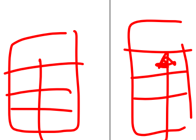
Network Layer

application | msg ↓ segment ↓ datagram
transport
network

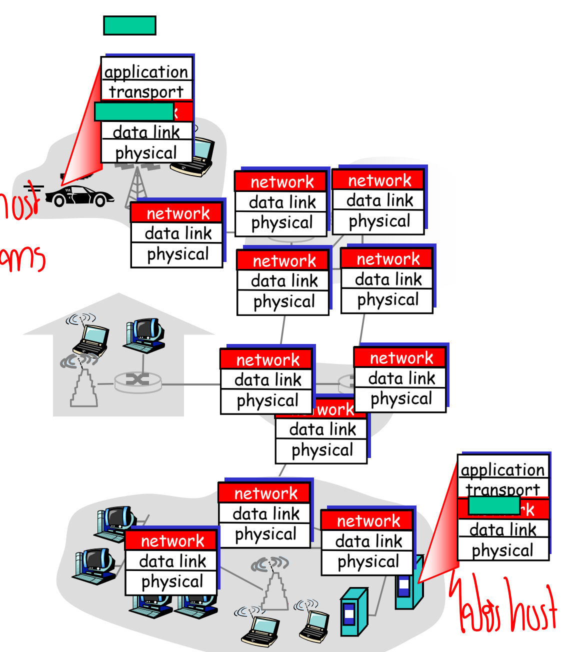
Computer Networks

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



จาก host ต้นทาง



ไปยัง host ปลายทาง

③ connection setup → มีเฉพาะบางระบบเครือข่ายเท่านั้น เช่น framrelay, atm เป็นต้น

Two Key Network-Layer Functions

งานของ network

① o *Forwarding Function:* move packets from router's input to appropriate router output

decide where to go

② o *Routing Function:* determine route taken by packets from source to destination

network → f: พิจารณาเส้นทางให้ datagram

data

phy

→ พิจารณาว่าจะส่งออกทาง interface ไหน

ส่งรวมกับ layer ด้านล่าง

o <u>analogy:</u>

o routing: process of planning trip from source to destination

→ วางแผนการเดินทาง → ใช้เส้นทางไหน → ไปถนนเส้นไหน

o forwarding: process of getting through single interchange

→ IP host ต้นทาง, IP host ปลายทาง

→ จะต้องส่งออกทาง interface ไหน

f: routing

→ เวลาที่เดินทางมาถึงทางแยก ไปทางแยกไหน

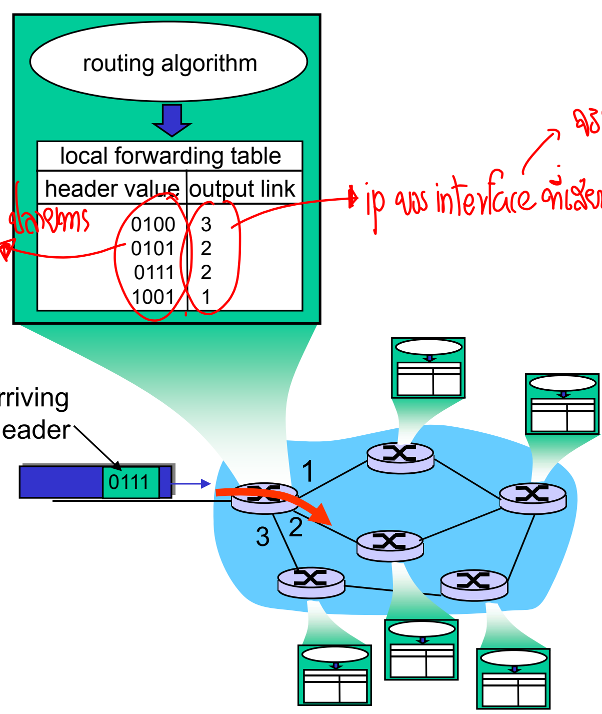
o *Routing Algorithms*

table

ใช้ในการ forward ส่วนที่เป็น datagram

ตาราง forwarding table และ routing table

Interplay between routing and forwarding



จะบอกเรื่องเกี่ยวกับ host ปลายทาง

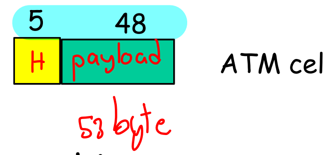
ในรูปแบบ network IP

ip ของ interface ที่เชื่อมอยู่

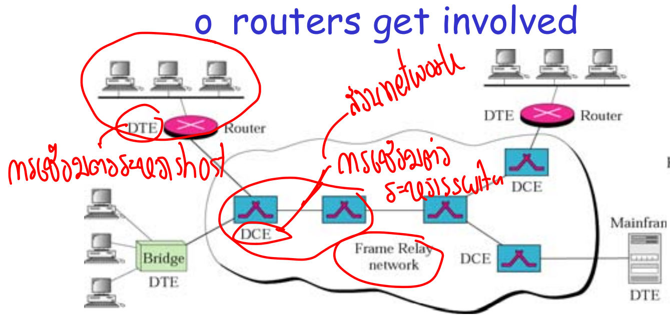
จริงๆ ต้องรวมการกำหนด ip ให้เส้น

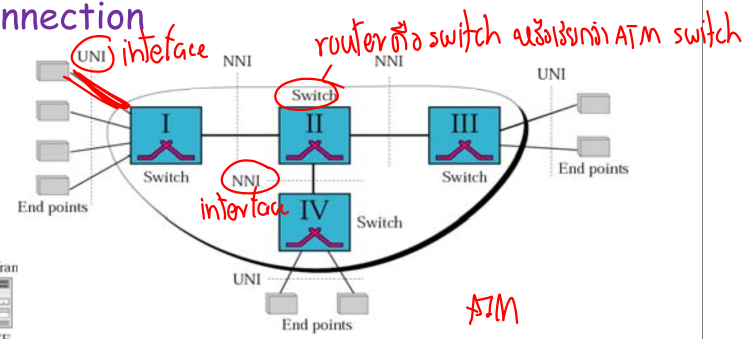
มีเฉพาะบางเครือข่าย

Connection setup



- 3rd important function in *some* network architectures: ปัจจุบันไม่ค่อยมีแล้ว
  - Asynchronous Transfer Mode (ATM), Frame Relay, X.25
  - Backbone networks in Internet are Switched WAN เครือข่ายที่ทำหน้าที่เป็น
  - Switched WAN : wide area network that cover large area and provide access at several point to users
- before datagrams flow, two end hosts *and* intervening routers establish virtual connection
  - routers get involved





DTE : Data Terminating Equipment
DCE : Data Circuit Equipment

UNI : User-to-Network Interface
NNI : Network-to-Network Interface

ในส่วนของ service ที่อยู่ใน network layer แยกเป็น service ที่เกี่ยวกับ connection จะ

Network layer connection and connection-less service

- Network layer Connection service แบบมี connection
  - Virtual Circuit เราเรียกเครือข่ายแบบนี้ว่า ↳ ตัวอย่าง f: ตอนที่ 3 f:
    - Connection setup
    - Forwarding
    - Routing

    พวก ATM, framerelay ↳ อ้างอิงหมายเลข VC ที่ถูกกำหนดไว้
- Network layer Connection-less service แบบนั้น
  - Datagram network เราเรียกเครือข่ายแบบนี้ว่า
    - Forwarding ↳ อ้างอิงหมายเลข ip
    - Routing

  ถ้าใน network : เครือข่ายใดก็ตามที่ใช้ service แบบนั้น จะต้องใช้แบบนั้นเท่านั้น
- Analogous to transport-layer services, but:
  - service: host-to-host
  - no choice: network provides one or the other
  - implementation: in network core

transport ของ host
network ที่ network core

network: host ต้นทาง ⟷ host ปลายทาง
transport: process ต้นทาง ⟷ process ปลายทาง

ใน transport: connection oriente service : TCP
connectionless ... : UDP

ใน transport ↳ เลือกใช้ protocol ตามความเหมาะสม

คล้ายกับใน TCP

ใน ATM ใช้การแบ่ง bandwidth → ไม่ได้เป็นการจัดสรรแบบ ?

เวลาเรียกเส้นว่า virtual circuit

กระบวนการ ① setup phase
② data transfer phase
③ tear down phase

Virtual circuits (VC)

ใช้หลักการ)

"source-to-dest path behaves much like telephone circuit" → ข้อมูลต้องรู้การสร้าง circuit ไว้ก่อนแล้ว → ข้อมูลส่งไปทางเดียว แบบเรียงลำดับ

- performance-wise
- network actions along source-to-dest path

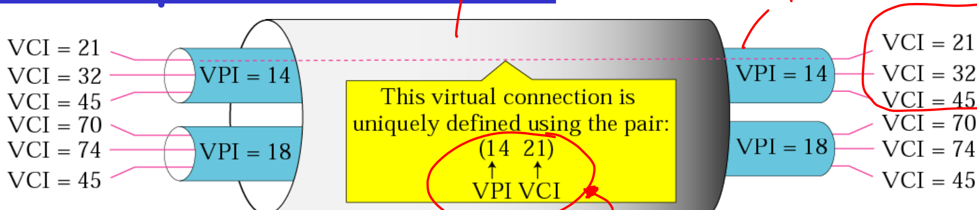
- call setup, teardown for each call *before* data can flow
  ข้อมูลเดินทางกำหนด
- each packet carries VC identifier (VCI) (not destination host address)
  มีการจดจำสถานะ
- *every* router on source-dest path maintains "state" for each passing connection
- link, router resources (bandwidth, buffers) may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

I : identifier

link สื่อสาร

VC implementation

VP

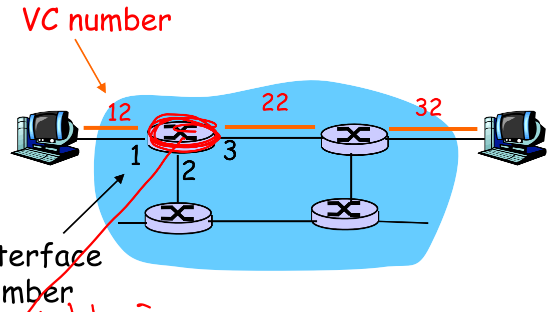


เลขสื่อสารต้องรู้

- VC consists of:
  - Path from source to destination (Virtual Path: VP)
  - VC numbers, one number for each link along path
  - entries in forwarding tables in routers along path
- packet belonging to VC carries VC number (rather than destination address)
- VC number can be changed on each link.
  - New VC number comes from forwarding table

Note that : virtual connection is defined by a pair of numbers: VPI and VCI.

Forwarding table



Forwarding table in northwest router:

table นี้

ตารางที่สร้างขึ้นมาตอน setup phase

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

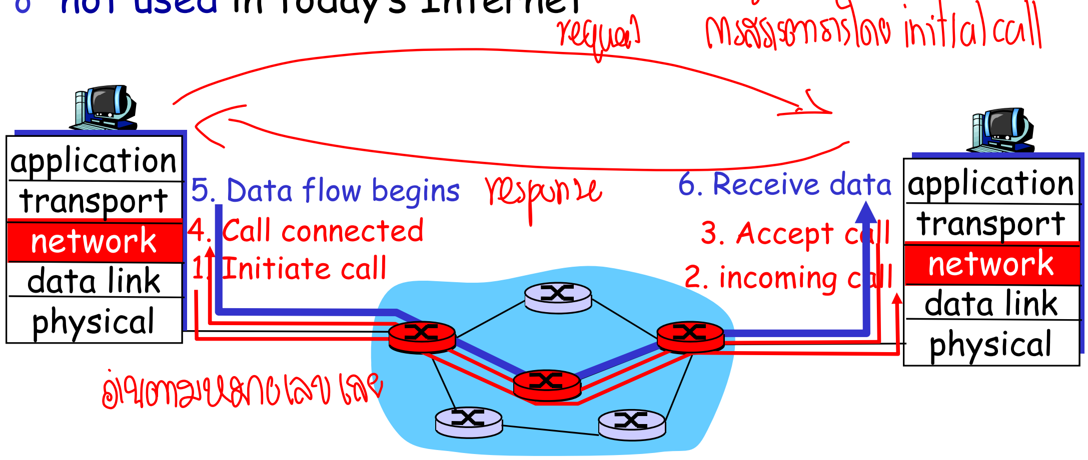
Routers maintain connection state information!

Virtual circuits: signaling protocols

- used to setup, maintain teardown VC
- used in ATM, Frame-Relay, X.25
- **not used** in today's Internet

ใน setup phase

การเชื่อมต่อหรือ initial call



อยู่ใน service ประเภท conectionless → เป็นเครือข่ายที่เรียกว่า datagram

Datagram networks

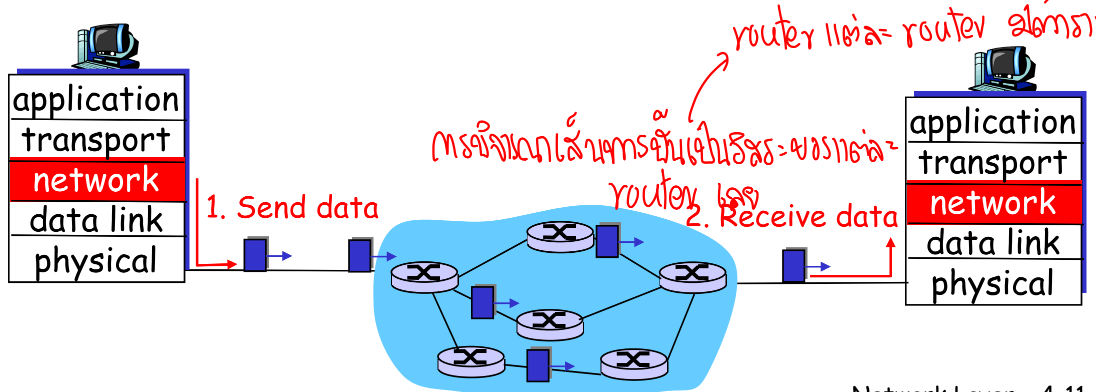
ข้อมูลจาก ip แทน

- no call setup at network layer → ไม่มีการ setup → router ไม่มีสถานะ
- 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 → ข้อมูลอาจไปไม่เหมือนกัน

ใช้ สร้างโดยตัวเอง

router แต่ละ router มีตารางของตัวเอง

การเลือกเส้นทางขึ้นเป็นระยะๆของแต่ละ router และ



กระโดดไป slide switching.pdf ก่อน

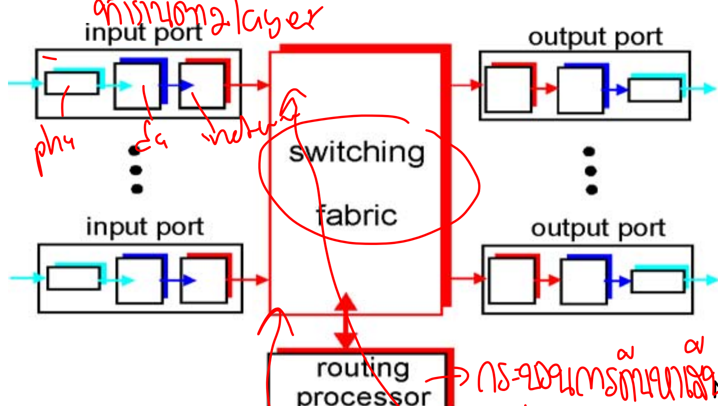
What is inside the router? : Router Architecture Overview

algorithm → link stage
↳ distance vector

- Two key router functions:
- run routing algorithms/protocol
  - Route Information Protocol (RIP),
  - Open Shortest Path First (OSPF),
  - Border Gateway Protocol (BGP)

  protocol ← สื่อสาร
- *forwarding* datagrams from incoming to outgoing link

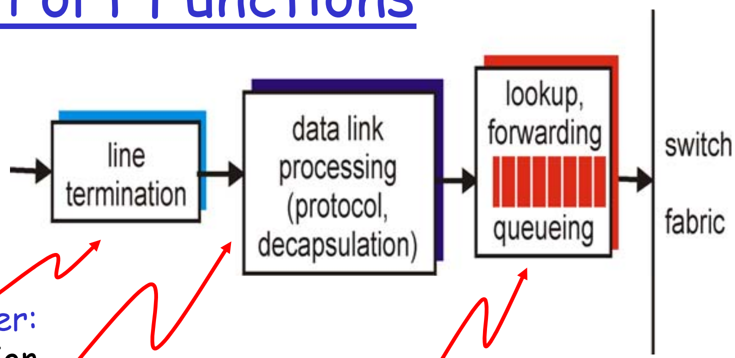
ทำงานบน 2 layer



กระบวนการที่นำมาสร้างตาราง

เหมือนตารางที่ไปดูข้อมูล

Input Port Functions



Physical layer:
bit-level reception

Data link layer:
e.g., Ethernet

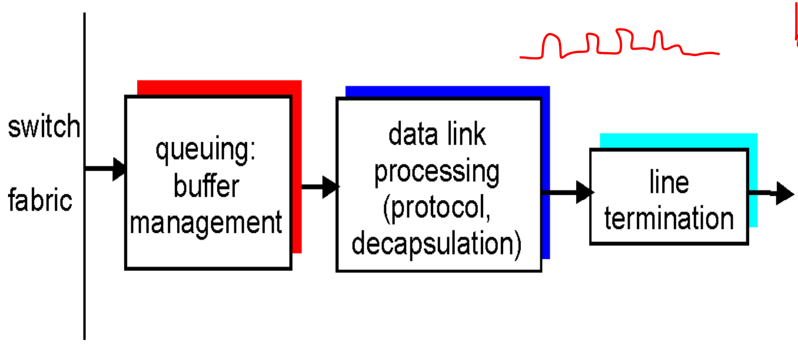
- **Decentralized switching**:
- given datagram dest., lookup output port using forwarding table in input port memory
- Goal: complete input port processing at 'line speed'
- Queuing: if datagrams arrive faster than forwarding rate into switch fabric

Output Ports

H, datagram

H, T frame

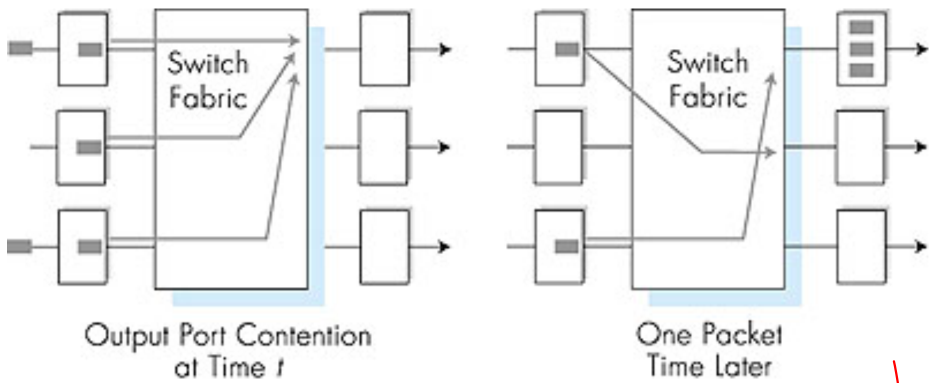
bit



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

ง่ายสุดก็มาก่อนส่งก่อน และอื่นๆ อีกมากมาย

Output port queueing



ถ้าการกำหนดขนาด buffer เผื่อ packet ล้นถึง

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

ไม่มีพื้นที่พอในการเก็บ output

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

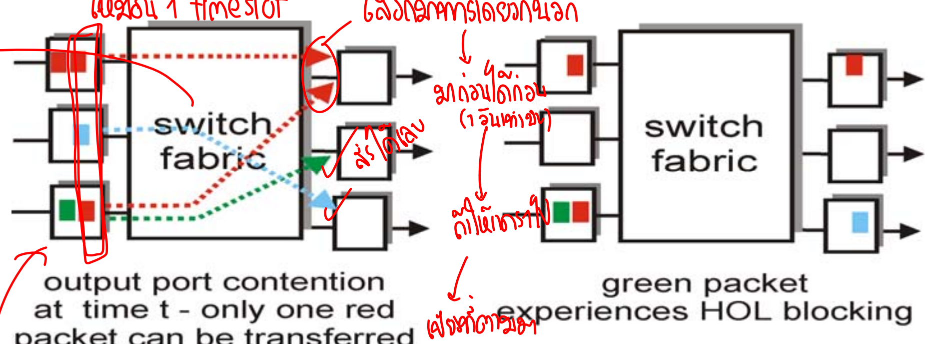
→ เพราะเกิด delay → ทำให้เกิด loss

- *queueing delay and loss* due to *input buffer overflow!*

ถ้าสวิช ทำงานช้า กว่าการเข้ามาของข้อมูล ก็ต้องมี buffer

เหมือน 1 timeslot

เสียเวลาทำการเดียวกันอีก



มาก่อนได้ก่อน (1 อันเท่านั้น)

ส่งได้เลย

ถ้าไม่ได้เอาไป

เพราะที่ตามมา

มี buffer เหมือนกัน

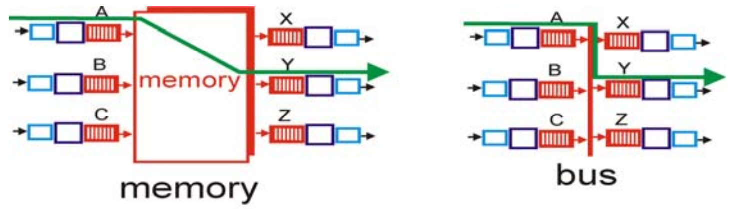
เสียโอกาสในการส่ง ทั้งๆที่เส้นทางว่าง

Three types of switching fabrics

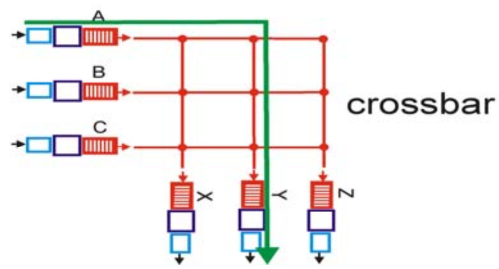
- Packet was copied from input port into processor memory
- routing processor extracted destination address, looked up appropriate output port in forwarding table
- Cisco's Catalyst 8500 series switches (10 Gbps)

มีหลายแบบมาก

- Input port transfer packet directly to output port over shared bus
- One packet at a time can be transferred over bus
- switching bandwidth of router is limited to bus speed
- Cisco 5600 – switches packets over 32 Gbps bus

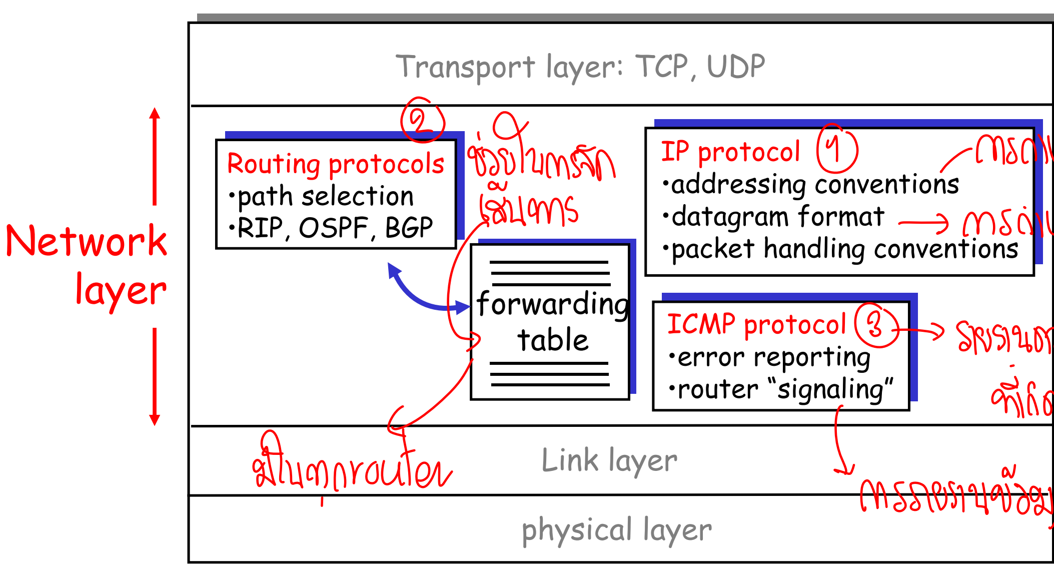


- Crossbar switch is interconnection network Consisting of 2n buses
- Cisco 12000 family switches use interconnection network, providing up to 60 Gbps through switching fabric



The Internet Network layer → มี 3 ส่วนที่สำคัญ

Host, router network layer functions:



การกำหนด address

→ การกำหนดรูปแบบ

ความแตกต่างของ Field

→ รายงานความผิดพลาดที่เกิดจากการส่งข้อมูล

การรายงานข้อมูลต่างๆ ของ router ที่วิ่ง

IP datagram format

IPV4

ขนาดเริ่ม 20 byte → 60 byte

32 bits

4 bits	4 bits	8 bits	16 bits	
ver	head. len	type of service	length	
16-bit identifier			3 bits flgs	13 bits fragment offset
time to live (8 bits)		upper layer (8 bits)	16 bits header checksum	
32 bit source IP address				
32 bit destination IP address				
Options (if any)				
data (variable length, typically a TCP or UDP segment)				

- IP protocol version number
- header length (bytes)
- "type" of data
- max number remaining hops (decremented at each router)
- upper layer protocol to deliver payload to (TCP=6, UDP=17)
- total datagram length (bytes)
- for fragmentation/ reassembly
- E.g. timestamp, record route taken, specify list of routers to visit.

จำนวน hop ที่เหลือ(ครั้ง) ผ่านไปได้

protocol เหนือ layer นี้

detect false

ใช้ในการ 3 flag นี้

max = 40 byte

- how much overhead with TCP?
- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + application layer overhead

แต่ละเครื่องมี MTU ที่ต่างกัน

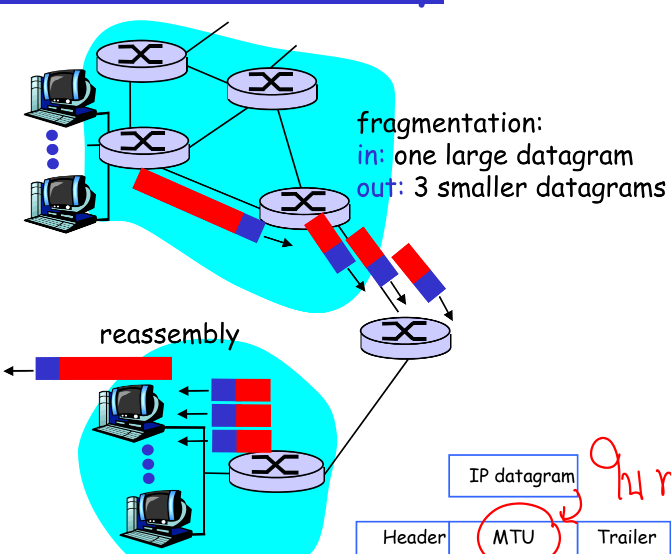
การส่งจาก host ต้นทางไปยังปลายทาง อาจจะผ่านหลาย network

การแตกของหรือการประกอบกลับ datagram

MTU = Maximum Transmission Units

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 datagram

Header	MTU	Trailer

ใน network

Token Ring (16 Mbps)	– 17,914 bytes
Token Ring (4 Mbps)	– 4,464 bytes
Ethernet	**– 1,500 bytes**
Point-to-Point Protocol (PPP)	–296 bytes

ขนาดการ MTU มากที่สุดที่บรรจุได้ใน frame ข้อมูล

C → MTU 4794 → X → MTU 1500 → ประกอบ → O

ถ้าขนาดมากกว่า จะต้องทำการแตกของ ก็จะส่งไปต่อได้

พอ host ปลายทางได้แล้ว host ปลายทางจะประกอบกลับมาเป็น datagram อีกครั้งหนึ่ง

IP Fragmentation and Reassembly

Example

ต้นทางส่ง: ❑ 4000 byte datagram

ผ่านตัวกลางซึ่งมี: ❑ MTU = 1500 bytes

(no option) header(20) + data(3980)

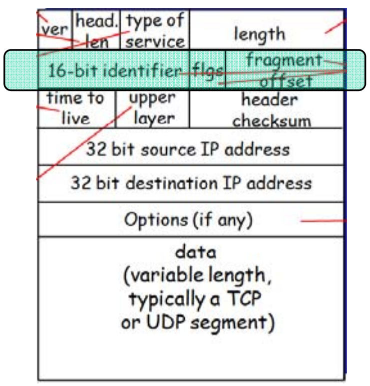
ลำดับของ datagram ย่อยกรณีที่มีการแยก

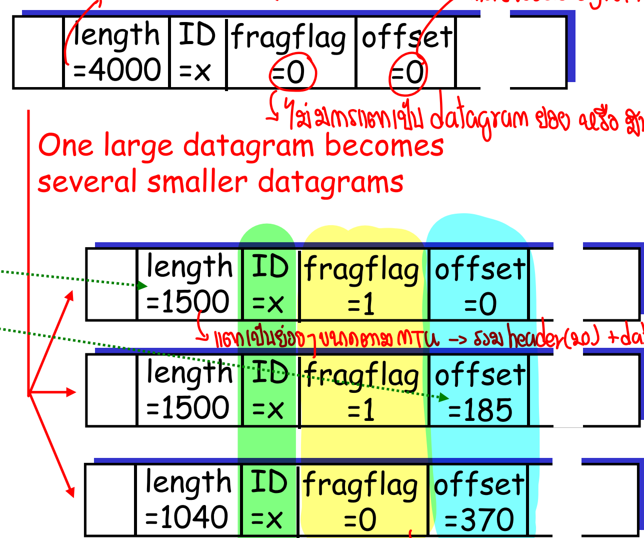
One large datagram becomes several smaller datagrams

ไม่มีการแตกเป็น datagram ย่อย หรือ เป็นตัว datagram สุดท้าย

1480 bytes in data field

offset = 1480/8





แตกเป็นย่อยๆ ขนาดตาม MTU -> รวม header(20) + data(1480)

ทุกตัว offset ต่างกัน

1480/8 = 185

ขนาดข้อมูลสะสมหารด้วย /8

มีการแยก

0 คือ ไม่มีการแยก หรือตัวสุดท้าย

id เหมือนกัน

การจะระบุตัว host ต้นทาง กับ ปลายทาง ต้องมี global address สากล

Network Layer : Logical Addressing

ip address

- Communication at network layer is host-to-host
- Computer somewhere in the world need to communicate with another computer somewhere else in the world through Internet
- Packet transmitted by sending computer may pass through several LANs or WANs before reaching destination computer
- We need *global addressing scheme* called *logical addressing*
- Today, we use the term *IP address* to mean a logical address in network layer of TCP/IP protocol suite

IP Addresses

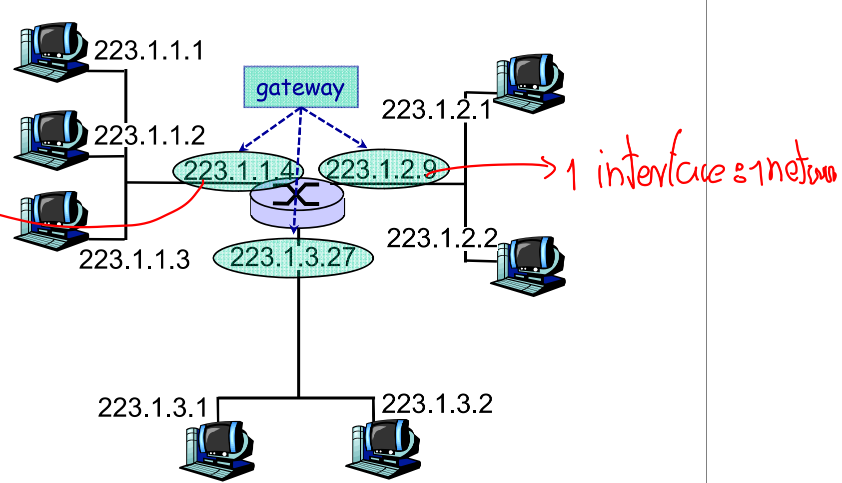
- The Internet address are 32 bits in length
  - Address space is 2^{32} or 4,294,967,296
  - These addresses are referred to as IPv4 (IP version 4) addresses or simply IP address
- The need for more addresses motivated a new design of the IP layer called new generation of IP or IPv6 (IP version 6)
  - The Internet uses 128-bit addresses that give much greater flexibility in address location
  - These addresses are referred to as IPv6 (IP version 6) address

ส่วนที่เชื่อมกับ router

IPv4 Addressing: introduction

interface ⇒ ส่วนที่เชื่อมต่อ

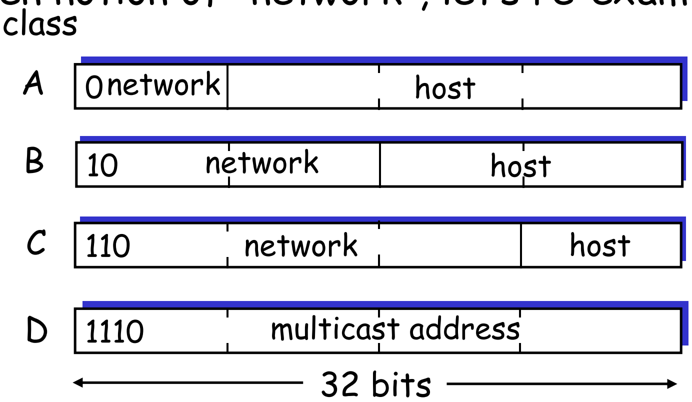
- IPv4 address: 32-bit identifier for host, router *interface*
- *Interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one interface
  - IP addresses associated with each interface



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

IP Addresses "class-full" addressing:

given notion of "network", let's re-examine IP addresses:

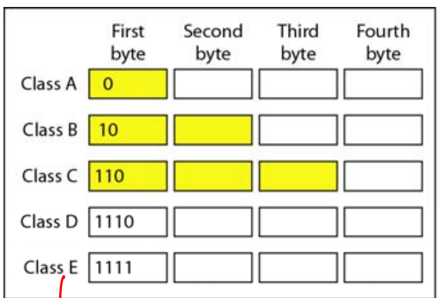


1.0.0.0 to 127.255.255.255 0

128.0.0.0 to 191.255.255.255 10

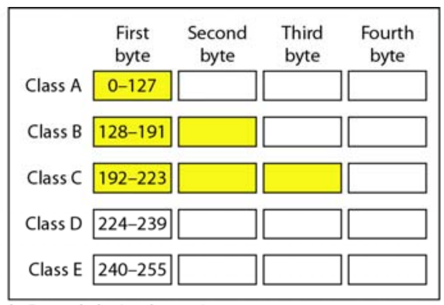
192.0.0.0 to 223.255.255.255 110

224.0.0.0 to 239.255.255.255 1110



a. Binary notation

reserve



b. Dotted-decimal notation

Class	*Number of Blocks*	*Block Size*	*Application*
A	128	16,777,216	Unicast
B	16,384	65,536	Unicast
C	2,097,152	256	Unicast
D	1	268,435,456	Multicast
E	1	268,435,456	Reserved

Class	Network Octets (blanks in the IP address are used for octets identifying hosts)	Total Number of Possible Networks or Licenses (จำนวนของnetwork)	Host Octets (blanks in IP address are used for octets identifying networks)	Total Number of Possible IP Addresses in Each Networks (จำนวนของ ip ต่อ 1 network)
A	0.__.__.__ to 127.__.__.__	128	__.0.0.1 0.0.0 to __.255.255.254 (255 -> broad cas)	ip netw 16,777,214 256x256x254
B	128.0.__.__ to 191.255.__.__	64x256 16,384	__.__.0.1 to __.__.255.254	65,534
C	192.0.0.__ to 223.255.255.__	32x256x256 2,097,152	__.__.__.1 to __.__.__.254	254

Address for Private Networks

ช่วง private ip ของ class ต่างๆ

	Range			Total
Class A	10.0.0.0	to	10.255.255.255	2^{24}
Class B	172.16.0.0	to	172.31.255.255	2^{20}
Class C	192.168.0.0	to	192.168.255.255	2^{16}

ip → global

private → ที่ไม่ซ้ำ ๆ ip ใช้ได้หลาย host

internet

global

priv priv

อุปกรณ์ที่อยู่ใน subnet เดียวกัน สามารถส่งข้อมูลหากันได้โดยไม่ผ่าน router ได้เลย

Subnets

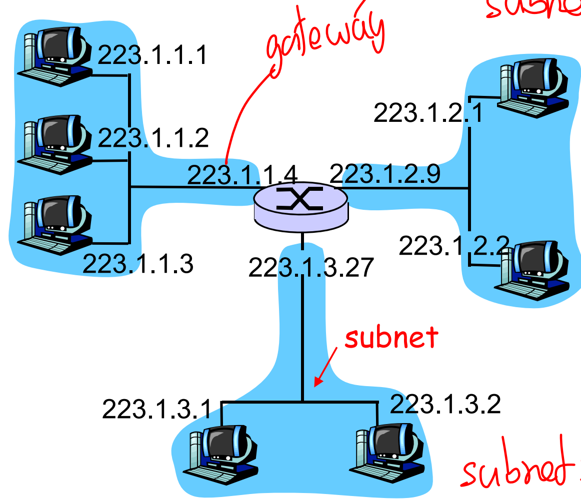
- IP address:
  - subnet part (high order bits)
  - host part (low order bits)
- *What's a subnet ?*
  - device interfaces with same subnet part of IP address
  - can physically reach each other without intervening router

subnet 1

gateway

ทางออกของแต่ละ subnet

subnet 2

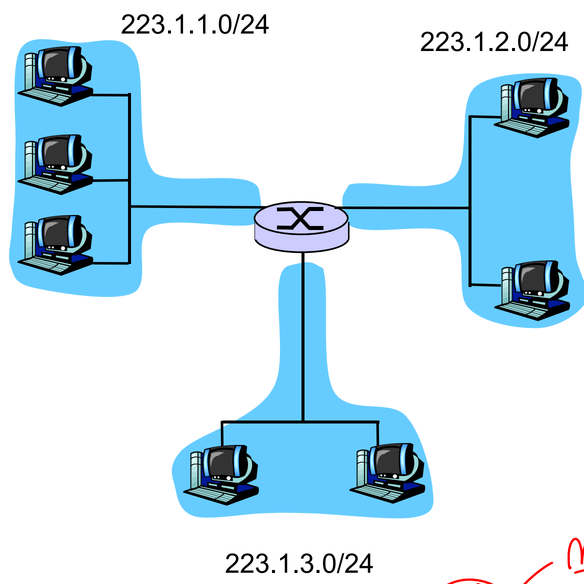


subnet 3

network consisting of 3 subnets

Class C

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

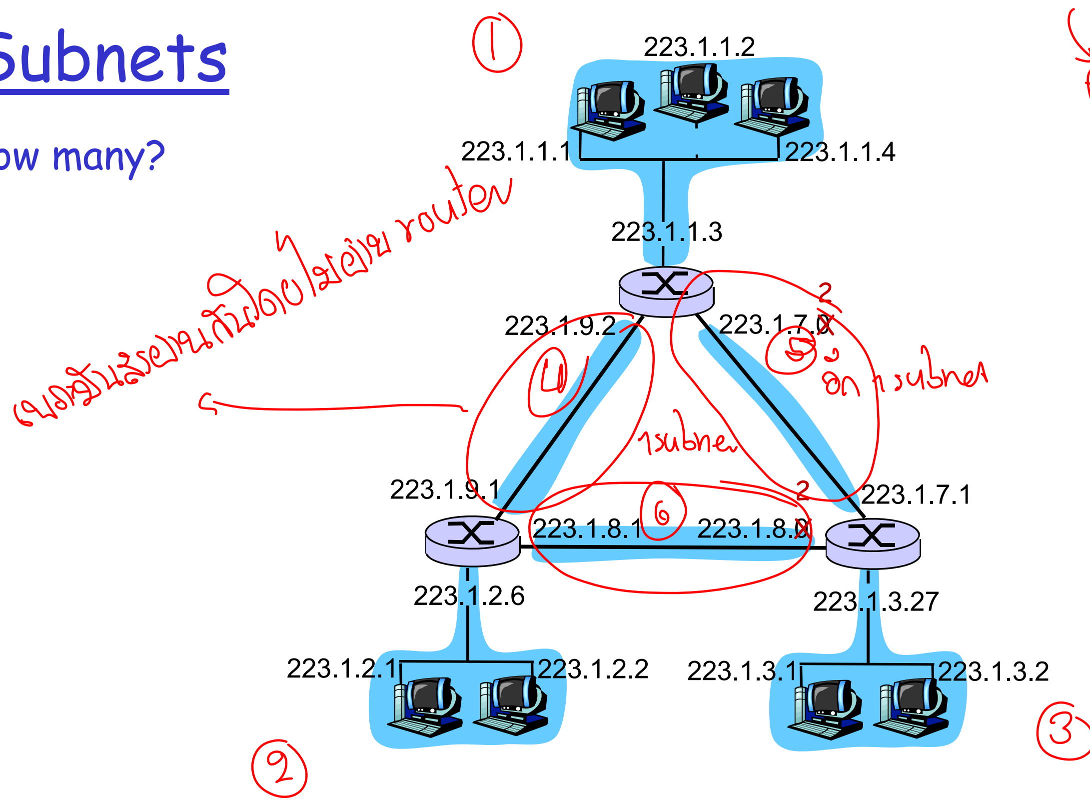
mask n bit

Class C

Subnets

How many?

1 interface = 1 subnet
free send inside



IP addressing: CIDR

กำหนดแบบไม่สนใจ class

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

ส่วนที่เป็น network 20 บิท

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

ที่เหลือจัดการ subnet เองได้

20+3

IP addresses: how to get one?

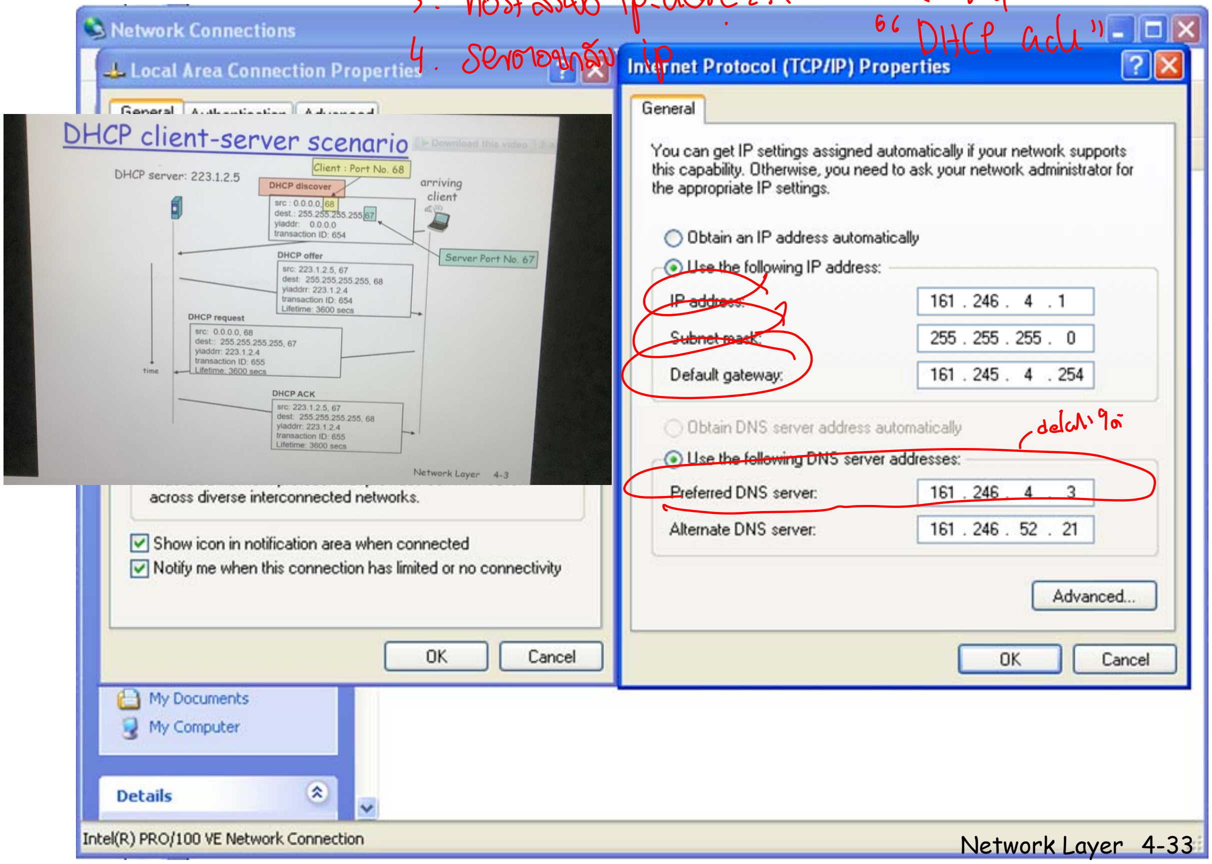
- **Q:** How does *host* get IP address?
- hard-coded by system admin in a file → กำหนดแบบ manual ได้
  - Windows:
    - control-panel->network connections->properties ->Internet Protocol (TCP/IP)
  - UNIX: /etc/rc.config

  ตามระบบ OS
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server → การกำหนด ip จาก server โดยอัตโนมัติ
  - "plug-and-play" → เชื่อมปุ๊ปใช้ได้ปั๊บ
  - allow host to *dynamically* obtain its IP address from network server when it joins network

หลักการทำงาน: 1. host ใดต้องการเข้ามาเชื่อมต่อ

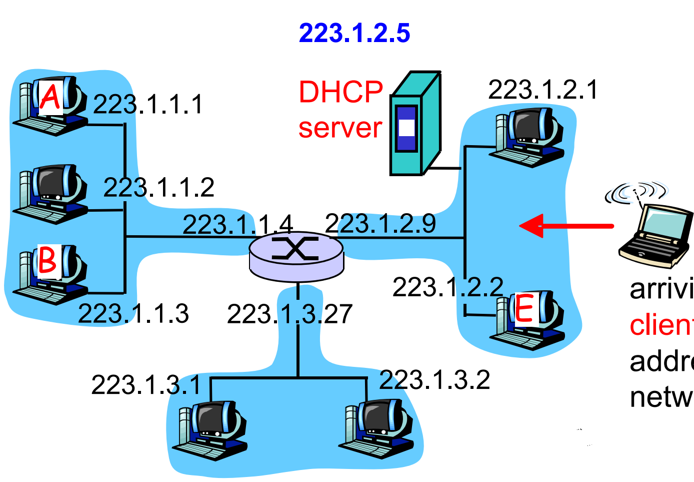
ต้องทำการส่ง msg ชื่อ "DHCP discover"

2. serได้รับ msg ตอบกลับมา "DHCP offer"
3. host ส่งขอ ip address "DHCP request"
4. serตอบกลับ ip "DHCP ack"



DHCP ใช้ UDP protocol และใช้ port
server : 67
cilent : 68

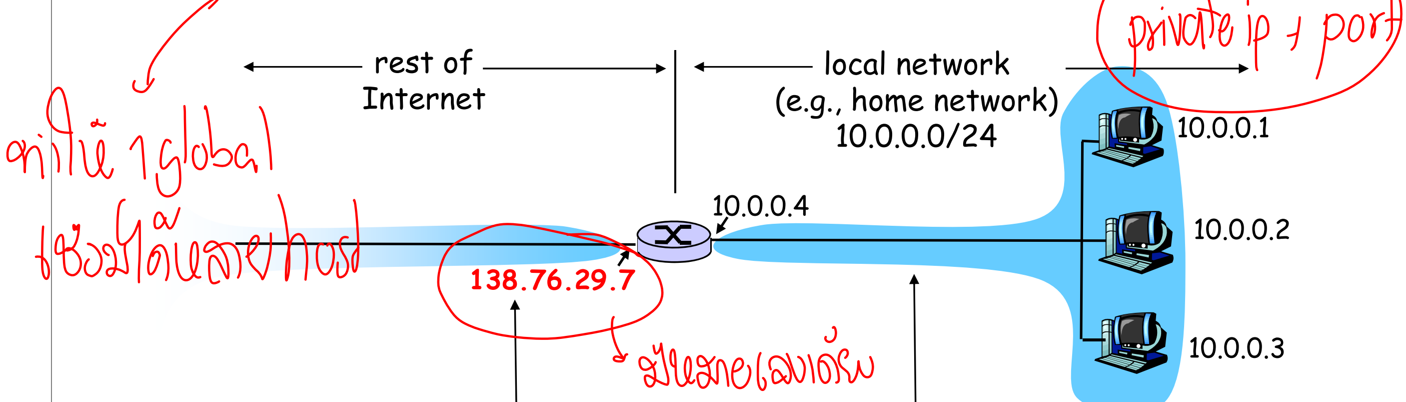
DHCP client-server scenario



arriving DHCP client needs address in this network

ปกติ 1 global : 1 host

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 10.0.0.0/24 address for source, destination (as usual)

	Range			Total
Class A	10.0.0.0	to	10.255.255.255	2^{24}
Class B	172.16.0.0	to	172.31.255.255	2^{20}
Class C	192.168.0.0	to	192.168.255.255	2^{16}

การจองกลุ่ม ip ใช้ใน ip ภายใน local network

ข้อดี

NAT: Network Address Translation

- Motivation: local network uses just *one IP address* as far as outside world is concerned:
  - range of addresses not needed from ISP: just one IP address for all devices
  - can *change addresses of devices* in local network without notifying outside world
  - can *change ISP* without changing addresses of devices in local network
  - devices inside local net not explicitly addressable, visible by outside world (a security plus).

NAT: Network Address Translation

Implementation: NAT router must:

ส่ง private ip+port nat แปลง global+port

เปลี่ยน datagram

- *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 address.

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

global + port → private+port เปลี่ยนไป host

- *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

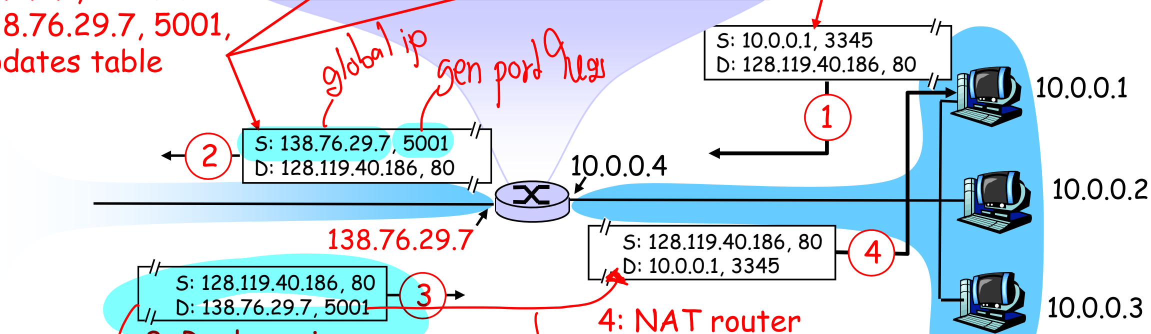
NAT: Network Address Translation

NAT translation table

WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……

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

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80



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

ไปดูในตาราง เพื่อหาว่าจะส่งไปหาใคร

map address back

Address Mapping

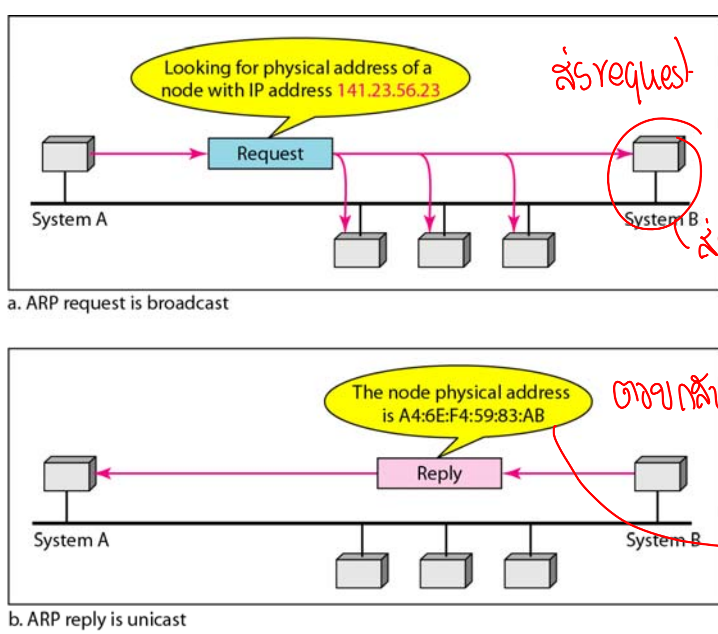
ในการส่งข้อมูลไปให้ตัว physical address
จะรู้แค่ตัวมันเองรู้จักตัวมันเอง
แต่ไม่รู้ของคนอื่น เราเลยใช้ ARP

- Delivery of packet to host or router requires two levels of addressing: logical address and physical address
- We need to be able to map a logical address to its corresponding physical address and vice versa.
- Mapping Logical Address to Physical Address can be done by Address Resolution Protocol (ARP) RFC 826

Logical address → Physical address

Address Resolution Protocol (ARP)

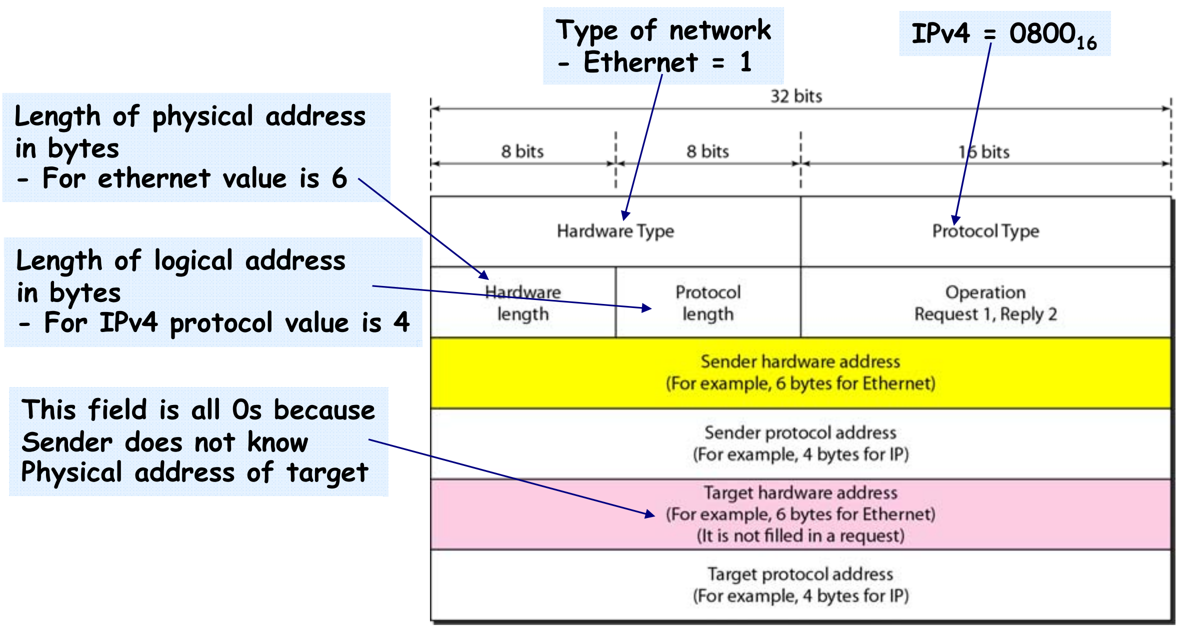
ส่ง request
ถามว่าใคร ip อันนี้อ่ะ ?
ส่งมาหาคนนี้



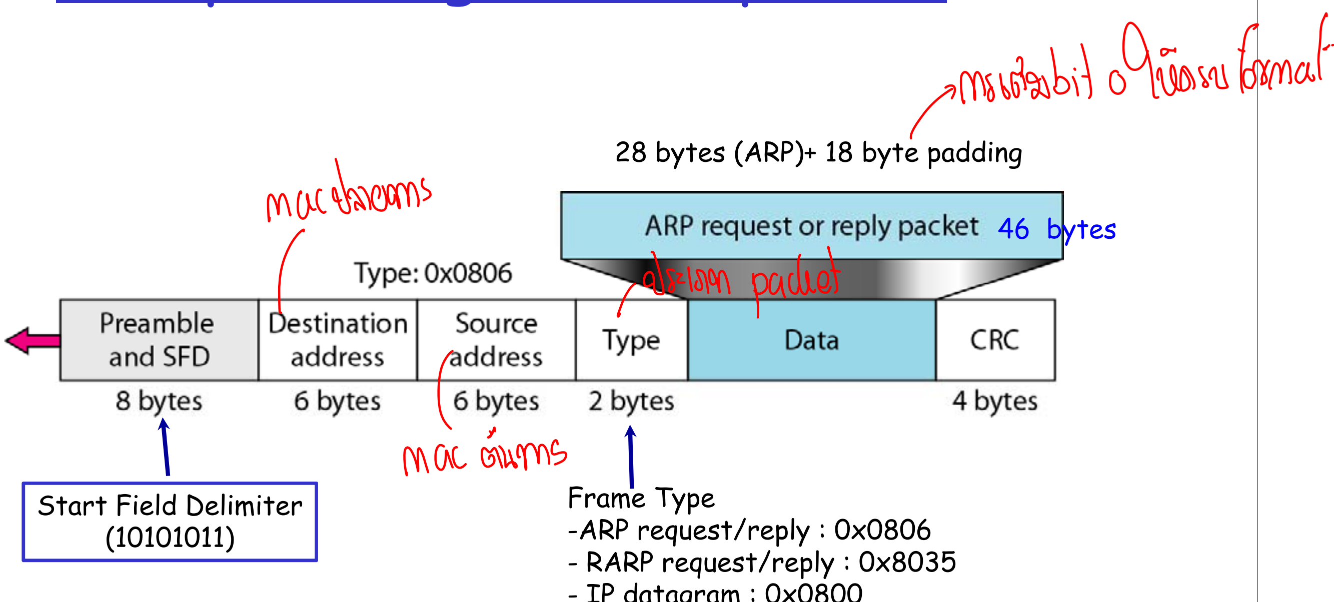
ตอบกลับ reply
ของเรา mac เราคือ

ARP Operation

ARP Packet

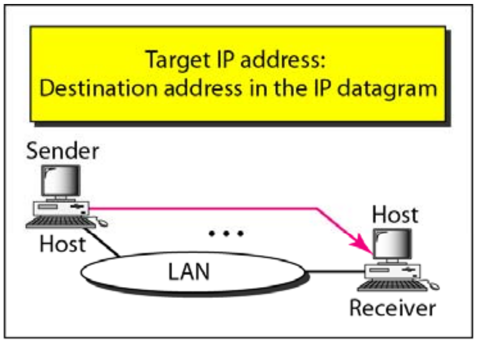


Encapsulating of ARP packet



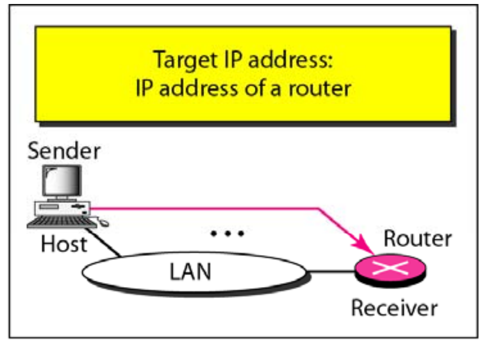
กรณีการใช้งาน

① อยู่ในเครือข่ายเดียวกัน



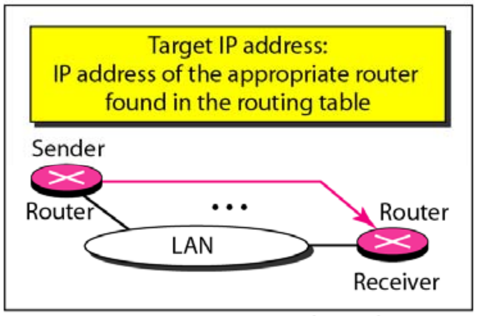
Case 1. A host has a packet to send to another host on the same network.

② คนละเครือข่าย ไปหา router



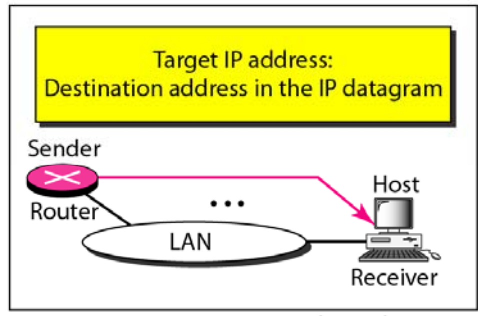
Case 2. A host wants to send a packet to another host on another network. It must first be delivered to a router.

③ จาก interface router ไปหา router



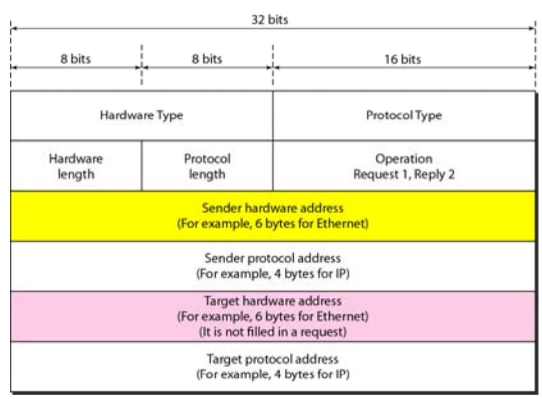
Case 3. A router receives a packet to be sent to a host on another network. It must first be delivered to the appropriate router.

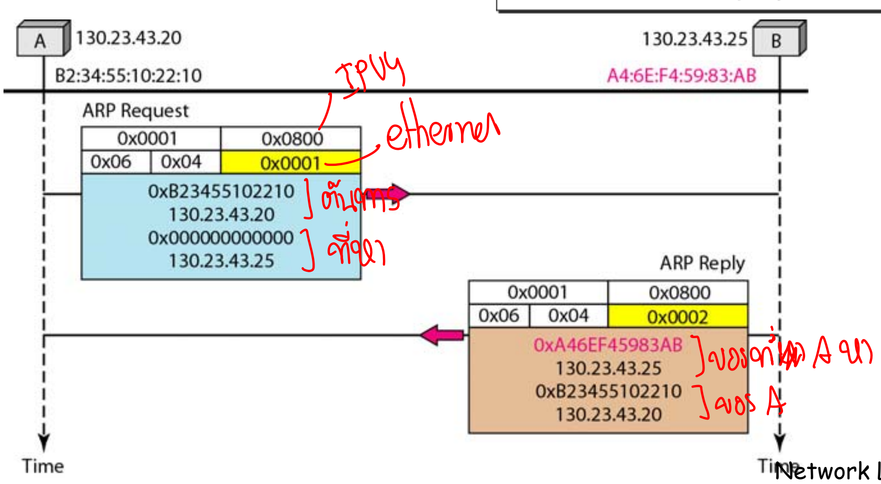
④ จาก router ไป หา host



Case 4. A router receives a packet to be sent to a host on the same network.

Example





Internet Control Message Protocol (ICMP)

- IP provides unreliable and connectionless datagram delivery
  การส่งข้อมูลอาจมีการ loss ได้
- IP protocol is a best-effort delivery service that delivers datagram from original source to final destination
- Two deficiencies
  - **Lack of error control** ไม่มีกระบวนการรายงาน error , error correcting
    - No error-reporting or error-correcting mechanism
  - **Lack of assistance mechanism for host and management queries**
    host ไม่สามารถรู้ว่าจะส่งข้อมูลได้หรือไม่เมื่อส่งไปยัง router
    - Host sometimes needs to determine if router or another host is alive
    - Sometimes a network administrator needs information from another host or router

เอาเลขน้อยมา

ICMP : Type of Messages

- ICMP message are divided into two broad categories
  - **Error-reporting message**
    - Report problems that router or host (destination) may encounter when it processes IP packet
  - **Query message**
    - Help host or network manager get specific information from router or another host
      - Ex. Nodes can discover their neighbors ค้นหาเพื่อนใน network เจอ
      - Hosts can discover and learn about routers on their network

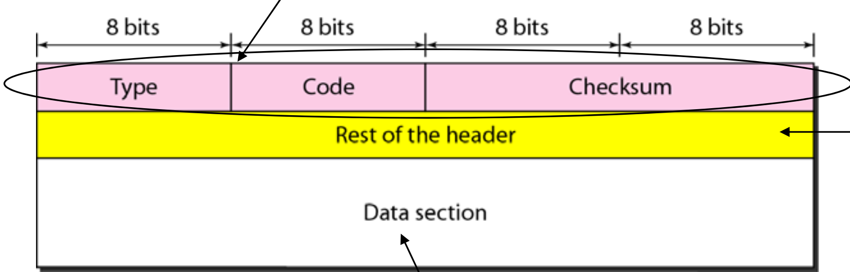
ICMP : Message Format

- ICMP message has 8-byte header and variable-size data section

ขนาดที่แน่นอน (max)

เปลี่ยนแปลงได้

First 4 bytes are common to all



specific for each message type

ระบุ type

- In **error messages** carries information for finding original packet that had error
- In **query messages** data section carries extra information based on type of query

โครงสร้าง section

ICMP : Error Reporting

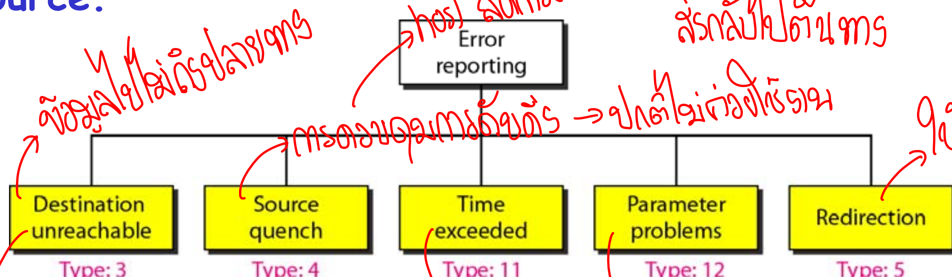
- **ICMP always reports error messages to original source.**

host ออกการส่งข้อมูล

ส่งกลับไปต้นทาง

ข้อมูลไปไม่ถึงปลายทาง

การควบคุมการส่งข้อมูล → ปกติไม่ค่อยใช้งาน



ใช้

TTL

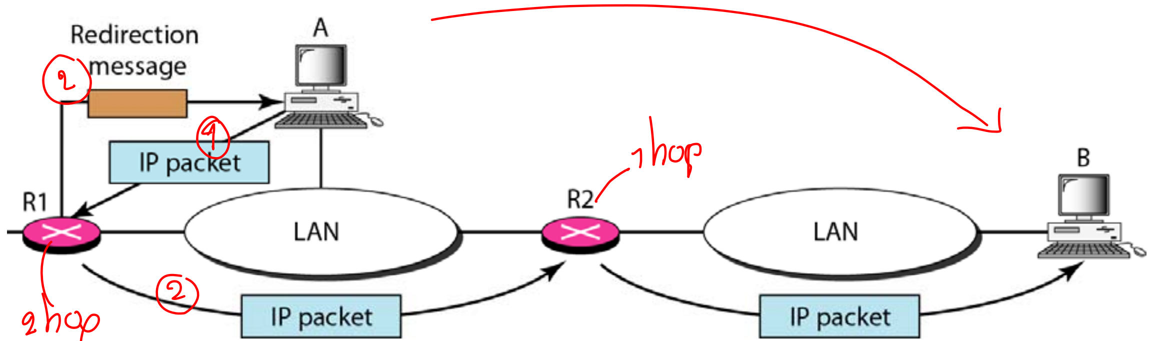
ข้อมูลผิดจาก parameter

เช่น ที่ส่วนของ header

Type	Code	description
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)
11	0	TTL expired
12	0	bad IP header

ย่อย

Redirection Concept

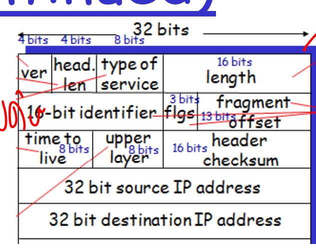


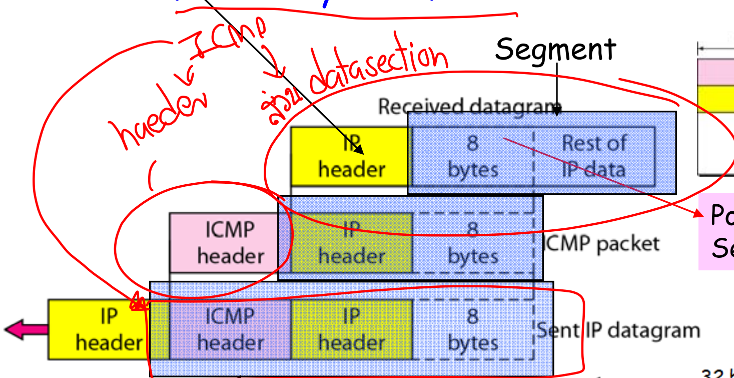
- Host A wants to send a datagram to host B
- Router R2 is obviously most efficient routing choice, but host A did not choose router R2
- Datagram goes to R1 instead
- Router R1, after consulting its table, finds that packet should have gone to R2
- R1 sends packet to R2 and,at the same time, sends a redirection message to host A
- Host A's routing table can now be updated

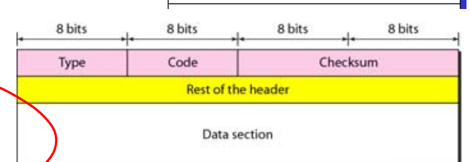
เมื่อจะส่งครั้งถัดไป ให้ส่งไปยัง R2 แทน

ICMP : Error Reporting (continued)

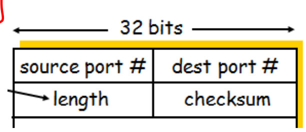
- All error messages contain data section that includes
  ถ้า error msg จะมี data section ประกอบด้วย
  - IP header of the original datagram plus
  - the first 8 bytes of data in that datagram

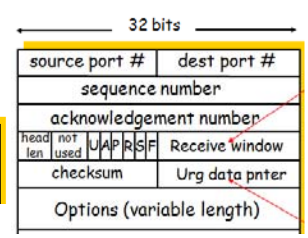






Port No. (UDP, TCP) (16 + 16 bits)
Sequence No (TCP) (32 bits)





- network-layer "above" IP:
  - ICMP messages carried in IP datagrams

ส่งลายรูปแบบทำไปไม่ได้

Type: 3	Code: 0 to 15	Checksum
Unused (All 0s)		
Part of the received IP datagram including IP header plus the first 8 bytes of datagram data		

Destination Unreachable

Type: 4	Code: 0	Checksum
Unused (All 0s)		
Part of the received IP datagram including IP header plus the first 8 bytes of datagram data		

ให้ต้นทางลดการส่งข้อมูล

Source Quench

0 : Time to Live exceeded in Transit
1 : Fragment Reassembly Time Exceeded

Type: 11	Code: 0 or 1	Checksum
Unused (All 0s)		
Part of the received IP datagram including IP header plus the first 8 bytes of datagram data		

TTL Expired

Code 0 : Error in one of header field, value in pointer field points to byte with problem
Code 1 : required part of option is missing

Type: 12	Code: 0 or 1	Checksum
Pointer	Unused (All 0s)	
Part of the received IP datagram including IP header plus the first 8 bytes of datagram data		

Parameter Problem

Type: 5	Code: 0 to 3	Checksum
IP address of the target router		
Part of the received IP datagram including IP header plus the first 8 bytes of datagram data		

Redirection message format

Code 0 – Redirection for network (or Subnet) -specific route → การกำหนดเส้นทาง
Code 1 – Redirection for host-specific route
Code 2 – same as code 0, based on specified type of service
Code 3 – same as code 1, based on specified type of service

เหมือน 0, 1

ICMP : Query

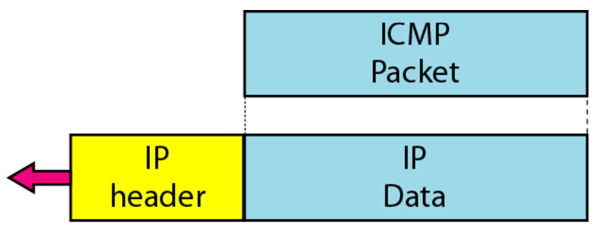
การส่ง msg เพื่อสอบถามข้อมูล

Type	Code	description
0	0	echo reply (ping)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery

ping: ส่งจากต้นทางไปปลายทางเพื่อที่จะถามว่า host ปลายทางยังอยู่หรือไม่ สามารถรับข้อมูลได้หรือไม่

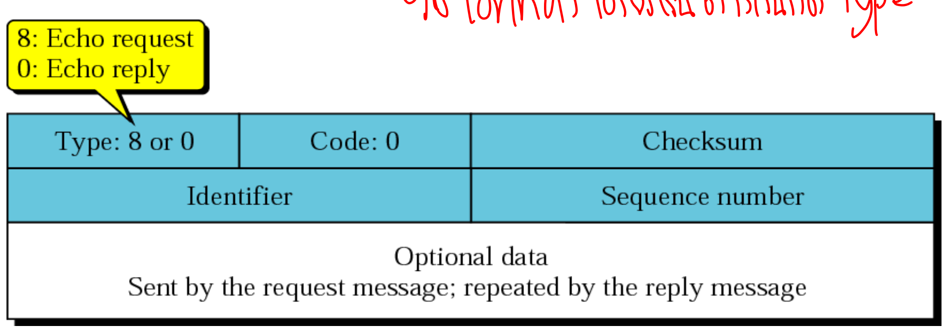
ส่วนที่ query จะถูกบรรจุใน ip datagram

- Query message is encapsulated in IP packet, which in turn is encapsulated in data link layer frame
- In this case, no bytes of original IP are included in message



Echo-request and echo-reply messages

ใช้ format เดียวกัน ต่างกันแค่ type



Route discovery and Route Advertisement

- Route discovery → ส่งเพื่อใช้ในการค้นหาเส้นทาง

Type: 10	Code: 0	Checksum
Identifier		Sequence number

การประกาศเส้นทาง

- Route advertisement

information for each router address entry in the list. The value is normally set to 2 (router address + preference level).

มี 2 ส่วน

number of router advertisements in message

จำนวนเราเตอร์ ที่ประกาศใน msg

Type: 9	Code: 0	Checksum
Number of addresses	Address entry size	Lifetime
IPv4 address router — Router address 1		
Address preference 1		
Router address 2		
Address preference 2		
⋮		

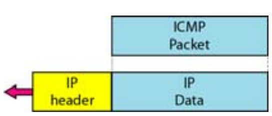
Ranking
0 – default
80000000_{16}

Ranking of router and used to select router as default router

Ping

- Ping program is used to find whether a host is alive or not

56+8 bytes of ICMP header + 20 bytes of IP header



56+8 bytes of ICMP header

```
$ ping fhda.edu
PING fhda.edu (153.18.8.1)    56 (84) bytes of data.
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=0    ttl=62    time=1.91 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=1    ttl=62    time=2.04 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=2    ttl=62    time=1.90 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=3    ttl=62    time=1.97 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=4    ttl=62    time=1.93 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=5    ttl=62    time=2.00 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=6    ttl=62    time=1.94 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=7    ttl=62    time=1.94 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=8    ttl=62    time=1.97 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=9    ttl=62    time=1.89 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=10   ttl=62    time=1.98 ms

--- fhda.edu ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10103ms
    rtt min/avg/max = 1.899/1.955/2.041 ms
```

โปรแกรม

ข้อมูลของ router จาก host ต้นทางไปยังปลายทาง

Traceroute

- 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 n^{th} router:
  - Router discards datagram
  - And sends to source an ICMP message (type 11, code 0) : TTL expired
  - Message includes name of router& IP address

router ๆ จะทำการลด ttl ลง 1 และเมื่อ ttl = 0 จะทำการ discard datagram นั้นแล้วส่ง ICMP ไปยังต้นทาง

ที่ส่งกลับ

- When ICMP message arrives, source calculates RTT

(เมื่อ packet ที่ปลายทางกลับมาต้นทางก็จะแสดง RTT

- Traceroute does this 3 times
- Stopping criterion
- UDP segment eventually arrives at destination host

หยุดเมื่อถึง host ปลายทาง

- Destination returns ICMP "port unreachable" packet (type 3, code 3 : Destination Port Unreachable)

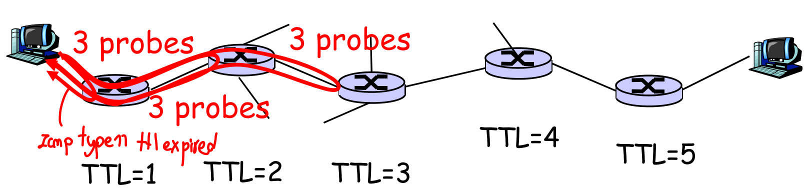
ตอบกลับ

- When source gets this ICMP, stops.

การทำงาน

"Real" Internet delays and routes

- What do "real" Internet delay & loss look like?
- Traceroute program (tracert for windows) : provides delay measurement from source to router along end-end Internet path towards destination. For all *i*:
  - sends three packets that will reach router *i* on path towards destination
  - router *i* will return packets to sender
  - sender times interval between transmission and reply.



"Real" Internet delays and routes

traceroute: gaia.cs.umass.edu to www.eurecom.fr

ทำ ส่ง 3 ครั้ง Three delay measurements from gaia.cs.umass.edu to cs-gw.cs.umass.edu

ชื่อrouter ip router

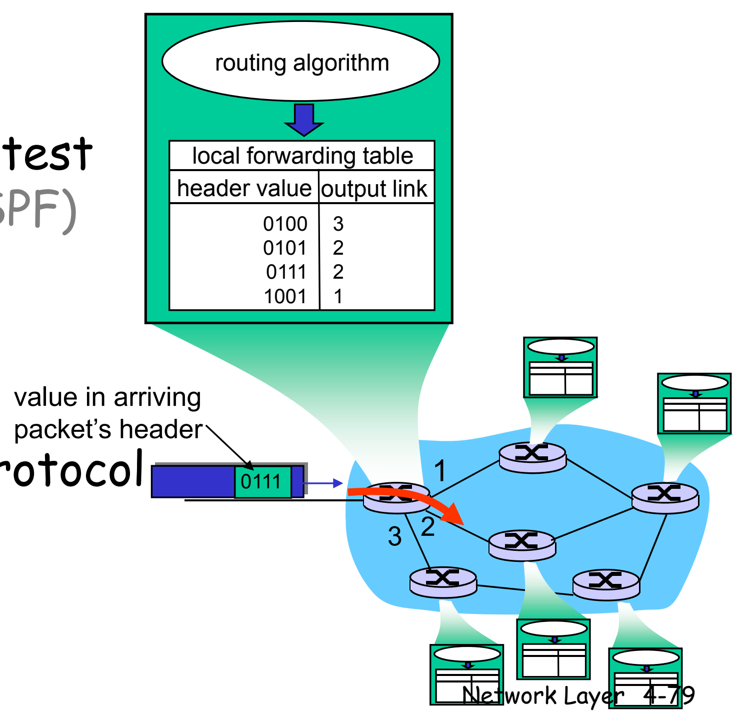
```
1  cs-gw (128.119.240.254)  1 ms  1 ms  2 ms
2  border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145)  1 ms  1 ms  2 ms
3  cht-vbns.gw.umass.edu (128.119.3.130)  6 ms 5 ms 5 ms
4  jn1-at1-0-0-19.wor.vbns.net (204.147.132.129)  16 ms 11 ms 13 ms
5  jn1-so7-0-0-0.wae.vbns.net (204.147.136.136)  21 ms 18 ms 18 ms
6  abilene-vbns.abilene.ucaid.edu (198.32.11.9)  22 ms  18 ms  22 ms
7  nycm-wash.abilene.ucaid.edu (198.32.8.46)  22 ms  22 ms  22 ms
8  62.40.103.253 (62.40.103.253)  104 ms 109 ms 106 ms
9  de2-1.de1.de.geant.net (62.40.96.129)  109 ms 102 ms 104 ms
10  de.fr1.fr.geant.net (62.40.96.50)  113 ms 121 ms 114 ms
11  renater-gw.fr1.fr.geant.net (62.40.103.54)  112 ms  114 ms  112 ms
12  nio-n2.cssi.renater.fr (193.51.206.13)  111 ms  114 ms  116 ms
13  nice.cssi.renater.fr (195.220.98.102)  123 ms  125 ms  124 ms
14  r3t2-nice.cssi.renater.fr (195.220.98.110)  126 ms  126 ms  124 ms
15  eurecom-valbonne.r3t2.ft.net (193.48.50.54)  135 ms  128 ms  133 ms
16  194.214.211.25 (194.214.211.25)  126 ms  128 ms  126 ms
17  * * *
18  * * *
19  fantasia.eurecom.fr (193.55.113.142)  132 ms  128 ms  136 ms
```

trans-oceanic link

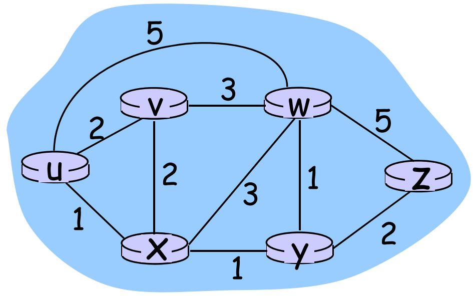
* means no response (probe lost, router not replying)

Routing Algorithms

- Link state
  - Ex. Open Shortest Path First (OSPF)
- Distance Vector
  - Ex. Routing Information Protocol (RIP)



Graph abstraction



Graph: G = (N,E)

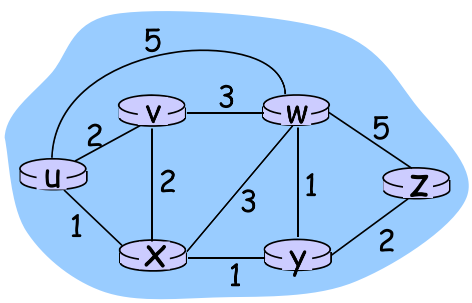
N = set of routers = { u, v, w, x, y, z }

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 ?

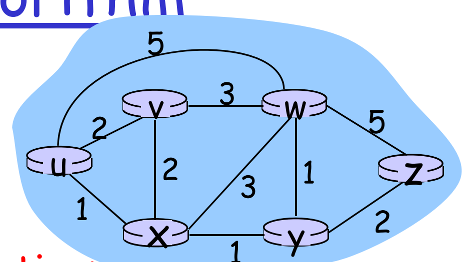
Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

- Global or decentralized information?
- Global:
  - all routers have complete topology, link cost information
  - "link state" algorithms
- Decentralized:
  - router knows physically-connected neighbors, link costs to neighbors
  - iterative process of computation, exchange of information 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

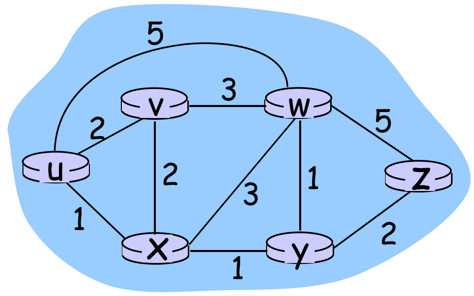
A Link-State Routing Algorithm



- Dijkstra's algorithm
- network topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same information
- 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 destination'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 destination v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijsktra's Algorithm



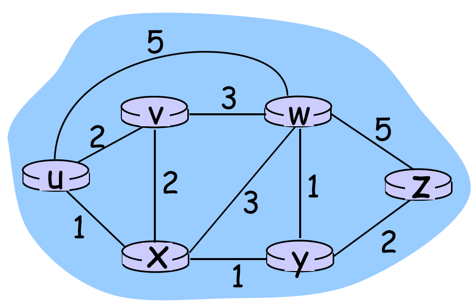

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
       then D(v) = c(u,v)
    else D(v) = ∞

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'
```

หา Cost จาก U ไปยัง Node ข้างเคียงก่อน

เลือก node ที่ให้ cost น้อยที่สุด เป็นสมาชิกของ N’

P : โหนดใดๆ ในเครือข่ายที่ไม่เป็นสมาชิกของ N’

Dijsktra's Algorithm




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

Loop
  find Q not in N' such that D(Q) is a minimum
   add Q to N'
   update D(P) for all P adjacent to Q and not in N' :
      D(P) = min( D(P), D(Q) + c(Q, P) )
   /* new cost to P is either old cost to P or known
    shortest path cost to Q plus cost from Q to P */
until all nodes in N'
```

หา Cost จาก U ไปยัง Node ข้างเคียงก่อน

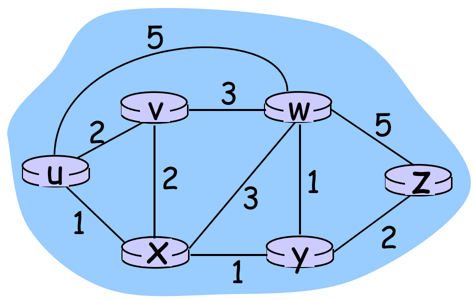
เลือก node ที่ให้ cost น้อยที่สุด เป็นสมาชิกของ N’

Q : โหนดใดๆ ในเครือข่ายที่ไม่เป็นสมาชิกของ N’ และมีค่า Cost น้อยที่สุด

Dijkstra's algorithm: example

D(P): current value of cost of path from source to destination P
p(P): predecessor node along path from source to P

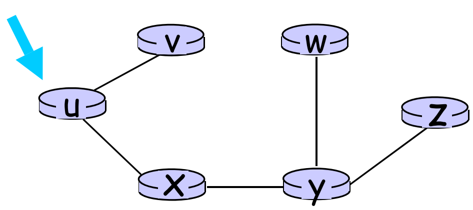
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



ต้องการหา **Shortest Path** จาก **Node u** ไปยังทุก **Node**

Dijkstra's algorithm: example (2)

Resulting shortest-path tree from u:



Resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

Define

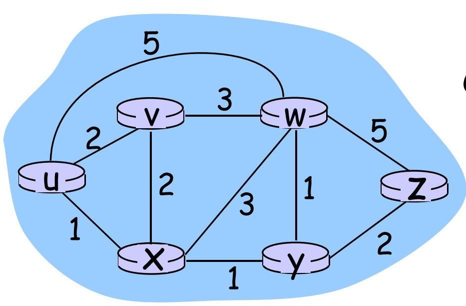
$d_x(y)$:= cost of least-cost path from x to y

Then

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

where min is taken over all neighbors v of x

Bellman-Ford example



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

B-F equation says:

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

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

Distance Vector Algorithm

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

Each node:

wait for (change in local link cost or messg from neighbor)

recompute estimates

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

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

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

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

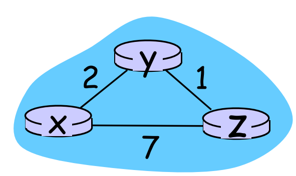
node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

time



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

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

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

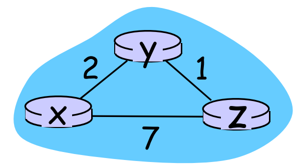
node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

time →



IPv6 Header (Cont)

เกิด IPV4 (32b) → ไม่เพียงพอต่อความต้องการใช้ → เกิด IPV6 (128b)

- ขนาด header fix ที่ 40 byte

Packet priority(8-bits - รองรับ QoS
Priority values subdivide into range:
Traffic where source provides
- Congestion control
- Non-congestion control

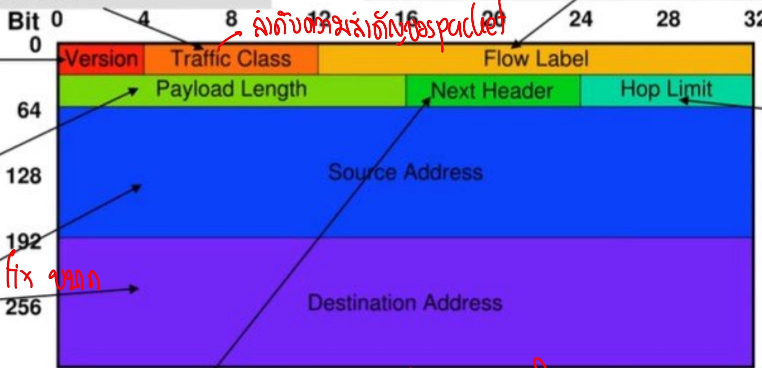
- ไม่มี fragmentation

การแตกของ datagram (ใน v4)

มากกว่าเดิม 2^{96} เท่า

QoS management (20bits)
Originally created for giving realtime appplications
Special service, but currently unused

Version 6
(4-bit IP version)



Payload length in Bytes (16 bits)

ตัดเก่า header length
แต่ว่ามันไม่มีเพราะ header fix ขนาด

128 bits)

Replaces the TTL field of IPv4 (8 bits)

ip protocol → เหมือนใน v4

Specifies the next encapsulated protocol. Values are compatible with those specified for IPv4 protocol field (8 bits) (UDP=17, TCP=6)

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

ยังรับรองรับแค่ Ipv4

- Not all routers can be upgraded simultaneous
  - no "flag days" → ไม่มีวันเวลาที่แน่นอนในการเปลี่ยน standard → ipv6
  - How will the network operate with mixed IPv4 and IPv6 routers?
- ระ *Tunneling:* IPv6 carried as payload in IPv4 datagram among IPv4 routers

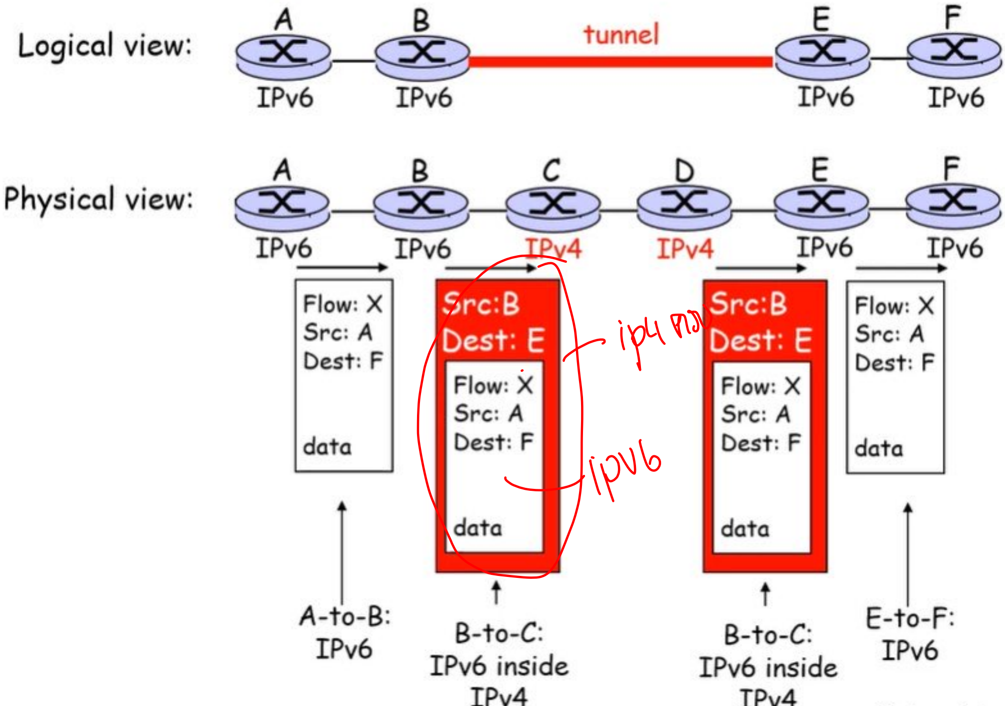
อุปกรณ์ที่ $\boxed{\text{IPv6}}$ → X

IPv4

(payload)

ข้อมูล IPv6 จะถูกใส่ลงใน IPv4

Tunneling



Logical view:
A
B
tunnel
E
F
IPv6
IPv6
IPv6
IPv6
Physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
Flow: X
Src: A
Dest: F
data
Src:B
Dest: E
Flow: X
Src: A
Dest: F
data
Src:B
Dest: E
Flow: X
Src: A
Dest: F
data
Flow: X
Src: A
Dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6
ip4
ipv6

Routing Algorithms

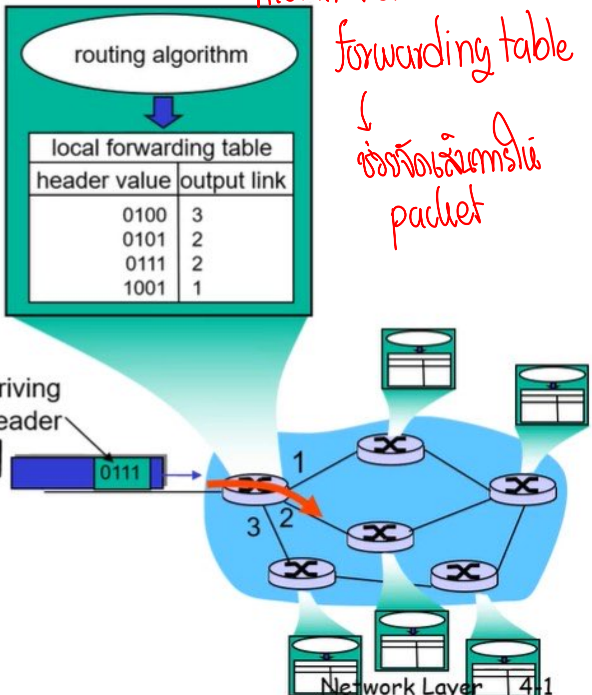
→ ถูกใช้โดย routing protocol
ที่เป็นกรรมวิธีการ

al 1st   มีแผนหาทาง แต่ละรอบแต่ละ

- Link state
  - Ex. Open Shortest Path First (OSPF)

al 2nd

- Distance Vector
  - Ex. Routing Information Protocol (RIP)

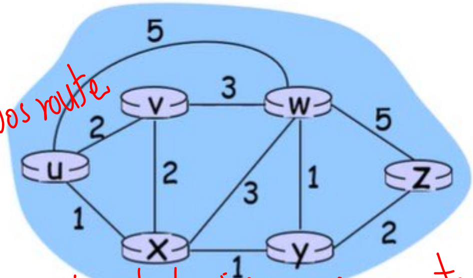


forwarding table
↓
ช่วยจัดเส้นทางให้
packet

Graph abstraction

set ของ route

node



Graph: G = (N,E)

edge – set ของ link ที่เชื่อมระหว่าง 2 router

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

อาจแทนเหมือนความยาวระยะ

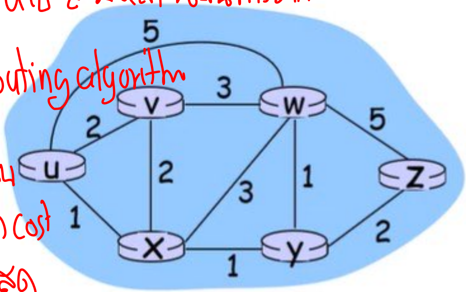
จาก u ไป z มีหลายเส้นทางมาก

หน้าที่ routing algorithm

ค้นหาเส้นทาง

ที่ค่า cost

น้อยที่สุด



router 2 router

given:
- $c(x,x')$ = 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

สรุปรวมระยะทางจะผ่านมากกว่า 1 router → path

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

↳ cost ของเส้นทางทั้งหมดรวมกัน

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

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

ข้อมูลของเส้นทาง

- Global or decentralized information? แบ่งได้ตามลักษณะของข้อมูล
- Global: → ทุก router มีข้อมูลของ cost ทั้งหมด
  - all routers have complete topology, link cost information ของทั้ง network ตัวเอง
  - เรียกว่า "link state" algorithms
- Decentralized: → router มีข้อมูลแค่ของ router เพื่อนบ้าน (ที่เชื่อมต่อกันโดยตรง)
  - router knows physically-connected neighbors, link costs to neighbors
  - iterative process of computation, exchange of information with neighbors
  - เรียกว่า "distance vector" algorithms

สามารถปรับตามการเปลี่ยนแปลงได้ด้วย

- Static or dynamic?
- Static: เส้นทางจะมีการเปลี่ยนแปลงแบบ
  - routes change slowly over time ช้าๆ ตลอดเวลา
- Dynamic:
  - routes change more quickly
    - periodic update มีการอัปเดต เป็นช่วงๆ
    - in response to link cost changes

A Link-State Routing Algorithm

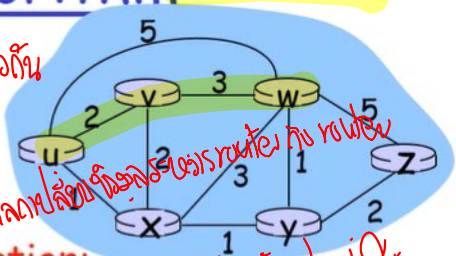
จาก D(w) = 2+3 = 5

(ใช้ algorithm นี้

ทุก node รู้ถึงข้อมูลเดียวกัน

ได้ไง?

ใช้ broadcast ในการแลกเปลี่ยนข้อมูลระหว่าง router กับ router



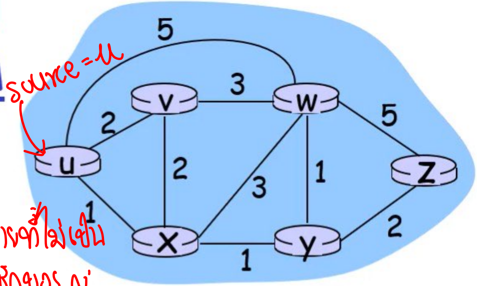
- Dijkstra's algorithm
- network topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same information
- computes least cost paths from one node ('source") to all other nodes หาเส้นทางที่สั้นที่สุด
  - gives forwarding table for that node สำหรับ node ต้นทาง, ปลายทาง(s) อื่นๆ ที่เราเลือกพิจารณา
- iterative: after k iterations, know least cost path to k destination's

- Notation: ความหมายของตัวแปรที่จะใช้
- c(x,y): link cost from node x to y; = ∞ if not direct neighbors (ถ้าไม่อยู่ติดกันนะ
- D(v): current value of cost of path from source to destination v ผลรวมได้ ตั้งแต่ต้นทางถึง D(destination) = value
- p(v): predecessor node along path from source to v node ก่อนหน้า กำหนด u = source ไป y คือ destination P(y) => node ที่อยู่ก่อน node y คือ node x P(y) = x
- N': set of nodes whose least cost path definitively known

set ของ node ต่างๆ เหล่านี้ที่รู้ค่า cost ที่น้อยที่สุดแล้ว

P : โหนดใดๆ ในเครือข่ายที่ไม่เป็นสมาชิกของ N'

Dijsktra's Algorithm



ตารางแสดง node u only

Initialization: การกำหนดค่า
  N' = {u}
  for all nodes P node ใดๆ ในเครือข่ายที่ไม่เป็นสมาชิกของ N'
    if P adjacent to u #ถ้าติดกันมีค่า
      then D(P) = c(u, P)
    else D(P) = ∞
ถ้าไม่ใช่เพื่อนบ้านจะเป็น ∞

หา Cost จาก U ไปยัง Node ข้างเคียงก่อน

เลือก node ที่ให้ cost น้อยที่สุด เป็นสมาชิกของ N'

Loop การวน loop
  find Q not in N' such that D(Q) is a minimum หา node ที่ไม่อยู่ใน N' และมีค่าน้อยที่สุด ↓ ที่มีค่า D(Q)
  add Q to N'
  update ค่าจากเพื่อนบ้าน(1) ไปยังเพื่อนบ้าน(2)
  update D(P) for all P adjacent to Q and not in N' :
    D(P) = min(D(P), D(Q) + c(Q, P))
  /* new cost to P is either old cost to P or known
   shortest path cost to Q plus cost from Q to P */
until all nodes in N'

Q : โหนดใดๆ ในเครือข่ายที่ไม่เป็นสมาชิกของ N' และมีค่า Cost น้อยที่สุด

Dijkstra's algorithm: example

D(P): current value of cost of path from source to destination P

p(P): predecessor node along path from source to P

มีแต่เริ่มต้น

node ที่มาก

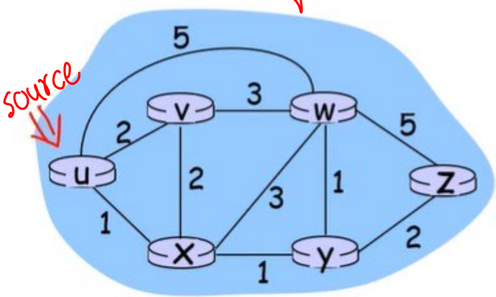
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0 เริ่มต้น	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

NO ไม่ได้เชื่อมต่อ

เลือกที่น้อยที่สุด

เอาน้อยสุดที่

เจอแล้วติดต่อทำไม?



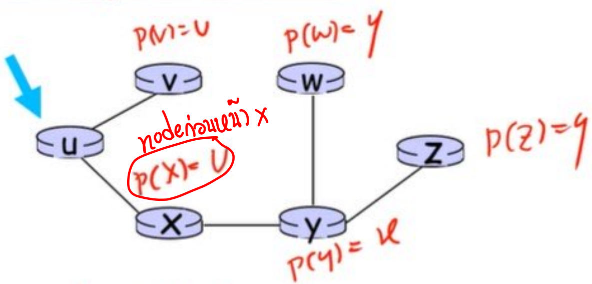
โดยหลักการ: เอา min (เส้นทางที่เชื่อมกันโดยตรง,

หรือ เส้นทางที่เชื่อมผ่าน N')

ต้องการหา **Shortest Path** จาก **Node u** ไปยังทุก **Node**

Dijkstra's algorithm: example (2)

Resulting shortest-path tree from u:



1.7 บาท = 1ท

50x1.7 ⇒ 85 บาท

40ทx1.7 = 68 บาท

Resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

ปลายทางต่อจากอันที่เป็นเพื่อนบ้าน

Distance Vector Algorithm

อีก algorithm การ routing

จุด

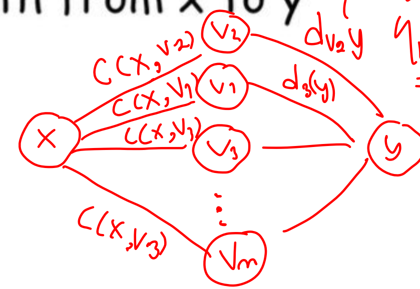
↳ Bellman-Ford Equation (dynamic programming)

Define

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

สั้นที่สุดจาก $x \to y$

สั้นที่สุดจาก เพื่อนบ้าน ๆ ไปหาค่า



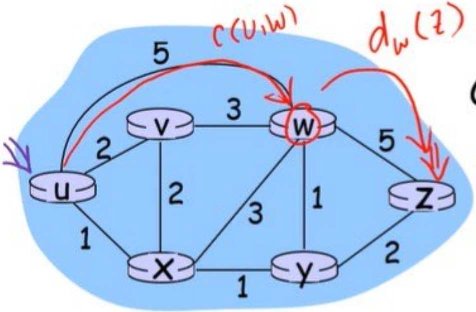
Then

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

v คือเพื่อนบ้านของ x ที่อยู่ติด

where <u>min</u> is taken over all neighbors v of x

Bellman-Ford example



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

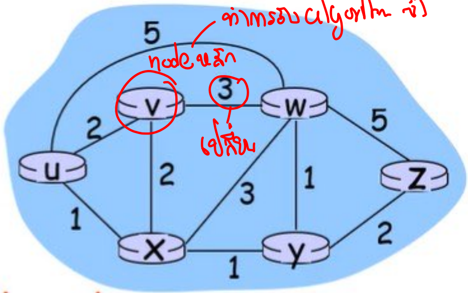
B-F equation says:

ตามจำนวนเพื่อนบ้าน

$$d_u(z) = \min \{ c(u,v) + d_v(z), \quad 2+5=7$$
$$\boxed{c(u,x) + d_x(z), \quad 1+3=4}$$
$$c(u,w) + d_w(z) \} \quad 5+3=8$$
$$= \min \{2 + 5,\ 1 + 3,\ 5 + 3\}$$
$$= 4$$

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

Distance Vector Algorithm



ใช้หลักการ

- **Iterative, asynchronous:** each local iteration caused by:
  - local link cost change
    ค่า cost เปลี่ยนไป
  - DV update message from neighbor
    (ได้รับ msg จากเพื่อนบ้าน
- **Distributed:**
  แจ้งการเปลี่ยนแปลงข้อมูลให้กับเพื่อนบ้านของตัวเองครบ
  - each node notifies neighbors *only* when its DV changes
    อาจจะแจ้งแบบ chain ได้
    - neighbors then notify their neighbors if necessary

Each node:

วน loop ต่อไป

wait for (change in local link cost or messg from neighbor)

↓

recompute estimates

↓

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

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

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

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

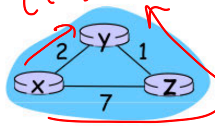
from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

time

แล้วไปปรเพื่อนบ้าน

$2+0 = 2$

$C(x,y)+D_y(y)$



$C(x,z)+d_z(y)$

$7+1 = 8$

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

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

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

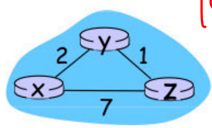
from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

time

ไม่มีการเปลี่ยนแปลง
ไม่ต้องแจ้งไป
ยัง node เพื่อน
บ้าน



การคำนวณ ip address

↳ เมื่อเราได้ ip มา 1 ชุดต้องบอก ① subnet mask คือ?
② network ip คือ?
③ broadcast ip คือ?
④ ช่วง ip , ช่วงที่ assign ได้ให้กับ host
⑤ จำนวน subnet , host ต่อ subnet

เน้นแค่ ABC

ตารางเลขฐาน

7	6	5	4	3	2	1	0	
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
128	64	32	16	8	4	2	1	จำนวน ip ต่อ subnet
128	192	224	240	248	252	254	256	ค่า sub

ในการ setup host ให้สามารถเชื่อมต่อ network ได้ เราต้องกำหนด ip address, subnet mask, default gateway
dns

subnet mask → แบ่งเครือข่ายเป็นส่วนย่อยๆ

ip /24 → "mask 24"
↓ 24 บิตแรกจากซ้าย => '1' ที่เหลือ '0'

max = 31

classful subnet

↳ class A => /8 → mask 8 bit
B /16
C /24

ปรับ mask เองก็ได้ตามแต่สะดวก

การหา network ip , broadcast ip

network ip
ช่วง ip ของ subnet
broadcast ip

network ip คือ ip ตัวแรกของ subnet ไม่สามารถ assign ให้ host
ใช้สำหรับประกาศเกี่ยวกับ routing ในอุปกรณ์

broadcast ip คือ ตัวท้าย, ไม่ assign ด้วย ค่านี้เป็น broadcast
ให้อุปกรณ์ทุกตัวใน เครือข่ายเดียวกัน

↳ex

192.168.22.50/30 → มีทั้งหมด 4 ตัวต่อ 1 network

subnet → 255.255.255.252

network ip

host 49–50

192.168.22.50 → อยู่ในช่วง 48 – 51 แต่ละช่วง

board cast ip

หา network ip จาก ipaddress & subnet ได้ว่า | หรือ → ip ต่ำสุด / ip ต่อ subnet

↳ ค่าที่เป็นเลขอยู่หน้า host จะคือ network ip พอไปเจอ 1 ก็จะรู้เลย

แล้วนับลงไปให้ตรง

การ assign ip ให้ host ต้องอยู่ใน subnet กับที่ได้มา

10.10.10.3/13 → 8 ip ต่อวงนึง

↓

มีหลาย

11111111.11111000 .

↳ 8 ip ใหม่ต่อ 1 network

↳ 8×255×255 #

$\frac{256}{8}$ subnet

32 subnet

subnet ละ: 8×256×256 − 2

และจำนวน host ต่อ subnet

$= 2^{32-n} - 2$; n คือ mask

$2^{18} = 524288 - 2$

เหมือนกัน

จำนวน subnet $= 2^n$; จำนวน bit ที่อยู่หน้า mask ตัว • ที่ใกล้ที่สุด

$2^5 = 32$

ได้เหมือนกัน

จากตัวอย่าง /20 แล้วแบ่ง /27 จะได้กี่ subnet

↳ ปกติ 1 subnet : 32 ip

จำนวน subnet = 2^p ; p จำนวน bit ระหว่าง mask

$= 2^{27-20} = 2^7 = 128$ subnet

Variable Length Subnet Masks (VLSM)

↳ เครือข่ายไม่จำเป็นต้องมีขนาดเท่ากันในแต่ละ subnet (mask ของแต่ละ subnet ไม่จำเป็นต้องเท่ากัน)

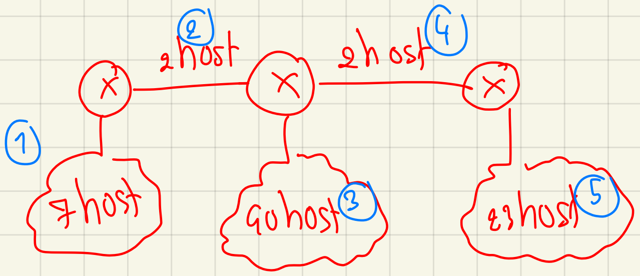
↳ ประหยัดการใช้งาน ip

↳ vlsm จะช่วยให้การแบ่ง subnet มากกว่าหนึ่งครั้ง สำหรับแต่ละชุดของ ip เพื่อให้ได้ขนาด ip ตามต้องการ

↳ vlsm ช่วยลดความสูญเสียการจัดสรร ip → ใช้ที่อยู่อย่างมีประสิทธิภาพ

↳ vlsm ช่วยให้ router ทำงานได้เร็วขึ้น → ขนาด routing น้อยลง

↳ ตัวอย่างการจัดสรร ip ให้กับเครือข่ายด้วย vlsm



(n) เครือข่าย

จากเครือข่าย ip ที่กำหนดให้คือ 192.168.55.0/24 จงทำการกำหนด ip ให้กับแต่ละ subnet (1-5)

ให้เหมาะสมกับจำนวน hosts ในแต่ละ subnet โดยการทำ vlsm

↳ subnet mask = 255.255.255.0 → 1 subnet : 256 ip address

192.168.55.0 – 255

↳ ทุกเครือข่าย ①-⑤ อยู่ใน ip space นี้

subnet ที่ ① ต้องการ 7 host → +2 (broadcast, network ip)

= 9 ip

ใช้ /28 → รองรับได้ 16 ip → สามารถกำหนด ip ได้เป็น 192.168.55.0 – 15

② ต้องการ 2 host + 2 = 4 ip

ใช้ /30 → ได้ 4 ip → 192.168.55.16-19

③ ต้องการ 90 host + 2 = 92 ip

ใช้ /25 → ได้ 128 ip → 192.168.55.128-255 ← ไม่ต้องเรียงก็ได้ แต่ห้ามซ้ำกัน

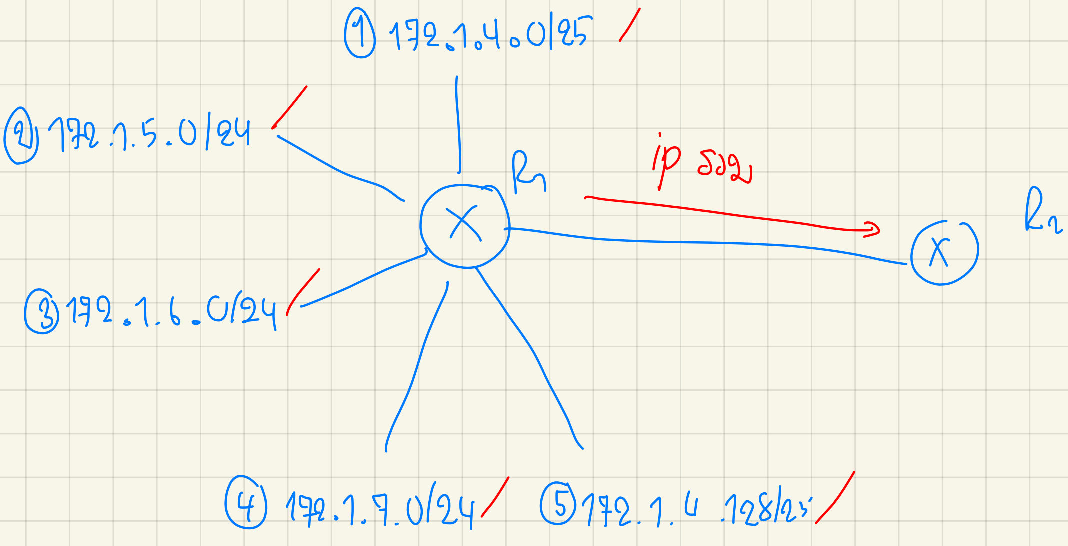
④ เหมือนข้อ ② /30 → $\boxed{}$.20-23

⑤ 23+2 = 25 ip ใช้ /27 → 32 ip → $\boxed{}$.$\underbrace{32 - 63}_{\text{ความกว้างตาม mask นะ}}$

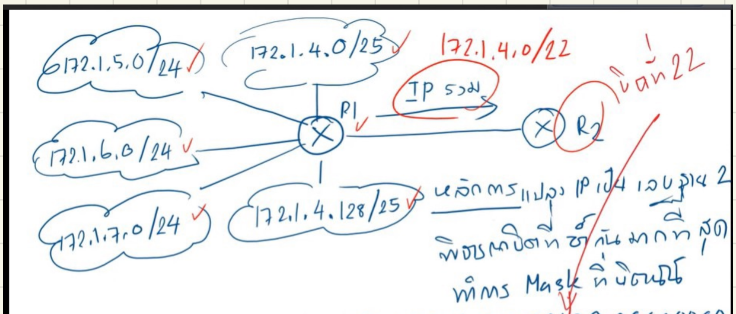
Summarization

↳ การรวม ip จำนวนหลายๆ เครือข่ายให้เหลือเพียงเครือข่ายเดียว ซึ่งครอบคลุม ip หลายๆ เครือข่าย

↳ ทำให้ตาราง routing table เล็กลง



หลักการแปลง ip เป็นเลขฐาน 2 พิจารณา bit ที่ซ้ำกันมากที่สุด ทำการ mask bit นั้น



172.1.4.0/25 → 10101100.00000001.00000100.00000000

172.1.5.0/25 → 10101100.00000001.00000101.00000000

172.1.6.0/24 → 10101100.00000001.00000110.00000000

172.1.7.0/24 → 10101100.00000001.00000111.00000000

172.1.4.128/25 → 10101100.00000001.00000100.10000000

IP summarization 172 . 1 . 4 . 0/22