Network Layer Protocols

- **Goals:**
  - Understand principles behind Network layer services
  - Instantiation and implementation in the Internet

How the network layer moves a segment from the transport layer of an origin host to the transport layer of the destination host?

Internet Network Layer: Overview

- Network layer services
- IP – Internet Protocols: Format and Addressing schemes
- How Does Internet works? What’s inside a router?
- IP Subnetworking and routing
- ARP/DNS/NAT/DHCP/CIDR
- ICMP – for error control/report
- Routing principle: Hierarchical routing and path selection*
- Routing Algorithms and Internet Routing Protocols
- IPv6 Protocol – header, addressing (a brief), migration

Network Layer Functions

- Transport packet from sending to receiving hosts via internet
- Network layer protocols exist in every host and router
- Three important functions:
  - **path determination**: route taken by packets from source to destination (Routing algorithms)
  - **forwarding**: move packets from a router’s input to a appropriate router’s output
  - **call setup**: some networks require router call setup along path before data flows (e.g., MPLS)

Network Service Model

**Q: What’s service model for “channel” transporting packets from sender to receiver?**

- guaranteed bandwidth?
- preservation of inter-packet timing (no jitter)?
- loss-free delivery?
- in-order delivery?
- congestion feedback to sender? (protective CC)

Network-level Service model

Determines end-to-end Characteristics of transporting Data between network edges

Service Abstraction (properties)

Network service model

virtual circuit or Datagram

The most important abstraction provided by network layer.
Virtual Circuit Networks

- "source-to-destination path" behaves much like "telephone circuit"
  - performance-wise
  - network actions along source-to-destination path from VC to VC

- call setup for each call before data can flow
- each packet carries VC identifier (not destination host OD)
- every router on source-destination path maintains "state" for each passing (undergoing) connection (through it)
  - transport-layer connection only involved two end systems
  - Link and router resources (bandwidth, buffers) may be allocated to VC
    - to get (real) circuit-like performance

Virtual Circuits: via signaling protocols

- Protocols used to exchange signaling messages
  - setup, maintain ("initiation" mostly), and teardown VC
- Ex: ATM, frame-relay, X.25 VC-based networks (see chp.5)
- But . . . not used in today's Internet

Datagram Networks: the Internet Model

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of "connection" → out-of-order
- packets typically routed using destination host ID
  - packets between same source-dest pair may take different paths

Internet Architecture: TCP/IP Protocol suite

- Based on protocol number
  - Interface, so-called SAP
- Based on protocol type
  - TCP
- UTP
- TP Layer
- Segment or Datagram
- Packet (Datagram)
More on Internet Network (IP) layer

Network layer for host and router: datagram-oriented functions

1. IP protocol
   - addressing conventions
   - datagram format
   - packet handling conventions

2. Routing protocols
   - path selection
   - Ex: RIP, OSPF, BGP

3. ICMP protocol
   - error reporting
   - router "signaling"

Transport layer: TCP, UDP

Network layer

Routing table

Link layer

physical layer

1. IP overview - I

- IP is designed to interconnect packet switched (datagram) communication networks to form an internet.
- It transmits blocks of data known as datagrams received from IP’s upper-layer software to and from hosts.
- IP provides best-effort or connectionless delivery.
- IP is responsible for addressing.
- Two versions of IP: version 4 (RFC 791, currently) and version 6 (RFC 2373, 2460)
  * Network information is distributed via routing protocols.

2. IP overview - II

- IP’s main function is to provide for the interconnection of subnetworks to form an internet in order to pass data.

- Summary of IP functions:
  - Basic unit for data transfer (format, encapsulation)
  - Addressing
  - Routing (supporting packet forwarding)
  - Fragmentation of packets/datagrams

IP packet/datagram format

<table>
<thead>
<tr>
<th>Bit</th>
<th>0</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Version</td>
<td>HHL</td>
<td>Type-of-service</td>
<td>Total length</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Identification</td>
<td>Flags</td>
<td>Fragment offset</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Time-to-live</td>
<td>Protocol</td>
<td>Header checksum</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source IP address
Destination IP address
IP Option (IF any)
Padding
Data (from TP layer)

IP over Ethernet

Ethernet frame

<table>
<thead>
<tr>
<th>Dest MAC</th>
<th>SRC MAC</th>
<th>TYPE</th>
<th>IP Header + Data</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bytes</td>
<td>6 bytes</td>
<td>2 bytes</td>
<td>46-1500 bytes</td>
<td>4 bytes</td>
</tr>
<tr>
<td>(MAC Address)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**IP datagram format – an Overview**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP protocol version</td>
<td>4-bit version of IP protocol</td>
</tr>
<tr>
<td>header length (bytes)</td>
<td>4-bit header length</td>
</tr>
<tr>
<td>“type” of data</td>
<td>Length of the 16-bit identifier</td>
</tr>
<tr>
<td>max number remaining</td>
<td>8-bit max number of remaining hops</td>
</tr>
<tr>
<td>hops (decrypted at each router)</td>
<td>8-bit maximum remaining hops (decremented at each router)</td>
</tr>
<tr>
<td>upper layer protocol</td>
<td>Upper layer protocol to deliver payload to</td>
</tr>
<tr>
<td>to deliver payload to</td>
<td></td>
</tr>
<tr>
<td>32 bits source IP addr</td>
<td>32-bit source IP address</td>
</tr>
<tr>
<td>32 bits destination IP addr</td>
<td>32-bit destination IP address</td>
</tr>
<tr>
<td>Options (if any)</td>
<td>Options (if any)</td>
</tr>
<tr>
<td>data</td>
<td>Data (variable length, typically a TCP or UDP segment)</td>
</tr>
</tbody>
</table>

**IPv4 Header - TTL**

- **Time-to-live** — Indicating the amount of time (in second) that a datagram is allowed to stay on the network. (A 8-bit field, Max TTL = 255.)

- A datagram or some of its fragments may loop indefinitely through the internet when dynamic or alternate routing is used.
- The 「 lifetime 」 should be a true measure of time. This requires some global clocking mechanism - estimating exact time is difficult.
- A simple way to implement lifetime is to use a hop count. The initial is set by the packet originator and it varies.

**IPv4 Header Checksum**

- **Checksum** ~ to ensure the integrity of the IP header (RFC 1071)
- Calculated over IP header only ((5 ~ 15) x 32-bit in total since HLEN = 4-bit long. Can you see it ?)
- Each router will recompute the checksum while forwarding a DG. Why ? Because TTL is changed by each router the DG traverses.
- How ? (refer to : RFC 1141)
Checksum (CKS) Computing

In sending station:
- CKS = 0 initially for an outgoing DG
- Calculate the 16-bit one's complement sum of the header (i.e., the entire header is considered a sequence of 16-bit words) and store the 16-bit one’s complement of the sum in CKS field

Example (downsize to 8-bit)
CKS = 00000000 (initial)
Header = 10010110 11001000 01000010 00010110 . . .
CKS = (01101001 + 00110111 + 10111101 + 11101001)’
  = . . .
  = 1110101

In receiving station:
- When an IP DG is received, the 16-bit one's complement sum of the header is re-calculated
- CKS = all 1’s, if no error (since the receiver’s CKS calculation contains the CKS stored by the sender)
- Declare checksum error if CKS ≠ all 1’s, and IP discards the received DG (No error message is generated upon error detected. It is up to the higher layers to somehow detect the missing datagram and retransmit it.)

Example (cont’d) – if no error
CKS_new = ((data_1)’ + (data_2)’ + . . . + (CKS_old)’)’
  = . . .
  = 11111111

IP Addressing
- IP address: 32-bit identifier for host and router's interface
- Interface: connection between host, router and physical link
  - Routers - typically have multiple interfaces
  - Host - may have multiple interfaces
  - IP addresses associated with interface(s), not host or router

IP address Structure
- Divided 32-bit into 4 octets of binary
- Use Dotted-decimal representation

A Note on IP Address
- An IP address does not identify a specific computer.
  Instead, each IP address identifies a connection between a computer and a network. A computer with multiple network connections (e.g., a router) must be assigned one IP address for each connection.
  - Global authority (IANA) assigns unique prefix to network
  - Local administrator assigns unique suffix to host (interface)

NetID HostID
### IP Addressing (cont'd)

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

### Routers Interconnect Networks

- **How to find the networks?**
  - Detach each interface from routers and hosts
  - create "islands" of isolated networks
  - call each a "network"

### IP Addressing: Classful and Classless

- **Classful addressing** (RFC 791)
  - Fixed number of bits for network and host portions
  - *inefficient* use (giving out) of address space (IP address exhaustion)
    - e.g., class B net allocated enough addresses for 65K hosts,
      even if only 2K hosts in that network ✏ waste address space

- **CIDR: Classless InterDomain Routing** (RFC 1519)
  - network portion of address of arbitrary length
  - address format: `a.b.c.d/x`, where *x* is the number of leading bits in network portion of address

### Classes of Original IP Address

- **Primary IP Classes**: Class A, B, and C

  **Class A**
  - 7-bit network address: `24` bits of host address
  - First byte
  - Last three bytes
  - (0000000) = 0  ~  (1111111) = $2^7 - 1$

  **Class A**: `0.0.0.0` ~ `127.255.255.255`
  - Networks (Max): `1.0.0.0` ~ `126.0.0.0` (128 - 2)
  - Max Hosts/network: `16,777,216 - 2`
  - \[ (2^{24} - 1) \]
Classes of Original IP Address (cont’d)

Class B

10

14-bit network address

16 bits of host address

First two bytes

Last two bytes

• Class B : 128.0.0.0 ~ 191.255.255.255
  – Networks (Max) : 128.1.0.0 ~ 191.254.0.0 (16386 - 2)
  – Max Hosts/network : 65,536 - 2

• The most popular IP address assignment
• 20% of class B were assigned by July 1990 and DOUBLING every 14 months → will be exhausted by Early 1994 (Not so bad)
  (projection on March, 1994) ← May you find the up-to-date info?

Class C

110

21-bit network address

8 bits of host address

First three bytes

Last byte

• Class C : 192.0.0.0 ~ 223.255.255.255
  – Networks (Max) : 192.0.1.0 ~ 223.255.254.0 (2097154 - 2)
  – Max Hosts/network : 256 - 2

Class D

1110

Multicast address (group ID)

• Class D : 224.0.0.0 ~ 239.255.255.255 (Multicasting)
  - mapped to a physical address (assigned in the NIC)

Class E

11110

Reserved for future use

• Class E : 240.0.0.0 ~ 247.255.255.255 (Reserved)

Classful IP Addresses Summary

given notion of “network”, let’s re-examine IP addresses:

• Called “classful” addressing:

  class
  1.0.0.0 to 127.255.255.255

  128.0.0.0 to 191.255.255.255

  192.0.0.0 to 223.255.255.255

  224.0.0.0 to 239.255.255.255

  32 bits
Special IP Addresses

- Hostid = all 0 ⇒ This network
- Netid = all 0 & Hostid = any ⇒ Specified host on this Net
- Netid = all 0 & Hostid = all 0 ⇒ The host on this Net
- Hostid = all 1 ⇒ Broadcast on the specified network (directed broadcast, getting through Router)
- Netid = all 1 & Hostid = all 1 ⇒ LAN broadcast address (limited broadcast, w/o getting through a Router)
- 127.x.x.x ⇒ loop back address ~ for internal testing (packets would not be sent onto network)
  * Mostly, use 127.0.0.1 (Try: ping 127.0.0.1 under MS-DOS window)
  * Those SIPs are applicable to the networks with subnetting

Private IP Addresses

- For local use only (allowing duplication among different LANs)
- A legal IP address only in a LAN (private use) but would not be recognized by Internet (eg., routers)
- Three PIP’s range (RFC 1918):

<table>
<thead>
<tr>
<th>IP range</th>
<th>Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0 ~ 10.255.255.255</td>
<td>255.0.0.0</td>
</tr>
<tr>
<td>172.16.0.0 ~ 172.31.255.255</td>
<td>255.240.0.0</td>
</tr>
<tr>
<td>192.168.0.0 ~ 192.168.255.255</td>
<td>255.255.255.0</td>
</tr>
</tbody>
</table>

Note on PIP

A Note on Private IP Address

In 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
How Does IP Handle Fragmentations?

<table>
<thead>
<tr>
<th>VERS</th>
<th>HLEN</th>
<th>Service Type</th>
<th>Total Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Identification | Flags | Fragment Offset |
| Time to Live | Protocol | Header Checksum |

- Different media allows for different-sized datagrams to be transmitted and received.
- *Fragmentation* allows a datagram that is too large to be forwarded to the next LAN segment to be broken up into smaller segments to be reassembled at the destination.
- The fragmentation occurs at the router that cannot forward packet to the next interface directly.
- Applications should use *path MTU discovery* (RFC 1191) to find the smallest datagram size. Send 576 if no information.

Fragmentation Control in the IP Header

- **Identification** — Contains an integer that identifies the current datagram. The receiving IP layer use this field and source IP address to help piece together datagram fragments. (16-bit)
- **Flags** — A 3-bit field of which the low-order 2 bits control fragmentation. One bit specifies whether the packet can be fragmented; the second bit specifies whether the packet is the last fragment in a series of fragmented packets.
- **Fragmentation offset** — Indicating the offset (in bytes) from the previous datagram that continues the complete datagram.
- Fragments’ header duplicate most of the original datagram header.

IP Fragmentation and Reassembly

- **Disadvantages:**
  - Larger buffers are required at intermediate gateways
  - All fragments of a datagram may be forced to pass through the same gateway. This inhibits the use of dynamic routing.
- **Example**

```
<table>
<thead>
<tr>
<th>MTU examples: FDDI ~ 4352; X.25 ~ 576; PPP ~ 296 bytes</th>
</tr>
</thead>
</table>
```

(Done by routers in today’s Internet)

IP FAR (cont’d)

- **Flag**
  - DF: Don’t Fragment
    - 0 ~ May Fragment
    - 1 ~ Don’t Fragment
  - MF: More Fragments
    - 0 ~ Last Fragment
    - 1 ~ More Fragment

```
<table>
<thead>
<tr>
<th>Flag</th>
<th>DF</th>
<th>MF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
```

```
IN: 3980 data bytes; IP header = 20 bytes
```

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

```
One large datagram becomes several smaller datagrams
```

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

```
<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>fragflag</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>=1500</td>
<td>x</td>
<td>=1</td>
<td>=1480</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>fragflag</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>=1040</td>
<td>x</td>
<td>=0</td>
<td>=2960</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 (RFC 2050)
  ➢ allocates addresses by three regional Internet registries
    • American Registry for Internet Number (ARIN)
      – handles registrations for North and South America, as well as parts of Africa
    • Reseaux IP Europeans (RIPE)
      – covers Europe and nearby countries
    • Asia Pacific Network Information Center (APNIC)
  ➢ manages DNS (root DNS servers)
  ➢ assigns domain names, resolves disputes

IP problems? (reseau ~ 網狀組織)

IP Routing - Getting a Datagram from Src to Dest

Forwarding a datagram - Scenario 1

Starting at A, given IP datagram addressed to B:
• look up net. address of B first
• find B is on same network as A
  ➢ By consulting its IRT
• link layer will send datagram directly to B inside link-layer frame (discussed later)
  ➔ B and A are directly/physical connected

Here, 223.1.1 ➔ 223.1.1.0/24

Forwarding a datagram - Scenario 2

Starting at A, dest. E:
• look up network address of E
• E on different network (since Nhop=2)
  ➔ A, E are not directly connected
  ➢ routing table ➔ next hop router to E is 223.1.1.4
  ➢ link layer sends datagram to router 223.1.1.4 inside link-layer frame
  ➢ datagram arrives at 223.1.1.4 (continued.....)
**Forwarding a datagram - router’s action**

Arriving at 223.1.1.4, destined for 223.1.2.2
- look up network address of E
- E on same network as router’s interface 223.1.2.9
  => router, E directly attached
- link layer sends datagram to 223.1.2.2 inside link-layer frame via interface 223.1.2.9
- datagram arrives at 223.1.2.2!!! (hooray!)

**Actions Taken by a Router upon Receiving a Packet**

<table>
<thead>
<tr>
<th>Packet Received</th>
<th>Header and checksum valid?</th>
<th>Decrement TTL; TTL &gt;= 0?</th>
<th>Route Table lookup based on destination address</th>
<th>Route found?</th>
<th>Default route available?</th>
<th>YES</th>
<th>NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If route is available, search for MAC address in ARP cache
- Send ICMP error message to originator
- Send ARP request and wait for a response

Received ARP reply?
- Build new packet with MAC address and route packet through port found in routing table.
- Received ARP reply, insert MAC and IP address into ARP table

**IP Packet Forwarding over Internet**

What if the MAC address is unknown

**Router - the Internet Packet Forwarding Device**

- Routing is an overhead activity (store-and-forward)
- Performance is derived from switching mechanism (forwarding rate in PPS)
- Capability depends upon the protocols (e.g., IP, IPX) and interfaces (ports) [e.g., E, FE, ATM(25,155.5,622.02 Mbps or more), T1/E1, T3] it can support
- Separate networks from broadcast and independent of OS and networks
**Some a prior information are needed before sending out a Packet**

- What if the MAC address of a destination is unknown giving IP address?
  - Calls for **ARP** to find it out

- What if the IP address is unknown giving host name only?
  - Calls for **DNS lookup**

---

**Address Resolution Protocol (ARP)**

- RFC 826.
- TCP/IP addresses are **32** bits and represent a network, subnet, and host ID.
- Addresses on LANs are represented by physical (MAC) layer addresses and they are **48** bits in length.
- ARP provides the mapping between a host’s 32-bit IP address and its 48-bit MAC address.
- ARP works only on the local subnet (it cannot traverse routers).
- ARP builds a table of IP/MAC addresses to properly format a source and destination address field in a packet (ARP cache).

---

**ARP Operations : Request and Reply**

~ IP address (4-byte) ⇒ MAC address (6-byte) service

- **ARP request** (via broadcasting):
  - A asks MAC address of B (with IP address given) = ?

- **ARP reply**:
  - B (only) replies its MAC address; others keep silent

---

**ARP Packet Format**

- **Operation**
  - 1 (request) or 2 (response)

- **Type of hardware address**
  - HA type = 1 (for Ethernet)

- **Type of protocol address**
  - PA type = 0x0800 (w/ IP)

- **Length of header**
- **Length of protocol address**

**Operation**

- Hardware address of the source station
- Protocol address of the source station
- Hardware address of the destination station
- Protocol address of the destination station

**Data**

- (ARP over Ethernet directly)

- HA type = 1 (for Ethernet) ; PA type = 0x0800 (w/ IP)

- Operation = 1 (request) or 2 (response)
ARP : Sending, Identifying, and Caching

- Transmit ARP in a frame

- Identifying ARP frame

- ARP software maintains a small table of bindings (IP ↔ MAC address) in memory – called ARP cache ~ to reduce network traffic (but not last for too long)

ARP : Local and Remote

- Router replies its MAC address if B is located out of the segment

To find B’s MAC addr

To see the ARP cache (maintained by the station for a short period)
- **arp -a** (empty, when you just open it)
- **arp -a** (check arp cache after a few connections)

ARP broadcast packet ~ Get router’s MAC address

ARP broadcast packet ~ Get local destination’s MAC address

- **RARP** (Reverse ARP) ~ In contrast to ARP
  ~ MAC address ⇒ IP address mapping services
  ~ For diskless devices (a way of managing IP address)

DNS Naming Rule

**Example**: Naming hierarchy in internet

- **Domain Name Service (DNS)**
  - Domain (網域) ~ the coverage of a Netid (network)
  - **DNS** ~ Hierarchical (階層式) naming rules in internet
    - Example: Host name -- en.ntut.edu.tw, www.edu.tw
      www.epson.com.tw
  - Domain Name System Server (**DNS server**) ~ containing a database (look-up table) for host name to IP address mapping
    - Example: Host name
      en.ntut.edu.tw  IP address = 140.124.70.26
• Links among DNS servers
  - All DNS servers are linked together to form a unified system.
  - Each server knows how to reach a root server and servers that are authorized by itself (the further down the hierarchy).

**Question:** How does a DNS server know which other DNS server is the authority for a given name?

**Answer:** It does not know. Gee!

• Resolving a name (machine)
  - The DNS server knows where is the root DNS server and resolve the name from there.
  - The resolver acts as a client by sending DNS request to other DNS servers for name resolving (call for help).

---

**(URL Uniform Resource Locator)**

• **URL (網址)**
  ~ providing a uniform way to access resources in the Web.
  • Format
    – “protocol”:// “host name” “port” “path”
    – Examples:
      2. file:///C:/ProgramFiles/Netscape/kwk/bookmark.htm

---

**Name Resolving Operation**

**Example:** (iterative resolution - one of query methods)
A remote site (say, syl.ntu.edu.tw) sends a request to its local DNS server 'ntu.edu.tw' for resolving the name “kwk.en.ntut.edu.tw”

1. The authority server of “x.en.ntut.edu.tw”
2. Ftp kwk.en.ntut.edu.tw
3. Reply ntut.edu.tw’s IP
4. ntu
5. Reply en.ntut.edu.tw’s IP
6. ftp
7. Reply kwk.en.ntut.edu.tw’s IP
8. sys.ntu.edu.tw

---

**URL Examples**

1. Gopher://mitdir.mit.edu:105
2. Mailto:kwk@en.ntut.edu.tw
3. News:com.dcom.cell-relay
Summary of IP Transfer over Internet

- Case 1: A Æ B ~ direct (no router involved)
- Case 2: A Æ C ~ indirect (via default router first)

Dest & Src MAC addr altered hop-by-hop, but IP addresses keep unchanged through.

Example

IP Addresses: How to get one?

Hosts (host portion):
- Manual configuration: ~ 固定IP
  - hard-coded by system administrator into the host (in a file)
- DHCP: Dynamic Host Configuration Protocol: (RFC 2131) (dynamically get address: "plug-and-play")

DHCP Protocol’s Four-step process: ~ 自動取得
- 1. "DHCP server discovery"
- 2. "DHC server offer(s)"
- 3. "DHCP request"
- 4. "DHCP ack"
  - DHCP is an extension of BOOTP (RFC1542) and is used extensively in LANs and in residential Internet access.

Immediate IPv4 Problems

- Running out of network ID’s . . .
- Solution:
  - Subnetworking (subnetting)

Subnetting – Dividing the Network Locally

- What is a Subnet （子網路）?
  ~ A network segment (with different network ID in their IP address) separated by routers.

- Why need a subnet?
  ~ to create more networks ID (by reducing host ID)
  ~ to help conserve the IP address space
Subnetting by Subnet Mask

• How to do it?
  ~ One network assigned to a site and it is allowed to “chop” up that network number to create subnets.
  ~ by “subnet mask” that divides the IP address into subnets (RFC 950)

Subnet mask = 255.255.255.224

Subnetting Example

• IP address assignment 150.5.0.0.
• Requirements of 75 subnets and 75 hosts per subnet.
• First set is to find out how many bits are needed for 75 subnets?
  – $2^5 = 32, 2^6 = 64$ and $2^7 = 128 > 75$ therefore we need $n = 7$ bits to subnet for 75 subnets
  – This leaves 9 bits for host assignment (16 - 7) which allows for 510 hosts
• Subnets are masked starting from the left and hosts are configured starting from the right.

Result: 150.5.XXXXXXXXX.XXXXXXXXX

Subnetting Exercise

A Class C IP address = 203.54.48.0 ~ 256 nodes max.
Wanted: Subnetting it to FOUR subnets.
Require to set Subnet Mask = 255.255.255.

⇒ 11111111.11111111.11111111.11111111.

• The resulting subnet IP addresses:
  IP address for subnet #1 = 203.54.48.0 ~ 203.54.48.
  IP address for subnet #2 = 203.54.48.0 ~ 203.54.48.
  IP address for subnet #3 = 203.54.48.0 ~ 203.54.48.
  IP address for subnet #4 = 203.54.48.0 ~ 203.54.48.

• What is the broadcast addresses for each subnet?
**DHCP: Dynamic Host Configuration Protocol**

* Obtaining a host address automatically, Method - I:

**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 an "on")
- Support for mobile users who want to join network (more shortly)

**DHCP operations:**
- Host broadcasts "DHCP discover" msg
- DHCP server responds with "DHCP offer" msg
- Host requests IP address: "DHCP request" msg
- DHCP server sends address: "DHCP ack" msg

**DHCP client-server scenario (explained)**

DHCP discover

- **DHCP discover**
  - src: 0.0.0.0, 68 (BOOTP client port)
  - dest: 255.255.255.255, 67 (server port)
  - Your inet addr: 0.0.0.0
  - transaction ID: 654

DHCP offer

- **DHCP offer**
  - src: 223.1.2.5, 67
  - dest: 255.255.255.255, 68
  - Your inet addr: 223.1.2.4
  - transaction ID: 654
  - Lifetime: 3600 secs

DHCP request

- **DHCP request**
  - src: 0.0.0.0, 68
  - dest: 255.255.255.255, 67
  - Your inet addr: 223.1.2.4
  - transaction ID: 655
  - Lifetime: 3600 secs

DHCP ACK

- **DHCP ACK**
  - src: 223.1.2.5, 67
  - dest: 255.255.255.255, 68
  - Your inet addr: 223.1.2.4
  - transaction ID: 655
  - Lifetime: 3600 secs

**DHCP Client-Server Scenario**

- **DHCP server:** 223.1.2.5
- **DHCP client:** 223.1.2.2

**NAT: Network Address Translation**

* Obtaining a host address automatically, Method - II:

**DHCP client-server scenario (explained)**

DHCP discover

- **DHCP discover**
  - src: 0.0.0.0, 68 (BOOTP client port)
  - dest: 255.255.255.255, 67 (server port)
  - Your inet addr: 0.0.0.0
  - transaction ID: 654

DHCP offer

- **DHCP offer**
  - src: 223.1.2.5, 67
  - dest: 255.255.255.255, 68
  - Your inet addr: 223.1.2.4
  - transaction ID: 654
  - Lifetime: 3600 secs

DHCP request

- **DHCP request**
  - src: 0.0.0.0, 68
  - dest: 255.255.255.255, 67
  - Your inet addr: 223.1.2.4
  - transaction ID: 655
  - Lifetime: 3600 secs

DHCP ACK

- **DHCP ACK**
  - src: 223.1.2.5, 67
  - dest: 255.255.255.255, 68
  - Your inet addr: 223.1.2.4
  - transaction ID: 655
  - Lifetime: 3600 secs

**NAT: Network Address Translation**

**DHCP client-server scenario (explained)**

DHCP discover

- **DHCP discover**
  - src: 0.0.0.0, 68 (BOOTP client port)
  - dest: 255.255.255.255, 67 (server port)
  - Your inet addr: 0.0.0.0
  - transaction ID: 654

DHCP offer

- **DHCP offer**
  - src: 223.1.2.5, 67
  - dest: 255.255.255.255, 68
  - Your inet addr: 223.1.2.4
  - transaction ID: 654
  - Lifetime: 3600 secs

DHCP request

- **DHCP request**
  - src: 0.0.0.0, 68
  - dest: 255.255.255.255, 67
  - Your inet addr: 223.1.2.4
  - transaction ID: 655
  - Lifetime: 3600 secs

DHCP ACK

- **DHCP ACK**
  - src: 223.1.2.5, 67
  - dest: 255.255.255.255, 68
  - Your inet addr: 223.1.2.4
  - transaction ID: 655
  - Lifetime: 3600 secs

**DHCP client-server scenario (explained)**

DHCP discover

- **DHCP discover**
  - src: 0.0.0.0, 68 (BOOTP client port)
  - dest: 255.255.255.255, 67 (server port)
  - Your inet addr: 0.0.0.0
  - transaction ID: 654

DHCP offer

- **DHCP offer**
  - src: 223.1.2.5, 67
  - dest: 255.255.255.255, 68
  - Your inet addr: 223.1.2.4
  - transaction ID: 654
  - Lifetime: 3600 secs

DHCP request

- **DHCP request**
  - src: 0.0.0.0, 68
  - dest: 255.255.255.255, 67
  - Your inet addr: 223.1.2.4
  - transaction ID: 655
  - Lifetime: 3600 secs

DHCP ACK

- **DHCP ACK**
  - src: 223.1.2.5, 67
  - dest: 255.255.255.255, 68
  - Your inet addr: 223.1.2.4
  - transaction ID: 655
  - Lifetime: 3600 secs
**Motivation:**
local/private network uses just one IP address as far as outside world is concerned:
- no need to be allocated range of addresses from ISP: - just one (or more?) IP address is used for all devices
- can change addresses of devices in local network without notifying outside world (but this violates end-to-end argument in Internet)
- can change ISP without changing addresses of devices in local network
- devices inside local net are not explicitly addressable and visible by outside world (a security plus).

**Implementation:**
NAT router must (do for)
- 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

**Example:**

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

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

3: Reply arrives dest. address: 138.76.29.7, 5001

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

**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
- NAT is a stopgap solution IPv6 enthusiastic person’s complaint
**What’s Wrong with the IP Addressing?**

- 32-bit IP address
  - Allows for \(4,294,967,296\) unique addresses
  - Addresses are grouped in a *class* (group), *most of which are wasted*.

- Addresses were arbitrarily handed out without regard to geographic location:
  - Class A stopped being handed out (running out).
  - Class B was near depletion (ex., 3000 out-of-65534 left)
  - Class C addresses were *overtaxing the Internet routing tables*.

- RFC 1338 introduced *subnetting* for fixing.

---

**Problem with Class C IP Address Assignment**

- Subnetting (RFC 950) – provides better address space granularity within each network
  \(\Rightarrow\) Solves problem – “running out of class B Address”

- However, it introduces another problem . . .
  - *more than 50 percent of the businesses were small and medium-sized. Class C addresses were needed*.

  - *Every class C address requires a routing table entry*  
  \(\Rightarrow\) *Internet routing tables explosion*

- *How to prevent this?*

  - **Solution** \(\Rightarrow\) Using “*Classless Inter-Domain Routing*”

---

**Flexible and Efficient IP Address Assignments**

- **Variable Length Subnet Masks (VLSM)**
  - placing a *variable-length* subnet mask on a single IP network number. – the first thought

- **Terminology:**
  - *Subnetting* \(\Rightarrow\) *... longer than ...*
  - *Supernetting* \~ applying a mask to an IP address that is *shorter than* its natural (default) mask.
  - *Address aggregation* \~ *summarizing contiguous blocks* of IP networks as one advertisement.

- **Classless Inter-Domain Routing (CIDR)**
  - *An advertisement mechanism that allows for advertising routes without regard to Class assignment*. The route could be identified by a *supernet* or by an *extended subnet mask*.

---

**Classless Inter-Domain Routing: A ISP’s Partition of IP addresses**

Network (network portion):

- get allocated portion of ISP’s address space:

*ISP’s block*  
\[11001000\ 00010111\ 0001\ 000\ 0000000\ 200.23.16.0/20\]

*Organizations 0*  
\[11001000\ 00010111\ 0001\ 000\ 0000000\ 200.23.16.0/23\]

*Organizations 1*  
\[11001000\ 00010111\ 0001\ 000\ 0000000\ 200.23.18.0/23\]

*Organizations 2*  
\[11001000\ 00010111\ 0001\ 000\ 0000000\ 200.23.20.0/23\]

*Organizations 3*  
\[11001000\ 00010111\ 0001\ 000\ 0000000\ 200.23.22.0/23\]

*Organizations 4*  
\[11001000\ 00010111\ 0001\ 000\ 0000000\ 200.23.24.0/23\]

*Organizations 5*  
\[11001000\ 00010111\ 0001\ 000\ 0000000\ 200.23.26.0/23\]

*Organizations 6*  
\[11001000\ 00010111\ 0001\ 000\ 0000000\ 200.23.28.0/23\]

*Organizations 7*  
\[11001000\ 00010111\ 0001\ 000\ 0000000\ 200.23.30.0/23\]

(divide into 8 blocks)

*CIDR* facilitates hierarchical routing
Hierarchical addressing: route aggregation

- Hierarchical addressing allows efficient advertisement of routing information - use a single network prefix to advertise multiple networks → route aggregation

ISP - A

Organization 0
200.23.16.0/23
Organization 1
200.23.18.0/23
Organization 2
200.23.20.0/23
Organization 7
200.23.30.0/23

ISP - B

"Send me anything with addresses beginning 200.23.16.0/20"

The rest of the world need not know that within the address block 200.23.16.0/20 there are 8 other organizations/networks.

ISP - B

Organization 1
200.23.16.0/23
Organization 2
200.23.20.0/23
Organization 7
200.23.30.0/23

Longest Prefix Matching: more specific routes

- ISP-B has a more specific route to Organization 1

ISP - A

Organization 0
200.23.16.0/23
ISP - B

Organization 1
200.23.16.0/23

"Send me anything with addresses beginning 200.23.16.0/20"

Internet

Organization 1
200.23.16.0/23
ISP - A

"Send me anything with addresses beginning 199.31.0.0/16"

ISP - B

Organization 1
200.23.16.0/23
ISP - A

"Send me anything with addresses beginning 200.23.18.0/23"

Internet

Organization 1
200.23.18.0/23
ISP - B

"Send me anything with addresses beginning 199.31.0.0/16 or 200.23.18.0/23"

Router's operation

Internet Control Message Protocol (ICMP)

- The internetworking facility does not guarantee successful delivery of every datagram
- Datagram may be discarded for a number of reasons:
  - lifetime expiration (in TTL field)
  - congestion (in Network)
  - bit error (being detected)
- Internet flow control allows gateways or receiving stations, or both, to limit the rate at which they receive data.
- The best approach would seem to be to send flow control packets. This can be done using the ICMP protocol (RFC 792)
- ICMP provides feedback about problems in the communication environment that require attention.

ICMP Packet Format

0 8 15 31
IP Header (20 bytes)

ICMP message

Type Code Checksum
Contents depends on type and code

Parameters

- Type ~ specify the ICMP message is a query or an error (e.g., type = 3, 4, 5, 11, 12)
- Code ~ distinct type of error or query
  Ex: type & code = 3 & 3 ⇒ port unreachable
- Checksum ~ covers the entire ICMP message (required)
- An ICMP error message is never generated in response to an ICMP error message.
**ICMP Message Formats**

- **Destination Unreachable (network, host, protocol, port)**
- **Time Exceeded** (TTL = 0 during transit, in *Traceroute*)
- **Parameter Problem** (bad IP header, e.g., improper options)
- **Source Quench** (elementary Flow control, reduce the transmission rate of sending-end)
- **Redirect** (Migrate to a better path for network or host)
  - **Echo Request** Testing the communication path, e.g., used in *ping* program
  - **Echo Reply**
- **Timestamp Request** (Sampling the delay characteristics of the internet)
- **Timestamp Reply**

**ICMP Message Types – Currently Used**

**Protocol Number (type) in IP Packet**

- 0 Reserved
- 1 *Internet Control Message Protocol (ICMP)*
- 2 *Internet Group Management Protocol (IGMP)*
- 3 *Gateway-to-Gateway Protocol (GGP)*
- 4 *IP (IP encapsulation)*
- 5 *Stream*
- 6 *Transmission Control Protocol (TCP)*
- 8 *Exterior Gateway Protocol (EGP)*
- 9 *Private Interior Routing Protocol*
- 17 *User Datagram Protocol (UDP)*
- 41 *IP Version 6 (IPv6)*
- 50 *Encap Security Payload for IPv6 (ESP)*
- 51 *Authentication Header for IPv6 (AH)*
- 89 *Open Shortest Path First*

Reference: Internet STD 2 — Assigned Internet Numbers. (or RFC 1700)
Ping - Packet INternet Groper

- Ping uses the ICMP **Echo and Echo Reply** messages to determine whether a host is reachable.

  - **ping loopback-address**: (Ex: 127.0.0.1) ~ verifies the operation of NIC and base TCP/IP software
  - **ping a-Host’s-IP-address** (self & a subnet neighbor) ~ verifies whether the physical network device can be addressed. (still be ok even without a router/gateway)
  - **ping gateway’s-IP-address** ~ verifies whether the gateway (your LAN’s router) works
  - **ping a-remote-host-name** (or DNS server’s IP directly) ~ verifies the operation of the name server

Routing Principle

- **Routing protocol**
  - **Goal**: determine "good" path (sequence of routers) thru network from source to DEST.

  - "Good" path:
    - typically means minimum **cost** path
    - Cost → hops/links/delay
      - link cost: delay, $ cost, or congestion level
        (other definitions are possible)
  - Graph abstraction for routing algorithms:
    - graph nodes are routers
      - link cost: delay, $ cost, or congestion level
    - graph edges are physical links

- **Ex:**  
  - 17 paths possible!

Routing Algorithm Classifications

- **Global or decentralized information?** ✓
  - **Global**:
    - All routers have complete topology (connectivity) and link cost info
    - Known as **link state** algorithms
  - **Decentralized**:
    - Router only knows physically-connected neighbors and link costs to neighbors
    - Iterative process of computation, exchange of info with neighbors
    - Known as **distance vector** algorithm

- **Static or dynamic?** ✓
  - **Static**:
    - Routes (routing information) change slowly over time
  - **Dynamic**:
    - Routes change more quickly
      - periodic update, or
      - in response to link cost changes
A Link-State Routing Algorithm

**Dijkstra’s algorithm**

- Net topology and link costs are 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 (candidate destinations)
  - gives **routing table** for that node
- Iterative: after k iterations, know least cost path to k destinations

**Notation:**

- **c(i,j):** link cost from node i to j. 
  - c(i,j) = infinite, if not direct neighbors (directly connected)
- **D(v):** current value of cost of path from source to dest. V
- **p(v):** predecessor node along path from source to v, that is next v
- **N:** set of nodes whose least cost path definitively known

---

**Dijsktra’s Algorithm**

1. **Initialization (step 0):**
   2. \( N = \{A\} \)
   3. for all nodes \( v \)
   4. if \( v \) adjacent to \( A \) /* directly attached */
   5. \( D(v) = c(A,v) \)
   6. else \( D(v) = \infty \)

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

---

**Dijsktra’s Algorithm - Example**

<table>
<thead>
<tr>
<th>Step</th>
<th>start N</th>
<th>D(B),p(B)</th>
<th>D(C),p(C)</th>
<th>D(D),p(D)</th>
<th>D(E),p(E)</th>
<th>D(F),p(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>2, A</td>
<td>5, A</td>
<td>1, A</td>
<td>( \infty )</td>
<td>( \infty )</td>
</tr>
<tr>
<td>1</td>
<td>AD</td>
<td>2, A</td>
<td>4, D</td>
<td>2, D</td>
<td>( \infty )</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ADE</td>
<td>2, A</td>
<td>3, E</td>
<td>4, E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ADEB</td>
<td>3, E</td>
<td>4, E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ADEBC</td>
<td>3, E</td>
<td>4, E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ADEBCF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- For each node, we got its predecessor along the least-cost path from the src, and so forth for each predecessor.
- Then the source node constructs its next-hop node to the given destination.

**FYI**

**Dijkstra’s Algorithm’s Complexity**

- **Algorithm complexity**: suppose \( n \) nodes (excluding the source node)
  - each iteration: need to check all nodes, \( w \), not in \( N \)
  - Need \( n(n+1)/2 \) comparisons: \( O(n^{*2}) \)
- Calls for more efficient implementations \( \Rightarrow O(n \cdot \log n) \) (A more sophisticated implementation of this algorithm, using a data structure known as a heap, can find the minimum in line 9 in logarithmic rather than linear time, thus reducing the complexity.)
**Dijkstra's Algorithm's Instability**

- **Oscillations** for asymmetry links
  - Ex: link cost $\equiv$ amount of carried traffic (e.g., delay)

(a) Initial routing

(b) B, C detect better path to A, CW

(c) B, C, D detect better path to A, CCW

(d) B, C, D detect better path to A, CW

**Distance Vector Routing Algorithm**

- **DV is . . .**
  - distributed: each node communicates only with directly-attached neighbors
  - Iterative: continues until no info exchanged between neighbors (self-terminating: no "signal" to stop)
  - asynchronous: nodes need not exchange info/iterate in lock step with each other

**Distance Table: an Example**

- Distance:
  
  \[ D^X(Y,Z) = \text{distance from } X \text{ to } Y, \text{ via } Z \text{ as next hop} \]

  \[ D^X(Y,Z) = c(X,Z) + \min_w \{D^X(Y,w)\} \]

**Routing Table – from Distance Table**

<table>
<thead>
<tr>
<th>Destination</th>
<th>Cost to Destination (E)</th>
<th>Outgoing Link to Use, Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1 14 5</td>
<td>A, A,1</td>
</tr>
<tr>
<td>B</td>
<td>7 8 5</td>
<td>B, D,5</td>
</tr>
<tr>
<td>C</td>
<td>6 9 4</td>
<td>C, D,4</td>
</tr>
<tr>
<td>D</td>
<td>4 11 2</td>
<td>D, D,2</td>
</tr>
</tbody>
</table>
**Distance Vector Routing: Overview**

- **Iterative, asynchronous**: each local iteration caused by:
  - local link cost change
  - message from neighbor: its least cost path change from neighbor
- **Distributed**:
  - each node notifies neighbors only when its least cost path to any destination changes
  - neighbors then notify their neighbors if necessary

**Each node:**

- wait for (change in local link cost of msg from neighbor)
- **recompute** distance table
- if least cost path to any dest has changed, notify neighbors

---

**Bellman-Ford Algorithm**

**At all nodes, \( X \):**

1. **Initialization:**
2. for all adjacent nodes \( v \):
3. \( D^{X}(\ast,v) = \infty \)
4. \( D^{X}(v,v) = c(X,v) \)
5. for all destinations, \( y \)
6. send \( \min_w D^{X}(y,w) \) to each neighbor

/* the " \( \ast \) " operator means "for all rows" */

---

**Bellman-Ford Algorithm (cont.)** (at each node \( X \))

8. **loop**
9. **wait** (until it sees a link cost change to neighbor \( V \)
10. or until it receives an update from neighbor \( V \))
11. **if** ( \( c(X,V) \) changes by an amount of \( d \))
12. /* change cost to all dest's via neighbor \( v \) by \( d \) */
13. /* note: \( d \) could be positive or negative */
14. **for all destinations** \( y \):
15. \( D^{X}(y,V) = D^{X}(y,V) + d \) \( \leftarrow \text{recomp.} \)
16. **else if** ( update received from \( V \) w.r.t. destination \( Y \))
17. /* shortest path from \( V \) to some \( Y \) has changed */
18. **for the single destination** \( y \):
19. \( D^{X}(Y,V) = c(X,V) + \text{newval} \) \( \leftarrow \text{recomp.} \)
20. **else if** we have a new \( \min_w D^{X}(Y,w) \) for any destination \( Y \)
21. **send new value of** \( \min_w D^{X}(Y,w) \) to all neighbors
22. **forever**

---

**Distance Vector Algorithm: an Example**

- \( \bigcirc \) ~ current minimum path cost to a destination
- \( \bigcirc \) ~ a new minimum cost has been computed
Distance Vector Algorithm: example

- update received from \( V \) w.r.t. destination \( Y \)

\[
\begin{align*}
D^X(Y,Z) &= c(X,Z) + \min_w (D^Z(Y,w)) \\
&= 7 + 1 = 8
\end{align*}
\]

Triggered by \( Z \):

\[
D^X(Y,Z) = c(X,Z) + \min_w (D^Z(Y,w))
\]

need to be recomputed (B-F algorithm line 21)

Triggered by \( Y \):

\[
D^X(Z,Y) = c(X,Y) + \min_w (D^Z(Y,w))
\]

Comparison of LS and DV algorithms

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

2. **Robustness**:
   - **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 indirectly

3. **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

Realistic Routing Issues in Internet

- Our routing study thus far - idealization
  - all routers identical
  - Network structure is "flat"
    ... Nooooo! It is not true in practice.

- **scale**:
  - with more than 50 million destinations and routers:
    - can’t store all dest’s in routing tables!
    - routing table exchange by LS or DV would swamp links
    - No more BW left for sending data packets

- **administrative autonomy**:
  - internet = network of networks
  - each network administrator may want to control routing in its own network

Hierarchical Routing

- **Solution**: aggregate routers into regions “autonomous systems” (AS)
- Routers in same AS run same routing protocol (DV or LS)
  - “intra-AS” routing protocol
- Routers in different ASs can run different intra-AS routing protocols via gateway routers

(Idea: next page)
Hierarchical - Intra-AS and Inter-AS RP

Routing in the Today’s Internet

- The Global Internet consists of Autonomous Systems (AS) interconnected with each other:
  - **AS**: comprised of multiple networks/routers sharing all routing information under a single administration (i.e., routing protocol)
  - **Stub AS**: small corporation: one connection to other AS’s
  - **Multihomed AS**: large corporation (no transit) with multiple connections to other AS’s
  - **Transit AS**: service provider, hooking many AS’s together

- Two-level routing:
  - **Intra-AS**: administrator responsible for choice of routing algorithm within network ~ RIP, OSPF
  - **Inter-AS**: unique standard for inter-AS routing ~ BGP

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)

- Adopted by Xerox Network System architecture firstly
- Included in BSD-UNIX Distribution in 1982
- Use **Distance vector** algorithm
  - Distance/cost metric: # of hops (max = 15 hops)
  - *Can you guess why?*
- Distance vectors: exchanged every 30 sec via Response Message (called **RIP advertisement**) with neighboring routers
  - Each advertisement contains up to 25 destination networks
- RIPv1 ~ RFC 1058; RIPv2 ~ RFC 2453 (allows route aggregation)
- For classful addressing, no VLSM supporting
**RIP Version 2 – Packet Format**

<table>
<thead>
<tr>
<th>Command</th>
<th>Version</th>
<th>Unused</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Family Identifier</td>
<td>Route Tag</td>
<td></td>
</tr>
<tr>
<td>Net 1 address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subnet mask (v2 only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Next-Hop IP Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metric</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address Family Identifier</td>
<td>Route Tag</td>
<td></td>
</tr>
<tr>
<td>Net 2 address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subnet mask</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Next Hop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metric</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(UDP Data) [DA SA TF IP Hdr UDP Hdr UDP Data CRC]

(use multicast instead of broadcasting in RIP v1)

**How does the RIP Advertisement Work?**

**Advertisement sent from router A**

<table>
<thead>
<tr>
<th>Destination Network</th>
<th>Next Router</th>
<th>Num. of hops to dest.</th>
</tr>
</thead>
<tbody>
<tr>
<td>z</td>
<td>C</td>
<td>4</td>
</tr>
<tr>
<td>w</td>
<td>--</td>
<td>1</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 D updated to (after rerunning DV algorithm) …

**Routing table in D**

<table>
<thead>
<tr>
<th>Destination Network</th>
<th>Next Router</th>
<th>Num. of 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>z</td>
<td>A</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**RIP: Link Failure and Recovery**

- If no advertisement heard after 180 sec (missing 6 updates)
  - **Route-timeout timer**
  - routes via neighbor invalidated (no longer reachable)
  - new advertisements sent to neighbors
  - neighbors in turn send out new advertisements (if tables changed)
  - link failure info quickly propagates to entire net

- Route-flush timer: removed a route from routing table after being declared invalid for 240 sec.
- Poison reverse used to prevent ping-pong loops (infinite distance = 16 hops)
RIP Table processing

- RIP routing tables managed by **application-level** process called "routed (route de)" (daemon) - executes RIP protocol
- Advertisements sent in UDP packets, periodically repeated
- Use "netstat -rn" to view routing table

```
router:
+---------+---------+-----+-----+-----+
| 127.0.0.1| 127.0.0.1| UH | 0   | lo0 |
| 192.168.2.5| 192.168.2.5| U  | 2   | fa0 |
| 193.55.114.6| 193.55.114.6| U  | 3   | le0 |
| 192.168.3.5| 193.55.114.6| U  | 2   | qaa0|
| default (0.0.0.0)| 193.55.114.129| UG | 0   | 1e0 |
```

- Three attached class C networks (LANs)
- Router only knows routes to attached LANs
- Default router used to "go up"
- Route multicast address: 224.0.0.0
- Loopback interface (for debugging)

OSPF (Open Shortest Path First)

- "Open" → publicly available ← opposed to Cisco's EIGRP
- RFC 2178, 2328
- Uses **Link State** algorithm (LS - the topology of the AS) with
  - LS packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra's lease-cost 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
  - Supports VLSM and CIDR

OSPF's "Advanced" Features (not in RIP)

- **Security**: all OSPF messages authenticated (to prevent malicious intrusion - injecting incorrect info into routing tables)
- **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 database as OSPF
  - OSPF allows to **structure an AS hierarchically** within a large domains (to ease routing)
Metrics (cost)

- Cost = a configurable parameter with the output side of each router interface
- Reference RFC 1253 (OSPF v.2 MIB)
- Metric = $10^8$ / interface speed (recommended)

**Examples:**
- 100 Mbps 1
- 10 Mbps 10
- E1 (2.048 Mbps) 48
- T1 (1.544 Mbps) 65
- 64 kbps 1562
- 19.2 kbps 5208
- 9.6 kbps 10416

Generic Packet Formula

<table>
<thead>
<tr>
<th>Version</th>
<th>Type</th>
<th>Packet Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Area ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td>Authentication Type</td>
<td></td>
</tr>
<tr>
<td>Authentication</td>
<td>LSA Specific</td>
<td></td>
</tr>
<tr>
<td>1 – Hello, 2 – DB Description, 3 – LS Request, 4 – LS Update, 5 – LS Ack</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DA</th>
<th>SA</th>
<th>TF</th>
<th>IP Header Protocol ID 89</th>
<th>IP Data</th>
<th>CRC</th>
</tr>
</thead>
</table>

Hierarchical OSPF

- Two-level hierarchy in an OSPF AS: local area and backbone
  - Area routers broadcasts LS advertisements to routers inside the area
  - Each router has detailed area topology; but the details are invisible to the outside routers
  - Each router only knows 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.
**Inter-AS Routing Protocol – BGP4**

- BGP (Border Gateway Protocol): *the* de facto standard
- RFC 1771-1773
- Supports CIDRized address (e.g., 140.124.115.0/24)
- Use **Path Vector protocol**:
  - similar to Distance Vector protocol
  - each Border Gateway broadcast to neighbors (peers) *entire path* (i.e., sequence of AS's) to destination
  - BGP routes/paths to networks (ASs), not individual hosts/routers
  - E.g., Gateway X may send its path to dest. Z:

  $\text{Path (X,Z)} = X, Y_1, Y_2, Y_3, \ldots, Z$

**IPv6 – IP Version 6**

- **Initial motivation**: 32-bit address space completely allocated by 2008(class B) and 2018(class C).
- **Additional motivation**:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS
  - new “anycast” address: route to “best” of several replicated servers
- **IPv6 datagram format**: RFC 2460
  - fixed-length 40 byte header + extension headers
  - no fragmentation allowed
- **IPv6 Header**
  - **Priority**: identify priority among datagrams in flow
  - **Flow Label**: identify datagrams in same “flow.”
    (concept of “flow” not well defined).
  - **Next header**: identify extension or upper layer protocol
IPv6 Next Header Field

- Next header values (Extension header)
  0 ~ Hop-by-Hop Options Header
  43 ~ IPv6 Routing Header
  44 ~ IPv6 Fragment Header
  50 ~ Encapsulating Security Payload *
  51 ~ IPv6 Authentication Header *
  59 ~ No Next Header
  60 ~ Destination Options Header
- Upper-layer Protocol type identifier (new)
  45 ~ Interdomain Routing Protocol
  46 ~ Resource Reservation Protocol
  58 ~ IPv6 ICMP Packet

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

IPv4 and IPv6 Differences in Header

- IPv6 is a static 40 bytes in length
  - not variable any longer; checksum being removed
- IPv6 allows for jumbogrames (jumbo datagram > 65,536 bytes)
  - allows for various network attachments (RFC 2146)
- IPv6 supports Extension header (concatenated headers)
  - IPv4’s Total length field is replaced with payload length
  - IPv4 TTL field is replaced with the hop limit
  - Many IPv4 options were moved to independent protocols

* Question:
  How many fields have been suppressed, renamed, and newly added? **Ans.: 6,3,2**
Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneously
  - no "flag days"
  - How will the network operate with mixed IPv4 and IPv6 routers?
- Two proposed basic approaches:
  - **Dual Stack**: some routers with dual stack (v6, v4) can "translate" between v4 and v6 formats
  - **Tunneling**: IPv6 carried as payload in IPv4 datagram among IPv4 routers