Computer Networking

谢 逸

中山大学 • 计算机学院

2023. Fall

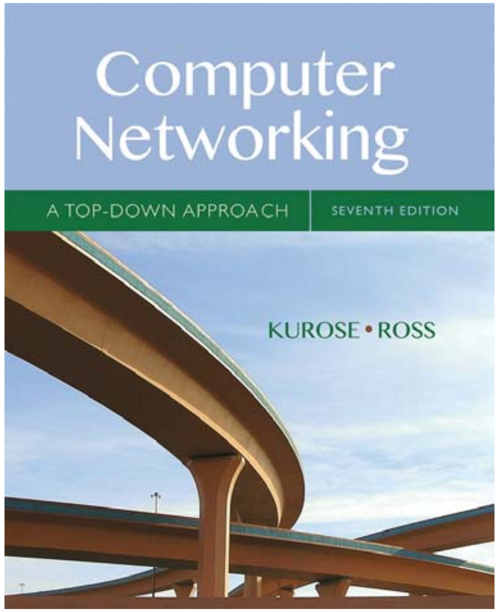
Chapter 4 Network Layer: The Data Plane

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.

Thanks and enjoy! JFK/KWR



Computer Networking: A Top Down Approach

7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

Assignments :

- Ch4 (ver 7, cn): 2, 4, 6, 8, 11, 14, 17, 19, 21, 22
- A team project
- Keywords: Forwarding and Routing, Virtual Circuit and Datagram Networks, Switch, IPv4, Addressing, DHCP, NAT, IPv6, SDN, flow table, Openflow

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

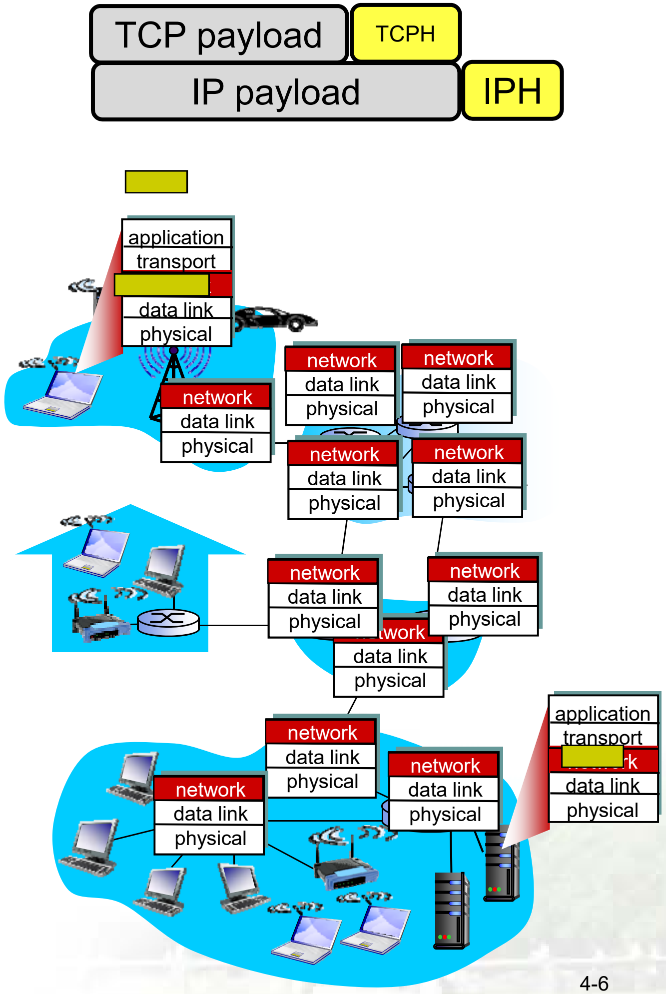
Chapter 4: network layer

chapter goals:

- **understand principles behind network layer services, focusing on data plane:**
  - **network layer service models**
  - **forwarding versus routing**
  - **how a router works**
  - **generalized forwarding**
- **instantiation, implementation in the Internet**

Network layer

- transport segment from sending to receiving host
- on sending side encapsulates **segments** into **datagrams**
- on receiving side, delivers segments to transport layer
- network layer protocols in ***every* host**, **router**
- router examines header fields in all IP datagrams passing through it



Two key network-layer functions

- ***routing:*** **determine route taken by packets from source to dest.**
  - ***routing algorithms***
- ***forwarding:*** **move packets from router's input to appropriate router output**
- **Connection Setup**
  - **Not included in IP**

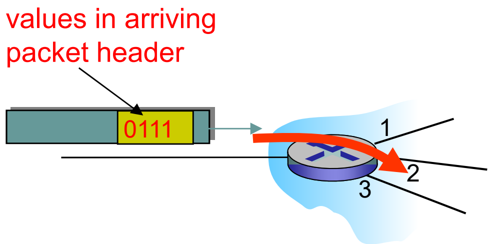
analogy:

- *routing:* process of **planning trip** from source to dest
- *forwarding:* process of **getting through** single interchange

Network layer: data plane, control plane

Data plane

- **local, per-router function**
- **determines how datagram arriving on router input port is forwarded to router output port**
- **forwarding function**

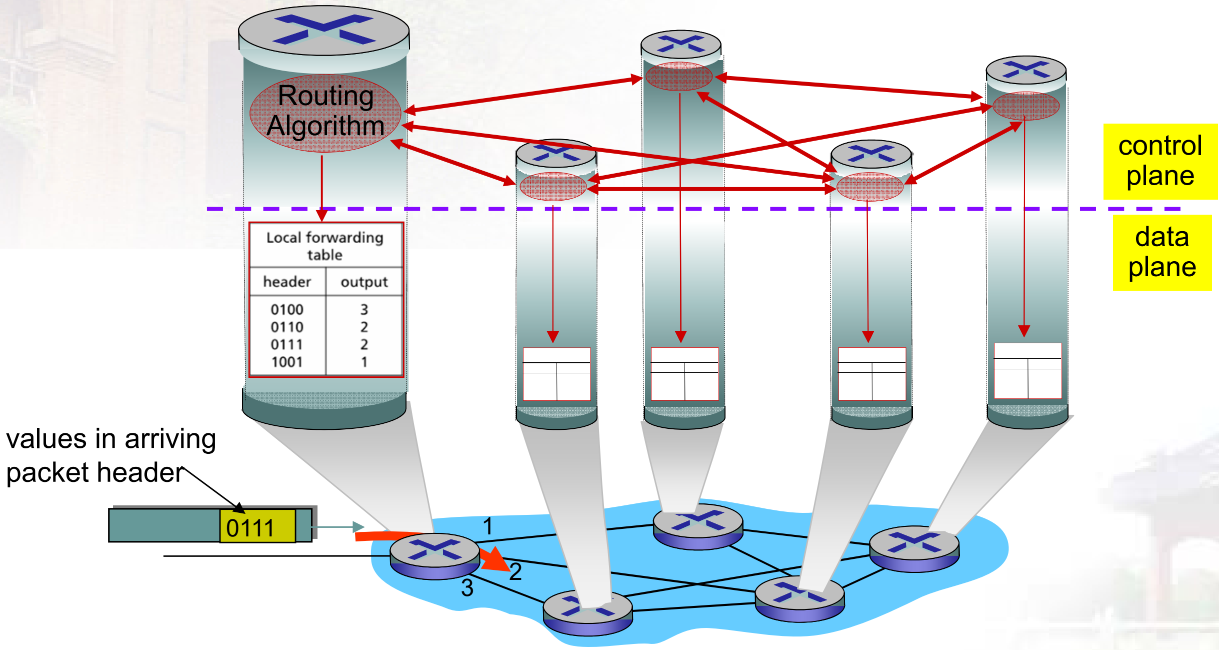


Control plane

- network-wide **logic**
- determines how datagram is **routed** among routers along end-end path from source host to destination host
- two control-plane approaches:
  - *traditional routing algorithms:* implemented in routers
  - *software-defined networking (SDN)*: implemented in (remote) servers

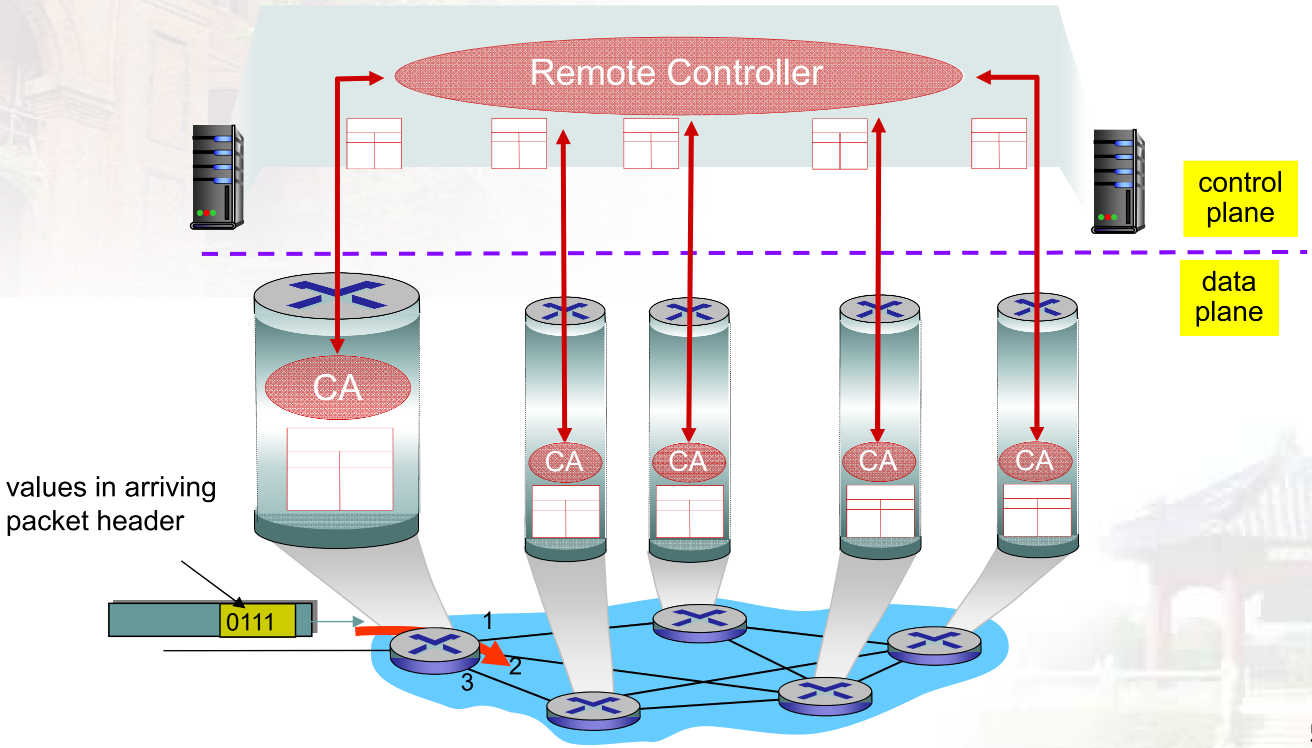
Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane



Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs)



Network service model

Q: What *service model* for "channel" transporting datagrams from sender to receiver?

example services for individual datagrams:

- **guaranteed delivery**
- **guaranteed delivery with less than 40 msec delay**

example services for a flow of datagrams:

- **in-order datagram delivery**
- **guaranteed minimum bandwidth to flow**
- **restrictions on changes in inter-packet spacing**

Network-layer service model

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ? Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no

Internet "best effort" service model

No guarantees on*:*

i. successful datagram delivery to destination
ii. timing or order of delivery
iii. bandwidth available to end-end flow

Network layer service models:

Network Architecture	Service Model	Guarantees ? Bandwidth	Loss	Order	Timing	Congestion feedback
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

CBR：Constant Bit Rate

VBR：Variable Bit Rate

ABR：Available Bit Rate

UBR：Unspecified Bit Rate

Reflections on best-effort service:

- **simplicity of mechanism** has allowed Internet to be widely deployed adopted
- sufficient **provisioning of bandwidth** allows performance of real-time applications (e.g., interactive voice, video) to be "good enough" for "most of the time"
- **replicated, application-layer distributed services** (datacenters, content distribution networks) connecting close to clients' networks, allow services to be provided from multiple locations
- congestion control of "elastic" services helps

It's hard to argue with success of best-effort service model

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

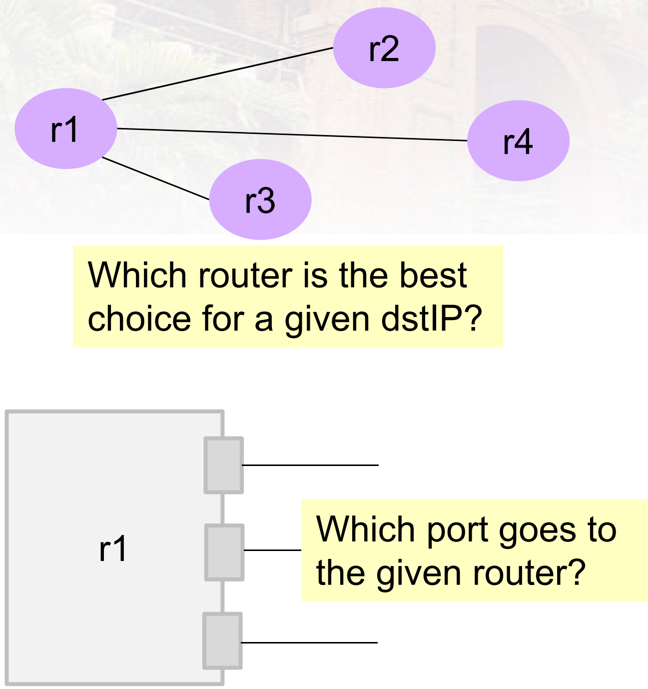
4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

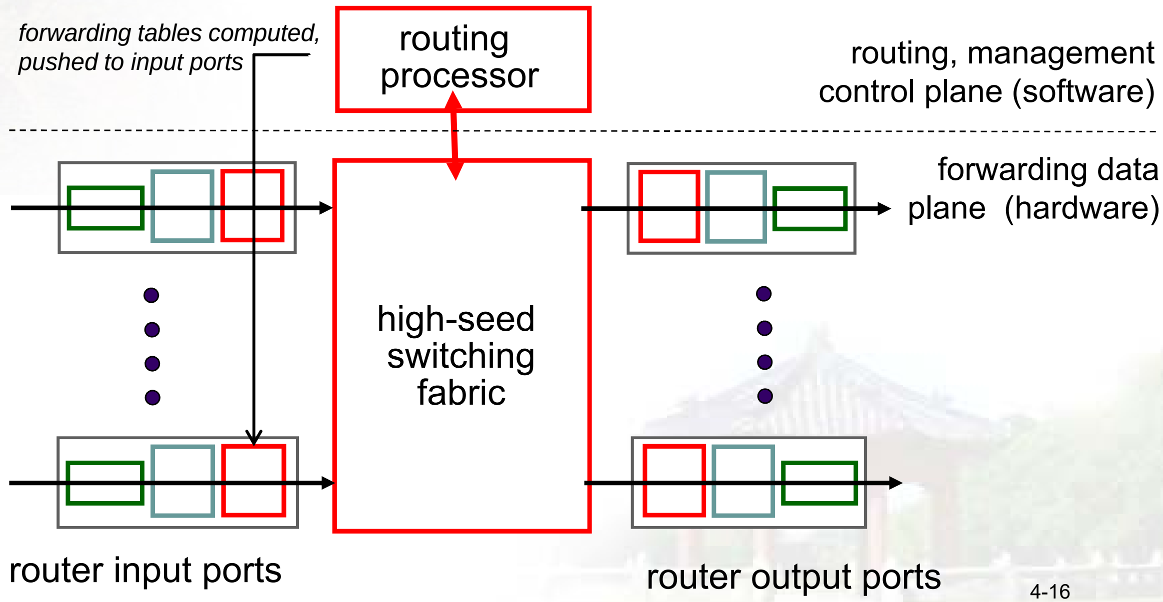
4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

Router architecture overview

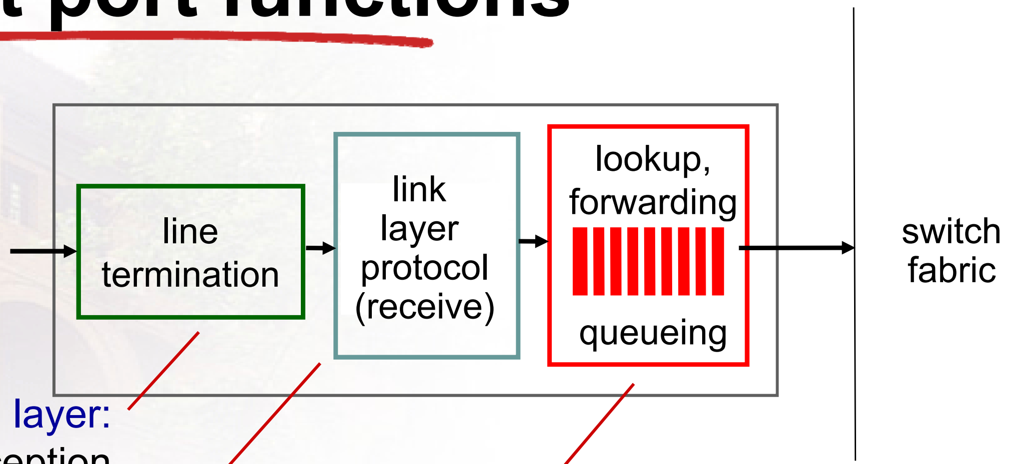
two key router functions:

- **run routing algorithms/protocol (RIP, OSPF, BGP)**
- ***forwarding* datagrams from incoming to outgoing link**





Input port functions



physical layer:
bit-level reception

data link layer:
e.g., Ethernet
see chapter 5

decentralized switching*:*

- **given datagram dest., lookup output port using forwarding table in input port memory (*"match plus action"*)**
- **goal: complete input port processing at 'line speed'**
- **queuing: if datagrams arrive faster than forwarding rate into switch fabric**

Destination-based forwarding

forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through **11001000 00010111 00010111 11111111**	0
11001000 00010111 00011000 00000000 through **11001000 00010111 00011000 11111111**	1
11001000 00010111 00011001 00000000 through **11001000 00010111 00011111 11111111**	2
otherwise	3

Q: but what happens if ranges don't divide up so nicely?

192.168.1.1 ~ 192.168.1.5 r1
192.168.1.6 ~ 192.168.1.10 r2
192.168.1.11 ~ 192.168.1.15 r1
192.168.1.16 ~ 192.168.1.20 r2
…

Longest prefix matching

longest prefix matching

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *********	0
11001000 00010111 00011000 *********	1
11001000 00010111 00011*** *********	2
otherwise	3

examples:

DA: 11001000 00010111 00010110 10100001 which interface?

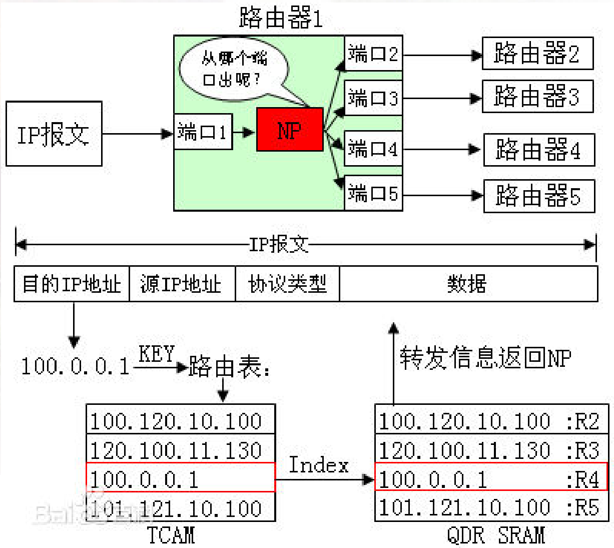
DA: 11001000 00010111 00011000 10101010 which interface?

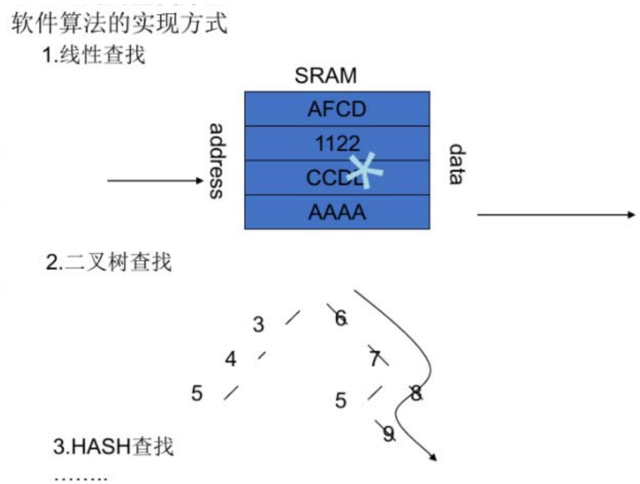
Longest prefix matching

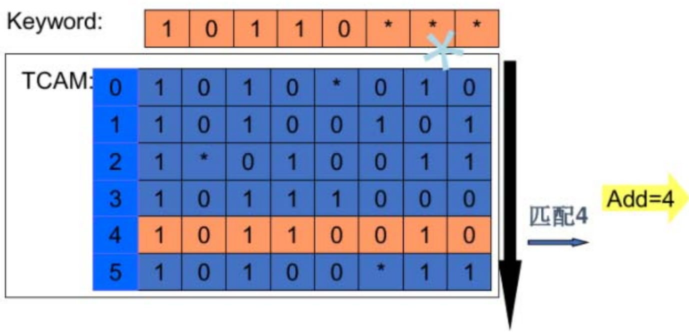
- we'll see *why* longest prefix matching is used shortly, when we study addressing
- longest prefix matching: often performed using ternary content addressable memories (TCAMs)
  - *content addressable:* present address to TCAM: retrieve address in one clock cycle, regardless of table size
  - Cisco Catalyst: can up ~1M routing table entries in TCAM

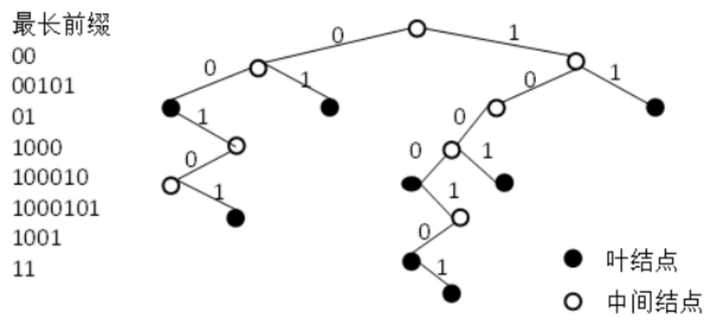
Longest prefix matching

ternary content addressable memories (TCAMs)



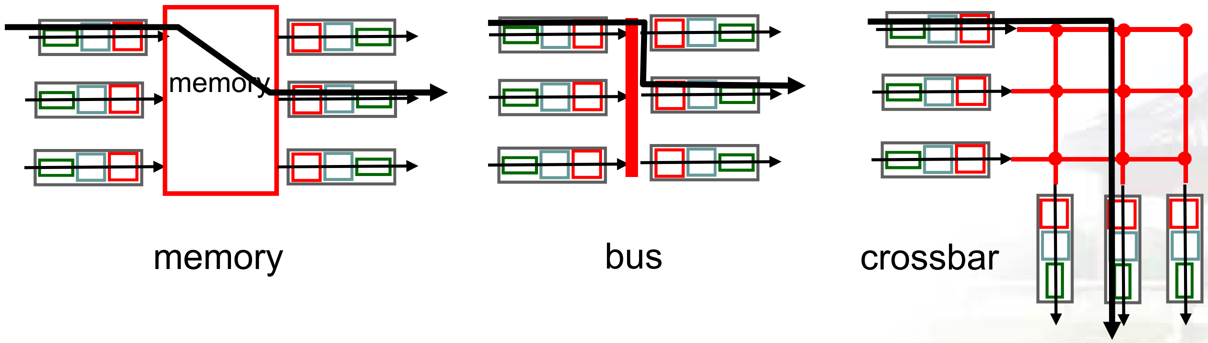






Switching fabrics

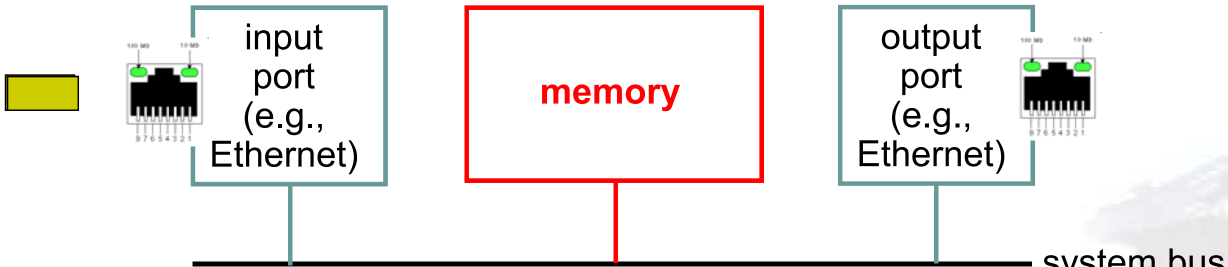
- transfer packet from input buffer to appropriate output buffer
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three types of switching fabrics



Switching via memory

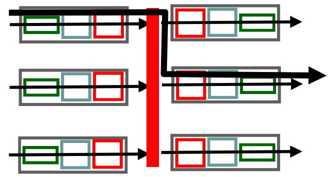
first generation routers:

- **traditional computers with switching under direct control of CPU**
- **packet copied to system’s memory**
- **speed limited by memory bandwidth (2 bus crossings per datagram)**



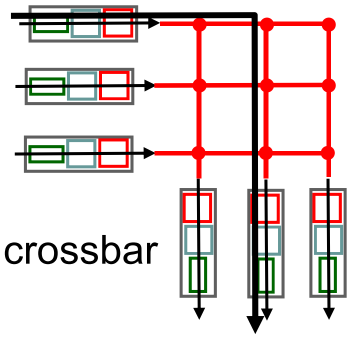
Switching via a bus

- datagram from input port memory to output port memory via a shared bus
- *bus contention:* switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers

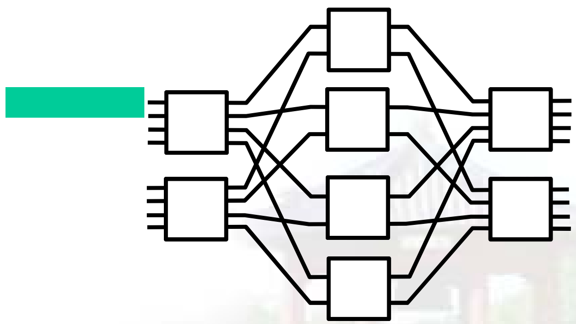


Switching via interconnection network

- overcome bus bandwidth limitations
- banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection network

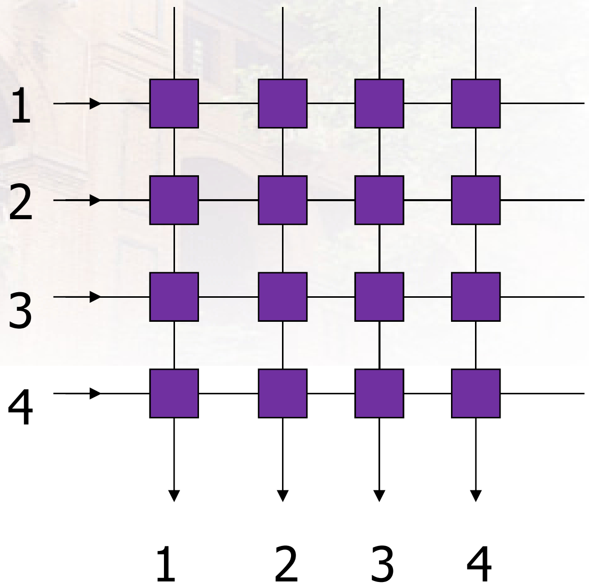


crossbar

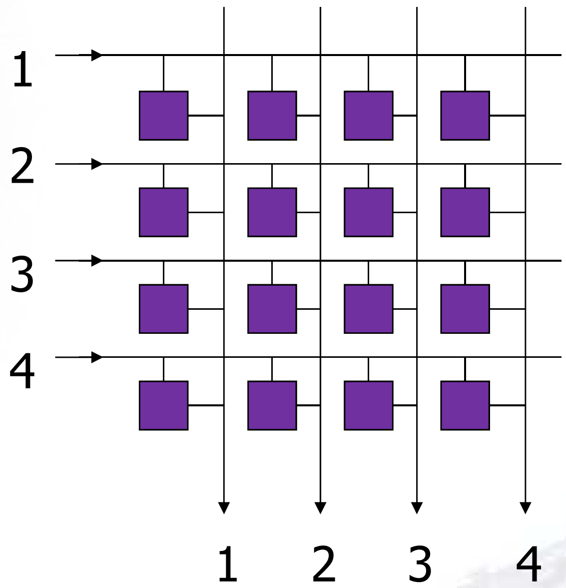


8x8 multistage switch
built from smaller-sized switches

Switching via interconnection network



crossbar

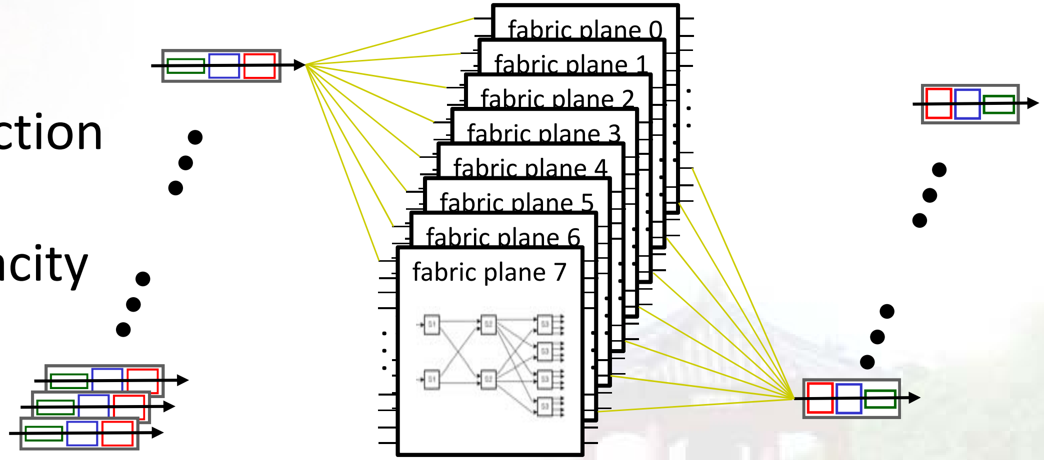


通/断开关

Switching via interconnection network

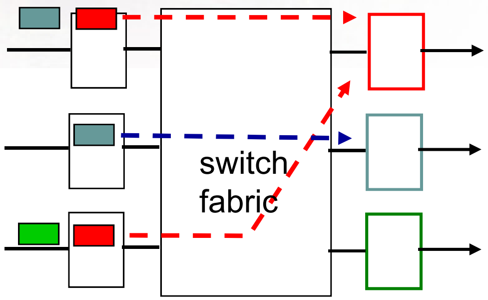
- scaling, using multiple switching "planes" in parallel:
  - speedup, scaleup via parallelism

- Cisco CRS router:
  - basic unit: 8 switching planes
  - each plane: 3-stage interconnection network
  - up to 100's Tbps switching capacity

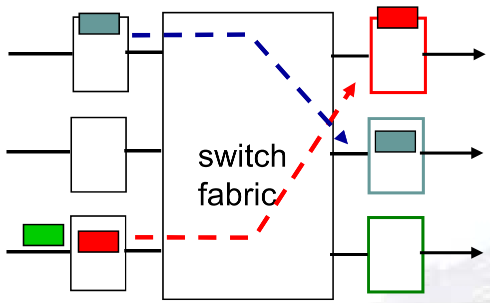


Input port queuing

- **fabric slower than input ports combined -> queueing may occur at input queues**
  - ***queueing delay and loss due to input buffer overflow!***
- **Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward**



output port contention:
only one red datagram can be transferred.
lower red packet is blocked

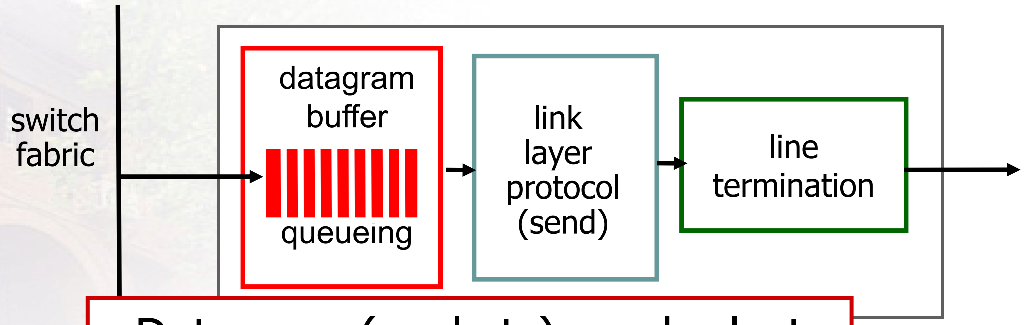


one packet time later: green packet experiences HOL blocking

4-28

Output ports

This slide in HUGELY important!



Datagram (packets) can be lost due to congestion, lack of buffers

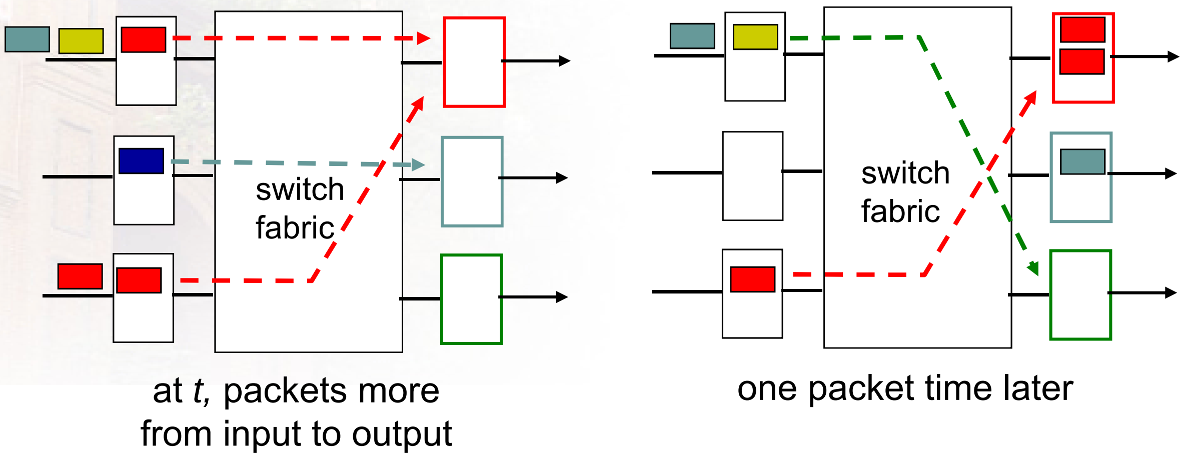
- ***buffering*** **required when datagrams arrive from fabric faster than the transmission rate**
- ***scheduling discipline*** **chooses among queued datagrams for transmission**

Priority scheduling – who gets best performance, network neutrality

10.26

Why is the lose in output serious than the input?

Output port queueing



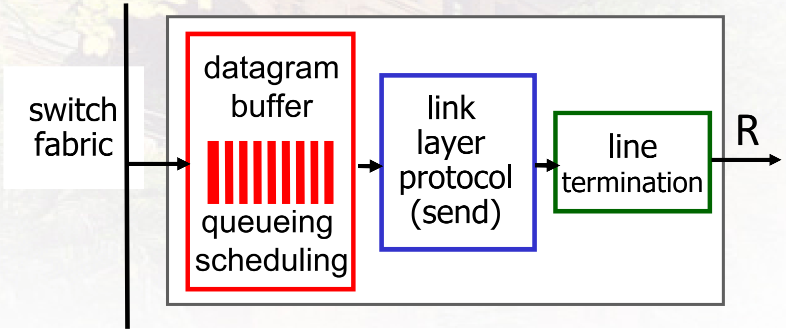
- **buffering when arrival rate via switch exceeds output line speed**
- ***queueing (delay) and loss due to output port buffer overflow!***

How much buffering?

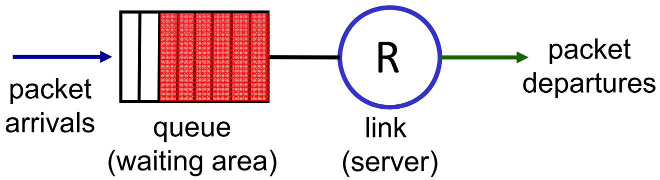
- RFC 3439 rule of thumb: average buffering equal to "typical" RTT (say 250 msec) times link capacity C
  - e.g., C = 10 Gpbs link: 2.5 Gbit buffer
- recent recommendation: with *N* flows, buffering equal to

$$\frac{RTT \cdot C}{\sqrt{N}}$$

Buffer Management



Abstraction: queue

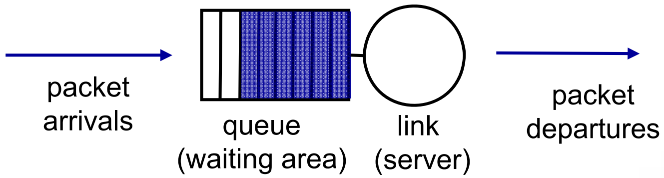


buffer management:

- drop: which packet to add, drop when buffers are full
  - tail drop: drop arriving packet
  - priority: drop/remove on priority basis
- marking: which packets to mark to signal congestion (ECN, RED)

Scheduling mechanisms

- ***scheduling:*** **choose next packet to send on link**
- ***FIFO (first in first out) scheduling:*** **send in order of arrival to queue**
  - **real-world example?**
  - ***discard policy:*** **if packet arrives to full queue: who to discard?**
    - ***tail drop:*** **drop arriving packet**
    - ***priority:*** **drop/remove on priority basis**
    - ***random:*** **drop/remove randomly**

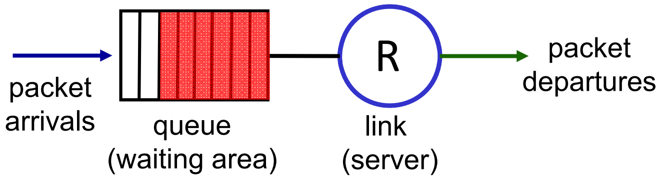


Packet Scheduling: FCFS

packet scheduling: deciding which packet to send next on link

- **first come, first served**
- **priority**
- **round robin**
- **weighted fair queueing**

Abstraction: queue



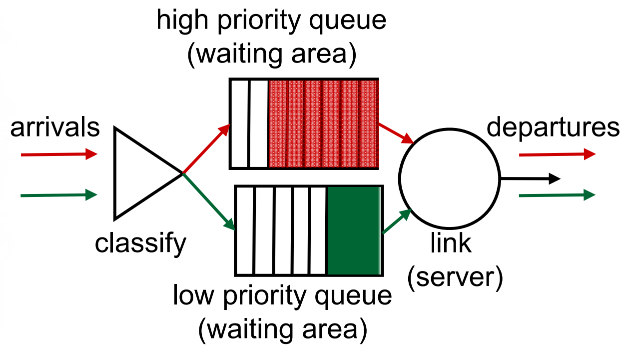
FCFS: packets transmitted in order of arrival to output port

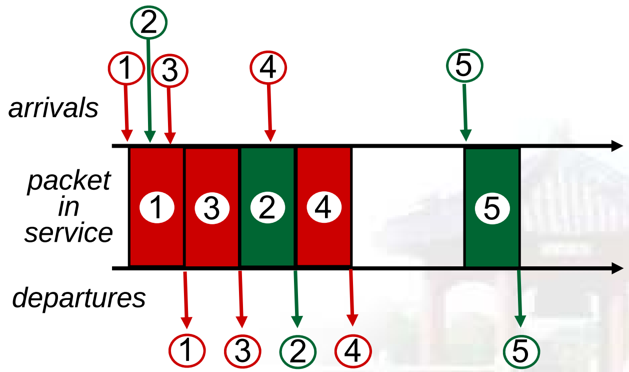
- also known as: First-in-first-out (FIFO)
- real world examples?

Scheduling policies: priority

priority scheduling: **send highest priority queued packet**

- **multiple *classes*, with different priorities**
  - **class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.**
  - **real world example?**

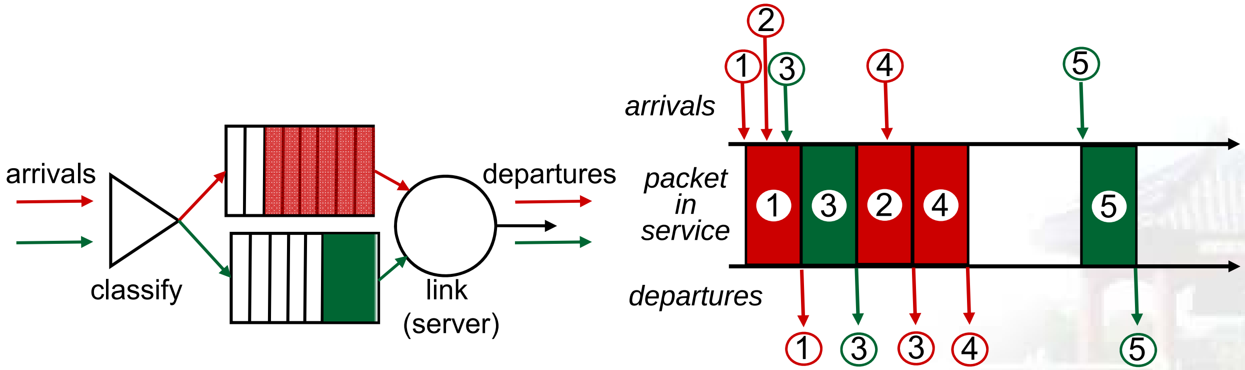




Scheduling policies: still more

Round Robin (RR) scheduling:

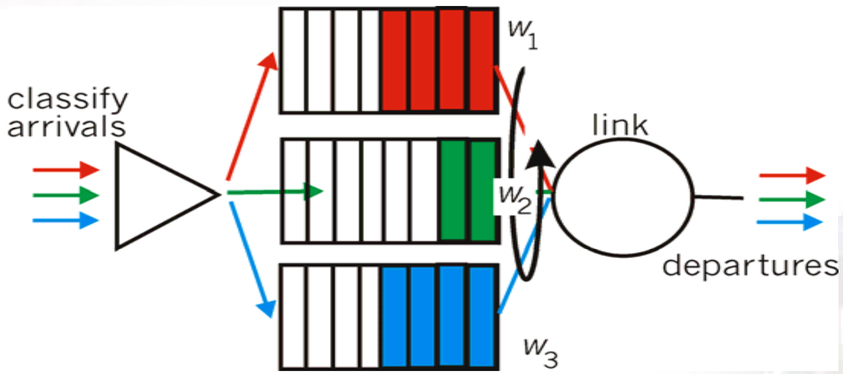
- multiple classes
- cyclically scan class queues, sending **one complete packet from each class** (if available)
- real world example?



Scheduling policies: still more

Weighted Fair Queuing (WFQ):

- **generalized Round Robin**
- **each class gets weighted amount of service in each cycle**
- **real-world example?**



Sidebar: Network Neutrality

What is network neutrality?

- *technical:* how an ISP should share/allocation its resources
  - packet scheduling, buffer management are the *mechanisms*
- *social, economic* principles
  - protecting free speech
  - encouraging innovation, competition
- enforced *legal* rules and policies

Different countries have different "takes" on network neutrality

Sidebar: Network Neutrality

2015 US FCC *Order on Protecting and Promoting an Open Internet:* three "clear, bright line" rules:

- **no blocking … "shall not block lawful content, applications, services, or non-harmful devices, subject to reasonable network management."**
- **no throttling … "shall not impair or degrade lawful Internet traffic on the basis of Internet content, application, or service, or use of a non-harmful device, subject to reasonable network management."**
- **no paid prioritization. … "shall not engage in paid prioritization"**

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

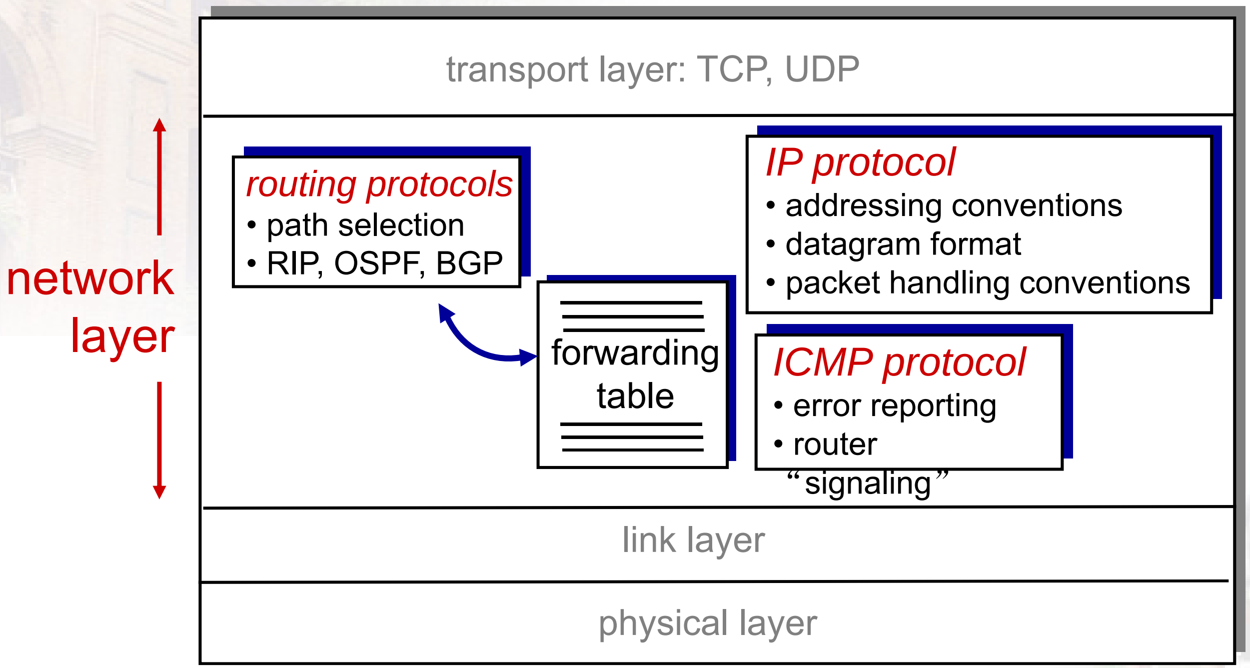
4.2 What' s inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

The Internet network layer

host, router network layer functions:



IP datagram format

IP protocol version number

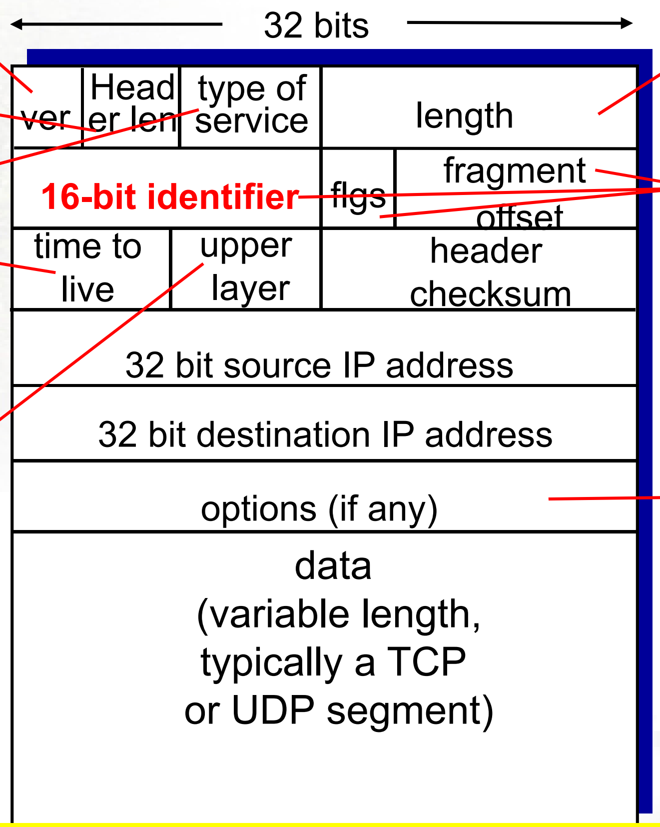
header length (bytes)

"type" of data

max number remaining hops (decremented at each router)

TCP=6;
UDP=17

upper layer protocol to deliver payload to



total datagram length (bytes)

for fragmentation/ reassembly

e.g. timestamp, record route taken, specify list of routers to visit.

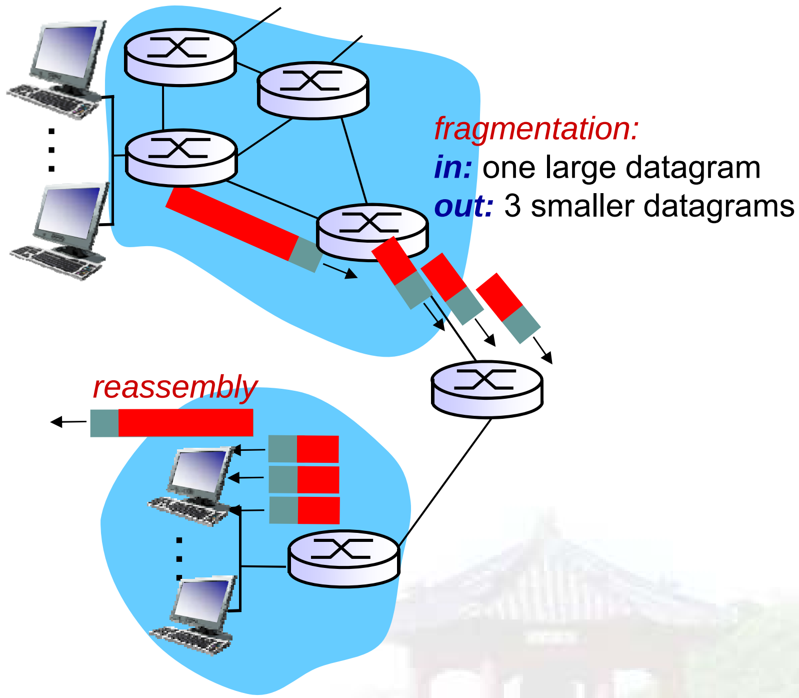
how much overhead?

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

"16-bit identifier" is NOT a number for reordering
No reordering in Net Layer

IP fragmentation, reassembly

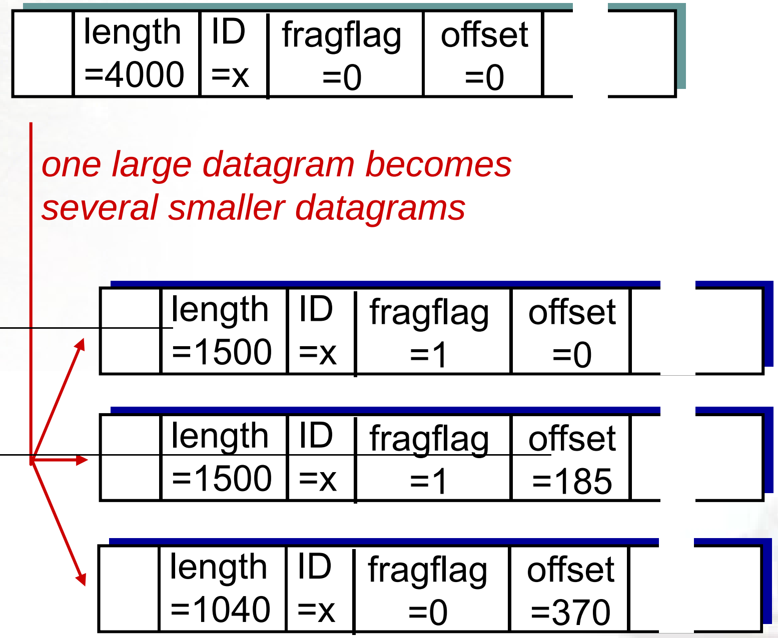
- network links have MTU (max.transfer size) - largest possible link-level frame
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order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



IP fragmentation, reassembly

example:

- 4000 byte datagram
- Payload=3980
- MTU = 1500 bytes



1480 bytes in data field

offset = **1480/8**

(1500-20)+(1500-20)+(1040-20)=3980

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

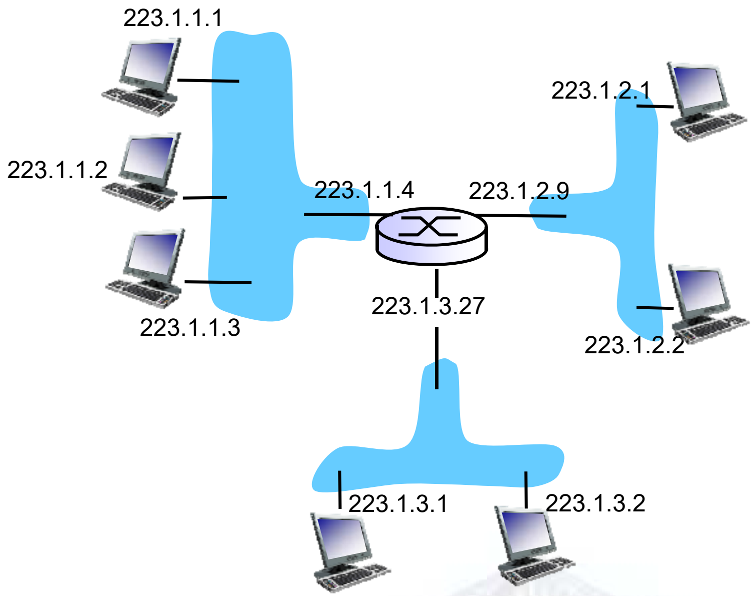
4.2 What’s inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

IP addressing: introduction

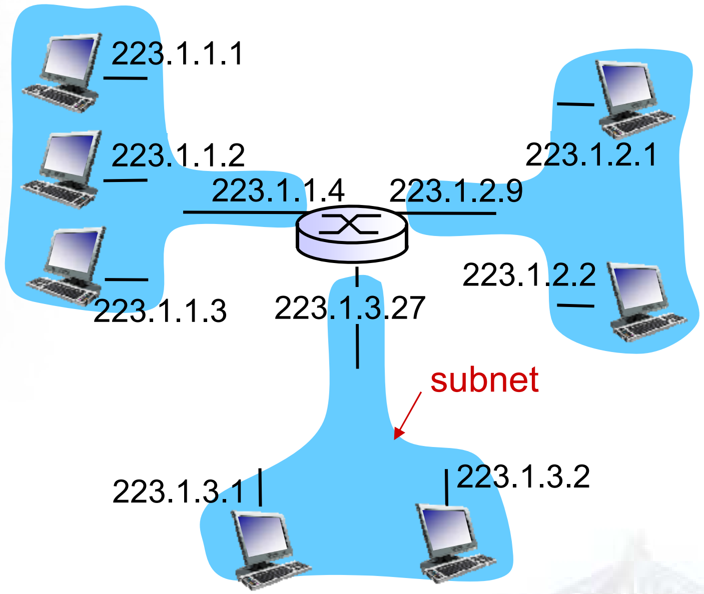
- ***IP address:*** **32-bit identifier for host, router *interface***
- ***interface:*** **connection between host/router and physical link**
  - **Router's typically have multiple interfaces**
  - **host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)**
- ***IP addresses associated with each interface***



$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

Subnets

- **IP address:**
  - **subnet part - high order bits**
  - **host part - low order bits**
- ***what's a subnet ?***
  - **device interfaces with same subnet part of IP address**
  - **can physically reach each other *without intervening router***

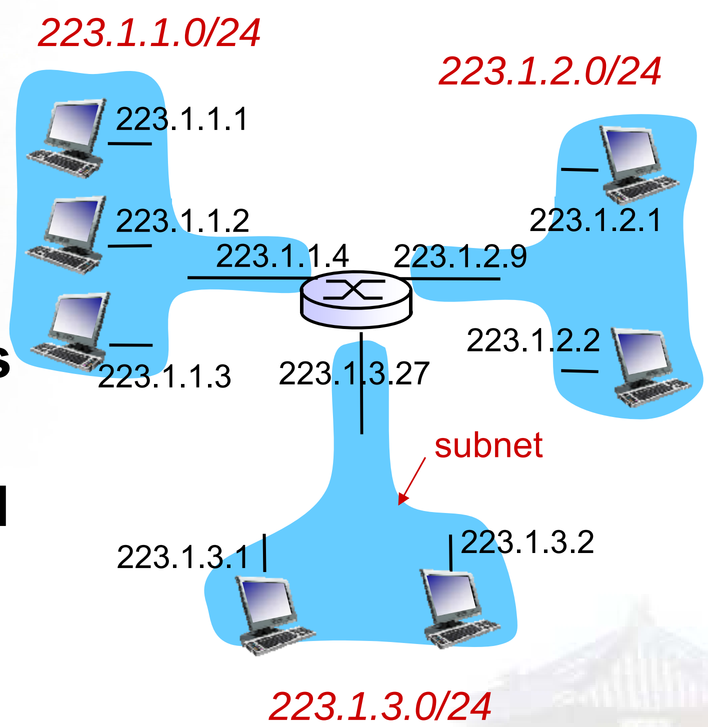


network consisting of 3 subnets

Subnets

recipe

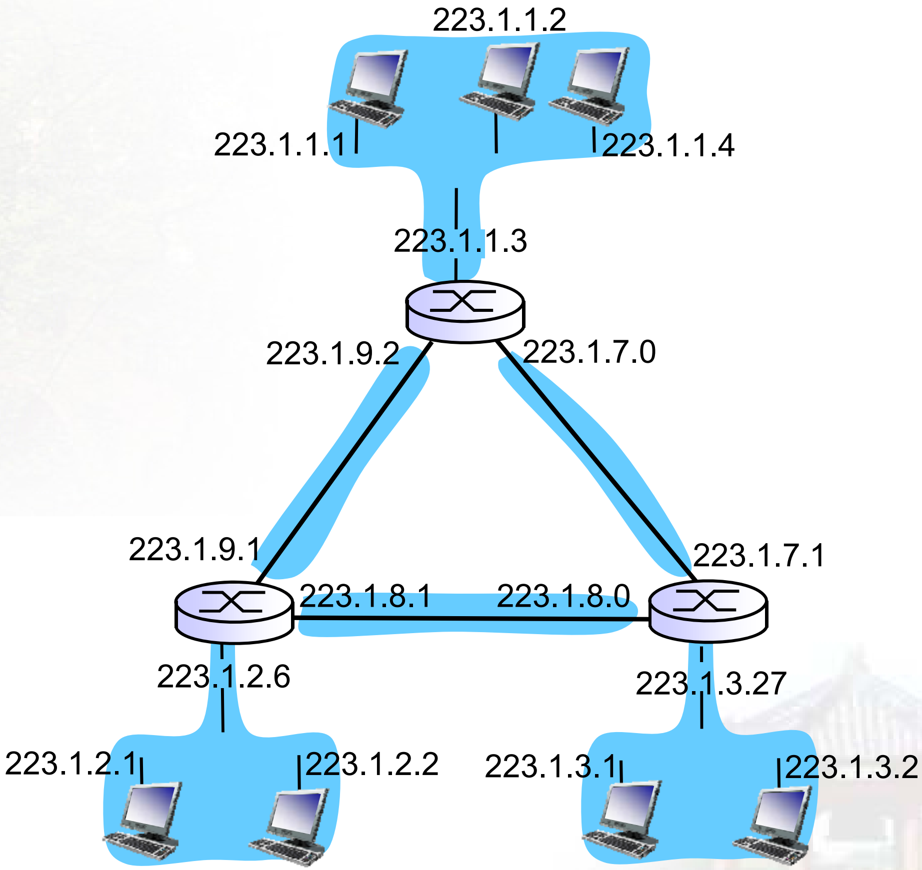
- **to determine the subnets, detach each interface from its host or router, creating islands of isolated networks**
- **each isolated network is called a *subnet***



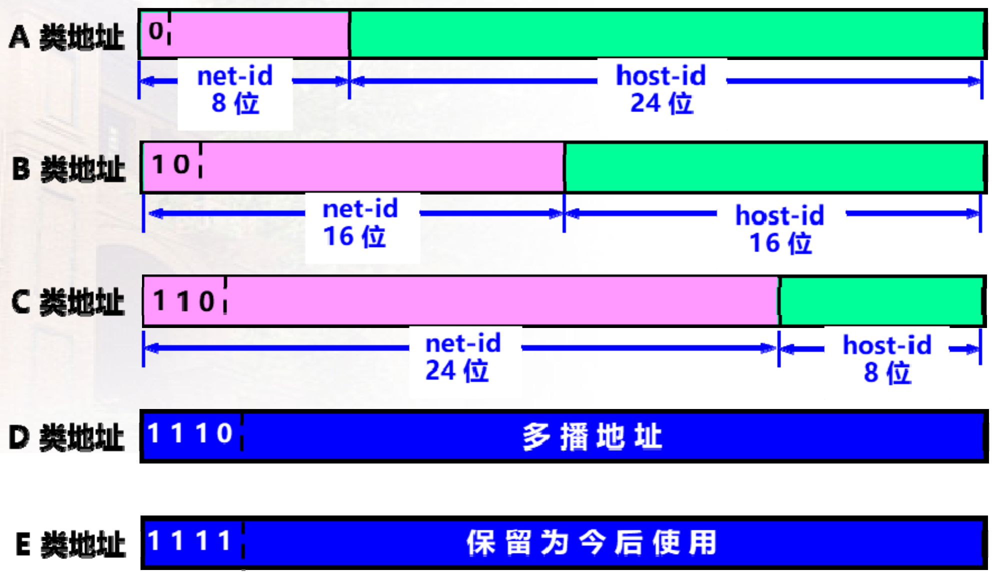
subnet mask: /24

Subnets

how many?



IP Address Classes



网络类别	最大可指派的网络数	第一个可指派的网络号	最后一个可指派的网络号	每个网络中最大主机数
A	126 ($2^7 - 2$)	1	126	16777214
B	16383 ($2^{14} - 1$)	128.1	191.255	65534
C	2097151 ($2^{21} - 1$)	192.0.1	223.255.255	254

IP Address Classes

私有IP地址是一段保留的IP地址。只是使用在局域网中，在Internet上是不使用的。 私有IP地址的范围有:

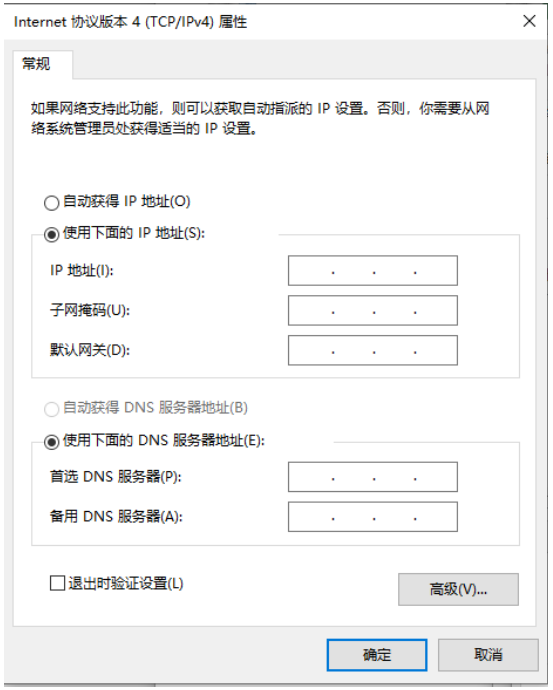
类别	IP范围	IP范围	网络数量
A	10.x.x.x/8	10.0.0.0~10.255.255.255	1
B	172.16-32.x.x/16	172.16.0.0~172.31.255.255	16
C	192.168.x.x/24	192.168.0.0~192.168.255.255	255

网络号	主机号	源地址使用	目的地址使用	代表的意思
0	0	可以	不可	在本网络上的本主机（见 6.6 节 DHCP 协议）
0	host-id	可以	不可	在本网络上的某台主机 host-id
全 1	全 1	不可	可以	只在本网络上进行广播（各路由器均不转发）
net-id	全 1	不可	可以	对 net-id 上的所有主机进行广播
127	非全 0 或全 1 的任何数	可以	可以	用于本地软件环回测试

IP addressing: CIDR

CIDR: Classless InterDomain Routing

- **subnet portion of address of arbitrary length**
- **address format: a.b.c.d/x, where x is # bits in subnet portion of address**



subnet part | host part

11001000 00010111 00010000 00000000

$\underbrace{\text{11001000 00010111 0001000}}_{\text{23bits}}$

23bits 200.23.16.0/23

IP addresses: how to get one?

Q: **How does a *host* get IP address?**

- **hard-coded by system admin in a file**
  - **Windows: control-panel->network->configuration->tcp/ip->properties**
  - **UNIX: /etc/rc.config**
- **DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server**
  - **“plug-and-play”**

DHCP: Dynamic Host Configuration Protocol

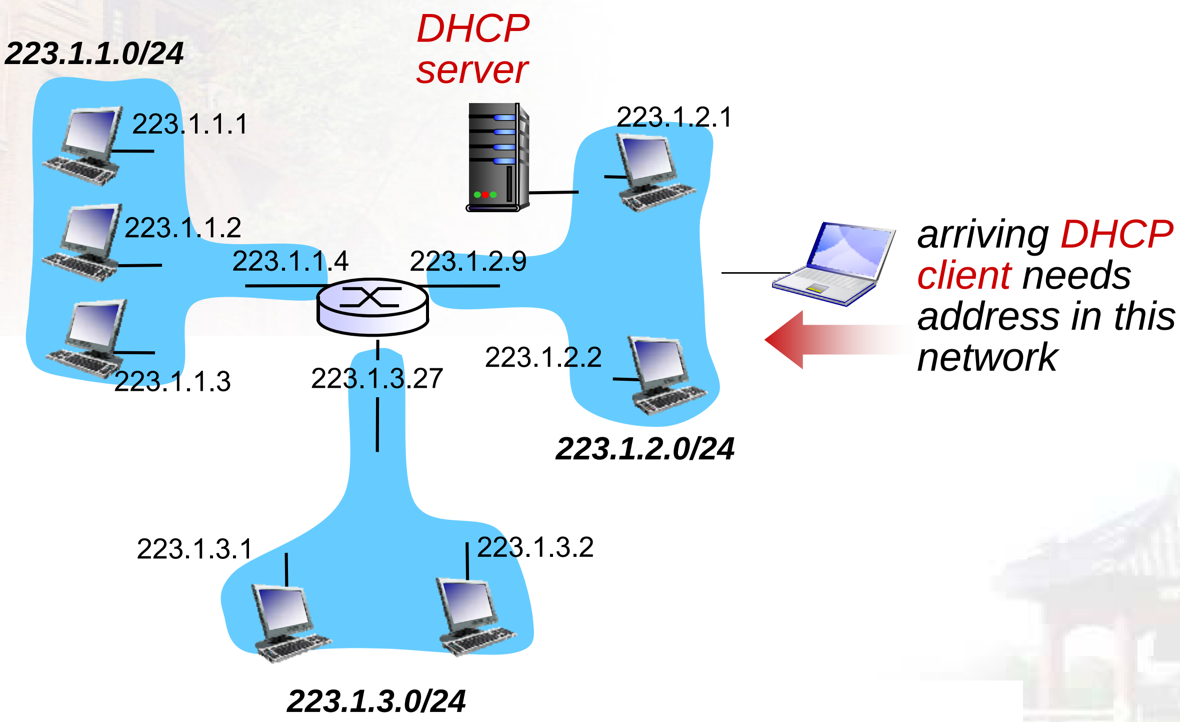
goal: allow host to *dynamically* obtain its IP address from network server when it joins network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/“on”)
- support for mobile users who want to join network (more shortly)

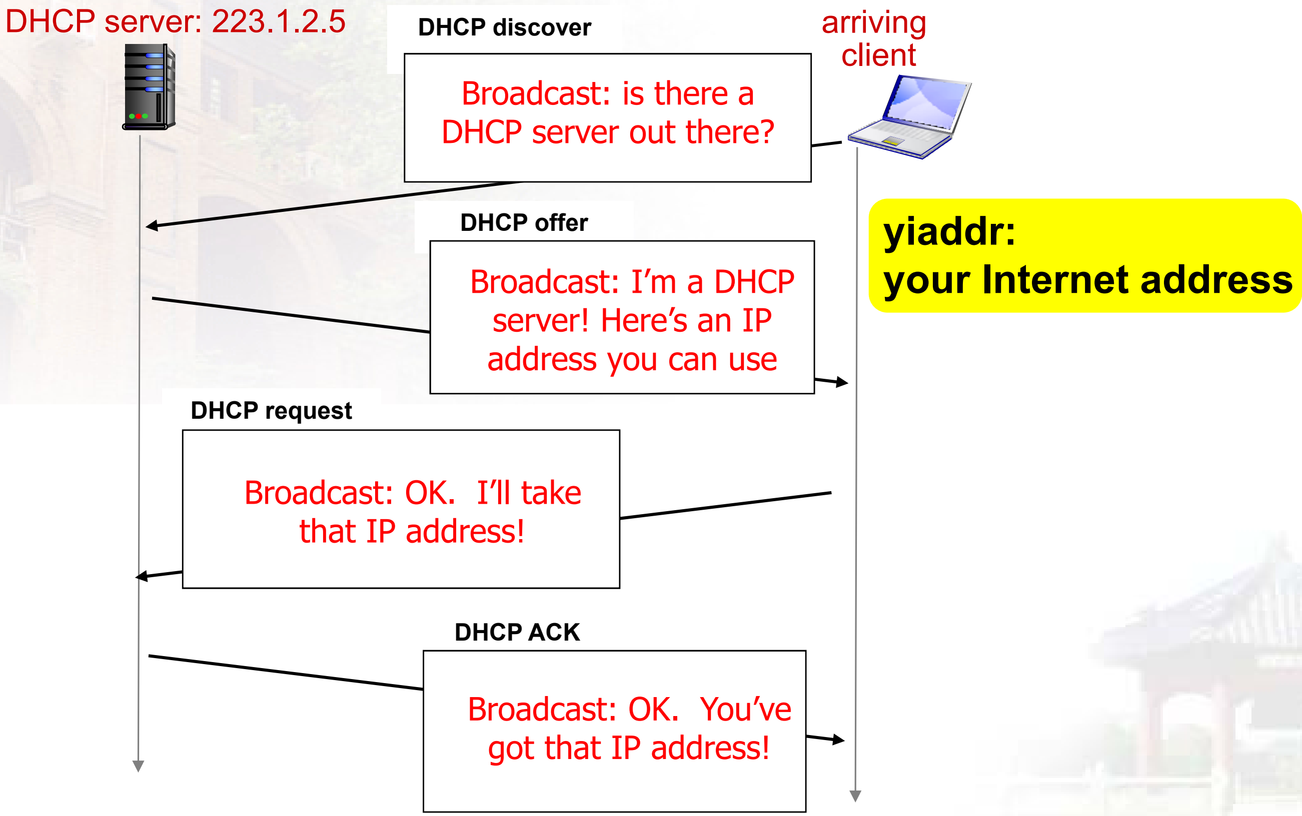
DHCP overview:

- 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



DHCP client-server scenario

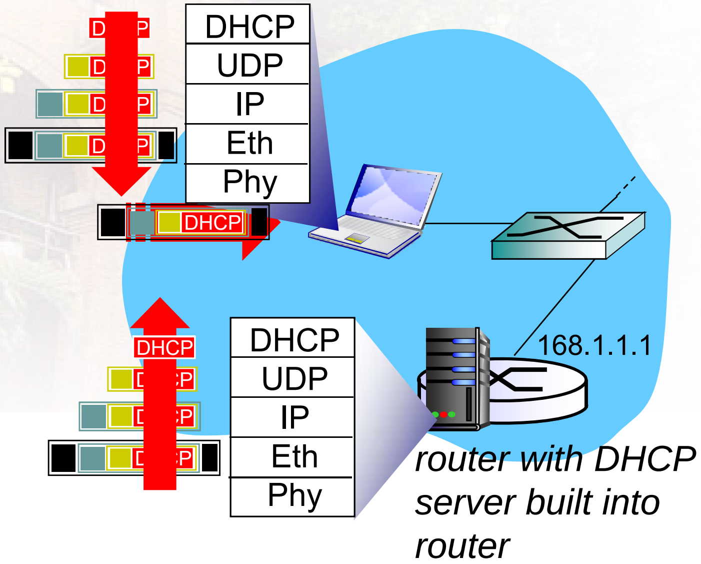


DHCP: more than IP addresses

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

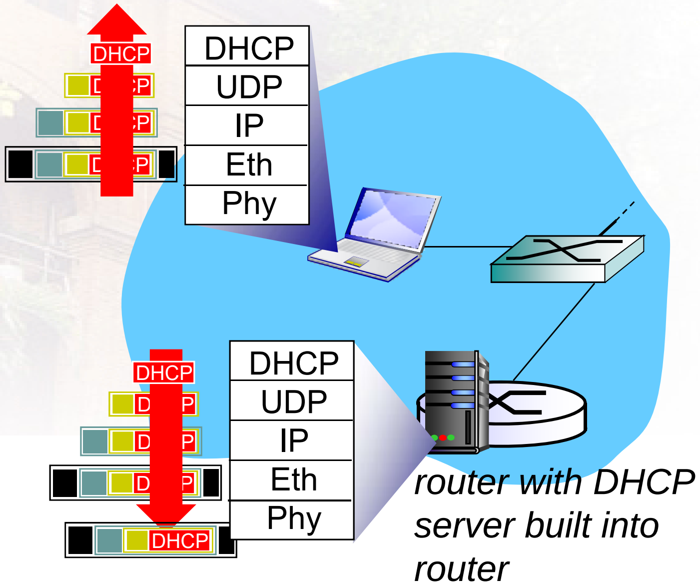
- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

DHCP: example



- **connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP**
- DHCP request encapsulated in **UDP**, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

DHCP: example



- **DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server**

- encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client

- client now knows its IP address, name and IP address of DSN server, IP address of its first-hop router

DHCP: Wireshark output (home LAN)

request

Message type: **Boot Request (1)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) **DHCP Message Type = DHCP Request**
Option: (61) Client identifier
  Length: 7; Value: 010016D323688A;
  Hardware type: Ethernet
  Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Option: (t=50,l=4) Requested IP Address = 192.168.1.101
Option: (t=12,l=5) Host Name = "nomad"
Option: (55) Parameter Request List
  Length: 11; Value: 010F03062C2E2F1F21F92B
  1 = Subnet Mask; 15 = Domain Name
  3 = Router; 6 = Domain Name Server
  44 = NetBIOS over TCP/IP Name Server
  ……

reply

Message type: **Boot Reply (2)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 192.168.1.101 (192.168.1.101)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 192.168.1.1 (192.168.1.1)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) DHCP Message Type = DHCP ACK
Option: (t=54,l=4) Server Identifier = 192.168.1.1
Option: (t=1,l=4) Subnet Mask = 255.255.255.0
Option: (t=3,l=4) Router = 192.168.1.1
Option: (6) Domain Name Server
  Length: 12; Value: 445747E2445749F244574092;
  IP Address: 68.87.71.226;
  IP Address: 68.87.73.242;
  IP Address: 68.87.64.146
Option: (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."

IP addresses: how to get one?

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

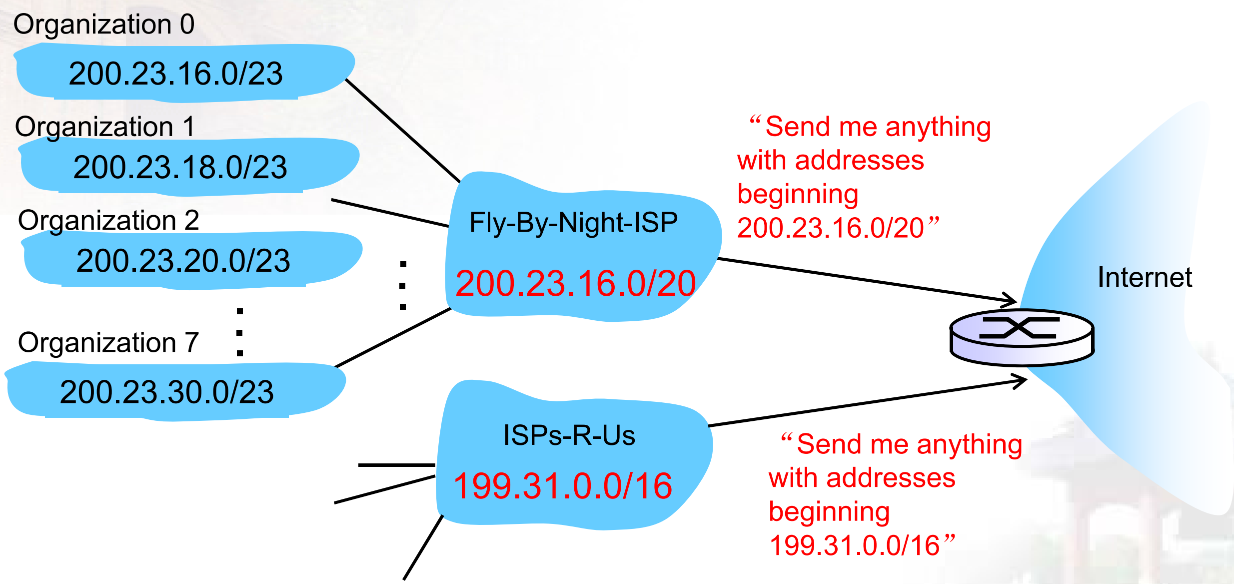
A: gets allocated portion of its provider ISP's address space

ISP's block	11001000 00010111 00010000 00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000 00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010 00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100 00000000	200.23.20.0/23
...	..... ….	….
Organization 7	11001000 00010111 00011110 00000000	200.23.30.0/23

4-64

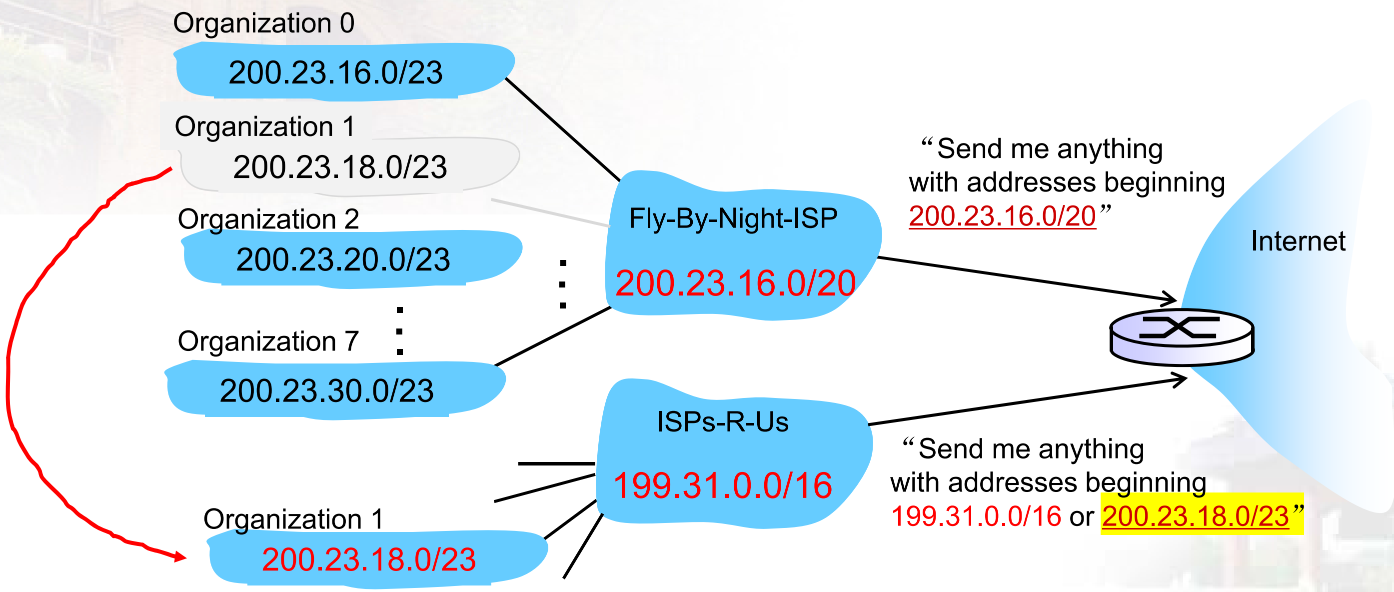
Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

ISPs-R-Us has a more specific route to Organization 1



10.29

IP addressing: the last word...

Q: how does an ISP get block of addresses?

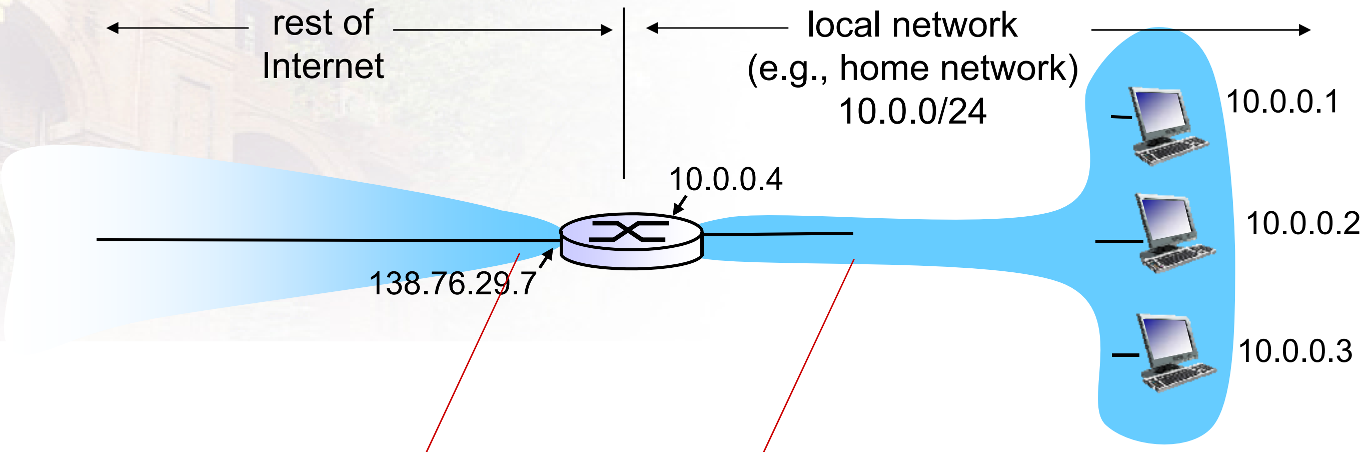
A: ICANN: Internet Corporation for Assigned Names and Numbers http://www.icann.org/

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

ICANN

One World, One Internet

NAT: network address translation



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

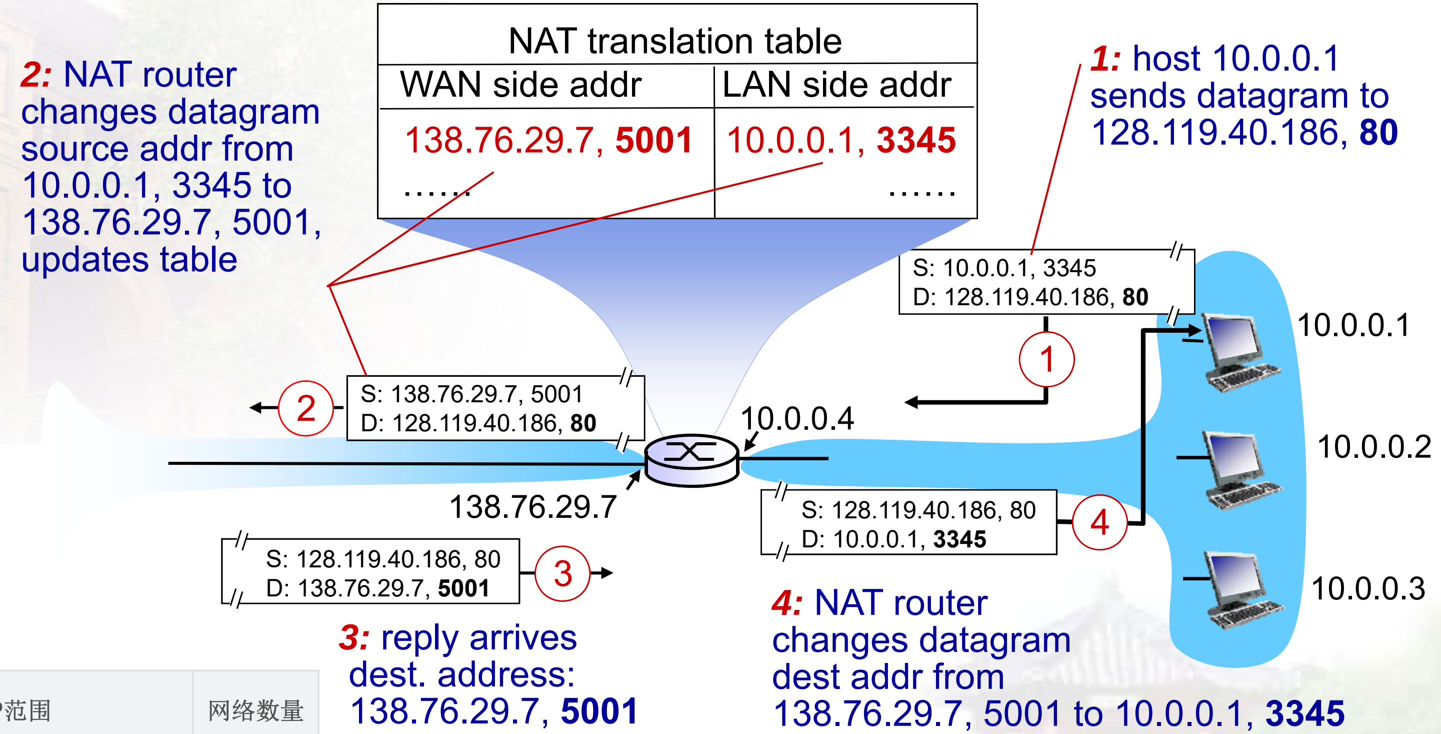
datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: network address translation

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

- range of addresses not needed from ISP: just one IP address for all devices
- 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



类别	IP范围	IP范围	网络数量
A	10.x.x.x/8	10.0.0.0~10.255.255.255	1
B	172.16-32.x.x/16	172.16.0.0~172.31.255.255	16
C	192.168.x.x/24	192.168.0.0~192.168.255.255	255

NAT: network address translation

implementation: **NAT router must:**

- ***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 addr

- ***remember (in NAT translation table)*** **every (source IP address, port #) to (NAT IP address, new port #) translation pair**

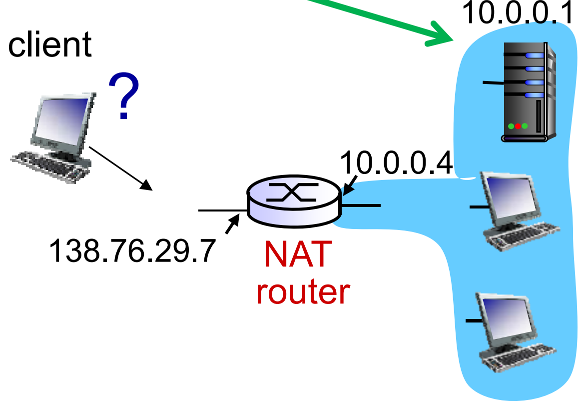
- ***incoming datagrams: replace*** **(NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table**

NAT: network address translation

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
  - routers should only process up to layer 3
  - 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

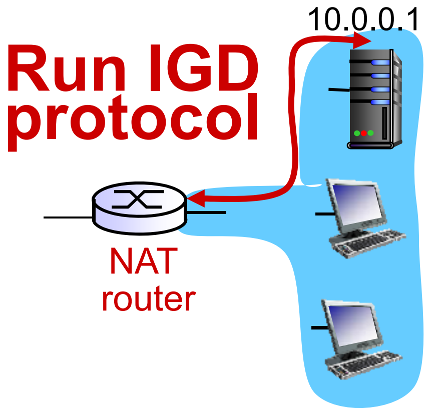
- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7
- *solution1: statically configure* NAT to forward incoming connection requests at given port to server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



NAT traversal problem

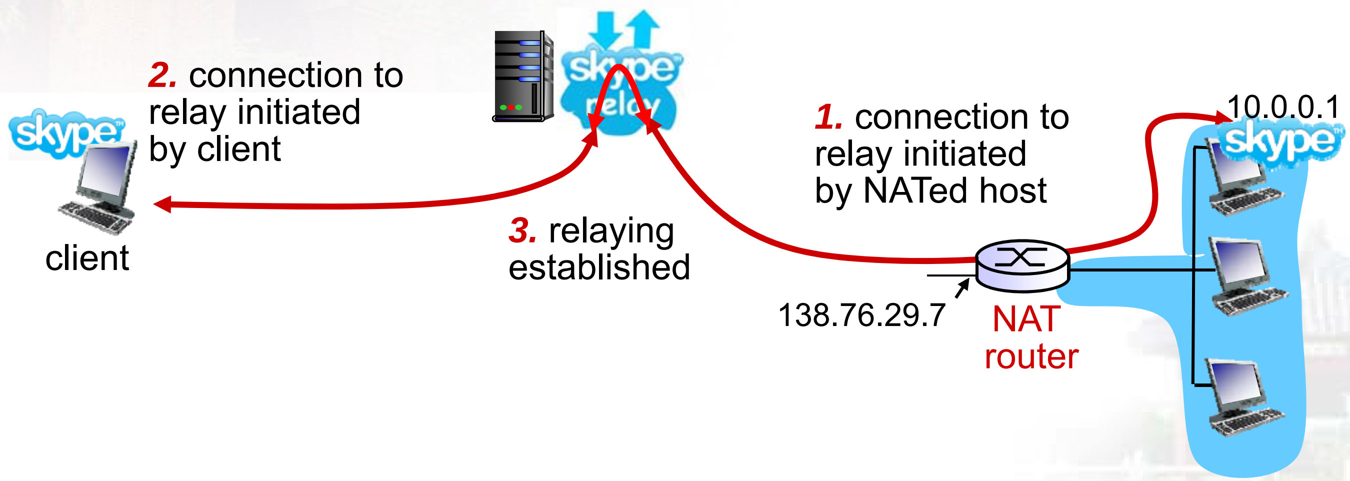
- *solution 2:* Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:
  - learn public IP address (138.76.29.7)
  - add/remove port mappings (with lease times)

i.e., automate static NAT port map configuration



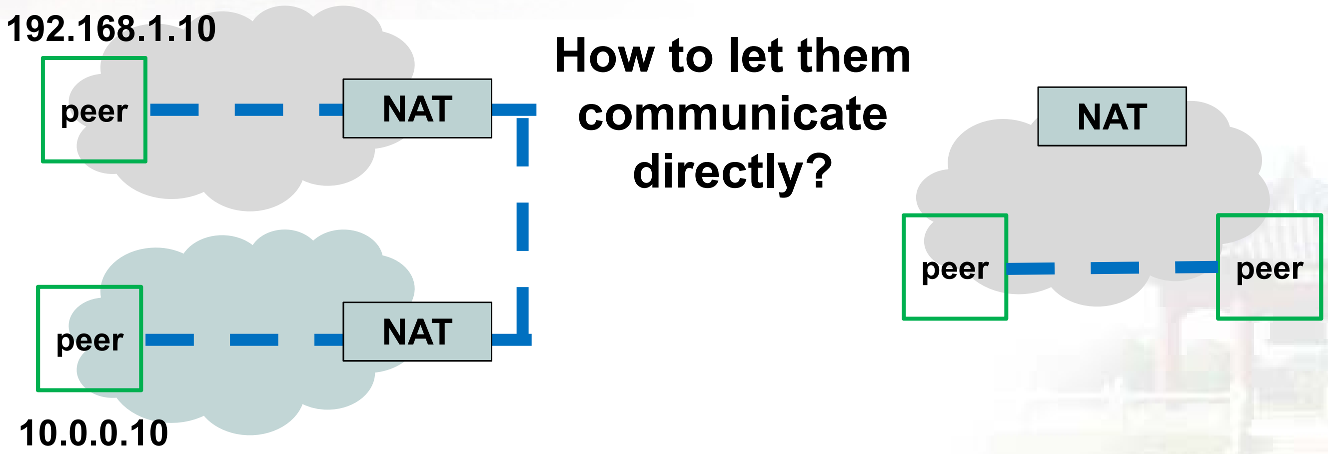
NAT traversal problem

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



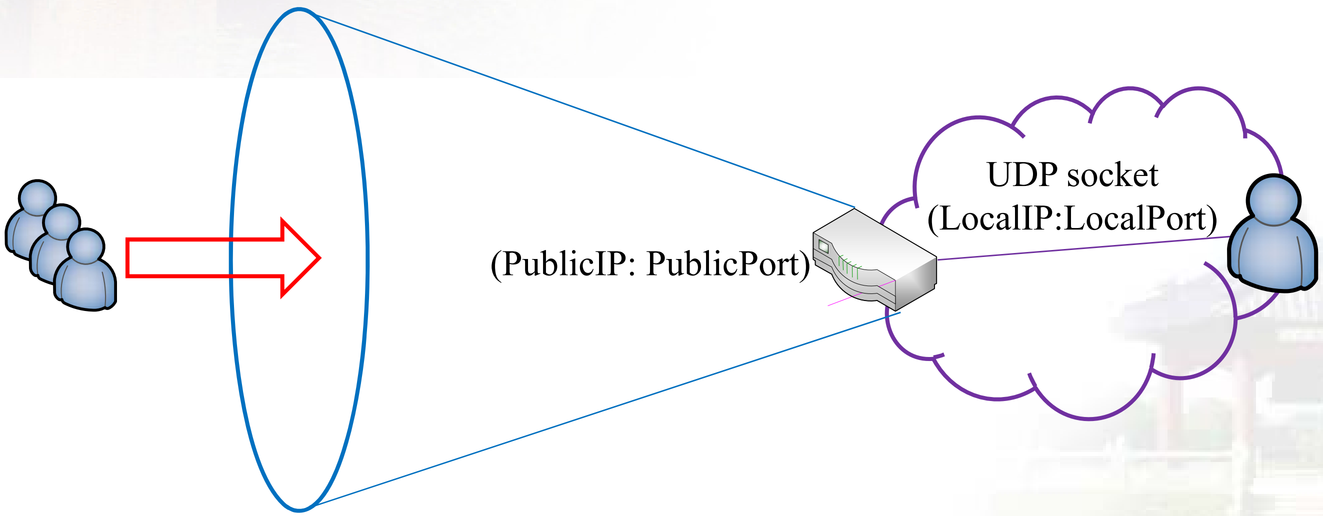
How to make a hole

- The firewall refuses incoming TCP conn.
- the firewall refuses the first incoming UDP.
- If A and B locate behind their NAT, respectively, how to communicate directly via a hole?

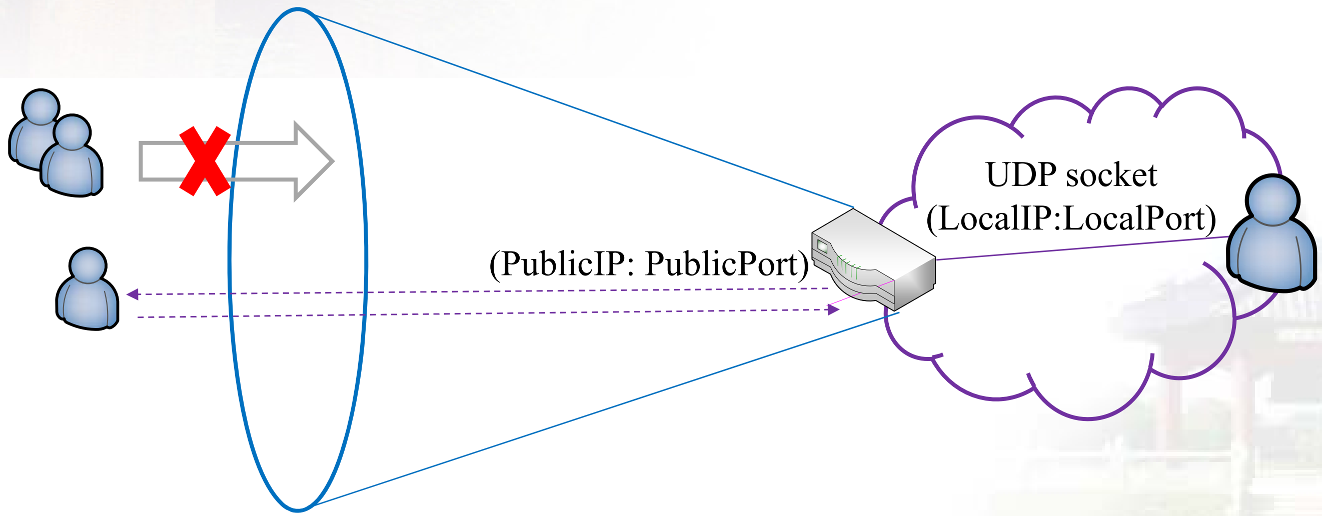


打洞方法:

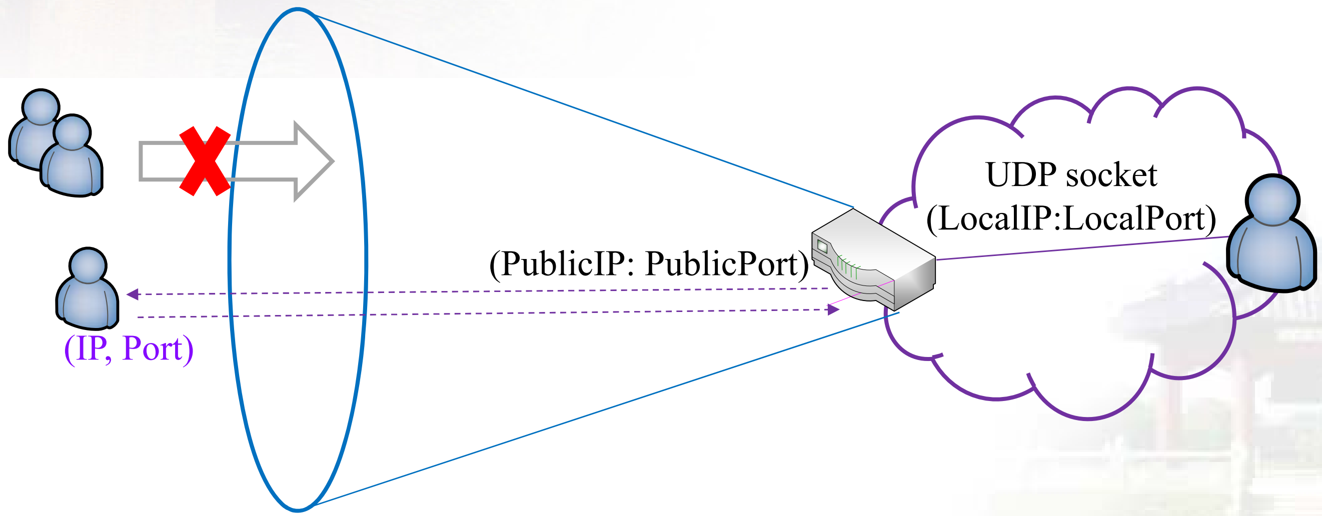
- **Full Cone(全锥型)**
  - 内网主机建立一个**UDP socket(LocalIP:LocalPort)** 第一次使用这个**socket**给外部主机发送数据时**NAT**会给其分配一个公网**(PublicIP:PublicPort)**, 以后用这个**socket**向外面任何主机发送数据都将使用这对**(PublicIP:PublicPort)**。此外，任何外部主机只要知道这个**(PublicIP:PublicPort)**就可以发送数据给**(PublicIP:PublicPort)**，内网的主机就能收到这个数据包.



- **Restricted Cone**
  - 内网主机建立一个**UDP socket(LocalIP:LocalPort)** 第一次使用这个**socket**给外部主机发送数据时**NAT**会给其分配一个公网**(PublicIP:PublicPort),** 以后用这个**socket**向外面任何主机发送数据都将使用这对**(PublicIP:PublicPort)**。此外，如果任何外部主机想要发送数据给这个内网主机，只要知道这个**(PublicIP:PublicPort)**并且内网主机之前曾用这个**socket**向这个外部主机**IP**发送过数据。只要满足这两个条件，这个外部主机就可以用自己的**(IP,** 任何端口**)**发送数据给**(PublicIP:PublicPort)**，内网的主机就能收到这个数据包

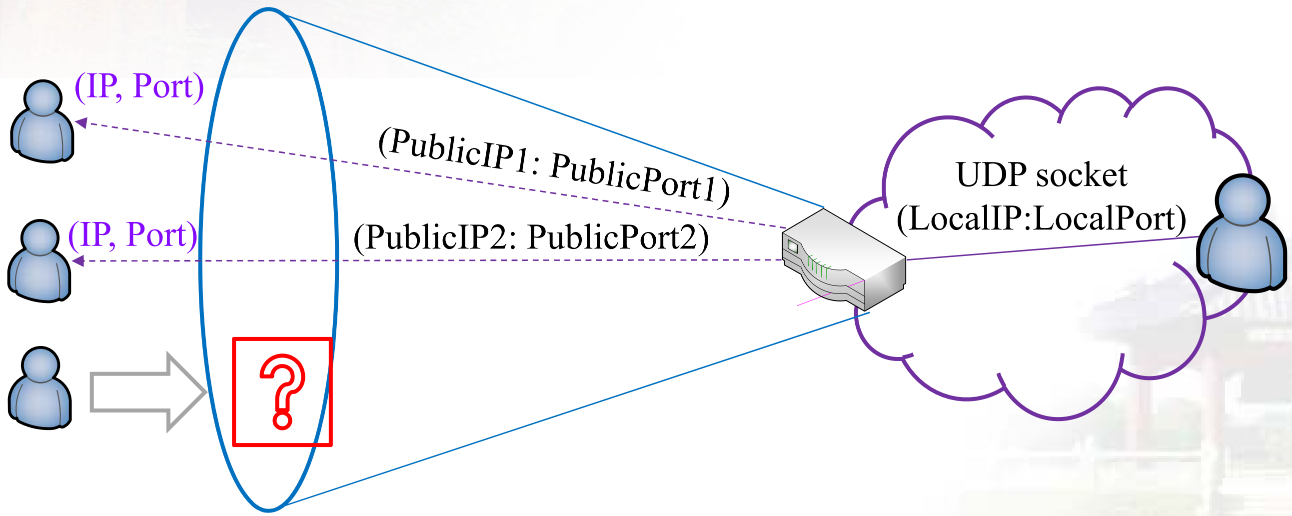


- **Port Restricted Cone**
  - 内网主机建立一个**UDP socket(LocalIP:LocalPort)** 第一次使用这个**socket**给外部主机发送数据时**NAT**会给其分配一个公网**(PublicIP:PublicPort)**,以后用这个**socket**向外面任何主机发送数据都将使用这对**(PublicIP:PublicPort)**。此外，如果任何外部主机想要发送数据给这个内网主机，只要知道这个**(PublicIP:PublicPort)**并且内网主机之前曾用这个**socket**向这个外部主机**(IP, Port)**发送过数据。只要满足这两个条件，这个外部主机就可以用自己的**(IP,Port)**发送数据给**(PublicIP:PublicPort)**，内网的主机就能收到这个数据包.



- **Symmetric（对称形）**

  - **内网主机建立一个UDP socket(LocalIP, LocalPort),当用这个socket第一次发数据给外部主机1时,NAT为其映射一个(PublicIP1,Port1),以后内网主机发送给外部主机1的所有数据都是用这个(PublicIP1,Port1)； 如果内网主机同时用这个socket给外部主机2发送数据，第一次发送时，NAT会为其分配一个(PublicIP2,Port2), 以后内网主机发送给外部主机2的所有数据都是用这个(PublicIP2,Port2).**



Chapter 4: outline

IPv6: motivation

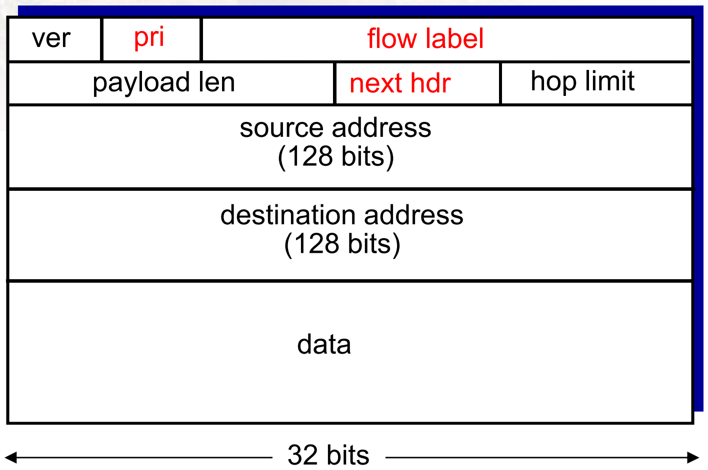
- *initial motivation:* **32-bit address space soon to be completely allocated.**
- **additional motivation:**
  - **header format helps speed processing/forwarding**
  - **header changes to facilitate QoS**

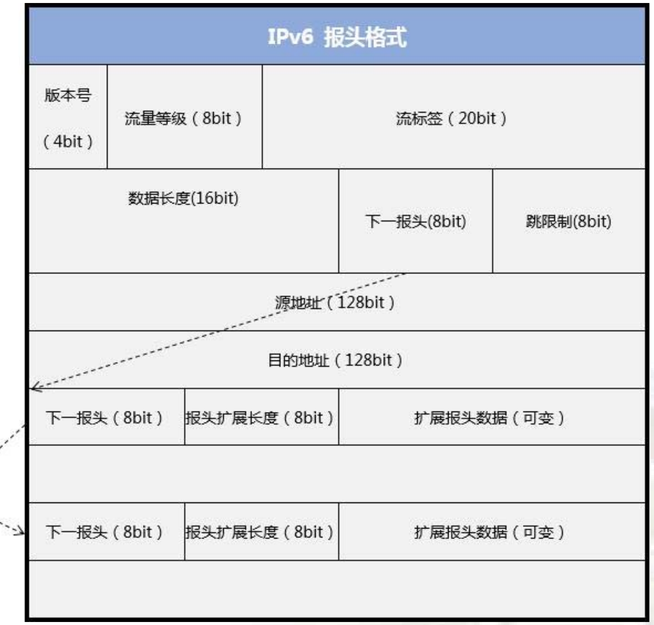
IPv6 datagram format:

- **fixed-length 40 byte header**
- **no fragmentation allowed**

IPv6 datagram format

priority: 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



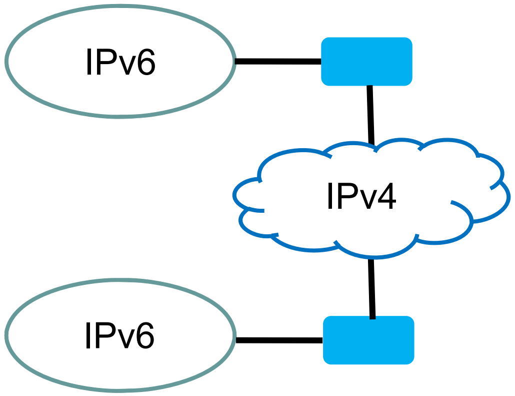


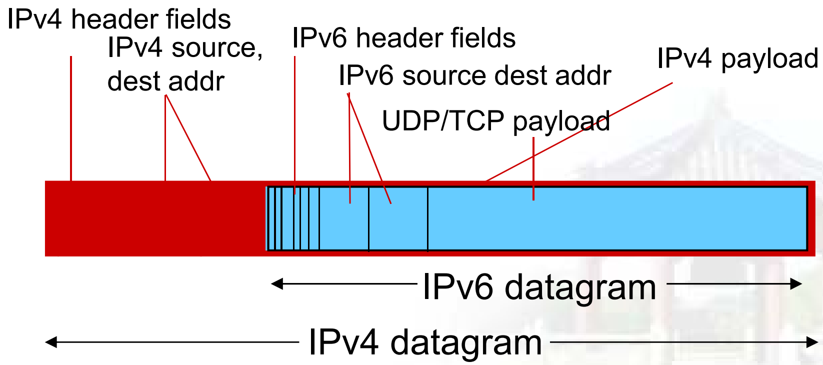
Other changes from IPv4

- ***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

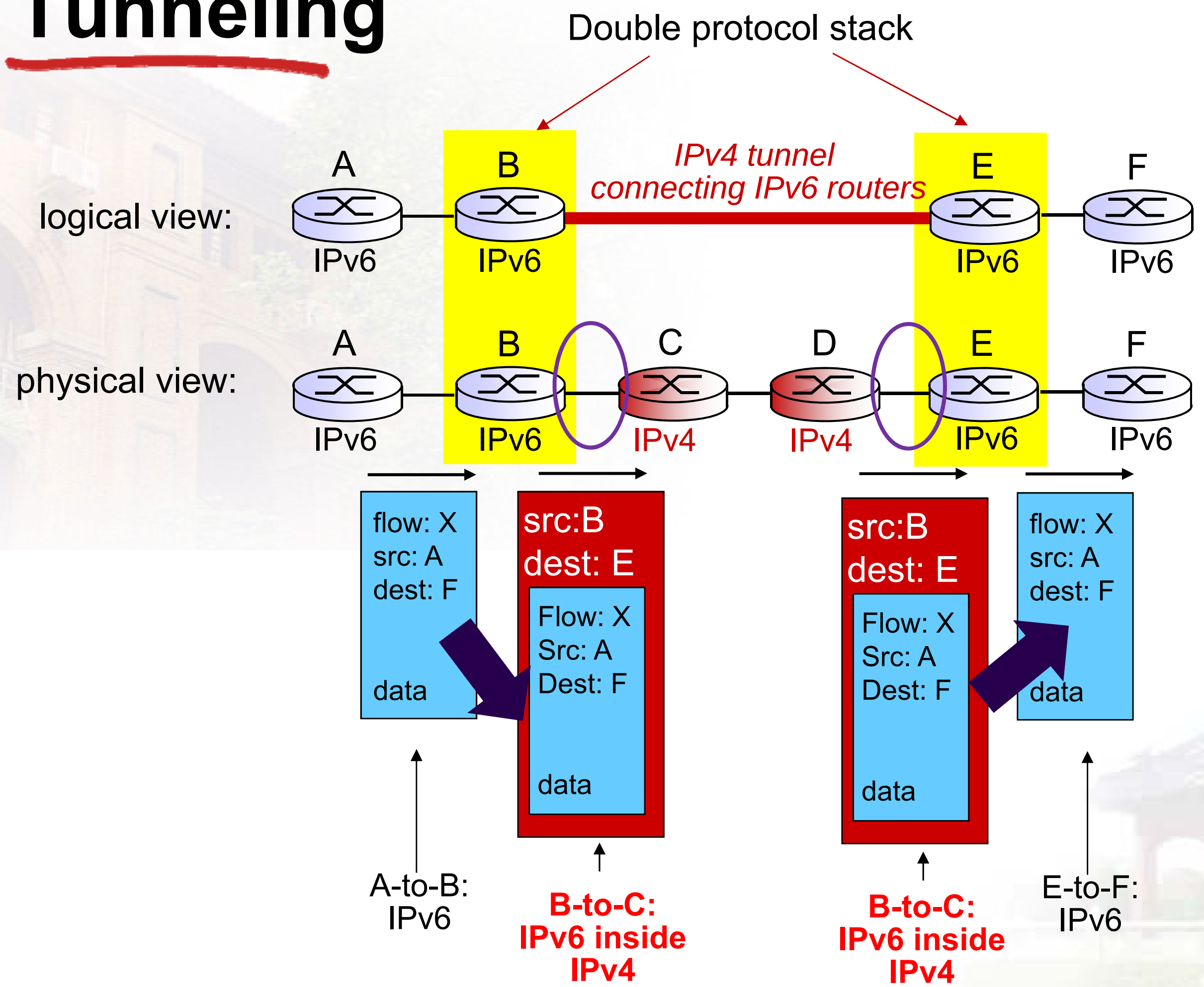
Transition from IPv4 to IPv6

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





Tunneling



4-87

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

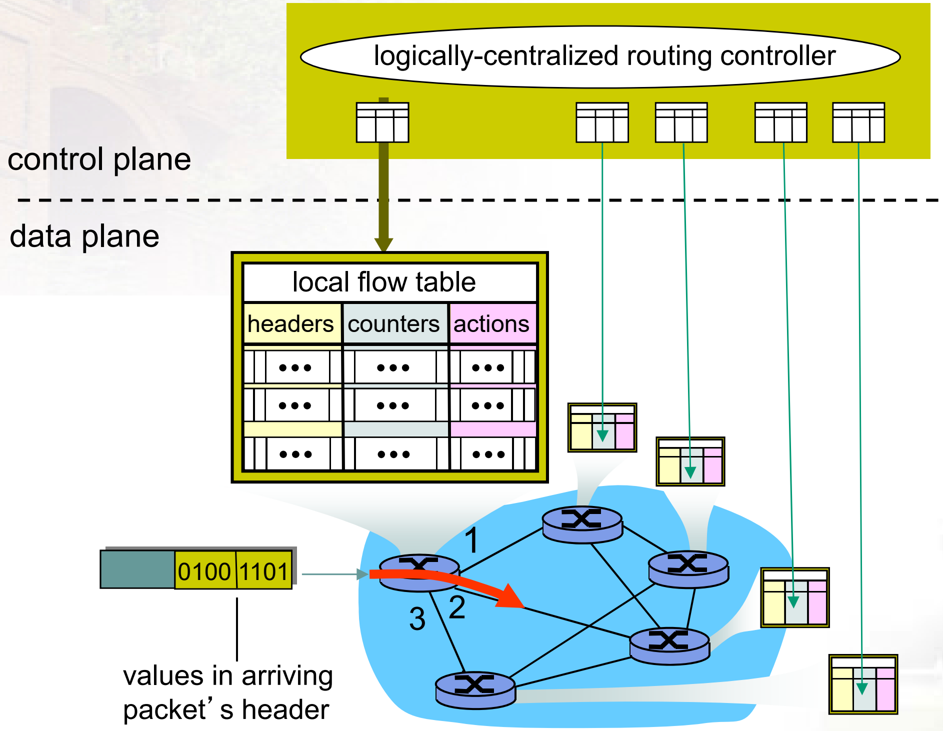
4.2 What’ s inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

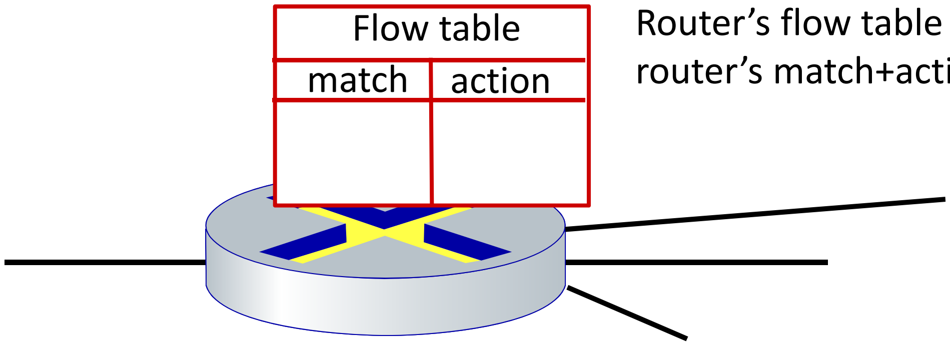
Generalized Forwarding and SDN

Each router contains a *flow table* that is computed and distributed by a *logically centralized* routing controller



Flow table abstraction

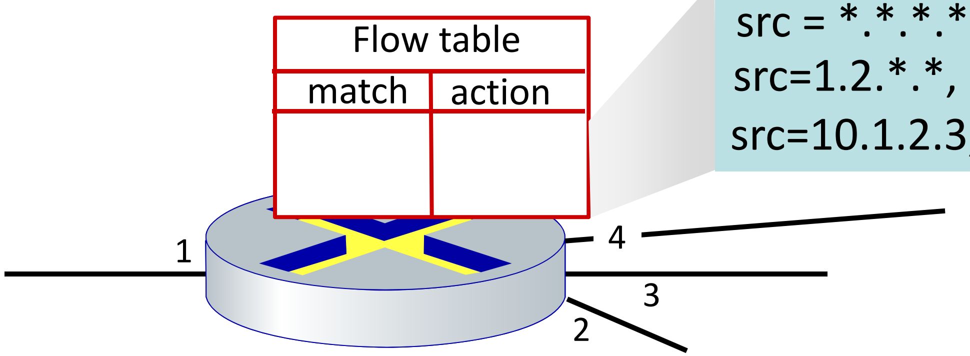
- **flow:** defined by header field values (in link-, network-, transport-layer fields)
- **generalized forwarding:** simple packet-handling rules
  - **match:** pattern values in packet header fields
  - **actions:** for matched packet: drop, forward, modify, matched packet or send matched packet to 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

Rule	Action	Stats

Packet + byte counters

1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields

Switch Port	VLAN ID	MAC src	MAC dst	Eth type	IP Src	IP Dst	IP Prot	TCP sport	TCP dport

Link layer | Network layer | Transport layer

Examples

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	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	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

do not forward (block) all datagrams destined to TCP port 22

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	128.119.1.1	*	*	*	*	drop

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

Examples

Destination-based layer 2 (switch) forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	22:A7:23:11:E1:02	*	*	*	*	*	*	*	*	port3

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

OpenFlow abstraction

- *match+action:* unifies different kinds of devices

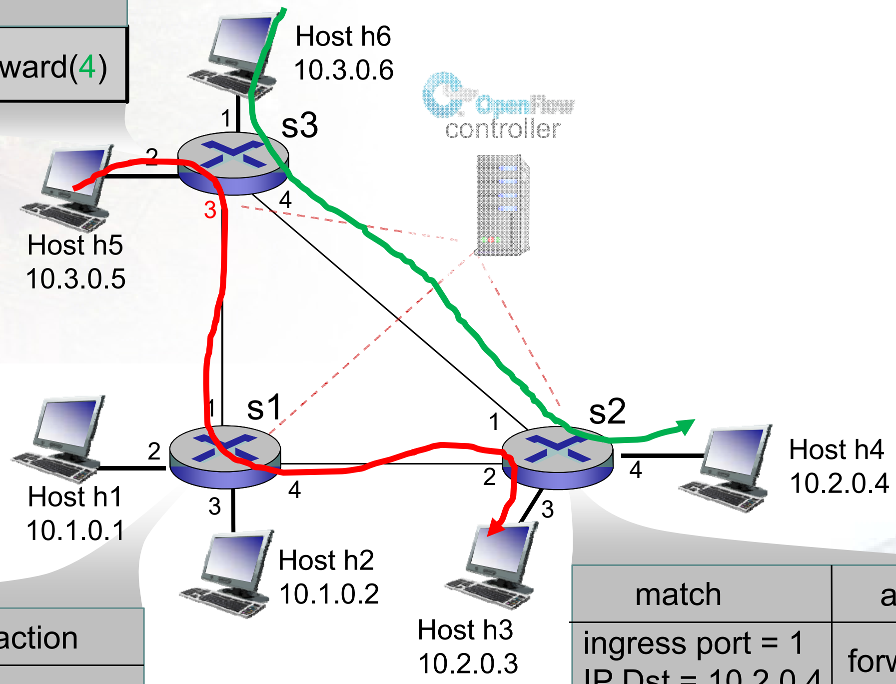
- **Router**
  - ***match:*** **longest destination IP prefix**
  - ***action:*** **forward out a link**
- **Switch**
  - ***match:*** **destination MAC address**
  - ***action:*** **forward or flood**

- **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

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

match	action
IP Src = 10.3.0.5 IP Dst = 10.2.*.*	forward(3)
IP Src = 10.3.0.6 IP Dst = 10.2.*.*	forward(4)

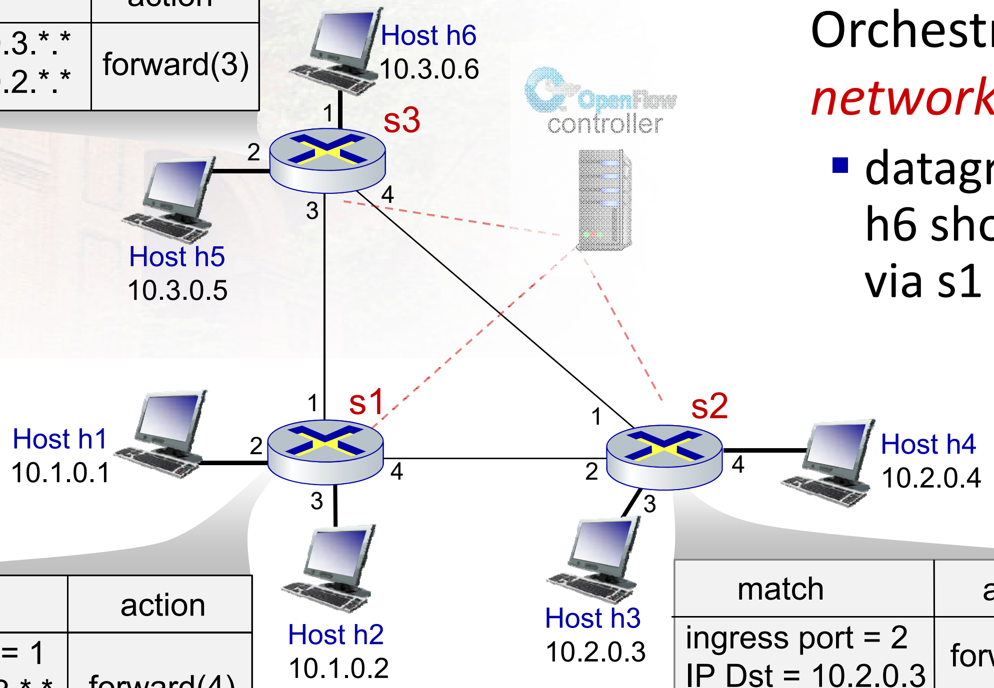


match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 1 IP Dst = 10.2.0.4	forward(4)
ingress port = 2 IP Dst = 10.2.0.3	forward(3)

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

Middleboxes

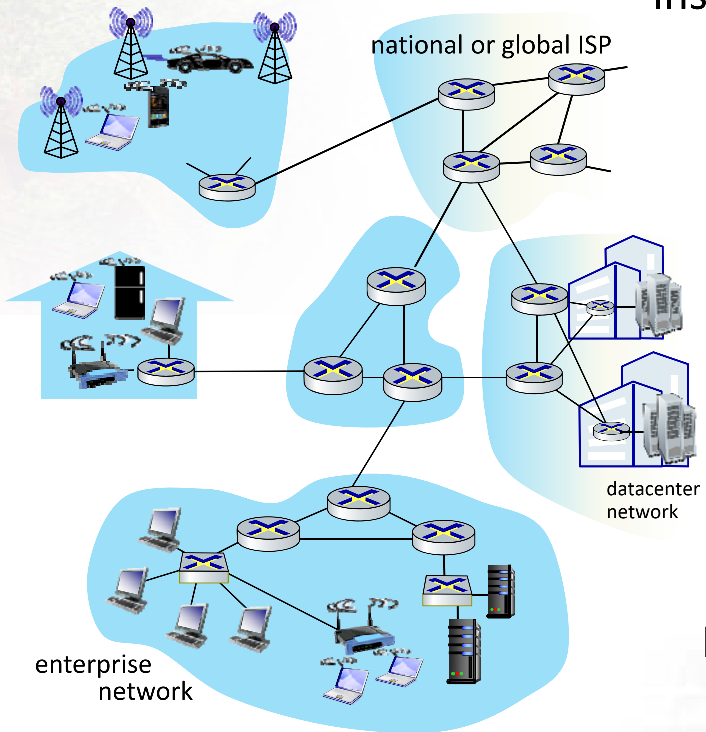
Middlebox (RFC 3234)

“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

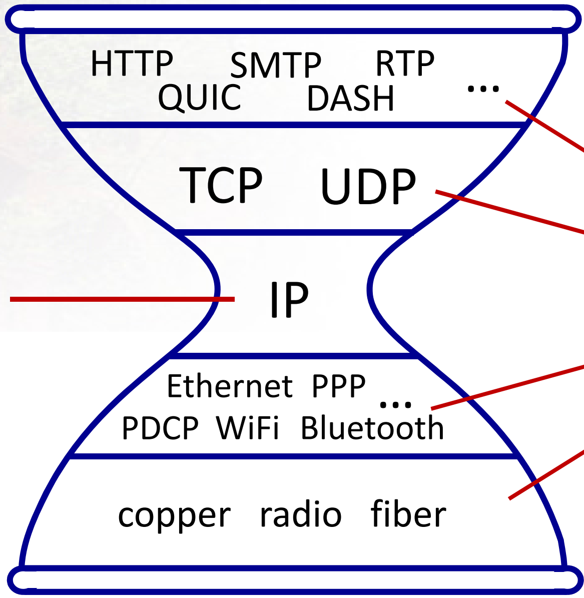
Middleboxes

- 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

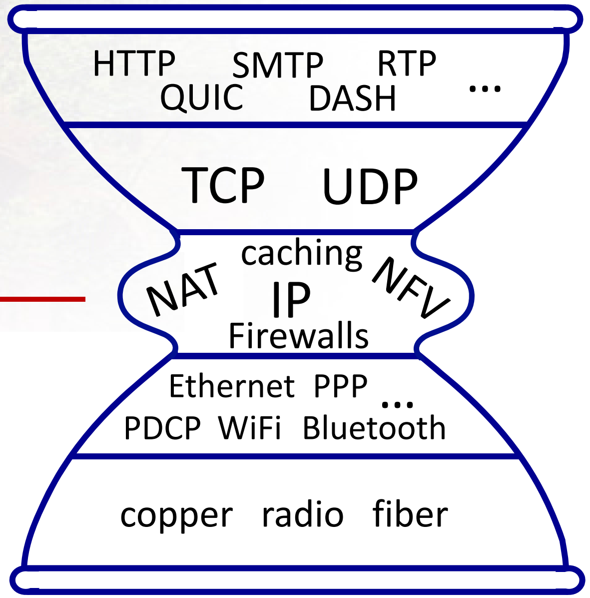


many protocols in physical, link, transport, and application layers

The IP hourglass, at middle age

Internet's middle age "love handles"?

- middleboxes, operating inside the network



Architectural Principles of the Internet

RFC 1958

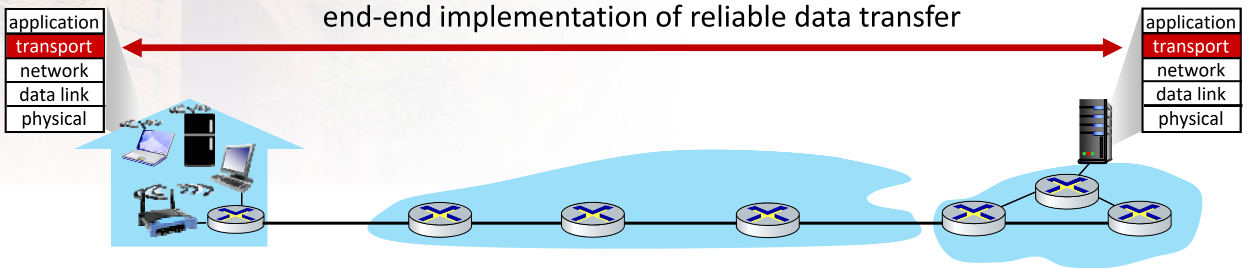
"Many members of the Internet community would argue that there is no architecture, but only a tradition, which was not written down for the first 25 years (or at least not by the IAB). However, in very general terms, the community believes that the goal is connectivity, the tool is the Internet Protocol,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

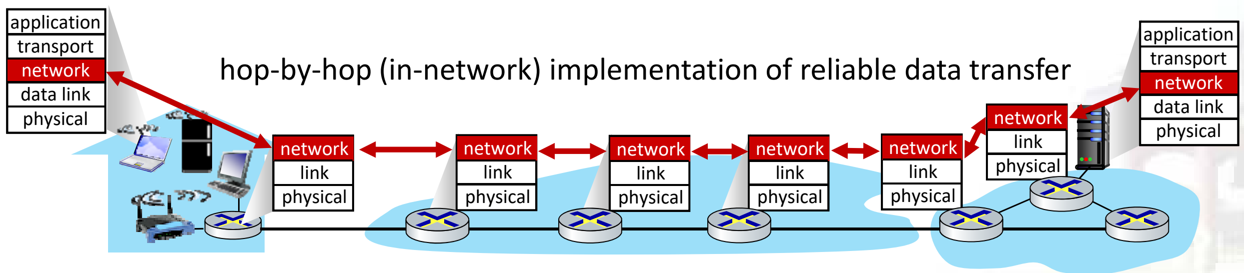
Three cornerstone beliefs:

- 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





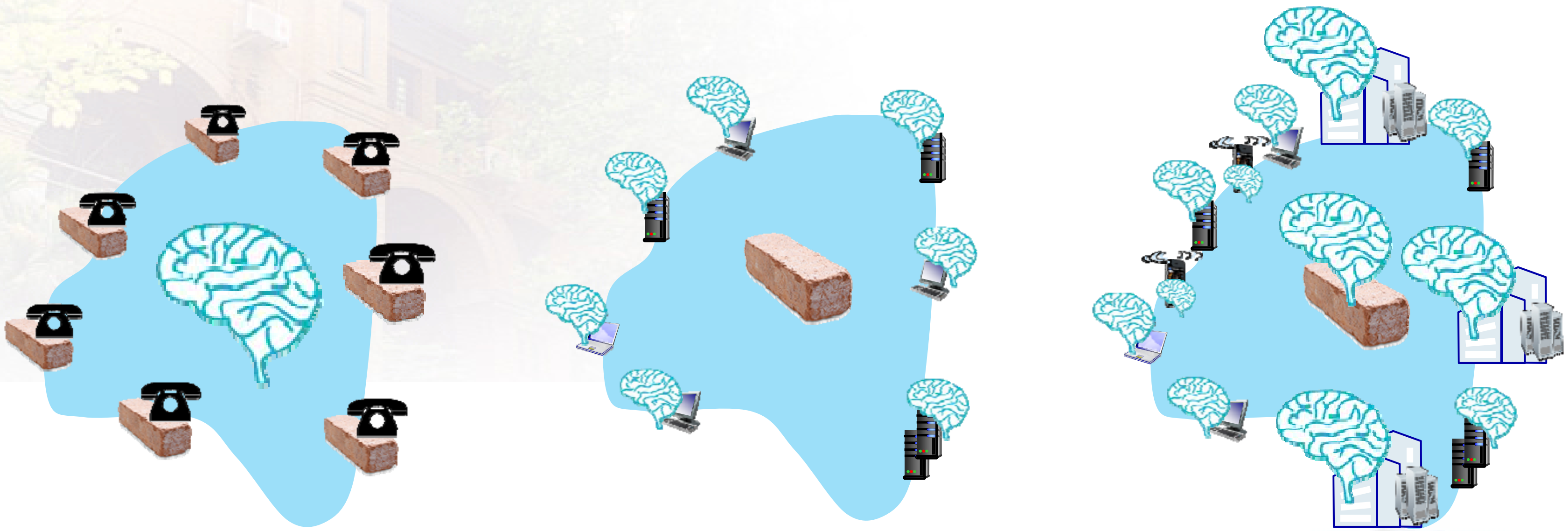
The end-end argument

- 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."

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!*

4.1 Overview of Network layer: data plane and control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- NAT
- IPv6

4.4 Generalized Forward and SDN
- match plus action
- OpenFlow example

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

Answer: **by the control plane (next chapter)**

Thanks

Q & A

Email: xieyi5@mail.sysu.edu.cn
https://cse.sysu.edu.cn/content/2462