 /16)
  - Can be automatically assigned to the local host by the operating system in environments where no IP configuration is available.
  - Microsoft calls this APIPA (Automatic Private IP Addressing)

- **TEST-NET Addresses**
  - 192.0.2.0 to 192.0.2.255 (192.0.2.0 /24)
  - Set aside for teaching and learning purposes.
  - These addresses can be used in documentation and network examples.

subnetting by hand

```
0000 0001 = 1
0000 0010 = 2
0000 0100 = 4
0000 1000 = 8
0001 0000 = 16
0010 0000 = 32
0100 0000 = 64
1000 0000 = 128

1100 0000 = 192
1110 0000 = 224
1111 0000 = 240
1111 1000 = 248
1111 1100 = 252
1111 1110 = 254
1111 1111 = 255
```

When subnetting by hand I like to make these two tables first

subnetting using the ipcalc command

```
[root@elrond ~]# ipcalc -n 192.168.2.107 255.255.255.0
NETWORK=192.168.2.0

[root@elrond ~]# ipcalc -b 192.168.2.107 255.255.255.0
BROADCAST=192.168.2.255

[root@elrond ~]# ipcalc -p 192.168.2.107 255.255.255.0
PREFIX=24

[root@elrond ~]# ipcalc -nbp 172.30.1.0/24
PREFIX=24
BROADCAST=172.30.1.255
NETWORK=172.30.1.0
```

The ipcalc on Ubuntu is nicer but you have to install it with: ***apt-get install ipcalc***

```
cis192@frodo:~$ ipcalc 172.30.4.0/24
Address:   172.30.4.0           10101100.00011110.00000100. 00000000
Netmask:   255.255.255.0 = 24   11111111.11111111.11111111. 00000000
Wildcard:  0.0.0.255            00000000.00000000.00000000. 11111111
=>
Network:   172.30.4.0/24        10101100.00011110.00000100. 00000000
HostMin:   172.30.4.1           10101100.00011110.00000100. 00000001
HostMax:   172.30.4.254         10101100.00011110.00000100. 11111110
Broadcast: 172.30.4.255         10101100.00011110.00000100. 11111111
Hosts/Net: 254                   Class B, Private Internet
```

subnetting example problem

Given the following IP address and network mask, what is the network address?

IP: 192.168.30.100
Netmask: 255.255.240.0

The first two octets of the mask are 255 so we will start the network address as 192.168.?.0. This mask indicates a /20 network (8 + 8 + 4). Next we need to apply the decimal 240 mask (1111 0000) to decimal 30 (0001 1110) which gives us binary 0001 0000 or decimal 16. Our network address is 192.168.16.0.

a) 192.168.30.0
b) 192.168.24.0
c) 192.168.15.0
d) 192.168.16.0

[root@elrond ~]# ipcalc -n 192.168.30.100 255.255.240.0
NETWORK=192.168.16.0

Team Exercise – IPv4 Addressing

http://simms-teach.com/docs/cis192/ip-exercise.pdf

Table 1-4 & remote students: Do Q1, Q7

Table 5-8: Do Q2, Q8

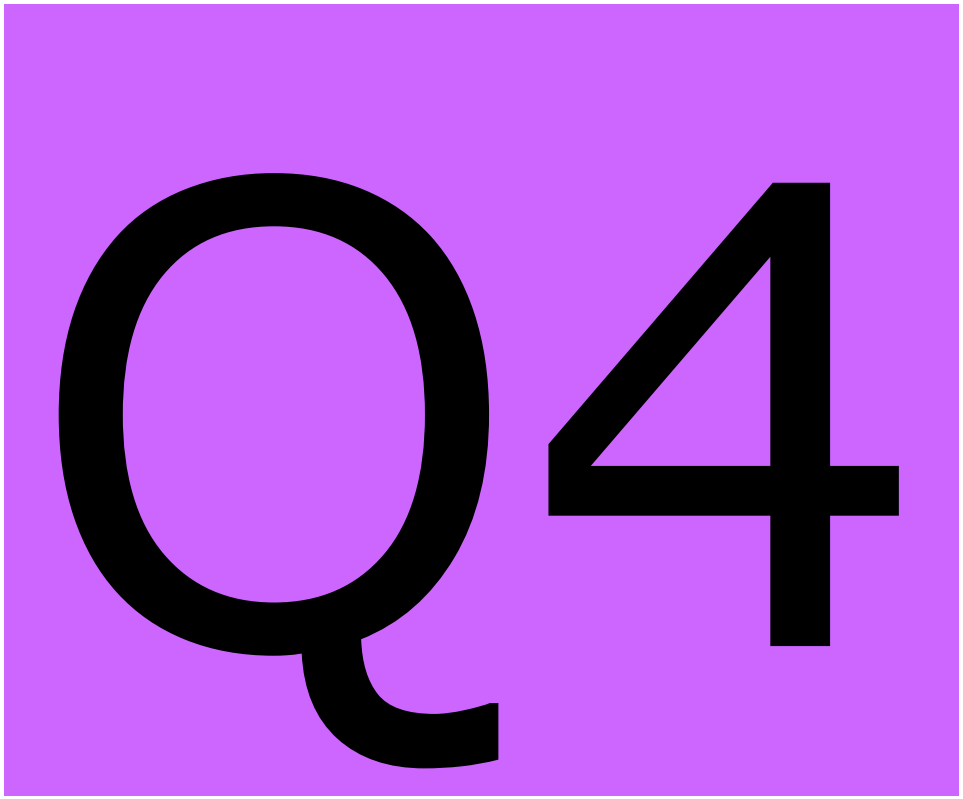
Table 9-12: Do Q3, Q9

Table 13-16: Do Q4, Q10

Table 17-20: Do Q5, Q11

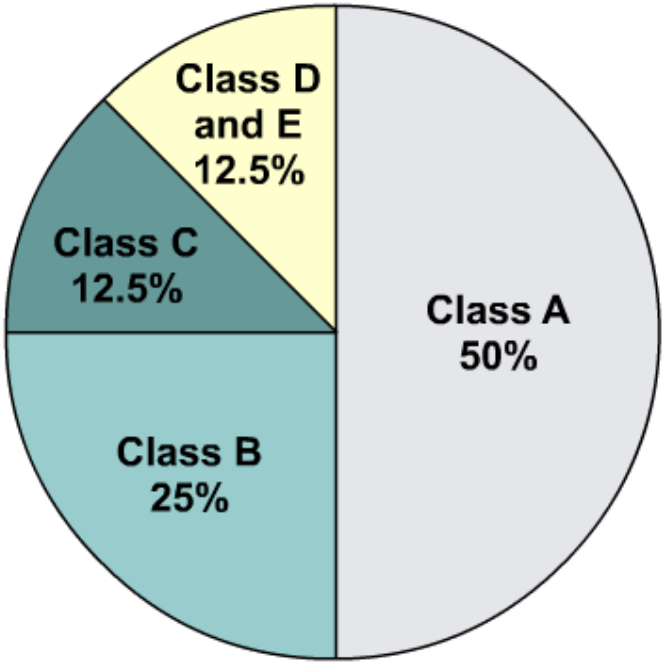
Table 21-24: Do Q6, Q12

Station numbers


Q4

NAT/PAT and IPv6

IP addressing crisis



RS: This has been a growing problem with 32 bit IP addresses

With Class A and B addresses virtually exhausted, Class C addresses (12.5 percent of the total space) are left to assign to new networks.

- Address Depletion
- Internet Routing Table Explosion

Short Term Solutions: IPv4 Enhancements

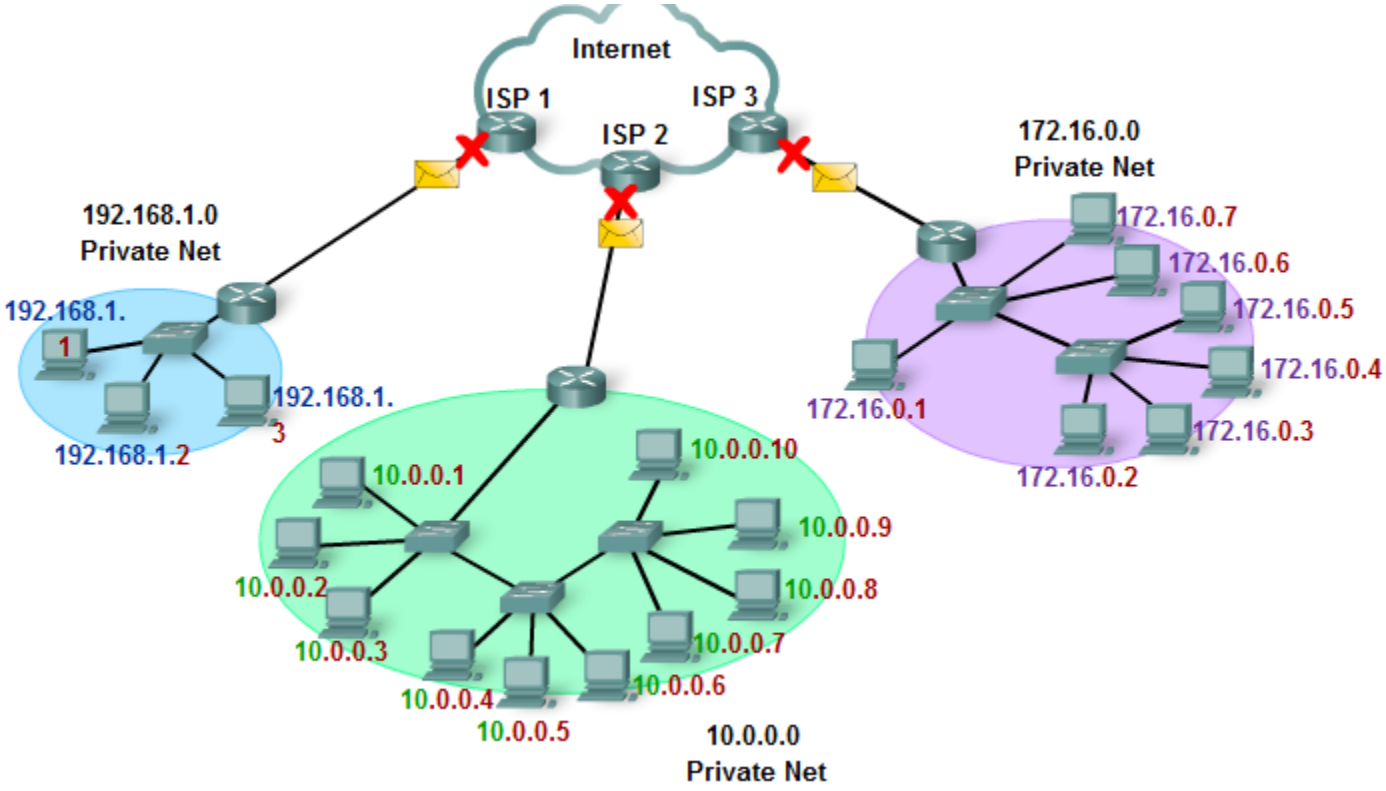
Class	RFC 1918 Internal Address Range	CIDR Prefix
A	10.0.0.0 to 10.255.255.255	10.0.0.0/8
B	172.16.0.0 to 172.31.255.255	172.16.0.0/12
C	192.168.0.0 to 192.168.255.255	192.168.0.0/16

- CIDR (Classless Inter-Domain Routing) – RFCs 1517, 1518, 1519, 1520
- VLSM (Variable Length Subnet Mask) – RFC 1009

- Private Addressing - RFC 1918
- NAT/PAT (Network Address Translation / Port Address Translation)
  - More later when we discuss TCP

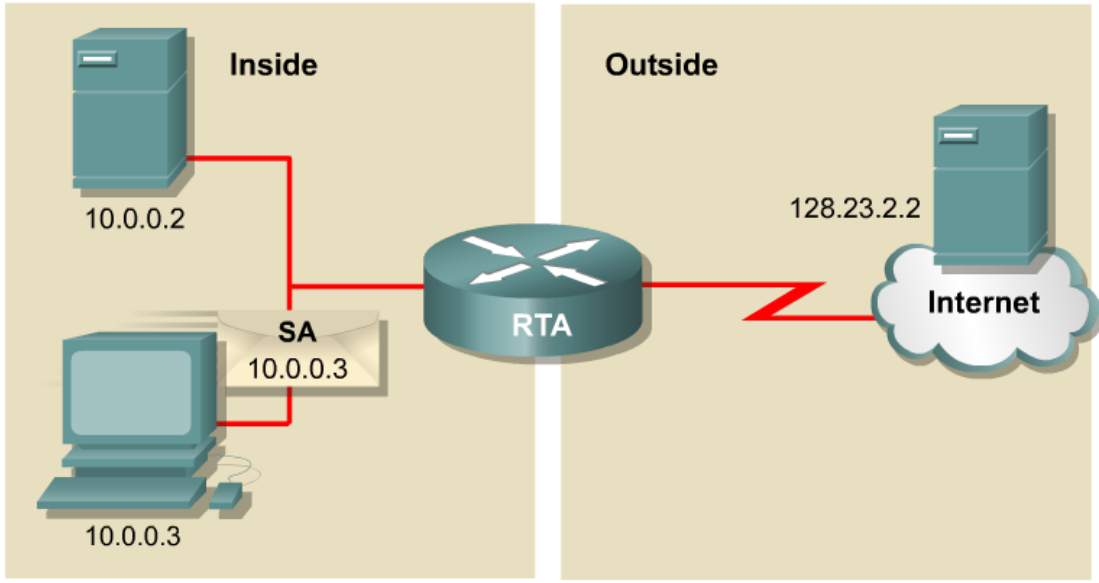
RS: CIDR IP addresses use the / notation

Private IP Addresses



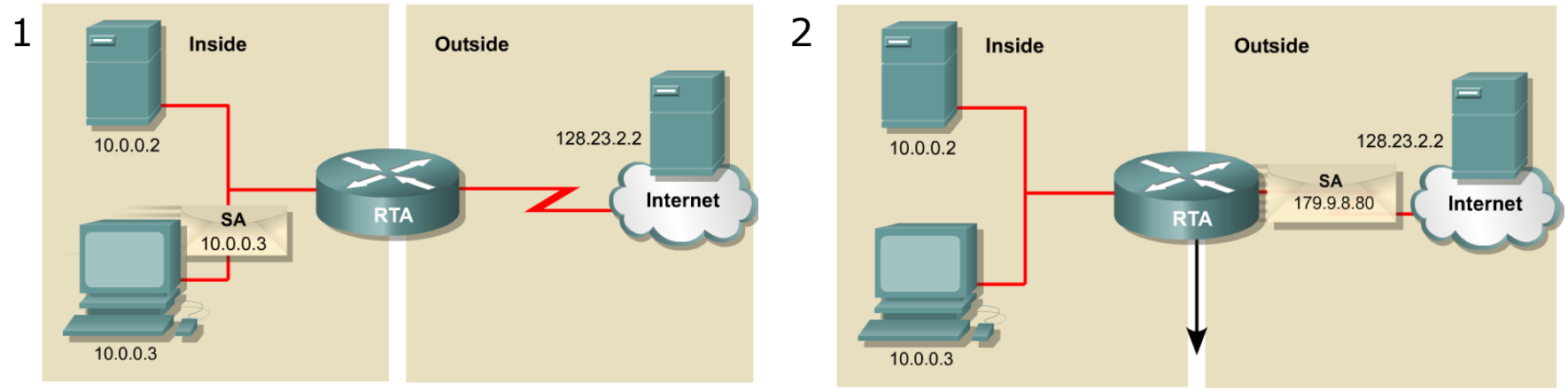
- RFC 1918
  - 10.0.0.0 to
  - 172.16.0.0 to 172.31.255.255 (172.16.0.0 /12)
  - 192.168.0.0 to 192.168.255.255 (192.168.0.0 /16)
- The addresses will not be routed in the Internet
  - Need NAT/PAT (next)
- Should be blocked by your ISP
- Allows for any network to have up to 16,777,216 hosts (/8)

Introducing NAT and PAT

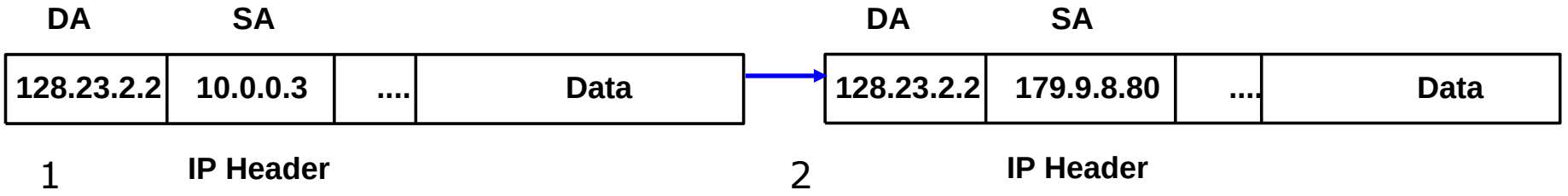


- NAT is designed to conserve IP addresses and enable networks to use private IP addresses on internal networks.
- These private, internal addresses are translated to routable, public addresses.
- IPv4 addresses are almost depleted.
- NAT/PAT has allowed IPv4 to be the predominant network protocol, keeping IPv6 at-bay (for now).

NAT Example

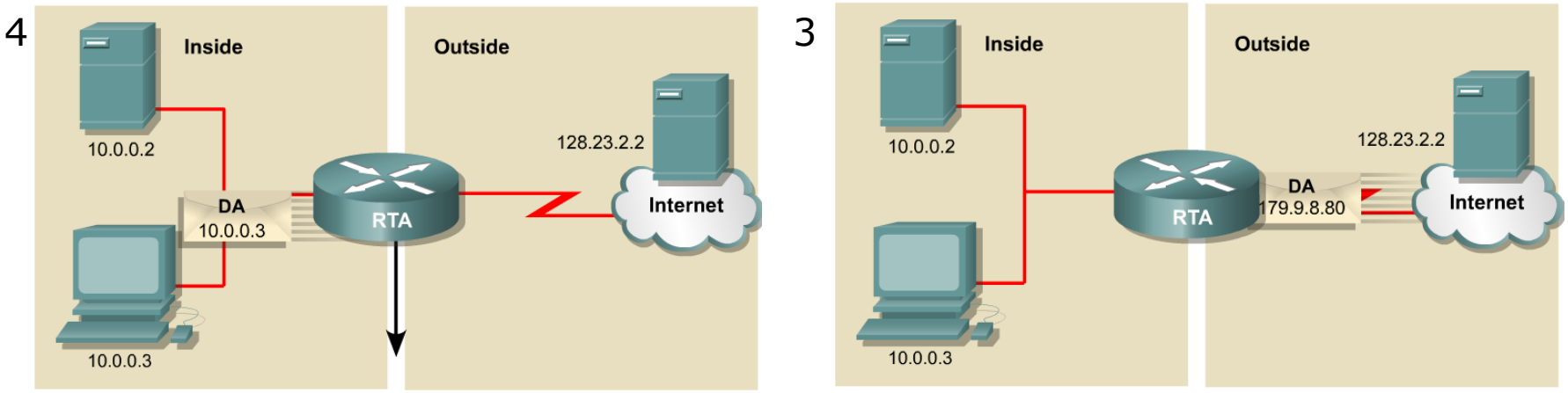


NAT Table		
Inside Local IP Address	Inside Global IP Address	Outside Global IP Address
10.0.0.3	179.9.8.80	128.23.2.2



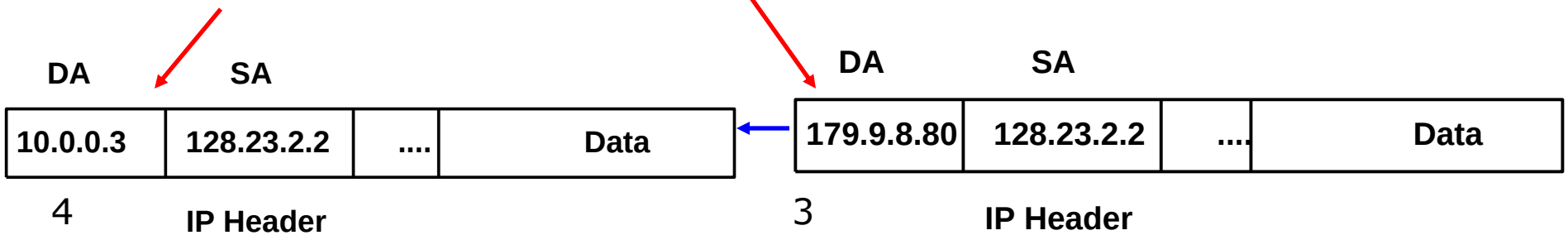
The translation from Private source IP address to Public source IP address.

NAT Example



NAT Table

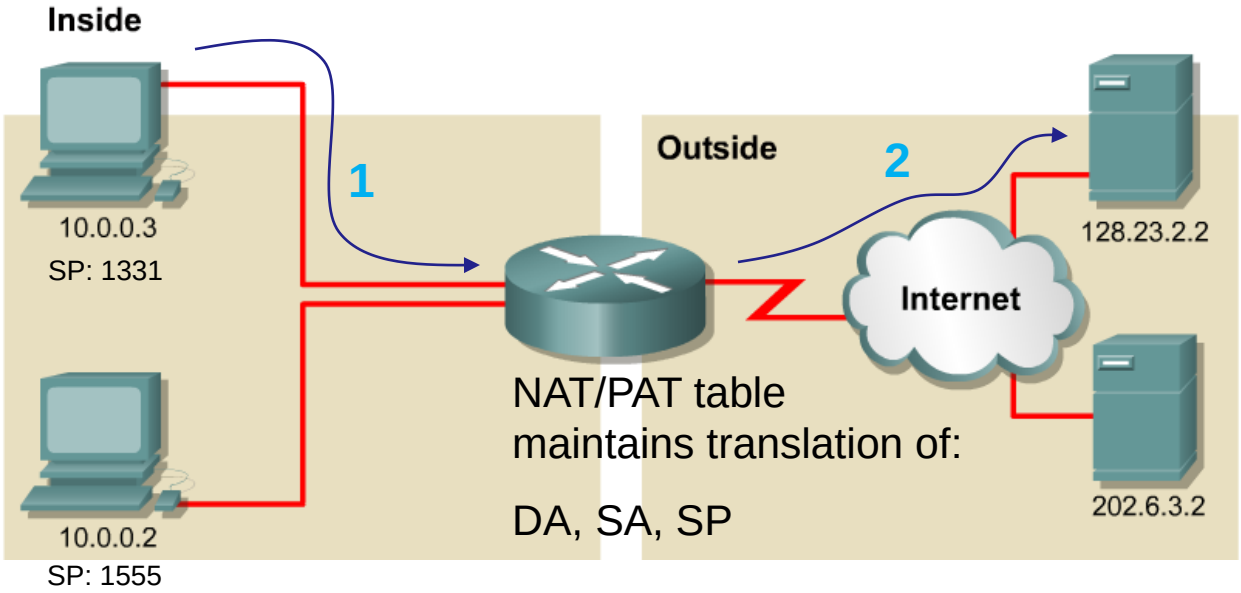
Inside Local IP Address	Inside Global IP Address	Outside Global IP Address
10.0.0.2	179.9.8.80	128.23.2.2
10.0.0.3	179.9.8.80	128.23.2.2



Translation back, from Public destination IP address to Private destination IP address.

RS: The main downfall of NAT is that you may not have a big enough pool of public addresses for every internal host needing to use the Internet at the same time.

PAT Example



DA	SA	DP	SP	
128.23.2.2	10.0.0.3	80	1331	Data

1 IP Header TCP/UDP Header

translated

DA	SA	DP	SP	
128.23.2.2	179.9.8.80	80	3333	Data

2 IP Header TCP/UDP Header

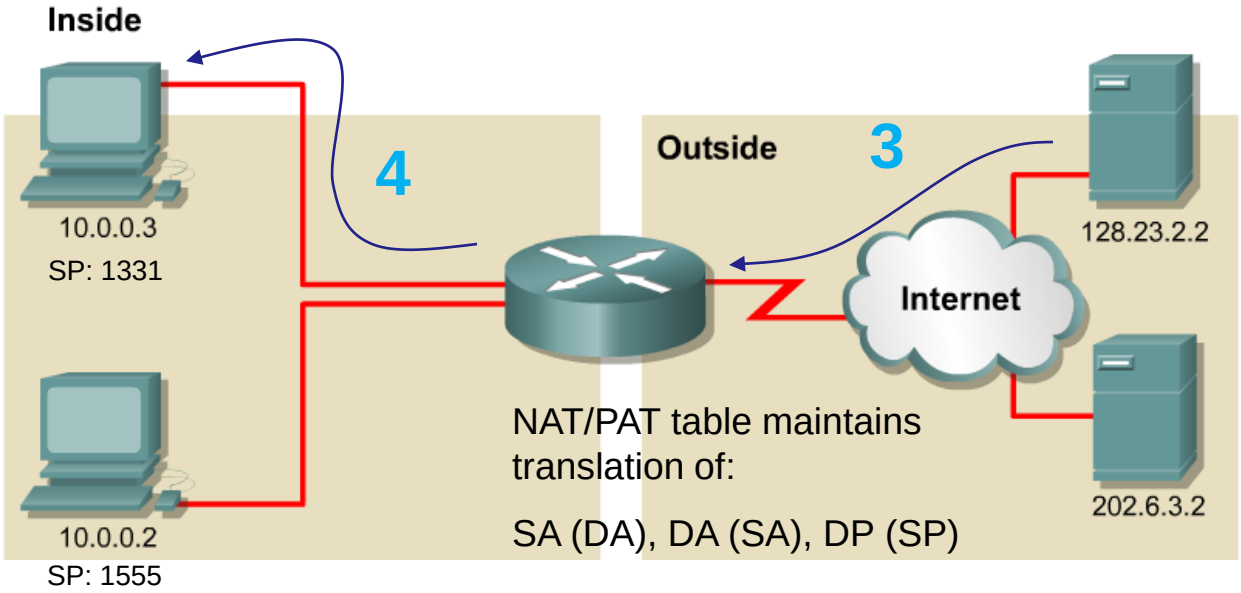
DA	SA	DP	SP	
128.23.2.2	10.0.0.2	80	1555	Data

IP Header TCP/UDP Header

DA	SA	DP	SP	
128.23.2.2	179.9.8.80	80	2222	Data

IP Header TCP/UDP Header

PAT Example



DA	SA	DP	SP	
10.0.0.3	128.23.2.2	1331	80	Data

4 IP Header TCP/UDP Header

translated

DA	SA	DP	SP	
179.9.8.80	128.23.2.2	3333	80	Data

3 IP Header TCP/UDP Header

DA	SA	DP	SP	
10.0.0.2	128.23.2.2	1555	80	Data

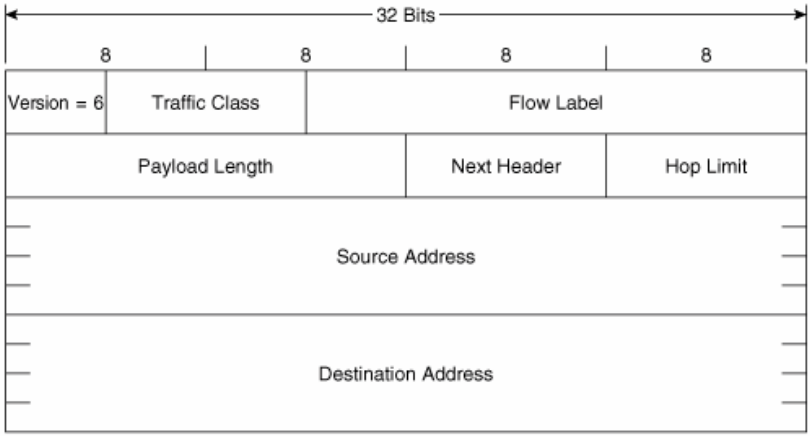
IP Header TCP/UDP Header

DA	SA	DP	SP	
179.9.8.80	128.23.2.2	2222	80	Data

IP Header TCP/UDP Header

Long Term Solution: IPv6

Figure 2-5. The IPv6 packet header.



- IPv6 replaces the 32-bit IPv4 address with a **128-bit address**, making **340 *trillion trillion trillion* IP addresses** available.

  340,282,366,920,938,463,463,374,607,431,768,211,456 addresses

  - Represented by breaking them up into **eight 16-bit segments.**
  - **Each segment** is written in **hexadecimal** between 0x0000 and 0xFFFF, separated by colons.

- An example of a written IPv6 address is

  3ffe:1944:0100:000a:0000:00bc:2500:0d0b

Long Term Solution: IPv6 (coming)

- IPv6 has been slow to arrive
- IPv6 requires new software; IT staffs must be retrained
- IPv6 will most likely coexist with IPv4 for years to come.
- Some experts believe IPv4 will remain for more than 10 years.

See Rick's presentation on IPv6 for an excellent overview

Trouble shooting

more commands

Troubleshooting ping command

- ping command tests for connectivity
- Uses ICMP protocol to send **echo requests** and **echo replies**
- Default is continuous pinging and requires Ctrl-C (a SIGINT signal) to stop.
- Use –c option to set the ping count.
- Use –R option to see route information
- Use –I option to set source address (when you have more than one interface).
- User –b option for broadcast pings

  echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts (on other nodes being pinged)

Troubleshooting ping command

Ping command using –R and –c options

```
root@frodo:~# ping -R -c 1 opus.cabrillo.edu
PING opus.cabrillo.edu (207.62.186.9) 56(124) bytes of data.
64 bytes from opus.cabrillo.edu (207.62.186.9): icmp_seq=1 ttl=63 time=2.73 ms
RR:     frodo.local (172.30.4.150)
        207.62.186.30
        opus.cabrillo.edu (207.62.186.9)
        opus.cabrillo.edu (207.62.186.9)
        172.30.4.1
        frodo.local (172.30.4.150)

--- opus.cabrillo.edu ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.732/2.732/2.732/0.000 ms
root@frodo:~#
```

Similar to traceroute

-R records the route used for the ping, -c sets the count of how many pings to send

Troubleshooting traceroute command

```
[root@elrond ~]# traceroute google.com
traceroute to google.com (209.85.171.100), 30 hops max, 40 byte packets
 1  172.30.4.1 (172.30.4.1)  5.649 ms  6.507 ms  7.695 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
```

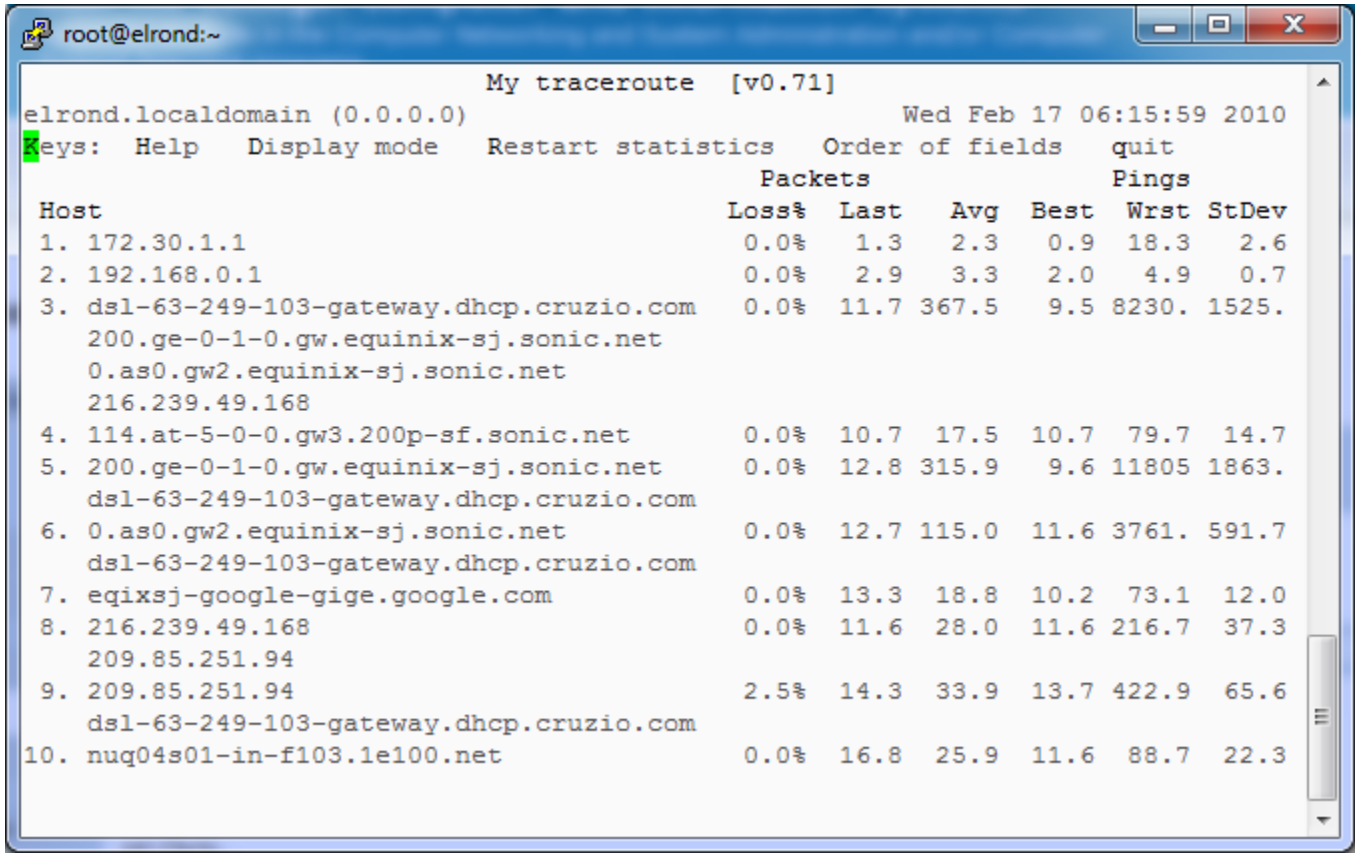
Ctrl-C to stop

Using –I option to use ICMP instead of UDP

```
[root@elrond ~]# traceroute -I google.com
traceroute to google.com (209.85.171.100), 30 hops max, 40 byte packets
 1  172.30.4.1 (172.30.4.1)  4.756 ms  6.571 ms  7.829 ms
 2  207.62.184.4 (207.62.184.4)  14.907 ms  15.631 ms  15.996 ms
 3  dc-oak-dc1--cab-cc-egm.cenic.net (137.164.34.120)  16.785 ms  17.534 ms  17.862 ms
 4  dc-oak-core1--oak-agg1-ge.cenic.net (137.164.46.55)  18.490 ms  19.003 ms  19.769 ms
 5  dc-svl-core1--oak-core1-ge-1.cenic.net (137.164.46.212)  20.769 ms  23.570 ms  26.460 ms
 6  dc-svl-peer1--svl-core1-10ge.cenic.net (137.164.46.205)  27.112 ms  10.025 ms  10.635 ms
 7  te4-4--482.tr01-plalca01.transitrail.net (137.164.131.237)  10.969 ms  9.992 ms  10.718 ms
 8   (137.164.130.94)  10.735 ms  10.675 ms  11.063 ms
 9  209.85.240.114 (209.85.240.114)  11.610 ms  10.864 ms  11.106 ms
10  216.239.49.198 (216.239.49.198)  24.040 ms  21.596 ms  21.487 ms
11  216.239.48.34 (216.239.48.34)  23.582 ms  25.061 ms  25.734 ms
12  64.233.174.101 (64.233.174.101)  20.129 ms 64.233.174.125 (64.233.174.125)  19.820 ms  19.706 ms
13  209.85.251.137 (209.85.251.137)  22.856 ms 209.85.251.129 (209.85.251.129)  33.682 ms 209.85.251.149
    (209.85.251.149)  29.731 ms
14  74.125.31.6 (74.125.31.6)  23.278 ms 74.125.31.134 (74.125.31.134)  20.824 ms 74.125.31.6 (74.125.31.6)
    21.776 ms
15  cg-in-f100.google.com (209.85.171.100)  20.158 ms  19.939 ms  19.710 ms
[root@elrond ~]#
```

Troubleshooting
mtr command

```
[root@elrond ~]# mtr google.com
```



A very nice alternative to traceroute

Troubleshooting
netstat -i command

Shows ifconfig output in tabular format

```
[root@elrond ~]# netstat -i
Kernel Interface table
Iface       MTU Met    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0       1500   0     3328      0      0      0     2827      0      0      0 BMRU
eth1       1500   0        0      0      0      0       48      0      0      0 BMRU
lo        16436   0       42      0      0      0       42      0      0      0 LRU


[root@elrond ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:82:68:7A
          inet addr:172.30.4.121  Bcast:172.30.4.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe82:687a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3344 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2840 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:354885 (346.5 KiB)  TX bytes:309367 (302.1 KiB)
          Interrupt:177 Base address:0x1400

[root@elrond ~]#
```

Class Exercise – Troubleshooting

1. Try -I, -R and -c options on the **ping** command
2. Use **traceroute google.com** and **traceroute opus.cabrillo.edu** with and without the –I option
3. Try **mtr google.com**
4. Compare **ifconfig** and **netstat –i** output

Lab

Cabrillo College

Lab 2: Joining a network

The purpose of this lab is to configure the network settings of several systems to join one or more networks. This includes setting the IP address, network mask, default gateway, and DNS settings for different distributions of Linux. Once joined, the connectivity will be tested and network traffic observed.

Supplies

- Frodo, Elrond and William VMs (CIS Lab or VLab)

Some essentials for doing labs

The "I've tried everything and it still won't work" problem

- Use the forum to ask questions and to clarify things
- Review Lesson Powerpoints which usually have examples aimed at doing the lab assignments
- Make a network diagram with all interfaces labeled. Confirm your configuration matches the diagram.
- Go back and methodically verify each step was completed. For example, if you modified /etc/hosts then cat it out and review your changes. If you set the default gateway, use route –n command to verify. If you configured an IP address, use ifconfig to verify.
- If your VM is completely "hosed": Use **Revert to snapshot** to restore to a pristine version.

Wrap

New commands, tools and services:

arp
ifconfig
ipcalc
mtr
netconfig or system-config-network
netstat
ping
ping6
tcpdump
traceroute

service network restart
/etc/init.d/networking start (Ubuntu)

service arpwatch restart (Red Hat)
/etc/init.d/arpwatch start (Ubuntu)

wireshark

New Files and Directories:
/etc/resolv.conf
/var/arpwatch/arp.dat
/var/lib/arpwatch/arp.dat

VMware:

Next Class

Assignment: Check Calendar Page on web site to see what is due next week.

Quiz questions for next class:

- What does the C flag mean when viewing ARP cache entries with arp -n?
- What Wireshark display filter would only show ARP and ICMP protocol packets?
- With an IP address of 172.30.4.100 and a netmask of 255.255.0.0, what is the broadcast address?

Backup