

Telematics I

Chapter 7

Network Layer

Acknowledgement: These slides have been prepared by J.F. Kurose and K.W. Ross with a couple of additions from various sources (see references)



Goals of this Chapter

- ❑ Building larger networks by simply interconnecting LANs is limited, it does not scale
- ❑ To build larger networks, the following questions have to be explicitly solved:
 - ❑ What are good paths that a packet should take to get from a source node to a destination node?
 - ❑ How to represent these paths by *routing tables* and how to construct them efficiently?
 - ❑ How to *use* routing tables (once constructed) efficiently?
 - ❑ How to organize larger networks with respect to an addressing structure that allows efficient & compact routing tables?
- ❑ In addition, we will look at:
 - ❑ two main service models for the network layer virtual circuits vs. datagram
 - ❑ the structure of a router
 - ❑ the Internet Protocol (IP)
 - ❑ how the Internet's routing structure looks like as a case study



- ❑ 7.1 Introduction
- ❑ 7.2 Virtual circuit and datagram networks
- ❑ 7.3 What's inside a router
- ❑ 7.4 IP: Internet Protocol
 - ❑ Datagram format
 - ❑ IPv4 addressing
 - ❑ ICMP
 - ❑ IPv6
- ❑ 7.5 Routing algorithms
 - ❑ Overview
 - ❑ Distance Vector
 - ❑ Link state
 - ❑ Hierarchical routing
- ❑ 7.6 Routing in the Internet
 - ❑ RIP
 - ❑ OSPF
 - ❑ BGP



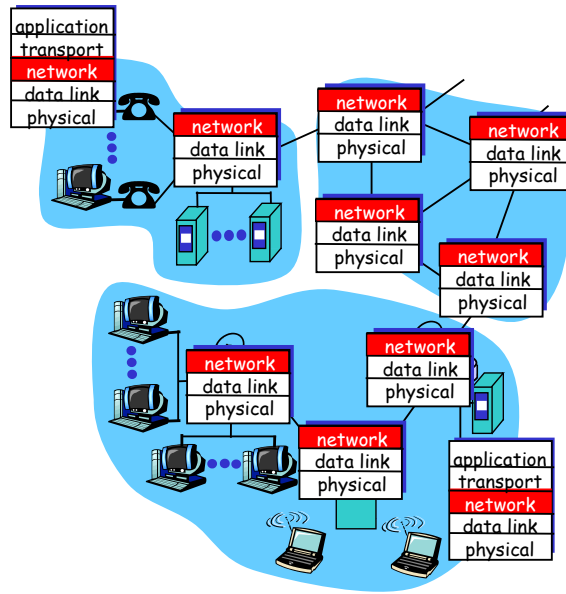
Goal: Build Large Networks

- ❑ The “internetworking” mechanisms described in previous chapter can help packets to reach their destination
 - ❑ Hubs: broadcast
 - ❑ Switch: send to directly connected network
 - ❑ Bridge: Flooding causes problems
 - Spanning tree solves some of them
- ❑ Can we directly extend these mechanisms to large networks (millions or more nodes)?
 - ❑ Flooding clearly not a good idea
 - ❑ Need some structure to decide where a packet should go
 - ❑ Spanning tree is actually a good start:
 - Once spanning tree for a destination is known, it is clear how to send a packet to this destination
 - However, once a link goes down, the tree gets disconnected
⇒ insufficient fault tolerance



Network Layer: The Big Picture

- ❑ 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



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



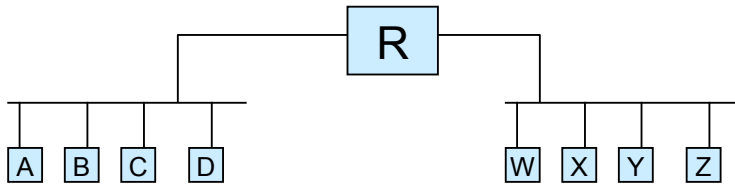
Forwarding

Question: How do you get a packet from one network to another?

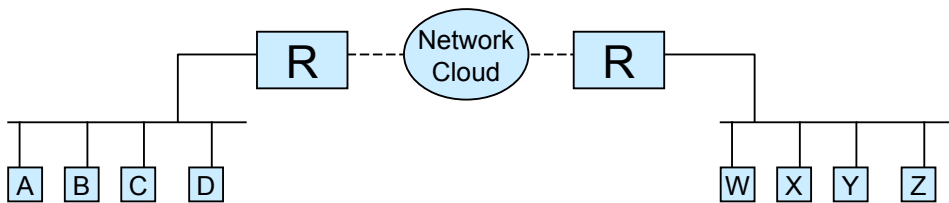


Answer: with a switch / bridge / router (or a series of them)

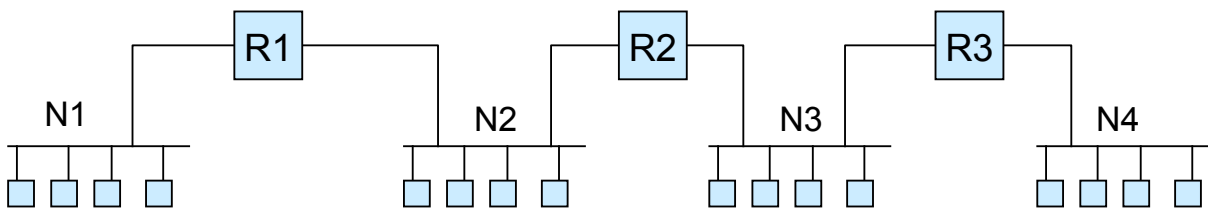
Case 1:
Single hop



Case 2:
Multi-hop



Forwarding: Example



routing table @ R2

Dest	Next hop
N1	R1
N2	Deliver directly
N3	Deliver directly
N4	R3

Actual routing table contains IP addresses, flags indicating type of entries, net mask etc.

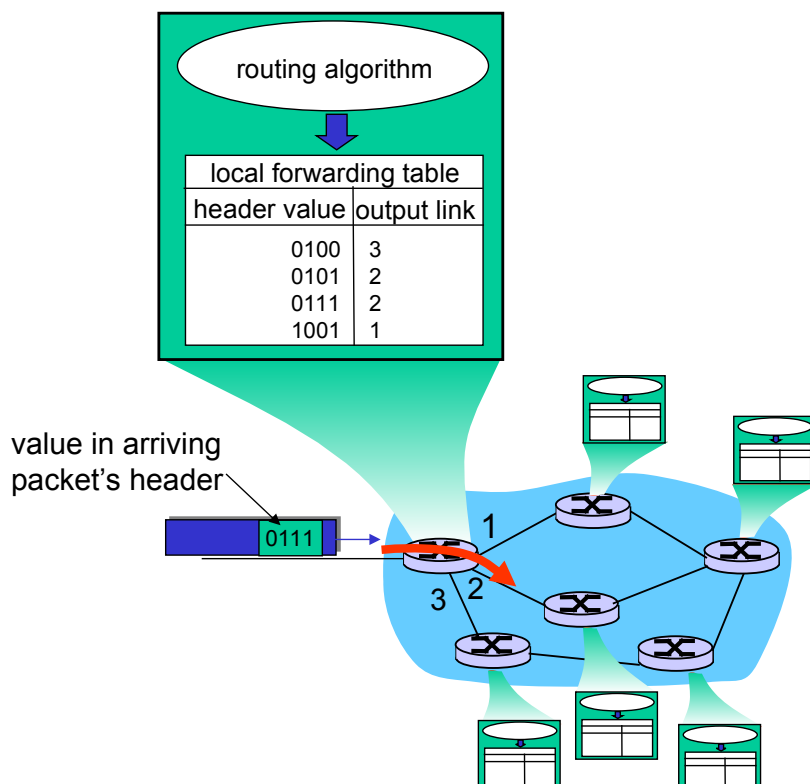


Searching the Routing Table

- ❑ First, search for a matching host address
 - ❑ Flag H is set
- ❑ Second, search for a matching network address
 - ❑ Need to know the number of bits to use for network ID
- ❑ Third, search for a default entry
 - Execute `netstat -rn` on your machine and find the contents of the routing table
 - Default entry allows for a single entry for a list of entries that have the same next-hop value
- ❑ This leads to the interesting question:
 - ❑ How is the information stored in this routing / forwarding table obtained?
 - ❑ Answer: this is the task of a *routing algorithm* (more later)



Interplay Between Routing and Forwarding (Summary)



- ❑ 3rd important function in *some* network architectures:
 - ❑ ATM, Frame Relay, X.25
- ❑ Before datagrams flow, two hosts and intervening routers establish virtual connection
 - ❑ Routers get involved and establish state for the virtual connection
- ❑ Network and transport layer connection service:
 - ❑ **Network:** between two hosts
 - ❑ **Transport:** between two processes

- ❑ In the Internet, there is no connection setup in the network layer



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 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	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no



- ❑ 7.1 Introduction
- ❑ **7.2 Virtual circuit and datagram networks**
- ❑ 7.3 What's inside a router
- ❑ 7.4 IP: Internet Protocol
 - ❑ Datagram format
 - ❑ IPv4 addressing
 - ❑ ICMP
 - ❑ IPv6
- ❑ 7.5 Routing algorithms
 - ❑ Overview
 - ❑ Distance Vector
 - ❑ Link state
 - ❑ Hierarchical routing
- ❑ 7.6 Routing in the Internet
 - ❑ RIP
 - ❑ OSPF
 - ❑ BGP



- ❑ 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 the core



“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

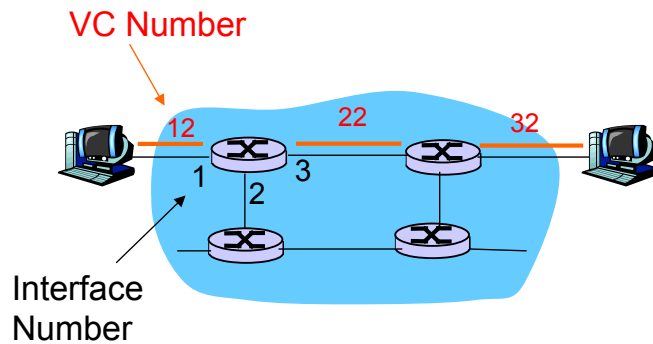


- ❑ A VC consists of:
 - ❑ Path from source to destination
 - ❑ VC numbers, one number for each link along path
 - ❑ Entries in forwarding tables in routers along path
- ❑ Packet belonging to VC carries a VC number.
- ❑ VC number must be changed on each link.
 - ❑ New VC number comes from forwarding table



Forwarding Table

Forwarding table in northwest router:

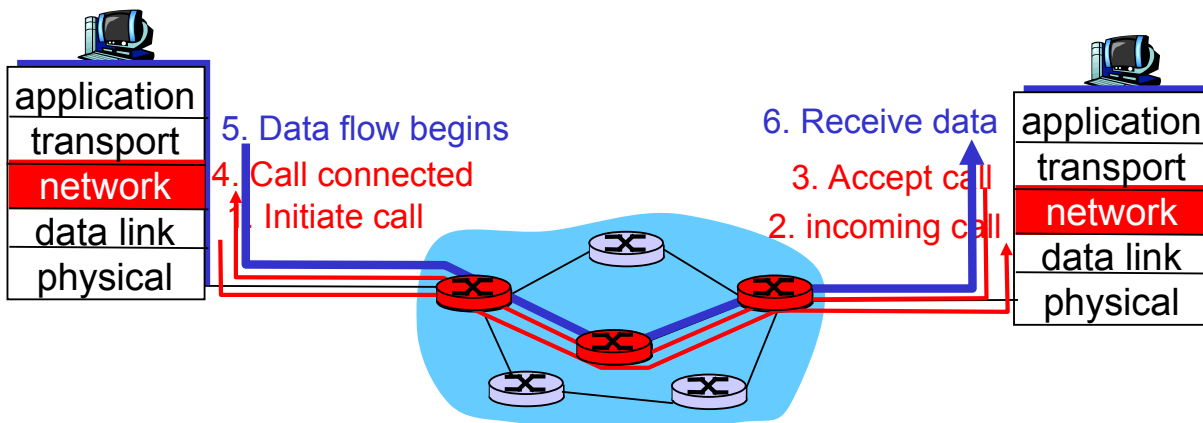


Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
...

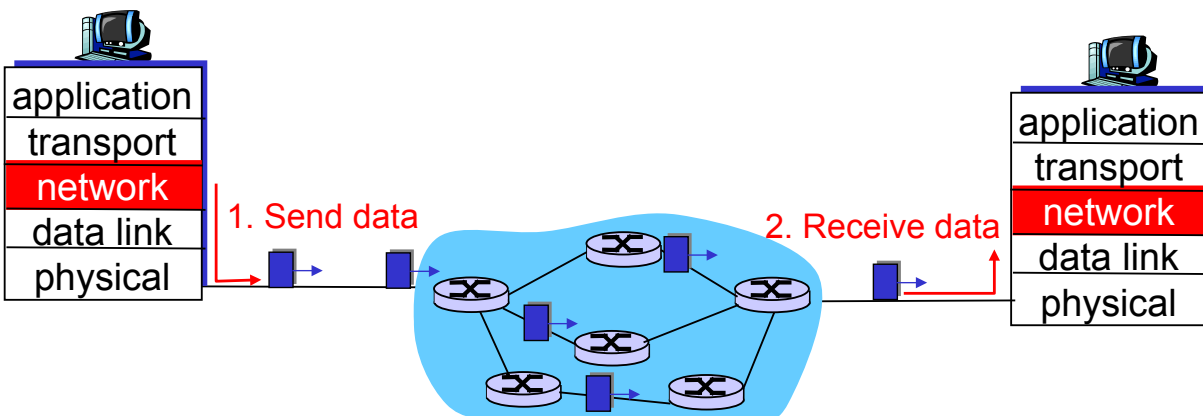
Routers maintain connection state information!



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



- ❑ 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

<u>Destination Address Range</u>	<u>Link Interface</u>
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

4 billion possible entries!



Longest Prefix Matching

<u>Prefix Match</u>	<u>Link Interface</u>
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?**



Internet

- ❑ 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

- ❑ Evolved from telephony
- ❑ Human conversation:
 - ❑ Strict timing, reliability requirements
 - ❑ Need for guaranteed service
- ❑ “Dumb” end systems
 - ❑ telephones
 - ❑ complexity inside network



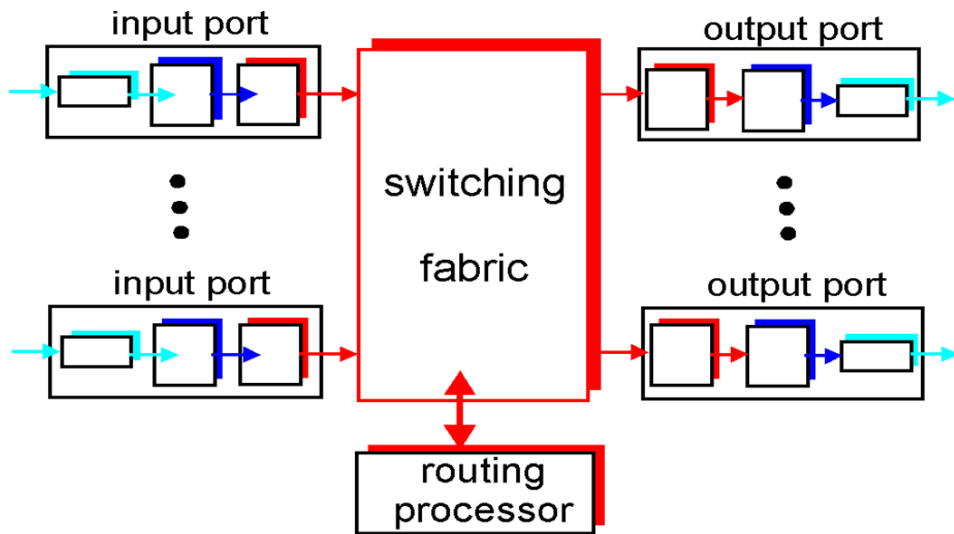
- ❑ 7.1 Introduction
- ❑ 7.2 Virtual circuit and datagram networks
- ❑ **7.3 What's inside a router**
- ❑ 7.4 IP: Internet Protocol
 - ❑ Datagram format
 - ❑ IPv4 addressing
 - ❑ ICMP
 - ❑ IPv6
- ❑ 7.5 Routing algorithms
 - ❑ Overview
 - ❑ Distance Vector
 - ❑ Link state
 - ❑ Hierarchical routing
- ❑ 7.6 Routing in the Internet
 - ❑ RIP
 - ❑ OSPF
 - ❑ BGP



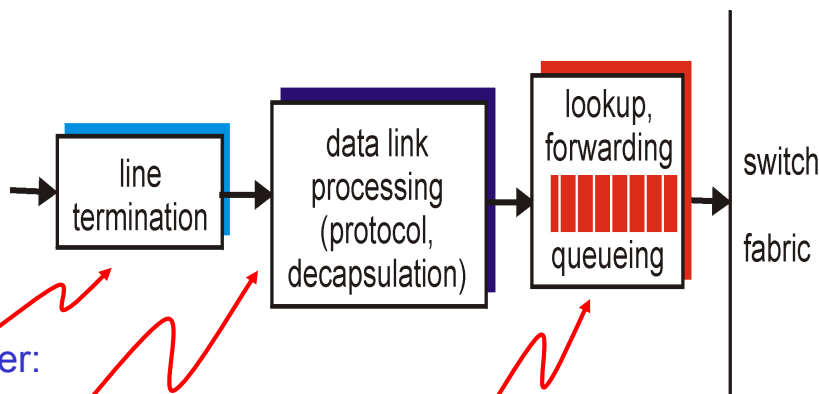
Router Architecture Overview

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

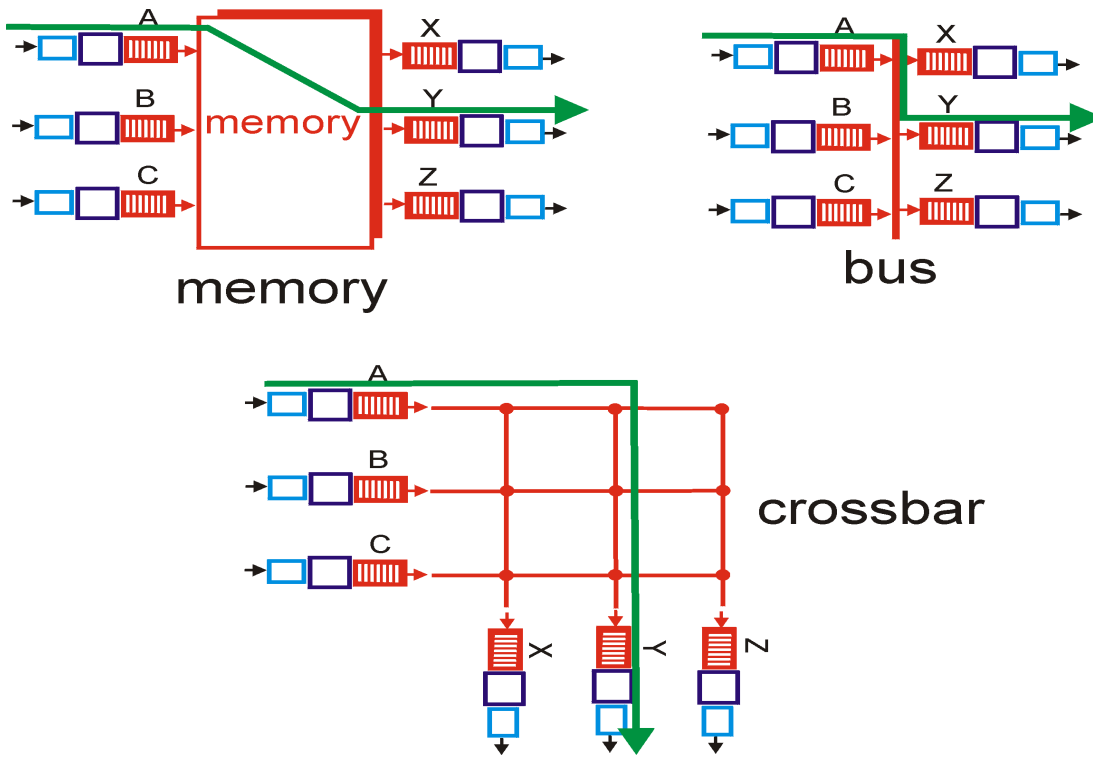
Data link layer:
e.g., Ethernet
see chapter 5

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



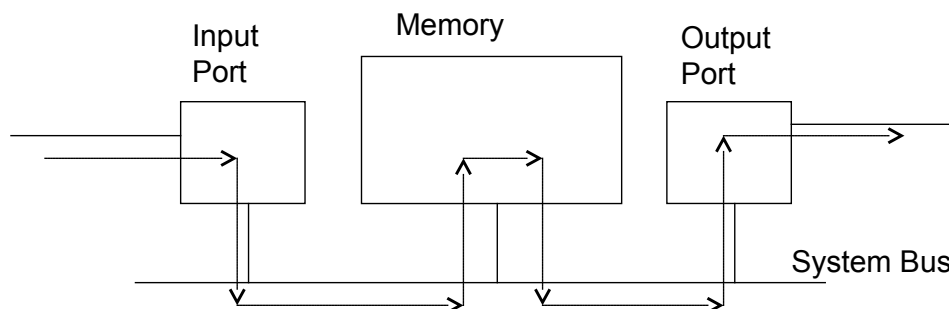
Three Types of Switching Fabrics



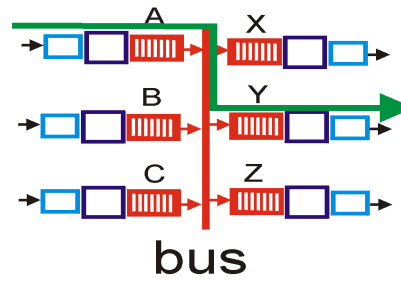
Switching Via Memory

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

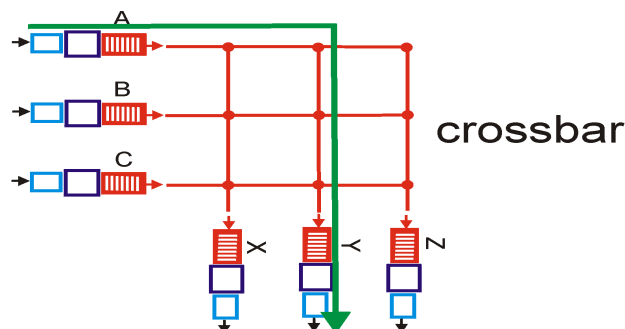


- ❑ Datagram from input port memory to output port memory via a shared bus
- ❑ **Bus contention:** switching speed limited by bus bandwidth
- ❑ 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)

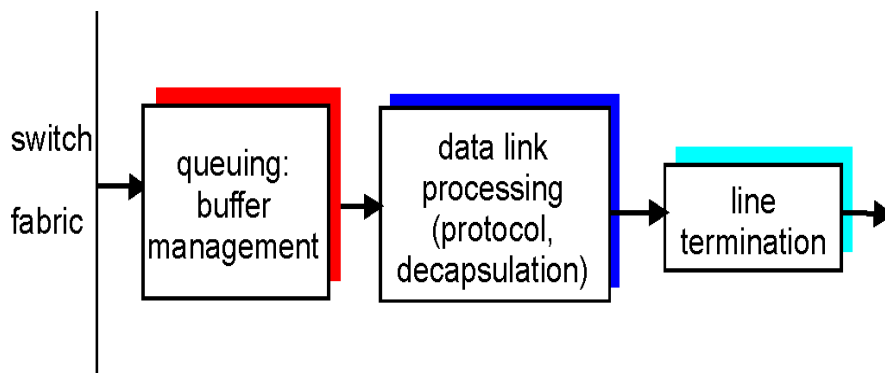


Switching Via An Interconnection Network

- ❑ 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 series: switches Gigabits per second (Gbps) through the interconnection network
 - ❑ Cisco 12816: up to 1.28 Tbps, 16 slots with 40 Gbps (full duplex)



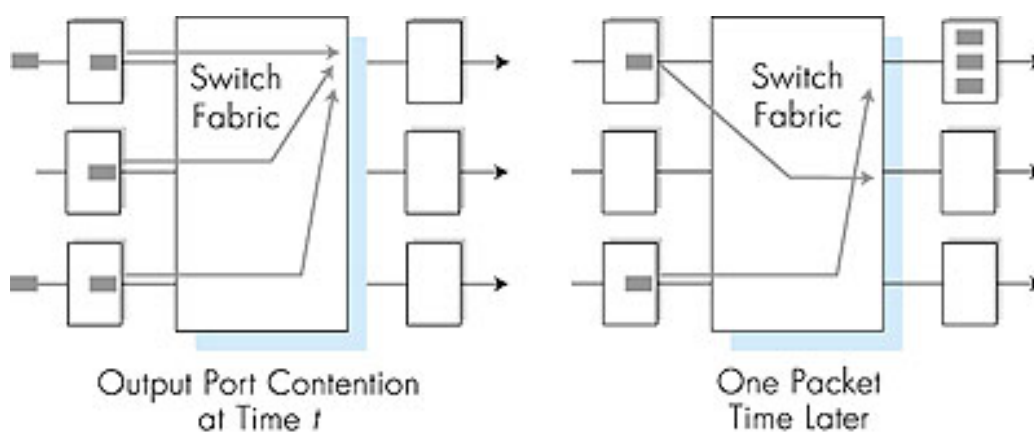
Output Ports



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



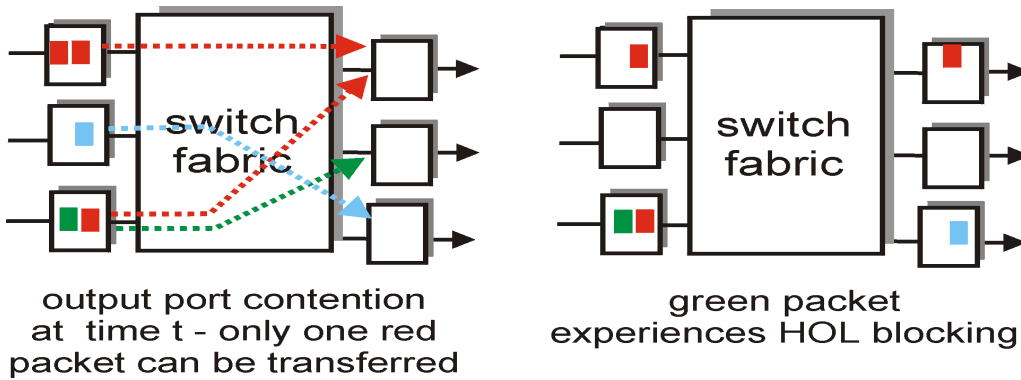
Output Port Queuing



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



- ❑ 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!*

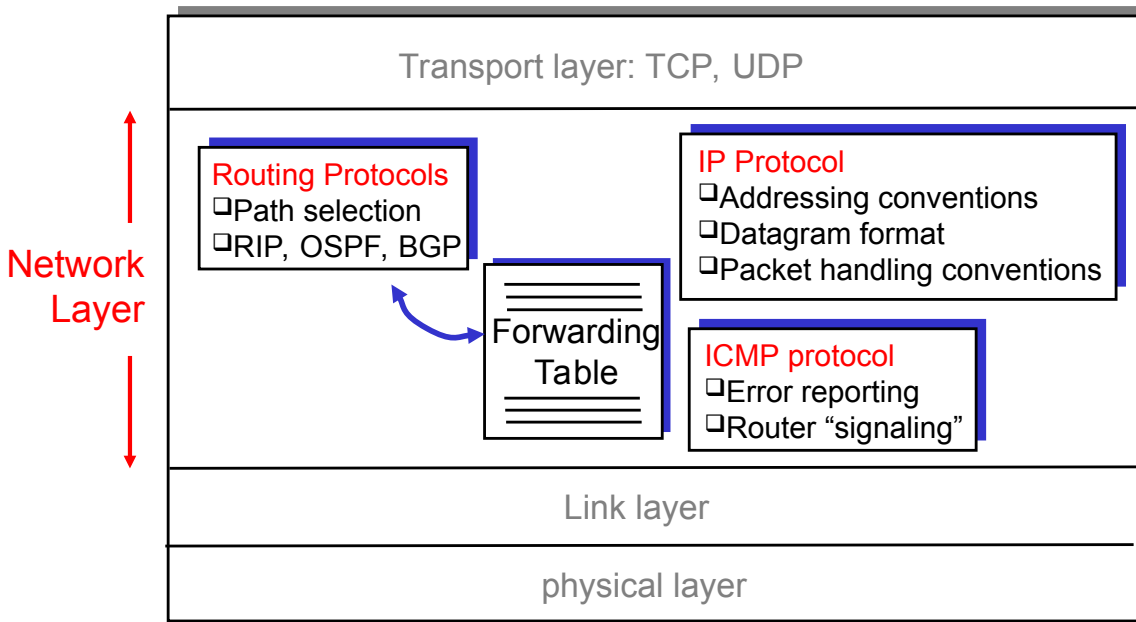


- ❑ 7.1 Introduction
- ❑ 7.2 Virtual circuit and datagram networks
- ❑ 7.3 What's inside a router
- ❑ **7.4 IP: Internet Protocol**
 - ❑ **Datagram format**
 - ❑ IPv4 addressing
 - ❑ ICMP
 - ❑ IPv6
- ❑ 7.5 Routing algorithms
 - ❑ Overview
 - ❑ Distance Vector
 - ❑ Link state
 - ❑ Hierarchical routing
- ❑ 7.6 Routing in the Internet
 - ❑ RIP
 - ❑ OSPF
 - ❑ BGP

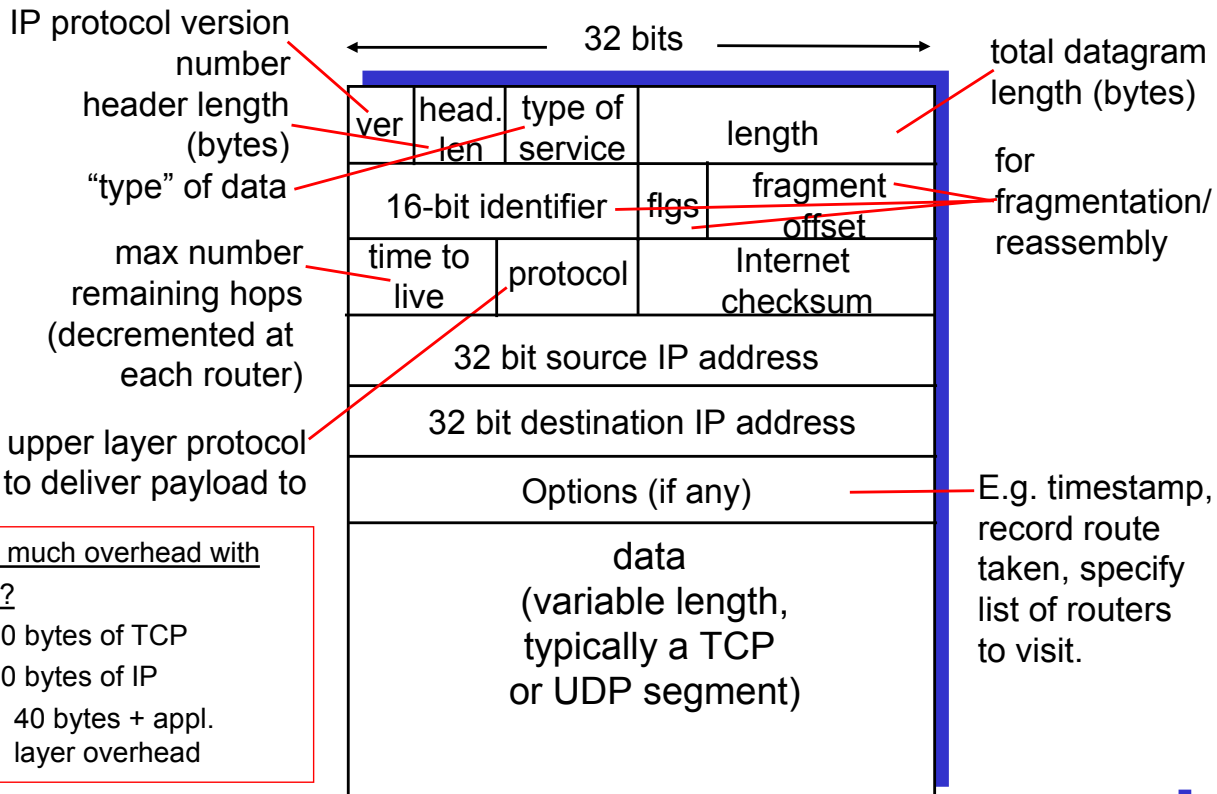


The Internet Network Layer

Host & router network layer functions:



IP Packet Format (1)



IP Packet Format (2)

- ❑ *Version*: the IP version number (currently still 4 even though 6 exists)
- ❑ *IHL*: IP Header Length in 32-bit words
- ❑ *Type of Service*: contains priority information, rarely used
- ❑ *Total Length*: the total length of the datagram in bytes (incl. header)
- ❑ *Identification*: when an IP packet is segmented into multiple fragments, each fragment is given the same identification; this field is used to reassemble fragments
- ❑ *Flags*:
 - ❑ *DF*: Don't Fragment
 - ❑ *MF*: More Fragments; when a packet is fragmented, all fragments except the last one have this bit set
- ❑ *Fragment Offset*: the fragment's position within the original packet (specified in units of 8 octets)
- ❑ *Time to Live*: hop count, decremented each time the packet reaches a new router; when hop count = 0, packet is discarded



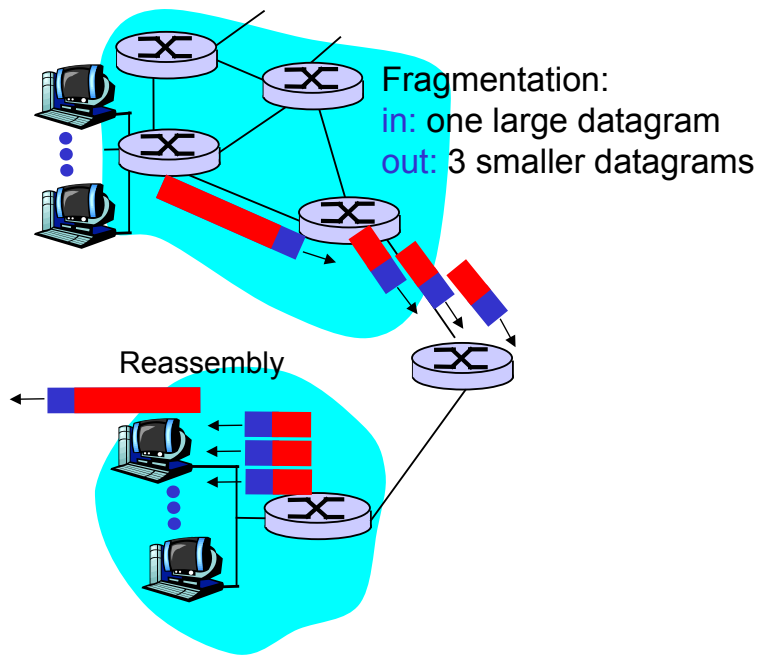
IP Packet Format (3)

- ❑ *Protocol*: identifies which transport layer protocol is being used for this packet (most of the time: either TCP or UDP)
- ❑ *Header Checksum*: allows to verify the contents of the IP header (not polynomial-based)
- ❑ *Source and Destination Addresses*: uniquely identify sender and receiver of the packet
- ❑ *Options*: up to 40 bytes in length; used to extend functionality of IP (examples: source routing, record route)
- ❑ *IP addresses*:
 - ❑ 32 bits long (4 bytes)
 - ❑ Each byte is written in decimal in MSB order, separated by decimals (example: 128.195.1.80)
 - ❑ 0.0.0.0 (lowest) to 255.255.255.255 (highest)
 - ❑ Address Classes: Class A, B, C, D, E, Loopback, Broadcast



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	ID	fragflag	offset
=4000	=x	=0	=0

One large datagram becomes several smaller datagrams

1480 bytes in data field

offset = 1480/8

length	ID	fragflag	offset
=1500	=x	=1	=0

length	ID	fragflag	offset
=1500	=x	=1	=185

length	ID	fragflag	offset
=1040	=x	=0	=370



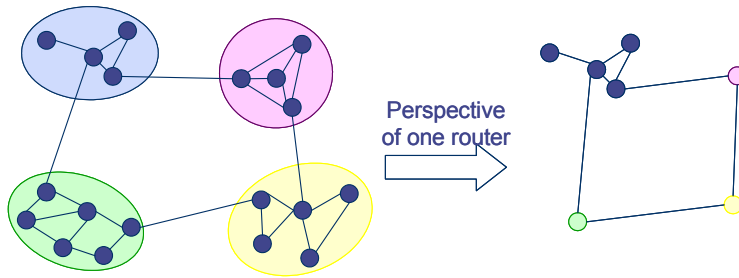
- ❑ 7.1 Introduction
- ❑ 7.2 Virtual circuit and datagram networks
- ❑ 7.3 What's inside a router
- ❑ **7.4 IP: Internet Protocol**
 - ❑ Datagram format
 - ❑ **IPv4 addressing**
 - ❑ ICMP
 - ❑ IPv6
- ❑ 7.5 Routing algorithms
 - ❑ Overview
 - ❑ Distance Vector
 - ❑ Link state
 - ❑ Hierarchical routing
- ❑ 7.6 Routing in the Internet
 - ❑ RIP
 - ❑ OSPF
 - ❑ BGP



- ❑ Think back to the MAC/LLC layer: Each device has a globally unique MAC address
 - ❑ Why is there then a need to talk about some other addressing scheme?
 - ❑ How did spanning tree algorithm for bridges work?
 - ❑ Each bridge had to store a **separate entry** for each device to which it was routing packets
 - ❑ Lot's of memory, CPU overhead (for searching)
- ! Clearly, this does not scale to large networks



- ❑ “Flat” addresses – addresses that express no structure – do not work well together with hierarchical routing

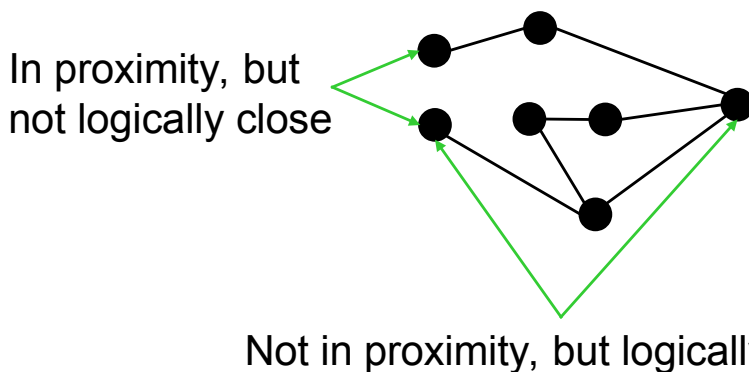


- ❑ Necessary: Addresses that express/respect the hierarchical routing structure
 - ❑ Essentially, something like:
Group-ID_n:Group-ID_{n-1}:...:Group-ID₁:Device-ID
 - ❑ **Hierarchical addresses** – addresses are relative to higher groups

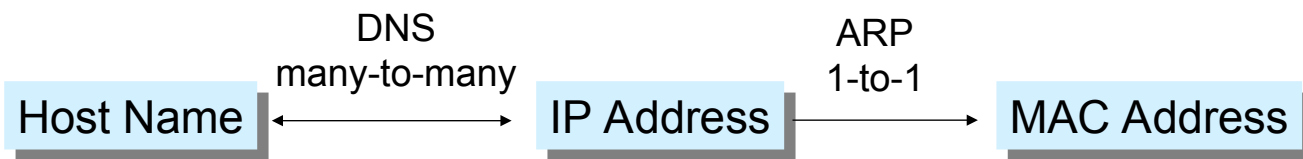


“Closeness”

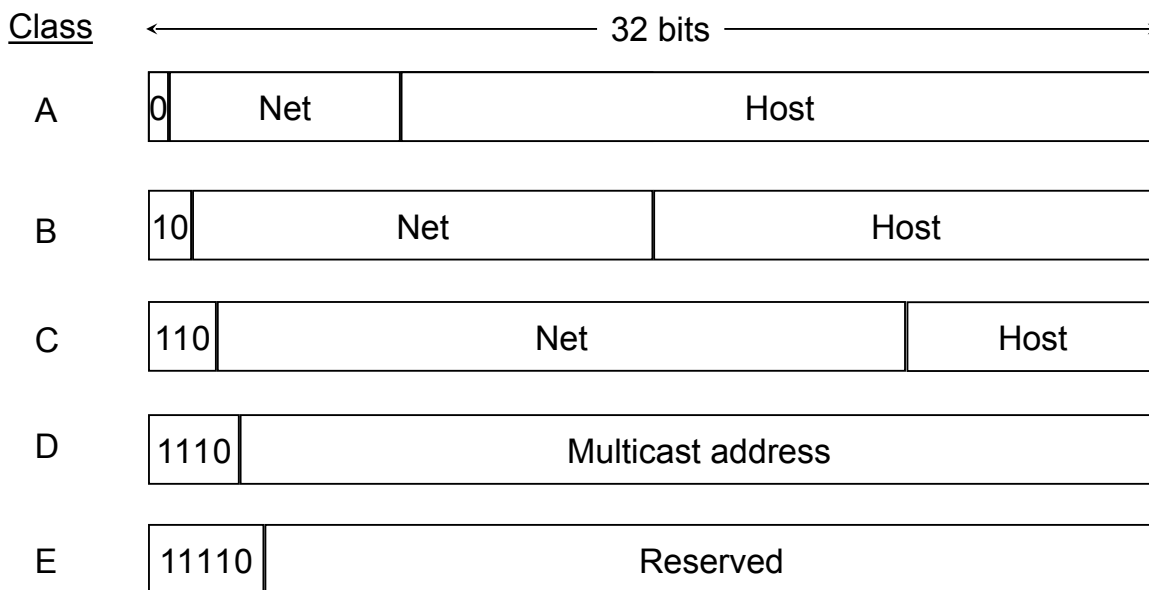
- ❑ A note of caution: Closeness <> proximity
 - ❑ Proximity: Nearby physical location
 - ❑ Closeness: Structurally, logically within a short distance (e.g., in number of hops)
 - Closeness is not a standard nomenclature



	Example	Organization
MAC address	8:0:20:72:93:18	flat, permanent
IP address	132.151.1.35	topological (mostly)
Host name	www.ietf.org	hierarchical



IP Address Classes (1)



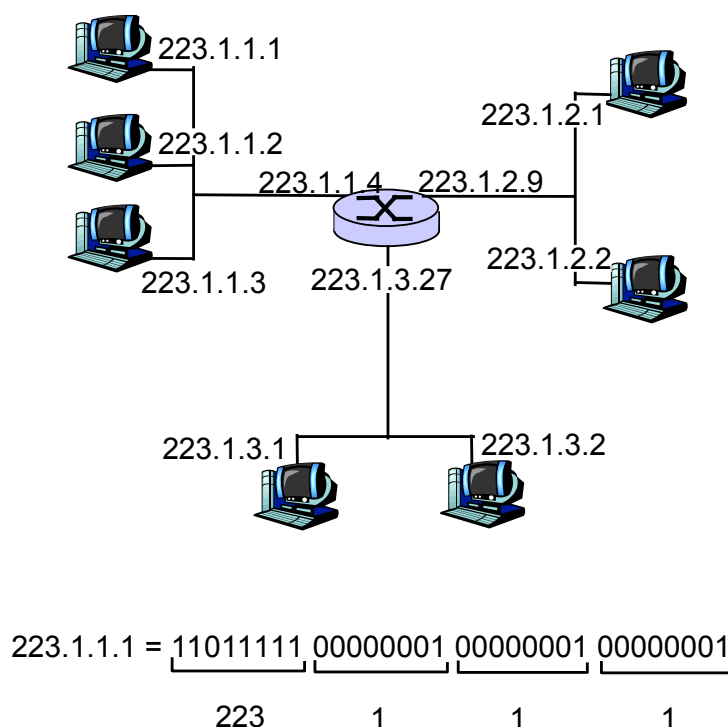
IP Address Classes (2)

- ❑ Class A: for very large organizations; 16 million hosts allowed
- ❑ Class B: for large organizations; 65 thousand hosts allowed
- ❑ Class C: for small organizations; 255 hosts allowed
- ❑ Class D: multicast addresses; no network/host hierarchy
- ❑ Class E: reserved
- ❑ Loopback: 127.xx.yy.zz (127.anything) is reserved for loopback testing; packets sent to this address are not put out onto the wire; they are processed locally and treated as incoming packets.
- ❑ Broadcast: all 1s
- ❑ Address Hierarchy:
 - ❑ Each address contains a network and a host portion, meaning two levels of hierarchy
 - ❑ Note that Class A, Class B, and Class C addresses only support two levels of hierarchy
 - ❑ However, the host portion can be further split into “subnets” by the address class owner

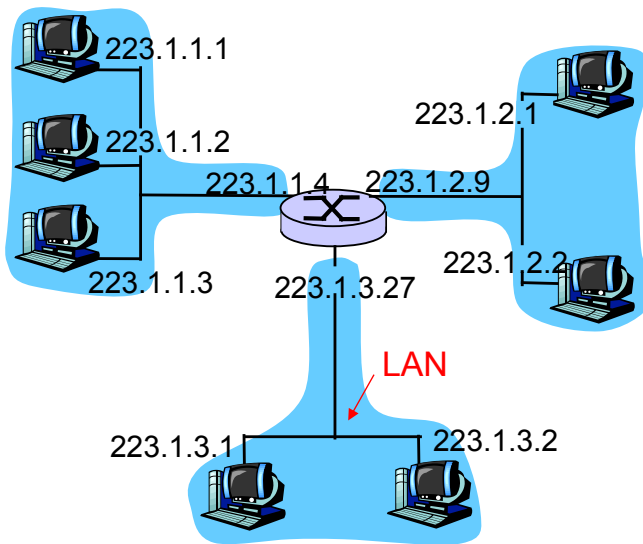


IP Addressing: Introduction

- ❑ **IP address:** 32-bit identifier for host, router *interface*
- ❑ **Interface:** connection between host/router and physical link
 - ❑ Router's typically have multiple interfaces
 - ❑ Host may have multiple interfaces
 - ❑ IP addresses associated with each interface



- ❑ 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

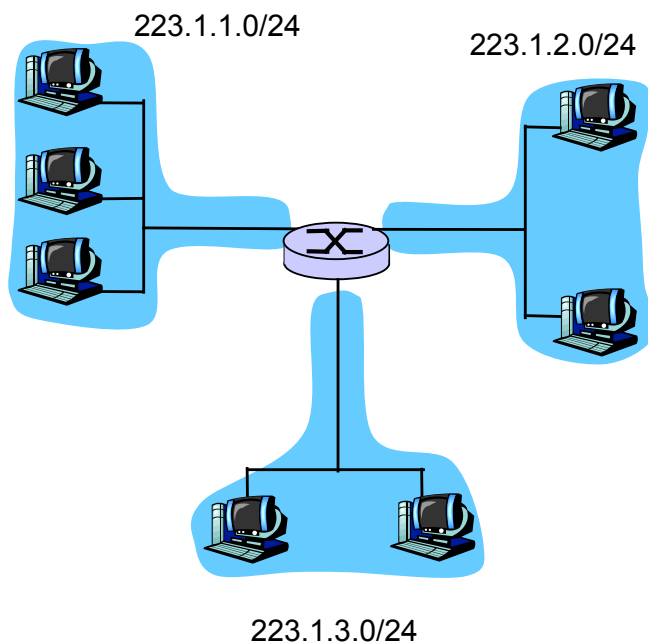


network consisting of 3 subnets



Recipe

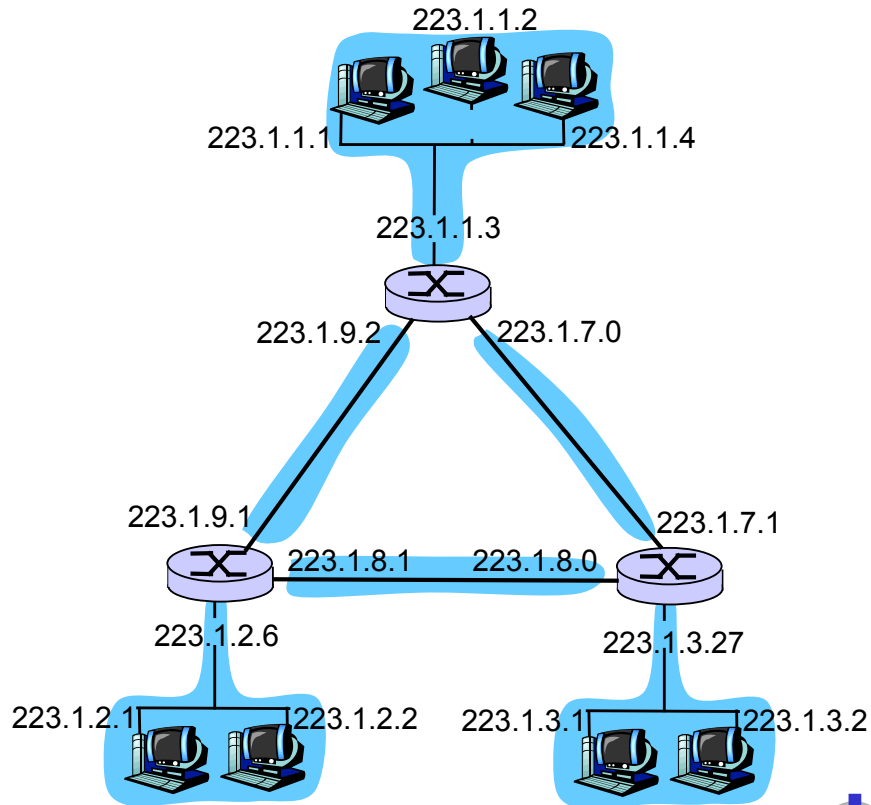
- ❑ To determine the subnets, detach each interface from its host or router, creating islands of isolated networks. Each isolated network is called a **subnet**.



Subnet mask: /24



How many?



IP Addresses (1)

Given the notion of “network”, let’s re-examine IP addresses:

“Class-full” addressing:

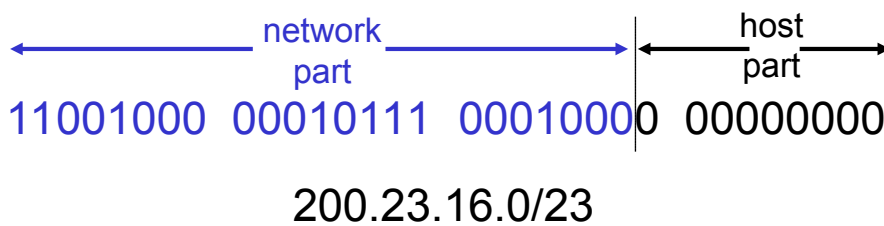
class

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 →



- ❑ Classful addressing:
 - ❑ Inefficient use of address space, address space exhaustion
 - ❑ E.g., class B net allocated enough addresses for 65K hosts, even if only 2K hosts in that network
- ❑ *CIDR: Classless InterDomain Routing*
 - ❑ Network portion of address of arbitrary length
 - ❑ Address format: a.b.c.d/x, where x is # bits in network portion of address



- ❑ Discard class boundaries → “supernetting”
- ❑ ISP assigns a contiguous group of class C blocks
- ❑ “Longest match routing” on masked address; e.g. 192.175.132.0/22

Address/mask	Next hop
192.175.132.0/22	1
192.175.133.0/23	2
192.175.128.0/17	3

- ❑ Example:
 - All sites in Europe common prefix
 - Only single entry in most U.S. routers
- ❑ Details:
 - RFC 1519: Supernet address extensions and CIDR



Q: How does *host* get IP address?

- ❑ Hard-coded by system admin in a file
 - ❑ Wintel (short for Windows/Intel):
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”



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

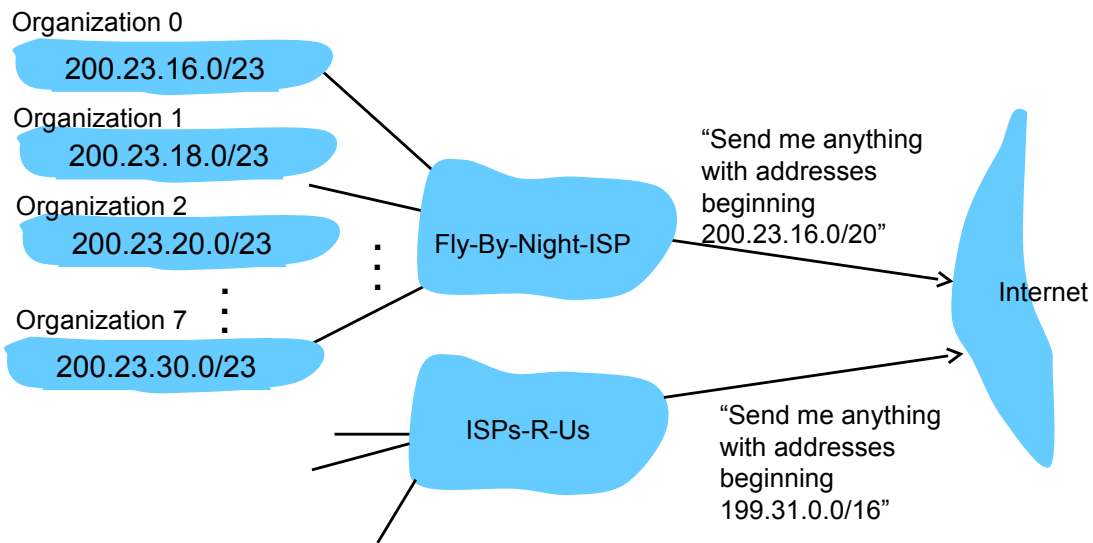
A: gets allocated portion of its provider ISP's address space

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



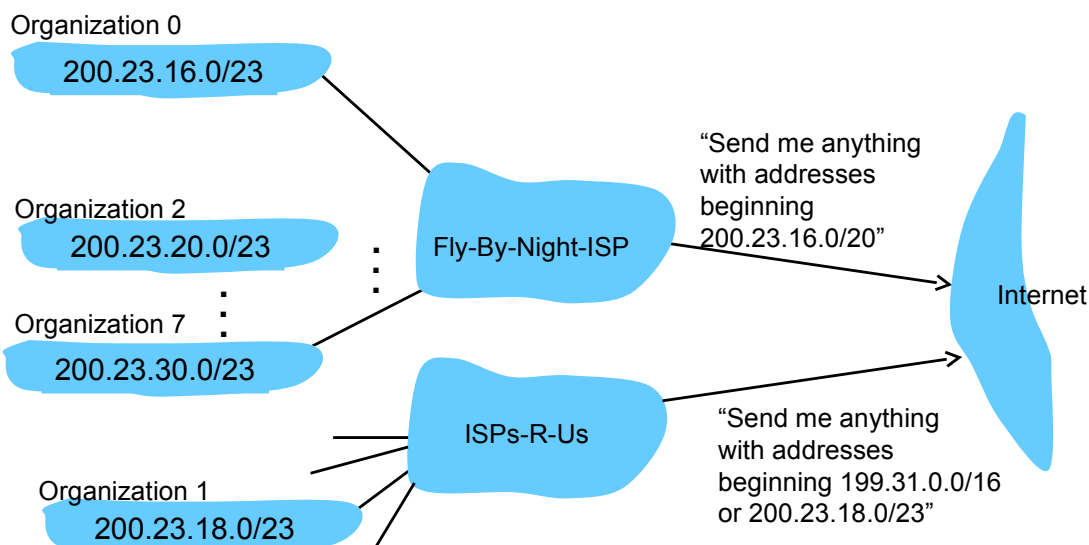
Hierarchical Addressing: Route Aggregation

Hierarchical addressing allows efficient advertisement of routing information:



Hierarchical Addressing: More Specific Routes

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



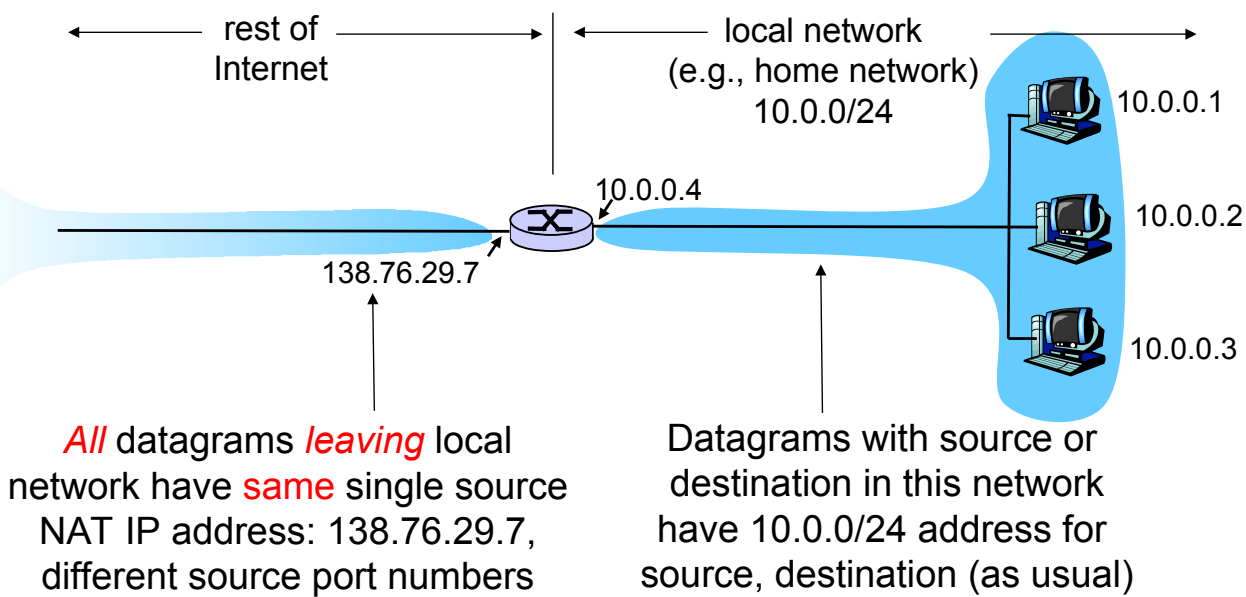
Q: How does an ISP get block of addresses?

A: **ICANN:** Internet Corporation for Assigned Names and Numbers

- ❑ Allocates addresses
- ❑ Manages DNS
- ❑ Assigns domain names, resolves disputes



NAT: Network Address Translation (1)



- ❑ **Main Motivation:** Shortage of IP addresses
- ❑ Basic Idea:
 - ❑ Local network uses just one IP address as far as outside world is concerned
 - ❑ No need to be allocated range of addresses from ISP - just one IP address is used for all devices
- ❑ Further Advantages:
 - ❑ 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)



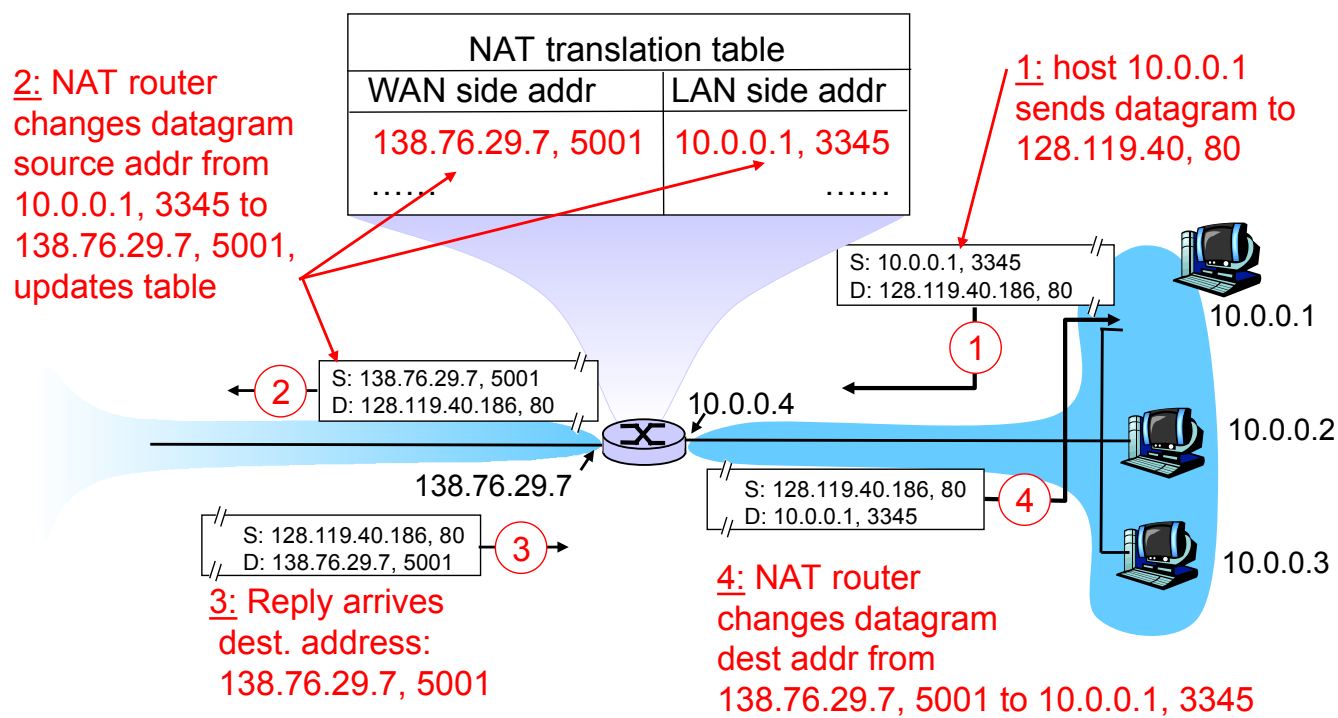
Implementation – NAT router must:

- ❑ **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 (4)



NAT: Network Address Translation (5)

- ❑ 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 principle
 - NAT possibility must be taken into account by app designers, eg, P2P applications
 - ❑ Address shortage should instead be solved by IPv6



- ❑ What happens once a packet arrives at its destination network / LAN?
 - ❑ How to turn an IP address (which is all that is known about the destination) into a MAC address that corresponds to the MAC address?
- ❑ Simple solution: Yell!
 - ❑ Broadcast on the LAN, asking which node has IP address x
 - ❑ Node answers with its MAC address
 - ❑ Router can then address packet to that MAC address
- ❑ **Address Resolution Protocol (ARP)**



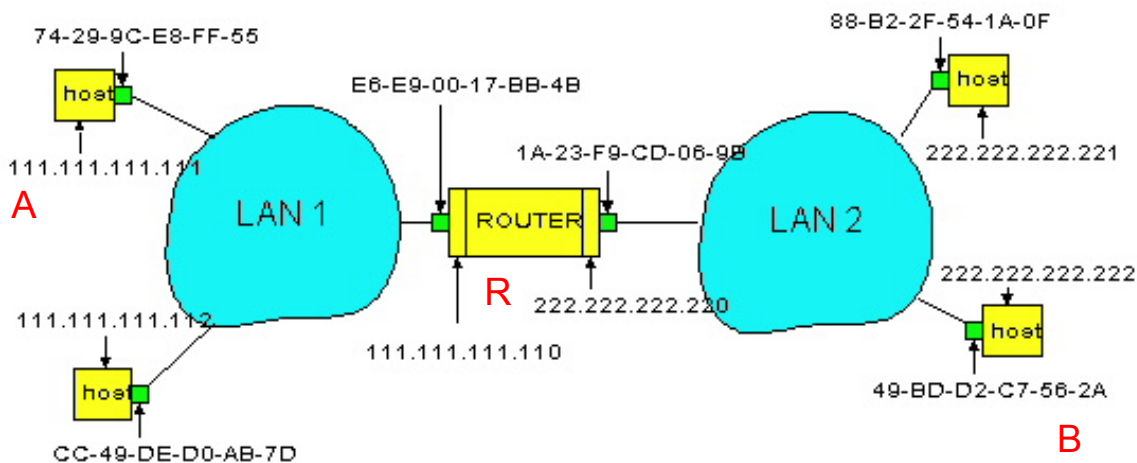
ARP Protocol: Same LAN (Network)

- ❑ A wants to send datagram to B, and B's MAC address not in A's ARP table.
- ❑ A **broadcasts** ARP query packet, containing B's IP address
 - ❑ Dest MAC address = FF-FF-FF-FF-FF-FF
 - ❑ all machines on LAN receive ARP query
- ❑ B receives ARP packet, replies to A with its (B's) MAC address
 - ❑ frame sent to A's MAC address (unicast)
- ❑ A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
 - ❑ Soft state: information that times out (goes away) unless refreshed
- ❑ ARP is "plug-and-play":
 - ❑ Nodes create their ARP tables without intervention from net administrator



Routing to Another LAN (1)

Walkthrough: **Send datagram from A to B via R**
assume A know's B IP address

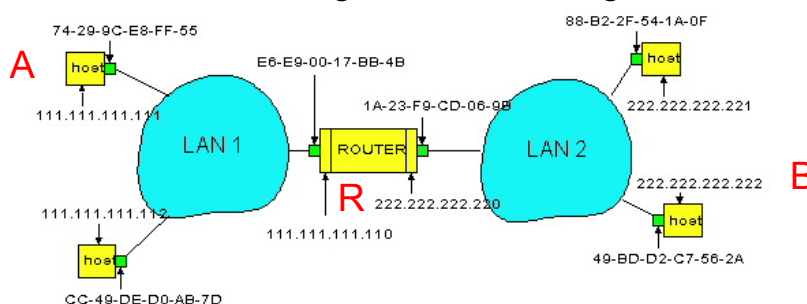


- ❑ Two ARP tables in router R, one for each IP network (LAN)
- ❑ In routing table at source Host, find router 111.111.111.110
- ❑ In ARP table at source, find MAC address E6-E9-00-17-BB-4B, etc



Routing to Another LAN (2)

- ❑ A creates datagram with source A, destination B
- ❑ A uses ARP to get R's MAC address for 111.111.111.110
- ❑ A creates link-layer frame with R's MAC address as dest, frame contains A-to-B IP datagram
- ❑ A's adapter sends frame
- ❑ R's adapter receives frame
- ❑ R removes IP datagram from Ethernet frame, sees its destined to B
- ❑ R uses ARP to get B's MAC address
- ❑ R creates frame containing A-to-B IP datagram sends to B



- ❑ 7.1 Introduction
- ❑ 7.2 Virtual circuit and datagram networks
- ❑ 7.3 What's inside a router
- ❑ **7.4 IP: Internet Protocol**
 - ❑ Datagram format
 - ❑ IPv4 addressing
 - ❑ **ICMP**
 - ❑ IPv6
- ❑ 7.5 Routing algorithms
 - ❑ Overview
 - ❑ Distance Vector
 - ❑ Link state
 - ❑ Hierarchical routing
- ❑ 7.6 Routing in the Internet
 - ❑ RIP
 - ❑ OSPF
 - ❑ BGP



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



- ❑ Source sends series of UDP segments to dest
 - ❑ First has TTL =1
 - ❑ Second has TTL=2, etc.
 - ❑ Unlikely port number
 - ❑ When n^{th} datagram arrives to n^{th} router:
 - ❑ Router discards datagram
 - ❑ And sends to source an ICMP message (type 11, code 0)
 - ❑ Message includes name of router & IP address
 - ❑ Back at source:
 - ❑ When ICMP message arrives, source calculates & displays 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, it stops



- ❑ 7.1 Introduction
- ❑ 7.2 Virtual circuit and datagram networks
- ❑ 7.3 What's inside a router
- ❑ **7.4 IP: Internet Protocol**
 - ❑ Datagram format
 - ❑ IPv4 addressing
 - ❑ ICMP
 - ❑ **IPv6**
- ❑ 7.5 Routing algorithms
 - ❑ Overview
 - ❑ Distance Vector
 - ❑ Link state
 - ❑ Hierarchical routing
- ❑ 7.6 Routing in the Internet
 - ❑ RIP
 - ❑ OSPF
 - ❑ BGP



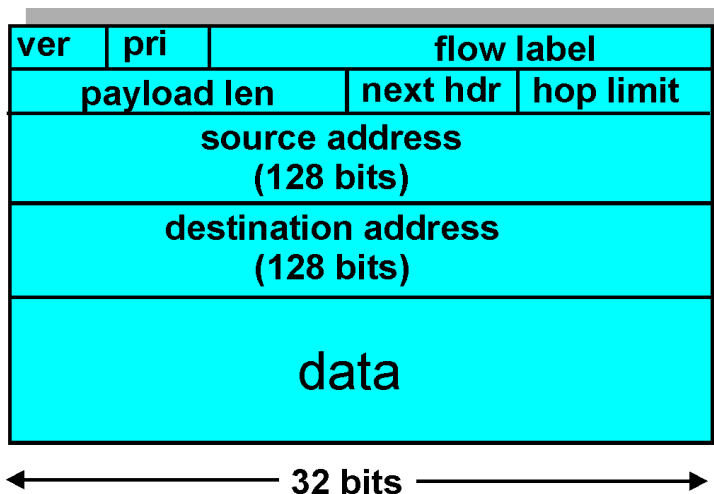
- ❑ **Initial motivation:**
 - ❑ 32-bit address space soon to be completely allocated
- ❑ **Additional motivation:**
 - ❑ Header format helps speed processing/forwarding
 - ❑ Header changes to facilitate QoS
- ❑ **IPv6 datagram format:**
 - ❑ Fixed-length 40 byte header
 - ❑ No fragmentation allowed



Priority: Identify priority among datagrams in flow

Flow Label: Identify datagrams in same “flow.”
(concept of “flow” not well defined).

Next header: Identify upper layer protocol for data



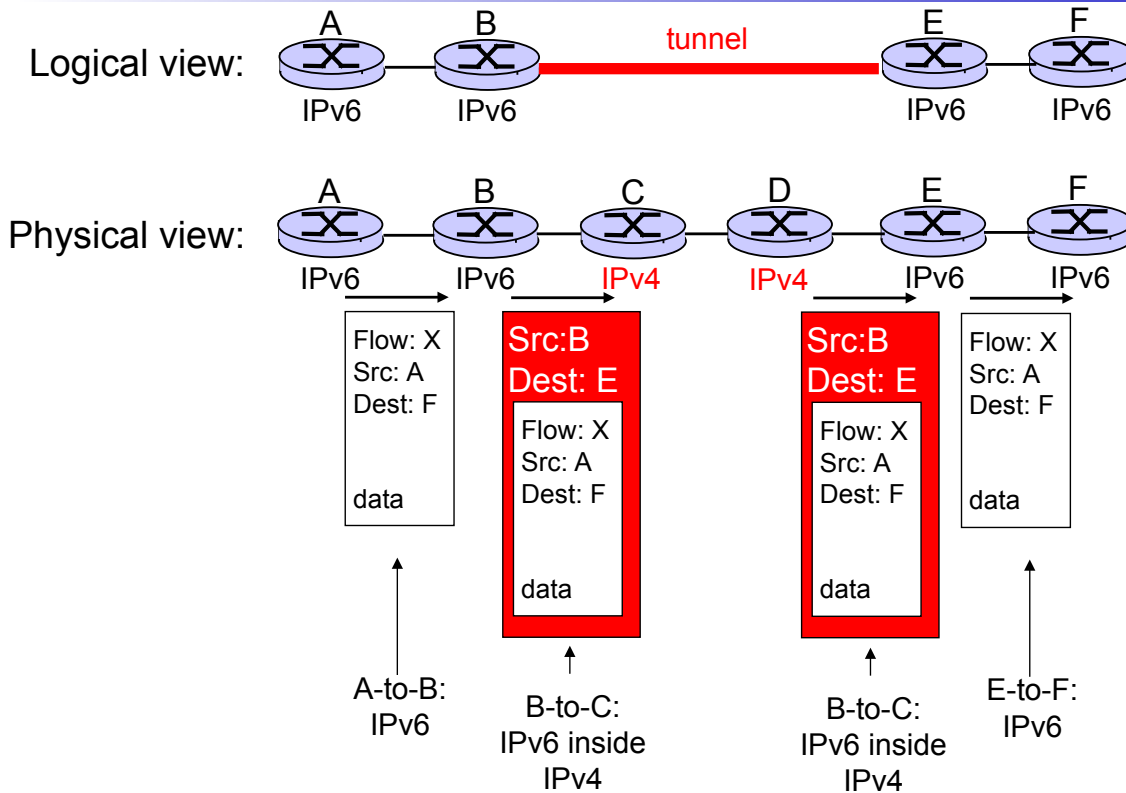
- ❑ **Checksum:**
 - ❑ Removed entirely to reduce processing time at each hop
- ❑ **Options:**
 - ❑ Allowed, but outside of header, indicated by “Next Header” field
- ❑ **ICMPv6:** new version of ICMP
 - ❑ Additional message types, e.g. “Packet Too Big”
 - ❑ Multicast group management functions



- ❑ Not all routers can be upgraded simultaneously
 - ❑ No “flag days” (= day on which all systems commit a change)
 - ❑ How will the network operate with mixed IPv4 and IPv6 routers?
- ❑ **Solution “Tunneling”:**
 - ❑ IPv6 carried as payload in IPv4 datagram among IPv4 routers
 - ❑ Drawbacks:
 - Additional IPv4 header
 - Processing overhead at tunnel endpoints
 - No preferential QoS treatment inside IPv4 tunnel



Tunneling

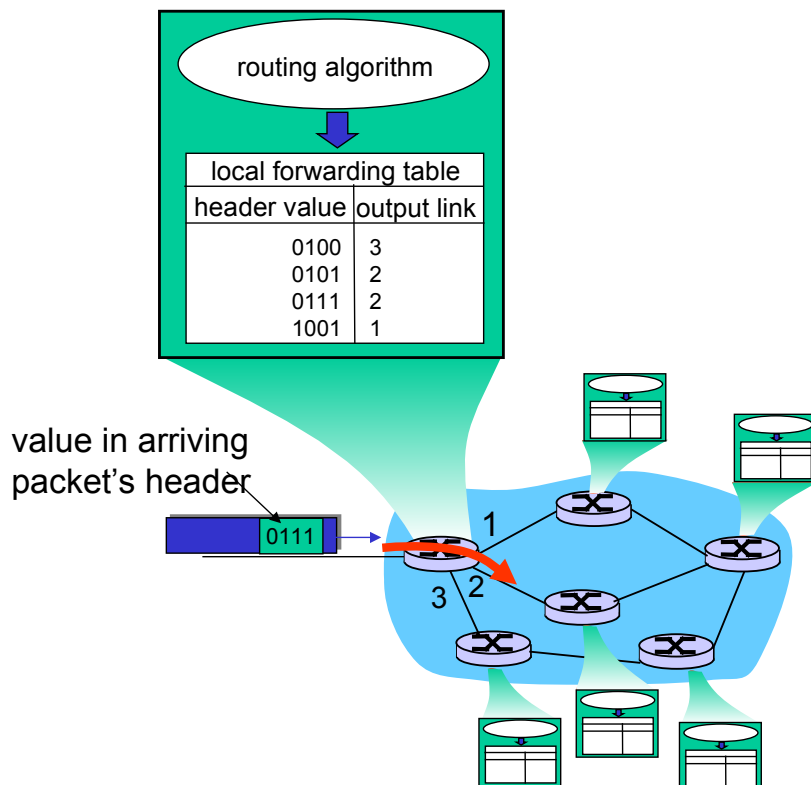


Chapter 7: Network Layer

- ❑ 7.1 Introduction
- ❑ 7.2 Virtual circuit and datagram networks
- ❑ 7.3 What's inside a router
- ❑ 7.4 IP: Internet Protocol
 - ❑ Datagram format
 - ❑ IPv4 addressing
 - ❑ ICMP
 - ❑ IPv6
- ❑ 7.5 Routing algorithms
 - ❑ Overview
 - ❑ Distance Vector
 - ❑ Link state
 - ❑ Hierarchical routing
- ❑ 7.6 Routing in the Internet
 - ❑ RIP
 - ❑ OSPF
 - ❑ BGP



Recall: Interplay Between Routing and Forwarding



Overview on Routing Algorithms (1)

- ❑ An router executes a routing algorithm to decide which output line an incoming packet should be transmitted on:
 - ❑ In connection-oriented service, the routing algorithm is performed only during connection setup
 - ❑ In connectionless service, the routing algorithm is either performed as each packet arrives, or performed periodically and the results of this execution updated in the forwarding table
- ❑ Often, routing algorithms take a so-called *metric* into account when making routing decisions:
 - ❑ In this context, a metric assigns a cost to each network link
 - ❑ This allows to compute a metric for each route in the network
 - ❑ A metric may take into account parameters like number of hops, “€ cost” of a link, delay, length of output queue, etc.
 - ❑ The “cheapest” path according to some metric is often also referred to as the *shortest path* (even though it might actually contain more hops than an alternative path)



- Two basic types of routing algorithms:
 - *Non-adaptive routing algorithms*: do not base their routing decisions on the current state of the network (example: flooding)
 - *Adaptive routing algorithms*: take into account the current network state when making routing decisions (examples: distance vector routing, link state routing)
- Remark: additionally, hierarchical routing can be used to make these algorithms scale to large networks



- Basic strategy:
 - Every incoming packet is sent out on every outgoing line except the one it arrived on
 - Problem: vast number of duplicated packets
- Reducing the number of duplicated packets:
 - Solution 1:
 - Have a hop counter in the packet header; routers decrement each arriving packet's hop counter; routers discard a packet with hop count=0
 - Ideally, the hop counter should be initialized to the length of the path from the source to the destination
 - Solution 2:
 - Require the first router hop to put a sequence number in each packet it receives from its hosts
 - Each router maintains a table listing the sequence numbers it has seen from each first-hop router; the router can then discard packets it has already seen



- ❑ Military applications
 - ❑ Large number of routers is desirable (all systems act as a router)
 - ❑ If one router is taken out (e.g. by a bomb) flooding will still get packets to their destinations
- ❑ Distributed databases
 - ❑ Simultaneous updates of multiple databases can be done with a single packet transmission
- ❑ Networks with frequent topology changes
 - ❑ Adhoc networks



- ❑ Problems with non-adaptive algorithms:
 - ❑ If traffic levels in different parts of the subnet change dramatically and often, non-adaptive routing algorithms are unable to cope with these changes
 - ❑ Lots of computer traffic is *bursty* (~ very variable in intensity), but non-adaptive routing algorithms are usually based on average traffic conditions
- ➔ Adaptive routing algorithms can deal with these situations
- ❑ Three types:
 - ❑ *Centralized adaptive routing*:
 - One central routing controller
 - ❑ *Isolated adaptive routing*:
 - Based on local information
 - Does not require exchange of information between routers
 - ❑ *Distributed adaptive routing*:
 - Routers periodically exchange information and compute updated routing information to be stored in their forwarding table



- ❑ Basic strategy:
 - ❑ Routing table adapts to network traffic
 - ❑ A routing control center is somewhere in the network
 - ❑ Periodically, each router forwards link status information to the control center
 - ❑ The center can compute the best routes, e.g. with Dijkstra's shortest path algorithm (explained later)
 - ❑ Best routes are dispatched to each router
- ❑ Problems:
 - ❑ Vulnerability: if the control center goes down, routing becomes non-adaptive
 - ❑ Scalability: the control center must handle a great deal of routing information, especially for larger networks



- ❑ Basic idea:
 - ❑ Routing decisions are made only on the basis of information available locally in each router
- ❑ Examples:
 - ❑ Hot potato
 - ❑ Backward learning
- ❑ Hot potato routing:
 - ❑ When a packet arrives, the router tries to get rid of it as fast as it can by putting it on the output line that has the shortest queue
 - ❑ Hot potato does not care where the output line leads
 - ❑ Not very effective



- ❑ Basic idea:
 - ❑ Packet headers include destination and source addresses; they also include a hop counter
→ learn from this data as packets pass by
 - ❑ Network nodes, initially ignorant of network topology, acquire knowledge of the network state as packets are handled
- ❑ Algorithm:
 - ❑ Routing is originally random (or hot potato, or flooding)
 - ❑ A packet with a hop count of one is from a directly connected node; thus, neighboring nodes are identified with their connecting links
 - ❑ A packet with a hop count of two is from a source two hops away, etc.
 - ❑ As packets arrive, the IMP compares the hop count for a given source address with the minimum hop count already registered; if the new one is less, it is substituted for the previous one
 - ❑ Remark: in order to be able to adapt to deterioration of routes (e.g. link failures) the acquired information has to be “forgotten” periodically

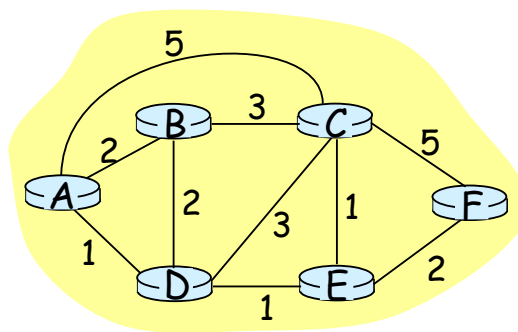


Routing Protocol

Goal: Determine “good” path (sequence of routers) through network from source to dest.

Graph abstraction for routing algorithms:

- ❑ Graph nodes are routers
- ❑ Graph edges are physical links
 - ❑ Link cost: delay, \$ cost, or congestion level
 - ❑ Path cost: sum of the link costs on the path
- ❑ “Good” path:
 - ❑ Typically means minimum cost path
 - ❑ Other definitions possible



Global or decentralized information?

Decentralized:

- ❑ Router knows physically-connected neighbors, link costs to neighbors
- ❑ Iterative process of computation, exchange of info with neighbors
- ❑ “Distance vector” algorithms
 - RIP protocol
 - BGP protocol (“path vector”)

Global:

- ❑ All routers have complete topology, link cost info
- ❑ “Link state” algorithms
 - Dijkstra’s algorithm
 - OSPF protocol

Static or dynamic?

Static:

- ❑ Routes change slowly over time

Dynamic:

- ❑ Routes change more quickly
 - ❑ Periodic update
 - ❑ In response to link cost changes



- ❑ 7.1 Introduction
- ❑ 7.2 Virtual circuit and datagram networks
- ❑ 7.3 What’s inside a router
- ❑ 7.4 IP: Internet Protocol
 - ❑ Datagram format
 - ❑ IPv4 addressing
 - ❑ ICMP
 - ❑ IPv6
- ❑ 7.5 Routing algorithms
 - ❑ Overview
 - ❑ Distance Vector
 - ❑ Link state
 - ❑ Hierarchical routing
- ❑ 7.6 Routing in the Internet
 - ❑ RIP
 - ❑ OSPF
 - ❑ BGP



Iterative:

- Continues until no nodes exchange info.
- *Self-terminating*: no "signal" to stop

Asynchronous:

- Nodes need *not* exchange info/iterate in lock step!

Distributed:

- Each node communicates *only* with directly-attached neighbors

Distance Table data structure

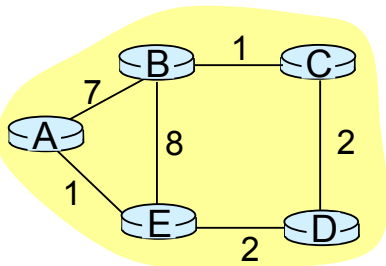
- Each node has its own
 - Row for each possible destination
 - Column for each directly-attached neighbor to node
- Example: in node X, for dest. Y via neighbor Z:

$$D^X(Y,Z) = \text{distance from X to Y, via Z as next hop}$$

$$= c(X,Z) + \min_w \{D^Z(Y,w)\}$$



Example for Distance Table



$$D^E(C,D) = c(E,D) + \min_w \{D^D(C,w)\}$$

$$= 2+2 = 4$$

$$D^E(A,D) = c(E,D) + \min_w \{D^D(A,w)\}$$

$$= 2+3 = 5 \text{ loop!}$$

$$D^E(A,B) = c(E,B) + \min_w \{D^B(A,w)\}$$

$$= 8+6 = 14 \text{ loop!}$$

		cost to destination via		
$D^E()$		A	B	D
destination	A	1	14	5
	B	7	8	5
	C	6	9	4
	D	4	11	2



Constructing Routing Table from Distance Table

		Cost to destination via				
		A	B	D		
Destination	D ^E ()				Outgoing link to use, cost	
	A	1	14	5	A	A,1
	B	7	8	5	B	D,5
	C	6	9	4	C	D,4
D	4	11	2	D	D,4	

Distance table → Routing table



Distance Vector Routing: Overview

Iterative, asynchronous:

Each local iteration caused by:

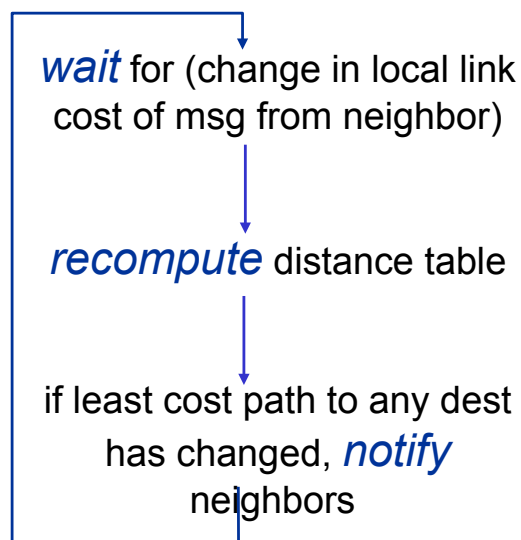
- ❑ Local link cost change
- ❑ Message from neighbor: its least cost path change from neighbor

Distributed:

Each node notifies neighbors *only* when its least cost path to any destination changes

- ❑ Neighbors then notify their neighbors if necessary

Each node:



At all nodes, X:

- 1 Initialization:
- 2 for all adjacent nodes v:
- 3 $D^X(*,v) = \text{infinity}$ /* the * operator means "for all rows" */
- 4 $D^X(v,v) = c(X,v)$
- 5 for all destinations, y
- 6 send $\min_w D^X(y,w)$ to each neighbor /* w over all X's neighbors */

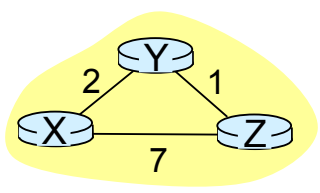


```

8 loop
9  wait (until I see a link cost change to neighbor V
10     or until I receive update from neighbor V)
11
12  if (c(X,V) changes by d)
13     /* change cost to all dest's via neighbor v by d */
14     /* note: d could be positive or negative */
15     for all destinations y:  $D^X(y,V) = D^X(y,V) + d$ 
16
17  else if (update received from V w.r.t. destination Y)
18     /* shortest path from V to some Y has changed */
19     /* V has sent a new value for its  $\min_w D^V(Y,w)$  */
20     /* call this received new value "newval" */
21     for the single destination y:  $D^X(y,V) = c(X,V) + \text{newval}$ 
22
23  if we have a new  $\min_w D^X(Y,w)$  for any destination Y
24     send new value of  $\min_w D^X(Y,w)$  to all neighbors
25
26 forever
    
```



Distance Vector Algorithm: Example (1)



		cost via	
		Y	Z
D ^X	Y	2	∞
	Z	∞	7
	d _{est}		

		cost via	
		X	Z
D ^Y	X	2	∞
	Z	∞	1
	d _{est}		

		cost via	
		X	Y
D ^Z	X	7	∞
	Y	∞	1
	d _{est}		

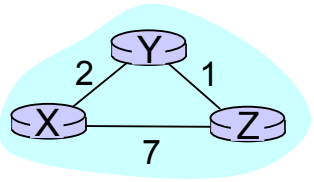
		cost via	
		Y	Z
D ^X	Y	2	8
	Z	3	7
	d _{est}		

		cost via	
		X	Z
D ^Y	X	2	8
	Z	9	1
	d _{est}		

		cost via	
		X	Y
D ^Z	X	7	3
	Y	9	1
	d _{est}		



Distance Vector Algorithm: Example (2)



		cost via	
		Y	Z
D ^X	Y	2	∞
	Z	∞	7
	d _{est}		

		cost via	
		X	Z
D ^Y	X	2	∞
	Z	∞	1
	d _{est}		

		cost via	
		X	Y
D ^Z	X	7	∞
	Y	∞	1
	d _{est}		

		cost via	
		Y	Z
D ^X	Y	2	8
	Z	3	7
	d _{est}		

$$D^X(Y,Z) = c(X,Z) + \min_w \{D^Z(Y,w)\}$$

$$= 7 + 1 = 8$$

$$D^X(Z,Y) = c(X,Y) + \min_w \{D^Y(Z,w)\}$$

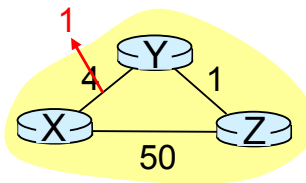
$$= 2 + 1 = 3$$



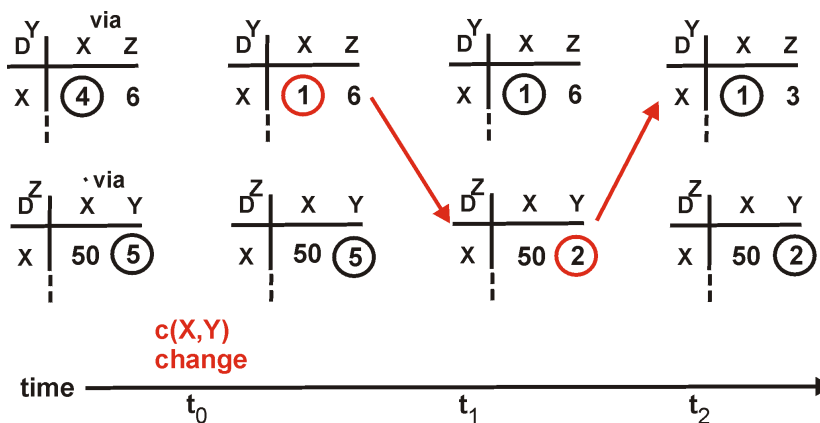
Distance Vector: Reaction to Link Cost Changes (1)

Link cost changes:

- Node detects local link cost change
- Updates distance table (line 15)
- If cost change in least cost path, notify neighbors (lines 23,24)



“Good news travels fast”



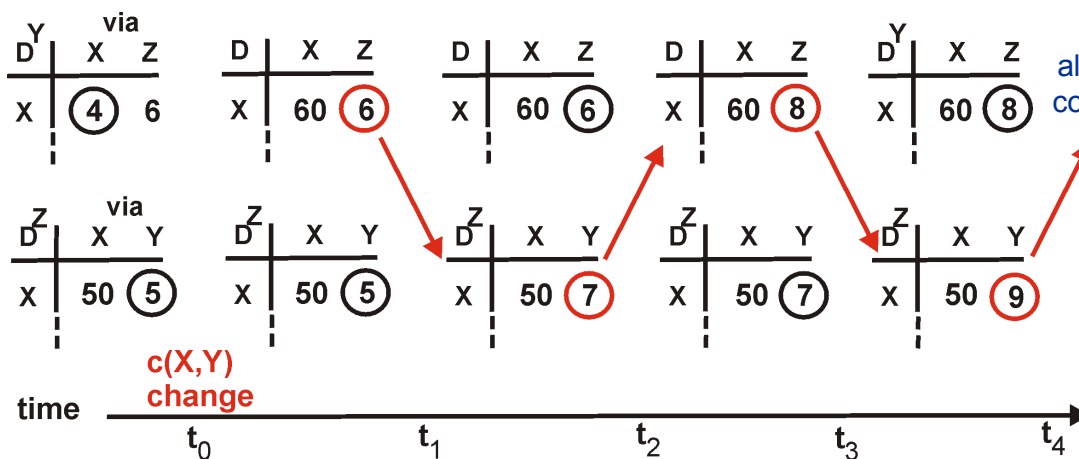
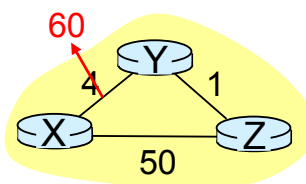
Algorithm “terminates”



Distance Vector: Reaction to Link Cost Changes (2)

Link cost changes:

- Good news travels fast
- Bad news travels slow - “count to infinity” problem!

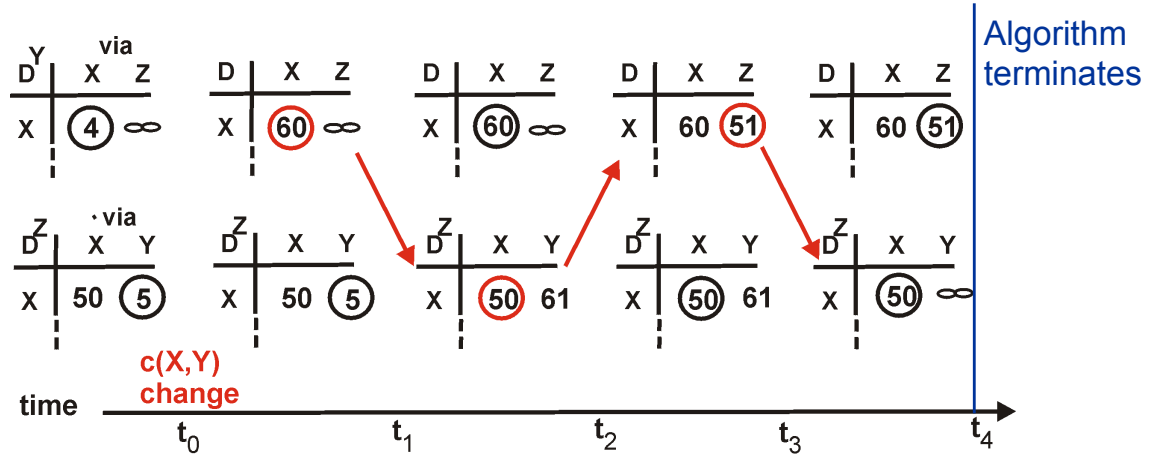
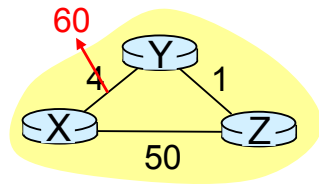


algorithm continues on!



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?



- ❑ 7.1 Introduction
- ❑ 7.2 Virtual circuit and datagram networks
- ❑ 7.3 What's inside a router
- ❑ 7.4 IP: Internet Protocol
 - ❑ Datagram format
 - ❑ IPv4 addressing
 - ❑ ICMP
 - ❑ IPv6
- ❑ 7.5 Routing algorithms
 - ❑ Overview
 - ❑ Distance Vector
 - ❑ Link state
 - ❑ Hierarchical routing
- ❑ 7.6 Routing in the Internet
 - ❑ RIP
 - ❑ OSPF
 - ❑ BGP



- Link state routing usually uses *Dijkstra's algorithm*:
 - Network topology, link costs known to all nodes:
 - Accomplished via “link state broadcast”
 - All nodes have same information
 - Input: a graph (V, E) with
 - V being the set of vertices (nodes)
 - E being the set of edges (links)
 - A mapping $c(v, w)$ representing the cost of edge (v, w) if (v, w) is in E (otherwise $c(v, w) = \text{infinity}$)
 - Goal: compute the least cost paths from one node s (“source”) to all other nodes v :
 - Can be used to obtain *routing table* for that node s



- Basic algorithm idea:
 - Start with a set N containing only the source node s and incrementally add one node at a time, to which the shortest path can be determined with the knowledge available at that point in time
 - Initially, the shortest path to node v is estimated to be infinity for all vertices except the source vertex s
 - In each step:
 - Select the node v not yet in the set N that can be reached from s with the cheapest estimated cost using only nodes, that are already in N
 - Include v in N
 - Update the estimates for all direct neighbors w of v if a path over v would be cheaper than the current estimate
 - When updating an estimate for node w , also memorize the predecessor node v leading to this estimate



Dijkstra's Algorithm to Compute Shortest Paths (2)

- Thus, the algorithm maintains for every vertex v :
 - A shortest path “estimate” $d(v)$
 - This “estimate” $d(v)$ is what the algorithm currently assumes to be the shortest path from s to v
 - The predecessor node $p(v)$ on the currently assumed shortest path from s to v
 - A list of vertices N for whom the shortest path is definitely known
- Some remarks:
 - It holds that whenever the estimate $d(v)$ is finite for a vertex v , there exists a path from the source s to v with weight $d(v)$
 - It turns out that the estimate is accurate for the vertex with the minimum value of this estimate
 - Shortest path is known for this vertex v
 - This vertex v is added to the set N of vertices for which shortest path is known
 - Shortest path estimates are upgraded for every vertex which has an edge from v , and is not in N



Dijkstra's Algorithm to Compute Shortest Paths (3)

- Estimate upgrade procedure:
 - Suppose vertex v is added to the list newly, and we are upgrading the estimate for vertex w :
 - $d(v)$ is the shortest path estimate for v , $d(w)$ is the estimate for w
 - $c(v, w)$ is the cost of the edge from v to w

```

if (  $d(v) + c(v, w) < d(w)$  ) {
     $d(w) = d(v) + c(v, w)$ ;
     $p(w) = v$ ; }
                    
```
- Intuition behind this procedure:
 - Assume that $d(w)$ and $d(v)$ are finite
 - So, there exists a path (s, v_1, v_2, \dots, v) from s to v of weight $d(v)$
 - Hence, there exists a path $(s, v_1, v_2, \dots, v, w)$ from s to w of weight $d(v) + c(v, w)$
 - Also, there exists another path to w of weight $d(w)$
 - So the shortest path to w can not have weight more than either $d(w)$ or $d(v) + c(v, w)$



Dijkstra's Algorithm in Pseudocode

Computes least-cost path from one node to other nodes in the net

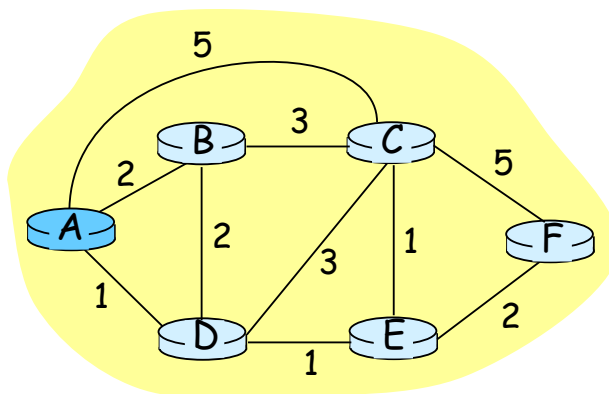
```

1 Initialization (for Node s):
2 N = {s} /* Set of Nodes with known least-cost path */
3 for all nodes v
4   if v adjacent to s
5     then { d(v) = c(s,v); p(v) = s; }
6     else d(v) = infinity;
7
8 Loop
9   find v not in N such that d(v) is a minimum
10  add v to N
11  for all w adjacent to v and not in N do /* update d(w) */
12    { if (d(v) + c(v, w) < d(w))
13      { d(w) = d(v) + c(v, w); p(w) = v; } }
14  /* new cost to w is either old cost to w or known
15  shortest path cost to v plus cost from v to w */
16 until all nodes in N
  
```



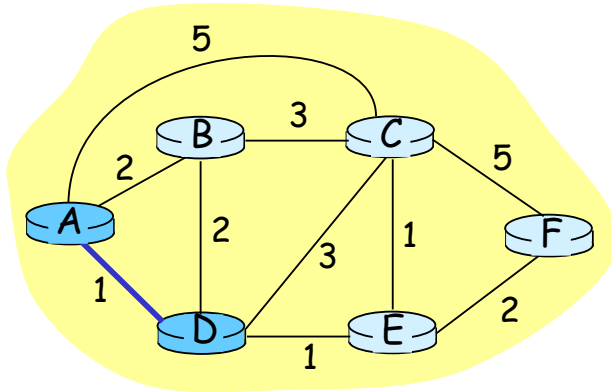
Example Run of Dijkstra's Algorithm (1)

Step	start N	d(B),p(B)	d(C),p(C)	d(D),p(D)	d(E),p(E)	d(F),p(F)
→ 0	A	2,A	5,A	1,A	infinity	infinity



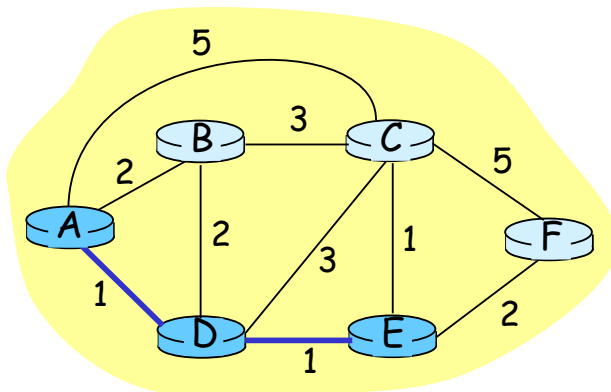
Example Run of Dijkstra's Algorithm (2)

Step	start N	d(B),p(B)	d(C),p(C)	d(D),p(D)	d(E),p(E)	d(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
→ 1	AD	2,A	4,D		2,D	infinity



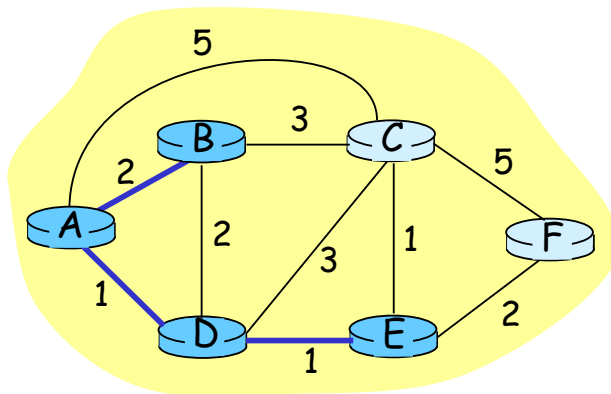
Example Run of Dijkstra's Algorithm (3)

Step	start N	d(B),p(B)	d(C),p(C)	d(D),p(D)	d(E),p(E)	d(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
→ 2	ADE	2,A	3,E			4,E
3						
4						
5						



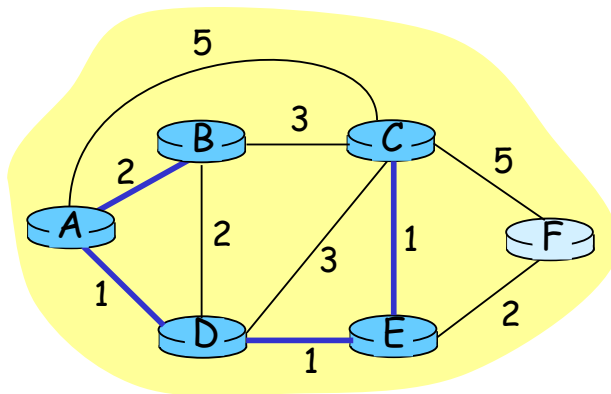
Example Run of Dijkstra's Algorithm (4)

Step	start N	d(B),p(B)	d(C),p(C)	d(D),p(D)	d(E),p(E)	d(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
→ 3	ADEB		3,E			4,E



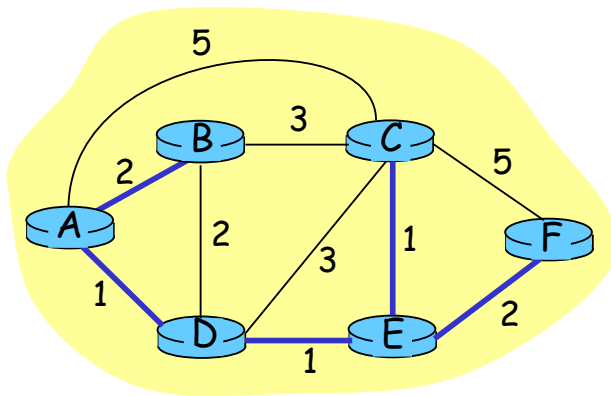
Example Run of Dijkstra's Algorithm (5)

Step	start N	d(B),p(B)	d(C),p(C)	d(D),p(D)	d(E),p(E)	d(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
→ 4	ADEBC					4,E



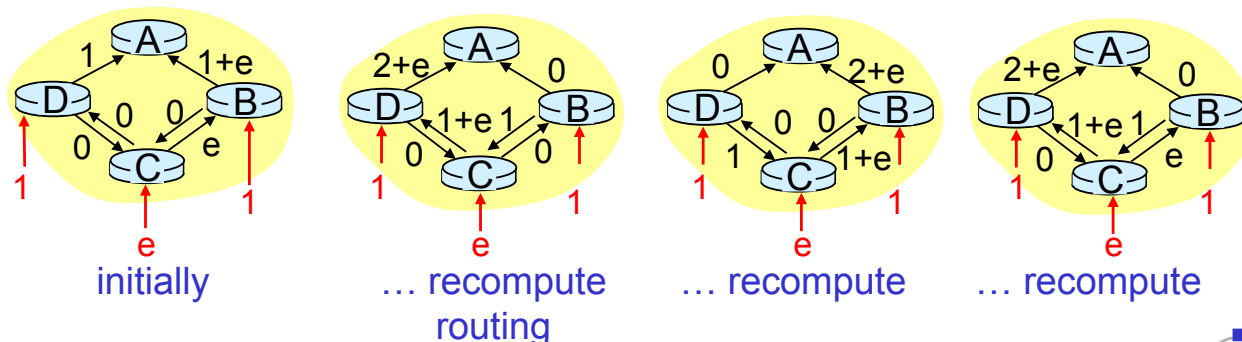
Example Run of Dijkstra's Algorithm (6)

Step	start N	d(B),p(B)	d(C),p(C)	d(D),p(D)	d(E),p(E)	d(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
→ 5	ADEBCF					



Further Discussion of Dijkstra's Algorithm

- Algorithm complexity for $|V|$ nodes:
 - Each iteration: need to check all nodes v that are not yet in N
 - This requires $|V| \cdot (|V|+1)/2$ comparisons, leading to $O(|V|^2)$
 - This is optimal in dense graphs (where $|E| \sim |V|^2$)
 - In sparse graphs, more efficient implementations are possible: $O(|V| \cdot \log|V| + |E|)$ using so-called Fibonacci-heaps
- Variable link costs \rightarrow oscillations possible:
 - E.g., link cost = amount of carried traffic



- ❑ Each router measures the cost (in delay, hop count, etc.) between itself and its adjacent routers
- ❑ The router builds a packet containing all these distances
 - ❑ The packet also contains a sequence number and an age field
- ❑ Each router distributes these packets using flooding
- ❑ To control flooding, the sequence numbers are used by routers to discard flood packets they have already seen from a given router
- ❑ The age field in the packet is an expiration date that specifies how long the information in the packet is valid
- ❑ Once a router receives all the link state packets from the network, it can reconstruct the complete topology and compute a shortest path between itself and any other node using Dijkstra's algorithm



Message complexity

- ❑ LS: with n nodes, E links, $O(n \cdot E)$ msgs sent each
- ❑ DV: exchange between neighbors only
 - ❑ convergence time varies

Speed of Convergence

- ❑ LS: $O(n^2)$ algorithm requires $O(n \cdot E)$ msgs
 - ❑ may have oscillations (with dynamic link weights)
- ❑ DV: convergence time varies
 - ❑ may be routing loops
 - ❑ count-to-infinity problem
 - ❑ may have oscillations (with dynamic link weights)

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 other routers:
 - Errors propagate through network



- ❑ 7.1 Introduction
- ❑ 7.2 Virtual circuit and datagram networks
- ❑ 7.3 What's inside a router
- ❑ 7.4 IP: Internet Protocol
 - ❑ Datagram format
 - ❑ IPv4 addressing
 - ❑ ICMP
 - ❑ IPv6
- ❑ 7.5 **Routing algorithms**
 - ❑ Overview
 - ❑ Distance Vector
 - ❑ Link state
 - ❑ **Hierarchical routing**
- ❑ 7.6 Routing in the Internet
 - ❑ RIP
 - ❑ OSPF
 - ❑ BGP



Our routing study so far is an idealization:

- ❑ All routers are assumed to be identical
- ❑ Network is assumed to be “flat”

... Practice, however, looks different

Scale (50 million destinations!):

- ❑ Can't store all destinations in routing tables
- ❑ Routing table exchange would overload links

Administrative autonomy:

- ❑ Internet = network of networks
- ❑ Each network admin may want to control routing in its own network



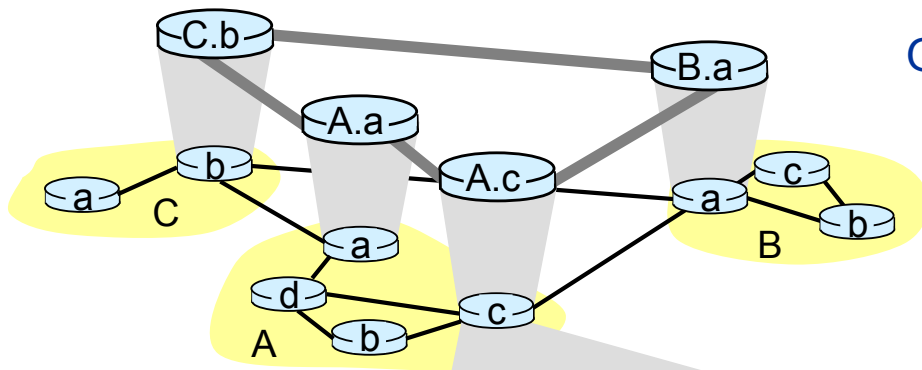
- ❑ Aggregate routers into regions, “Autonomous systems” (AS)
- ❑ Routers in same AS run same routing protocol
 - ❑ “Intra-AS” routing protocol
 - ❑ Routers in different AS can run different intra-AS routing protocol

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

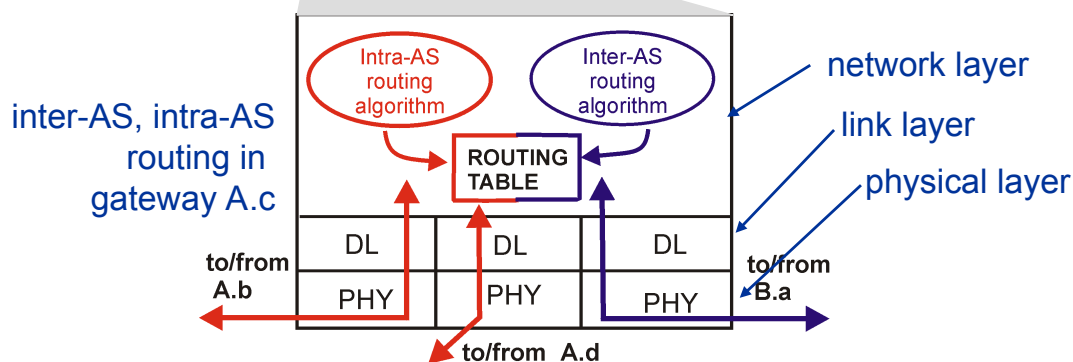


Inter-AS and Intra-AS Routing



Gateways:

- ❑ Perform inter-AS routing amongst themselves
- ❑ Perform intra-AS routing with other routers in their AS

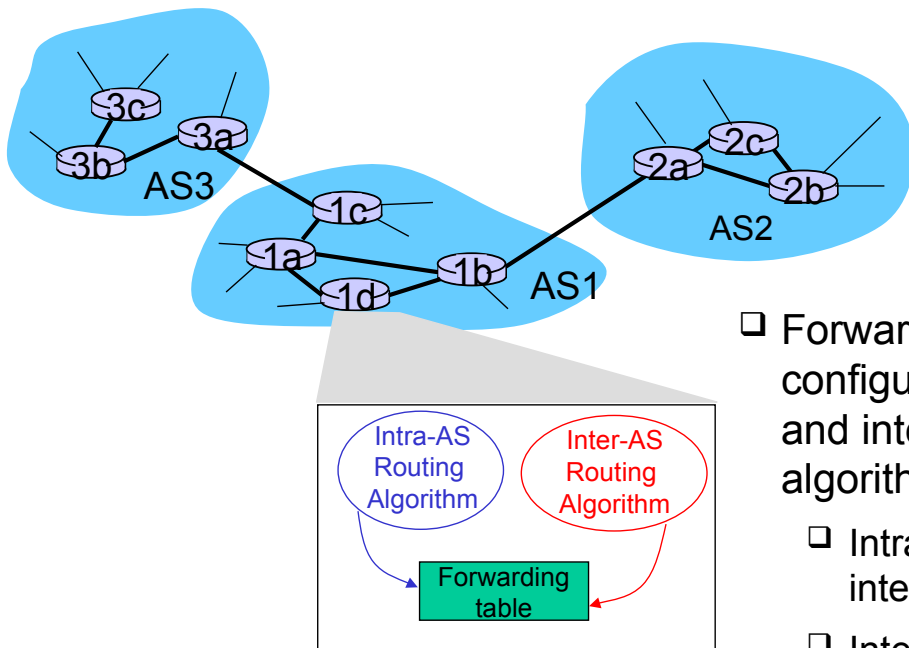


- ❑ The Global Internet consists of **Autonomous Systems (AS)** interconnected with each other:
 - ❑ Stub AS: small corporation (only one link to the Internet)
 - ❑ Multihomed AS: large corporation (multiple links, but no transit)
 - ❑ Transit AS: provider
- ❑ Two-level routing:
 - ❑ **Intra-AS**: administrator is responsible for choice
 - **Routing Information Protocol (RIP)**: Distance Vector
 - **Open Shortest Path First (OSPF)**: Link State
 - **Interior Gateway Routing Protocol (IGRP)**: Distance Vector (Cisco proprietary)
 - ❑ **Inter-AS**: unique standard
 - **Border Gateway Protocol (BGP)**: Path Vector (sort of distance vector, but with path information for loop avoidance)



- ❑ **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





- Forwarding table is configured by both intra- and inter-AS routing algorithm
 - Intra-AS sets entries for internal destinations
 - Inter-AS & Intra-AS sets entries for external destinations



Inter-AS Tasks

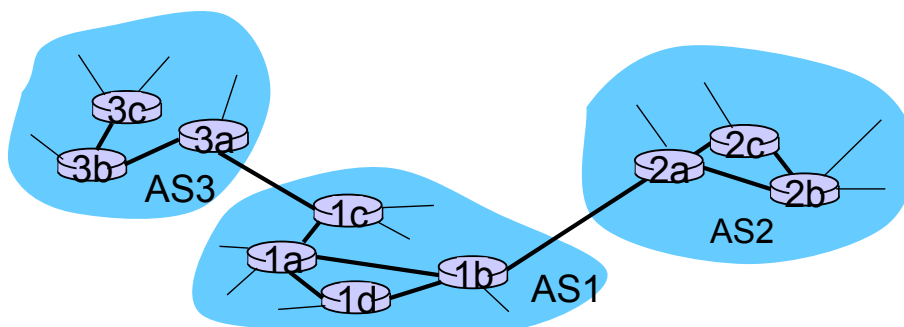
Problem:

- Suppose router in AS1 receives datagram for which destination is outside of AS1
- Router should forward packet towards one of the gateway routers, but which one?

AS1 needs to:

- Learn which destinations are reachable through AS2 and which through AS3
- Propagate this reachability info to all routers in AS1

Job of inter-AS routing!



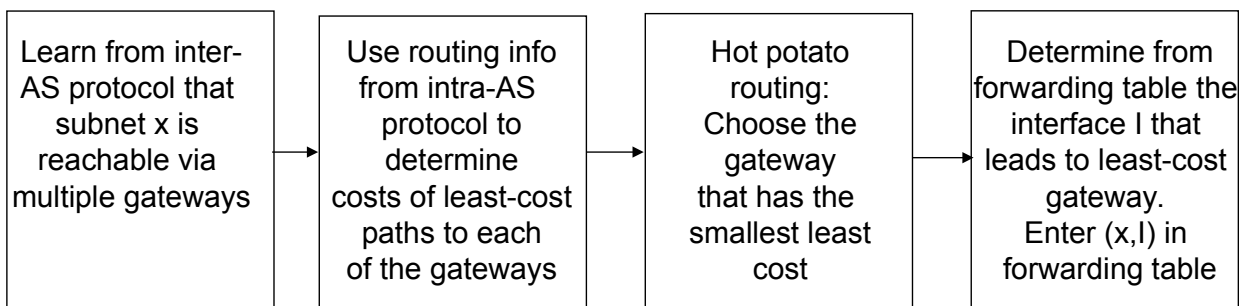
Example: Setting Forwarding Table in Router 1d

- ❑ Suppose AS1 learns from the inter-AS protocol that subnet x is reachable from AS3 (gateway 1c) but not from 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.
- ❑ Puts in forwarding table entry (x, I) .



Example: Choosing Among Multiple ASes

- ❑ Now suppose AS1 learns from the 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 the job on inter-AS routing protocol!
- ❑ **Hot potato routing:** send packet towards closest of two routers (the AS tries to get rid of the packet as quickly as possible)



- ❑ 7.1 Introduction
- ❑ 7.2 Virtual circuit and datagram networks
- ❑ 7.3 What's inside a router
- ❑ 7.4 IP: Internet Protocol
 - ❑ Datagram format
 - ❑ IPv4 addressing
 - ❑ ICMP
 - ❑ IPv6
- ❑ 7.5 Routing algorithms
 - ❑ Overview
 - ❑ Distance Vector
 - ❑ Link state
 - ❑ Hierarchical routing
- ❑ 7.6 Routing in the Internet
 - ❑ RIP
 - ❑ OSPF
 - ❑ BGP

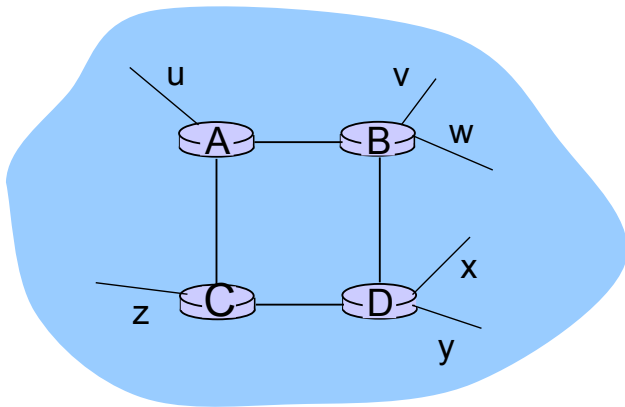


- ❑ Also known as **Interior Gateway Protocols (IGP)**
- ❑ Most common Intra-AS routing protocols:
 - ❑ RIP: Routing Information Protocol
 - ❑ OSPF: Open Shortest Path First
 - ❑ IGRP: Interior Gateway Routing Protocol (Cisco proprietary)



Basic characteristics:

- ❑ Distance vector algorithm
- ❑ Included in BSD-UNIX Distribution in 1982
- ❑ Distance metric: # of hops (max = 15 hops)



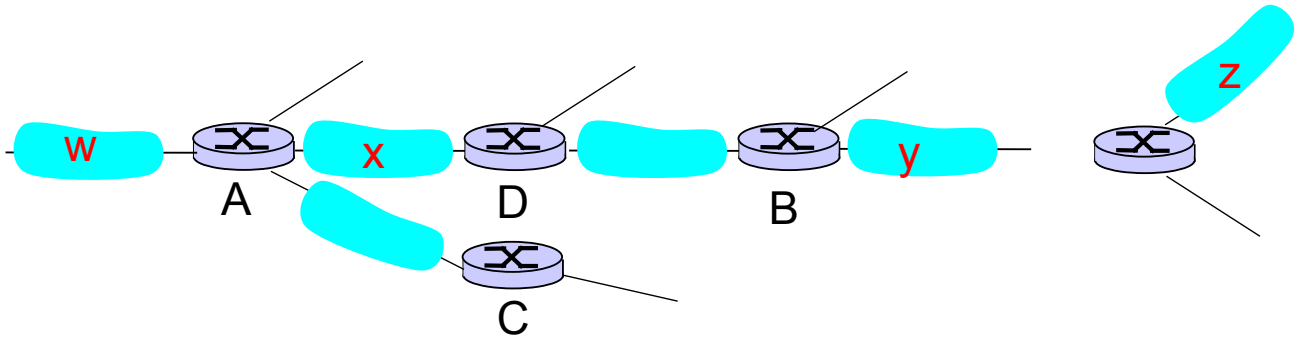
Destination	Hops
u	1
v	2
w	2
x	3
y	3
z	2



- ❑ Distance vectors:
 - ❑ Exchanged among neighbors every 30 sec via Response Message
 - ❑ Also called **advertisement**
- ❑ Each advertisement:
 - ❑ List of up to 25 destination networks within AS



RIP: Example



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

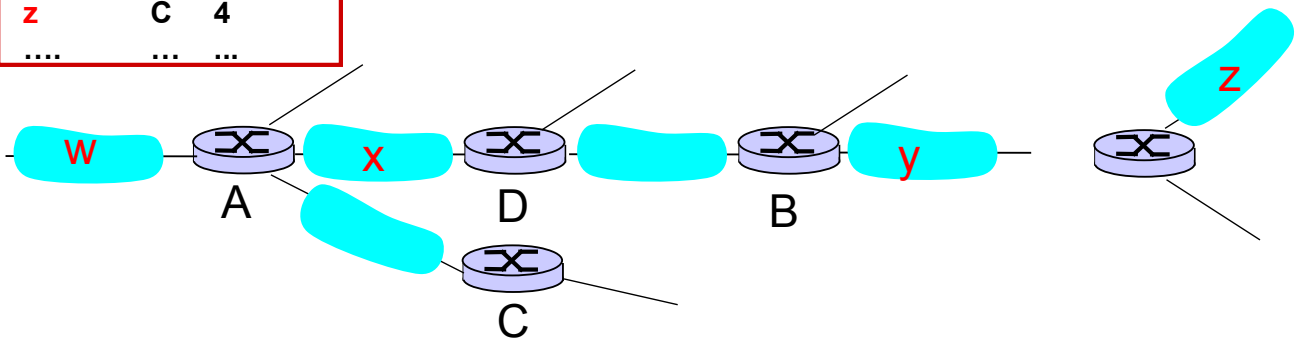
Routing table in D



RIP: Example

Dest	Next hops
w	- -
x	- -
z	C 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
....



If no advertisement heard after 180 sec

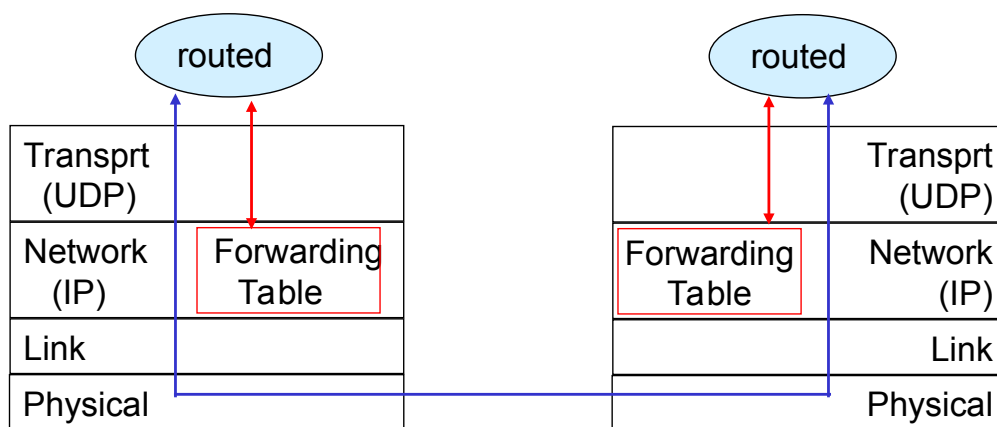
⇒ neighbor/link declared dead:

- ❑ 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



- ❑ 7.1 Introduction
- ❑ 7.2 Virtual circuit and datagram networks
- ❑ 7.3 What's inside a router
- ❑ 7.4 IP: Internet Protocol
 - ❑ Datagram format
 - ❑ IPv4 addressing
 - ❑ ICMP
 - ❑ IPv6
- ❑ 7.5 Routing algorithms
 - ❑ Overview
 - ❑ Distance Vector
 - ❑ Link state
 - ❑ Hierarchical routing
- ❑ 7.6 Routing in the Internet
 - ❑ RIP
 - ❑ OSPF
 - ❑ BGP



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)

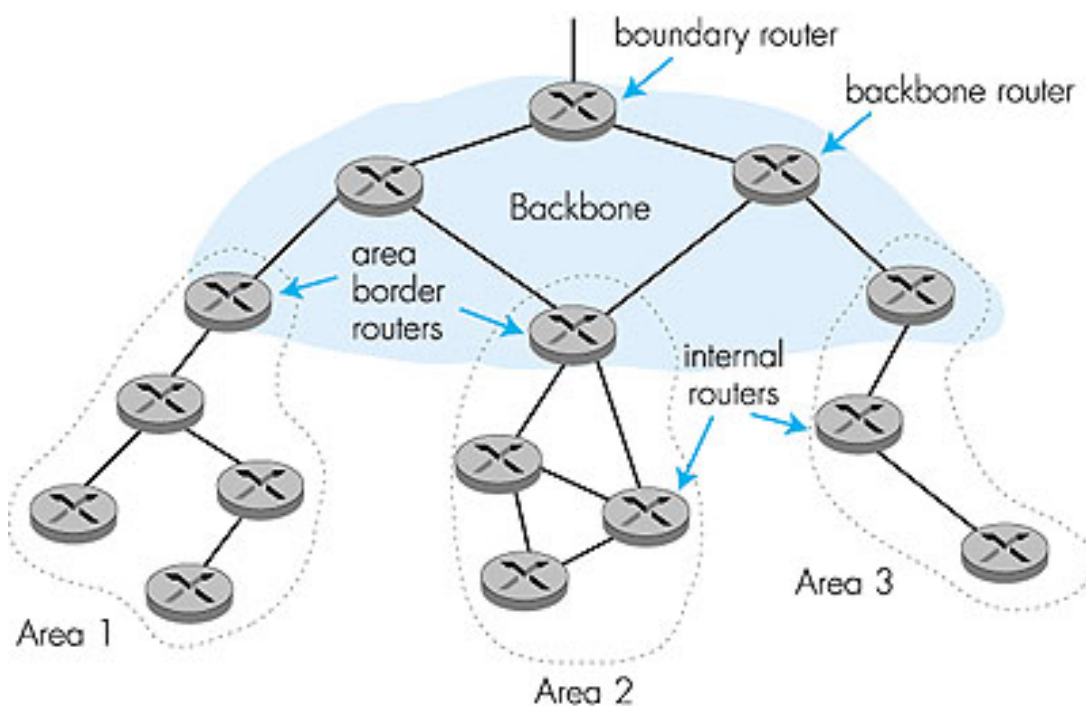


OSPF "Advanced" Features (not in RIP)

- ❑ **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 (1)



- ❑ **Two-level hierarchy:** local area, backbone.
 - ❑ Link-state advertisements only in area
 - ❑ Each node has detailed area topology; only knows direction (shortest path) to networks 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.



- ❑ 7.1 Introduction
- ❑ 7.2 Virtual circuit and datagram networks
- ❑ 7.3 What's inside a router
- ❑ 7.4 IP: Internet Protocol
 - ❑ Datagram format
 - ❑ IPv4 addressing
 - ❑ ICMP
 - ❑ IPv6
- ❑ 7.5 Routing algorithms
 - ❑ Overview
 - ❑ Distance Vector
 - ❑ Link state
 - ❑ Hierarchical routing
- ❑ **7.6 Routing in the Internet**
 - ❑ RIP
 - ❑ OSPF
 - ❑ **BGP**

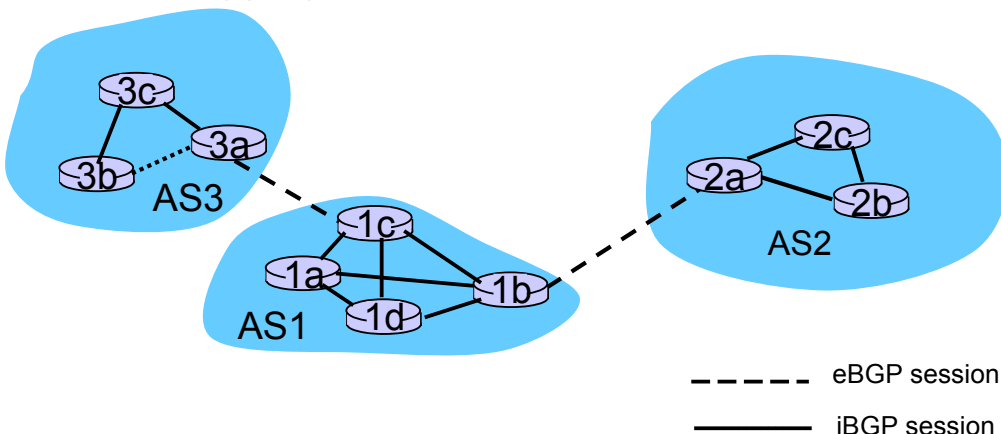


- ❑ **BGP (Border Gateway Protocol):** *the de facto standard*
- ❑ BGP provides each AS a means to:
 - ❑ Obtain subnet reachability information from neighboring ASs
 - ❑ Propagate the reachability information to all routers internal to the AS
 - ❑ Determine “good” routes to subnets based on reachability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



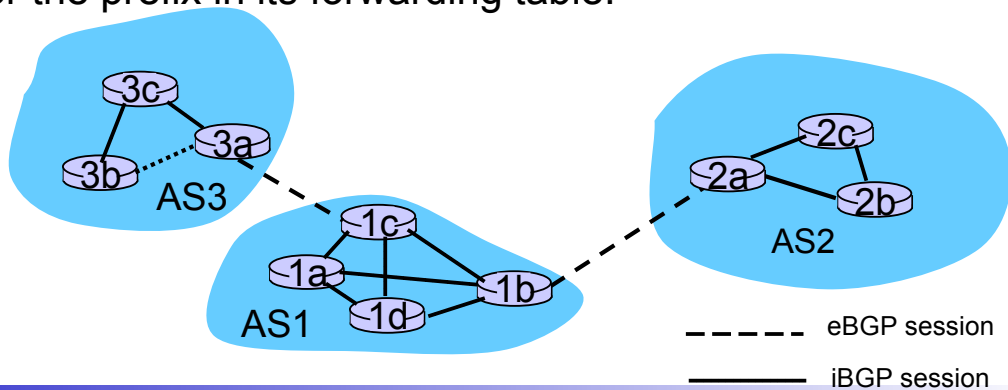
BGP Basics

- ❑ Pairs of routers (BGP peers) exchange routing info over semi-permanent TCP connections: **BGP sessions**
- ❑ Note that BGP sessions do not correspond to physical links.
- ❑ When AS2 advertises a prefix to AS1, AS2 is **promising** it will forward any datagrams destined to that prefix towards the prefix
 - ❑ AS2 can aggregate prefixes in its advertisement



Distributing Reachability Info

- ❑ With eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
- ❑ 1c can then use iBGP to distribute this new prefix reach info to all routers in AS1
- ❑ 1b can then re-advertise the new reach info to AS2 over the 1b-to-2a eBGP session
- ❑ When router learns about a new prefix, it creates an entry for the prefix in its forwarding table.



Path Attributes & BGP Routes

- ❑ When advertising a prefix, the advertisement includes BGP attributes.
 - ❑ Prefix + attributes = “route”
- ❑ Two important attributes:
 - ❑ **AS-PATH**: contains the ASs through which the advert for the prefix passed: AS 67 AS 17
 - ❑ **NEXT-HOP**: Indicates the specific internal-AS router to next-hop AS. (There may be multiple links from current AS to next-hop-AS.)
- ❑ When gateway router receives route advert, uses **import policy** to accept/decline.



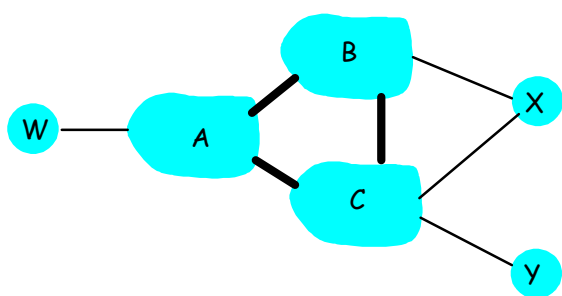
- ❑ Router may learn about more than 1 route to some prefix:
 - ❑ Router must select route
- ❑ Elimination rules:
 - ❑ Local preference value attribute: policy decision
 - ❑ Shortest AS-PATH
 - ❑ Closest NEXT-HOP router: hot potato routing
 - ❑ Additional criteria



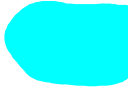

- ❑ BGP messages exchanged using TCP
- ❑ BGP messages:
 - ❑ **OPEN**: opens TCP connection to peer and authenticates sender
 - ❑ **UPDATE**: advertises new path (or withdraws old)
 - ❑ **KEEPALIVE** keeps connection alive in absence of UPDATES; also ACKs OPEN request
 - ❑ **NOTIFICATION**: reports errors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



BGP Routing Policy (1)



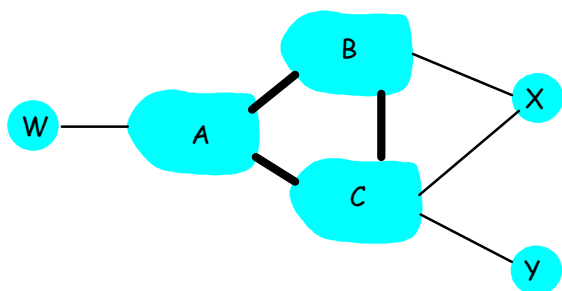
legend:

-  provider network
-  customer network:

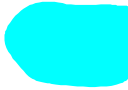

- ❑ A,B,C are **provider networks**
- ❑ X,W,Y are customer (of provider networks)
- ❑ X is **dual-homed**: attached to two networks
 - ❑ X does not want to route from B via X to C
 - ❑ .. so X will not advertise to B a route to C



BGP Routing Policy (2)

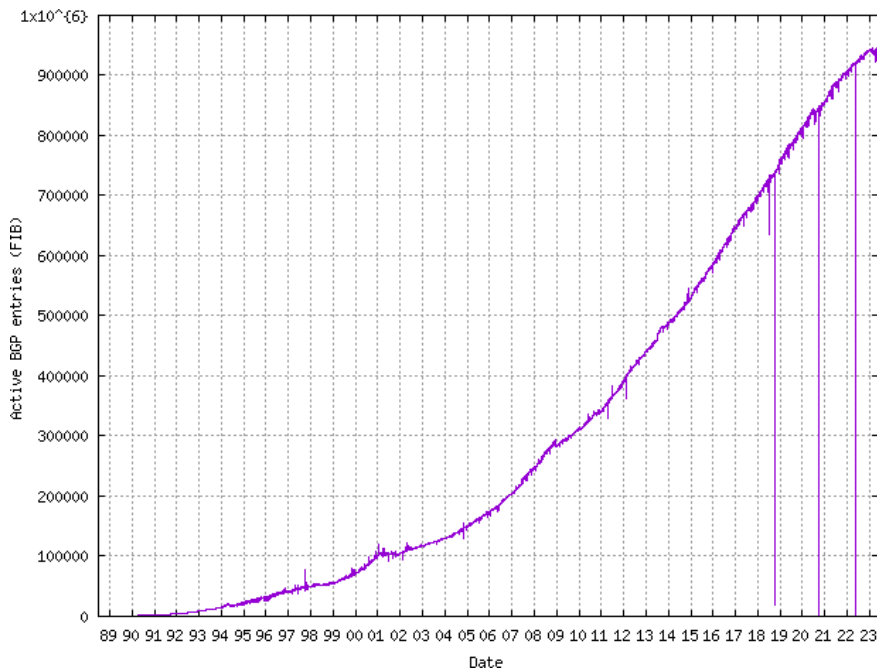


legend:

-  provider network
-  customer network:

- ❑ A advertises to B the path AW
- ❑ B advertises to X the path BAW
- ❑ Should B advertise to C the path BAW?
 - ❑ 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!





Source: <https://www.cidr-report.org/>



What we've covered:

- Network layer services
- What's inside a router?
- IPv4 & IPv6
- Routing algorithms & hierarchical routing
- Internet routing protocols RIP, OSPF, BGP

Some conclusions:

- Routing in large networks not only requires adequate routing algorithms for general graphs
 - Also an appropriate, hierarchical network structure is required
- Network structure has to be reflected in the addressing structure
 - Flat addresses would result in prohibitive overhead
- Different metrics and goals have to be fulfilled, in particular in inter-domain routing where optimality is only a single aspect



- [Car03] G. Carle. *Internet-Protokolle und -Komponenten*. Vorlesungsfolien, Universität Tübingen, 2003.
<http://net.informatik.uni-tuebingen.de/teaching/ipuk/index.php>
- [CLR90] T. Cormen, C. Leiserson, R. Rivest. *Introduction to Algorithms*. MIT Press, MA, USA, 1990.
- [KR04] J. F. Kurose, K. W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. 3rd edition, Addison Wesley, 2004.
- [Sar04] S. Sarkar. *Algorithms in Networking - Lecture 1*. Course slides of course TCOM 799, School of Engineering & Applied Science, University of Pennsylvania, PA, USA, 2004.
- [Ste95] M. Steenstrup. *Routing in Communication Networks*. Prentice Hall, 1995.
- [Sud04] T. Suda. *Computer Networks*. Course slides, Department of Information and Computer Science, University of California, Irvine, USA, 2004.
<http://www.ics.uci.edu/~ics243a>

