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

Two key network-layer functions
- forwarding: move packets from router’s input to appropriate router output
- routing: determine route taken by packets from source to dest.
  - routing algorithms

Interplay between routing and forwarding
- routing algorithm determines end-end path through network
- forwarding table determines local forwarding at this router
- value in arriving packet’s header
- routing algorithm determines end-end path through network
- forwarding table determines local forwarding at this router
**Connection setup**

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

**Network service model**

Q: What service model for “channel” transporting datagrams from sender to receiver?

- example services for individual datagrams:
  - guaranteed delivery
  - guaranteed delivery with less than 40 msec delay
- example services for a flow of datagrams:
  - in-order datagram delivery
  - guaranteed minimum bandwidth to flow
  - restrictions on changes in inter-packet spacing

**Network layer service models:***

<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</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no congestion</td>
</tr>
<tr>
<td>ATM</td>
<td>VBR</td>
<td>guaranteed</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no congestion</td>
</tr>
<tr>
<td>ATM</td>
<td>ABR</td>
<td>guaranteed minimum</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>ATM</td>
<td>UBR</td>
<td>none</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></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 connecton-oriented / connectionless transport-layer services, but:
  - **service**: host-to-host
  - **no choice**: network provides one or the other
  - **implementation**: in network core

**Virtual circuits**

“source-to-dest path behaves much like telephone circuit”

- 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**

<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>4</td>
<td>97</td>
<td>3</td>
<td>87</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 network-level concept of “connection”
  - Packets forwarded using destination host address

**Datagram forwarding table**

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

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*** 00000000</td>
<td>0</td>
</tr>
<tr>
<td>11001000 00010111 00010100 00000000</td>
<td>1</td>
</tr>
<tr>
<td>11001000 00010111 00010010 00000000</td>
<td>2</td>
</tr>
<tr>
<td>otherwise</td>
<td>3</td>
</tr>
</tbody>
</table>

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.
- 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

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

Router architecture overview

two key router functions:
- run routing algorithms/protocol (RIP, OSPF, BGP)
- forwarding datagrams from incoming to outgoing link

Switching fabrics

- transfer packet from input buffer to appropriate output buffer
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three types of switching fabrics
  - memory
  - bus
  - crossbar
Switching via memory

first generation routers:
- traditional computers with switching under direct control of CPU
- packet copied to system’s memory
- speed limited by memory bandwidth (2 bus crossings per datagram)

Switching via a bus

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

Switching via interconnection network

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

Output ports

- buffering required when datagrams arrive from fabric faster than the transmission rate
- scheduling discipline chooses among queued datagrams for transmission

Output port queueing

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

How much buffering?

- RFC 3439 rule of thumb: average buffering equal to "typical" RTT (say 250 msec) times link capacity C
  - e.g., C = 10 Gbps link: 2.5 Gbit buffer
- recent recommendation: with N flows, buffering equal to
  \[
  \text{buffering} = \frac{\text{RTT} \cdot C}{N}
  \]
**Input port queuing**

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

**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

**The Internet network layer**

Host, router network layer functions:

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

**IP datagram format**

- IP protocol version number / header length (bytes)
- ‘type’ of data
- Max number remaining hops (decremented at each router)
- Upper layer protocol to deliver payload to
- Data (variable length, typically a TCP or UDP segment)
- Options (if any)

**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
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

IP addressing: introduction

Q: How are interfaces actually connected?
A: We’ll learn about that in chapter 5, 6.

- Wired Ethernet interfaces connected by Ethernet switches
- Wireless WiFi interfaces connected by WiFi base station

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

Subnets

- 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

IP addresses associated with each interface

Subnets

Subnets

Recipe

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

Subnets

How many?
IP addressing: CIDR

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

<table>
<thead>
<tr>
<th>subnet part</th>
<th>host part</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000</td>
<td>00010111</td>
</tr>
<tr>
<td>00010000</td>
<td>00000000</td>
</tr>
<tr>
<td>200.23.16.0/23</td>
<td></td>
</tr>
</tbody>
</table>

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 a 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: 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)
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 (desc: 00:01:00:00:00:00) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

DHCP: Wireshark output (home LAN)

Message type: Boot Request (2)
Hardware type: Ethernet
Hardware address length: 6
Transaction ID: 00000000
Seconds elapsed: 3
Client IP address: 0.0.0.0
Bootfile name not given
Client Identifier: 00000000
Server host name not given
Client MAC address: Wistron_23:68:8a
Relay agent IP address: 0.0.0.0
Next server IP address: 0.0.0.0
Your (client) IP address: 0.0.0.0
Client ID: 00000000
Hop count: 0
Hops: 0
Hardware address length: 6
Hardware type: Ethernet
Message type: DHCP Request
Length: 11
Value: 010016D323688A
DHCP: example
DHCP: Wireshark
DHCP: example

IP addresses: how to get one?

Q: how does network get subnet part of IP addr?
A: gets allocated portion of its provider ISP's address space

ISPs-R-Us

<table>
<thead>
<tr>
<th>ISP's block</th>
<th>11001000</th>
<th>00010111</th>
<th>00010000</th>
<th>00000000</th>
<th>200.23.16.20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization 0</td>
<td>11001000</td>
<td>00010111</td>
<td>00010000</td>
<td>00000000</td>
<td>200.23.16.0/23</td>
</tr>
<tr>
<td>Organization 1</td>
<td>11001000</td>
<td>00010111</td>
<td>00110101</td>
<td>00000000</td>
<td>200.23.18.0/23</td>
</tr>
<tr>
<td>Organization 2</td>
<td>11001000</td>
<td>00010111</td>
<td>00110101</td>
<td>00000000</td>
<td>200.23.20.0/23</td>
</tr>
<tr>
<td>Organization 7</td>
<td>11001000</td>
<td>00010111</td>
<td>00111110</td>
<td>00000000</td>
<td>200.23.30.0/23</td>
</tr>
</tbody>
</table>

Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:

<table>
<thead>
<tr>
<th>Organization 0</th>
<th>200.23.16.0/23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization 1</td>
<td>200.23.16.0/23</td>
</tr>
<tr>
<td>Organization 2</td>
<td>200.23.16.0/23</td>
</tr>
<tr>
<td>Organization 7</td>
<td>200.23.30.0/23</td>
</tr>
<tr>
<td>ISPs-R-Us</td>
<td>200.23.16.0/23</td>
</tr>
<tr>
<td>Fly-By-Night-ISP</td>
<td>200.23.16.0/23</td>
</tr>
<tr>
<td>Internet</td>
<td></td>
</tr>
</tbody>
</table>

Hierarchical addressing: more specific routes

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

<table>
<thead>
<tr>
<th>ISP's block</th>
<th>200.23.16.0/23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization 0</td>
<td>200.23.16.0/23</td>
</tr>
<tr>
<td>Organization 2</td>
<td>200.23.16.0/23</td>
</tr>
<tr>
<td>Organization 7</td>
<td>200.23.30.0/23</td>
</tr>
<tr>
<td>Organization 1</td>
<td>200.23.18.0/23</td>
</tr>
<tr>
<td>ISPs-R-Us</td>
<td>200.23.16.0/23</td>
</tr>
<tr>
<td>Fly-By-Night-ISP</td>
<td>200.23.16.0/23</td>
</tr>
<tr>
<td>Internet</td>
<td></td>
</tr>
</tbody>
</table>
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

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)

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

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

NAT is controversial:
  routers should only process up to layer 3
  violates end-to-end argument
  NAT possibility must be taken into account by app designers, e.g., P2P applications
  address shortage should instead be solved by IPv6
**Network Layer 4-61**

*Network Layer 4-62*

**NAT traversal problem**

- 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 2500

- Solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol
  - Allows NATed host to:
    - learn public IP address
    - add/remove port mappings (with lease times)
  - i.e., automate static NAT port map configuration

**Network Layer 4-63**

**NAT traversal problem**

- Solution 3: relaying (used in Skype)
  - NATed client establishes connection to relay
  - external client connects to relay
  - relay bridges packets between to connections
  - 1. connection to relay initiated by NATed host
  - 2. connection to relay initiated by client
  - 3. relaying established

**Network Layer 4-64**

**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

**Network Layer 4-65**

**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>0</td>
<td>1</td>
<td>dest. network unreachable</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>dest host unreachable</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>dest protocol unreachable</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>dest port unreachable</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>dest network unknown</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>source quench (congestion control - not used)</td>
</tr>
</tbody>
</table>

**Network Layer 4-66**

**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
  - 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

**Network Layer 4-67**

**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
- Solution 1: statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 2500

**Network Layer 4-68**

**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
IPv6: motivation

- **Initial motivation:** 32-bit address space soon to be completely allocated.
- **Additional motivation:**
  - Header format helps speed processing/forwarding
  - Header changes to facilitate QoS

IPv6 datagram format:
- Fixed-length 40 byte header
- No fragmentation allowed
- **Priority:** identify priority among datagrams in flow
- **Flow Label:** identify datagrams in same “flow.” (concept of “flow” not well defined).
- **Next Header:** Identify upper layer protocol for data

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

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:**

<table>
<thead>
<tr>
<th>IPv4 tunnel connecting IPv6 router</th>
</tr>
</thead>
</table>

**Physical view:**

A IPv4

B IPv4 to C IPv4

C IPv4 to D IPv4

D IPv4 to E IPv4

E IPv4 to F IPv4

IPv6

IPv6

IPv6
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

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,y), (w,y), (w,z), (x,z) \}
aside: graph abstraction is useful in other network contexts, e.g., 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

key question: what is the least-cost path between u and z?

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

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
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:
   \( N' = \{ u \} \)

2. for all nodes \( v \)
   - if \( v \) adjacent to \( u \)
     \( D(v) = c(u,v) \)
   - else \( D(v) = \infty \)

3. Loop:
   - find \( w \) not in \( N' \) such that \( D(w) \) is a minimum
   - add \( w \) to \( N' \)
   - update \( D(v) \) for all \( v \) adjacent to \( w \) and not in \( N' \):
     \( D(v) = \min(D(v), D(w) + c(w,v)) \)

   /* new cost to v is either old cost to v or known shortest path cost to w plus cost from w to v */

   until all nodes in \( N' \)

Dijkstra's algorithm: example

\[
\begin{align*}
\text{Step} & \quad N' & \quad D(v), p(v) \\
0 & \quad \{ u \} & \quad \begin{cases} 
7, u \ \\ 5, u \ \\
\infty \end{cases} \\
1 & \quad \{ u, v \} & \quad \begin{cases} 
11, u \ \\ 14, x \end{cases} \\
2 & \quad \{ u, v, w \} & \quad \begin{cases} 
10, v \ \\
14, x \end{cases} \\
3 & \quad \{ u, v, w, y \} & \quad \begin{cases} 
11, w \ \\
14, x \end{cases} \\
4 & \quad \{ u, v, w, x \} & \quad \begin{cases} 
15, x \end{cases} \\
5 & \quad \{ u, v, w, y, z \} & \quad \begin{cases} 
15, w \ \\
14, x \ \\
14, x \end{cases}
\end{align*}
\]

notes:
- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)

Dijkstra's algorithm: another example

\[
\begin{align*}
\text{Step} & \quad N' & \quad D(v), p(v) \\
0 & \quad \{ u \} & \quad \begin{cases} 
2, u \ \\
5, u \ \\
\infty \end{cases} \\
1 & \quad \{ u, x \} & \quad \begin{cases} 
2, u \ \\
4, x \ \\
2, u \ \\
4, x \ \\
\infty \end{cases} \\
2 & \quad \{ u, x, y \} & \quad \begin{cases} 
3, y \ \\
3, y \ \\
4, y \ \\
4, y \ \\
4, y \end{cases} \\
3 & \quad \{ u, x, y, w \} & \quad \begin{cases} 
3, y \ \\
4, y \ \\
4, y \ \\
4, y \ \\
4, y \ \\
4, y \ \\
\infty \end{cases} \\
4 & \quad \{ u, x, y, w, z \} & \quad \begin{cases} 
4, y \ \\
4, y \ \\
4, y \ \\
4, y \ \\
4, y \ \\
4, y \ \\
4, y \ \\
\infty \end{cases}
\end{align*}
\]

Dijkstra's algorithm: example (2)

resulting shortest-path tree from u:

resulting forwarding table in u:

Dijkstra's algorithm, discussion

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

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

initially given these costs, find new routing... resulting in new costs
given these costs, find new routing... resulting in new costs
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

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let
d_x(y) := cost of least-cost path from x to y
then
d_x(y) = \min \{ c(x,v) + d_v(y) \}
    \quad \text{cost from neighbor v to destination y}
    \quad \text{cost to neighbor v}
    \quad \text{min taken over all neighbors v of x}

Bellman-Ford example

clearly, d_v(z) = 5, d_x(z) = 3, d_w(z) = 3

B-F equation says:
d_x(y) = \min \{ c(u,v) + d_v(z),
          c(u,x) + d_x(z),
          c(u,w) + d_w(z) \}
= \min \{ 2 + 5,
        1 + 3,
        5 + 3 \} = 4

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

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:
  \[ D_x(y) \leftarrow \min \{ c(x,v) + D_v(y) \} \text{ for each node } y \in N \]
- 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

iterative, asynchronous:
each local iteration caused by:
- wait for (change in local link cost or msg from neighbor)

distributed:
each node:
- recompute estimates
- if DV to any dest has changed, notify neighbors
Distance vector: link cost changes

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

"Good news travels fast"

$t_0$: y detects link-cost change, updates its DV, informs its neighbors.

$t_1$: z receives update from y, updates its table, computes new least cost to x, sends its neighbors its DV.

$t_2$: y receives z’s update, updates its distance table. y’s least costs do not change, so y does not send a message to z.

Comparison of LS and DV algorithms

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

Speed of convergence
- LS: O(n^2) algorithm requires O(nE) msgs • may have oscillations
- DV: convergence time varies • may be routing loops • count-to-infinity problem

Robustness: what happens if router malfunctions?

LS:
- Node can advertise incorrect link cost
- Each node computes only its own table

DV:
- DV node can advertise incorrect path cost
- Each node’s table used by others
- Error propagate thru network

Chapter 4: 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

Example: setting forwarding table in router 1d
- suppose AS1 learns (via inter-AS protocol) that subnet x is 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!

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: choosing among multiple ASes

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

Chapter 4: 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

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)

RIP: example

<table>
<thead>
<tr>
<th>Subnet</th>
<th>Next Router</th>
<th># Hops to Dest</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w )</td>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>( y )</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>( x )</td>
<td>--</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Routing table in router D
RIP: link failure, recovery
if no advertisement heard after 180 sec -->
neighbor/link declared dead
- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly (?) propagates to entire net
- poison reverse used to prevent ping-pong loops (infinite distance = 16 hops)

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

OSPF (Open Shortest Path First)
- "open": publicly available
- uses link state algorithm
- 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

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 ToS; high for real time ToS)
- integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- hierarchical OSPF in large domains.

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

- **BGP** (Border Gateway Protocol): the de facto 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”

### BGP basics

- **BGP session**: two BGP routers (“peers”) exchange BGP messages:
  - advertising paths to different destination network prefixes (“path vector” protocol)
  - exchanged over semi-permanent TCP connections
- when AS3 advertises a prefix to AS1:
  - AS3 promises it will forward datagrams towards that prefix
  - AS3 can aggregate prefixes in its advertisement

### BGP basics: distributing path information

- 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)
- gateway router receiving route advertisement uses **import policy** to accept/decline
  - e.g., never route through AS x
  - **policy-based** routing

### BGP route selection

- router may learn about more than 1 route to destination AS, selects route based on:
  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 between peers over TCP connection
- BGP messages:
  - **OPEN**: opens TCP connection to peer and authenticates sender
  - **UPDATE**: advertises new path (or withdraws old)
  - **KEEPALIVE**: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - **NOTIFICATION**: reports errors in previous msg; also used to close connection
**BGP routing policy**

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

---

**Why different Intra-, Inter-AS routing?**

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

**Scale:**
- Hierarchical routing saves table size, reduced update traffic.

**Performance:**
- Intra-AS: Can focus on performance.
- Inter-AS: Policy may dominate over performance.

---

**Broadcast routing**

- Deliver packets from source to all other nodes.
- Source duplication is inefficient.

**In-network duplication**

- Flooding: When node receives broadcast packet, sends copy to all neighbors.
  - Problems: Cycles & broadcast storm.
- Controlled flooding: Node only broadcasts pkt if it hasn’t broadcast same packet before.
  - Node keeps track of packet ids already broadcasted.
  - Or reverse path forwarding (RPF): Only forward packet if it arrived on shortest path between node and source.
- Spanning tree:
  - No redundant packets received by any node.

---

**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
Spanning tree

- first construct a spanning tree
- nodes then forward/make copies only along spanning tree

(a) broadcast initiated at A  
(b) broadcast initiated at D

Spanning tree: creation

- center node
- each node sends unicast join message to center node
- message forwarded until it arrives at a node already belonging to spanning tree

(a) stepwise construction of spanning tree (center: E)  
(b) constructed spanning tree

Multicast routing: problem statement

**goal:** find a tree (or trees) connecting routers having local mcast group members

- tree: not all paths between routers used
- shared-tree: same tree used by all group members
- source-based: different tree from each sender to receivers

shared tree  
source-based trees

Legend:

- group member
- not group member
- router with a group member
- router without a group member

Approaches for building mcast trees

**approaches:**

- **source-based tree:** one tree per source
  - shortest path trees
  - reverse path forwarding
- **group-shared tree:** group uses one tree
  - minimal spanning (Steiner)
  - center-based trees

...we first look at basic approaches, then specific protocols adopting these approaches

Shortest path tree

- mcast forwarding tree: tree of shortest path routes from source to all receivers
  - Dijkstra’s algorithm

Reverse path forwarding

- rely on router’s knowledge of unicast shortest path from it to sender
- each router has simple forwarding behavior:

```python
if (mcast datagram received on incoming link on shortest path back to center)
then flood datagram onto all outgoing links
else ignore datagram
```
Reverse path forwarding: example

- result is a source-specific reverse SPT
  - may be a bad choice with asymmetric links

Reverse path forwarding: pruning

- forwarding tree contains subtrees with no mcast group members
  - no need to forward datagrams down subtree
  - “prune” msgs sent upstream by router with no downstream group members

Shared-tree: steiner tree

- steiner tree: minimum cost tree connecting all routers with attached group members
- problem is NP-complete
- excellent heuristics exists
- not used in practice:
  - computational complexity
  - information about entire network needed
  - monolithic: rerun whenever a router needs to join/leave

Center-based trees

- single delivery tree shared by all
- one router identified as “center” of tree
- to join:
  - edge router sends unicast join-msg addressed to center router
  - join-msg “processed” by intermediate routers and forwarded towards center
  - join-msg either hits existing tree branch for this center, or arrives at center
  - path taken by join-msg becomes new branch of tree for this router

Center-based trees: example

- suppose R6 chosen as center:

Internet Multicasting Routing: DVMRP

- DVMRP: distance vector multicast routing protocol, RFC1075
- flood and prune: reverse path forwarding, source-based tree
  - RPF tree based on DVMRP’s own routing tables constructed by communicating DVMRP routers
  - no assumptions about underlying unicast
  - initial datagram to mcast group flooded everywhere via RPF
  - routers not wanting group: send upstream prune msgs
DVMRP: continued…

- **soft state**: DVMRP router periodically (1 min.) “forgets” branches are pruned:
  - multicast data again flows down unpruned branch
  - downstream router: re-prune or else continue to receive data
- routers can quickly regraft to tree
  - following IGMP join at leaf
- odds and ends
  - commonly implemented in commercial router

Tunneling

Q: how to connect “islands” of multicast routers in a “sea” of unicast routers?

- multicast data encapsulated inside “normal” (non-multicast-addressed) datagram
- normal IP datagram sent thru “tunnel” via regular IP unicast to receiving multicast router (recall IPv6 inside IPv4 tunneling)
- receiving multicast router unencapsulates to get multicast datagram

PIM: Protocol Independent Multicast

- not dependent on any specific underlying unicast routing algorithm (works with all)
- two different multicast distribution scenarios:

  - **dense**:
    - group members densely packed, in “close” proximity.
    - bandwidth more plentiful

  - **sparse**:
    - # networks with group members small wrt # interconnected networks
    - group members “widely dispersed”
    - bandwidth not plentiful

Consequences of sparse-dense dichotomy:

- **dense**:
  - group membership by routers assumed until routers explicitly prune
  - data-driven construction on multicast tree (e.g., RPF)
  - bandwidth and non-group-router processing profligate

- **sparse**:
  - no membership until routers explicitly join
  - receiver-driven construction of multicast tree (e.g., center-based)
  - bandwidth and non-group-router processing conservative

PIM- dense mode

- flood-and-prune RPF: similar to DVMRP but…
- underlying unicast protocol provides RPF info for incoming datagram
- less complicated (less efficient) downstream flood than DVMRP reduces reliance on underlying routing algorithm
- has protocol mechanism for router to detect it is a leaf-node router

PIM- sparse mode

- center-based approach
- router sends join msg to rendezvous point (RP)
  - intermediate routers update state and forward join
  - after joining via RP, router can switch to source-specific tree
    - increased performance: less concentration, shorter paths
PIM - sparse mode

sender(s):
- unicasts data to RP, which distributes down RP-rooted tree
- RP can extend mcast tree upstream to source
- RP can send stop msg if no attached receivers
  - “no one is listening!”

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
4.7 broadcast and multicast routing

- 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