IS 450/IS 650–Data Communications and Networks

Network Layer

Nirmalya Roy
Department of Information Systems
University of Maryland Baltimore County
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)
  ▪ broadcast, multicast

❖ 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
Recall Layering

- 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 actual traffic intersections
Interplay between routing and forwarding

Routing algorithm determines end-end-path through network.

Forwarding table determines local forwarding at this router.

Table:

<table>
<thead>
<tr>
<th>header value</th>
<th>output link</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100</td>
<td>3</td>
</tr>
<tr>
<td>0101</td>
<td>2</td>
</tr>
<tr>
<td>0111</td>
<td>2</td>
</tr>
<tr>
<td>1001</td>
<td>1</td>
</tr>
</tbody>
</table>

Value in arriving packet’s header: 0111

1 - 2 - 3
Connection setup

- 3rd important function in *some* network architectures:
  - ATM, frame relay, X.25

- before datagrams flow, two end hosts *and* intervening routers establish virtual connection
  - routers get involved

- network vs transport layer connection service:
  - *network*: between two hosts (may also involve intervening routers in case of VCs)
  - *transport*: between two processes
Q: What service model for “channel” transporting datagrams from sender to receiver?

**example services for individual datagrams:**
- guaranteed delivery
- guaranteed delivery with bounded delay (less than 40 msec delay)

**example services for a flow of datagrams:**
- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- guaranteed maximum jitter (restrictions on changes in inter-packet spacing)
- security
## Network layer service models:

<table>
<thead>
<tr>
<th>Network Architecture</th>
<th>Service Model</th>
<th>Bandwidth</th>
<th>Loss</th>
<th>Order</th>
<th>Timing</th>
<th>Congestion feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet</td>
<td>best effort</td>
<td>none</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no (inferred via loss)</td>
</tr>
<tr>
<td>ATM</td>
<td>CBR</td>
<td>constant rate</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no congestion</td>
</tr>
<tr>
<td>ATM</td>
<td>VBR</td>
<td>guaranteed rate</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no congestion</td>
</tr>
<tr>
<td>ATM</td>
<td>ABR</td>
<td>guaranteed minimum</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>ATM</td>
<td>UBR</td>
<td>none</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
Chapter 4: outline

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
Connection, connection-less service

- **datagram** network provides network-layer connectionless service
- **virtual-circuit** network provides network-layer connection service
- analogous to TCP/UDP connection-oriented/connectionless transport-layer services, but:
  - **service**: host-to-host
  - **no choice**: network provides one or the other
  - **implementation**: in network core
Two types of Network Architecture

- Connection-Oriented and Connection-Less

Virtual Circuit Switching
Example: ATM, X.25
Analogy: Telephone

Datagram forwarding
Example: IP networks
Analogy: Postal service
Virtual Circuits

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

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

A VC consists of:

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

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

forwarding table in northwest router:

<table>
<thead>
<tr>
<th>Incoming interface</th>
<th>Incoming VC #</th>
<th>Outgoing interface</th>
<th>Outgoing VC #</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>3</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>63</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>2</td>
<td>17</td>
</tr>
<tr>
<td>1</td>
<td>97</td>
<td>3</td>
<td>87</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

VC routers maintain connection state information!
Virtual circuits: signaling protocols

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

- No call setup at network layer
- @ routers: no state about end-to-end connections
  - no concept of “connection”
- packets forwarded using destination host address
  - May take different path for same source-dest pair
Datagram forwarding table

4 billion IP addresses, so rather than list individual destination address list range of addresses (aggregate table entries)

IP destination address in arriving packet’s header

<table>
<thead>
<tr>
<th>dest address</th>
<th>output link</th>
</tr>
</thead>
<tbody>
<tr>
<td>address-range 1</td>
<td>3</td>
</tr>
<tr>
<td>address-range 2</td>
<td>2</td>
</tr>
<tr>
<td>address-range 3</td>
<td>2</td>
</tr>
<tr>
<td>address-range 4</td>
<td>1</td>
</tr>
</tbody>
</table>
Datagram forwarding table

<table>
<thead>
<tr>
<th>Destination Address Range</th>
<th>Link Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111</td>
<td>0</td>
</tr>
<tr>
<td>11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111</td>
<td>1</td>
</tr>
<tr>
<td>11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111</td>
<td>2</td>
</tr>
<tr>
<td>otherwise</td>
<td>3</td>
</tr>
</tbody>
</table>

Q: but what happens if ranges don’t divide up so nicely?
Longest prefix matching

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

<table>
<thead>
<tr>
<th>Destination Address Range</th>
<th>Link interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000 00010111 00010*** ***********</td>
<td>0</td>
</tr>
<tr>
<td>11001000 00010111 00011000 ***********</td>
<td>1</td>
</tr>
<tr>
<td>11001000 00010111 00011*** ***********</td>
<td>2</td>
</tr>
<tr>
<td>otherwise</td>
<td>3</td>
</tr>
</tbody>
</table>

examples:

- **DA: 11001000 00010111 00010110 10100001**
- **DA: 11001000 00010111 00011000 10101010**

which interface?
Datagram or VC network: why?

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

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

- Thoughts on why VC isn’t great?

- Thoughts on why datagram may not be great?
  - Think of an application that’s better with VC
<table>
<thead>
<tr>
<th>Section</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>introduction</td>
</tr>
<tr>
<td>4.2</td>
<td>virtual circuit and datagram networks</td>
</tr>
<tr>
<td>4.3</td>
<td>what’s inside a router</td>
</tr>
<tr>
<td>4.4</td>
<td>IP: Internet Protocol</td>
</tr>
<tr>
<td></td>
<td>- datagram format</td>
</tr>
<tr>
<td></td>
<td>- IPv4 addressing</td>
</tr>
<tr>
<td></td>
<td>- ICMP</td>
</tr>
<tr>
<td></td>
<td>- IPv6</td>
</tr>
<tr>
<td>4.5</td>
<td>routing algorithms</td>
</tr>
<tr>
<td></td>
<td>- link state</td>
</tr>
<tr>
<td></td>
<td>- distance vector</td>
</tr>
<tr>
<td></td>
<td>- hierarchical routing</td>
</tr>
<tr>
<td>4.6</td>
<td>routing in the Internet</td>
</tr>
<tr>
<td></td>
<td>- RIP</td>
</tr>
<tr>
<td></td>
<td>- OSPF</td>
</tr>
<tr>
<td></td>
<td>- BGP</td>
</tr>
<tr>
<td>4.7</td>
<td>broadcast and multicast routing</td>
</tr>
</tbody>
</table>
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 (**match plus action**)
- goal: complete input port processing at ‘line speed’
- queuing: if datagrams arrive faster than forwarding rate into switch fabric
Chapter 4: outline

4.1 introduction
4.2 virtual circuit and datagram networks
4.3 what’s inside a router
4.4 IP: Internet Protocol
  o datagram format
  o IPv4 addressing
  o ICMP
  o IPv6
4.5 routing algorithms
  o link state
  o distance vector
  o hierarchical routing
4.6 routing in the Internet
  o RIP
  o OSPF
  o BGP
4.7 broadcast and multicast routing
The Internet network layer

host, router network layer functions:

- **transport layer**: TCP, UDP
  - **routing protocols**
    - path selection
    - RIP, OSPF, BGP
  - **IP protocol**
    - addressing conventions
    - datagram format
    - packet handling conventions
  - **ICMP protocol**
    - error reporting
    - router “signaling”

network layer

link layer

physical layer
# IP datagram format

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP protocol version</td>
<td>Number of the IP protocol version</td>
</tr>
<tr>
<td>header length (bytes)</td>
<td>Length of the IP header in bytes</td>
</tr>
<tr>
<td>“type” of data</td>
<td>Type of data being transported</td>
</tr>
<tr>
<td>max number remaining</td>
<td>Maximum number of remaining hops (decremented at each router)</td>
</tr>
<tr>
<td>time to live</td>
<td>Time to live for the IP header (decremented at each router)</td>
</tr>
<tr>
<td>upper layer protocol</td>
<td>Protocol used to deliver payload to the upper layer</td>
</tr>
<tr>
<td>32 bit source IP address</td>
<td>Source IP address for the IP datagram</td>
</tr>
<tr>
<td>32 bit destination IP address</td>
<td>Destination IP address for the IP datagram</td>
</tr>
<tr>
<td>options (if any)</td>
<td>Options if any specified</td>
</tr>
<tr>
<td>fragment offset</td>
<td>Fragment offset for fragmentation/reassembly</td>
</tr>
<tr>
<td>checksum</td>
<td>Header checksum for the IP datagram</td>
</tr>
<tr>
<td>data</td>
<td>Data portion of the IP datagram (variable length, typically a TCP or UDP segment)</td>
</tr>
</tbody>
</table>

## How much overhead?

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

[Diagram of IP datagram format with annotations]
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

**Diagram:**
- **Fragmentation:** in: one large datagram
  out: 3 smaller datagrams
- **Reassembly:**
# IP fragmentation, reassembly

**example:**

- 4000 byte datagram
- MTU = 1500 bytes

One large datagram becomes several smaller datagrams.

<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>fragflag</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

1480 bytes in data field

offset = \( \frac{1480}{8} \)

<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>fragflag</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1500</td>
<td>x</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

1020 bytes in data field

<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>fragflag</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1500</td>
<td>x</td>
<td>1</td>
<td>185</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>fragflag</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1040</td>
<td>x</td>
<td>0</td>
<td>370</td>
</tr>
</tbody>
</table>
Chapter 4: outline

4.1 Introduction
4.2 virtual circuit and datagram networks
4.3 what’s inside a router
4.4 IP: Internet Protocol
   o datagram format
   o IPv4 addressing
   o ICMP
   o IPv6
4.5 routing algorithms
   o link state
   o distance vector
   o hierarchical routing
4.6 routing in the Internet
   o RIP
   o OSPF
   o BGP
4.7 broadcast and multicast routing
IP Addressing: introduction

- **IP address**: 32-bit identifier for host, router *interface*

- **interface**: connection between host/router and physical link
  - router’s typically have multiple interfaces
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
  - IP addresses associated with each interface

**Example IP Addresses:**
- 223.1.1.1
- 223.1.1.2
- 223.1.1.3
- 223.1.1.4
- 223.1.2.1
- 223.1.2.2
- 223.1.2.9
- 223.1.3.1
- 223.1.3.2
- 223.1.3.27

**Binary Representation:**
- 223.1.1.1 = 11011111 00000001 00000001 00000001
IP addressing: introduction

**Q:** how are interfaces actually connected?

**A:** wired Ethernet interfaces connected by Ethernet switches

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

**A:** wireless WiFi interfaces connected by WiFi base station
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

Subnet

223.1.1.1
223.1.1.2
223.1.1.3
223.1.1.4
223.1.2.9
223.1.2.1
223.1.2.2
223.1.3.1
223.1.3.2
223.1.3.27
**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*

```
Subnets
223.1.1.0/24
223.1.2.0/24
223.1.3.0/24

223.1.1.1
223.1.1.2
223.1.1.3
223.1.1.4
223.1.2.9
223.1.3.2
223.1.3.27
223.1.3.28
223.1.2.2

subnet mask: /24
```
Subnets

how many?
IP addressing: CIDR

CIDR: Classless InterDomain Routing
- subnet portion of address of arbitrary length
- address format: \(a.b.c.d/x\), where \(x\) is \# bits in subnet portion of address

11001000 00010111 00010000 00000000

200.23.16.0/23
## Q: How does network get subnet part of IP addr?

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

<table>
<thead>
<tr>
<th>ISP's block</th>
<th>11001000_00010111_00010000_00000000</th>
<th>200.23.16.0/20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization 0</td>
<td>11001000_00010111_00010000_00000000</td>
<td>200.23.16.0/23</td>
</tr>
<tr>
<td>Organization 1</td>
<td>11001000_00010111_00010010_00000000</td>
<td>200.23.18.0/23</td>
</tr>
<tr>
<td>Organization 2</td>
<td>11001000_00010111_00010100_00000000</td>
<td>200.23.20.0/23</td>
</tr>
<tr>
<td>...</td>
<td>......</td>
<td>....</td>
</tr>
<tr>
<td>Organization 7</td>
<td>11001000_00010111_00011110_00000000</td>
<td>200.23.30.0/23</td>
</tr>
</tbody>
</table>
Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:

Organizations:
- Organization 0: 200.23.16.0/23
- Organization 1: 200.23.18.0/23
- Organization 2: 200.23.20.0/23
- Organization 7: 200.23.30.0/23

ISP Organizations:
- Fly-By-Night-ISP
- ISPs-R-Us

Internet:“Send me anything with addresses beginning 200.23.16.0/20”

“Send me anything with addresses beginning 199.31.0.0/16”
Hierarchical addressing: more specific routes

ISP's-R-Us has a more specific route to Organization 1.
IP addressing: the last word...

**Q:** how does an ISP get block of addresses?

**A:** ICANN: Internet Corporation for Assigned Names and Numbers http://www.icann.org/

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes
**IP addresses: how to get one?**

**Q:** How does a *host* get IP address?

- **hard-coded by system admin in a file**
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config

- **DHCP: Dynamic Host Configuration Protocol:** dynamically get address from as server
  - “plug-and-play”
**DHCP: Dynamic Host Configuration Protocol**

**Goal:** allow host to *dynamically* obtain its IP address from network server when it joins network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/“on”)
- support for mobile users who want to join network (more shortly)

**DHCP overview:**

- host broadcasts “DHCP discover” msg [optional]
- DHCP server responds with “DHCP offer” msg [optional]
- host requests IP address: “DHCP request” msg
- DHCP server sends address: “DHCP ack” msg
DHCP client-server scenario

DHCP server

arriving DHCP client needs address in this network
DHCP client-server scenario

DHCP server: 223.1.2.5

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

DHCP offer
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 654
lifetime: 3600 secs

DHCP request
src: 0.0.0.0, 68
dest: 255.255.255.255, 67
yiaddr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

DHCP ACK
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs
DHCP: more than IP addresses

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)
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 802.11 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP, UDP demuxed 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.
- Client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router.
- Encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client.

Diagram:
- DHCP server
- UDP
- IP
- Eth
- Phy

Router with DHCP server built into router

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)

Message type: **Boot Request (1)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
**Transaction ID: 0x6b3a11b7**
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
**Option:** (t=53,l=1) DHCP Message Type = DHCP Request

**Option:** (61) Client identifier
  - Length: 7; Value: 010016D323688A;
  - Hardware type: Ethernet
  - Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
**Option:** (t=50,l=4) Requested IP Address = 192.168.1.101
**Option:** (t=12,l=5) Host Name = "nomad"

**Option:** (55) Parameter Request List
  - Length: 11; Value: 010F03062C2E2F1F21F92B
  1 = Subnet Mask; 15 = Domain Name
  3 = Router; 6 = Domain Name Server
  44 = NetBIOS over TCP/IP Name Server
  ……

Message type: **Boot Reply (2)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
**Transaction ID: 0x6b3a11b7**
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 192.168.1.101 (192.168.1.101)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 192.168.1.1 (192.168.1.1)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
**Option:** (t=53,l=1) DHCP Message Type = DHCP ACK
**Option:** (t=54,l=4) Server Identifier = 192.168.1.1
**Option:** (t=1,l=4) Subnet Mask = 255.255.255.0
**Option:** (t=3,l=4) Router = 192.168.1.1
**Option:** (6) Domain Name Server
  - Length: 12; Value: 445747E2445749F244574092;
  - IP Address: 68.87.71.226;
  - IP Address: 68.87.73.242;
  - IP Address: 68.87.64.146
**Option:** (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."
Network Address Translation
Scalability Problem

- Internet growing very fast
  - Many million devices
  - Each device needs an address for communication

- Question is
  - How do you address each of them
  - IP addressing can give you $2^{32}$
  - May not be enough
NAT: Network Address Translation

All datagrams leaving local network have same single source NAT IP address: 138.76.29.7, different source port numbers.

Datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual).
NAT: network address translation

**motivation:** local network uses just one IP address as far as outside world is concerned:

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

**Implementation: **NAT router must:

- **Outgoing datagrams: replace** (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #) . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr

- **Remember (in NAT translation table) **every (source IP address, port #) to (NAT IP address, new port #) translation pair

- **Incoming datagrams: replace** (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table
**NAT: network address translation**

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

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

3: reply arrives dest. address: 138.76.29.7, 5001

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

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!

- NAT is controversial:
  - port # are for addressing processes, not for addressing hosts
  - routers should only process up to layer 3
  - violates end-to-end argument
    - Host should be talking directly with each other, without interfering nodes modifying IP addresses and port numbers
    - 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 NATed address: 138.76.29.7

- **solution1:** statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000
NAT traversal problem

- **solution 2**: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed 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
solution 3: relaying (used in Skype)
- NATed client establishes connection to relay
- external client connects to relay
- relay bridges packets between two connections
NAT makes Globally non-routable hosts

- Non-routable
  - Means you cannot ping 192.168.0.3 (your home machines) from UMBC Lab

- But, Skype, GotoMyPC, etc. can access / call your home machine
  - How?
Chapter 4: outline

4.1 introduction
4.2 virtual circuit and datagram networks
4.3 what’s inside a router
4.4 IP: Internet Protocol
   o datagram format
   o IPv4 addressing
   o ICMP
   o IPv6

4.5 routing algorithms
   o link state
   o distance vector
   o hierarchical routing

4.6 routing in the Internet
   o RIP
   o OSPF
   o 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

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>echo reply (ping)</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>dest. network unreachable</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>dest host unreachable</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>dest protocol unreachable</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>dest port unreachable</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>dest network unknown</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>dest host unknown</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>source quench (congestion control - not used)</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>echo request (ping)</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>route advertisement</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>router discovery</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>TTL expired</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>bad IP header</td>
</tr>
</tbody>
</table>
Traceroute and ICMP

- Source sends series of UDP segments to dest:
  - First set has TTL = 1
  - Second set has TTL = 2, etc.
  - Unlikely port number

- When nth set of datagrams arrives to nth router:
  - Router discards datagrams
  - And sends source ICMP messages (type 11, code 0)
  - ICMP messages includes name of router & IP address

- When ICMP messages arrives, source records RTTs

Stopping criteria:
- UDP segment eventually arrives at destination host
- Destination returns ICMP “port unreachable” message (type 3, code 3)
- Source stops
IPv6: motivation

- initial motivation: 32-bit address space soon to be completely allocated.
- expanded addressing capabilities

- additional motivation:
  - A streamlined 40-byte header
    - header format helps speed processing/forwarding
  - Flow labeling and priority
    - header changes to facilitate QoS

IPv6 datagram format:

- fixed-length 40 byte header
- no fragmentation allowed
IPv6 datagram format

**Priority/traffic class**: 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

![IPv6 datagram format diagram](image_url)
IPv4 & IPv6 side-by-side comparison

Header size?
- 20 byte for IPv4
- 40 byte fixed length for IPv6

<table>
<thead>
<tr>
<th>ver</th>
<th>head. len</th>
<th>type of service</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit identifier</td>
<td>flgs</td>
<td>fragment offset</td>
<td></td>
</tr>
<tr>
<td>time to live</td>
<td>upper layer protocol</td>
<td>header checksum</td>
<td></td>
</tr>
</tbody>
</table>

32 bit source IP address
32 bit destination IP address
options (if any)
data
(variable length, typically a TCP or UDP segment)

<table>
<thead>
<tr>
<th>ver</th>
<th>pri</th>
<th>flow label</th>
</tr>
</thead>
<tbody>
<tr>
<td>payload len</td>
<td>next hdr</td>
<td>hop limit</td>
</tr>
</tbody>
</table>

source address (128 bits)
destination address (128 bits)
data

32 bits
Other changes from IPv4

- **fragmentation/reassembly**: only be performed by the source and destination host
- **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
    - Internet Group Management Protocol (IGMP)
Transition from IPv4 to IPv6

- not all routers can be upgraded simultaneously
  - no “flag days”
  - how will network operate with mixed IPv4 and IPv6 routers?

- **tunneling**: IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers
Tunneling

**logical view:**
IPv4 tunnel connecting IPv6 routers

**physical view:**
Tunneling

logical view:

physical view:

IPv4 tunnel connecting IPv6 routers
An Alternate Approach: IPv6

- **Initial motivation:** Make space for 64 bit address space
  - How can this be made compatible to IPv4 routers?

- **IPv6 not flying**
  - NAT coping fine with today’s needs
Chapter 4: outline

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
Routing - Why Difficult?

- Several algorithmic problems:
  - Many many paths - which is the best?
  - Each path has changing characteristics
    - Queuing time varies, losses happen, router down ...
  - How do you broadcast (find where someone is)
  - How do you multicast (webTV, conference call)
  - How do routers perform routing at GBbps scale

- Several management problems:
  - How do you detect/diagnose faults
  - How do you do pricing, accounting
Interplay between routing, forwarding

The routing algorithm determines the end-end-path through the network. The local forwarding table determines the output link based on the destination address in the arriving packet's header.
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

What factors influence this cost?

Should costs be only on links?

c(x, x') = cost of link (x, x')
e.g., c(w, z) = 5

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

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

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

Routing algorithm: algorithm that finds least-cost path
Routing algorithm classification

Q: 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

Q: static or dynamic?

*static:*
- routes change slowly over time

*dynamic:*
- routes change more quickly
  - periodic update
  - in response to link cost changes
A Link-State Routing Algorithm

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

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

1 **Initialization:**
2 \( N' = \{u\} \)
3 for all nodes \( v \)
4 if \( v \) adjacent to \( u \)
5 then \( D(v) = c(u,v) \)
6 else \( D(v) = \infty \)

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

**Notation:**
- \( c(x,y) \): link cost from node \( x \) to \( y \); \( = \infty \) if not direct neighbors
- \( D(v) \): current value of cost of path from source to dest. \( v \)
Dijkstra’s algorithm: example

<table>
<thead>
<tr>
<th>Step</th>
<th>N'</th>
<th>D(v)</th>
<th>D(w)</th>
<th>D(x)</th>
<th>D(y)</th>
<th>D(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>p(v)</td>
<td>p(w)</td>
<td>p(x)</td>
<td>p(y)</td>
<td>p(z)</td>
</tr>
<tr>
<td>0</td>
<td>u</td>
<td>7,u</td>
<td>3,u</td>
<td>5,u</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td>uw</td>
<td>6,w</td>
<td>5,u</td>
<td>11,w</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>uwx</td>
<td>x</td>
<td>6,w</td>
<td>11,w</td>
<td>14,x</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>uwxv</td>
<td></td>
<td>10,v</td>
<td>14,x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>uwxvy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12,y</td>
</tr>
<tr>
<td>5</td>
<td>uwxvzy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes:
- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)
### Dijkstra’s algorithm: another example

<table>
<thead>
<tr>
<th>Step</th>
<th>N'</th>
<th>D(v),p(v)</th>
<th>D(w),p(w)</th>
<th>D(x),p(x)</th>
<th>D(y),p(y)</th>
<th>D(z),p(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>u</td>
<td>2, u</td>
<td>5, u</td>
<td>1, u</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td>ux</td>
<td>2, u</td>
<td>4, x</td>
<td>2, x</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>2</td>
<td>uxy</td>
<td>2, u</td>
<td>3, y</td>
<td>2, x</td>
<td>4, y</td>
<td>4, y</td>
</tr>
<tr>
<td>3</td>
<td>uxyv</td>
<td>2, u</td>
<td>3, y</td>
<td>2, x</td>
<td>4, y</td>
<td>4, y</td>
</tr>
<tr>
<td>4</td>
<td>uxyvw</td>
<td>2, u</td>
<td>3, y</td>
<td>2, x</td>
<td>4, y</td>
<td>4, y</td>
</tr>
<tr>
<td>5</td>
<td>uxyv wz</td>
<td>2, u</td>
<td>3, y</td>
<td>2, x</td>
<td>4, y</td>
<td>4, y</td>
</tr>
</tbody>
</table>

**Diagram:**

- Node U connected to X and V with weights 1 and 2 respectively.
- Node V connected to W with weight 5.
- Node W connected to Y with weight 3.
- Node Y connected to Z with weight 2.
- Node X connected to V with weight 3.
- Node Z connected to Y with weight 1.
- Path from U to Z via X, Y, and W.

This diagram illustrates the step-by-step process of Dijkstra’s algorithm as nodes are added to the set N'.
Dijkstra’s algorithm: example (2)

resulting shortest-path tree from u:

![Graph diagram]

resulting forwarding table in u:

<table>
<thead>
<tr>
<th>destination</th>
<th>link</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>(u,v)</td>
</tr>
<tr>
<td>x</td>
<td>(u,x)</td>
</tr>
<tr>
<td>y</td>
<td>(u,x)</td>
</tr>
<tr>
<td>w</td>
<td>(u,x)</td>
</tr>
<tr>
<td>z</td>
<td>(u,x)</td>
</tr>
</tbody>
</table>
Dijkstra’s algorithm, discussion

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

**Oscillations possible:**
- e.g., support link cost equals amount of carried traffic:

---

Initially

\[
\begin{array}{c}
\text{A} \\
\downarrow 1 \\
\text{D} \\
\| \\
\text{0} \\
\| \\
\text{C} \\
\| \\
\text{B} \\
\end{array}
\]

\[
\begin{array}{c}
\text{A} \\
\| \\
\text{D} \\
\| \\
\text{1} \\
\| \\
\text{B} \\
\| \\
\text{C} \\
\end{array}
\]

\[
\begin{array}{c}
\text{A} \\
\| \\
\text{D} \\
\| \\
\text{1+e} \\
\| \\
\text{B} \\
\| \\
\text{C} \\
\end{array}
\]

\[
\begin{array}{c}
\text{A} \\
\| \\
\text{D} \\
\| \\
\text{2+e} \\
\| \\
\text{B} \\
\| \\
\text{C} \\
\end{array}
\]

---

Given these costs, find new routing.... resulting in new costs

\[
\begin{array}{c}
\text{A} \\
\downarrow 1 \\
\text{D} \\
\| \\
\text{0} \\
\| \\
\text{C} \\
\| \\
\text{B} \\
\end{array}
\]

\[
\begin{array}{c}
\text{A} \\
\| \\
\text{D} \\
\| \\
\text{1+e} \\
\| \\
\text{B} \\
\| \\
\text{C} \\
\end{array}
\]

\[
\begin{array}{c}
\text{A} \\
\| \\
\text{D} \\
\| \\
\text{0} \\
\| \\
\text{B} \\
\| \\
\text{C} \\
\end{array}
\]

\[
\begin{array}{c}
\text{A} \\
\| \\
\text{D} \\
\| \\
\text{2+e} \\
\| \\
\text{B} \\
\| \\
\text{C} \\
\end{array}
\]

---

Given these costs, find new routing.... resulting in new costs

\[
\begin{array}{c}
\text{A} \\
\downarrow 1 \\
\text{D} \\
\| \\
\text{0} \\
\| \\
\text{C} \\
\| \\
\text{B} \\
\end{array}
\]

\[
\begin{array}{c}
\text{A} \\
\| \\
\text{D} \\
\| \\
\text{1+e} \\
\| \\
\text{B} \\
\| \\
\text{C} \\
\end{array}
\]

\[
\begin{array}{c}
\text{A} \\
\| \\
\text{D} \\
\| \\
\text{0} \\
\| \\
\text{B} \\
\| \\
\text{C} \\
\end{array}
\]

\[
\begin{array}{c}
\text{A} \\
\| \\
\text{D} \\
\| \\
\text{1+e} \\
\| \\
\text{B} \\
\| \\
\text{C} \\
\end{array}
\]

---

Given these costs, find new routing.... resulting in new costs

\[
\begin{array}{c}
\text{A} \\
\downarrow 1 \\
\text{D} \\
\| \\
\text{0} \\
\| \\
\text{C} \\
\| \\
\text{B} \\
\end{array}
\]

\[
\begin{array}{c}
\text{A} \\
\| \\
\text{D} \\
\| \\
\text{1+e} \\
\| \\
\text{B} \\
\| \\
\text{C} \\
\end{array}
\]

\[
\begin{array}{c}
\text{A} \\
\| \\
\text{D} \\
\| \\
\text{0} \\
\| \\
\text{B} \\
\| \\
\text{C} \\
\end{array}
\]

\[
\begin{array}{c}
\text{A} \\
\| \\
\text{D} \\
\| \\
\text{2+e} \\
\| \\
\text{B} \\
\| \\
\text{C} \\
\end{array}
\]

---

Given these costs, find new routing.... resulting in new costs
Chapter 4: outline

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
To find D, node S asks each neighbor X

- How far X is from D
- X asks its neighbors ... comes back and says C(X,D)
- Node S deduces \( C(S,D) = C(S,X) + C(X,D) \)
- S chooses neighbor \( X_i \) that provides min \( C(S,D) \)

Later, \( X_j \) may find better route to D

- \( X_j \) advertizes \( C(X_j,D) \)
- All nodes update their cost to D if new min found
Distance vector algorithm

*Bellman-Ford equation (dynamic programming)*

let

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

then

\[ d_x(y) = \min \{ c(x,v) + d_v(y) \} \]

\[ \text{cost from neighbor } v \text{ to destination } y \]
\[ \text{cost to neighbor } v \]

\[ \min \text{ taken over all neighbors } v \text{ of } x \]
clearly, \( d_v(z) = 5 \), \( d_x(z) = 3 \), \( d_w(z) = 3 \)

B-F equation says:

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

\[
= \min \{ 2 + 5, \quad 1 + 3, \quad 5 + 3 \} = 4
\]

node achieving minimum is next hop in shortest path, used in forwarding table
Distance vector algorithm

- $D_x(y) = \text{estimate of least cost from } x \text{ to } y$
  - $x$ maintains distance vector $D_x = [D_x(y) : y \in N]$

- node $x$:
  - knows cost to each neighbor $v$: $c(x,v)$
  - maintains its neighbors’ distance vectors. For each neighbor $v$, $x$ maintains $D_v = [D_v(y) : y \in N]$
Distance vector algorithm

**key idea:**

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when $x$ receives new DV estimate from neighbor, it updates its own DV using B-F equation:
  \[
  D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N
  \]
- under minor, natural conditions, the estimate $D_x(y)$ converge to the actual least cost $d_x(y)$
Distance vector algorithm

**iterative, asynchronous:**
- each local iteration caused by:
  - local link cost change
  - DV update message from neighbor

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

**each node:**
- *wait* for (change in local link cost or msg from neighbor)
- *recompute* estimates
- if DV to any dest has changed, *notify* neighbors
\( D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} \)
\[ = \min\{2+0, 7+1\} = 2 \]

\( D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} \)
\[ = \min\{2+1, 7+0\} = 3 \]
### node x table

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>y</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>z</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>

### node y table

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>y</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>z</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>

### node z table

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>y</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>z</td>
<td>7</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

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

---

![Graph](image.png)
Chapter 4: outline

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 600 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:*
- at “edge” of its own AS
- has link to router in another AS
Interconnected ASes

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

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

**AS1 must:**

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

*job of inter-AS routing!*
Example: setting forwarding table in router 1d

- Suppose AS1 learns (via inter-AS protocol) that subnet $x$ reachable via AS3 (gateway 1c), but not via AS2
  - inter-AS protocol propagates reachability info to all internal routers
- Router 1d determines from intra-AS routing info that its interface $I$ is on the least cost path to 1c
  - Installs forwarding table entry $(x,I)$
Example: choosing among multiple ASes

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

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

- *hot potato routing:* send packet towards closest of two routers.

Learn from inter-AS protocol that subnet $x$ is reachable via multiple gateways

Use routing info from intra-AS protocol to determine costs of least-cost paths to each of the gateways

Hot potato routing: choose the gateway that has the smallest least cost

Determine from forwarding table the interface $I$ that leads to least-cost gateway. Enter $(x, I)$ in forwarding table.
Chapter 4: outline

4.1 introduction
4.2 virtual circuit and datagram networks
4.3 what’s inside a router
4.4 IP: Internet Protocol
   o datagram format
   o IPv4 addressing
   o ICMP
   o IPv6
4.5 routing algorithms
   o link state
   o distance vector
   o hierarchical routing
4.6 routing in the Internet
   o RIP
   o OSPF
   o 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)
RIP (Routing Information Protocol)

- included in BSD-UNIX distribution in 1982
- distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors every 30 sec in response message (aka advertisement)
  - each advertisement: list of up to 25 destination subnets (in IP addressing sense)

from router A to destination subnets:

<table>
<thead>
<tr>
<th>subnet</th>
<th>hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>1</td>
</tr>
<tr>
<td>v</td>
<td>2</td>
</tr>
<tr>
<td>w</td>
<td>2</td>
</tr>
<tr>
<td>x</td>
<td>3</td>
</tr>
<tr>
<td>y</td>
<td>3</td>
</tr>
<tr>
<td>z</td>
<td>2</td>
</tr>
</tbody>
</table>
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
- advertisements flooded to entire AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP)
- IS-IS routing protocol: nearly identical to OSPF
Internet inter-AS routing: BGP

■ BGP (Border Gateway Protocol): *the* de facto inter-domain routing protocol
  ○ “glue that holds the Internet together”

■ BGP provides each AS a means to:
  ○ eBGP: obtain subnet reachability information from neighboring ASs.
  ○ iBGP: propagate reachability information to all AS-internal routers.
  ○ determine “good” routes to other networks based on reachability information and policy.

■ allows subnet to advertise its existence to rest of Internet: “I am here”
Routing in Internet Analogy

- Similar to International FedEx routing
  - FedEx figures out the best route within country
    - Uses Google maps
    - This is link state -- All info available

- USA FedEx does not have international map, also no permission to operate outside USA
  - Gets price quote from Germany FedEx, Japan FedEx etc. to route to India
  - Chooses minimum price and handles package to say Germany (Distance Vector)
  - Germany has country map (link state)
  - Germany asks for cost from Egypt, South Africa ...
Internet Routing

- Think of each country FedEx as ISPs
  - Routing on internet very similar to prior example

- The link state and DV routing protocols used in internet routing
  - RIP (routing information protocol)
  - OSPF (Open shortest path first)
  - BGP (Border gateway protocol)

- They utilize the concepts of
  - Link state
  - Distance vector routing
How is this different in wireless?
Routing in Wireless Mobile Networks

- Imagine hundreds of hosts moving
  - Routing algorithm needs to cope up with varying wireless channel and node mobility

Where's RED guy
Chapter 4: done!

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

- understand principles behind network layer services:
  - network layer service models, forwarding versus routing how a router works, routing (path selection), broadcast, multicast
- instantiation, implementation in the Internet
Questions ?
Practice Problem (Subnet)

For an IP v4 address 200.23.16.0/22 (with binary format as 11001000 00010111 00010000 00000000), answer the following questions.

a) What is the subnet address of this IP? Show it in binary format.

b) How many hosts can this subnet accommodate?
Practice Problem (Congestion Window)

Assume that a TCP connection is just entering slow start phase at time 0. The congestion window (CWND) is set to 1 MSS (maximum segment size). Assume that RTT is 10ms. Omit the transmission time (this means packets can be sent out instantly and corresponding ACKs can be received simultaneously if no loss). The receiving window is set to be infinite large and the slow start threshold (ssthresh) is set to be 16 MSS initially. Answer the following questions and give brief reasoning.

(a) Assume no packet loss till 31ms, then what is the congestion window size?

(b) Assume no packet loss till 61ms, then what is the congestion window size?
Practice Problem (Distance Vector Routing)

Assume that in a network, Router X has two and only two neighbors Y and Z. Y and Z are neighbors as well. Assume that initially X is only aware of the link costs to Y and Z: \( c(X, Y) = 5 \), \( c(X, Z) = 1 \). Give brief reasoning.

(a) X first receives a DV update from Y. The content is \( DV(Y, X) = 5 \), \( DV(Y, Y) = 0 \), \( DV(Y, Z) = 2 \), \( DV(Y, O) = 10 \).

X will update its DV to
\[
DV(X, X) = 0, \quad DV(X, Y) = \_\_\_, \quad DV(X, Z) = \_\_\_, \quad DV(X, O) = \_\_\_
\]

(b) X THEN receives a DV update from Z. The content is \( DV(Z, X) = 1 \), \( DV(Z, Y) = 2 \), \( DV(Z, Z) = 0 \), \( DV(Z, O) = 12 \).

X will update its DV to
\[
DV(X, X) = 0, \quad DV(X, Y) = \_\_\_, \quad DV(X, Z) = \_\_\_, \quad DV(X, O) = \_\_\_
\]