Lesson Module Status

- Slides – draft
- Properties - done
- Flashcards - na
- 1st minute quiz – done
- Web Calendar summary –
- Web book pages –
- Commands –
- Howtos –
- Skills pacing - na
- Lab – done
- Depot (VMs) – done
- Special:
  - Extra credit bounty on the intermittent bridged connection – done
  - log/code name strips - done

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/

Quiz

Please take out a blank piece of paper, switch off your monitor, close your books, put away your notes and answer these questions:

- What command would you use to add 172.30.4.1 as the default gateway?
- What command would you use to remove (unload) the pcnet32 NIC driver?
- At what OSI layer are IP addresses used?

Online students may emails their answers to risimms@cabrillo.edu

ARP and the Internet Layer

Related Course Objectives	Agenda
• Identify the protocols used for establishing connections between network nodes, as well as the common conventions used by each protocol. • Install and configure a local area network (LAN) that meets the resource needs of a small to medium business. • 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. • 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 • NIC Configuration • 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 command

*The **scp** command can be used to make a local copy of a file*

```
[root@elrond ~]# echo abc > fromfile
[root@elrond ~]# scp fromfile tofile
[root@elrond ~]# cat tofile
abc
```

*The **scp** command above has two arguments, the 1st argument is the source file and the second argument is the destination file*

scp command

*The **scp** command can be used to copy a local file to a remote system. To do this, specify **logname@hostname:** as a prefix to the remote file pathname*

```
[root@elrond ~]# scp fromfile rsimms@opus.cabrillo.edu:tofile
rsimms@opus.cabrillo.edu's password:
fromfile                                      100%    4     0.0KB/s   00:00
[root@elrond ~]#
```

*The **scp** command above has two arguments, the 1st argument is the source file and the second argument is the destination file*

*The **remote directory is the user's home directory***

scp command

*The **scp** command can be used to copy a remote file to a local file. To do this, specify **logname@hostname:** as a prefix to the remote file pathname*

```
[root@elrond ~]# scp rsimms@opus.cabrillo.edu:tofile myfile
rsimms@opus.cabrillo.edu's password:
tofile                                        100%    4     0.0KB/s   00:00
[root@elrond ~]#
```

*The **scp** command above has two arguments, the 1st argument is the source file and the second argument is the destination file*

*The **remote directory is the user's home directory***

scp command

*The **scp** command can be used to copy remote files between remote systems. To do this, specify **logname@hostname:** as a prefix for both files*

```
[root@elrond ~]# scp root@172.30.1.196:tofile  rsimms@opus.cabrillo.edu:myfile
root@172.30.1.196's password:
rsimms@opus.cabrillo.edu's password:
tofile                                                  100%    4     0.0KB/s   00:00
Connection to 172.30.1.196 closed.
[root@elrond ~]#
```

*The **scp** command above has two arguments, the 1st argument is the source file and the second argument is the destination file*

*The **remote directory is the user's home directory***

scp command examples

```
scp root@172.30.1.196:bin/init* .

scp mylab rsimms@opus.cabrillo.edu:

scp mylab cis192@opus.cabrillo.edu:lab1.simmsric

scp mylab rsimms@opus.cabrillo.edu:labs/lab01/lab1.simmsric

scp myfile root@172.30.1.196:/tofile

scp myfile cis192@172.30.1.196:/tofile
```

scp command examples

```
scp root@172.30.1.196:bin/init* .
```

Copies all files whose names start with "init" from the bin directory of root's home directory on 172.30.1.196 to the local working directory

```
scp mylab rsimms@opus.cabrillo.edu:
```

Copies local mylab file to the home directory of the rsimms user on Opus

```
scp mylab cis192@opus.cabrillo.edu:lab1.simmsric
```

Copies local mylab file to the home directory of the rsimms user on Opus and renames it to lab1.simmsric

```
scp mylab rsimms@opus.cabrillo.edu:labs/lab01/lab1.simmsric
```

Copies local mylab file to the lab01 directory in the labs directory in the home directory of the rsimms user on Opus and renames it to lab1.simmsric

```
scp myfile root@172.30.1.196:/tofile
```

Copies local myfile to the / directory on 172.30.1.196 (uses absolute path)

```
scp myfile cis192@172.30.1.196:/tofile
```

Attempts to copy the local myfile to the / directory on 172.30.1.196 and fails because cis192 user does not have write permissions to that directory

How to submit your work for grading

scp *fromfile tofile*

- From Windows:

```
C:\>pscp labx.txt cis192@opus.cabrillo.edu:lab1.logname
cis192@opus.cabrillo.edu's password:
C:\>
```

Replace labx.txt with your local file name and logname with your Opus login name.

- From Linux:

```
[root@arwen ~]$ scp myfile cis192@opus.cabrillo.edu:labx.logname
cis192@opus.cabrillo.edu's password:
lab1                          100%    5     0.0KB/s   00:00
[root@arwen ~]$
```

Replace myfile with your local file name, x with the lab number and logname with your Opus login name.

- Verify your submittal from Opus:

```
[simmsben@opus ~]$ ls /home/turnin/cis192
lab1.simmsben  lab2.simmsben
[simmsben@opus ~]$
```

Housekeeping

- Lab 1 is due by midnight tonight (Opus time)
- Last day to add is 2/19 (tomorrow)
- Please use the sign in sheets in the lab which are used for tracking the TBA portion of the course
- Roll call

Our CIS 192 class of Spring 2010

I have five photos so far … thanks!

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

Joe P.

Joe A.

Robert

Ryan

Chuck

Rich

John

Chris B.

Josh

Patrick

Casady

Lieven

Chris H.

Jack

Bill

Drew

Edwin

Julio

Kay

Junious

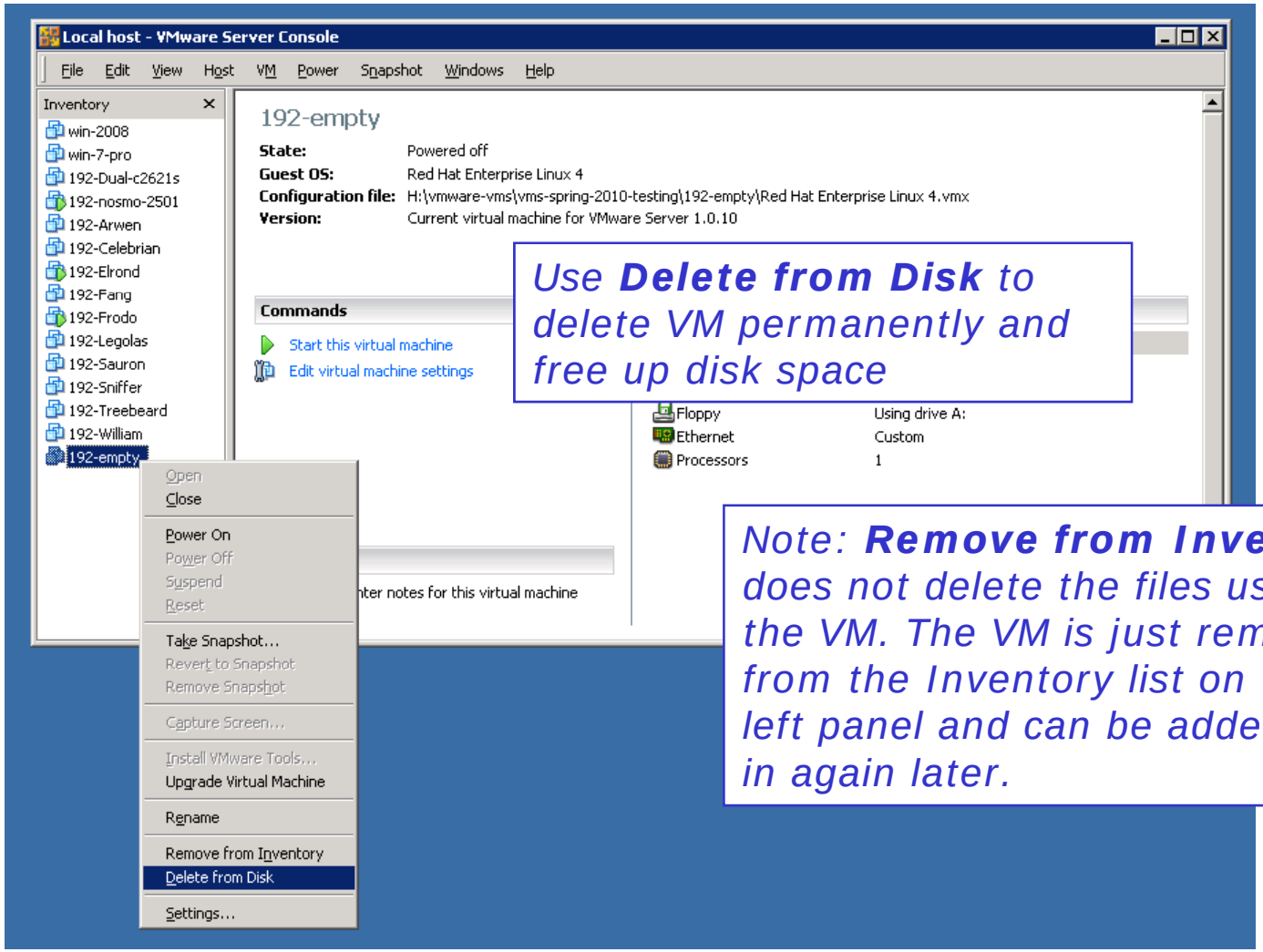
Brynden

Aaron

Randall

Joe B.

Please delete **the Empty VM** we made last time



Student Survey

UNIX/Linux Network Administration (CIS 192AB-66522)

Spring 2010 -- 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)
- CCC Confer will be used to record each class. There is a video cam option which may be used which could record student's faces. Do you give permission to post on the web any recordings that show your face? ❑ yes ❑ no

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?

Need to get surveys from Aaron and Lieven

Baseline Summary

	No understanding or experience		Some understanding or experience		Strong understanding and experience	Average	Count
Use the mtr command to trace a route	30.4% (7)	**39.1% (9)**	21.7% (5)	8.7% (2)	0.0% (0)	2.09	23
Connect virtual machines together using a virtual network	13.0% (3)	26.1% (6)	**52.2% (12)**	8.7% (2)	0.0% (0)	2.57	23
Move a virtual machine to a different physical computer	21.7% (5)	13.0% (3)	**43.5% (10)**	13.0% (3)	8.7% (2)	2.74	23
Use yum to install software packages	21.7% (5)	8.7% (2)	**43.5% (10)**	21.7% (5)	4.3% (1)	2.78	23
Leap frog from system to system across networks using SSH	17.4% (4)	21.7% (5)	17.4% (4)	**34.8% (8)**	8.7% (2)	2.96	23
Use the arp command to show the arp cache	8.7% (2)	**34.8% (8)**	26.1% (6)	13.0% (3)	17.4% (4)	2.96	23
Follow a TCP stream using Wireshark	17.4% (4)	4.3% (1)	**43.5% (10)**	30.4% (7)	4.3% (1)	3	23
Use the route command to show the routing table	8.7% (2)	13.0% (3)	**47.8% (11)**	17.4% (4)	13.0% (3)	3.13	23
Use Wireshark to capture and view packets	13.0% (3)	13.0% (3)	**30.4% (7)**	**30.4% (7)**	13.0% (3)	3.17	23
Create a new virtual machine using VMware Server	13.0% (3)	13.0% (3)	17.4% (4)	**34.8% (8)**	21.7% (5)	3.39	23
Use the dhclient command to get and release an IP address	4.3% (1)	8.7% (2)	**43.5% (10)**	26.1% (6)	17.4% (4)	3.43	23
Use the ifconfig command to configure network settings	0.0% (0)	13.0% (3)	26.1% (6)	**56.5% (13)**	4.3% (1)	3.52	23
Mount devices on the UNIX file tree	8.7% (2)	13.0% (3)	21.7% (5)	**30.4% (7)**	26.1% (6)	3.52	23
Specify absolute and relative file paths	4.3% (1)	13.0% (3)	26.1% (6)	21.7% (5)	**34.8% (8)**	3.7	23
Use virtualization software like VMware or VirtualBox	8.7% (2)	0.0% (0)	17.4% (4)	**56.5% (13)**	17.4% (4)	3.74	23
Identify subnet network, host and broadcast addresses	0.0% (0)	8.7% (2)	30.4% (7)	**39.1% (9)**	21.7% (5)	3.74	23
Use the ping command to test connectivity	0.0% (0)	0.0% (0)	17.4% (4)	13.0% (3)	**69.6% (16)**	4.52	23

Tools

Baseline Summary

	No understanding or experience		Some understanding or experience		Strong understanding and experience	Average	Count
Build a NIS server to centralize user accounts and files	65.2% (15)	34.8% (8)	0.0% (0)	0.0% (0)	0.0% (0)	1.35	23
Configure and manage printers with CUPS	69.6% (16)	17.4% (4)	13.0% (3)	0.0% (0)	0.0% (0)	1.43	23
Configure RIP or OSPF on a Linux router	69.6% (16)	17.4% (4)	8.7% (2)	0.0% (0)	4.3% (1)	1.52	23
Build a DNS server with bind	60.9% (14)	21.7% (5)	17.4% (4)	0.0% (0)	0.0% (0)	1.57	23
Configure NAT using iptables	52.2% (12)	34.8% (8)	13.0% (3)	0.0% (0)	0.0% (0)	1.61	23
Configure an LDAP directory service	56.5% (13)	26.1% (6)	17.4% (4)	0.0% (0)	0.0% (0)	1.61	23
Build a NFS server for remote directory mounts	60.9% (14)	21.7% (5)	8.7% (2)	8.7% (2)	0.0% (0)	1.65	23
Build a PXE server to automate OS/application installation	56.5% (13)	30.4% (7)	4.3% (1)	8.7% (2)	0.0% (0)	1.65	23
Configure a custom firewall using iptables	47.8% (11)	34.8% (8)	17.4% (4)	0.0% (0)	0.0% (0)	1.7	23
Build and configure a Linux router	60.9% (14)	17.4% (4)	13.0% (3)	4.3% (1)	4.3% (1)	1.74	23
Build a Samba server to share files with Windows users	56.5% (13)	21.7% (5)	13.0% (3)	8.7% (2)	0.0% (0)	1.74	23
Build a DHCP server or service	43.5% (10)	26.1% (6)	17.4% (4)	0.0% (0)	13.0% (3)	2.13	23
Build an Apache Web server to publish HTML and PHP web pages	43.5% (10)	21.7% (5)	8.7% (2)	17.4% (4)	8.7% (2)	2.26	23
Build a FTP server to share files on the Internet	39.1% (9)	21.7% (5)	8.7% (2)	13.0% (3)	17.4% (4)	2.48	23

Technologies

Baseline Summary

Modify SELinux settings to allow access to services	69.6% (16)	21.7% (5)	8.7% (2)	0.0% (0)	0.0% (0)	1.39	23
Setup and use a IPv6 network	65.2% (15)	21.7% (5)	8.7% (2)	4.3% (1)	0.0% (0)	1.52	23
Automate OS installation for a "bare-metal" computer (PXE)	60.9% (14)	26.1% (6)	8.7% (2)	4.3% (1)	0.0% (0)	1.57	23
Locate NIC drivers within the Linux file tree	47.8% (11)	34.8% (8)	17.4% (4)	0.0% (0)	0.0% (0)	1.7	23
Configure a PPP connection over a serial line	56.5% (13)	26.1% (6)	8.7% (2)	4.3% (1)	4.3% (1)	1.74	23
Load and remove kernel modules	34.8% (8)	26.1% (6)	34.8% (8)	4.3% (1)	0.0% (0)	2.09	23
Mount a directory on a remote computer	30.4% (7)	39.1% (9)	17.4% (4)	4.3% (1)	8.7% (2)	2.22	23
Identifying a socket using Wireshark packet captures	30.4% (7)	34.8% (8)	21.7% (5)	0.0% (0)	13.0% (3)	2.3	23
Configure network settings using files in /etc	17.4% (4)	43.5% (10)	21.7% (5)	17.4% (4)	0.0% (0)	2.39	23
Tunnel telnet traffic securely inside SSH over the Internet	26.1% (6)	39.1% (9)	13.0% (3)	13.0% (3)	8.7% (2)	2.39	23
Configure network settings from the command line	26.1% (6)	26.1% (6)	26.1% (6)	17.4% (4)	4.3% (1)	2.48	23
Troubleshoot and pinpoint the source of a network problem	13.0% (3)	30.4% (7)	34.8% (8)	17.4% (4)	4.3% (1)	2.7	23
Use the telnet command to access web and mail servers	17.4% (4)	26.1% (6)	30.4% (7)	8.7% (2)	17.4% (4)	2.83	23

Linux networking skills

Baseline Summary

	No understanding or experience		Some understanding or experience		Strong understanding and experience	Average	Count
Use Dynamips/Dynagen to practice CCNA skills	73.9% (17)	8.7% (2)	13.0% (3)	0.0% (0)	4.3% (1)	1.52	23
Use NetLab to practice CCNA skills	47.8% (11)	30.4% (7)	13.0% (3)	0.0% (0)	8.7% (2)	1.91	23
Configure RIP or OSPF on a Cisco router	39.1% (9)	26.1% (6)	17.4% (4)	8.7% (2)	8.7% (2)	2.22	23
Configure VLANs on a Cisco switch	39.1% (9)	13.0% (3)	17.4% (4)	21.7% (5)	8.7% (2)	2.48	23
Configure interfaces on a Cisco router	21.7% (5)	21.7% (5)	26.1% (6)	21.7% (5)	8.7% (2)	2.74	23
Manually cable together Cisco routers and systems	21.7% (5)	8.7% (2)	34.8% (8)	21.7% (5)	13.0% (3)	2.96	23
Use PacketTracer program to practice CCNA skills	21.7% (5)	8.7% (2)	30.4% (7)	13.0% (3)	26.1% (6)	3.13	23

Cisco Skills

Baseline Summary

	Needs		Ok		Very strong	Average	Count
Public presentation skills	8.7% (2)	8.7% (2)	39.1% (9)	21.7% (5)	21.7% (5)	3.39	23
Technical documentation skills	4.3% (1)	4.3% (1)	43.5% (10)	26.1% (6)	21.7% (5)	3.57	23
Planning skills	4.3% (1)	8.7% (2)	30.4% (7)	30.4% (7)	26.1% (6)	3.65	23
Teamwork skills	8.7% (2)	0.0% (0)	34.8% (8)	26.1% (6)	30.4% (7)	3.7	23
Organization skills	4.3% (1)	13.0% (3)	21.7% (5)	30.4% (7)	30.4% (7)	3.7	23
Electronic communication (forum, email, chat, etc.) skills	0.0% (0)	0.0% (0)	26.1% (6)	26.1% (6)	47.8% (11)	4.22	23

General Job Skills

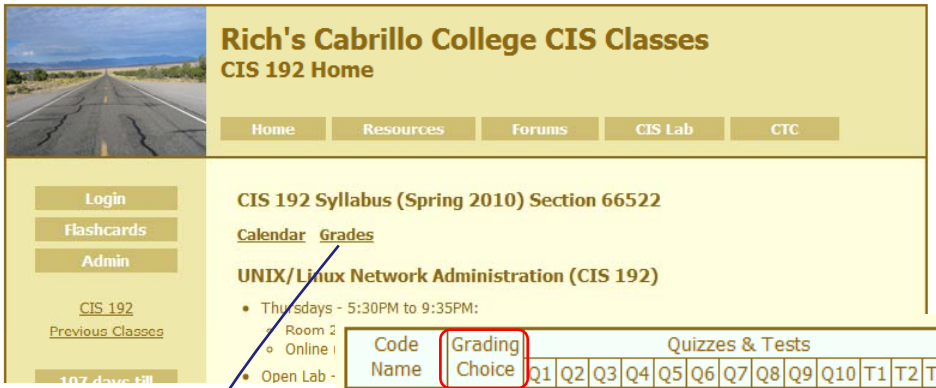
CIS 192 – Code Names
Lord of the Rings Characters

Pass out paper strips with code names and Opus lognames

If you are attending class online please email the instructor to get the information after class is over.

CIS 192 – Code Names

Lord of the Rings Characters



http://simms-teach.com/cis192grades.php

Code Name	Grading Choice	Quizzes & Tests													Forum				Labs										Final	Extra Credit	Total	Grade
		Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	T1	T2	T3	F1	F2	F3	F4	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10				
Max Points		3	3	3	3	3	3	3	3	3	3	30	30	30	20	20	20	20	30	30	30	30	30	30	30	30	30	30	60	90	560	
Arwen	Grade																													3		
Aragorn	Grade																															
Balrog	Grade																															
Bilbo	P/NP																															
Bombadil	Grade																													3		
Denethor	Grade																															
Dwalin	Grade																															
Elrond	Grade																															
Eomer	Grade																															
Frodo	Grade																															
Gimli	Grade																															
Goldberry	P/NP																													4		
Gwaihir	Grade																															
Ioreth	Grade																															
Legolas	Grade																															
Pippen	Grade																															
Samwise	Grade																															
Saruman	Grade																															
Sauron	Grade																															
Smeagol	Grade																															
Strider	Grade																													4		
Theoden	Grade																															
Treebeard	Grade																															

The code names are now available.

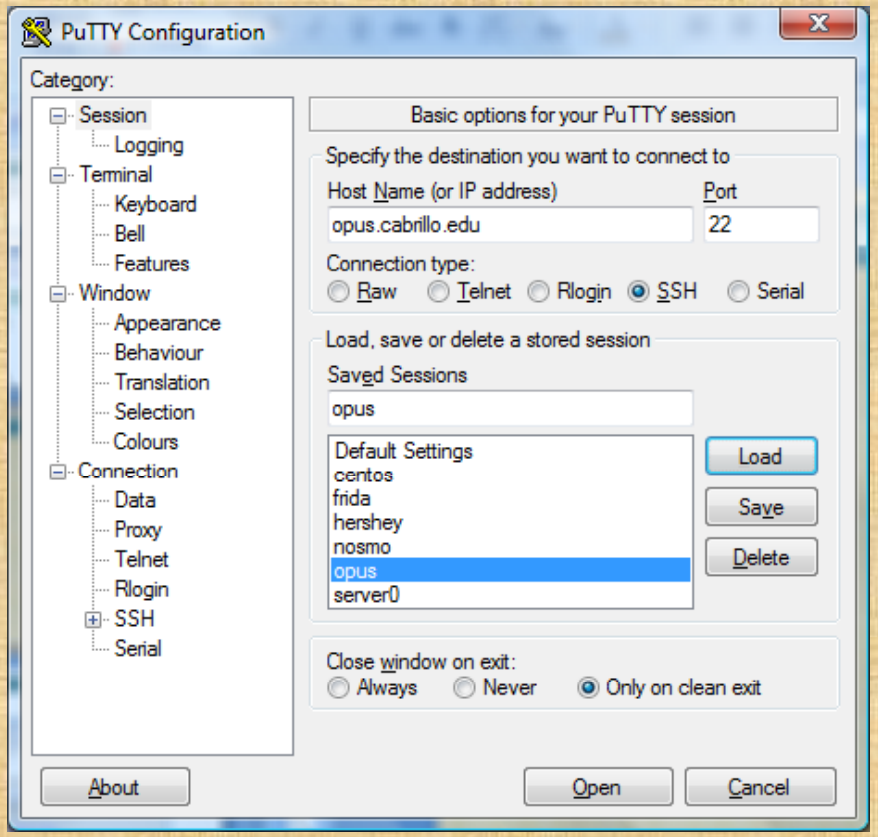
Please verify your current Grading Choice

Opus

- Opus.cabrillo.edu is a RHEL 5 server available on and off campus via Putty or ssh.
- The original 1U rackmount Opus in the “cold room” passed away last term and has been replaced by a virtual machine
- Be sure and set a STRONG password – the bots are ALWAYS trying to break into UNIX/Linux computers, like Opus, on the Internet with dictionary attacks on port 22 (ssh)
- 193 students own a directory in /home/cis192 and are members of the cis192 group.
- All other 192 students have their home directory in /home/cis192
- Use **ls /home/turnin/cis192** to verify your lab assignment submittals.

Class Exercise

Login to Opus and change passwords



Login to Opus:

1. Use new student accounts.
2. Change passwords with **passwd** command.

FYI, some Howtos for installing and configuring Putty:

http://simms-teach.com/howtos/103-install-putty.html
http://simms-teach.com/howtos/106-config-putty.html

http://simms-teach.com

Link for tips on how to repair broken VMs

http://simms-teach.com

GAH! – An intermittent connection issue in the classroom and lab with incoming packets not reaching the VM.

pcnet32 driver? the real NIC hardware? the real NIC driver? the VMware Tools vmxnet module? the part of VMware that transfers packets from the real NIC to the VM's NIC?

Intermittent Network Problem

Bridged VMs

DEAD OR ALIVE

Intermittent problem where a bridged VM cannot send or receive packets on the 172.30.x.0 /24 network. A reward is offered for a permanent solution that works every time.

25 EXTRA CREDIT POINTS REWARD

Intermittent Network Problem

Bridged VMs

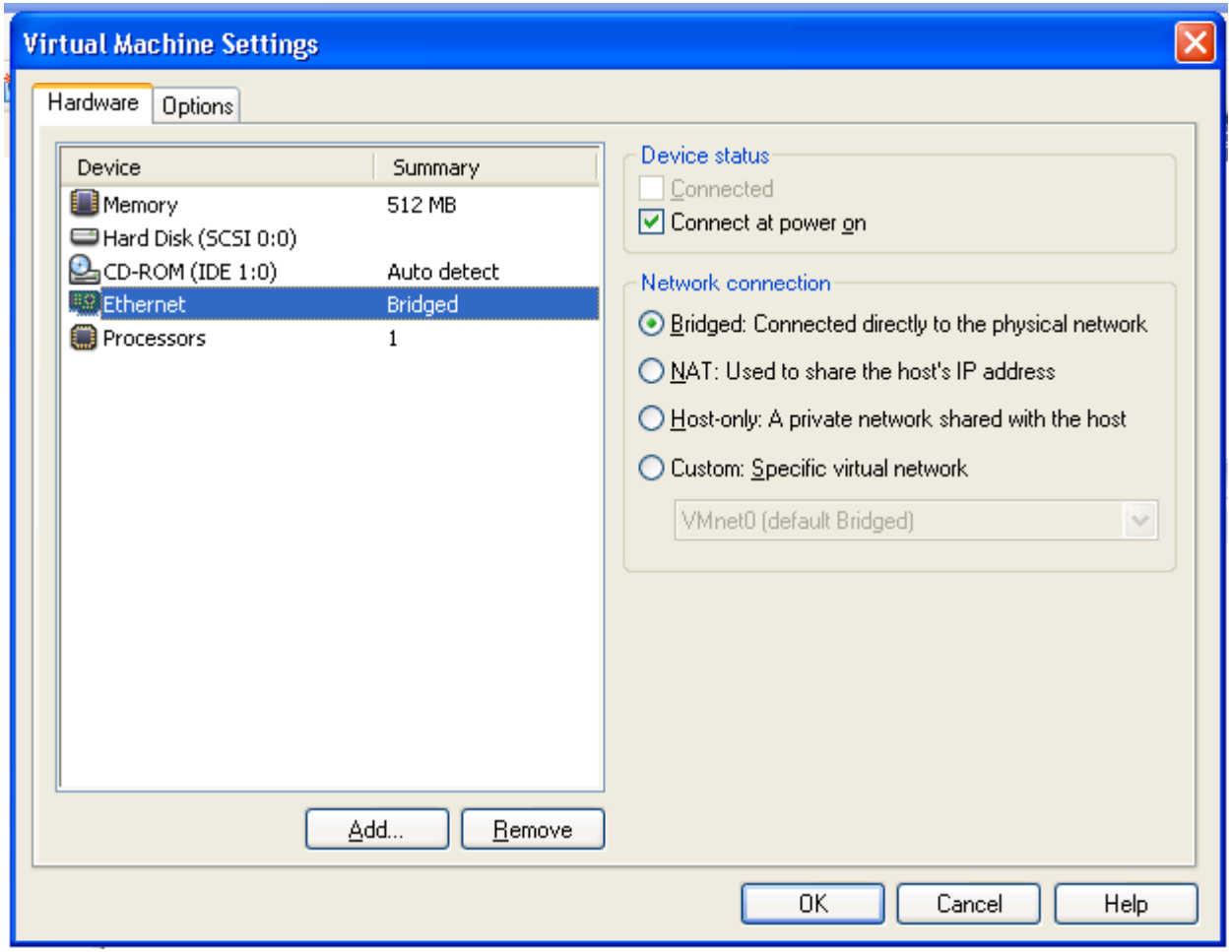
If your bridged VM fails to connect to the 172.30.1.0/24 network in room 2501 or the 172.30.4.0/24 network in room 2504/CTC try the following potential fixes in the order listed.

After applying a fix, test by pinging the router with **ping 172.30.1.1** or **ping 172.30.4.1** and if that fails try the next fix on the list:

1. Check the VM settings to make sure the Ethernet device is connected as **bridged**.
2. Try and get a new DHCP address with **dhclient –r** (to release the current address) and then **dhclient eth0** (to request a new address).
3. Restart the network service using **/etc/init.d/networking restart** on Ubuntu VMs or **service network restart** on CentOS VMs.
4. Restart the VM with init 6.
5. Restart the VMware services on the Windows station.
6. Restart the Windows VMware station.
7. Revert the VM to its snapshot (you will lose any configurations you have made)

Review

Cabling VMs



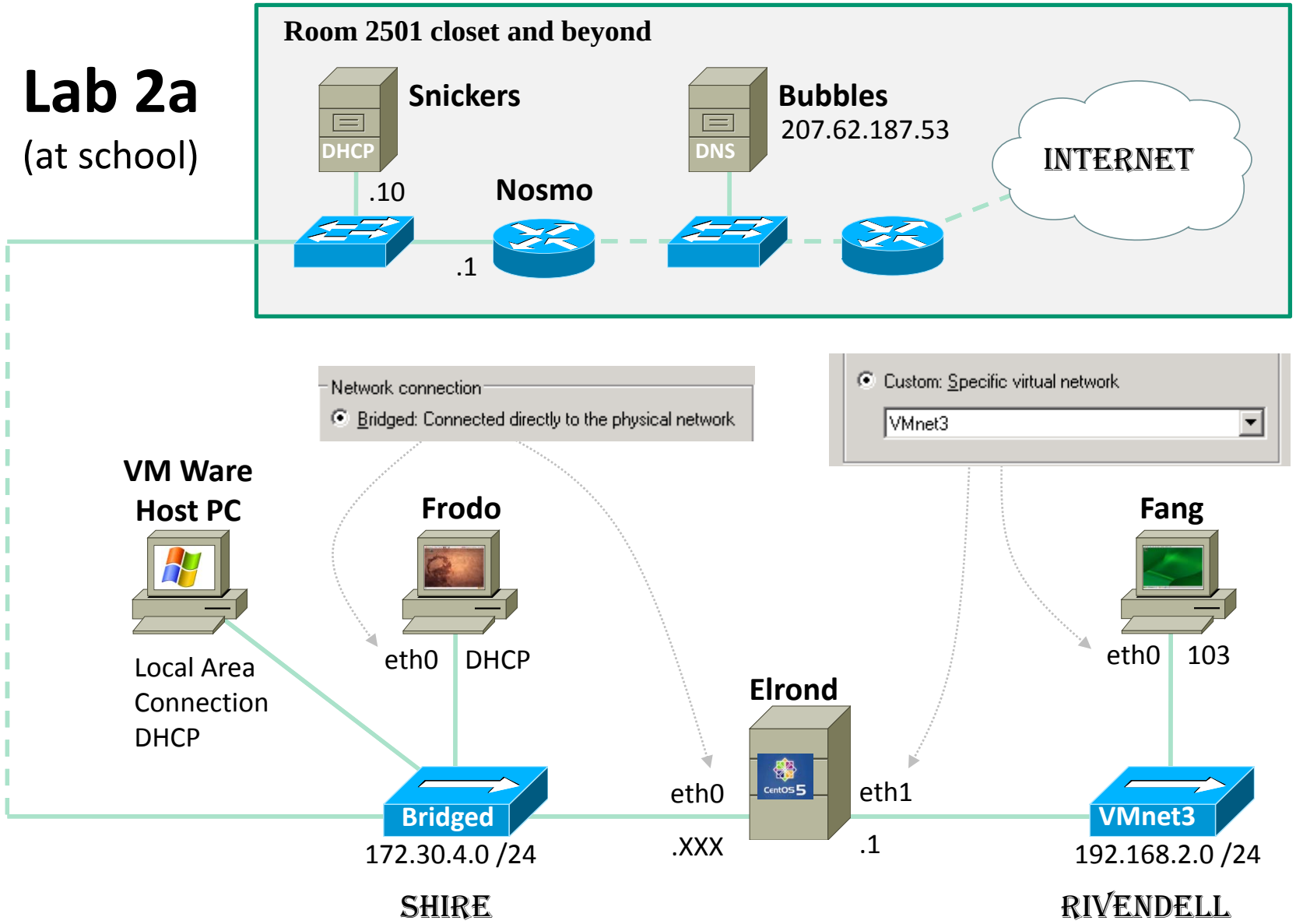
Cabling is done with the VM Settings for the Ethernet device (the NIC)

***Bridged** means the VM's NIC will use the host's physical NIC and be attached to the same network the host is. The virtual NIC will have its own MAC and IP address.*

***VMnets** can be though of as virtual hubs the VM can be cabled to.*

Lab 2a

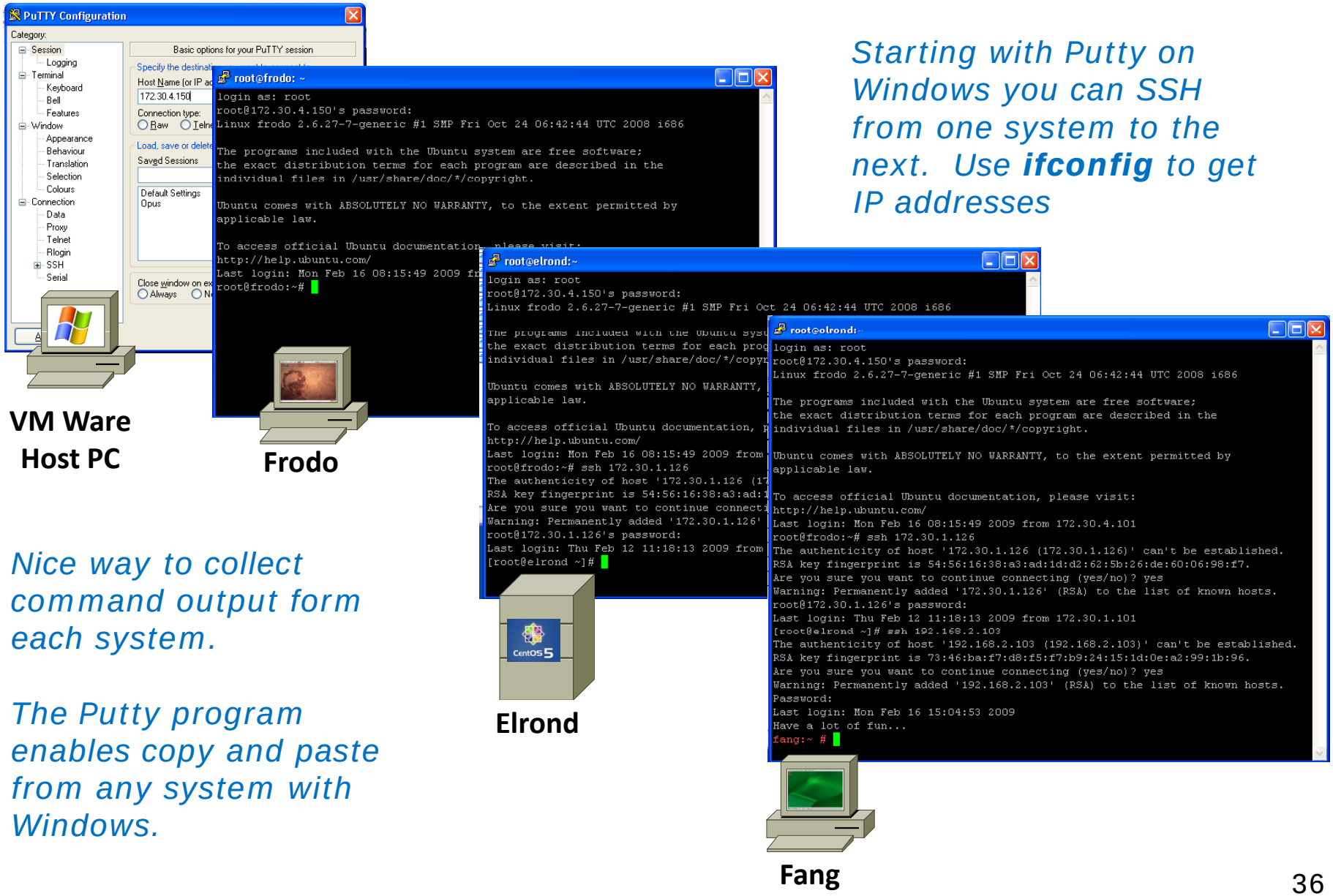
(at school)



Connecting your Linux system to the Network

1. Identify the NIC in your system (vendor and model)
   - Use **lspci**
   - Other ways: Open chassis and examine NIC card, read PC vendor specs, observe NIC driver related messages in **dmesg** output
2. Locate a driver for your NIC
   - Get driver name using http://tldp.org/HOWTO/Ethernet-HOWTO.html
   - Locate driver in /lib/modules/$(uname –r)/kernel/drivers/net
   - If not found, try to download driver from NIC vendor web site
3. Load the driver (**insmod** or **modprobe** command)
4. Bring up and configure the interface (ifconfig)

Steps 1-3 will be done automatically if your Linux distribution has the correct driver for your NIC. Step 4 will be done automatically if the interface is configured to use DHCP.



*Starting with Putty on Windows you can SSH from one system to the next. Use **ifconfig** to get IP addresses*

Nice way to collect command output form each system.

The Putty program enables copy and paste from any system with Windows.

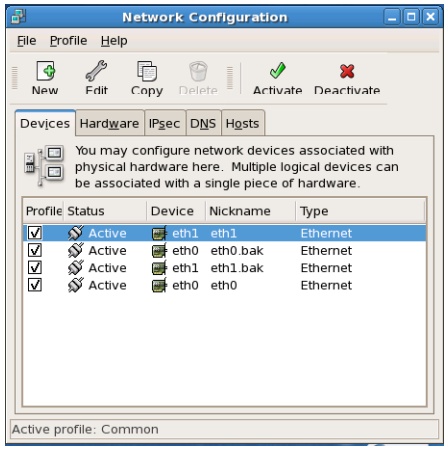
NIC Configuration

(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

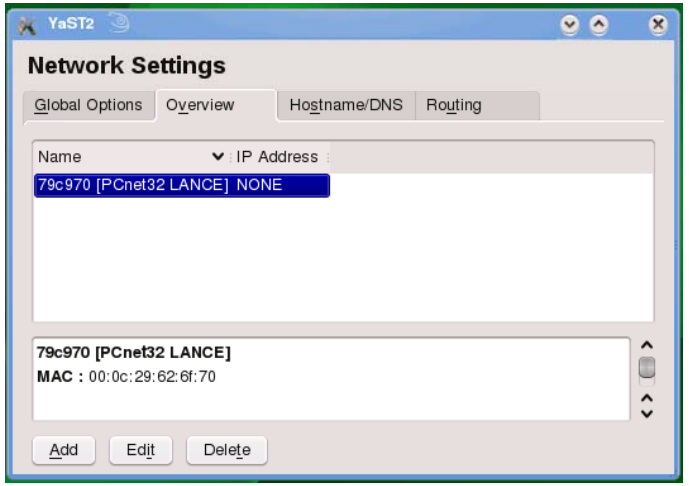


System
> Administration
> Network

Ubuntu 9.10

System
> Preferences
> Network Connections

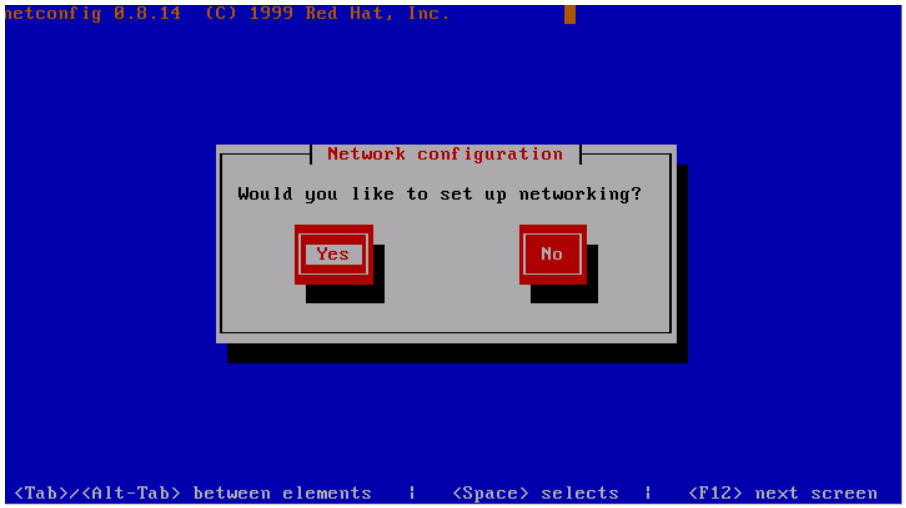
OpenSUSE 11.2



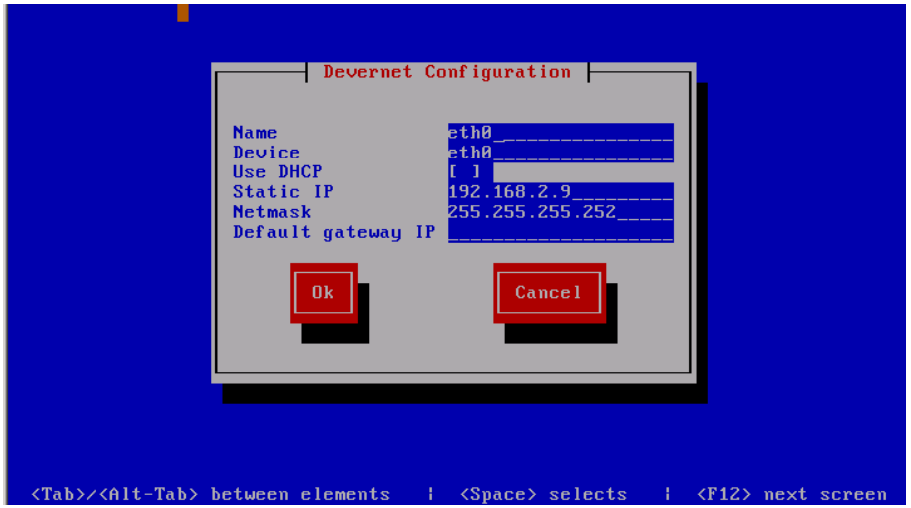
Application Launcher
> Computer
> YaST
> YaST Control Center
> Network Devices
> Network Settings

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

Yes, there is no "e"!

- 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**

*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*

Set IP Address and Subnet Mask

Set

- To set ip address and subnet mask:
  ifconfig ethx 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 interface:
  ifconfig ethx

Example

```
[root@elrond ~]# ifconfig eth1 192.168.2.107 netmask 255.255.255.0 broadcast 192.168.2.255
[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

[root@elrond ~]#
```

Configuring the default gateway

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 DNS

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 207.62.187.54 > /etc/resolv.conf
[root@elrond ~]# cat /etc/resolv.conf
nameserver 207.62.187.53
[root@elrond ~]#
```

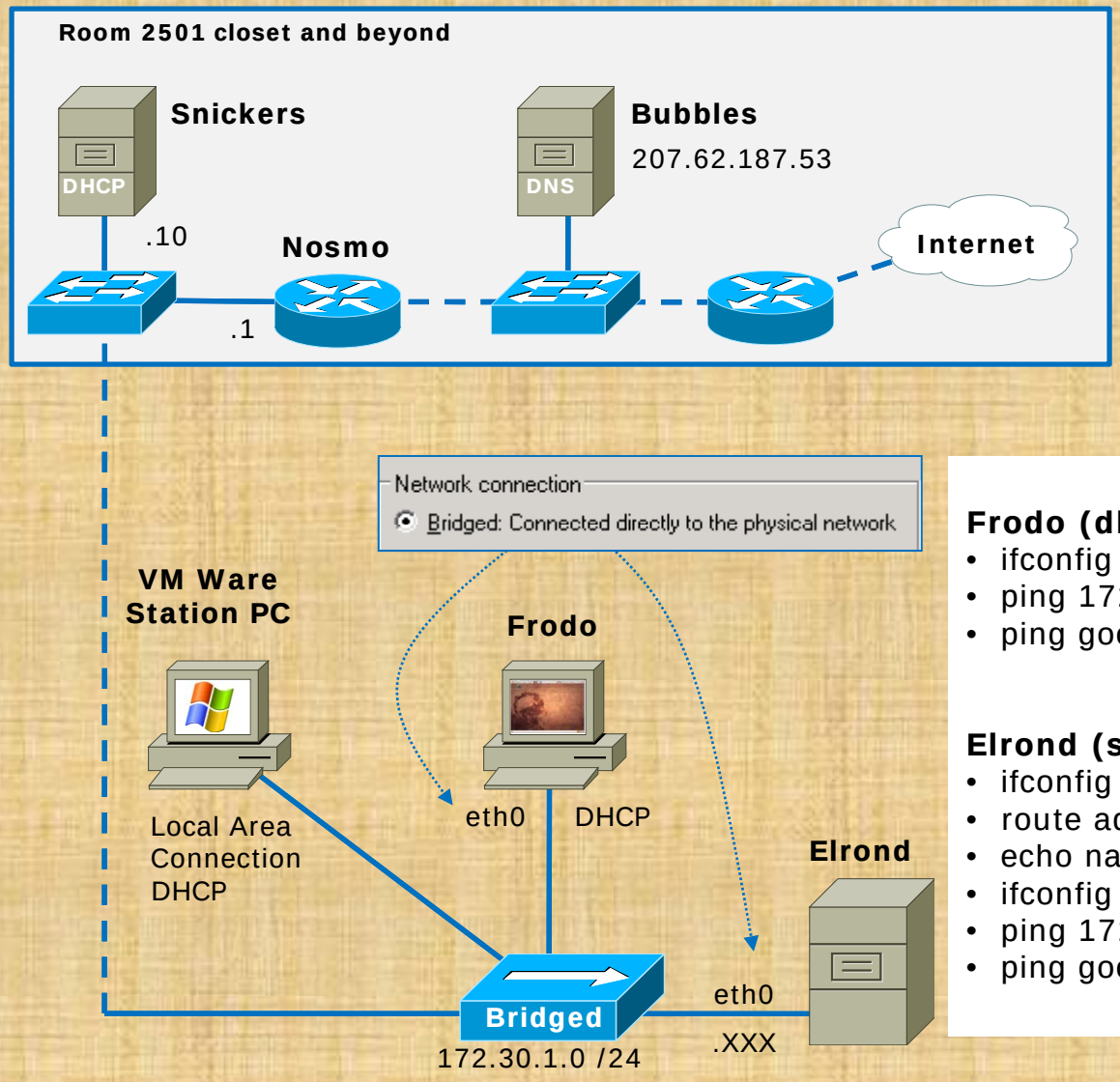
IP addresses for VM's in the classroom

Station	IP	Static 1
Instructor	172.30.1.100	172.30.1.125
Station-01	172.30.1.101	172.30.1.126
Station-02	172.30.1.102	172.30.1.127
Station-03	172.30.1.103	172.30.1.128
Station-04	172.30.1.104	172.30.1.129
Station-05	172.30.1.105	172.30.1.130
Station-06	172.30.1.106	172.30.1.131
Station-07	172.30.1.107	172.30.1.132
Station-08	172.30.1.108	172.30.1.133
Station-09	172.30.1.109	172.30.1.134
Station-10	172.30.1.110	172.30.1.135
Station-11	172.30.1.111	172.30.1.136
Station-12	172.30.1.112	172.30.1.137

Station	IP	Static 1
Station-13	172.30.1.113	172.30.1.138
Station-14	172.30.1.114	172.30.1.139
Station-15	172.30.1.115	172.30.1.140
Station-16	172.30.1.116	172.30.1.141
Station-17	172.30.1.117	172.30.1.142
Station-18	172.30.1.118	172.30.1.143
Station-19	172.30.1.119	172.30.1.144
Station-20	172.30.1.120	172.30.1.145
Station-21	172.30.1.121	172.30.1.146
Station-22	172.30.1.122	172.30.1.147
Station-23	172.30.1.123	172.30.1.148
Station-24	172.30.1.124	172.30.1.149

Note the static IP address for your station to use in the next class exercise

Class Exercise – Join Frodo and Elrond to classroom network



Frodo (dhcp)

- ifconfig eth0
- ping 172.30.1.1
- ping google.com

Elrond (static)

- ifconfig eth0 172.30.1.1xx netmask 255.255.255.0
- route add default gw 172.30.1.1
- echo nameserver 207.62.187.53 > /etc/resolv.conf
- ifconfig eth0
- ping 172.30.1.1
- ping google.com

ifconfig and aliases

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

Set

- To set an alias IP address and subnet mask:
  ifconfig ethx:n 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 ethx:n

It is possible to have more that 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)

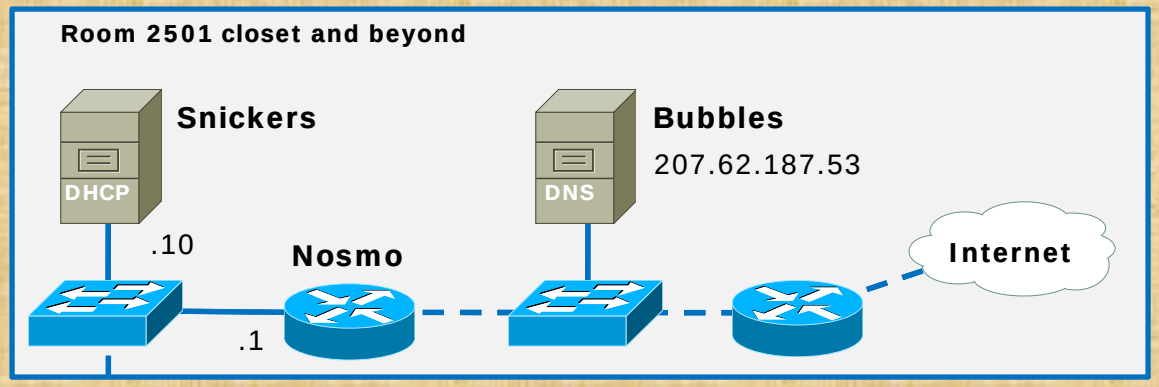
Example

```
[root@elrond ~]# ifconfig eth0:1 172.30.4.122 netmask 255.255.255.0 broadcast 172.30.4.255
[root@elrond ~]#
[root@elrond ~]# ifconfig eth0:1
eth0:1    Link encap:Ethernet  HWaddr 00:0C:29:82:68:7A
          inet addr:172.30.4.122  Bcast:172.30.4.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:177 Base address:0x1400

[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:4863 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3442 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:566772 (553.4 KiB)  TX bytes:382355 (373.3 KiB)
          Interrupt:177 Base address:0x1400

[root@elrond ~]#
```

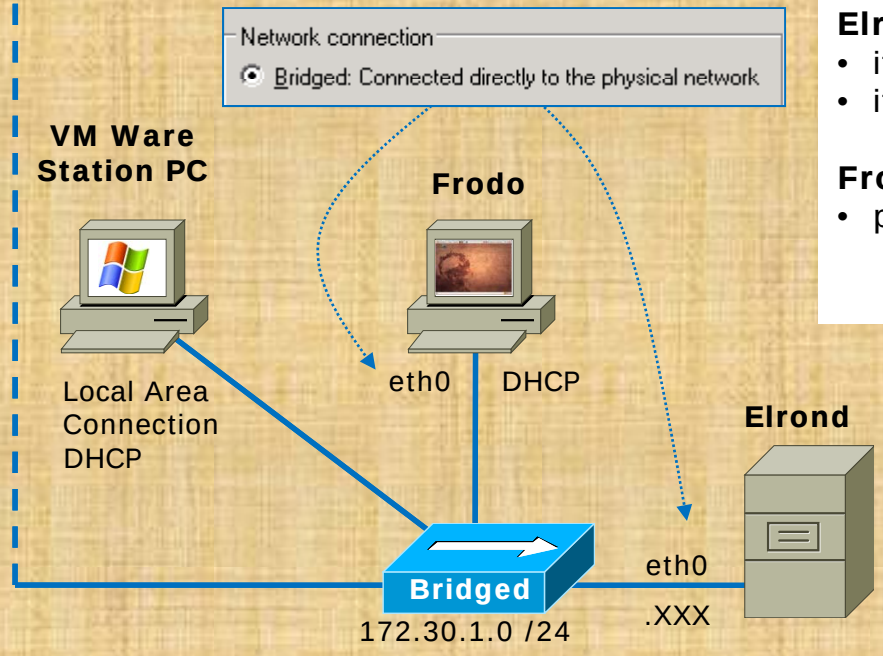
```
root@frodo:~# ping -c 1  172.30.1.121
PING 172.30.1.121 (172.30.1.121) 56(84) bytes of data.
64 bytes from 172.30.1.121: icmp_seq=1 ttl=127 time=6.04 ms

--- 172.30.1.121 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 6.049/6.049/6.049/0.000 ms
root@frodo:~# ping -c 1  172.30.1.122
PING 172.30.1.122 (172.30.1.122) 56(84) bytes of data.
64 bytes from 172.30.1.122: icmp_seq=1 ttl=127 time=0.900 ms

--- 172.30.1.122 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.900/0.900/0.900/0.000 ms
root@frodo:~#
```

Frodo now can ping either of Elrond's two host IP addresses

Class Exercise – Add an alias IP address



Make your 2nd static IP address for Elrond be 172.30.1.2xx where xx is your station number



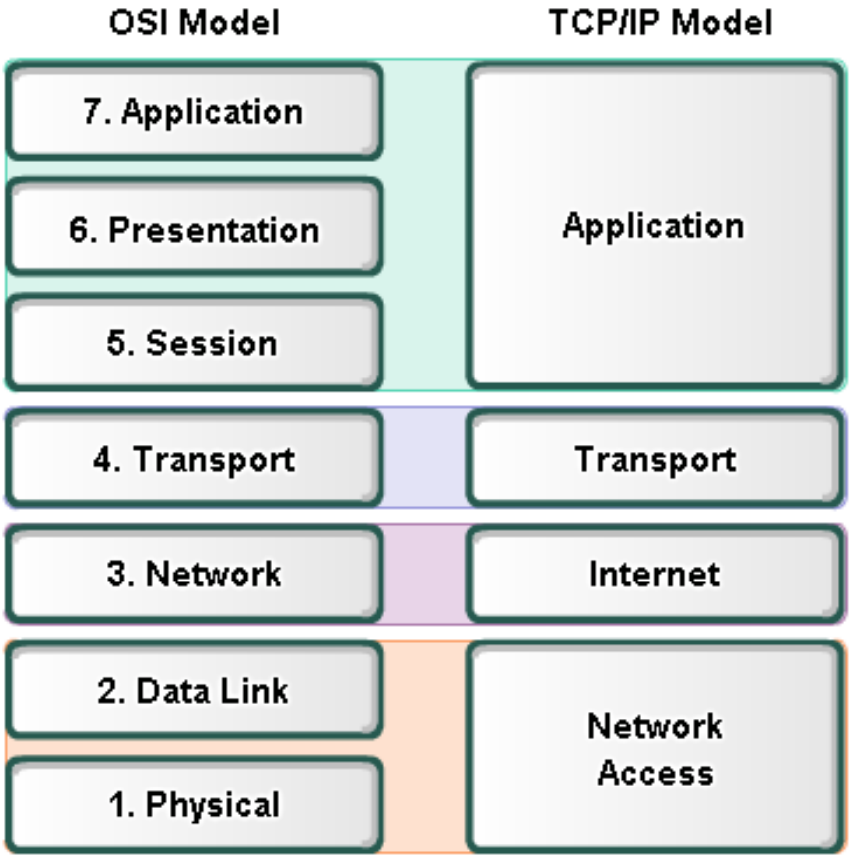
Elrond (static)
- ifconfig eth0:1 172.30.1.2xx netmask 255.255.255.0
- ifconfig

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

ARP

Protocol and Reference Models

Late 70's, International Organization for Standardization (ISO)



Early 70's, Defense Advanced Research Projects Agency (DARPA)

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

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.

Rich's note: The layering here is blurry. IP address are being determined (Layer 3). The request and replies use frames addressed with MAC addresses (Layer 2) with the IP information held in the payload.

ARP – Address Resolution Protocol

Overview

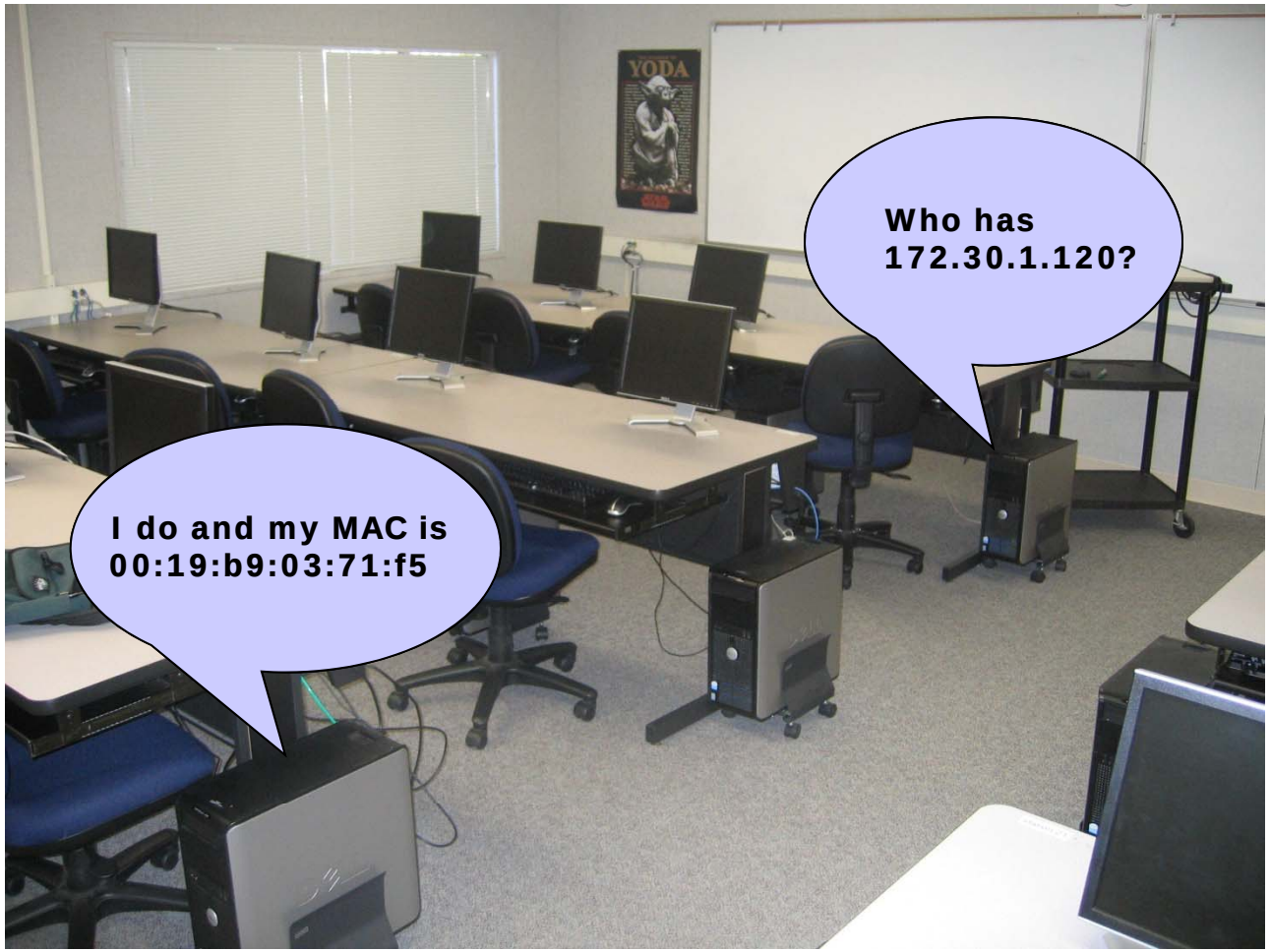
The purpose of ARP is to provide the correct destination physical address given the destination IP address.

- 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 replay: targeted to the requestor's address (unicast) – "I do and my MAC address is *xx:xx:xx:xx:xx:xx*"

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

RARP

Reverse Address Resolution Protocol

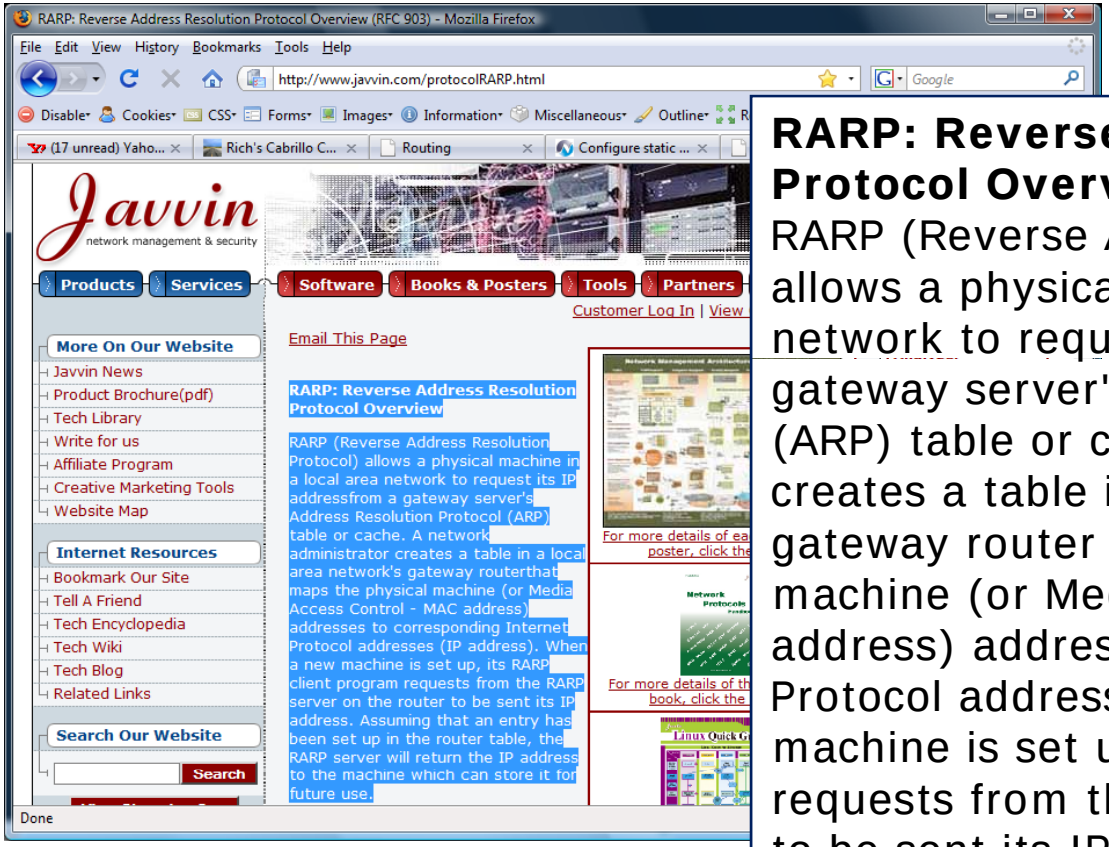
- For diskless clients and X workstations that need an IP address when they start up.
- Requires a RARP server.
- RARP request is like an ARP request but instead is "Who has this MAC address?"
- RARP reply is "I have that MAC and it's IP address is *xxx.xxx.xxx.xxx*"
- RARP is pretty much obsolete now that DHCP is used to provide IP addresses.

RARP

- RARP, or Reverse Address Resolution Protocol.
- Like ARP, used to map MAC address to IP addresses.
- Unlike ARP, used by devices to find their own IP address, not MAC address.
- What kind of device would not know its own IP address?
- Dumb terminals are diskless workstations.
- Diskless workstations have no permanent storage (like a hard drive) to store network configurations.
- Dumb terminals will know their own MAC address because it's burned in to the card, but they have to use RARP to find their IP.

Dumb Terminals

RARP



RARP: Reverse Address Resolution Protocol Overview

RARP (Reverse Address Resolution Protocol) allows a physical machine in a local area network to request its IP address from a gateway server's Address Resolution Protocol (ARP) table or cache. A network administrator creates a table in a local area network's gateway router that maps the physical machine (or Media Access Control - MAC address) addresses to corresponding Internet Protocol addresses (IP address). When a new machine is set up, its RARP client program requests from the RARP server on the router to be sent its IP address. Assuming that an entry has been set up in the router table, the RARP server will return the IP address to the machine which can store it for future use.

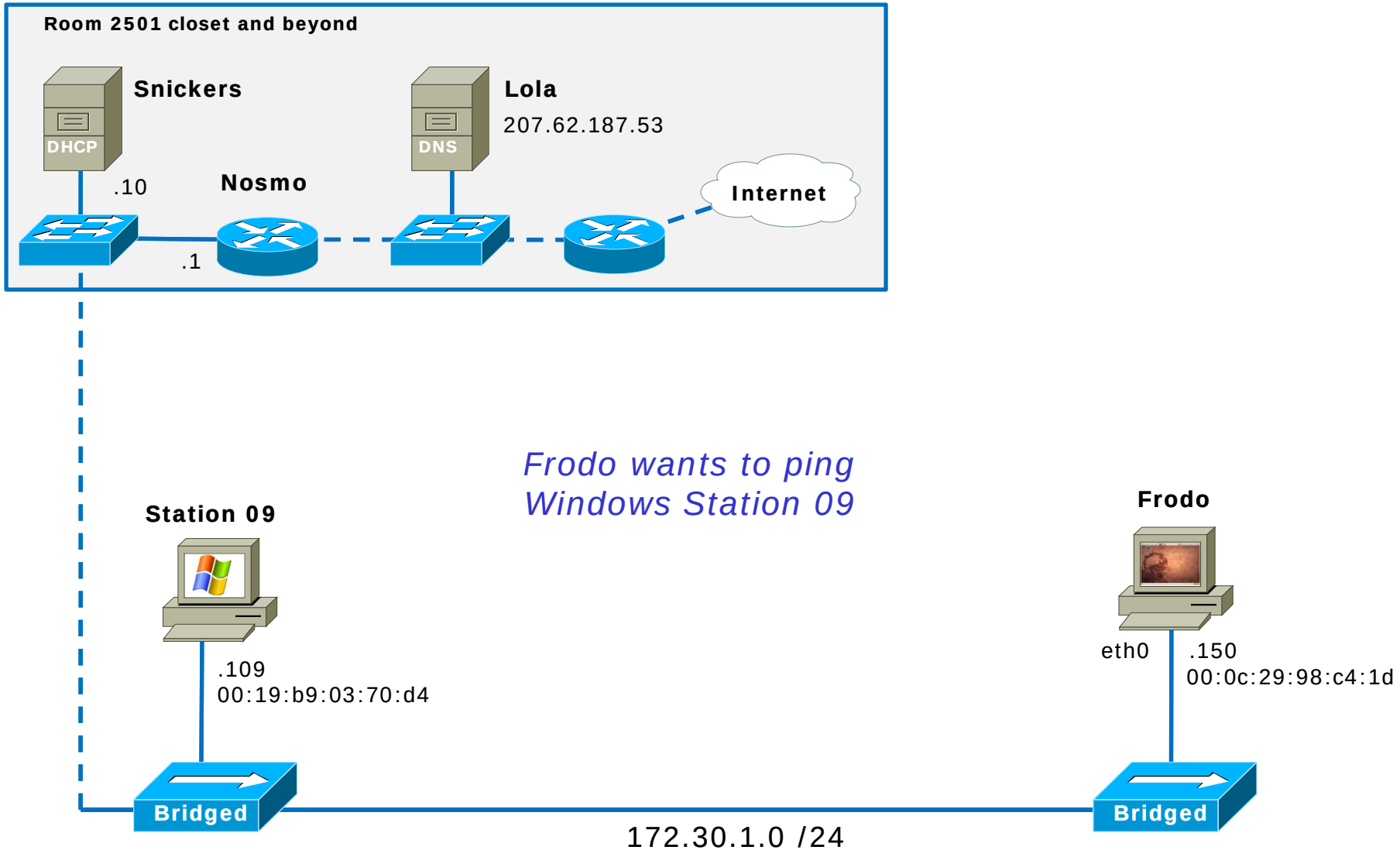
http://www.javvin.com/protocolRARP.html

ARP Poisoning

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

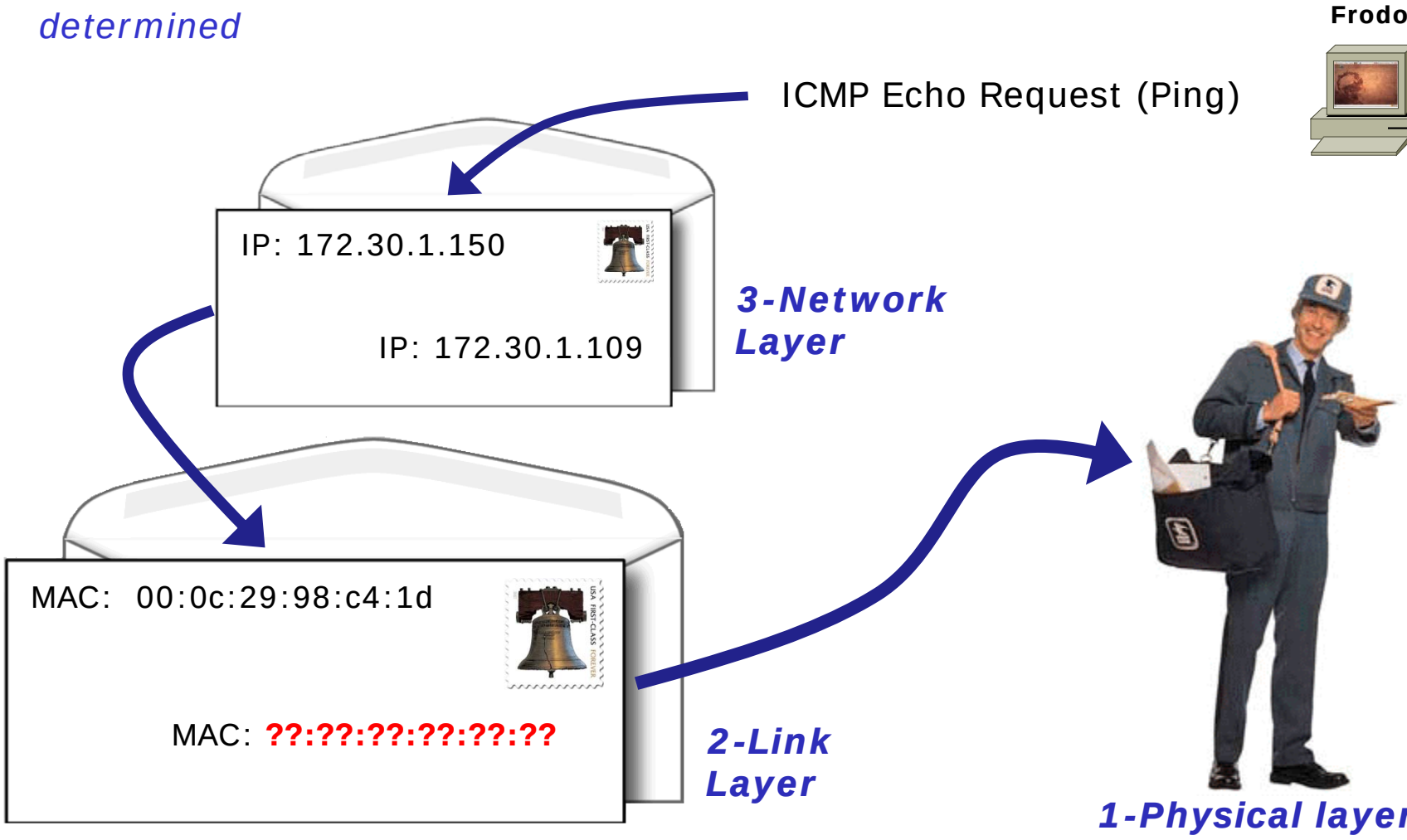
- An attacker can send arp replies (even as a broadcast) to populate arp caches with bogus MAC/IP pairs
  - Denial of service: pair a non-existing MAC address with the router's IP address. External destination packets can never leave the subnet.
  - Man-in-the-middle: pair an existing hosts IP address 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

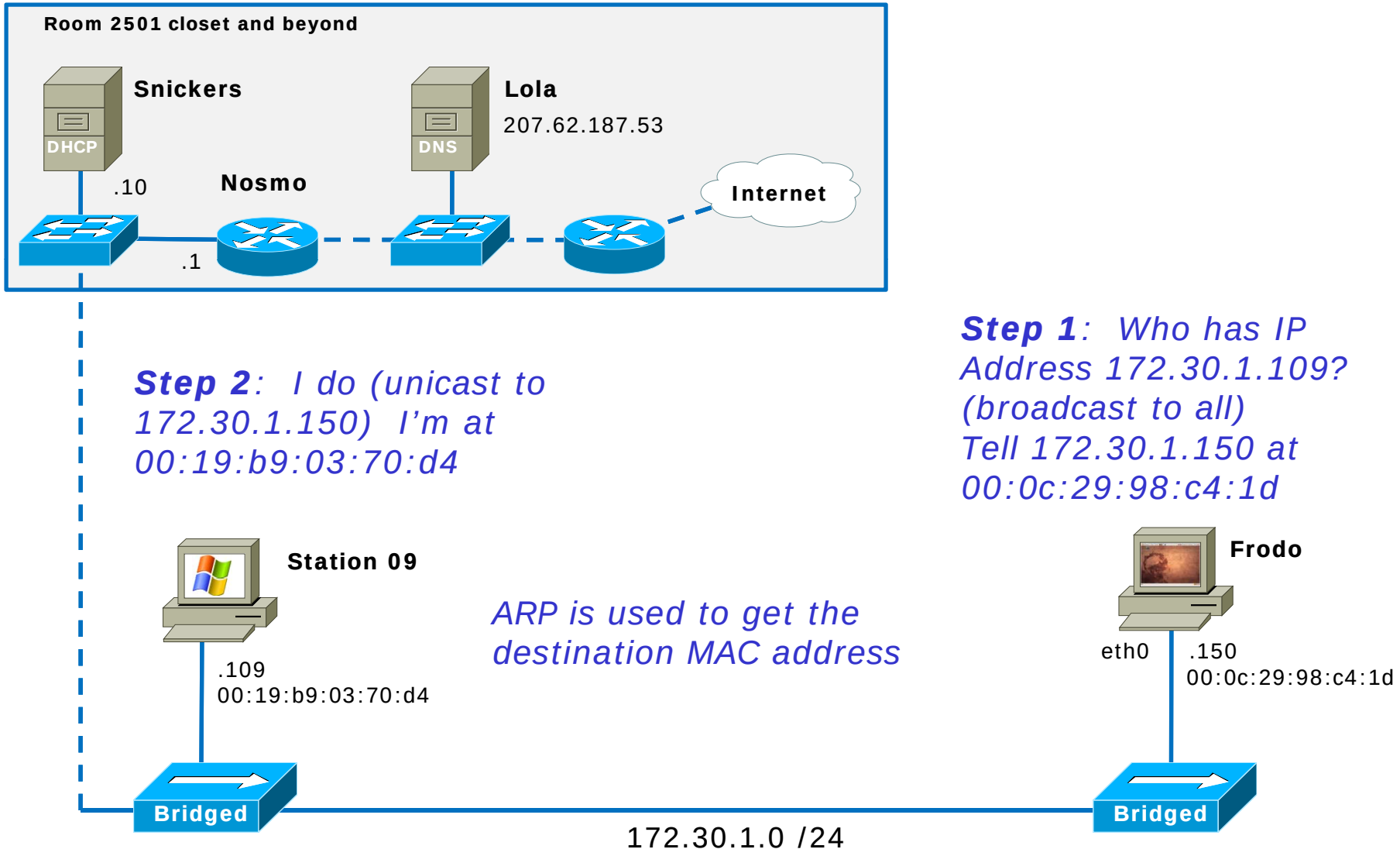


ARP Example - Frodo pings Station09

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

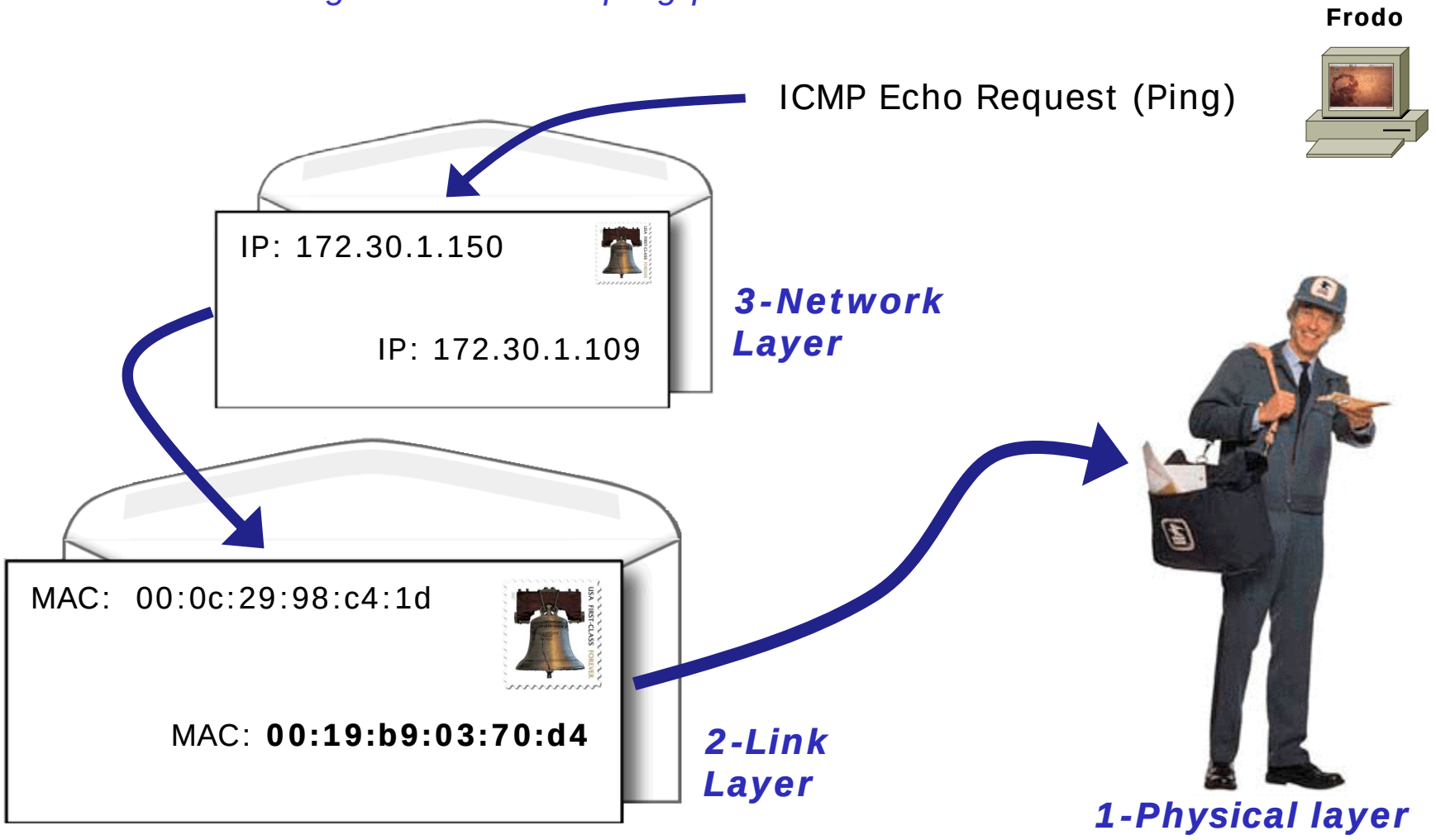


ARP Example - Frodo pings Station09

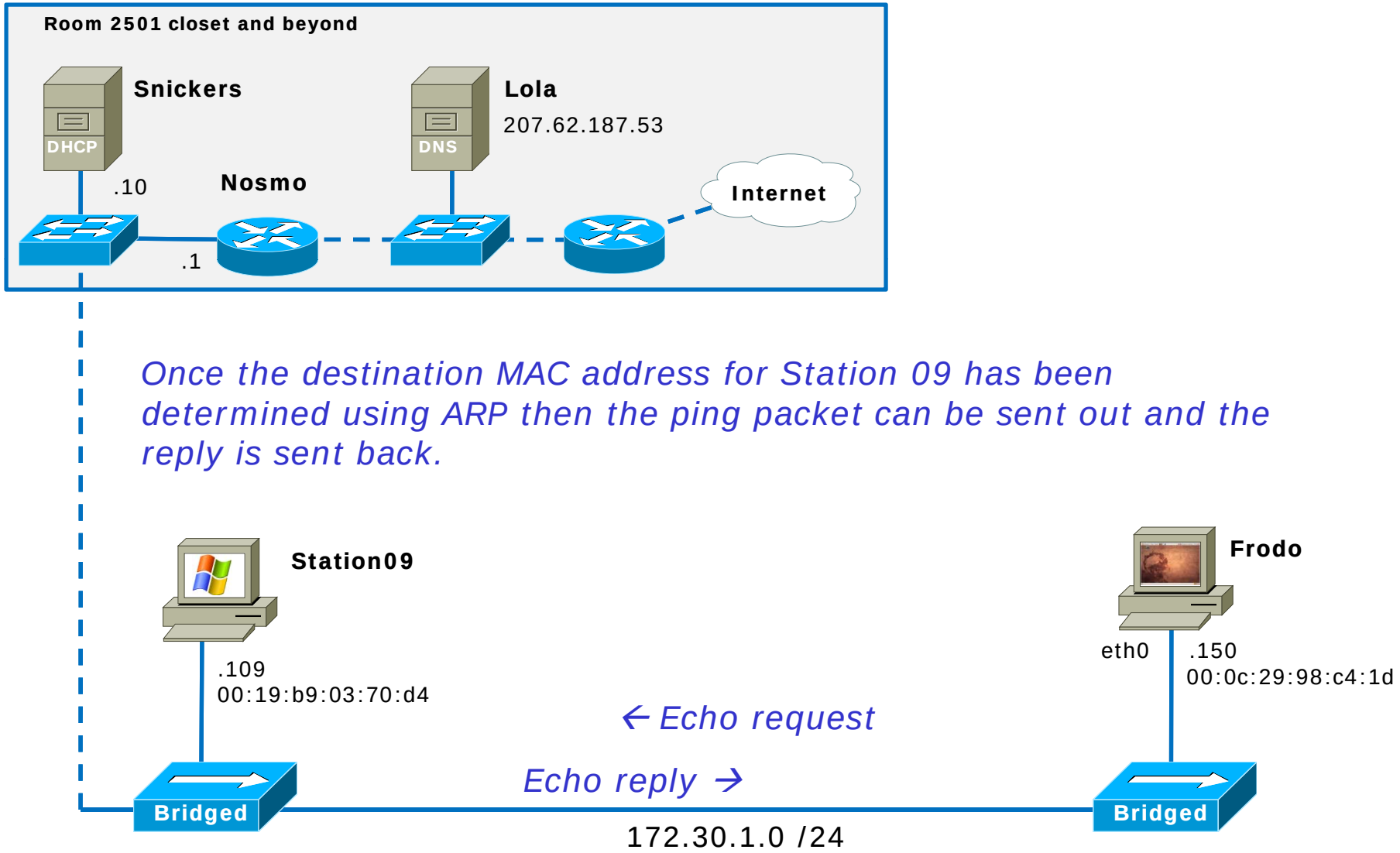


ARP Example - Frodo pings Station09

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



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 if for the router

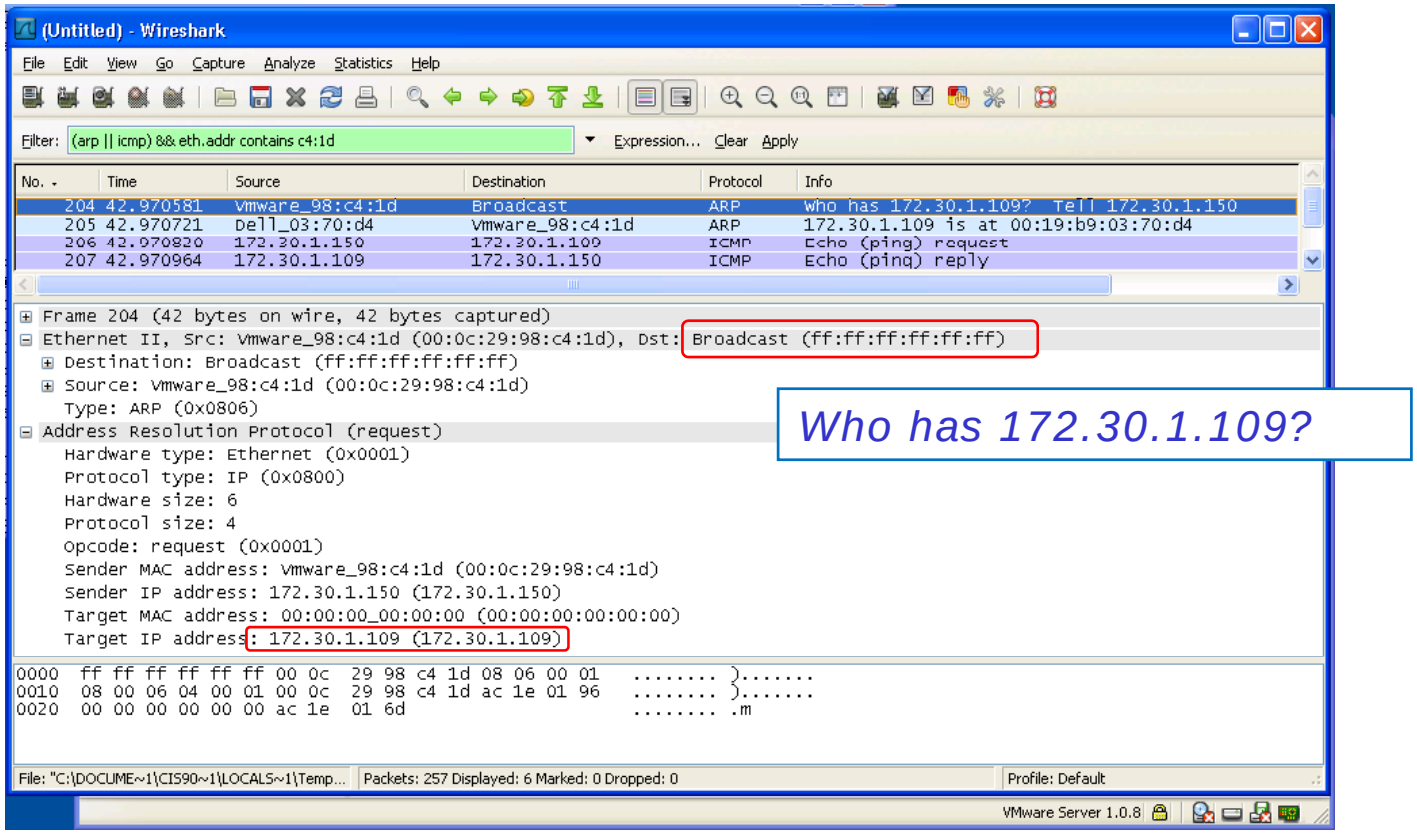
```
root@frodo:~# ping -c 1 172.30.1.109
PING 172.30.1.109 (172.30.1.109) 56(84) bytes of data.
64 bytes from 172.30.1.109: icmp_seq=1 ttl=128 time=3.71 ms
< snipped >
```

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
```

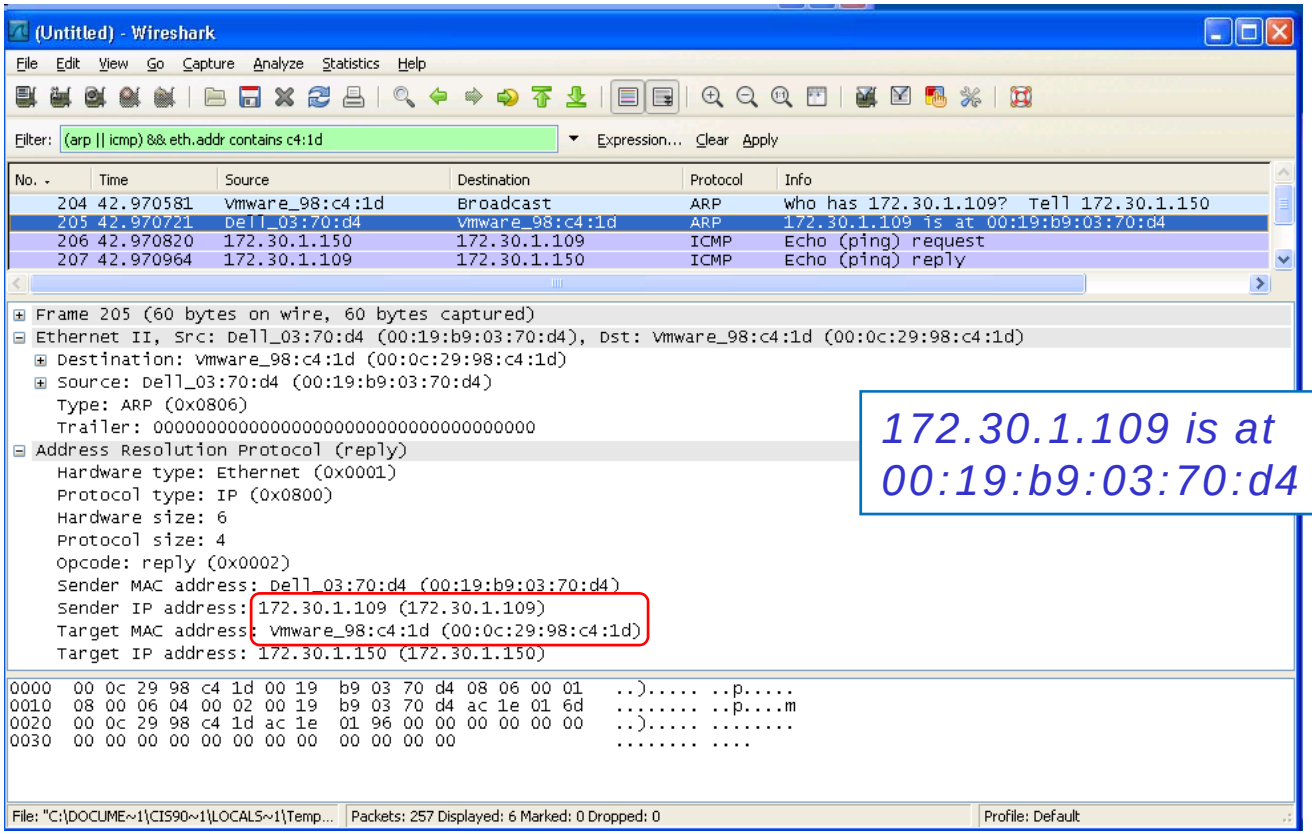
The new MAC/IP pair for Station 09 has been added to the ARP cache

ARP Example - Frodo pings Station09



Frodo's ARP request is a broadcast. Every NIC on the subnet will hear it and check to see if the requested IP address belongs to them.

ARP Example - Frodo pings Station09



Station09's ARP reply sent as a unicast directly back to Frodo.

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)*

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*

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 = still reachable but needs to be verified

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
  192.168.0.20          00-1e-65-68-ab-3a     dynamic
  192.168.0.23          00-13-46-77-eb-4b     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

More examples – make a permanent entry

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 command

More examples – populate 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 cache

populating the arp cache – via 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
```

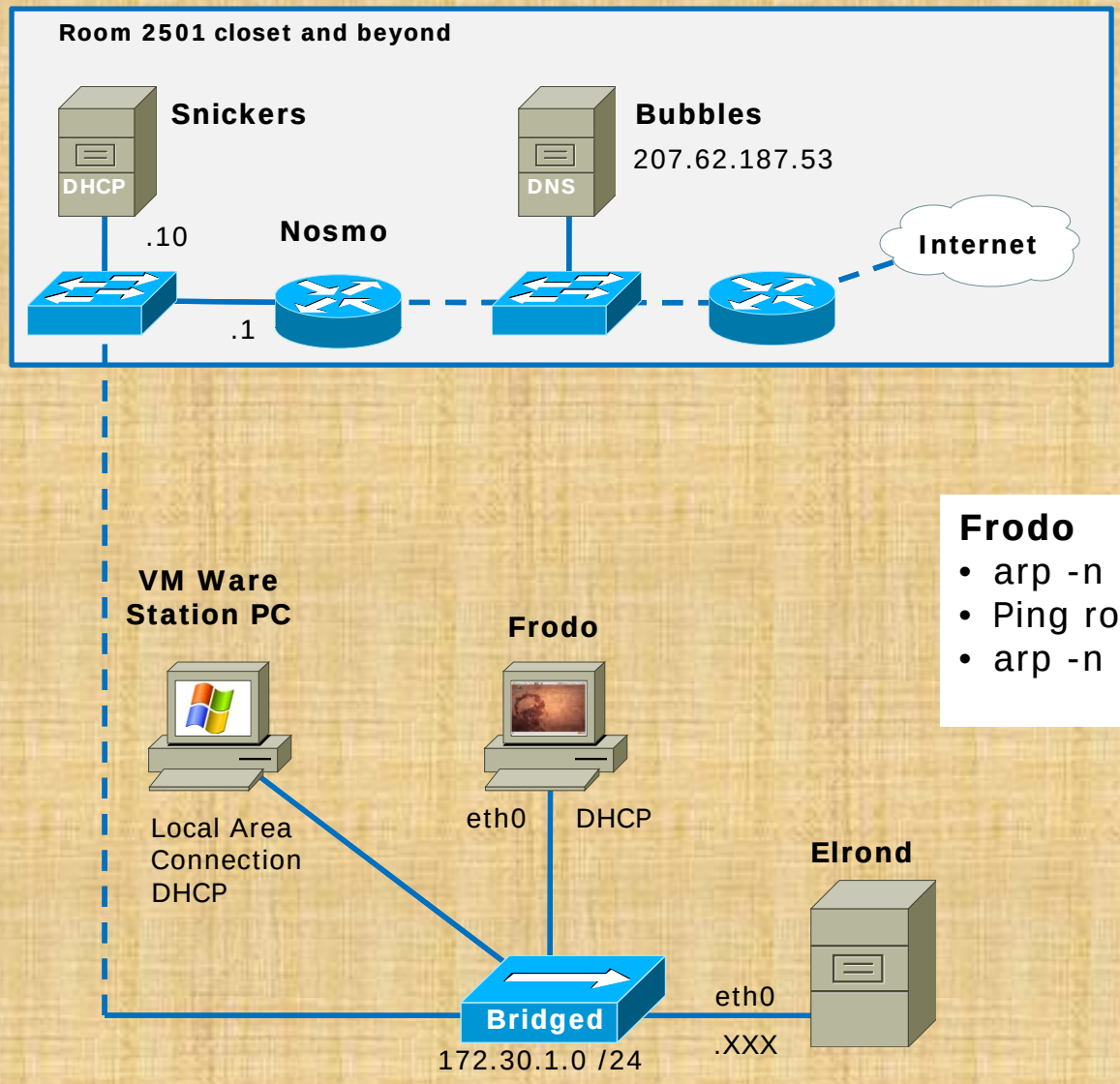
```
root@frodo:~# vi /etc/ethers
root@frodo:~# cat /etc/ethers
172.30.1.1      00:b0:64:53:42:01
172.30.1.10     00:90:27:76:97:ab

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 and saves them in a file
- Must specify an existing database log file: arp.dat
- Emails root as pairs are found
- Great way to inventory MAC addresses or monitor for fraudulent activity

arpwatch

Collect MAC / IP pairs

Centos 5.2

if needed: **yum install arpwatch**

service arpwatch start

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

service arpwatch restart

```
[root@elrond ~]# cat /var/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
0:19:b9:3:39:d1         172.30.1.104    1234303583
0:19:b9:3:71:7          172.30.1.101    1234304181
0:c:29:98:c4:1d         172.30.1.150    1234303456
0:c:29:99:bd:c0         172.30.1.151    1234303460
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
[root@elrond ~]#
```

arpwatch

New pairs are emailed

Centos 5.2

```
[root@elrond ~]# mail
Mail version 8.1 6/6/93.  Type ?
"/var/spool/mail/root": 34 messa
>N  1 logwatch@legolas.riv  Mon
 N  2 logwatch@legolas.riv  Mon
 N  3 logwatch@legolas.loc  Tue
 N  4 logwatch@legolas.loc  Wed
 N  5 logwatch@elrond.loca  Wed
 N  6 root@elrond.localdom  Tue
 N  7 root@elrond.localdom  Tue
 N  8 root@elrond.localdom  Tue
 N  9 root@elrond.localdom  Tue
 N 10 root@elrond.localdom  Tue
 N 11 root@elrond.localdom  Tue
 N 12 root@elrond.localdom  Tue
 N 13 root@elrond.localdom  Tue
 N 14 root@elrond.localdom  Tue Feb 10 14:00  20/832   "new station"
 N 15 root@elrond.localdom  Tue Feb 10 14:00  20/827   "new station"
 N 16 root@elrond.localdom  Tue Feb 10 14:00  20/832   "new station"
 N 17 root@elrond.localdom  Tue Feb 10 14:00  20/824   "new station"
 N 18 root@elrond.localdom  Tue Feb 10 14:00  20/824   "new station"
 N 19 root@elrond.localdom  Tue Feb 10 14:00  20/825   "new station"
 N 20 root@elrond.localdom  Tue Feb 10 14:00  20/823   "new station"
&
```

```
Message 14:
From pcap@elrond.localdomain  Tue Feb 10 14:00:08 2009
Date: Tue, 10 Feb 2009 14:00:08 -0800
From: root@elrond.localdomain (Arpwatch)
To: root@elrond.localdomain
Subject: new station

            hostname: <unknown>
          ip address: 172.30.1.10
    ethernet address: 0:90:27:76:97:ab
     ethernet vendor: INTEL CORPORATION
           timestamp: Tuesday, February 10, 2009 14:00:08 -0800

&
```

arpwatch

Collect MAC / IP pairs

Ubuntu 8.10

Check if installed: **dpkg –l | grep arpwatch**
Intall: **apt-get install arpwatch**

Take defaults:
Listening address: 0.0.0.0
Admin username: Administrator
External authentication: no

Start service: **/etc/init.d/arpwatch start**

The collection starts now. The arp.dat file is not updated till arpwatch is restarted

Restart service: **/etc/init.d/arpwatch restart**

View pairings **cat /var/lib/arpwatch/arp.dat**

arpwatch

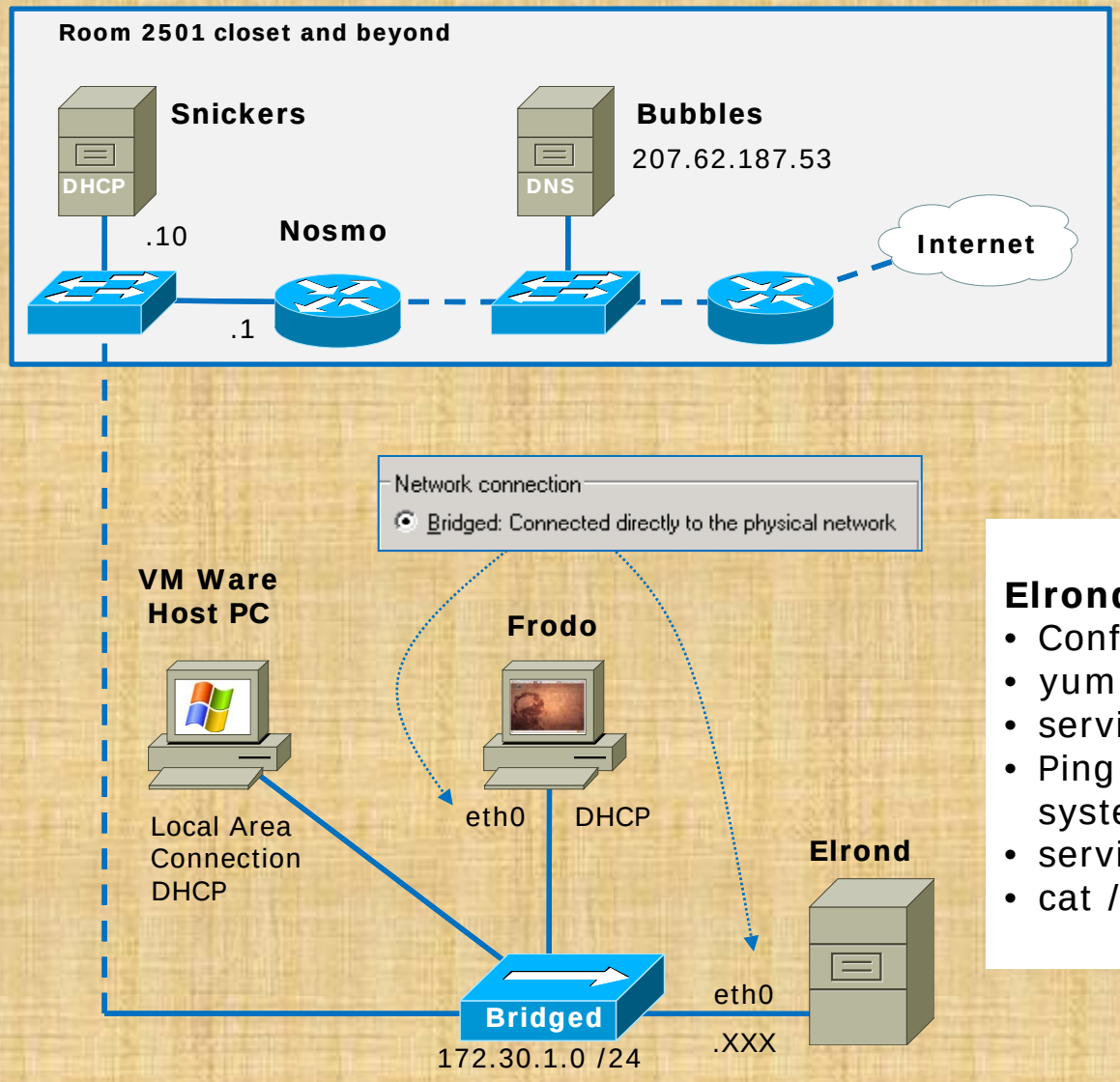
Collect MAC / IP pairs

Ubuntu 8.10

View pairs: **cat /var/lib/arpwatch/arp.dat**

```
root@frodo:~# cat /var/lib/arpwatch/arp.dat
0:90:27:76:97:ab        172.30.1.10         1234570859
0:c:29:46:5:73          172.30.1.153        1234571003      frodo
0:b0:64:53:42:1         172.30.1.1          1234570458
0:18:8b:28:ac:50        172.30.1.121        1234570939              eth0
0:19:b9:3:39:d1         172.30.1.104        1234571003              eth0
0:19:b9:3:71:7          172.30.1.101        1234570607              eth0
0:c:29:a4:83:bc         172.30.1.126        1234570238      elrond  eth0
0:19:b9:3:71:ed         172.30.1.107        1234570408              eth0
0:18:8b:28:ac:ca        172.30.1.108        1234570414              eth0
0:19:b9:3:71:3a         172.30.1.114        1234570426              eth0
0:18:8b:28:ac:9f        172.30.1.116        1234570432              eth0
0:19:b9:3:71:0          172.30.1.110        1234570448              eth0
0:19:b9:3:70:d4         172.30.1.109        1234570449              eth0
0:18:8b:28:a2:68        172.30.1.119        1234570459              eth0
0:19:b9:3:71:b5         172.30.1.118        1234570465              eth0
0:c:29:99:bd:c0         172.30.1.151        1234570470      frodo-2 eth0
root@frodo:~#
```

Class Exercise – Setting up arpwatch on Elrond



Elrond

- Configure eth0 (previous exercise)
- yum install arpwatch
- service arpwatch start
- Ping some other 172.30.1.1xx systems
- service arpwatch restart
- cat /var/arpwatch/arp.dat

Viewing Packets

Viewing Network Packets

Some sniffer options:

- Use tcpdump command on the Linux system
- Run Wireshark on the Windows VMware station (172.30.1.0 /24 network)
- Run the Sniffer VM (has Wireshark installed)
- 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 on Elrond

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

```
[root@elrond ~]# tcpdump 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
11:20:31.850225 IP 172.30.1.126 > 172.30.1.105: ICMP echo request, id
  52755, seq 1, length 64
11:20:31.856842 arp who-has 172.30.1.126 tell 172.30.1.105
11:20:31.857217 arp reply 172.30.1.126 is-at 00:0c:29:a4:83:bc (oui
  Unknown)
11:20:31.857736 IP 172.30.1.105 > 172.30.1.126: ICMP echo reply, id
  52755, seq 1, length 64
```

Ctrl-C to end

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

On another terminal we do a single ping of 172.30.1.105

Viewing Network Packets
tcpdump on Elrond

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

Provide link–level header
Buffer stdout
Don't convert addresses to names

```
[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
```

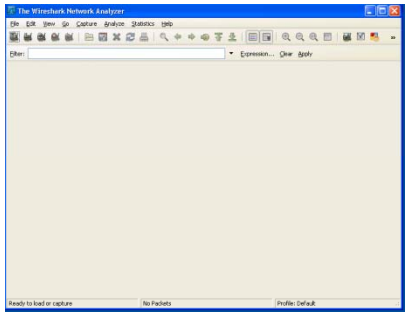
Ctrl-C to end

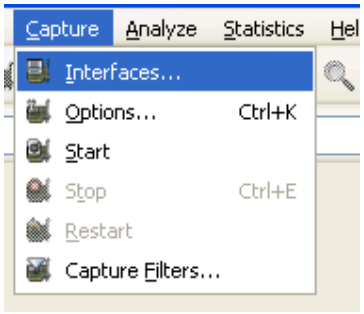
```
4 packets captured
12 packets received by filter
0 packets dropped by kernel
[root@elrond ~]#
```

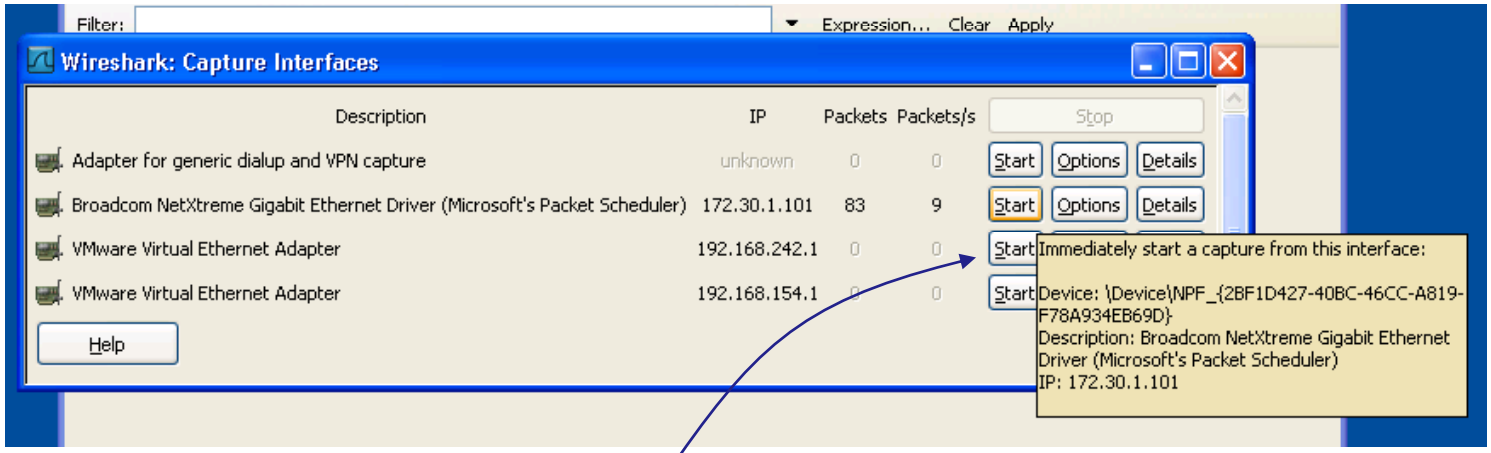
On another terminal we do a single ping of 172.30.1.105

Viewing Network Packets
Wireshark on VMware station

Wireshark

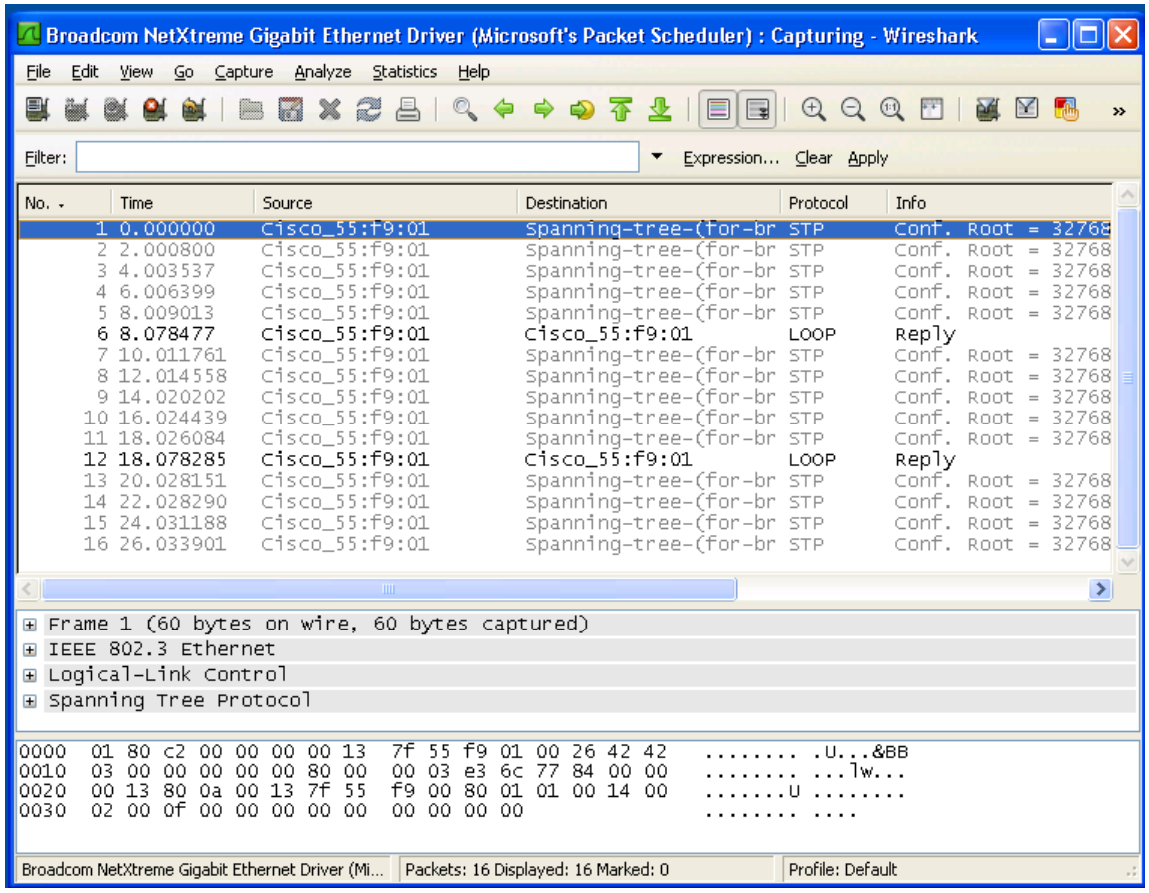






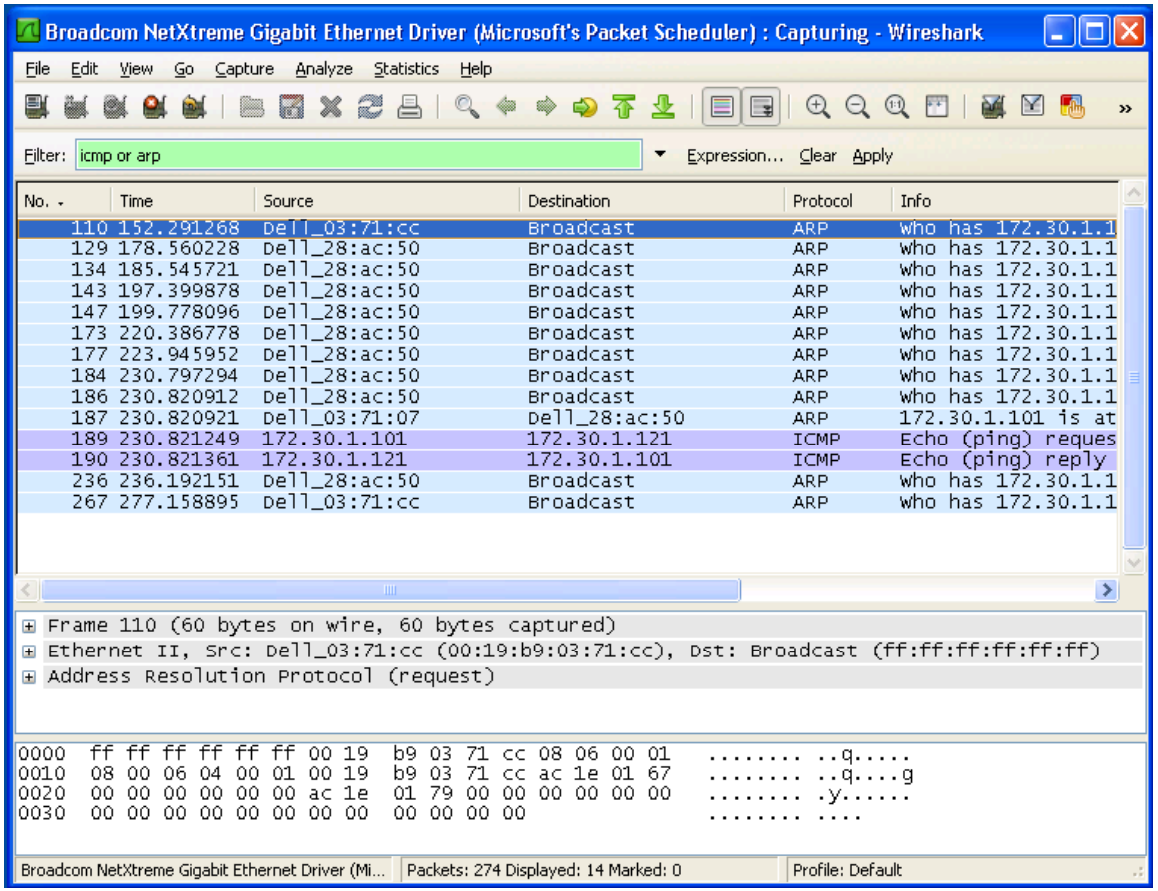
Click on the Start button for the Broadcom NIC interface

Viewing Network Packets
Wireshark



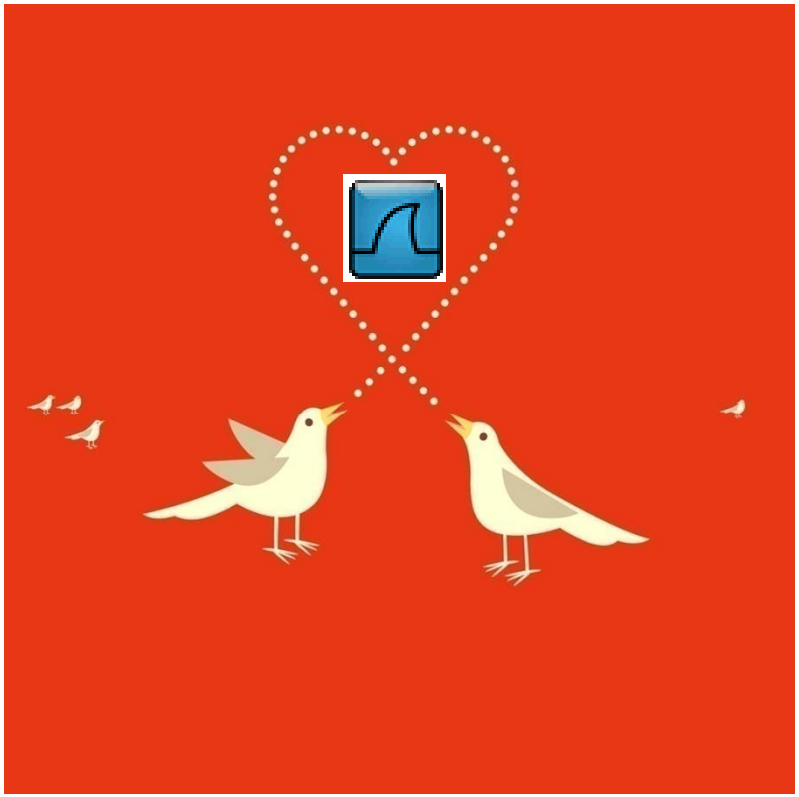
Without any filters set you will see all the packets

Viewing Network Packets
Wireshark



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

Viewing Network Packets
Wireshark

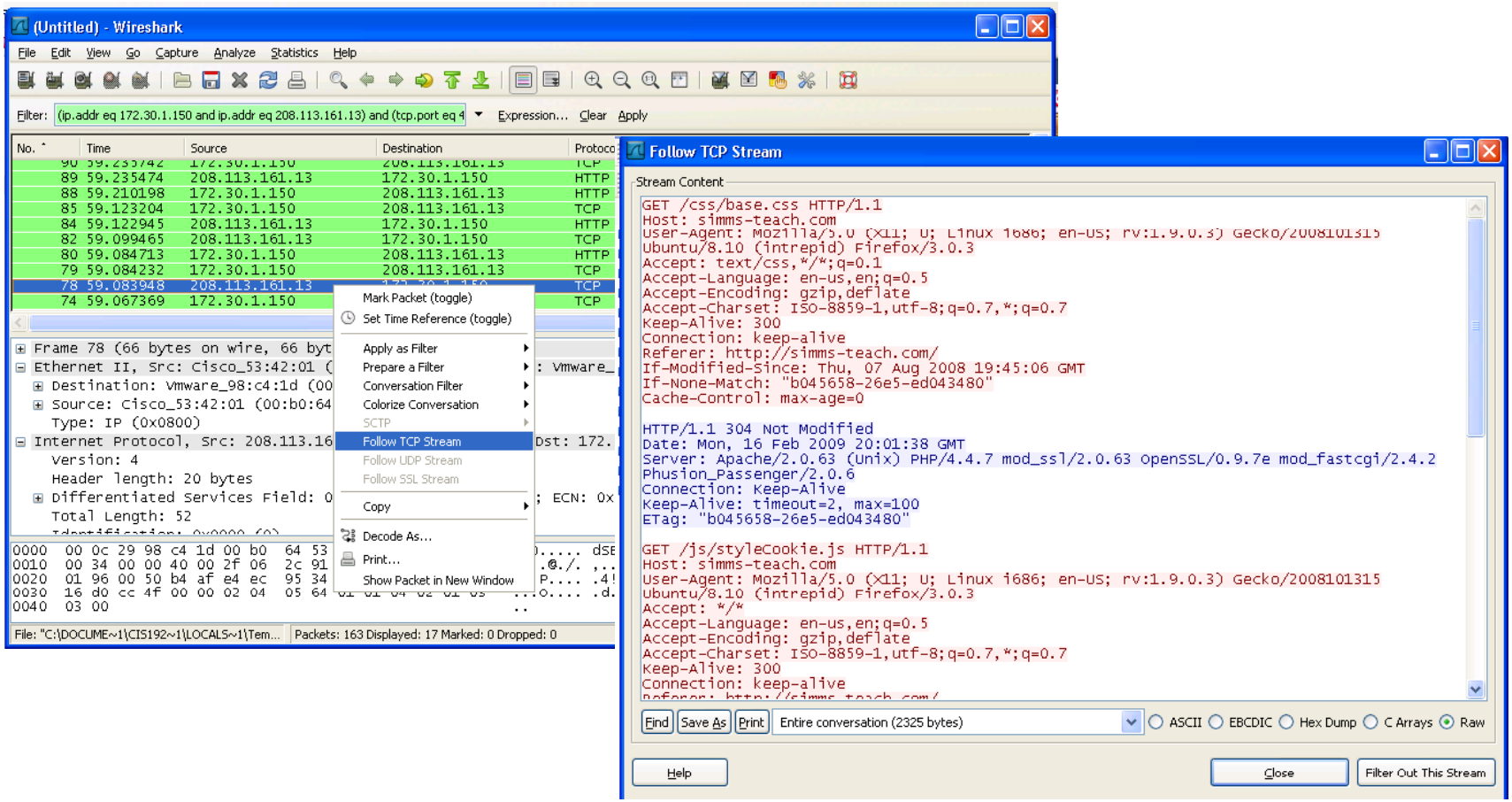


Some really nice options:

- Follow TCP stream
- Prepare a filter

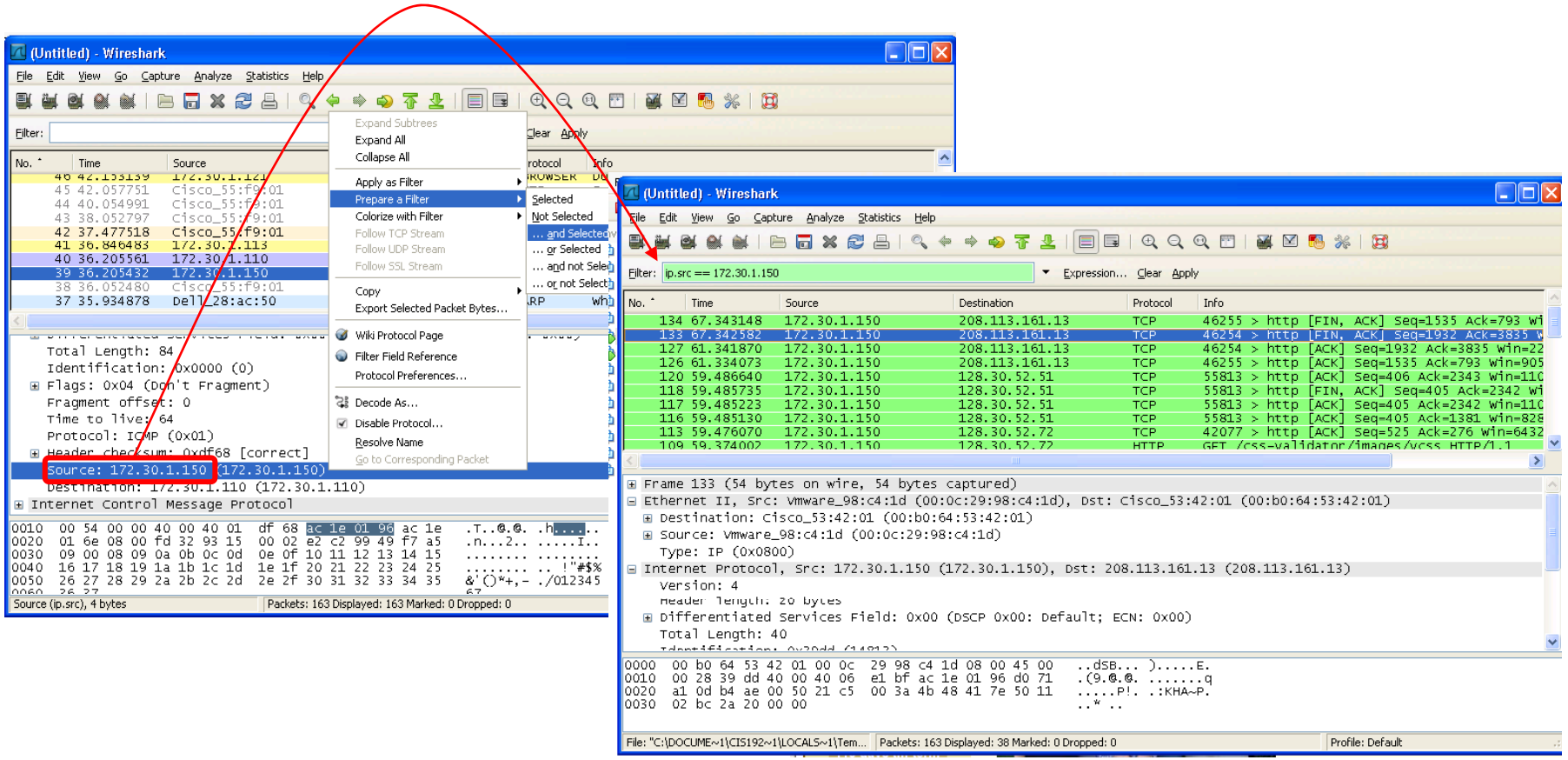
Use icmp or arp as a display filter to view only those packets

Viewing Network Packets
Wireshark – Follow TCP Stream



Following the TCP stream of viewing a web page

Viewing Network Packets
Wireshark – Prepare a filter



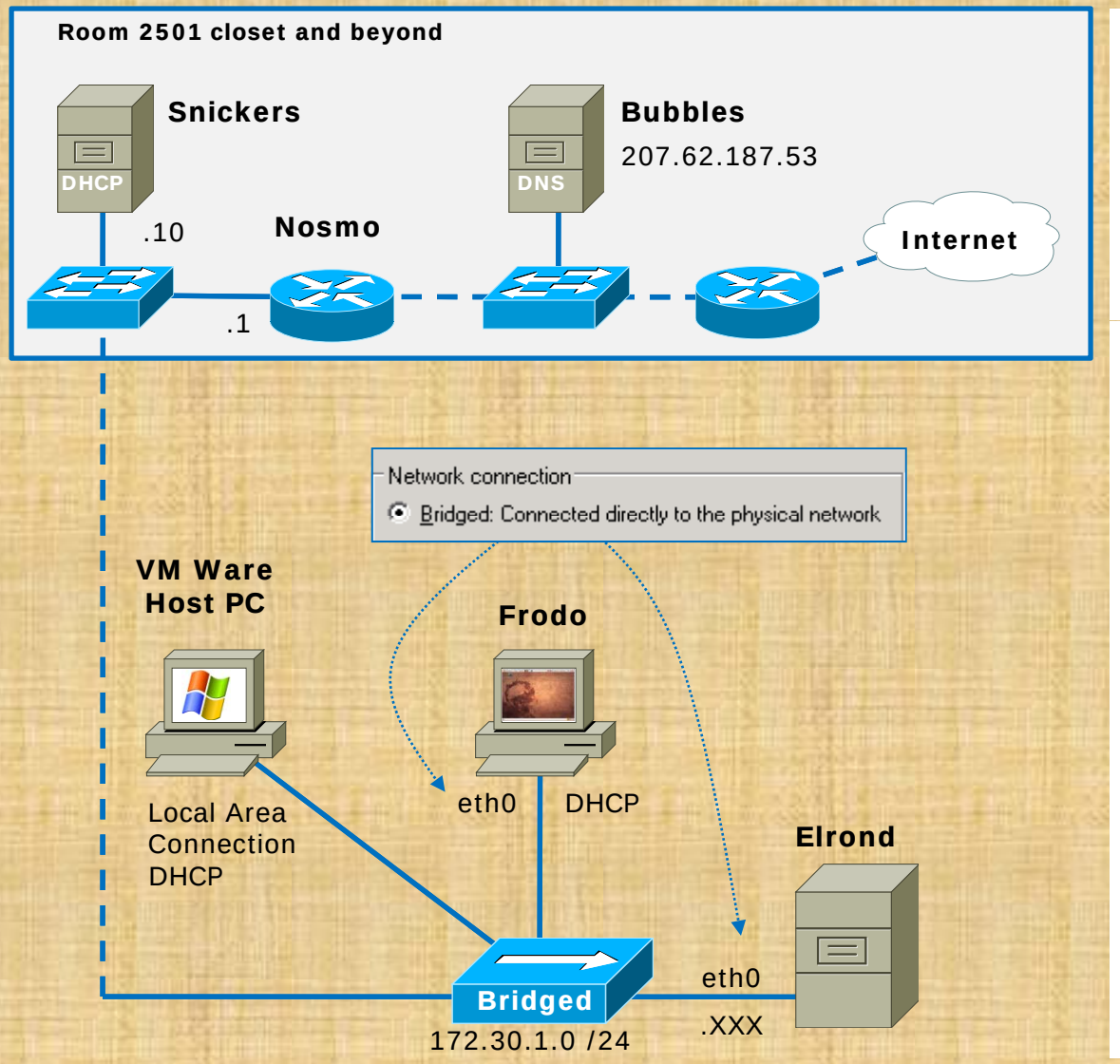
Select the source IP address of a packet and used it to make a display filter to only see packets from that IP address

Viewing Network Packets
Wireshark – 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*

Filter by MAC address, IP address, protocol and many other ways

Class Exercise – View Packets



On Elrond

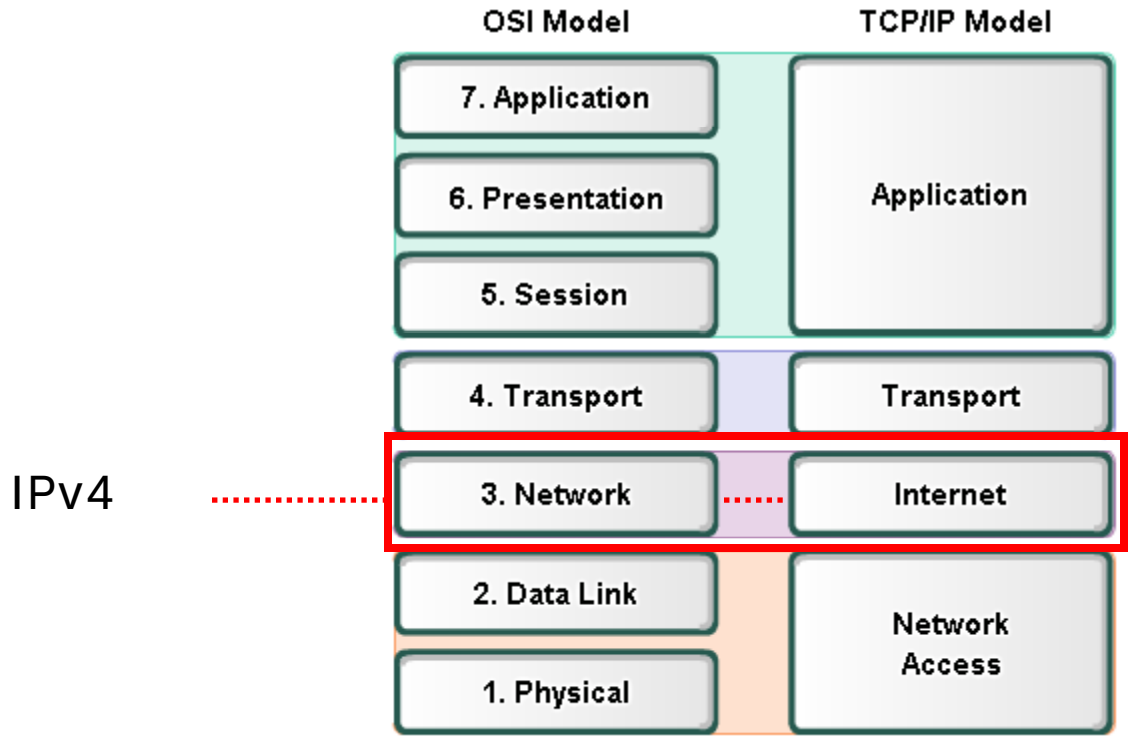
- In one terminal start tcpdump with a filter to view only your Elrond VM's traffic.
- **tcpdump -eln src 172.30.1.1xx or dst 172.30.1.1xx**
- In another terminal start pinging the router.
- Check you are capturing your traffic to and from the router.
- Ctrl-C to end tcpdump and ping

VMware Station Run Wireshark

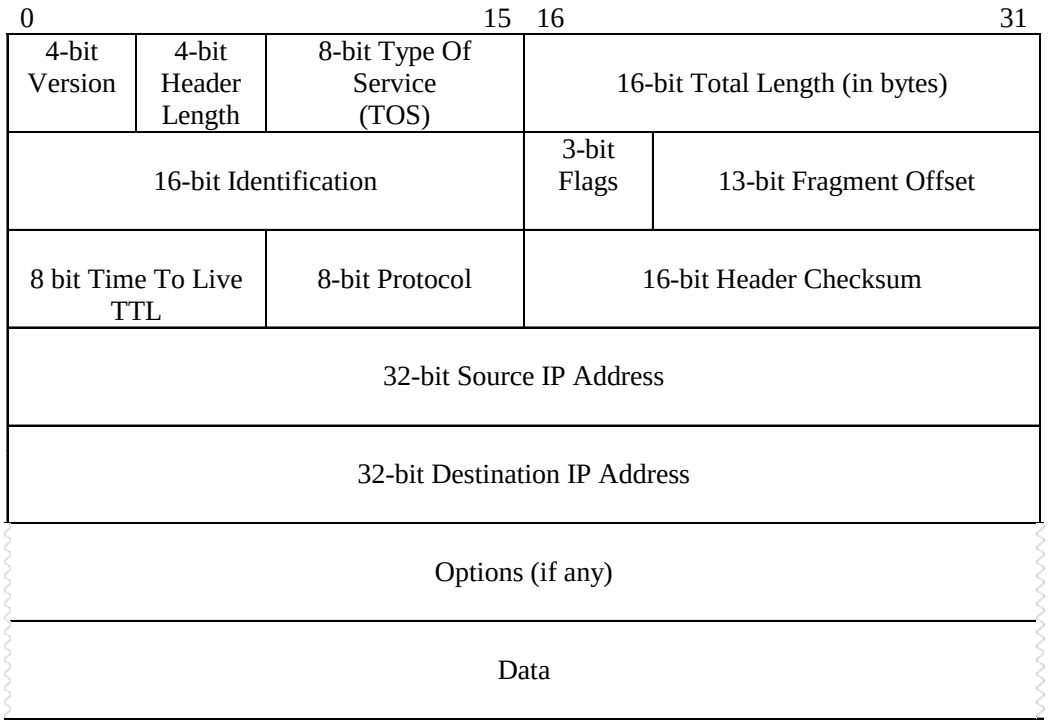
- Start a capture on the real NIC
- Do several pings from Elrond
- View a web site on Frodo
- Stop capture
- Create a display filter to only see ARP packets
- Create a display filter to only see HTTP packets for your Frodo.

Layer 3

Network Layer

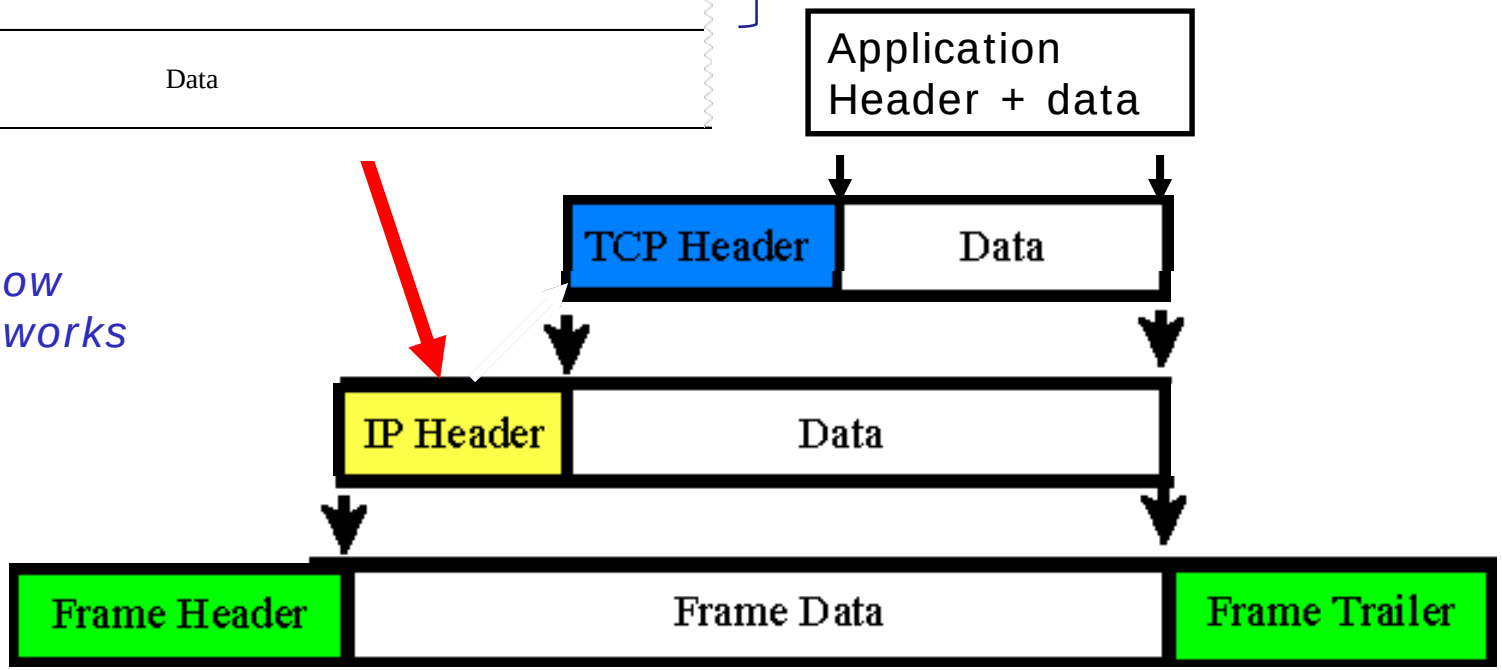


RS: More on Layer 3 tonight

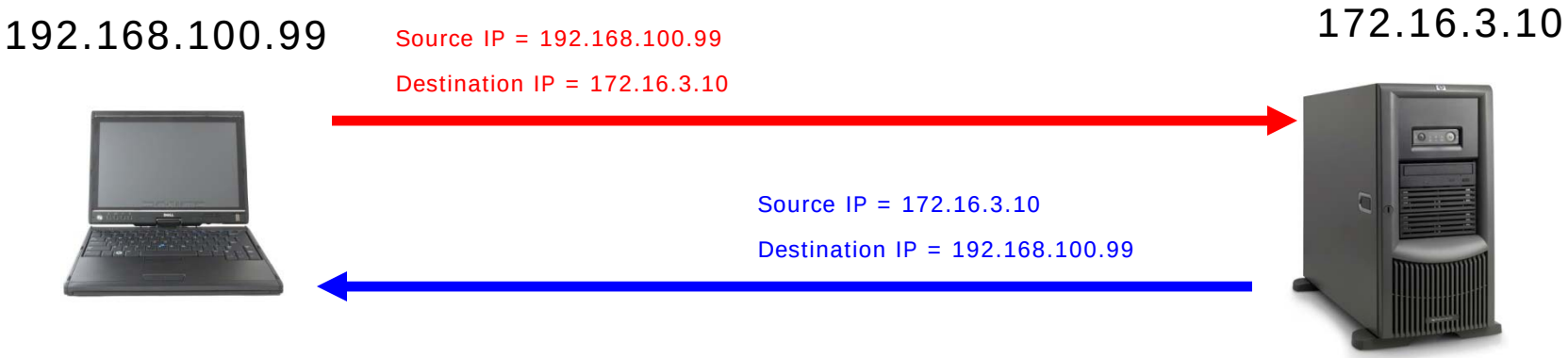


IP Header

RS: showing how encapsulation works



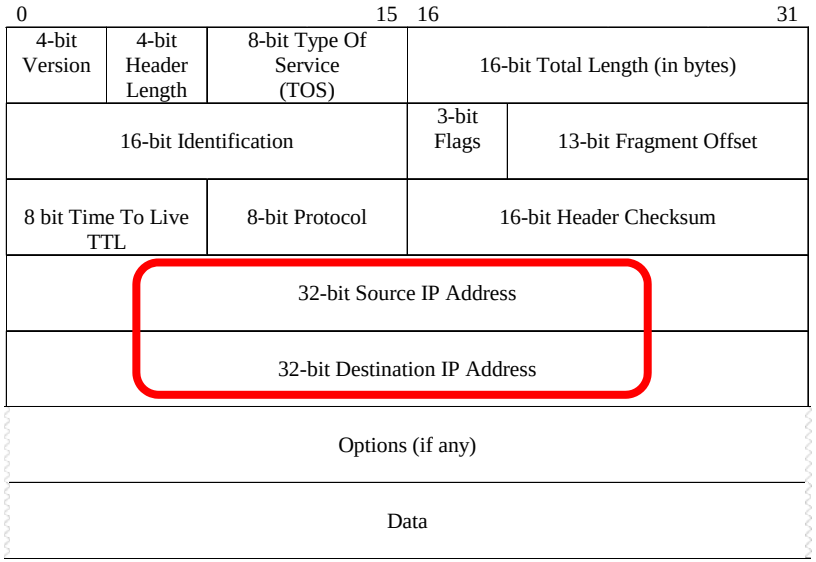
Addressing



- 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.



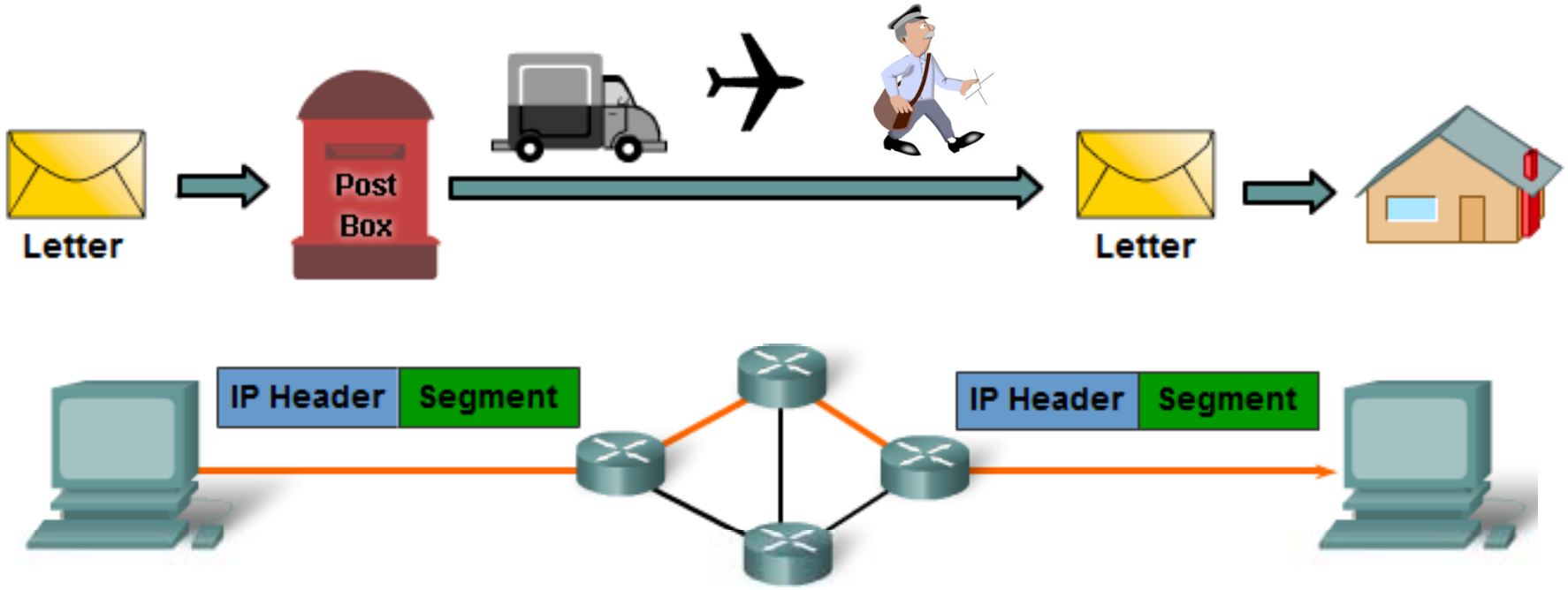
Network Layer Protocols

- 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.

RS: Same for CIS 192

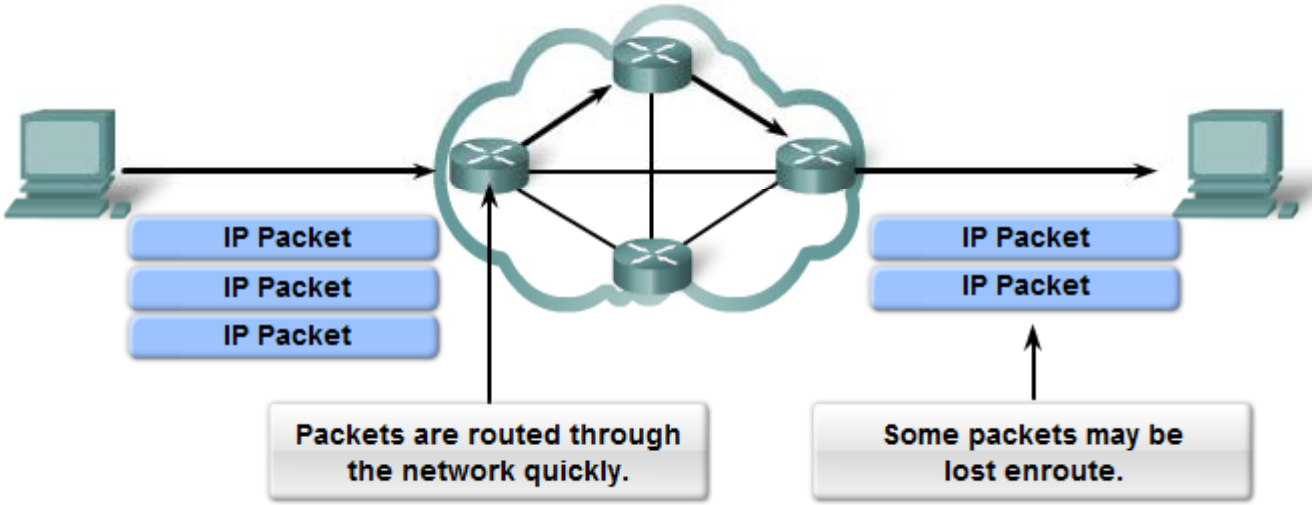
Connectionless



- IP packets are sent without notifying the end host that they are coming.
  - **TCP**: A connection-oriented protocol does requires a connection to be established prior to sending TCP segments.
  - **UDP**: A connectionless protocol does not require a session to be established.

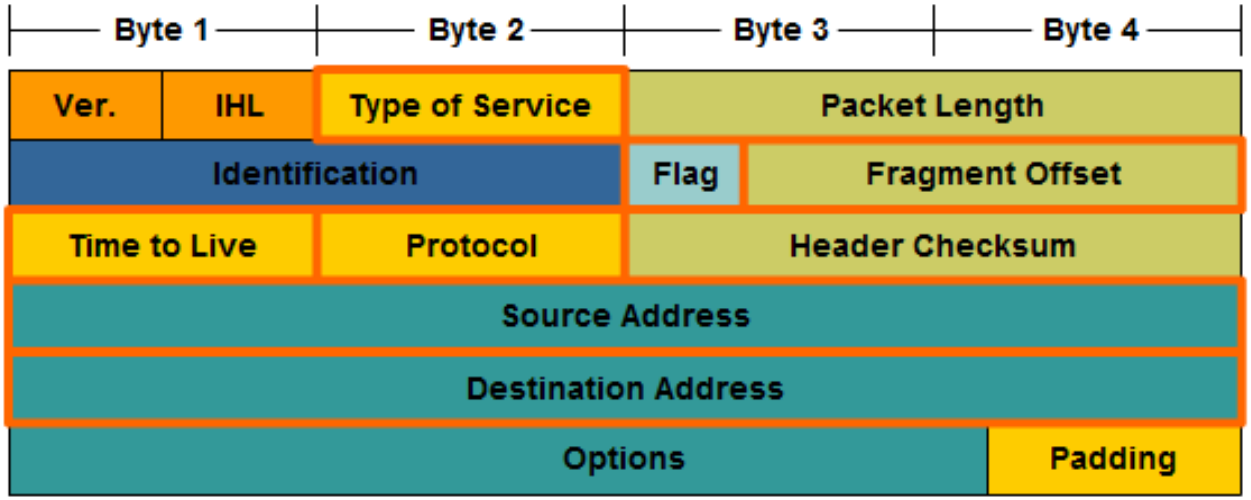
RS: SNMP traps use UDP … they are "fire and forget"

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

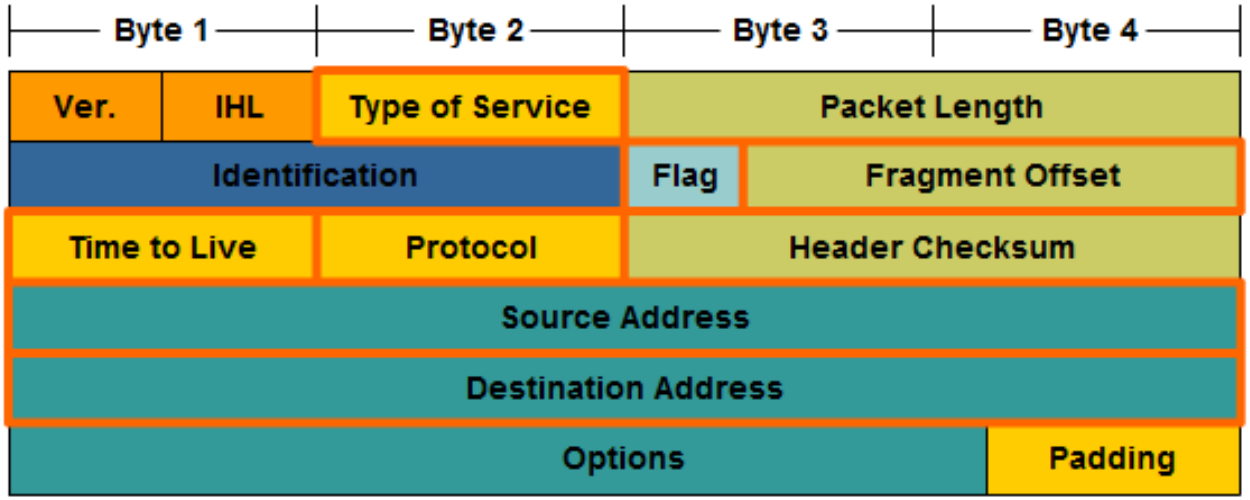


- **IP Destination Address**
  - 32-bit binary value that represents the packet destination Network layer host address.
- **IP Source Address**
  - 32-bit binary value that represents the packet source Network layer host address.

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.

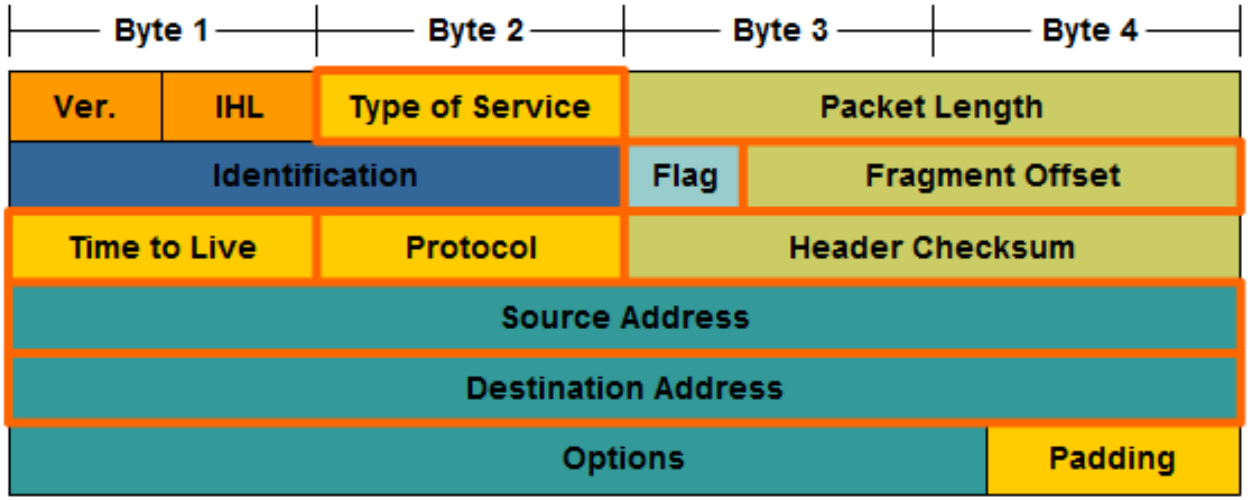


- If the router decrements the TTL field to 0, it will then drop the packet (unless the packet is destined specifically for the router, i.e. ping, telnet, etc.).
- Common operating system TTL values are:
  - UNIX: **255**
  - Linux: **64 or 255** depending upon vendor and version
  - Microsoft Windows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

RS: TTL keeps packets from endlessly wandering about the Internet forever

IP's TTL – Time To Live field

Decrement by 1, if 0 drop the packet.

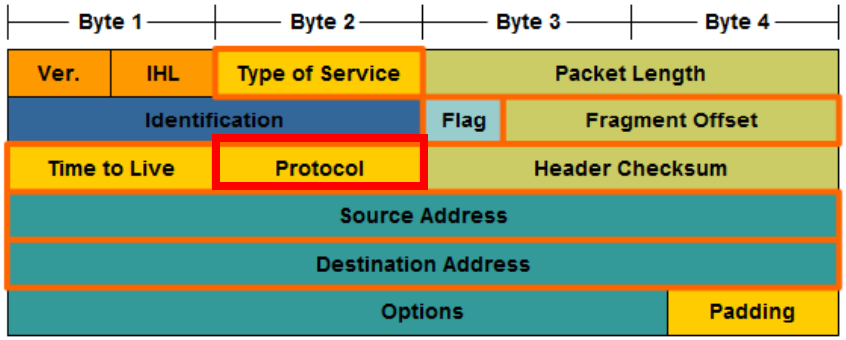


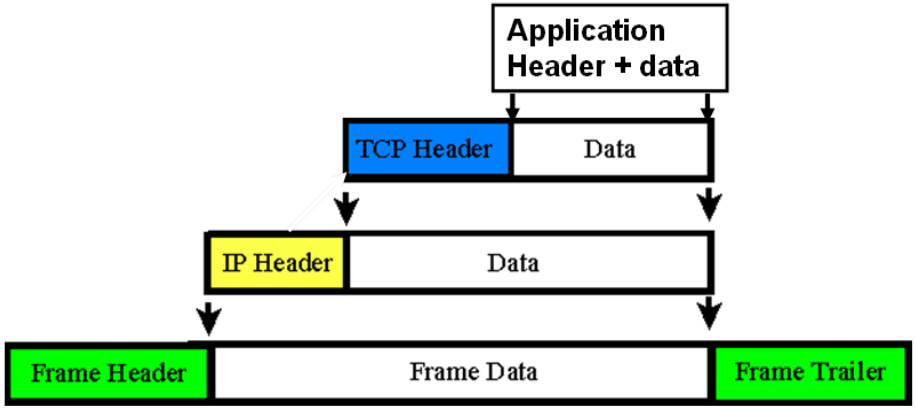
- The idea behind the TTL field is that IP packets can not travel around the Internet forever, from router to router.

- 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

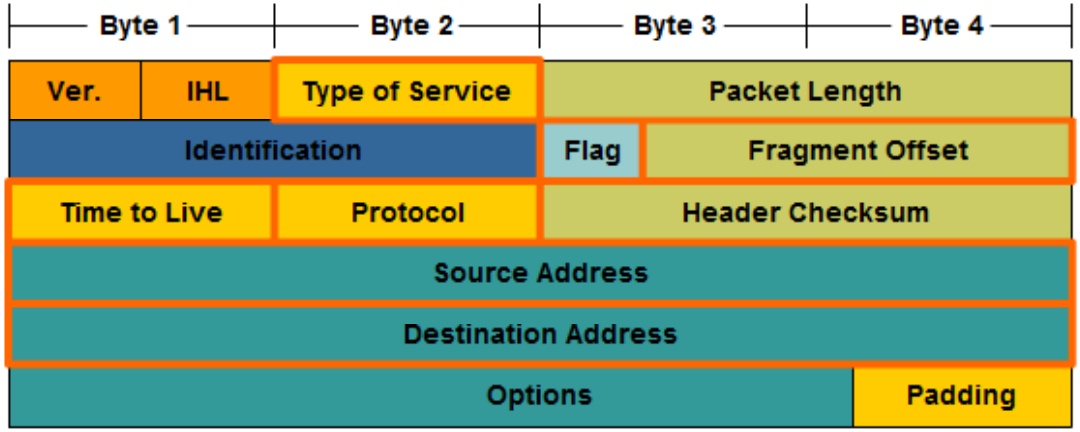




- **Protocol field** enables the Network layer to pass the data to the appropriate upper-layer protocol.
- Example values are:
  - 01 ICMP
  - 06 TCP
  - 17 UDP

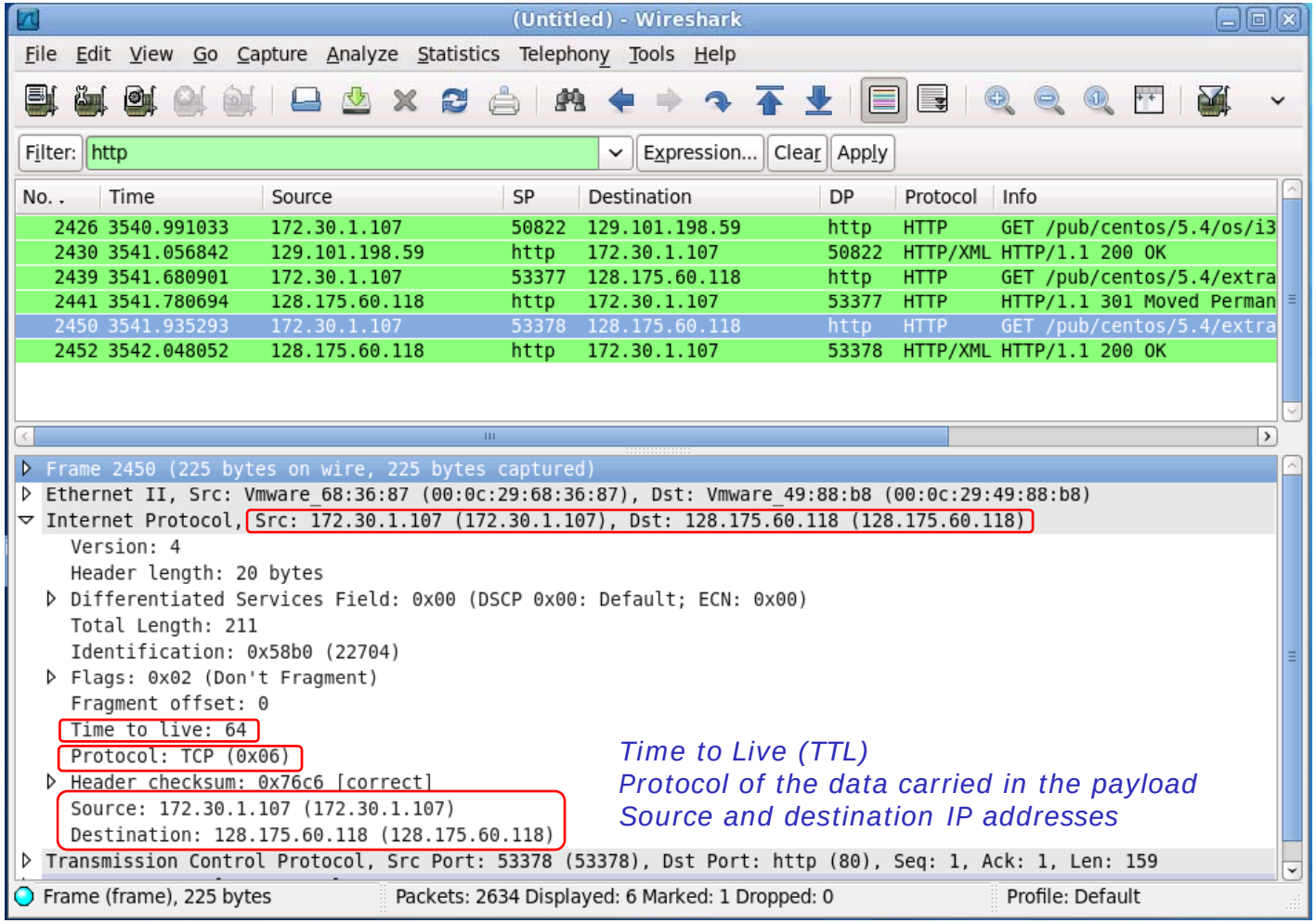
RS: The protocol is used to identify the format of the data payload

Other IPv4 fields



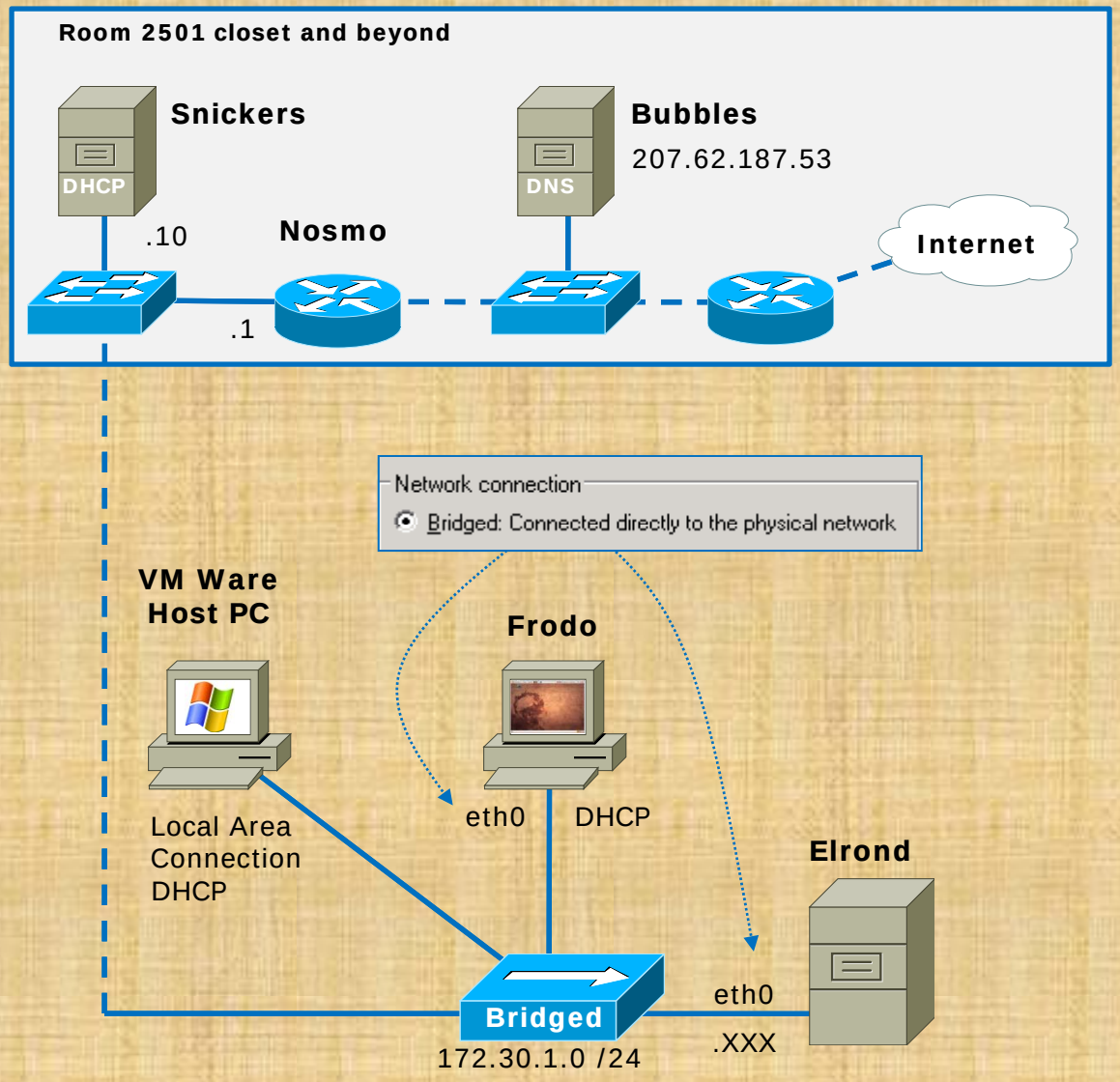
- **Version** - Contains the IP version number (4)
- **Header Length (IHL)** - Specifies the size of the packet header.
- Packet Length - This field gives the entire packet size, including header and data, in bytes.
- **Identification** - This field is primarily used for uniquely identifying fragments 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

Class Exercise – View IP Header with Wireshark

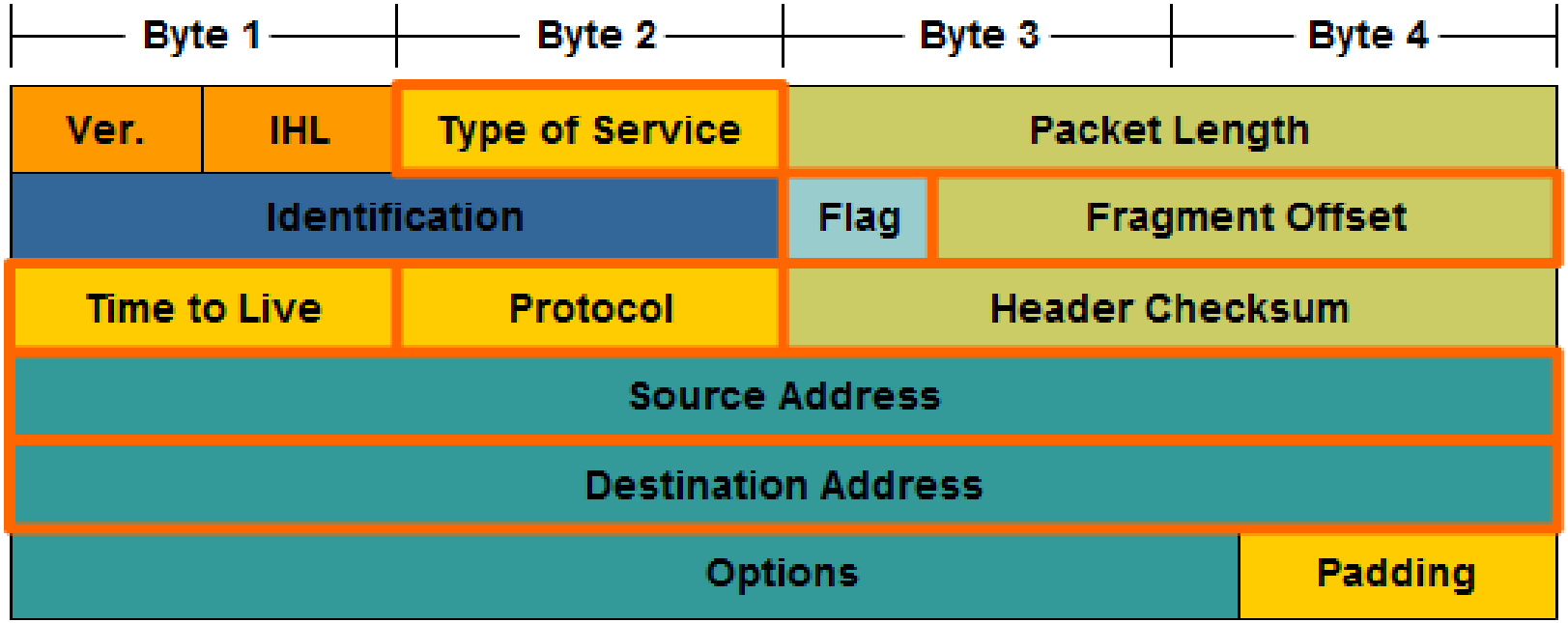


VMware Station
Run Wireshark

- Remove all filters
- Start a capture using the hardware NIC
- Browse a web site on Frodo
- Stop the capture
- Set HTTP filter
- Locate Source and Destination IP addresses
- Locate TTL
- Locate protocol (of its data payload)

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

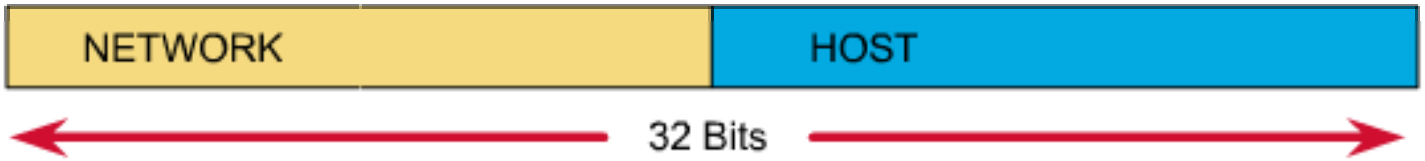
- We use dotted notation (or dotted decimal notation) to represent the value of each byte (octet) of the IP address in decimal.

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

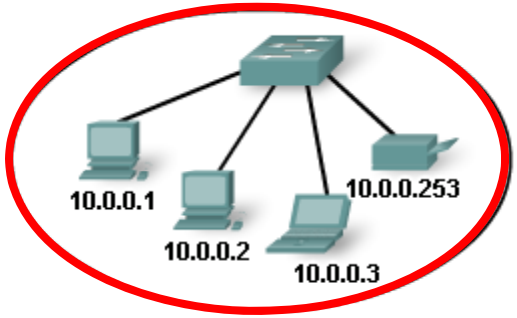
Broadcast Address

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.

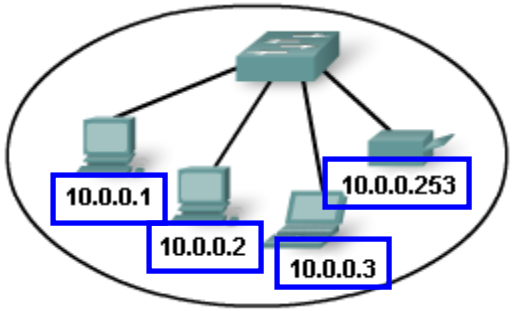
	Network		Host
10	0	0	0
00001010	00000000	00000000	00000000

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

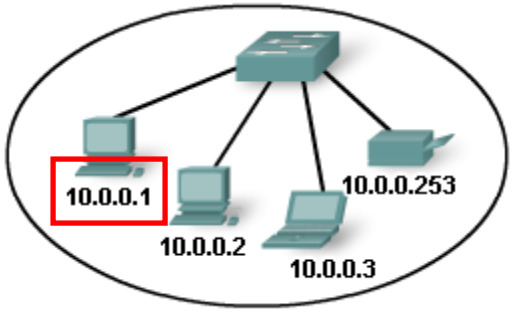
Host Address

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.

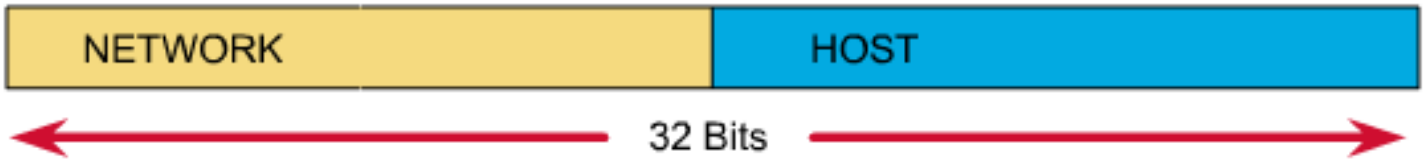
Subnet Mask: 255.255.255.0



- **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

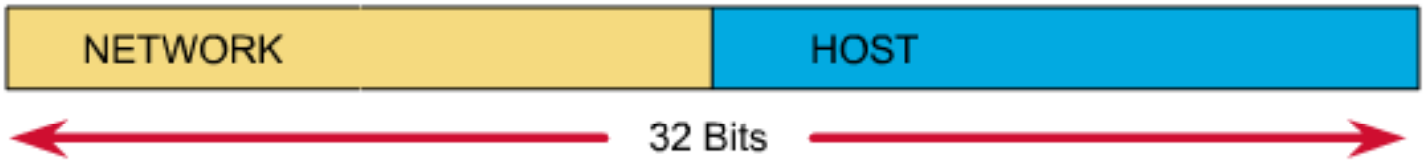


11111111111111110000000000000000

- **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

RS: The mask is a way to specify what portion of the IP address is the network and which portion is for the hosts.

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)

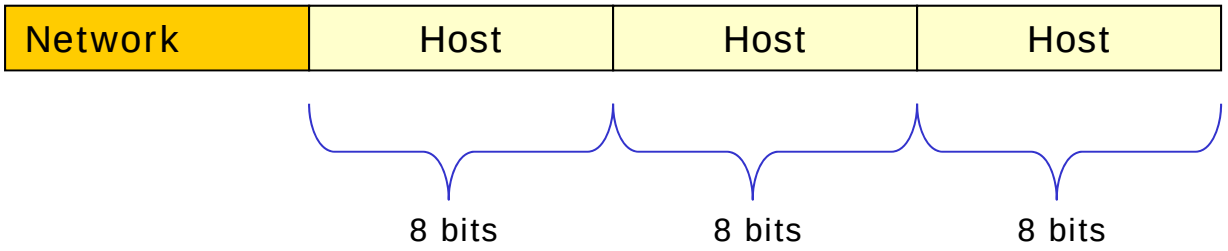
RS: We will use both dotted and slash notations in CIS 192

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”

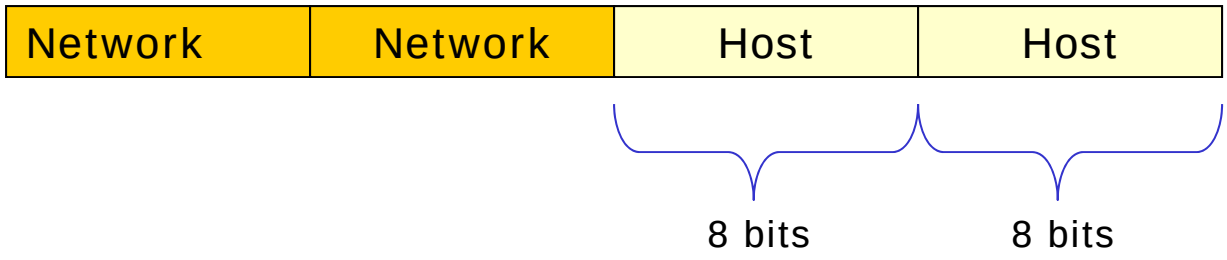
Subnet: 255.0.0.0 (/8)



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

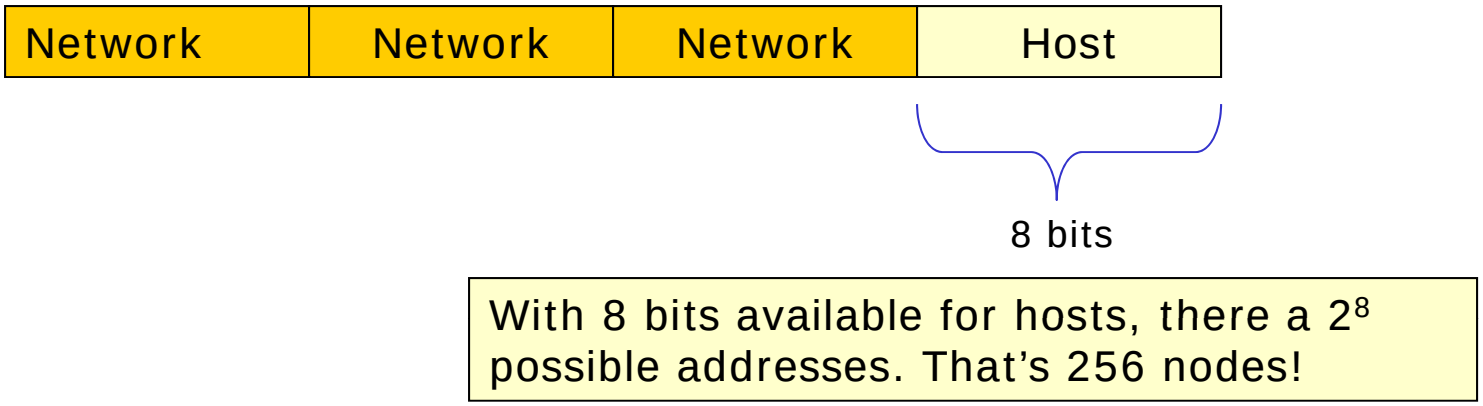
Subnet: 255.255.0.0 (/16)



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.

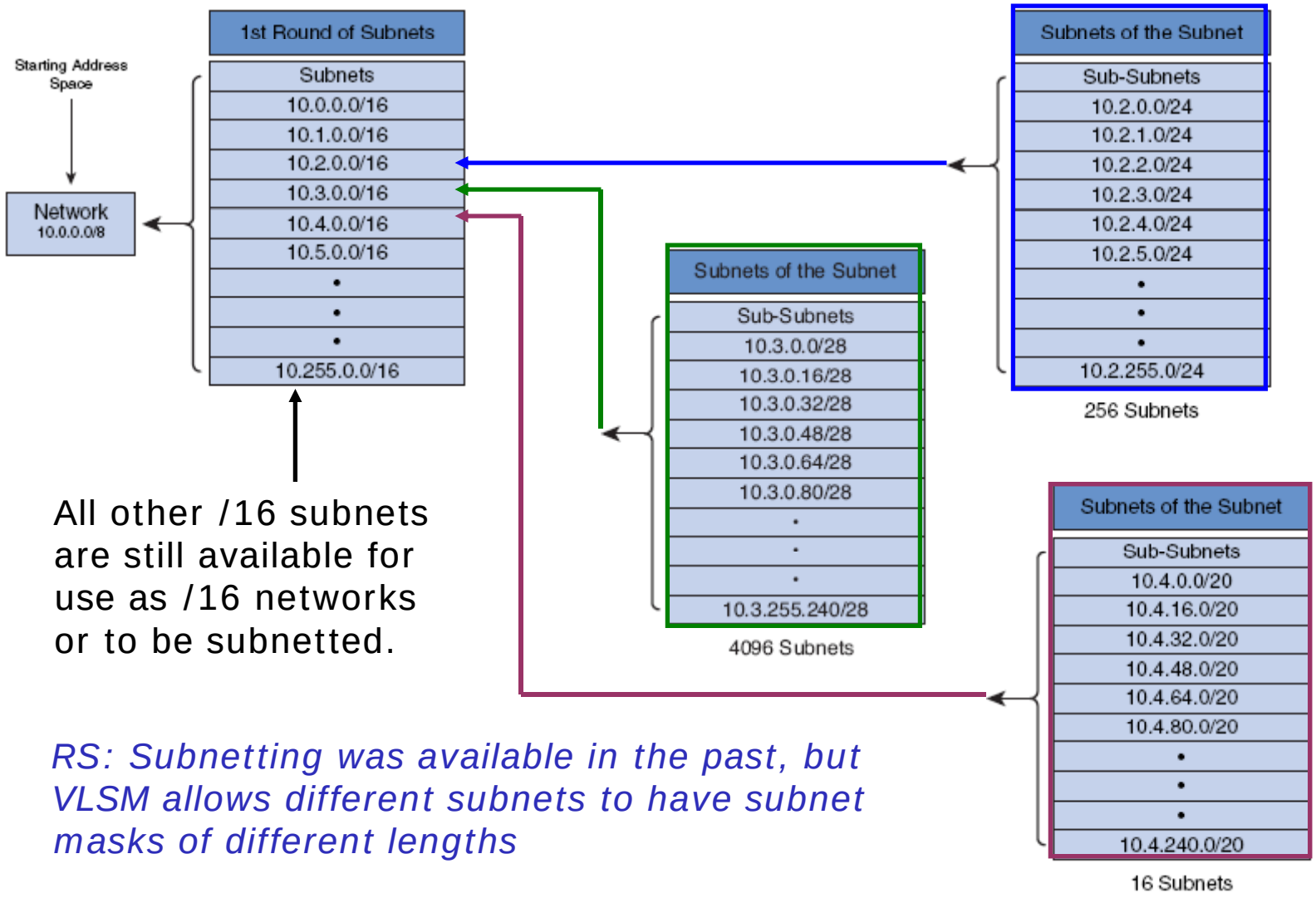
Subnet: 255.255.255.0 (/24)



- 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

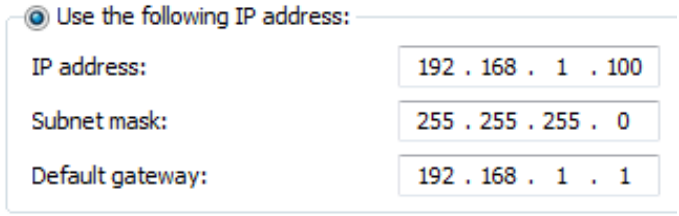
Old days: Address Classes

	1st octet	2nd octet	3rd octet	4th octet
Class A	Network	Host	Host	Host
Class B	Network	Network	Host	Host
Class C	Network	Network	Network	Host

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 -p 15.107.34.45 255.255.248.0
PREFIX=21

[root@elrond ~]# ipcalc -n 15.107.34.45 255.255.248.0
NETWORK=15.107.32.0

[root@elrond ~]# ipcalc -b 172.30.4.101/24
BROADCAST=172.30.4.255

[root@elrond ~]# ipcalc -b 172.30.4.101/16
BROADCAST=172.30.255.255
```

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: Do Q1, Q7

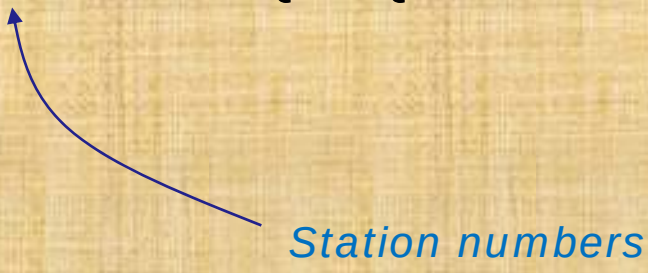
Table 5-8: Do Q2, Q8

Table 9-12: Do Q3, Q9

Table 13-16: Do Q4, Q10

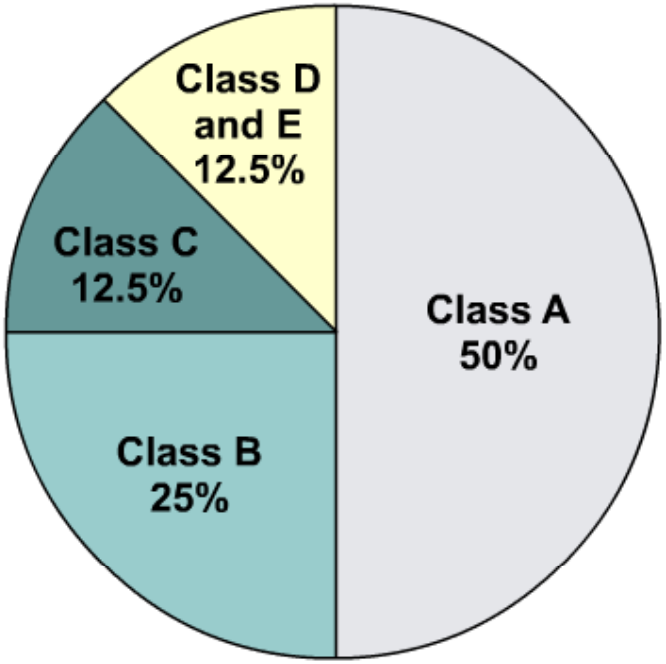
Table 17-20: Do Q5, Q11

Table 21-24: Do Q6, Q12



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

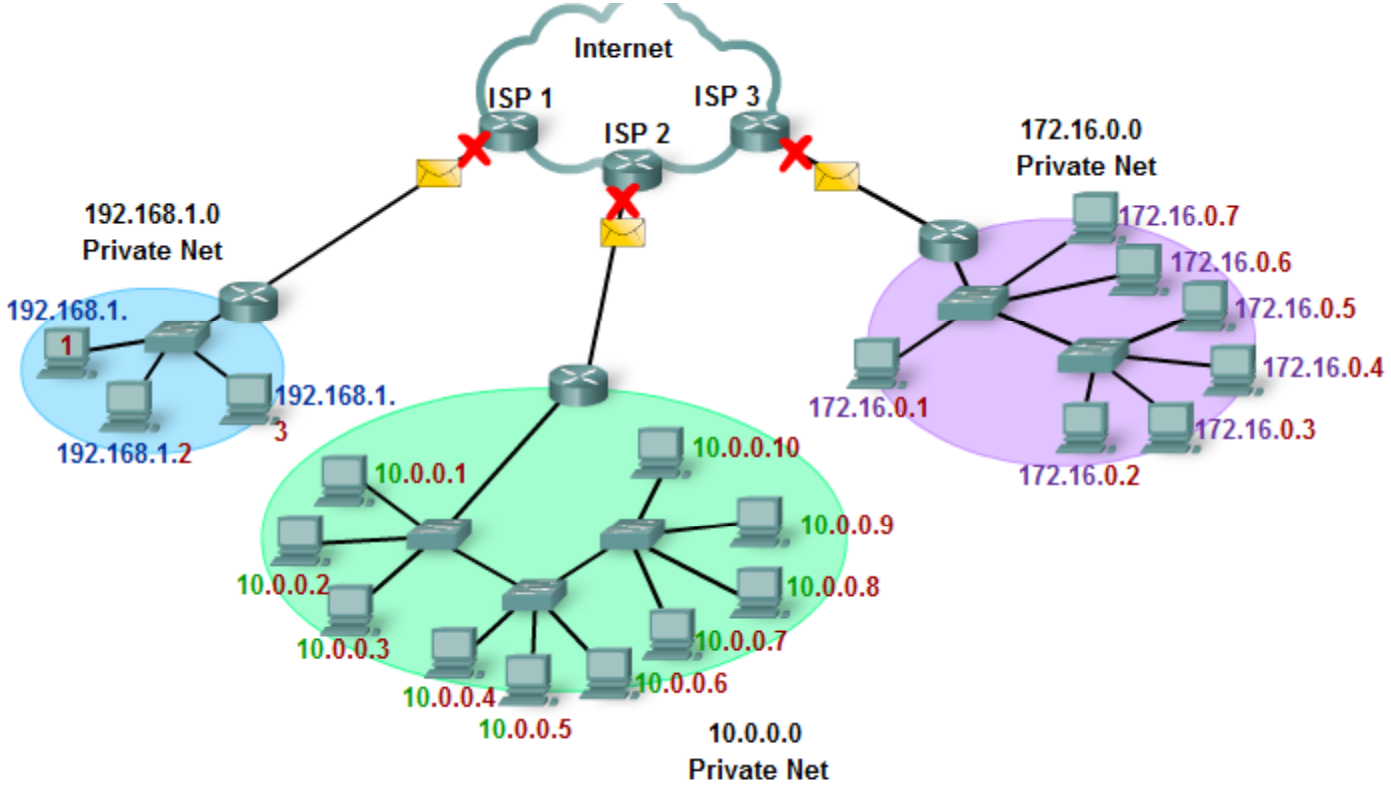
Class	RFC 1918 Internal Address Range	CIDR Prefix
A	10.0.0.0 to 10.255.255.255	10.0.0.0/8
B	172.16.0.0 to 172.31.255.255	172.16.0.0/12
C	192.168.0.0 to 192.168.255.255	192.168.0.0/16

- 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

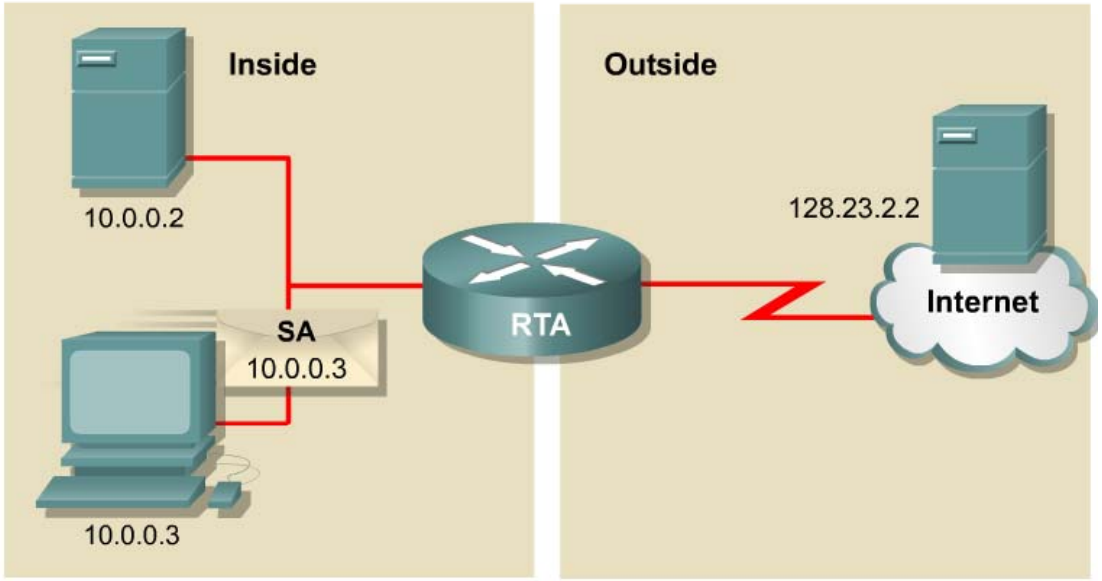
RS: CIDR IP addresses use the / notation

Private IP Addresses



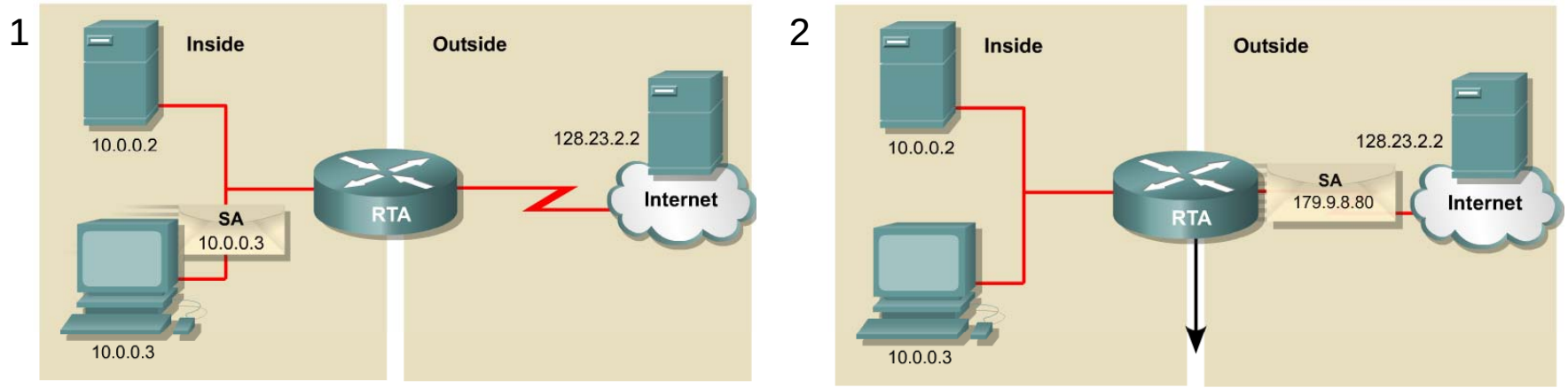
- RFC 1918
  - 10.0.0.0 to
  - 172.16.0.0 to 172.31.255.255 (172.16.0.0 /12)
  - 192.168.0.0 to 192.168.255.255 (192.168.0.0 /16)
- The addresses will not be routed in the Internet
  - Need NAT/PAT (next)
- Should be blocked by your ISP
- Allows for any network to have up to 16,777,216 hosts (/8)

Introducing NAT and PAT

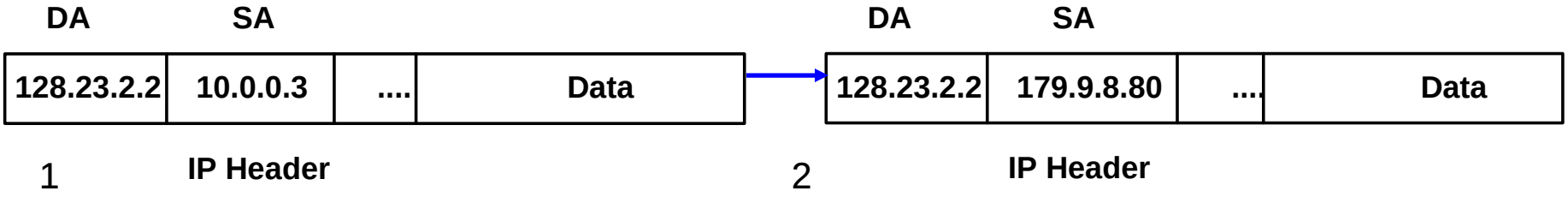


- NAT is designed to conserve IP addresses and enable networks to use private IP addresses on internal networks.
- These private, internal addresses are translated to routable, public addresses.
- IPv4 addresses are almost depleted.
- NAT/PAT has allowed IPv4 to be the predominant network protocol, keeping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

NAT Example

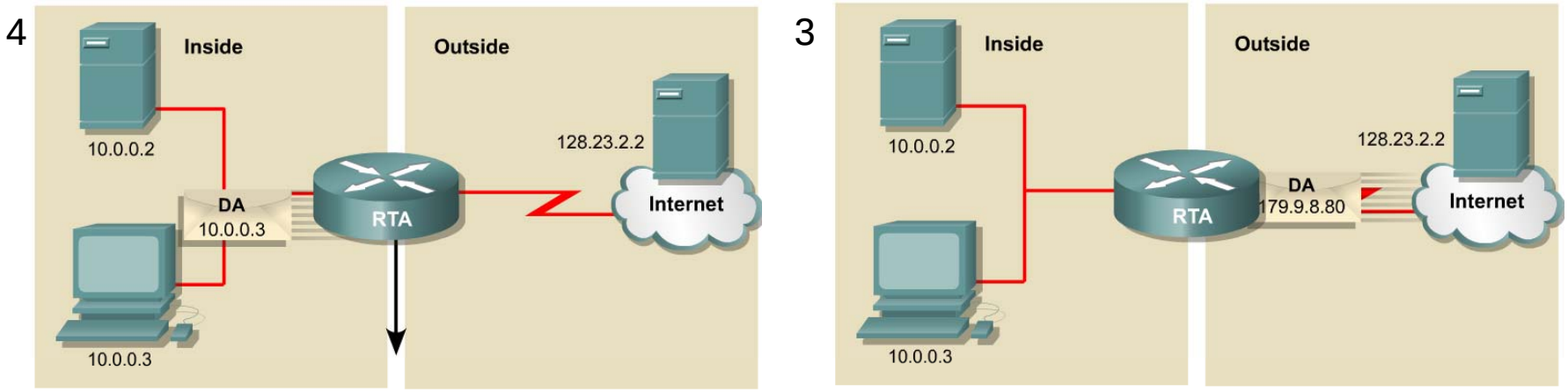


NAT Table		
Inside Local IP Address	Inside Global IP Address	Outside Global IP Address
10.0.0.3	179.9.8.80	128.23.2.2



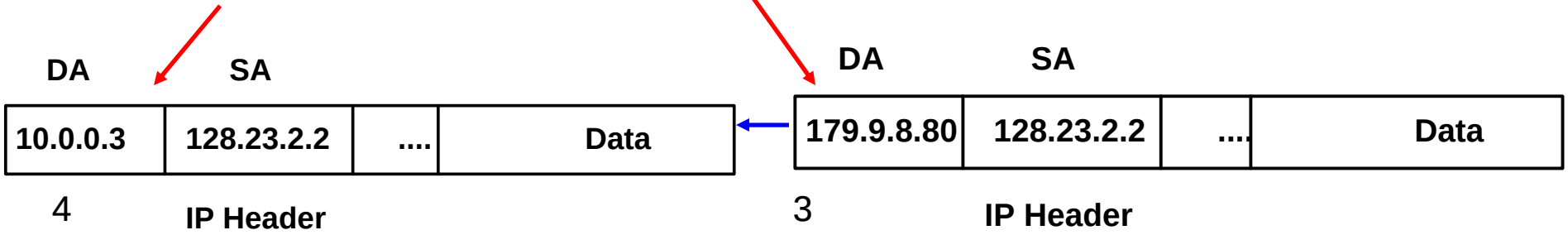
The translation from Private source IP address to Public source IP address.

NAT Example



NAT Table

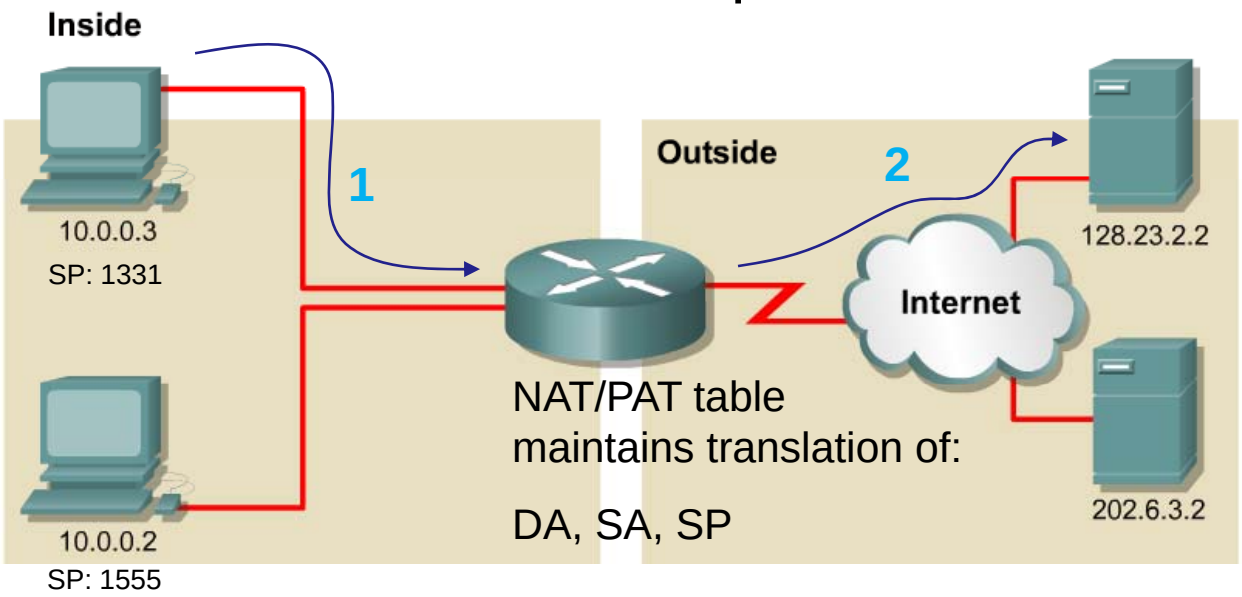
Inside Local IP Address	Inside Global IP Address	Outside Global IP Address
10.0.0.2	179.9.8.80	128.23.2.2
10.0.0.3	179.9.8.80	128.23.2.2



Translation back, from Public <u>destination</u> IP address to Private <u>destination</u> 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

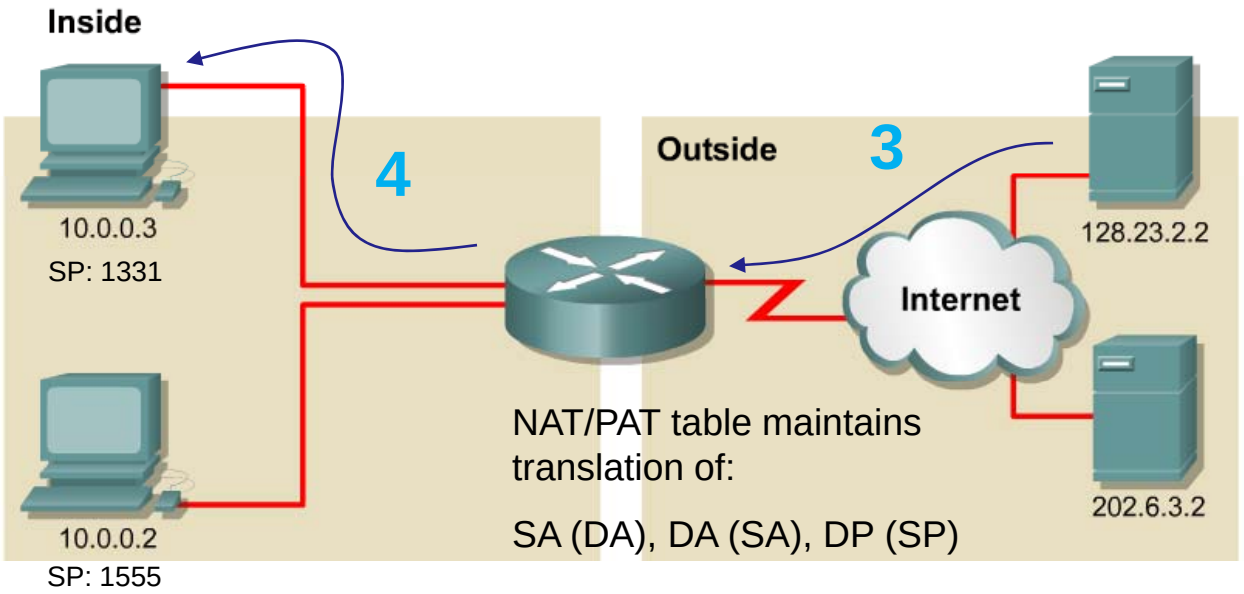
DA	SA	DP	SP	
128.23.2.2	10.0.0.2	80	1555	Data

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



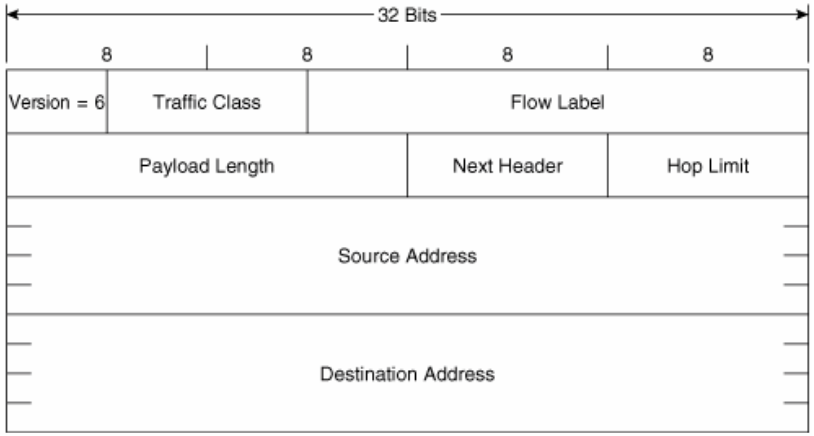
DA	SA	DP	SP		DA	SA	DP	SP	
10.0.0.3	128.23.2.2	1331	80	Data	179.9.8.80	128.23.2.2	3333	80	Data
4 IP Header	TCP/UDP Header				3 IP Header	TCP/UDP Header			

translated

DA	SA	DP	SP		DA	SA	DP	SP	
10.0.0.2	128.23.2.2	1555	80	Data	179.9.8.80	128.23.2.2	2222	80	Data
IP Header	TCP/UDP Header				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

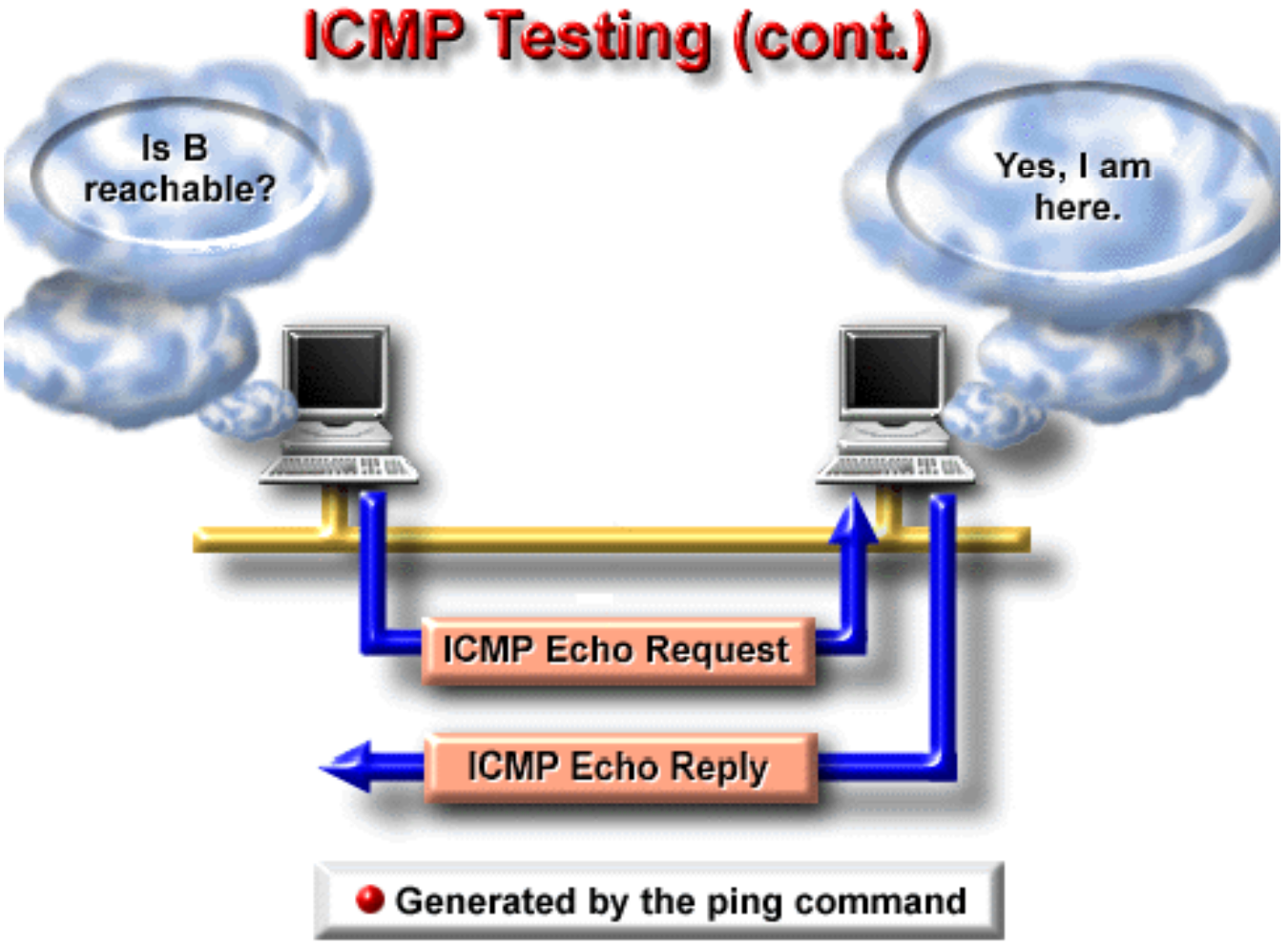
Long Term Solution: IPv6 (coming)

- 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.

Trouble shooting

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).


ICMP Testing (cont.)
Is B reachable?
Yes, I am here.
ICMP Echo Request
ICMP Echo Reply
Generated by the ping command
© Cisco Systems, Inc. 1999

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 ping command

Ping command using –I option to "ping from …"

```
[root@elrond ~]# ping -I eth0 opus.cabrillo.edu
PING opus.cabrillo.edu (207.62.186.9) from 172.30.4.121 eth0: 56(84) bytes of data.
64 bytes from opus.cabrillo.edu (207.62.186.9): icmp_seq=1 ttl=63 time=1.26 ms
64 bytes from opus.cabrillo.edu (207.62.186.9): icmp_seq=2 ttl=63 time=1.43 ms

--- opus.cabrillo.edu ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.261/1.348/1.435/0.087 ms
[root@elrond ~]# ping -I eth1 opus.cabrillo.edu
PING opus.cabrillo.edu (207.62.186.9) from 192.168.2.107 eth1: 56(84) bytes of data.
From 192.168.2.107 icmp_seq=1 Destination Host Unreachable
From 192.168.2.107 icmp_seq=2 Destination Host Unreachable
From 192.168.2.107 icmp_seq=3 Destination Host Unreachable

--- opus.cabrillo.edu ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss, time 4119ms
, pipe 3
[root@elrond ~]#
```

arp fails to get an IP addresss of the router when forced to go out eth1

Troubleshooting ping command

Ping command using –I option to "ping from …"

```
[root@elrond ~]# ping -I 192.168.2.107 opus.cabrillo.edu
PING opus.cabrillo.edu (207.62.186.9) from 192.168.2.107 : 56(84) bytes of data.

--- opus.cabrillo.edu ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6007ms
```

Nothing seems to happen …. until you hit Ctrl-C

There is no path back to the private network. Echo requests go out but the echo replies can't get back!

Example ping troubleshooting

Network is unreachable (1 of 3)

An example with Elrond is on CIS-Lab-01 in the CIS Lab

```
[[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:2085 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1020 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:276179 (269.7 KiB)  TX bytes:159336 (155.6 KiB)
          Interrupt:177 Base address:0x1400

[root@elrond ~]# ping -c 2 172.30.4.1
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=1.35 ms
64 bytes from 172.30.4.1: icmp_seq=2 ttl=255 time=1.46 ms

--- 172.30.4.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.352/1.406/1.461/0.066 ms

[root@elrond ~]# ping -c 2 172.30.1.1
connect: Network is unreachable
[root@elrond ~]#
```

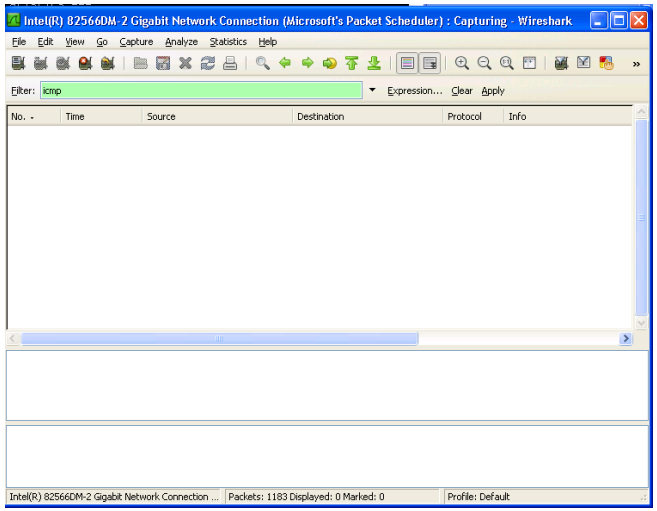
Pinging another device on the same subnet succeeds.

However, pinging a device on another subnet fails. Why? Lets find out

Note: 172.30.1.1 is the router interface on Nosmo used in the classroom

Example ping troubleshooting

Network is unreachable (2 of 3)



Using Wireshark, we see no packets even left from the NIC. The error was detected locally.

Lets check the routing table next … uh oh, no routes to the 172.30.1.0 /24 network!

```
[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
[root@elrond ~]#
```

We forgot to add a default gateway!
And that explains why the "Network is unreachable"

Example ping troubleshooting

Network is unreachable (3 of 3)

Adding a default gateway solves the problem.

```
[root@elrond ~]# route add default gw 172.30.4.1
[root@elrond ~]# ping 172.30.1.1
PING 172.30.1.1 (172.30.1.1) 56(84) bytes of data.
64 bytes from 172.30.1.1: icmp_seq=1 ttl=255 time=3.77 ms
64 bytes from 172.30.1.1: icmp_seq=2 ttl=255 time=1.57 ms

--- 172.30.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.571/2.670/3.770/1.100 ms
[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
0.0.0.0         172.30.4.1      0.0.0.0         UG    0      0        0 eth0
[root@elrond ~]#
```

We need a router to get to the classroom network 172.30.1.0/24 from the Lab network.

Troubleshooting traceroute command

- traceroute command show route to destination address
- Increments TTL by 1 each time and uses the ICMP time exceeded response to "hop" from router to router.
- Uses UDP (may get blocked by a firewall)
- Use –I option to use ICMP instead of UDP
- Note: tracert on Windows always uses ICMP

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  * * *
```

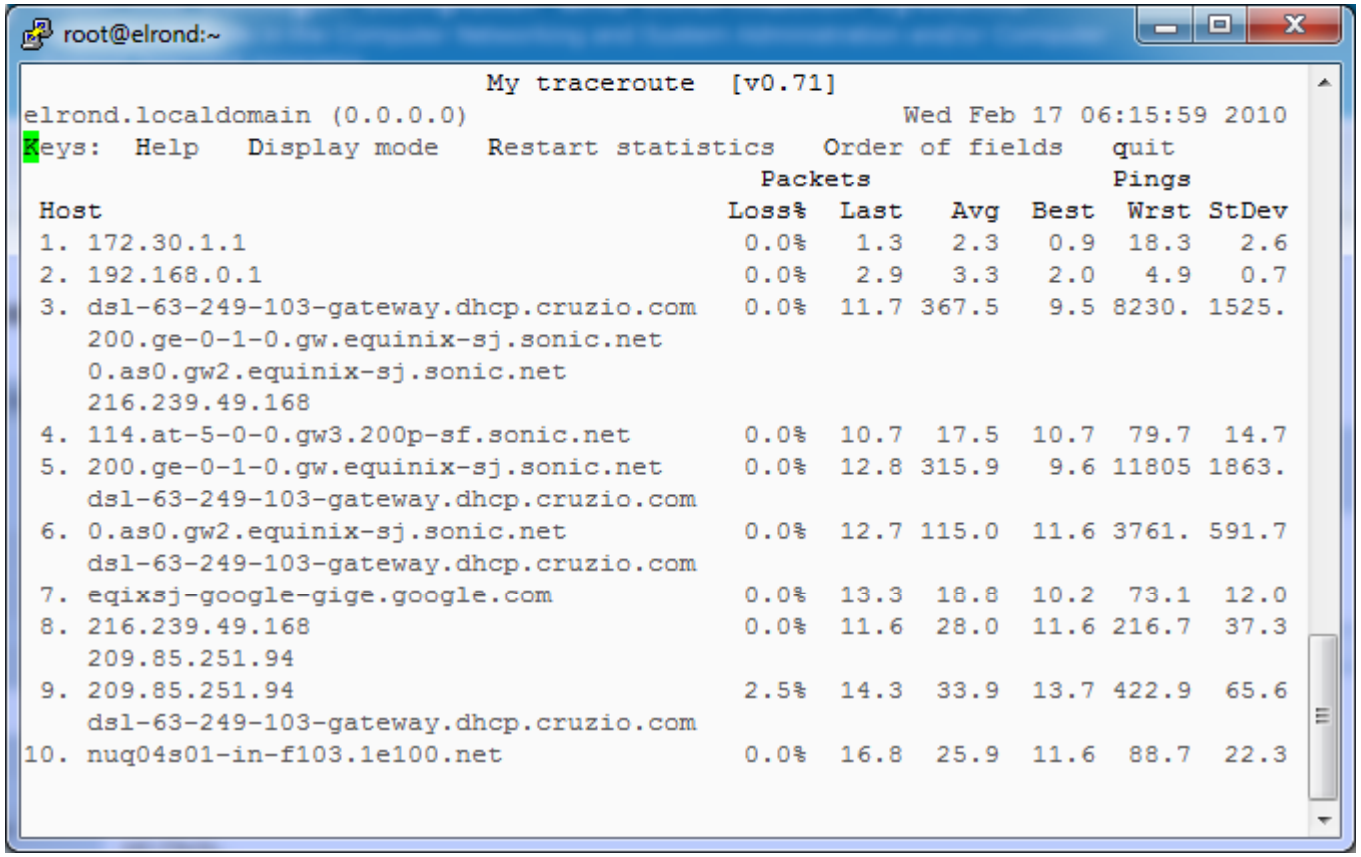
Ctrl-C to stop

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 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: Temporary Network Interface Card Configuration

The purpose of this lab is to configure the NICs (network interface controllers) of several Linux 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. Putty and SSH will be used to traverse through the various systems after the interfaces have been configured.

Supplies

- VMWare Server 1.08 or higher
- 192 VMs: Frodo, Elrond, and Fang
- Virtual networks: VMnet3

Some essentials for doing labs

- Becoming root:
  - sudo *command*
  - su –

The "–" is very important as this gets you root's environment

- To try again for a DHCP address: dhclient

If Frodo's DHCP interface fails to get an IP address after booting up use this command

- Use Google to research error messages
  - Google *network is unreachable*

You will need to login as root to do most labs.
Be careful as root can do anything !!

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
netstat -i
netconfig
ipcalc
ping –cIR
traceroute

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:

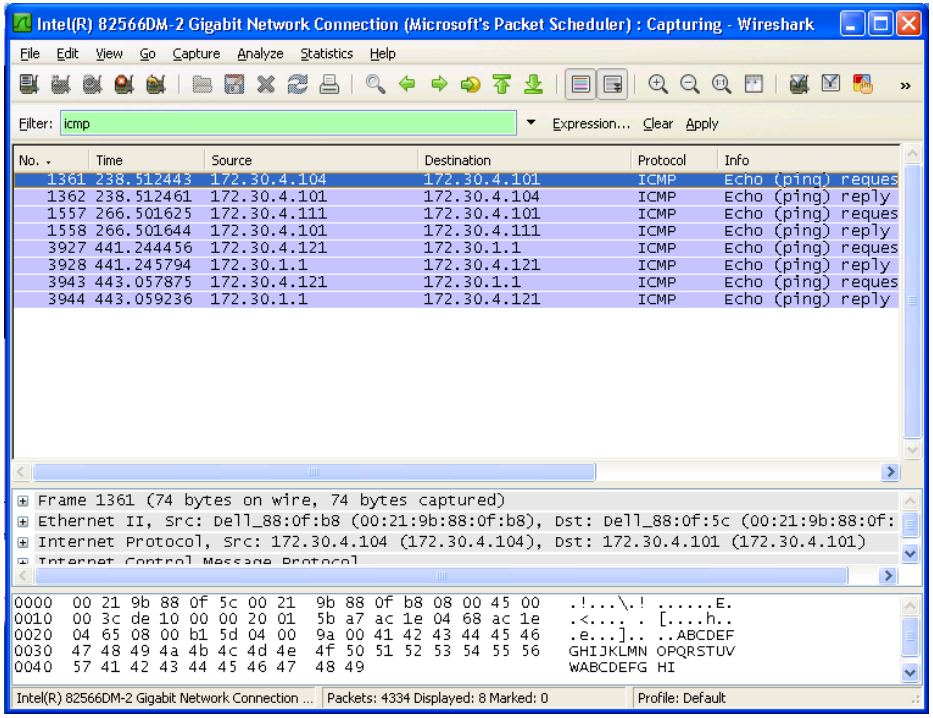
- What does the C flag mean when viewing ARP cache entries with arp -n?
- What Wireshark display filter would only show ARP and ICMP protocol packets?
- With an IP address of 172.30.4.100 and a netmask of 255.255.0.0, what is the broadcast address?

Backup

Example ping troubleshooting

Network is unreachable

Adding a default gateway solves the problem.



The ICMP packets can be viewed using Wireshark. Looks like someone else is in the lab right now pinging CIS-Lab-101 from CIS-Lab-04 and CIS-Lab-11.

This wireshark is running on CIS-Lab-01. Note: It sees all the packets traveling to itself (172.30.4.101) or the Elrond VM (172.30.4.121) which is cabled using the VMware bridged option.