National University of Computer & Emerging Sciences

CS 3001 - COMPUTER NETWORKS

Lecture 19

Chapter 4

8th April, 2025

Nauman Moazzam Hayat

nauman.moazzam@lhr.nu.edu.pk

Office Hours: 11:30 am till 01:00 pm (Every Tuesday & Thursday)

Chapter 4
Network Layer: Data Plane

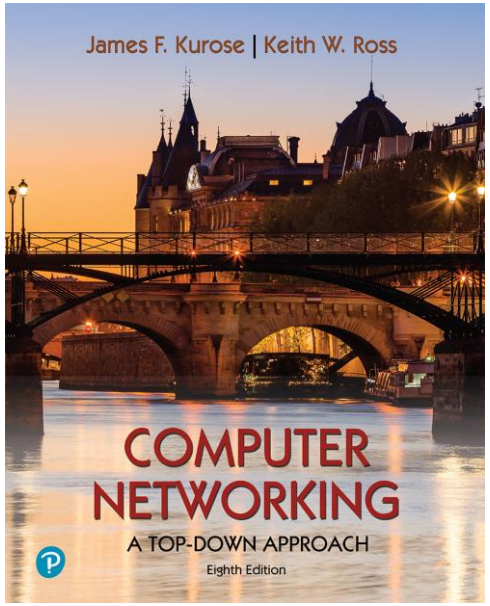
A note on the use of these PowerPoint slides:

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



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

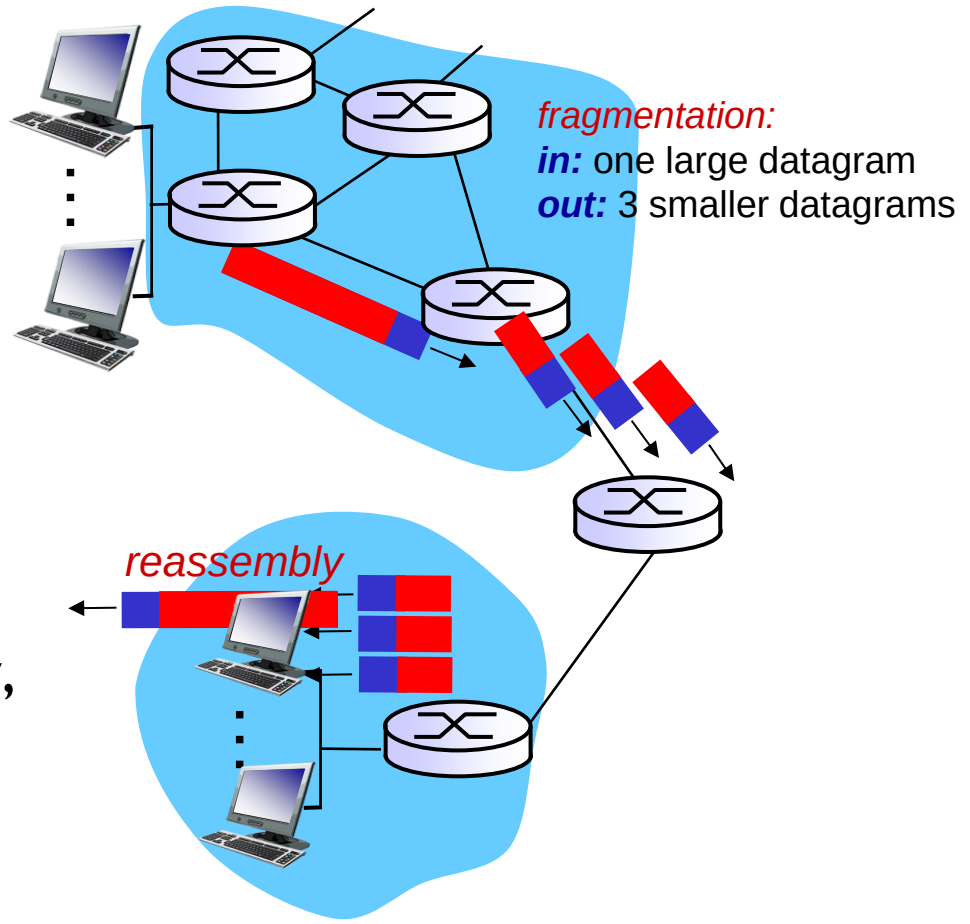
Network layer: "data plane" roadmap

- 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

IPv4 fragmentation, reassembly

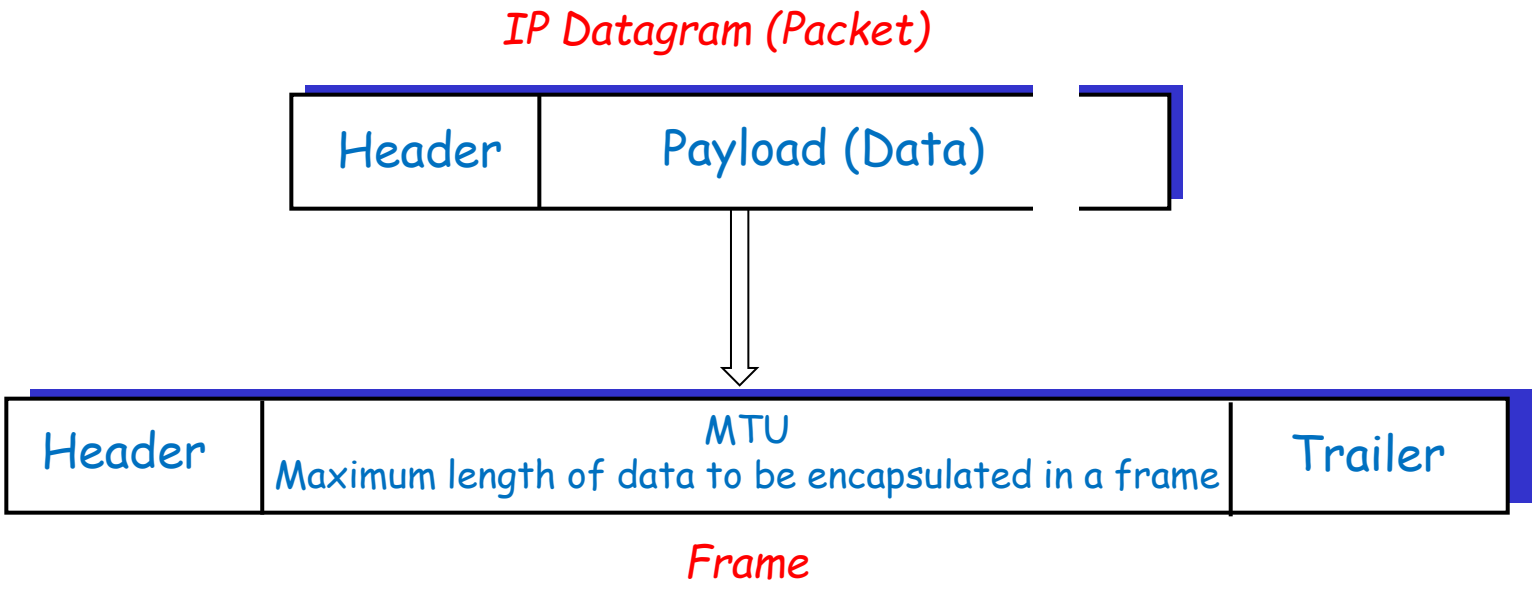
- network links have MTU (max. transfer size) - largest possible link-level frame
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - IP header bits used to identify, order related fragments
  - The network (i.e. routers) treat fragments as independent units / packets i.e. independent datagrams



IPv4 Fragmentation

- IP packet may travel over different networks (LANs and WANs)
- A router de-capsulate an IP packet from the frame it receives, process it, and encapsulate it in another frame
- Frame size and format varies depend on the data link protocol used by the physical network through which the frame is traveling
- MTU (Maximum Transmission Unit) is the maximum size of the data field (payload) in the frame
- If Packet size > MTU, Need for Fragmentation

Maximum Transmission Unit (MTU)



Protocol	MTU (Bytes)
Ethernet	1500
Token Ring (4 Mbps)	4464
Token Ring (16 Mbps)	17914
FDDI	4352
X.25	576
PPP	296
Wireless	2312

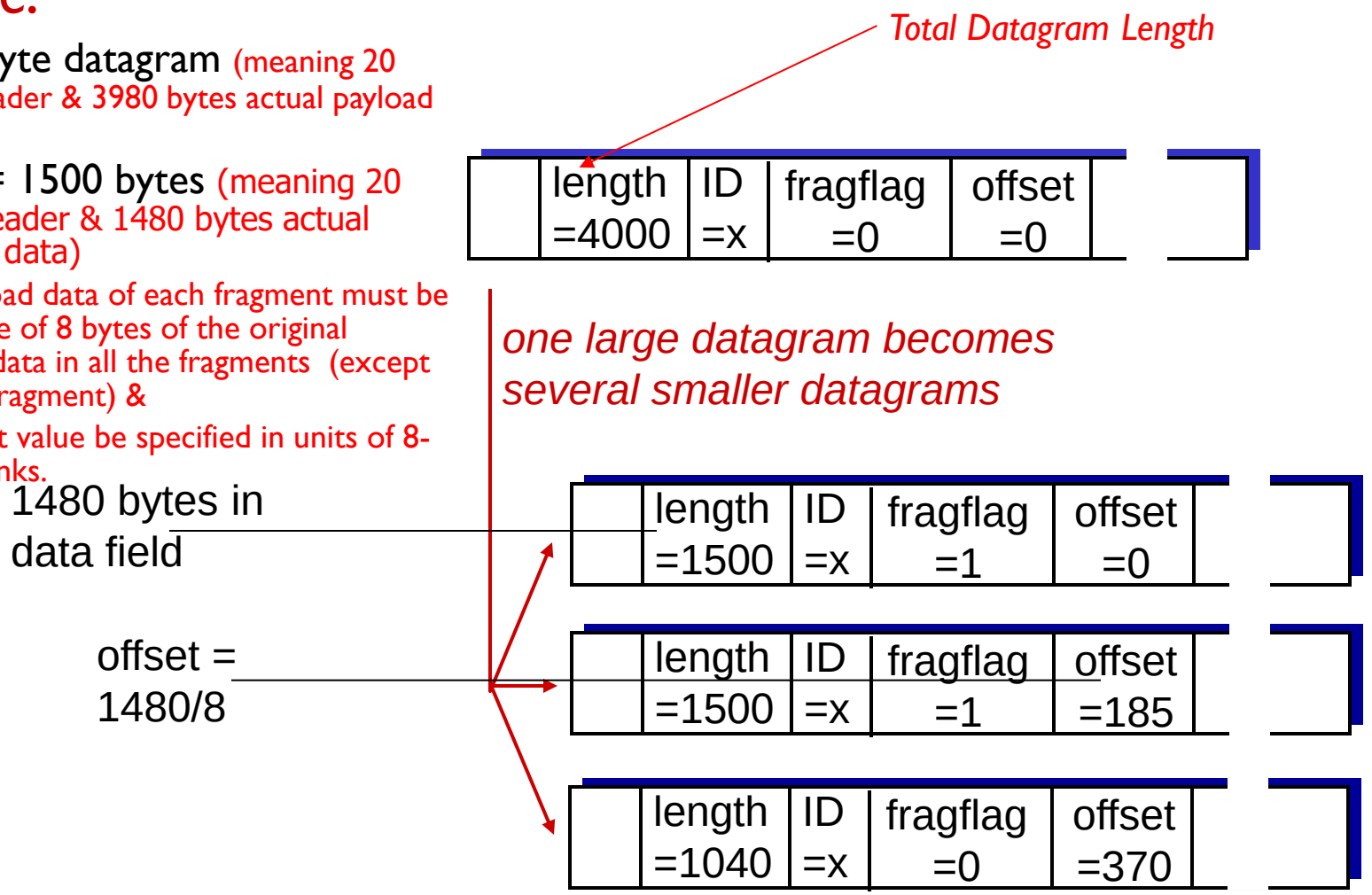
IPv4 Fragmentation (cont'd)

- Each fragment has its own header (most of fields are copied, some will change, including the total length, the Flags and the fragmentation offset fields)
- A fragmented datagram may itself be fragmented if it encounters a network with smaller MTU
- A packet can be fragmented by a source host or by any router in the path. Re-assembly of the packet must be done at the destination host because those fragments become independent packets and may travel different routes

IPv4 fragmentation, reassembly

example:

- 4000 byte datagram (meaning 20 bytes header & 3980 bytes actual payload data)
- MTU = 1500 bytes (meaning 20 bytes header & 1480 bytes actual payload data)
- the payload data of each fragment must be a multiple of 8 bytes of the original payload data in all the fragments (except the last fragment) &
- the offset value be specified in units of 8-byte chunks.



IPv4 Fragmentation (example cont'd)

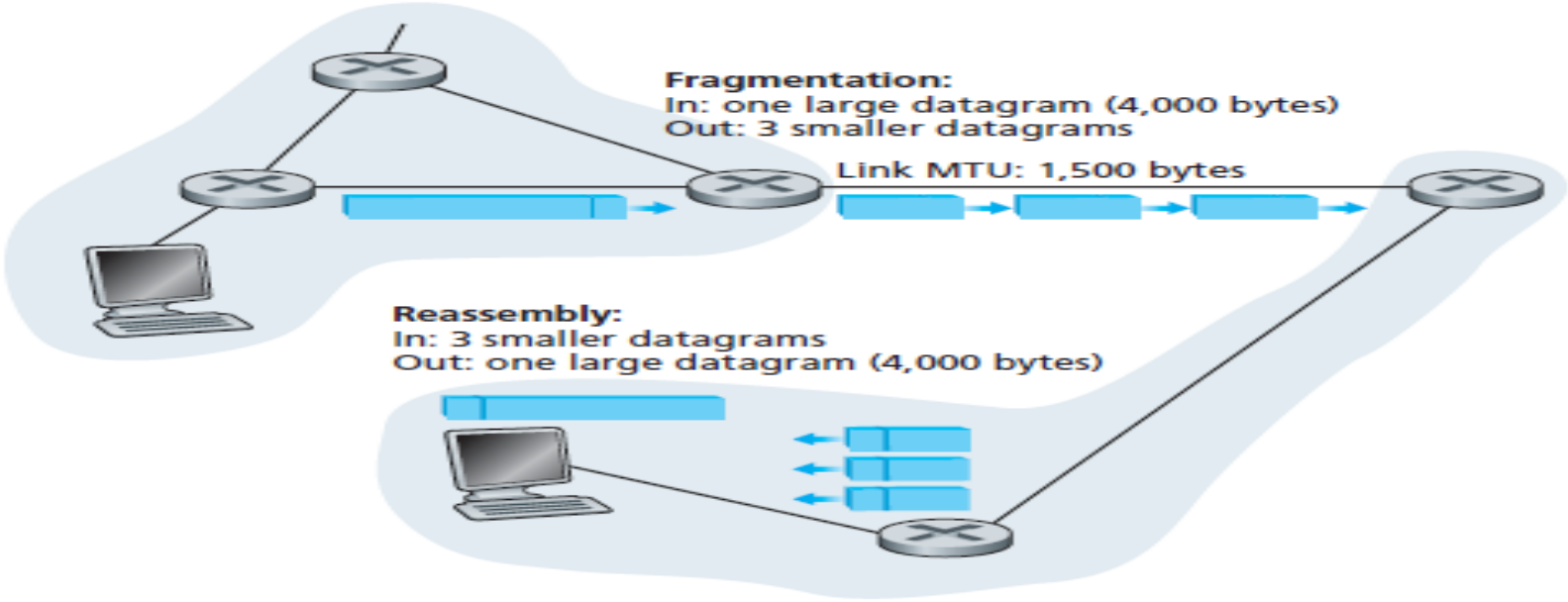


Table 4.2 IP fragments

Fragment	Bytes	ID	Offset	Flag
1st fragment	1,480 bytes in the data field of the IP datagram	identification = 777	offset = 0 (meaning the data should be inserted beginning at byte 0)	flag = 1 (meaning there is more)
2nd fragment	1,480 bytes of data	identification = 777	offset = 185 (meaning the data should be inserted beginning at byte 1,480. Note that 185 · 8 = 1,480)	flag = 1 (meaning there is more)
3rd fragment	1,020 bytes (= 3,980–1,480–1,480 of data	identification = 777	offset = 370 (meaning the data should be inserted beginning at byte 2,960. Note that 370 · 8 = 2,960)	flag = 0 (meaning this is the last fragment)

Why Fragmentation at Layer 3 (IPv4), when TCP is already ensuring MSS based on Path MTU at Layer 4?

TCP handles the Path MTU Discovery (PMTUD) to ensure the Maximum Segment Size (MSS) doesn't exceed the path's maximum transmission unit, but this process isn't foolproof. Routers along the path can have varying capabilities, and sometimes ICMP messages (crucial for PMTUD) can be blocked or lost, leading to an incorrect Path MTU. When that happens, fragmentation at Layer 3 (IP level) becomes the safety net, ensuring that the data still reaches its destination, even if it means breaking down larger packets.

Fragmentation at Layer 3 is the fallback in case the mechanisms at Layer 4 encounter unexpected issues. It's the assurance that the data keeps flowing, even if not in the most efficient manner.

Network layer: "data plane" roadmap

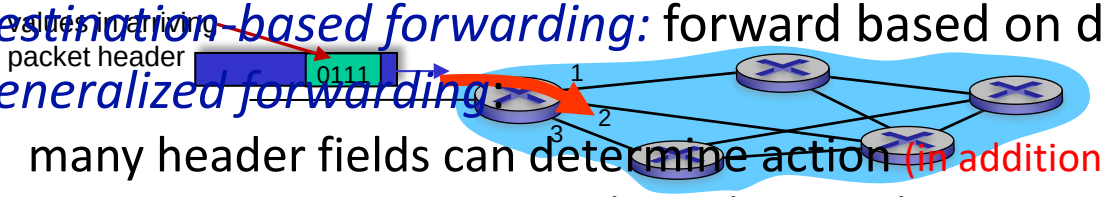
- Network layer: overview
  - data plane
  - control plane
- What's inside a router
  - input ports, switching, output ports
  - buffer management, scheduling
- IP: the Internet Protocol
  - datagram format
  - addressing
  - network address translation
  - IPv6

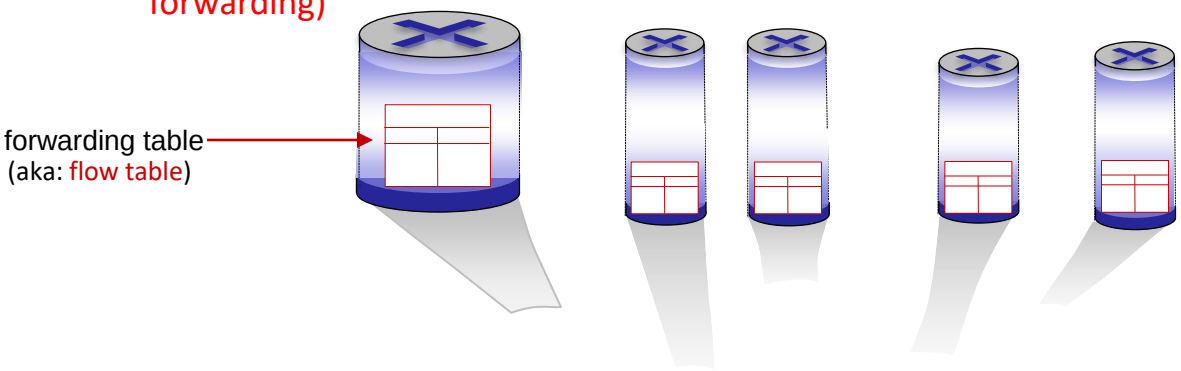
- Generalized Forwarding, SDN
  - Match+action
  - OpenFlow: match+action in action
- Middleboxes

Generalized forwarding: match plus action

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

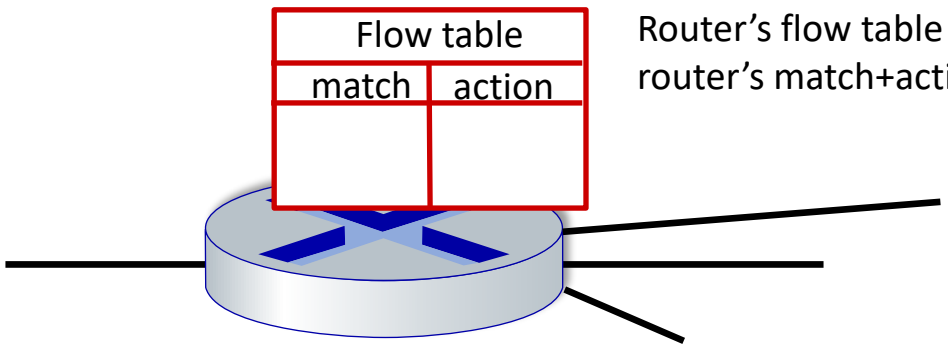
- "match plus action" abstraction: match bits in arriving packet, take action
  - *destination-based forwarding:* forward based on dest. IP address
  - *generalized forwarding:*
    - many header fields can determine action (in addition to only dest. IP)
    - many action possible: drop/copy/modify/log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





Flow table abstraction

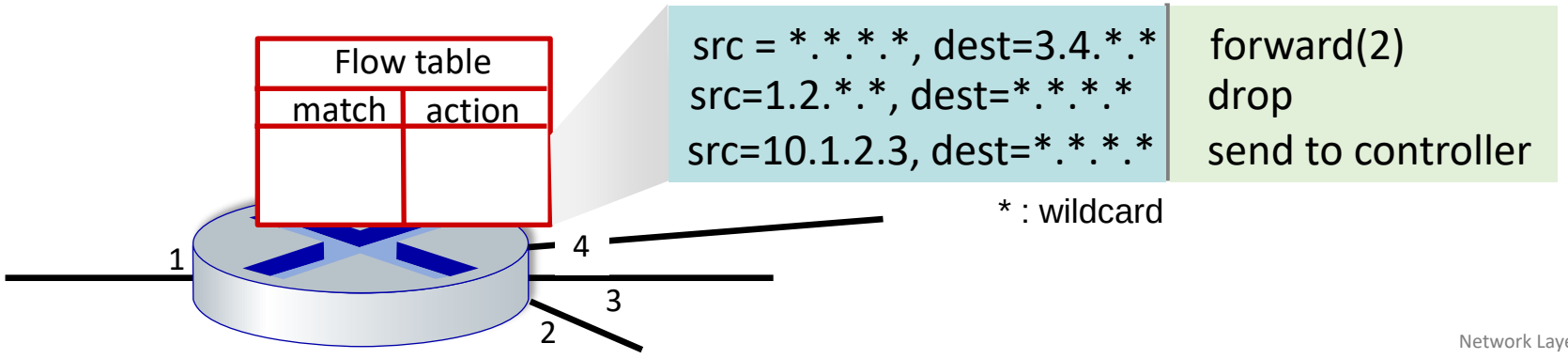
- flow: defined by header field values (in link-, network-, transport-layer fields)
- Open flow device can easily perform as a router (layer 3 device) or a switch (layer 2 device), thus a new term "*packet switch*" is being used (a terminology that is gaining widespread adoption in SDN literature)
- 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



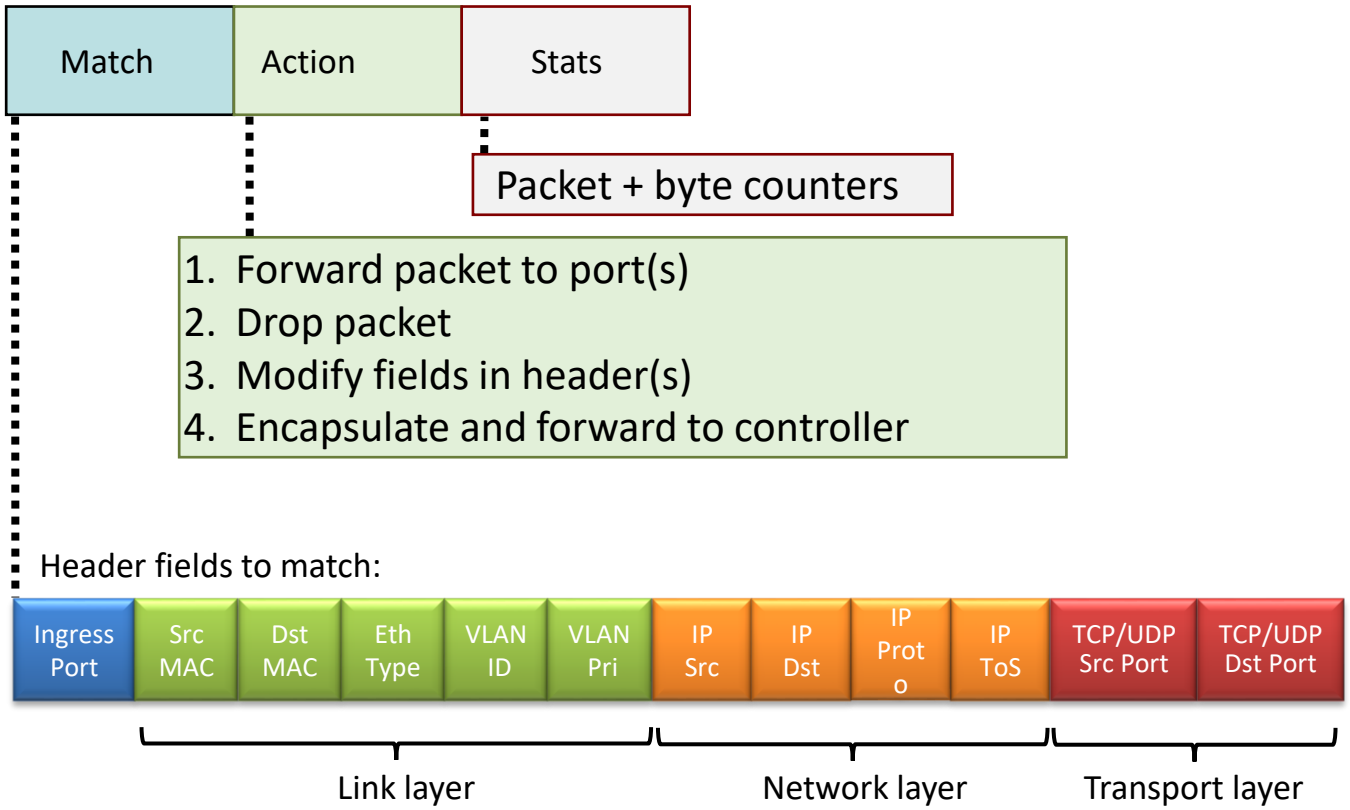
Router's flow table define router's match+action rules

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



OpenFlow: flow table entries (11 header entries in Open Flow 1.0, more in newer specifications of Open Flow)



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 (into a device called packet switch)

Router

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

Switch

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

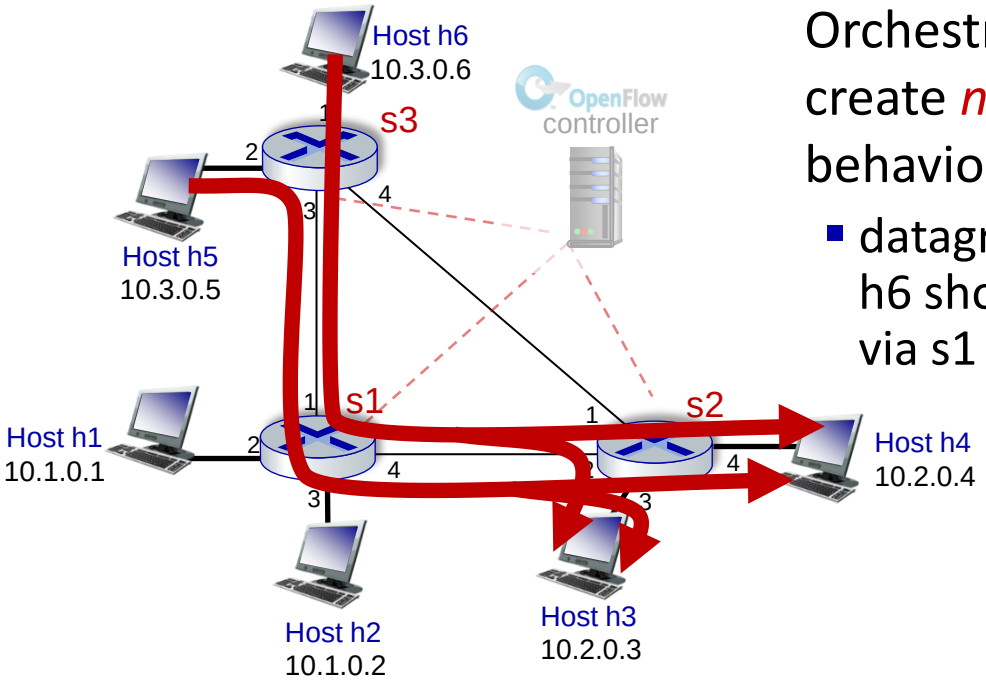
Firewall

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

NAT

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

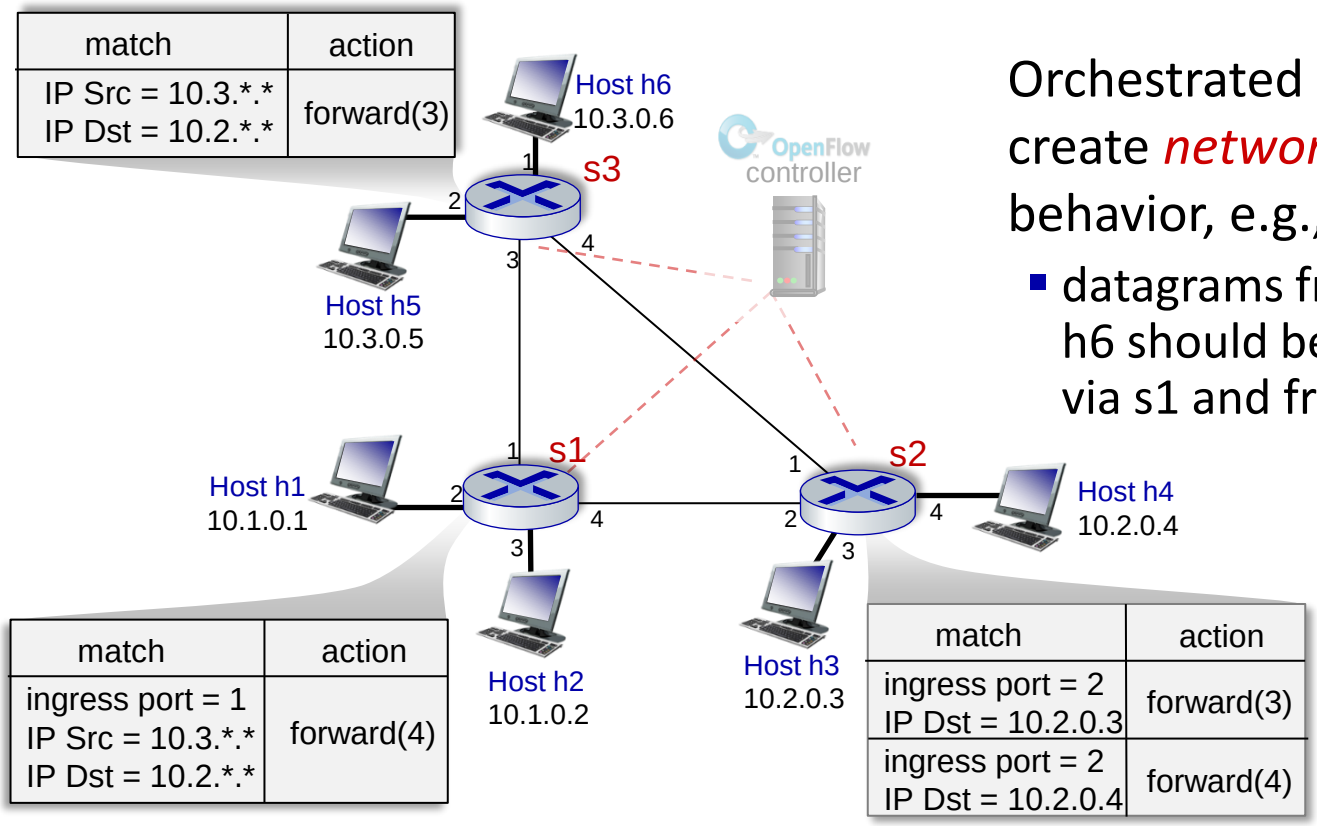
OpenFlow example



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



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" n*etwork-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

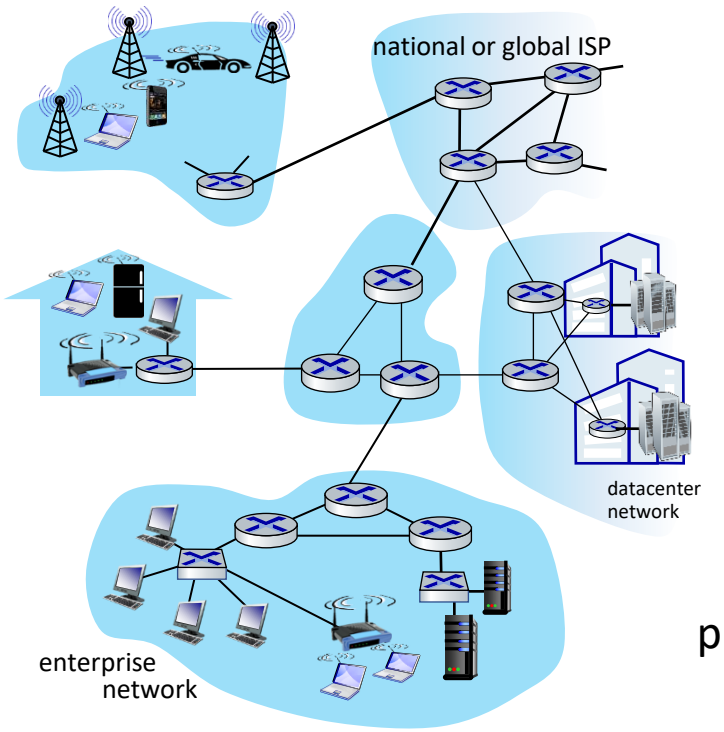
Middlebox (RFC 3234)

"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 (Intrusion Detection Systems): corporate, institutional, service providers, ISPs

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

Caches: service provider, mobile, CDNs

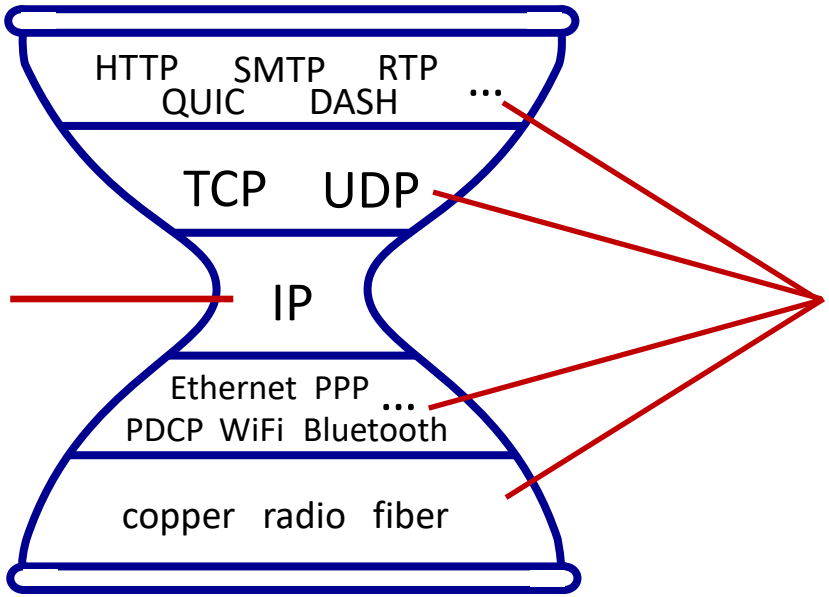
Middleboxes

- 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 Researchers are exploring the use of commodity hardware (networking, computing, and storage) with specialized software built on top of a common software stack to implement these services (Network translation, Security Services & Performance Enhancement.) This approach has become known as **network function virtualization (NFV)**. An alternate approach that has also been explored is to outsource middlebox functionality to the cloud.

The IP hourglass (initially)

Internet's "thin waist":

- *one* network layer protocol: IP (also called spanning layer)
- *must* be implemented by every (billions) of Internet-connected devices
- This narrow waist has played a critical role in the phenomenal growth of the Internet

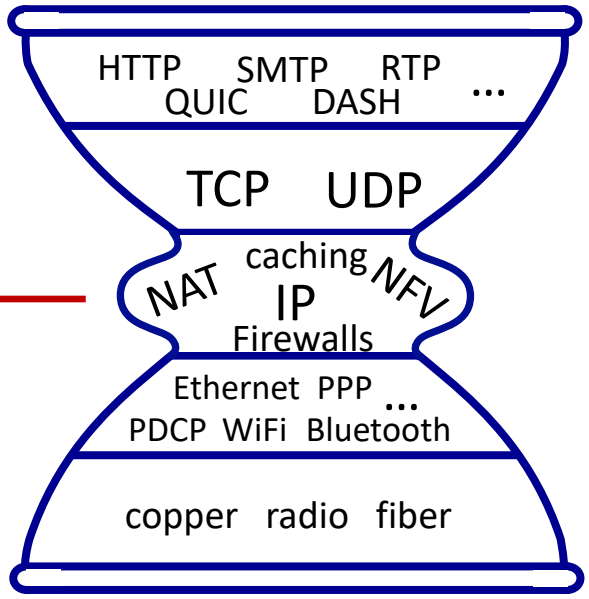


many protocols in physical, link, transport, and application layers

The IP hourglass, at middle age

Internet's middle age "love handles"?

- middleboxes, operating inside the network



Architectural Principles of the Internet (RFC 1958)

RFC 1958

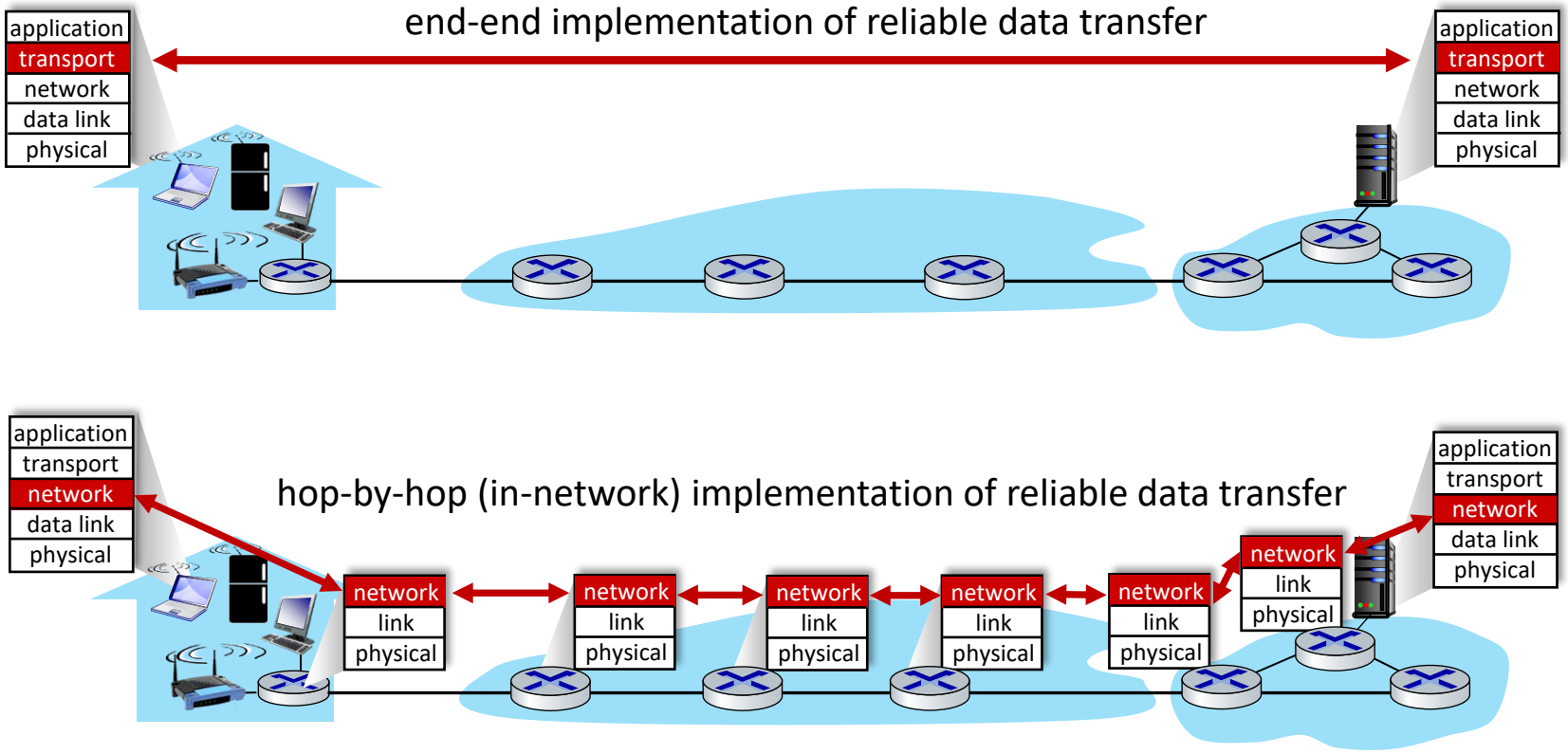
"Many members of the Internet community would argue that there is no architecture, but only a tradition, which was not written down for the first 25 years (or at least not by the IAB). However, in very general terms, the community believes that the goal is connectivity, the tool is the Internet Protocol,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

Three cornerstone beliefs: (i.e. three principles of RFC 1958)

- simple connectivity (Goal)
- IP protocol: that narrow waist (Just one protocol)
- intelligence, complexity at network edge (Rather than network core)

The end-end argument

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

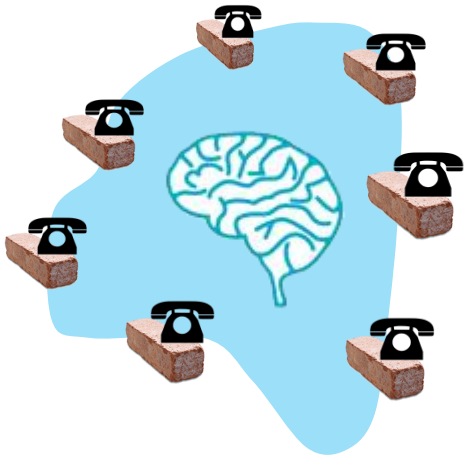


The end-end argument (Third principle)

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

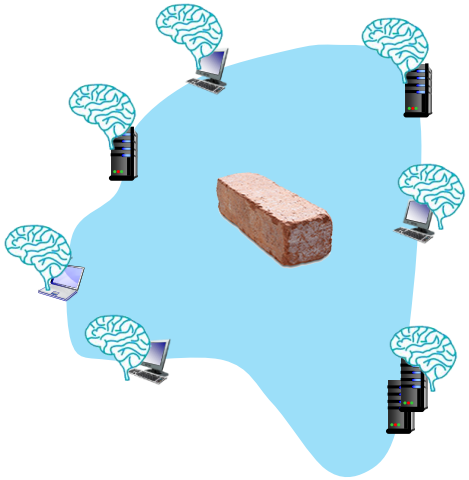
> "The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes 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."
>
> Saltzer, Reed, Clark 1981

Where's the intelligence?



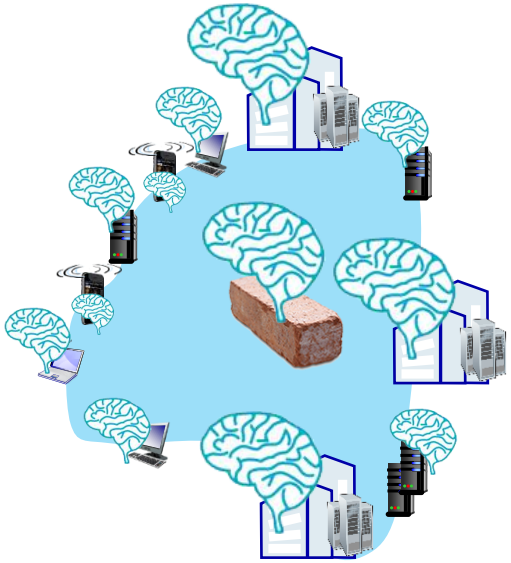
20th century phone net:

- 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

Chapter 4: done!

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

Question: how are forwarding tables (destination-based forwarding) or flow tables (generalized forwarding) computed?

Answer: by the control plane (next chapter)

Quiz 4 – Chapter 4

Midterm II