4장. 네트워크 계층

Computer Networking:
***A Top Down Approach Featuring the Internet*,**
3rd edition. Jim Kurose, Keith Ross

컴퓨터네트워크(2005-2학기)
박영호

Chapter 4: Network Layer

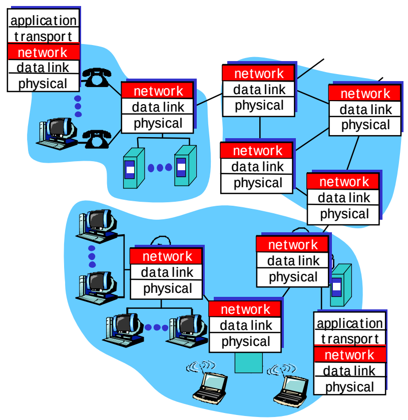
Chapter goals:

- understand principles behind network layer services:
  - routing (path selection)
  - dealing with scale
  - how a router works
  - advanced topics: IPv6
- instantiation and implementation in the Internet

Chapter 4: Network Layer

Network layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on 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



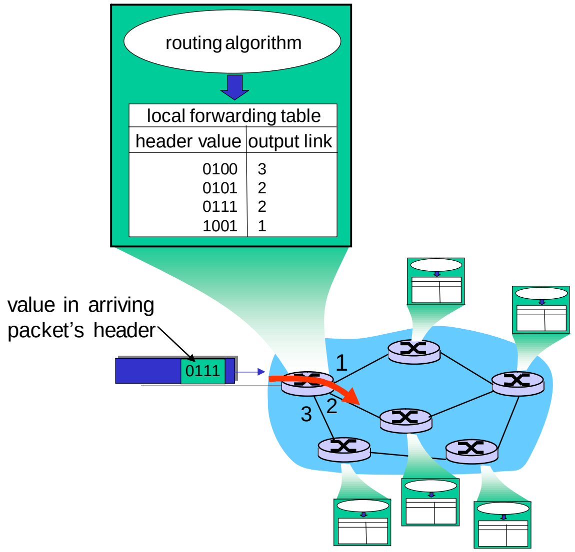
Key Network-Layer Functions

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

analogy:

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

Interplay between routing and forwarding



Connection setup

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

Network service model

Q: What *service model* for "channel" transporting datagrams from sender to receiver?

Example services for individual datagrams:

- guaranteed delivery
- Guaranteed delivery with less than 40 msec delay

Example services for a flow of datagrams:

- In-order datagram delivery
- Guaranteed minimum bandwidth to flow
- Restrictions on changes in inter-packet spacing

Network layer service models:

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

Chapter 4: Network Layer

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

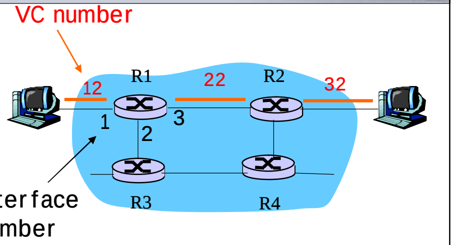
VC implementation

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 a VC number.
- VC number must be changed on each link.
  - New VC number comes from forwarding table

Forwarding table



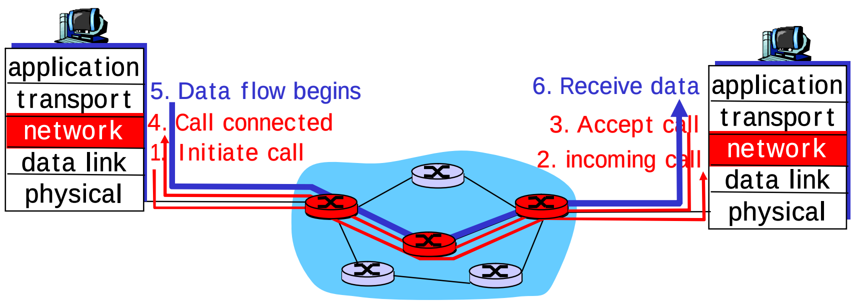
Forwarding table in R1:

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

Routers maintain connection state information!

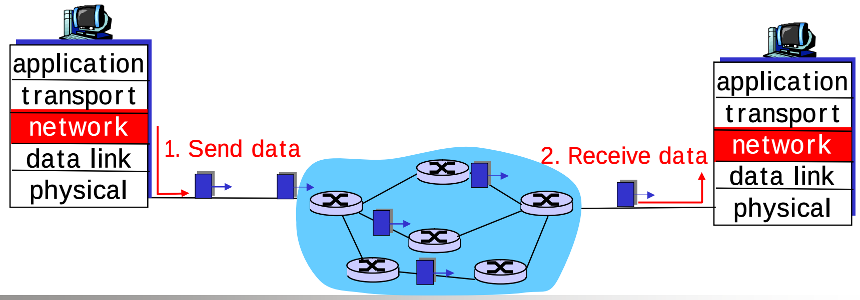
Virtual circuits: signaling protocols

- used to setup, maintain teardown VC
- used in ATM, frame-relay, X.25
- 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
  - packets between same source-dest pair may take different paths



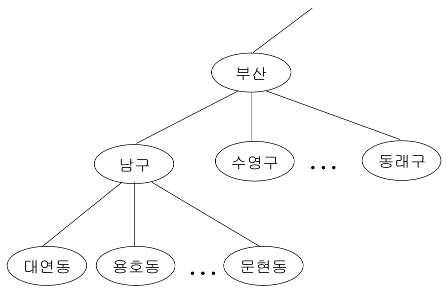
Forwarding table

4 billion
possible entries

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Longest prefix matching

Prefix Match	Link Interface
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3



현재 인터넷 프로토콜(IP)은 클래스 기반의 서브넷(subnet)을 이용하여 계층적으로 분류됨

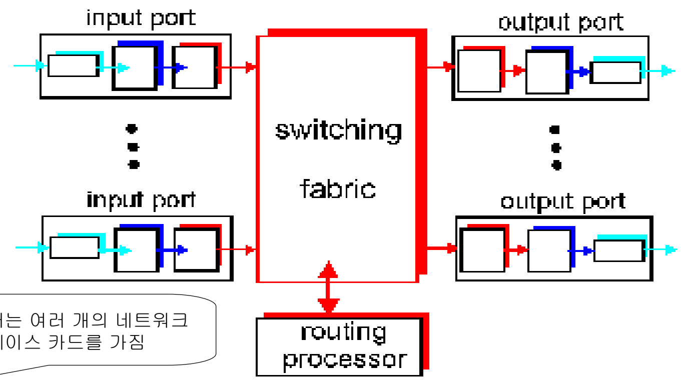
Chapter 4: Network Layer

- 4. 1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What's inside a router
- 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 Broadcast and multicast routing

Router Architecture Overview

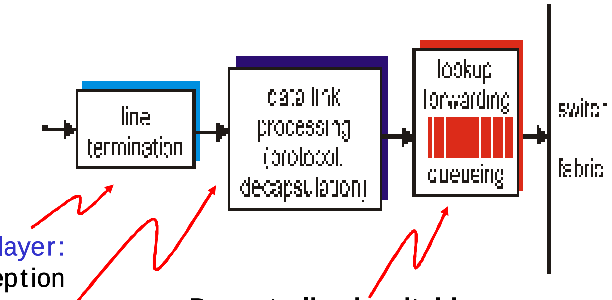
Two key router functions:

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



라우터는 여러 개의 네트워크 인터페이스 카드를 가짐

Input Port Functions



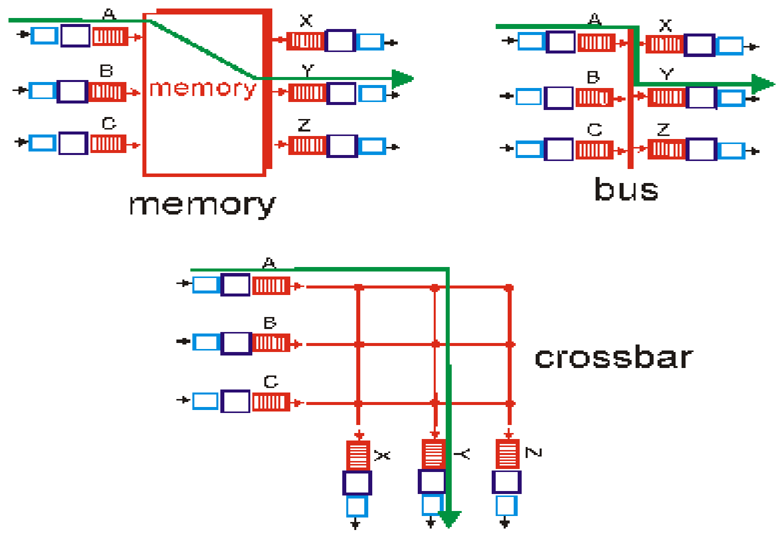
Physical layer:
bit-level reception

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

Decentralized switching:

- given datagram dest., lookup output port using forwarding table in input port memory
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

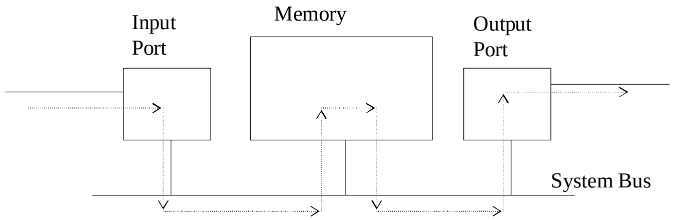
Three types of switching fabrics



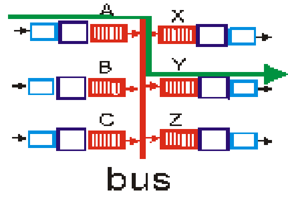
Switching Via Memory

First generation routers:

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

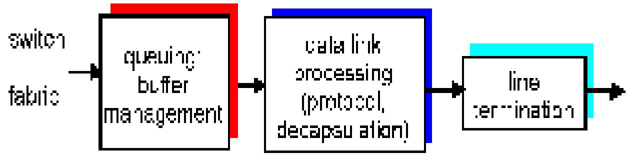


Switching Via a Bus



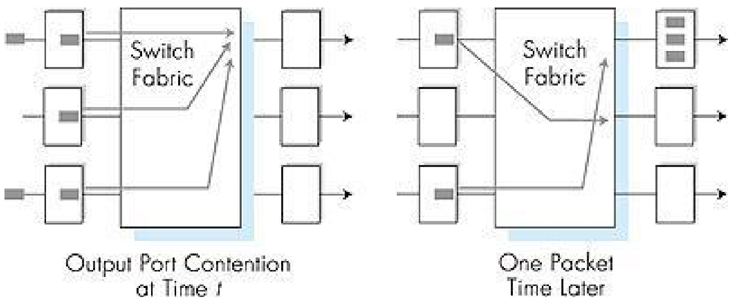
- datagram from input port memory to output port memory via a shared bus
- bus contention: switching speed limited by bus bandwidth
- 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)

Output Ports



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

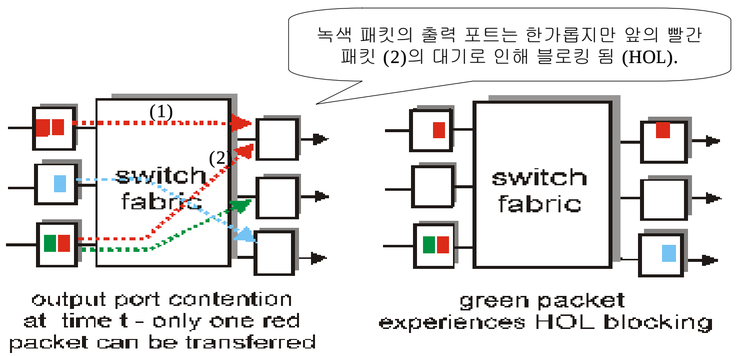
Output port queueing



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

Input Port Queuing

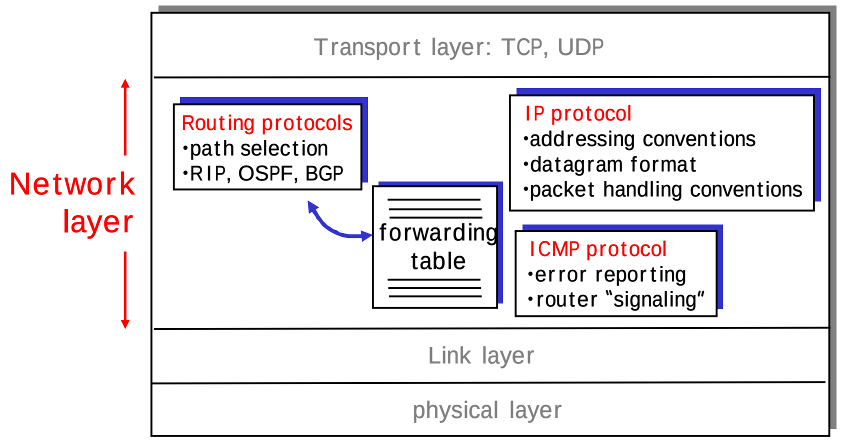
- Fabric slower than input ports combined -> queueing may occur at input queues
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward
- *queueing delay and loss due to input buffer overflow!*



Chapter 4: Network Layer

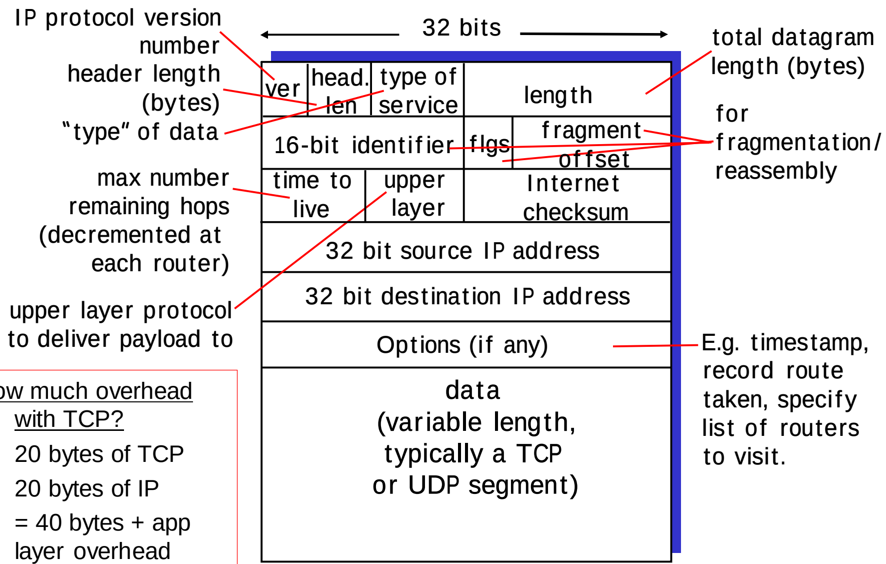
The Internet Network layer

Host, router network layer functions:



Chapter 4: Network Layer

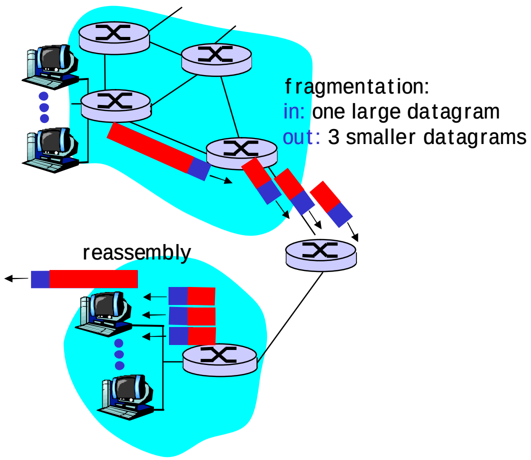
IP datagram format



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

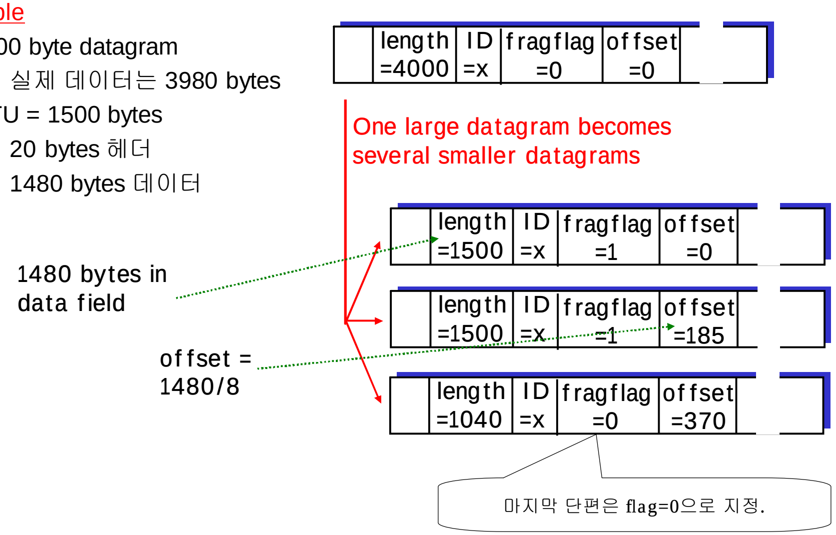
IP Fragmentation & Reassembly

- network links have MTU (max.transfer size) - largest possible link-level frame.
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related fragments



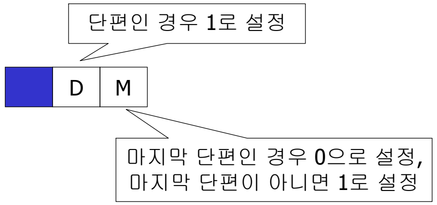
IP Fragmentation and Reassembly

Example
- 4000 byte datagram
  - 실제 데이터는 3980 bytes
- MTU = 1500 bytes
  - 20 bytes 헤더
  - 1480 bytes 데이터



단편화 재조립(Reassembly)

- 단편들을 원래의 데이터그램으로 합치는 것
- 단편의 재조립은 최종 수신 호스트에서 담당
  - 라우터에서 유지해야 하는 상태정보를 줄임
  - 각 단편들은 서로 다른 경로로 전송될 수 있음
    - 모든 단편들이 동일한 라우터를 경유할 필요가 없음
- 마지막 단편은 헤더의 more fragment 플래그에 0 포함



데이터그램 식별

- 동일한 데이터그램의 단편들은 헤더에 동일한 식별자(identification)를 가짐
  - 각 단편은 서로 다른 경로로 전송
  - IP는 전달의 순서를 보증하지 않음
- 수신측 호스트는 IP 헤더의 식별자를 이용하여 단편들의 재조립에 이용
  - 동일한 식별자를 가지는 단편들을 offset에 따라 재조립

단편 손실

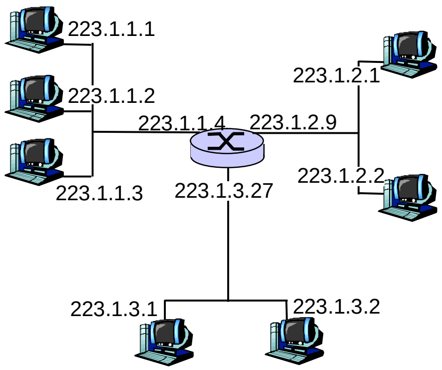
- IP는 전송에 대한 신뢰성을 보증하지 않음
  - 패킷의 손실이 발생 가능
- 단편들의 재조립을 위해 모든 단편이 도착할 때까지 메모리에 저장
  - 메모리 낭비를 방지하기 위해 일정 시간 타이머 동작
  - 타이머 시간내에 모든 단편들이 도착하지 않으면 단편들을 삭제
    - 전부 또는 전무 (All-or-nothing)
  - 단편화된 데이터그램의 재전송을 요청하지 않음
    - 데이터그램이 이전과 동일하게 단편화 된다는 보장이 없으므로

Chapter 4: Network Layer

- 4. 1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What's inside a router
- 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 Broadcast and multicast routing

IP Addressing: introduction

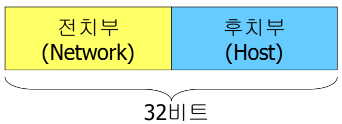
- IP address: 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host may have multiple interfaces
  - IP addresses associated with each interface



223.1.1.1 = $\underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$

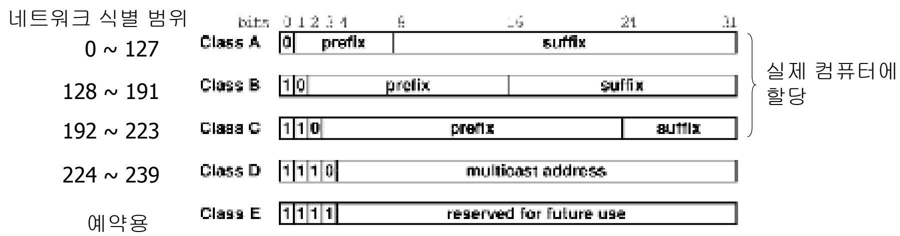
주소 계층

- 32비트 주소를 전치부(prefix)와 후치부(suffix)로 구분
  - 전치부
    - 네트워크를 식별
    - 전역적인 조정이 필요
      - 인터넷에서 유일한 네트워크 식별 번호를 할당
  - 후치부
    - 해당 네트워크상의 개별적인 컴퓨터를 식별
    - 지역적으로 할당
      - 주어진 물리적 네트워크 내에서 유일한 주소를 할당
      - 인터넷상에서 유일한 주소



IP 주소의 클래스

- 전치부, 후치부 길이에 따라 네트워크 규모 결정
  - 긴 후치부 길이는
    - 한 네트워크에 많은 컴퓨터를 할당
    - 사용할 수 있는 네트워크 주소 범위는 감소
- IP 주소는 클래스별로 주소 범위의 비트 수를 구분
  - 자가 식별(self-identifying)



점있는 십진 표기와 주소 공간

- IP 주소는 8비트 블록 4개로 구성 – 32비트
  - 판독성을 위해 4개의 십진값과 점(dot)으로 표기
  - 각 십진값은 0~255 범위 내

32-bit Binary Number	Equivalent Dotted Decimal
10000001 00110100 00000110 00000000	129 . 52 . 6 . 0
11000000 00000101 00110000 00000011	192 . 5 . 48 . 3
00001010 00000010 00000000 00100101	10 . 2 . 0 . 37
10000000 00001010 00000010 00000011	128 . 10 . 2 . 3
10000000 10000000 11111111 00000000	128 . 128 . 255 . 0

Class	Range of Values
A	0 through 127
B	128 through 191
C	192 through 223
D	224 through 239
E	240 through 255

Address Class	Bits In Prefix	Maximum Number of Networks	Bits In Suffix	Maximum Number Of Hosts Per Network
A	7	128	24	16777216
B	14	16384	16	65536
C	21	2097152	8	256

주소 마스크(Mask)

- 라우터는 패킷의 목적지를 결정하기 위해 라우팅 표와 IP 주소의 네트워크 값을 비교
- IP 주소의 전치부와 후치부의 경계를 구별하기 위해 마스크 값 사용
  - Net Mask
  - IP 주소와 마스크 값에 대해 bit-wise AND 연산
  - 예) IP 주소 : 203.247.166.178 (C class)
    Mask : 255.255.255.0
    IP 주소 & Mask = 203.247.166.0
    - 네트워크 식별값 203.247.166.0
- 일반적으로 IP 클래스로부터 마스크 값 결정 가능

특수 IP 주소

- 네트워크 주소
  - 전치부만 할당되고 후치부는 모두 0으로 지정
  - 네트워크 참조용이며, 목적지 주소로 사용하지 못함
- 방향적 방송 주소
  - 후치부가 모두 1로 할당
  - 해당 네트워크에서 방송용
    - 해당 네트워크의 모든 컴퓨터가 패킷의 수신지가 됨
- 제한된 방송주소
  - IP 주소가 모두 1로 할당
  - 시스템 시작동안에만 사용
    - 주로 주소 획득에 이용

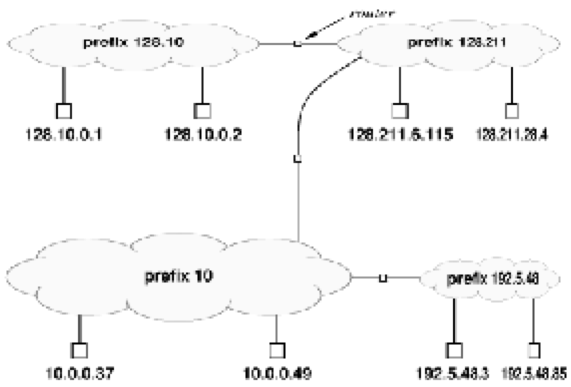
특수 IP 주소

- 컴퓨터 자신 주소
  - IP 주소가 모두 0으로 할당
- local loopback 주소
  - 전치부로 127을 사용
    - 일반적으로 127.0.0.1로 지정
  - 자신이 송신과 수신 컴퓨터로 동작
  - 네트워크 프로그램 테스트용으로 활용

전치부	후치부	주소 유형	목적
모두 0	모두 0	컴퓨터 자신	부트스트랩 동안
네트워크	모두 0	네트워크	네트워크 식별
네트워크	모두 1	방향적 방송	지정 네트워크 방송
모두 1	모두 1	제한된 방송	지역 네트워크 방송
127	임의의 값	loopback	테스트용

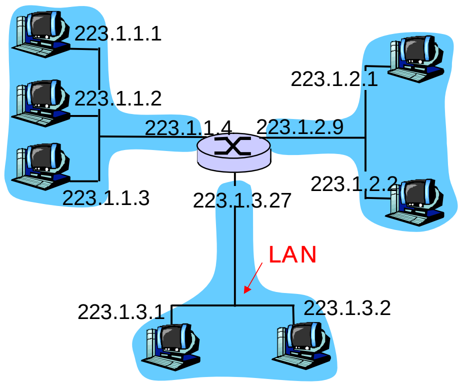
주소 지정 예제

- IP 주소의 네트워크 식별값은 ICANN에서 할당



Subnets

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

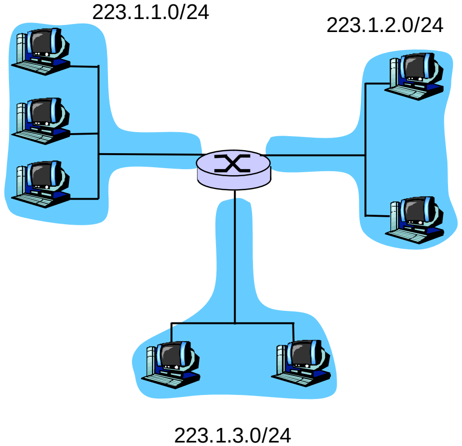


network consisting of 3 subnets

Subnets

Recipe

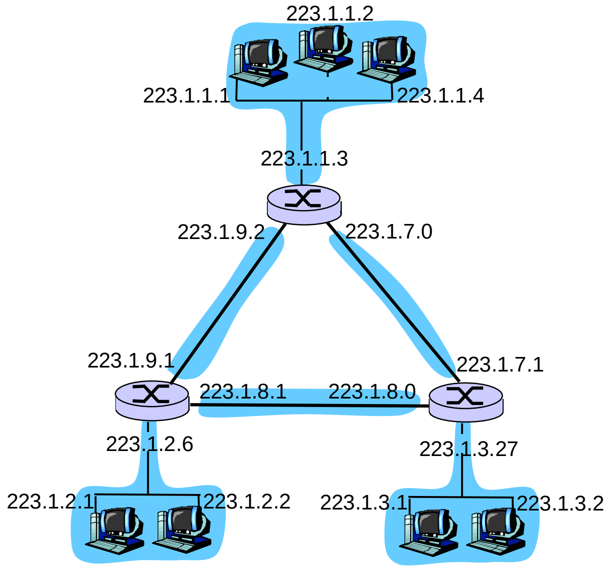
- To determine the subnets, detach each interface from its host or router, creating islands of isolated networks. Each isolated network is called a subnet.



Subnet mask: /24

Subnets

How many?



IP addressing: CIDR

CIDR: Classless InterDomain Routing

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

subnet part | host part

11001000 00010111 00010000 00000000

200.23.16.0/23

IP addresses: how to get one?

Q: How does *host* get IP address?

- hard-coded by system admin in a file
  - Wintel: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

(more in next chapter)

IP addresses: how to get one?

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

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

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

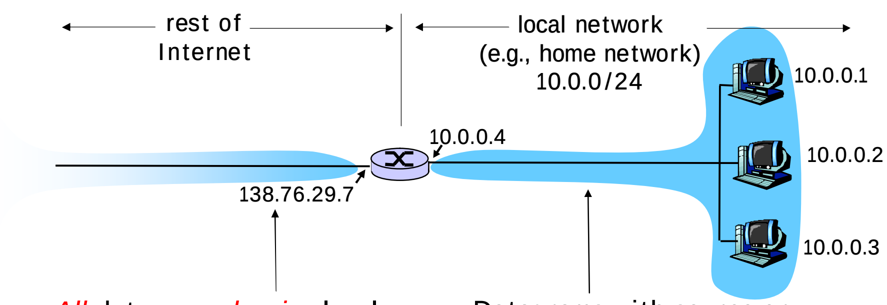
IP addressing: the last word...

Q: How does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers

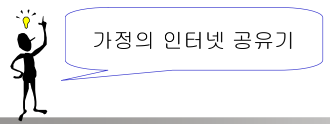
- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

NAT: Network Address Translation



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

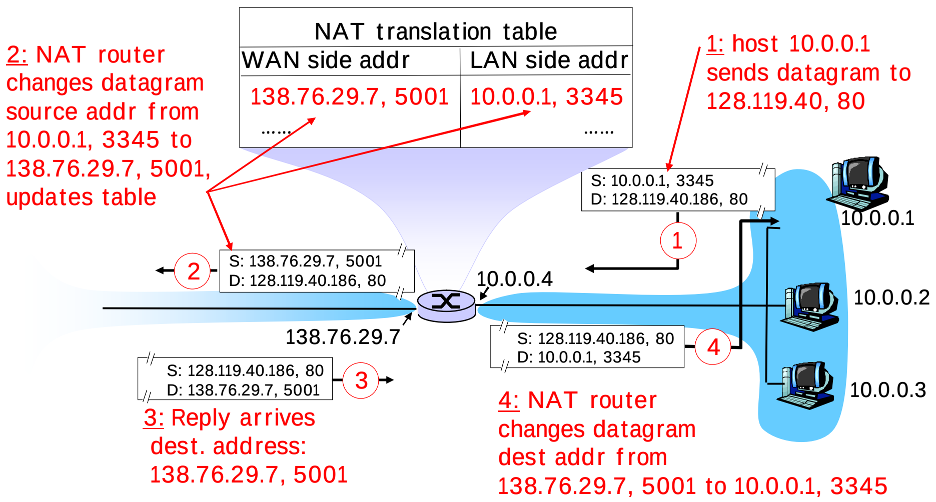
- Motivation: local network uses just one IP address as far as outside word is concerned:
  - no need to be allocated range of addresses from ISP: - just one IP address is used for all devices
  - can change addresses of devices in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - devices inside local net not explicitly addressable, visible by outside world (a security plus).

NAT: Network Address Translation

Implementation: NAT router must:

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

NAT: Network Address Translation



NAT: Network Address Translation

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
  - routers should only process up to layer 3
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, eg, P2P applications
  - address shortage should instead be solved by IPv6

Chapter 4: Network Layer

- 4. 1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What's inside a router
- 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 Broadcast and multicast routing

ICMP: Internet Control Message Protocol

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

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

Traceroute and ICMP

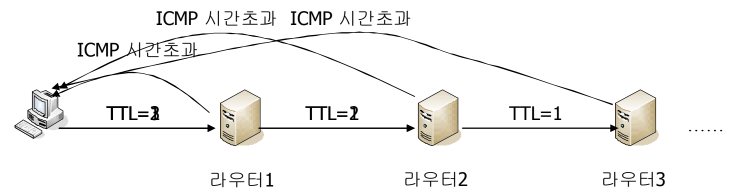
- Source sends series of UDP segments to dest
  - First has TTL =1
  - Second has TTL=2, etc.
  - Unlikely port number
- When nth datagram arrives to nth router:
  - Router discards datagram
  - And sends to source an ICMP message (type 11, code 0)
  - Message includes name of router& IP address
- When ICMP message arrives, source calculates RTT
- Traceroute does this 3 times

Stopping criterion

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

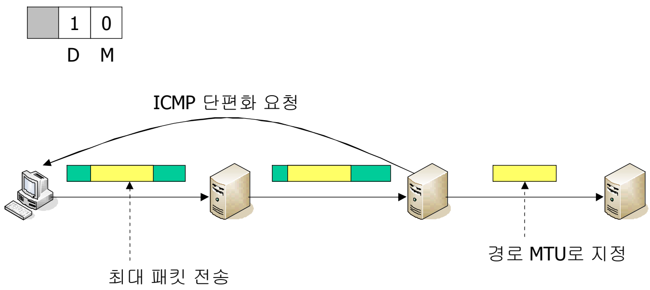
라우터 추적

- ICMP 시간초과 응답에서 해당 라우터의 IP 주소 추출



경로 MTU 발견

- 최대 크기의 패킷을 전송하여 각 라우터의 단편화 요청 메시지로 경로 MTU 결정
  - 라우터가 단편화를 수행하지 않도록 flag 지정

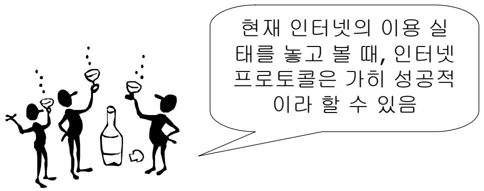


Chapter 4: Network Layer

- 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

인터넷 프로토콜(IP)의 성공

- 인터넷 프로토콜
  - 이기종 네트워크의 연결
  - 균일한 패킷 구조와 패킷 전송 방법을 정의
    - IP 데이터그램
    - 주소지정
    - 경로설정
  - H/W 기술변화와 확장성을 수용
    - 전 세계의 거의 모든 사람들이 인터넷을 이용



변화의 동기

- 제한된 주소공간
  - 현재 IP는 32비트 주소공간을 사용
    - 총 2^{32}개의 주소와 클래스별 주소 할당
    - 향후 네트워크 성장률을 만족하지 못함
- 새로운 인터넷 응용의 필요성
  - 실시간 멀티미디어 전송 서비스
    - 고품질의 서비스를 요구(QoS, Quality of Service)
  - 그룹 통신
    - 동일한 서비스를 여러 사용자에게 제공하기 위함
    - Multicast

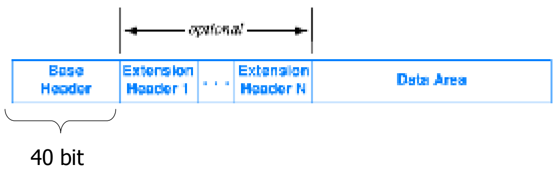
    - 새로운 주소지정과 경로 설정을 요구

IPv6 특징

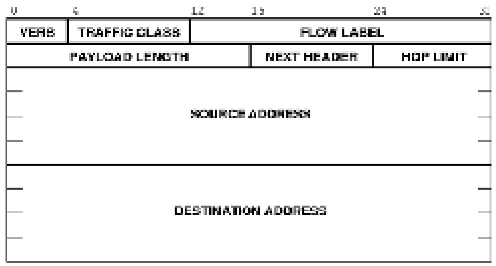
- IPv4의 기본 개념은 유지하면서 상세 항목들을 변경
  - 새로운 주소체계와 데이터그램 형식
    - 128비트 주소공간 (약 3.40 x 10^{38}개)
    - 가변길이의 다중 확장 헤더사용
  - 오디오와 비디오 전송 지원
    - 고품질의 높은 성능을 보장하는 경로 설정
  - 확장 프로토콜
    - 보안 기능의 추가
      - IPSec
    - 새로운 기능의 추가에 대한 유연성 제공

IPv6 데이터그램 구조

- 기본헤더 + 확장헤더(0개 이상) + 자료 영역
  - 40비트 고정길이의 기본헤더
  - 확장헤더는 가변길이



IPv6 기본헤더



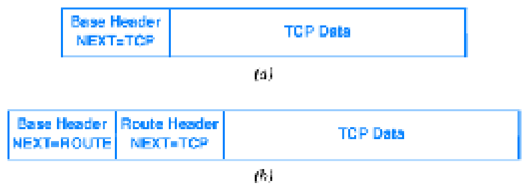
flow(흐름) 란?
동일한 특성을 가지는 패킷의 연속을 말함

VERS	4	IP 버전 : 6
TRAFFIC CLASS	8	패킷의 우선순위
FLOW LEBEL	20	특정 흐름에 대한 경로 인식
PATLOAD LENGTH	16	페이로드 길이를 바이트 단위로
NEXT HEADER	8	다음에 오는 자료의 유형을 명시
HOP LIMIT	8	TTL 값

두 필드를 이용하여 경로에 대한 자원을 미리 예약.
실제 구현은 아직 개발 중…

IPv6 기본헤더

- Next Header는 뒤 따라 오는 자료의 유형을 명시
  - 다른 확장 헤더이거나 TCP 데이터가 됨



IPv6의 다중 헤더 처리

- 표준안은 각 가능한 헤더 유형의 유일한 값을 명시
  - Next Header 필드값으로 이용
- 선택 헤더(Optional header)
  - 가변 길이의 여러 확장헤더를 처리

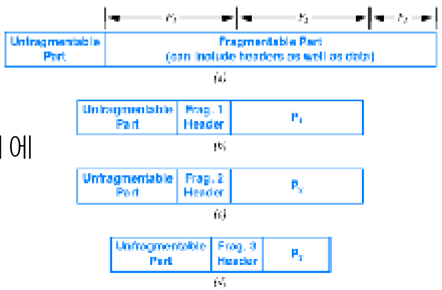
code	Next Header
0	Hop-by-hop option
2	ICMP
6	TCP
17	UDP
43	Source routing
44	Fragmentation
50	Encrypted Security Payload
51	Authentication
59	Null (Not next header)
60	Destination option

단편화, 재조립, 경로 MTU

- IPv6는 단편 확장 헤더를 포함
  - Fragmentation (코드 44)
- 송신 호스트가 단편화를 담당
  - IPv4는 라우터도 단편화
  - 송신 호스트가 목적지까지의 경로 MTU를 발견함.
  - 경로 MTU 발견은 ICMP를 반복적으로 이용.
  - 송신측은 경로 MTU의 크기에 맞게 데이터그램을 구성

경로 MTU (path MTU)란?

소스에서 목적지까지의 최소 MTU를 말함. 경로 MTU를 구하는 과정을 경로 MTU 발견이라 함.



다중 헤더의 목적

- 경제성
  - 기능별 헤더 분할로 인한 공간 절약.
  - 작은 데이터그램은 적은 전송 시간을 요구.
  - cf.) IPv4는 헤더에 모든 기능을 표현.
    - 사용되지 않는 부분에 대한 낭비
- 확장성
  - 새로운 기능의 추가가 용이
    - 새로운 확장 헤더와 NEXT HEADER 유형만 정의
  - 프로토콜 설계에 대한 유연성 제공
  - cf.)IPv4같은 고정된 헤더는 새로운 기능을 추가하기 위해 헤더 전체를 수정해야 함.

IPv6의 주소 지정

- 다중 계층 구조의 주소

Type identifier	Registry identifier	Provider identifier	Subscriber identifier	Subnet identifier	Node identifier

16 24 32 48

권고 비트수 (고정아님)

  - Type ID(3^5-bit) Provider-based address 정의
    - 010 : unicast address
  - Registry ID(5-bit)
    - 11000 : INTERNIC
    - 01000 : RIPNIC
    - 10100 : APNIC
  - Provider ID : ISP 할당 값
  - Subscriber ID : ISP에서 가입자에게 부여하는 값
  - Subnet ID
  - Node ID : 각 컴퓨터에 할당

IPv6의 주소 지정

- 주소 유형
  - 단일전송(Unicast)
    - 단일 컴퓨터에 대응되는 주소
  - 다중전송(Multicast)
    - 컴퓨터들의 집합에 대응되는 주소
    - 여러 컴퓨터에게 동일한 데이터그램의 복사본을 전송
  - 임의전송(Anycast)
    - 공통된 주소 전치부에 속하는 컴퓨터 집합의 대응 주소
    - 클러스터 주소
    - 접근이 용이한 어느 한 컴퓨터에게 전달

IPv6 콜론 16진 주소 표기

- 128비트 주소 표기의 어려움
  - 예) 105.220.136.100.255.2.255.255.0.0.18.128.140.10.255.255
- 16비트씩 콜론(:)으로 구별하여 16진수로 표기
  - 예) 69DC:8864:FFFF:FFFF:0:1280:8C0A:FFFF
- 제로 압축(Zero Compression)
  - 0의 나열들(zero run)을 축약하여 표기
    - 예) FF0C:0:0:0:0:0:0:00B1 -> FF0C::B1

    주소의 나머지 부분은 모두 0을 의미
  - 단, 한번의 zero run에 대해서만 가능
    - 예) FF0C:0:0:0:8864:0:0:B125 -> FF0C::8864::B125 (X)

    0인 부분이 몇 개인지 구분 안됨
- IPv4의 주소를 IPv6의 주소로 매핑
  - 96개의 0비트로 시작
  - IPv6의 하위 32비트 주소는 IPv4의 주소를 포함

요약

- 현재 IP 버전이 성공적이지만 인터넷의 성장으로 32비트 주소공간이 모두 고갈될 것으로 전망.
  - 128비트의 새로운 IP 개발
- IPv6은 현재 IPv4의 기본 개념은 유지하면서 새로운 기술을 개발
  - 확장헤더와 새로운 데이터그램 형식
  - 고품질의 서비스 제공
- 새로운 주소 형식
  - IPv4의 주소도 포함

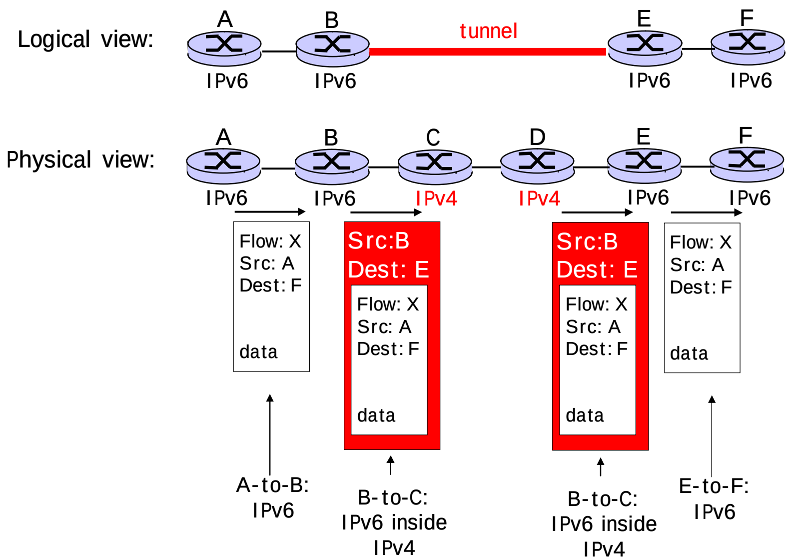
Other Changes from IPv4

- *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

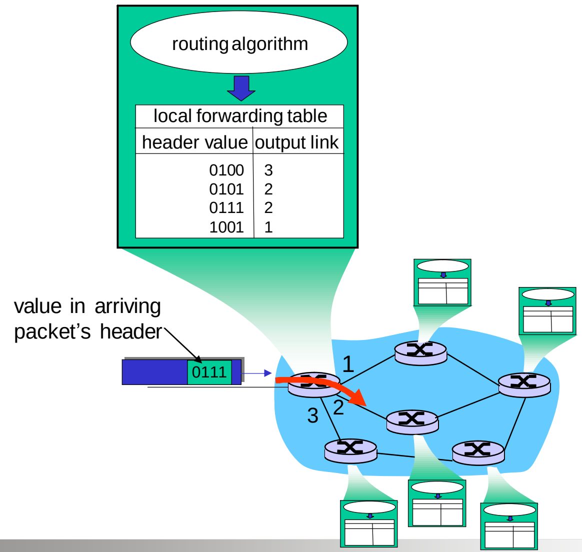
- Not all routers can be upgraded simultaneous
  - no "flag days"
  - How will the network operate with mixed IPv4 and IPv6 routers?
- *Tunneling:* IPv6 carried as payload in IPv4 datagram among IPv4 routers

Tunneling

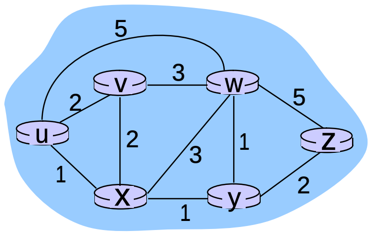


Chapter 4: Network Layer

Interplay between routing and forwarding



Graph abstraction



Graph: G = (N,E)

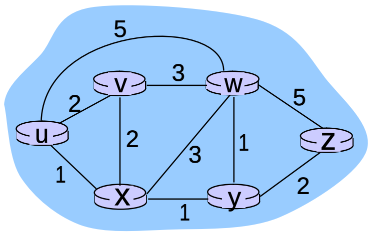
N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where N is set of peers and E is set of TCP connections

Graph abstraction: costs



• c(x,x') = cost of link (x,x')

- e.g., c(w,z) = 5

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

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

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

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

Global or decentralized information?

Global:

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

Decentralized:

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

Static or dynamic?

Static:

- routes change slowly over time

Dynamic:

- routes change more quickly
  - periodic update
  - in response to link cost changes

Chapter 4: Network Layer

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

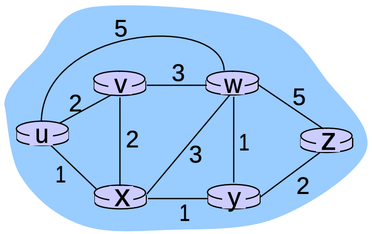
Dijsktra's Algorithm

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

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

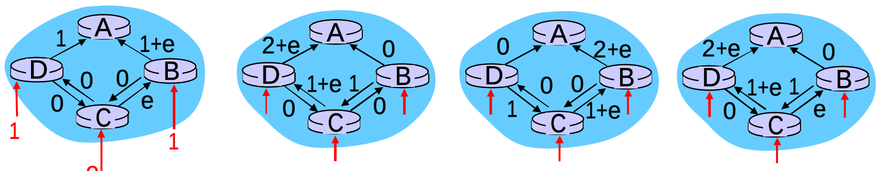
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Dijkstra's algorithm, 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(nlogn)

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



initially ... recompute routing ... recompute ... recompute

Chapter 4: Network Layer

- 4. 1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What's inside a router
- 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 Broadcast and multicast routing

Distance Vector Algorithm (1)

Bellman-Ford Equation (dynamic programming)

Define

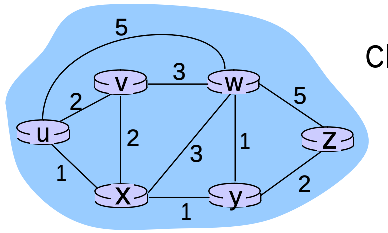
$d_x(y)$:= cost of least-cost path from x to y

Then

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

where min is taken over all neighbors of x

Bellman-Ford example (2)



Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

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

Node that achieves minimum is next hop in shortest path ➜ forwarding table

Distance Vector Algorithm (3)

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

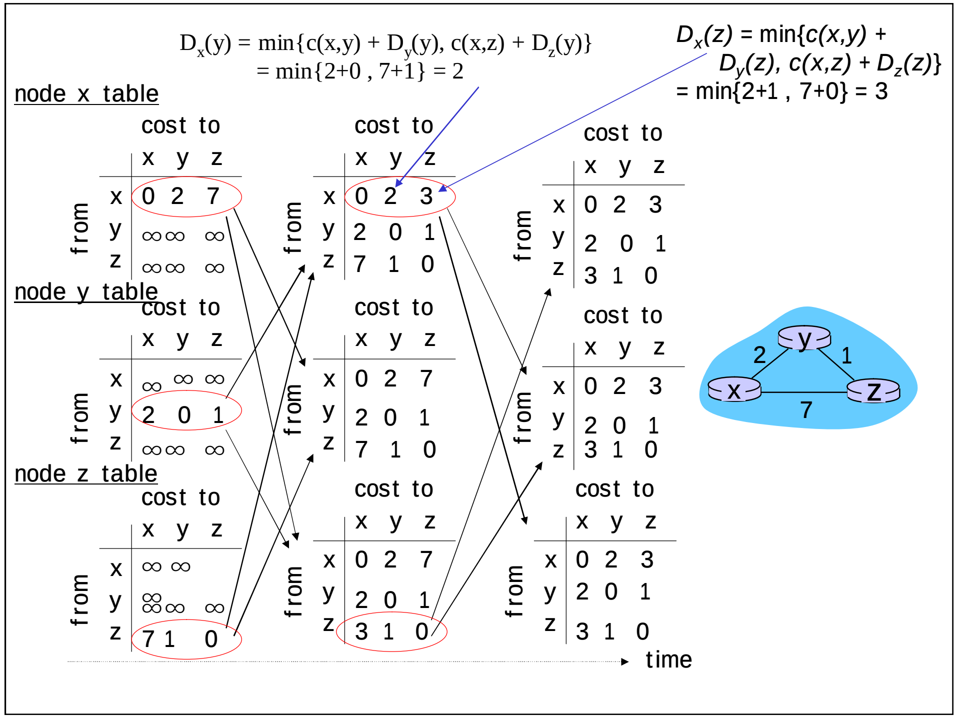
Distance vector algorithm (4)

Basic idea:

- Each node periodically sends its own distance vector estimate to neighbors
- When node a node x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- Under minor, natural conditions, the estimate $D_x(y)$ *converge the actual least cost* $d_x(y)$



Distance Vector Algorithm (5)

Iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

Distributed:

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

Each node:

wait for (change in local link cost of msg from neighbor)

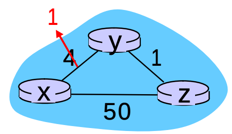
recompute estimates

if DV to any dest has changed, *notify* neighbors

Distance Vector: link cost changes

Link cost changes:

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors



"good news travels fast"

At time t_0, *y* detects the link-cost change, updates its DV, and informs its neighbors.

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

At time t_2, *y* receives *z*'s update and updates its distance table. *y*'s least costs do not change and hence *y* does *not* send any message to *z*.

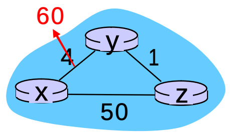
Distance Vector: link cost changes

Link cost changes:

- good news travels fast
- bad news travels slow - "count to infinity" problem!
- 44 iterations before algorithm stabilizes: see text

Poissoned reverse:

- If Z routes through Y to get to X :
  - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- will this completely solve count to infinity problem?



Comparison of LS and DV algorithms

Message complexity

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

Speed of Convergence

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

Robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

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

Chapter 4: Network Layer

Hierarchical Routing

Our routing study thus far - idealization

- all routers identical
- network "flat"

… *not* true in practice

scale: with 200 million destinations:

- can't store all dest's in routing tables!
- routing table exchange would swamp links!

administrative autonomy

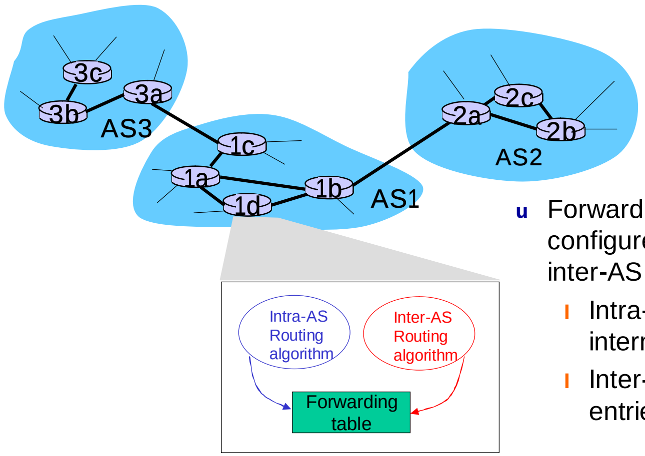
- internet = network of networks
- each network admin may want to control routing in its own network

Hierarchical Routing

- aggregate routers into regions, "autonomous systems" (AS)
- routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

Gateway router
- Direct link to router in another AS

Interconnected ASes



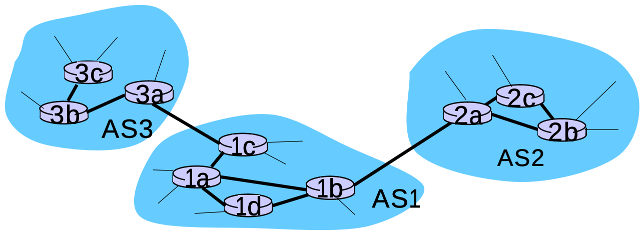
- Forwarding table is configured by both intra- and inter-AS routing algorithm
  - Intra-AS sets entries for internal dests
  - Inter-AS & Intra-As sets entries for external dests

Inter-AS tasks

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

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

Job of inter-AS routing!



Example: Setting forwarding table in router 1d

- Suppose AS1 learns from the inter-AS protocol that subnet *x* is reachable from AS3 (gateway 1c) but not from AS2.
- Inter-AS protocol propagates reachability info to all internal routers.
- Router 1d determines from intra-AS routing info that its interface *I* is on the least cost path to 1c.
- Puts in forwarding table entry *(x,I)*.

Example: Choosing among multiple ASes

- Now suppose AS1 learns from the inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- To configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest x.
- This is also the job on inter-AS routing protocol!
- Hot potato routing: send packet towards closest of two routers.

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: Network Layer

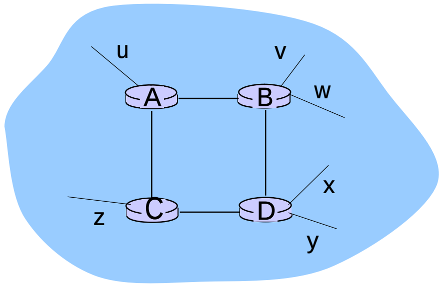
Intra-AS Routing

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

Chapter 4: Network Layer

RIP (Routing Information Protocol)

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

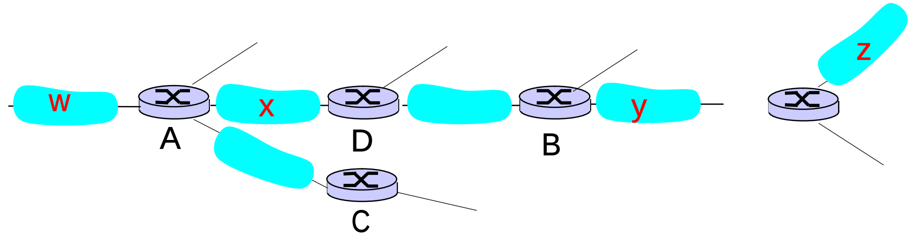


destination	hops
u	1
v	2
w	2
x	3
y	3
z	2

RIP advertisements

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

RIP: Example



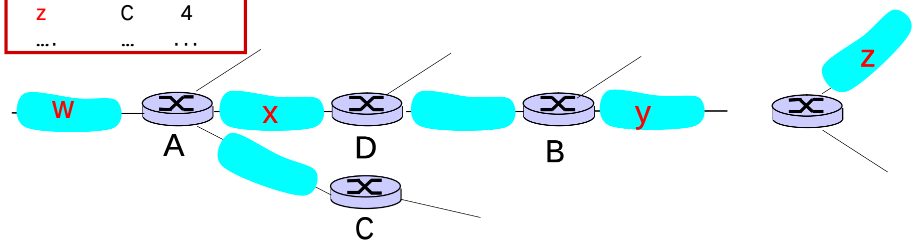
Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B	7
x	- -	1
….	….	. . . .

Routing table in D

RIP: Example

Dest	Next	hops
w	-	-
x	-	-
z	C	4
….	…	. . .

Advertisement from A to D



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	~~B~~ A	~~7~~ 5
x	- -	1
….	….	. . . .

Routing table in D

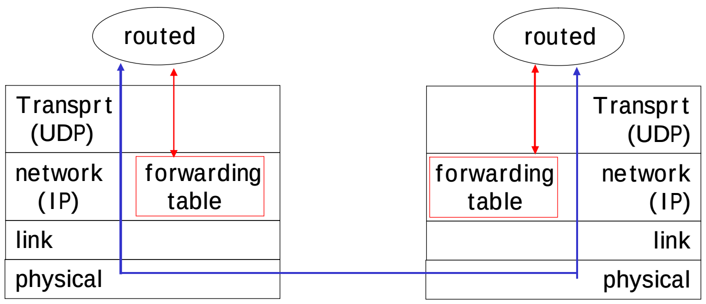
RIP: Link Failure and Recovery

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

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

RIP Table processing

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



Chapter 4: Network Layer

- 4. 1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What's inside a router
- 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 Broadcast and multicast routing

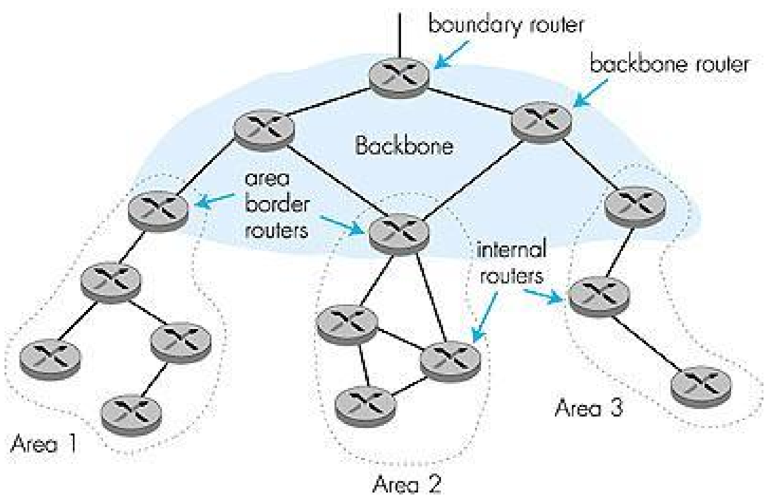
OSPF (Open Shortest Path First)

- "open": publicly available
- Uses Link State algorithm
  - LS packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra's algorithm
- OSPF advertisement carries one entry per neighbor router
- Advertisements disseminated to entire AS (via flooding)
  - Carried in OSPF messages directly over IP (rather than TCP or UDP)

OSPF "advanced" features (not in RIP)

- Security: all OSPF messages authenticated (to prevent malicious intrusion)
- Multiple same-cost paths allowed (only one path in RIP)
- For each link, multiple cost metrics for different TOS (e.g., satellite link cost set "low" for best effort; high for real time)
- Integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- Hierarchical OSPF in large domains.

Hierarchical OSPF



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.

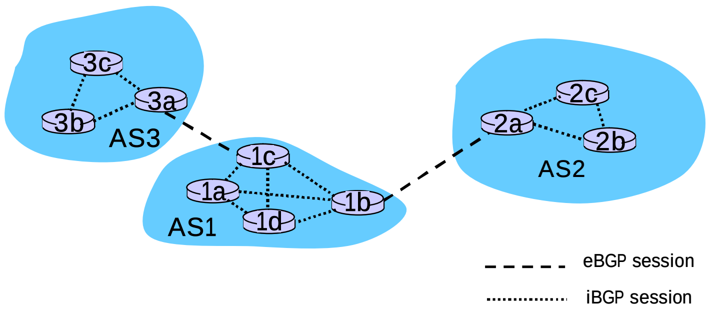
Chapter 4: Network Layer

Internet inter-AS routing: BGP

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

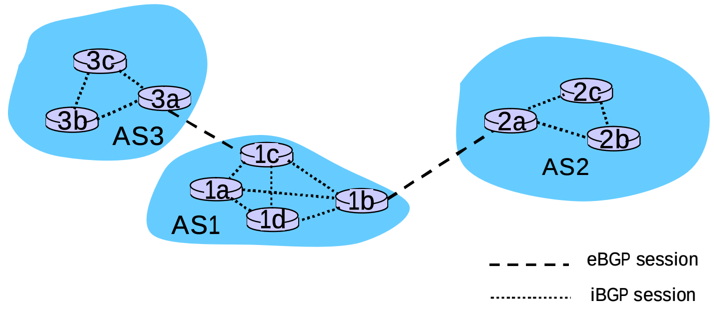
BGP basics

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



Distributing reachability info

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



Path attributes & BGP routes

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

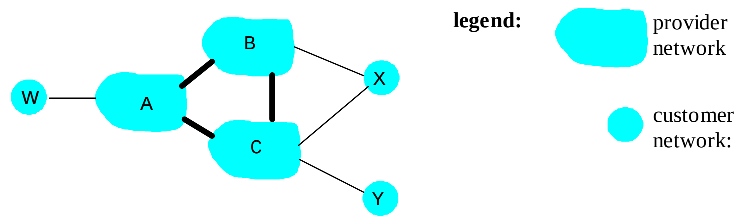
BGP route selection

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

BGP messages

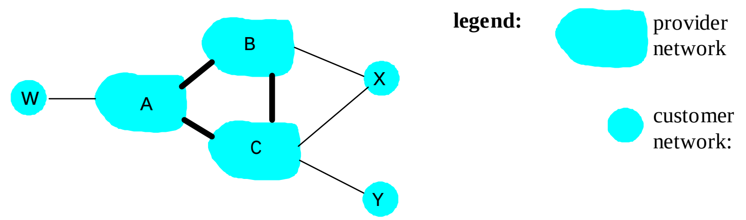
- BGP messages exchanged using TCP.
- BGP messages:
  - OPEN: opens TCP connection to peer and authenticates sender
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also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

BGP routing policy



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

BGP routing policy (2)



- A advertises to B the path AW
- B advertises to X the path BAW
- Should B advertise to C the path BAW?
  - No way! B gets no "revenue" for routing CBAW since neither W nor C are B's customers
  - B wants to force C to route to w via A
  - B wants to route *only* to/from its customers!

Why different Intra- and Inter-AS routing ?

Policy:

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

Scale:

- hierarchical routing saves table size, reduced update traffic

Performance:

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

Chapter 4: Network Layer

- 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

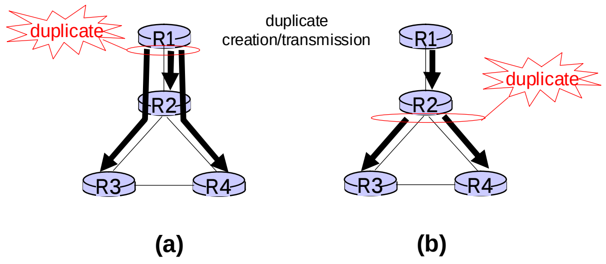


Figure 4.39 Source-duplication versus in-network duplication. (a) source duplication, (b) in-network duplication

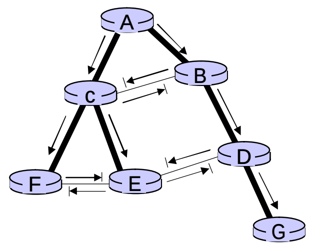
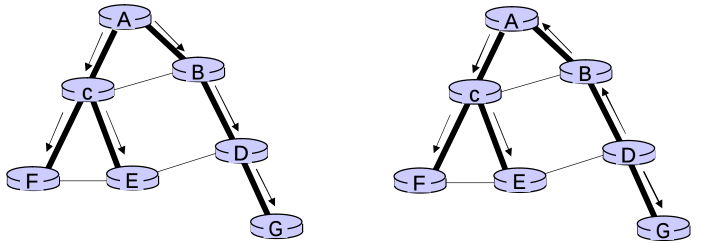
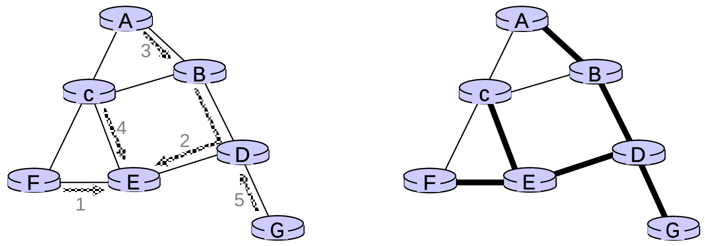


Figure 4.40: Reverse path forwarding



(a) Broadcast initiated at A **(b) Broadcast initiated at D**

Figure 4.41: Broadcast along a spanning tree

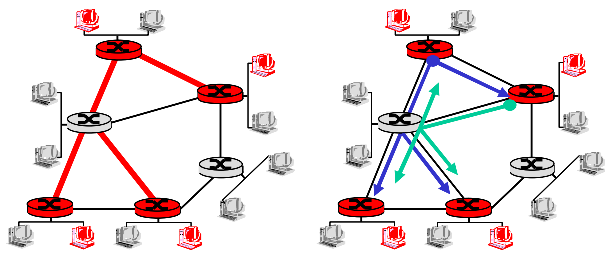


(a) Stepwise construction of spanning tree **(b) Constructed spanning tree**

Figure 4.42: Center-based construction of a 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
  - *source-based:* different tree from each sender to rcvrs
  - *shared-tree:* same tree used by all group members



Shared tree Source-based trees

Approaches for building mcast trees

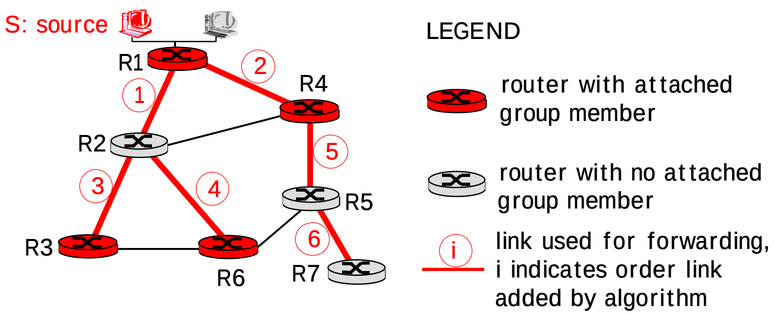
Approaches:

- 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

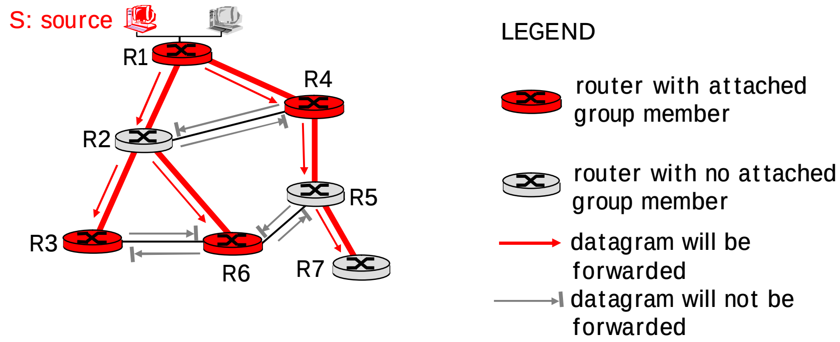


Reverse Path Forwarding

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

if (mcast datagram received on incoming link on shortest path back to center)
then flood datagram onto all outgoing links
else ignore datagram

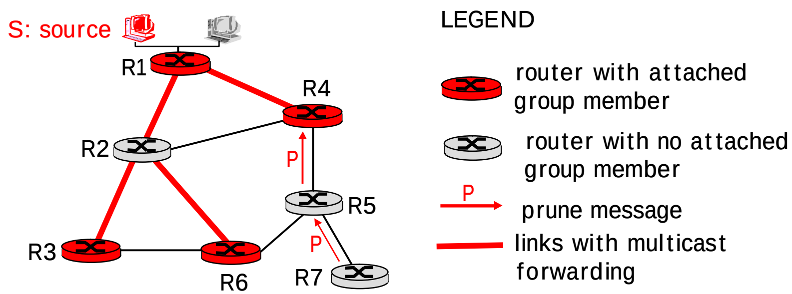
Reverse Path Forwarding: example



- 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

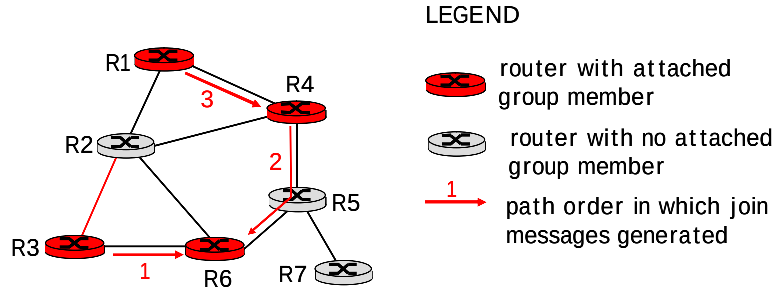
- Steiner Tree: minimum cost tree connecting all routers with attached group members
- problem is NP-complete
- excellent heuristics exists
- not used in practice:
  - computational complexity
  - information about entire network needed
  - monolithic: rerun whenever a router needs to join/leave

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: an example

Suppose R6 chosen as center:



Internet Multicasting Routing: DVMRP

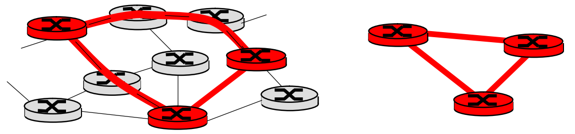
- DVMRP: distance vector multicast routing protocol, RFC1075
- *flood and prune:* reverse path forwarding, source-based tree
  - RPF tree based on DVMRP's own routing tables constructed by communicating DVMRP routers
  - no assumptions about underlying unicast
  - initial datagram to mcast group flooded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

DVMRP: continued…

- *soft state:* DVMRP router periodically (1 min.) "forgets" branches are pruned:
  - mcast data again flows down unpruned branch
  - downstream router: reprune or else continue to receive data
- routers can quickly regraft to tree
  - following IGMP join at leaf
- odds and ends
  - commonly implemented in commercial routers
  - Mbone routing done using DVMRP

Tunneling

Q: How to connect "islands" of multicast routers in a "sea" of unicast routers?



physical topology   logical topology

- mcast datagram encapsulated inside "normal" (non-multicast-addressed) datagram
- normal IP datagram sent thru "tunnel" via regular IP unicast to receiving mcast router
- receiving mcast router unencapsulates to get mcast 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 mcast tree (e.g., RPF)
- bandwidth and non-group-router processing *profligate*

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

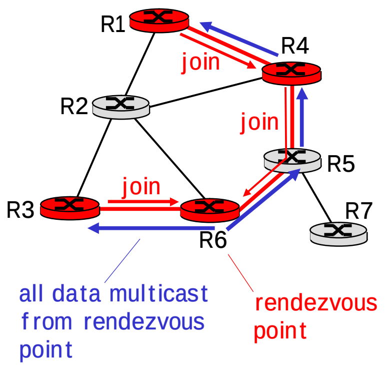
PIM- Dense Mode

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

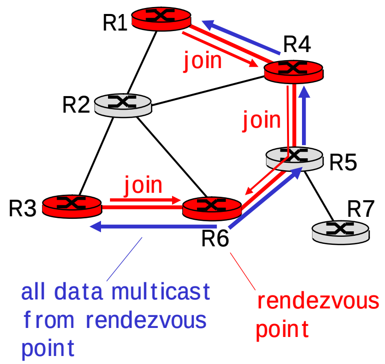
- center-based approach
- router sends *join* msg to rendezvous point (RP)
  - intermediate routers update state and forward *join*
- after joining via RP, router can switch to source-specific tree
  - increased performance: less concentration, shorter paths



PIM - Sparse Mode

sender(s):

- unicast 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!"



Network Layer: summary

What we've covered:

- network layer services
- routing principles: link state and distance vector
- hierarchical routing
- IP
- Internet routing protocols RIP, OSPF, BGP
- what's inside a router?
- IPv6

Next stop: the Data link layer!