Lecture 6: Network Layer. Part 2

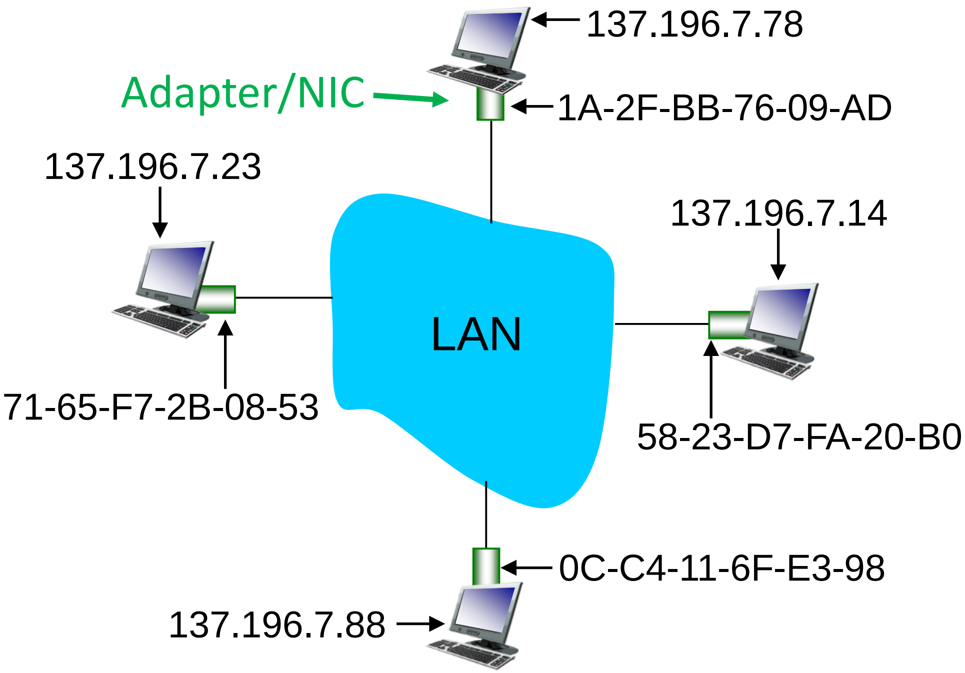
Acknowledgement: Materials presented in this lecture are predominantly based on slides from:

- *Computer Networking: A Top Down Approach,* J. Kurose, K. Ross, 6th ed., 2012, Addison-Wesley, Chapter 4
- *Business Data Communications and Networking,* J. Fitzgerald, A. Dennis, 10/11th ed., 2013, John Wiley & Sons, Chapter 5

Lecture 6: Network Layer. Part 2
Outline

- MAC Address Resolution
- Sending Messages using TCP/IP
  - Configuration file
- TCP/IP Flow of packets example:
  - Known Address, Same Subnet
  - Known Address, Different Subnets
  - Different Subnets, Unknown Addresses
- Hierarchical Addressing
- Network Address Translation (NAT)
- ICMP: internet control message protocol
- IPv6 packet format
- Tunneling

ARP: Address Resolution protocol



- The ARP table maintains mapping between the Physical and IP addresses
- Try **arp –a** in the command window

Recall from Lecture 3:

- Each adapter on LAN has unique MAC/PHY address
- Allocation administered by IEEE
- Approx: first three bytes: manufacturer, last three bytes: the product and its serial number
- PHY address is needed to send packets between the adjacent, or in the same subnet, computers

ARP protocol (RFC 826)

Host A:

- wants to send a packet to a host B
- knows the IP address of the host B
- checks that the host B is in the same subnet
- Does not know the PHY/MAC address of B
  - B’s MAC address not in A’s ARP table.

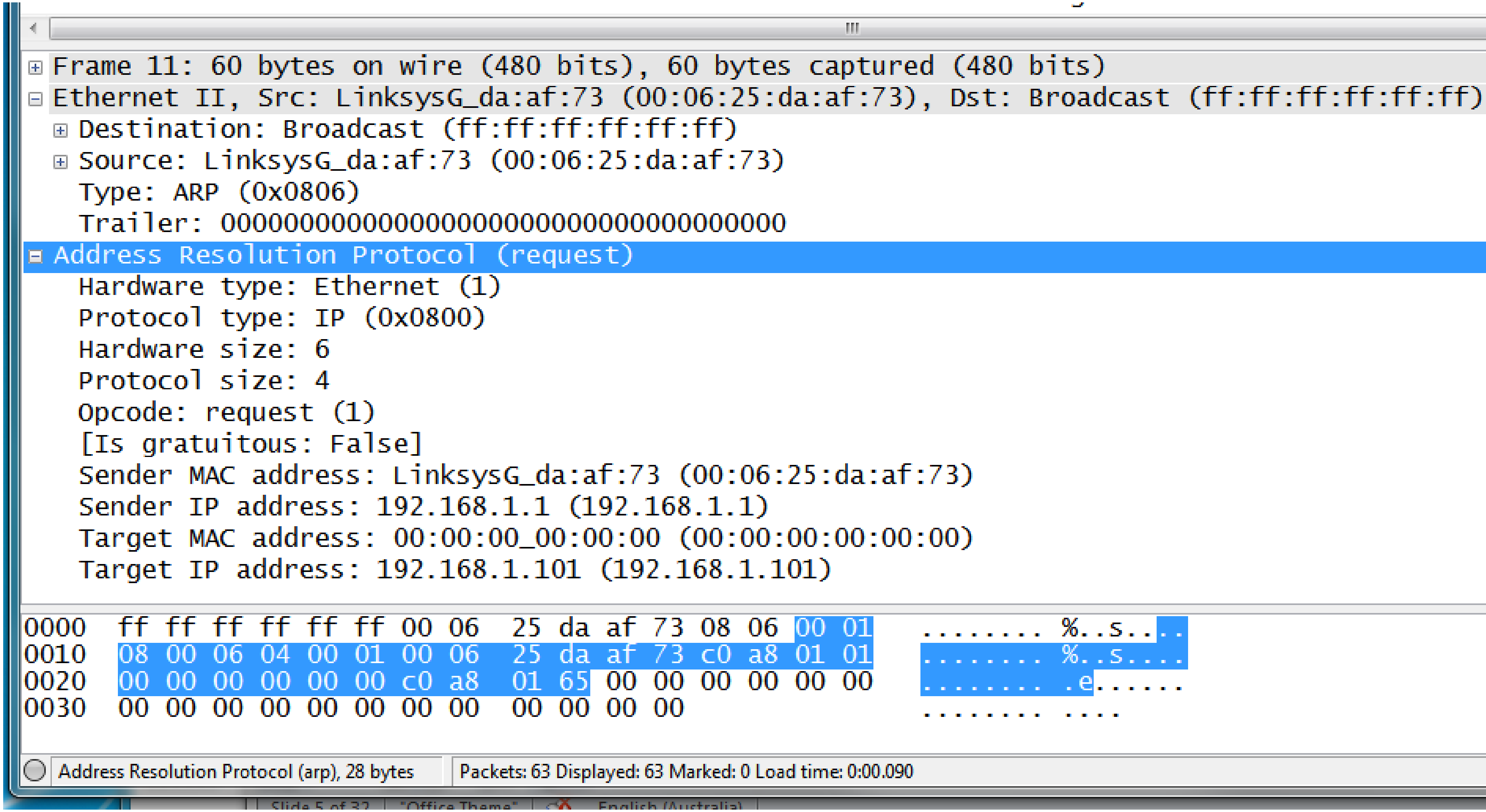
Host A broadcasts ARP query packet, containing B's IP address

- dest MAC address = FF-FF-FF-FF-FF-FF
- all nodes on LAN receive ARP query

- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A’s MAC address
- ARP is “plug-and-play”:
  - nodes create their ARP tables *without intervention* from the network administrator

ARP wireshark example

- Request (note the format of the frame)



ARP wireshark example

- Response (note the format of the frame)

```
⊞ Frame 12: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
⊟ Ethernet II, Src: DellComp_4f:36:23 (00:08:74:4f:36:23), Dst: LinksysG_da:af:73 (00:06:25:da:af:73)
  ⊞ Destination: LinksysG_da:af:73 (00:06:25:da:af:73)
  ⊞ Source: DellComp_4f:36:23 (00:08:74:4f:36:23)
    Type: ARP (0x0806)
⊟ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    [Is gratuitous: False]
    Sender MAC address: DellComp_4f:36:23 (00:08:74:4f:36:23)
    Sender IP address: 192.168.1.101 (192.168.1.101)
    Target MAC address: LinksysG_da:af:73 (00:06:25:da:af:73)
    Target IP address: 192.168.1.1 (192.168.1.1)
```

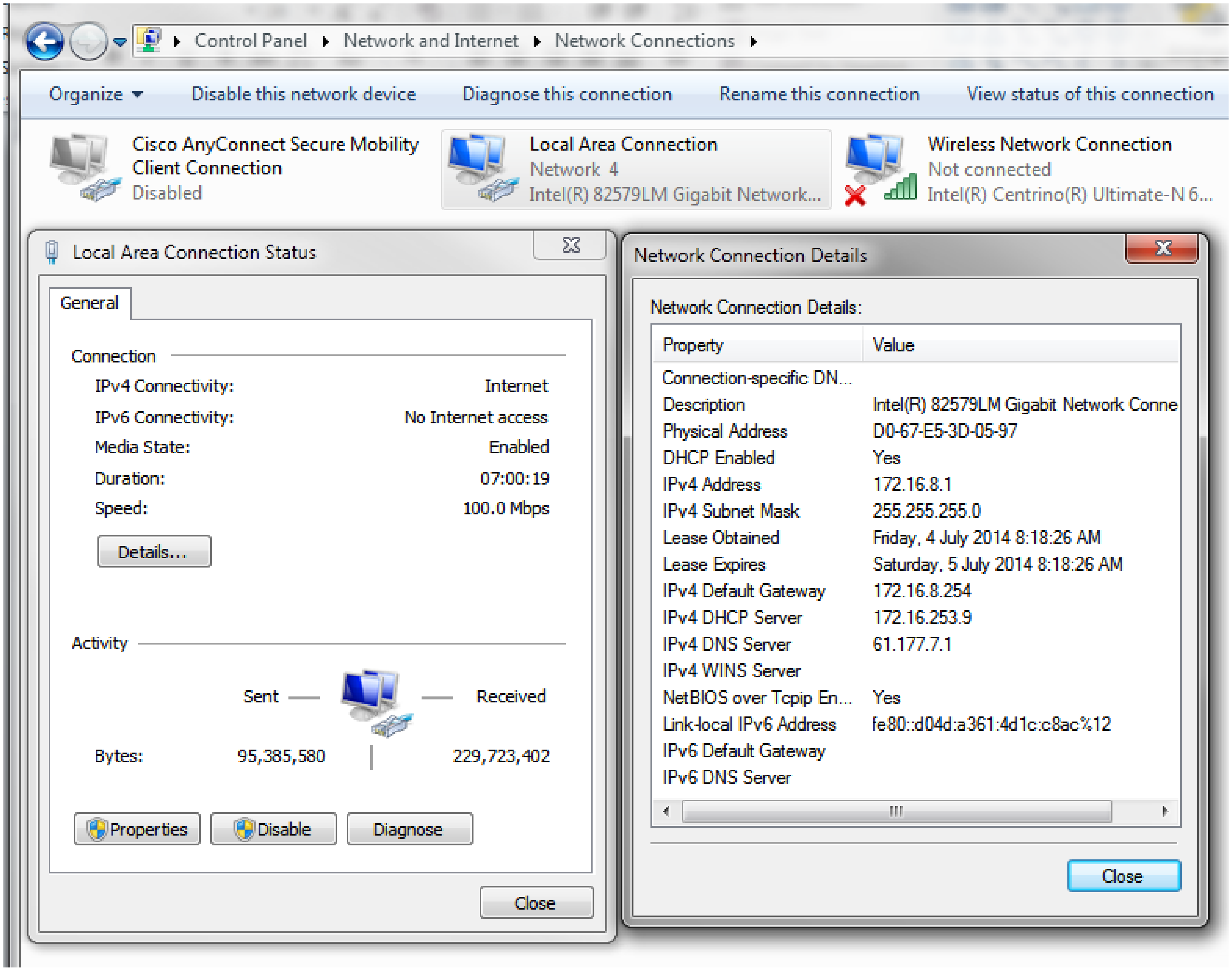
```
0000   00 06 25 da af 73 00 08  74 4f 36 23 08 06 00 01   ..%..s.. tO6#....
0010   08 00 06 04 00 02 00 08  74 4f 36 23 c0 a8 01 65   ........ tO6#...e
0020   00 06 25 da af 73 c0 a8  01 01                     ..%..s.. ..
```

Address Resolution Protocol (arp), 28 bytes | Packets: 63 Displayed: 63 Marked: 0 Load time: 0:00.090

Sending Messages using TCP/IP/Ethernet

- Required Network layer addressing information:
  - Computer's own IP address
  - Its subnet mask
    - to determine what addresses are part of its subnet
  - Local DNS server's IP address
    - to translate URLs into IP addresses
  - IP address of the router (gateway) on its subnet
    - to route messages going outside of its subnet
- Address information is obtained from a configuration file or provided by a DHCP server
  - Servers also need to know their own application layer addresses (domain names)

Network Configuration Information



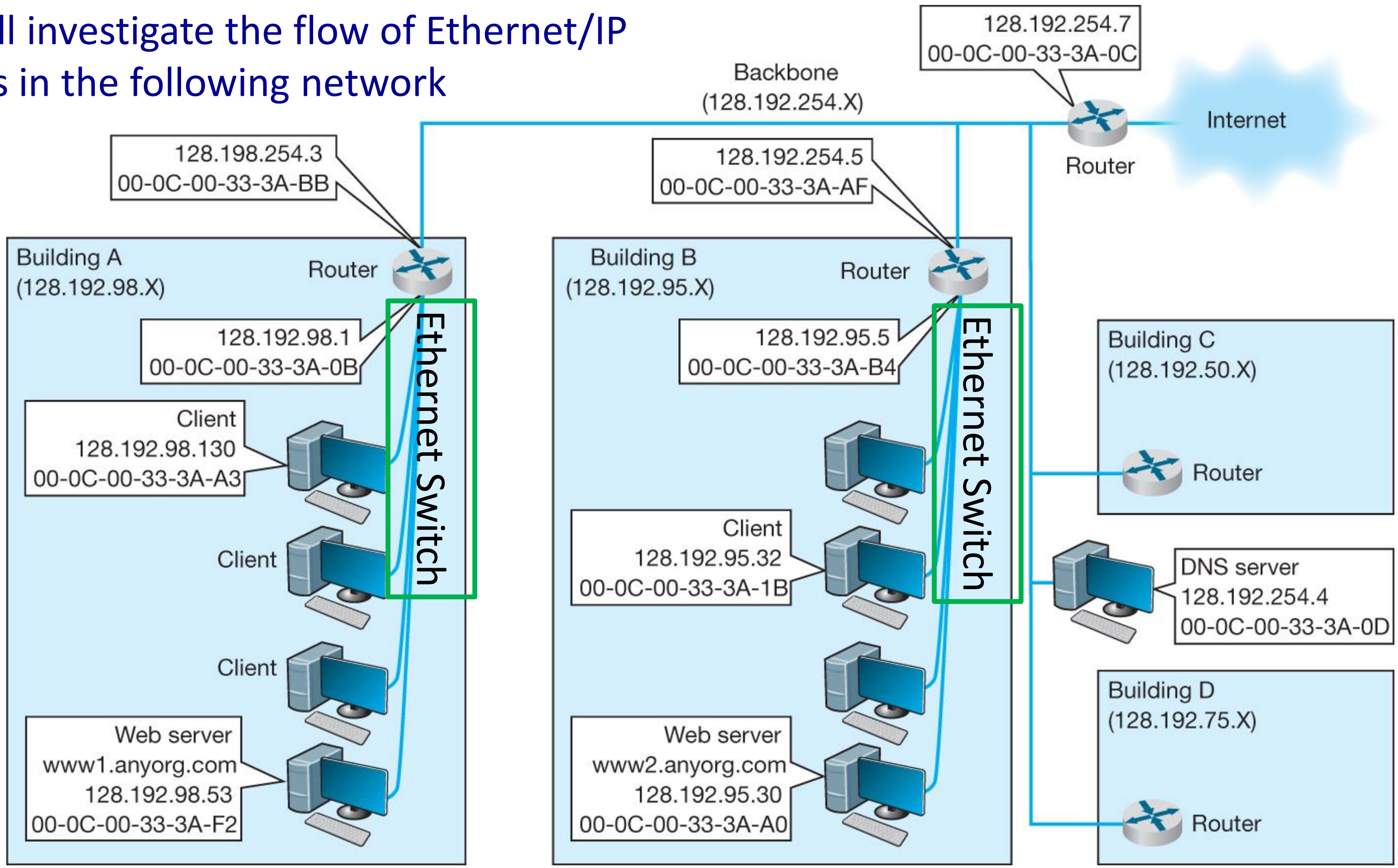
Monash PC Configuration Example

- Use **ipconfig /all** in the command window
- (filtered) response can be as follows:

Ethernet adapter Local Area Connection:
Physical Address. . . . . : D0-67-E5-3D-05-97
IPv6 Address. . . . . : 2001:388:608c:2c52:d04d:a361:4d1c:c8ac (Preferred)
IPv4 Address. . . . . : 130.194.69.92 (Preferred)
Subnet Mask . . . . : 255.255.255.0
Lease Obtained. . . . . : Monday, 2 September 2013 8:48:31 AM
Lease Expires . . . . : Monday, 2 September 2013 9:18:31 PM
Default Gateway . . . . : fe80::5:73ff:fea0:6%13
130.194.69.254
DHCP Server . . . . : 130.194.15.1
DNS Servers . . . . : 2001:388:608c:281::1
2001:388:608c:282::1
130.194.1.99
130.194.7.99

TCP/IP/Eth Example

We will investigate the flow of Ethernet/IP frames in the following network



Note: Buildings A, B, C, D subnets
Backbone subnet
DNS server

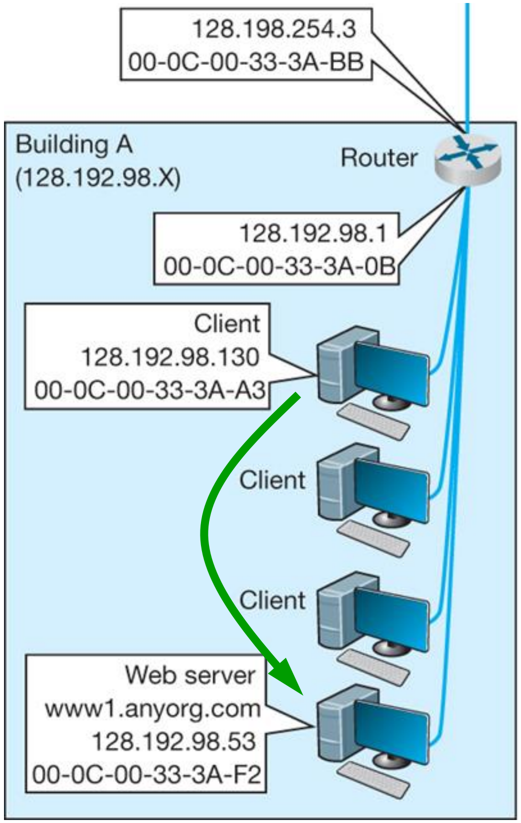
Each computer, including routers, has its MAC/PHY and IP addresses

TCP/IP/Eth Example

- In the examples that follow we ignore the TCP:
  - connection setup and termination
  - segmentation
  - details of the TCP header
- From the IP and Ethernet headers we will use only the addressing information:

Eth. Dest.	Eth. Src	IP Src	IP Dest.	TCP	HTTP

Case 1a: Known Address, Same Subnet



Flow of packet(s) when

- Client A (128.192.98.130) requests a Web page from a server (www1.anyorg.com)
- Client knows the server's IP and MAC addresses

Operations (performed by the client)

- Prepare HTTP packet and send it to TCP
- Place HTTP packet into a TCP packet and sent it to IP
- Place TCP packet into an IP packet, add destination IP address, 128.192.98.53
- Use its subnet mask to see that the destination is on the same subnet as itself
- Add server's MAC address into its destination address field, and send the frame to the Web server

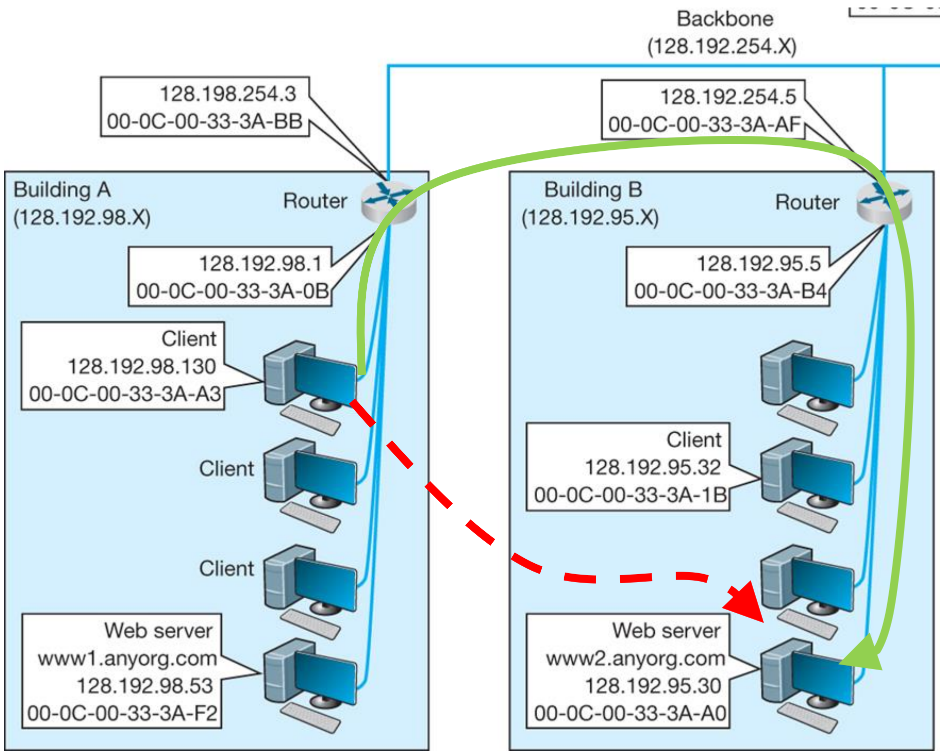
Using last two bytes of the addresses the (simplified, addresses only) Ethernet frame looks as follows:

Eth.Dst	Eth.Src	IP Src	IP Dst		
-3A-F2	**-3A-A3**	**.98.130**	**.98.53**	**TCP**	**HTTP**

Case 1b: Server response to client

- Operations (performed by the server)
  - Receive Ethernet frame, perform error checking and send back an ACK
  - Process incoming frame successively up the layers (data link, network, transport and application) until the HTTP request emerges
  - Process HTTP request and sends back an HTTP response (with requested Web page)
  - Process outgoing HTTP response successively down the layers until an Ethernet frame is created
  - Send Ethernet frame to the client
- Operations (performed by the client)
  - Receive Ethernet frame and process it successively up the layers until the HTTP response emerges at browser

Case 2: Known Address, Different Subnet



- Client A (128.192.98.130) requests a Web page from a server B (www2.anyorg.com)
- Client A knows the server's IP and uses the subnet mask to determine that the destination is NOT on the same subnet
- Prepares the Frame 1 and sends it to the subnet gateway/router A.
- Router A modifies the MAC addresses (but NOT the IP addresses) and sends the Frame 2 to the router B
- Router B knows that the server B is in its subnet, prepares the Frame 3 modifying the MAC addresses and sends the frame 3 to the server

Frame 1:

-3A-0B	-3A-A3	.98.130	.95.30	TCP	HTTP

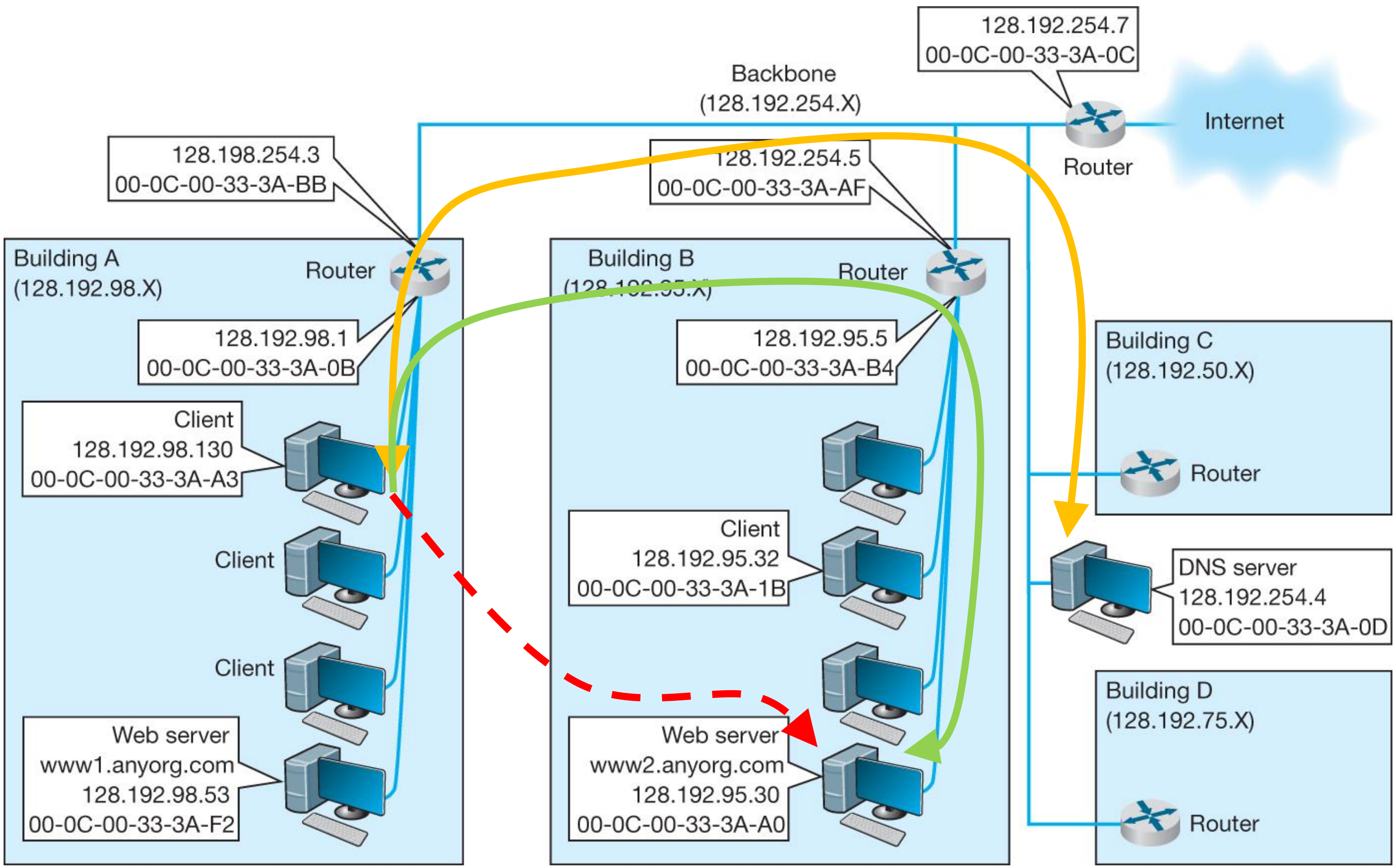
Frame 2:

-3A-AF	-3A-BB	.98.130	.95.30	TCP	HTTP

Frame 3:

-3A-A0	-3A-B4	.98.130	.95.30	TCP	HTTP

Case 3: Different Subnets, Unknown Addresses 1



Client A requests the web page from the www2.anyorg.com server but knows only addresses as in the configuration files

Needs to use ARP and DNS to get required IP addresses

Case 3: Different Subnets, Unknown Addresses 2

- Client A knows the IP address of its gateway/router but does not know its MAC address. Needs to **broadcast an ARP request** (who has 128.192.98.1 ?) inside its subnet:

brdcast	**-3A-A3**	**.98.130**	**.98.1**	**ARP**	

- The router A replies with the ARP response frame

-3A-A3	**-3A-0B**	**.98.1**	**.98.130**	**ARP**	

- Client A can now issue its DNS request (what is the IP address of www2.anyorg.com) to its DNS server through its gateway/router

-3A-0B	**-3A-A3**	**.98.130**	**.254.4**	**DNS**	**www2.anyorg .com**

- Assuming that the Gateway A knows MAC addresses in the backbone subnet, it passes the DNS request to the DNS server:

-3A-0D	**-3A-BB**	**.98.130**	**.254.4**	**DNS**	**www2.anyorg .com**

- The DNS server replies with the IP address to the Gateway A

-3A-BB	**-3A-0D**	**.254.4**	**.98.130**	**www2.anyorg .com**	**...95.30**

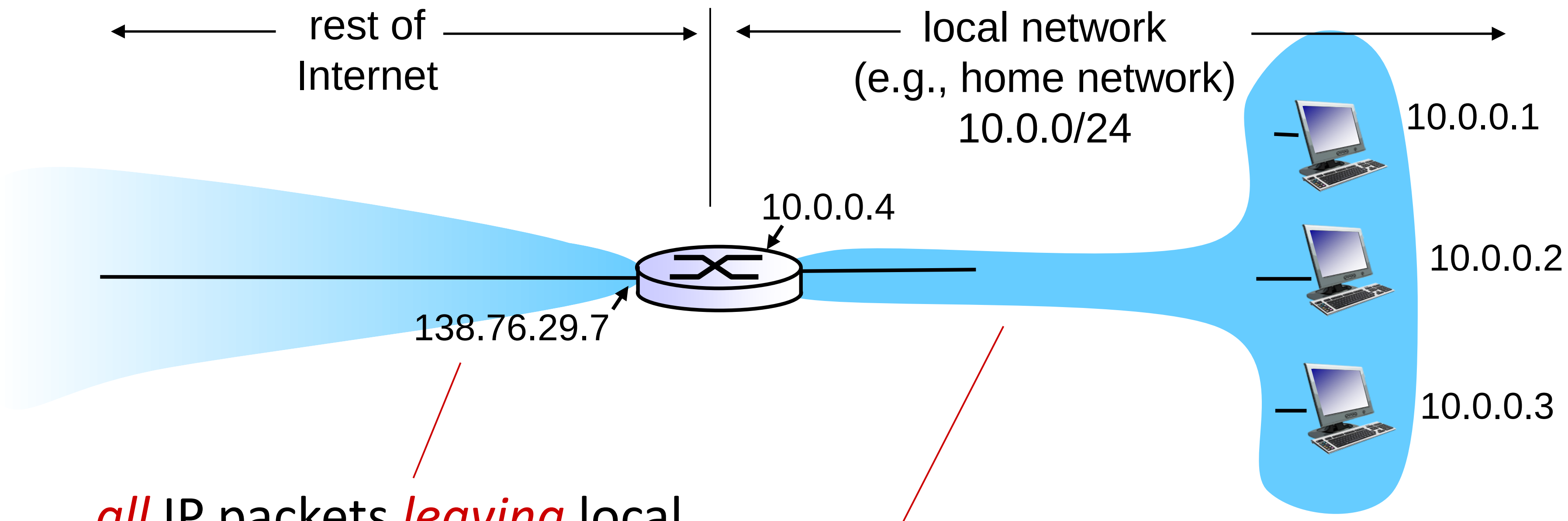
- The Gateway A passes the DNS response to the Client A:

-3A-A3	**-3A-0B**	**.254.4**	**.98.130**	**www2.anyorg .com**	**...95.30**

Case 3: Different Subnets, Unknown Addresses 3

- Once the Client A knows the IP address of the web server, it follows the steps as in
  - Case 2: Known Address, Different Subnet
  - It requests the web page through its Gateway/router A.

NAT: Network Address Translation



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

IP packets in this network use 10.0.0/24 IP addresses as source and destination

NAT: network address translation

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

– just one IP address for all devices is needed from the ISP (internet Service Provider)
– 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: use new port numbers to talk to a specific computer

The *NAT translation table* stores translation pairs in the NAT router:

source IP address, port number <=> NAT IP address, new port number

NAT router processes:

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

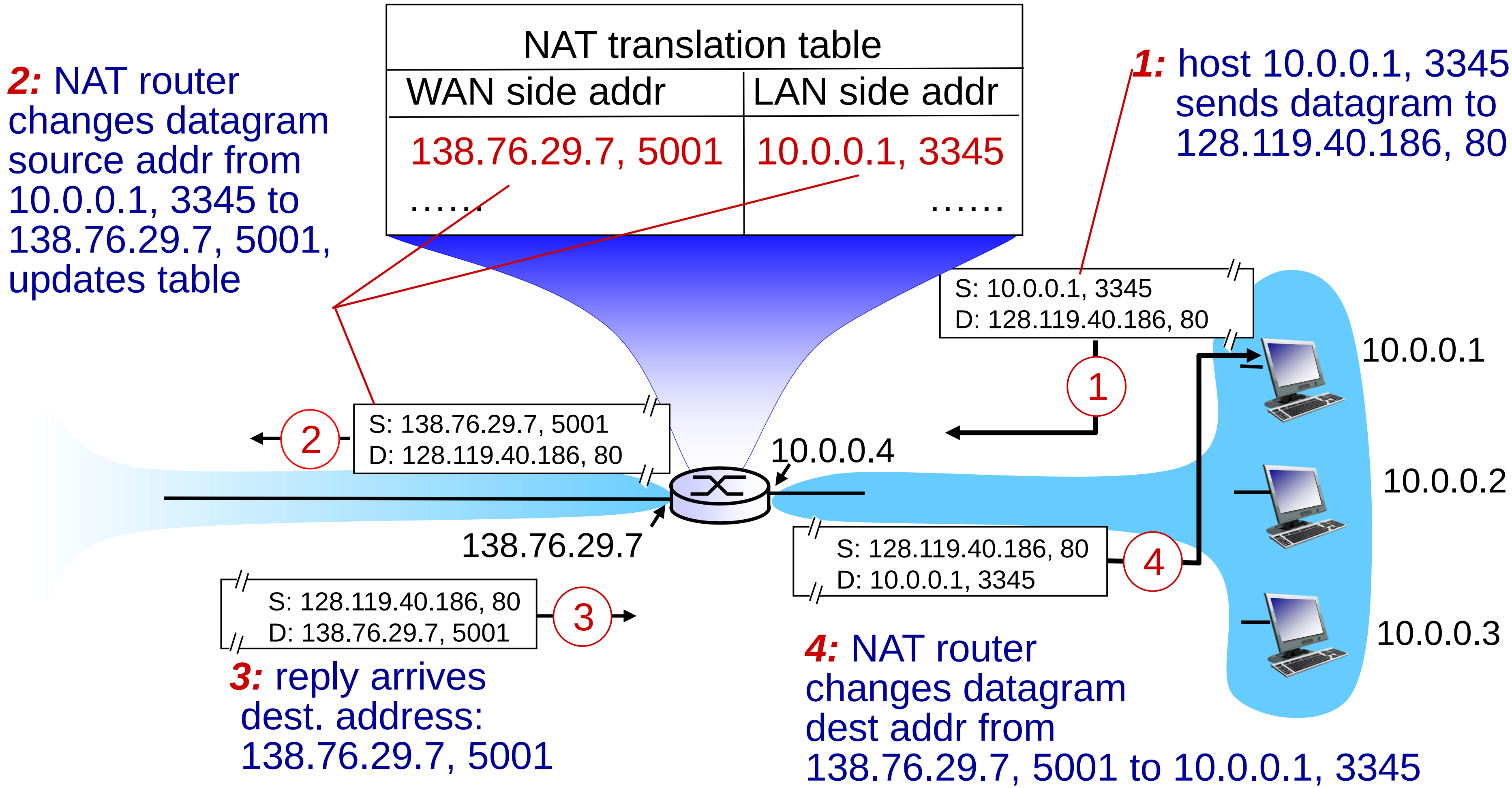
  . . . remote clients/servers will respond using

  (NAT IP address, new port #) as destination addr

- *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 Example

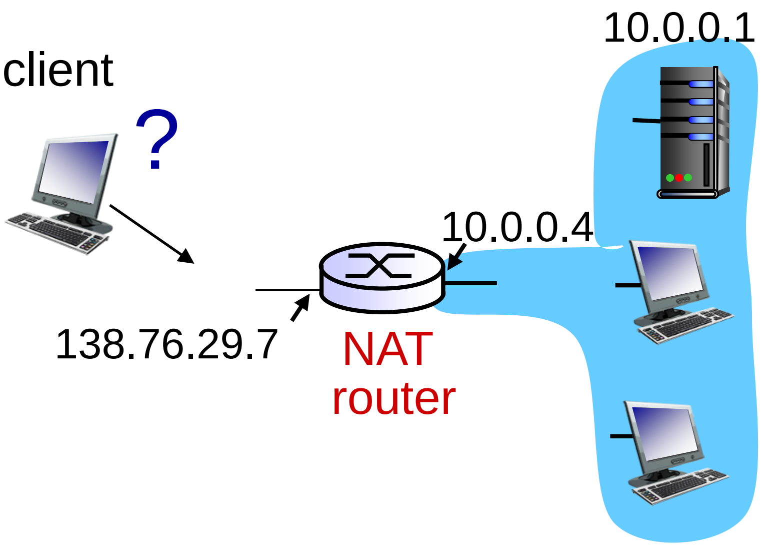

NAT translation table
WAN side addr
LAN side addr
138.76.29.7, 5001
10.0.0.1, 3345
……
……
1: host 10.0.0.1, 3345 sends datagram to 128.119.40.186, 80
2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table
S: 10.0.0.1, 3345
D: 128.119.40.186, 80
1
10.0.0.1
10.0.0.2
10.0.0.3
10.0.0.4
138.76.29.7
2
S: 138.76.29.7, 5001
D: 128.119.40.186, 80
S: 128.119.40.186, 80
D: 138.76.29.7, 5001
3
3: reply arrives dest. address: 138.76.29.7, 5001
S: 128.119.40.186, 80
D: 10.0.0.1, 3345
4
4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

NAT points to consider:

- 16-bit port-number field:
  - 64,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
  - routers should only process up to layer 3 (ports are for apps layer)
  - 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

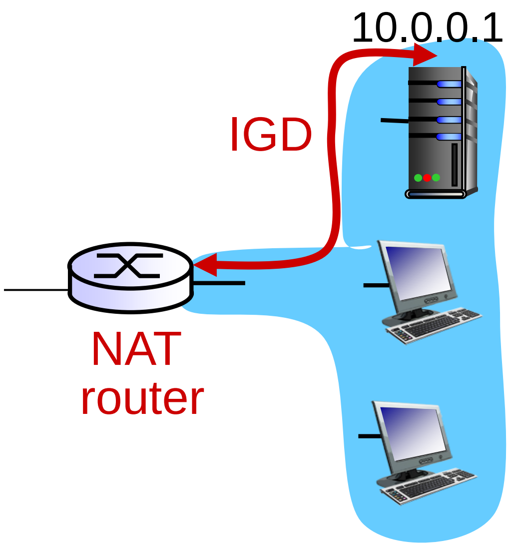
NAT traversal problem

- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7
- *solution1:* statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000



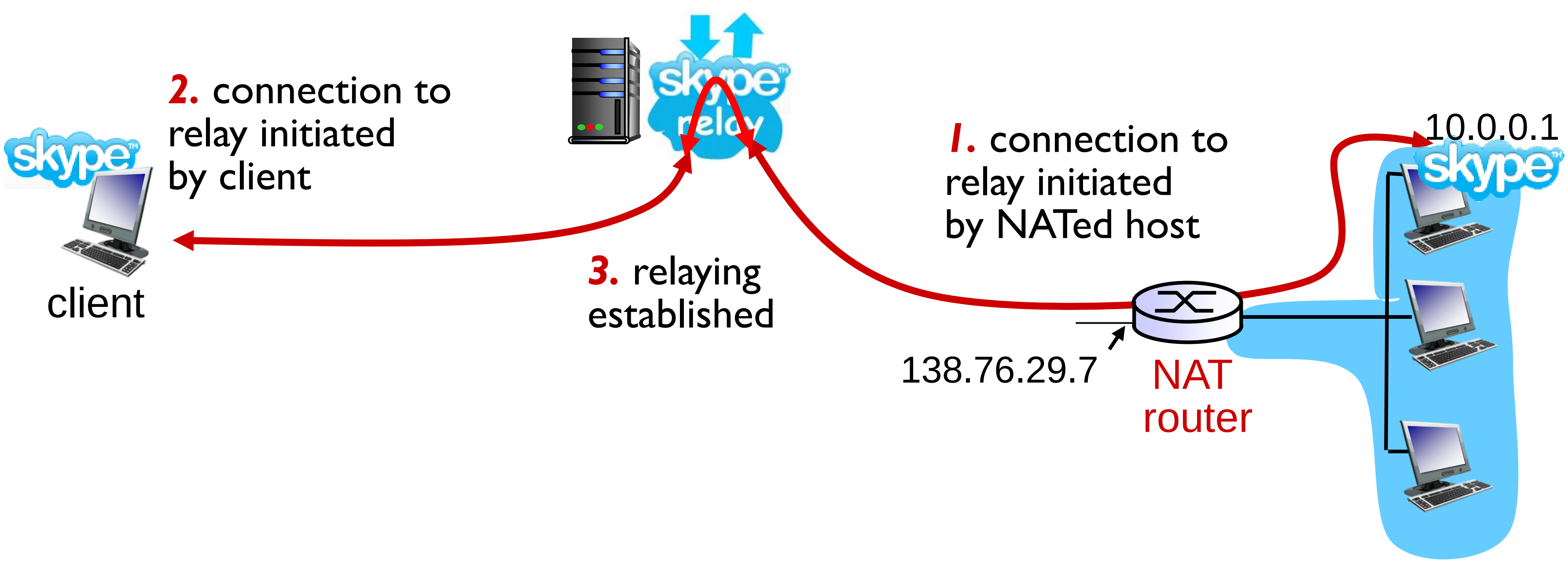
NAT traversal problem

- *solution 2:* Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol.
- Allows NATed host to:
  - learn public IP address (138.76.29.7)
  - add/remove port mappings
  - i.e., automate static NAT port map configuration



NAT traversal problem

- *solution 3:* relaying (used in Skype)
  - NATed client establishes connection to relay
  - external client connects to relay
  - relay bridges packets between to connections



ICMP: Internet Control Message Protocol

used by hosts & routers to communicate network-level information

- error reporting: unreachable host, network, port, protocol
- echo request/reply (used by ping)
- network-layer "above" IP:
  - ICMP **msgs** carried in IP datagrams
- ICMP message: type, code, header plus first 8 bytes of IP datagram causing error

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

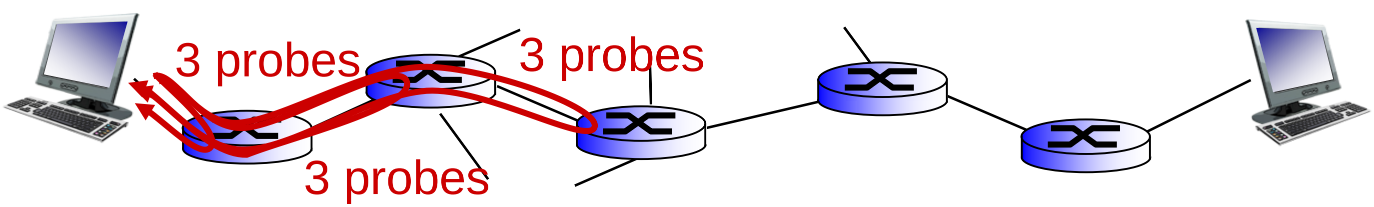
Traceroute and ICMP

- 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



IPv6: motivation

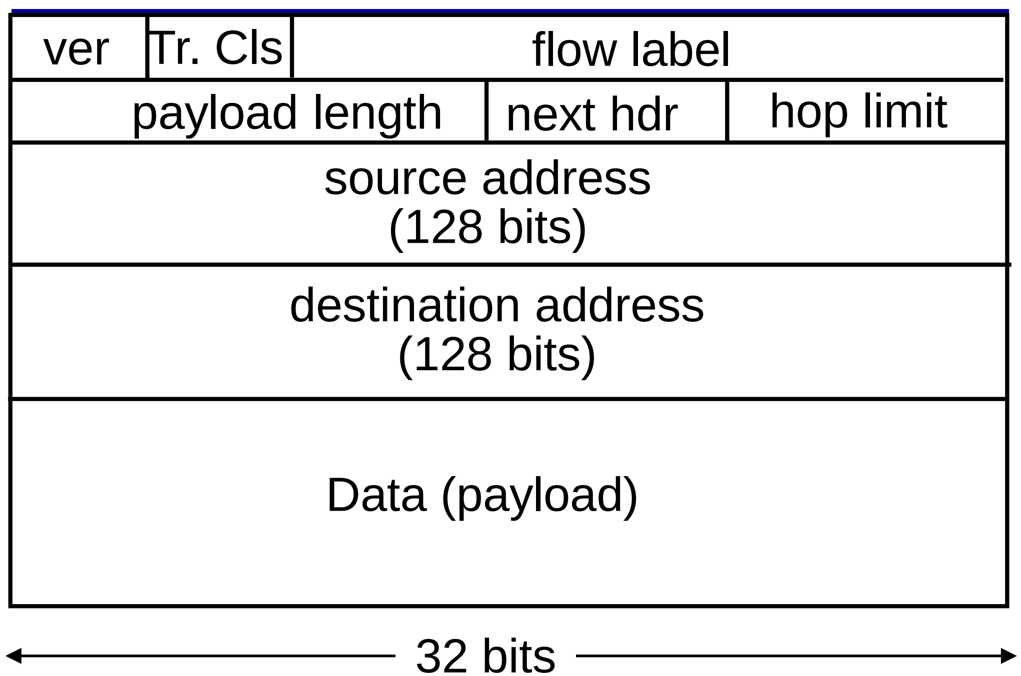
- *initial motivation:* 32-bit address space soon to be completely allocated.
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

IPv6 datagram format:

- fixed-length 40 byte header
- no fragmentation allowed

IPv6 datagram format

Traffic Class: 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



IP Packet Formats

IPv4 Header: 160 bits = 20 bytes plus 4 optional bytes

Version number	Header length	Type of service	Total length	Identifiers	Flags	Packet offset	Hop limit	Protocol	CRC 16	Source address	Destination address	Options	User data
4 bits	4 bits	8 bits	16 bits	16 bits	3 bits	13 bits	8 bits	8 bits	16 bits	32 bits	32 bits	32 bits	Varies

IPv6 Header: 320 bits (40 bytes)

Version number	Priority	flow label	Payload length	Next header	Hop limit	Source address	Destination address	User data
4 bits	4 bits	24 bits	16 bits	8 bits	8 bits	128 bits	128 bits	Varies

IPv6 Address Representation

- Unlike an IPv4, an IPv6 address is represented as
  - eight groups of four hexadecimal digits,
  - each group representing 16 bits (two octets).
- The groups are separated by colons (:).
- An example of an IPv6 address is:

2001:0db8:85a3:0000:0000:8a2e:0370:7334

- Leading zeroes in a group may be omitted.
- The example address may be written as:

2001:db8:85a3:0:0:8a2e:370:7334

- One or more consecutive groups of zero value may be replaced with a single empty group using two consecutive colons (::).
- The example address can be further simplified:

2001:db8:85a3::8a2e:370:7334

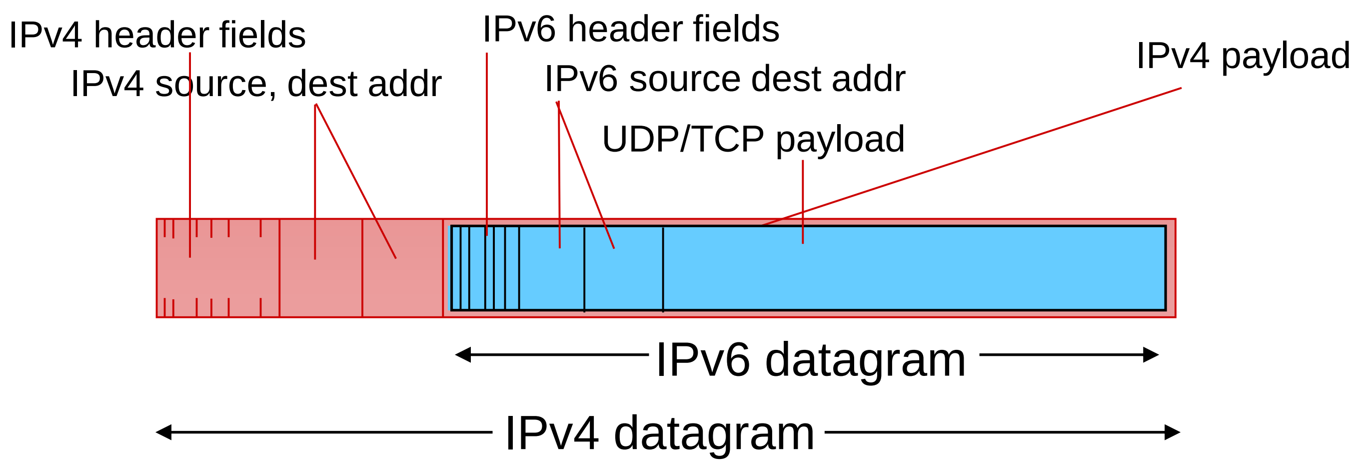
More IPv6 rules on Wikipedia

Other changes from IPv4

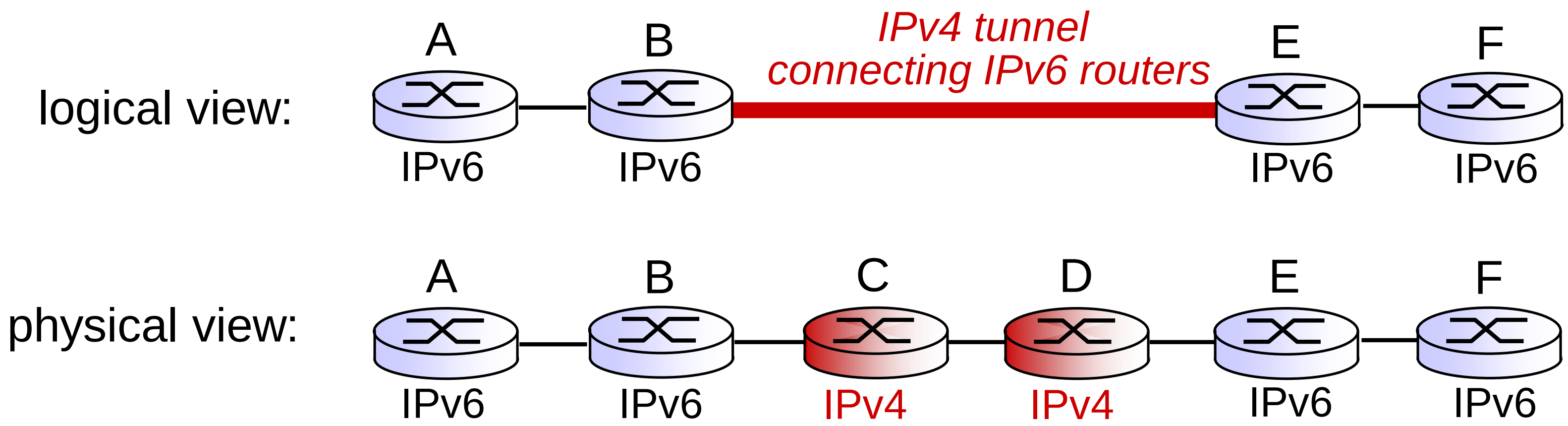
- *checksum*: removed entirely to reduce processing time at each hop
- *options:* allowed, but outside of header, indicated by "Next Header" field
- *ICMPv6:* new version of ICMP
  - additional message types, e.g. "Packet Too Big"
  - multicast group management functions

Transition from IPv4 to IPv6

- not all routers can be upgraded simultaneously
  - no specified transition day
  - how will network operate with mixed IPv4 and IPv6 routers?
- *tunneling:* IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers

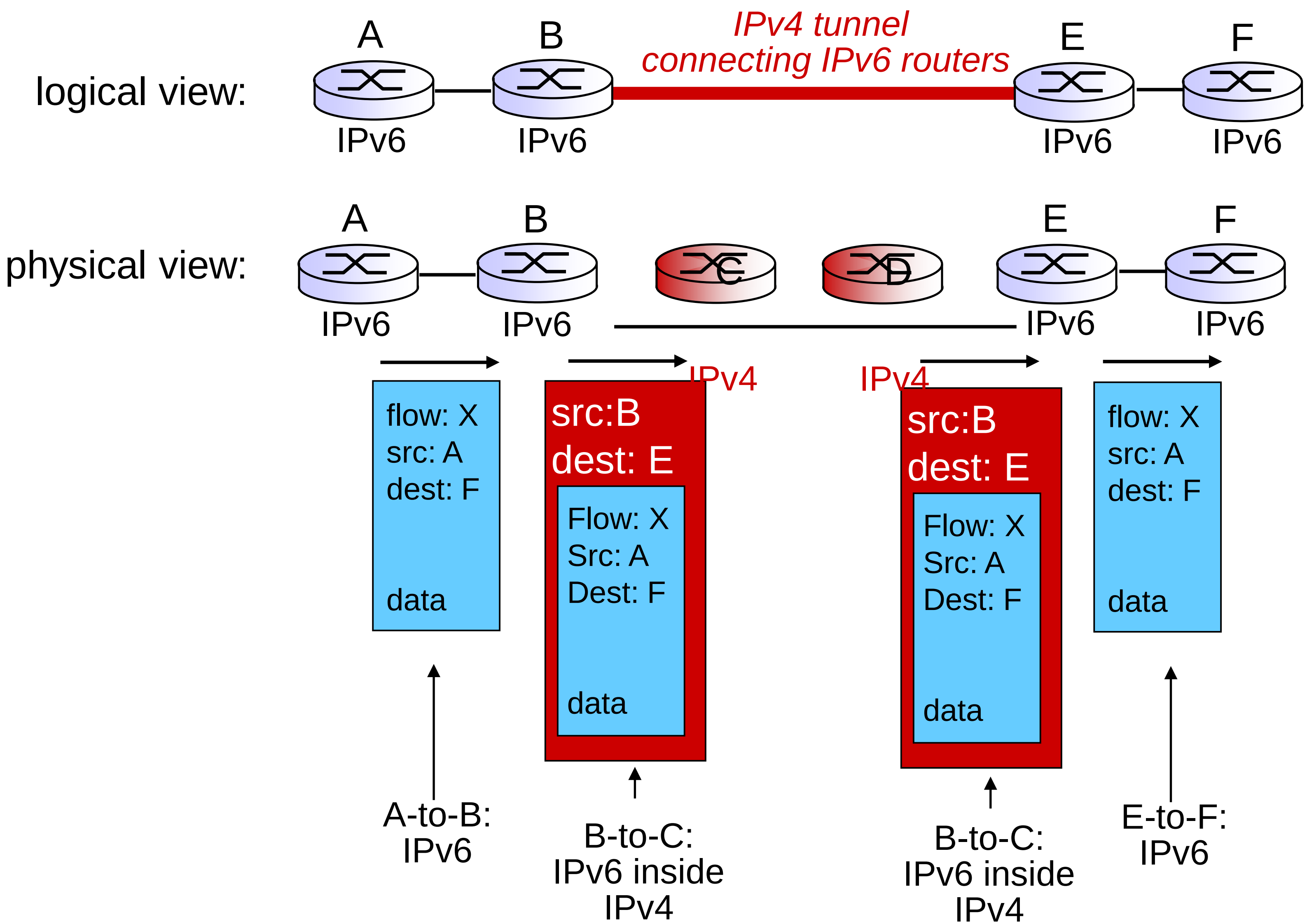


Tunneling



logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6

Tunneling



logical view:
A
B
E
F
IPv4 tunnel
connecting IPv6 routers
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
E
F
C
D
IPv6
IPv6
IPv6
IPv6
IPv4
IPv4
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6