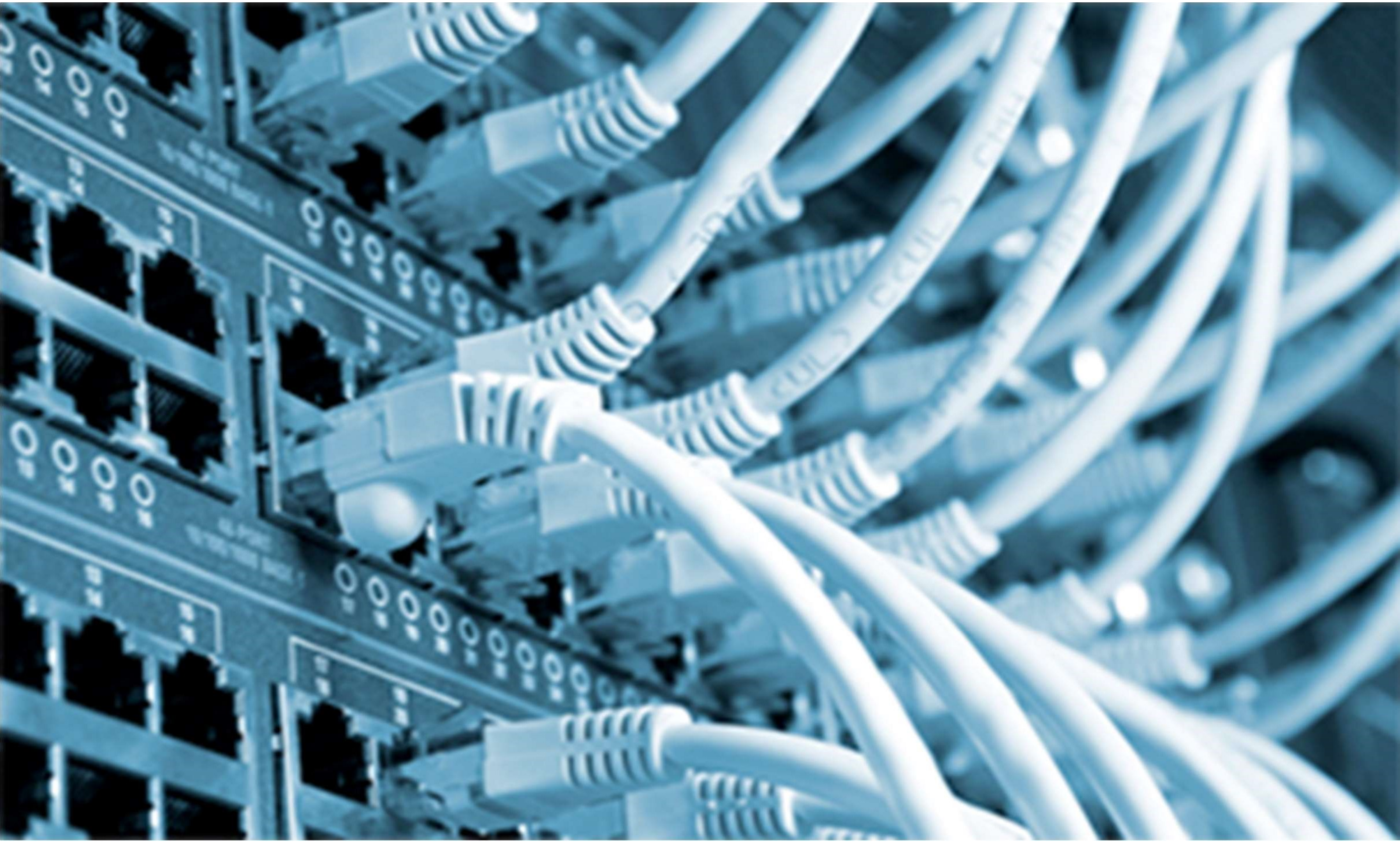


CS118 Discussion Week 7

Seungbae Kim

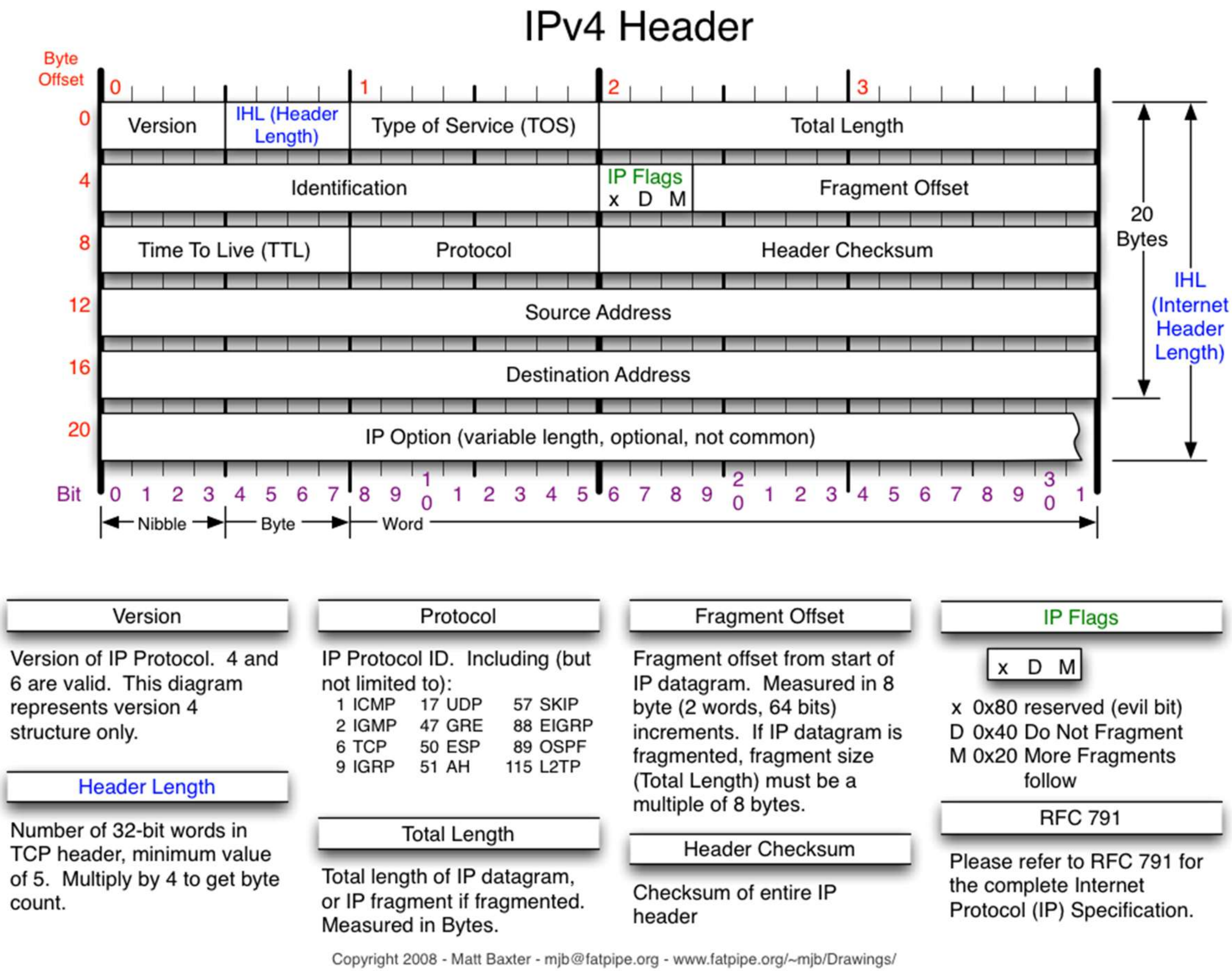
Outline

- IPv4
- ARP
- QnA for programming assignment

Network Layer

- Basic functions for network layer
  - Addressing
  - Routing
  - Fragmentation
- Network layer protocols
  - Addressing and fragmentation: IPv4, IPv6
  - Routing: RIP, OSPF, BGP, DVMRP, PIM, …
  - Others: DHCP, ICMP, NAT

IPv4 Header

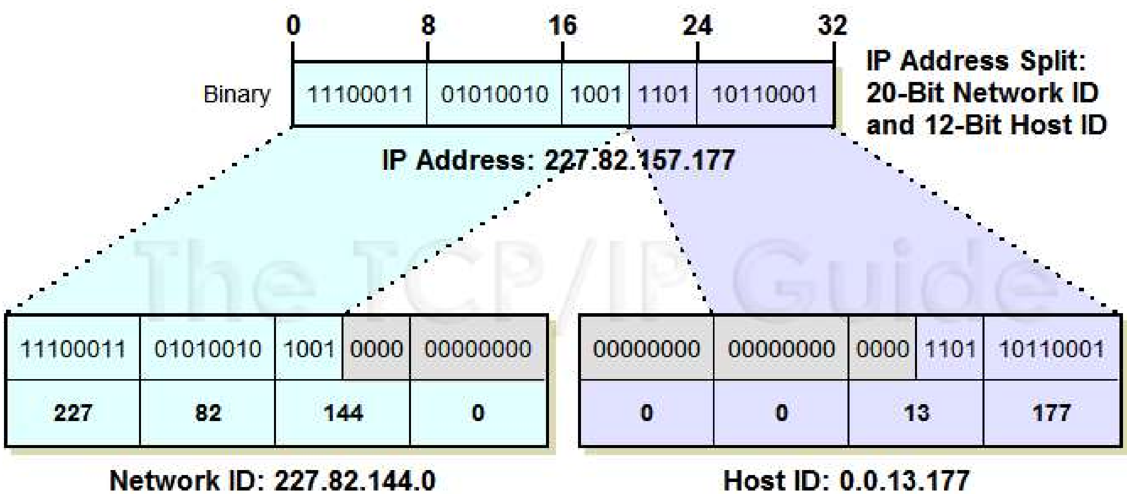

IPv4 Header
Byte Offset
0
4
8
12
16
20
0
1
2
3
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags
x D M
Fragment Offset
Time To Live (TTL)
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
20 Bytes
IHL (Internet Header Length)
Bit
0 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 20 1 2 3 4 5 6 7 8 9 30 1
Nibble
Byte
Word
Version
Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.
Header Length
Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.
Protocol
IP Protocol ID. Including (but not limited to):
1 ICMP 17 UDP 57 SKIP
2 IGMP 47 GRE 88 EIGRP
6 TCP 50 ESP 89 OSPF
9 IGRP 51 AH 115 L2TP
Total Length
Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.
Fragment Offset
Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.
Header Checksum
Checksum of entire IP header
IP Flags
x D M
x 0x80 reserved (evil bit)
D 0x40 Do Not Fragment
M 0x20 More Fragments follow
RFC 791
Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.
Copyright 2008 - Matt Baxter - mjb@fatpipe.org - www.fatpipe.org/~mjb/Drawings/

IP Address

- Globally Recognizable Identifier
  - IPv4: 0.0.0.0~255.255.255.255 (4 bytes)
  - Most IP addresses are globally unique
  - Exception:
    10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
    - Why?
- Hierarchical Addressing
  - Subnet: a portion of addressing space (CIDR)
    - <network address>/<subnet mask>

IP Address

- Hierarchical Addressing
  - Network ID + Host ID



  - Classless Inter-Domain Routing (CIDR)
    - <IP address>/<network prefix (mask)>
  - Subnetworking
  - Why hierarchical addressing?
    - ➢ **Route aggregation (Longest prefix matching)**

Exercise: Subnet

Consider a router that interconnects three subnets: Subnet 1, Subnet 2, and Subnet 3. Suppose all of the interfaces in each of these three subnets are required to have the prefix 223.1.17/24. Also suppose that Subnet 1 is required to support at least 60 interfaces, Subnet 2 is to support at least 90 interfaces, and Subnet 3 is to support at least 12 interfaces. Provide three network addresses (of the form a.b.c.d/x) that satisfy these constraints.

IP Fragmentation

- IP header contains three fields for the fragmentation
  - ID: All the fragments of an IP packet have same ID value
  - Flags: Don’t Fragment (DF) & More Fragment (MF)
  - Fragment offset: Specifies the offset of a particular fragment relative to the beginning of the original unfragmented IP packet (Measured in units of 8 bytes)

Routing

- How IP packets are delivered to a destination host?
  - Routing table lookup

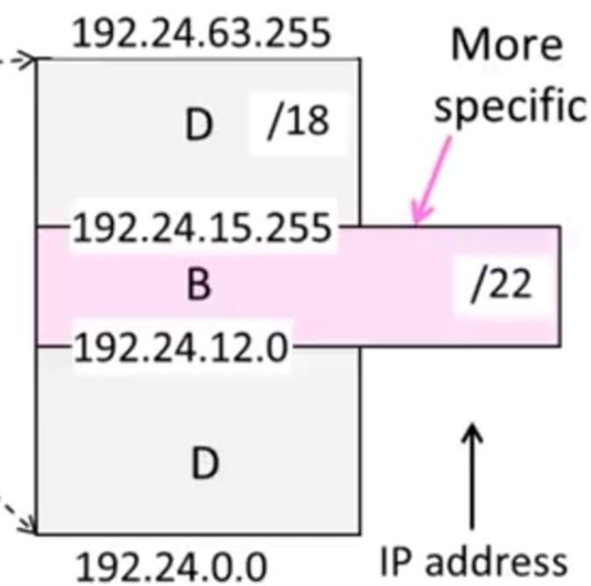
Network Destination	Netmask	Gateway	Interface	Metric
0.0.0.0	0.0.0.0	192.168.0.1	192.168.0.100	10
127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1
192.168.0.0	255.255.255.0	192.168.0.100	192.168.0.100	10
192.168.0.100	255.255.255.255	127.0.0.1	127.0.0.1	10
192.168.0.1	255.255.255.255	192.168.0.100	192.168.0.100	10

    - When an IP packet is received, a router looks up its routing table, compares with a destination IP address, and finds a next hop (gateway) to forward the packet.

Routing

- Longest Prefix Matching
  - Entries in routing table might have overlapped network prefix
  - Find the entry which has the longest prefix matching with incoming packet's destination IP

Prefix	Next Hop
192.24.0.0/18	D
192.24.12.0/22	B

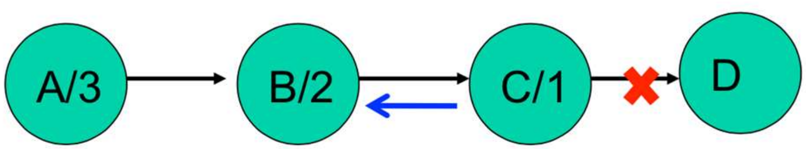


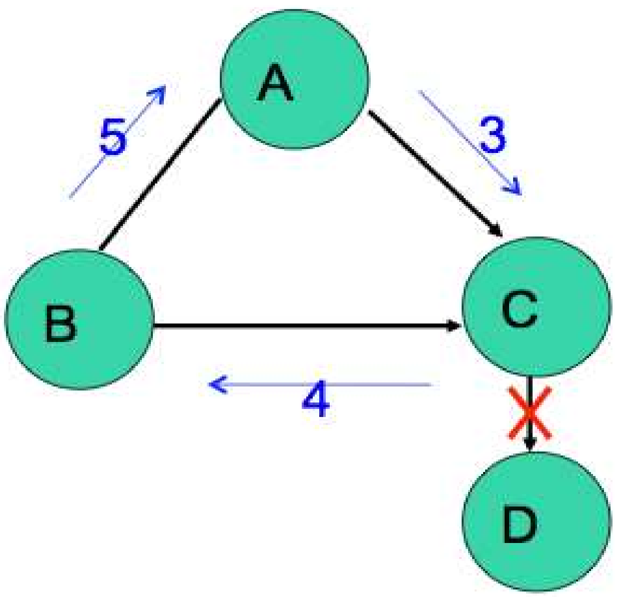
Routing

- Forwarding vs. Routing
  - Forwarding is the actual process of sending IP packets to an output interface
  - Routing determines a route from source to destination
- Route computation algorithms vs. Routing protocols
  - Algorithms: Distance-vector routing V.S. Link-state routing
  - Routing protocols: Intra-domain routing V.S. Inter-domain routing
    - Performance V.S. Policy

Distance-Vector Algorithm

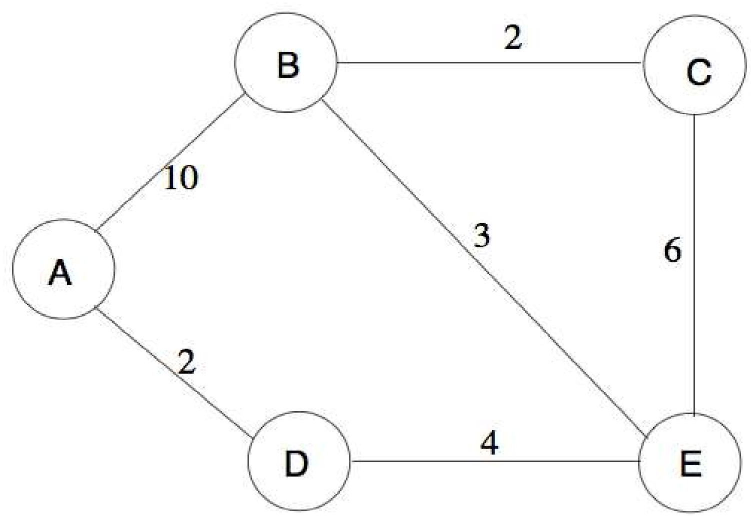
- Each node sends the **distance vector** to **all its neighbors**
  - Each node receives distance vectors from other neighbors, then calculate minimum cost for all Dsts.
- Count-to-infinity problem





Link-State Algorithm

- Each node broadcasts **its distance to all neighbors** to **entire network**
- Each node computes shortest paths too all nodes

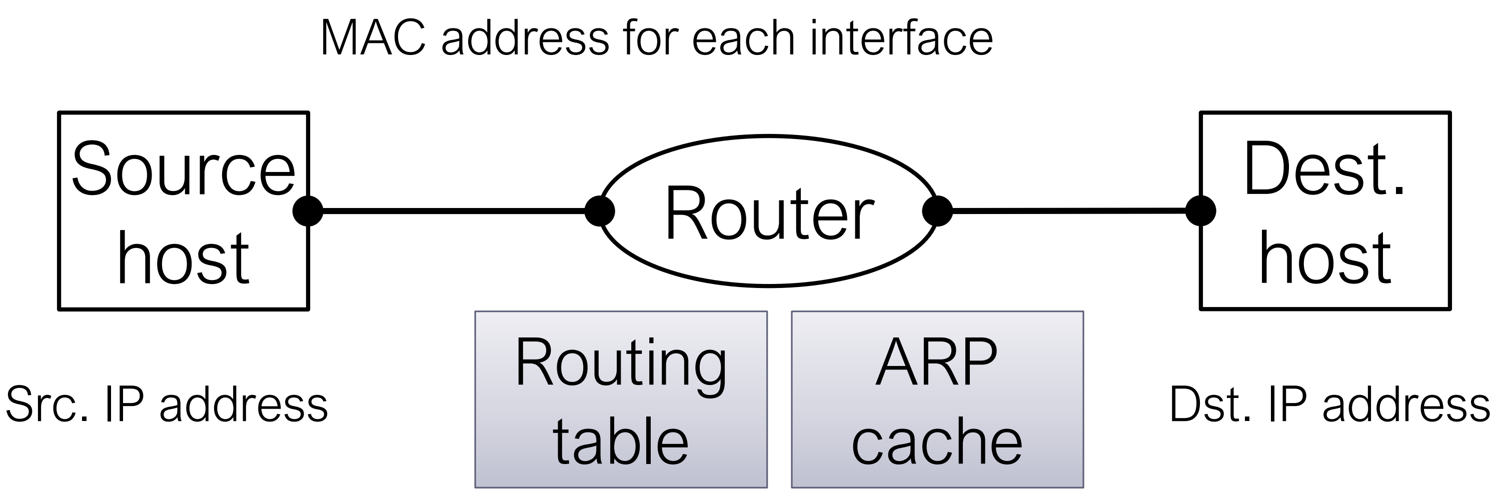


Link-State Algorithm

- Routing protocol based on Link State
- Link-State Packet (LSP)
  - Carries list of direct neighbors link cost
- How to broadcast (flood) LSP updates?
  - Sequence number and TTL
- Update every 30 minutes
- When detect link failure → update immediately
- How to detect link failure?

Address Resolution Protocol (ARP)

- Communication between two hosts
  - Source IP address and destination IP address
- Data transferring between two network interfaces
  - "Who has this IP address?" (Broadcast request)
  - "That is my IP! My MAC address is …" (Unicast response)



Address Resolution Protocol (ARP)

- ARP request
  - Src MAC address = Requester’s MAC address
  - Dst MAC address = Broadcast Mac address
    - Broadcast MAC address?
      - FF:FF:FF:FF:FF:FF
- ARP reply
  - Src MAC address = Respondent's MAC address
  - Dst MAC address = Requester’s MAC address

Programming Assignment

- Please carefully take a look the source codes to understand what you already have.
  - core/protocol.hpp
  - core/utils.hpp and utils.cpp
  - show-arp.py
- const uint8_t* buf = packet.data();

Programming Assignment
Handle Packet Procedure

1. Parse Ethernet header of the received packet
   A. Check dst. MAC address (router's interface or broadcast)
2. Check Ethernet type (Use *ethertype()* function in utils.cpp)
   A. If ARP? → handle ARP!
      I. ARP request? → Send ARP reply
      II. ARP reply?
      → Store IP/MAC info (*insertArpEntry()* function)
      then send cached packets out!
   B. If IPv4? → handle IPv4!

B. If IPv4? → handle IPv4!

I. Verify checksum and length (*cksum()* function)

II. Check destination IP

① Destined to the router

a) If ICMP packet? → handle Ping

i. Verify ICMP checksum

ii. Send ICMP echo reply

b) Else → discard

② Otherwise, forward the packet

a) Update IP header (TTL and checksum)

b) Lookup routing table and find next hop

c) Lookup ARP cache

i. If ARP entry found → Forward the packet

ii. If ARP entry not found → Cache the packet and send ARP reqeust

Programming Assignment Tips

- Checksum
  - When you build a (IP/ICMP) header,
    set checksum as 0 then calculate checksum.
- Check out the ***arp-cache.hpp*** file
  - You have *PendingPacket* & *ArpRequest* structs
  - Each ArpRequest holds list of pending packets