

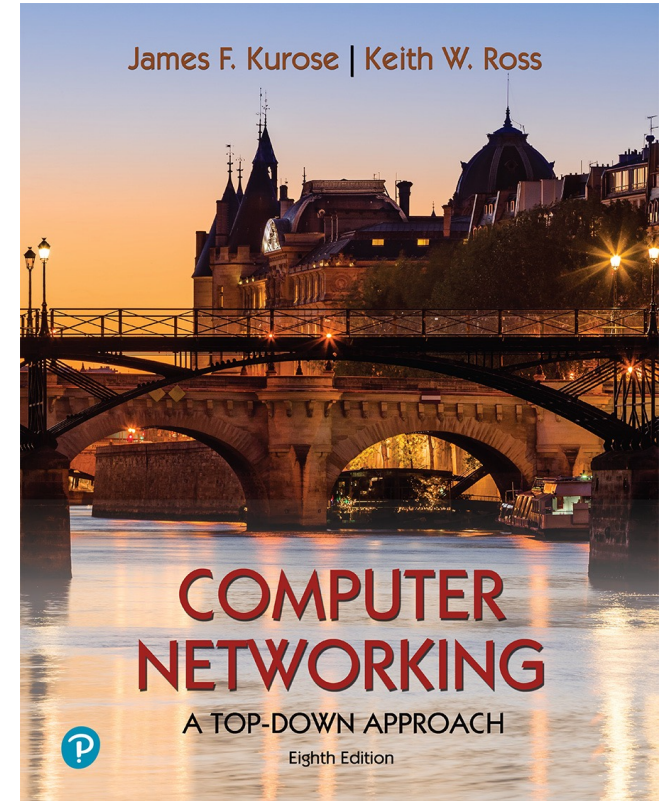
Chapter 4

Network Layer: Data Plane

Yaxiong Xie

Department of Computer Science and Engineering
University at Buffalo, SUNY

Adapted from the slides of the book's authors



*Computer Networking: A
Top-Down Approach*

8th edition

Jim Kurose, Keith Ross
Pearson, 2020

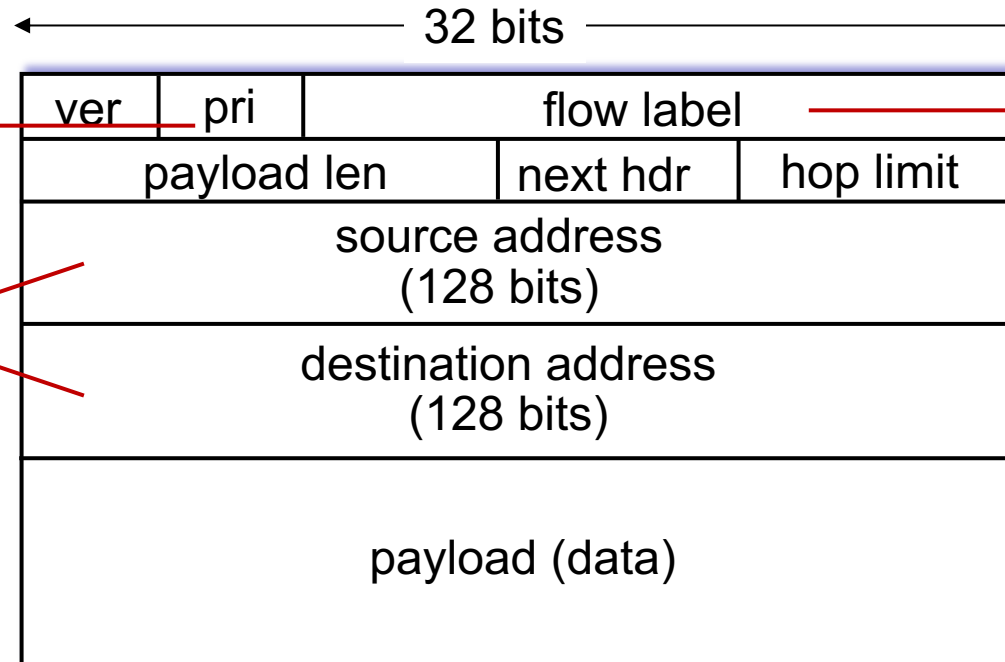
IPv6: motivation

- **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

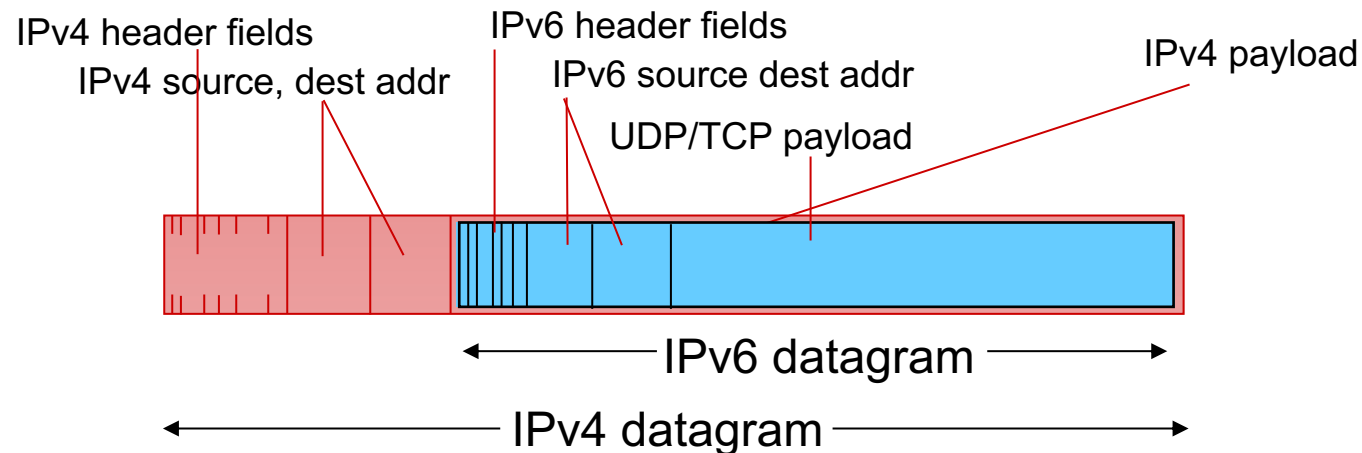
128-bit
IPv6 addresses



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

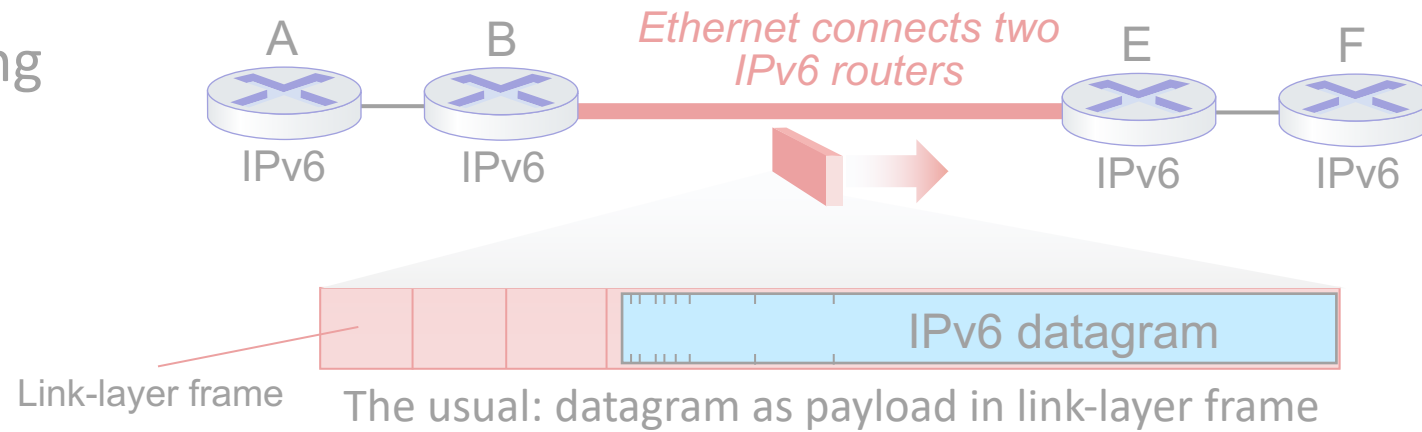
Transition from IPv4 to IPv6

- 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)

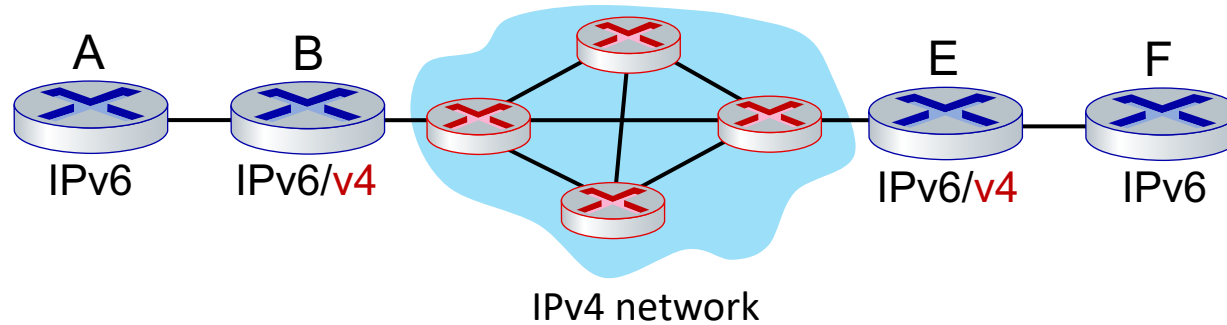


Tunneling and encapsulation

Ethernet connecting two IPv6 routers:

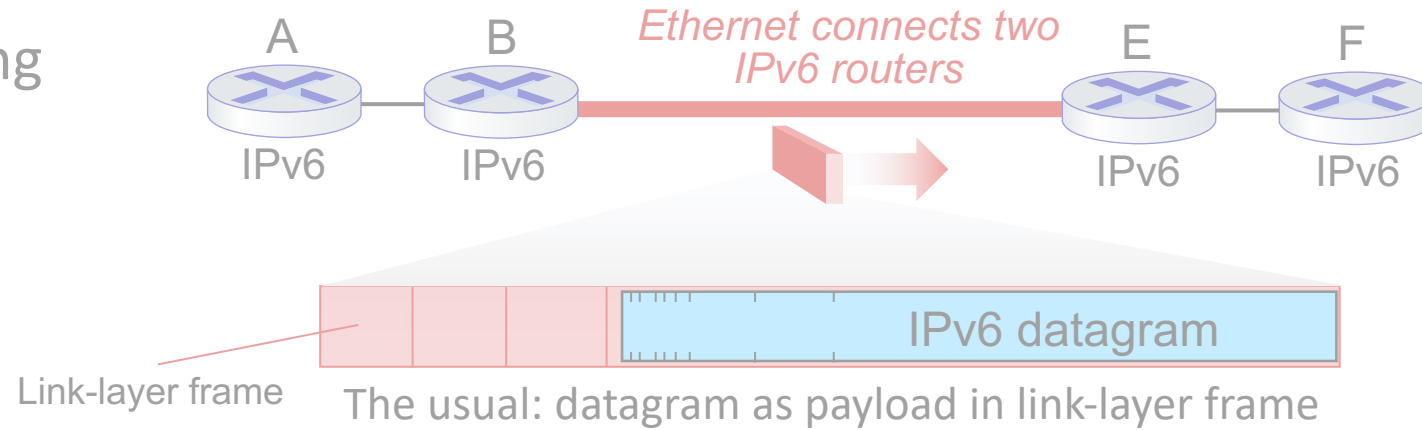


IPv4 network connecting two IPv6 routers

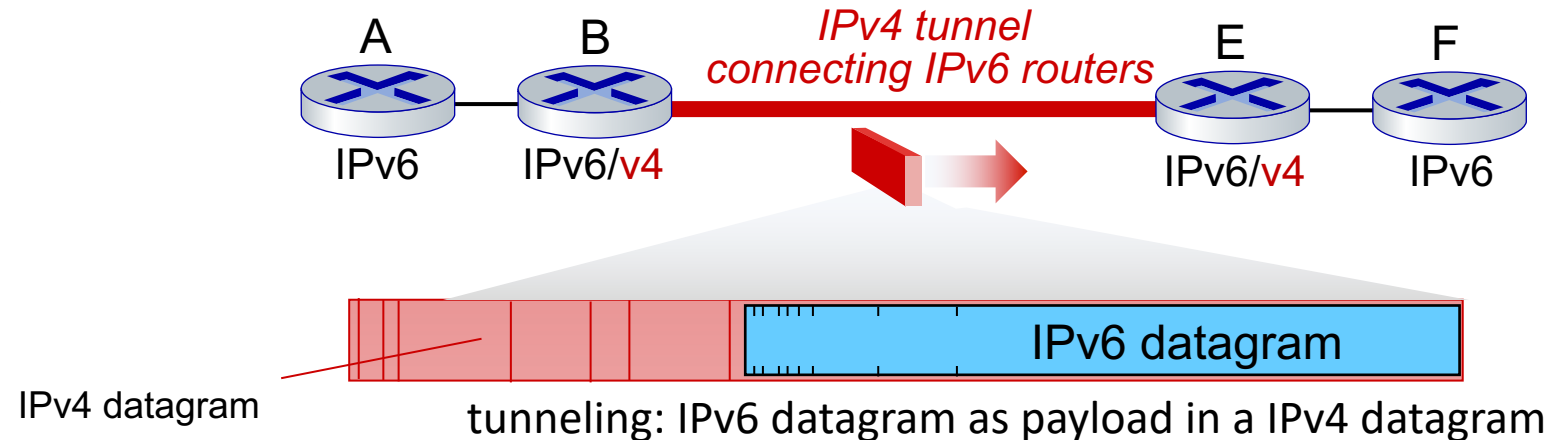


Tunneling and encapsulation

Ethernet connecting two IPv6 routers:



IPv4 tunnel connecting two IPv6 routers

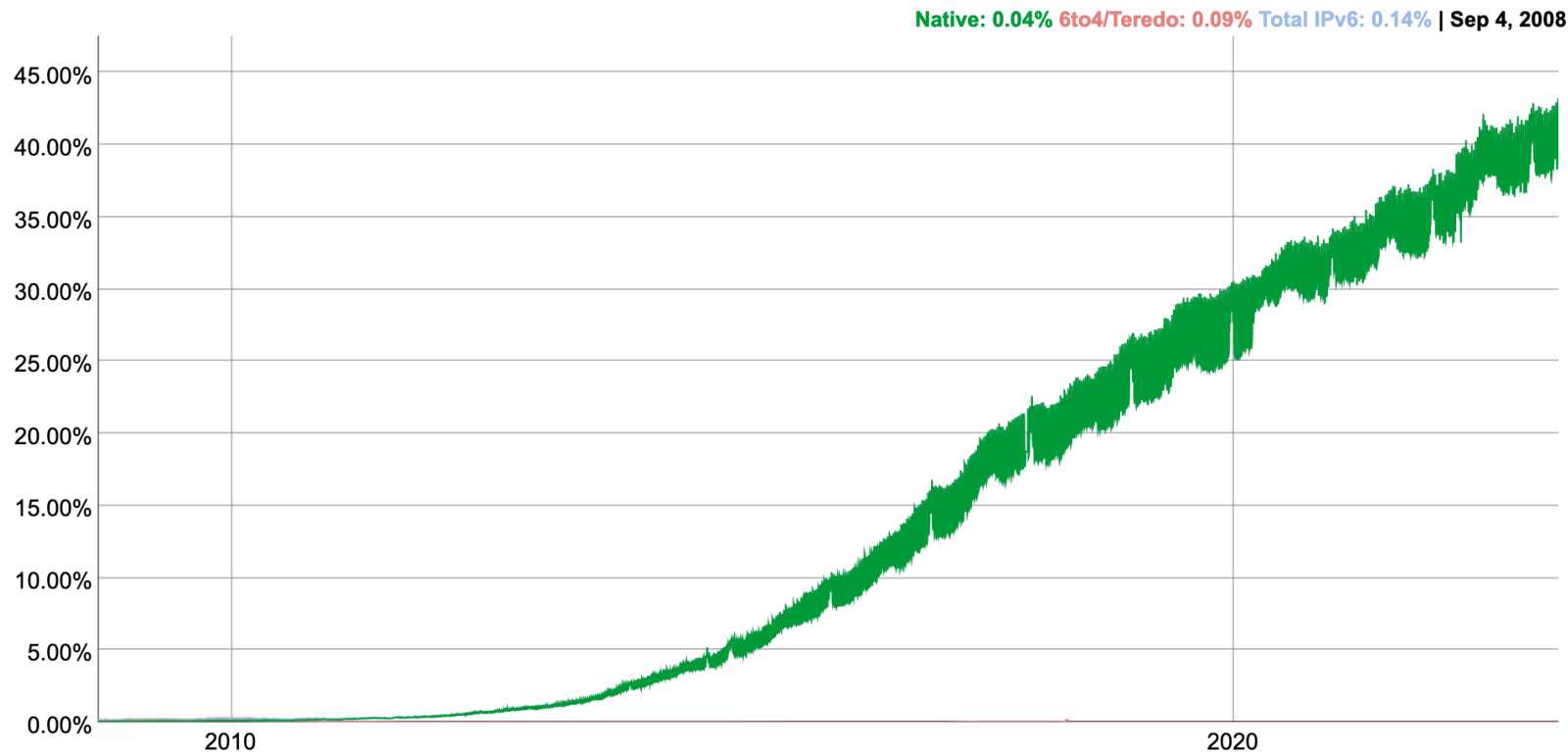


IPv6: adoption

- Google¹: ~ 40% of clients access services via IPv6 (2023)
- 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.



IPv6: adoption

- Google¹: ~ 40% of clients access services via IPv6 (2023)
- 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, ...

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

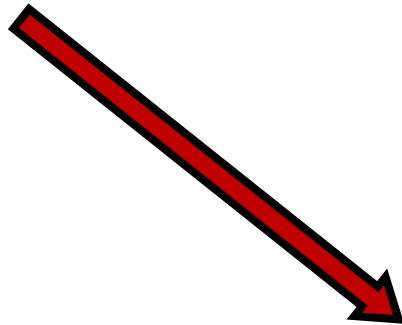
Network layer: “data plane” roadmap

- Network layer: overview
 - data plane
 - control plane
- What’s inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- IP: the Internet Protocol
 - datagram format
 - addressing
 - network address translation
 - IPv6
- Generalized Forwarding, SDN
 - Match+action
 - OpenFlow: match+action in action
- Middleboxes



Generalized Forwarding

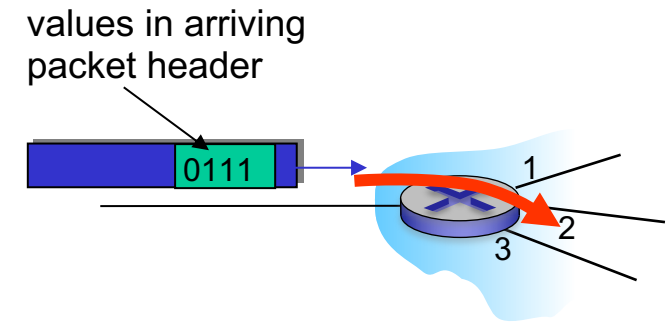
Classic IP Forwarding



Generalized Forwarding

Classic IP Forwarding: Match and Forward

IP Address Range	Forwarding Interface
200.23.16.0/23	1
200.23.18.0/23	2
200.23.20.0/23	3

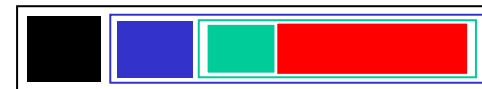
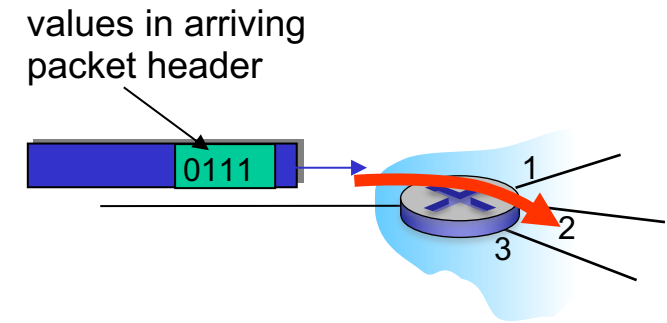


Action: Match and forward

Router: Forwarding Table

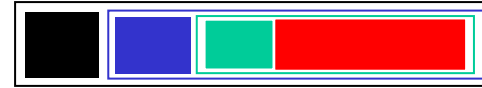
IP Address Range	Forwarding Interface
200.23.16.0/23	1
200.23.18.0/23	2
200.23.20.0/23	3

Action: Match and forward



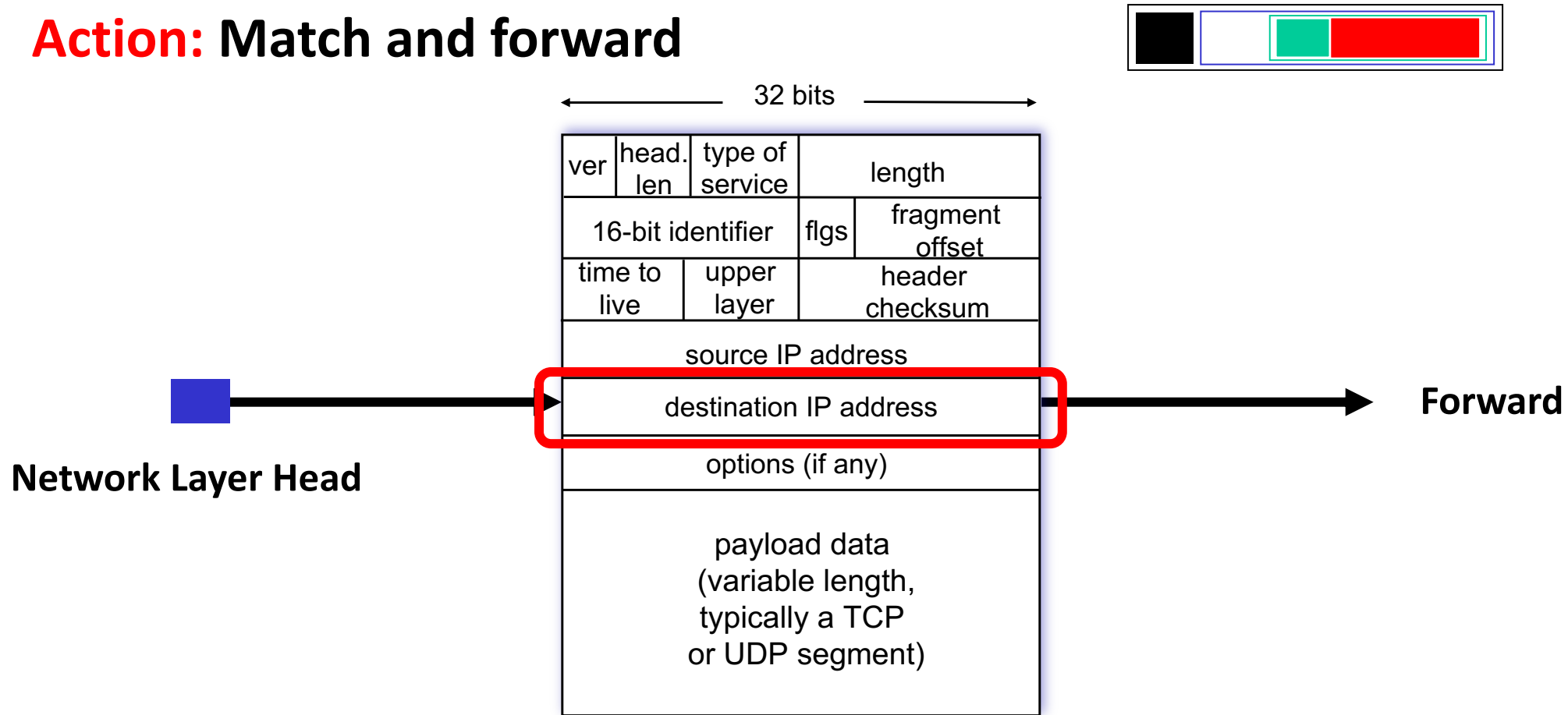
Router: Forwarding Table

Action: Match and forward



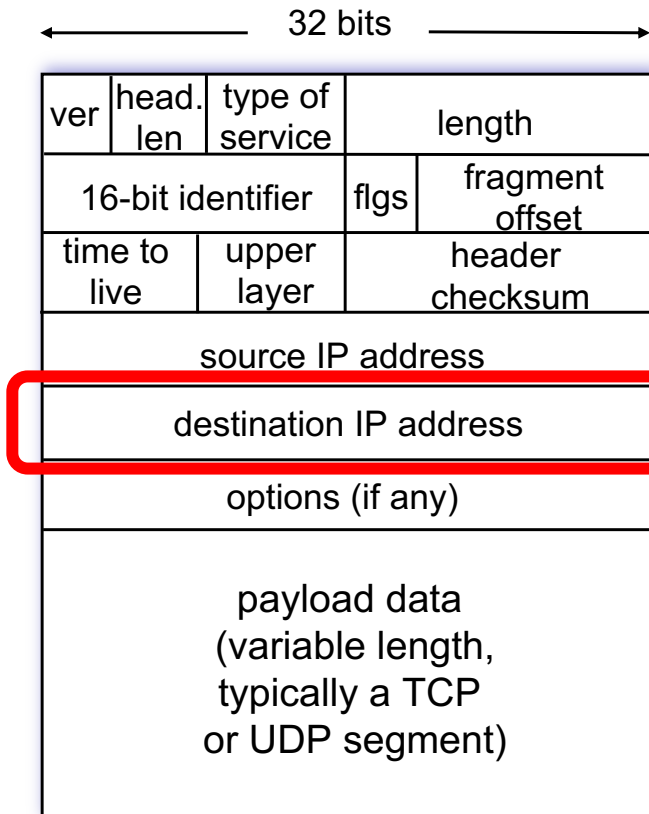
Router: Forwarding Table

Action: Match and forward



Router: Forwarding Table

Action: Match and forward

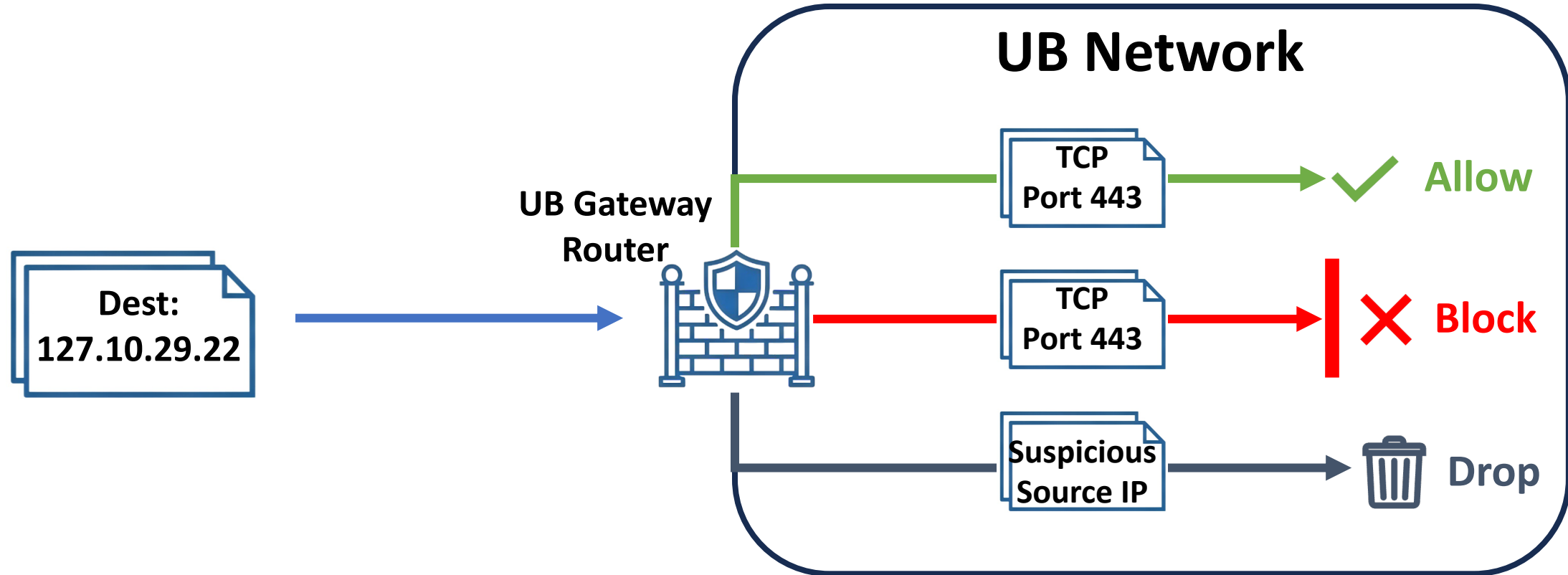


Forward

Generalized Forwarding:

Can we ask the router to perform operations
beyond simple forwarding?

Scenario 1: The Security Gatekeeper



Packets may share a destination IP, but require entirely different security actions based on source IP, protocol, and Port

Scenario 2: Application-Aware Routing

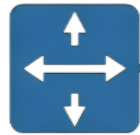
Incoming Packets

Required Network Action



The destination tells us where the packet goes, but not how urgently it should be handled. Same destination $\neq$ same performance requirement

Scenario 3: Load Balancing at Scale

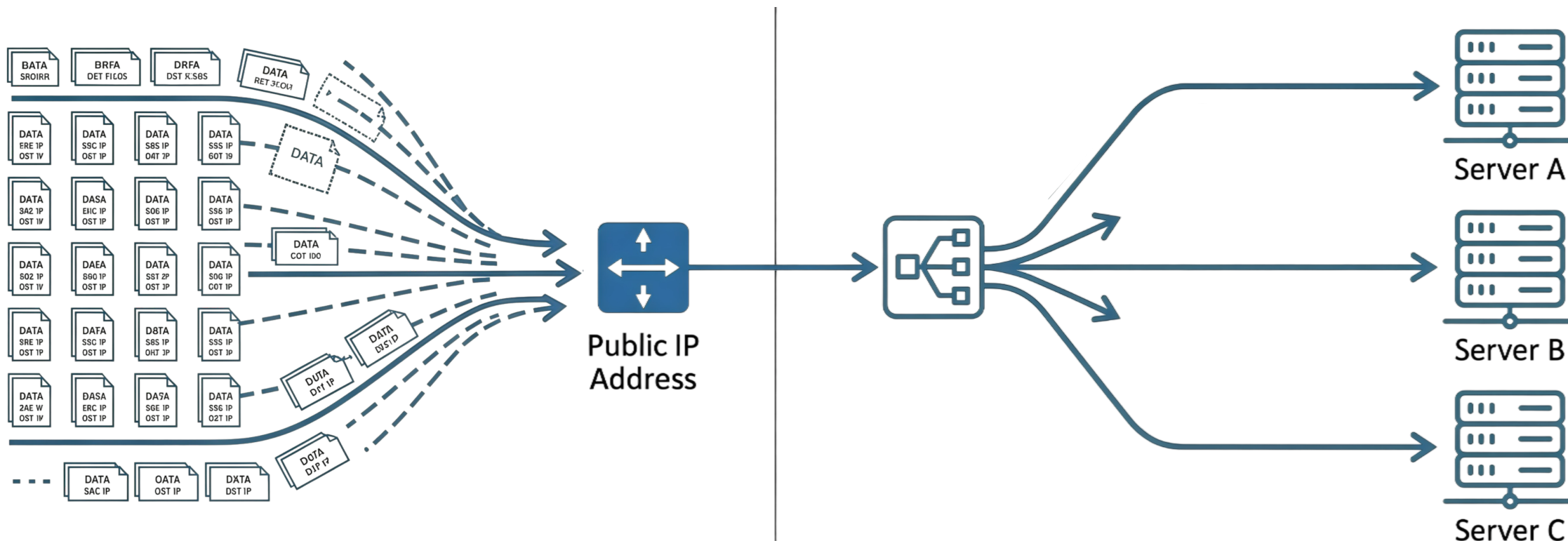


Public IP
Address

Scenario 3: Load Balancing at Scale

Huge Amount of Incoming Traffic

Load Balancing in Data Center



Even when packets goes to the same service, the network must distinguish distinct flows (Src IP/Port, Dest IP/Port, and Protocol) to balance the load

Generalized Forwarding: Match + Actions

Packet Pattern Matching

Dest IP

Dest IP + Src IP

Dest IP + Dest Port + Src IP + Src Port

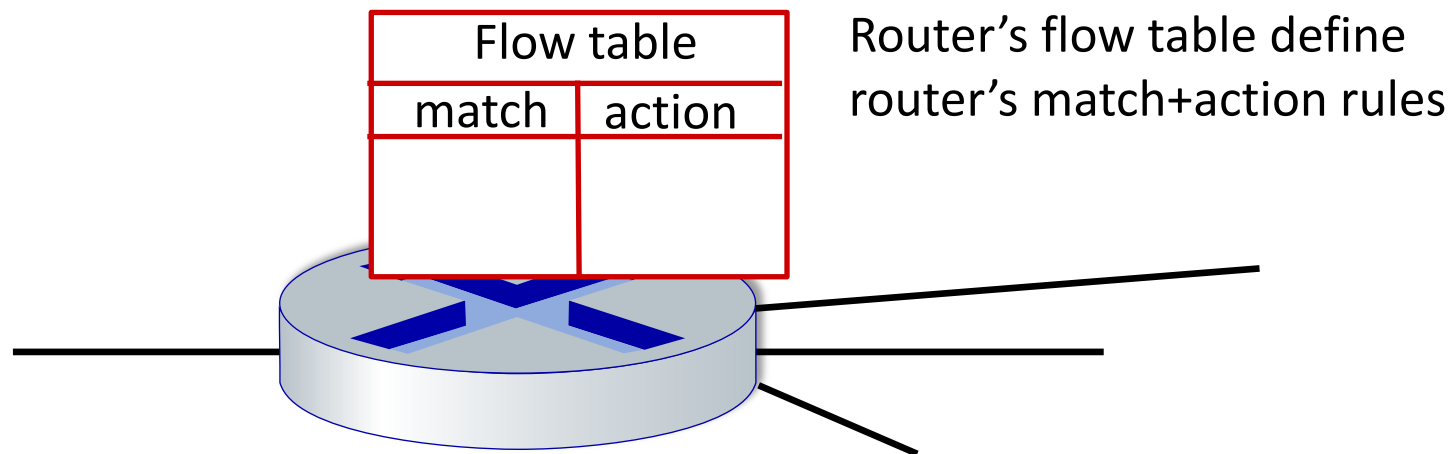


Required Network Actions

The action itself has become a generalized toolset.

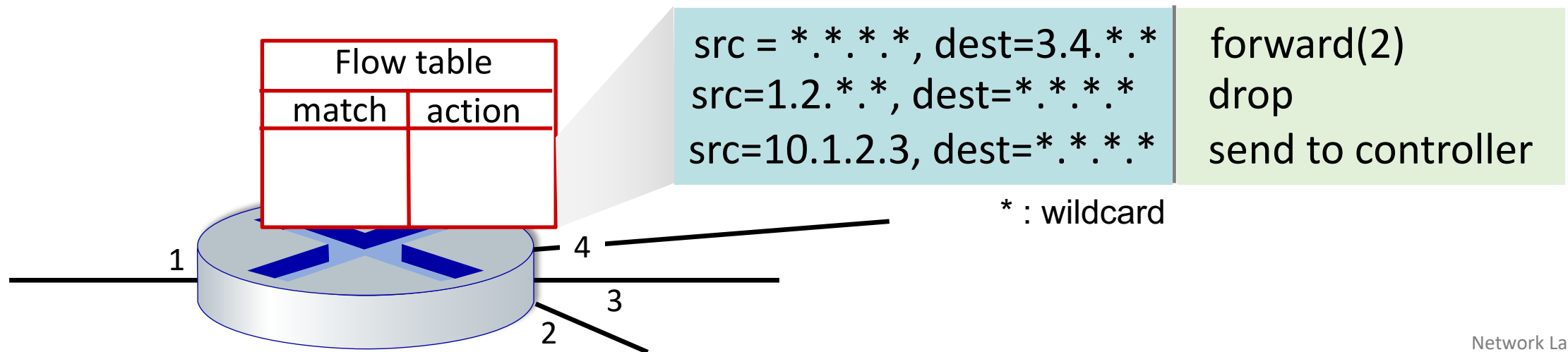
From Forwarding Table to Flow table

- **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



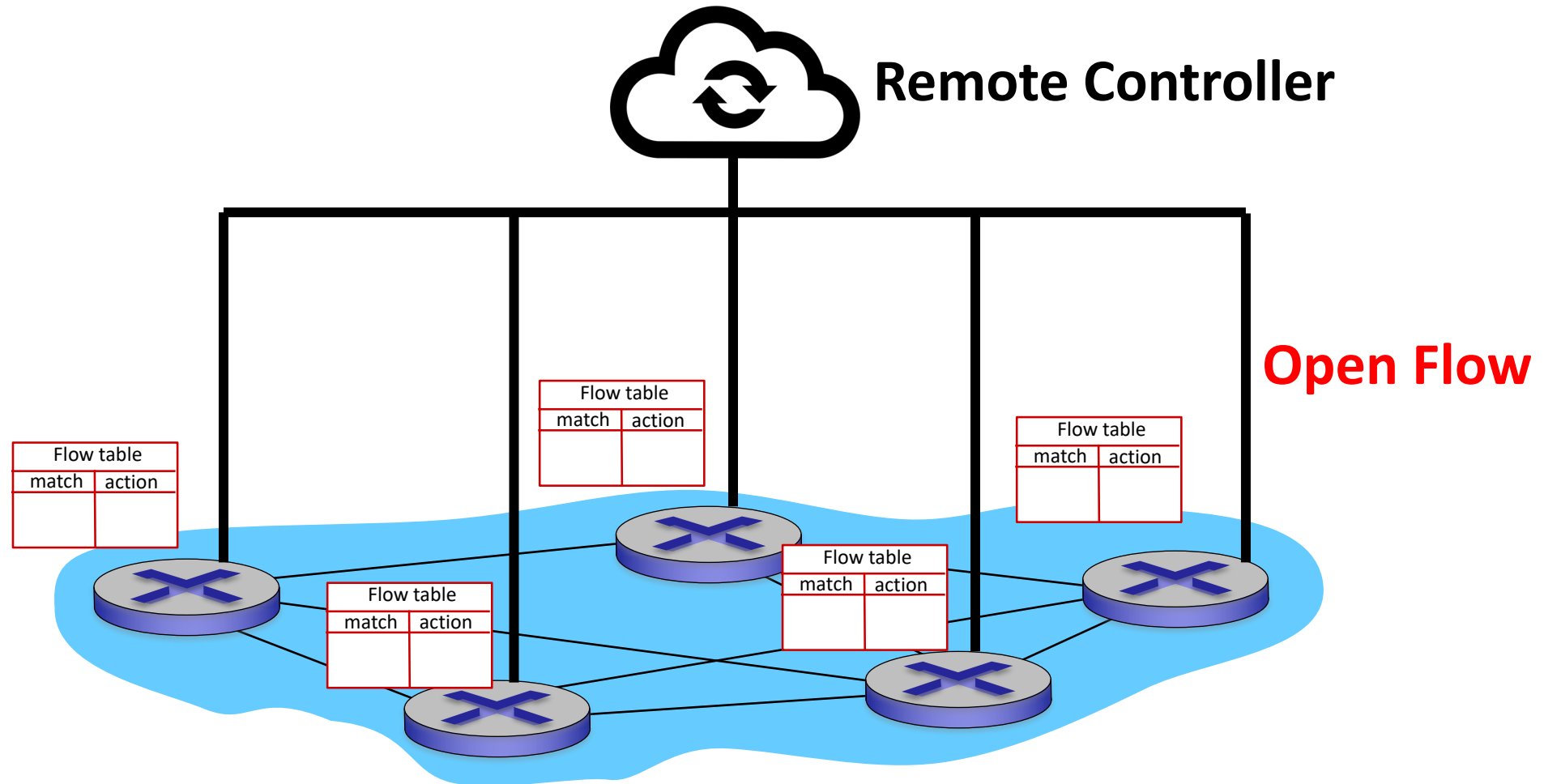
Flow table abstraction

- **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

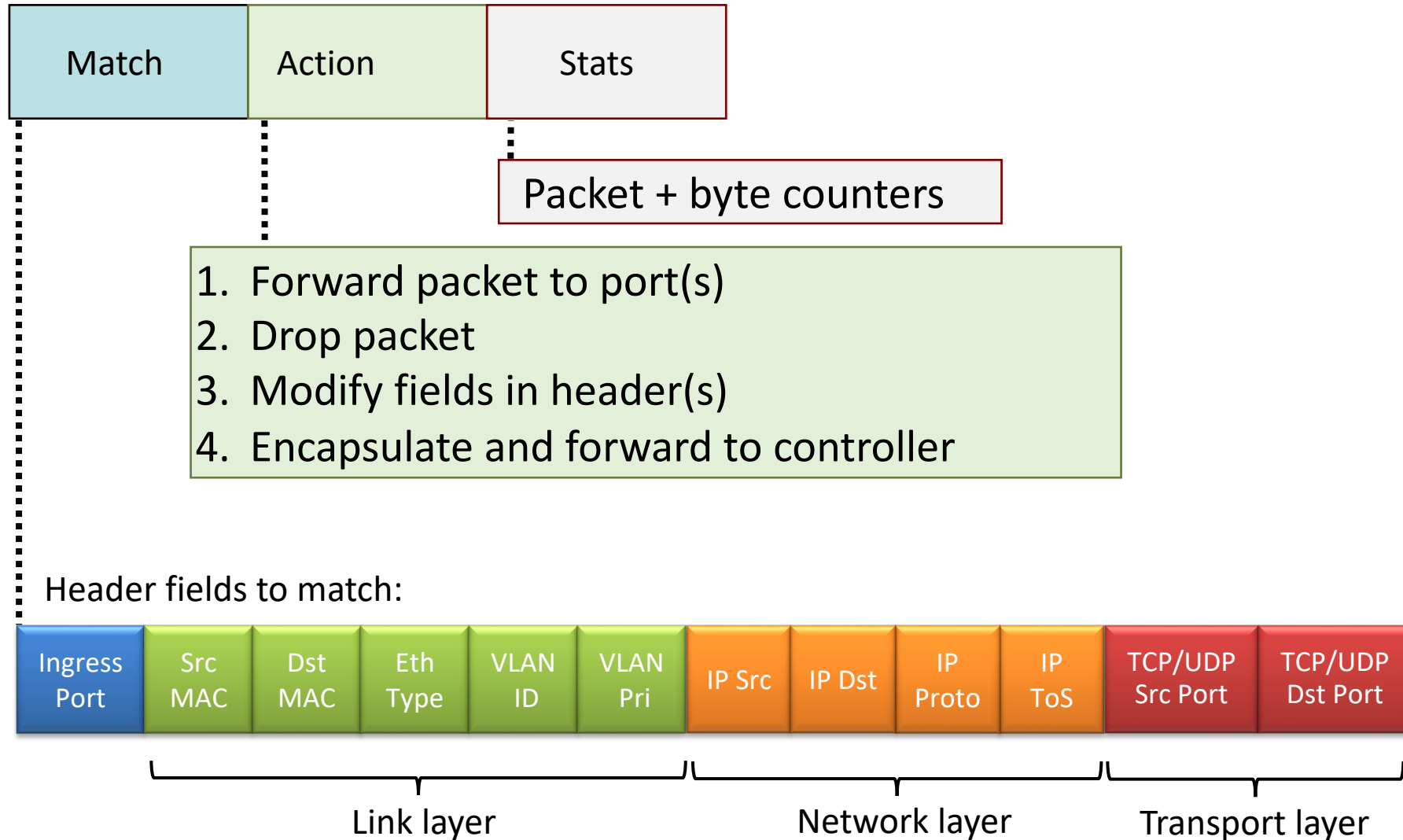


Who generates those flow tables?
How to deploy those flow table on
routers?

Open Flow: the communication protocol



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

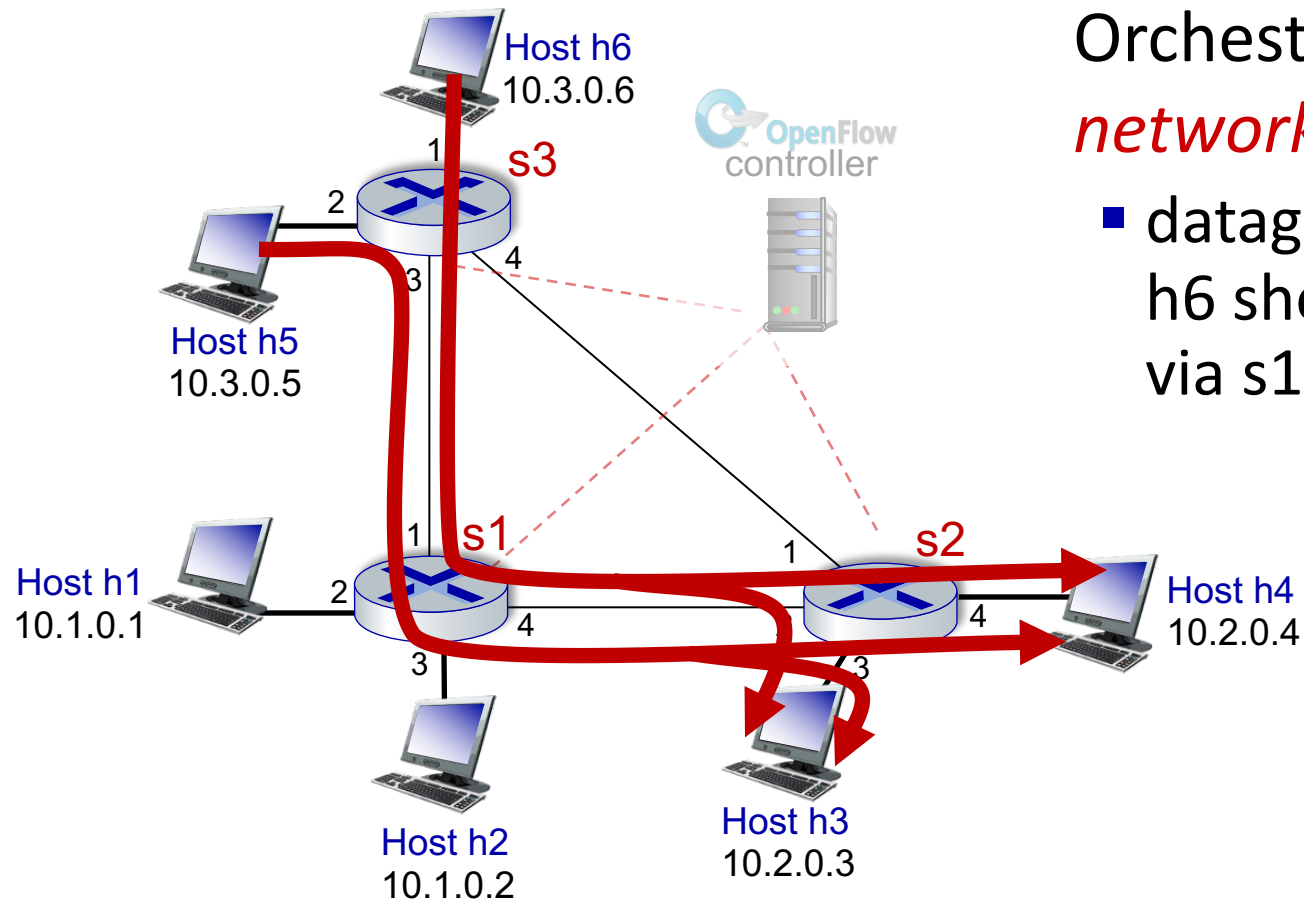
Firewall

- *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

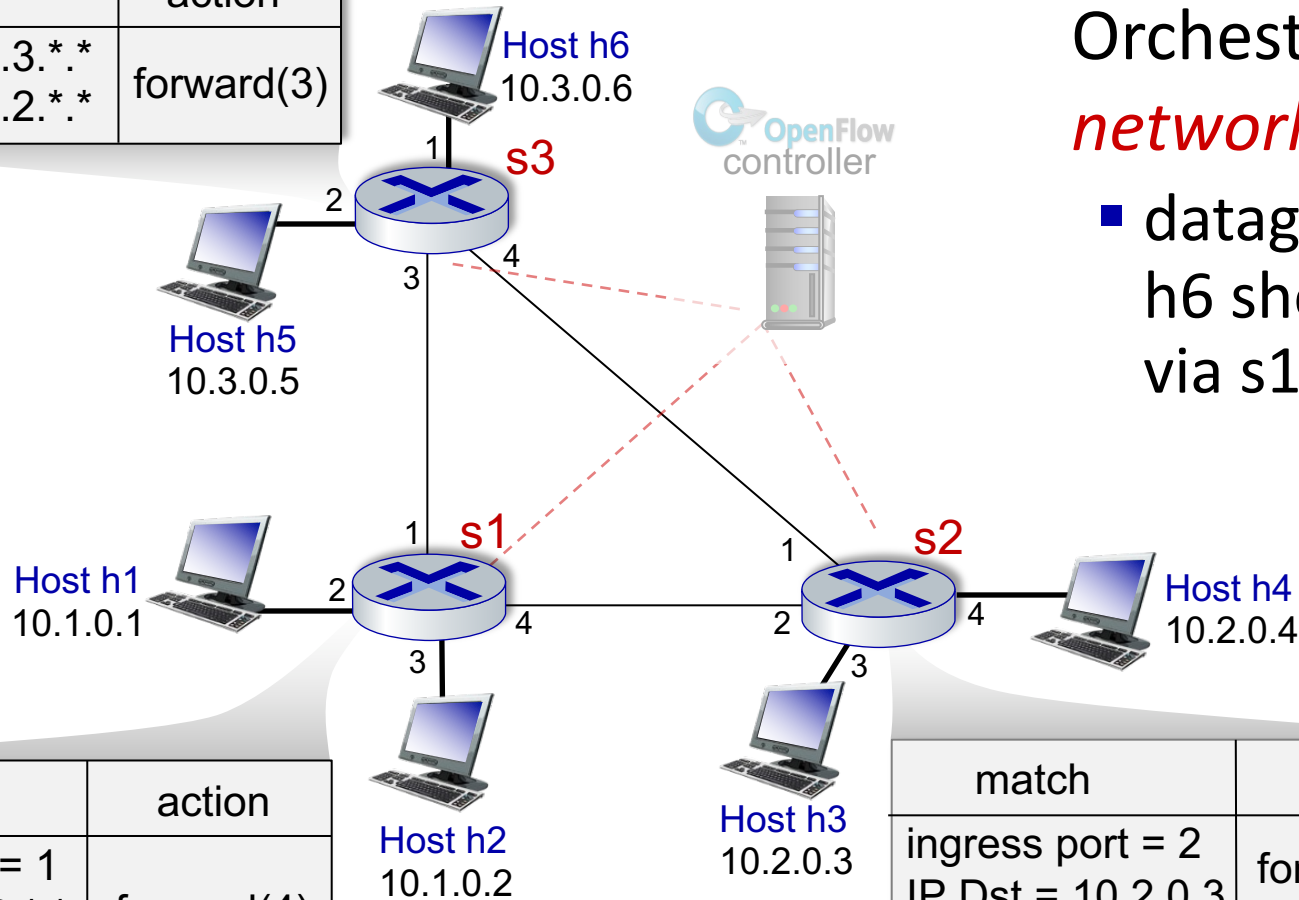


Orchestrated tables can create *network-wide* behavior, e.g.,:

- 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)

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

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

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).

Network layer: “data plane” roadmap

- Network layer: overview
- What’s inside a router
- IP: the Internet Protocol
- Generalized Forwarding
- **Middleboxes**
 - middlebox functions
 - evolution, architectural principles of the Internet



Middleboxes

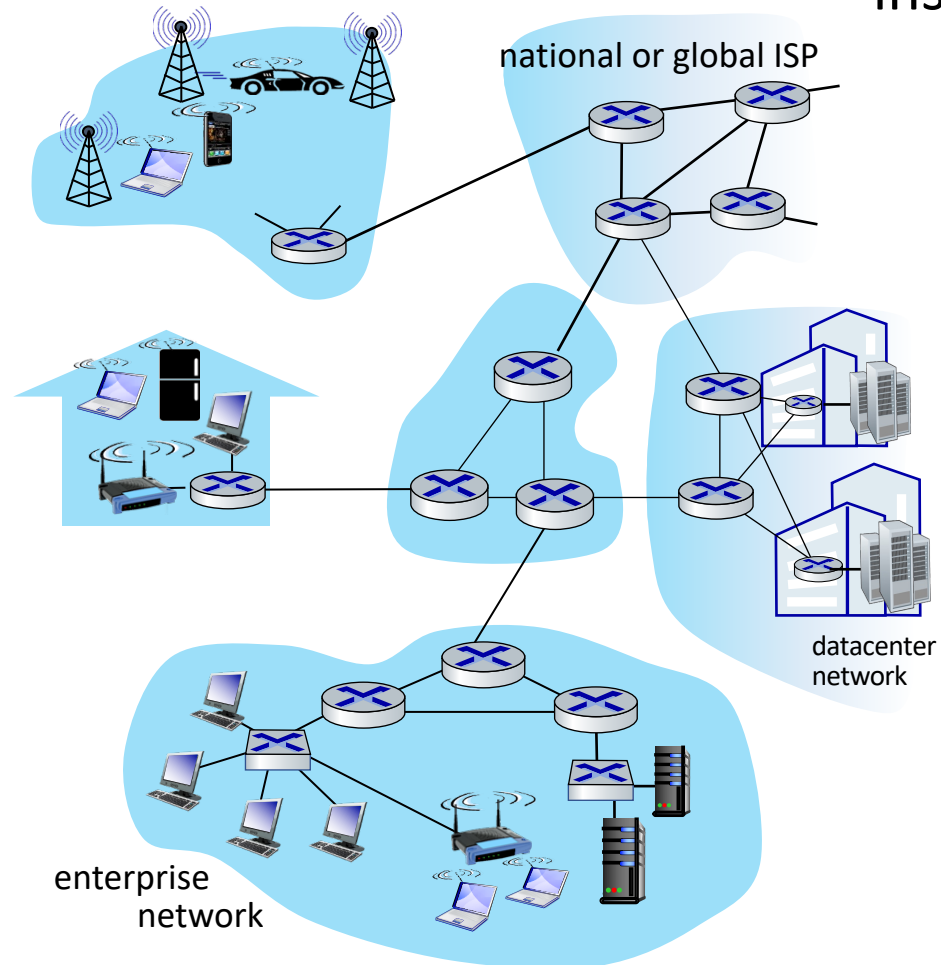
Middlebox (RFC 3234)

“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

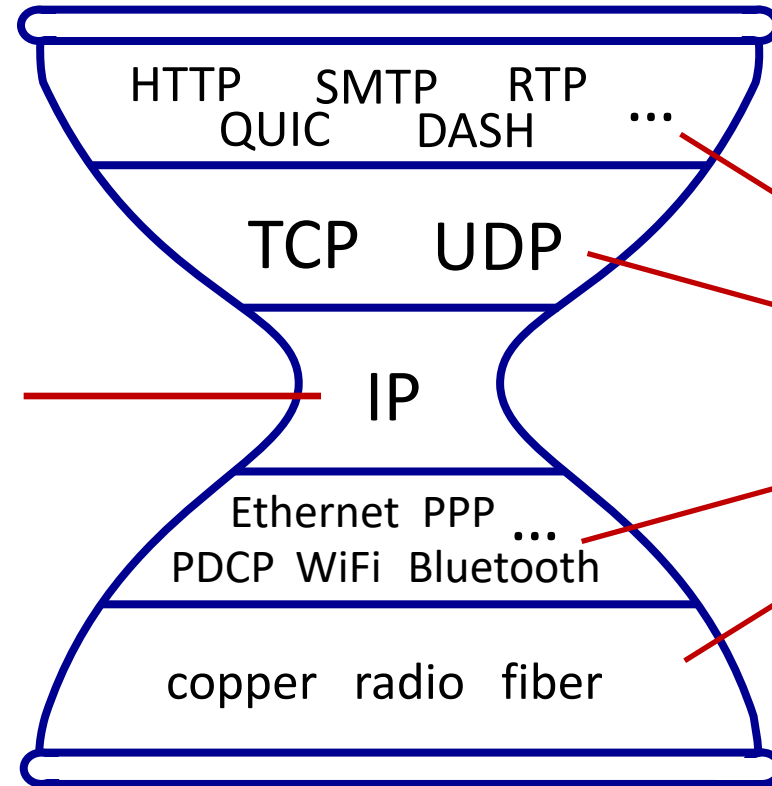
Middleboxes

- 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

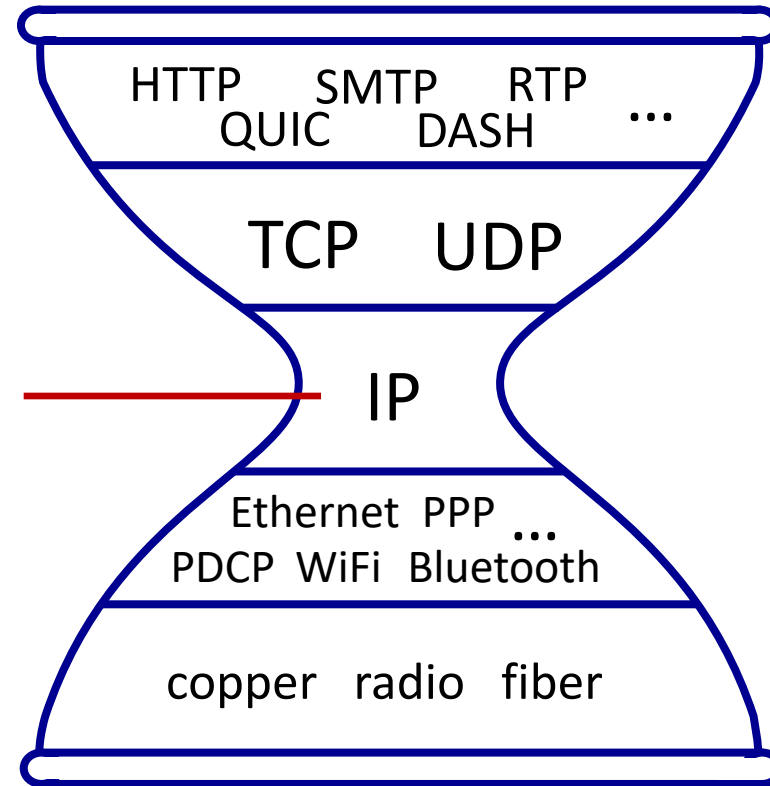


many protocols in physical, link, transport, and application layers

The IP hourglass

Internet's "thin waist":

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



Upper Layers:

Providing unified service



Lower Layers:

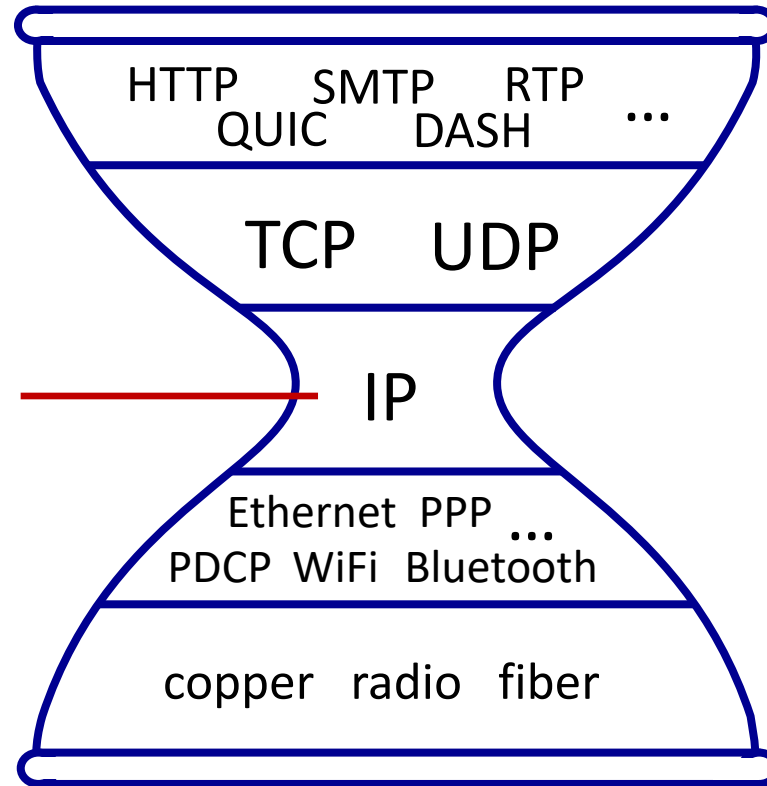
Abstracting away the differences in the underlying network technologies

IP is the common layer that connects diverse applications to diverse networks

The IP hourglass

Internet's "thin waist":

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



M types of applications

N types of transport layer protocols

O types of network layer protocols

P types of link layer protocols

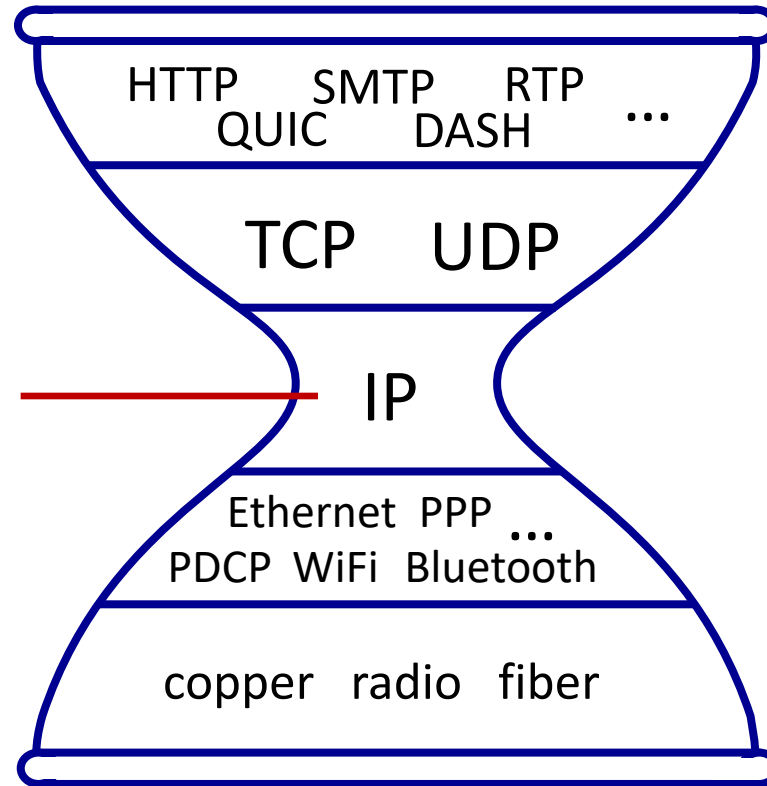
Q types of physical networks

M x N x O x P x Q

The IP hourglass

Internet's "thin waist":

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



M types of applications

N types of transport layer protocols

1 types of network layer protocols

P types of link layer protocols

Q types of physical networks

M x N and P x Q

Architectural Principles of the Internet

RFC 1958

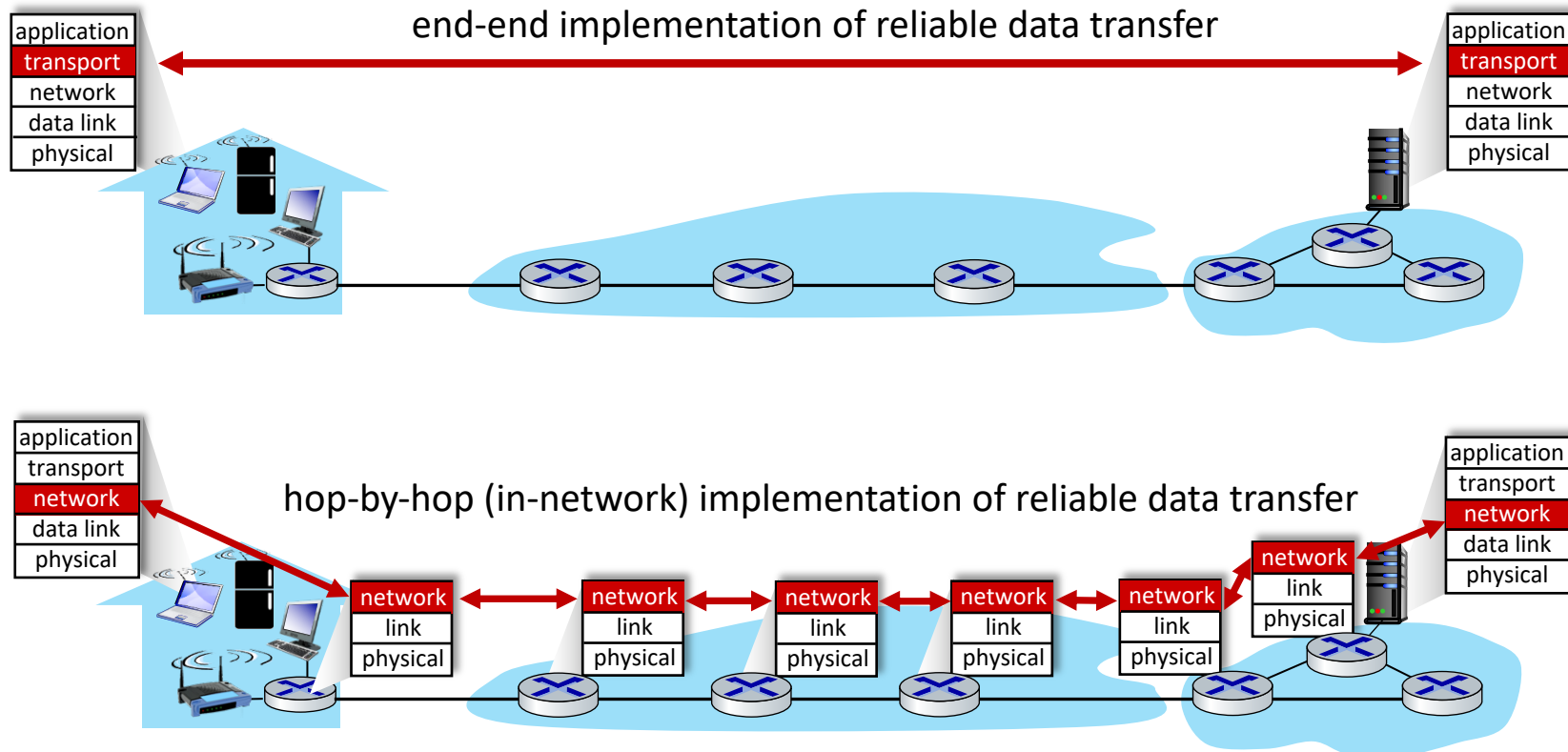
“Many members of the Internet community would argue that there is no architecture, but only a tradition, which was not written down for the first 25 years (or at least not by the IAB). However, in very general terms, the community believes that **the goal is connectivity, the tool is the Internet Protocol,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



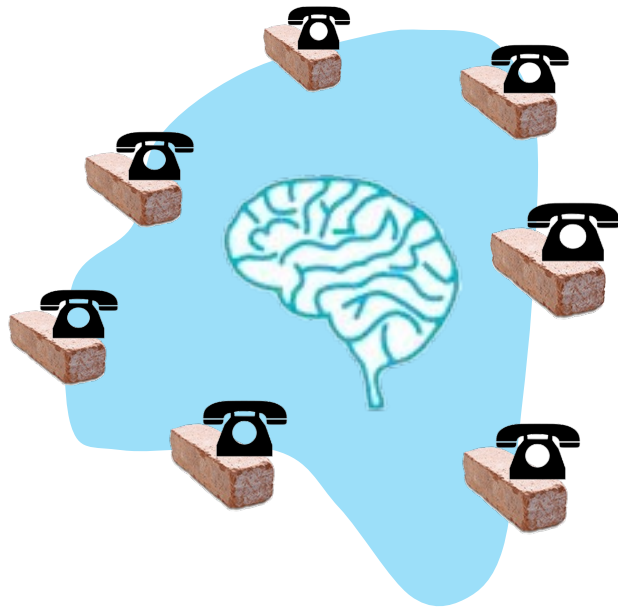
The end-end argument

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

“The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)

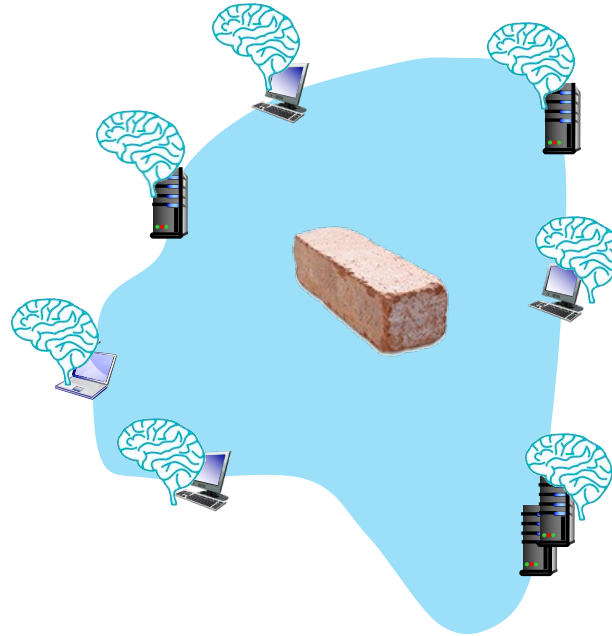
We call this line of reasoning against low-level function implementation the “end-to-end argument.”

Where's the intelligence?



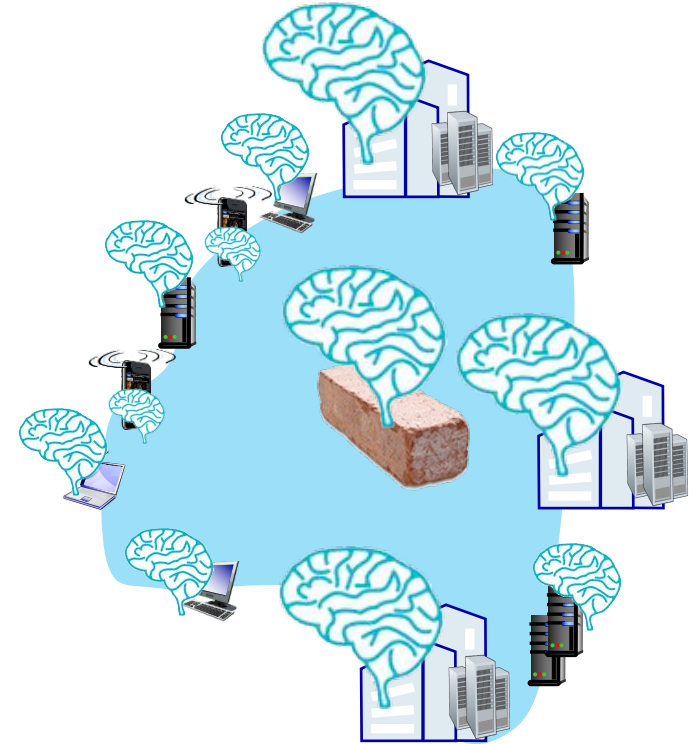
20th century phone net:

- 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

Generalized Forwarding: Match + Actions

Packet Pattern Matching

Dest IP

Dest IP + Src IP

Dest IP + Dest Port + Src IP + Src Port

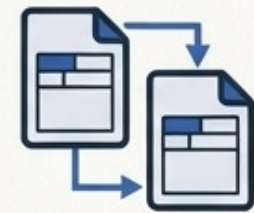


Required Network Actions



Drop

(e.g., suspicious traffic)



Mirror

(e.g., send DNS packets to a monitoring system)



Modify

(e.g., rewrite headers before forwarding)



Prioritize

(e.g., latency-sensitive traffic)

The action itself has become a generalized toolset.

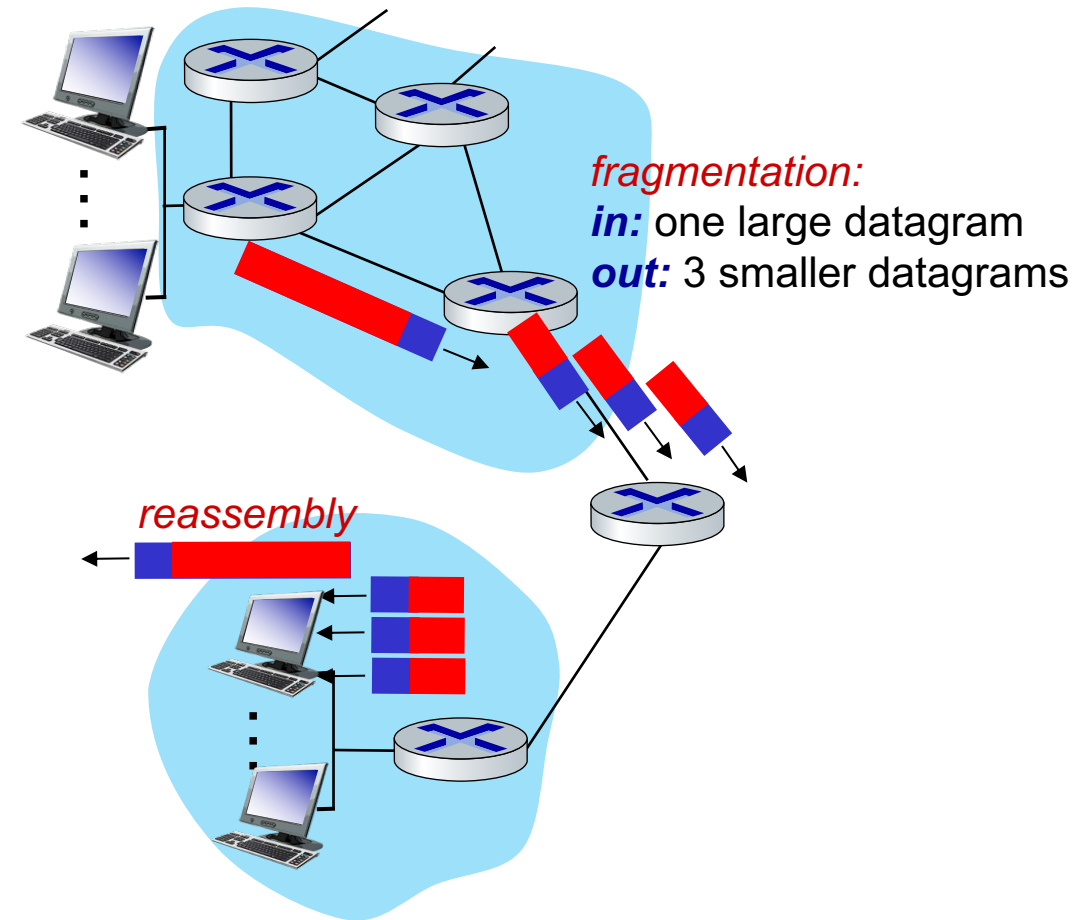
Chapter 4: done!

- Network layer: overview
- What's inside a router
- IP: the Internet Protocol
- Generalized Forwarding, SDN
- Middleboxes



IP fragmentation/reassembly

- network links have MTU (max. transfer size) - largest possible link-level frame
 - different link types, different MTUs
- large IP datagram divided (“fragmented”) within net
 - one datagram becomes several datagrams
 - “reassembled” only at *destination*
 - IP header bits used to identify, order related fragments



IP fragmentation/reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes

	length =4000	ID =x	fragflag =0	offset =0	
--	-----------------	----------	----------------	--------------	--

*one large datagram becomes
several smaller datagrams*

1480 bytes in
data field

offset =
 $1480/8$

	length =1500	ID =x	fragflag =1	offset =0	
--	-----------------	----------	----------------	--------------	--

	length =1500	ID =x	fragflag =1	offset =185	
--	-----------------	----------	----------------	----------------	--

	length =1040	ID =x	fragflag =0	offset =370	
--	-----------------	----------	----------------	----------------	--

Chapter 4: done!

