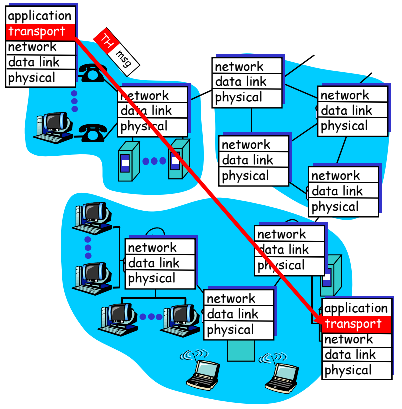
Chapter 4
Network Layer

Network layer functions

- transport packet from sending to receiving hosts
- network layer entity in *every* host, router

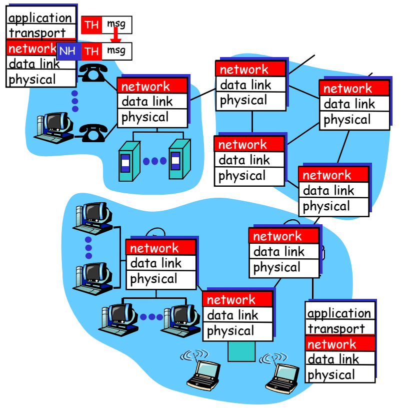
functions:
- *path determination:* route taken by packets from source to dest. *Routing algorithms*
- *forwarding:* move packets from router's input to appropriate router output
- *Call setup (VC networks):* Set-up routes state before sending packet



Network layer functions

- transport packet from sending to receiving hosts
- network layer entity in *every* host, router

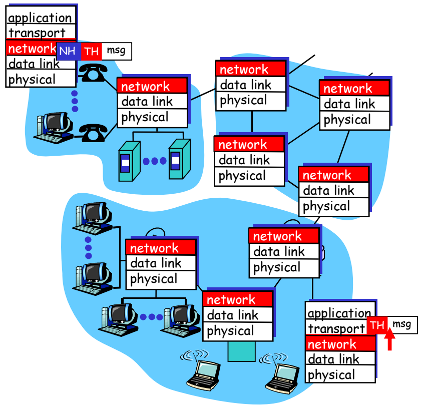
functions:
- *path determination:* route taken by packets from source to dest. *Routing algorithms*
- *forwarding:* move packets from router's input to appropriate router output
- *Call setup (VC networks):* Set-up routes state before sending packet



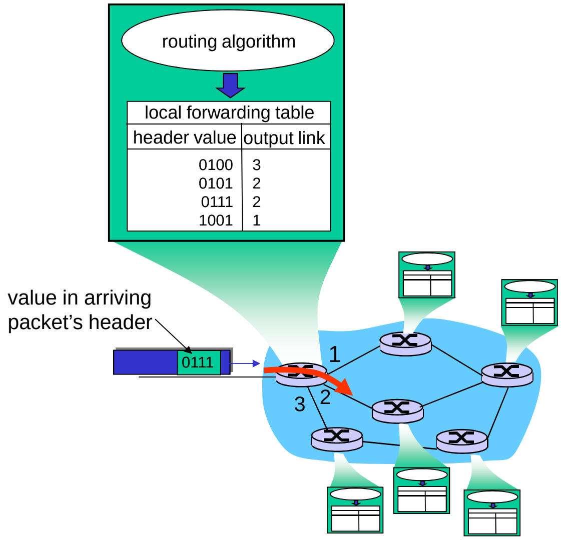
Network layer functions

- transport packet from sending to receiving hosts
- network layer entity in *every* host, router

functions:
- *path determination:* route taken by packets from source to dest. *Routing algorithms*
- *forwarding:* move packets from router's input to appropriate router output
- *Call setup (VC networks):* Set-up routes state before sending packet



Interplay between routing and forwarding



Connection setup

- 3rd important function in *some* network architectures:
  - ATM, frame relay, X.25
- before datagrams flow, two end hosts *and* intervening routers establish virtual connection
  - routers get involved
- network vs transport layer connection service:
  - network: between two hosts (may also involve intervening routers in case of VCs)
  - transport: between two processes

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 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

Network layer connection and connection-less service

- datagram network provides network-layer connectionless service
- VC network provides network-layer connection service
- analogous to the transport-layer services, but:
  - service: host-to-host
  - no choice: network provides one or the other
  - implementation: in network core

Virtual circuits

"source-to-dest path behaves much like telephone circuit"
- performance-wise
- network actions along source-to-dest path

- call setup, teardown for each call *before* data can flow
- each packet carries VC identifier (not destination host address)
- *every* router on source-dest path maintains "state" for each passing connection
- link, router resources (bandwidth, buffers) may be *allocated* to VC (dedicated resources = predictable service)

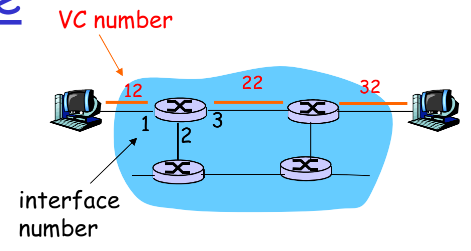
VC implementation

a VC consists of:

1. path from source to destination
2. VC numbers, one number for each link along path
3. entries in forwarding tables in routers along path

- packet belonging to VC carries VC number (rather than dest address)
- VC number can be changed on each link.
  - New VC number comes from forwarding table

Forwarding table



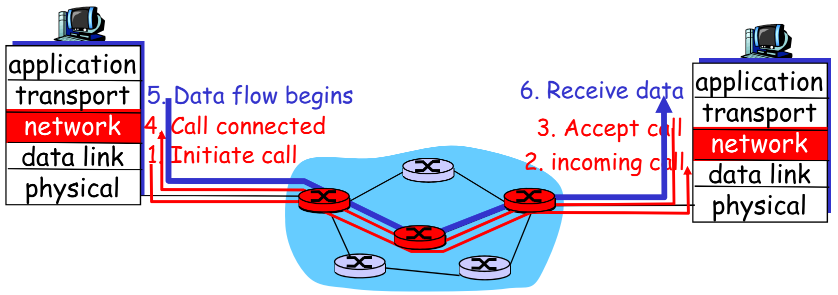
Forwarding table in northwest router:

Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
…	…	…	…

Routers maintain connection state information!

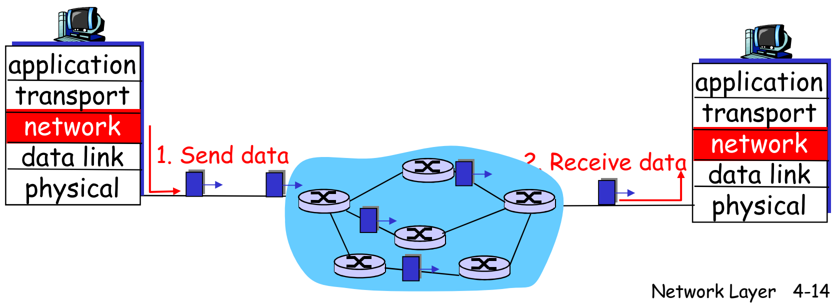
Virtual circuits: signaling protocols

- used to setup, maintain teardown VC
- used in ATM, frame-relay, X.25
- not used in today's Internet



Datagram networks

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of "connection"
- packets forwarded using destination host address
  - packets between same source-dest pair may take different paths



Forwarding table

4 billion possible entries

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

Datagram or VC network: why?

Internet (datagram)

- data exchange among computers
  - "elastic" service, no strict timing req.
- "smart" end systems (computers)
  - can adapt, perform control, error recovery
  - simple inside network, complexity at "edge"
- many link types
  - different characteristics
  - uniform service difficult

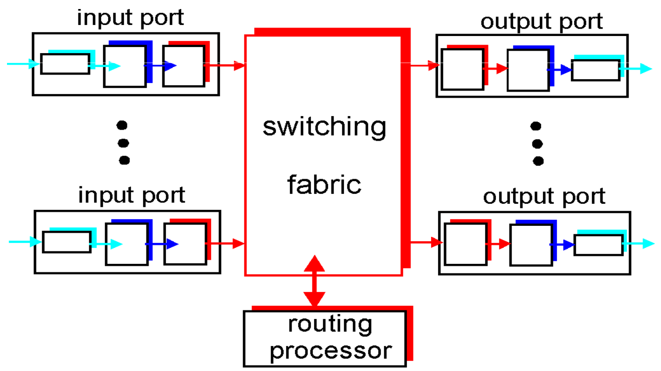
ATM (VC)

- evolved from telephony
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- "dumb" end systems
  - telephones
  - complexity inside network

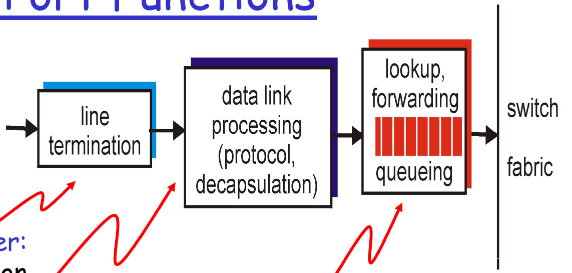
Router Architecture Overview

Two key router functions:

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



Input Port Functions



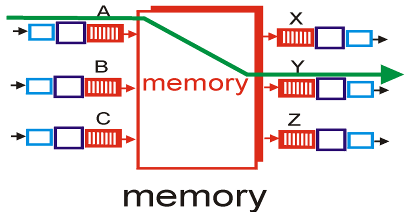
Physical layer:
bit-level reception

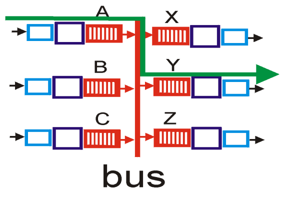
Data link layer:
e.g., Ethernet
see chapter 5

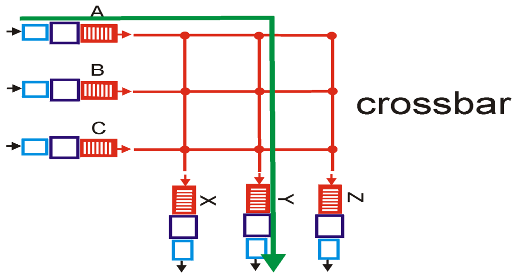
Decentralized switching:

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

Three types of switching fabrics



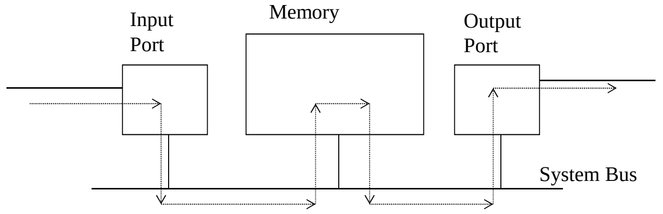




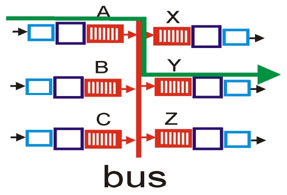
Switching Via Memory

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

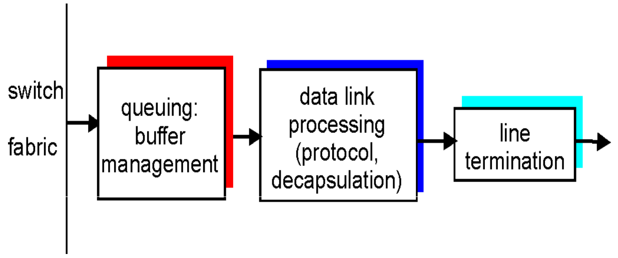


- datagram from input port memory to output port memory via a shared bus
- bus contention: switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers

Switching Via An Interconnection Network

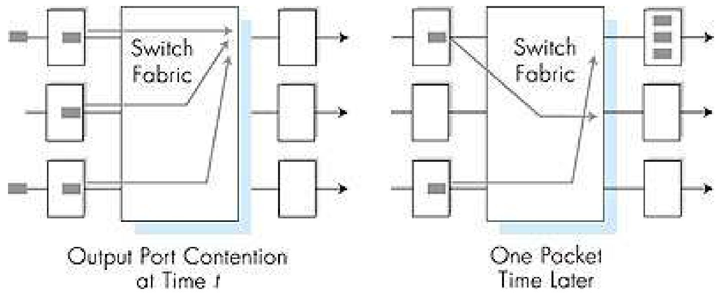
- overcome bus bandwidth limitations
- Banyan networks, other interconnection nets initially developed to connect processors in multiprocessor
- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection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

Output Ports


switch
fabric
queuing: buffer management
data link processing (protocol, decapsulation)
line termination

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

Output port queueing



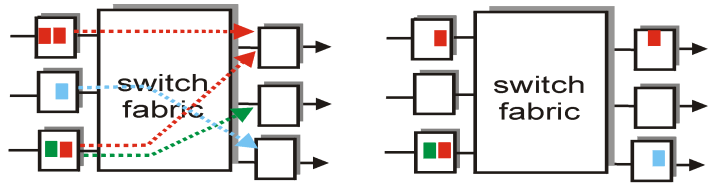
- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

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 Gps link: 2.5 Gbit buffer
- Recent recommendation: with *N* flows, buffering equal to $\frac{RTT \cdot C}{\sqrt{N}}$

Input Port Queuing

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

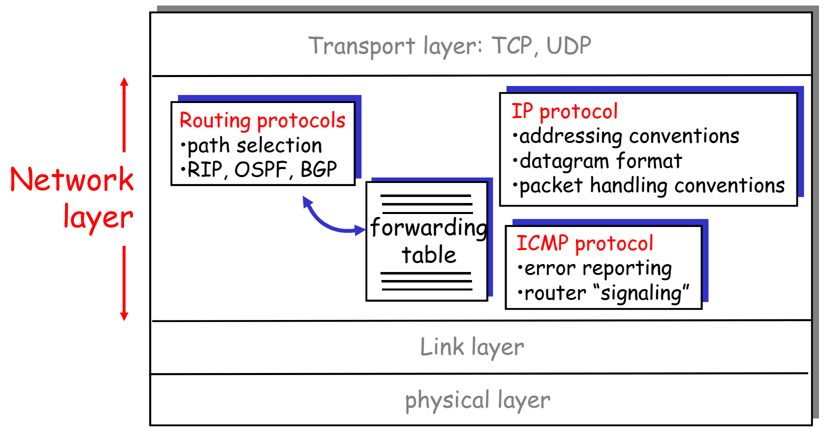


output port contention at time t - only one red packet can be transferred

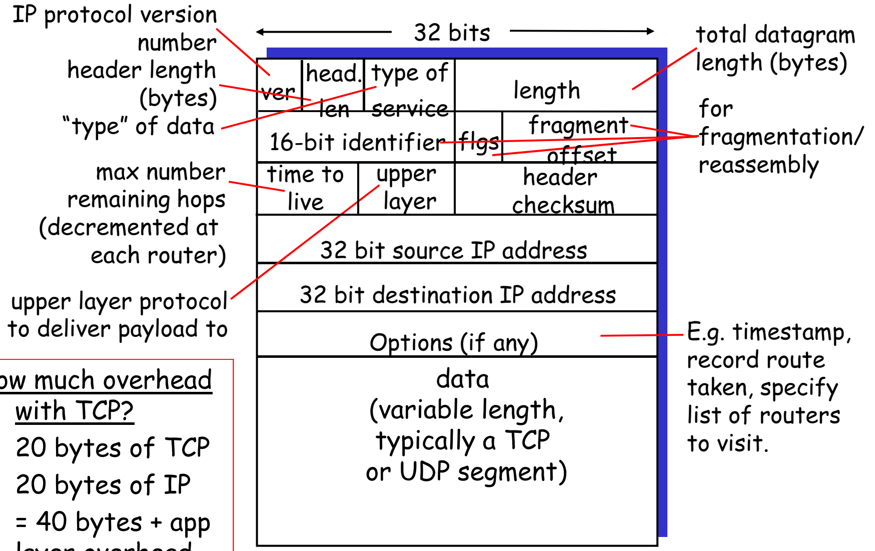
green packet experiences HOL blocking

The Internet Network layer

Host, router network layer functions:



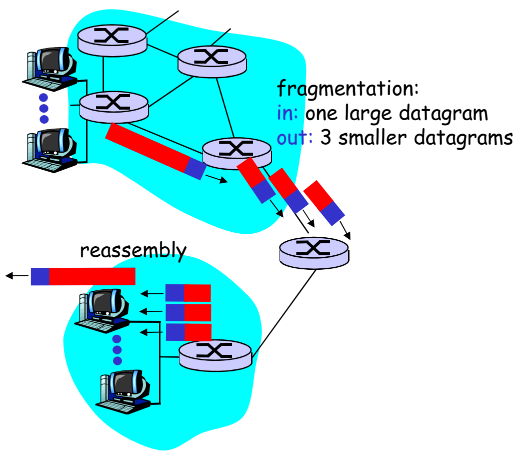
IP datagram format


IP protocol version number
header length (bytes)
"type" of data
max number remaining hops (decremented at each router)
upper layer protocol to deliver payload to
32 bits
ver
head. len
type of service
length
16-bit identifier
flgs
fragment offset
time to live
upper layer
header checksum
32 bit source IP address
32 bit destination IP address
Options (if any)
data (variable length, typically a TCP or UDP segment)
total datagram length (bytes)
for fragmentation/ reassembly
E.g. timestamp, record route taken, specify list of routers to visit.

how much overhead with TCP?
- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

IP Fragmentation & Reassembly

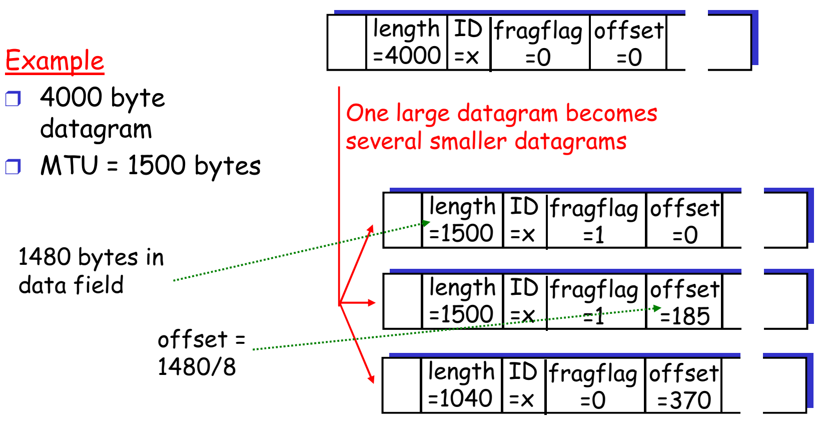
- network links have MTU (max.transfer size) - largest possible link-level frame.
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify,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



IP Fragmentation and Reassembly

Example

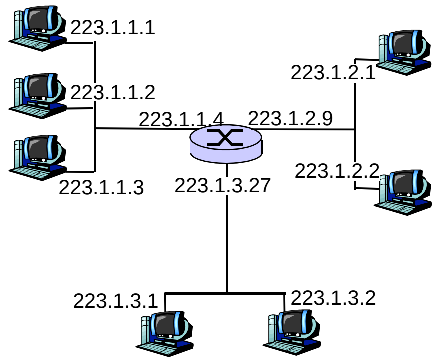
- 4000 byte datagram
- MTU = 1500 bytes



length =4000 | ID =x | fragflag =0 | offset =0
One large datagram becomes several smaller datagrams
length =1500 | ID =x | fragflag =1 | offset =0
length =1500 | ID =x | fragflag =1 | offset =185
length =1040 | ID =x | fragflag =0 | offset =370
1480 bytes in data field
offset = 1480/8

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 interface
  - IP addresses associated with each interface



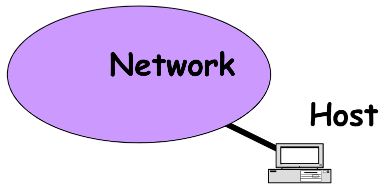
$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

IP Addressing

- Address can be divided in two parts

NetID	HostID

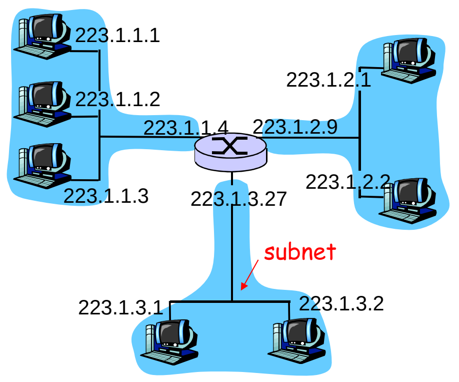
- NetID identifies the network
- HostID identifies the host within the network



Hosts within the same network have the same NetId

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

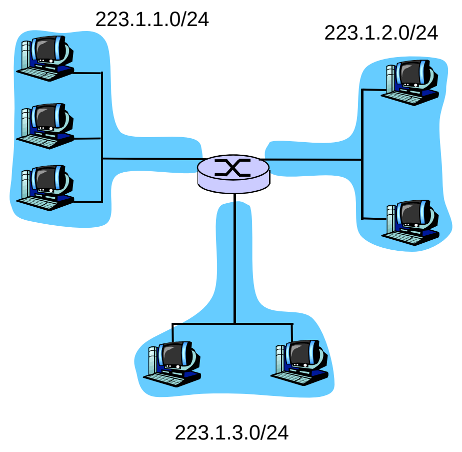


network consisting of 3 subnets

Subnets

Recipe

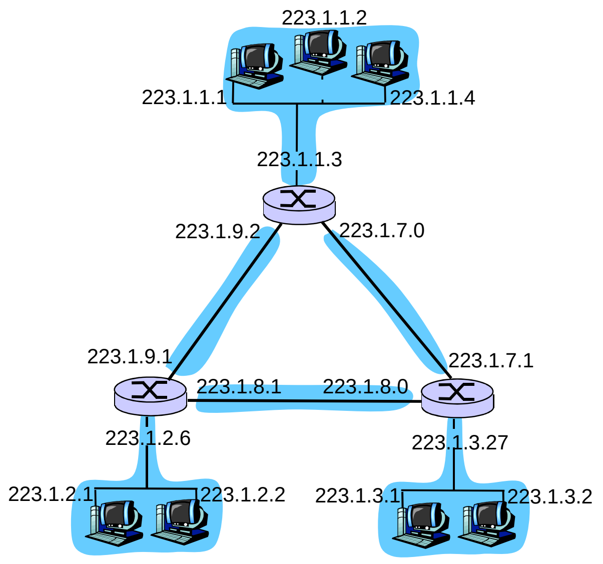
- To determine the subnets, detach each interface from its host or router, creating islands of isolated networks. Each isolated network is called a subnet.



Subnet mask: /24

Subnets

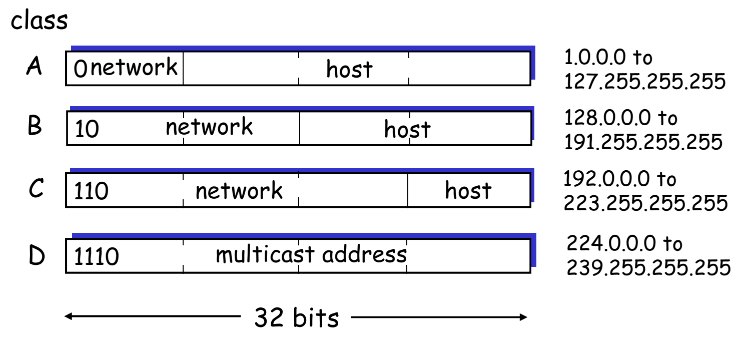
How many?



IP Addresses

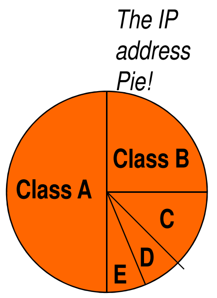
given notion of "network", let's re-examine IP addresses:

"class-full" addressing:


class
A
0 network
host
1.0.0.0 to 127.255.255.255
B
10
network
host
128.0.0.0 to 191.255.255.255
C
110
network
host
192.0.0.0 to 223.255.255.255
D
1110
multicast address
224.0.0.0 to 239.255.255.255
32 bits

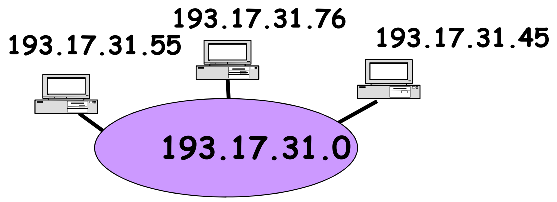
Counting up

- 32 bit IP address:
  - 2^{32} = 4.294.967.296 theoretical IP addresses
- class A:
  - 2^{7}-2 =126 networks [0.0.0.0 and 127.0.0.0 reserved]
  - 2^{24}-2 = 16.777.214 maximum hosts
    - **2.113.928.964** addressable hosts (49,22% of max)
- class B
  - 2^{14}=16.384 networks
  - 2^{16}-2 = 65.534 maximum hosts
    - **1.073.709.056** addressable hosts (24,99% of max)
- class C
  - 2^{21}=2.097.152 networks
  - 2^{8}-2 = 254 maximum hosts
    - **532.676.608** addressable hosts (12,40% of max)



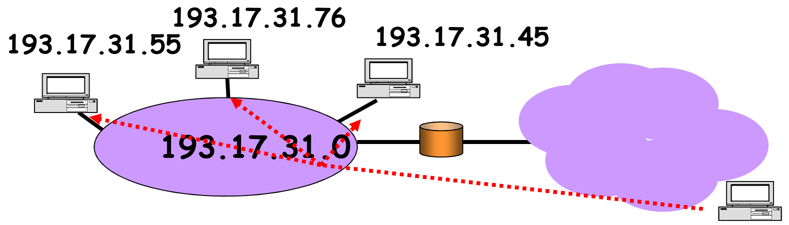
Special Addresses

- Network Address:
  - An address with the HostID bits set to 0 identifies the network with the given NetID (used in routing tables)
  - examples:
    - class B network: 131.175.0.0
    - class C network: 193.17.31.0



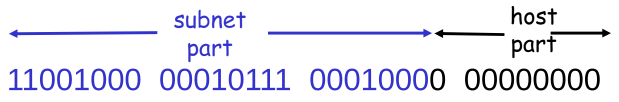
Special Addresses

- Direct Broadcast Address:
  - Address with HostID bit set to 1 is the broadcast address of the network identified by NetID.
  - example: 193.17.31.255



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



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

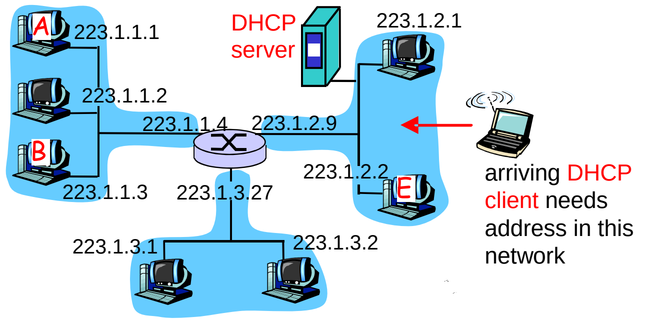
Can renew its lease on address in use

Allows reuse of addresses (only hold address while connected an "on")

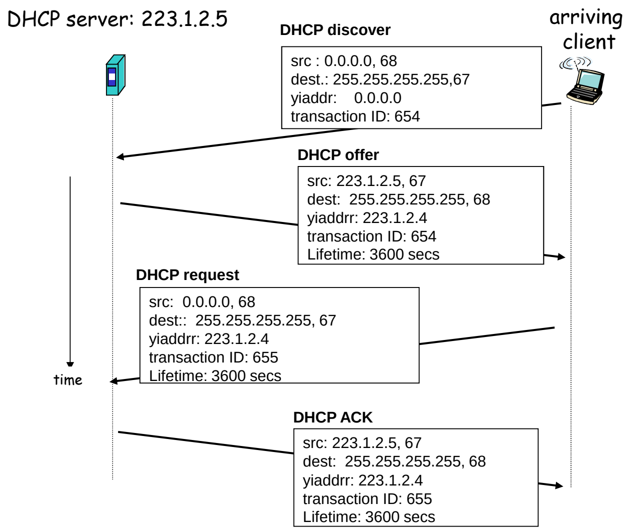
Support for mobile users who want to join network (more shortly)

DHCP overview:
- host broadcasts "DHCP discover" msg
- DHCP server responds with "DHCP offer" msg
- host requests IP address: "DHCP request" msg
- DHCP server sends address: "DHCP ack" msg

DHCP client-server scenario



DHCP client-server scenario



DHCP discover

src : 0.0.0.0, 68
dest.: 255.255.255.255,67
yiaddr: 0.0.0.0
transaction ID: 654

DHCP offer

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 654
Lifetime: 3600 secs

DHCP request

src: 0.0.0.0, 68
dest:: 255.255.255.255, 67
yiaddrr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs

DHCP ACK

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs

Getting a datagram from source to dest.

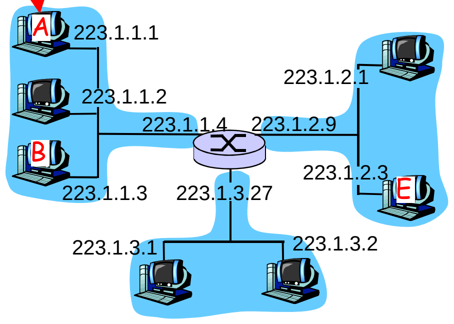
IP datagram:

misc fields	source IP addr	dest IP addr	data

- datagram remains unchanged, as it travels source to destination
- addr fields of interest here

forwarding table in A

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



Getting a datagram from source to dest.

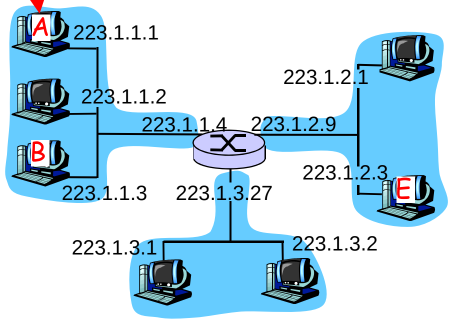
misc fields	223.1.1.1	223.1.1.3	data

Starting at A, send IP datagram addressed to B:

- look up net. address of B in forwarding table
- find B is on same net. as A
- link layer will send datagram directly to B inside link-layer frame
  - B and A are directly connected

forwarding table in A

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



Getting a datagram from source to dest.

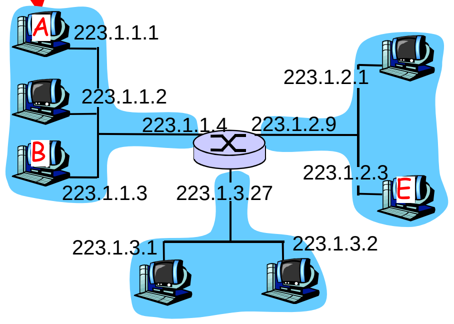
misc fields	223.1.1.1	223.1.2.3	data

Starting at A, dest. E:

- look up network address of E in forwarding table
- E on *different* network
  - A, E not directly attached
- routing table: next hop router to E is 223.1.1.4
- link layer sends datagram to router 223.1.1.4 inside link-layer frame
- datagram arrives at 223.1.1.4
- continued…..

forwarding table in A

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



Getting a datagram from source to dest.

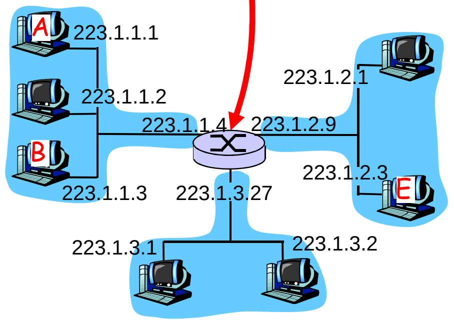
misc fields	223.1.1.1	223.1.2.3	data

Arriving at 223.1.4, destined for 223.1.2.2

- look up network address of E in router's forwarding table
- E on *same* network as router's interface 223.1.2.9
  - router, E directly attached
- link layer sends datagram to 223.1.2.3 inside link-layer frame via interface 223.1.2.9
- datagram arrives at 223.1.2.3!!! (hooray!)

forwarding table in router

Dest. Net	router	Nhops	interface
223.1.1	-	1	223.1.1.4
223.1.2	-	1	223.1.2.9
223.1.3	-	1	223.1.3.27



Forwarding table

4 billion possible entries

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

Longest prefix matching

Prefix Match	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?

IP addresses: how to get one?

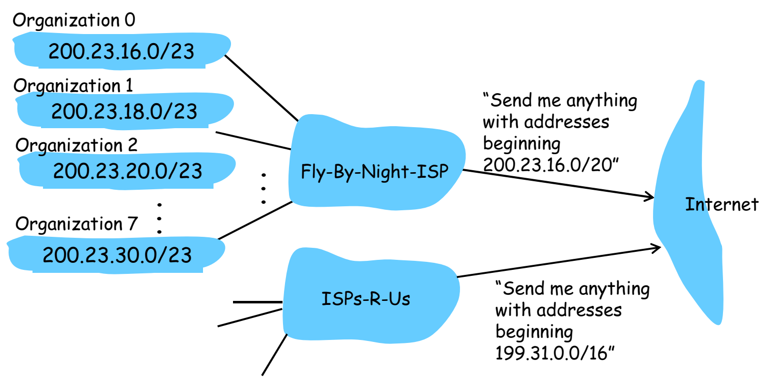
Q: How does *network* get subnet part of IP addr?

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

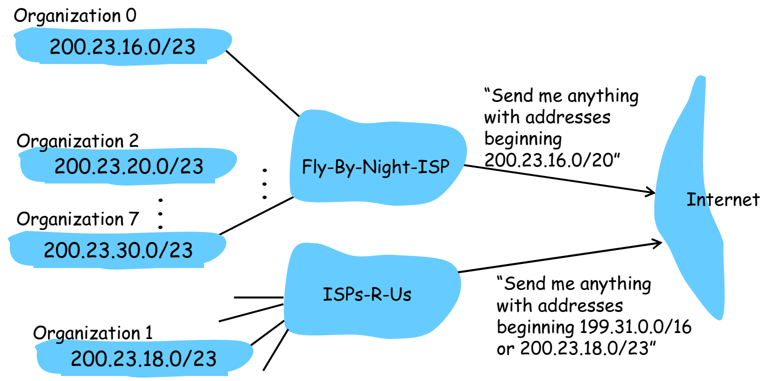
Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

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

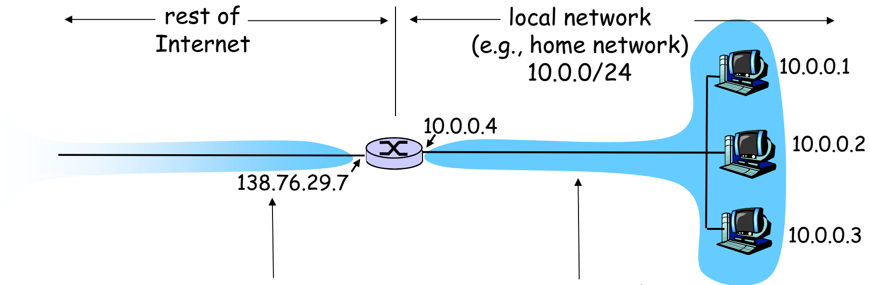


IP addressing: the last word...

Q: How does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers
- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

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

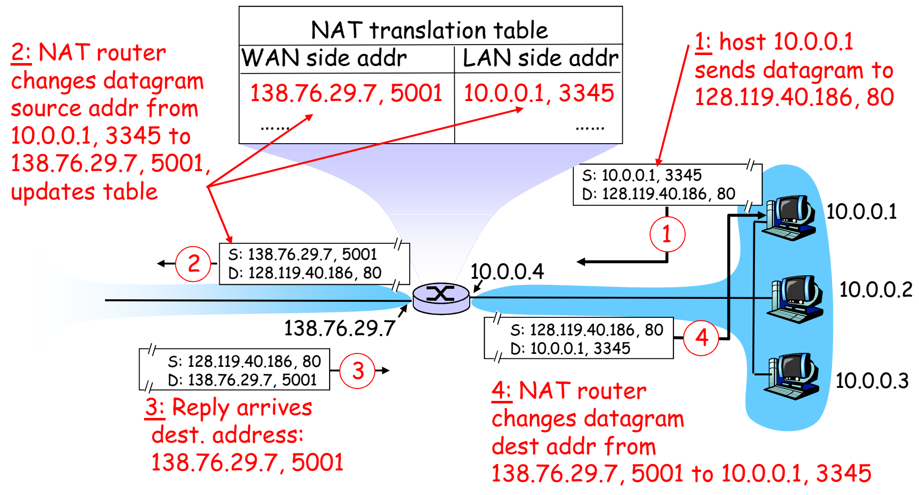
- Motivation: local network uses just one IP address as far as outside world is concerned:
  - range of addresses not needed from ISP: just one IP address for all devices
  - can change addresses of devices in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - devices inside local net not explicitly addressable, visible by outside world (a security plus).

NAT: Network Address Translation

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



NAT translation table

WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

3: Reply arrives dest. address: 138.76.29.7, 5001

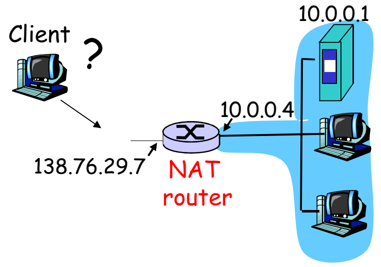
4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

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, eg, P2P applications
  - address shortage should instead be solved by IPv6

NAT traversal problem

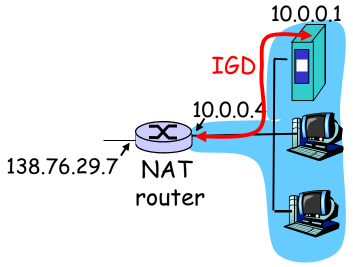
- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATted address: 138.76.29.7
- solution 1: statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000



NAT traversal problem

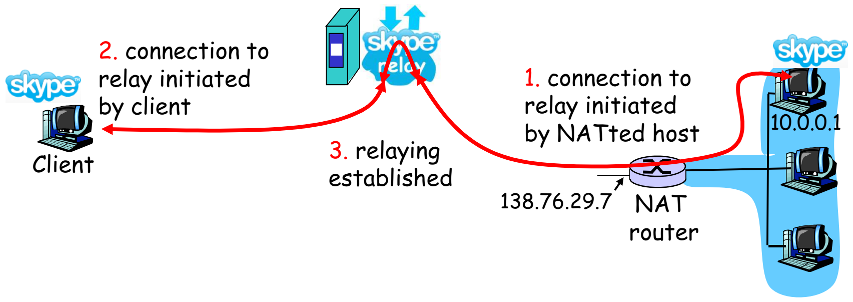
- solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATted host to:
  - learn public IP address (138.76.29.7)
  - add/remove port mappings (with lease times)

i.e., automate static NAT port map configuration

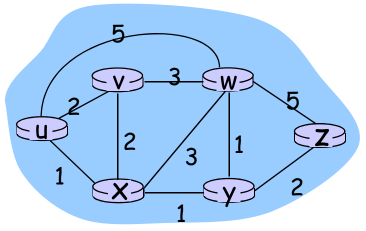


NAT traversal problem

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


2. connection to relay initiated by client
Client
skype relay
3. relaying established
1. connection to relay initiated by NATted host
10.0.0.1
138.76.29.7
NAT router

Graph abstraction



Graph: G = (N,E)

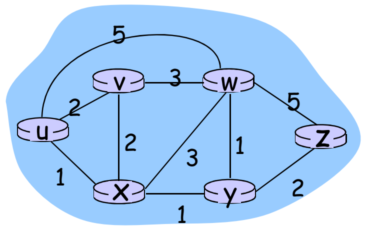
N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where N is set of peers and E is set of TCP connections

Graph abstraction: costs



- c(x,x') = cost of link (x,x')

  - e.g., c(w,z) = 5

- cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

Cost of path $(x_1, x_2, x_3,\dots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \dots + c(x_{p-1},x_p)$

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

Global or decentralized information?

Global:
- all routers have complete topology, link cost info
- "link state" algorithms

Decentralized:
- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Static or dynamic?

Static:
- routes change slowly over time

Dynamic:
- routes change more quickly
  - periodic update
  - in response to link cost changes

A Link-State Routing Algorithm

Dijkstra's algorithm
- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ('source") to all other nodes
  - gives forwarding table for that node
- iterative: after k iterations, know least cost path to k dest.'s

Notation:
- $c(x,y)$: link cost from node x to y; $= \infty$ if not direct neighbors
- $D(v)$: current value of cost of path from source to dest. v
- $p(v)$: predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

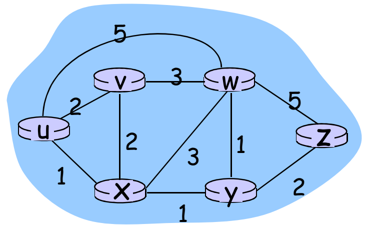
Dijsktra's Algorithm

```
Initialization:
 N' = {u}
 for all nodes v
   if v adjacent to u
       then $D(v) = c(u,v)$
   else $D(v) = \infty$

Loop
  find w not in N' such that D(w) is a minimum
  add w to N'
  update D(v) for all v adjacent to w and not in N' :
     $D(v) = \min( D(v), D(w) + c(w,v) )$
  /* new cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
until all nodes in N'
```

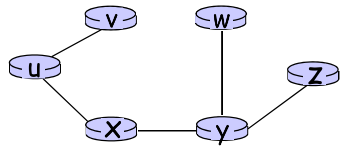
Dijkstra's algorithm: example

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Dijkstra's algorithm: example (2)

Resulting shortest-path tree from u:



Resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

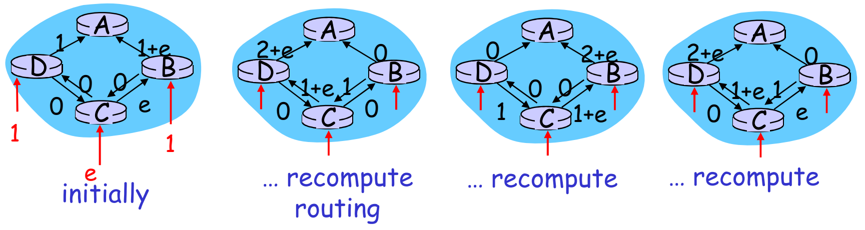
Dijkstra's algorithm, discussion

Algorithm complexity: n nodes

- each iteration: need to check all nodes, w, not in N
- n(n+1)/2 comparisons: $O(n^2)$
- more efficient implementations possible: O(nlogn)

Oscillations possible:

- e.g., link cost = amount of carried traffic



Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

Define

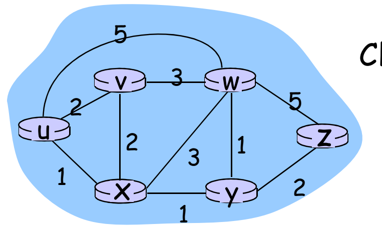
$d_x(y)$:= cost of least-cost path from x to y

Then

$$\boxed{d_x(y) = \min_v \{c(x,v) + d_v(y)\}}$$

where min is taken over all neighbors v of x

Bellman-Ford example



Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$\begin{aligned} d_u(z) &= \min \{ c(u,v) + d_v(z), \\ &\qquad c(u,x) + d_x(z), \\ &\qquad c(u,w) + d_w(z) \} \\ &= \min \{2 + 5, \\ &\qquad 1 + 3, \\ &\qquad 5 + 3\} = 4 \end{aligned}$$

Node that achieves minimum is next hop in shortest path ➔ forwarding table

Distance Vector Algorithm

- $D_x(y)$ = estimate of least cost from x to y
- Node x knows cost to each neighbor v: $c(x,v)$
- Node x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x also maintains its neighbors' distance vectors
  - For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$

Distance vector algorithm (4)

Basic idea:

- From time-to-time, each node sends its own distance vector estimate to neighbors
- *Asynchronous*
- When a node x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- Under minor, natural conditions, the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance Vector Algorithm (5)

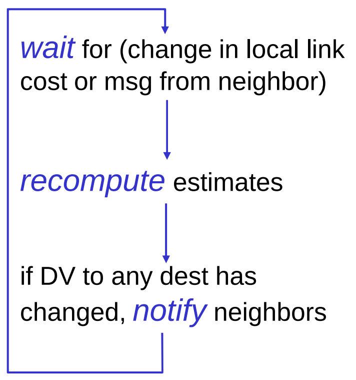
Iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

Distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

Each node:



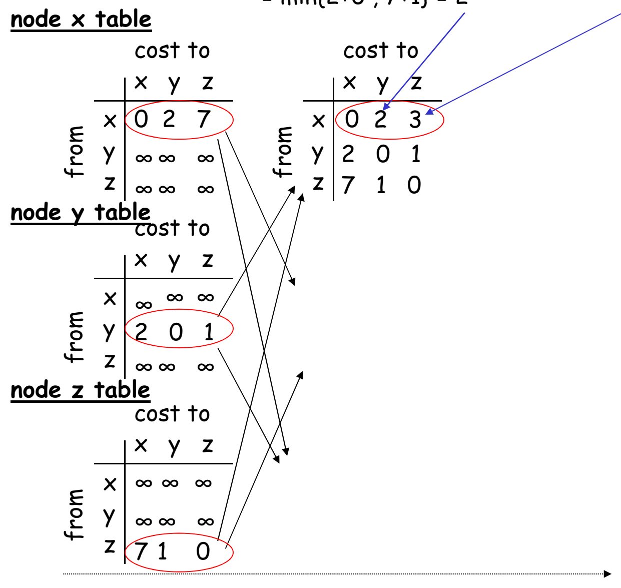
wait for (change in local link cost or msg from neighbor)

recompute estimates

if DV to any dest has changed, *notify* neighbors

$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$



node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

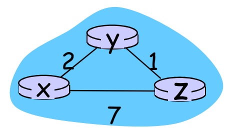
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

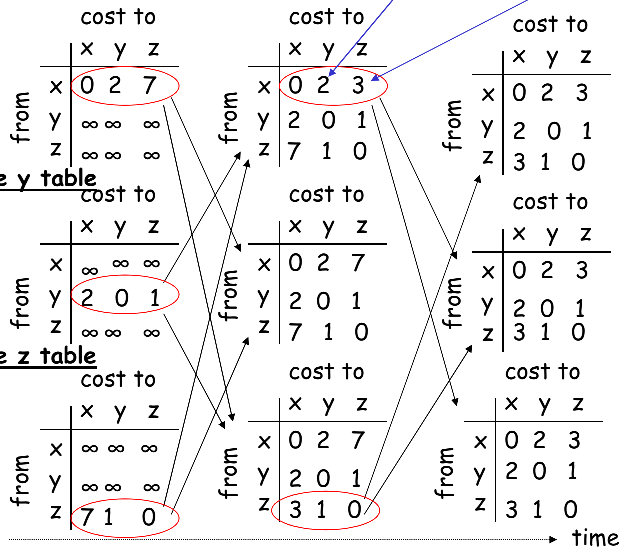
node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0



$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$



node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

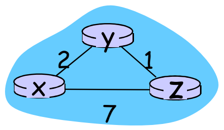
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

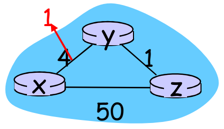
from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0



Distance Vector: link cost changes

Link cost changes:
- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors



"good news travels fast"

At time t_0, y detects the link-cost change, updates its DV, and informs its neighbors.

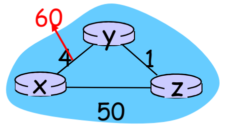
At time t_1, z receives the update from y and updates its table. It computes a new least cost to x and sends its neighbors its DV.

At time t_2, y receives z's update and updates its distance table. y's least costs do not change and hence y does *not* send any message to z.

Distance Vector: link cost changes

Link cost changes:
- good news travels fast
- bad news travels slow - "count to infinity" problem!
- 44 iterations before algorithm stabilizes: see text

Poisoned reverse:
- If Z routes through Y to get to X :
  - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- will this completely solve count to infinity problem?



Comparison of LS and DV algorithms

Message complexity
- LS: with n nodes, E links, O(nE) msgs sent
- DV: exchange between neighbors only
  - convergence time varies

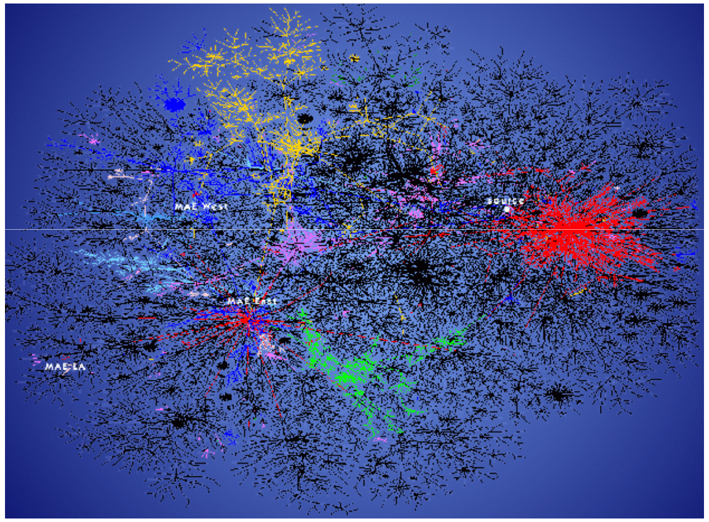
Speed of Convergence
- LS: O(n²) algorithm requires O(nE) msgs
  - may have oscillations
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?

LS:
- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:
- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

Hierarchical Routing



Hierarchical Routing

Our routing study thus far - idealization
- all routers identical
- network "flat"

... *not* true in practice

scale: with 200 million destinations:
- can't store all dest's in routing tables!
- routing table exchange would swamp links!

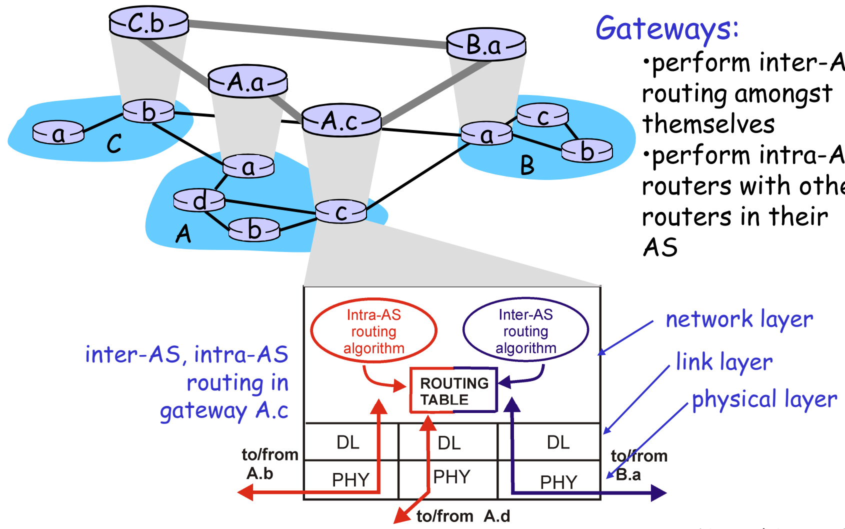
administrative autonomy
- internet = network of networks
- each network admin may want to control routing in its own network

Hierarchical Routing

- aggregate routers into regions, "autonomous systems" (AS)
- routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

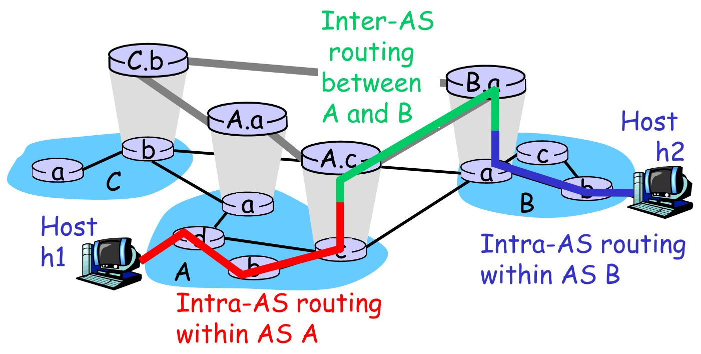
gateway routers
- special routers in AS
- run intra-AS routing protocol with all other routers in AS
- *also* responsible for routing to destinations outside AS
  - run *inter-AS routing* protocol with other gateway routers

Intra-AS and Inter-AS routing



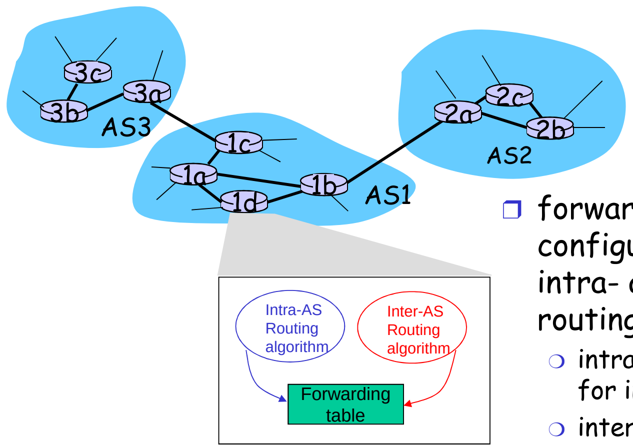
Gateways:
- perform inter-AS routing amongst themselves
- perform intra-AS routers with other routers in their AS

Intra-AS and Inter-AS routing


Inter-AS routing between A and B
C.b
B.a
A.a
A.c
a
b
C
c
a
b
B
Host h2
Host h1
a
d
b
c
A
Intra-AS routing within AS A
Intra-AS routing within AS B

- We'll examine specific inter-AS and intra-AS Internet routing protocols shortly

Interconnected ASes



- forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS sets entries for internal dests
  - inter-AS & intra-As sets entries for external dests

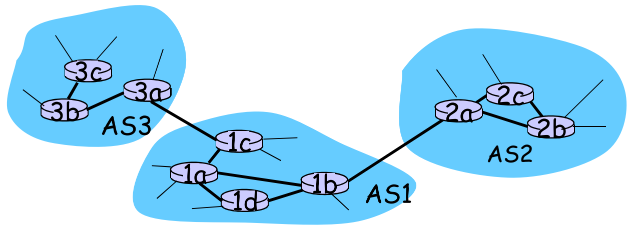
Inter-AS tasks

- suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

AS1 must:

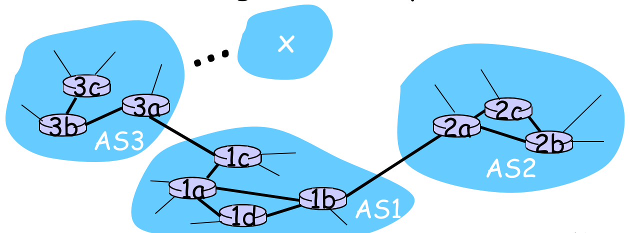
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

Job of inter-AS routing!



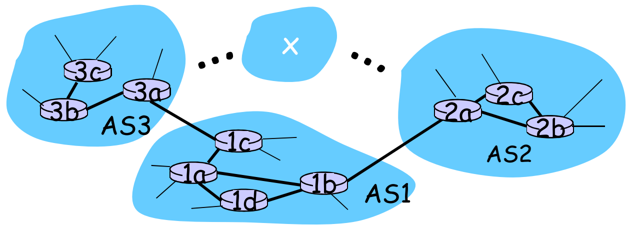
Example: Setting forwarding table in router 1d

- suppose AS1 learns (via inter-AS protocol) that subnet *x* reachable via AS3 (gateway 1c) but not via AS2.
- inter-AS protocol propagates reachability info to all internal routers.
- router 1d determines from intra-AS routing info that its interface *I* is on the least cost path to 1c.
  - installs forwarding table entry *(x,I)*



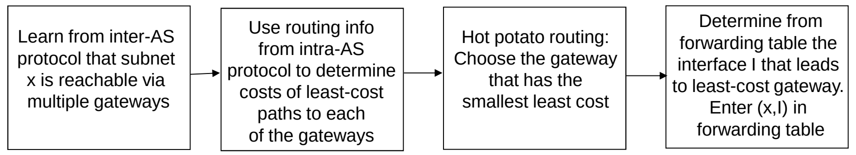
Example: Choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest x.
  - this is also job of inter-AS routing protocol!



Example: Choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest x.
  - this is also job of inter-AS routing protocol!
- hot potato routing: send packet towards closest of two routers.

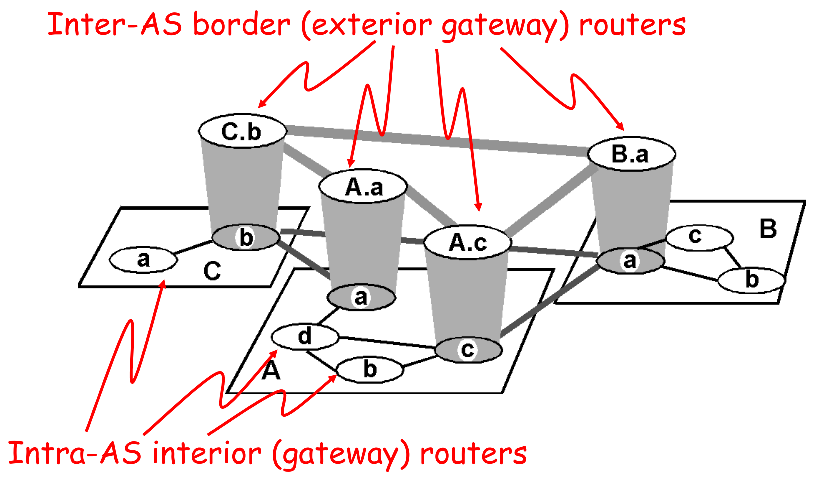


Learn from inter-AS protocol that subnet x is reachable via multiple gateways
Use routing info from intra-AS protocol to determine costs of least-cost paths to each of the gateways
Hot potato routing: Choose the gateway that has the smallest least cost
Determine from forwarding table the interface I that leads to least-cost gateway. Enter (x,I) in forwarding table

Routing in the Internet

- The Global Internet consists of Autonomous Systems (AS) interconnected with each other:
  - **Stub AS**: small corporation: one connection to other AS's
  - **Multihomed AS**: large corporation (no transit): multiple connections to other AS's
  - **Transit AS**: provider, hooking many AS's together
- Two-level routing:
  - **Intra-AS:** administrator responsible for choice of routing algorithm within network
  - **Inter-AS:** unique standard for inter-AS routing: BGP

Internet AS Hierarchy



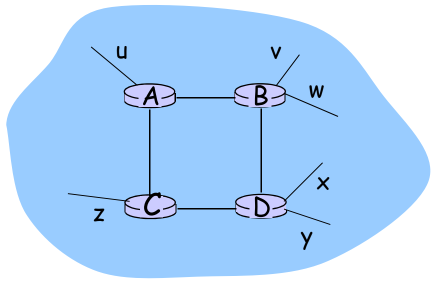
Inter-AS border (exterior gateway) routers
C.b
A.a
A.c
B.a
a
b
C
a
d
A
b
c
c
B
a
b
Intra-AS interior (gateway) routers

Intra-AS Routing

- also known as Interior Gateway Protocols (IGP)
- most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

RIP (Routing Information Protocol)

- distance vector algorithm
- included in BSD-UNIX Distribution in 1982
- distance metric: # of hops (max = 15 hops)



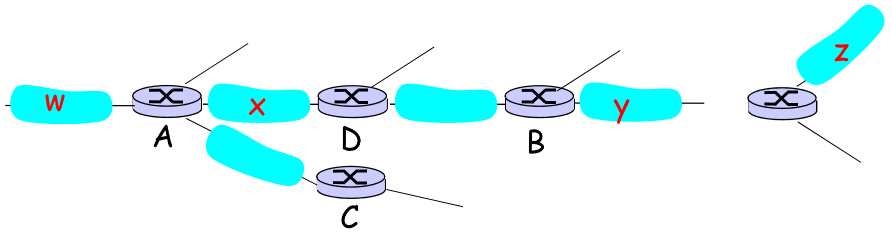
From router A to subnets:

destination	hops
u	1
v	2
w	2
x	3
y	3
z	2

RIP advertisements

- *distance vectors:* exchanged among neighbors every 30 sec via Response Message (also called advertisement)
- each advertisement: list of up to 25 destination subnets within AS

RIP: Example



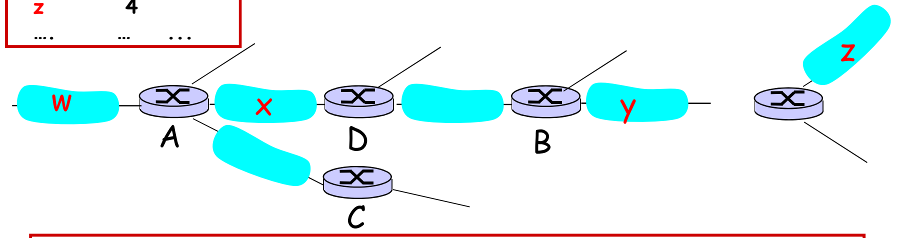
Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B	7
x	--	1
….	….	....

Routing/Forwarding table in D

RIP: Example

Dest	hops	
w	1	
x	1	
z	4	
….	…	. . .

Advertisement from A to D



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	~~B~~ A	~~7~~ 5
x	--	1
….	….	. . . .

Routing/Forwarding table in D

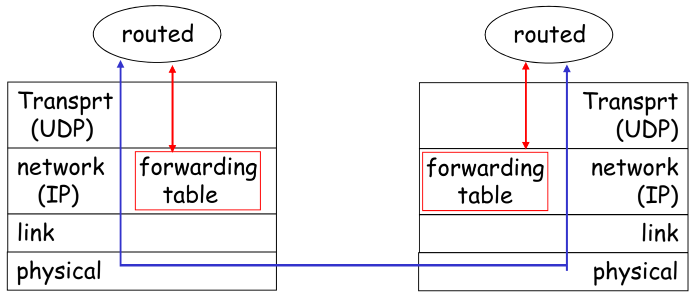
RIP: Link Failure and Recovery

If no advertisement heard after 180 sec --> neighbor/link declared dead

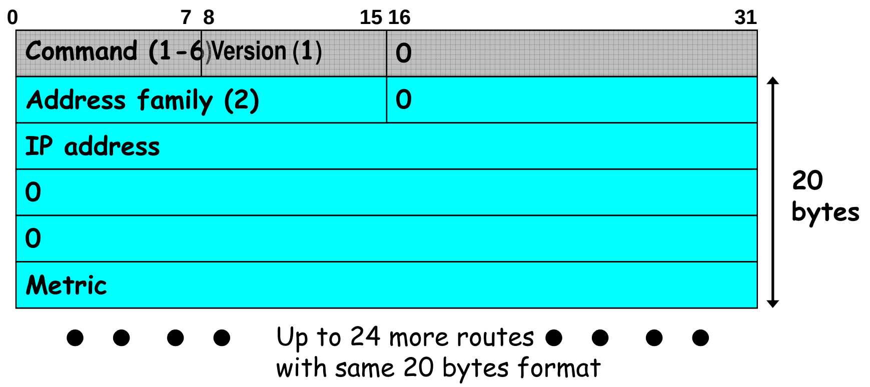
- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly (?) propagates to entire net
- *poison reverse* used to prevent ping-pong loops (infinite distance = 16 hops)

RIP Table processing

- RIP routing tables managed by **application-level** process called route-d (daemon)
- advertisements sent in UDP packets, periodically repeated



RIP message



Command: 1=request to send all or part of the routing table; 2=reply (3-6 obsolete or non documented)
Address family: 2=IP addresses
metric: distance of *emitting router* from the specified IP address in
number of hops (valid from 1 to 15; 16=infinite)

Message size

  - 8 UDP header
  - 4 bytes RIP header
  - 20 bytes x up to 25 entries
- total: maximum of 512 bytes UDP datagram

- 25 entries: too little to transfer an entire routing table
  - more than 1 UDP datagram generally needed

Initialization

- When routing daemon started, send special RIP request on every interface
  - command = 1 (request)
  - one entry all bit set to 0
- This asks for complete routing table from all connected routers
  - allows to discover adjacent routers!

OSPF (Open Shortest Path First)

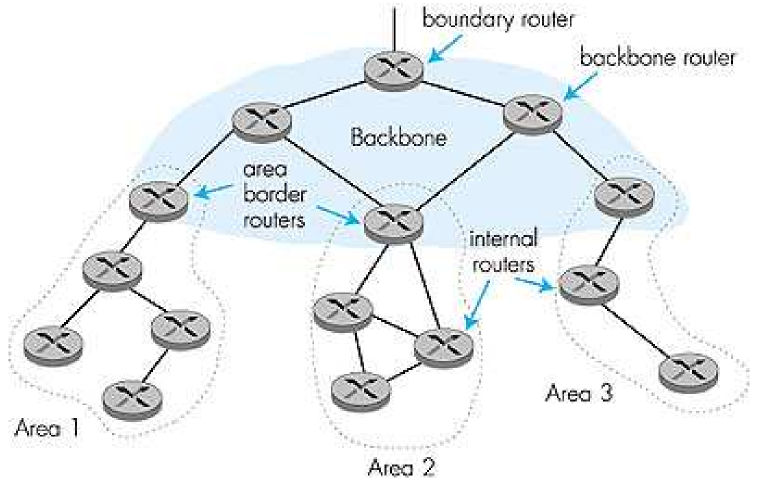
- "open": publicly available
- uses Link State algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm

- OSPF advertisement carries one entry per neighbor router
- advertisements disseminated to entire AS (via flooding)
  - carried in OSPF messages directly over IP (rather than TCP or UDP

OSPF "advanced" features (not in RIP)

- security: all OSPF messages authenticated (to prevent malicious intrusion)
- multiple same-cost paths allowed (only one path in RIP)
- For each link, multiple cost metrics for different TOS (e.g., satellite link cost set "low" for best effort; high for real time)
- integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- hierarchical OSPF in large domains.

Hierarchical OSPF



Hierarchical OSPF

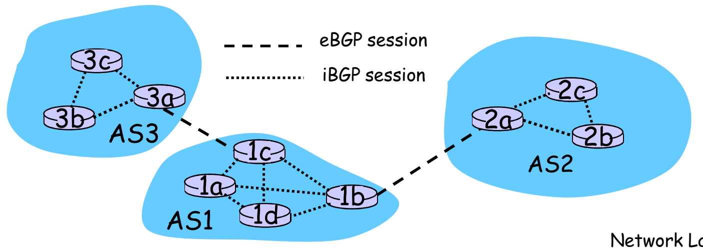
- two-level hierarchy: local area, backbone.
  - Link-state advertisements only in area
  - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- *area border routers:* "summarize" distances to nets in own area, advertise to other Area Border routers.
- *backbone routers:* run OSPF routing limited to backbone.
- *boundary routers:* connect to other AS's.

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto standard
- BGP provides each AS a means to:
  1. Obtain subnet reachability information from neighboring ASs.
  2. Propagate reachability information to all AS-internal routers.
  3. Determine "good" routes to subnets based on reachability information and policy.
- allows subnet to advertise its existence to rest of Internet: *"I am here"*

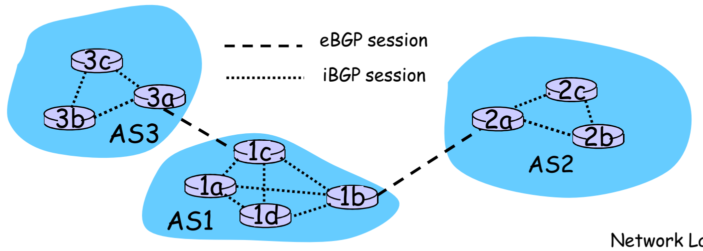
BGP basics

- pairs of routers (BGP peers) exchange routing info over semi-permanent TCP connections: BGP sessions
  - BGP sessions need not correspond to physical links.
- when AS2 advertises a prefix to AS1:
  - AS2 *promises* it will forward datagrams towards that prefix.
  - AS2 can aggregate prefixes in its advertisement



Distributing reachability info

- using eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
  - 1c can then use iBGP do distribute new prefix info to all routers in AS1
  - 1b can then re-advertise new reachability info to AS2 over 1b-to-2a eBGP session
- when router learns of new prefix, it creates entry for prefix in its forwarding table.



Path attributes & BGP routes

- advertised prefix includes BGP attributes.
  - prefix + attributes = "route"
- two important attributes:
  - AS-PATH: contains ASs through which prefix advertisement has passed: e.g, AS 67, AS 17
  - NEXT-HOP: indicates specific internal-AS router to next-hop AS. (may be multiple links from current AS to next-hop-AS)
- when gateway router receives route advertisement, uses import policy to accept/decline.

BGP route selection

- router may learn about more than 1 route to some prefix. Router must select route.
- elimination rules:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria

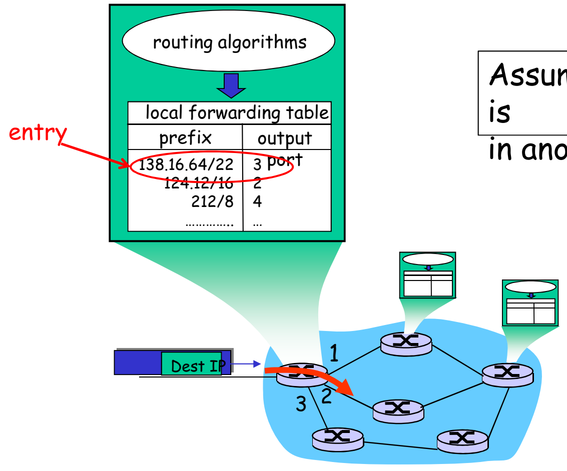
BGP messages

- BGP messages exchanged using TCP.
- BGP messages:
  - OPEN: opens TCP connection to peer and authenticates sender
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also used to close connection

Putting it Altogether: *How Does an Entry Get Into a Router's Forwarding Table?*

- Answer is complicated!
- Ties together hierarchical routing (Section 4.5.3) with BGP (4.6.3) and OSPF (4.6.2).
- Provides nice overview of BGP!

How does entry get in forwarding table?



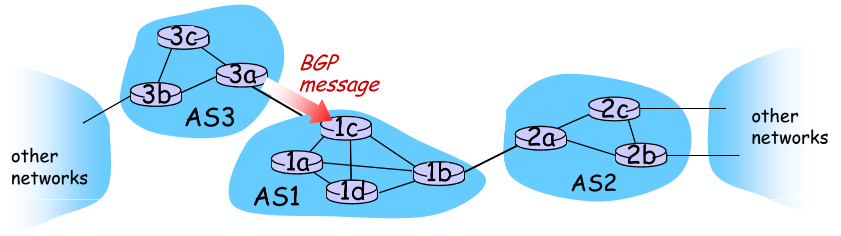
Assume prefix is in another AS.

How does entry get in forwarding table?

High-level overview

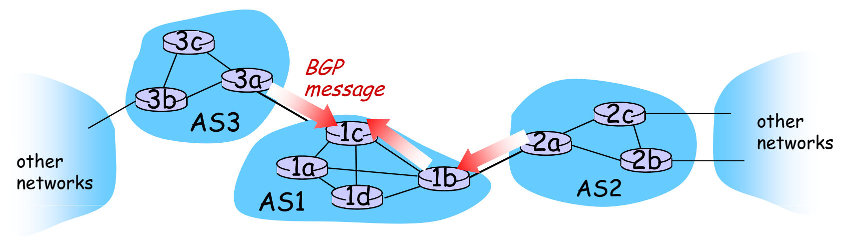
1. Router becomes aware of prefix
2. Router determines output port for prefix
3. Router enters prefix-port in forwarding table

Router becomes aware of prefix



- BGP message contains "routes"
- "route" is a prefix and attributes: AS-PATH, NEXT-HOP,...
- Example: route:
  - Prefix:138.16.64/22 ; AS-PATH: AS3 AS131 ; NEXT-HOP: 201.44.13.125

Router may receive multiple routes



- Router may receive multiple routes for <u>same</u> prefix
- Has to select one route

Select best BGP route to prefix

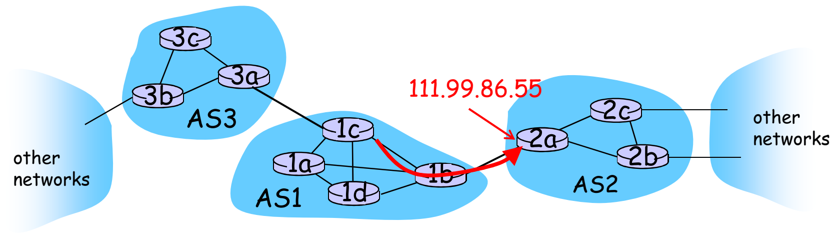
- Router selects route based on shortest AS-PATH
- Example:
  - AS2 AS17 to 138.16.64/22
  - AS3 AS131 AS201 to 138.16.64/22

select

- What if there is a tie? We'll come back to that!

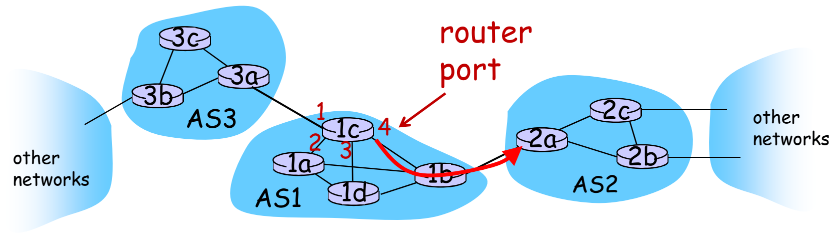
Find best intra-route to BGP route

- Use selected route's NEXT-HOP attribute
  - Route's NEXT-HOP attribute is the IP address of the router interface that begins the AS PATH.
- Example:
  - AS-PATH: AS2 AS17 ; NEXT-HOP: 111.99.86.55
- Router uses OSPF to find shortest path from 1c to 111.99.86.55



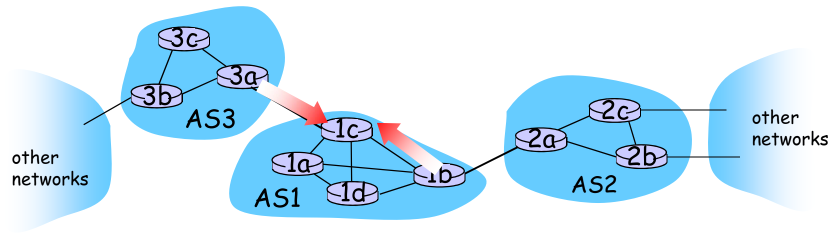
Router identifies port for route

- Identifies port along the OSPF shortest path
- Adds prefix-port entry to its forwarding table:
  - (138.16.64/22 , port 4)



Hot Potato Routing

- ❒ Suppose there two or more best inter-routes.
- ❒ Then choose route with closest NEXT-HOP
  - ❍ Use OSPF to determine which gateway is closest
  - ❍ Q: From 1c, chose AS3 AS131 or AS2 AS17?
  - ❍ A: route AS3 AS201 since it is closer

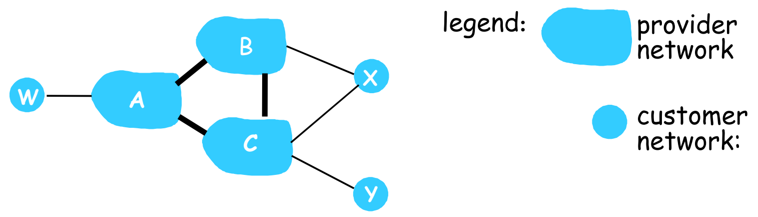


How does entry get in forwarding table?

Summary

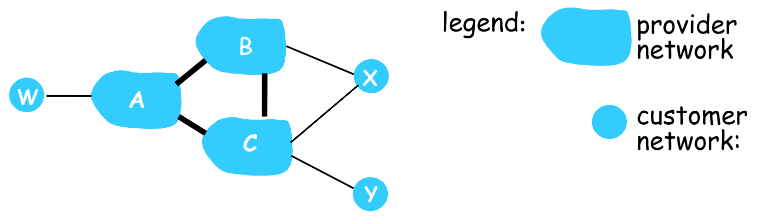
1. Router becomes aware of prefix
   - ❍ via BGP route advertisements from other routers
2. Determine router output port for prefix
   - ❍ Use BGP route selection to find best inter-AS route
   - ❍ Use OSPF to find best intra-AS route leading to best inter-AS route
   - ❍ Router identifies router port for that best route
3. Enter prefix-port entry in forwarding table

BGP routing policy



- ❒ A,B,C are provider networks
- ❒ X,W,Y are customer (of provider networks)
- ❒ X is dual-homed: attached to two networks
  - ❍ X does not want to route from B via X to C
  - ❍ .. so X will not advertise to B a route to C

BGP routing policy (2)



- ❒ A advertises path AW to B
- ❒ B advertises path BAW to X
- ❒ Should B advertise path BAW to C?
  - ❍ No way! B gets no "revenue" for routing CBAW since neither W nor C are B's customers
  - ❍ B wants to force C to route to w via A
  - ❍ B wants to route *only* to/from its customers!

Why different Intra- and Inter-AS routing ?

Policy:

- Inter-AS: admin wants control over how its traffic routed, who routes through its net.
- Intra-AS: single admin, so no policy decisions needed

Scale:

- hierarchical routing saves table size, reduced update traffic

Performance:

- Intra-AS: can focus on performance
- Inter-AS: policy may dominate over performance