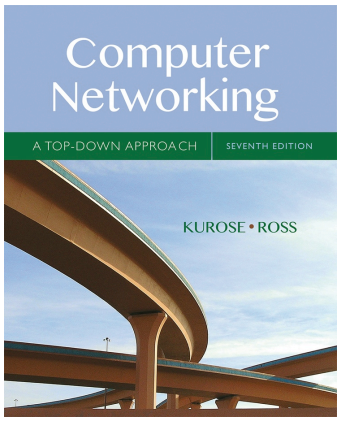
Chapter 4: Network Layer



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

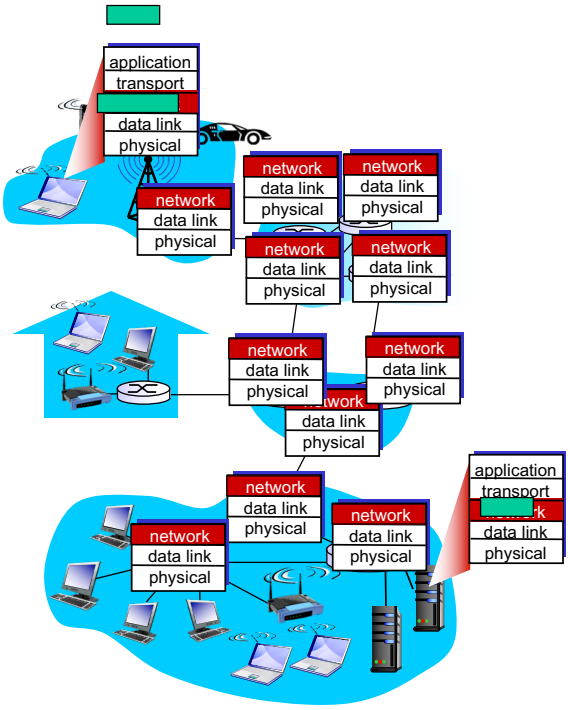
Computer Networking: A Top Down Approach
7th edition
Jim Kurose, Keith Ross
Pearson/Addison-Wesley
April 2016

Chapter 4: Network Layer: Data Plane

- 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 Forwarding and SDN
  - Match
  - Action
  - OpenFlow examples of match-plus-action in action

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

Network layer functions:

- *forwarding:* move packets from router's input link interface to appropriate router output link interface.
- *routing:* determine route taken by packets from source to destination.
  - *Routing algorithms*

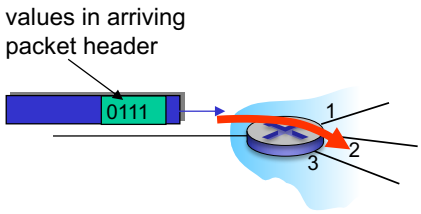
Analogy:

- routing: process of planning trip from source to destination
- 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

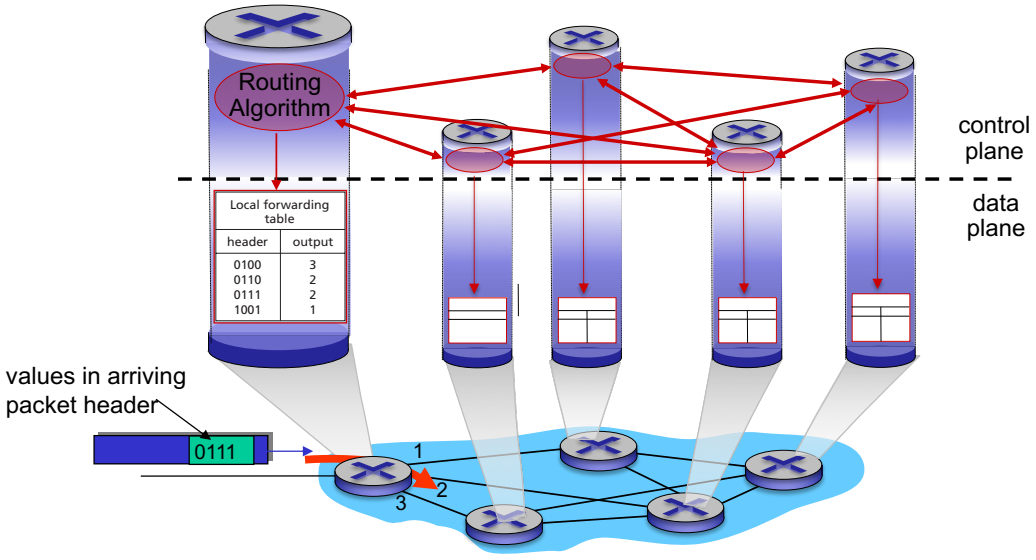


Control Plane

- 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

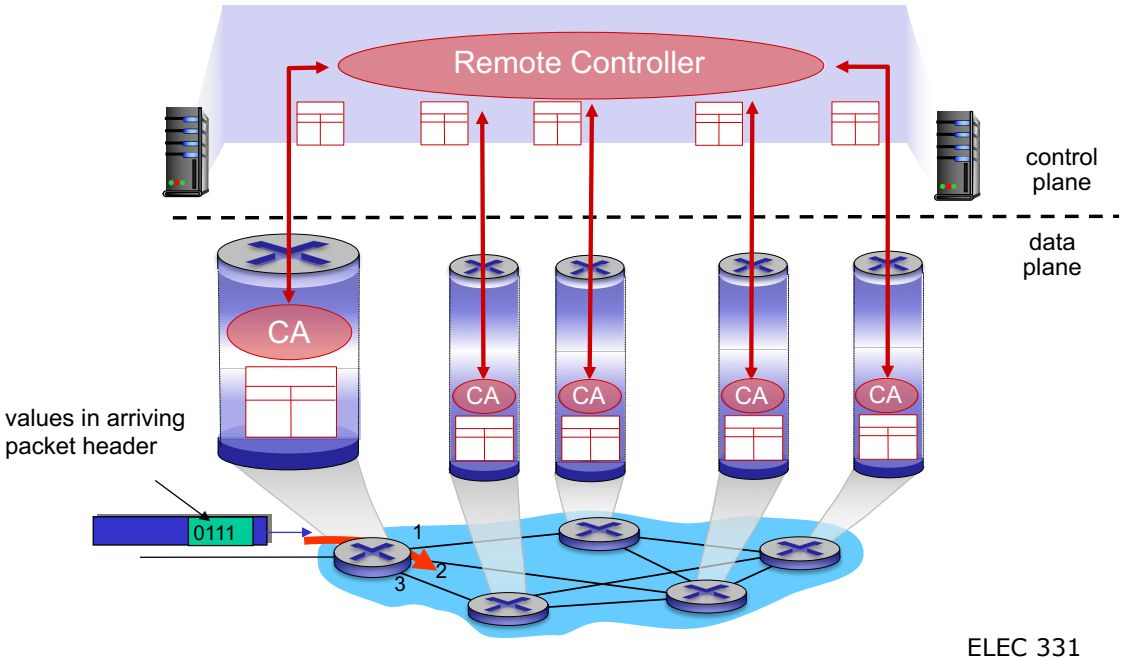
Per-Router Control Plane

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 packets:

- Guaranteed delivery
- Guaranteed delivery with less than 100 msec delay

Example services for a flow of packets:

- In-order datagram delivery
- Guaranteed minimum bandwidth to flow
- Restrictions on changes in inter-packet spacing
- Security services: confidentiality of data via encryption; source authentication.

Network Layer Service Models

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

ATM: Asynchronous Transfer Mode

CBR: Constant Bit Rate

ABR: Available Bit Rate

Other alternatives do exist

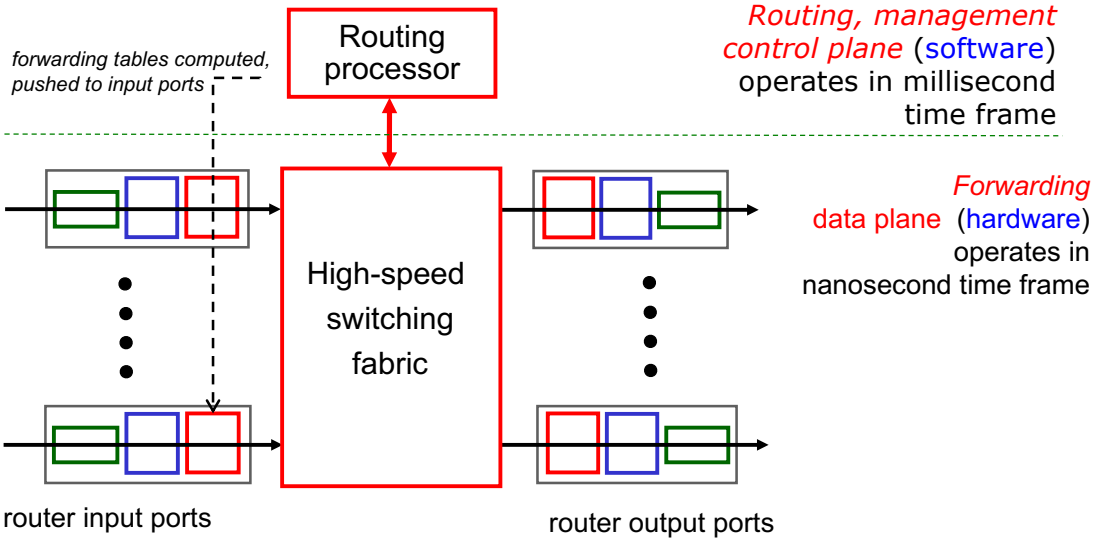
Chapter 4: Network Layer

- 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 Forwarding and SDN
  - Match
  - Action
  - OpenFlow examples of match-plus-action in action

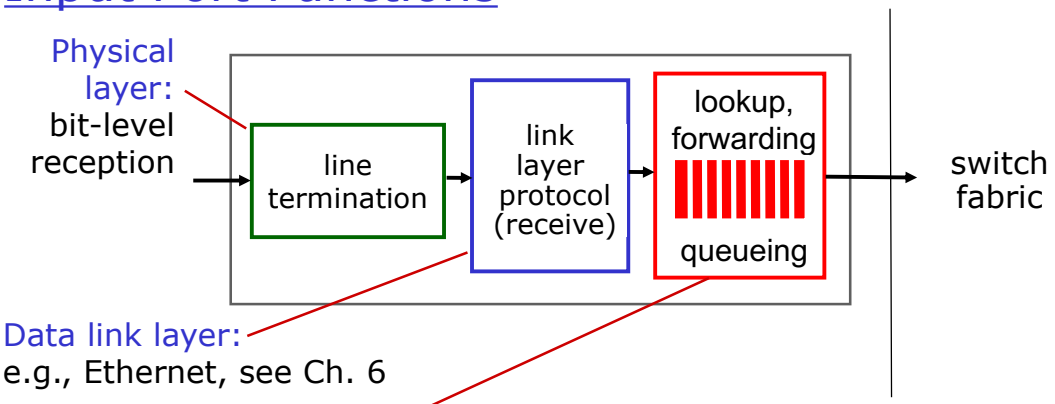
Router Architecture Overview

Two key router functions:

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



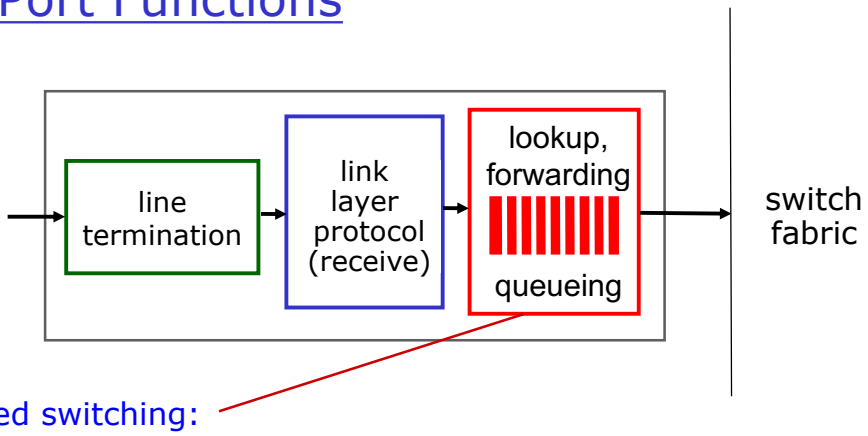
Input Port Functions



Decentralized switching:

- given datagram destination address, lookup output port using forwarding table in input port memory
- goal: complete input port processing at 'line speed'
- e.g., 10 Gbps input link, 64 byte IP datagram, input port has 51.2 ns to process the datagram before next one arrives.
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Input Port Functions



Decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory ("match plus action")
- destination-based forwarding: forward based only on destination IP address (traditional)
- generalized forwarding: forward based on any set of header field values

Destination-based Forwarding

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?

Longest Prefix Matching Rule

longest prefix matching

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

Prefix Match	Link interface
11001000 00010111 00010*** *********	0
11001000 00010111 00011000 *********	1
11001000 00010111 00011*** *********	2
otherwise	3

examples:

Dest Addr: 11001000 00010111 00010110 10100001 which interface?

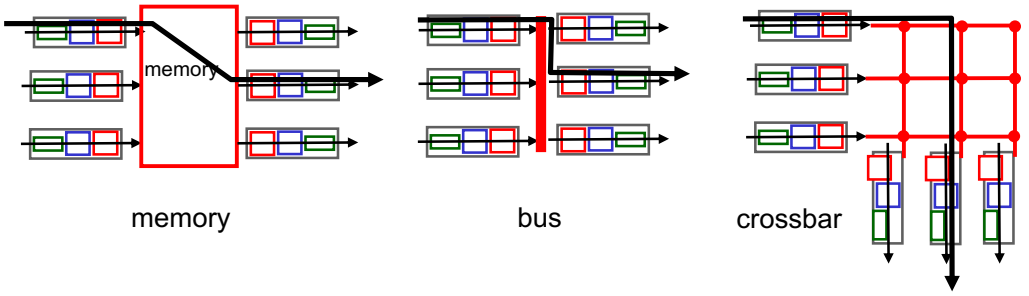
Dest Addr: 11001000 00010111 00011000 10101010 which interface?

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 6500 and 7600 Series routers: can up ~1M forwarding table entries in TCAM

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 fabric

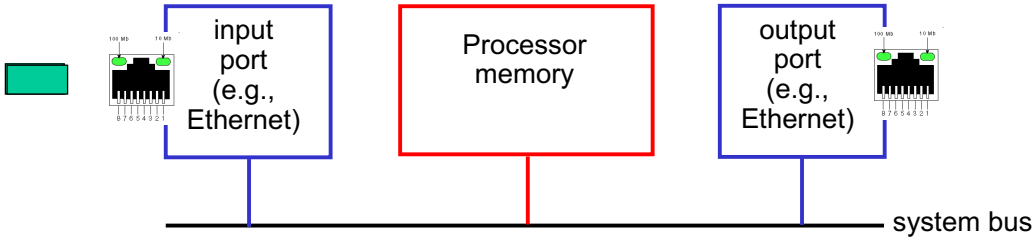


Analogy: Roundabout

Switching Via Memory

First generation routers:

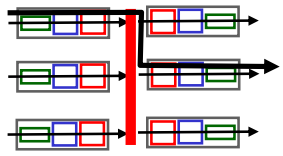
- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)



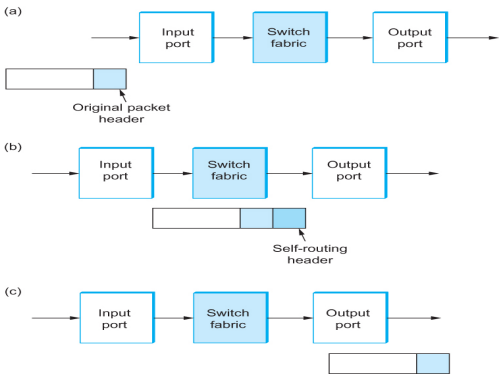
Modern routers: Lookup and storing of packet is performed by processors on the input line cards (e.g., Cisco's Catalyst 8500 series switches).

Switching Via a Bus

- datagram from input port memory to output port memory via a shared bus
- Input port pre-pend a switch-internal label (header) to packet indicating the local output port
- Only one packet at a time can be transferred over the bus
- bus contention: switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 6500: sufficient speed for access and enterprise routers

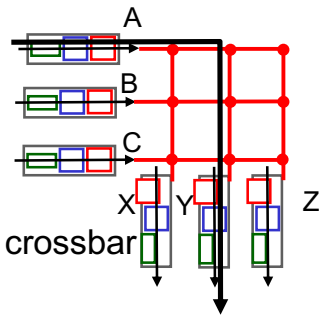


bus



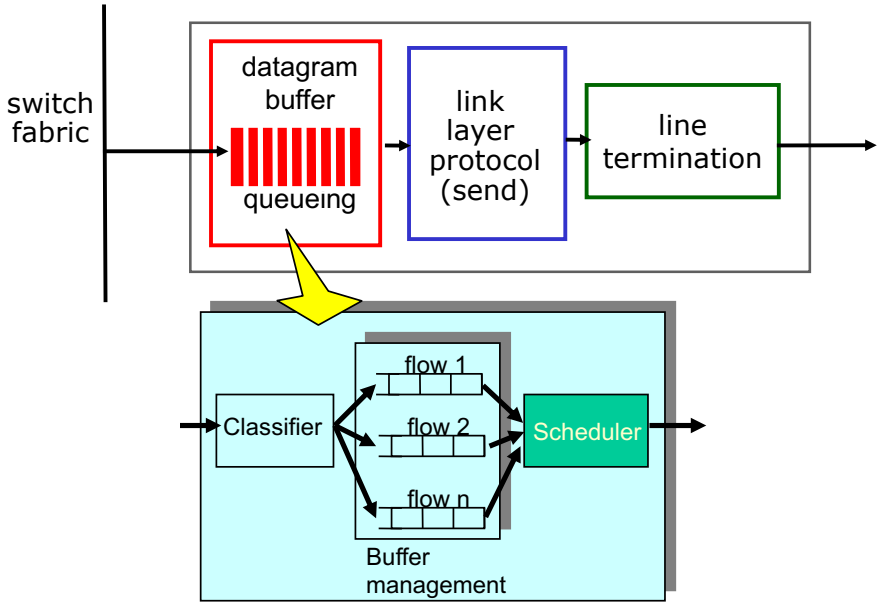
Switching Via an Interconnection Network

- Overcome bus bandwidth limitations
- Crossbar switch: an interconnection network consisting of 2*N* buses that connect *N* input and *N* output ports.
- Each vertical bus intersects each horizontal bus at a crosspoint, which can be opened or closed at any time by the controller.



- Capable of forwarding multiple packets in parallel.
- Advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through interconnection network

Output Ports

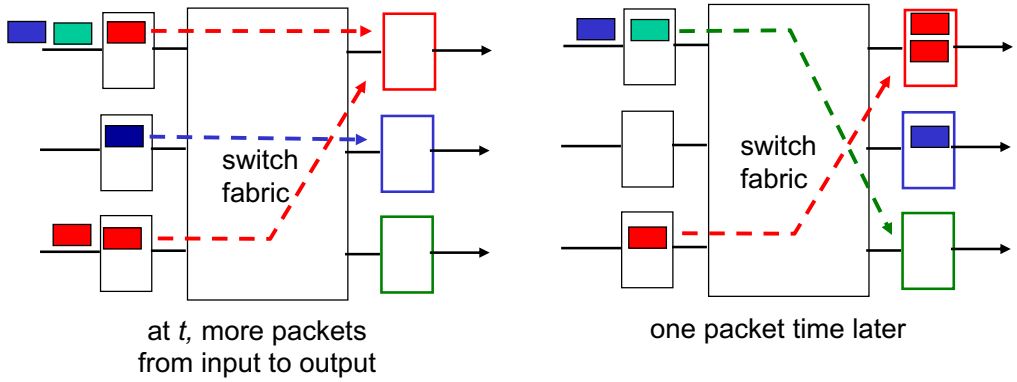


- *buffering* required when faster than the transmi
- *scheduling di* for transmiss

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

Priority scheduling – who gets best performance, network neutrality

Output Port Queueing



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

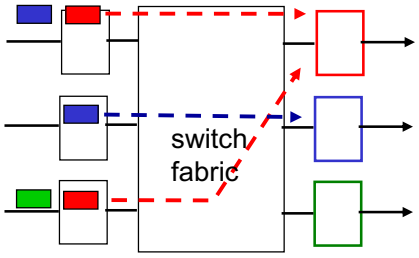
How much buffering is required?

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

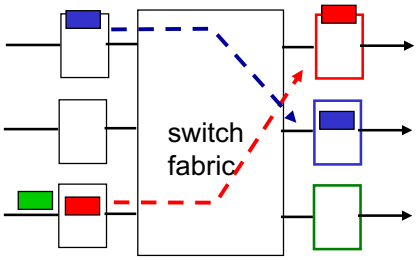
$$\frac{RTT \cdot C}{\sqrt{N}}$$

Input Port Queuing

- Switch fabric is *slower* than input ports combined. Multiple pkts can be transferred in parallel, as long as output ports are different. Fabric can only transfer one pkt to a given output port at a time.
- *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

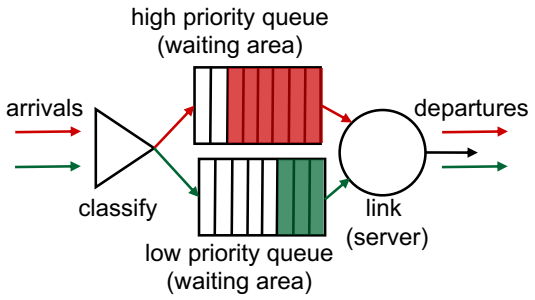


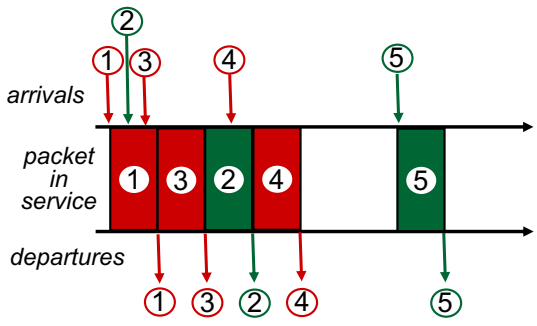
one packet time later:
green packet
experiences HOL
blocking

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?

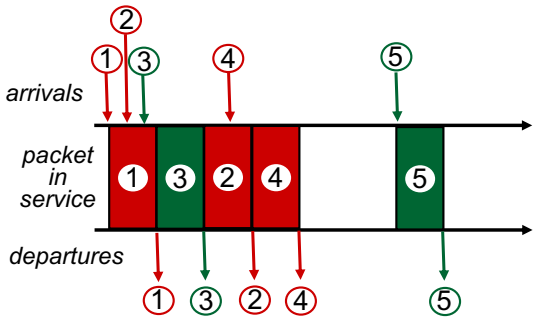




Scheduling Policies: Still More

Round Robin (RR) scheduling:

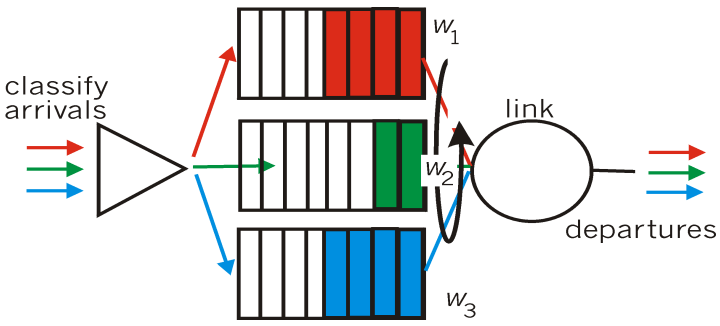
- multiple classes
- cyclically scan class queues, sending one complete packet from each class (if available)
- real world example?



Scheduling Policies: Still More

Weighted Fair Queuing (WFQ):

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

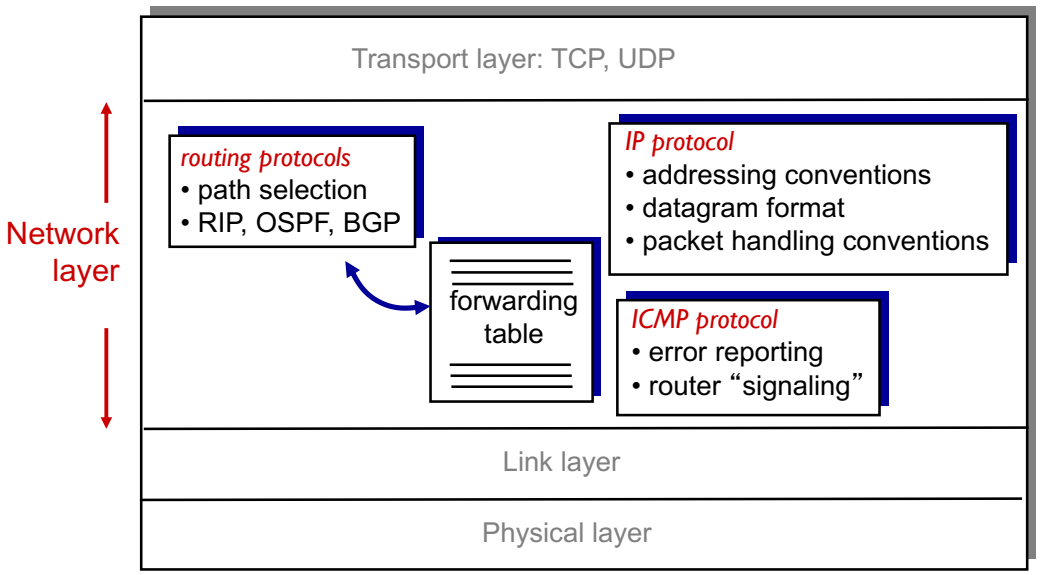


Chapter 4: Network Layer

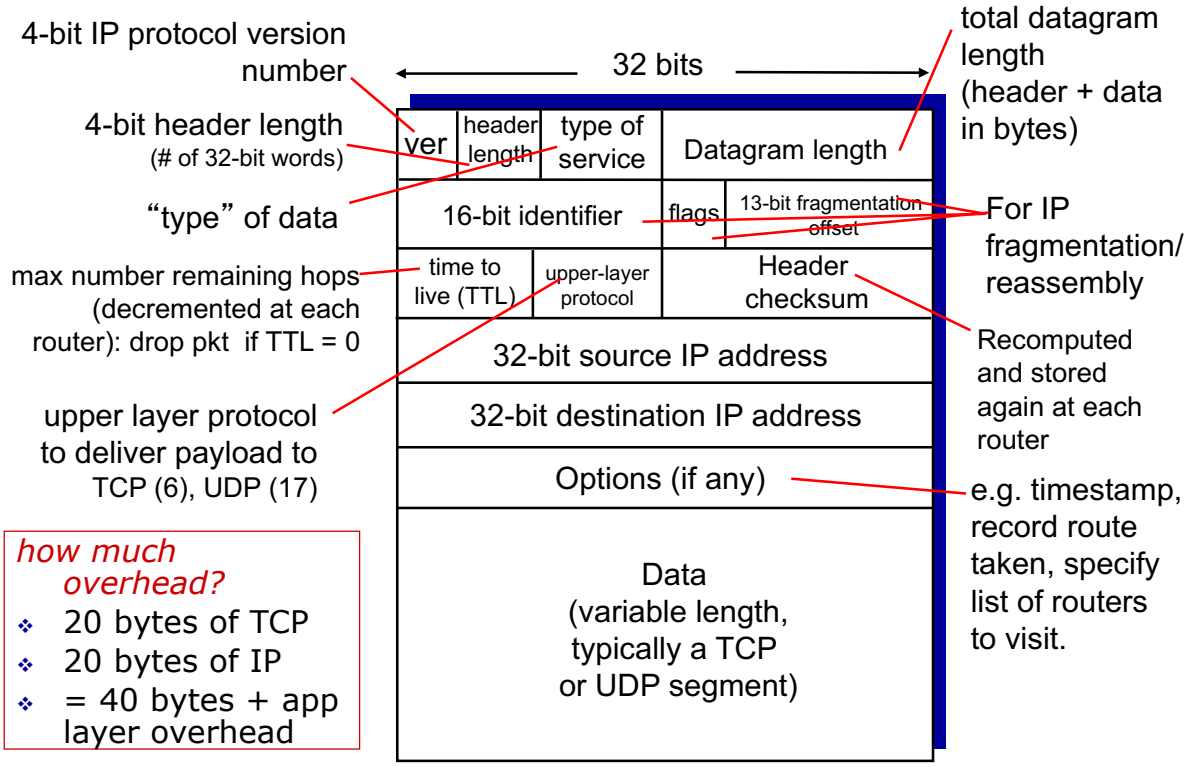
- 4.1 Overview of Network Layer
  - Data plane
  - Control plane
- 4.2 What's inside a router
- 4.3 IP: Internet Protocol
  - Datagram format
  - Fragmentation
  - IPv4 addressing
  - Network address translation
  - IPv6
- 4.4 Generalized Forwarding and SDN
  - Match
  - Action
  - OpenFlow examples of match-plus-action in action

The Internet Network Layer

Host, router network layer functions:



IPv4 Datagram Format

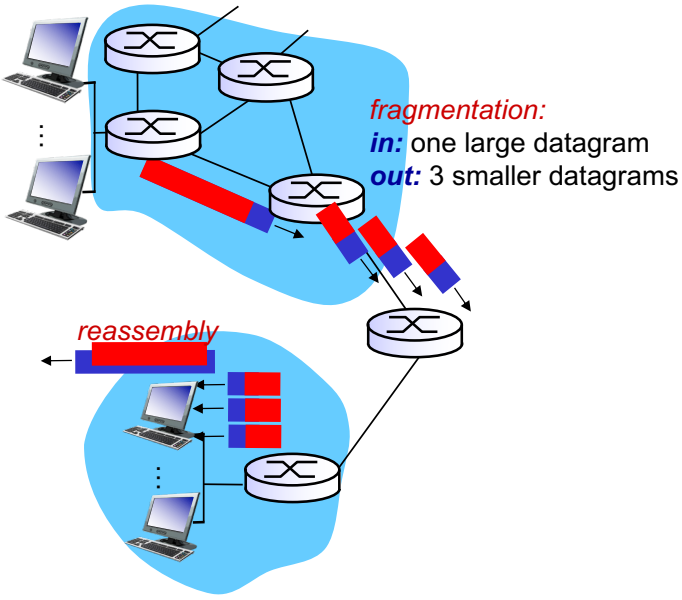


how much overhead?

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

IP Fragmentation & Reassembly

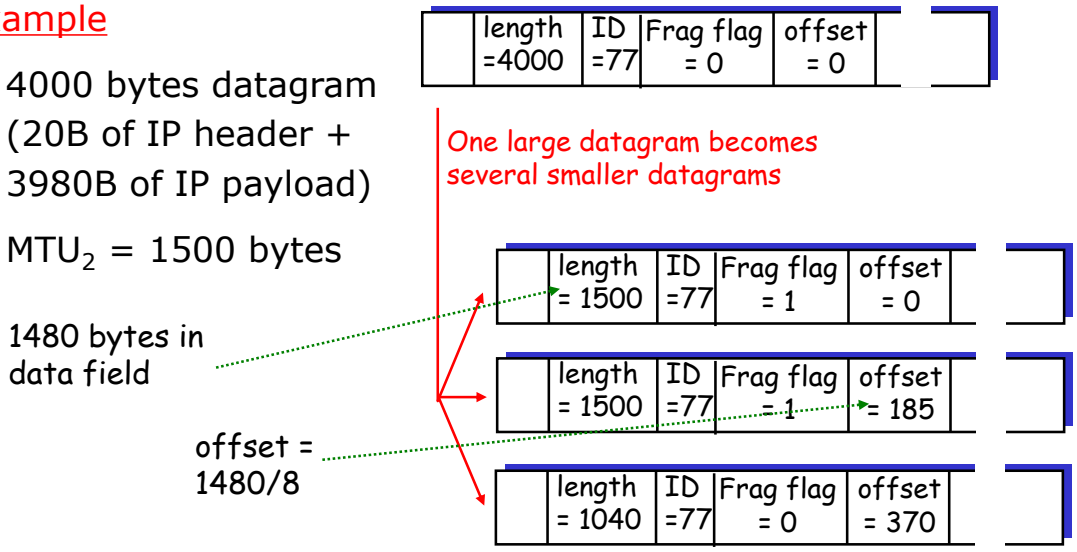
- network links have MTU (Maximum Transmission Unit) - largest possible data portion in link-layer frame.
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related fragments



IP Fragmentation and Reassembly

Example

- 4000 bytes datagram (20B of IP header + 3980B of IP payload)
- MTU_2 = 1500 bytes



- Last fragment has flag bit = 0, all other fragments have flag bit = 1.
- Offset value is specified in units of 8-byte chunks.
- Data in all fragments (except last fragment) are multiple of 8 bytes.

Path MTU Discovery

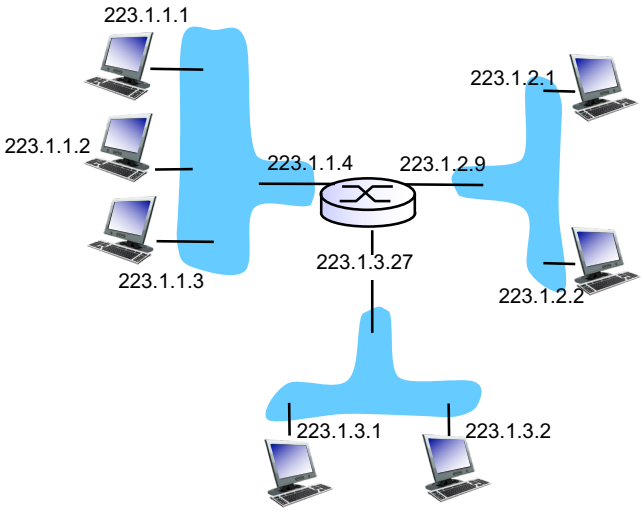
- How can a host choose a datagram size that will not result in fragmentation?
- Path MTU: Minimum MTU along a path from source to destination
- Path MTU Discovery: Process of learning the path MTU
- Iterative Procedure
  - A host sends a sequence of various-size datagrams (with *Don't Fragment* (DF) option bit set) to the destination to see if they arrive without error.
  - If fragmentation is required, the sending host will receive an ICMP error message.
  - Once a datagram is small enough to pass through without fragmentation, datagram size = path MTU.

Chapter 4: Network Layer

- 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 Forwarding 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:* boundary between host/router and physical link
  - router's typically have multiple interfaces
  - host may have multiple interfaces
- IP addresses associated with each interface



Dotted-decimal notation

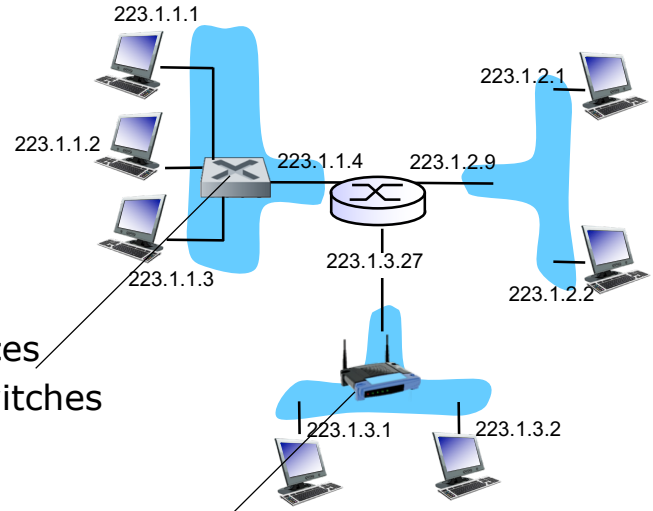
223.1.2.9 = 11011111 00000001 00000010 00001001

223 1 2 9

IP addressing: Introduction

Q: how are interfaces actually connected?

A: we'll learn about that in Chapters 6 and 7.



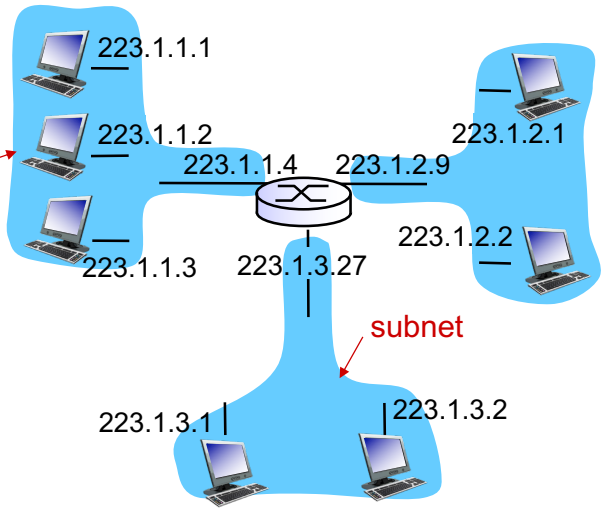
A: wired Ethernet interfaces connected by Ethernet switches

For now: don't need to worry about how one interface is connected to another (with no intervening router)

A: wireless WiFi interfaces connected by WiFi access point

Subnets

- Network interconnecting three host interfaces and one router interface forms a subnet (also called IP network or network).
- *What's a subnet ?*
  - Hosts can physically reach each other without intervening router
- IP address:
  - subnet part (high order bits)
  - host part (low order bits)
- The subnet part defines the subnet address

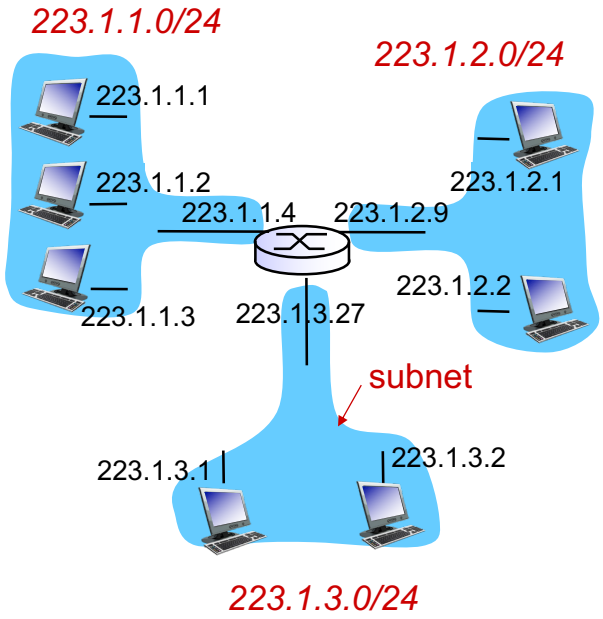


network consisting of 3 subnets

Subnets

Recipe

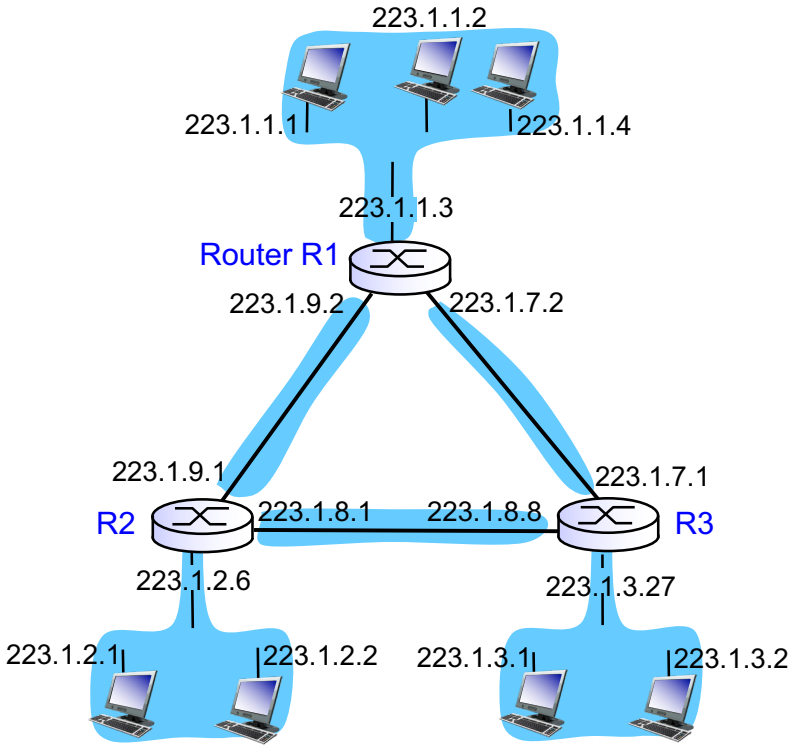
- To determine the subnets, detach each interface from its host or router, creating islands of isolated networks.
- Each isolated network is called a subnet.
- Subnet mask, /24 in this example, indicates the leftmost 24 bits define the subnet address.



subnet mask: /24

Subnets

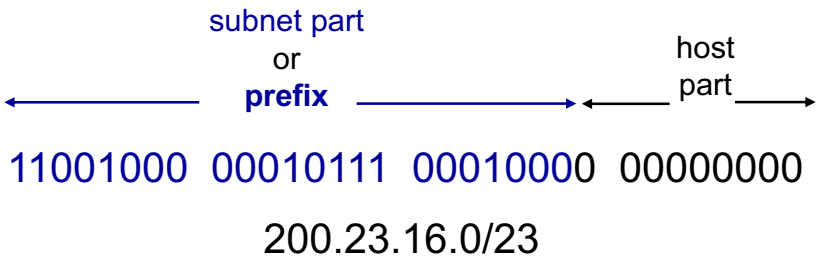
How many?



IP addressing: CIDR

CIDR: Classless InterDomain Routing [RFC 4632]

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



Summary of Special IP Addresses

Prefix	Suffix	Type of Address	Purpose
All-0s	All-0s	This computer	Used during bootstrap
Network	All-0s	Network	Identifies a network
All-1s	All-1s	Limited Broadcast	Broadcast on local network
Network	All-1s	Directed Broadcast	Broadcast on specific network
127/8	Any	Loopback	Testing

IP addressing

Q: How does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers

- allocates addresses to regional Internet registries
- manages DNS root servers
- assigns domain names, resolves disputes

IP addresses: how to get one?

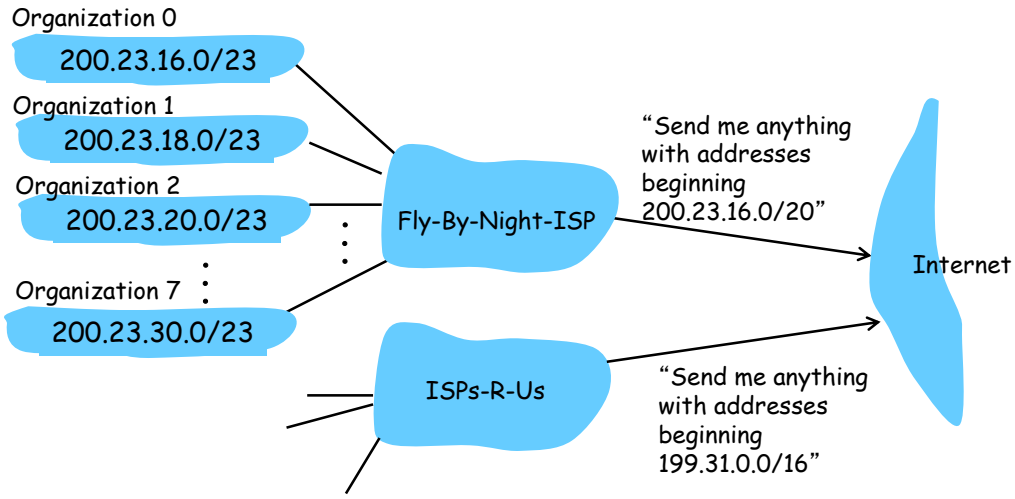
Q: How does *network* get subnet part of IP address?

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

ISP's block	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

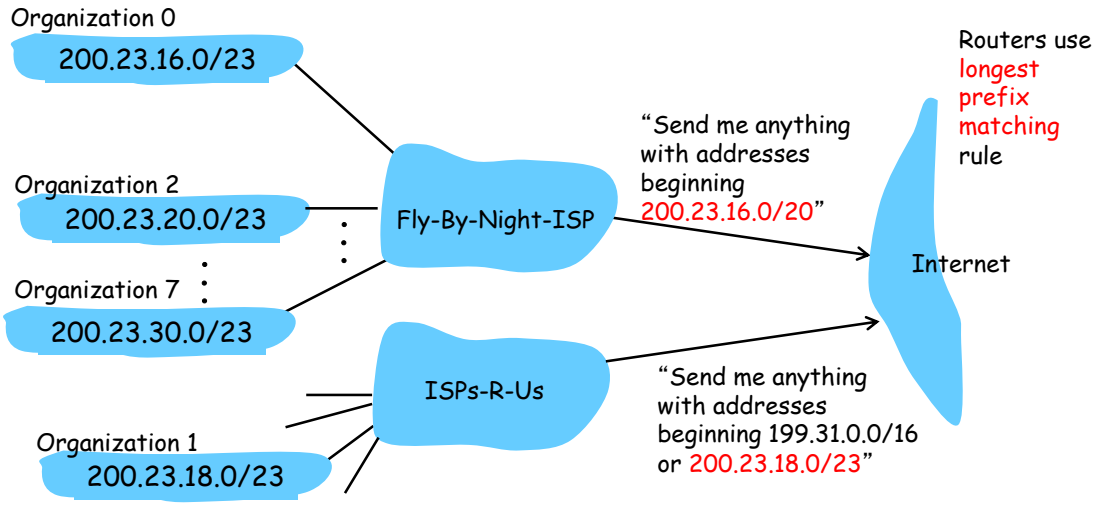
Hierarchical addressing: Route aggregation

Hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

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



Example:

Q: Consider a router that interconnects three subnets: Subnet 1, Subnet 2, and Subnet 3. Suppose all of the interfaces in each of these three subnets are required to have the prefix 223.1.17.0/24. Also suppose that Subnet 1 is required to support up to 125 interfaces, and Subnets 2 and 3 are each required to support up to 60 interfaces. Provide three network addresses (of the form a.b.c.d/x) that satisfy these constraints.

IP addresses: how to get one?

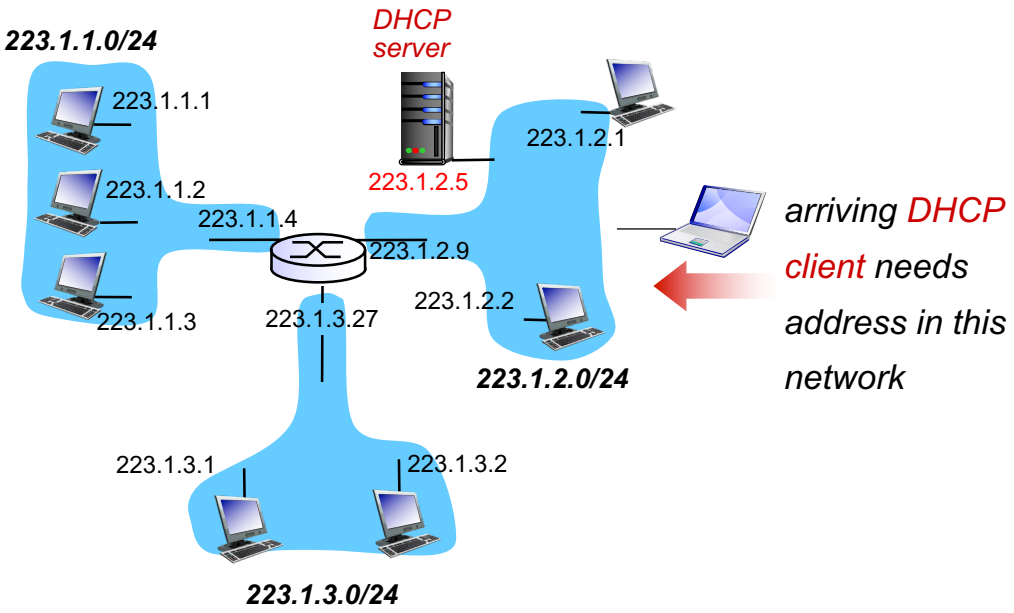
Q: How does *host* get IP address?

- hard-coded by system admin in a file
  - Wintel: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- DHCP: Dynamic Host Configuration Protocol [RFC 2131]: dynamically get address from as server
  - “plug-and-play”

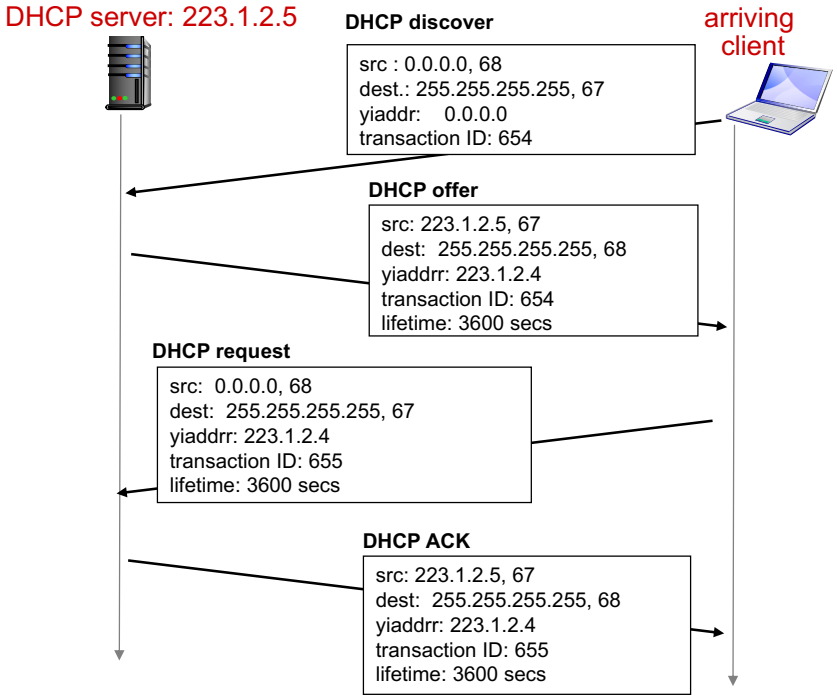
DHCP: Dynamic Host Configuration Protocol

- 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 as "on")
  - Support for mobile users who want to join network
- DHCP overview:
  - host broadcasts "DHCP discover" message
  - DHCP server responds with "DHCP offer" message
  - host requests IP address: "DHCP request" message
  - DHCP server sends address: "DHCP ACK" message

DHCP client-server scenario



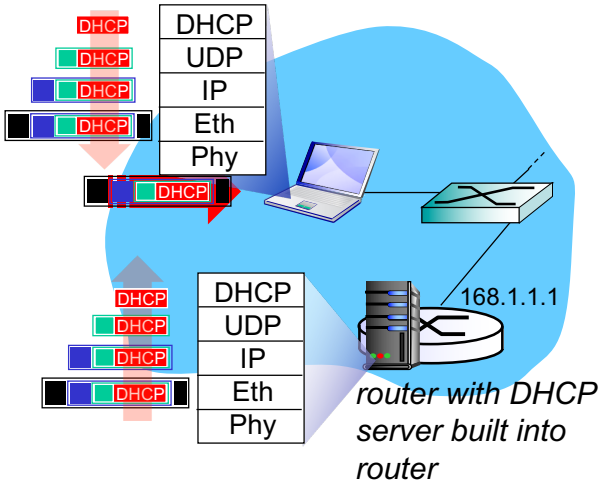
DHCP client-server scenario



DHCP: more than IP Address

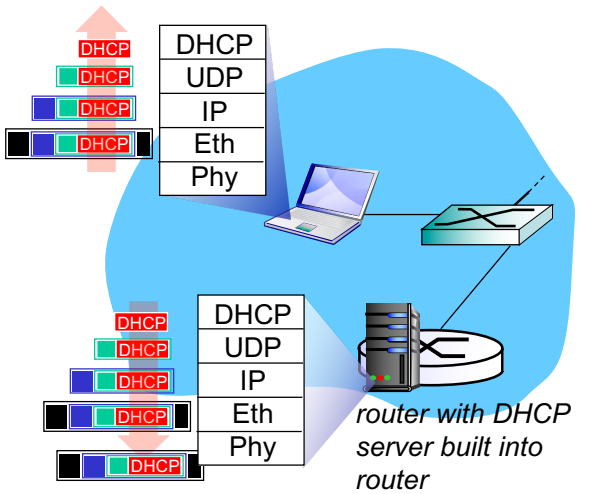
- DHCP can return more than just allocated IP address on subnet:
  - address of first-hop router for client
  - name and IP address of DNS server
  - subnet mask (indicating network versus host portion of address)

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 demux'ed to IP demux'ed, UDP demux'ed to DHCP

DHCP: Example (cont.)



- DHCP server creates 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, demux'ing 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: Wireshark Output (home LAN)

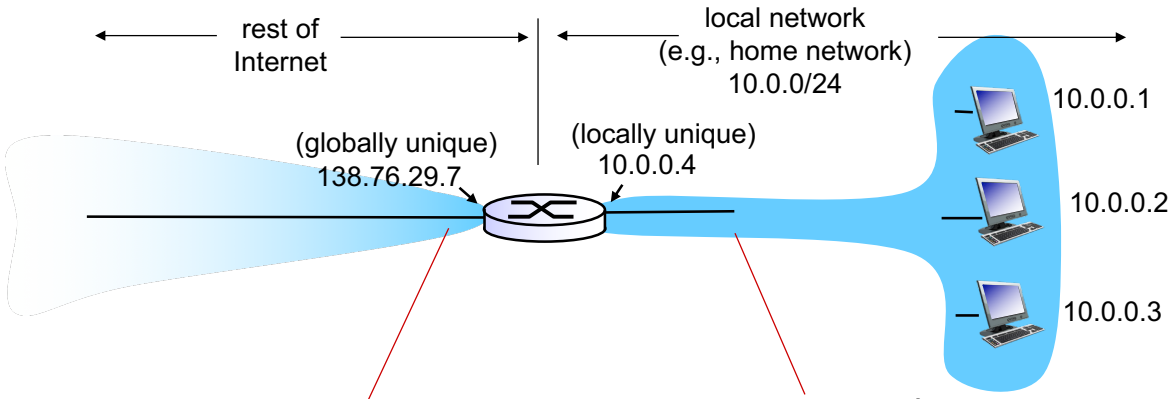
request

```
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."
```


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 either 10.0.0.0/8, 169.254.0.0/16, 172.16/12, or 192.168/16 private address for source, destination (as usual)

NAT: Network Address Translation

- Motivation: local network uses just one IP address as far as outside word is concerned:
  - no need to be allocated range of addresses from ISP, just one IP address is used for all devices
  - can change addresses of devices in local network without notifying the 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

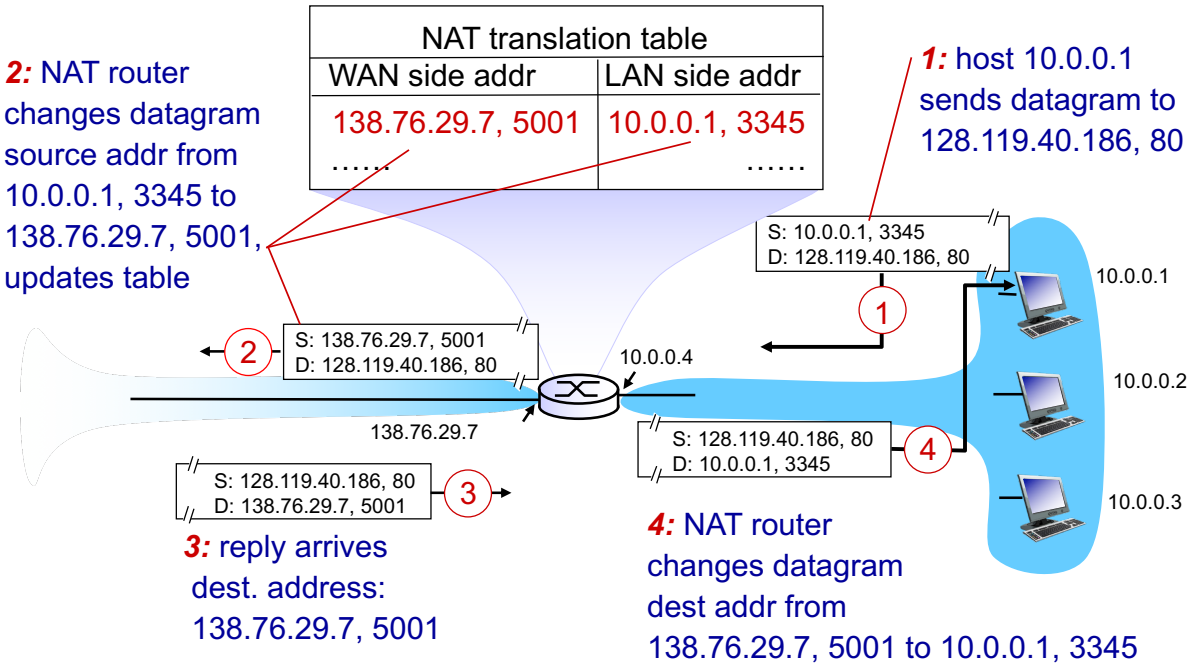
Implementation: NAT router must:

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

NAT: Network Address Translation

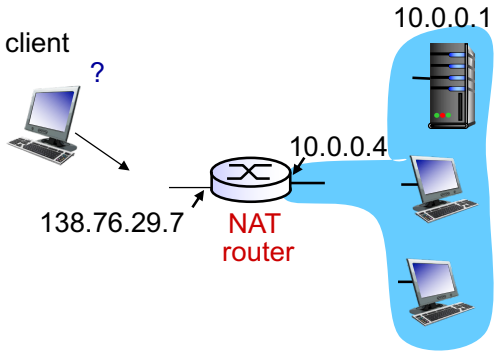


NAT: Network Address Translation

- 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

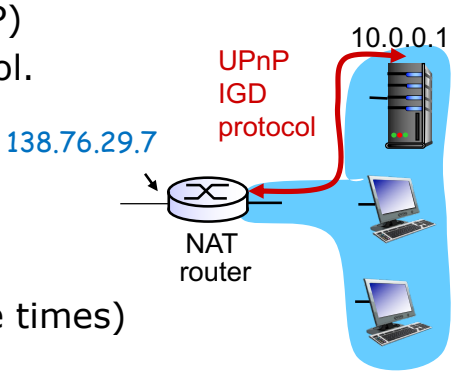
- client want to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7



- solution 1: statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (138.76.29.7, port 80) always forwarded to 10.0.0.1 port 2500

NAT traversal problem

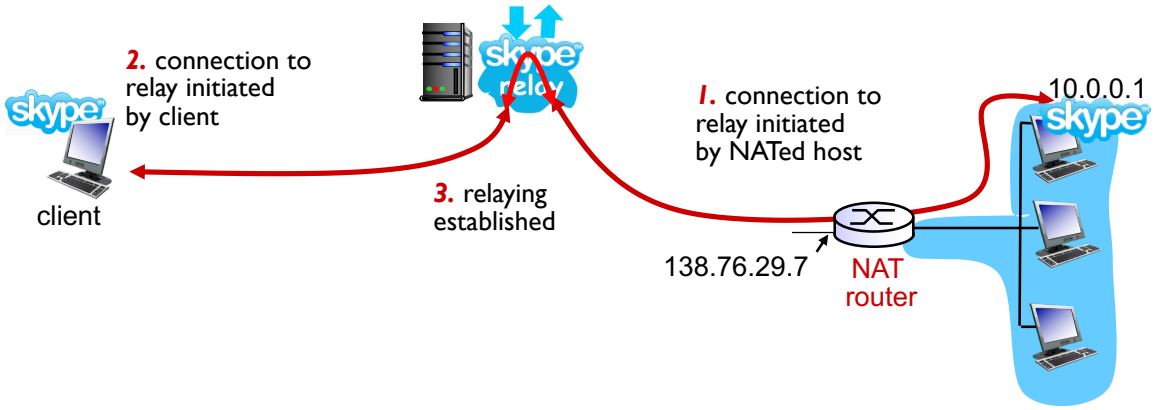
Solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:



- learn public IP address (138.76.29.7)
- enumerate existing port mappings
- add/remove port mappings (with lease times)
- An application running in a host can request a NAT mapping between its (private IP addr, private port #) and (public IP addr, public port #). E.g., mapping from (10.0.0.1, 3345) to (138.76.29.7, 5001).
- An application (e.g., BitTorrent) can advertise the (public IP addr, public port #) info (e.g., to its tracker).
- External hosts can initiate sessions to NATed hosts.

NAT traversal problem

- Solution 3: relaying (used in Skype)
  - NATed server establishes connection to relay
  - External client connects to relay
  - relay bridges packets between to connections



Chapter 4: Network Layer

- 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 Forwarding and SDN
  - Match
  - Action
  - OpenFlow examples of match-plus-action in action

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:
  - 128 bits (16 bytes) IP address
  - anycast address: allows a datagram to be delivered to any one of a group of hosts
  - fixed-length 40 byte header
  - Flow labeling: labeling of packets belonging to particular flows for which the sender requests special handling

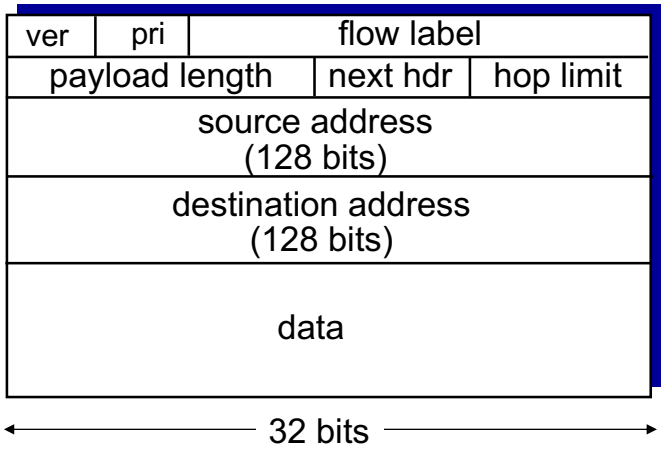
IPv6 Header (cont.)

priority: identify priority among datagrams in flow

Flow Label: identify datagrams in the same "flow".

e.g. of flows: realtime apps, high-priority users.

Next header: identify upper layer protocol (e.g., TCP) for data

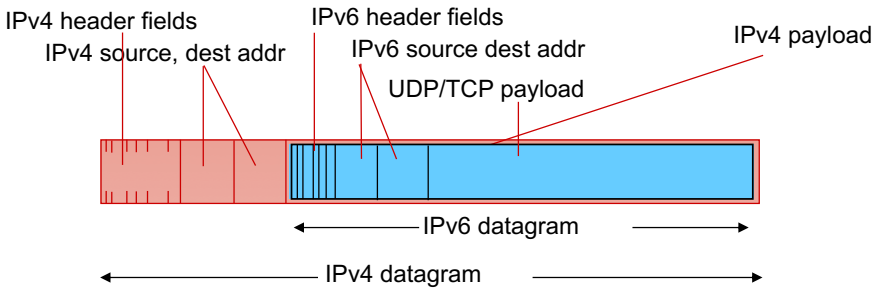


Other Changes from IPv4

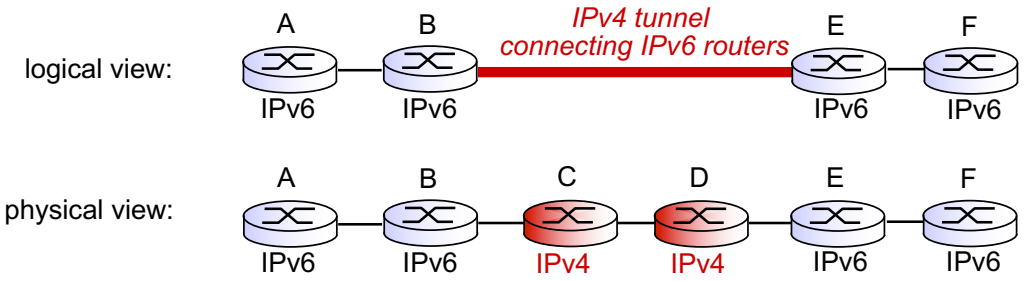
- *no fragmentation allowed*
- *Header Checksum*: removed entirely to reduce processing time at each hop
- *Options:* allowed, but outside of header, indicated by "Next Header" field
- *ICMPv6:* new version of ICMP
  - additional message types, e.g. "Packet Too Big"
  - multicast group management functions

Transition From IPv4 To IPv6

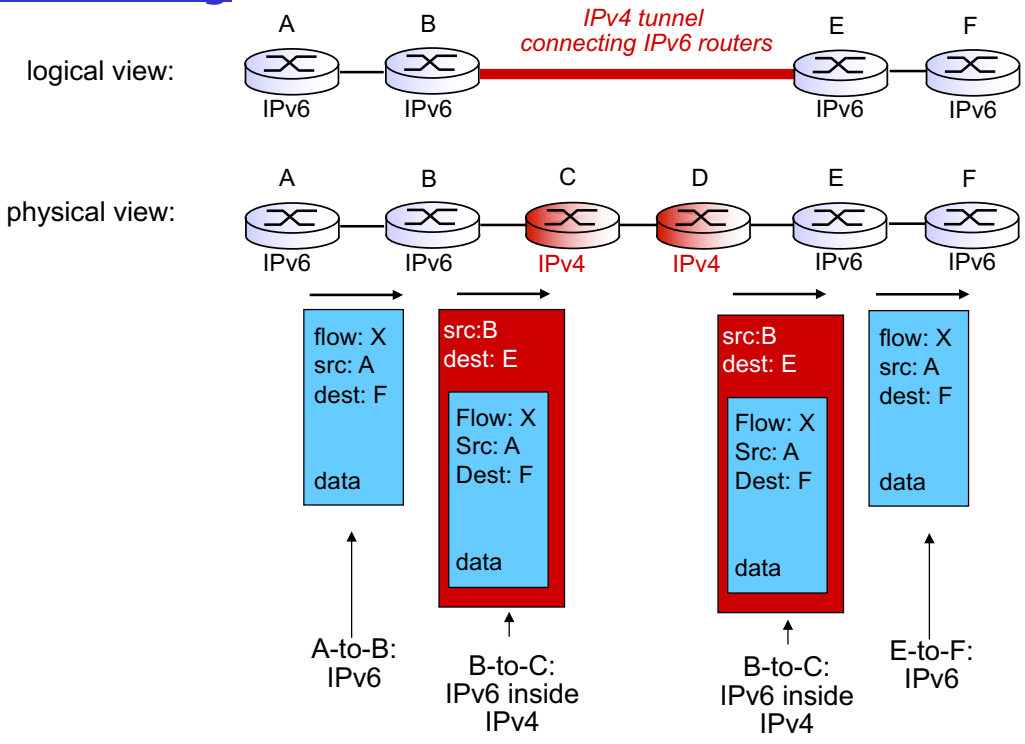
- Not all routers can be upgraded simultaneously
  - no "flag days"
  - How will the network operate with mixed IPv4 and IPv6 routers?
- *Tunneling:* IPv6 carried as payload in IPv4 datagram among IPv4 routers



Tunneling



Tunneling



IPv6: Adoption

- Google: 8% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable
- 3GPP has specified IPv6 as the standard addressing scheme for mobile multimedia

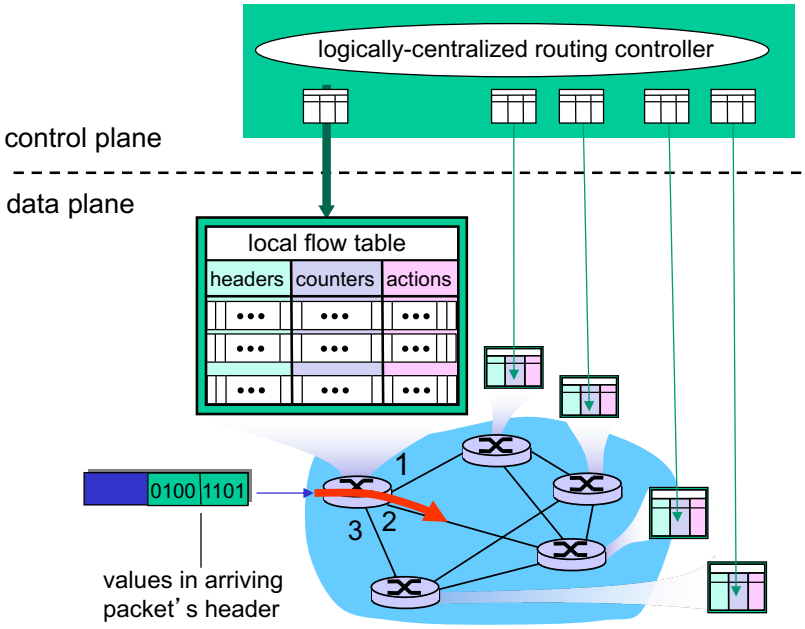
- *Long (long!) time for deployment, use*
  - 20 years and counting!
  - think of application-level changes in last 20 years: WWW, Facebook, streaming media, Skype, …
  - *Why?*

Chapter 4: Network Layer

- 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 Forwarding 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

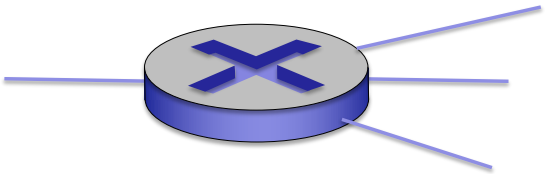


OpenFlow Data Plane Abstraction

- Traditional destination-based forwarding
  - *Match*: looking up a destination IP address
  - *Action*: sending packet into switching fabric to specific output port.
- Generalized forwarding
  - match values in packet header fields
  - *Actions: for matched packet:*
    - drop (e.g., firewall)
    - forward (e.g., destination-based forwarding, load balancing)
    - modify matched packet (e.g., NAT)
    - send matched packet to controller
    - *counters:* # of bytes and # of packets

OpenFlow Data Plane Abstraction

- flow: defined by a set of header fields

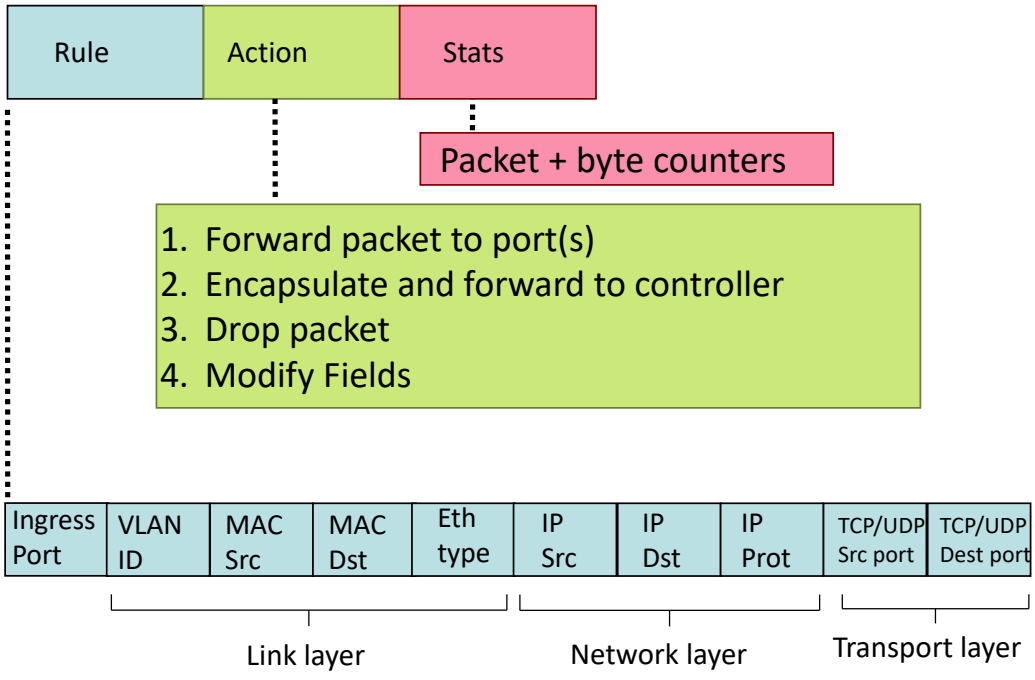


Flow table in a router (computed and distributed by controller) define router's match + action rules

* : wildcard

1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → send to controller

OpenFlow: Flow Table Entries



Examples

Destination-based forwarding:

Ingress Port	MAC Src	MAC Dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP/UDP Src port	TCP/UDP Dest port	Action
*	*	*	*	*	*	51.6.0.8	*	*	*	port 6

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

Firewall:

Ingress Port	MAC Src	MAC Dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP/UDP Src port	TCP/UDP Dest port	Forward
*	*	*	*	*	*	*	6	*	22	drop

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

Ingress Port	MAC Src	MAC Dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP/UDP Src port	TCP/UDP Dest port	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:

Ingress Port	MAC Src	MAC Dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP/UDP Src port	TCP/UDP Dest port	Action
*	22:A7:23: 11:E1:02	*	*	*	*	*	*	*	*	Port 3

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

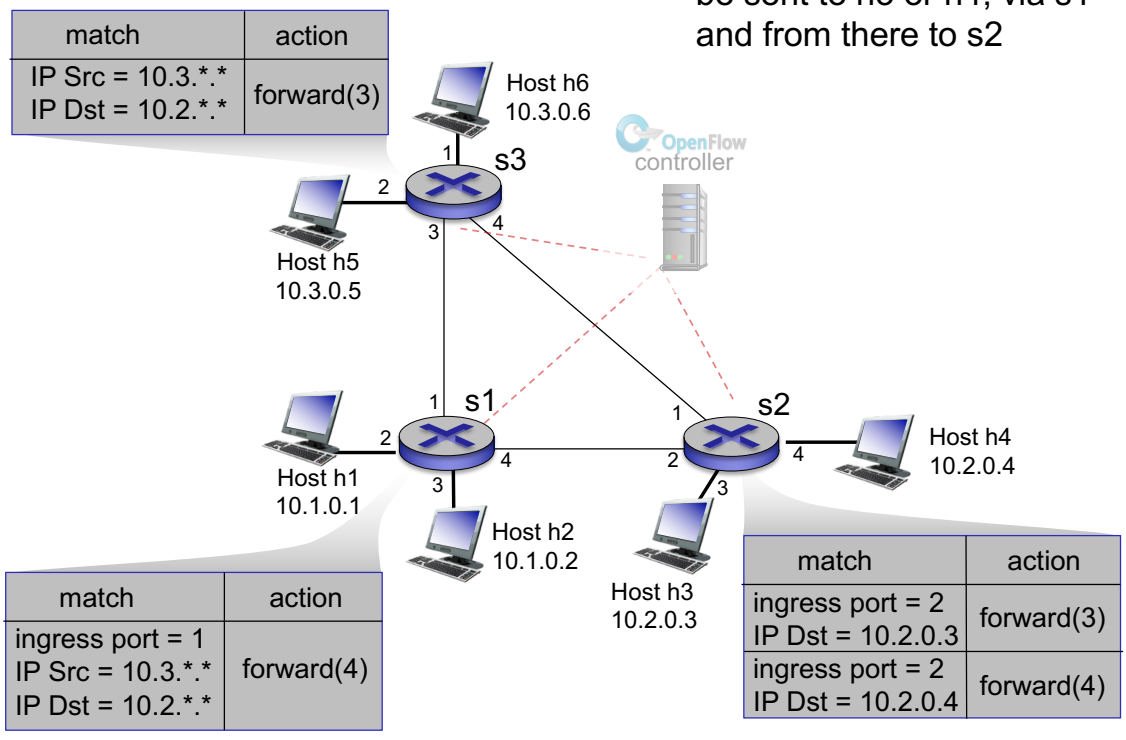
OpenFlow Abstraction

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

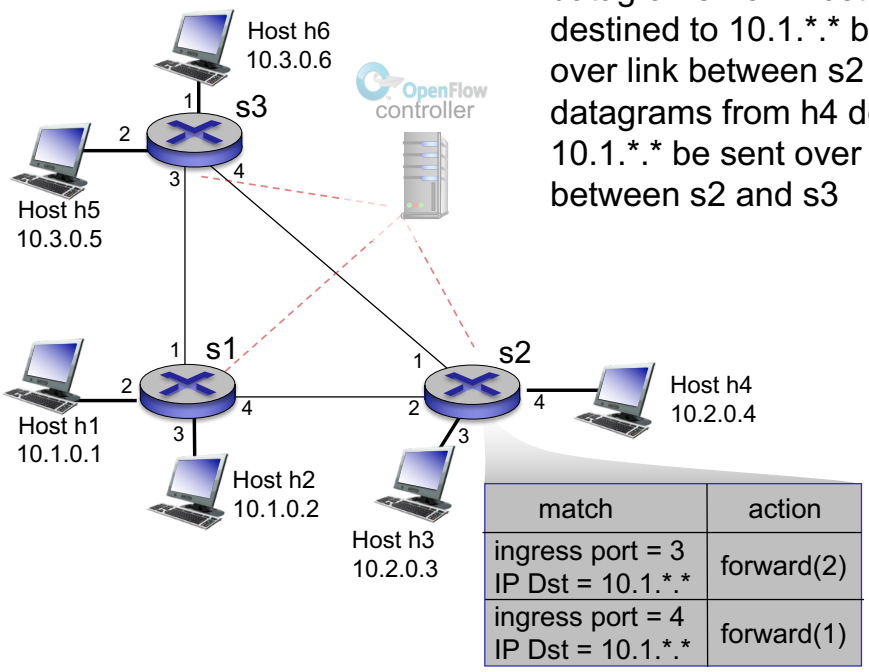
- Router
  - *match:* longest destination IP prefix
  - *action:* forward to output port
- Switch
  - *match:* destination MAC address
  - *action:* forward or flood
- Firewall
  - *match*: IP addresses and TCP/UDP port numbers
  - *action:* permit or deny
- NAT
  - *match:* IP address and port
  - *action:* rewrite address and port

OpenFlow Example 1

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



OpenFlow Example 2



Example:
datagrams from host h3 destined to 10.1.*.* be sent over link between s2 and s1; datagrams from h4 destined to 10.1.*.* be sent over link between s2 and s3

match	action
ingress port = 3 IP Dst = 10.1.*.*	forward(2)
ingress port = 4 IP Dst = 10.1.*.*	forward(1)

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)