Southern Connecticut State University

SCSU

CSC424 System Administration

Instructor: Dr. Hao Wu

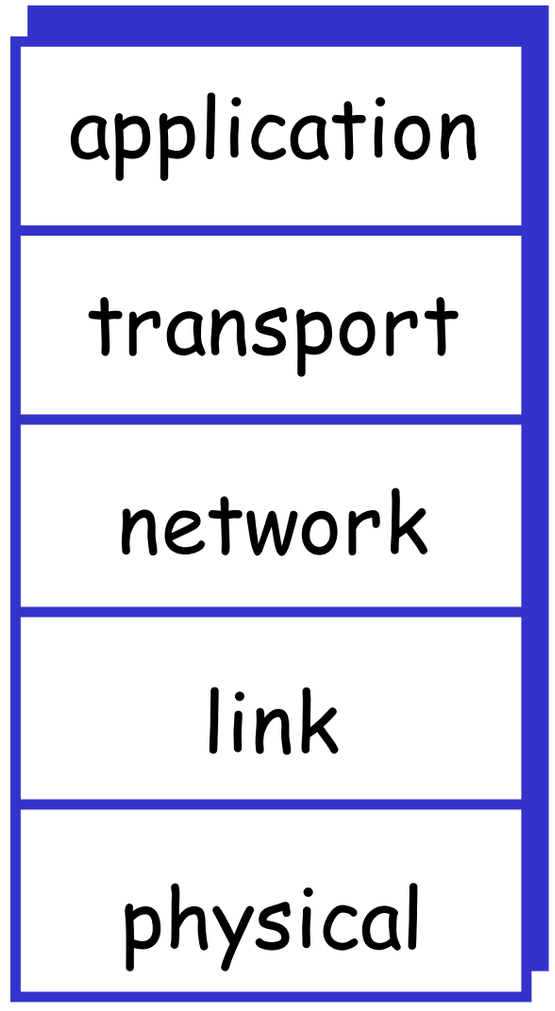
Week 11 Network and System Monitoring

Networking Basics

- Internet protocol layers
- MAC
- IP
- TCP/UDP
- Routing

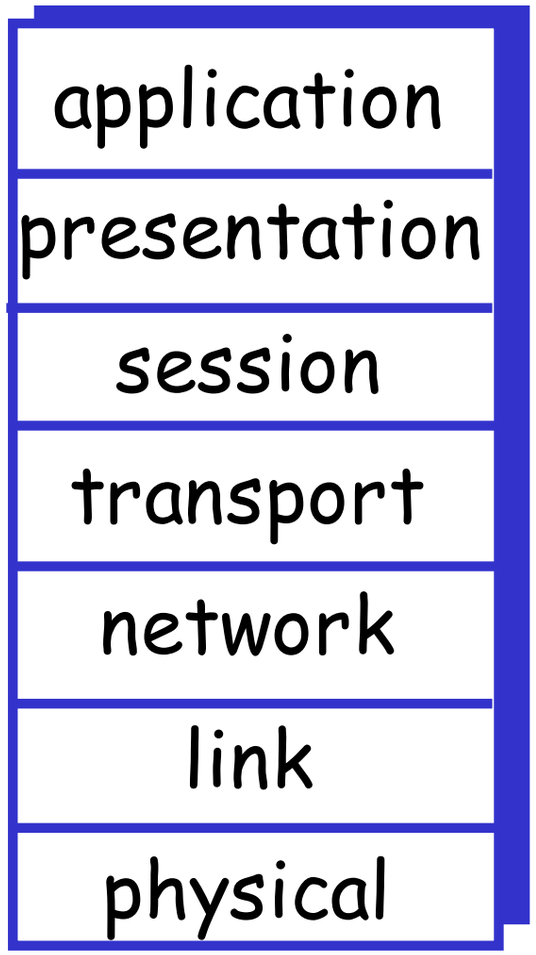
Internet protocol stack

- *application:* supporting network applications
  - FTP, SMTP, HTTP
- *transport:* process-process data transfer
  - TCP, UDP
- *network:* routing of datagrams from source to destination
  - IP, routing protocols
- *link:* data transfer between neighboring network elements
  - Ethernet, 802.111 (WiFi), PPP
- *physical:* bits "on the wire"

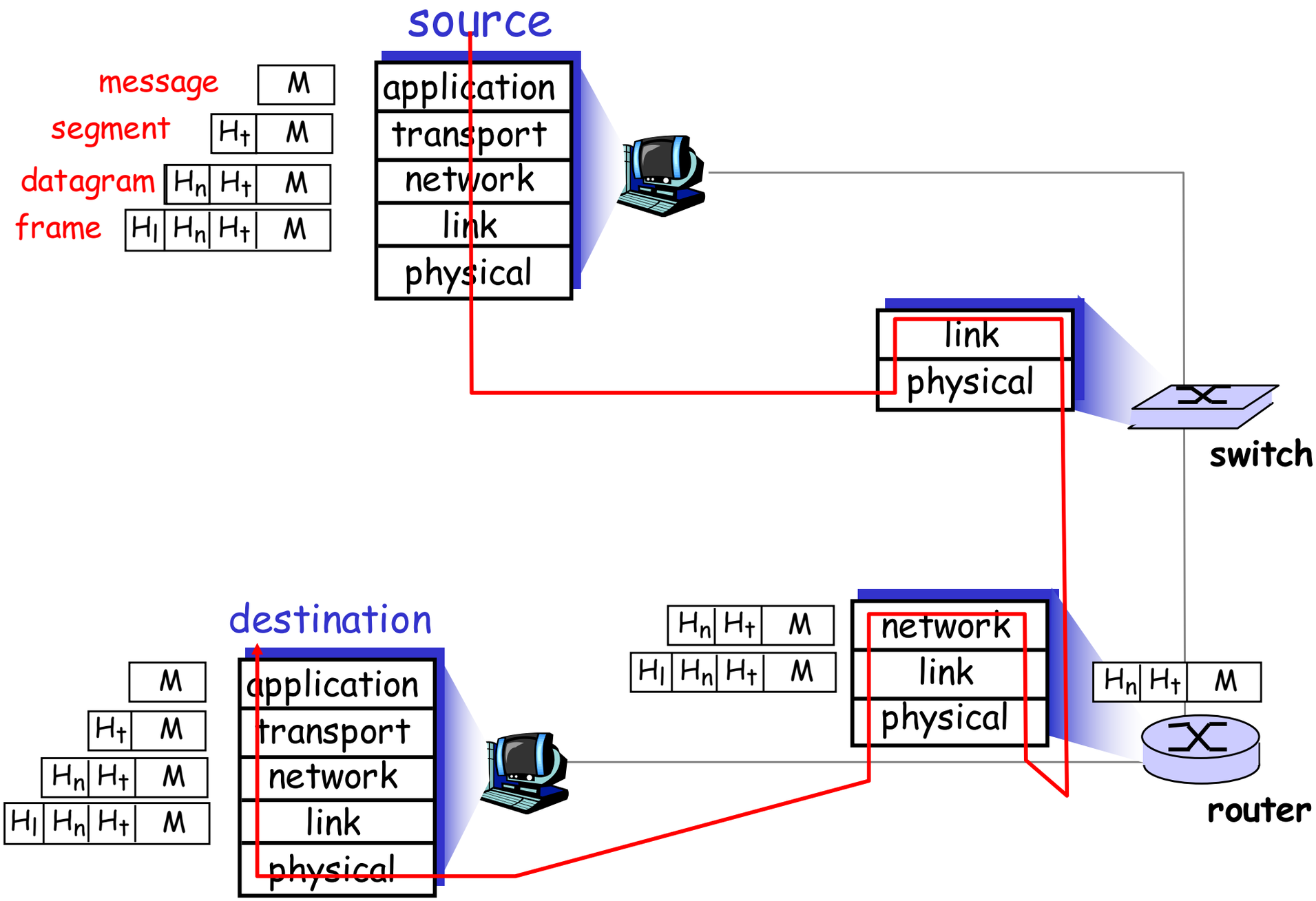


ISO/OSI (Open Systems Interconnection) reference model

- presentation: allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- *session:* synchronization, checkpointing, recovery of data exchange
- Internet stack "missing" these layers!
  - these services, *if needed*, must be implemented in application
  - needed?



Encapsulation

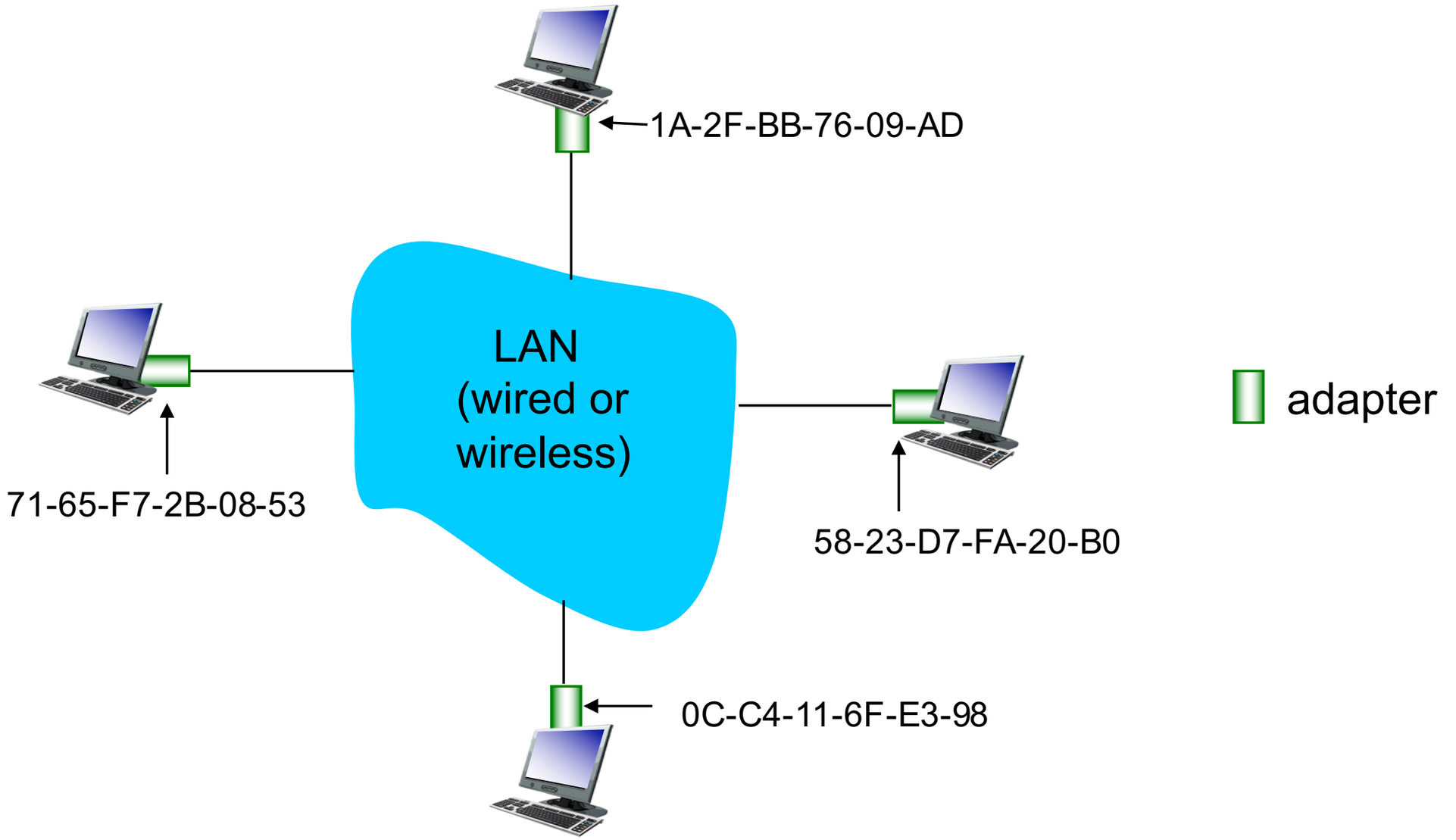


MAC addresses and ARP

- 32-bit IP address:
  - *network-layer* address for interface
  - used for layer 3 (network layer) forwarding
- MAC (or LAN or physical or Ethernet) address:
  - function: *used 'locally" to get frame from one interface to another physically-connected interface (same network, in IP-addressing sense)*
  - 48 bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
  - e.g.: 1A-2F-BB-76-09-AD

LAN addresses and ARP

each adapter on LAN has unique *LAN* address

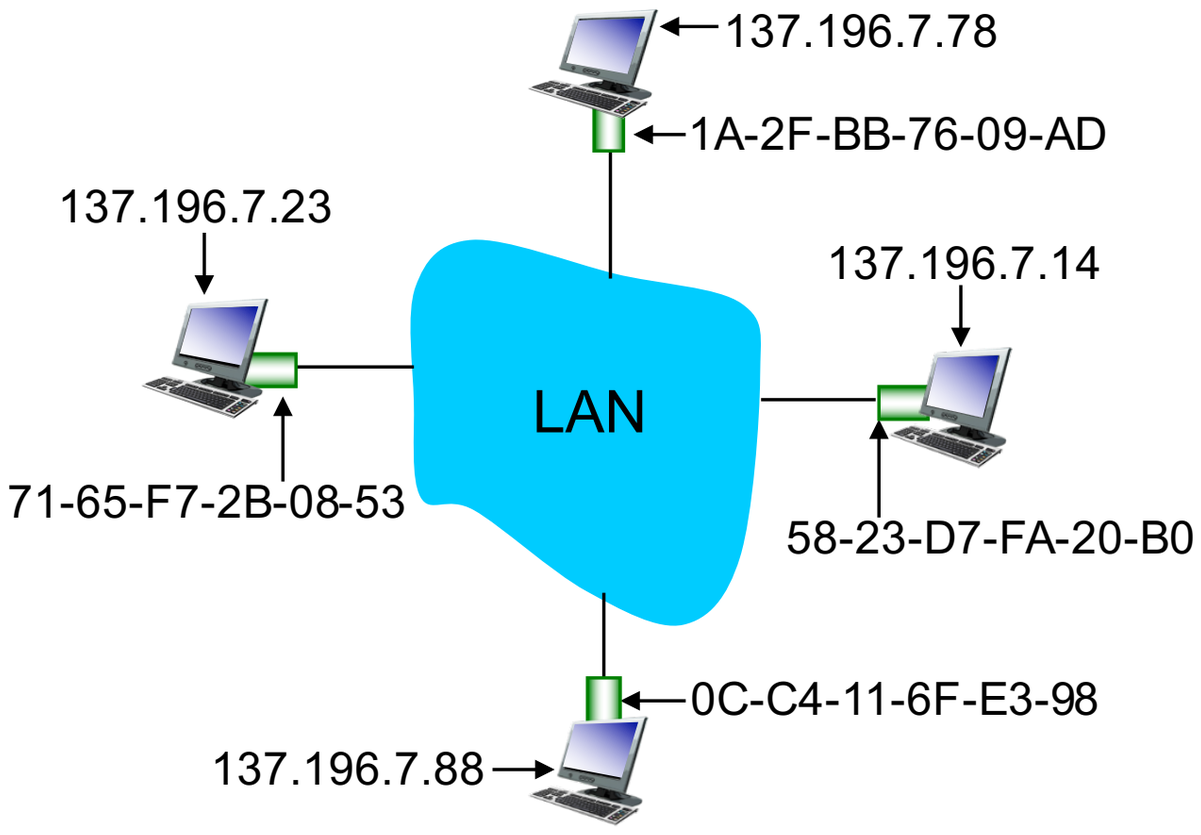


LAN addresses (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
  - MAC address: like Social Security Number
  - IP address: like postal address
- MAC flat address ➜ portability
  - can move LAN card from one LAN to another
- IP hierarchical address *not* portable
  - address depends on IP subnet to which node is attached

ARP: address resolution protocol

Question: how to determine interface's MAC address, knowing its IP address?



ARP table: each IP node (host, router) on LAN has table

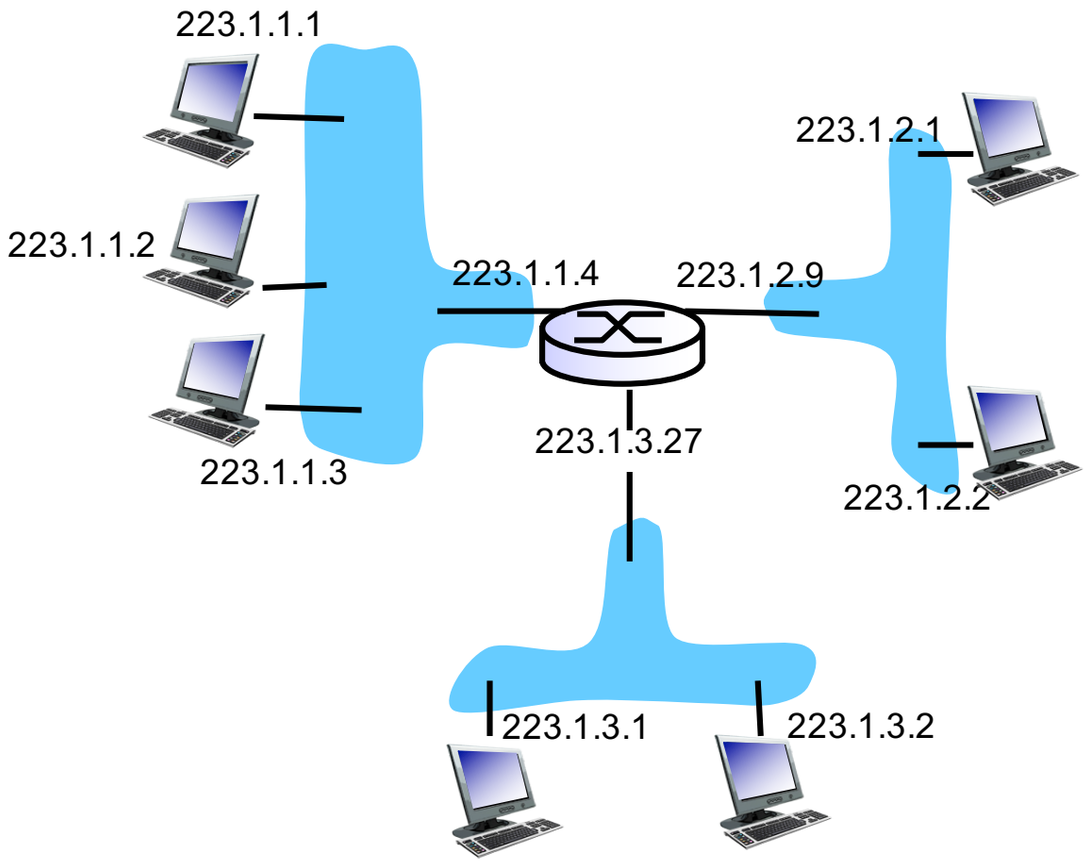
- IP/MAC address mappings for some LAN nodes:

  < IP address; MAC address; TTL>

- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

IP addressing: introduction

- *IP address:* 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
- *IP addresses associated with each interface*

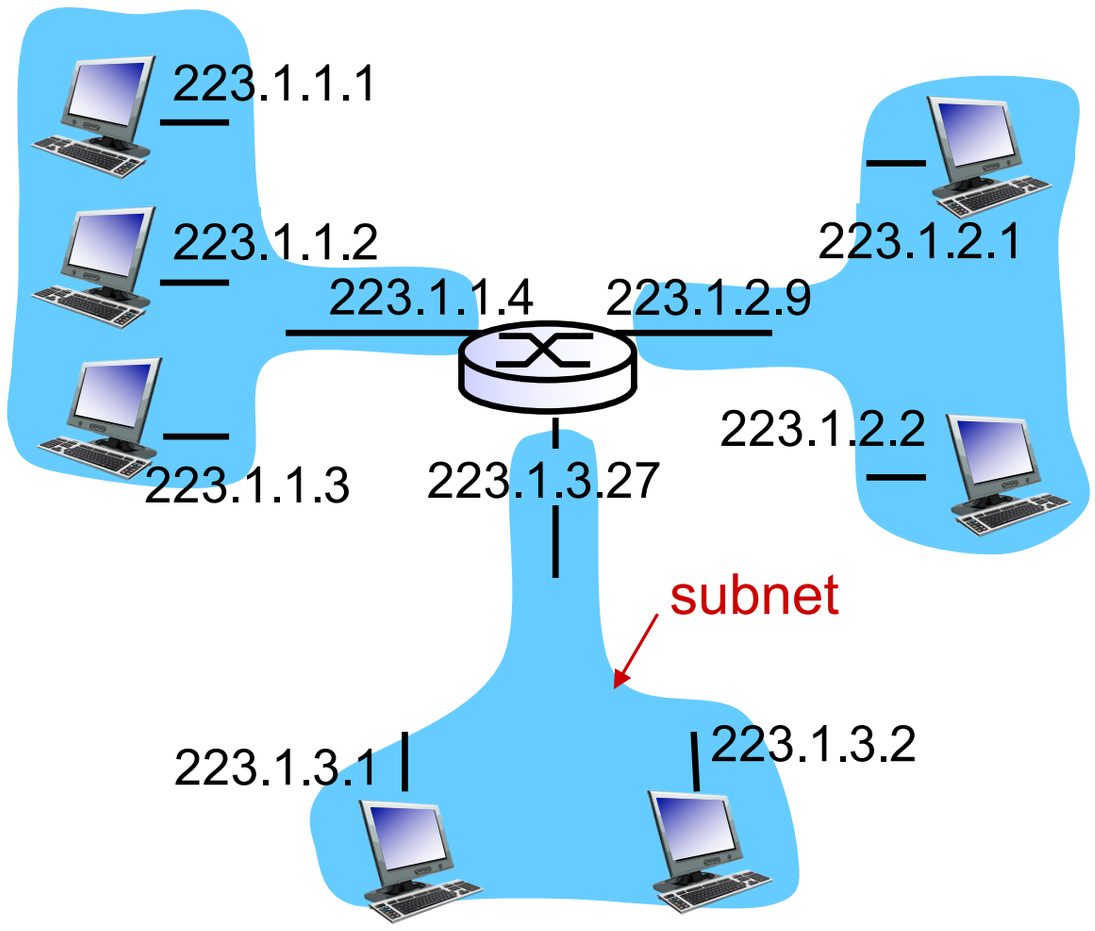


223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

Subnets

- IP address:
  - subnet part - high order bits
  - host part - low order bits
- *what's a subnet ?*
  - device interfaces with same subnet part of IP address
  - can physically reach each other *without intervening router*

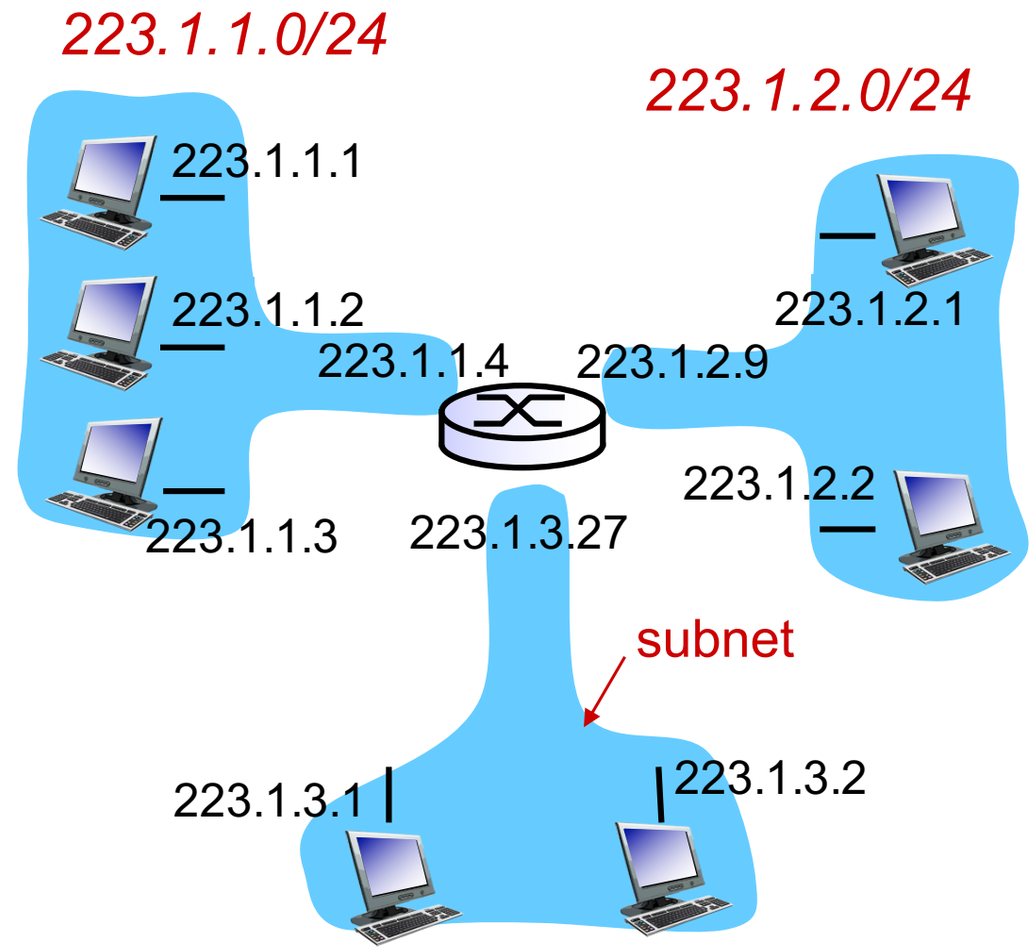


network consisting of 3 subnets

Subnets

recipe

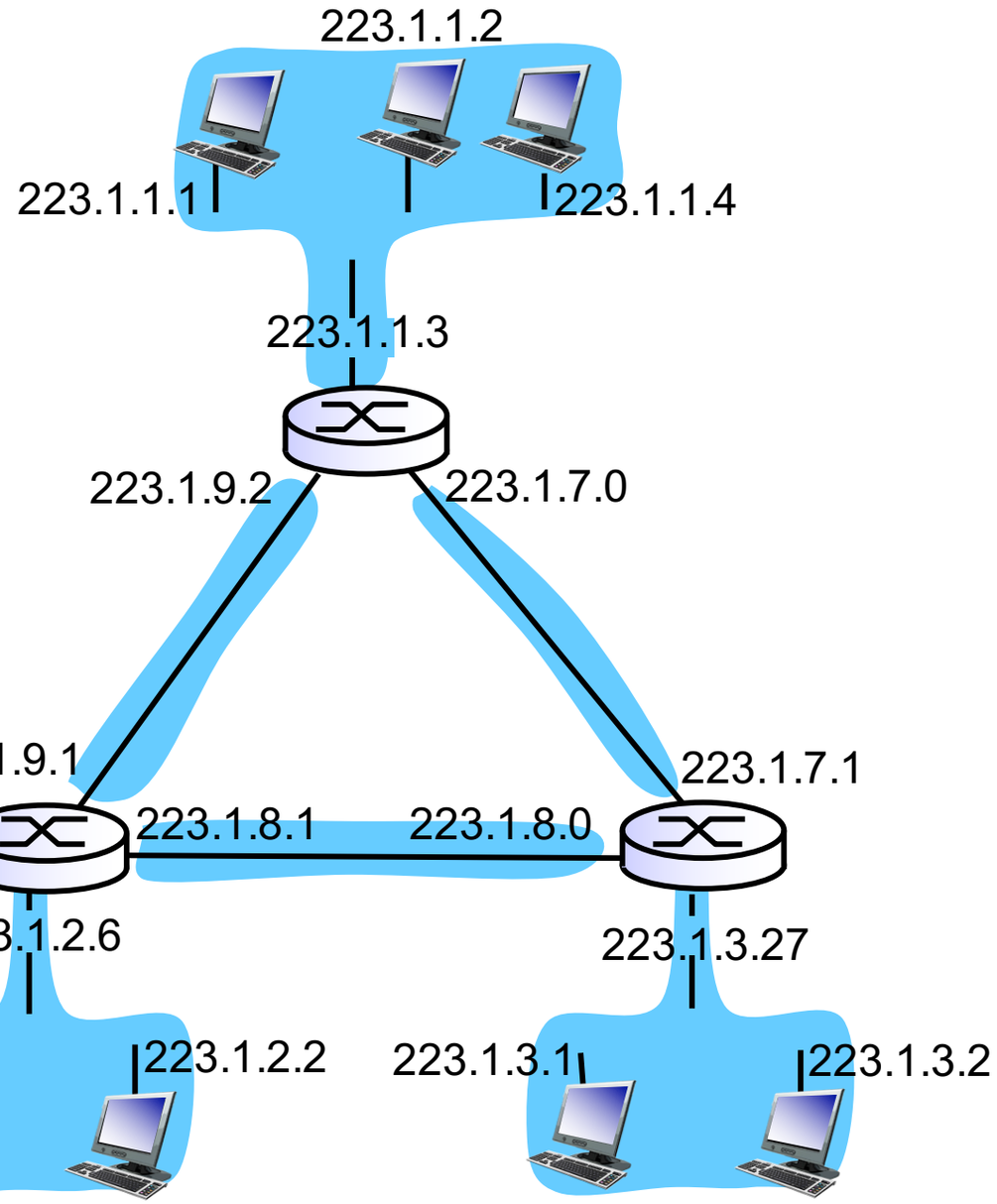
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a *subnet*



subnet mask: /24

Subnets

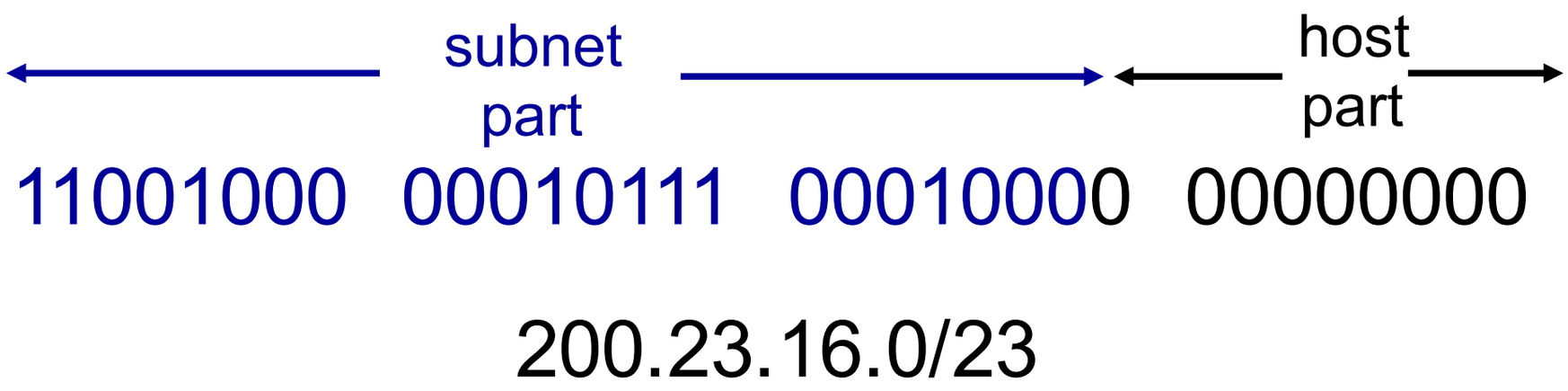
how many?



IP addressing: CIDR

CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: a.b.c.d/x, where x is # bits in subnet portion of address



200.23.16.0/23

IP addresses: how to get one?

Q: How does a *host* get IP address?

- hard-coded by system admin in a file
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

DHCP: Dynamic Host Configuration Protocol

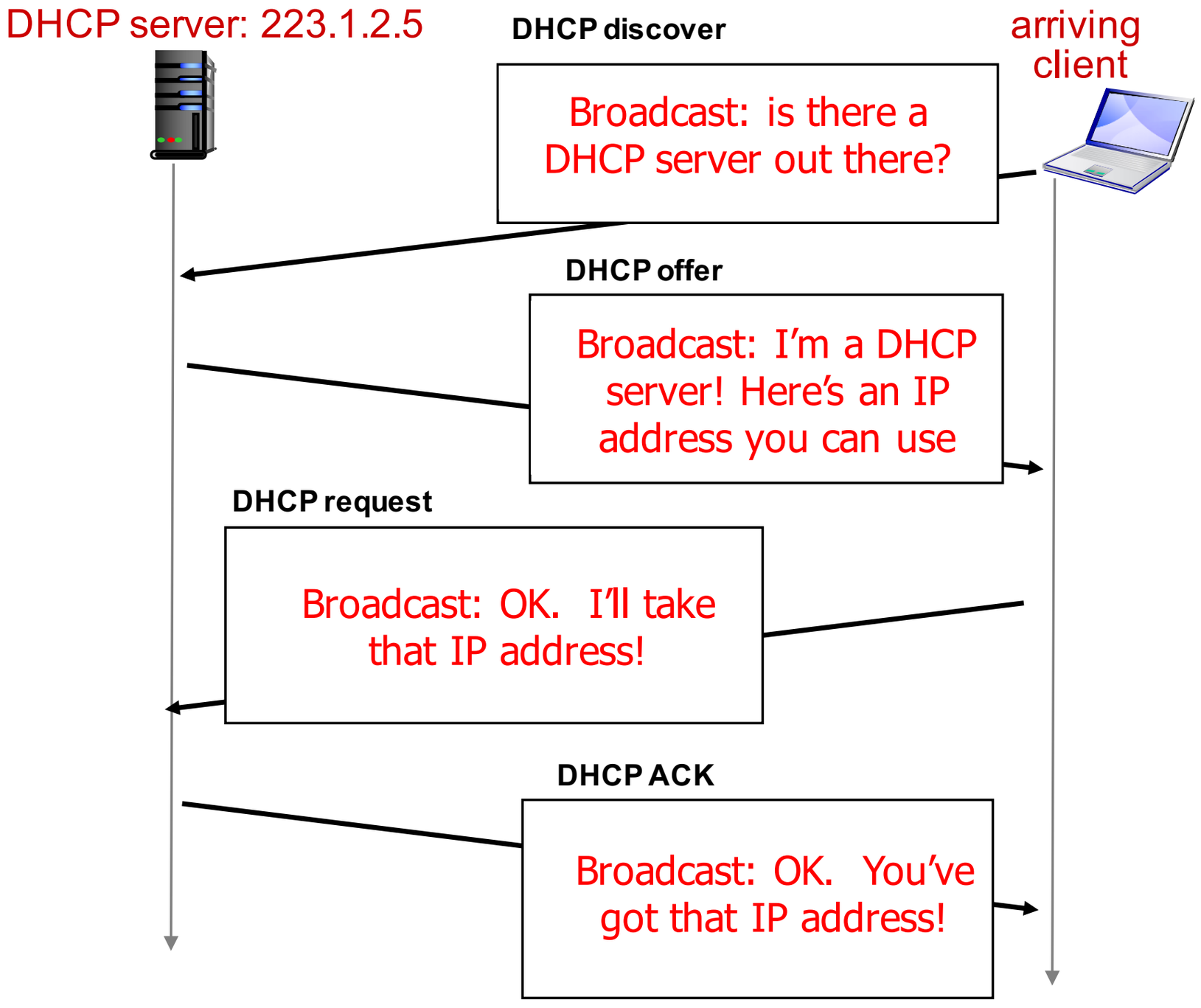
goal: allow host to *dynamically* obtain its IP address from network server when it joins network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/"on")
- support for mobile users who want to join network (more shortly)

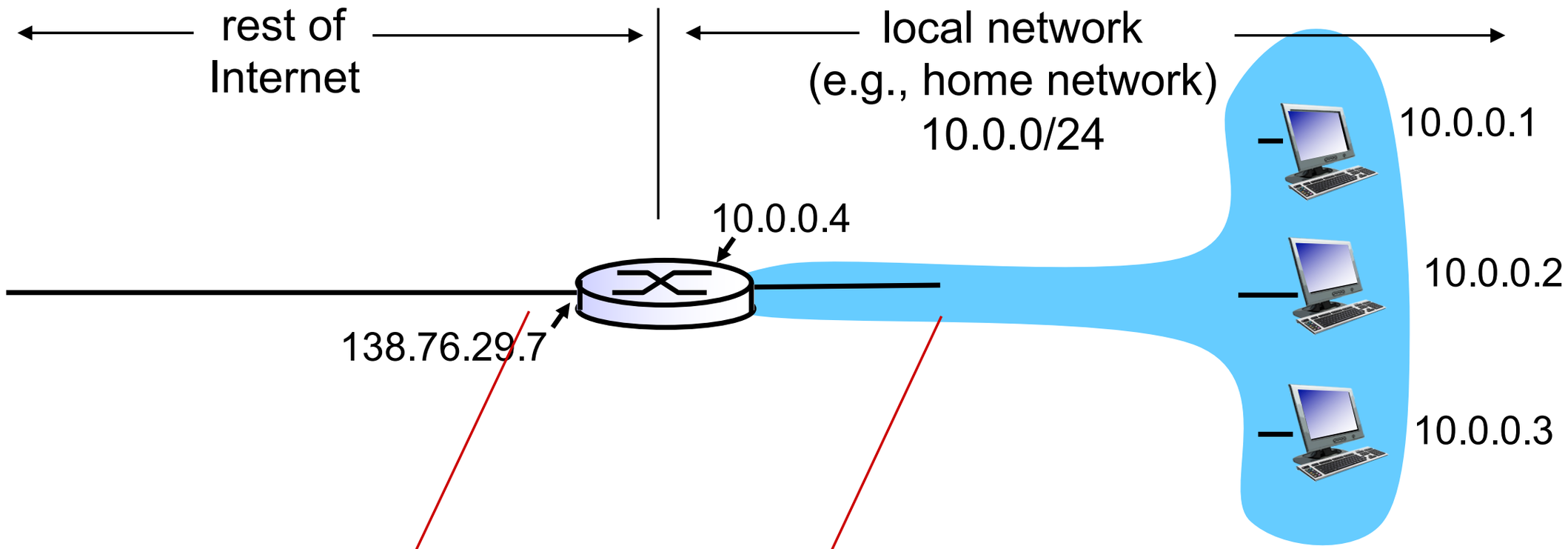
DHCP overview:

- host broadcasts "DHCP discover" msg [optional]
- DHCP server responds with "DHCP offer" msg [optional]
- host requests IP address: "DHCP request" msg
- DHCP server sends address: "DHCP ack" msg

DHCP client-server scenario



NAT: network address translation



all datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7,different source port numbers

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: network address translation

motivation: local network uses just one IP address as far as outside world is concerned:

- range of addresses not needed from ISP: just one IP address for all devices
- can change addresses of devices in local network without notifying outside world
- can change ISP without changing addresses of devices in local network
- devices inside local net not explicitly addressable, visible by outside world (a security plus)

NAT: network address translation

implementation: NAT router must:

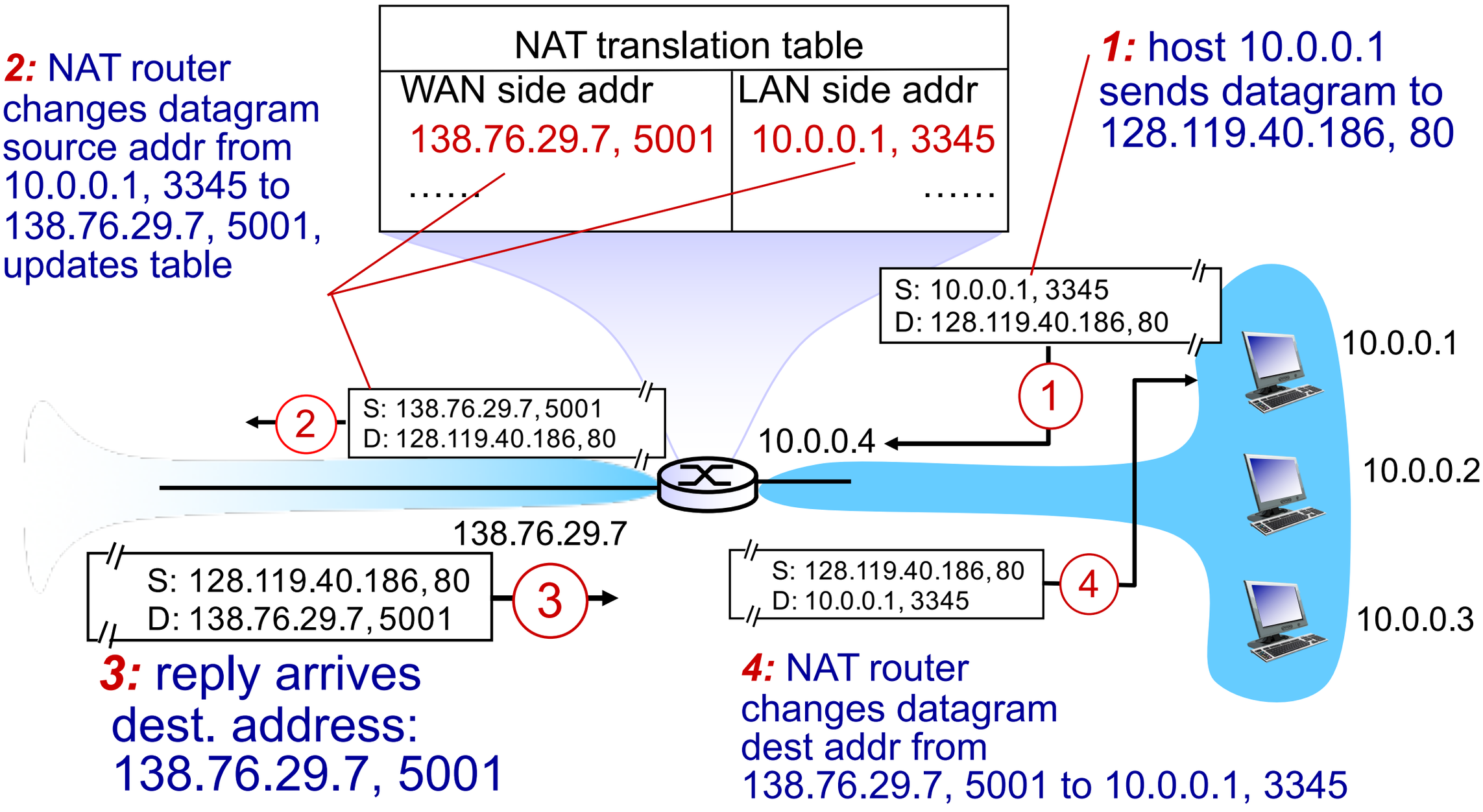
- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)

  . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr

- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP address, new port #) translation pair

- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation



NAT: network address translation

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
  - routers should only process up to layer 3
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - address shortage should instead be solved by IPv6

IPv6: motivation

- *initial motivation:* 32-bit address space soon to be completely allocated.
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

IPv6 datagram format:

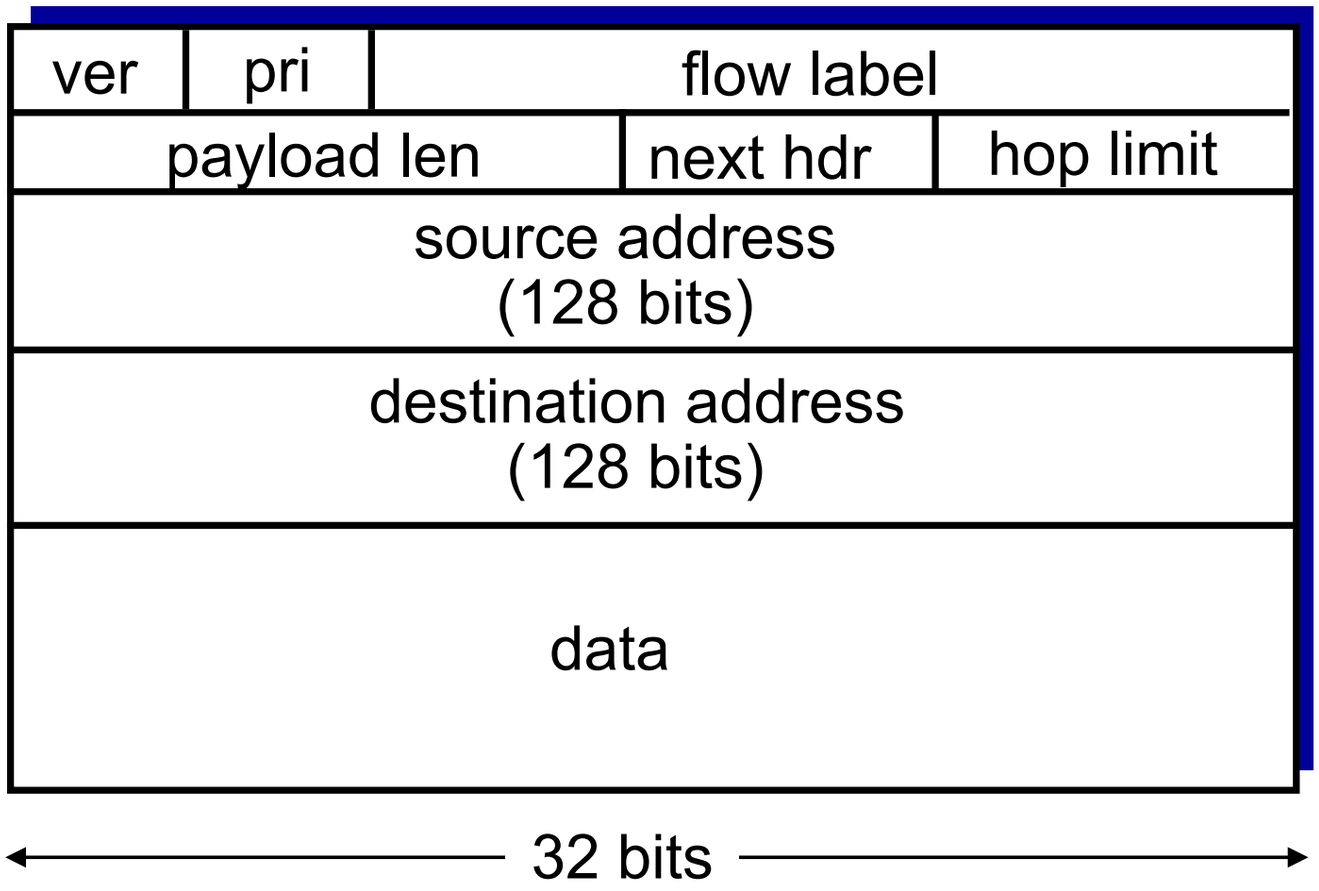
- fixed-length 40 byte header
- no fragmentation allowed

IPv6 datagram format

priority: identify priority among datagrams in flow

flow Label: identify datagrams in same "flow." (concept of"flow" not well defined).

next header: identify upper layer protocol for data



Network Interface Configuration

- Command for network interface configuration:
- ifconfig [option] [interface]
- If command not found, install the tool:

```
[root@localhost ~]# yum install net-tools
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.lug.udel.edu
 * extras: mirror.es.its.nyu.edu
 * updates: mirrors.lga7.us.voxel.net
```

Network Interface Configuration

- Usage: check all network interfaces:
- ifconfig -a

```
[root@localhost ~]# ifconfig -a
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 08:00:27:12:ea:9a  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.31  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::ae7e:dca9:dfa6:fabe  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:34:2b:0c  txqueuelen 1000  (Ethernet)
        RX packets 10374  bytes 14649660 (13.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1512  bytes 127449 (124.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Network Interface Configuration

- Understanding output information:

```
[root@localhost ~]# ifconfig enp0s8
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.31  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::ae7e:dca9:dfa6:fabe  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:34:2b:0c  txqueuelen 1000  (Ethernet)
        RX packets 10494  bytes 14663846 (13.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1566  bytes 134435 (131.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- enp0s8: name of the device
- UP: This flag indicates that the kernel modules related to the Ethernet interface has been loaded.
- BROADCAST: Denotes that the Ethernet device supports broadcasting - a necessary characteristic to obtain IP address via DHCP.
- RUNNING: The interface is ready to accept data

Network Interface Configuration

- Understanding output information:

```
[root@localhost ~]# ifconfig enp0s8
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.31  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::ae7e:dca9:dfa6:fabe  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:34:2b:0c  txqueuelen 1000  (Ethernet)
        RX packets 10494  bytes 14663846 (13.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1566  bytes 134435 (131.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- MULTICAST: This indicates that the Ethernet interface supports multicasting.
- mtu: short form for Maximum Transmission Unit is the size of each packet received by the Ethernet card.
- inet: IPv4 address
- inet6: IPv6 address
- ether: MAC address
- txqueuelen: This denotes the length of the transmit queue of the device.

Network Interface Configuration

- Understanding output information:

```
[root@localhost ~]# ifconfig enp0s8
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.31  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::ae7e:dca9:dfa6:fabe  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:34:2b:0c  txqueuelen 1000  (Ethernet)
        RX packets 10494  bytes 14663846 (13.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1566  bytes 134435 (131.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- RX and TX: total number of packets or bytes received and transmitted respectively.

Network Interface Configuration

- Change IP address using ifconfig:
- *ifconfig <interface> <address> netmask <netmask> [upldown]*
- *ifconfig <interface> <address> </prefixlen> [up l down]*
-

```
[root@localhost ~]# ifconfig enp0s3 192.168.2.3/24
[root@localhost ~]# ifconfig enp0s3
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.2.3  netmask 255.255.255.0  broadcast 192.168.2.255
        inet6 fe80::a00:27ff:fe12:ea9a  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:12:ea:9a  txqueuelen 1000  (Ethernet)
        RX packets 17  bytes 3135 (3.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 542  bytes 33328 (32.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Network Interface Configuration

- ifconfig:
  - changes network interface configurations temporarily
  - lose configurations after reboot
- Edit network configuration files:
  - Location: /etc/sysconfig/network-scripts/
  - Files start with "ifcfg-interfaceName" are configuration files for network interfaces

```
[root@localhost network-scripts]# ls
ifcfg-enp0s3  ifdown-isdn      ifdown-tunnel  ifup-isdn     ifup-Team
ifcfg-lo      ifdown-post      ifup           ifup-plip     ifup-TeamPort
ifdown        ifdown-ppp       ifup-aliases   ifup-plusb    ifup-tunnel
ifdown-bnep   ifdown-routes    ifup-bnep      ifup-post     ifup-wireless
ifdown-eth    ifdown-sit       ifup-eth       ifup-ppp      init.ipv6-global
ifdown-ippp   ifdown-Team      ifup-ippp      ifup-routes   network-functions
ifdown-ipv6   ifdown-TeamPort  ifup-ipv6      ifup-sit      network-functions-ipv6
```

Network Interface Configuration

- Understand the configuration file:
  - BOOTPROTO:
    - boot-time protocol used
    - none - No boot time protocol should be used
    - dhcp - The DHCP protocol should be used
    - static - Use static ip address
  - ONBOOT:
    - yes - This device should be activated at boot-time
    - no - This device should not be activated at boot-time

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=enp0s3
UUID=dd33654a-838f-4664-
abb4-38cf1b5980ac
DEVICE=enp0s3
ONBOOT=no
```

Network Interface Configuration

- Config a static IP interface:
  - IPADDR: IPv4 address
  - NETMASK: network mask

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
IPADDR=192.168.2.3
NETMASK=255.255.255.0
NAME=enp0s3
UUID=dd33654a-838f-4664-
abb4-38cf1b5980ac
DEVICE=enp0s3
ONBOOT=yes
~
```

Network Interface Configuration

- Take a network interface down:

```
[root@localhost network-scripts]# ifdown enp0s3
Device 'enp0s3' successfully disconnected.
```

- Bring a network interface up:

```
[root@localhost network-scripts]# ifup enp0s3
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/3)
```

Hostname

- Change your hostname by command:
  - Temporary

```
[root@localhost ~]# hostname
localhost.localdomain
[root@localhost ~]# hostname myhost.localdomain
[root@localhost ~]# hostname
myhost.localdomain
```

- Change your hostname by editing configuration file:
- Location: /etc/hostname

```
[root@localhost ~]# echo myhost.localhost > /etc/hostname
```

Hosts files

- /etc/hosts: file retains a mapping of host names and their ip addresses
-

```
[root@localhost ~]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
```

Network Tools

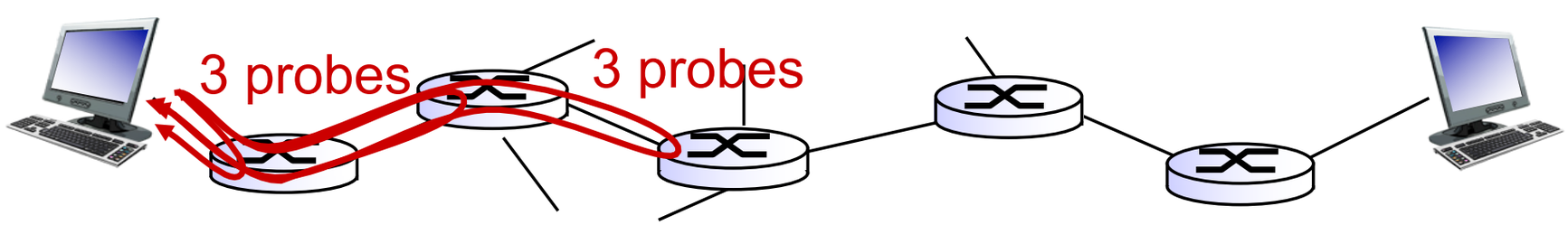
- ping: used to test connections between hosts
- traceroute: used to test connections between hosts and routers
- netstat: Print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships

Traceroute and ICMP

- source sends series of UDP segments to dest
  - first set has TTL =1
  - second set has TTL=2, etc.
  - unlikely port number
- when *n*th set of datagrams arrives to nth router:
  - router discards datagrams
  - and sends source ICMP messages (type 11, code 0)
  - ICMP messages includes name of router & IP address
- when ICMP messages arrives, source records RTTs

stopping criteria:

- UDP segment eventually arrives at destination host
- destination returns ICMP "port unreachable" message (type 3, code 3)
- source stops



netstat

```
[root@localhost ~]# netstat -tunpa
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      964/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      1066/master
tcp        0      0 192.168.1.31:22         192.168.1.8:61706       ESTABLISHED 1257/sshd: root@pts
tcp6       0      0 :::22                   :::*                    LISTEN      964/sshd
tcp6       0      0 ::1:25                  :::*                    LISTEN      1066/master
udp        0      0 0.0.0.0:68              0.0.0.0:*                           783/dhclient
udp        0      0 0.0.0.0:4317            0.0.0.0:*                           783/dhclient
udp        0      0 127.0.0.1:323           0.0.0.0:*                           644/chronyd
udp6       0      0 :::13571                :::*                                783/dhclient
udp6       0      0 ::1:323                 :::*                                644/chronyd
```

- Options:
  - a: Show both listening and non-listening sockets.
  - t: TCP
  - u: UDP
  - n: Show numerical addresses instead of trying to determine symbolic host, port or user names.
  - p: Show the PID and name of the program to which each socket belongs.