

Chapter 4

Network Layer: Data Plane

A note on the use of these PowerPoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part.

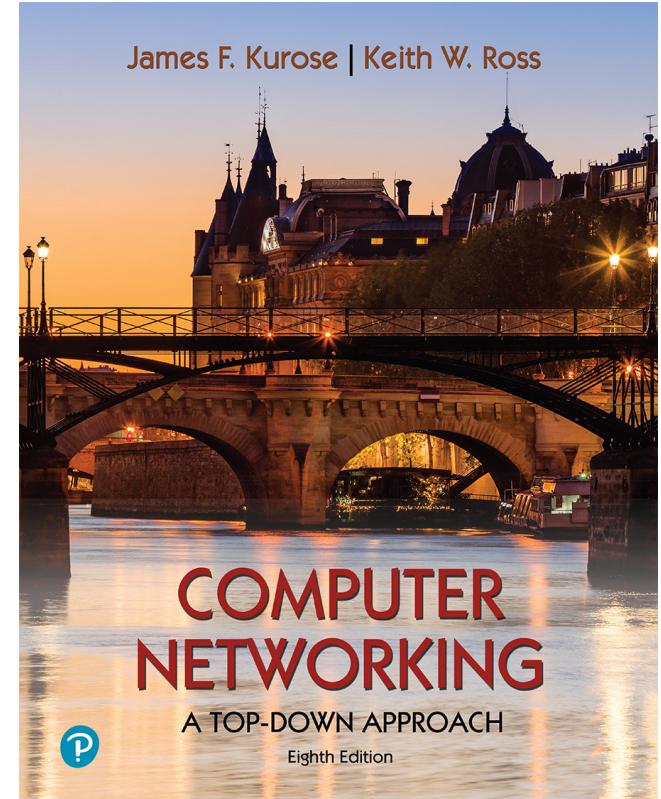
In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR

All material copyright 1996-2020
J.F Kurose and K.W. Ross, All Rights Reserved



*Computer Networking: A
Top-Down Approach*
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Objective

Assistance for a *system* that comes from an other era

Growing even larger ...

1. Limitation in address space
2. Getting rid of burdens
3. Transition to a new version



Network layer: “data plane” roadmap

- Network layer: overview
 - data plane
 - control plane
- Network layer service model
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- Forwarding of datagrams
- IP: the Internet Protocol
 - datagram format
 - addressing

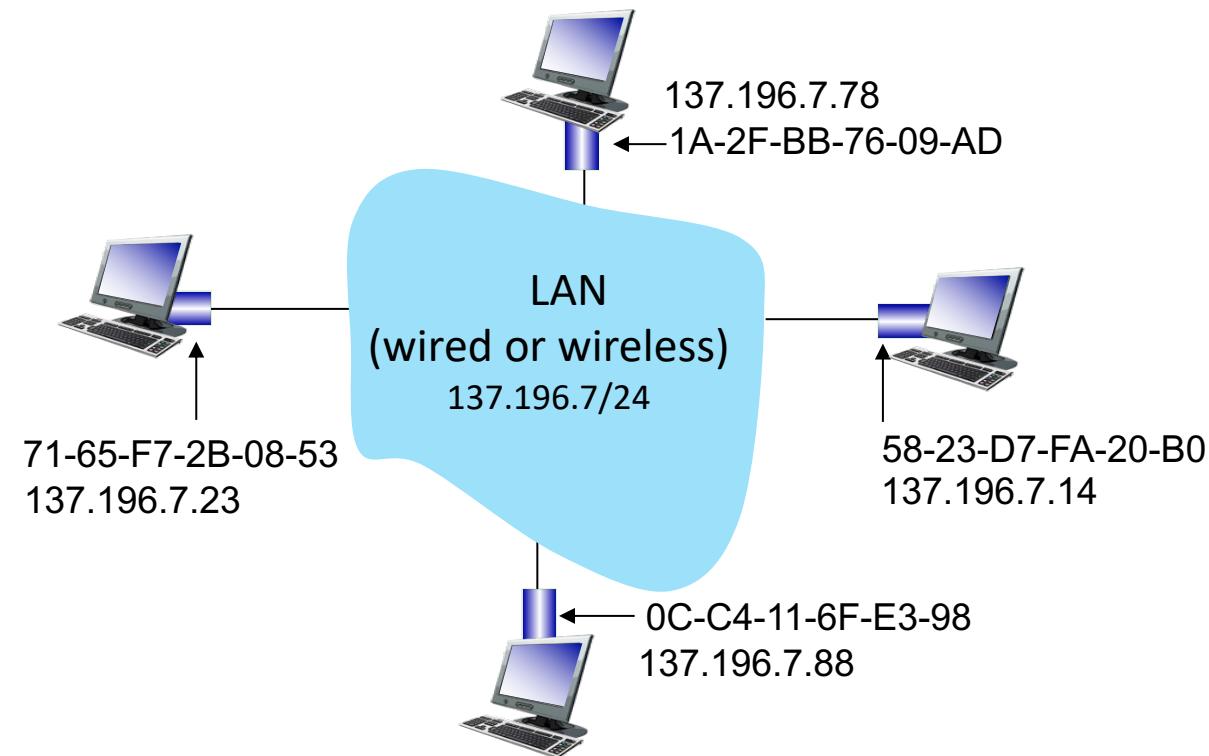


- IP/Ethernet cooperation
- Network Address Translation
- IPv6
- Tunneling

MAC addresses

each interface on LAN

- has unique 48-bit **MAC** address
- has a locally unique 32-bit IP address (as we've seen)

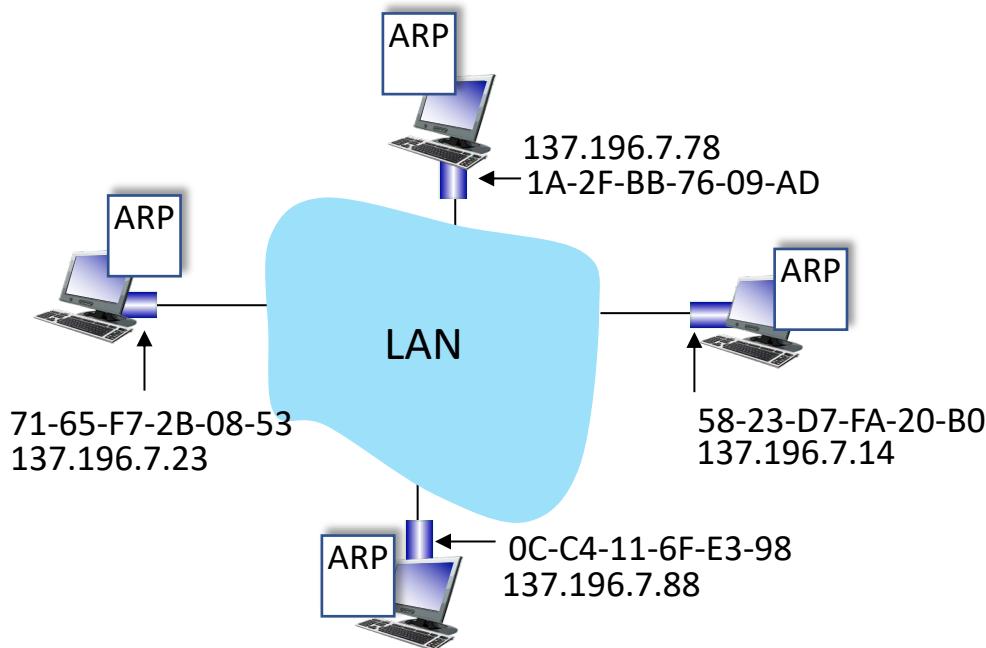


MAC addresses

- 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 interface from one LAN to another
 - recall IP address *not* portable: 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)

ARP protocol in action

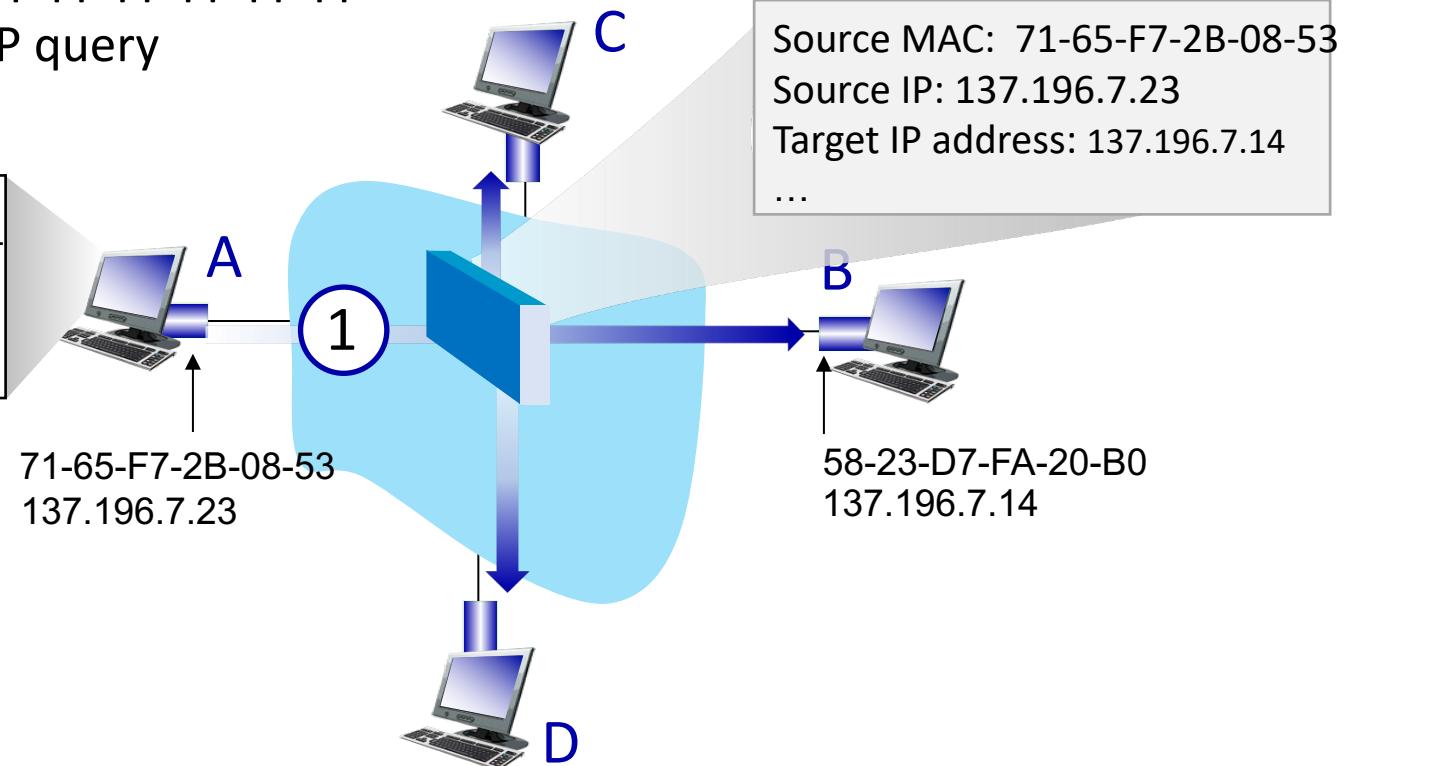
example: A wants to send datagram to B

- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address

- A broadcasts ARP query, containing B's IP address

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

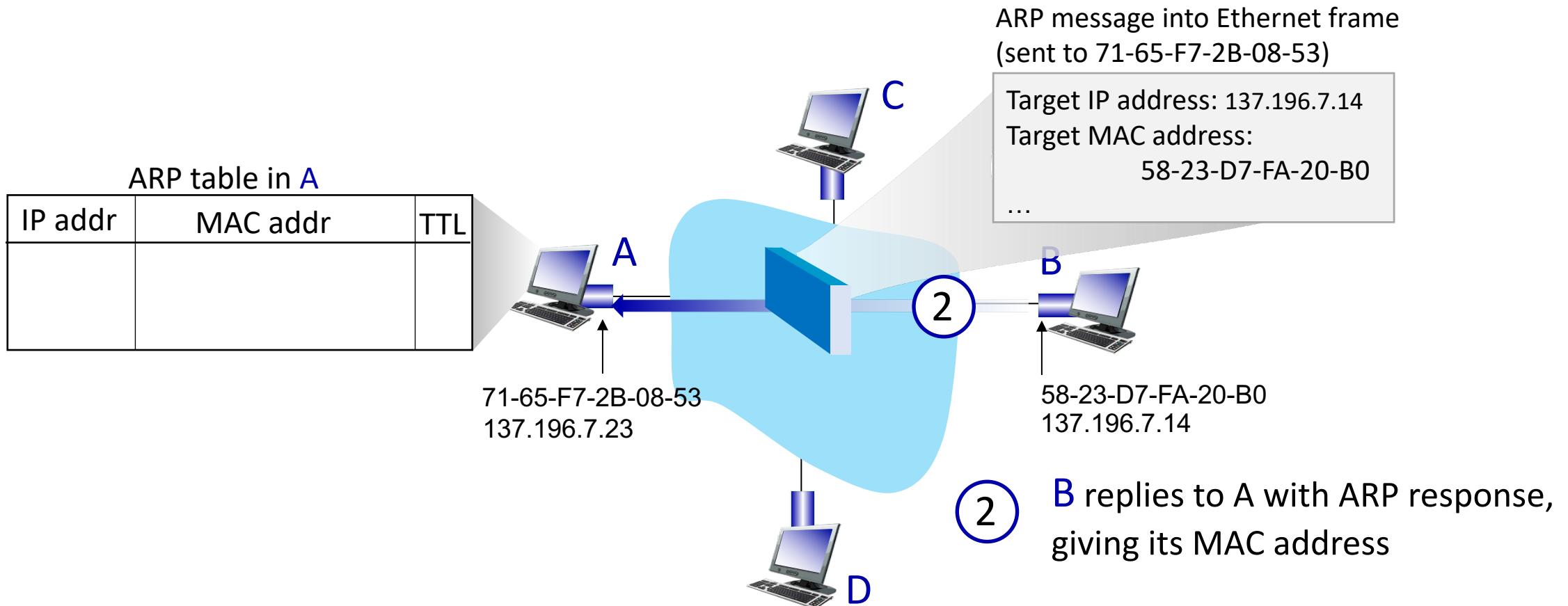
ARP table in A		
IP addr	MAC addr	TTL



ARP protocol in action

example: A wants to send datagram to B

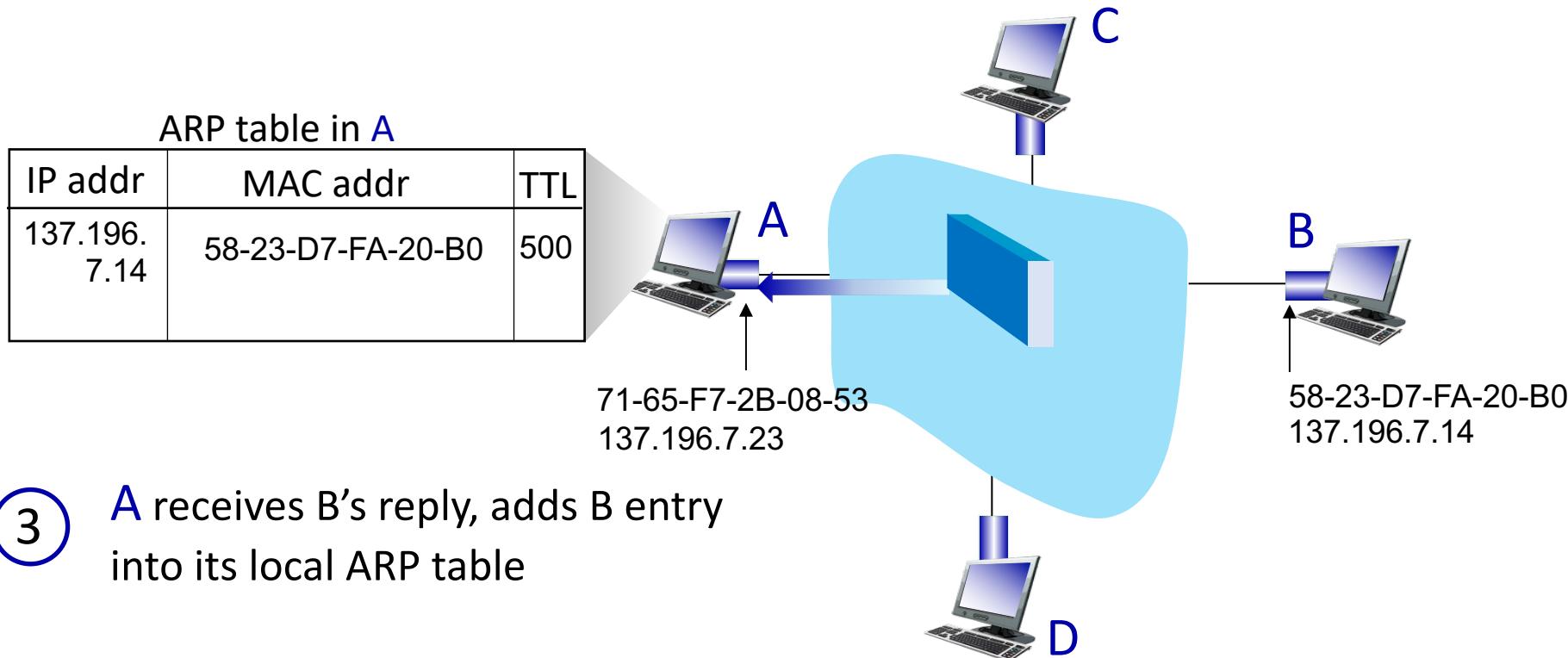
- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address



ARP protocol in action

example: A wants to send datagram to B

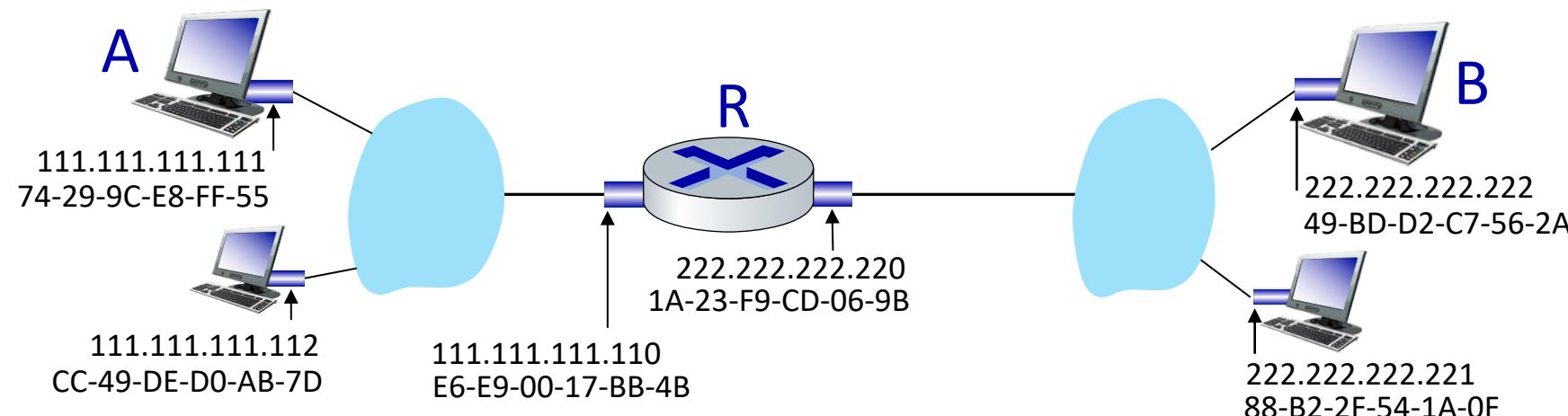
- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address



Routing to another subnet: addressing

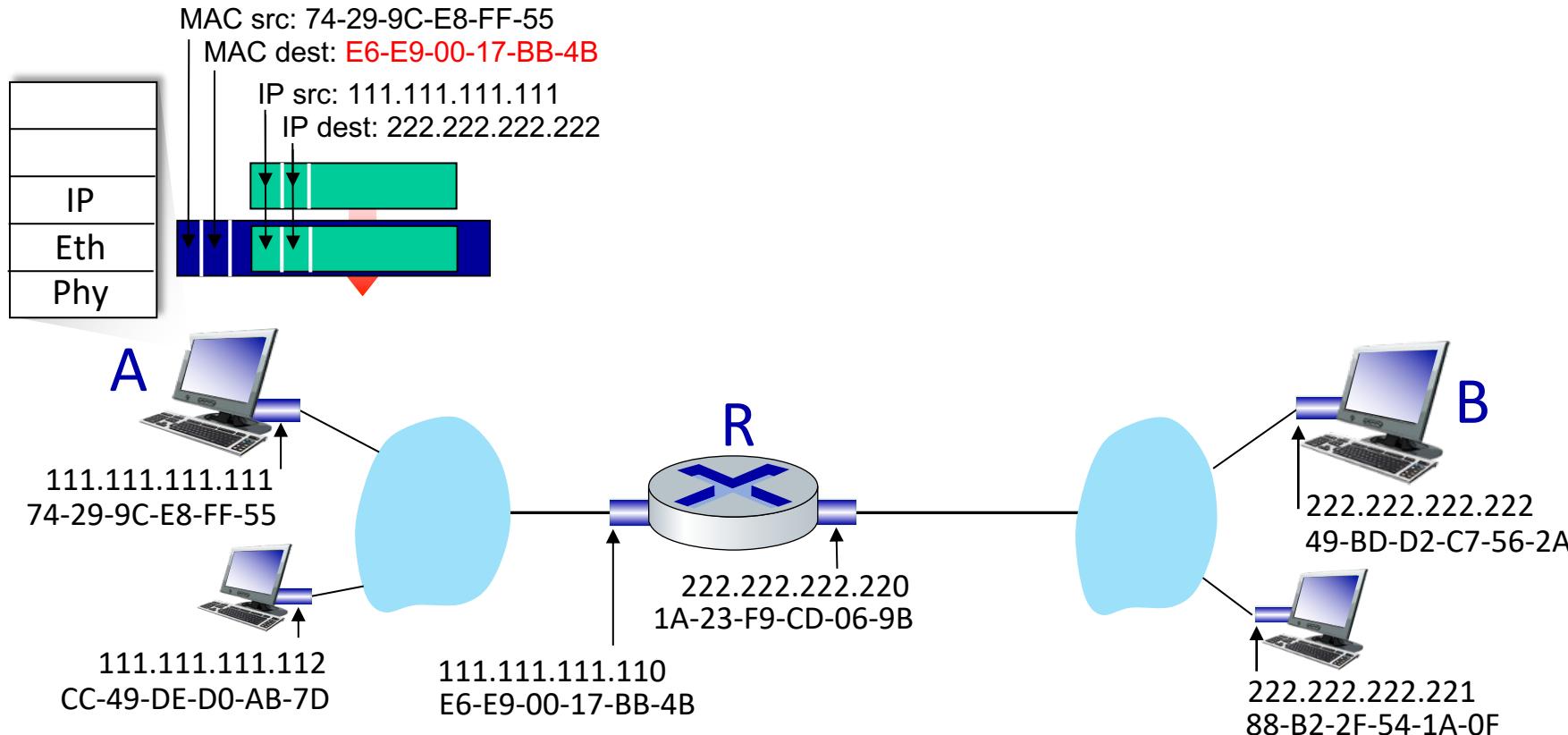
walkthrough: sending a datagram from *A* to *B* via *R*

- focus on addressing – at IP (datagram) and MAC layer (frame) levels
- assume that:
 - A knows B's IP address
 - A knows IP address of first hop router, R (how?)
 - A knows R's MAC address (how?)



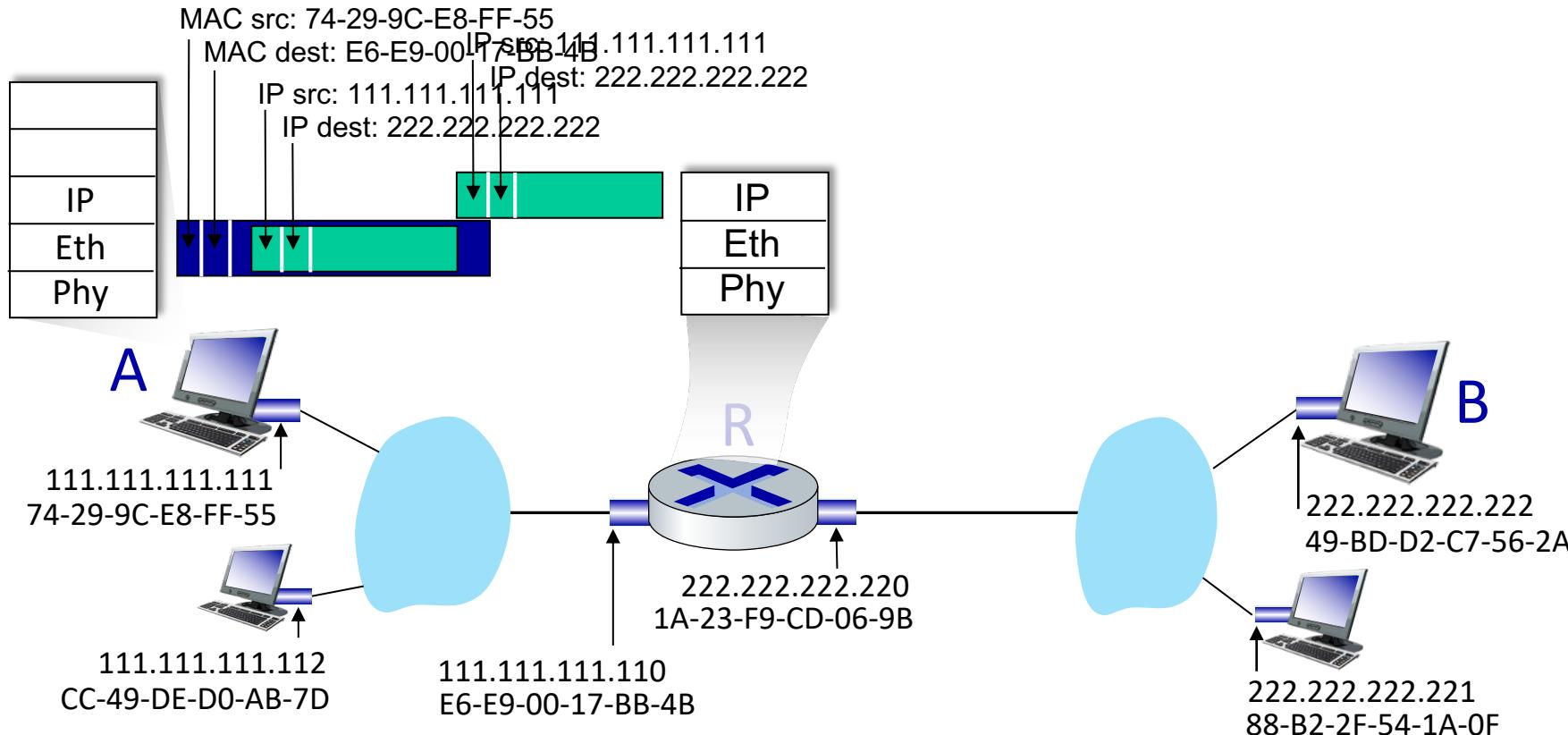
Routing to another subnet: addressing

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame containing A-to-B IP datagram
 - R's MAC address is frame's destination



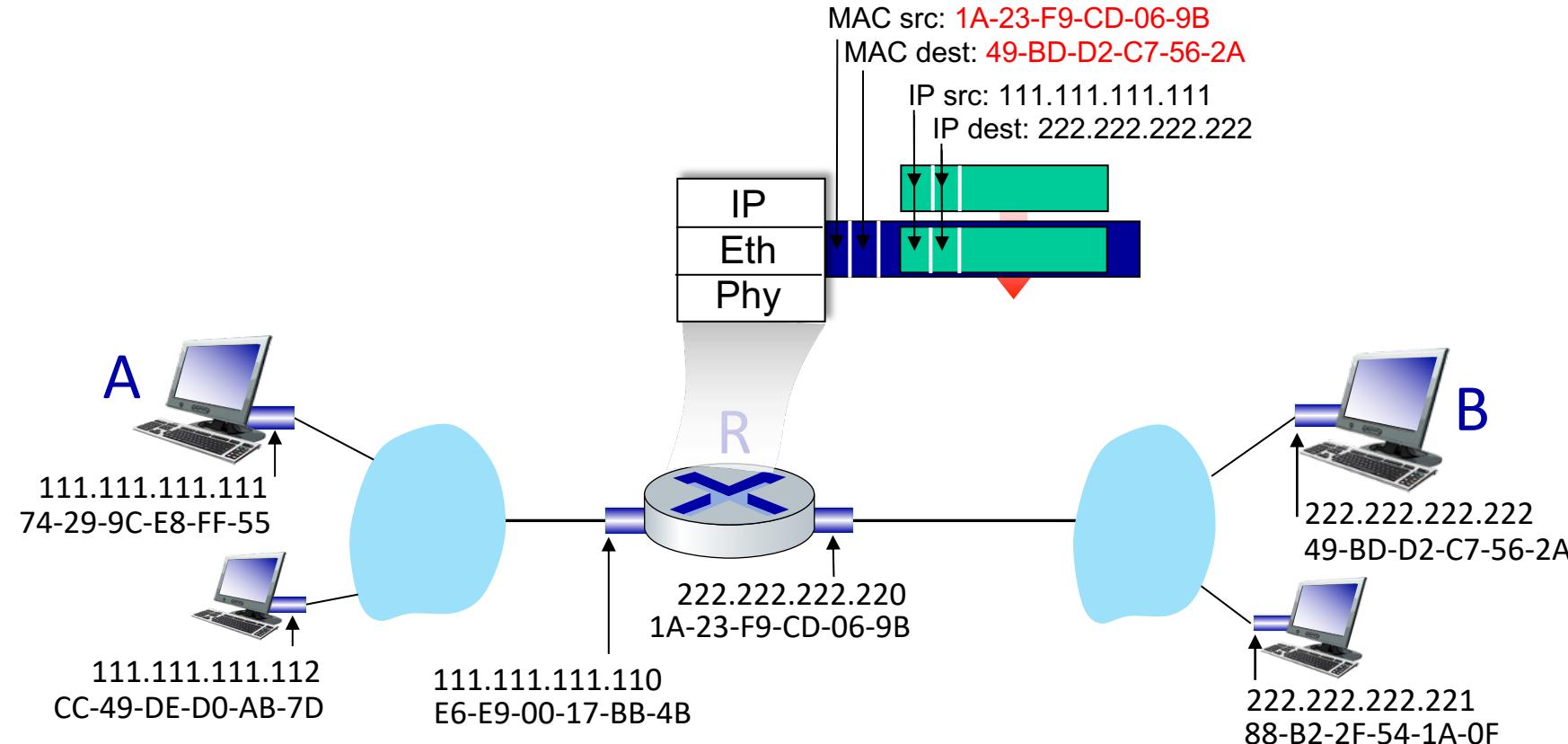
Routing to another subnet: addressing

- frame sent from A to R
- frame received at R, datagram removed, passed up to IP



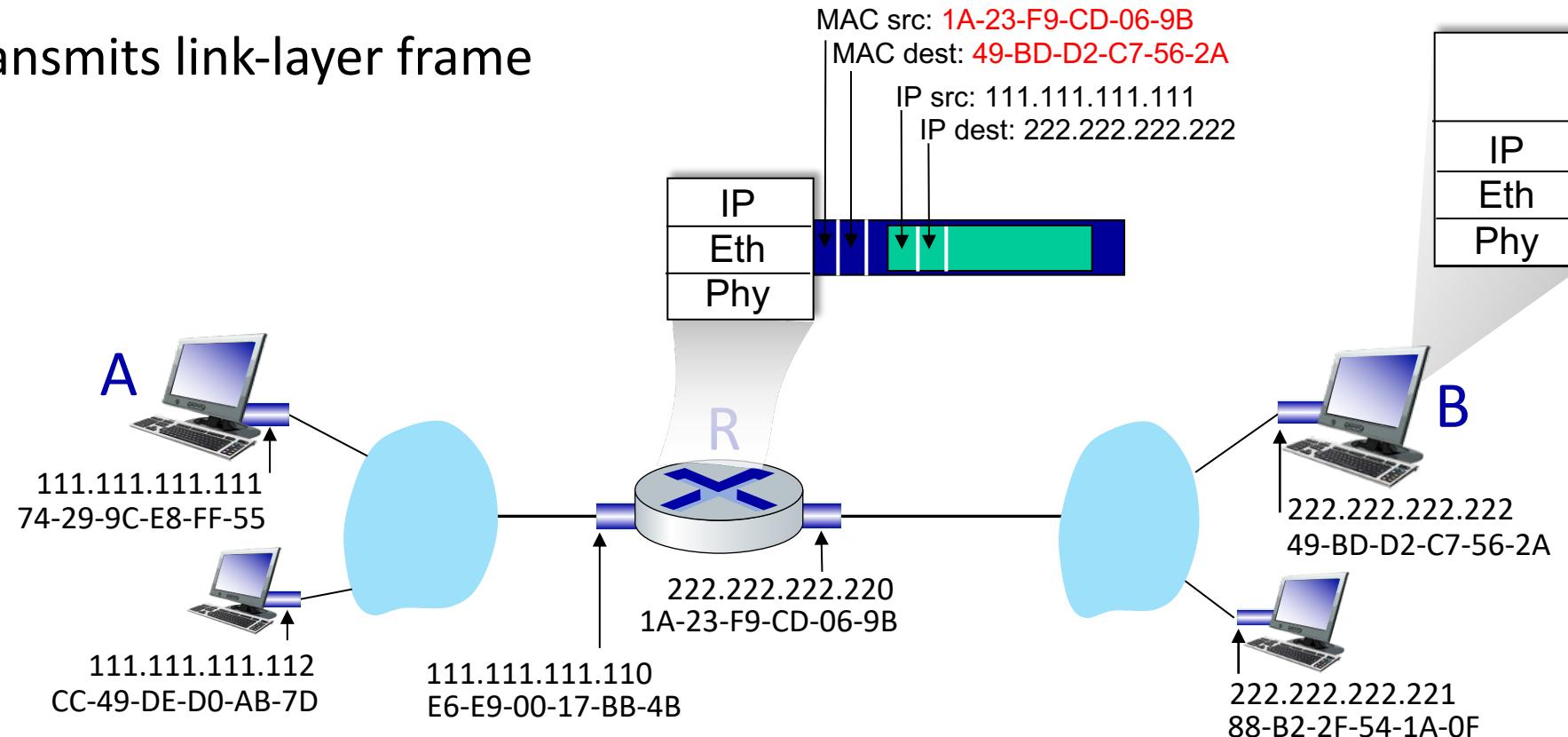
Routing to another subnet: addressing

- R determines outgoing interface, passes datagram with IP source A, destination B to link layer
- R creates link-layer frame containing A-to-B IP datagram. Frame destination address: B's MAC address



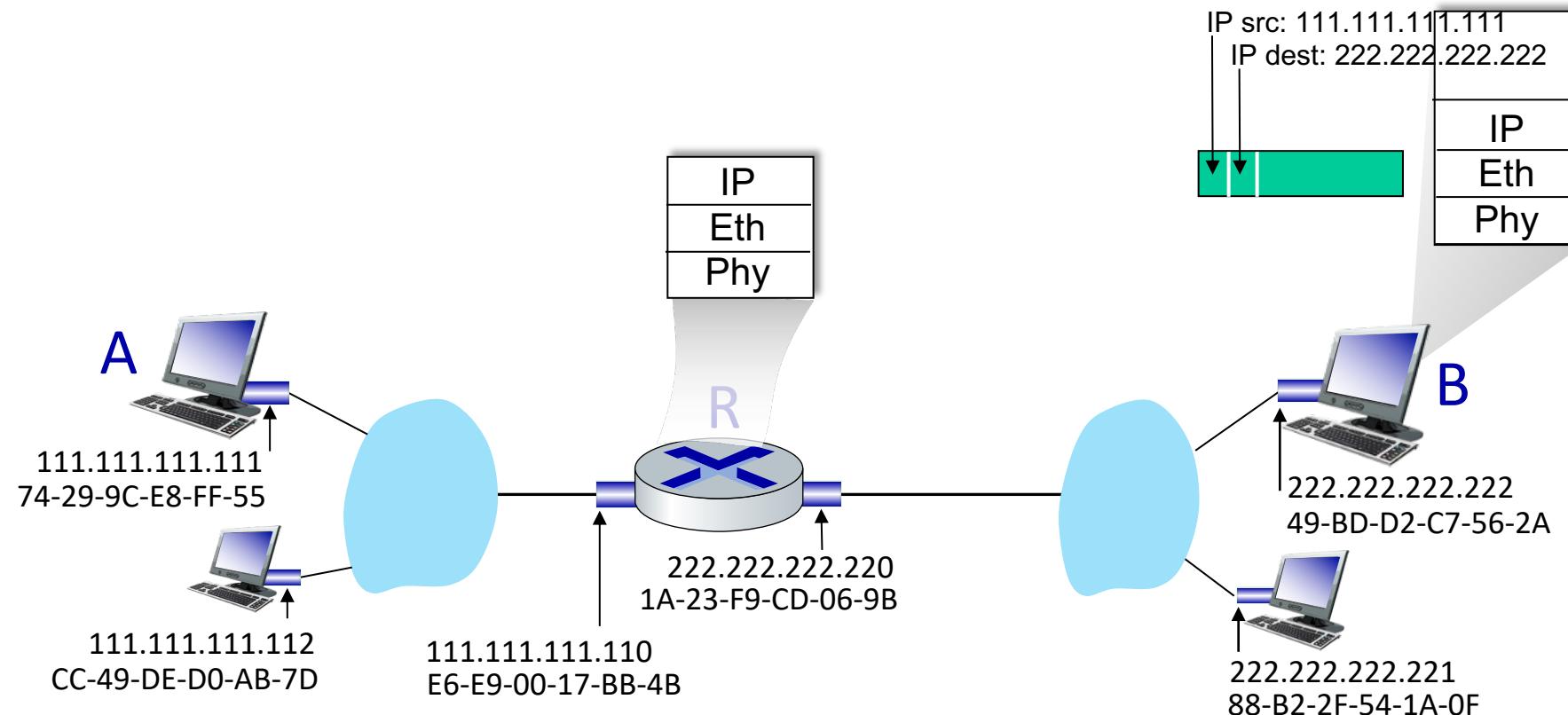
Routing to another subnet: addressing

- R determines outgoing interface, passes datagram with IP source A, destination B to link layer
- R creates link-layer frame containing A-to-B IP datagram. Frame destination address: B's MAC address
- transmits link-layer frame



Routing to another subnet: addressing

- B receives frame, extracts IP datagram destination B
- B passes datagram up protocol stack to IP



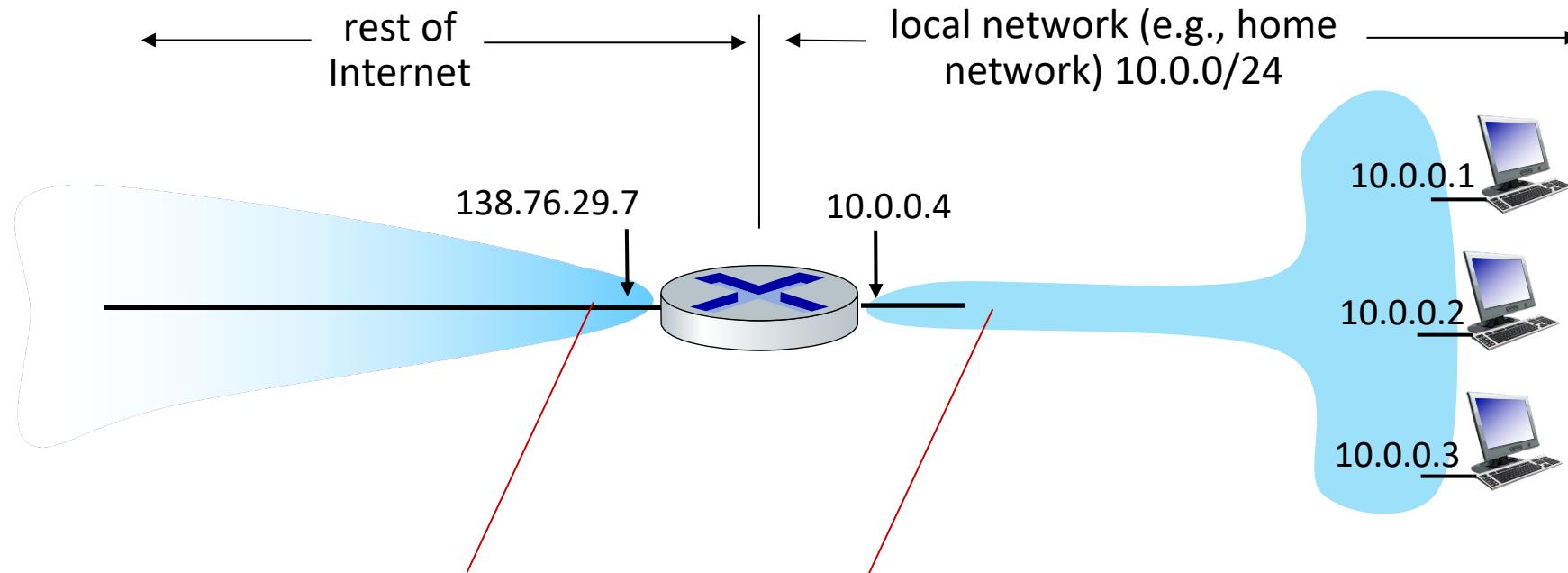
Network layer: “data plane” roadmap

- Network layer: overview
 - data plane
 - control plane
- Network layer service model
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- Forwarding of datagrams
- IP: the Internet Protocol
 - datagram format
 - addressing
- IP/Ethernet cooperation
- **Network Address Translation**
- IPv6
- Tunneling



NAT: network address translation

NAT: all devices in local network share just **one** IPv4 address as far as outside world is concerned



all datagrams *leaving* local network have
same source NAT IP address: 138.76.29.7,
but *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

- all devices in local network have 32-bit addresses in a “private” IP address space (10/8, 172.16/12, 192.168/16 prefixes) that can only be used in local network
- advantages:
 - just **one** IP address needed from provider ISP for ***all*** devices
 - can change addresses of host in local network without notifying outside world
 - can change ISP without changing addresses of devices in local network
 - security: devices inside local net not directly addressable, visible by outside world

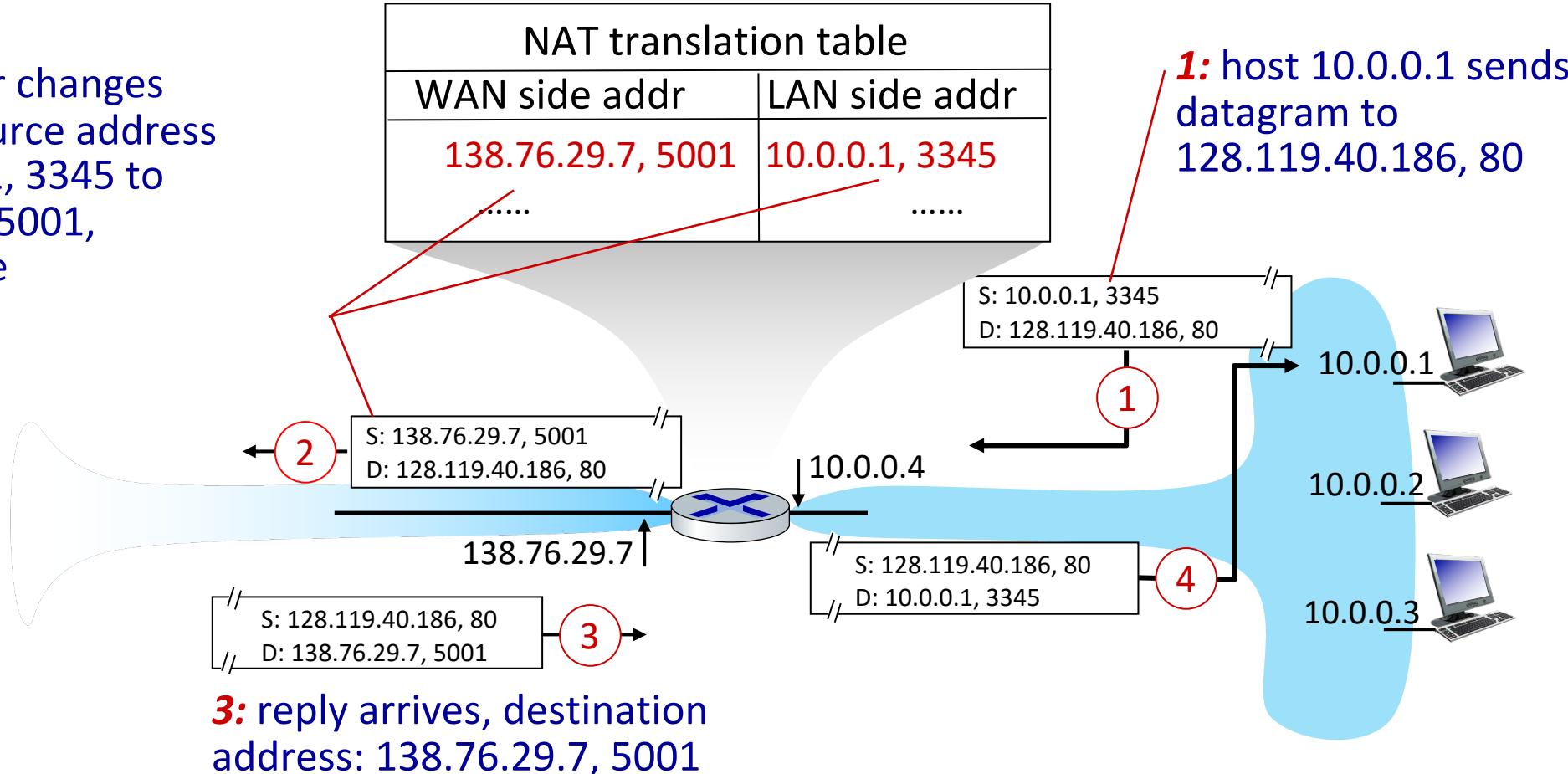
NAT: network address translation

implementation: NAT router must (transparently):

- 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 address
- 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 destination fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation

2: NAT router changes datagram source address from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table



NAT: network address translation

- NAT has been controversial:
 - routers “should” only process up to layer 3
 - address “shortage” should be solved by IPv6
 - violates end-to-end argument (port # manipulation by network-layer device)
 - NAT traversal: what if client wants to connect to server behind NAT?
- but NAT is here to stay:
 - extensively used in home and institutional nets, 4G/5G cellular nets

Network layer: “data plane” roadmap

- Network layer: overview
 - data plane
 - control plane
- Network layer service model
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- Forwarding of datagrams
- IP: the Internet Protocol
 - datagram format
 - addressing
- IP/Ethernet cooperation
- Network Address Translation
- **IPv6**
- Tunneling



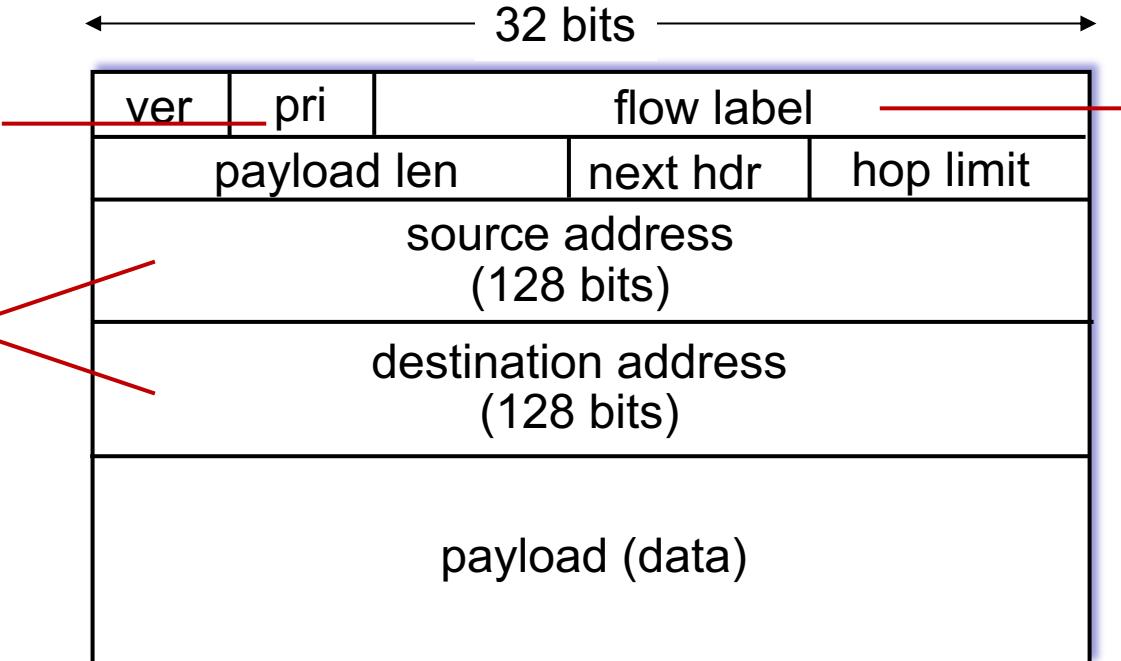
IPv6: motivation

- **initial motivation:** 32-bit IPv4 address space would be completely allocated
- additional motivation:
 - speed processing/forwarding: 40-byte fixed length header
 - enable different network-layer treatment of “flows”

IPv6 datagram format

priority: identify priority among datagrams in flow

128-bit
IPv6 addresses



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

What's missing (compared with IPv4):

- no checksum (to speed processing at routers)
- no fragmentation/reassembly
- no options (available as upper-layer, next-header protocol at router)

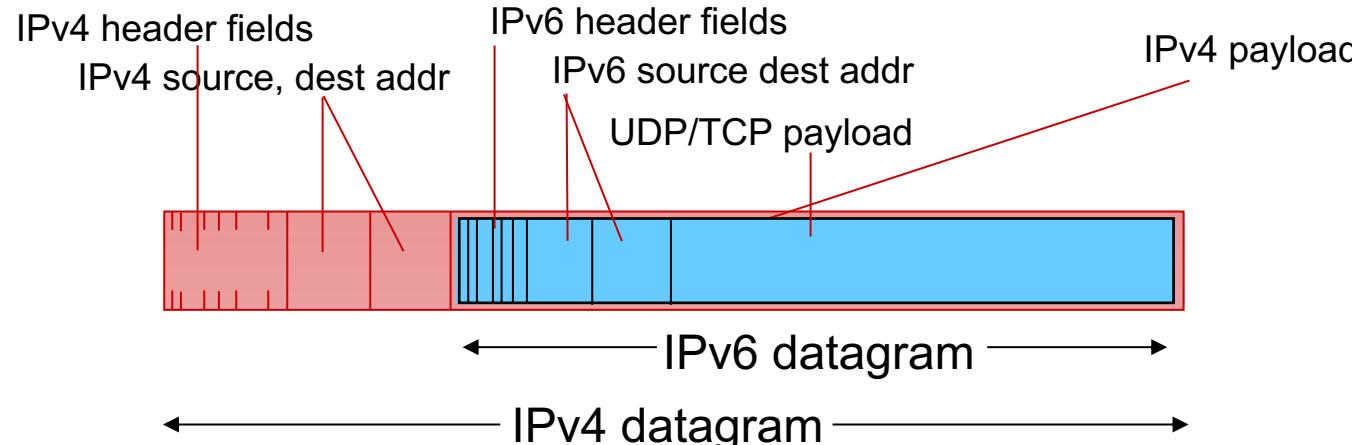
Network layer: “data plane” roadmap

- Network layer: overview
 - data plane
 - control plane
- Network layer service model
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- Forwarding of datagrams
- IP: the Internet Protocol
 - datagram format
 - addressing
- IP/Ethernet cooperation
- Network Address Translation
- IPv6
- **Tunneling**



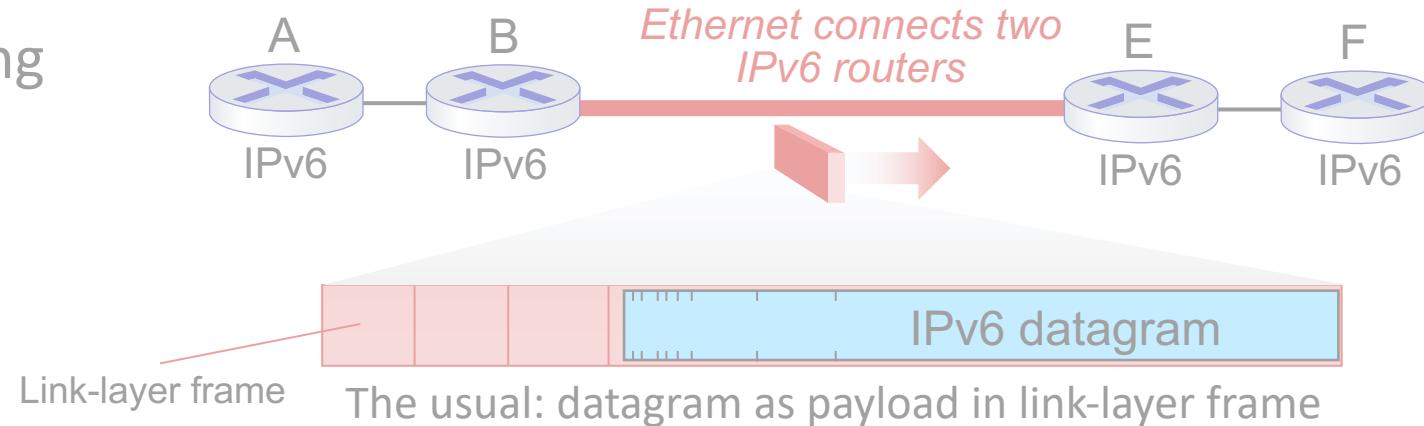
Transition from IPv4 to IPv6

- not all routers can be upgraded simultaneously
 - no “flag days”
 - how will network operate with mixed IPv4 and IPv6 routers?
- **tunneling:** IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers (“packet within a packet”)
 - tunneling used extensively in other contexts (4G/5G)

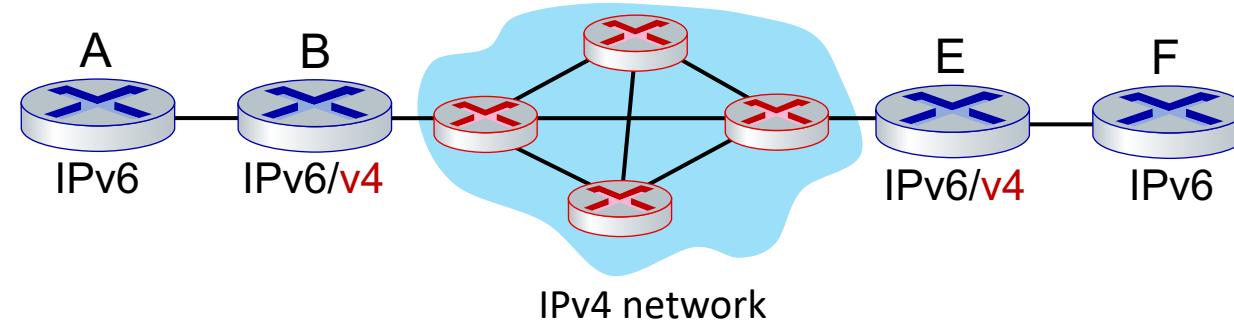


Tunneling and encapsulation

Ethernet connecting
two IPv6 routers:

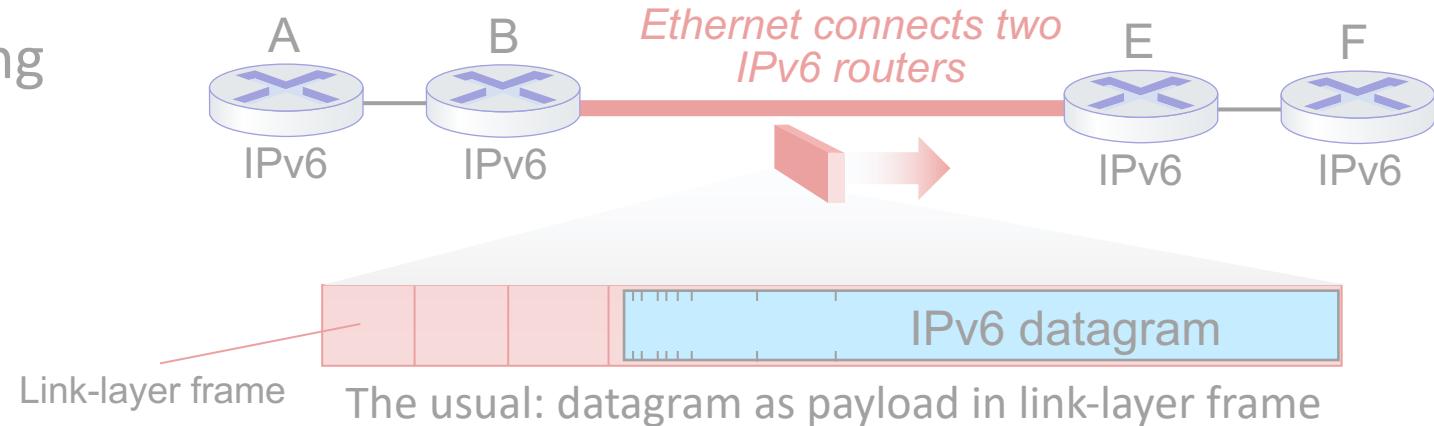


IPv4 network
connecting two
IPv6 routers

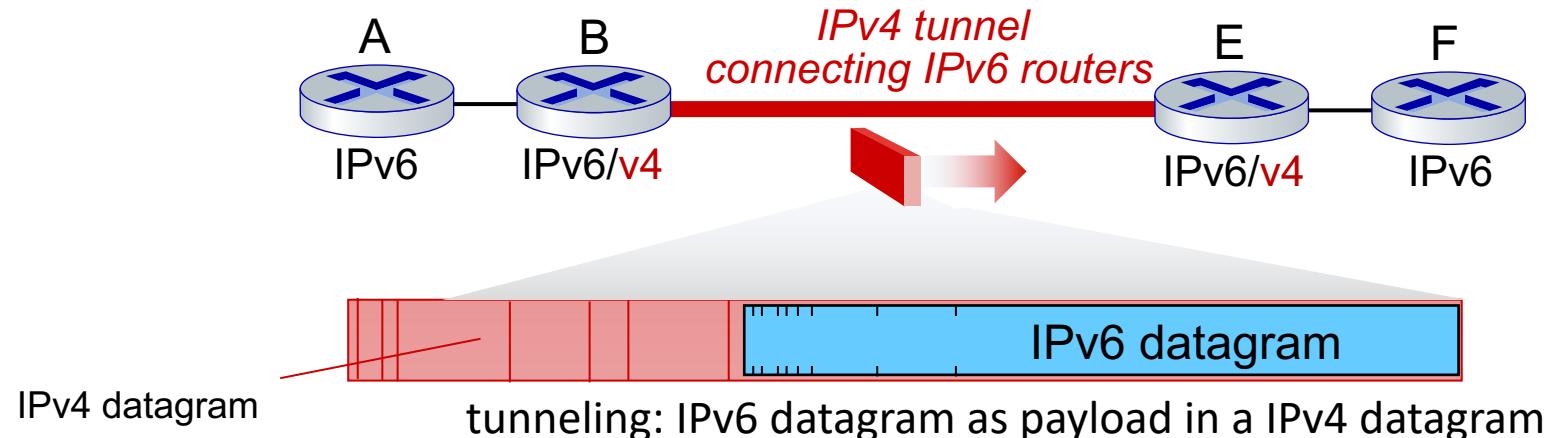


Tunneling and encapsulation

Ethernet connecting
two IPv6 routers:



IPv4 tunnel
connecting two
IPv6 routers



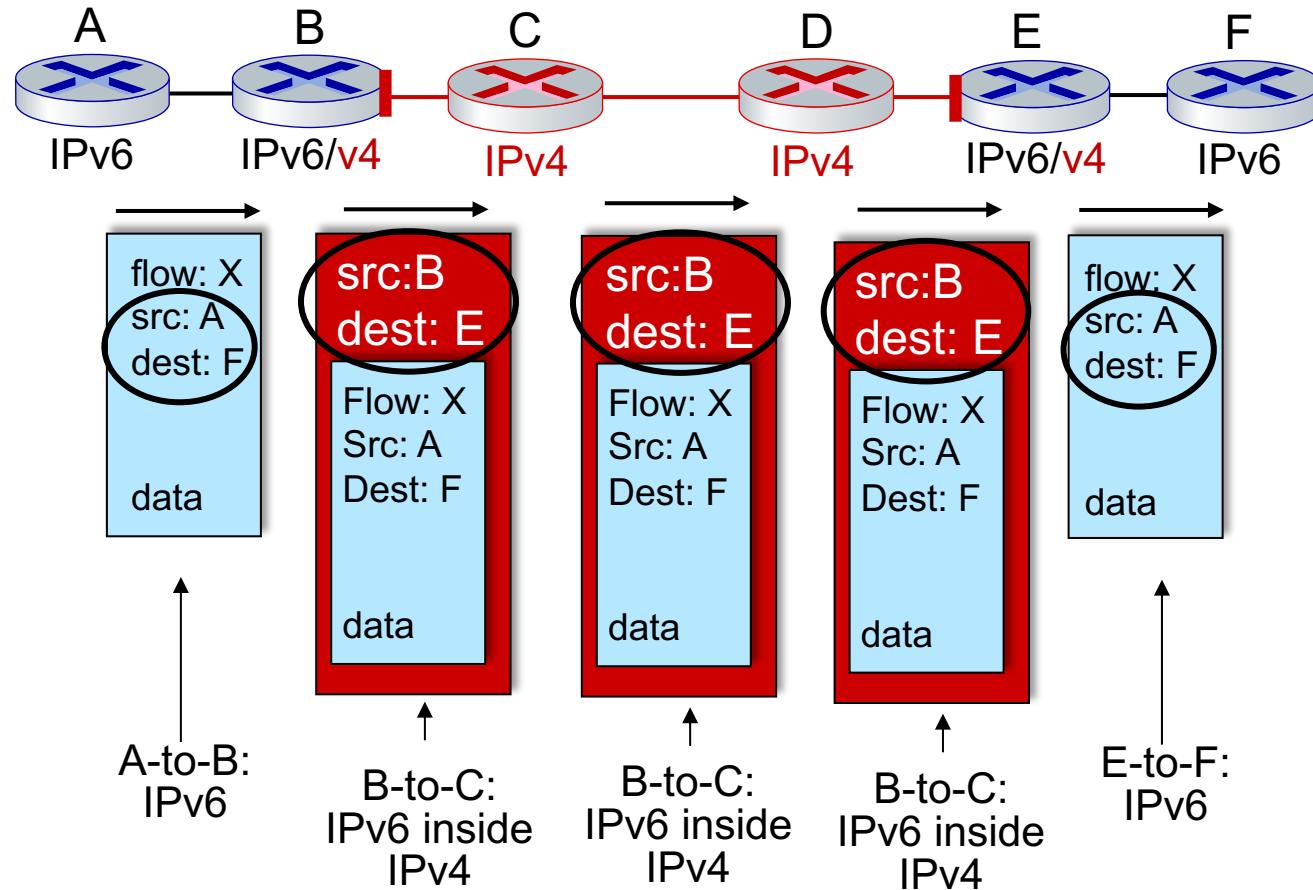
Tunneling

logical view:



physical view:

Note source and destination addresses!

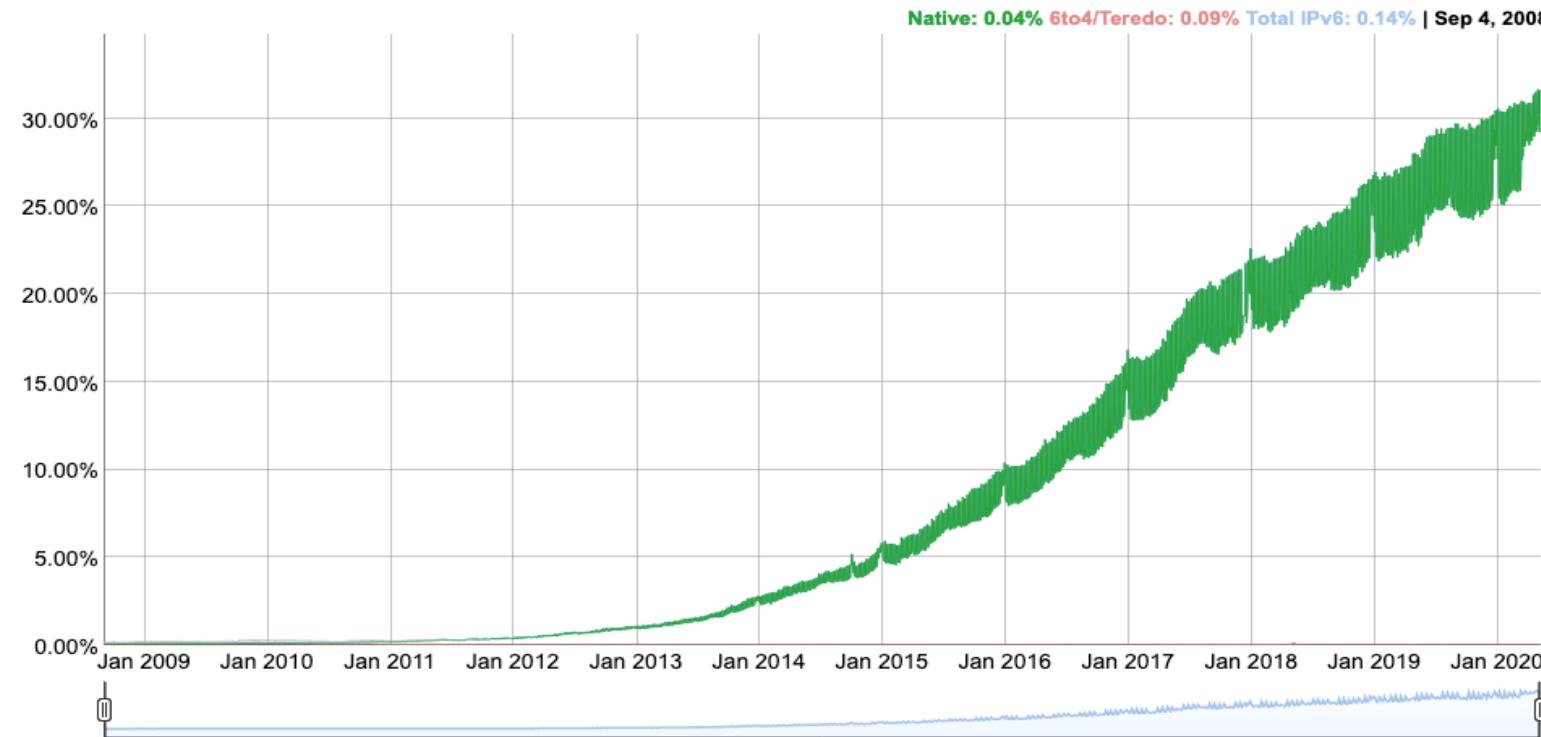


IPv6: adoption

- Google¹: ~ 30% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable

IPv6 Adoption

We are continuously measuring the availability of IPv6 connectivity among Google users. The graph shows the percentage of users that access Google over IPv6.



1

<https://www.google.com/intl/en/ipv6/statistics.html>

IPv6: adoption

- Google¹: ~ 30% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable
- Long (long!) time for deployment, use
 - 25 years and counting!
 - think of application-level changes in last 25 years: WWW, social media, streaming media, gaming, telepresence, ...
 - *Why?*

¹ <https://www.google.com/intl/en/ipv6/statistics.html>

Architectural Principles of the Internet

RFC 1958

“Many members of the Internet community would argue that there is no architecture, but only a tradition, which was not written down for the first 25 years (or at least not by the IAB). However, in very general terms, the community believes that **the goal is connectivity, the tool is the Internet Protocol, and the intelligence is end to end rather than hidden in the network.**”

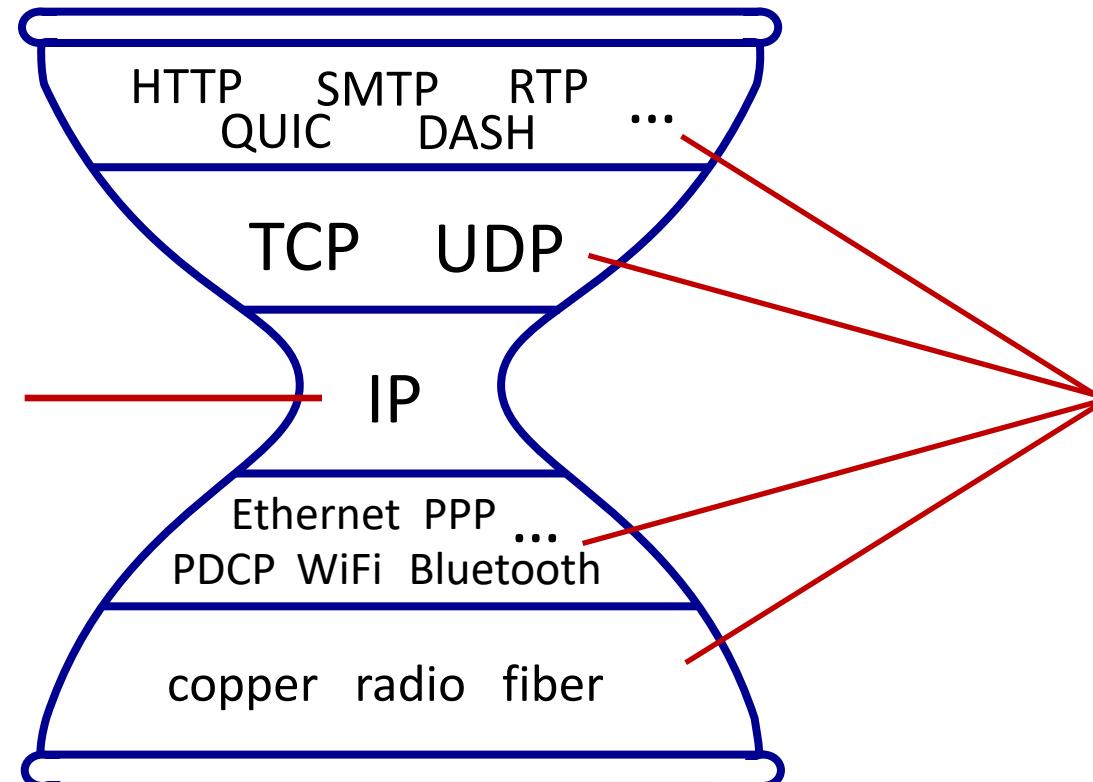
Three cornerstone beliefs:

- simple connectivity
- IP protocol: that narrow waist
- intelligence, complexity at network edge

The IP hourglass

Internet's "thin waist":

- *one* network layer protocol: IP
- *must* be implemented by every (billions) of Internet-connected devices

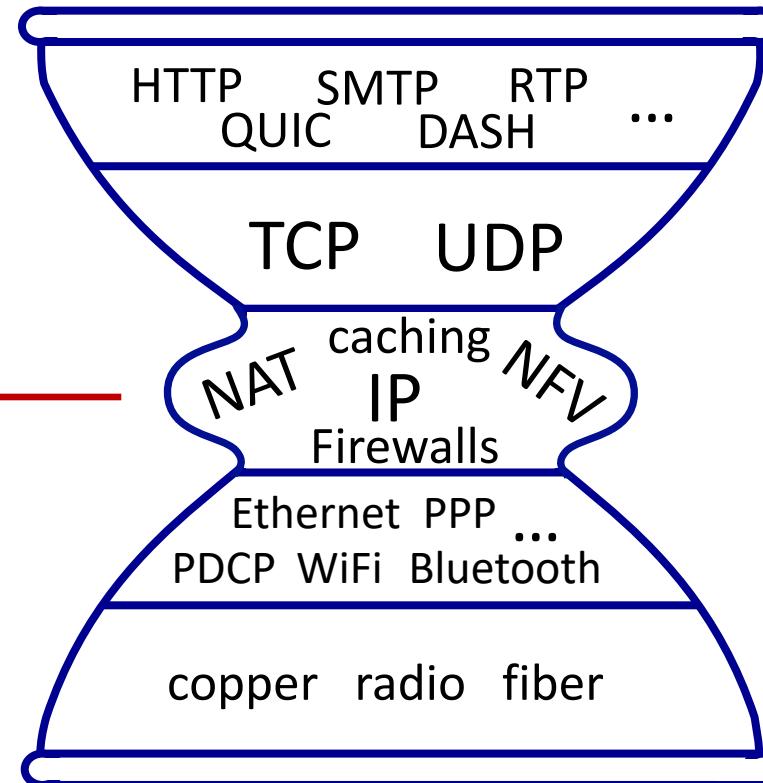


many protocols in physical, link, transport, and application layers

The IP hourglass, at middle age

Internet's middle age
“love handles”?

- middleboxes,
operating inside the
network



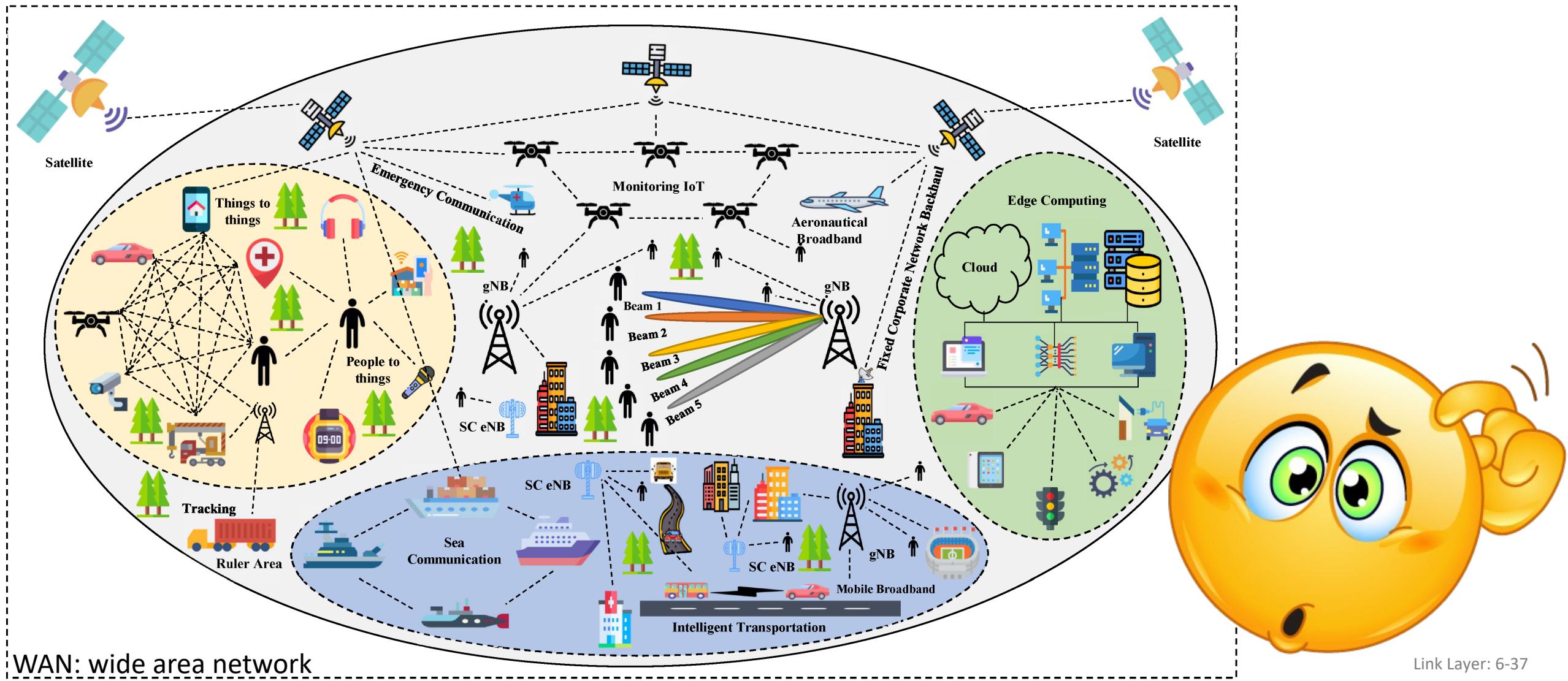
Chapter 4: Summary II

- Network Address Translation
 - Connect end devices without a proper IP address to the Internet
- IPv6
 - Transition is really hard
- Tunneling
 - General mechanism in networking
 - Encapsulation of data
 - Increasing the layer stack

More problems and solution ideas

- What problems remain?
 - Discussion in small groups

Real world application – finding the best path



Topics for next lecture

1. Routing algorithms: Dijkstra and Bellman-Ford
2. Intra domain routing
3. Inter domain routing