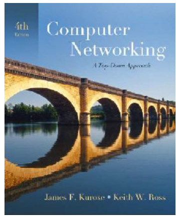
Chapter 4 Network Layer



A note on the use of these ppt slides:
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:
- If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we'd like people to use our book!)
- If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

Computer Networking: A Top Down Approach
4th edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2007.

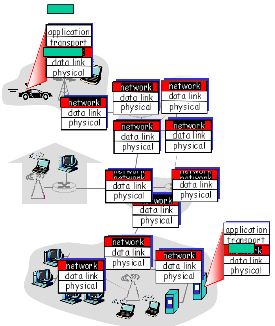
Chapter 4: Network Layer

Chapter goals:
- understand principles behind network layer services:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - routing (path selection)
  - dealing with scale
- instantiation, implementation in the Internet

Chapter 4: Network Layer

Network layer

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

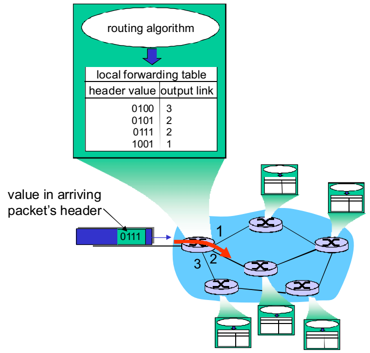


Two Key Network-Layer Functions

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

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

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 inervening 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

Chapter 4: Network Layer

4. 1 Introduction
4.2 Virtual circuit and datagram networks
4.3 What's inside a router
4.4 IP: Internet Protocol
- Datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 Routing algorithms
- Link state
- Distance Vector
- Hierarchical routing

4.6 Routing in the Internet
- RIP
- OSPF
- BGP

4.7 Broadcast and multicast routing

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

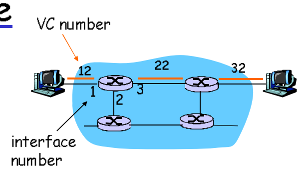
- call setup, teardown for each call *before* data can flow
- each packet carries VC identifier (not destination host address)
- *every* router on source-dest path maintains "state" for each passing connection
- link, router resources (bandwidth, buffers) may be *allocated* to VC (dedicated resources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



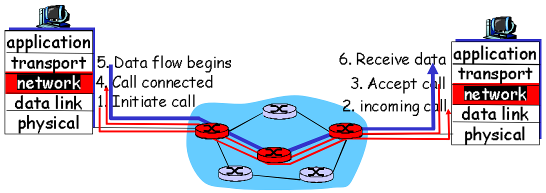
Forwarding table in northwest router:

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!

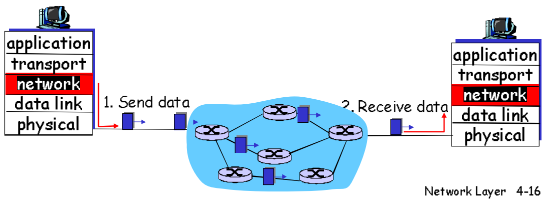
Virtual circuits: signaling protocols

- used to setup, maintain teardown VC
- used in ATM, frame-relay, X.25



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

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?

Datagram or VC network: why?

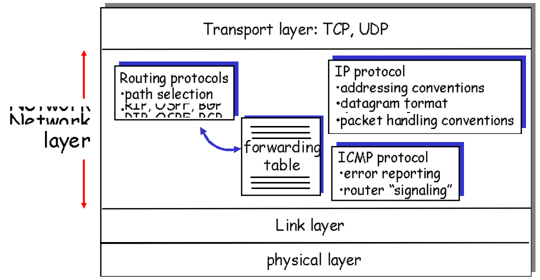
Internet (datagram)
- data exchange among computers
  - "elastic" service, no strict timing req.
- "smart" end systems (computers)
  - can adapt, perform control, error recovery
  - simple inside network, complexity at "edge"
- many link types
  - different characteristics
  - uniform service difficult

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

Chapter 4: Network Layer

The Internet Network layer

Host, router network layer functions:



Chapter 4: Network Layer

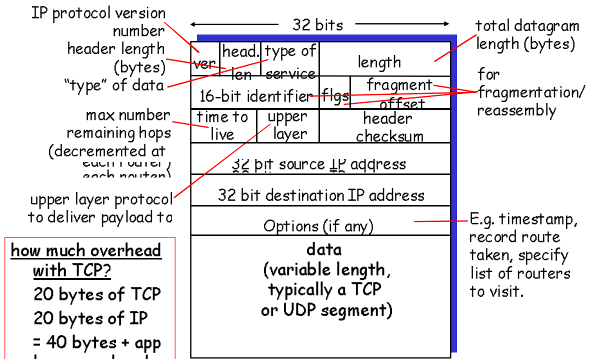
4. 1 Introduction
4.2 Virtual circuit and datagram networks
4.3 What's inside a router
4.4 IP: Internet Protocol
- Datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 Routing algorithms
- Link state
- Distance Vector
- Hierarchical routing

4.6 Routing in the Internet
- RIP
- OSPF
- BGP

4.7 Broadcast and multicast routing

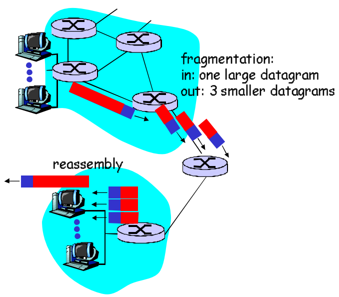
IP datagram format



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

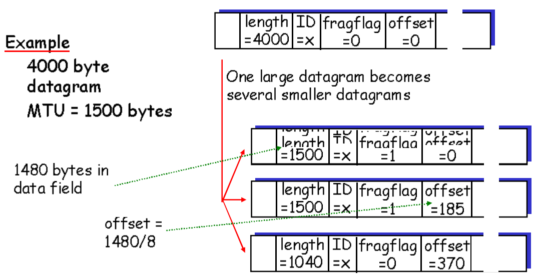
IP Fragmentation & Reassembly

- 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



Chapter 4: Network Layer

4. 1 Introduction
4.2 Virtual circuit and datagram networks
4.3 What's inside a router
4.4 IP: Internet Protocol
- Datagram format
- IPv4 addressing
- ICMP
- IPv6

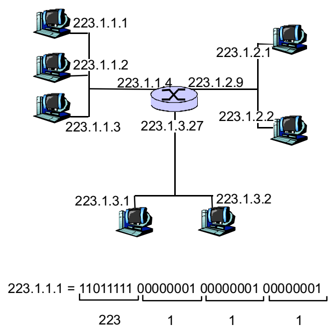
4.5 Routing algorithms
- Link state
- Distance Vector
- Hierarchical routing

4.6 Routing in the Internet
- RIP
- OSPF
- BGP

4.7 Broadcast and multicast routing

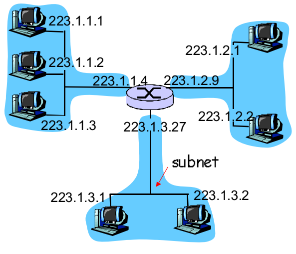
IP Addressing: introduction

- 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



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

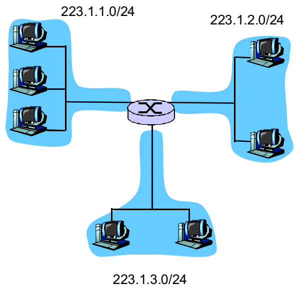


network consisting of 3 subnets

Subnets

Recipe

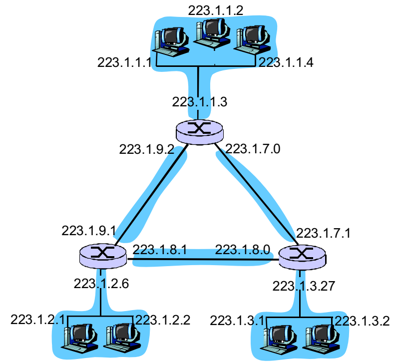
- To determine the subnets, detach each interface from its host or router, creating islands of isolated networks. Each isolated network is called a subnet.



Subnet mask: /24

Subnets

How many?



IP addressing: CIDR

CIDR: Classless InterDomain Routing

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

subnet part ← → host part

11001000 00010111 00010000 00000000

200.23.16.0/23

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: dynamically get address from as server
  - "plug-and-play"

IP addresses: how to get one?

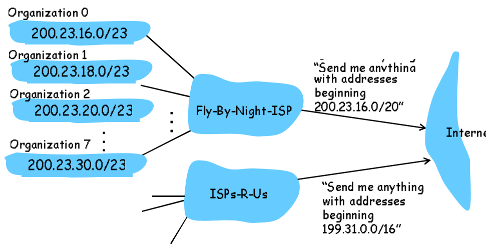
Q: How does *network* get subnet part of IP addr?

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

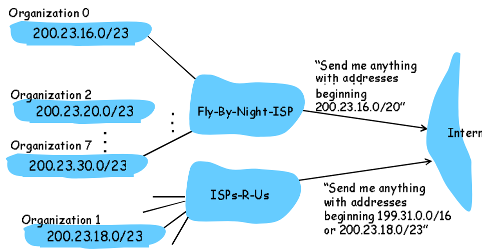
Hierarchical addressing: route aggregation

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

Chapter 4: Network Layer

ICMP: Internet Control Message Protocol

- used by hosts & routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Traceroute and ICMP

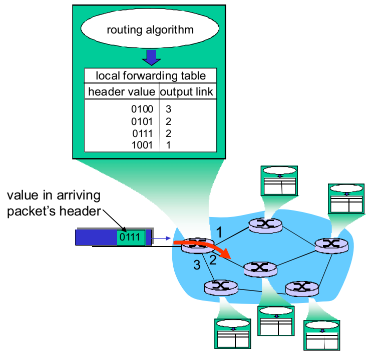
- Source sends series of UDP segments to dest
  - First has TTL =1
  - Second has TTL=2, etc.
  - Unlikely port number
- When nth datagram arrives to nth router:
  - Router discards datagram
  - And sends to source an ICMP message (type 11, code 0)
  - Message includes name of router& IP address

- When ICMP message arrives, source calculates RTT
- Traceroute does this 3 times

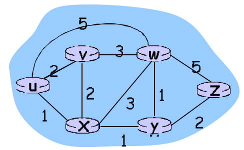
Stopping criterion
- UDP segment eventually arrives at destination host
- Destination returns ICMP "host unreachable" packet (type 3, code 3)
- When source gets this ICMP, stops.

Chapter 4: Network Layer

Interplay between routing, forwarding



Graph abstraction



Graph: G = (N,E)

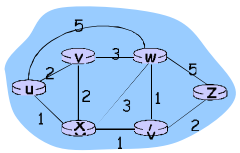
N = set of routers = { u, v, w, x, y, z }

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 $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

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

Chapter 4: Network Layer

Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

Define

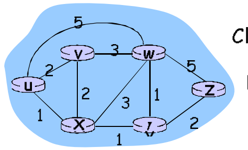
$d_x(y)$:= cost of least-cost path from x to y

Then

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

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:

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

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:

- Each node periodically sends its own distance vector estimate to neighbors
- 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

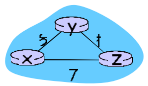
node y table

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

time →



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

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

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

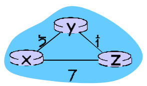
node z table

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

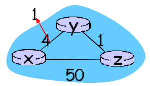
time →



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.

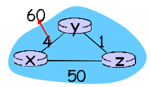
At time t_1, *z* receives the update from *y* and updates its table. It computes a new least cost to *x* and sends its neighbors its DV

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?

Chapter 4: Network Layer

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

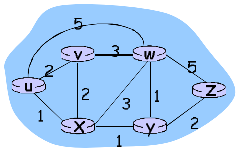
Notation:
- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijsktra's Algorithm

```
1  Initialization:
2   N' = {u}
3   for all nodes v
4     if v adjacent to u
5       then D(v) = c(u,v)
6     else D(v) = ∞
7
8  Loop
9   find w not in N' such that D(w) is a minimum
10  add w to N'
11  update D(v) for all v adjacent to w and not in N' :
12     D(v) = min( D(v), D(w) + c(w,v) )
13  /* new cost to v is either old cost to v or known
14   shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```

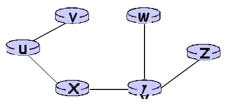

Dijkstra's algorithm: example

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:



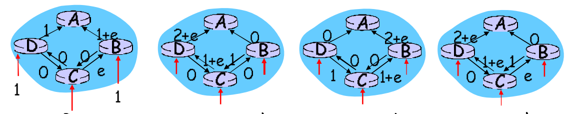
Resulting forwarding table in u:

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: $O(n\log n)$

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



initially … recompute routing … recompute … recompute

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: $O(n^2)$ 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

Chapter 4: Network Layer

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)

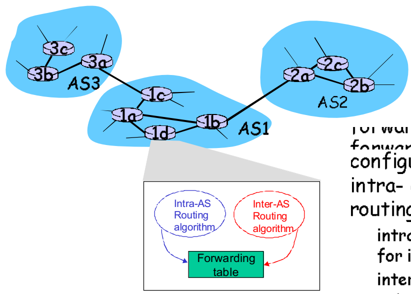
routers in same AS run same routing protocol

- "intra-AS" routing protocol
- routers in different AS can run different intra-AS routing protocol

Gateway router

Direct link to router in another AS

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

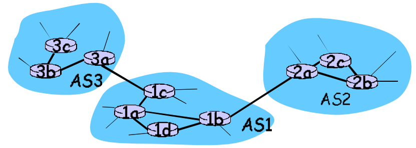
suppose router in AS1 receives datagram dest outside of AS1

- router should forward packet to gateway router, but which one?

AS1 must:

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

Job of inter-AS routing!



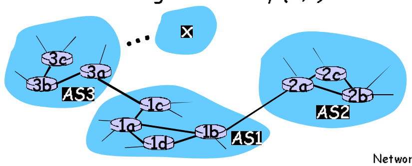
Example: Setting forwarding table in router 1d

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)*

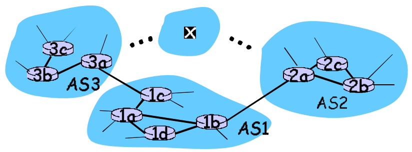


Example: Choosing among multiple ASes

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!



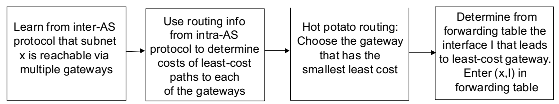
Example: Choosing among multiple ASes

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.



Chapter 4: Network Layer

4. 1 Introduction

4.2 Virtual circuit and datagram networks

4.3 What's inside a router

4.4 IP: Internet Protocol

- Datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 Routing algorithms

- Link state
- Distance Vector
- Hierarchical routing

4.6 Routing in the Internet

- RIP
- OSPF
- BGP

4.7 Broadcast and multicast routing

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)

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"*

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:

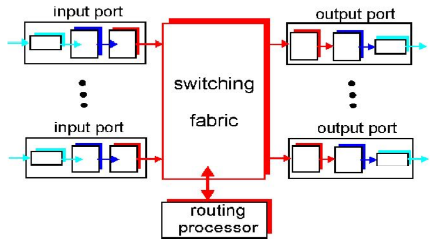
- Intra-AS: can focus on performance
- Inter-AS: policy may dominate over performance

Chapter 4: Network Layer

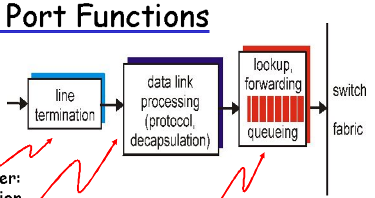
Router Architecture Overview

Two key router functions:

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



Input Port Functions



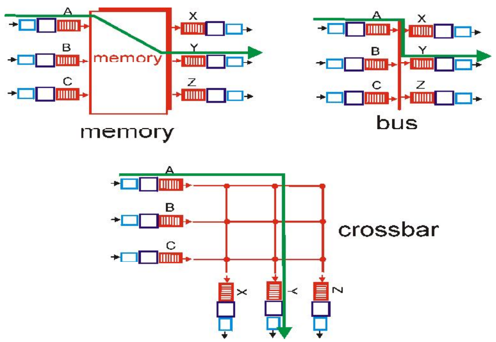
Physical layer: bit-level reception

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

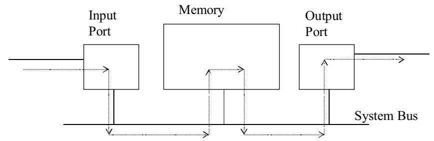
Three types of switching fabrics



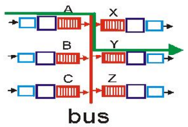
Switching Via Memory

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

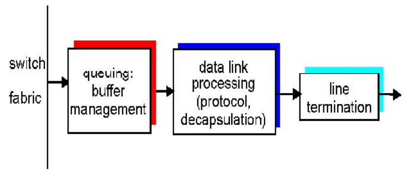


- datagram from input port memory to output port memory via a shared bus
- bus contention: switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers

Switching Via An Interconnection Network

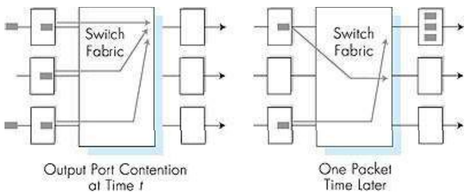
- overcome bus bandwidth limitations
- Banyan networks, other interconnection nets initially developed to connect processors in multiprocessor
- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection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

Output Ports



- *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!*