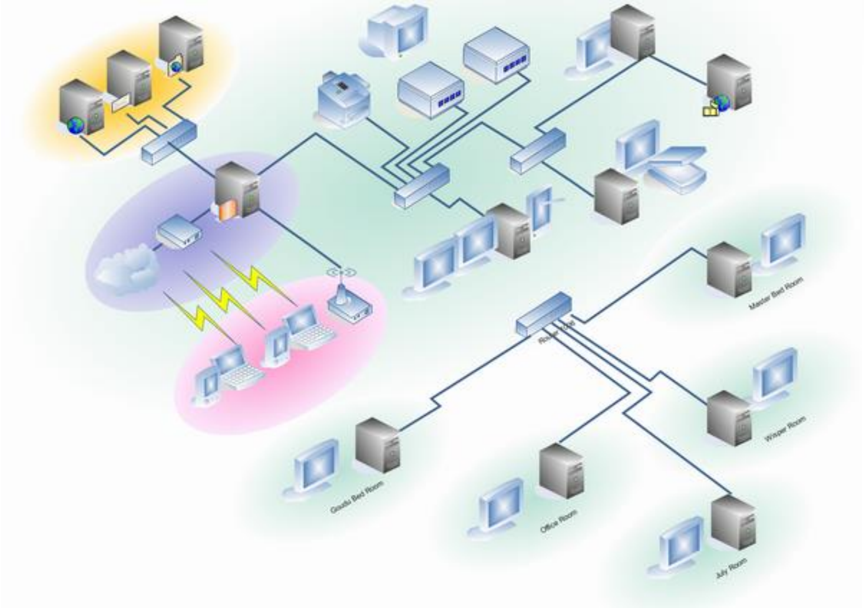
Linux For Embedded Systems

For Arabs

Course 102:

Understanding Linux

Ahmed ElArabawy



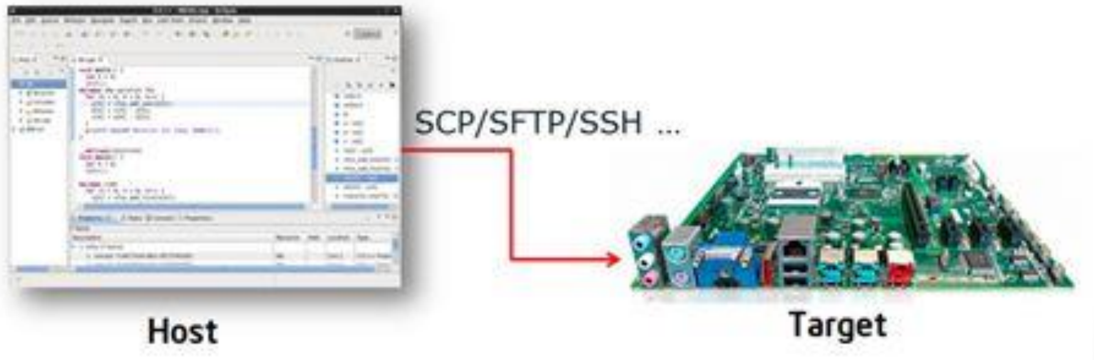
Lecture 20:
Networking in Linux (Basic Concepts)

Why Networking ??

- Networking is a very essential tool for Linux Users/developers
  - Setup our machine to be able to connect to the internet
  - Identify our machine information (interfaces/IP Address/ …)
  - Copy files to/from remote machines
  - Copy files to/from the web
  - Remotely access a remote machine
  - And Other important tasks….

Why Networking ??

- For an embedded developer, networking is even much more essential
- Embedded Systems normally comes without a display, and we will need to connect to it via a host machine
  - We will need to connect to the embedded target for:
    - Reading the logs
    - Debugging the program
    - Setting some configuration files
    - Uploading a new image



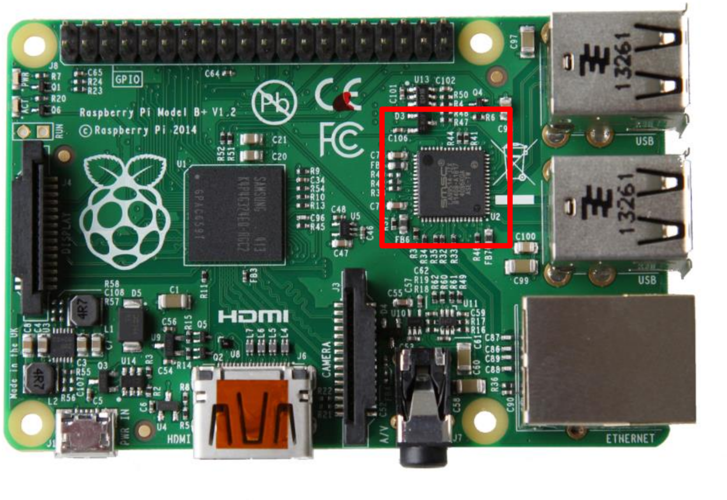
So ….

- Networking to an embedded user/developer aims for two main purposes
    - Connect to a remote machine or to the internet
        - This is typically done using an Ethernet connection
    - Connect a host machine to the embedded target
        - This can be achieved via,
            - JTAG Connection
            - Serial Console Connection
            - Ethernet Connection

Networking Basics

Network Interfaces

- The computer/board may have one or more network interfaces
- In Hardware, this is represented by,
  - An Ethernet NIC card
  - WiFi port
- In Linux, each interface will have a name,
  - Examples (eth0, eth1, ….)



Showing the Network Interfaces (ifconfig Command)

$ ifconfig

$ ifconfig <interface name> [up/down]

- This command is responsible for showing and managing interfaces
- To show the interfaces in the system

  $ ifconfig

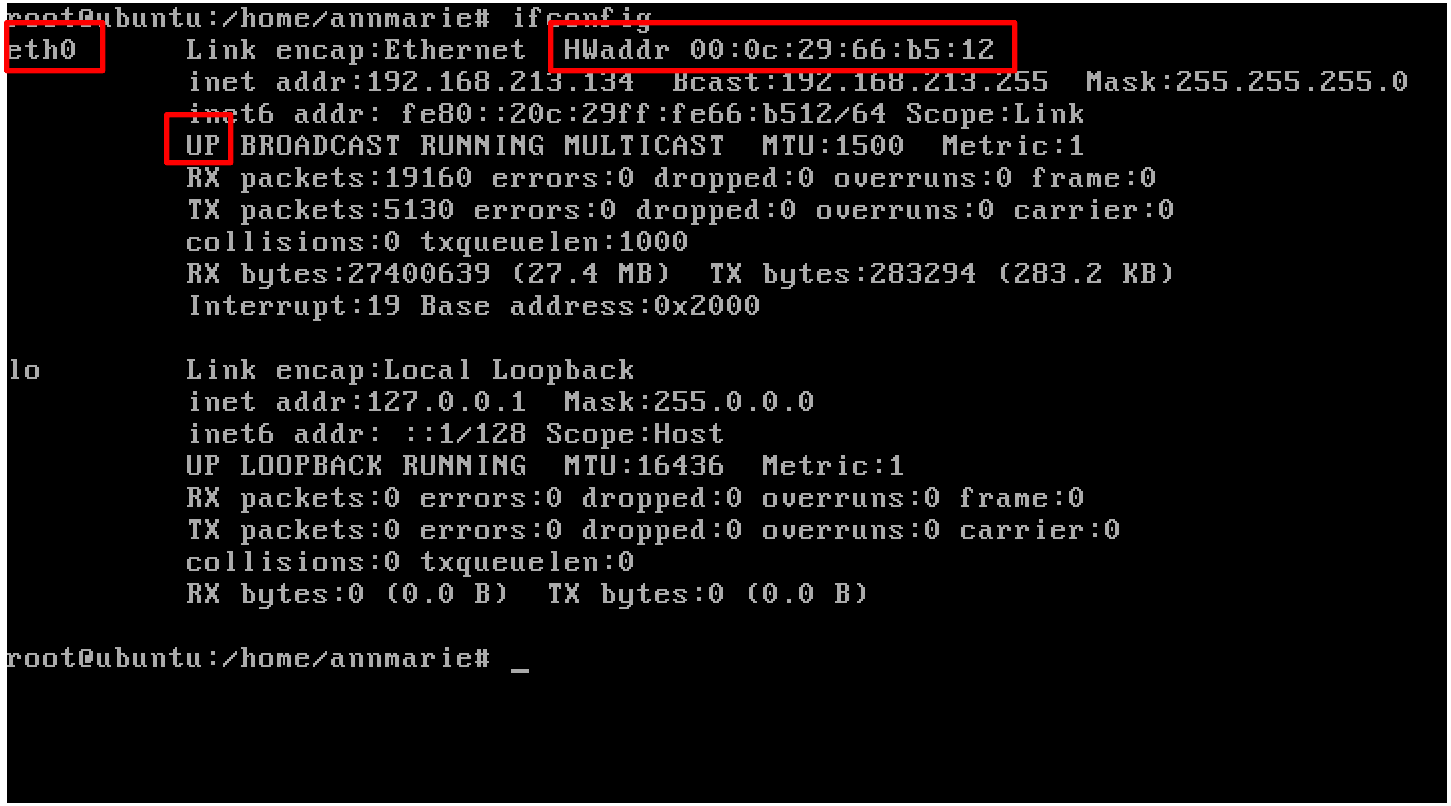
  $ ifconfig -s (shows an abstract description)
- To show info about a specific interface

  $ ifconfig eth1
- To activate/deactivate

  $ ifconfig eth1 up

  $ ifconfig eth2 down

Showing the Network Interfaces (ifconfig Command)



```
root@ubuntu:/home/annmarie# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:66:b5:12
          inet addr:192.168.213.134  Bcast:192.168.213.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe66:b512/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:19160 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5130 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:27400639 (27.4 MB)  TX bytes:283294 (283.2 KB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@ubuntu:/home/annmarie# _
```

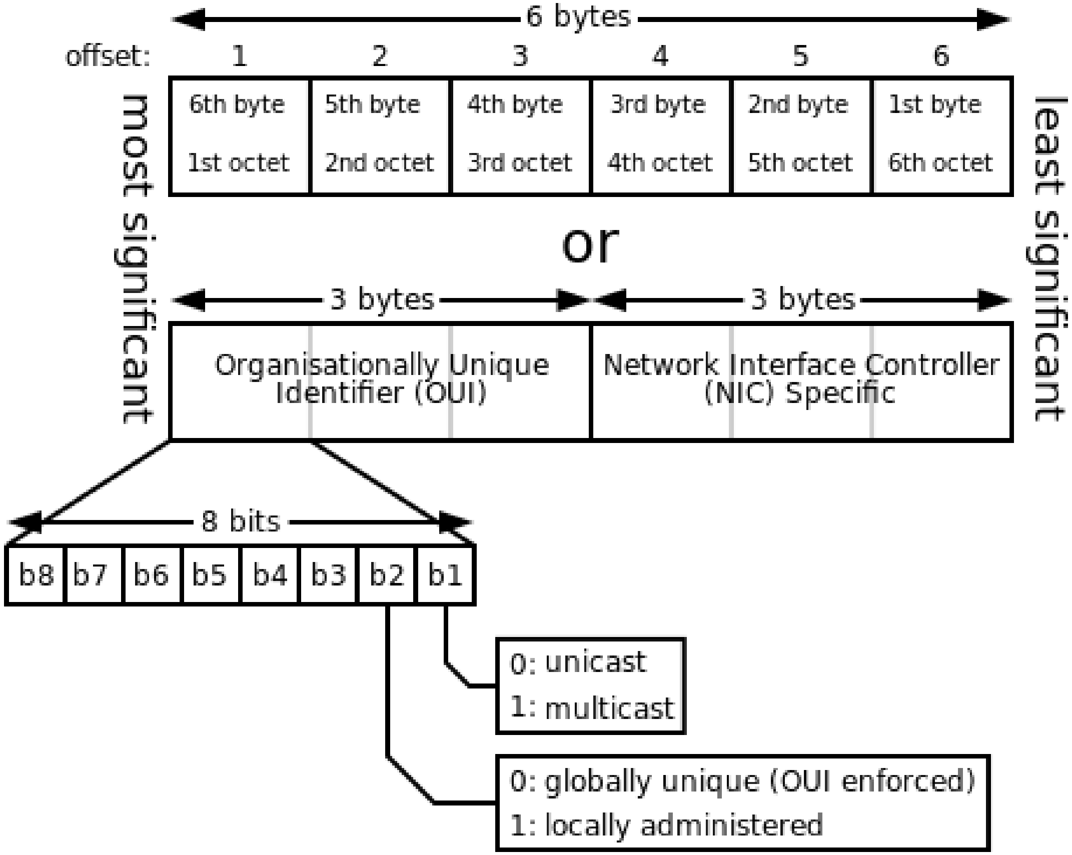
The MAC Address

- The MAC address is also called the Hardware Address
- It is a unique address for each NIC card (set by the manufacturer)
- It is a 48 bit number written in hexadecimal format (4 bit digits) as follows,

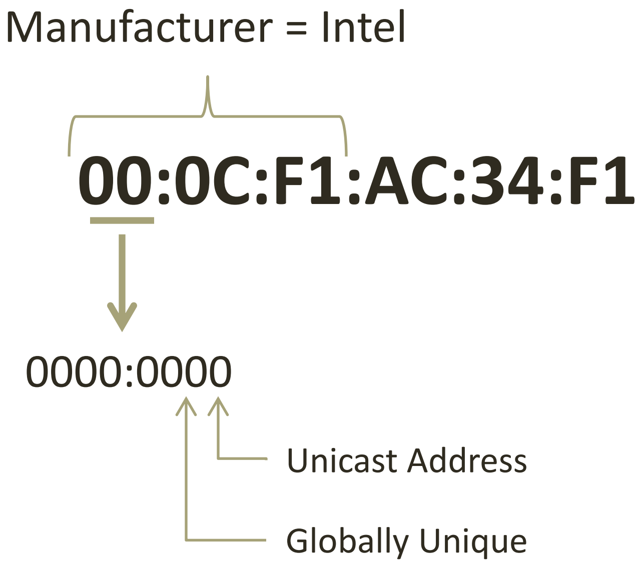
  68:05:CA:03:19:9C

- Note,
  - Hexadecimal format is a format that converts every 4 bits into a single digit
  - Since 4 bits require 16 combinations (takes values from 0 – 15), the digits (0-9, A,B,C,D,E,F) are used
  - Accordingly, since the MAC address is 48 bits, it will be written in 12 hexadecimal digits

MAC Address


6 bytes
offset:
1
2
3
4
5
6
6th byte
1st octet
5th byte
2nd octet
4th byte
3rd octet
3rd byte
4th octet
2nd byte
5th octet
1st byte
6th octet
most significant
least significant
or
3 bytes
3 bytes
Organisationally Unique Identifier (OUI)
Network Interface Controller (NIC) Specific
8 bits
b8
b7
b6
b5
b4
b3
b2
b1
0: unicast
1: multicast
0: globally unique (OUI enforced)
1: locally administered

MAC Address



Manufacturer = Intel

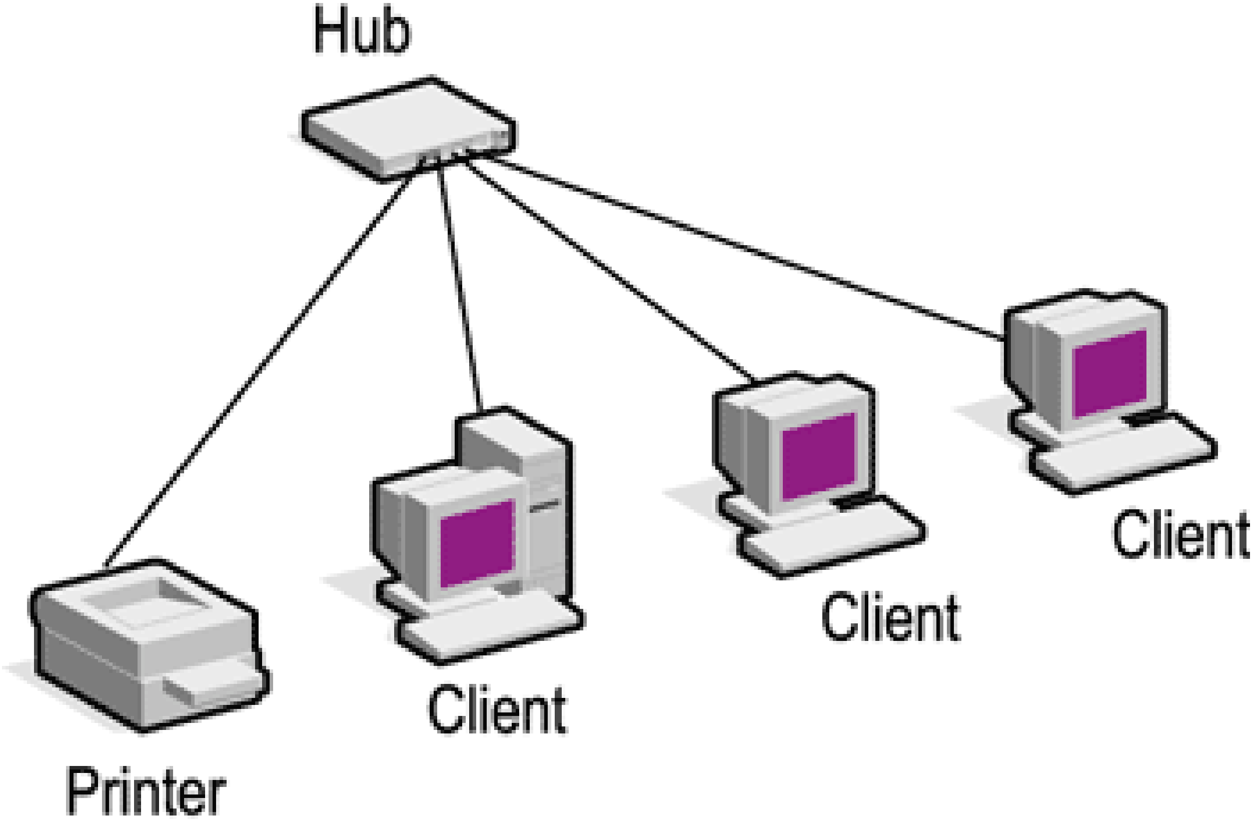
00:0C:F1:AC:34:F1

0000:0000

Unicast Address

Globally Unique

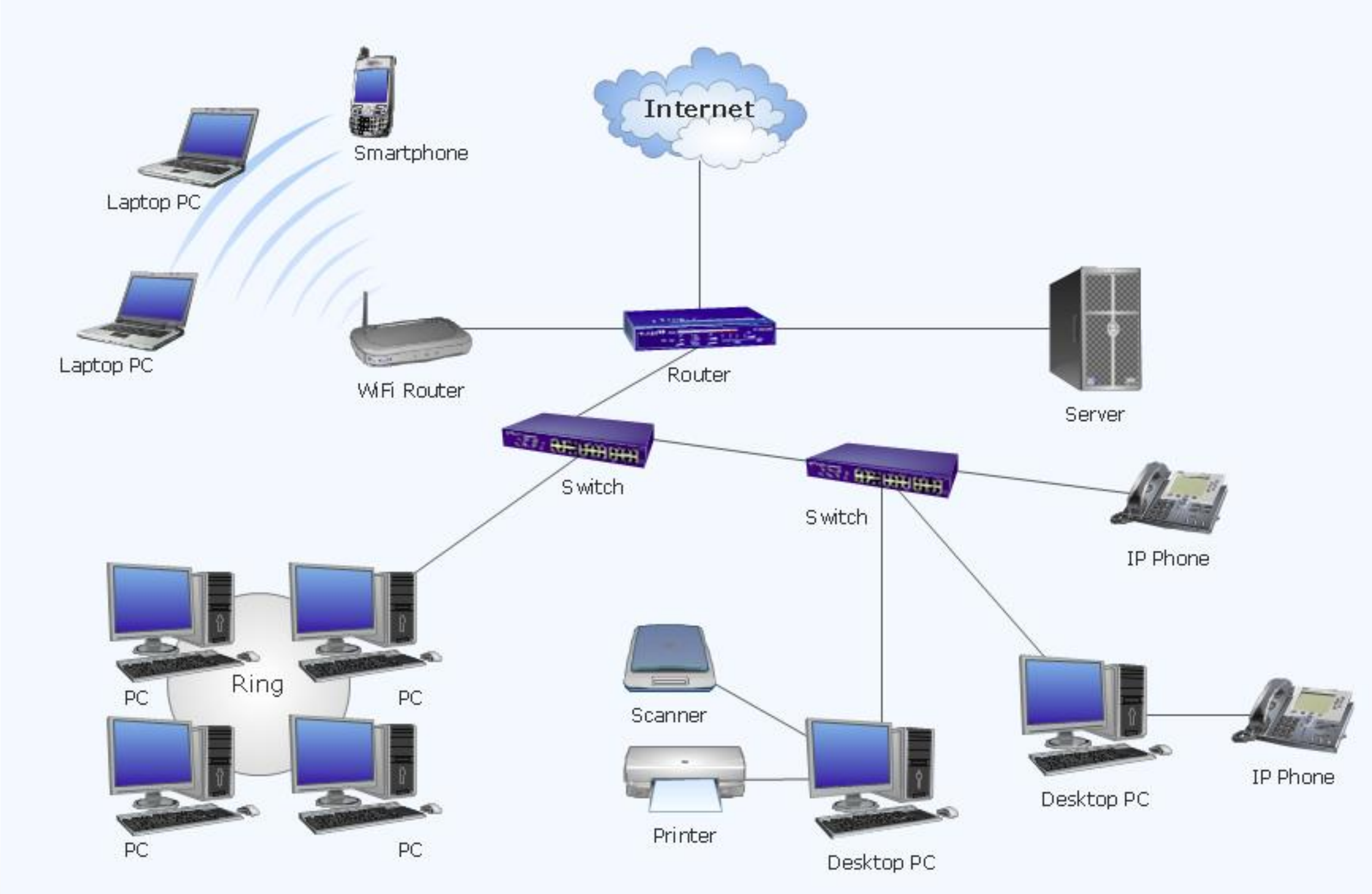
Local Area Network



Local Area Network

- A Local Area Network (LAN) is a set of computers connected together via hub/switch
- The devices (computers) are located in the same area (locally located)
- The computers have the same type of network interface (Ethernet, Wifi, …)
- Data is transferred between the machines based on their MAC addresses

A Real Network



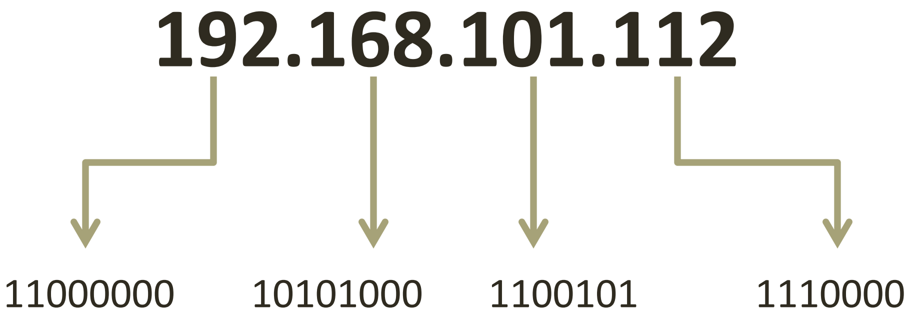
A Real Network

- In a more realistic network, not all computer reside in the same LAN,
  - Limitation on distance between the devices
  - Different kind of requirements (Wired/wireless/…)
  - Need to isolated devices in groups
  - Other…
- Hence A typical network is composed of a group of LANs interconnected with each other with some routing devices
- Within each LAN, MAC address is used for communication inside this LAN
- However, the MAC Address is no longer enough to communicate between the devices in different LANs, and we need a more global addressing scheme
- This is called network addressing which uses the ***IP Address***

IP Address

- Any network interface on a machine in the network is identified by its IP Address
  - Sometimes the interface may have multiple IP-Addresses
- The IP address is composed of 4 numbers (0-255) separated by a dot
  - Example: 192.168.101.112
  - This is assuming, we are using IPv4, another version which is IPv6 has a different format
  - Note that the number 0-255 is an 8 bit number

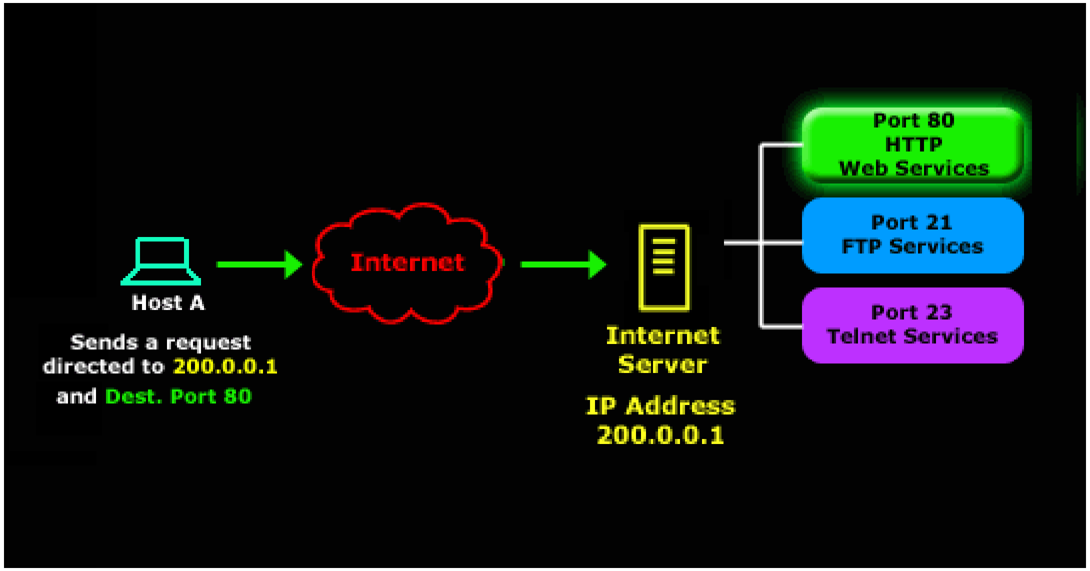
IP Address



192.168.101.112

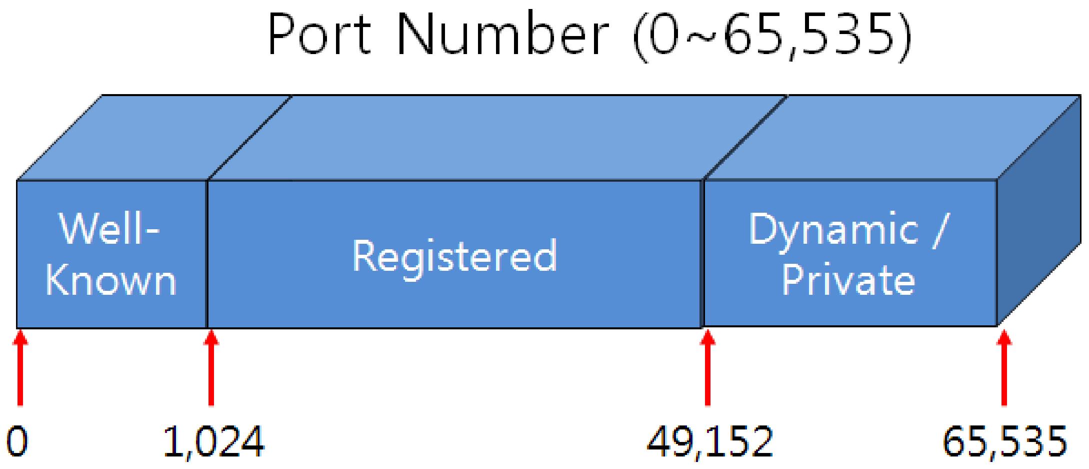
11000000 10101000 1100101 1110000

Is IP Address Enough ??



- Port numbers identify different services within the same IP Address
- Each connection will have both source port and destination port numbers

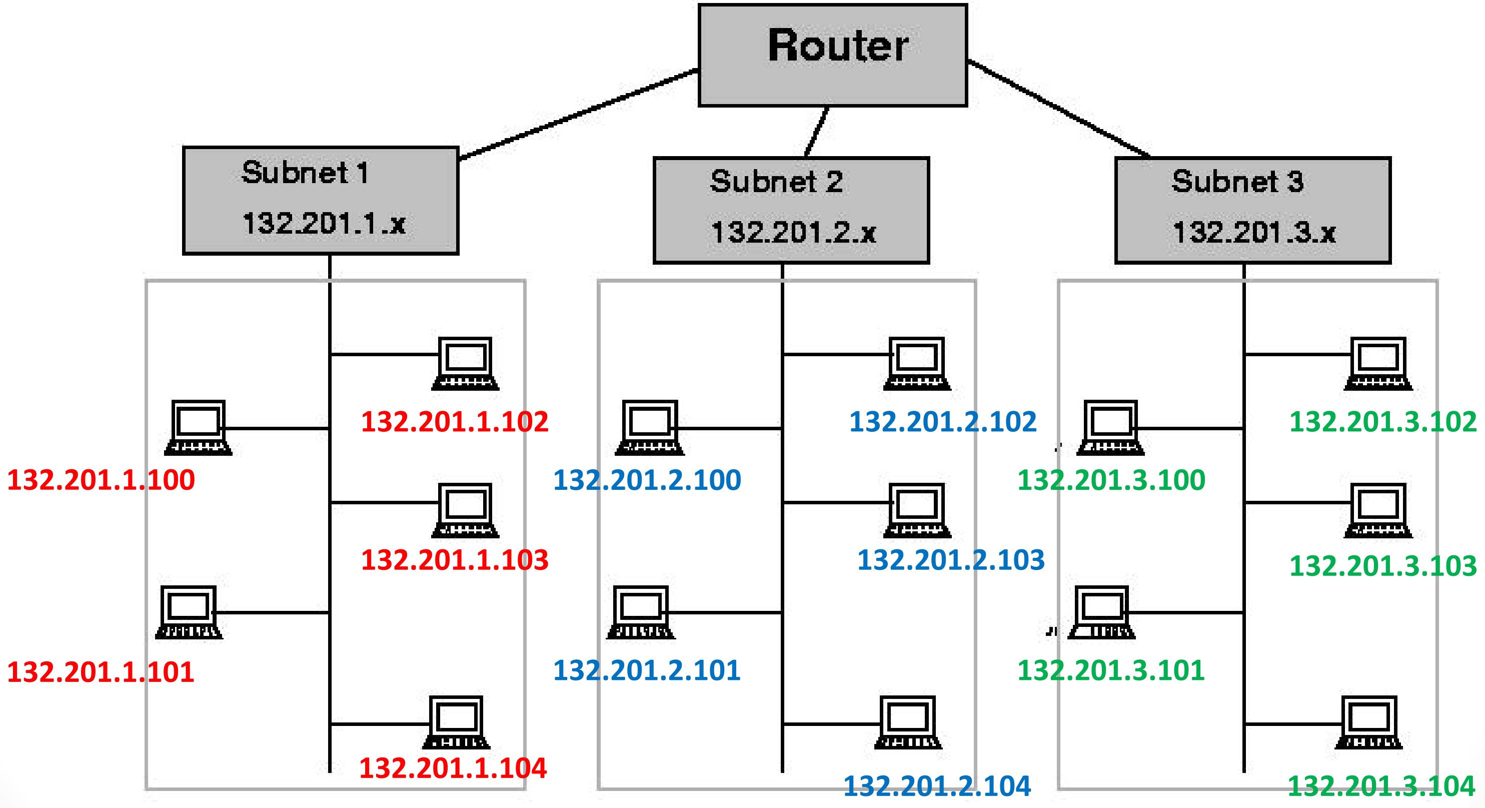
Port Numbers



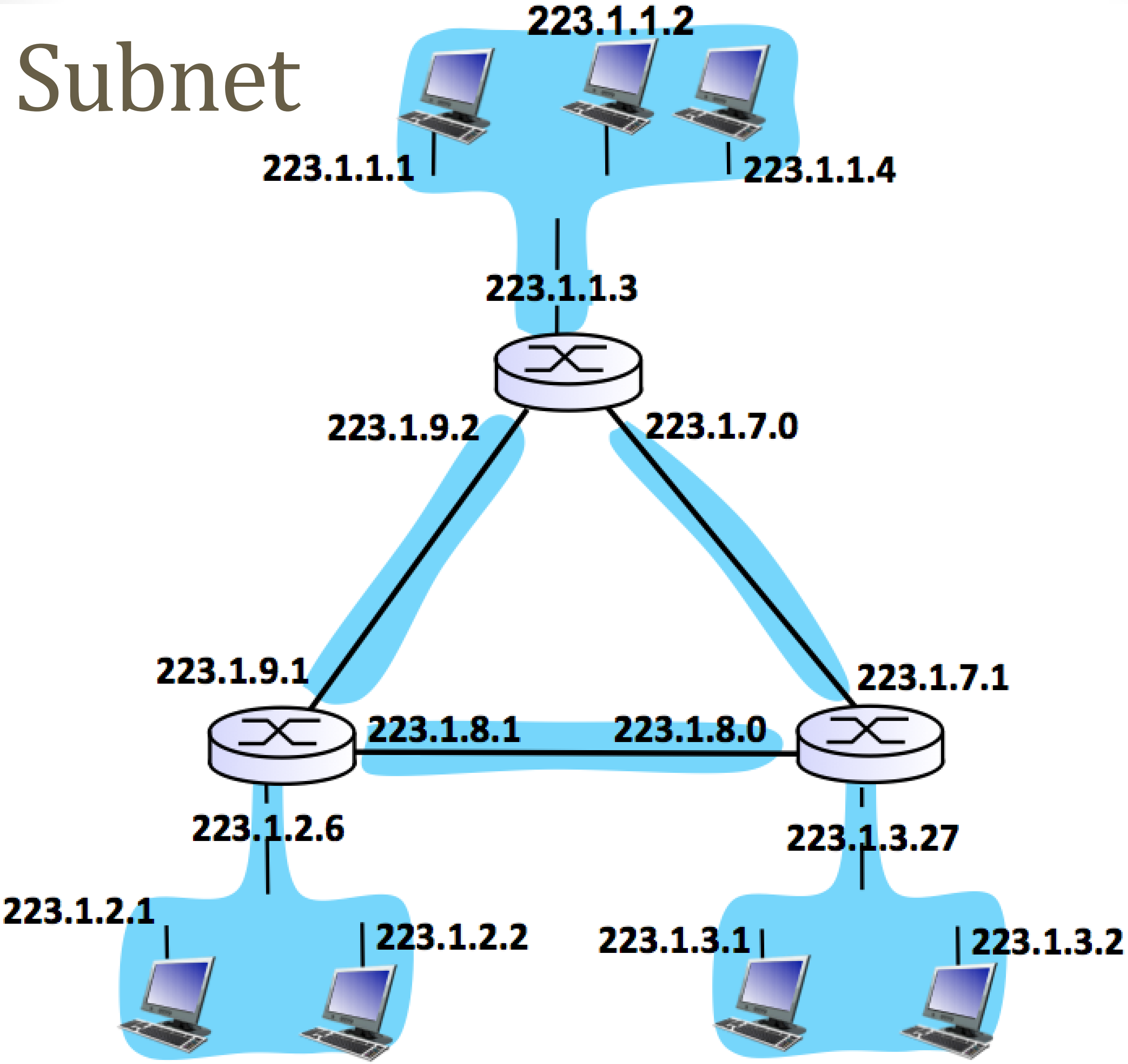
Some Well Know Port Numbers	Services
21	FTP
22	SSH
23	Telnet
25	SMTP
53	DNS
80	HTTP
110	POP3
143	IMAP

Subnet

- To Facilitate Routing,
  - The machines are organized into small networks (**subnets**)
  - Subnets share the upper part of the IP address



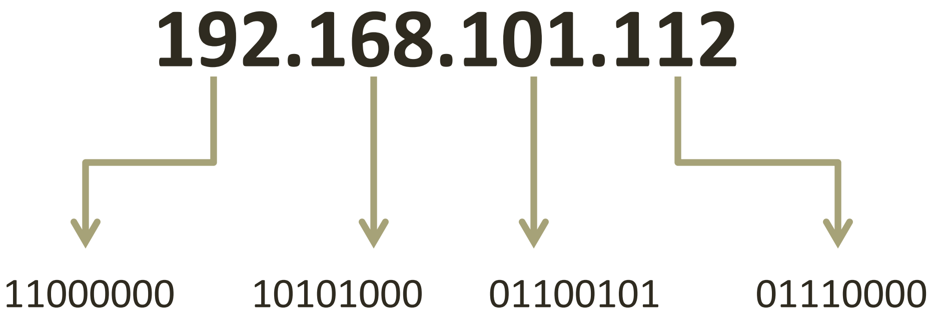
Subnet


223.1.1.2
223.1.1.1
223.1.1.4
223.1.1.3
223.1.9.2
223.1.7.0
223.1.9.1
223.1.7.1
223.1.8.1
223.1.8.0
223.1.2.6
223.1.3.27
223.1.2.1
223.1.2.2
223.1.3.1
223.1.3.2

Subnet

- Network interfaces within the same subnet share the upper part of the IP address, and differ in the lower part
- The number of bits shared within the subnet control the subnet size
- For example if we have the upper 24 bits (3 bytes) shared within the subnet, then the interfaces with the subnet can differ in 8 bits only, which leads to a maximum of 256 addresses
- If the shared part is reduced to 20 bits only, we will have 12 bits to change which lead to 4096 different addresses
- The part of the IP Address shared among the interfaces within the subnet, is expressed as the **subnet mask**

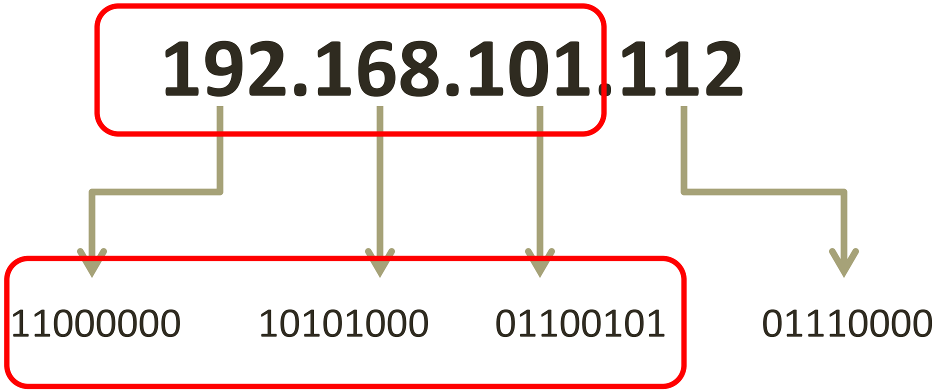
Subnet Mask



192.168.101.112

11000000 10101000 01100101 01110000

Subnet Mask



192.168.101.112

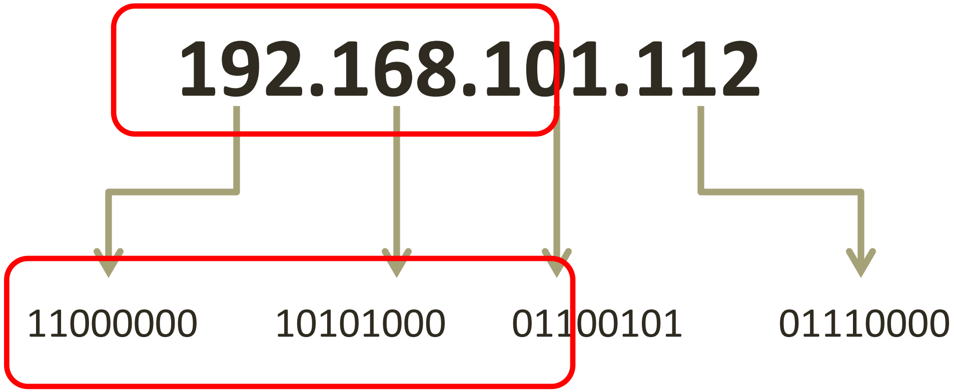
11000000 10101000 01100101 01110000

Subnet Address : 192.168.101.0

Subnet Mask : 255.255.255.0

Address : 192.168.101.112/24

Subnet Mask



192.168.101.112

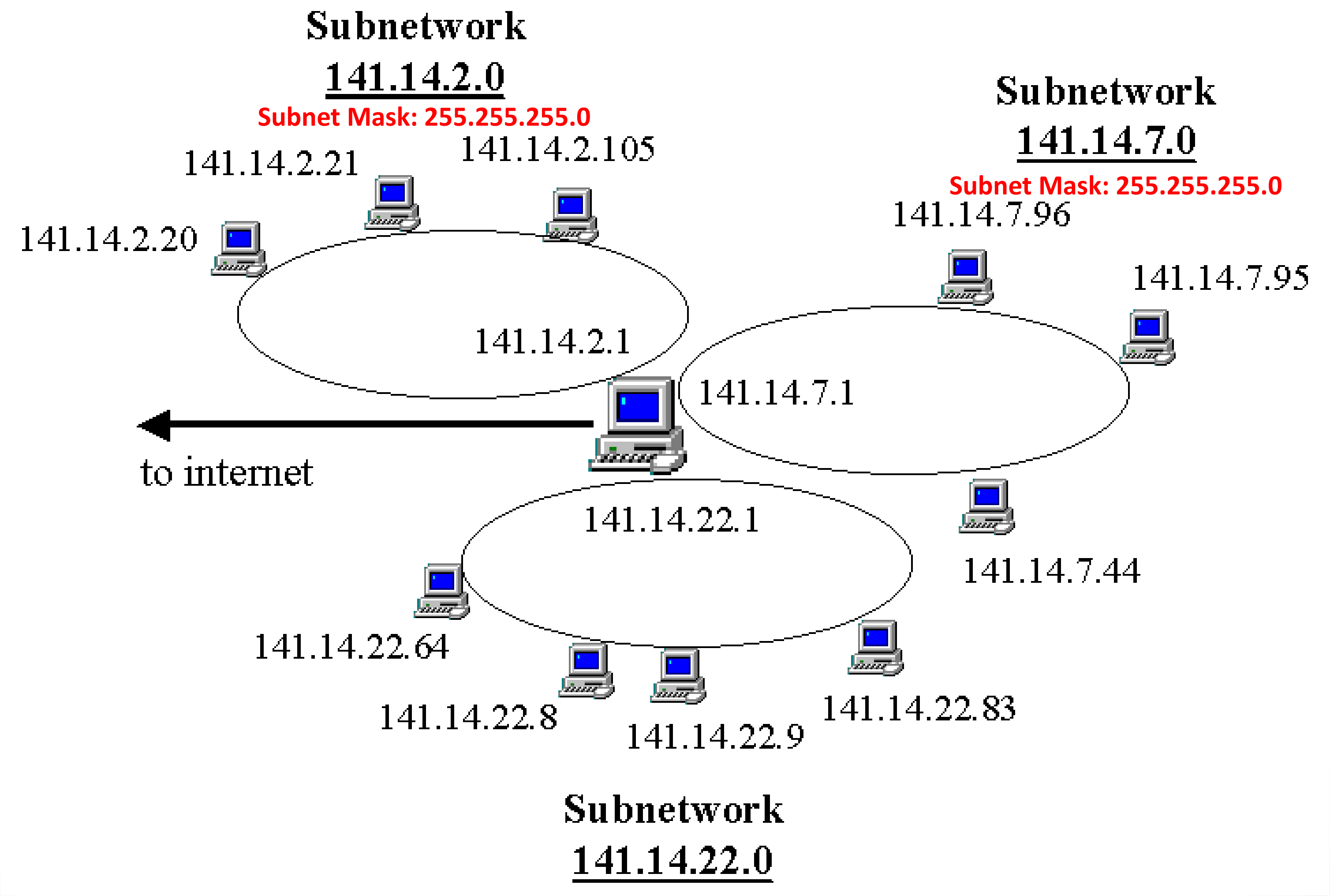
11000000 10101000 01100101 01110000

Subnet Address : 192.168.96.0

Subnet Mask : 255.255.240.0

Address : 192.168.101.112/19

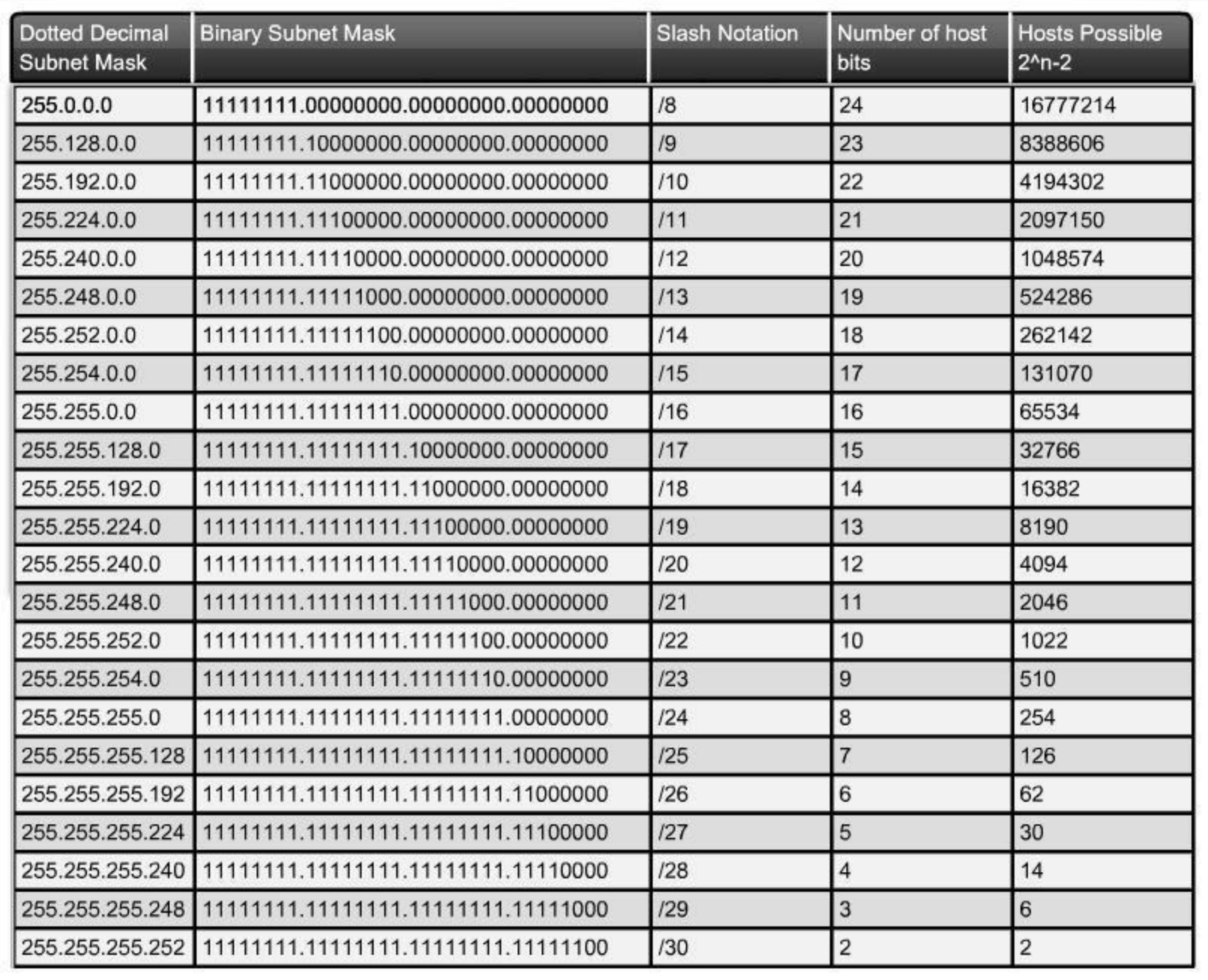
Subnet Mask



Subnet Mask: 255.255.255.0

Subnet Mask: 255.255.255.0

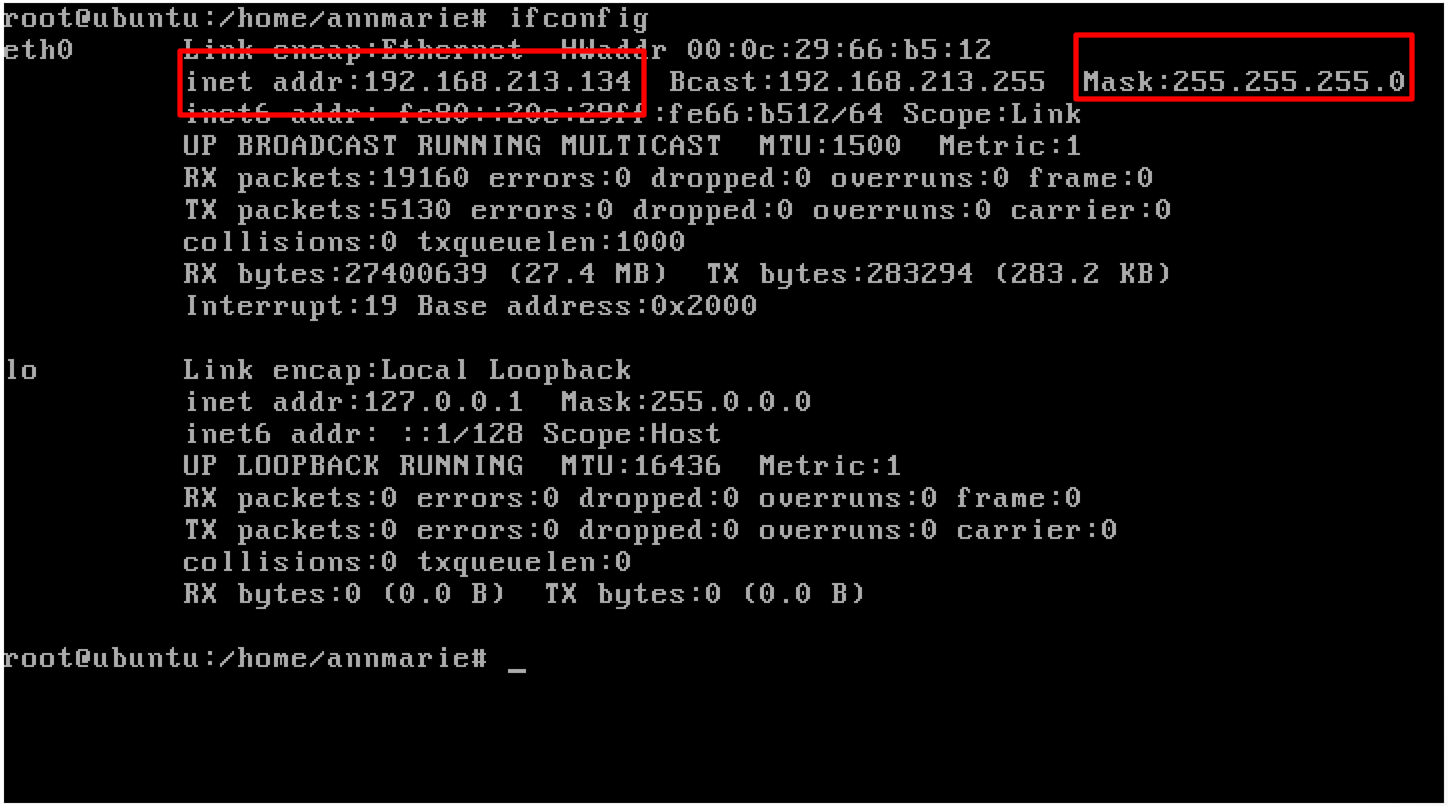
Subnet Mask: 255.255.255.0



Dotted Decimal Subnet Mask	Binary Subnet Mask	Slash Notation	Number of host bits	Hosts Possible 2^n-2
255.0.0.0	11111111.00000000.00000000.00000000	/8	24	16777214
255.128.0.0	11111111.10000000.00000000.00000000	/9	23	8388606
255.192.0.0	11111111.11000000.00000000.00000000	/10	22	4194302
255.224.0.0	11111111.11100000.00000000.00000000	/11	21	2097150
255.240.0.0	11111111.11110000.00000000.00000000	/12	20	1048574
255.248.0.0	11111111.11111000.00000000.00000000	/13	19	524286
255.252.0.0	11111111.11111100.00000000.00000000	/14	18	262142
255.254.0.0	11111111.11111110.00000000.00000000	/15	17	131070
255.255.0.0	11111111.11111111.00000000.00000000	/16	16	65534
255.255.128.0	11111111.11111111.10000000.00000000	/17	15	32766
255.255.192.0	11111111.11111111.11000000.00000000	/18	14	16382
255.255.224.0	11111111.11111111.11100000.00000000	/19	13	8190
255.255.240.0	11111111.11111111.11110000.00000000	/20	12	4094
255.255.248.0	11111111.11111111.11111000.00000000	/21	11	2046
255.255.252.0	11111111.11111111.11111100.00000000	/22	10	1022
255.255.254.0	11111111.11111111.11111110.00000000	/23	9	510
255.255.255.0	11111111.11111111.11111111.00000000	/24	8	254
255.255.255.128	11111111.11111111.11111111.10000000	/25	7	126
255.255.255.192	11111111.11111111.11111111.11000000	/26	6	62
255.255.255.224	11111111.11111111.11111111.11100000	/27	5	30
255.255.255.240	11111111.11111111.11111111.11110000	/28	4	14
255.255.255.248	11111111.11111111.11111111.11111000	/29	3	6
255.255.255.252	11111111.11111111.11111111.11111100	/30	2	2

Show Subnet Mask (ifconfig Command)

$ ifconfig



```
root@ubuntu:/home/annmarie# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:66:b5:12
          inet addr:192.168.213.134  Bcast:192.168.213.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe66:b512/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:19160 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5130 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:27400639 (27.4 MB)  TX bytes:283294 (283.2 KB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@ubuntu:/home/annmarie# _
```

IP Address Allocation

- IP Address can be allocated in two different ways,
  - Statically allocated
    - This is done manually, or via a startup script
    - Address is always the same
  - Dynamically allocated via a **DHCP server**
    - At startup, the machine asks a DHCP server to provide it with an IP Address
    - The DHCP server responds with an unused IP address within the proper subnet
    - Hence, the IP address can change every time the machine boots
    - The DHCP server can be configured so it will always assign a specific machine the same IP Address
- Using DHCP address allocation is more convenient and more flexible
- Since the DHCP server needs to be accessed before the IP address is allocated, it should reside within the same LAN as the machines (it is typically located inside the LAN switch)

Allocating IP Address (ifconfig Command)

```
$ ifconfig <interface> <IP Address>
$ ifconfig <interface> <IP Address> netmask <netmask>
$ ifconfig <interface> netmask <netmask>
```

Used to statically allocate IP address to an Interface and/or set its net-mask

- Example:

```
$ ifconfig eth1 192.268.0.10
$ ifconfig eth1 netmask 255.255.255.224
$ ifconfig eth1 192.268.0.10 netmask 255.255.255.224
```

Special IP Addresses

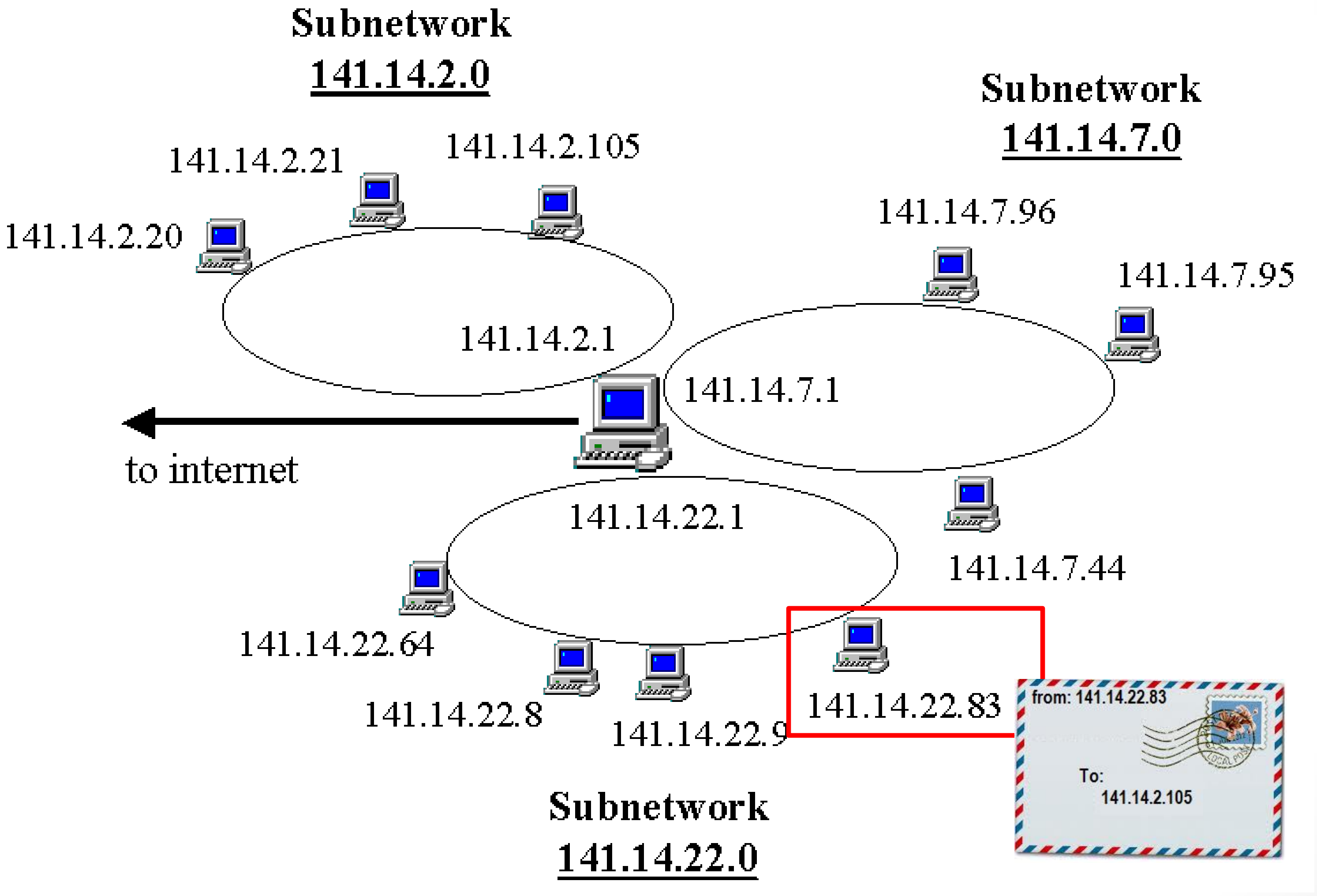
- Some IP Addresses have a special meaning,
  - The Address 127.0.0.1 is reserved for the Local Host. This means it is used as a loop back address
  - Any address ending with all ones (after the subnet mask), is a broadcast address within this subnet
    - Example,

      In the Subnet 192.168.224.0/19 the broadcast address would be, 192.168.255.255
  - Any address of the format xxx.xxx.xxx.1 is reserved for the default gateway
  - The following subnets are reserved for private addressing (to be discussed later)

Subnet	Starting from	Ending at
10.0.0.0/8	10.0.0.0	10.255.255.255
172.16.0.0/12	172.16.0.0	172.31.255.255
192.168.0.0/16	192.168.0.0	192.168.255.255

OK…So How is Routing Done ??

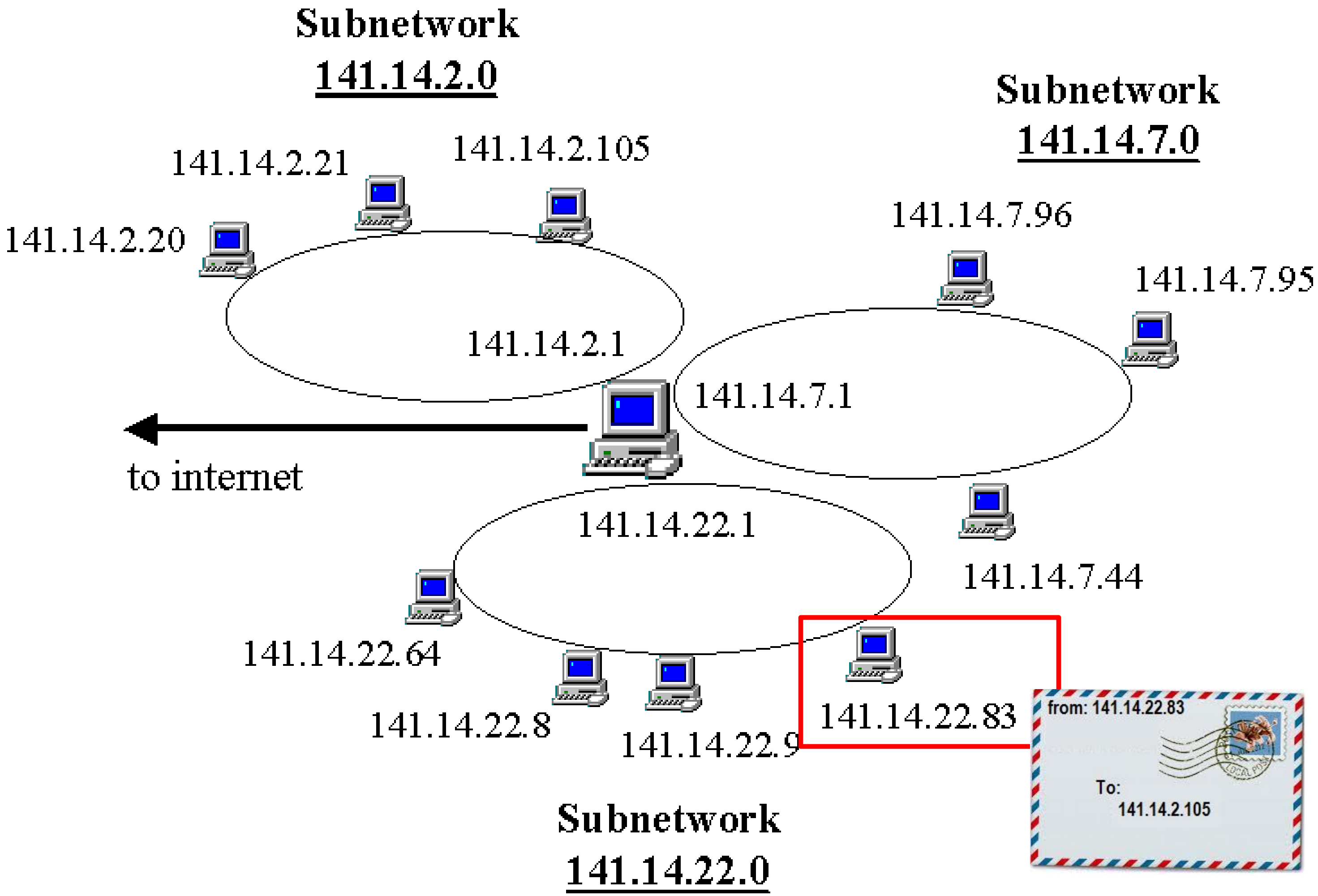
OK…So How is Routing Done ??



OK…So How is Routing Done ??

- First the destination address is checked to be within the same **subnet** of the sender or not
  - If it is on the same subnet,
    - Then delivering the packet will need to be done based on the MAC address
    - This means we will need to know the MAC address of the destination
    - We will not need this now, since the destination address does not belong to the sender subnet
  - If it is not on the same subnet,
    - Then we will need to deliver the packet to the proper next hop
    - The next hop is identified based on the **Routing Table**
    - The routing table is a table of routes that identifies the next hop based on the packet destination

OK…So How is Routing Done ??



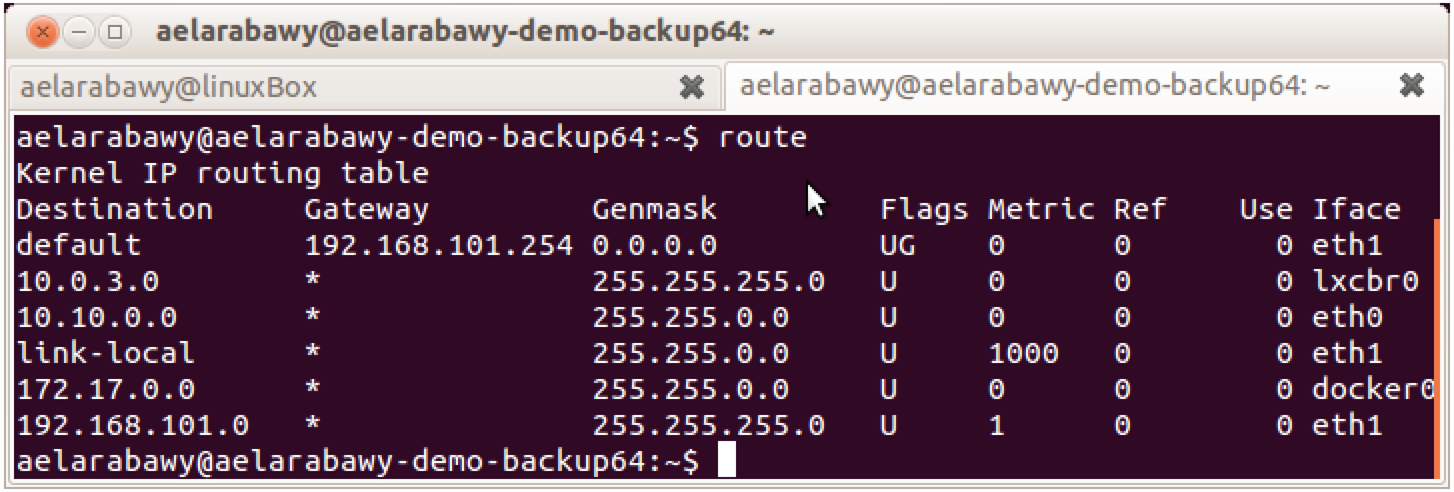
Show the Routing Table (route Command)

$ route

- The route command shows the entries of the routing table

  $ route
- Each route contains,
  - **Destination:** This is an address, whether a host address or a subnet address
  - **Network mask:** Used when a subnet address is used for destination
  - **Interface:** The used interface for this destination
  - **Gateway:** The next hop for this destination
- Upon the reception of a packet,
  - Linux tests the packet destination address to match the destination of the routes
  - If a match exists, Linux forwards the packet to the defined interface, towards the specified GW address

Show the Routing Table (route Command)


aelarabawy@aelarabawy-demo-backup64: ~
aelarabawy@linuxBox
aelarabawy@aelarabawy-demo-backup64: ~

```
aelarabawy@aelarabawy-demo-backup64:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.101.254 0.0.0.0         UG    0      0        0 eth1
10.0.3.0        *               255.255.255.0   U     0      0        0 lxcbr0
10.10.0.0       *               255.255.0.0     U     0      0        0 eth0
link-local      *               255.255.0.0     U     1000   0        0 eth1
172.17.0.0      *               255.255.0.0     U     0      0        0 docker0
192.168.101.0   *               255.255.255.0   U     1      0        0 eth1
aelarabawy@aelarabawy-demo-backup64:~$ 
```

Manage Routing Table (route command)

```
$ route add <address> dev <interface> gw <address>
$ route add default gw <address>
$ route del <address>
$ route del default
```

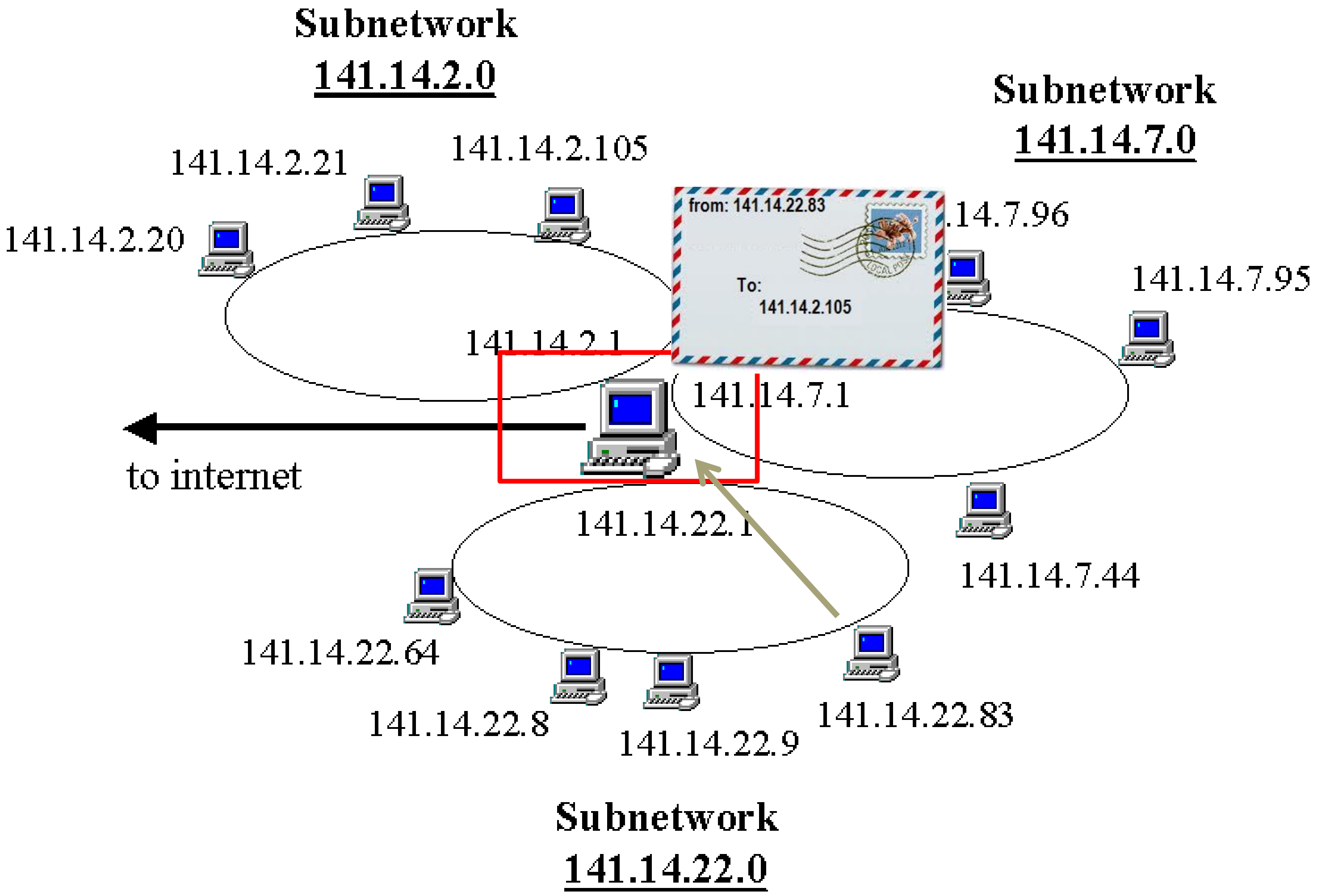
- To add a route

```
$ sudo route add 192.56.76.123 dev eth1 gw 192.168.101.1
$ sudo route add -net 192.56.76.0  netmask 255.255.255.0 dev eth0
$ sudo route add default gw 192.168.101.1
```

- To delete a route

```
$ sudo route del default
$ sudo route del 192.168.100.20
```

OK…So How is Routing Done ??



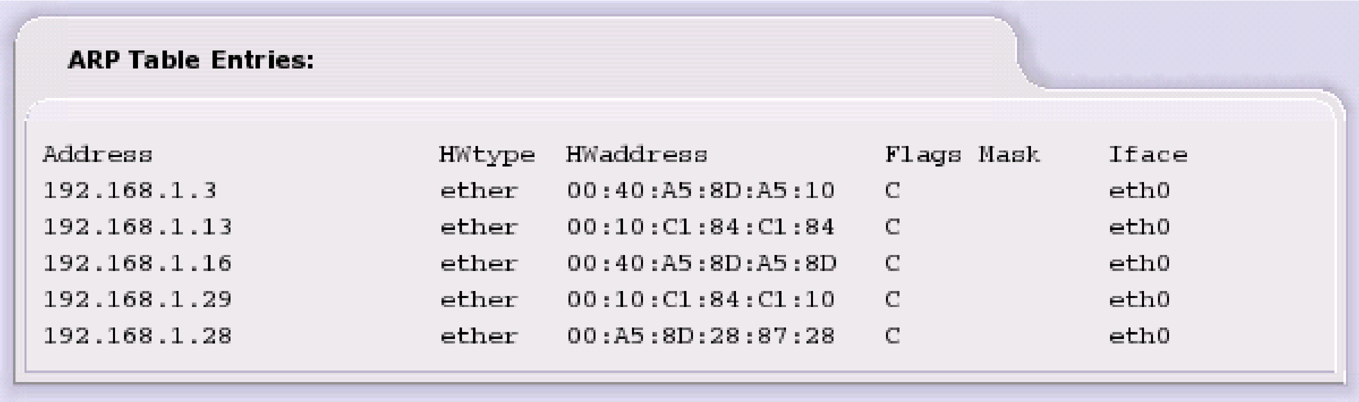
OK…So how is Routing Done ??

- Now the router has received the packet
- It should also check its **routing table** to identify the proper interface to use for forwarding the packet
- The destination is found to belong to the subnet 141.14.2.0/24, and the interface is identified
- Now we need to deliver the packet to its destination within the subnet
- Since delivering the packet within the subnet is based on the MAC Address, we will need to know the MAC address of the destination

What is the MAC Address for the Host 141.14.2.105 ???

Identifying the MAC Address … (The ARP Protocol)

- The machine should have a table that maps IP Addresses to MAC addresses within its subnet
- This table is called the ARP table

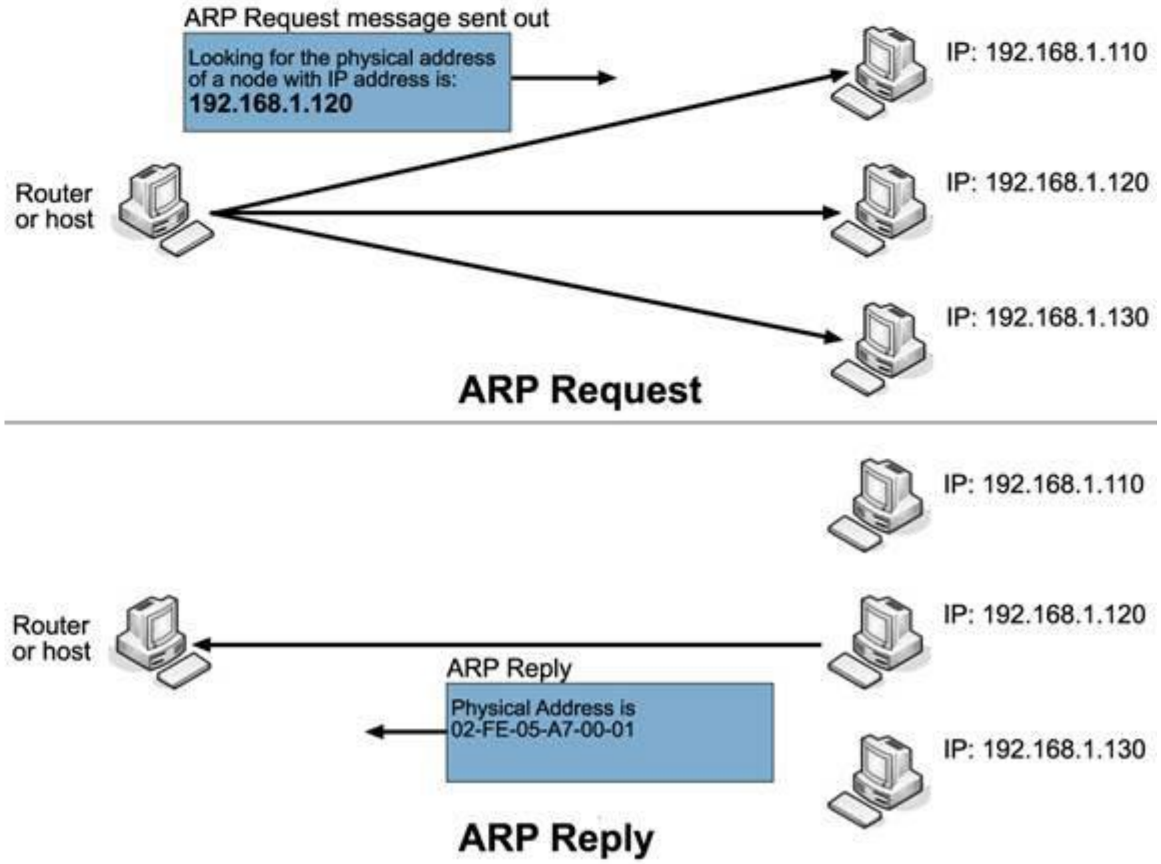


ARP Table Entries:

```
Address                  HWtype  HWaddress           Flags Mask     Iface
192.168.1.3              ether   00:40:A5:8D:A5:10   C              eth0
192.168.1.13             ether   00:10:C1:84:C1:84   C              eth0
192.168.1.16             ether   00:40:A5:8D:A5:8D   C              eth0
192.168.1.29             ether   00:10:C1:84:C1:10   C              eth0
192.168.1.28             ether   00:A5:8D:28:87:28   C              eth0
```

- If the machine does not have the address in its table, it queries it using the **ARP protocol** (Address Resolution Protocol)

ARP Protocol



Manage the ARP Table (arp Command)

```
$ arp
$ arp -s <IP Address> <Hardware Address>
$ arp -d <IP Address>
```

- The ***arp*** command is responsible for display and management of the ARP table
- To Display the ARP Table

```
$ arp
```

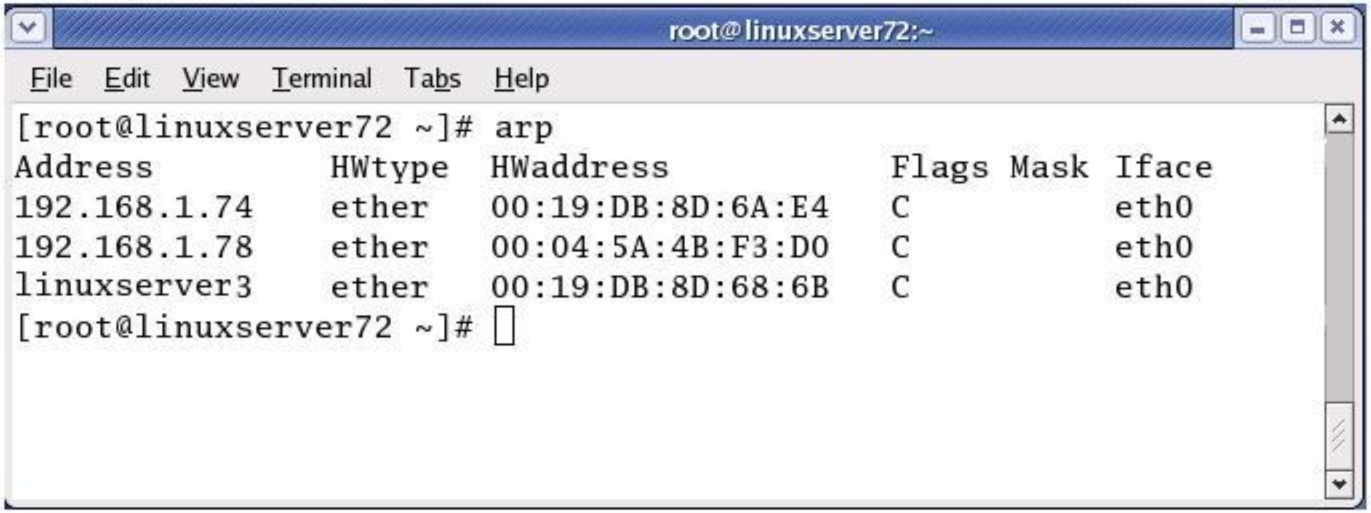
- To manually add an entry to the ARP Table (normally ARP Table entries are filled by the ARP Protocol)

```
$ sudo arp -s 192.168.101.105   00:0C:F1:AC:34:F1
```

- To manually delete an entry to the ARP Table (normally ARP table entries expire automatically)

```
$ sudo arp -d 192.168.101.105
```

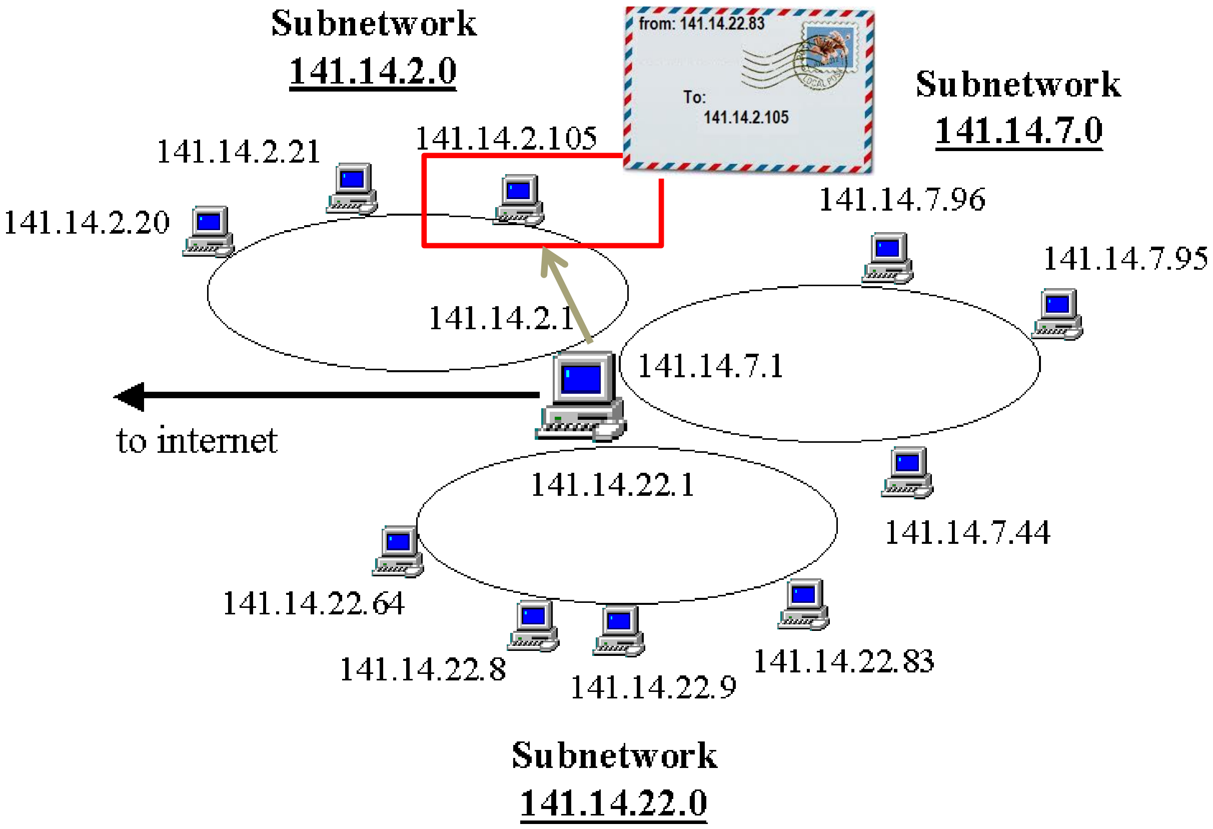
Show the ARP Table (arp Command)



```
[root@linuxserver72 ~]# arp
Address          HWtype  HWaddress           Flags Mask  Iface
192.168.1.74     ether   00:19:DB:8D:6A:E4   C           eth0
192.168.1.78     ether   00:04:5A:4B:F3:D0   C           eth0
linuxserver3     ether   00:19:DB:8D:68:6B   C           eth0
[root@linuxserver72 ~]# 
```

OK…So How is Routing Done ??

- Now the machine knows the MAC Address of the destination, and it can deliver the packet to it

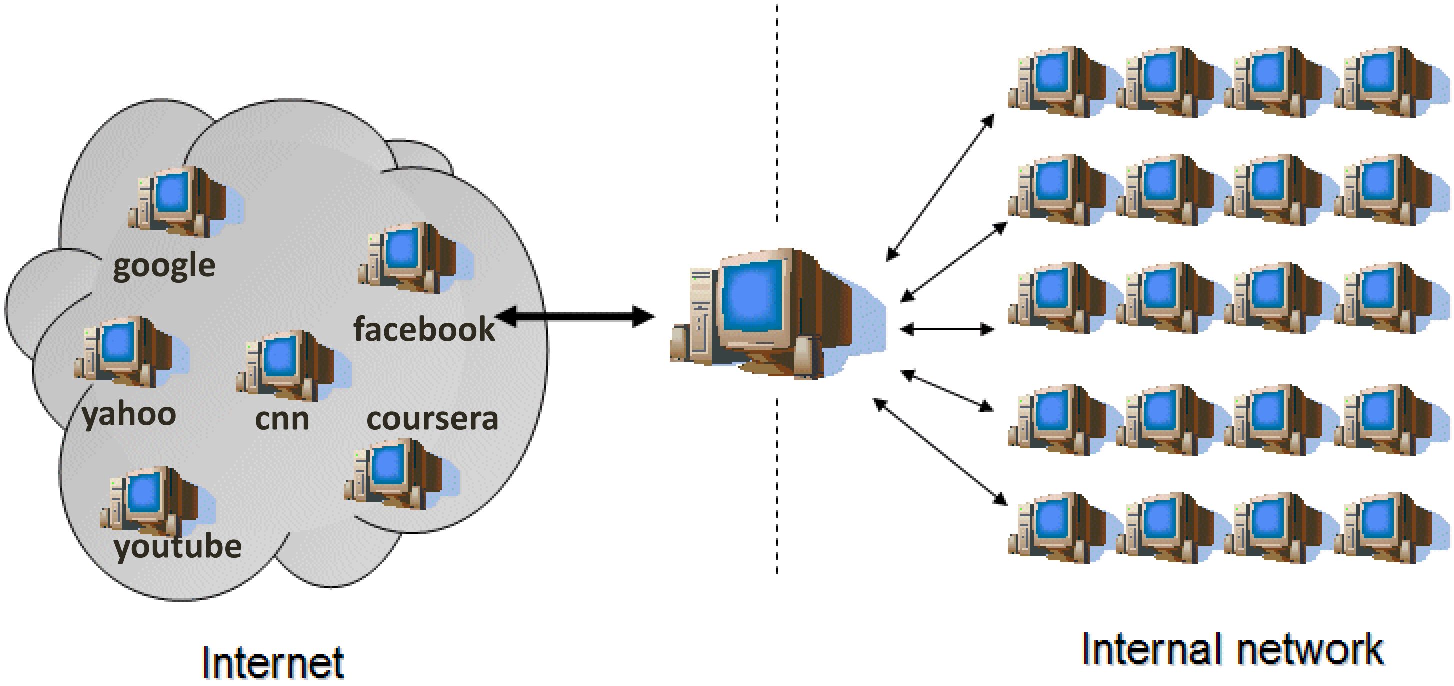


Internal IP Addressing and NATing

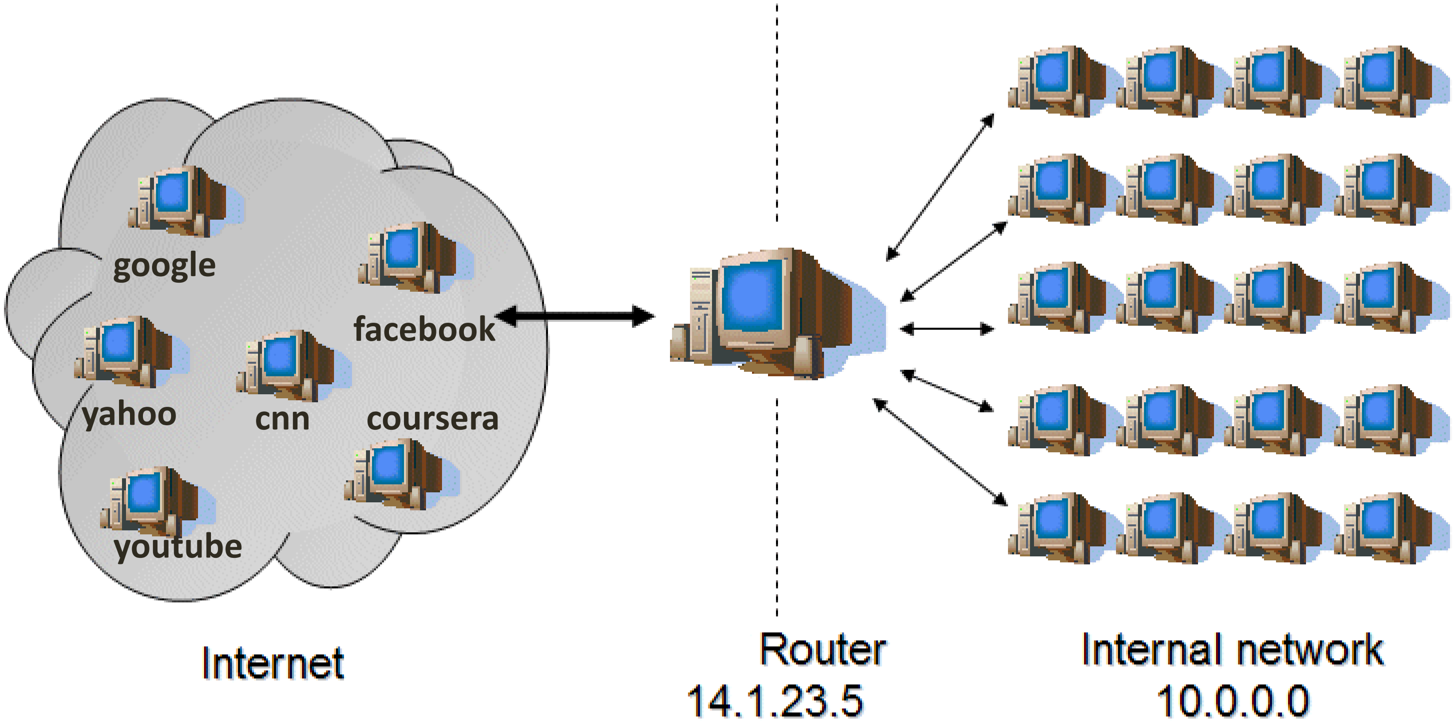
Why NATing ??

Internal network

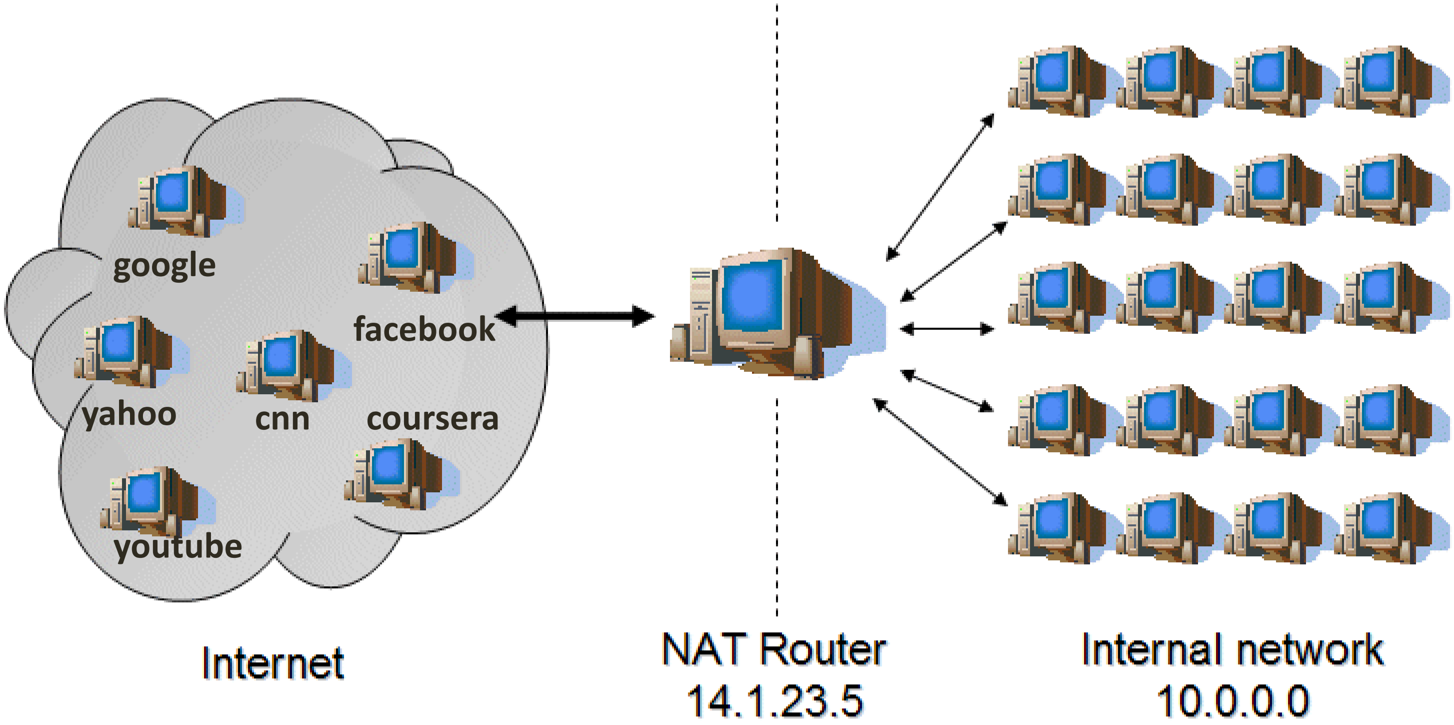
Why NATing ??

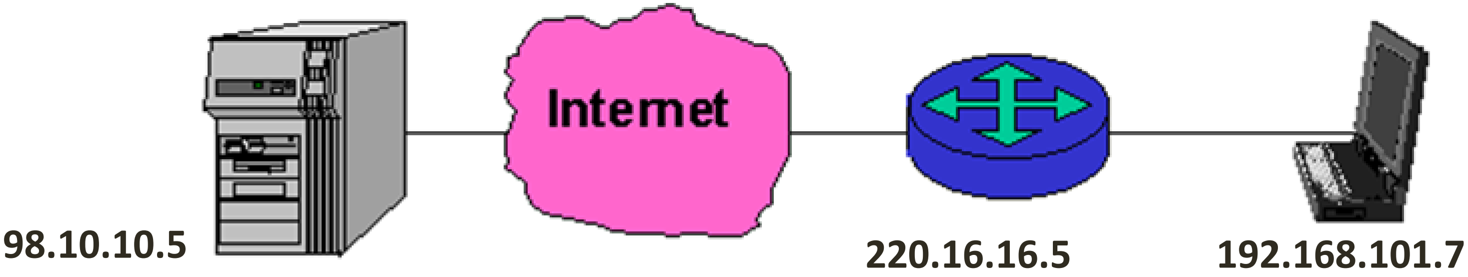

google
facebook
yahoo
cnn
coursera
youtube
Internet
Internal network

Why NATing ??


google
facebook
yahoo
cnn
coursera
youtube
Internet
Router
14.1.23.5
Internal network
10.0.0.0

Why NATing ??

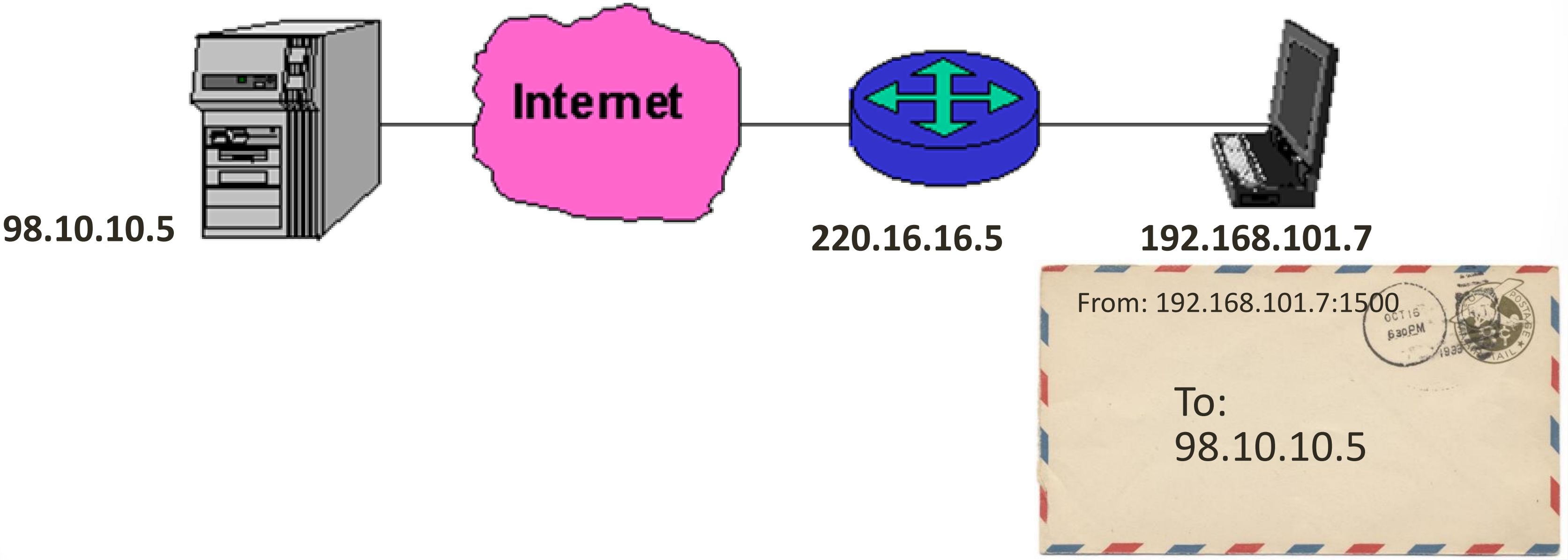

google
facebook
yahoo
cnn
coursera
youtube
Internet
NAT Router
14.1.23.5
Internal network
10.0.0.0



98.10.10.5

220.16.16.5

192.168.101.7



98.10.10.5

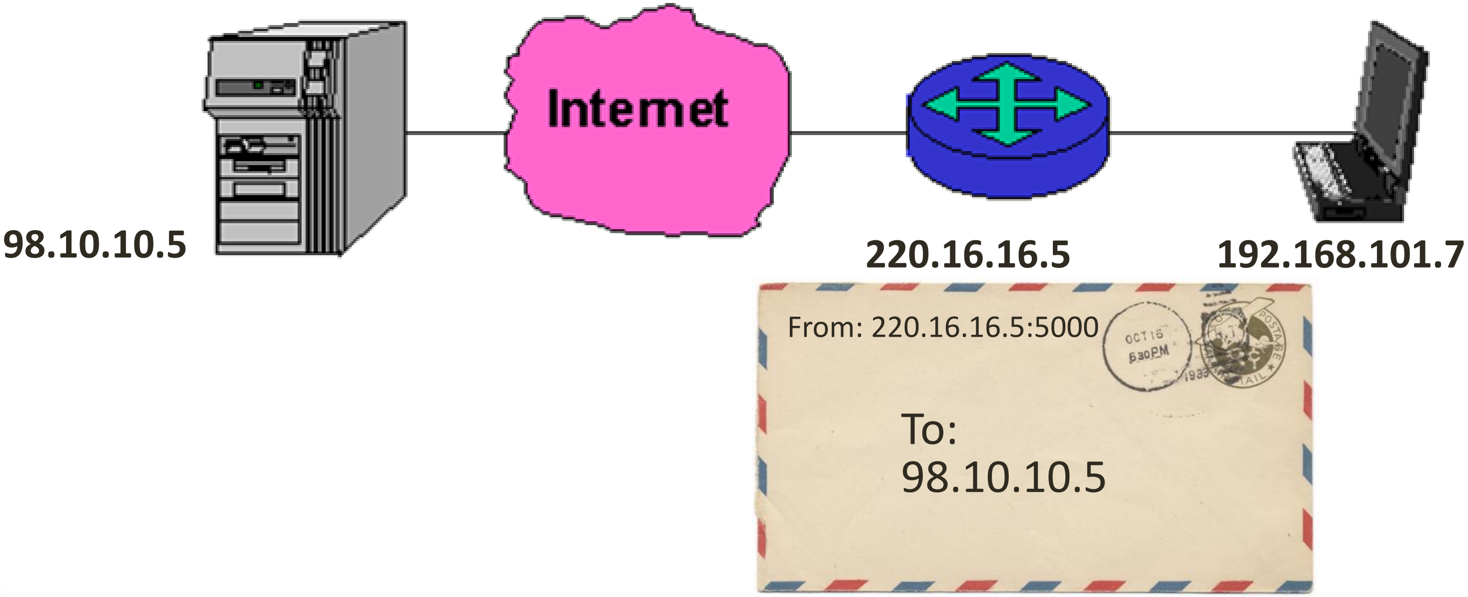
220.16.16.5

192.168.101.7

From: 192.168.101.7:1500

To:
98.10.10.5

Public		Private	
220.16.16.5	5000	192.168.101.7	1500



98.10.10.5

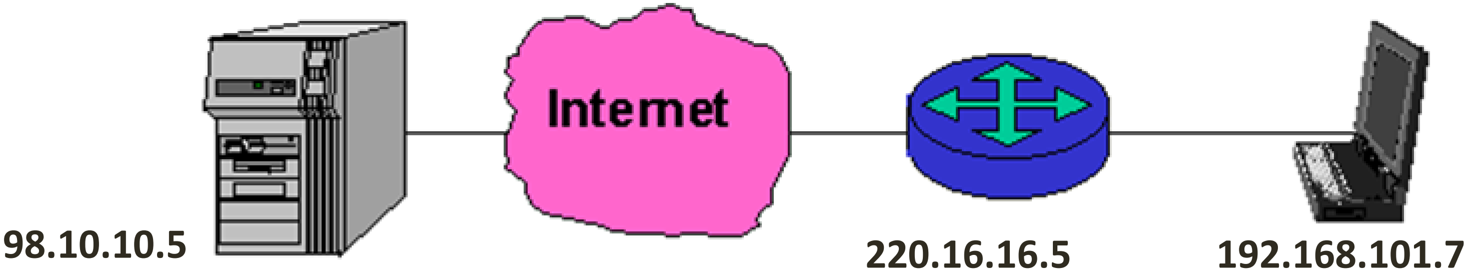
220.16.16.5

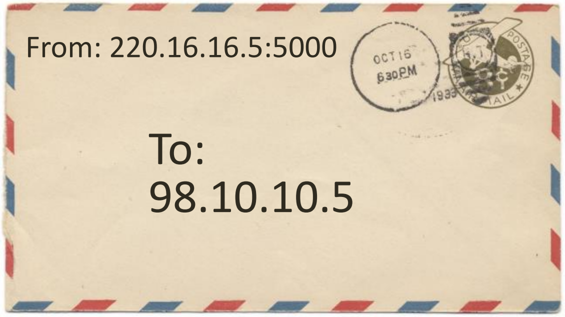
192.168.101.7

From: 220.16.16.5:5000

To:
98.10.10.5

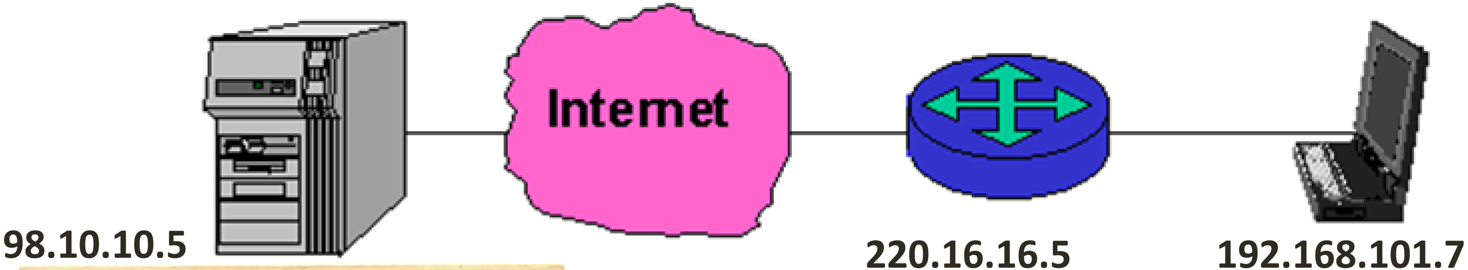
Public		Private	
220.16.16.5	5000	192.168.101.7	1500

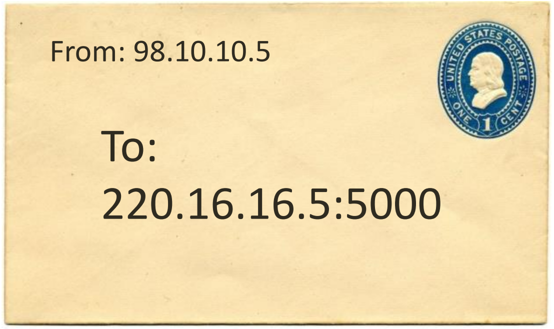



From: 220.16.16.5:5000

To:
98.10.10.5

Public		Private	
220.16.16.5	5000	192.168.101.7	1500

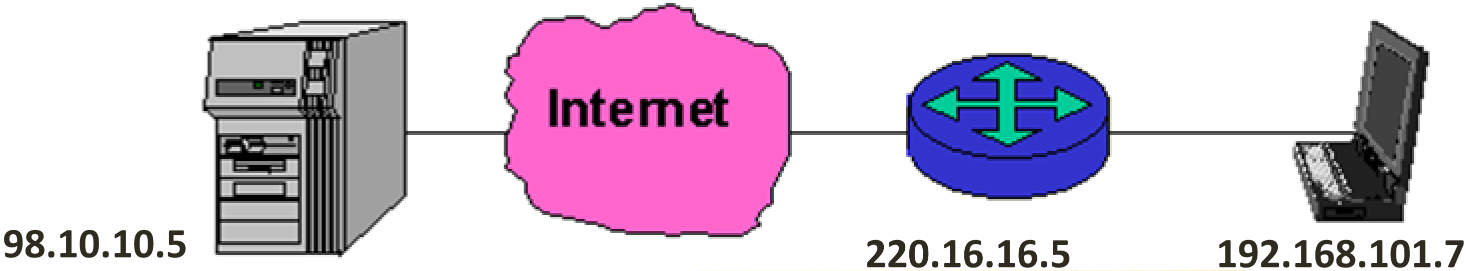


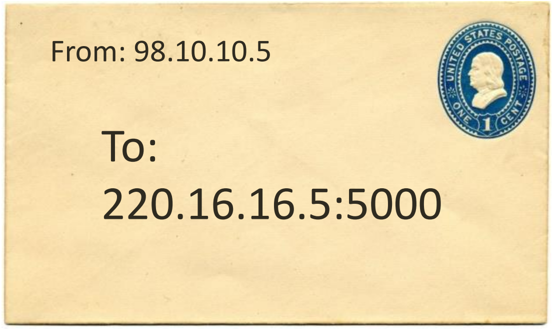


From: 98.10.10.5

To:
220.16.16.5:5000

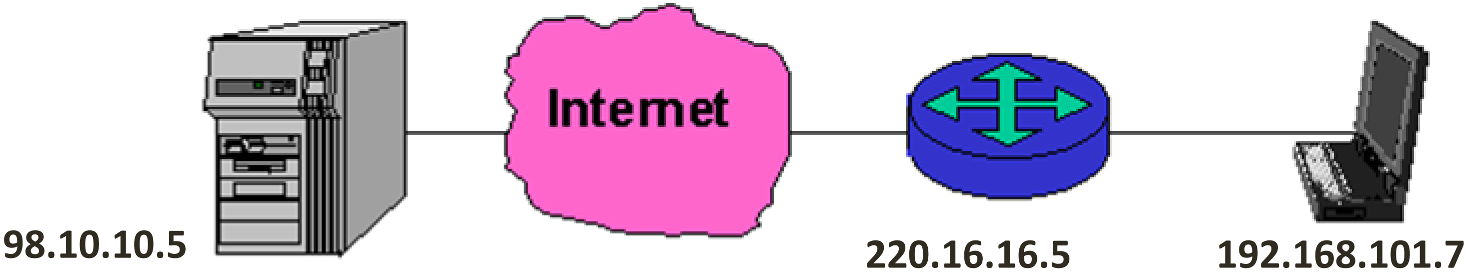
Public		Private	
220.16.16.5	5000	192.168.101.7	1500

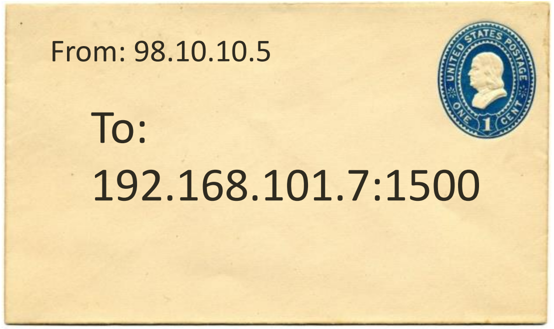



From: 98.10.10.5

To:
220.16.16.5:5000

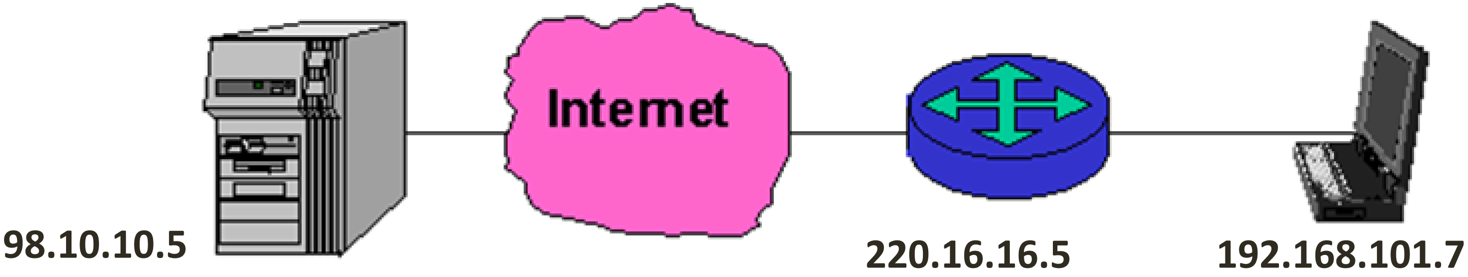
Public		Private	
220.16.16.5	5000	192.168.101.7	1500

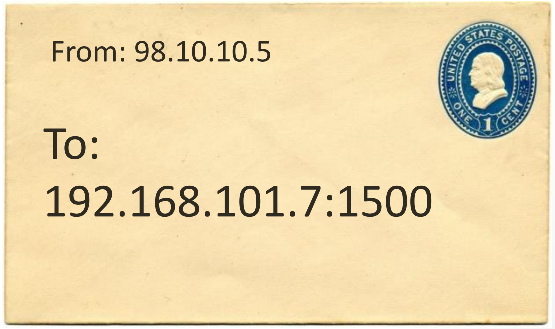



From: 98.10.10.5

To:
192.168.101.7:1500

Public		Private	
220.16.16.5	5000	192.168.101.7	1500




From: 98.10.10.5

To:
192.168.101.7:1500

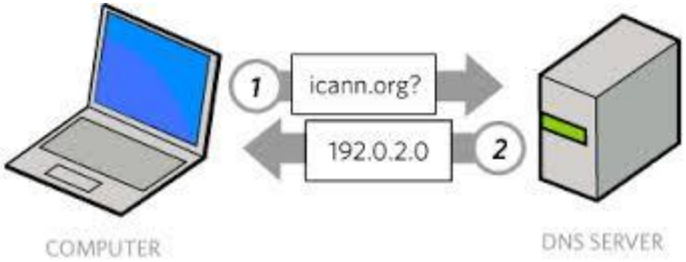
Network Address Translation

- NAT (Network Address Translation) is used to hide internal network from the rest of the network to,
  - Reduce the required number of administered IP Addresses
  - Protect the internal network from access from the external network
- The NAT Router builds a mapping table between the Internal addresses/ports, and its own address/ports (or from some list of administered addresses)
  - The external network sees only the external addresses
  - V
    r
- The
  te
  add

Subnet	Starting from	Ending at
10.0.0.0/8	10.0.0.0	10.255.255.255
172.16.0.0/12	172.16.0.0	172.31.255.255
192.168.0.0/16	192.168.0.0	192.168.255.255

Domain Names

- When accessing a server, it is normally identified by a domain name instead of IP Address,
  - More readable (www.google.com instead of 134.11.234.102)
  - Better portability (we can change the server IP Address)
  - Enable High Availability (Using a standby server in case of failure)
  - Enable Load sharing (Distribution of load among multiple server)
- We need a way to convert the domain name into the IP Address
- This is achieved through DNS Server



Setting DNS Server Info (/etc/resolv.conf)

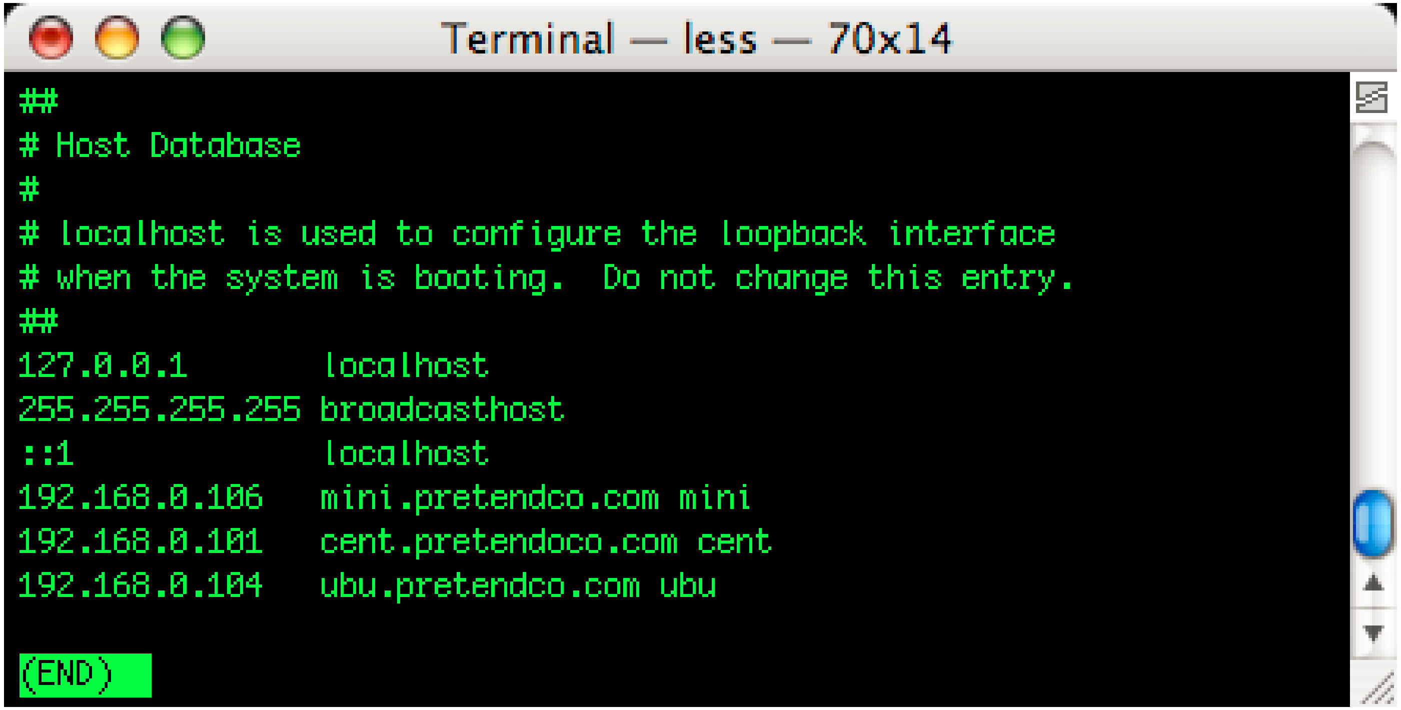
- To set the set of name servers to search for mapping Domain names to IP Address, edit the file **/etc/resolv.conf**
- Name servers are specified as follows,

```
nameserver 208.67.222.222
nameserver 208.67.220.220
```

- Any changes made to the file will take effect immediately

Machine Hostname Aliases (/etc/hosts)

- The file */etc/hosts* contains a list of aliases to hosts



```
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1       localhost
255.255.255.255 broadcasthost
::1             localhost
192.168.0.106   mini.pretendco.com mini
192.168.0.101   cent.pretendoco.com cent
192.168.0.104   ubu.pretendco.com ubu

(END)
```

Resolving Domain Names (host command)

$ host <Domain Name>

- This command is used to lookup a domain name and return with its IP Address(es)
- Example:

```
$ host www.google.com
```

- Note that the ***dig*** command can do a similar job

```
$ dig www.google.com
```

Linux4

Embedded Systems

http://Linux4EmbeddedSystems.com