


Network Layer


CS 3516 - Computer Networks



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



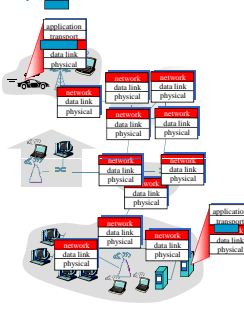

Chapter 4: Network Layer

- 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

- Transport segment from sending to receiving host
- On sending side encapsulates segments into datagrams
- On rcvng side, delivers segments to transport layer
- Network layer protocols in *every* host and router
- Router examines header fields in all IP datagrams passing through it





Two Key Network-Layer Functions

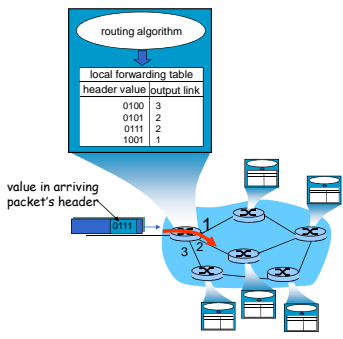
- **forwarding:** move packets from router's input to appropriate router output
- **routing:** determine route taken by packets from source to destination
 - routing algorithms

analogy:


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



Interplay Between Routing and Forwarding



| header value | output link |
|--------------|-------------|
| 0100 | 3 |
| 0101 | 2 |
| 0111 | 2 |
| 1001 | 1 |



Connection Setup

- 3rd important function in *some* network architectures:
 - ATM, frame relay, X.25
- Before datagrams flow, two end hosts *and* intervening routers establish virtual connection
 - routers get involved
- Network vs Transport Layer connection service:
 - **network**: between two hosts (may also involve intervening routers in case of Virtual Circuits (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



Example Network Layer Service Models

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

source-to-dest path behaves much like telephone circuit

- Performance-wise (predictable service)
- 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)



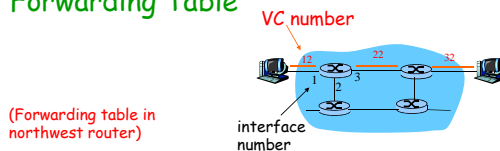
VC Implementation

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)

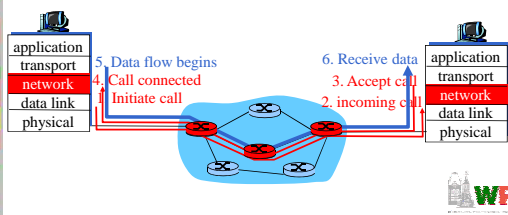
| Incoming interface | Incoming VC # | Outgoing interface | Outgoing VC # |
|--------------------|---------------|--------------------|---------------|
| 1 | 12 | 3 | 22 |
| 2 | 63 | 1 | 18 |
| 3 | 7 | 2 | 17 |
| 1 | 97 | 3 | 87 |
| ... | ... | ... | ... |

Routers maintain connection state information!



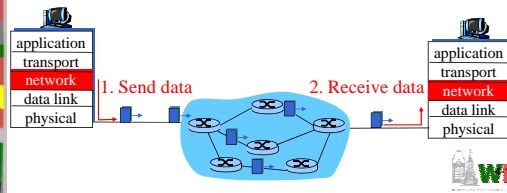
Virtual Circuits: Signaling Protocols

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



Datagram Networks

- Must do 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

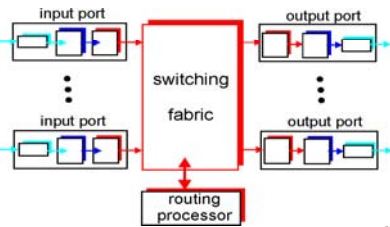
- 4.1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What's inside a router
- 4.4 IP: Internet Protocol
 - Datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
- 4.5 Routing algorithms
 - Link state
 - Distance Vector
 - Hierarchical routing
- 4.6 Routing in the Internet
 - RIP
 - OSPF
 - BGP
- 4.7 Broadcast and multicast routing



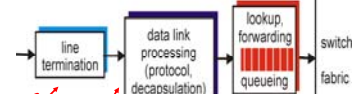
Router Architecture Overview

Two key router functions:

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



Input Port Functions



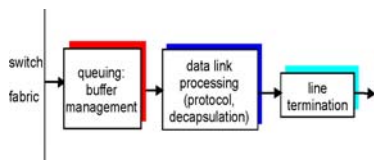
Physical layer:
bit-level reception
Data link layer:
e.g., Ethernet
(see chapter 5)

Decentralized switching:

- Given datagram destination, lookup output port using forwarding table in input port memory
- Goal: complete input port processing at 'line speed'
- Queueing: if datagrams arrive faster than forwarding rate into switch fabric



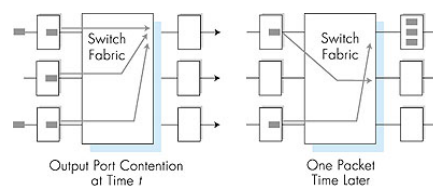
Output Ports



- Buffering** required when datagrams arrive from fabric faster than the transmission rate
- Scheduling discipline** chooses among queued datagrams for transmission
(More on queueing next slides...)



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?

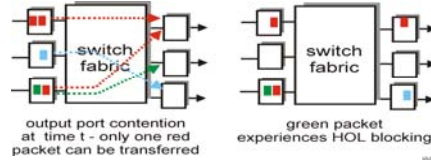
- RFC 3439 rule of thumb: average buffering equal to "typical" RTT (say 250 msec) times link capacity C (so, $RTT \cdot C$)
 - e.g., $C = 10$ Gps link \rightarrow 2.5 Gbit buffer
- Recent recommendation: with N flows, buffering equal to

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



Input Port Queuing

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



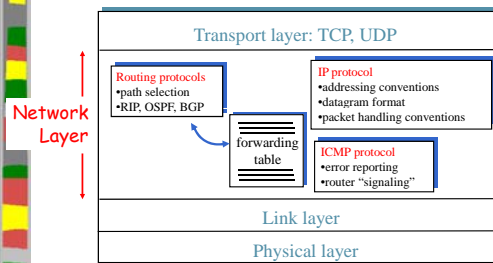
Chapter 4: Network Layer

- 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



The Internet Network layer

Host, router network layer functions:

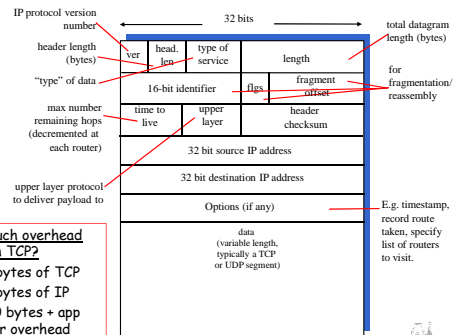


Chapter 4: Network Layer

- 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



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

fragmentation:
one large datagram
into 3 smaller datagrams

reassembly

IP Fragmentation and Reassembly

Example

- 4000 byte datagram
- MTU = 1500 bytes

One large datagram becomes several smaller datagrams

| length | ID | fragflag | offset |
|--------|----|----------|--------|
| =4000 | =X | =0 | =0 |
| =1500 | =X | =1 | =0 |
| =1500 | =X | =1 | =185 |
| =1040 | =X | =0 | =370 |

1480 bytes in data field

offset = 1480/8

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 Broadcasting and multicast routing

IP Addressing: Introduction

- IP address:** 32-bit identifier for host, router *interface*
- Interface:** connection between host/router and physical link
 - routers typically have multiple interfaces
 - hosts typically have one interface
 - IP addresses associated with each interface

223.1.1.1, 223.1.1.2, 223.1.1.3, 223.1.1.4, 223.1.2.1, 223.1.2.2, 223.1.3.1, 223.1.3.2, 223.1.3.27

223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

Subnets

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

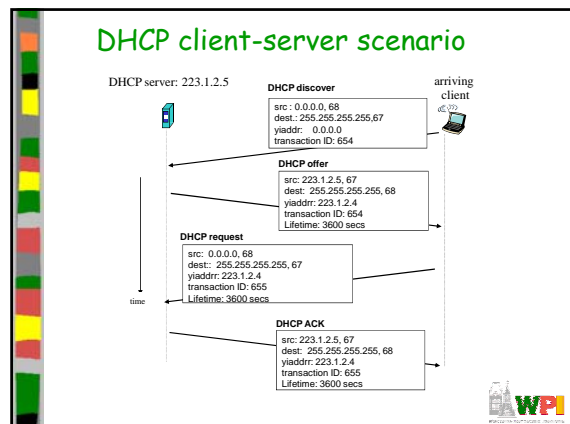
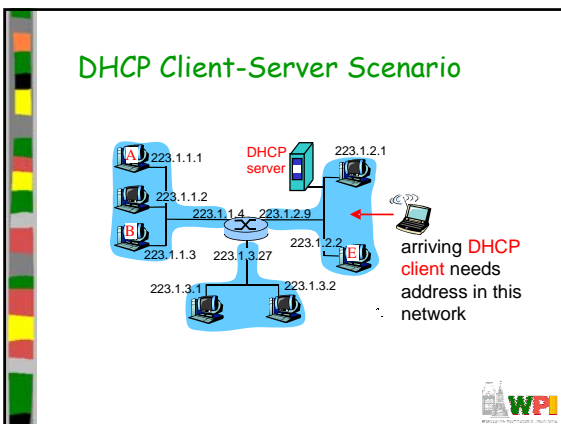
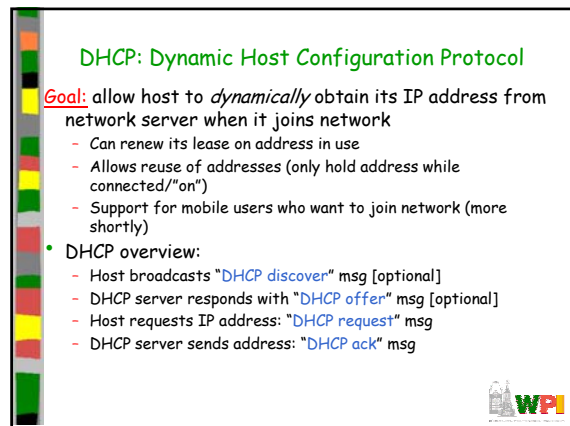
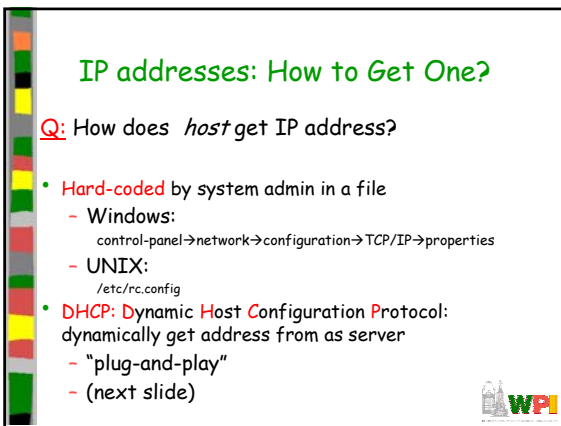
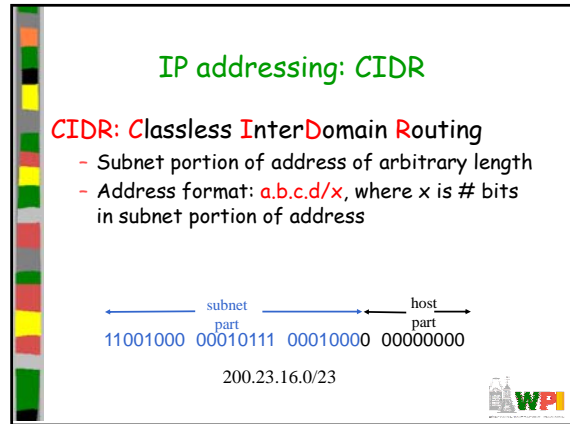
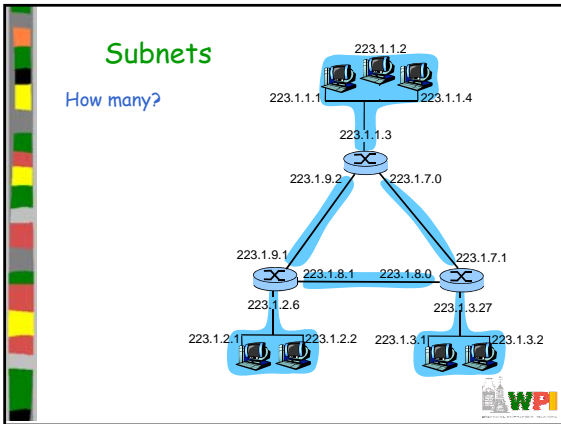
network consisting of 3 subnets

Subnets

Recipe

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

Subnet mask: /24



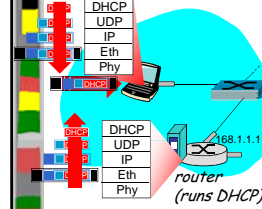
DHCP: more than IP address

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

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



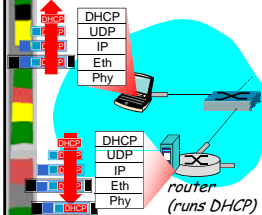
DHCP: Example



- Connecting laptop needs its IP address, addr of first-hop router, addr of DNS server → use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demux'd to IP demux'd, UDP demux'd to DHCP



DHCP: Example



- DHCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- Encapsulation of DHCP server, frame forwarded to client, demux'ing up to DHCP at client
- Client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router



DHCP: Wireshark Output (home LAN)

request

```

Message type: Boot Request (1)
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Scop flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0)
Relay agent IP address: 0.0.0.0 (0)
Client MAC address: Wistron_2
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (53, len=1) DHCP Message Type = DHCP Request
Option: (61) Client identifier
  Length: 7, Value: 010016D323688A;
  Hardware type: Ethernet
  Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Option: (50, len=4) Requested IP Address = 192.168.1.101
Option: (12, len=5) Host Name = "nomad"
Option: (55) Parameter Request List
  Length: 11, Value: 010F03062CE2F1F21F92B
  1 = Subnet Mask; 15 = Domain Name
  3 = Router; 6 = Domain Name Server
  44 = NetBIOS over TCP/IP Name Server
  
```

reply

```

Message type: Boot Reply (2)
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Scop flags: 0x0000 (Unicast)
Client IP address: 192.168.1.101 (192.168.1.101)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 192.168.1.1 (192.168.1.1)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (54) Message Type = DHCP ACK
Option: (51) Offered IP Address = 192.168.1.1
Option: (52) Offered IP Address = 192.168.1.1
Option: (53) Message Type = DHCP ACK
Option: (54) Offered IP Address = 192.168.1.1
Option: (55) Message Type = DHCP ACK
Option: (56) Offered IP Address = 192.168.1.1
Option: (57) Message Type = DHCP ACK
Option: (58) Offered IP Address = 192.168.1.1
Option: (59) Message Type = DHCP ACK
Option: (60) Offered IP Address = 192.168.1.1
Option: (61) Message Type = DHCP ACK
Option: (62) Offered IP Address = 192.168.1.1
Option: (63) Message Type = DHCP ACK
Option: (64) Offered IP Address = 192.168.1.1
Option: (65) Message Type = DHCP ACK
Option: (66) Offered IP Address = 192.168.1.1
Option: (67) Message Type = DHCP ACK
Option: (68) Offered IP Address = 192.168.1.1
Option: (69) Message Type = DHCP ACK
Option: (70) Offered IP Address = 192.168.1.1
Option: (71) Message Type = DHCP ACK
Option: (72) Offered IP Address = 192.168.1.1
Option: (73) Message Type = DHCP ACK
Option: (74) Offered IP Address = 192.168.1.1
Option: (75) Message Type = DHCP ACK
Option: (76) Offered IP Address = 192.168.1.1
Option: (77) Message Type = DHCP ACK
Option: (78) Offered IP Address = 192.168.1.1
Option: (79) Message Type = DHCP ACK
Option: (80) Offered IP Address = 192.168.1.1
Option: (81) Message Type = DHCP ACK
Option: (82) Offered IP Address = 192.168.1.1
Option: (83) Message Type = DHCP ACK
Option: (84) Offered IP Address = 192.168.1.1
Option: (85) Message Type = DHCP ACK
Option: (86) Offered IP Address = 192.168.1.1
Option: (87) Message Type = DHCP ACK
Option: (88) Offered IP Address = 192.168.1.1
Option: (89) Message Type = DHCP ACK
Option: (90) Offered IP Address = 192.168.1.1
Option: (91) Message Type = DHCP ACK
Option: (92) Offered IP Address = 192.168.1.1
Option: (93) Message Type = DHCP ACK
Option: (94) Offered IP Address = 192.168.1.1
Option: (95) Message Type = DHCP ACK
Option: (96) Offered IP Address = 192.168.1.1
Option: (97) Message Type = DHCP ACK
Option: (98) Offered IP Address = 192.168.1.1
Option: (99) Message Type = DHCP ACK
Option: (100) Offered IP Address = 192.168.1.1
Option: (101) Message Type = DHCP ACK
Option: (102) Offered IP Address = 192.168.1.1
Option: (103) Message Type = DHCP ACK
Option: (104) Offered IP Address = 192.168.1.1
Option: (105) Message Type = DHCP ACK
Option: (106) Offered IP Address = 192.168.1.1
Option: (107) Message Type = DHCP ACK
Option: (108) Offered IP Address = 192.168.1.1
Option: (109) Message Type = DHCP ACK
Option: (110) Offered IP Address = 192.168.1.1
Option: (111) Message Type = DHCP ACK
Option: (112) Offered IP Address = 192.168.1.1
Option: (113) Message Type = DHCP ACK
Option: (114) Offered IP Address = 192.168.1.1
Option: (115, len=20) Domain Name = "hsd1.ma.comcast.net."
  
```

3rd important function



IP Addressing: the Last Word...

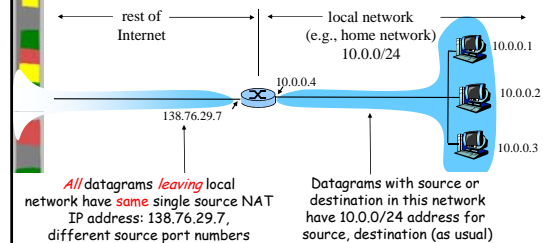
Q: How does an ISP get block of addresses?

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

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



NAT: Network Address Translation



NAT: Network Address Translation

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

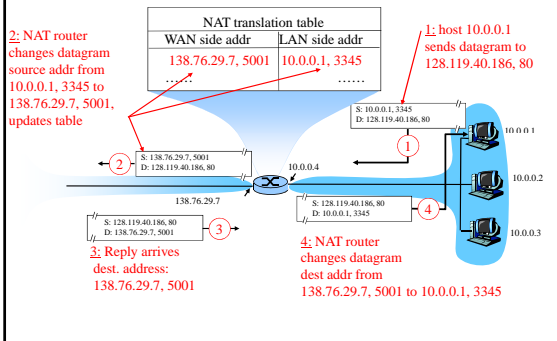


NAT: Network Address Translation

- **Implementation:** NAT router must:
 - *outgoing datagrams:* **replace** (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
 - (remote clients/servers will respond using (NAT IP address, new port #) as destination addr)
 - **remember** (in *NAT translation table*) every (source IP address, port #) to (NAT IP address, new port #) translation pair
 - *incoming datagrams:* **replace** (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table



NAT: Network Address Translation



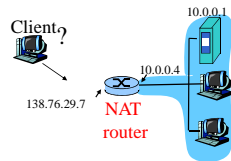
NAT: Network Address Translation

- **16-bit port-number field:**
 - ~60,000 simultaneous connections with a single LAN-side address!
- **NAT is controversial:**
 - Routers should only process up to layer 3
 - Violates "end-to-end" argument (complexity in ends)
 - NAT possibility must be taken into account by app designers, e.g., P2P applications
 - Address shortage should instead be solved by IPv6



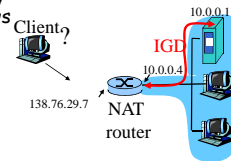
NAT Traversal Problem

- Client wants to connect to server with address 10.0.0.1
 - Server address 10.0.0.1 local to LAN (client can't use it as destination addr)
 - Only one externally visible NATted address: 138.76.29.7
- **Solution 1:** statically configure NAT to forward incoming connection requests at given port to server
 - e.g. (138.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000
 - But must be done ahead of time!



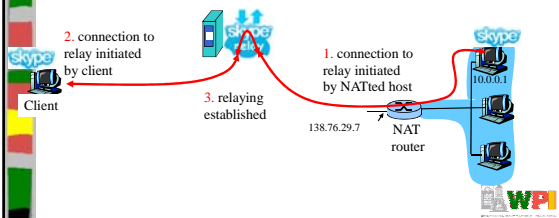
NAT Traversal Problem

- **Solution 2:** Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATted host to:
 - Learn public IP address (138.76.29.7)
 - Add/remove port mappings (with lease times)
 i.e. automate static NAT port map configuration
 - Still ahead of time, but automatic



NAT Traversal Problem

- **Solution 3: relaying (used in Skype)**
 1. NATed client establishes connection to relay
 2. External client connects to relay
 3. Relay bridges packets between to connections



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

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 m th datagram arrives to m th router:
 - Router discards datagram
 - And sends to source 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 for each router
- 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

- 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

IPv6

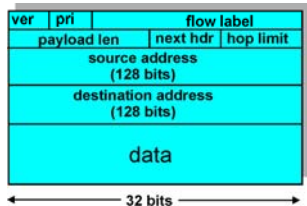
- **Initial motivation:** 32-bit address space soon to be completely allocated.
- **Additional motivation:**
 - header format helps speed processing/forwarding
 - header changes to facilitate QoS
- **IPv6 datagram format:**
 - fixed-length 40 byte header
 - no fragmentation allowed

IPv6 Header

Priority: identify priority among datagrams in flow

Flow Label: identify datagrams in same "flow" (concept of "flow" not well defined).

Next header: identify upper layer protocol for data



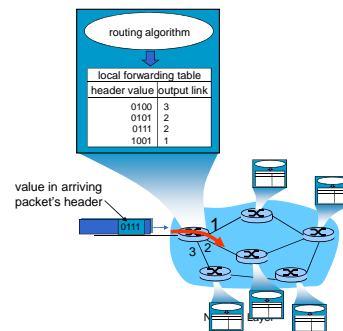
Other Changes from IPv4

- **Checksum:** removed entirely to reduce processing time at each hop
- **Options:** allowed, but outside of header, indicated by "Next Header" field
- **ICMPv6:** new version of ICMP
 - additional message types, e.g. "Packet Too Big"
 - multicast group management functions
- To help transition → **Tunneling:** IPv6 carried as payload in IPv4 datagram among IPv4 routers

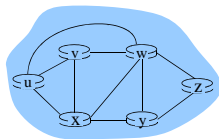
Chapter 4: Network Layer

- 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

Interplay between Routing, Forwarding



Graph Abstraction



Graph: $G = (N, E)$

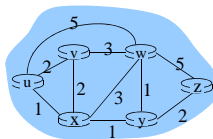
N = set of routers = $\{u, v, w, x, y, z\}$

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 (queuing)
- Note - cost of 1 means route is number of hops (common metric)

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

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

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm Classification

Global or decentralized information?

Global:

- All routers have complete topology, link cost info
- "link state" algorithms

Decentralized:

- Router knows physically-connected neighbors, link costs to neighbors
- Iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Static or dynamic?

Static:

- Routes change slowly over time

Dynamic:

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

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

A Link-State Routing Algorithm

Dijkstra's algorithm

- Network topology (graph), link costs known to all nodes
 - Accomplished via "link state broadcast"
 - All nodes have same info
- Compute least cost paths from one node ("source") to all other nodes
 - Gives forwarding table for that node
- Can be efficient ($O(n \log n)$, $n = \# \text{ nodes}$)

(See 4.5.1 for details and example)

Dijkstra's Algorithm - Output Example

From u, resulting shortest-path tree:

Resulting forwarding table in u:

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

Link State Updates via Flooding

- Send link information (cost, connection) to neighbors
- For each incoming packet, send to every outgoing link
 - Problems?
 - Vast numbers of duplicate packets
 - Infinite, actually, unless we stop. How?
 - Hop count: decrease each hop
 - Sequence number: don't flood twice
 - Selective flooding: send only in about the right direction

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

Distance Vector Algorithm

Bellman-Ford equation


Define

$$d_x(y) := \text{cost of least-cost path from } x \text{ to } y$$

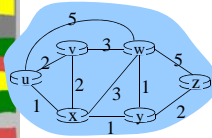
Then

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

where $\min\{\}$ is taken over all neighbors v of x



Bellman-Ford Example




Neighbors of u :
 $d_v(z) = 5, d_x(z) = 3, d_w(z) = 3$

B-F equation says:

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


$$= \min \{2 + 5, 1 + 3, 5 + 3\} = 4 \text{ (via } x \text{)}$$

Node that achieves minimum is next hop in shortest path (via x above)
 → that goes in forwarding table



Distance Vector Algorithm - State


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



Distance Vector Algorithm - Idea

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

$$D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \text{ for each node } y \in N$$
- Under most conditions estimate $D_x(y)$ converges to the actual least cost $d_x(y)$



Distance Vector Algorithm - Updates

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

↓

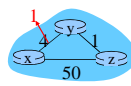
if DV to any dest has changed, notify neighbors



Distance Vector Algorithm - Link Cost Changes

Link cost changes:

- Node detects local link cost change
- Updates routing info, recalculates distance vector
- If DV changes, notify neighbors




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 .

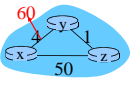
"good news travels fast"



Distance Vector Algorithm - Link Cost Changes


Link cost changes:

- Good news travels fast
- Bad news travels slowly
 - Right, 44 iterations before algorithm stabilizes (see text)
 - "Count to infinity" problem!



"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 not always completely solve count to infinity problem)



Comparison of LS and DV algorithms

Message complexity

- **LS:** with n nodes, E links, $O(nE)$ msgs sent
- **DV:** exchange between neighbors only

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
 - Somewhat limits damage


DV:

- DV node can advertise incorrect *path* cost
- Each node's table used by others
 - errors propagate thru network



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



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

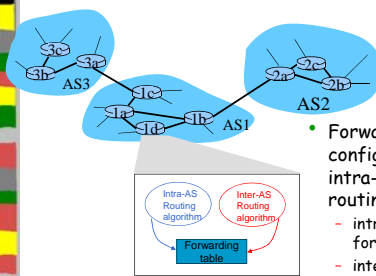


Hierarchical Routing


- 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



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)



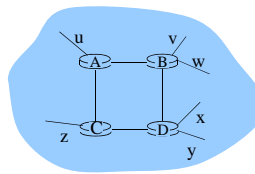
Chapter 4: Network Layer

- 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



RIP (Routing Information Protocol)

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



From router A to subnets:

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

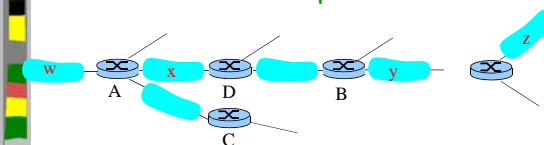


RIP Advertisements

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



RIP: Example



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

Routing/Forwarding table in D



RIP: Example

Advertisement from A to D

| Dest | Next hops |
|------|-----------|
| w | - 1 |
| x | - 1 |
| z | C 4 |
| | |

| Destination Network | Next Router | Num. of hops to dest. |
|---------------------|----------------|-----------------------|
| w | A | 2 |
| y | B | 2 |
| z | B A | 3 5 |
| x | -- | 1 |
| | | |

RIP: Link Failure and Recovery

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

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

RIP Table Processing

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

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

OSPF (Open Shortest Path First)

- "Open" means publicly available, in this context
- 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:
- **hierarchical** OSPF in large domains

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



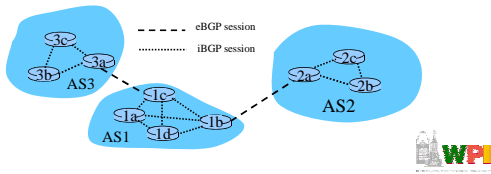
Internet Inter-AS routing: BGP

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



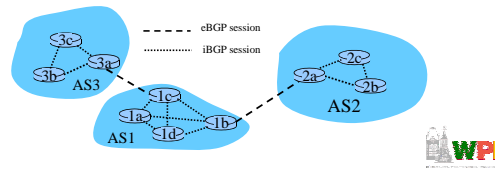
BGP Basics

- Pairs of routers (BGP peers) exchange routing info over semi-permanent TCP connections: **BGP sessions**
 - BGP sessions need not correspond to physical links.
- When AS2 advertises a prefix to AS1:
 - AS2 *promises* it will forward datagrams towards that prefix



Distributing Reachability Info

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



Path Attributes and BGP Routes

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



BGP Route Selection

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



BGP messages

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



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, reduces update traffic

Performance:

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



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

