Computer Networks

(SCC.203)

Week 16-2

Muhammad Bilal

IP Addressing

DHCP, Host IP, Network IP Address space

IP addresses: how to get one?

That's actually two questions:

1. Q: How does a *host* get IP address within its network (host part of address)?
2. Q: How does a *network* get IP address for itself (network part of address)

How does *host* get IP address?

- hard-coded by sysadmin in config file (e.g., /etc/rc.config in UNIX)
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

DHCP: Dynamic Host Configuration Protocol

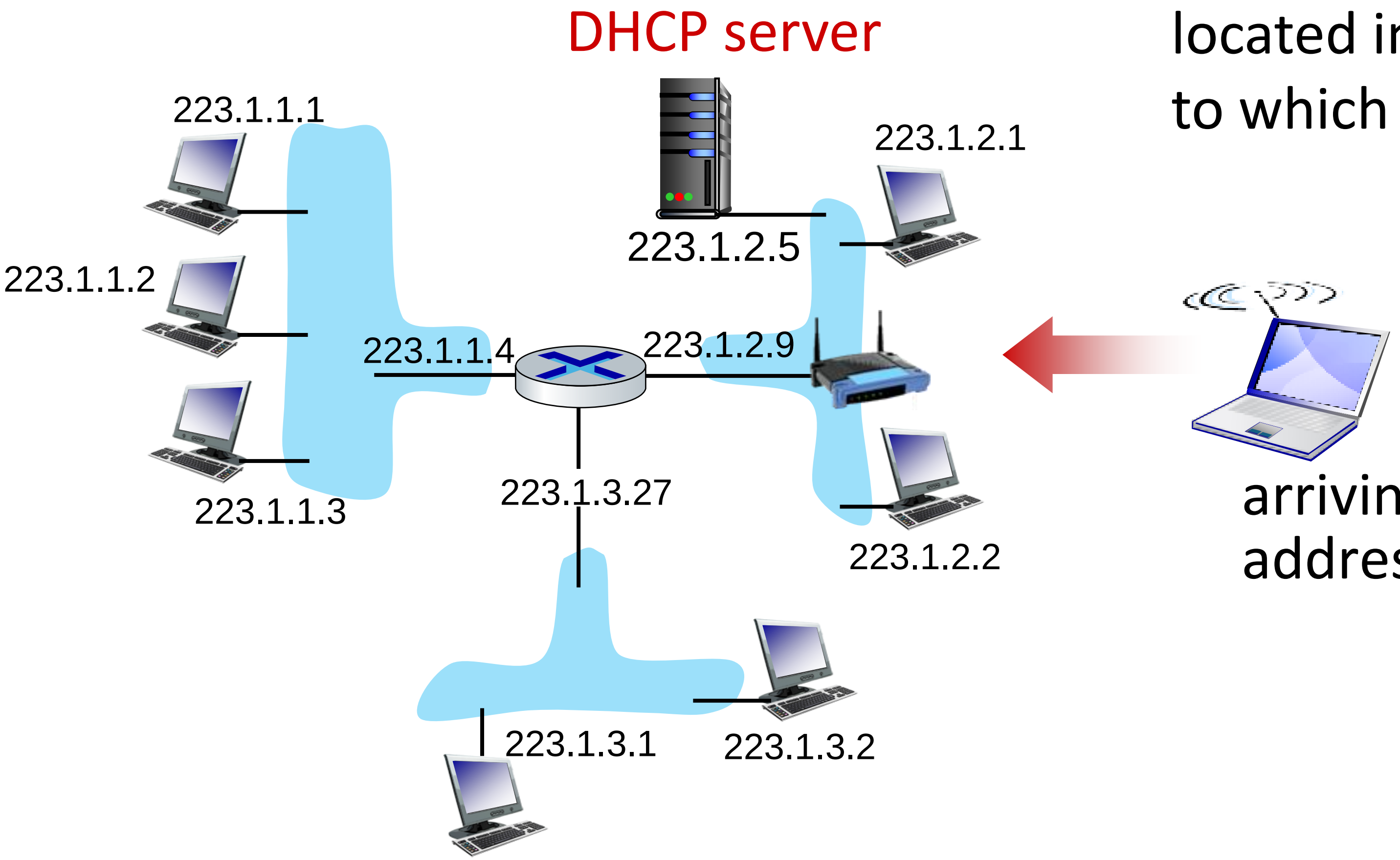
goal: host *dynamically* obtains IP address from network server when it "joins" network

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

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

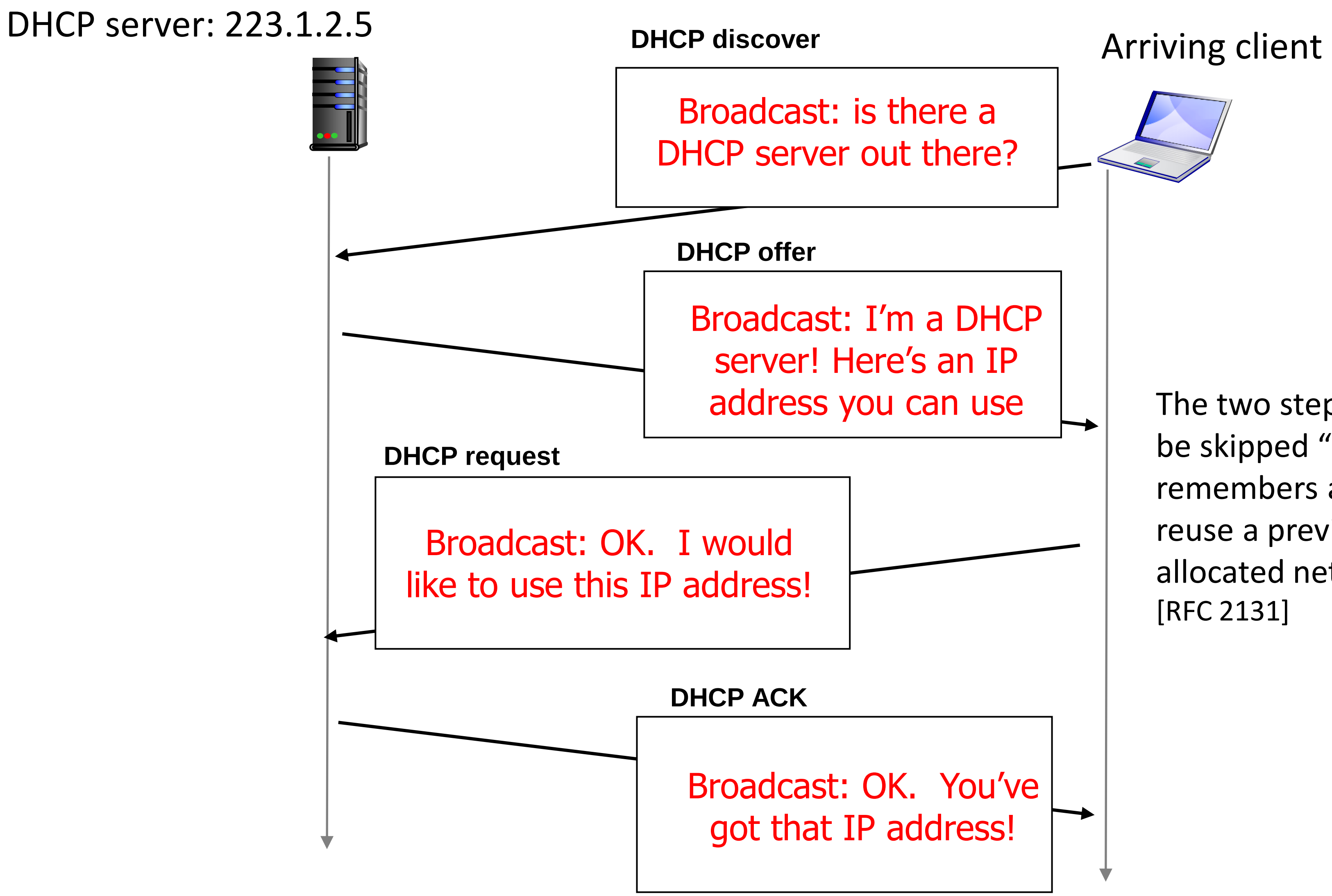
DHCP client-server scenario



Typically, DHCP server will be co-located in router, serving all subnets to which router is attached

arriving DHCP client needs address in this network

DHCP client-server scenario



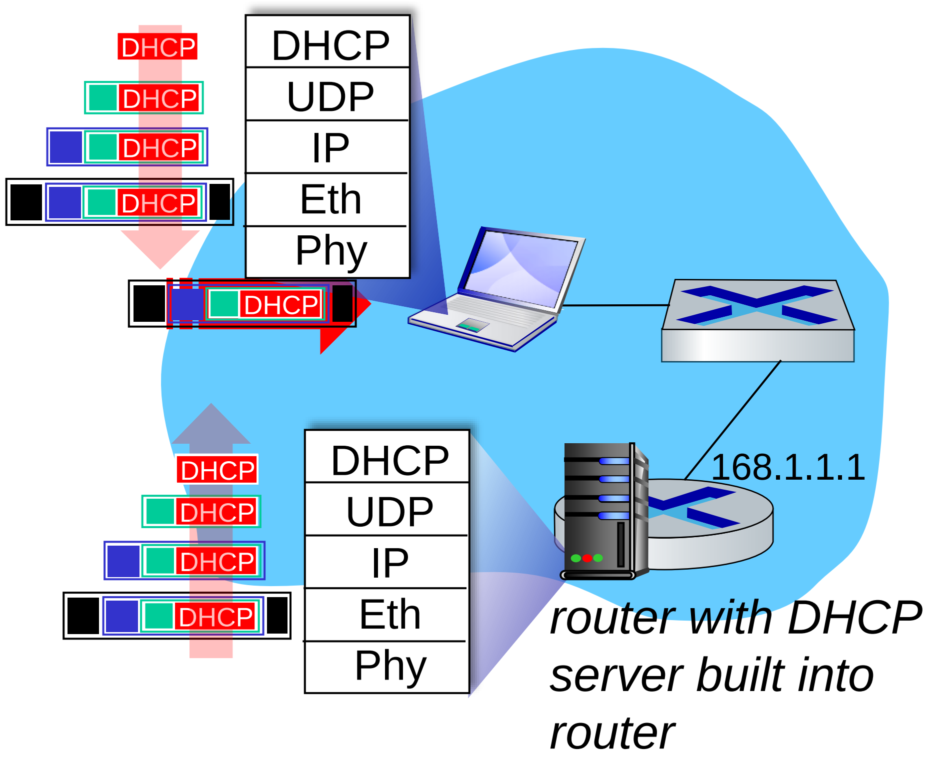
The two steps above can be skipped "if a client remembers and wishes to reuse a previously allocated network address" [RFC 2131]

DHCP: more than IP addresses

DHCP can return more than just allocated IP address on subnet:

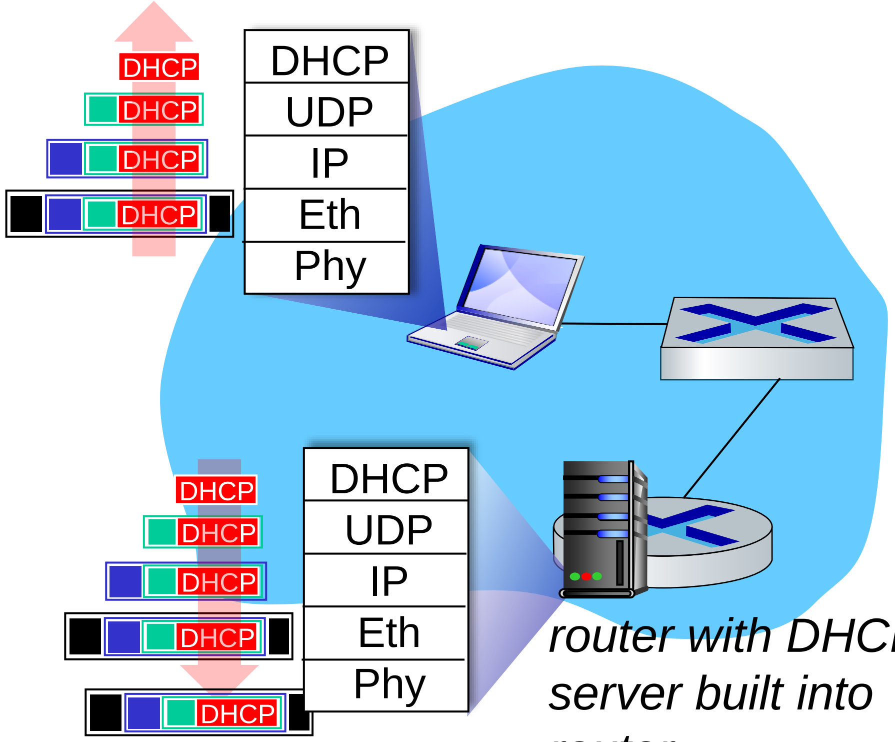
- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

DHCP: example



- Connecting laptop will use DHCP to get IP address, address of first-hop router, address of DNS server.
- DHCP REQUEST message encapsulated in UDP, encapsulated in IP, encapsulated in Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demux'ed to IP demux'ed, UDP demux'ed to DHCP

DHCP: example



router with DHCP server built into router

- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulated DHCP server reply forwarded to client, demuxing up to DHCP at client
- client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router

DHCP Pros and Cons

Pros

- Relieves the network administrator of manual configuration
- Device can be moved from network to network and automatically obtain valid configuration parameters for the current network
- IP addresses are only allocated when needed
- Conserve /reduce total number of addresses in us

Cons

- When DHCP server is unavailable, client is unable to access the enterprise's network
- Security Problems
  - Uses UDP, an unreliable and insecure protocol
  - DHCP is an unauthenticated protocol
  - When connecting to a network, the user is not required to provide credentials in order to obtain a lease
- DNS cannot be used for DHCP configured hosts

IP addresses: how to get one?

Q: how does *network* get subnet part of IP address?

A: gets allocated portion of its provider ISP's address space

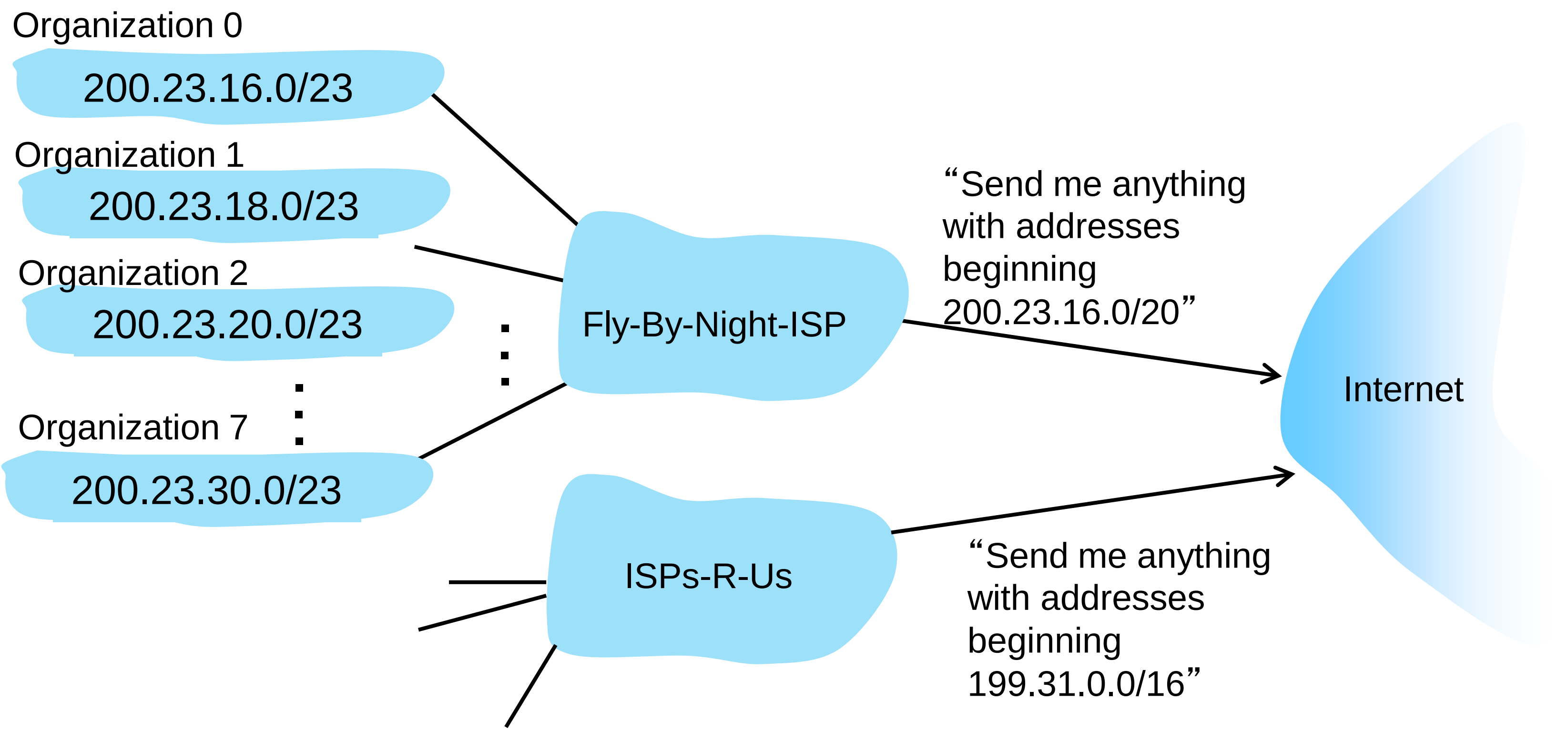
ISP's block	<u>11001000 00010111 0001</u>0000 00000000	200.23.16.0/20

ISP can then allocate out its address space in 8 blocks:

Organization 0	<u>11001000 00010111 0001000</u>0 00000000	200.23.16.0/23
Organization 1	<u>11001000 00010111 0001001</u>0 00000000	200.23.18.0/23
Organization 2	<u>11001000 00010111 0001010</u>0 00000000	200.23.20.0/23
...	....	....
Organization 7	<u>11001000 00010111 0001111</u>0 00000000	200.23.30.0/23

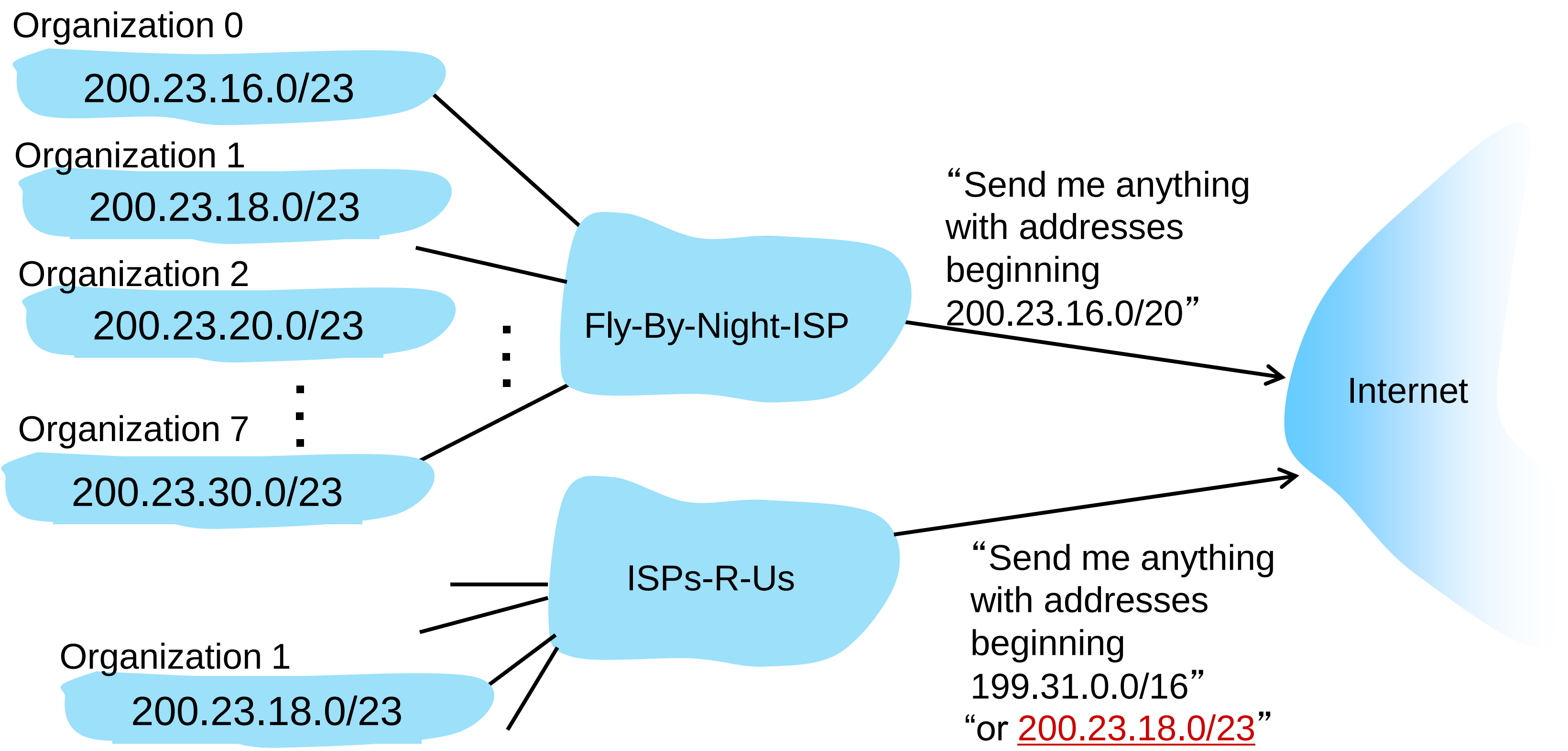
Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



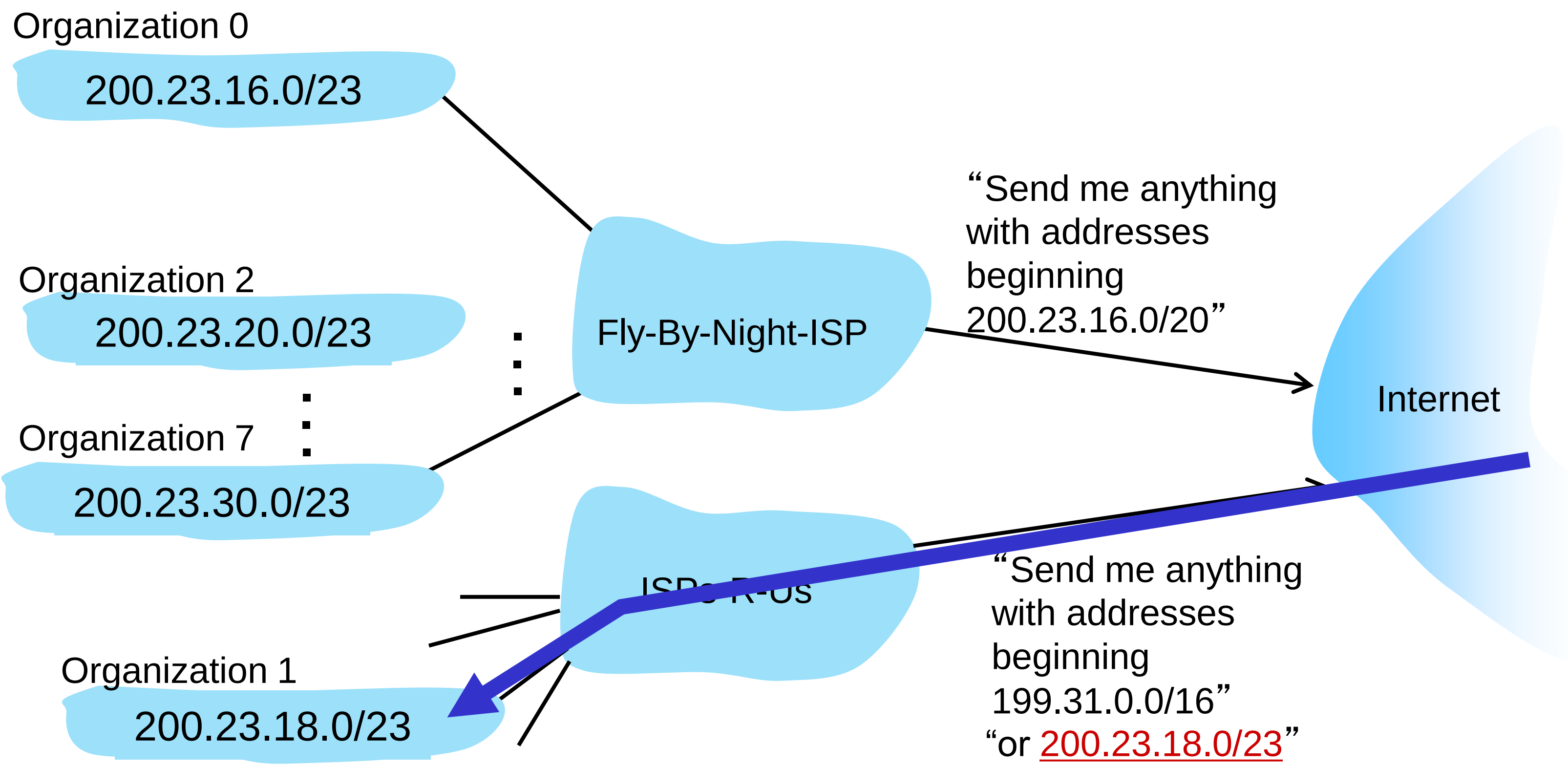
Hierarchical addressing: more specific routes

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



Hierarchical addressing: more specific routes

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1

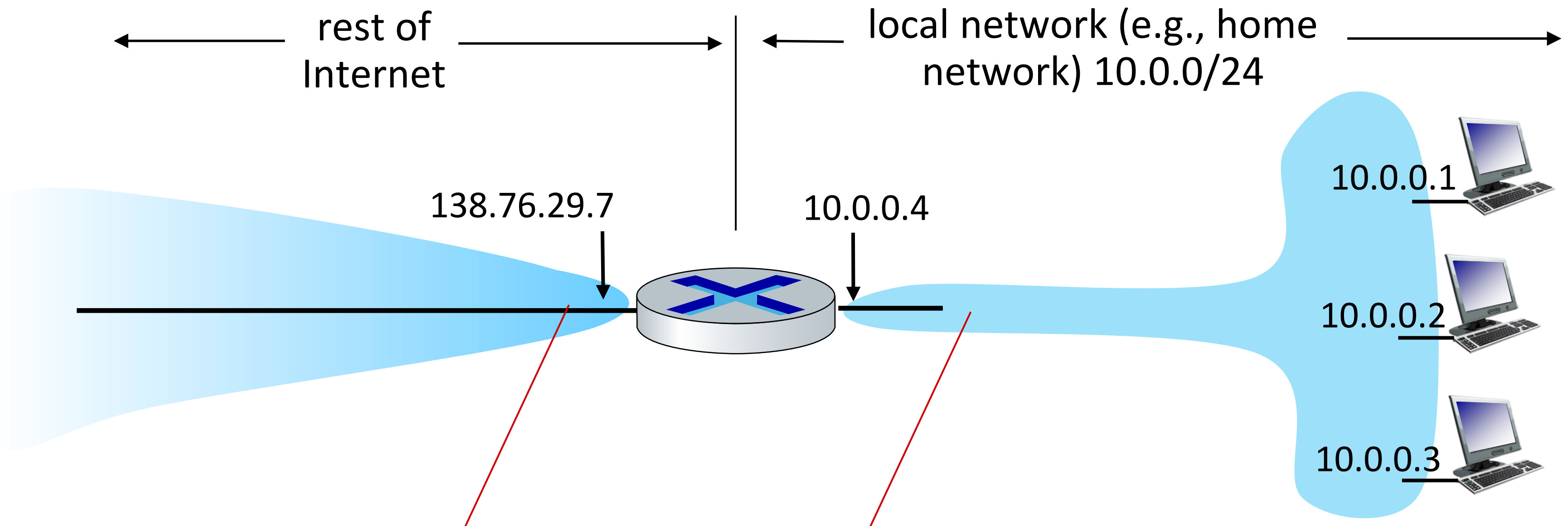


NAT: network address translation

Violating the fundamentals

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

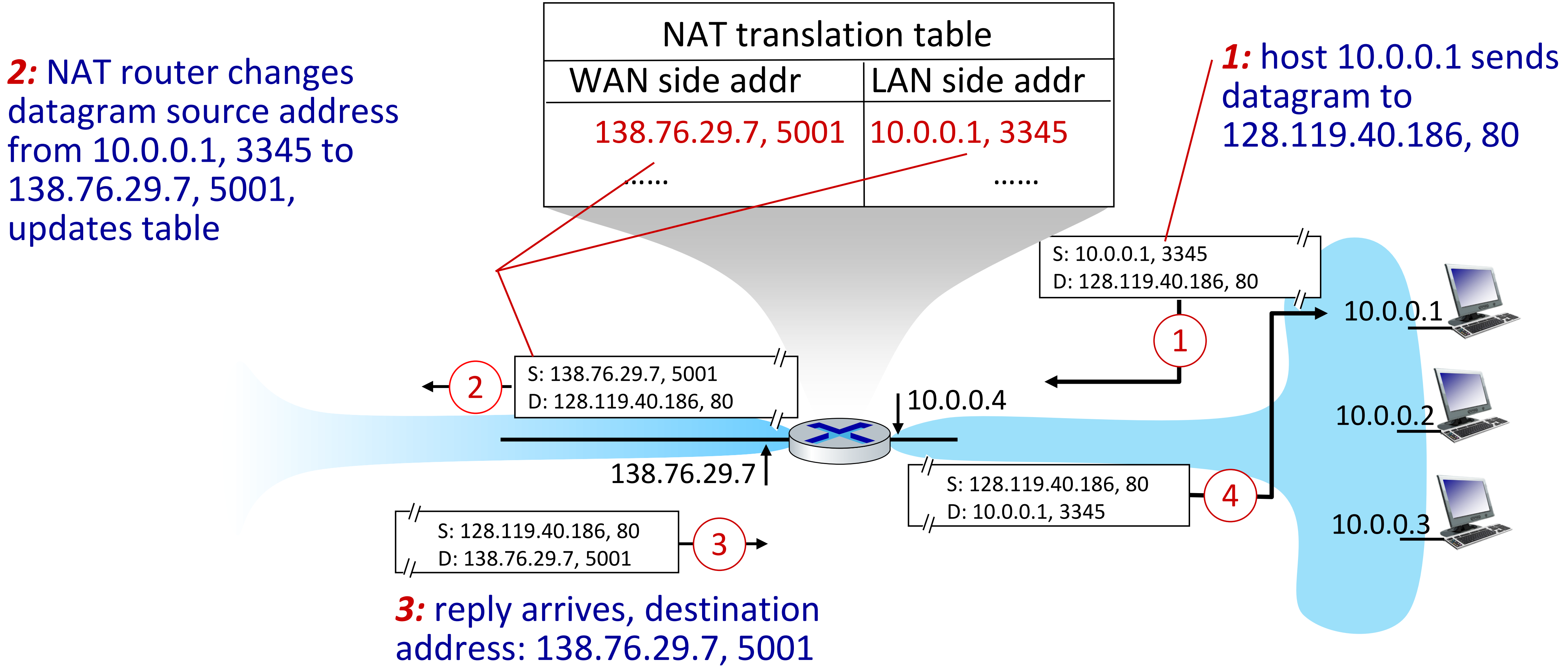
- all devices in local network have 32-bit addresses in a "private" IP address space (10/8, 172.16/12, 192.168/16 prefixes) that can only be used in local network
- advantages:
  - just one IP address needed from provider ISP for *all* devices
  - can change addresses of host in local network without 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



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)
- but NAT is here to stay:
  - extensively used in home and institutional nets, 4G/5G cellular nets

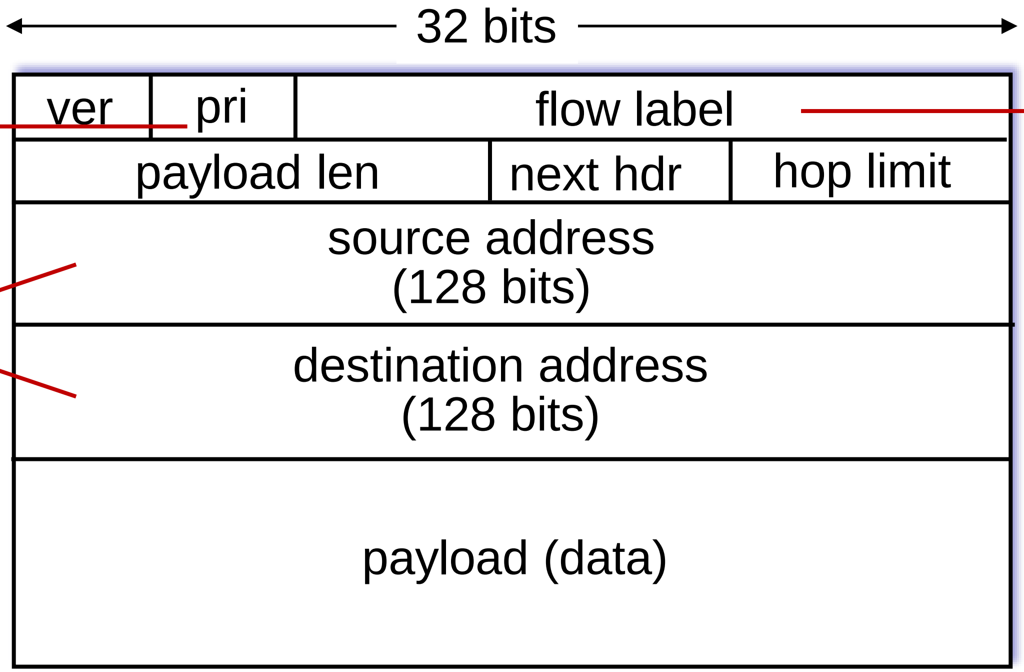
IPv6

More addresses, fixed header length, high speed processing

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).

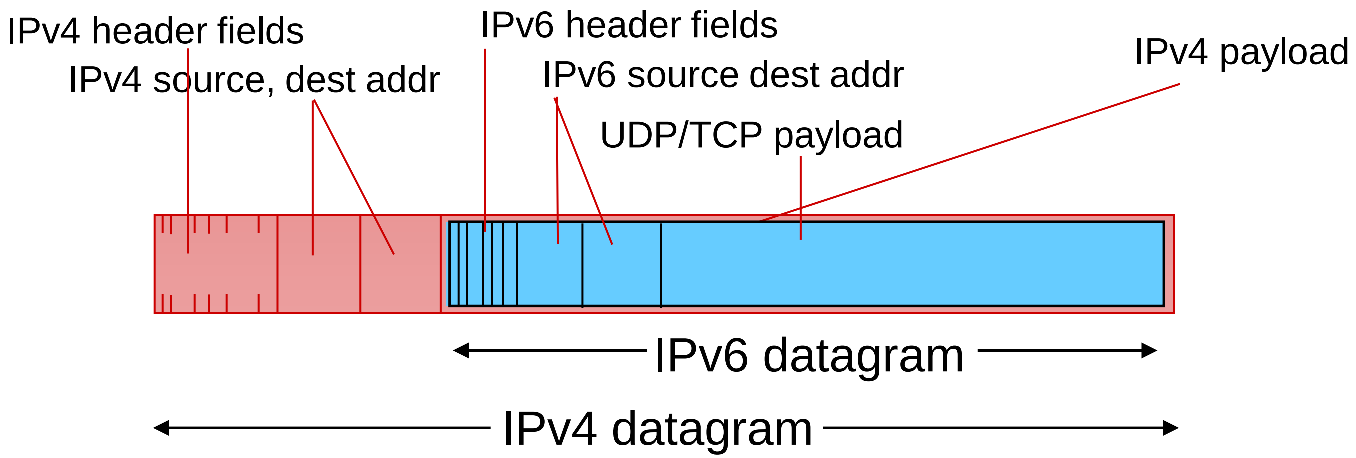
What's missing (compared with IPv4):

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

Transition from IPv4 to IPv6

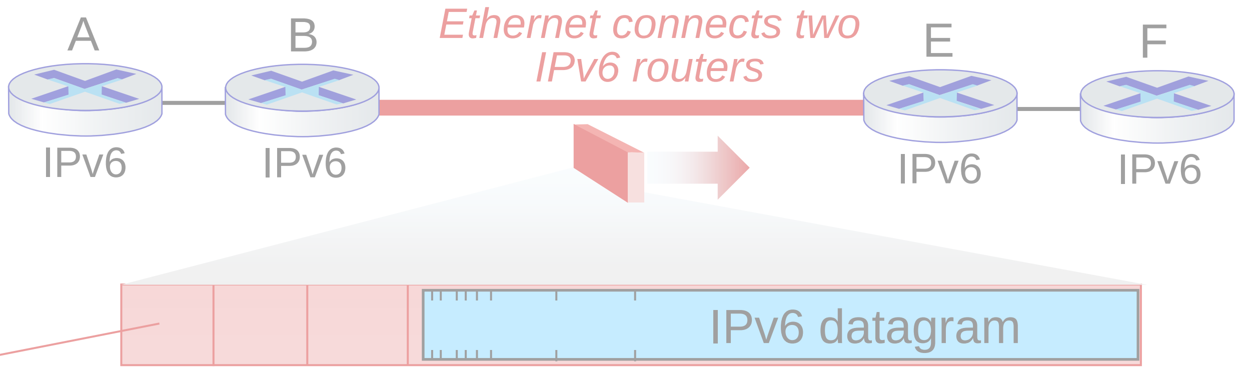
- not all routers can be upgraded simultaneously
  - no “flag days”
  - how will network operate with mixed IPv4 and IPv6 routers?

- tunneling: IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers (“packet within a packet”)
  - tunneling used extensively in other contexts (4G/5G)



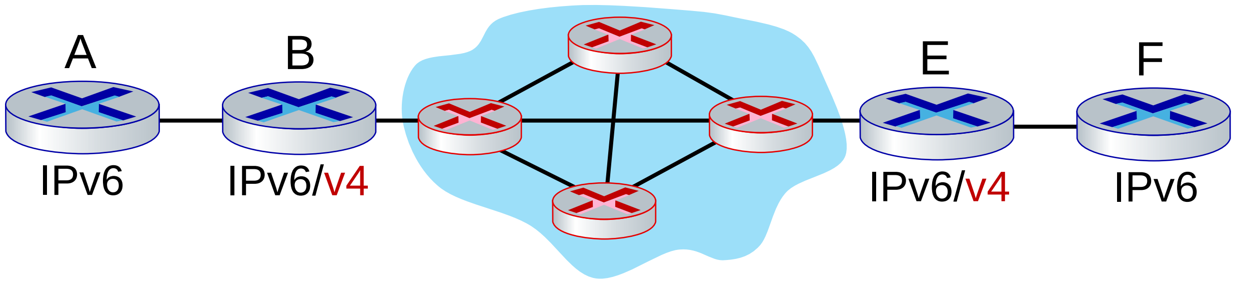
Tunneling and encapsulation

Ethernet connecting two IPv6 routers:



The usual: datagram as payload in link-layer frame

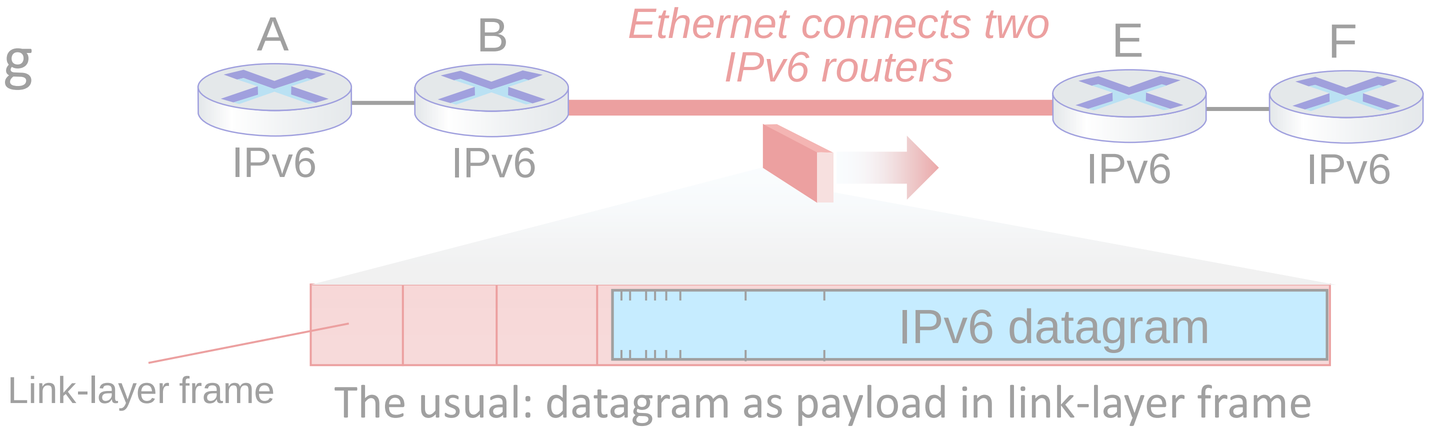
IPv4 network connecting two IPv6 routers



IPv4 network

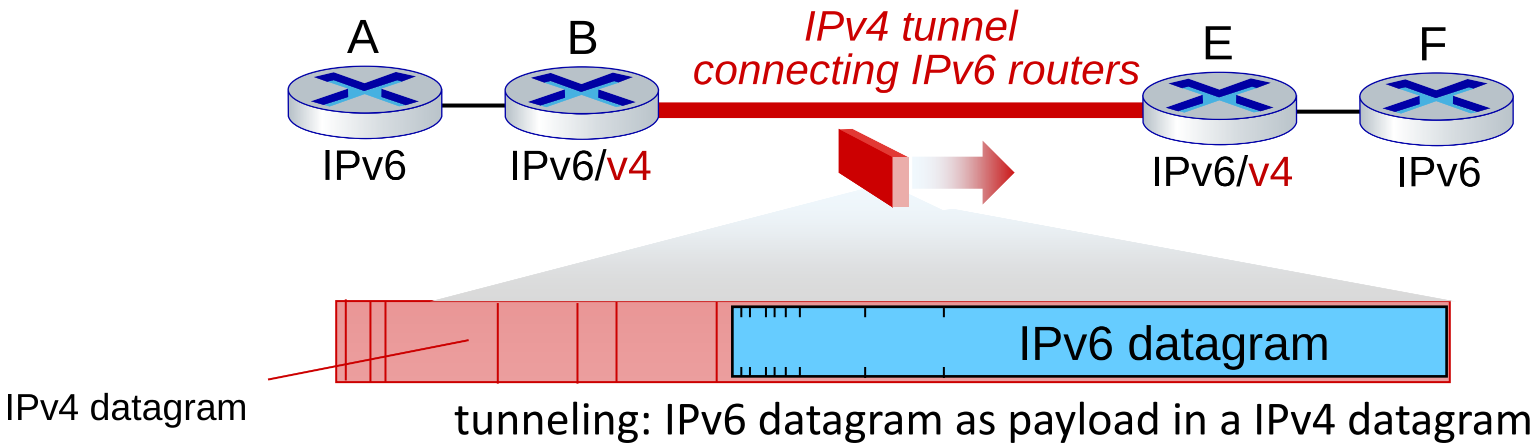
Tunneling and encapsulation

Ethernet connecting two IPv6 routers:



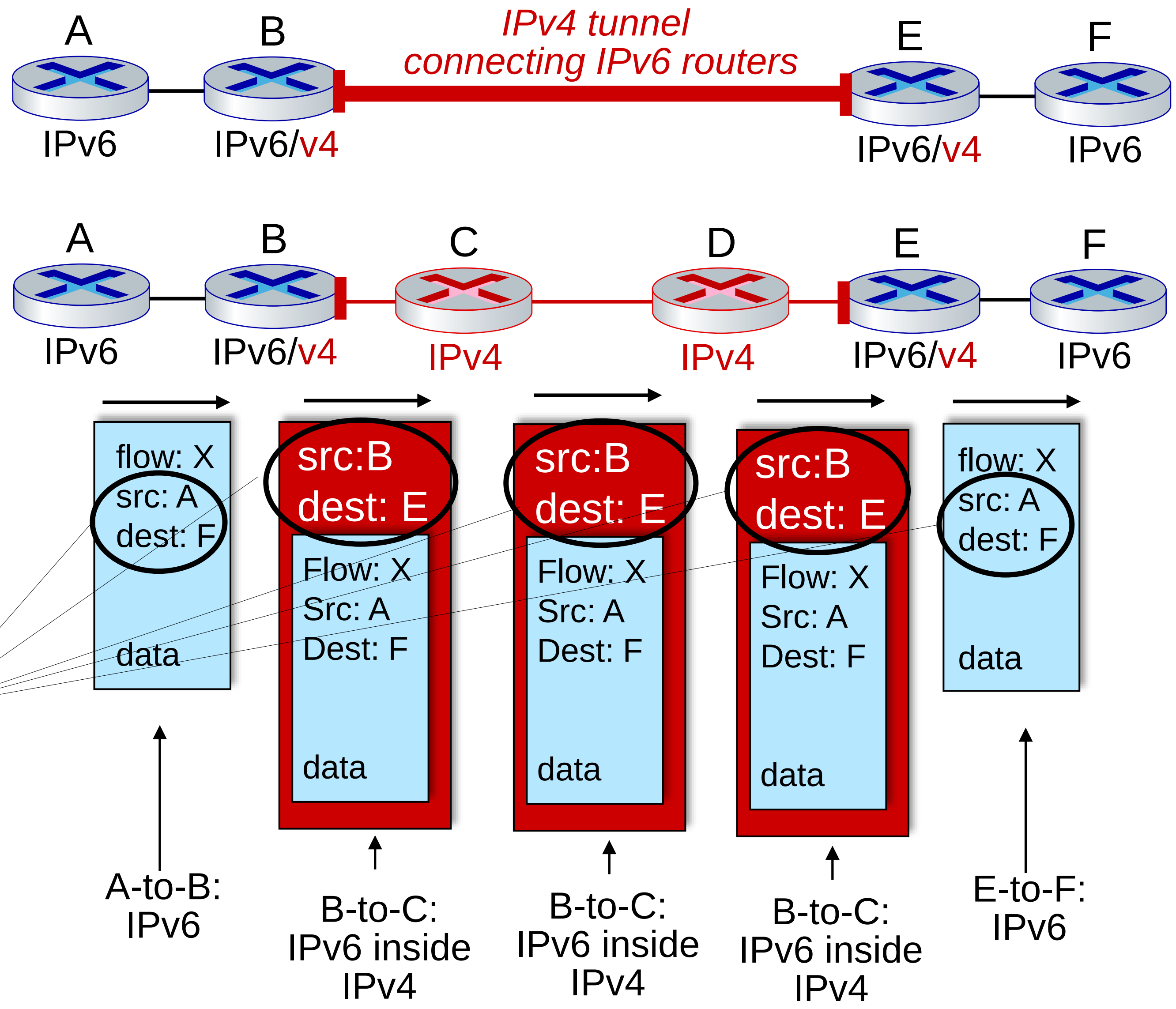
The usual: datagram as payload in link-layer frame

IPv4 tunnel connecting two IPv6 routers



tunneling: IPv6 datagram as payload in a IPv4 datagram

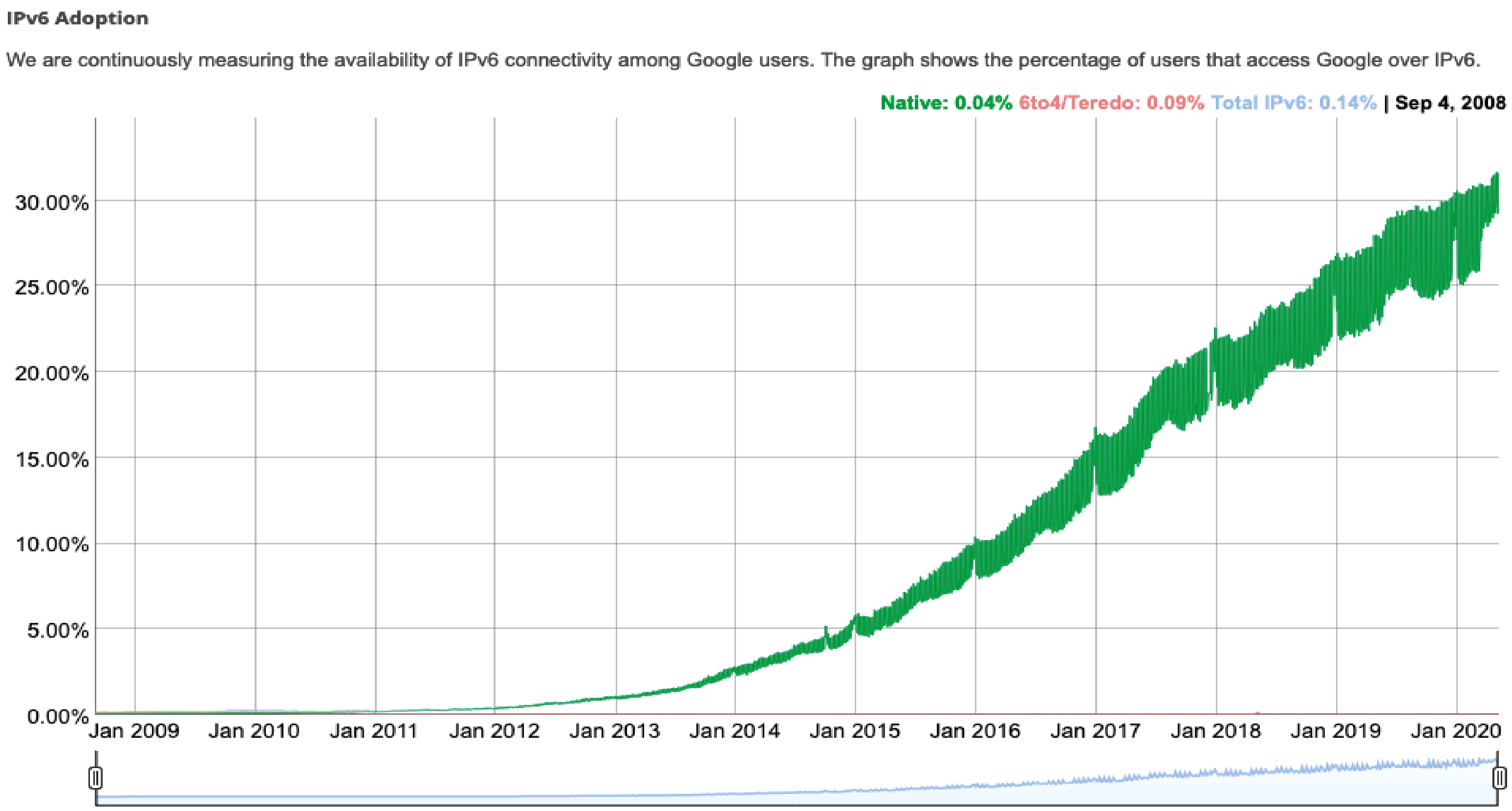
Tunneling



Note source and destination addresses!

IPv6: adoption

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



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

IPv6: adoption

- Google[1]: ~ 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

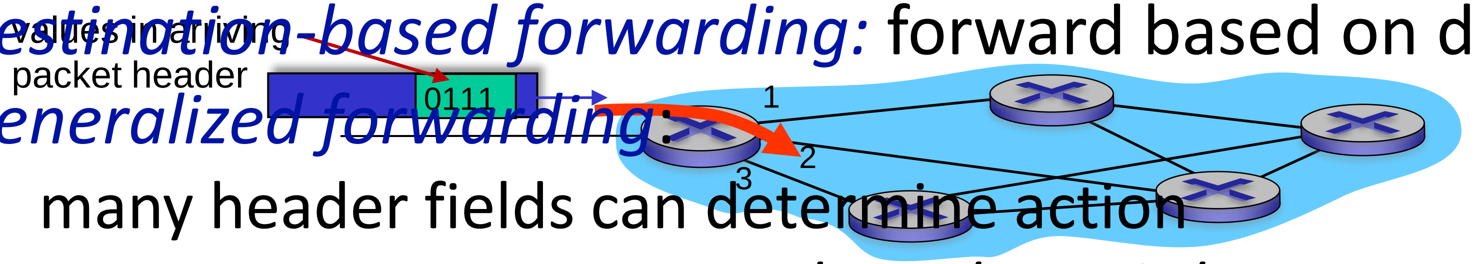
Network layer: Data Plane

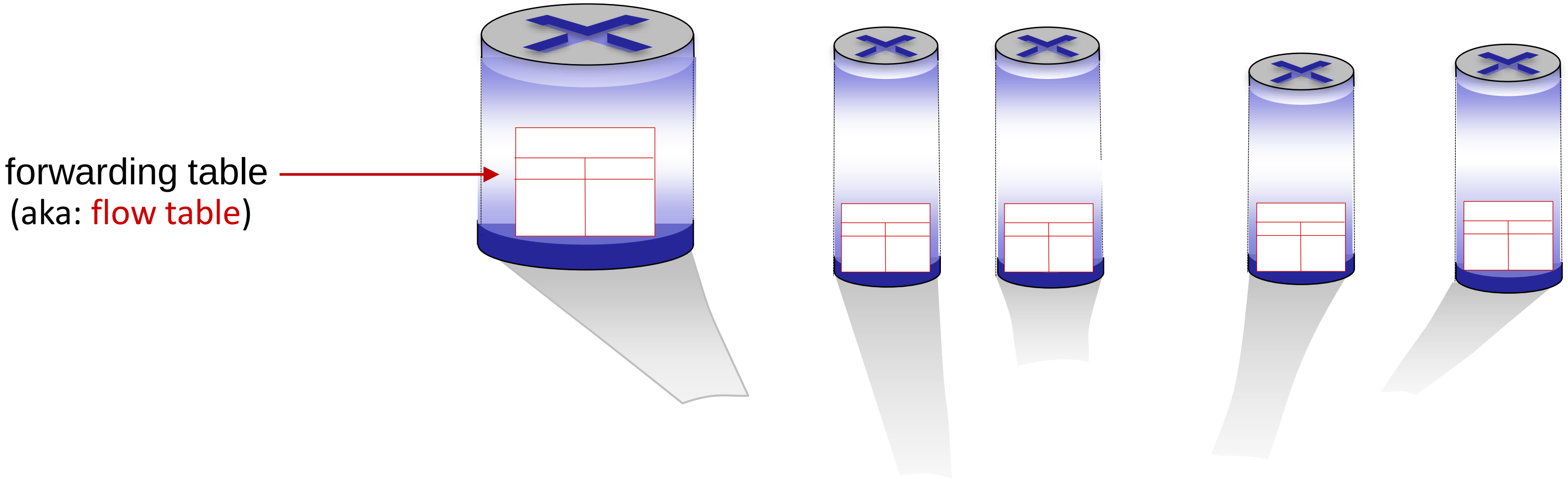
Generalized Forwarding, SDN, Middleboxes

Generalized forwarding: match plus action

Review: each router contains a forwarding table (aka: flow table)

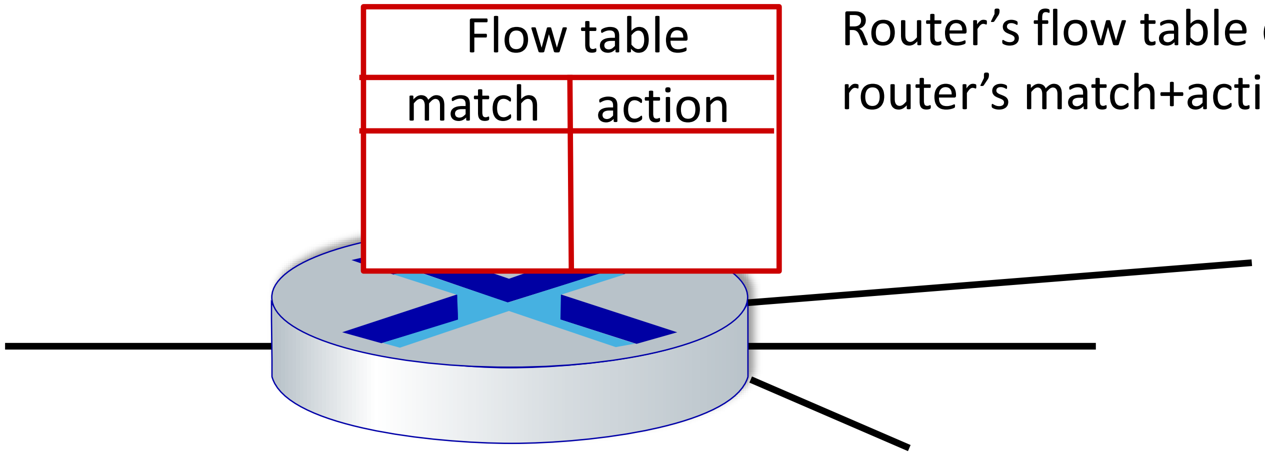
- "match plus action" abstraction: match bits in arriving packet, take action
  - *destination-based forwarding:* forward based on dest. IP address
  - *generalized forwarding:*
    - many header fields can determine action
    - many action possible: drop/copy/modify/log packet





Flow table abstraction

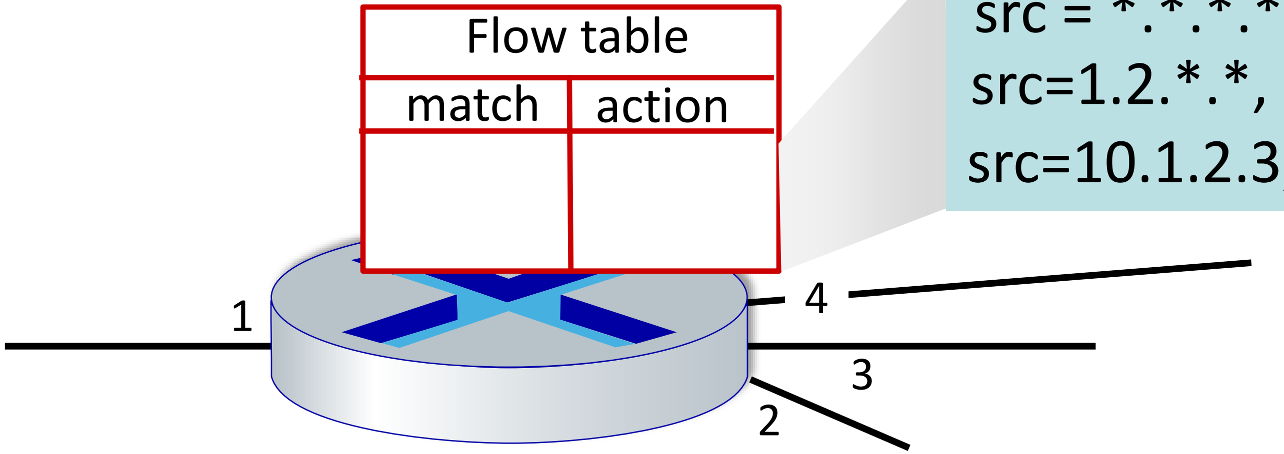
- flow: defined by header field values (in link-, network-, transport-layer fields)
- generalized forwarding: simple packet-handling rules
  - match: pattern values in packet header fields
  - actions: for matched packet: drop, forward, modify, matched packet or send matched packet to controller
  - priority: disambiguate overlapping patterns
  - counters: #bytes and #packets



Router's flow table define router's match+action rules

Flow table abstraction

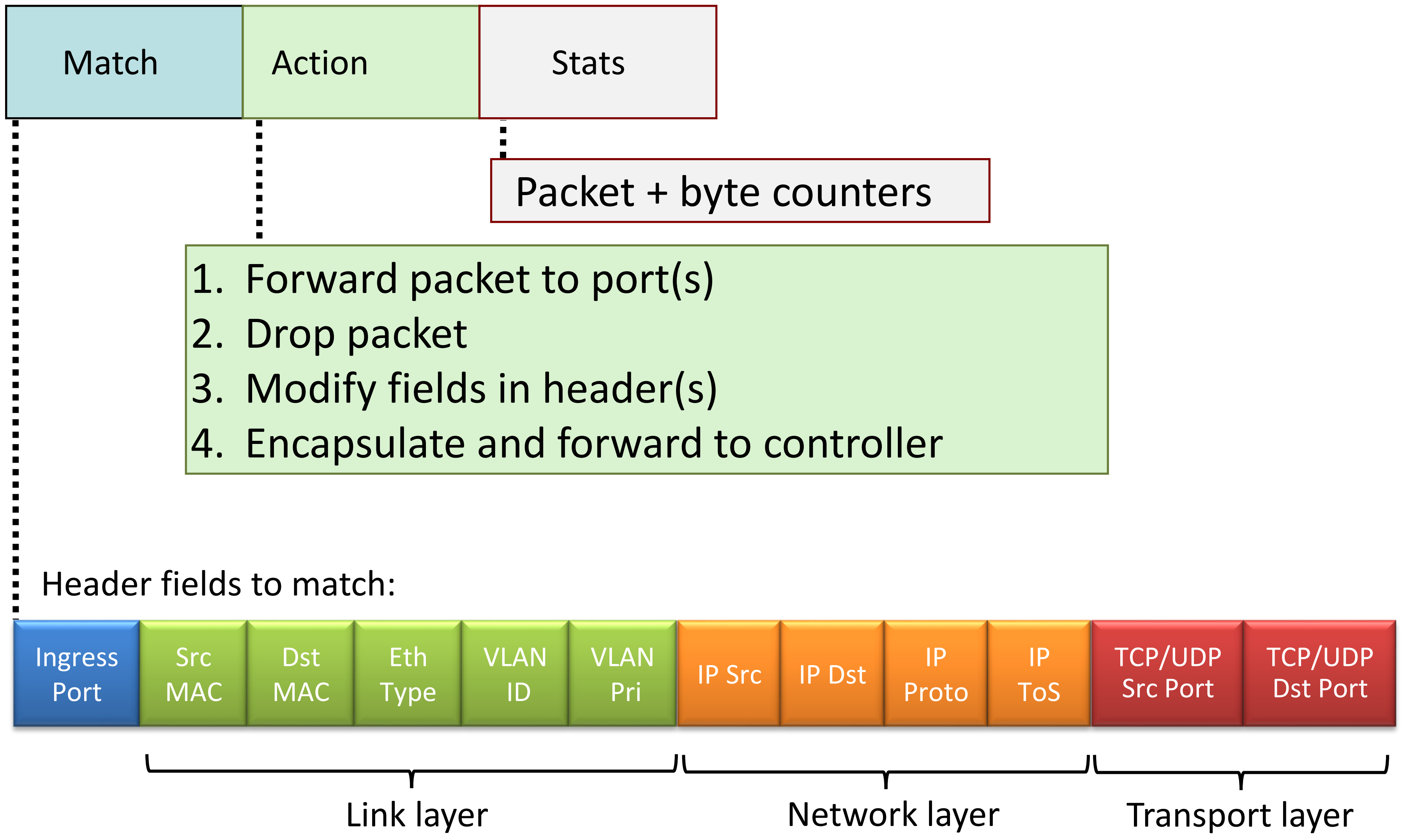
- flow: defined by header fields
- generalized forwarding: simple packet-handling rules
  - match: pattern values in packet header fields
  - actions: for matched packet: drop, forward, modify, matched packet or send matched packet to controller
  - priority: disambiguate overlapping patterns
  - counters: #bytes and #packets



src = *.*.*.*, dest=3.4.*.*	forward(2)
src=1.2.*.*, dest=*.*.*.*	drop
src=10.1.2.3, dest=*.*.*.*	send to controller

* : wildcard

OpenFlow: flow table entries



OpenFlow: examples

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	51.6.0.8	*	*	*	*	port6

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	*	*	*	*	22	drop

Block (do not forward) all datagrams destined to TCP port 22 (ssh port #)

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	128.119.1.1	*	*	*	*	*	drop

Block (do not forward) all datagrams sent by host 128.119.1.1

OpenFlow: examples

Layer 2 destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	22:A7:23:11:E1:02	*	*	*	*	*	*	*	*	*	port3

layer 2 frames with destination MAC address 22:A7:23:11:E1:02 should be forwarded to output port 3

OpenFlow abstraction

- match+action: abstraction unifies different kinds of devices

Router

- *match:* longest destination IP prefix
- *action:* forward out a link

Switch

- *match:* destination MAC address
- *action:* forward or flood

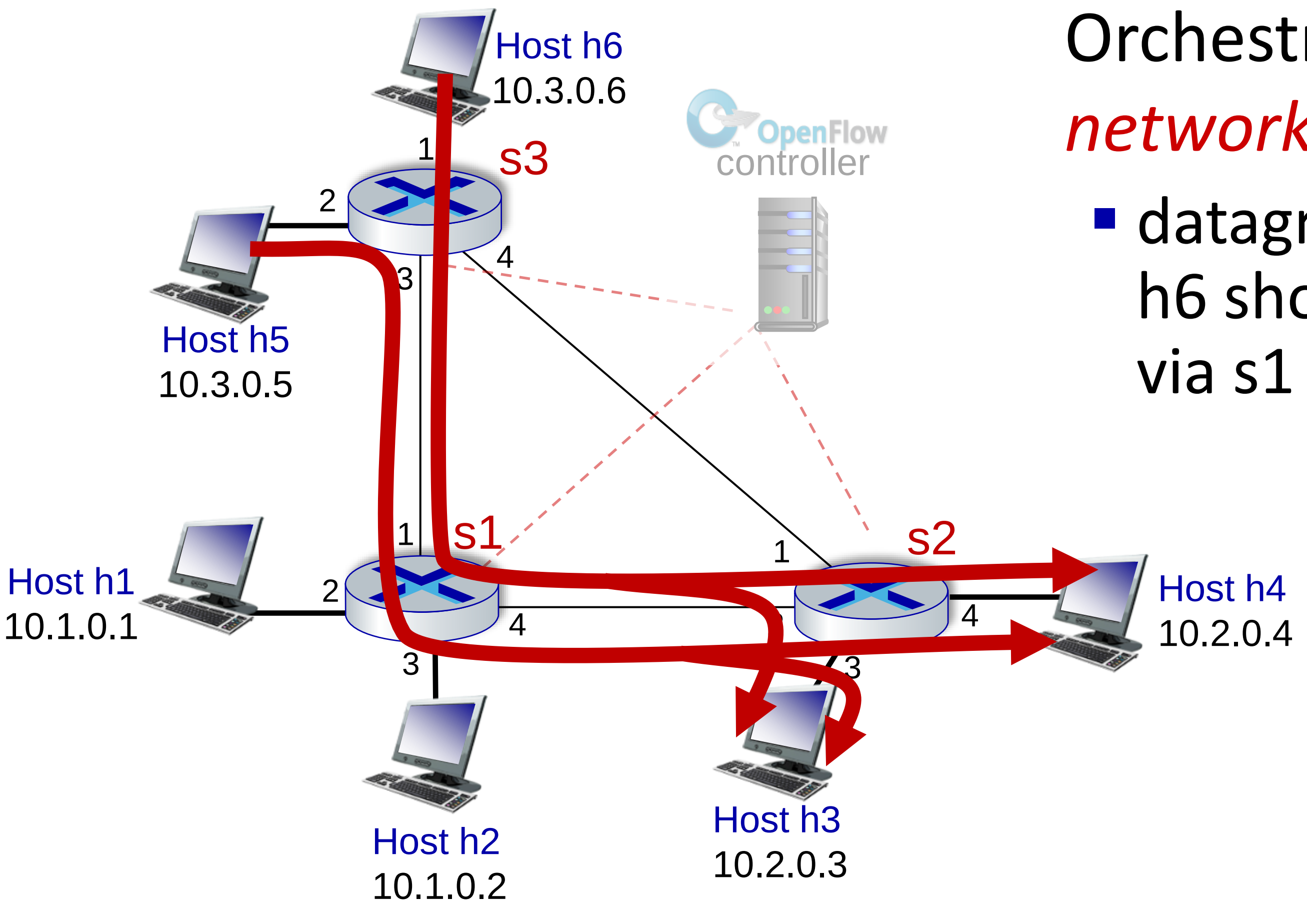
Firewall

- *match*: IP addresses and TCP/UDP port numbers
- *action:* permit or deny

NAT

- *match:* IP address and port
- *action:* rewrite address and port

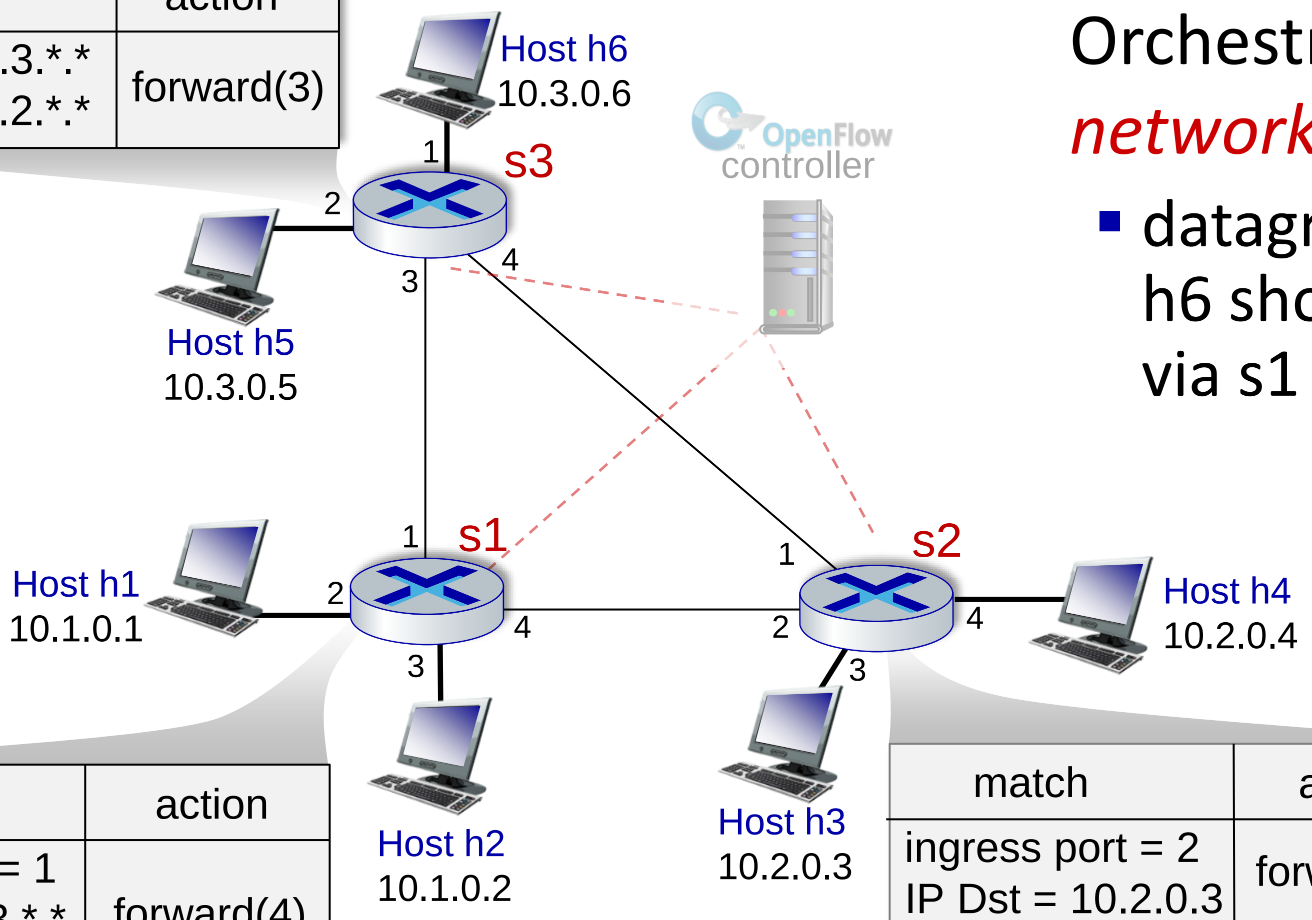
OpenFlow example



Orchestrated tables can create *network-wide* behavior, e.g.,:

- datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2

OpenFlow example



match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)

match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 2 IP Dst = 10.2.0.3	forward(3)
ingress port = 2 IP Dst = 10.2.0.4	forward(4)

Orchestrated tables can create *network-wide* behavior, e.g.,:

- datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2

Generalized forwarding: summary

- "match plus action" abstraction: match bits in arriving packet header(s) in any layers, take action
  - matching over many fields (link-, network-, transport-layer)
  - local actions: drop, forward, modify, or send matched packet to controller
  - "program" *network-wide* behaviors
- simple form of "network programmability"
  - programmable, per-packet "processing"
  - *historical roots:* active networking
  - *today:* more generalized programming: P4 (see p4.org).

Middleboxes

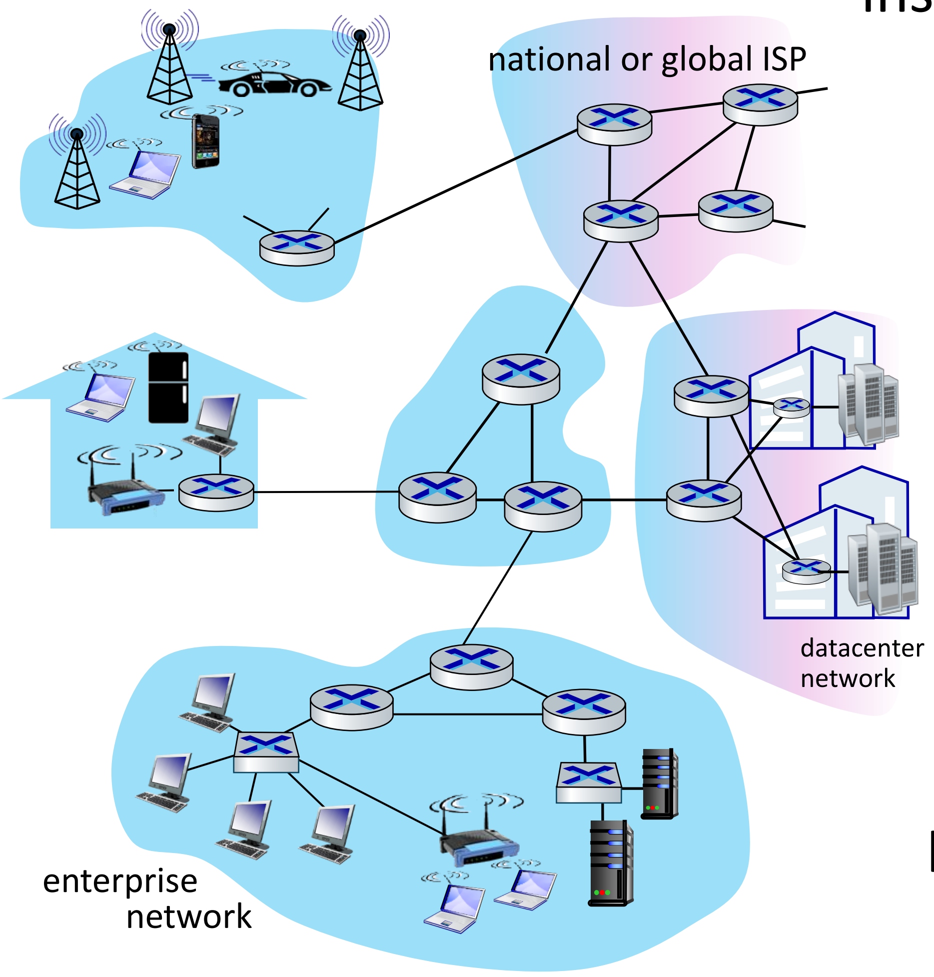
Middlebox (RFC 3234)

“any intermediary box performing functions apart from normal, standard functions of an IP router on the data path between a source host and destination host”

Middleboxes everywhere!

NAT: home, cellular, institutional

Application-specific: service providers, institutional, CDN



Firewalls, IDS: corporate, institutional, service providers, ISPs

Load balancers: corporate, service provider, data center, mobile nets

Caches: service provider, mobile, CDNs

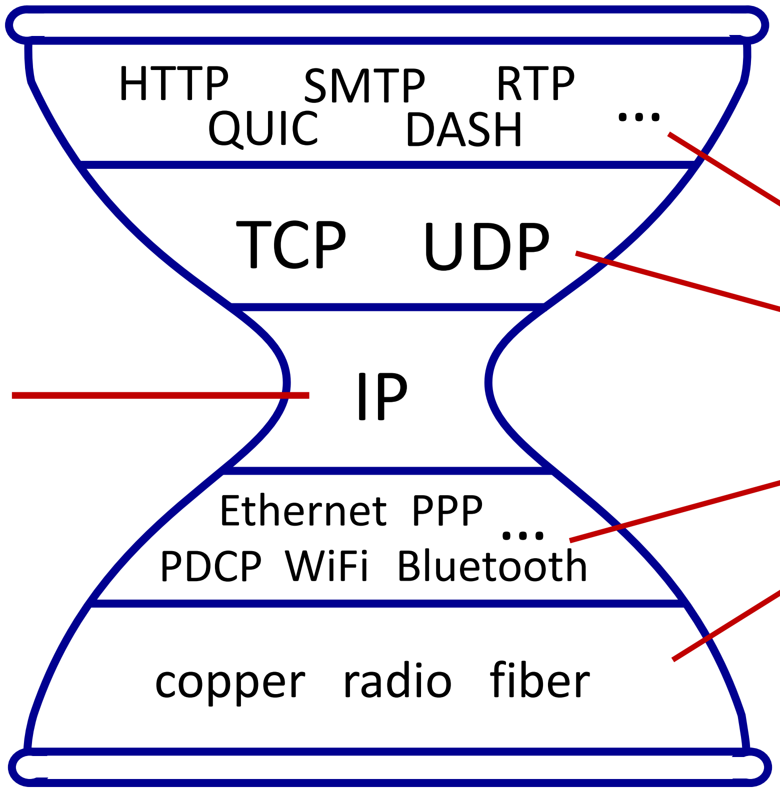
Middleboxes

- initially: proprietary (closed) hardware solutions
- move towards “whitebox” hardware implementing open API
  - move away from proprietary hardware solutions
  - programmable local actions via match+action
  - move towards innovation/differentiation in software
- SDN: (logically) centralized control and configuration management often in private/public cloud
- network functions virtualization (NFV): programmable services over white box networking, computation, storage

The IP hourglass

Internet's "thin waist":

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

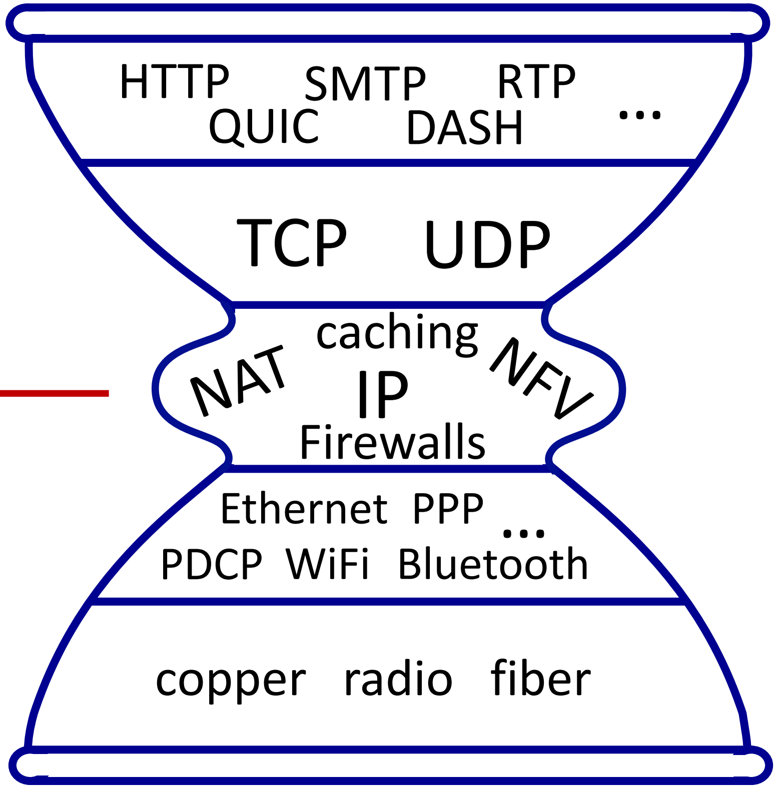


many protocols in physical, link, transport, and application layers

The IP hourglass, at middle age

Internet's middle age "love handles"?

- middleboxes, operating inside the network



Architectural Principles of the Internet

RFC 1958

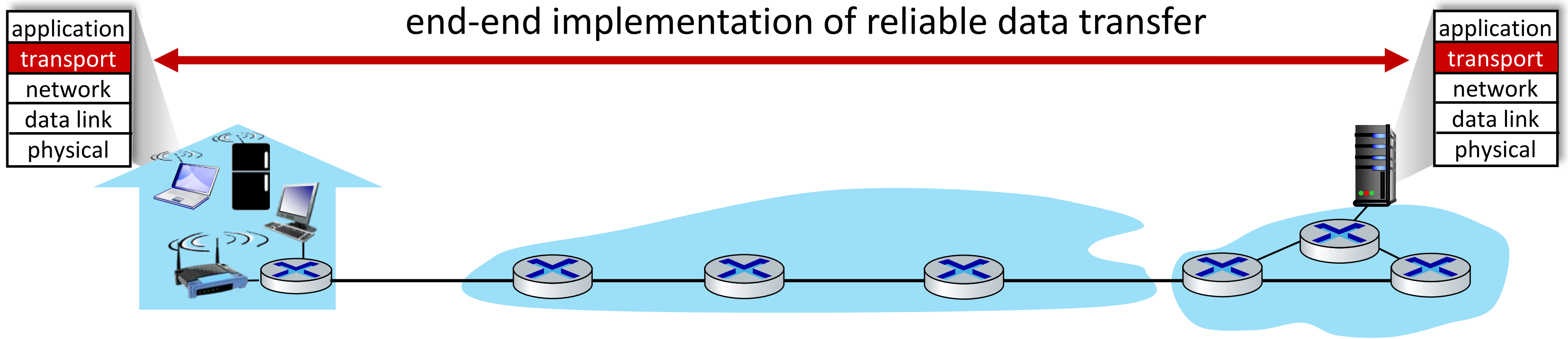
"Many members of the Internet community would argue that there is no architecture, but only a tradition, which was not written down for the first 25 years (or at least not by the IAB). However, in very general terms, the community believes that the goal is connectivity, the tool is the Internet Protocol,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

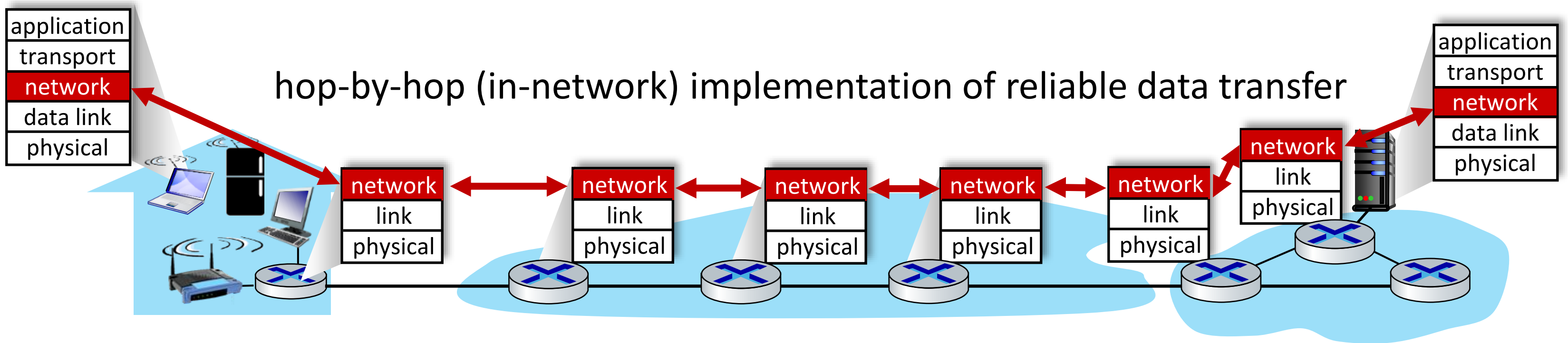
Three cornerstone beliefs:

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

The end-end argument

- some network functionality (e.g., reliable data transfer, congestion) can be implemented in network, or at network edge





Where's the intelligence?

20th century phone net:
- intelligence/computing at network switches

Internet (pre-2005)
- intelligence, computing at edge

Internet (post-2005)
- programmable network devices
- intelligence, computing, massive application-level infrastructure at edge

Thanks for listening!
Any questions?

Acknowledgment

- James F. Kurose University of Massachusetts, Amherst
- Keith W. Ross NYU and NYU Shanghai