

Chapter 4 Network Layer

การใช้สไลด์ :

เนื้อหาในสไลด์เหล่านี้ถูกแปลมาจากสไลด์ต้นฉบับประกอบหนังสือของผู้แต่งชื่อ Kurose และ Ross

ผู้แปลอนุญาตให้ทุกท่านสามารถใช้สไลด์ทั้งหมดได้ ดังนั้นท่านสามารถดูภาพเคลื่อนไหว สามารถเพิ่ม, แก้ไข และ ลบสไลด์ (นับรวมข้อความนี้) และเนื้อหาของสไลด์เพื่อให้เหมาะสมกับความต้องการของท่าน

สำหรับการแลกเปลี่ยน เราต้องการสิ่งต่อไปนี้เท่านั้น :

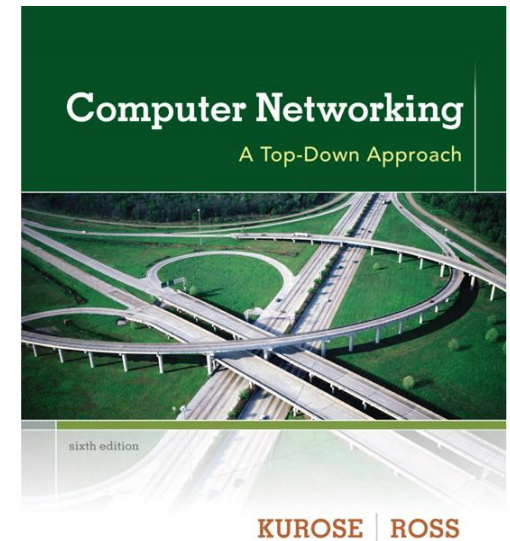
- ถ้าท่านใช้สไลด์เหล่านี้ (เป็นตัวอย่าง, ในห้องเรียน) อย่าลืมกล่าวถึงที่มาของสไลด์ (หลังจากนี้ เราต้องการให้ทุกคนอุดหนุนและใช้หนังสือของผู้แต่งด้านข้าง)
- ถ้าคุณโพสต์สไลด์ใด ๆ ในเว็บ, อย่าลืมกล่าวถึงว่า คุณแก้ไขจากสไลด์ต้นฉบับของเรา และ ระบุถึงลิขสิทธิ์ของเราด้วย

ขอขอบคุณและขอให้สนุก!

ณัฐนนท์ ลีลาตระกูล ผู้เรียบเรียง

สงวนลิขสิทธิ์ 2013

เนื้อหาทั้งหมดเป็นลิขสิทธิ์ของคณะวิทยาการสารสนเทศ



*Computer Networking: A
Top Down Approach*

6th edition

Jim Kurose, Keith Ross

Addison-Wesley

March 2012

Chapter 4: network layer

chapter goals:

- ❖ understand principles behind network layer services:
 - network layer service models
 - forwarding versus routing
 - how a router works
 - routing (path selection)
 - broadcast, multicast
- ❖ instantiation, implementation in the Internet

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

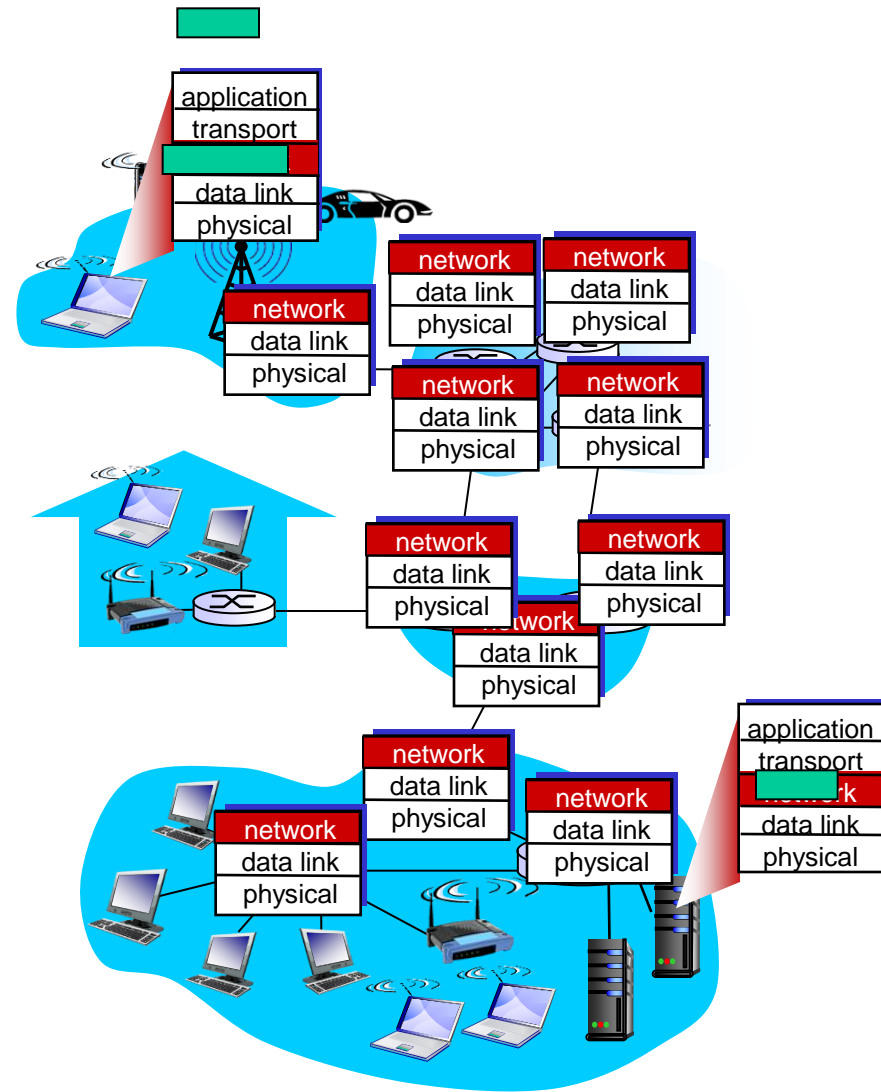
4.6 routing in the Internet

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Network layer

- ❖ เคลื่อนย้ายข้อมูลจากผู้ส่งไปยังผู้รับ
- ❖ ทางฝั่งผู้ส่งทำการ encapsulates segments ให้กลายเป็น datagrams
- ❖ ทางฝั่งผู้รับ นำส่ง segments ไปยัง transport layer
- ❖ network layer protocols มีอยู่ในทุกๆ host, router
- ❖ router ตรวจสอบ header fields ในทุกๆ IP datagrams ที่ถูกส่งผ่านตนเอง



Two key network-layer functions

❖ *forwarding*: เคลื่อนย้าย packets จาก router ที่ใส่ข้อมูล ไปยัง router ปลายทางอย่างถูกต้อง

❖ *routing*: กำหนดเส้นทางที่ packets จะใช้เดินทางจากต้นทางไปยังปลายทาง.

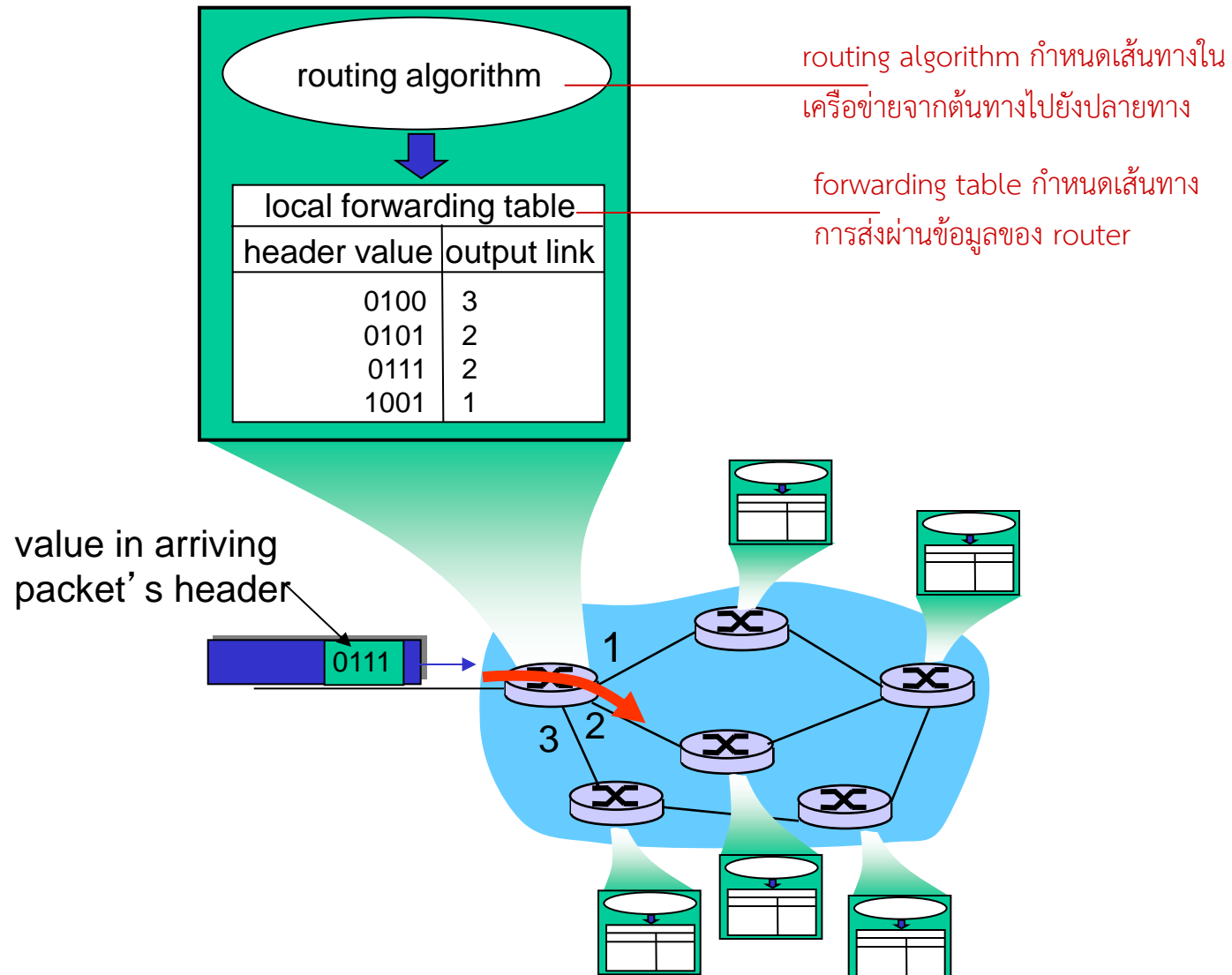
- *routing algorithms*

analogy:

❖ *routing*: กระบวนการวางแผนการเดินทางจากต้นทางไปยังปลายทาง

❖ *forwarding*: กระบวนการของการข้ามผ่านจุดสับเปลี่ยนเส้นทาง (เช่น ที่ switch, router)

Interplay between routing and forwarding



Connection setup (ข้าม)

- ❖ ในบางโครงสร้างเครือข่าย ถือเป็นฟังก์ชันที่สำคัญเป็นลำดับที่สาม :
 - ATM, frame relay, X.25
- ❖ ก่อนจะมีการไหลของข้อมูล ระหว่างเครื่องปลายทางสองเครื่อง และ Router ที่อยู่ระหว่างกลางทำการสร้าง การเชื่อมต่อเสมือน
 - Routers เข้าามีบทบาท
- ❖ network vs transport layer connection service:
 - *network*: เชื่อมต่อ 2 hosts (บางครั้งอาจมีบทบาทในการเชื่อมต่อระหว่าง routers ในกรณีของ VCs)
 - *transport*: เชื่อมต่อ 2 processes

Network service model

Q: โมเดลของการให้บริการอะไรที่ใช้สำหรับช่องทางการส่งดาต้าแกรมจากผู้ส่งไปยังผู้รับ?

ตัวอย่างการให้บริการสำหรับแต่ละดาต้าแกรม:

- ❖ รับประกันการส่ง
- ❖ รับประกันการส่งว่าค่าดีเลย์จะน้อยกว่า 40 msec

ตัวอย่างการให้บริการสำหรับการไหลของดาต้าแกรม:

- ❖ มีการส่งแบบเรียงลำดับ
- ❖ รับประกันอัตราแบนวิธขั้นต่ำในการส่ง
- ❖ จำกัดการเปลี่ยนแปลงของช่องว่างในแพคเกจ

Network layer service models:

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

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Connection, connection-less service

- ❖ *datagram* network ให้บริการบน network-layer แบบ *connectionless* service
- ❖ *virtual-circuit* network ให้บริการบน network-layer แบบ *connection* service
- ❖ ถึงแม้ว่า TCP/UDP จะเป็น connecton-oriented / connectionless transport-layer services, แต่:
 - *service*: ในระดับ host-to-host
 - *no choice*: network เป็นผู้ให้บริการ หรือใช้ผู้บริการอื่น
 - *implementation*: ในส่วนของ network core

Virtual circuits

“การสื่อสารระหว่างต้นทางไปยังปลายทาง สามารถทำงานได้ดี เหมือน telephone circuit”

- ประสิทธิภาพสูง
- network actions ระหว่างต้นทางและปลายทาง

- ❖ call setup, ก่อนจะมีการส่งข้อมูลระหว่างกัน
- ❖ each packet มีการระบุข้อมูล VC identifier (ไม่ระบุในกรณีที่เป็น host ปลายทาง)
- ❖ router แต่ละตัว ทั้งต้นทางและปลายทาง มีการเก็บ “state” ของแต่ละ connection
- ❖ link, router resources (bandwidth, buffers) อาจถูกจองไว้สำหรับ VC (dedicated resources = predictable service)

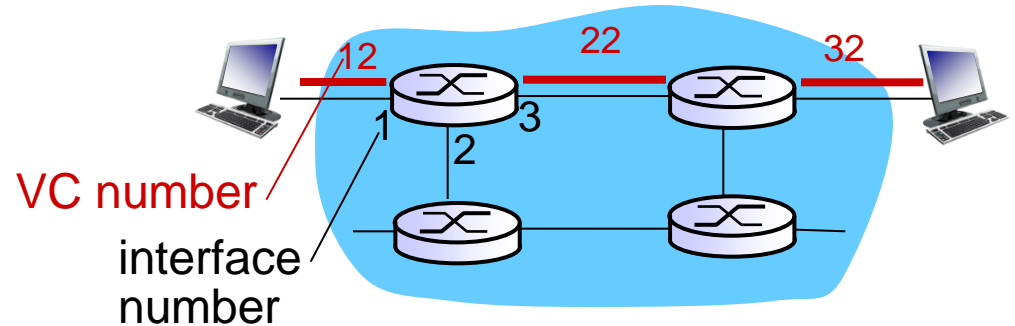
VC implementation (ข้าม)

a VC ประกอบไปด้วย:

1. *เส้นทาง* จากต้นทางไปยังปลายทาง
 2. *VC numbers*, 1 number สำหรับ แต่ละ link บนเครือข่าย
 3. *entries in forwarding tables* ใน routers บนเครือข่าย
- ❖ packet ที่เดินทางอยู่บน VC จะถูกระบุ VC number (แทนที่จะเป็น ที่อยู่ปลายทาง)
 - ❖ VC number สามารถเปลี่ยนแปลงได้ในแต่ละ link.
 - หมายเลข VC number ใหม่ จะมาจาก forwarding table

VC forwarding table

*forwarding table in
northwest router:*

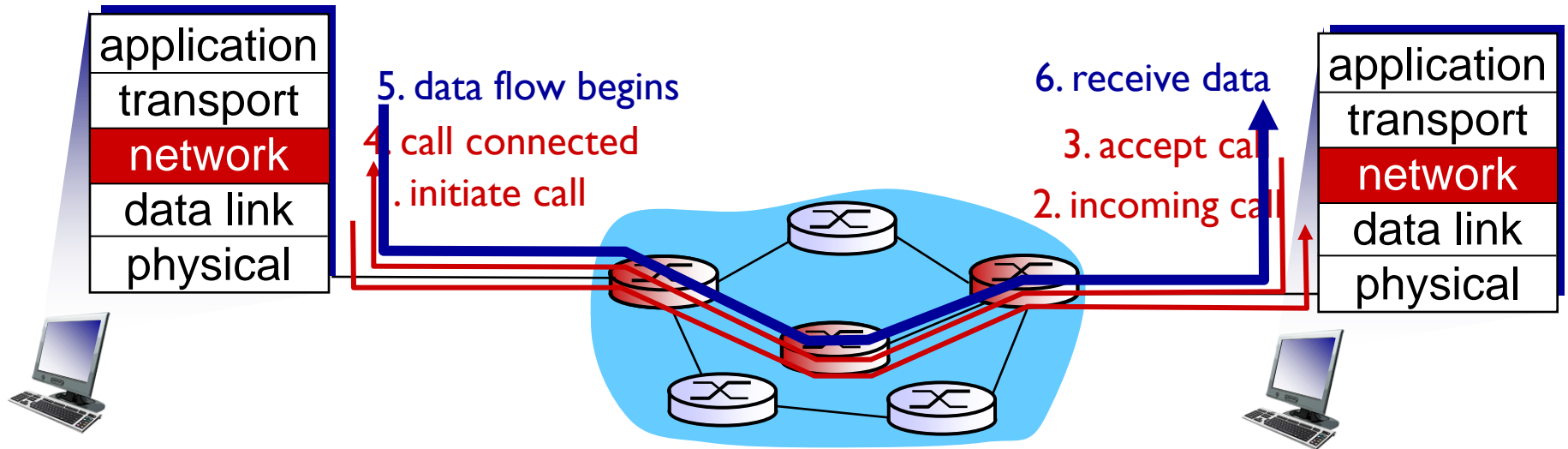


Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
...

VC routers maintain connection state information!

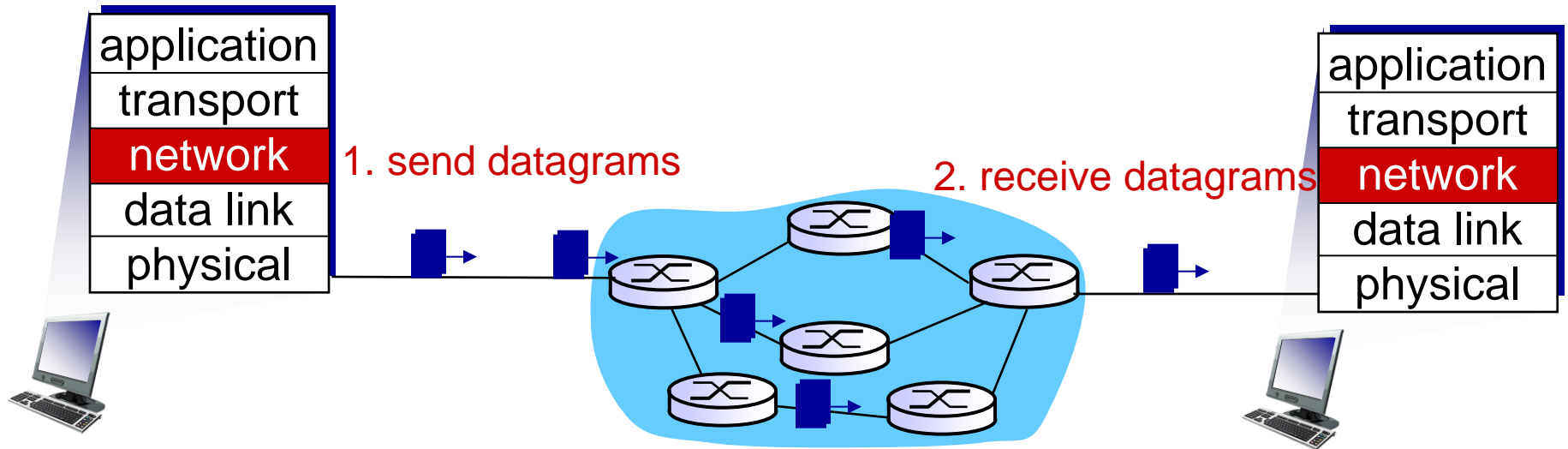
Virtual circuits: signaling protocols (จ้ำม)

- ❖ used to setup, maintain teardown VC
- ❖ used in ATM, frame-relay, X.25
- ❖ not used in today's Internet

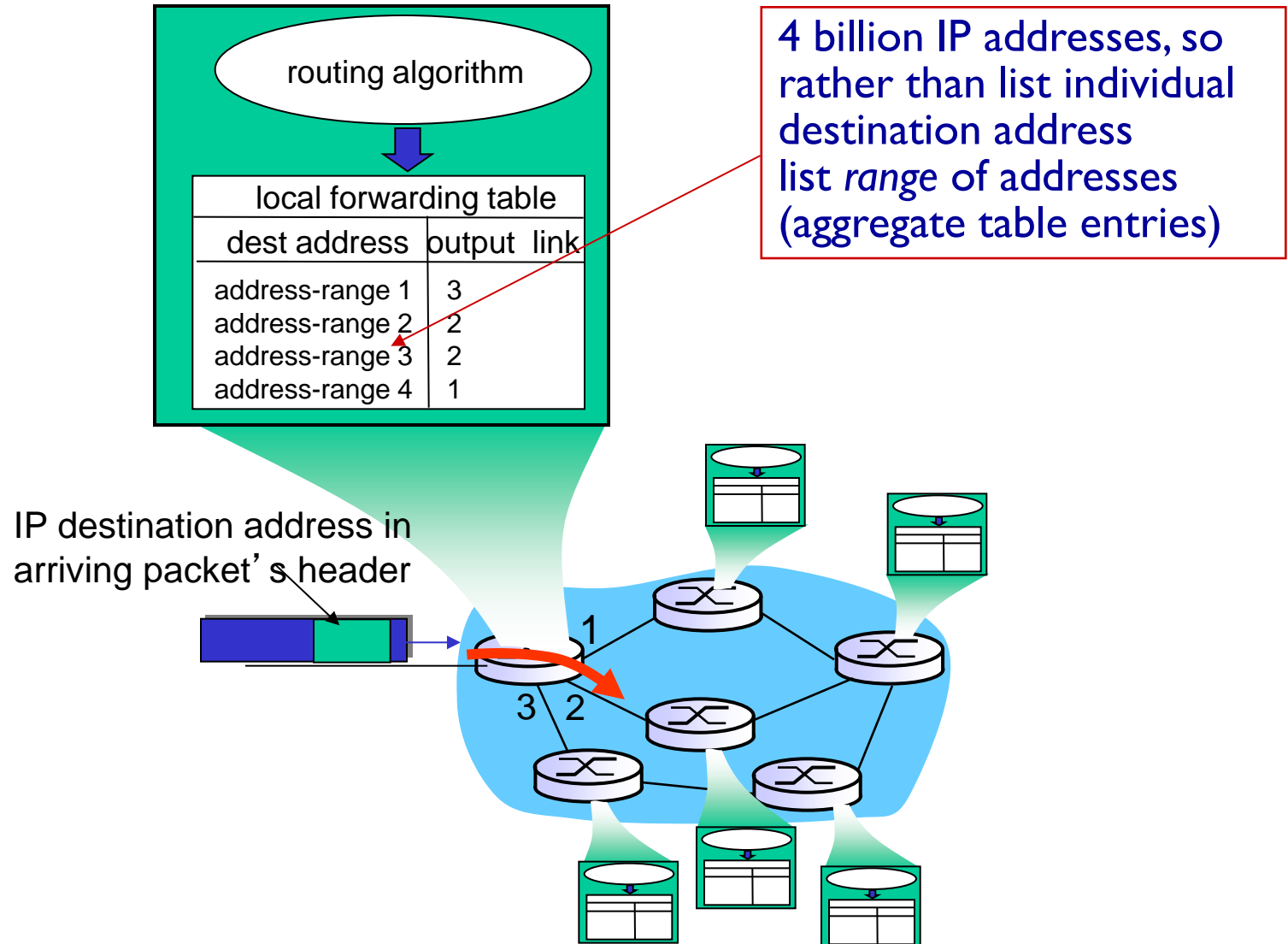


Datagram networks

- ❖ ชั้น network layer ไม่ต้อง call setup
- ❖ เราเตอร์ : ไม่เก็บสถานะการเชื่อมต่อของ end-to-end
 - ระดับเครือข่ายไม่ต้องมี “การเชื่อมต่อ”
- ❖ แพ็กเก็ตถูกส่งต่อไปโดยใช้ที่อยู่ของโฮสต์ปลายทาง



Datagram forwarding table



Datagram forwarding table

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

Q: but what happens if ranges don't divide up so nicely?

Longest prefix matching

longest prefix matching

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011*** *****	2
otherwise	3

examples:

DA: 11001000 00010111 00010110 10100001

which interface?

DA: 11001000 00010111 00011000 10101010

which interface?

Datagram หรือ VC network: ทำไม?

Internet (datagram)

- ❖ การแลกเปลี่ยนข้อมูลระหว่างคอมพิวเตอร์
 - “ยืดหยุ่น” บริการ, ไม่จำกัดความต้องการ.
- ❖ การเชื่อมต่อหลายแบบ
 - แต่ละการเชื่อมต่อคุณลักษณะแตกต่างกันไป
 - Internet ให้บริการแบบเดียว บริการให้ตรงกับแต่ละคุณลักษณะได้ยาก
- ❖ ระบบปลายทางต้องฉลาด
 - สามารถปรับได้, มีตัวควบคุมการดำเนินการ, มีการกู้คืนเมื่อเกิดข้อผิดพลาด
 - *จะทำให้ภายในเครือข่ายง่าย, แต่ไปซับซ้อนที่ “edge”*

ATM (VC)

- ❖ วิวัฒนาการมาจากโทรศัพท์
- ❖ การสนทนาของมนุษย์ :
 - Delay เป็นเรื่องสำคัญ, ต้องการความน่าเชื่อถือ
 - จำเป็นสำหรับรับประกันการบริการ
- ❖ ระบบปลายทางไม่ต้องฉลาด
 - โทรศัพท์
 - *ความซับซ้อนภายในเครือข่าย*

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

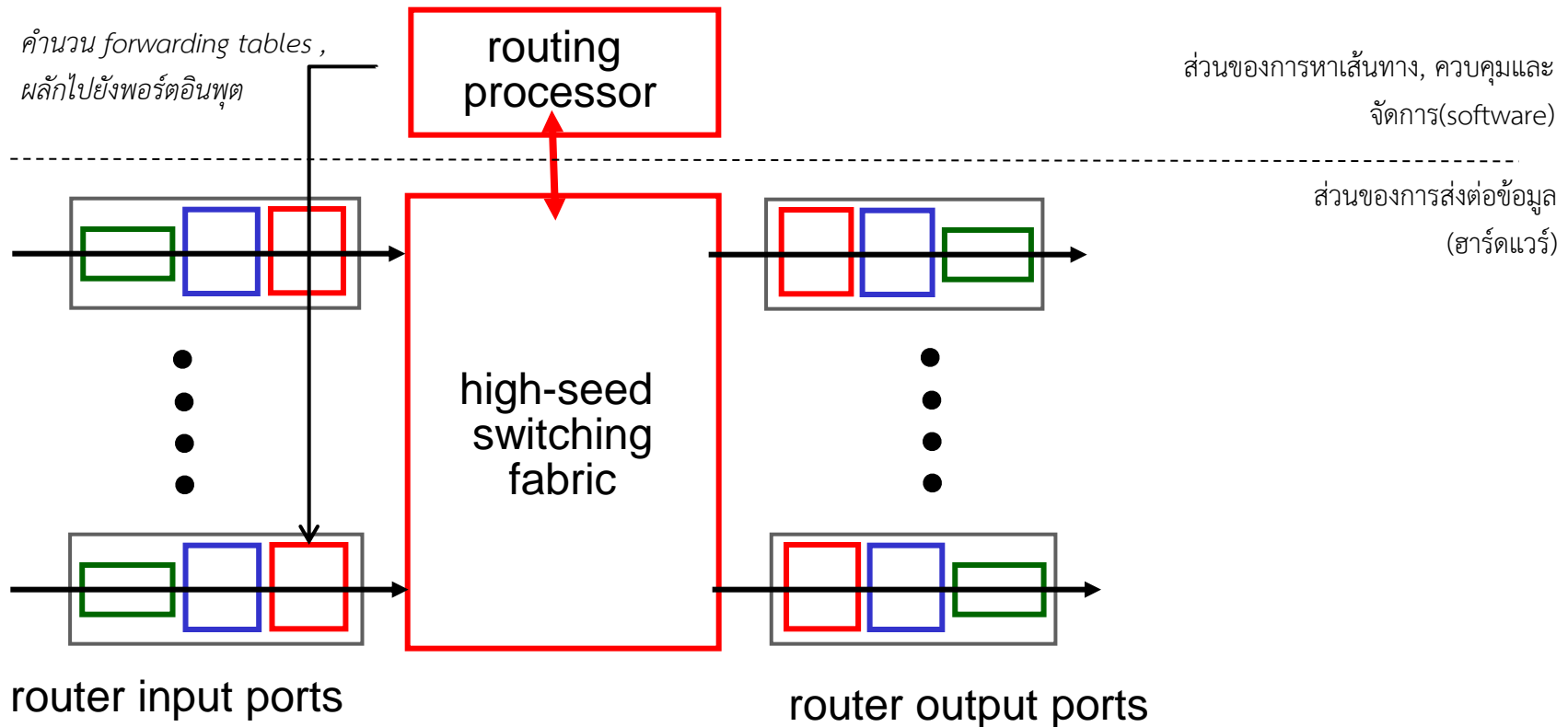
- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

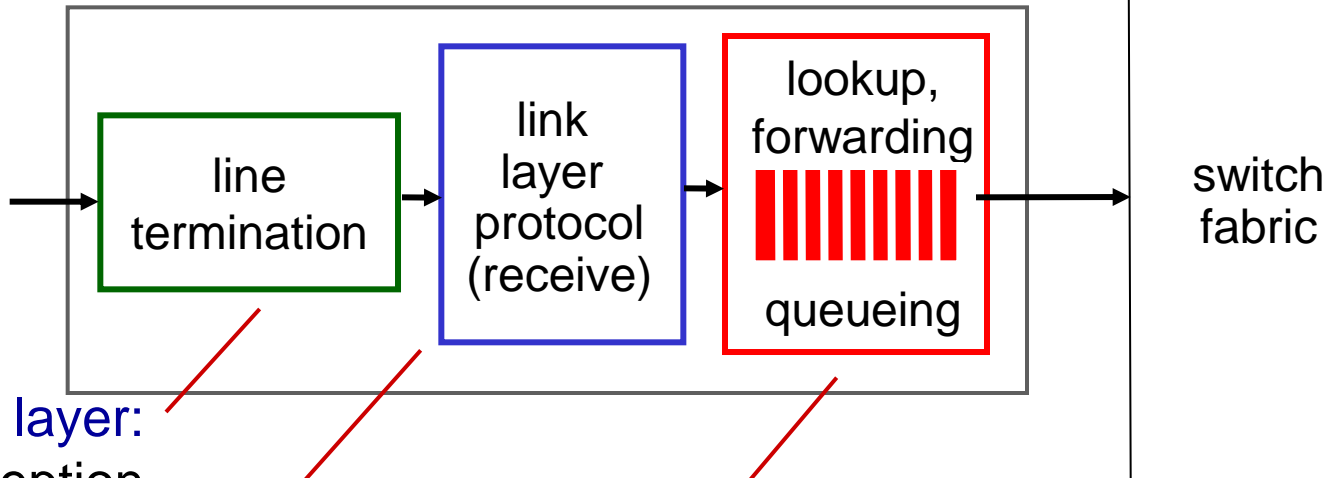
ภาพรวมของสถาปัตยกรรมเราเตอร์

สองหน้าที่ที่สำคัญของเราเตอร์ :

- ❖ ใช้ขั้นตอนวิธีในการหาเส้นทาง / โพรโทคอล (RIP, OSPF, BGP)
- ❖ ส่งต่อดาต้าแกรมจากลิงค์ขาเข้าถึงลิงค์ขาออก



Input port functions



physical layer:
bit-level reception

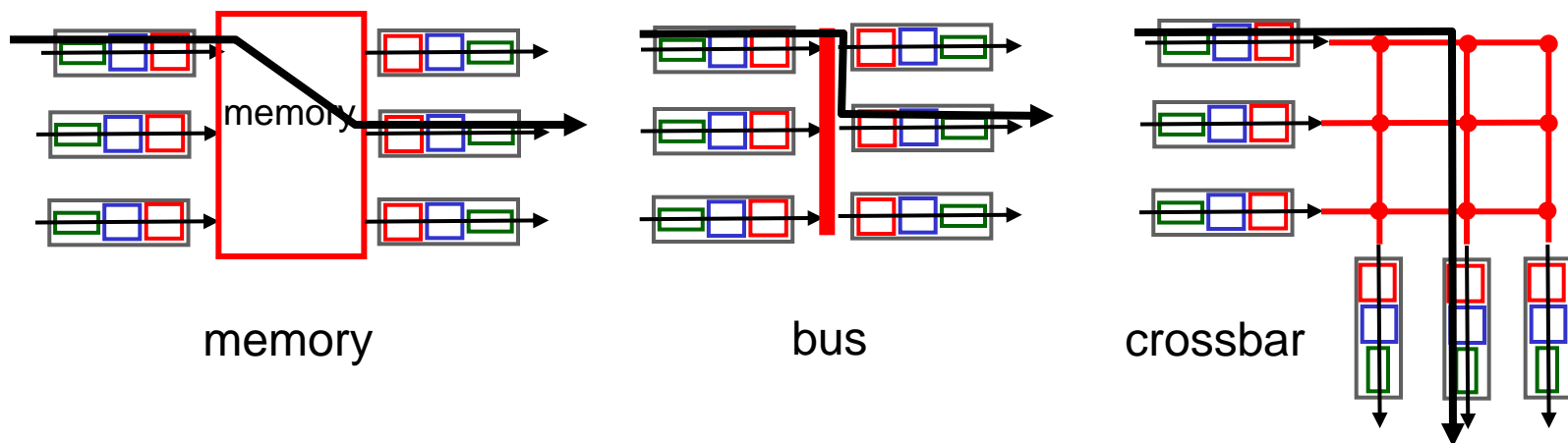
data link layer:
e.g., Ethernet
see chapter 5

decentralized switching:

- ❖ given datagram dest., lookup output port using forwarding table in input port memory (*“match plus action”*)
- ❖ goal: complete input port processing at ‘line speed’
- ❖ queuing: if datagrams arrive faster than forwarding rate into switch fabric

Switching fabrics

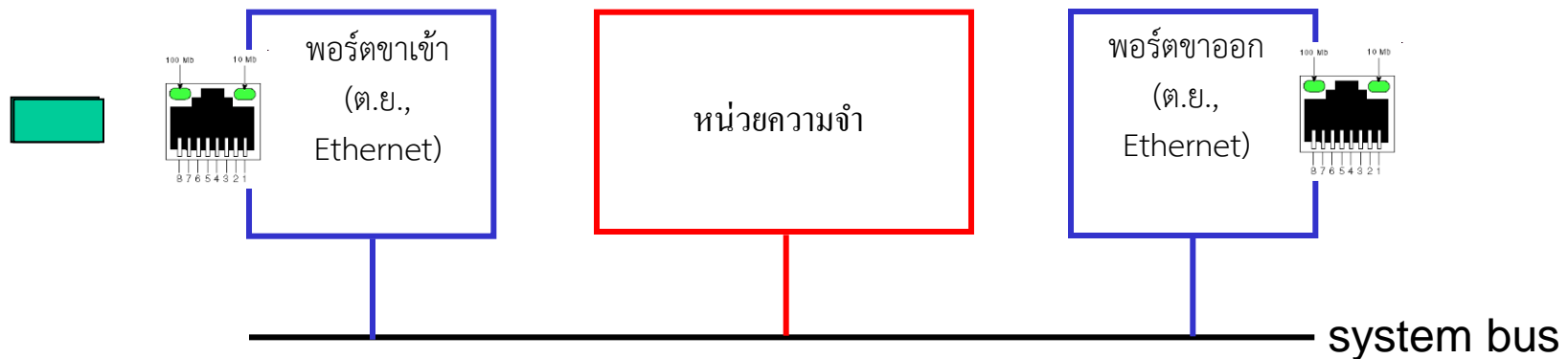
- ❖ transfer packet from input buffer to appropriate output buffer
- ❖ switching rate: rate at which packets can be transfer from inputs to outputs
 - often measured as multiple of input/output line rate
 - N inputs: switching rate N times line rate desirable
- ❖ three types of switching fabrics



การ Switching ผ่านหน่วยความจำ

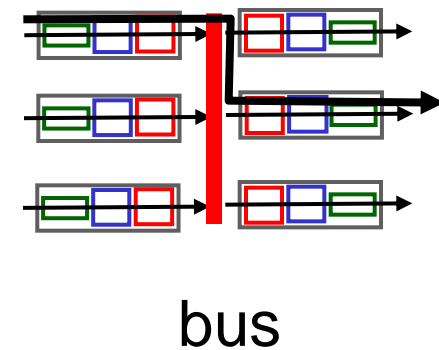
เราเตอร์รุ่นแรก:

- ❖ คอมพิวเตอร์แบบดั้งเดิมทำการ Switching ภายใต้การควบคุมโดยตรงของ CPU
- ❖ แพ็คเก็ตถูกคัดลอกไปยังหน่วยความจำของระบบ
- ❖ ความเร็วถูกจำกัดโดยหน่วยความจำและแบนด์วิดท์ (2 bus crossings ต่อดาต้าแกรม)



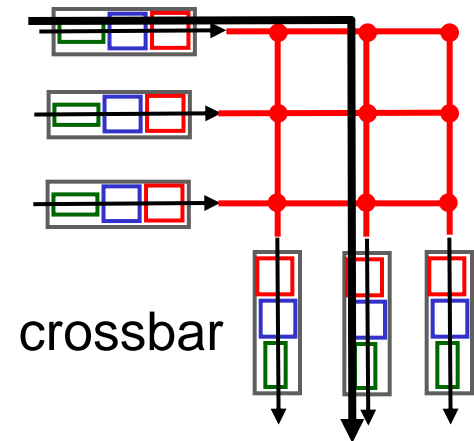
Switching ผ่าน bus

- ❖ รับดาต้าแกรมจากหน่วยความจำพอร์ตขาเข้า ถึงหน่วยความจำพอร์ตขาออกผ่านบัสนี้ใช้ร่วมกัน
- ❖ *bus contention*: ความเร็วในการ switching ถูกจำกัดโดยบัสแบนด์วิดท์
- ❖ 32 Gbps bus, Cisco 5600: ความเร็วที่เพียงพอสำหรับการเข้าถึงเราเตอร์ขององค์กร

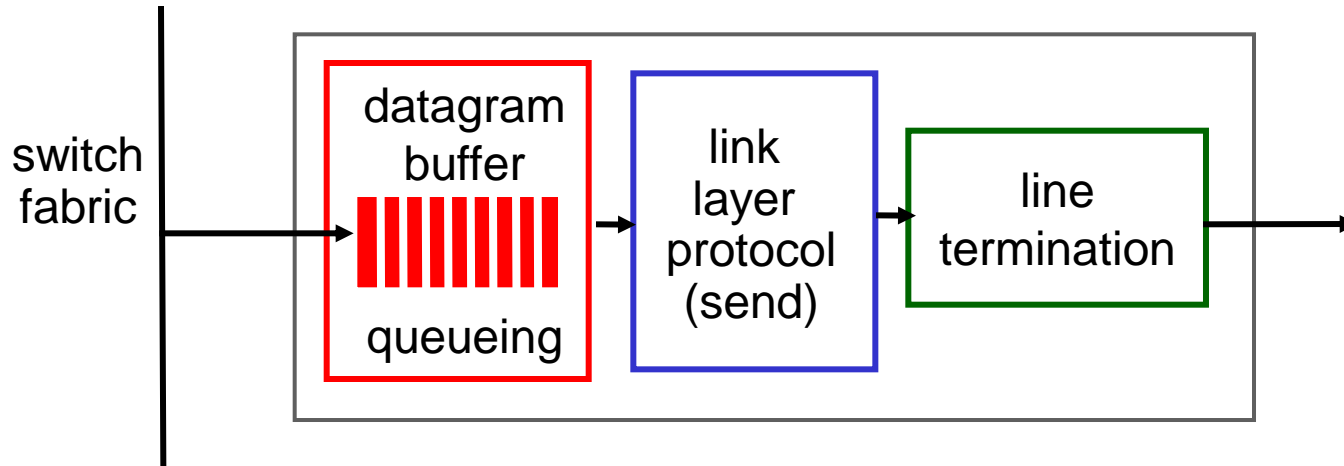


Switching ผ่านการเชื่อมโยงของเครือข่าย

- ❖ เอาชนะข้อจำกัดของบัสแบนด์วิดท์
- ❖ เครือข่ายต้นไทร, คานประตู่, การเชื่อมต่ออื่นๆ ในระยะแรก พัฒนาขึ้นเพื่อเชื่อมต่อการประมวลผลในแบบมัลติโพรเซสเซอร์
- ❖ ออกแบบที่ทันสมัย: ใส่ชิ้นดาต้าแกรมเข้าสู่เซลล์ที่จำกัดขนาด และสานกันเหมือนผ้าไหม
- ❖ Cisco 12000: switches 60 Gbps ผ่านการเชื่อมโยงแบบเครือข่าย

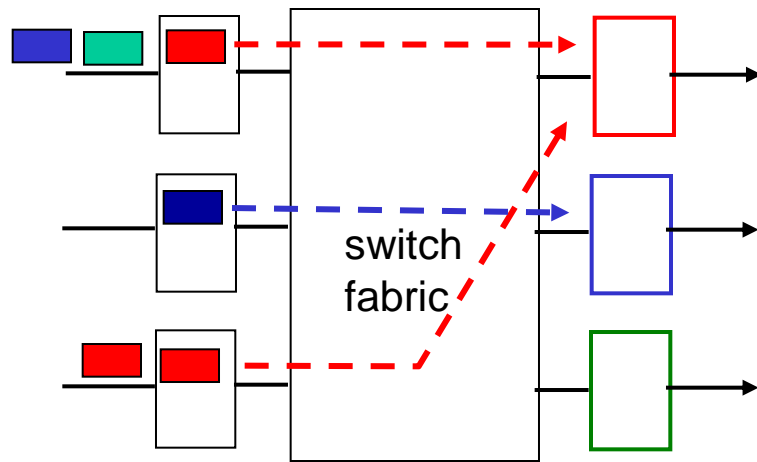


พอร์ตขาออก

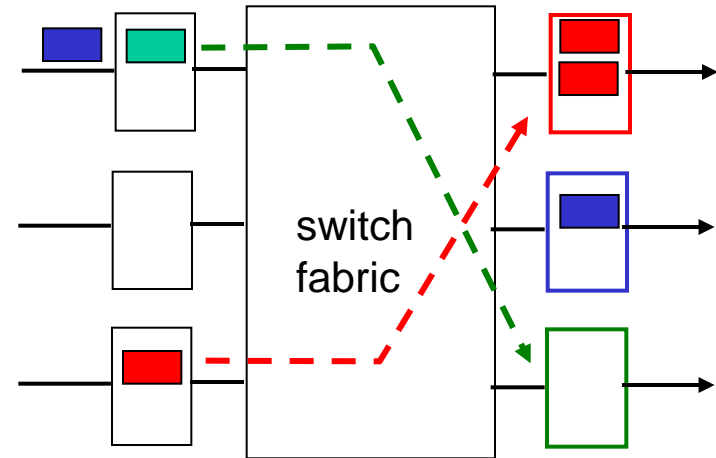


- ❖ *การกำหนดบัฟเฟอร์* จำเป็นต้องใช้เมื่อ datagrams มาจาก fabric เร็วกว่าอัตราการส่ง
- ❖ *การกำหนดตารางเวลา* เลือกดาต้าแกรมในกลุ่มของคิวที่จัดไว้สำหรับการส่ง

คิวที่ Output port



at t , packets move
from input to output



one packet time later

- ❖ มีการสำรองข้อมูลเมื่ออัตราการส่งข้อมูลของ switch เกินกว่าความเร็วของสายสัญญาณ
- ❖ เกิดการรอคิว(ล่าช้า)และสูญหายเนื่องจาก buffer ของ output port เต็ม!

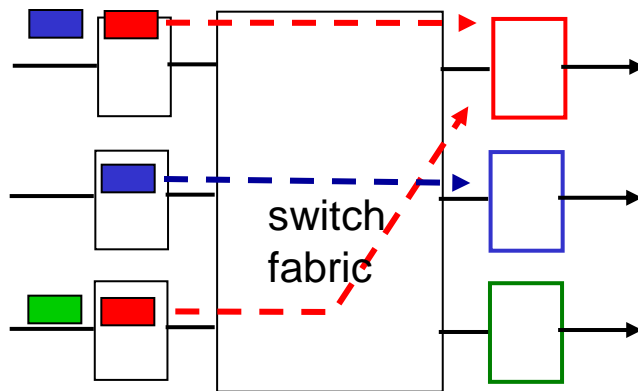
ขนาด buffer ควรมีเท่าไร? (ข้าม)

- ❖ RFC 3439 rule of thumb: ค่าเฉลี่ยของ buffer เท่ากับ “typical” RTT (say 250 msec) คูณกับขนาดของ link
 - e.g., C = 10 Gbps link: 2.5 Gbit buffer
- ❖ recent recommendation: with N flows, ขนาดของ buffer เท่ากับ

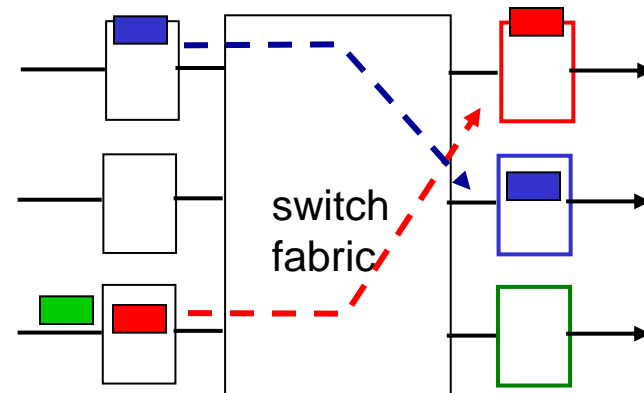
$$\frac{RTT \cdot C}{\sqrt{N}}$$

คิวที่ Input port

- ❖ fabric ทำงานช้ากว่าข้อมูลที่เข้า input ports -> จะเกิดคิวที่ input ports
 - *เกิดการล่าช้าของคิวและสูญหายเนื่องจาก buffer overflow!*
- ❖ **Head-of-the-Line (HOL) blocking:** ดาต้าแกรมที่เข้าคิวรออยู่ส่วนหน้าของคิวไปขัดกับตัวอื่น ทำให้ไม่สามารถส่งต่อไปได้



output port contention:
สามารถส่งดาต้าแกรมสีแดงได้เพียงตัวเดียว
แพคเกจสีแดงตัวล่างถูกบล็อก



one packet time later:
แพคเกจสีแดงเสียเวลาจากการบล็อก
โดย HOL

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

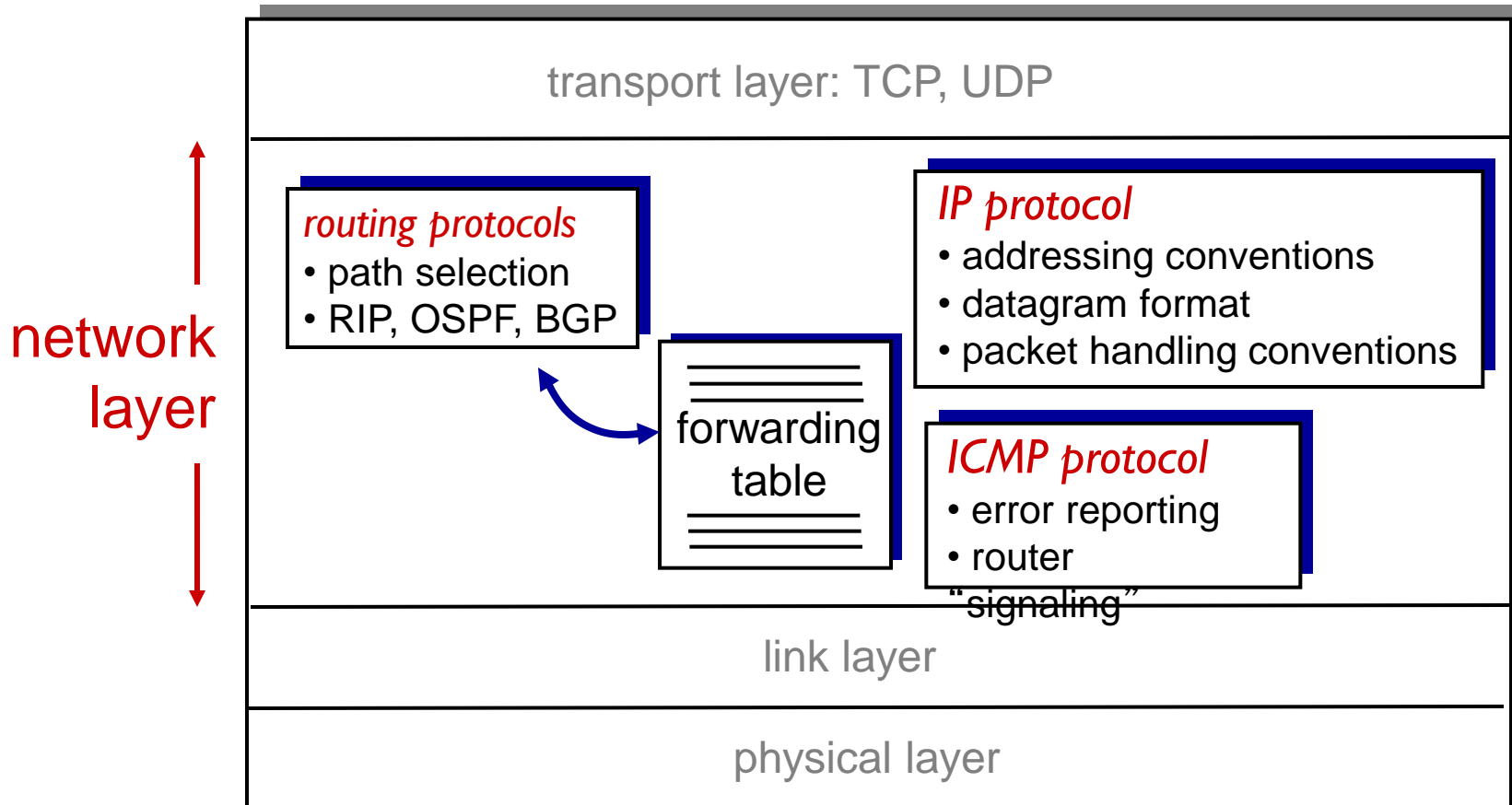
4.6 routing in the Internet

- RIP
- OSPF
- BGP

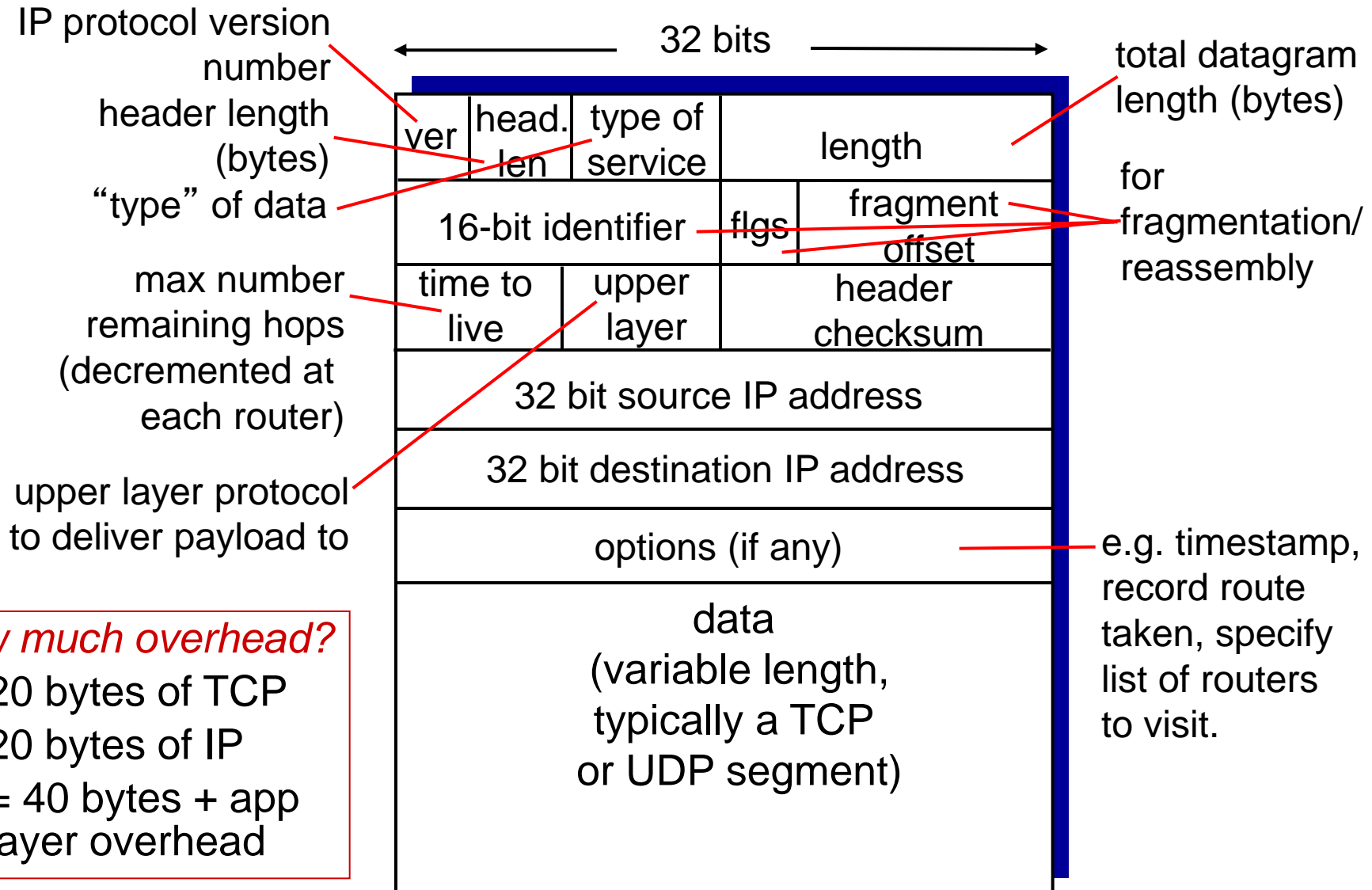
4.7 broadcast and multicast routing

The Internet network layer

หลักการทำงานของโฮส ,เราเตอร์ ในชั้นเน็ตเวิร์ค:

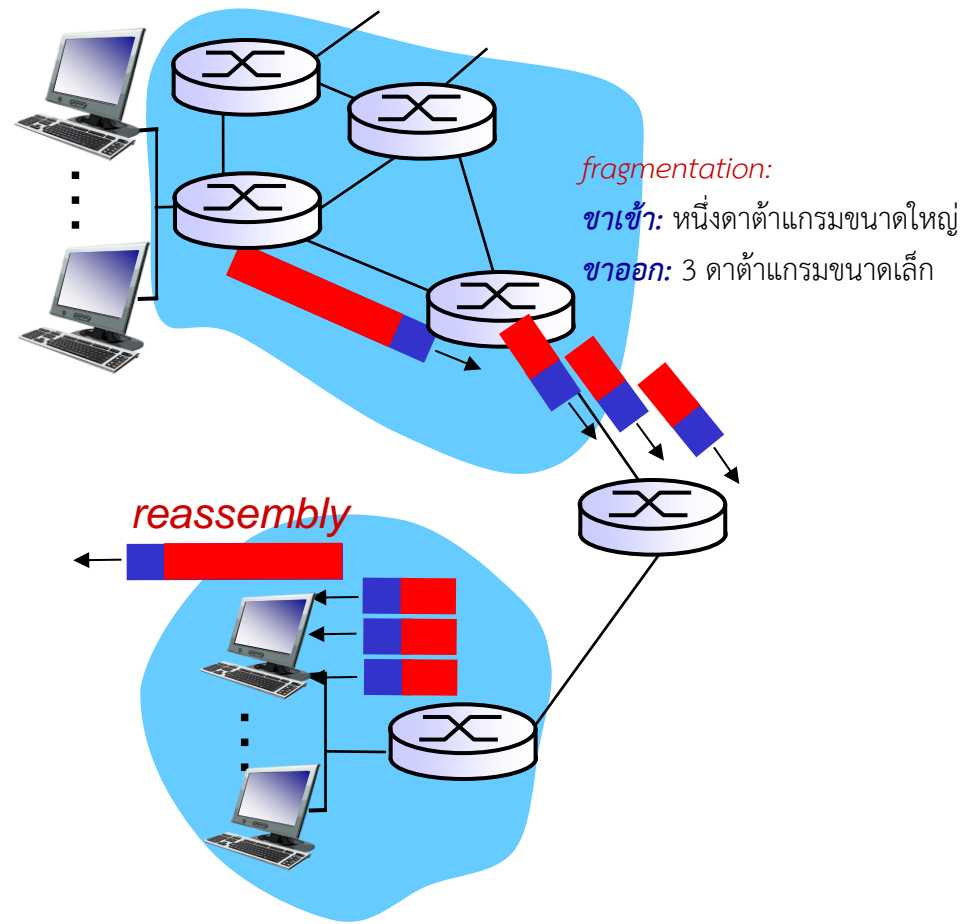


IP datagram format



IP fragmentation, reassembly

- ❖ การเชื่อมต่อของเครือข่ายจะมี MTU (max.transfer size) – ขนาดใหญ่สุดที่เป็นไปได้ของ link-level frame
 - ชนิดของการเชื่อมต่อที่ต่างกันจะมี MTU ไม่เท่ากัน
- ❖ IP datagram ขนาดใหญ่จะถูกแบ่งออกเป็นชิ้นส่วน
 - หนึ่งดาต้าแกรมจะกลายเป็นหลายๆ ดาต้าแกรม
 - จะถูกรวมอีกครั้งเมื่อถึงปลายทาง
 - IP header bits ใช้เพื่อกำหนดการเรียงชิ้นส่วนให้เป็นไปตามลำดับ



IP fragmentation, reassembly

ตัวอย่าง:

- ❖ 4000 byte datagram
- ❖ MTU = 1500 bytes

	length	ID	fragflag	offset	
	=4000	=x	=0	=0	

หนึ่งดาต้าแกรมขนาดใหญ่กลายเป็นดาต้าแกรมขนาดเล็กหลายๆชิ้น

1480 bytes in
data field

offset =
 $1480/8$

	length	ID	fragflag	offset	
	=1500	=x	=1	=0	

	length	ID	fragflag	offset	
	=1500	=x	=1	=185	

	length	ID	fragflag	offset	
	=1040	=x	=0	=370	

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

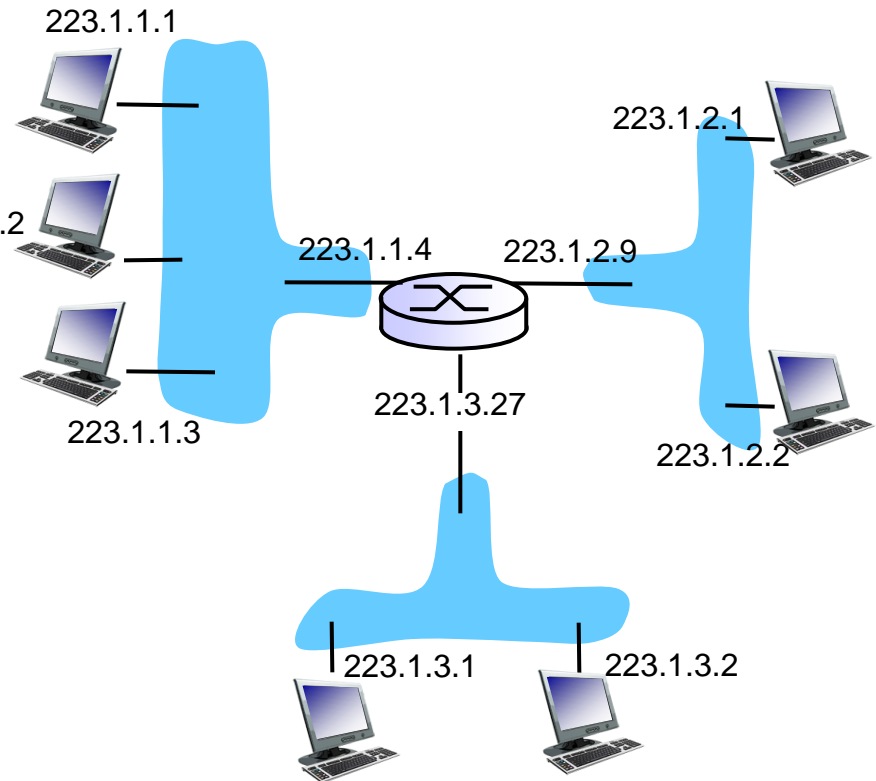
4.6 routing in the Internet

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

IP addressing: introduction

- ❖ *IP address*: เป็นเลข 32-bit ป่งบอกถึงโฮสหรือเราเตอร์
- ❖ *interface*: การเชื่อมต่อระหว่างโฮสหรือเราเตอร์กับลิงค์ทางกายภาพ
 - โดยทั่วไปเราเตอร์จะมีหลาย interface
 - โฮสจะมีเพียงหนึ่งหรือสอง interface เช่น wired Ethernet, wireless 802.11
- ❖ *IP address จะเกี่ยวข้องกับแต่ละ interface*

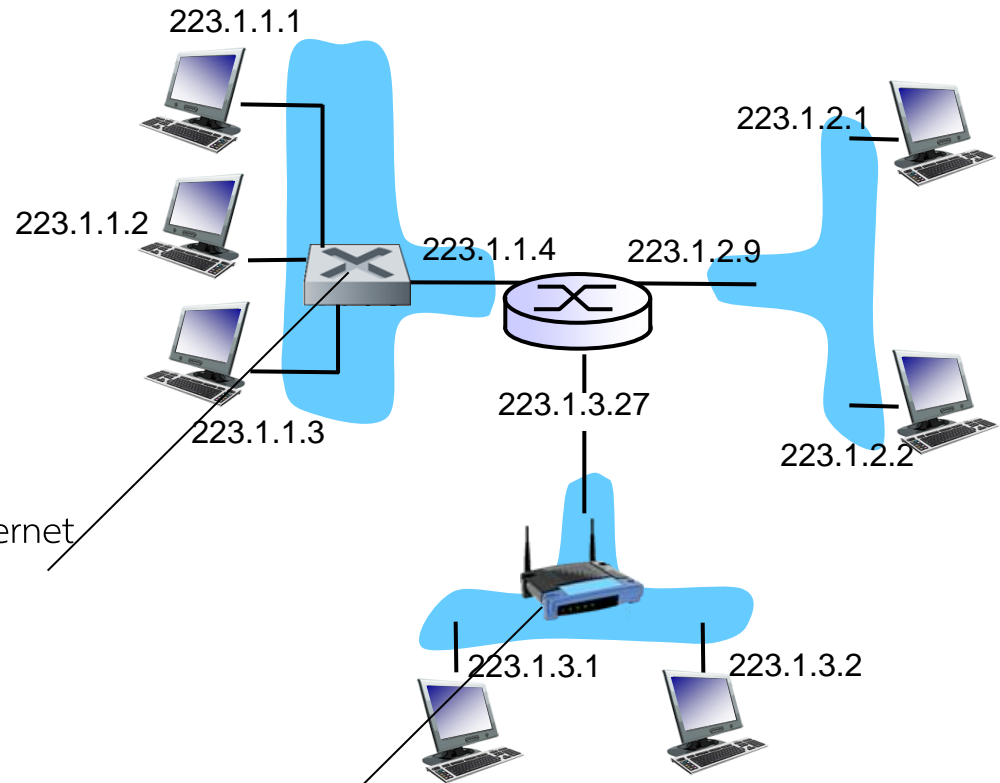


$$223.1.1.1 = \underbrace{11011111}_{223} \underbrace{00000001}_1 \underbrace{00000001}_1 \underbrace{00000001}_1$$

IP addressing: introduction

Q: interface เชื่อมต่อกันอย่างไร

A: เราจะศึกษาเกี่ยวกับมันใน chapter 5, 6



A: wired Ethernet interfaces เชื่อมต่อโดย Ethernet switches

ในตอนนี้: ยังไม่ต้องกังวลถึงการเชื่อมต่อของแต่ละ interface

A: wireless WiFi interfaces เชื่อมต่อโดย WiFi base station

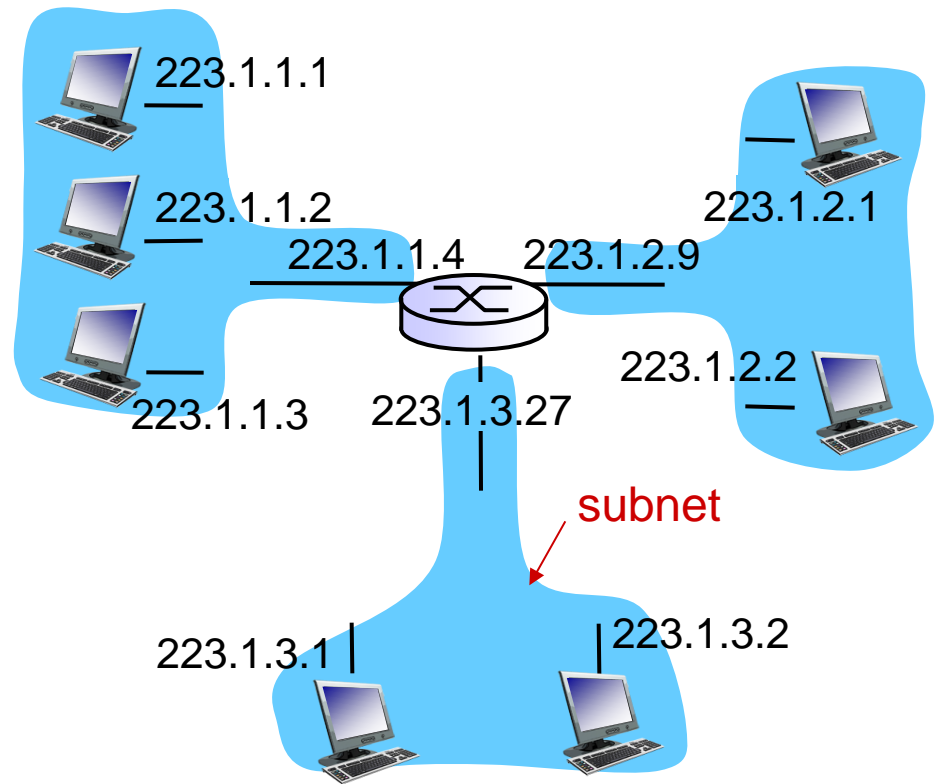
Subnets

❖ IP address:

- subnet part - high order bits
- host part - low order bits

❖ subnet คืออะไร ?

- อุปกรณ์ที่มี ip address อยู่ใน subnet part เดียวกัน
- สามารถเชื่อมต่อถึงกันได้โดยไม่ต้องอาศัยเราเตอร์

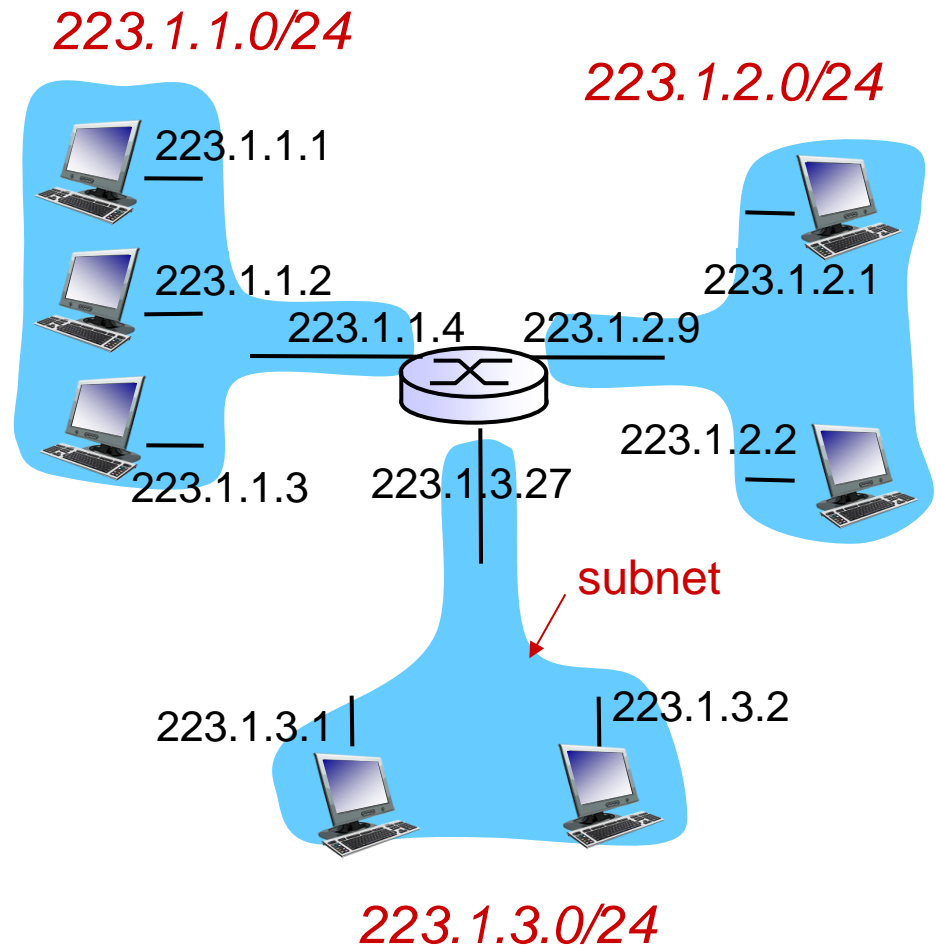


ระบบเครือข่ายนี้ประกอบด้วย 3 subnets

Subnets

วิธีการ

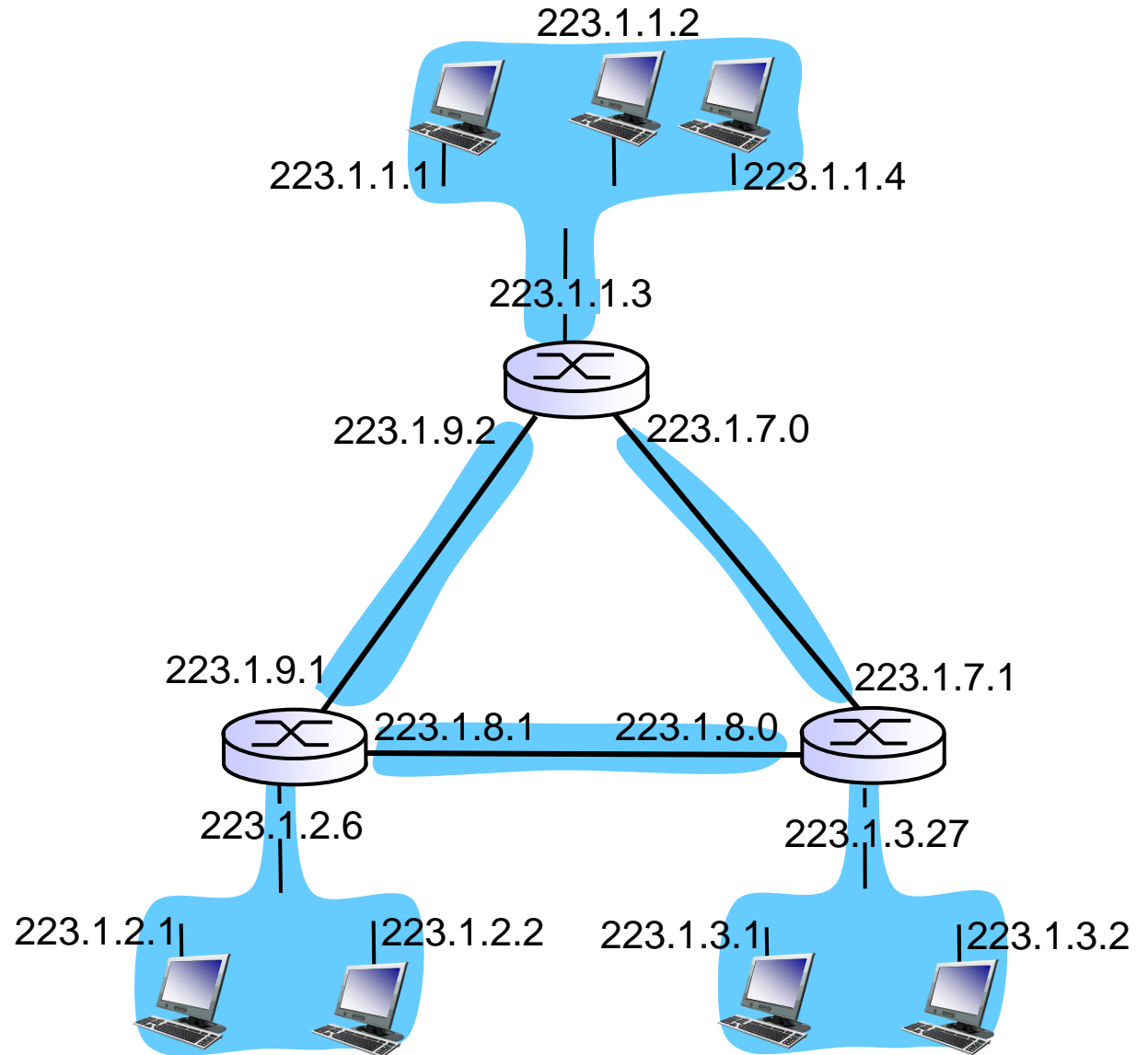
- ❖ ในการกำหนดซับเน็ต, ในแต่ละอินเทอร์เน็ตที่แยกออกจากเราท์เตอร์หรือโฮสต์, เป็นการสร้าง network ย่อยๆแยกออกมาจาก network หลัก
- ❖ แต่ละเน็ตเวิร์คที่ถูกแยกออกมาเรียกว่า ซับเน็ต



subnet mask: /24

Subnets

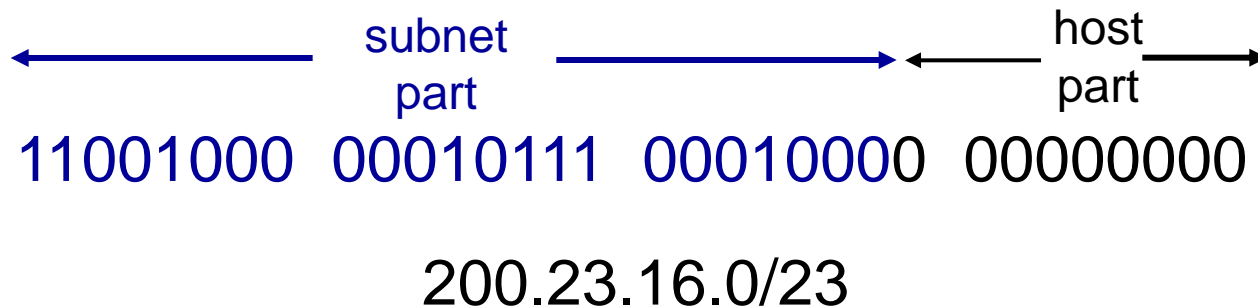
how many?



IP addressing: CIDR

CIDR: Classless InterDomain Routing

- การแบ่งซับเน็ตสามารถกำหนดขอบเขตได้ตามความพอใจ
- มีรูปแบบคือ: a.b.c.d/x, โดยที่ x เป็นจำนวนบิตในส่วนแบ่งของซับเน็ต



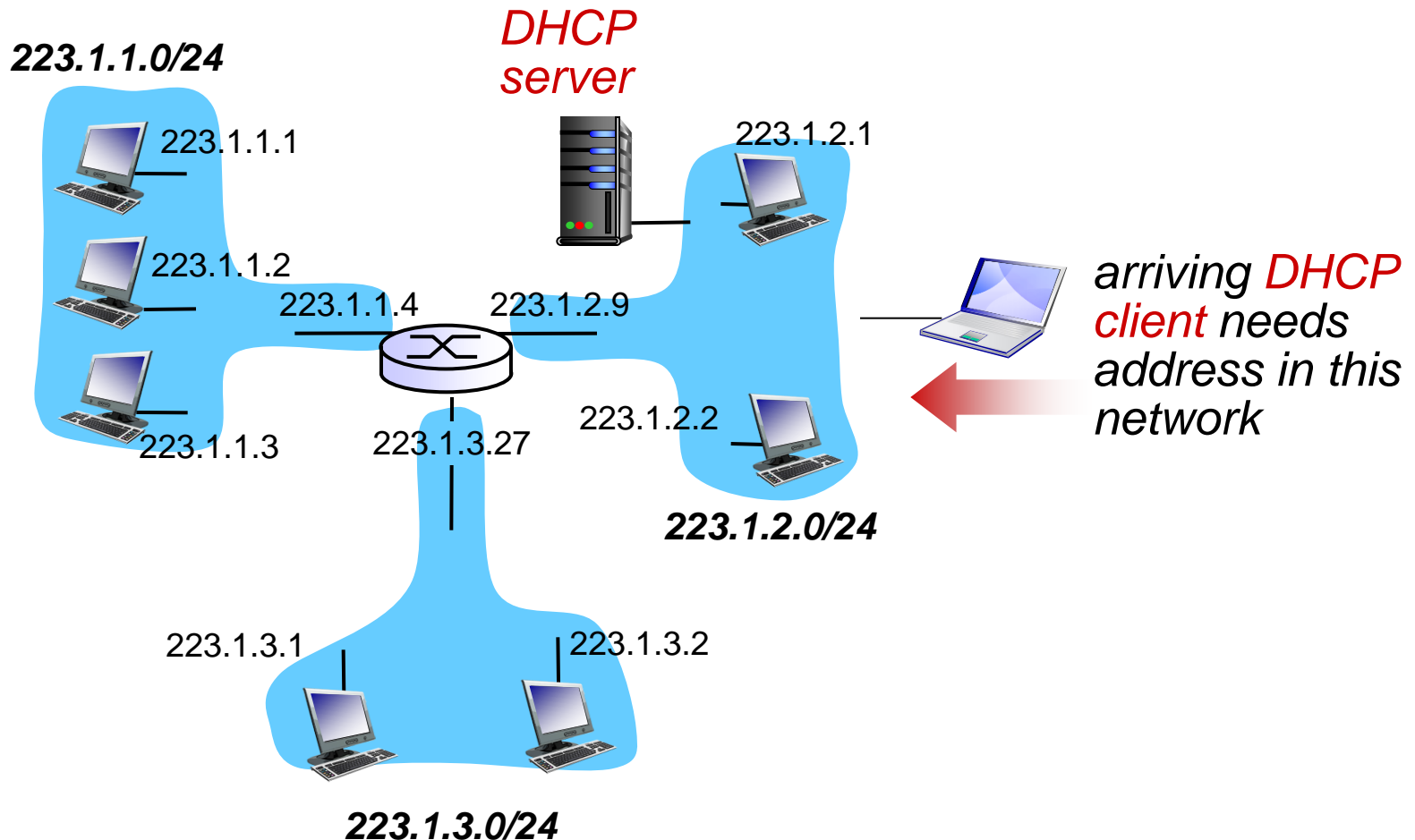
IP addresses: how to get one?

- ❖ Q: โฮสต์จะได้รับไอพีมาได้อย่างไร?
- ❖ คอนฟิกแบบตายตัว โดยผู้ดูแลระบบ
 - Windows: control-panel->network->configuration->tcp/ip->properties
 - UNIX: /etc/rc.config
- ❖ DHCP: Dynamic Host Configuration Protocol: ได้รับไอพีมาจากเซิร์ฟเวอร์แบบไดนามิก
 - “Plug-and-Play”

DHCP: Dynamic Host Configuration Protocol

- ❖ *goal*: อนุญาตให้โฮสต์ได้รับไอพีแบบไดนามิกจากเซิร์ฟเวอร์เมื่อมีการเชื่อมต่อเข้ามาในโครงข่าย
 - สามารถร้องขอใหม่เมื่อหมดอายุ
 - ยอมให้นำแอดเดรสกลับมาให้ใหม่ (only hold address while connected/“on”)
 - รองรับผู้ใช้งานแบบเคลื่อนที่ ซึ่งต้องการเชื่อมต่อมายังโครงข่าย
- ❖ *DHCP overview*:
 - โฮสต์ส่งบรอดแคสต์ออกไป “DHCP discover” msg [optional]
 - เซิร์ฟเวอร์ DHCP ตอบกลับมา “DHCP offer” msg [optional]
 - โฮสต์ร้องขอไอพี : “DHCP request” msg
 - เซิร์ฟเวอร์ DHCP ส่งแอดเดรสกลับมา: “DHCP ack” msg

DHCP client-server scenario



DHCP client-server scenario

DHCP server: 223.1.2.5

DHCP discover

src : 0.0.0.0, 68
dest.: 255.255.255.255,67
yiaddr: 0.0.0.0
transaction ID: 654

arriving
client



DHCP offer

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 654
lifetime: 3600 secs

DHCP request

src: 0.0.0.0, 68
dest.: 255.255.255.255, 67
yiaddr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

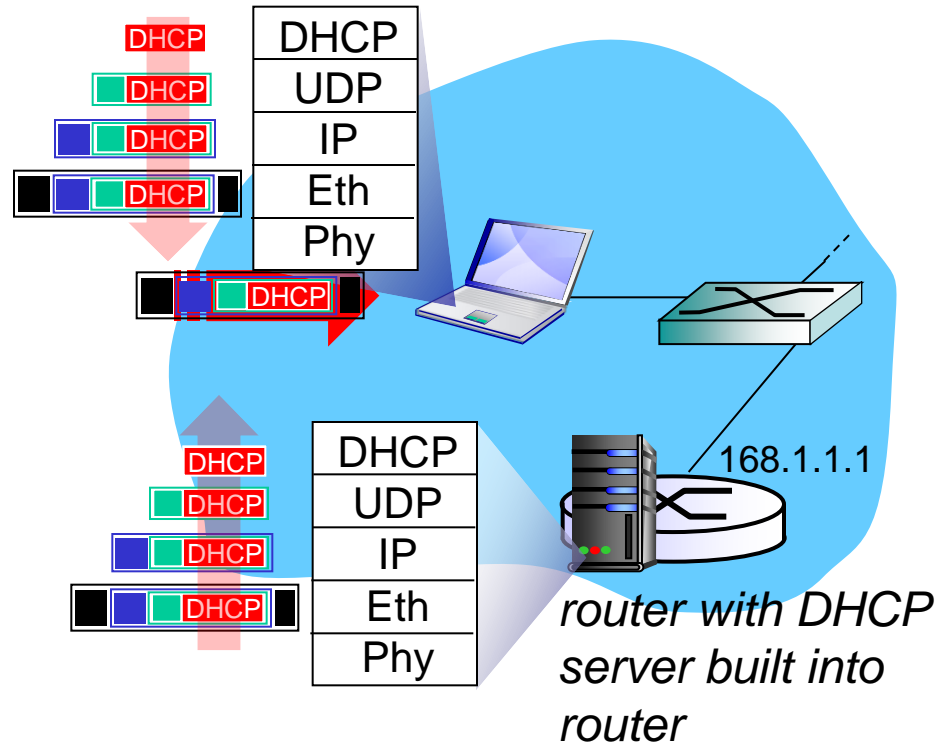
DHCP ACK

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

DHCP: เป็นมากกว่าการจ่ายไอพี

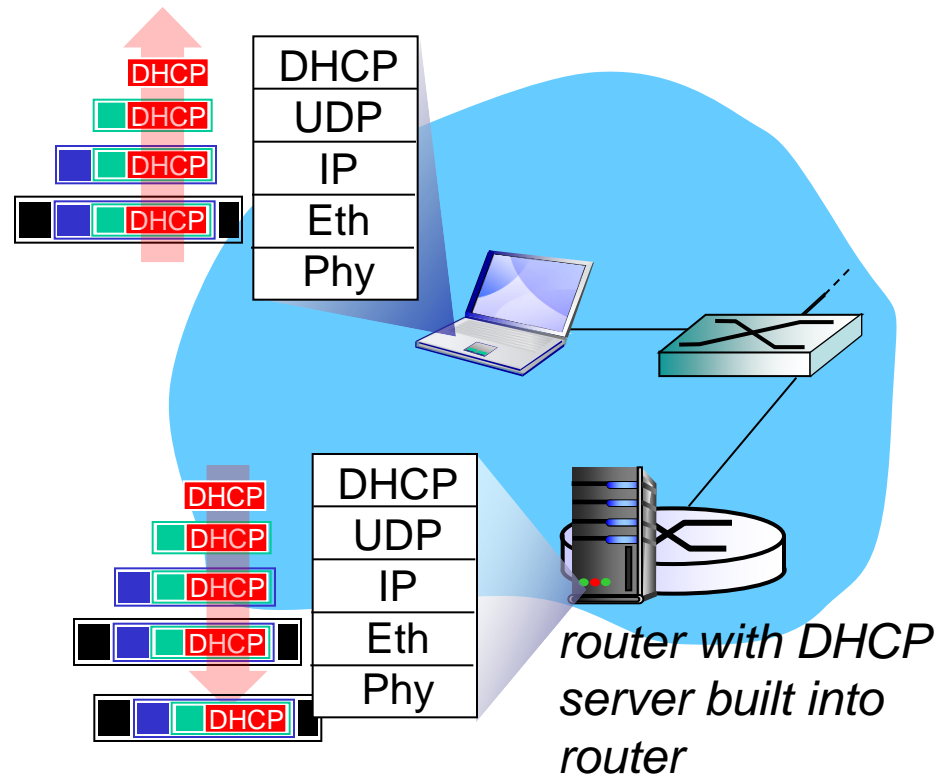
- ❖ DHCP สามารถส่งค่ากลับไปได้มากกว่าแค่การจัดสรรไอพีบนชั้นเน็ต:
 - ไอพีของเราท์เตอร์ที่เป็นเกตเวย์
 - ชื่อและไอพีของ DNS sever
 - network mask (แสดงถึงส่วนที่เป็น network และส่วนของโฮสต์)

DHCP: example



- ❖ laptop ที่เชื่อมต่อเข้ามาต้องการไอพี, ไอพีของเราเตอร์, ไอพีของเซิร์ฟเวอร์ DNS: ด้วย DHCP
- ❖ DHCP ถูกห่อหุ้มโดยUDP, หุ้มโดย IP, หุ้มโดย 802.1 Ethernet
- ❖ เฟรมบรอดแคสของอีเทอร์เน็ต (ปลายทาง: FFFFFFFFFFFFFFFF) on LAN, เราเตอร์ที่รัน DHCP Server จะได้รับเฟรมนี้
- ❖ จากอีเทอร์เน็ตถอด ออกเป็นไอพีถอดออกเป็น UDP และสุดท้ายออกเป็น DHCP

DHCP: example



- ❖ เซิร์ฟเวอร์ DHCP สร้าง DHCP ACK ประกอบด้วย ไอพีของไคลเอนต์, ไอพีของเราต์เตอร์ตัวแรกในโครงข่าย, ชื่อและไอพีของเซิร์ฟเวอร์ DNS
- ❖ การ encapsulation ของเซิร์ฟเวอร์ DHCP, เฟรมจะถูกส่งไปยังไคลเอนต์, การdemux จะเกิดขึ้นที่ไคลเอนต์ DHCP
- ❖ ไคลเอนต์จะรู้ถึงไอพีของตน ชื่อและไอพีของ เซิร์ฟเวอร์ DNS , ไอพีของเราท์เตอร์เกตเวย์

DHCP: Wireshark output (home LAN, บ้าน)

Message type: **Boot Request (1)**

Hardware type: Ethernet

Hardware address length: 6

Hops: 0

Transaction ID: 0x6b3a11b7

Seconds elapsed: 0

Bootp flags: 0x0000 (Unicast)

Client IP address: 0.0.0.0 (0.0.0.0)

Your (client) IP address: 0.0.0.0 (0.0.0.0)

Next server IP address: 0.0.0.0 (0.0.0.0)

Relay agent IP address: 0.0.0.0 (0.0.0.0)

Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)

Server host name not given

Boot file name not given

Magic cookie: (OK)

Option: (t=53,l=1) **DHCP Message Type = DHCP Request**

Option: (61) Client identifier

Length: 7; Value: 010016D323688A;

Hardware type: Ethernet

Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)

Option: (t=50,l=4) Requested IP Address = 192.168.1.101

Option: (t=12,l=5) Host Name = "nomad"

Option: (55) Parameter Request List

Length: 11; Value: 010F03062C2E2F1F21F92B

1 = Subnet Mask; 15 = Domain Name

3 = Router; 6 = Domain Name Server

44 = NetBIOS over TCP/IP Name Server

.....

request

Message type: **Boot Reply (2)**

Hardware type: Ethernet

Hardware address length: 6

Hops: 0

Transaction ID: 0x6b3a11b7

Seconds elapsed: 0

Bootp flags: 0x0000 (Unicast)

Client IP address: 192.168.1.101 (192.168.1.101)

Your (client) IP address: 0.0.0.0 (0.0.0.0)

Next server IP address: 192.168.1.1 (192.168.1.1)

Relay agent IP address: 0.0.0.0 (0.0.0.0)

Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)

Server host name not given

Boot file name not given

Magic cookie: (OK)

Option: (t=53,l=1) DHCP Message Type = DHCP ACK

Option: (t=54,l=4) Server Identifier = 192.168.1.1

Option: (t=1,l=4) Subnet Mask = 255.255.255.0

Option: (t=3,l=4) Router = 192.168.1.1

Option: (6) Domain Name Server

Length: 12; Value: 445747E2445749F244574092;

IP Address: 68.87.71.226;

IP Address: 68.87.73.242;

IP Address: 68.87.64.146

Option: (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."

reply

IP addresses: how to get one?

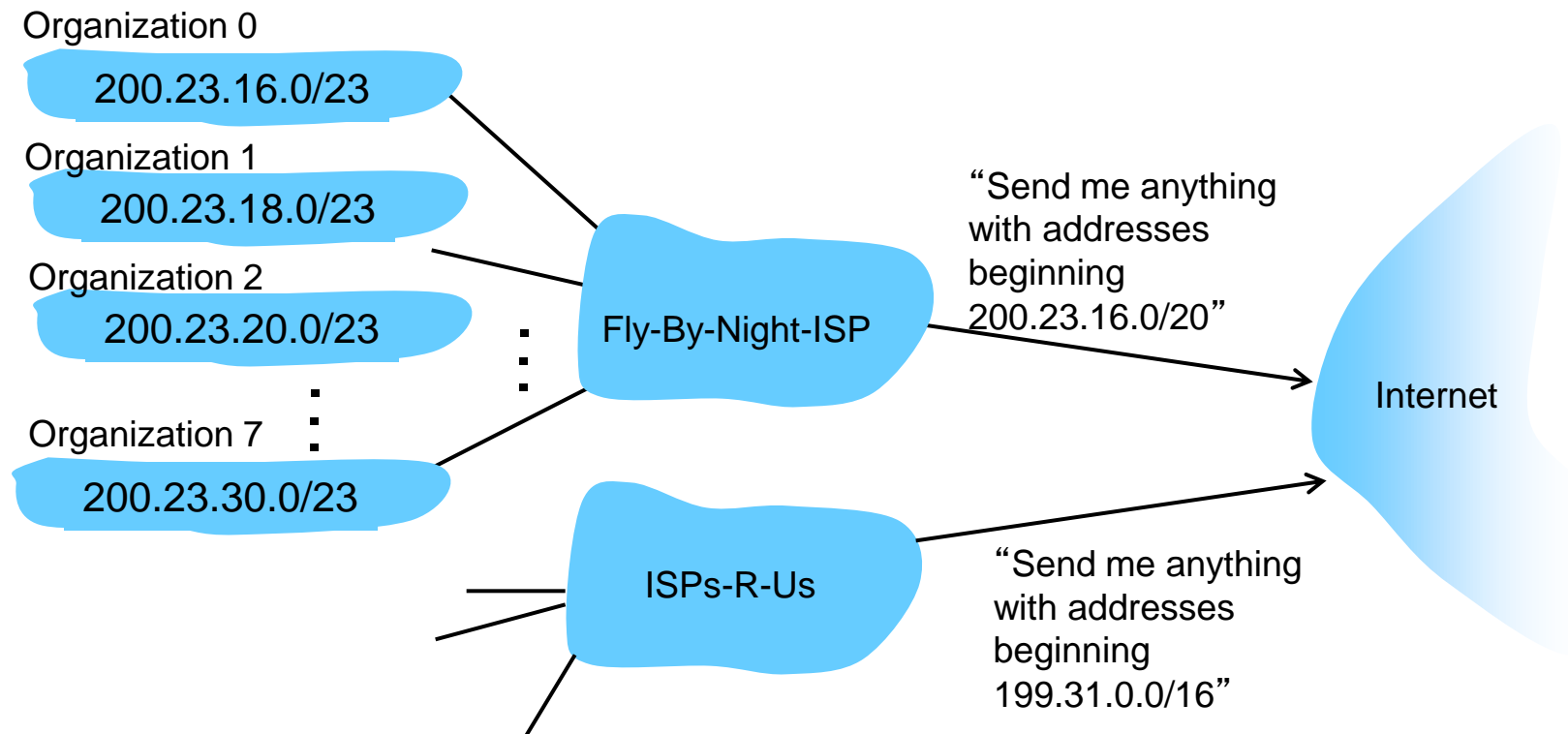
Q: how does *network* get subnet part of IP addr?

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

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

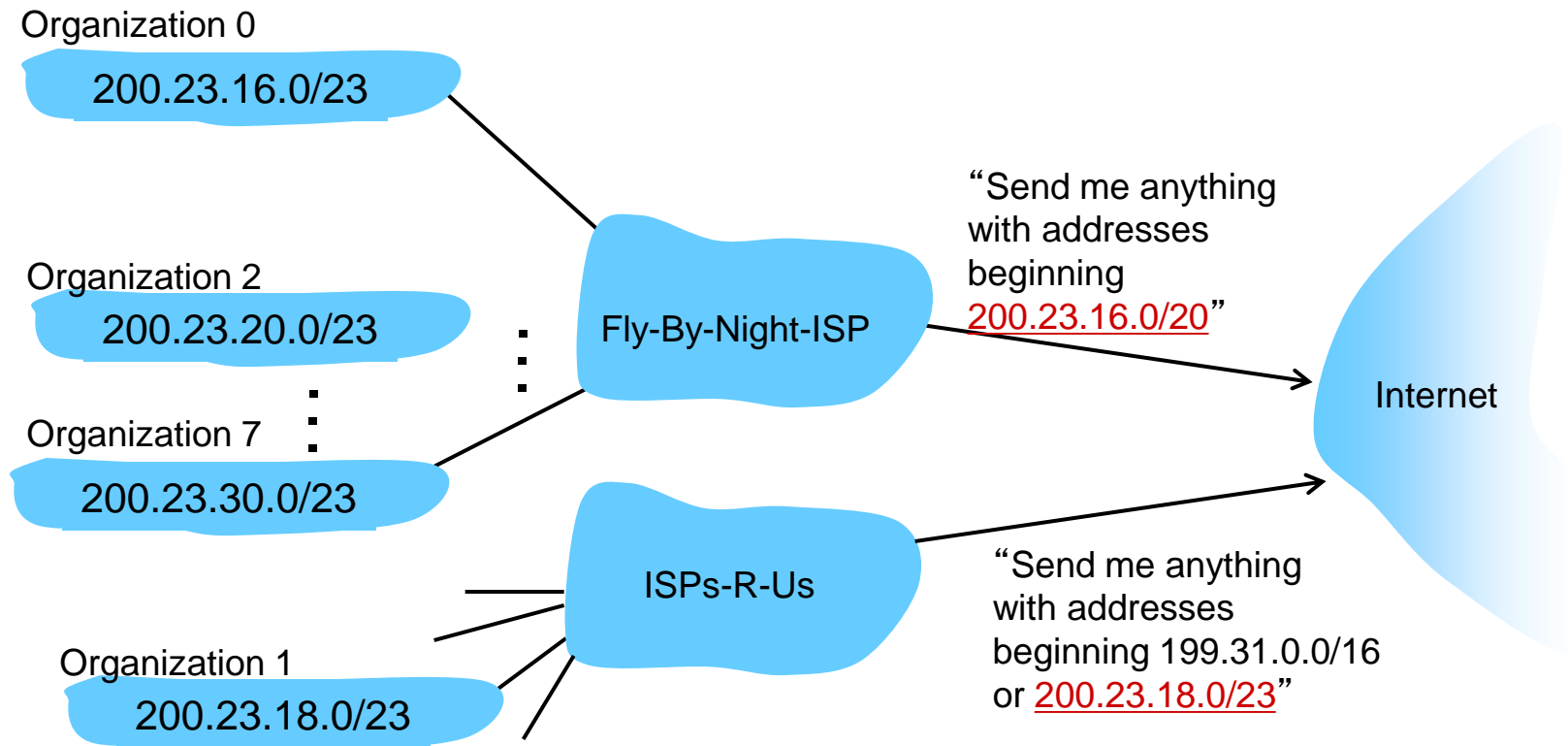
Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

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



IP addressing: the last word...

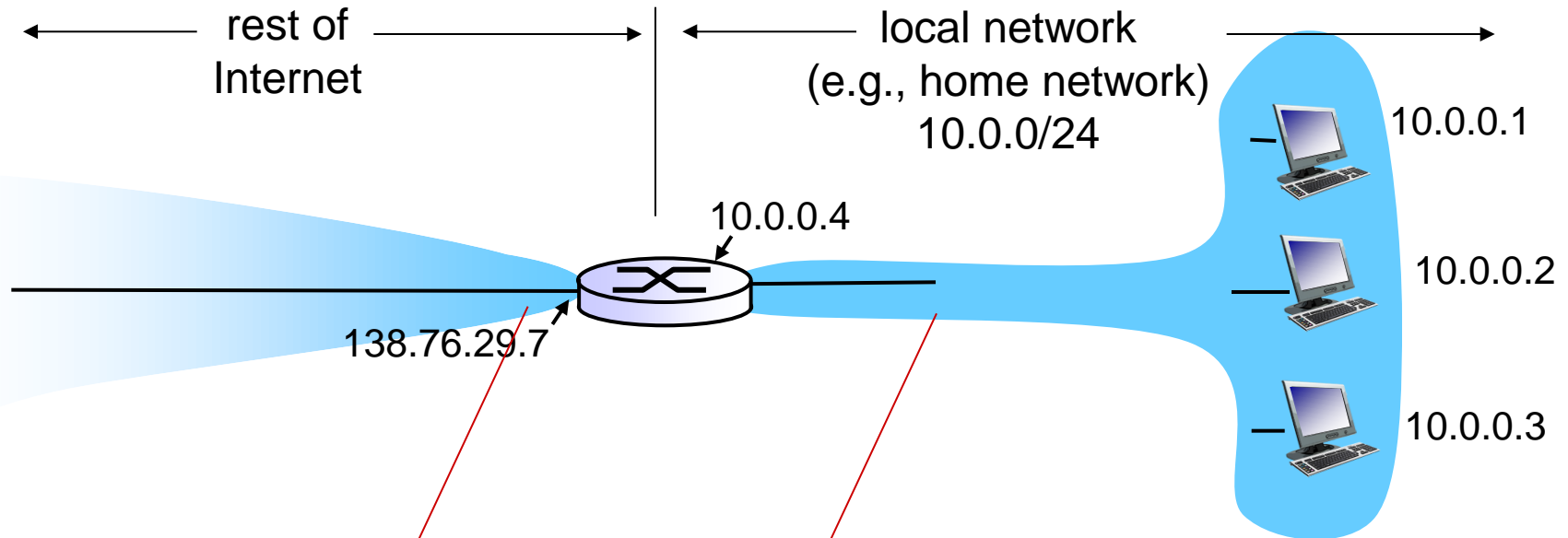
Q: ISP ได้รับบล็อกของที่อยู่ได้อย่างไร?

A: ICANN: Internet Corporation for Assigned

Names and Numbers <http://www.icann.org/>

- จัดสรร addresses
- manage DNS
- กำหนดชื่อโดเมน, ช่วยแก้ปัญหาความขัดแย้งที่เกิดขึ้น

NAT: network address translation



ดาต้าแกรมทั้งหมดออกจากเครือข่ายท้องถิ่น
จะมี Source IP address เบอร์เดียวกัน (ซึ่ง
เป็นของ NAT): 138.76.29.7,
ต่างที่ Source Port

ดาต้าแกรมจากต้นทาง หรือ ดาต้าแกรมไปยังปลายทาง
ในเครือข่ายนี้ จะมีที่อยู่ต้นทาง หรือ ปลายทางเป็น
10.0.0/24 (ตามปกติ)

NAT: network address translation

motivation: เครือข่ายท้องถิ่นใช้ IP address หมายเลขเดียว:

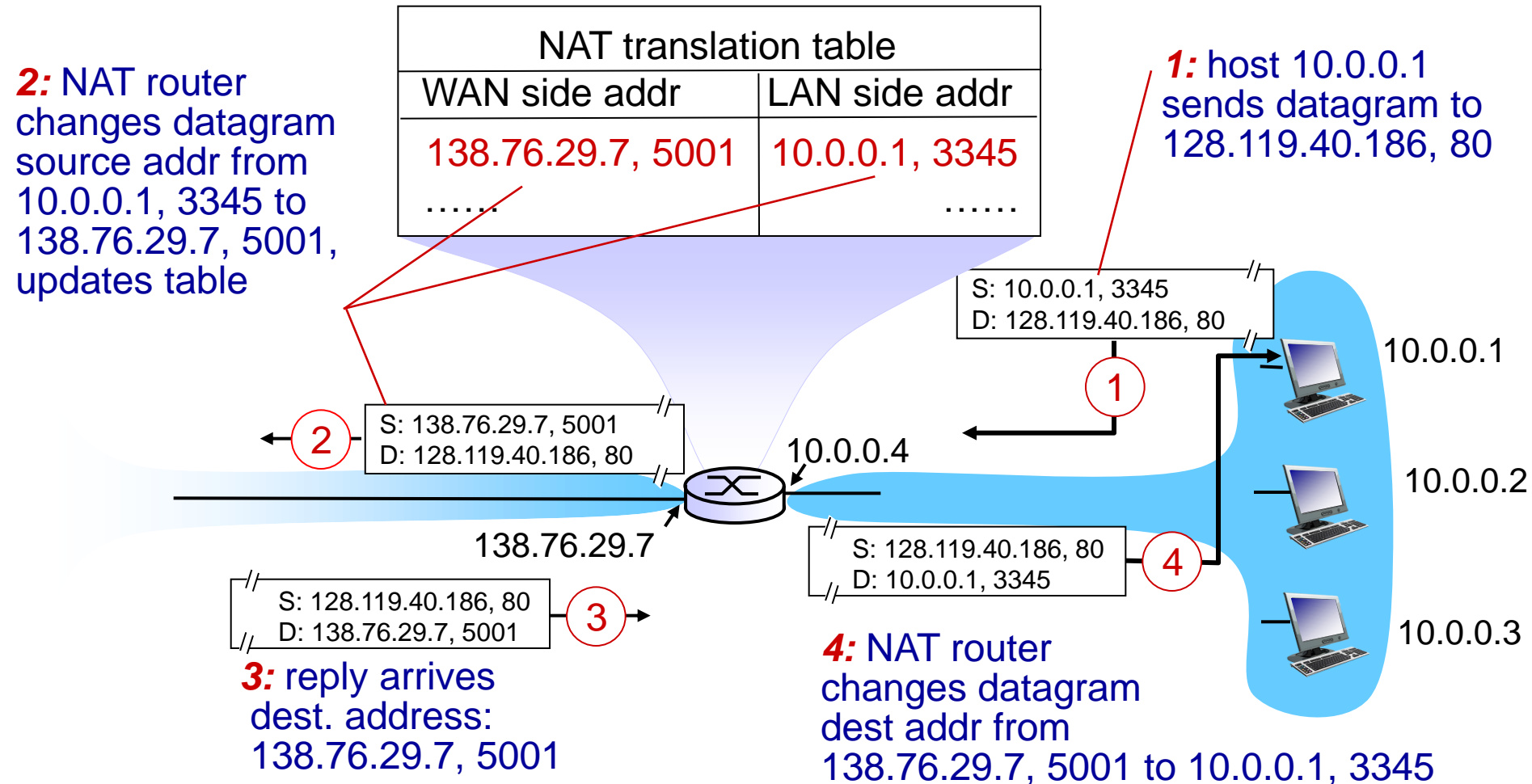
- โลกภายนอกจะเห็นทั้งเครือข่ายเป็น IP address เดียว (หมายเลข port ต่าง)
- ไม่จำเป็นต้องขอช่วงของ IP Address จาก ISP: แค่หนึ่ง IP address สำหรับทุกอุปกรณ์
- สามารถเปลี่ยนที่อยู่ของอุปกรณ์ในเครือข่ายท้องถิ่นโดยไม่ต้องแจ้งโลกภายนอก
- สามารถเปลี่ยน ISP โดยไม่ต้องเปลี่ยนที่อยู่ของอุปกรณ์ในเครือข่ายท้องถิ่น
- อุปกรณ์ในเครือข่ายท้องถิ่นไม่สามารถถูกระบุที่อยู่ได้อย่างชัดเจน, สามารถมองเห็นได้จากโลกภายนอก (ปลอดภัยเพิ่มขึ้น)

NAT: network address translation

การดำเนินการ: router ที่เป็น NAT ต้อง:

- *เดต้าแกรมขาออก: แทนที่* (source IP address, หมายเลขพอร์ต) ของทุกเดต้าแกรมขาออกให้เป็น (IP address ของ NAT, หมายเลขพอร์ตใหม่)
... clients/servers ที่อยู่ไกลจะตอบโดยใช้ (IP address ของ NAT, หมายเลขพอร์ตใหม่) เป็นที่อยู่ปลายทาง
- *จำ* ทุก ๆ คู่การแปลจาก (source IP address, หมายเลขพอร์ต) ไปเป็น (IP address ของ NAT, หมายเลขพอร์ตใหม่) ในตารางการแปลของ NAT
- *เดต้าแกรมขาเข้า: แทนที่* (IP address ของ NAT, หมายเลขพอร์ตใหม่) ใน field ปลายทางของทุก ๆ เดต้าแกรมด้วย (source IP address, หมายเลข port) ที่เป็นคู่ของมันที่ได้เก็บไว้ในตาราง NAT

NAT: network address translation

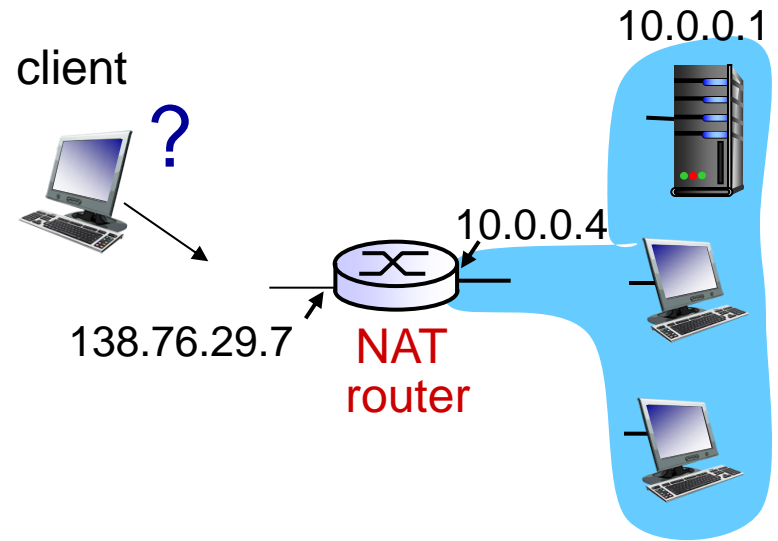


ควรใช้ NAT มั้ย ? (ข้าม)

- ❖ 16-bit port-number field:
 - 60,000 การเชื่อมต่อพร้อมกันกับที่อยู่ LAN ด้านเดียว!
- ❖ NAT is controversial:
 - เราเตอร์ควรจะประมวลผลได้ถึง 3 ชั้น
 - violates end-to-end argument
 - ความเป็นไปได้ NAT จะต้องนำเข้าบัญชีโดยนักออกแบบใน app เช่นโปรแกรม P2P
 - ปัญหาการขาดแคลนที่อยู่ควรจะเป็นแทนที่จะแก้ไขได้โดยการใช้ IPv6

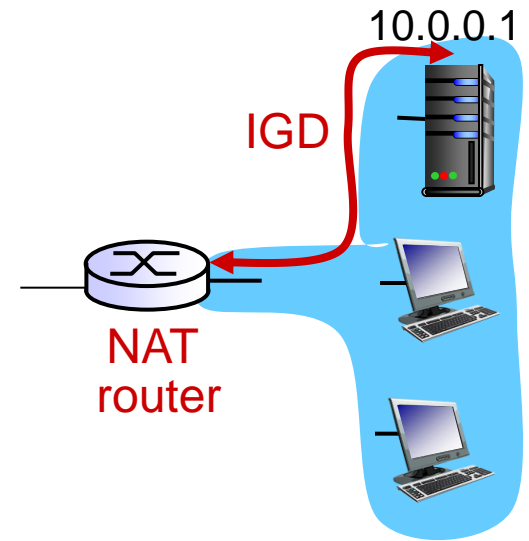
NAT traversal problem (ข้าม)

- ❖ client ต้องการที่จะเชื่อมต่อกับเซิร์ฟเวอร์ที่มี address 10.0.0.1
 - server address 10.0.0.1 local to LAN (client ไม่สามารถใช้เป็นปลายทาง address ได้)
 - ภายนอกเท่านั้นที่มอง NATed address: 138.76.29.7
- ❖ **ทางแก้1:** แบบคงที่กำหนดค่า NAT จะเข้ามาส่งต่อการร้องขอการเชื่อมต่อที่พอร์ตให้กับเซิร์ฟเวอร์
 - e.g., (138.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000



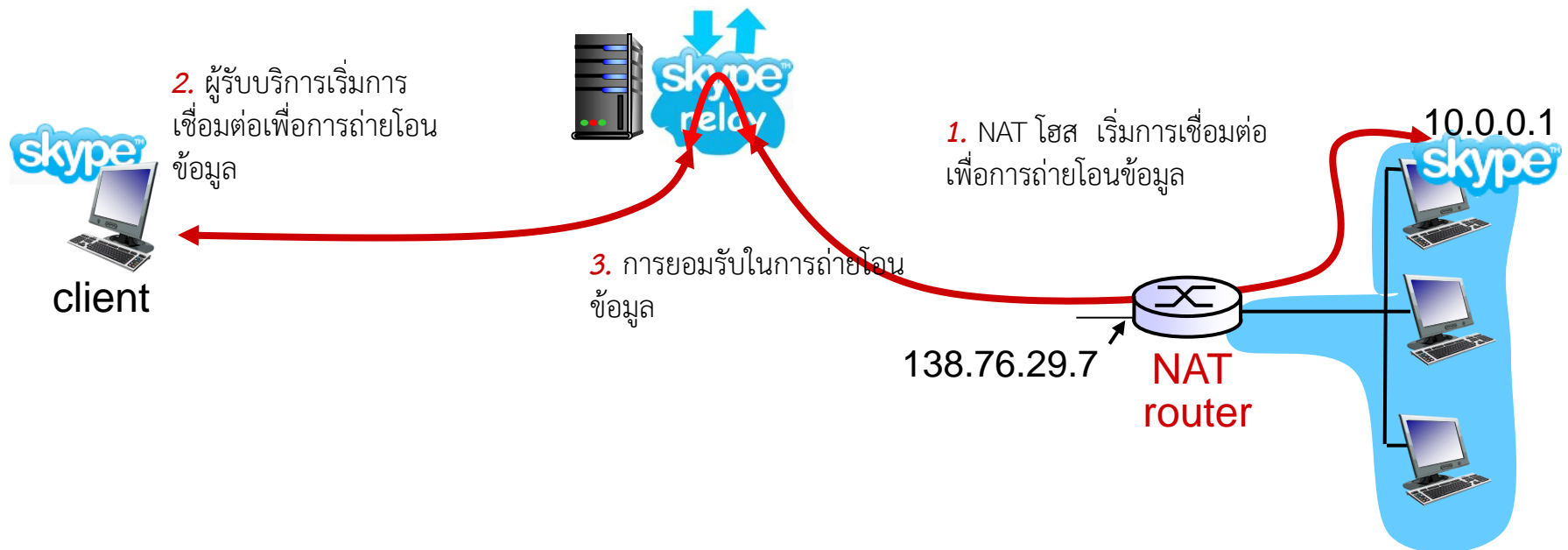
NAT traversal problem (ข้าม)

- ❖ *ทางเลือก 2:* ใช้ Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol.
Allows NATed เพื่อช่วยให้:
 - ❖ learn public IP address (138.76.29.7)
 - ❖ add/remove port mappings (with lease times)
- i.e., โดยอัตโนมัติการกำหนดค่า NAT port map configuration



NAT traversal problem (ข้าม)

- ❖ *solution 3*: การถ่ายโอนข้อมูล (ในการใช้โปรแกรม Skype)
 - ผู้รับบริการเริ่มการเชื่อมต่อเพื่อการถ่ายโอนข้อมูลด้วยวิธีการ NAT
 - ผู้รับบริการภายนอกทำการเชื่อมต่อที่จะถ่ายโอนข้อมูล
 - การเชื่อมต่อระหว่างกันแบบนี้จะทำให้เกิดเป็นสะพานในการถ่ายโอนข้อมูล



Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

ICMP: internet control message protocol (ข้าม)

- ❖ การใช้โฮสและระดับเครือข่ายสารสนเทศของการติดต่อสื่อสารใน routers

- การรายงานข้อผิดพลาด : ไม่สามารถเข้าถึง host, network, port, protocol
- การตอบสนองต่อ ความต้องการ/การถ่ายโอนข้อมูล (ด้วยการใช้คำสั่ง ping)

- ❖ network-layer “เหนือกว่า” IP:

- ข้อความที่ส่งไปใน IP datagrams ของ ICMP

- ❖ ICMP message: ใน IP datagram type ก่อให้เกิดข้อผิดพลาดของ รหัสคำสั่ง 8 ไบต์แรก

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

Traceroute and ICMP

- ❖ การส่งชุดต้นฉบับของส่วนปลายทางใน UDP

- ครั้งแรก กำหนดให้ TTL = 1
- ครั้งที่สอง กำหนดให้ TTL=2, และอื่นๆ
- หมายเลขของพอร์ตที่ไม่น่าจะเกิดขึ้น

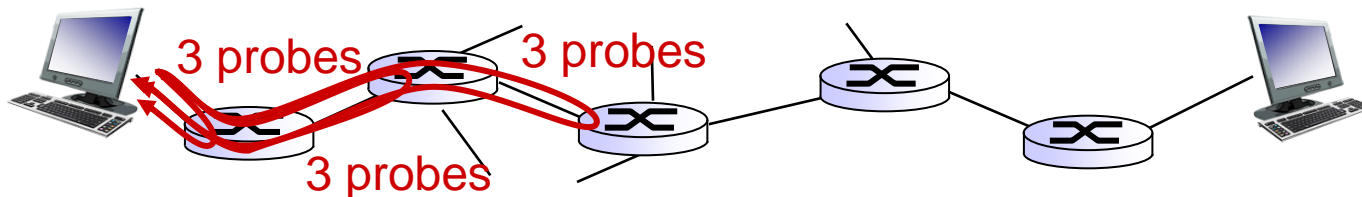
- ❖ เมื่อค่าของ datagrams ที่ n th กำหนดไปถึง n th router:

- Datagrams จะถูก router ตัดทิ้ง
- และทำการส่ง messages ต้นฉบับของ ICMP (type 11, code 0)
- ใน ICMP messages จะรวมถึง ชื่อของ router และหมายเลข IP

- ❖ เมื่อ messages ของ ICMP และ เรคคอร์ดของ RTTs มาถึง

บรรทัดฐานของการ stopping :

- ❖ เมื่อส่วนของ UDP มาถึงที่โฮสปลายทาง
- ❖ เมื่อ message ของ ICMP ย้อนกลับที่ปลายทาง “ไม่สามารถเข้าถึงพอร์ต” (type 3, code 3)
- ❖ source stops



IPv6: motivation

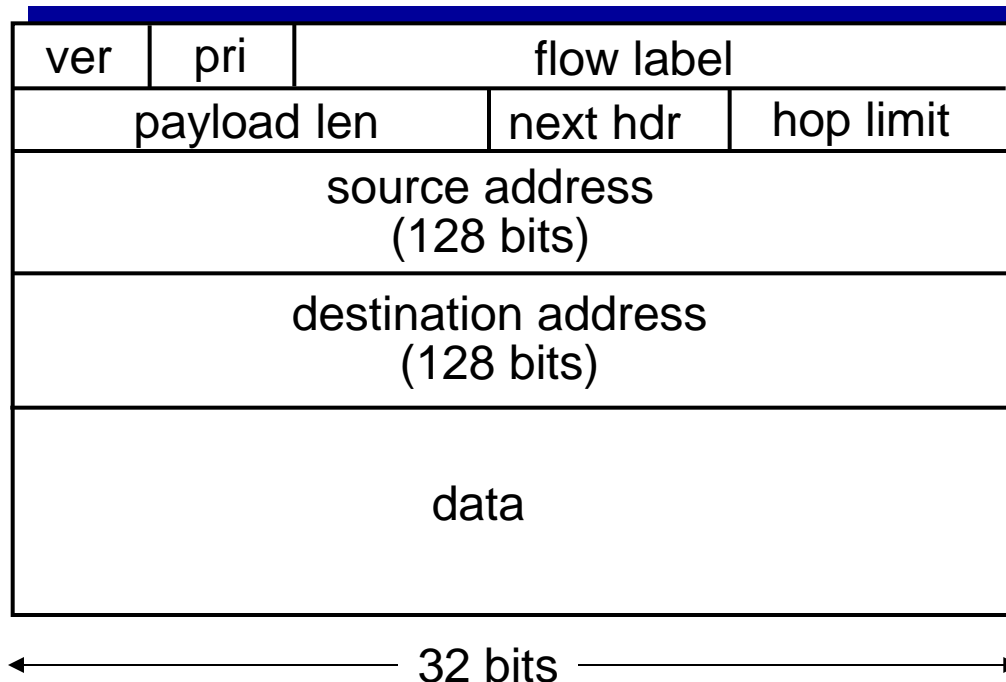
- ❖ *มูลเหตุจูงใจเริ่มแรก* : ความเป็นไปได้ที่จะเกิดสูญญากาศในเร็วๆ นี้ ของการจัดสรรหมายเลข IP แบบ 32-bit
- ❖ *มูลเหตุจูงใจเพิ่มเติม* :
 - รูปแบบที่ header จะช่วยให้การประมวลผลและการส่งต่อมีความรวดเร็วมากขึ้น
 - การเปลี่ยนแปลงที่ header ทำให้การทำ QoS สะดวกขึ้น

รูปแบบของ datagram IPv6 :

- กำหนดให้ header มีความยาว 40 byte
- ไม่ยินยอมให้มีการแตกกระจาย

IPv6 datagram format

- ❖ *ลำดับความสำคัญ* : ระบุความสำคัญของ datagrams ใน flow
- ❖ *ป้ายชื่อของ flow* : ระบุ datagrams ใน “flow” เดียวกัน
(แนวคิดของ “flow” ไม่ได้กำหนดเป็นนิยามไว้)
- ❖ *next header* : ระบุให้อยู่เหนือเลขเอร์โปรโตคอลของชั้น data

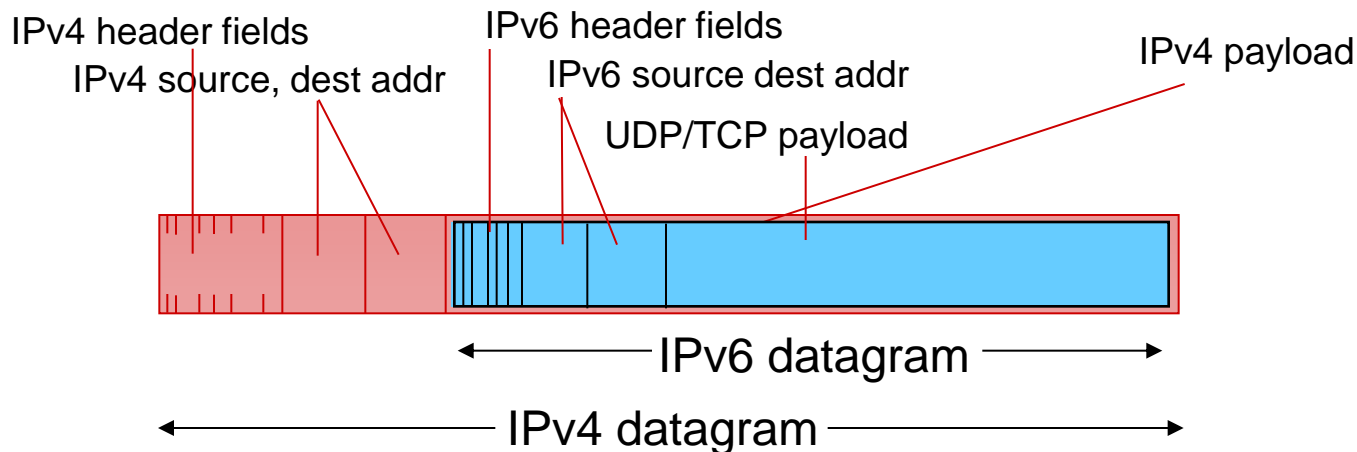


Other changes from IPv4

- ❖ *checksum*: เอา checksum ออก เพื่อให้เวลาในการประมวลผลของแต่ละ hop ลดลง
- ❖ *Options* : ยอมให้มี Options แต่ไม่ได้อยู่ใน Header จะนำไปไว้ในส่วนของ “Next Header” field
- ❖ *ICMPv6*: เวอร์ชันใหม่ของ ICMP
 - มีเพิ่มเติมในส่วนของชนิด message จะเป็นแบบ e.g. “Packet Too Big”
 - จะมีฟังก์ชันในการจัดการ การแพร่สัญญาณเฉพาะกลุ่ม

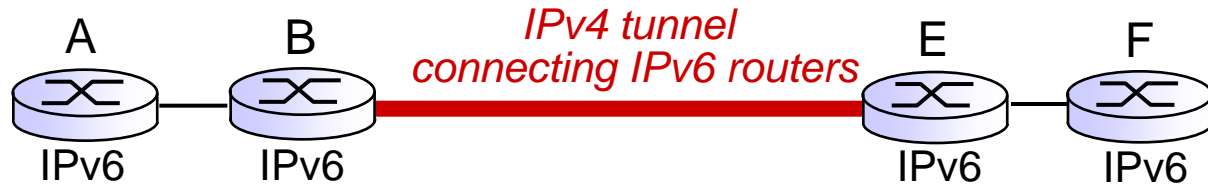
Transition from IPv4 to IPv6 (ข้าม)

- ❖ การเปลี่ยนผ่านไม่สามารถทำได้พร้อมกันทั้งหมด
 - ไม่มีตัวบ่งชี้ว่าจะสามารถเปลี่ยนผ่านให้กับทุกคนได้ในวันเดียว (no “flag days”)
 - ในการปฏิบัติทางเครือข่ายแสดงว่าจะต้องมีการผสมผสานกันระหว่าง IPv4 กับ IPv6
- ❖ *การใช้เทคนิคทางเครือข่ายแบบการขุดอุโมงค์ (tunneling)* : การสร้าง datagram ของ IPv6 ที่ถูกส่งไปใน datagram ของ IPv4

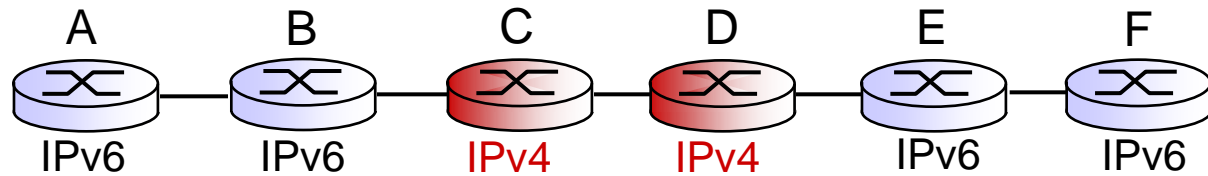


Tunneling (ขี้ม)

logical view:

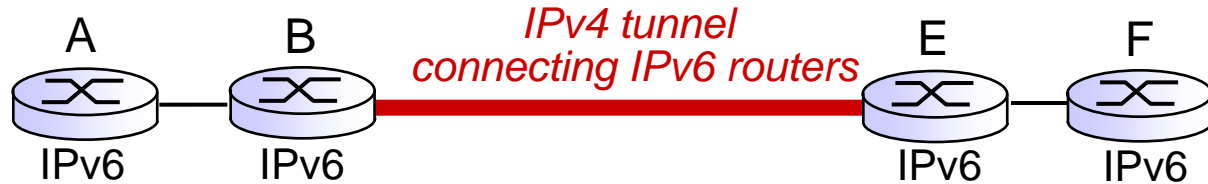


physical view:

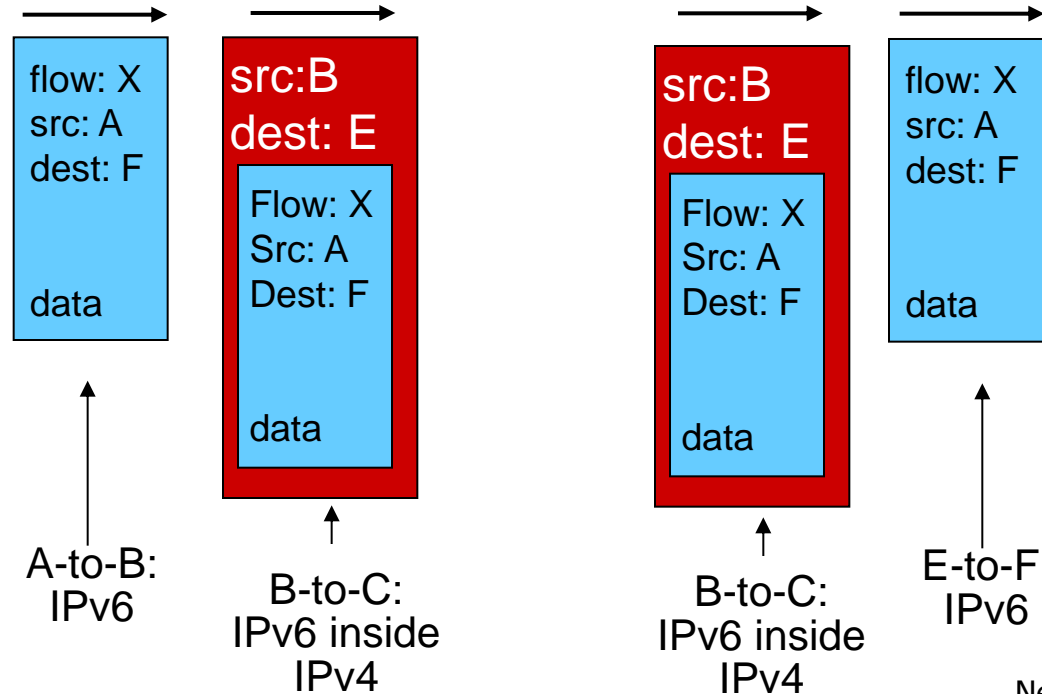
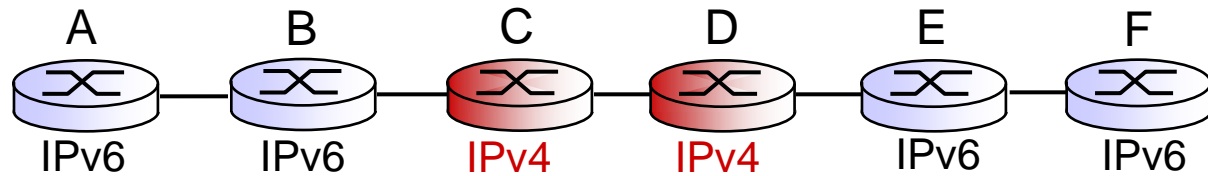


Tunneling (ขี้ม)

logical view:



physical view:



Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

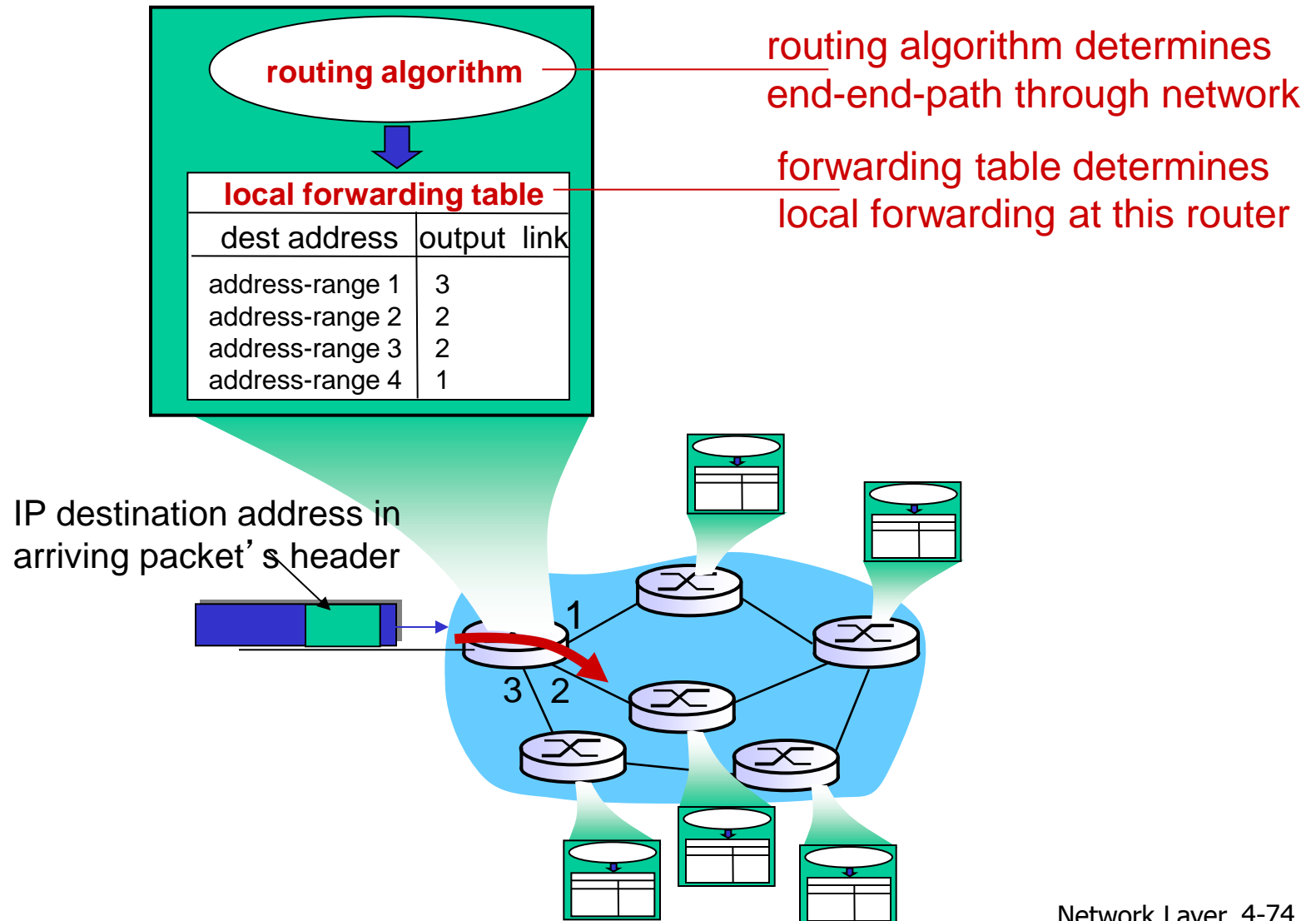
- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

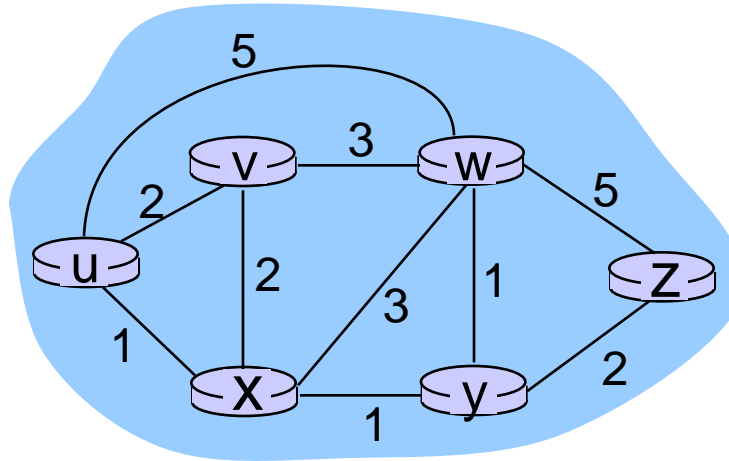
- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Interplay between routing, forwarding



Graph abstraction



graph: $G = (N, E)$

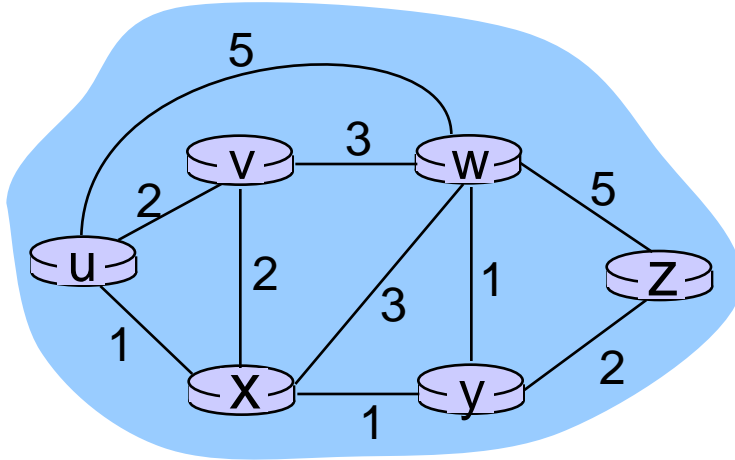
N = set of routers = $\{ u, v, w, x, y, z \}$

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

ในอีกด้าน : กราฟนั้นมีประโยชน์มากในการบรรยายเครือข่ายในรูปแบบต่างๆ เช่น

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

Graph abstraction: costs



$c(x, x') = \text{cost of link } (x, x')$

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

cost จะเป็น 1 เสมอ หรือ
ความแปรผันขึ้นอยู่กับ bandwidth
หรือ congestion

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

key question: อะไรคือเส้นทางที่สั้นที่สุดระหว่าง u และ z ?

routing algorithm: algorithm ที่หาเส้นทางที่สั้นที่สุด

Routing algorithm classification

Q: global or decentralized information?

global:

- ❖ Router ทุกตัวมีข้อมูลทั้งหมดของ topology (ข้อมูล link cost)
- ❖ “link state” algorithms

decentralized:

- ❖ router มีข้อมูลแค่เพื่อนบ้าน, (link costs ไปหาเพื่อนบ้าน)
- ❖ มีกระบวนการซ้ำๆ ในการแลกเปลี่ยนข้อมูลกับเพื่อนบ้าน
- ❖ “distance vector” algorithms

Q: static or dynamic?

static:

- ❖ เส้นทางเปลี่ยนช้า (ไม่ค่อยเปลี่ยน)

dynamic:

- ❖ เส้นทางเปลี่ยนเร็วมาก (บ่อย)
 - มีการอัปเดตเป็นช่วงๆ
 - เมื่อมีการตอบกลับว่า link cost เปลี่ยน

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

A Link-State Routing Algorithm

Dijkstra's algorithm

- ❖ ทุก node รู้ link cost ของทั้ง network topology
 - ทำงานผ่าน “link state broadcast”
 - ทุก node มีข้อมูลเหมือนกัน
- ❖ คำนวณเส้นทางที่สั้นที่สุดจาก node หนึ่ง (‘source’) ไป node อื่นทุก node
 - ส่ง *forwarding table* ไปให้ node นั้น
- ❖ iterative: after k iterations, know least cost path to k dest. 's

notation:

- ❖ $c(x,y)$: link cost from node x to y; $= \infty$ ถ้าไม่ได้เป็น node เพื่อนบ้าน
- ❖ $D(v)$: cost ปัจจุบันของเส้นทางจากต้นทางไปยังปลายทาง
- ❖ $p(v)$: predecessor node along path from source to v
- ❖ N' : set ของโหนดที่หาเส้นทางที่ cost น้อยที่สุดได้แล้ว

Dijkstra's Algorithm

1 **Initialization:**

2 $N' = \{u\}$

3 for all nodes v

4 if v adjacent to u

5 then $D(v) = c(u,v)$

6 else $D(v) = \infty$

7

8 **Loop**

9 find w not in N' such that $D(w)$ is a minimum

10 add w to N'

11 update $D(v)$ for all v adjacent to w and not in N' :

12 **$D(v) = \min(D(v), D(w) + c(w,v))$**

13 /* new cost to v is either old cost to v or known

14 shortest path cost to w plus cost from w to v */

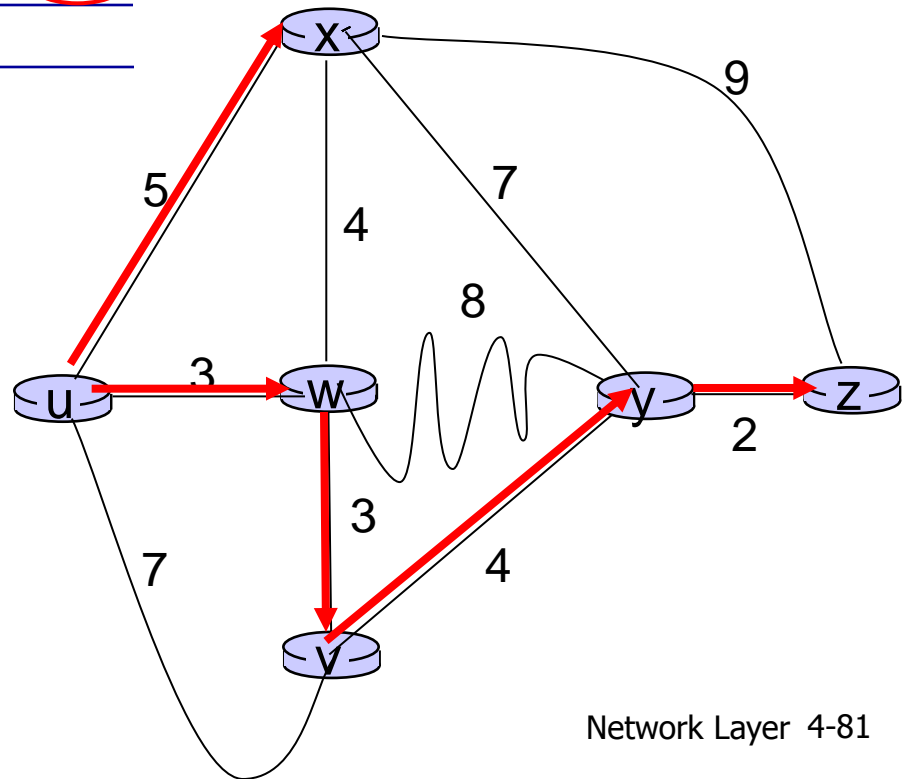
15 **until all nodes in N'**

Dijkstra's algorithm: example

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy				12,y	
5	uwxvyz					

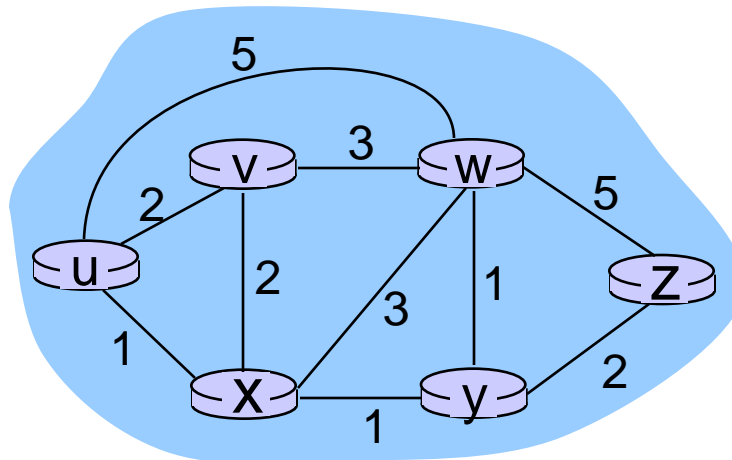
notes:

- ❖ สร้าง tree เส้นทางที่สั้นที่สุดโดยการ tracing predecessor nodes
- ❖ ความสัมพันธ์สามารถอยู่(อาจสามารถเปลี่ยนได้ตามความเหมาะสม)



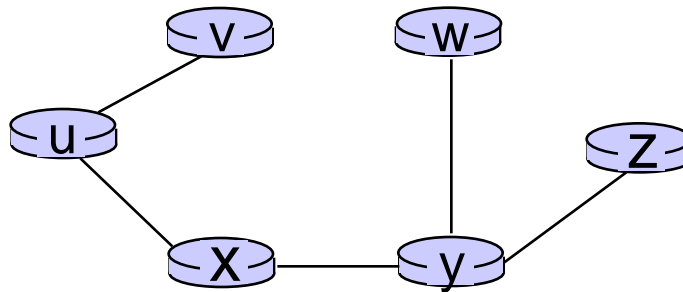
Dijkstra's algorithm: another example

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



Dijkstra's algorithm: example (2)

ผลลัพธ์ของ tree เส้นทางที่สั้นที่สุดจาก u:



ผลลัพธ์ของ forwarding table ใน u:

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

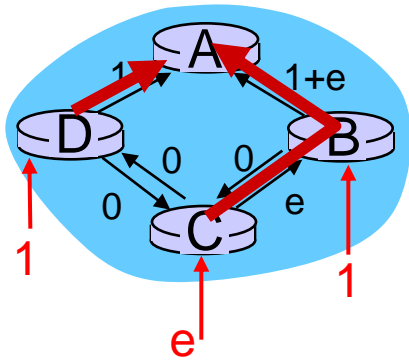
Dijkstra's algorithm, discussion (ข้าม)

algorithm complexity: n nodes

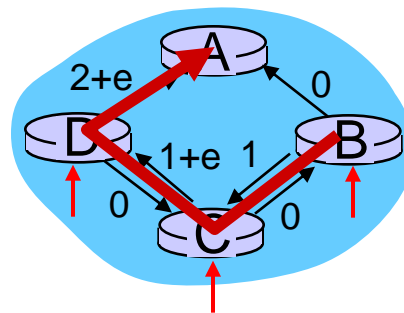
- ❖ แต่ละการทำซ้ำ : จำเป็นต้องเช็คทุก nodes, w, not in N
- ❖ $n(n+1)/2$ comparisons: $O(n^2)$
- ❖ ความเป็นไปได้ของประสิทธิภาพในการนำไปใช้ : $O(n \log n)$

oscillations possible:

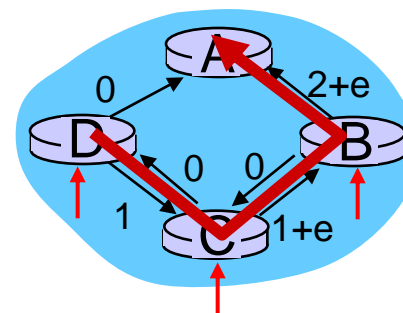
- ❖ e.g., support link cost equals amount of carried traffic:



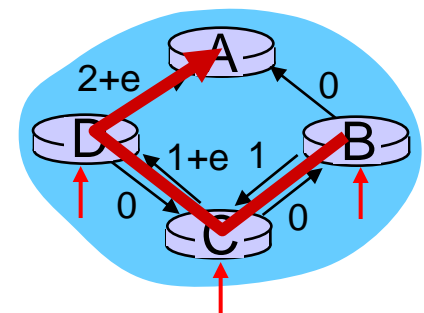
initially



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



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



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

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let

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

then

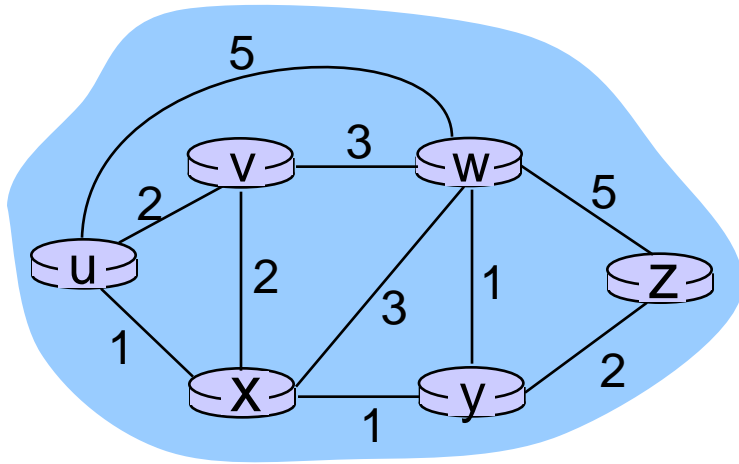
$$d_x(y) = \min_v \{ c(x,v) + d_v(y) \}$$

cost from neighbor v to destination y

cost to neighbor v

\min taken over all neighbors v of x

Bellman-Ford example



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

B-F equation says:

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

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

Distance vector algorithm

- ❖ $D_x(y)$ = ประมาณ least cost จาก x to y
 - x maintains distance vector $D_x = [D_x(y): y \in N]$
- ❖ node x:
 - รู้ cost ในการไปถึงแต่ละเพื่อนบ้าน : $c(x,v)$
 - การmaintainsระยะทางของเพื่อนบ้าน แต่ละเพื่อนบ้านจะต้องmaintains x,v
 $D_v = [D_v(y): y \in N]$

Distance vector algorithm

key idea:

- ❖ from time-to-time, แต่ละnodeส่งการประเมินระยะทางของตัวเองไปยังเพื่อนบ้าน
- ❖ เมื่อ x ได้รับการประเมินเส้นทางอันใหม่จากเพื่อนบ้าน, มันจะอัปเดตเส้นทางของมันโดยใช้ B-F equation:

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

- ❖ under minor, natural conditions, the estimate $D_x(y)$ converge to the actual least cost $d_x(y)$

Distance vector algorithm

iterative, asynchronous: แต่ละ

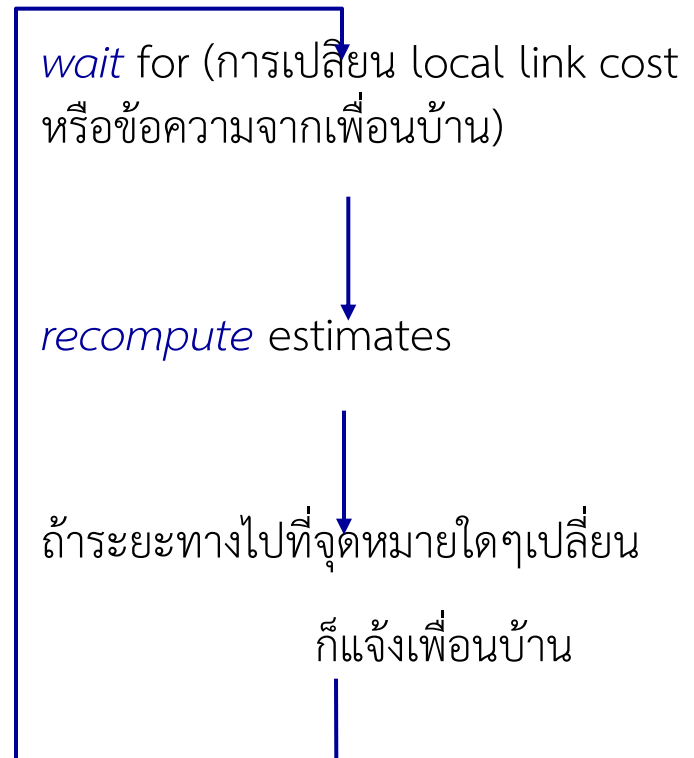
local iteration มีผลมาจาก:

- ❖ การเปลี่ยน local link cost
- ❖ มีข้อความอัปเดตระยะทางมาจากเพื่อนบ้าน

distributed:

- ❖ แต่ละ node จะมีการแจ้งเพื่อนบ้านเฉพาะเวลาที่ระยะทางเปลี่ยน
 - เพื่อนบ้านจะแจ้งเพื่อนบ้านเท่าที่จำเป็น

each node:



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

$$= \min\{2+0, 7+1\} = 2$$

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

$$= \min\{2+1, 7+0\} = 3$$

**node x
table**

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

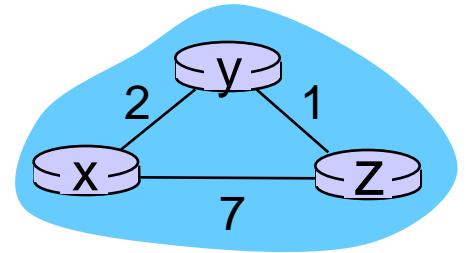
		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	7	1	0

**node y
table**

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

**node z
table**

		cost to		
		x	y	z
from	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**

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

**node y
table**

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

**node z
table**

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

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

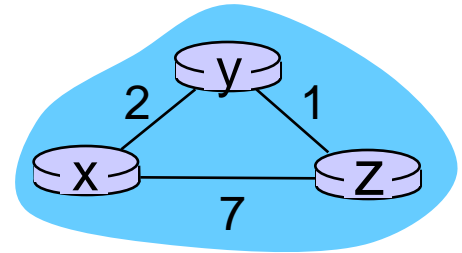
	cost to		
	x	y	z
from x	0	2	7
from y	2	0	1
from z	7	1	0

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

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

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

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

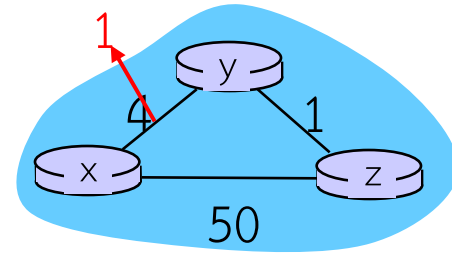


time →

Distance vector: ต้นทุนของlinkเปลี่ยน

ต้นทุนของ link เปลี่ยน:

- ❖ nodeจะตรวจพบว่าต้นทุนของlinkมีการเปลี่ยนยัง
- ❖ Updates ข้อมูลที่เกี่ยวกับ routing และคำนวณ distance vector
- ❖ ถ้า distance vector เปลี่ยนจะมีการบอกเพื่อนบ้าน



“**ข่าวดี**
จะกระจายเร็ว” t_0 : y ตรวจพบต้นทุนของ link เปลี่ยน Router จะมีการ updates distance vector ในตัวเองและจะมีการแจ้งไปยังเพื่อนบ้าน.

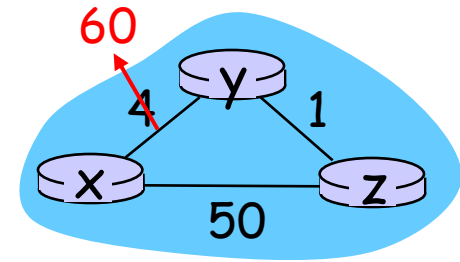
t_1 : z ได้รับการ update จาก from ของ y ซึ่งมีการ update ตารางแล้วมีการคำนวณต้นทุนหรือระยะทางที่น้อยที่สุดไปยัง x และจะมีการส่งข้อมูลไปยังเพื่อนบ้าน.

t_2 : y ได้รับข้อมูลจาก z 's ที่มีการ update แล้วก็มีการ updates ระยะทางในตารางของตัวเองถ้า y 's มีต้นทุนน้อยก็ไม่มีการเปลี่ยนแปลงดังนั้น y ไม่มีการส่งข้อความไปยัง z .

Distance vector: link cost changes

ต้นทุนของ link เปลี่ยน:

- ❖ nodeจะตรวจพบต้นทุนของ local link ที่มีการเปลี่ยน
- ❖ *bad news travels slow* - “นับไปจนถึงinfinity” ปัญหา!
- ❖ ก่อนจะทำวิธีการรักษาจะมีการทำงานซ้ำกันก่อน
จำนวน 44 รอบ: ถึงจะเห็นข้อความ



poisoned reverse:

- ❖ If Z routes through Y to get to X :
 - Z จะบอก Y โดยตัวของ (Z's) จะบอกระยะทางของ X ว่ามีระยะทางที่มาก(ดังนั้น Y จะไม่ไปเส้นทางของ X เพราะจะเลือกเส้นทางไป Z แทน)
- ❖ จะแก้ปัญหานี้สมบูรณ์ได้อย่างไรนับจากปัญหาที่ไม่มีวันสิ้นสุด?

Comparison of LS and DV algorithms

ความซับซ้อนของข้อความ

- ❖ **LS:** ด้วย n โหนด, E links, $O(nE)$ ทำการส่งข้อความ
- ❖ **DV:** มีการแลกเปลี่ยนระหว่างเพื่อนบ้านเท่านั้น
 - จะมีเวลาที่แตกต่างกันไปจนกว่าจะบรรจบกัน

ความเร็วของการบรรจบกัน

- ❖ **LS:** $O(n^2)$ ขั้นตอนที่ต้องการ $O(nE)$ ของข้อความ
 - อาจจะมีการแกว่ง
- ❖ **DV:** เวลาจะมีความแตกต่างกันไปจนกว่าจะบรรจบกัน
 - อาจจะมี routing loops
 - ปัญหาการนับถึง infinity

ความแข็งแรง: อะไรที่เกิดขึ้นถ้า router ทำงานผิดปกติ

LS:

- node สามารถโฆษณา *link* cost ไม่ถูกต้อง
- แต่ละ node จะคำนวณเพียงตารางของตนเอง

DV:

- DV node สามารถโฆษณาต้นทุนเส้นทางที่ไม่ถูกต้อง
- แต่ละ node 's จะมีตารางที่ใช้โดยคนอื่น
 - error ในการเผยแพร่ผ่านเครือข่าย

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Hierarchical routing

Routing ของเราได้เรียนรู้ thus far - idealization

- ❖ Routers เหมือนกันทั้งหมด
- ❖ network “เรียบ”

... ไม่เป็นจริงในทางปฏิบัติ

มาตราส่วน: ที่มี 600 ล้านสถานที่ :

- ❖ ไม่สามารถเก็บทุกปลายทางในตารางเส้นทาง!
- ❖ เส้นทางของตารางมีการแลกเปลี่ยนที่จะเชื่อมโยงท่วม!

อิสระในการบริหารจัดการ

- ❖ internet = เครือข่ายของเครือข่าย
- ❖ แต่ละเครือข่ายอาจจะต้องควบคุมเส้นทางในเครือข่ายของตัวเอง

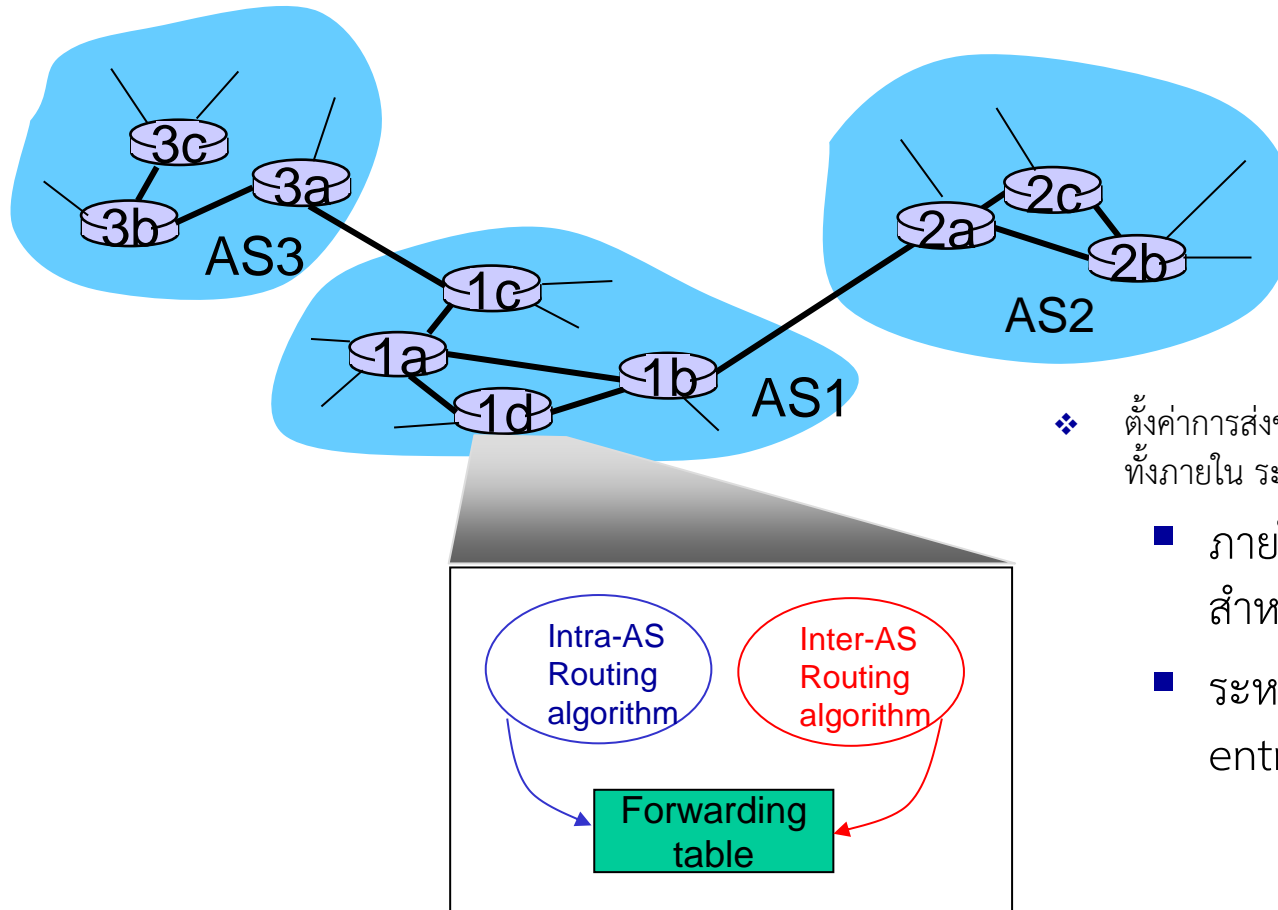
Hierarchical routing

- ❖ การรวบรวม routers ไว้ที่เดียวกันซึ่งเรียกกันว่า,
“autonomous systems” (AS)
- ❖ routers ใน AS เดียวกันมันก็จะใช้ routing protocol เดียวกัน
 - “intra-AS” routing protocol
 - routers ที่อยู่คนละ AS ก็จะมี routing protocol คนละตัว

gateway router:

- ❖ มี “edge” ส่วนปลายของ AS
- ❖ มี link ไปยัง router ในคนละ AS

Interconnected ASes



- ❖ ตั้งค่าการส่งของตารางโดย
ทั้งภายใน ระหว่าง AS routing algorithm
 - ภายในของ AS เป็นรายการชุด
สำหรับปลายทางภายใน
 - ระหว่าง AS & intra-AS sets
entries สำหรับ ปลายทางภายนอก

Inter-AS tasks

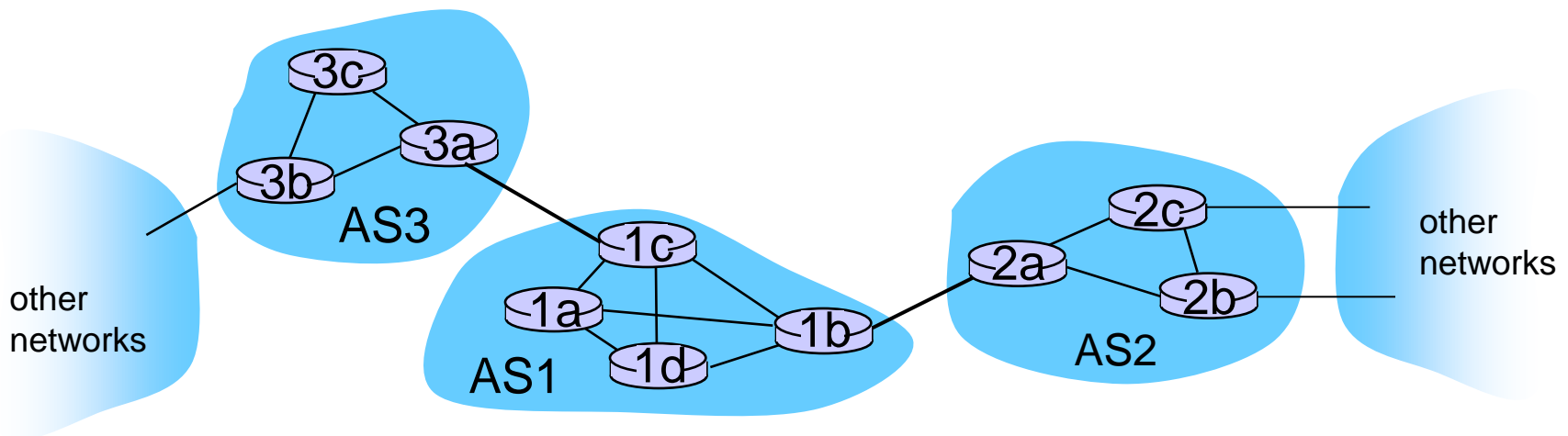
❖ สมมุติ router in AS1 ได้รับdatagram มุ่งข้างนอกของ AS1:

- router ควรส่ง packet ไปยัง gateway router, แต่เป็นที่ไหน?

AS1 ต้อง:

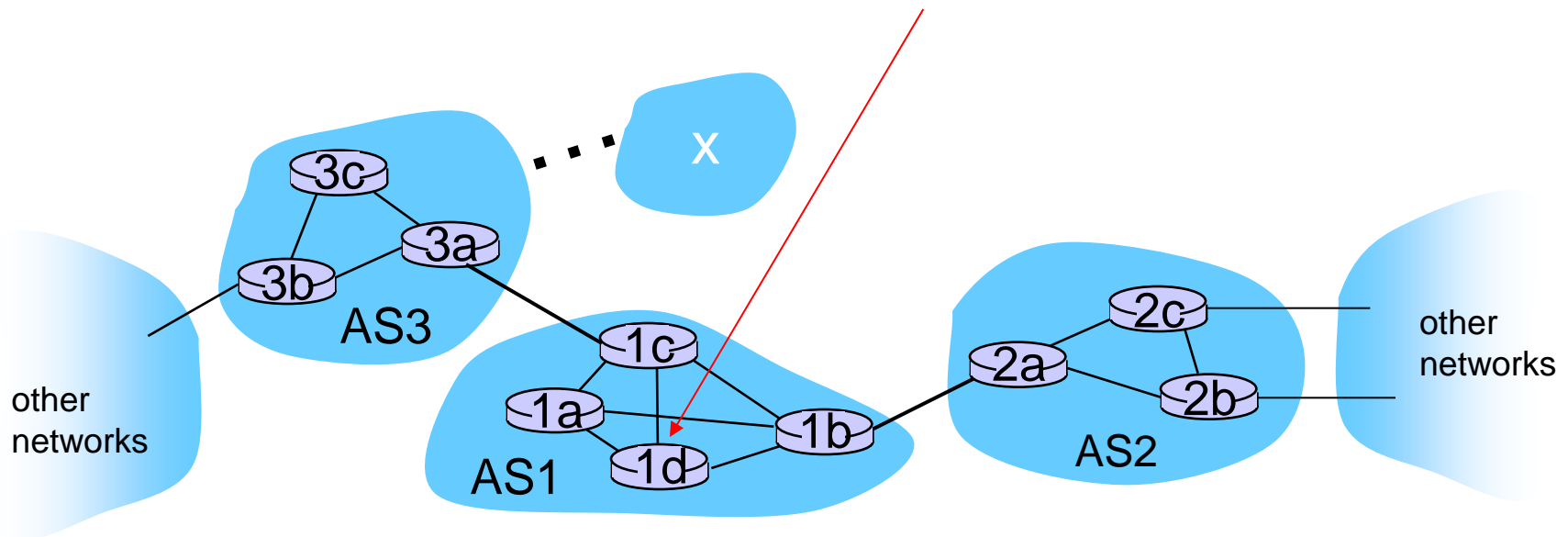
1. เรียนรู้ซึ่งปลายทางสามารถเข้าถึงได้ผ่าน AS2 ซึ่งผ่าน AS3
2. เผยแพร่ข้อมูลreachability เข้าไปทั้งหมด routers ใน AS1

งานของ inter-AS routing!



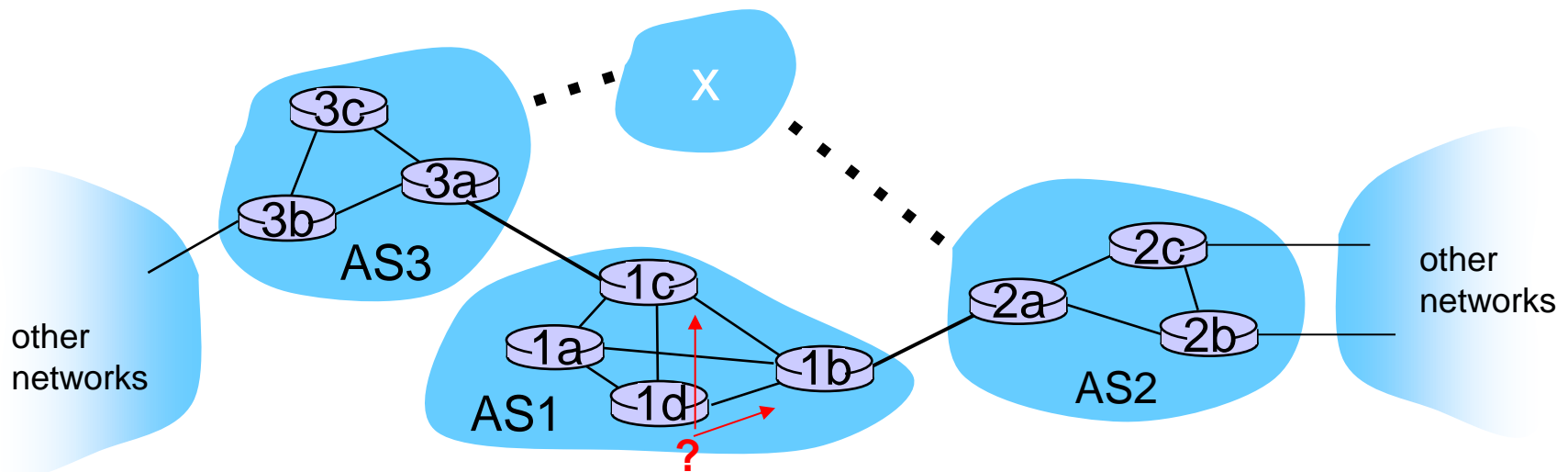
Example: setting forwarding table in router 1d

- ❖ สมมติว่า AS1 เรียนรู้ (ผ่านระหว่าง AS โปรโตคอล) ที่ x สามารถเข้าถึงได้ผ่านทางเครือข่ายย่อย AS3 (ประตู Lc) แต่ไม่ผ่าน AS2
- ❖ ระหว่างเป็นโปรโตคอลแพร่กระจายข้อมูล reachability ทุกเราเตอร์ภายใน
- ❖ router 1d determines from intra-AS routing info that its interface **l** is on the least cost path to 1c
 - installs forwarding table entry **(x,l)**



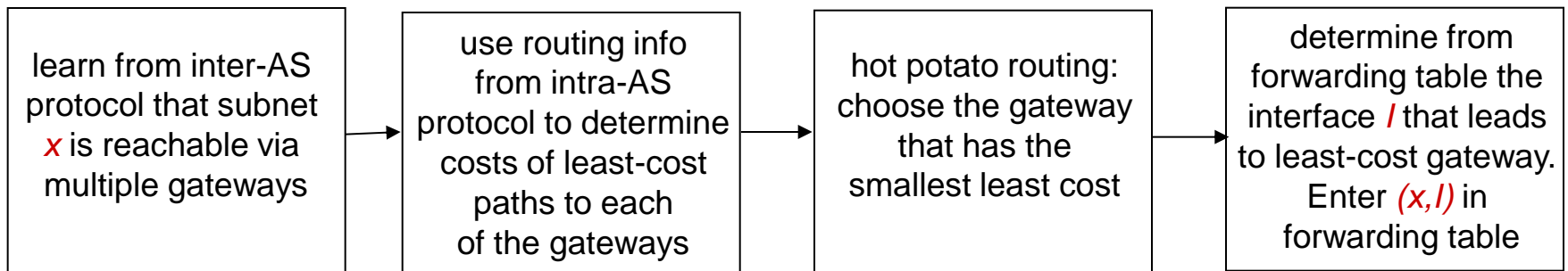
Example: choosing among multiple ASes

- ❖ now suppose AS1 learns from inter-AS protocol that subnet **x** is reachable from AS3 *and* from AS2.
- ❖ to configure forwarding table, router 1d must determine which gateway it should forward packets towards for dest **x**
 - this is also job of inter-AS routing protocol!



Example: choosing among multiple ASes

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



Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

- RIP
- OSPF
- BGP

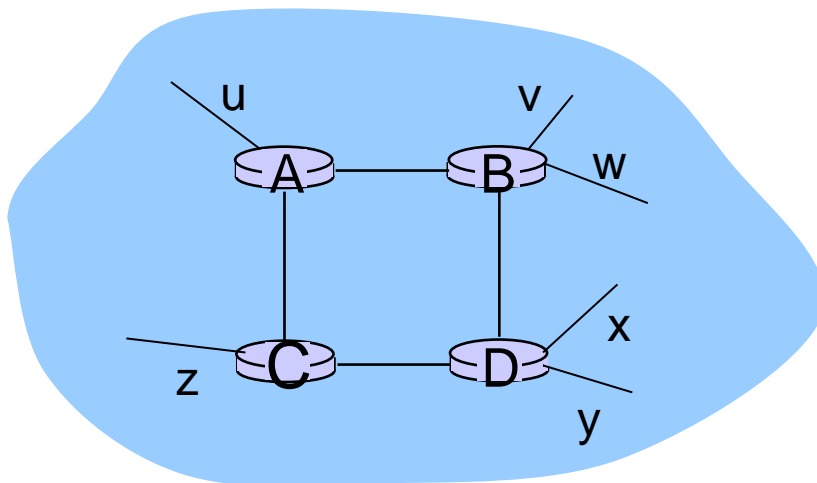
4.7 broadcast and multicast routing

Intra-AS Routing

- ❖ also known as *interior gateway protocols (IGP)*
- ❖ most common intra-AS routing protocols:
 - RIP: Routing Information Protocol
 - OSPF: Open Shortest Path First
 - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

RIP (Routing Information Protocol)

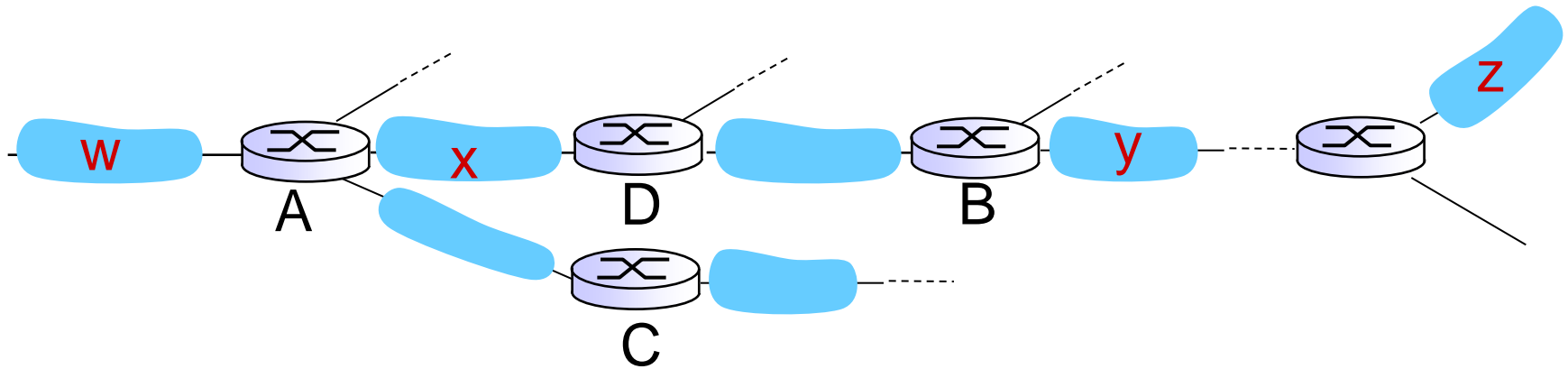
- ❖ included in BSD-UNIX distribution in 1982
- ❖ distance vector algorithm
 - distance metric: # hops (max = 15 hops), each link has cost 1
 - DVs exchanged with neighbors every 30 sec in response message (aka **advertisement**)
 - each advertisement: list of up to 25 destination **subnets** (in IP addressing sense)



from router A to destination **subnets**:

<u>subnet</u>	<u>hops</u>
u	1
v	2
w	2
x	3
y	3
z	2

RIP: example



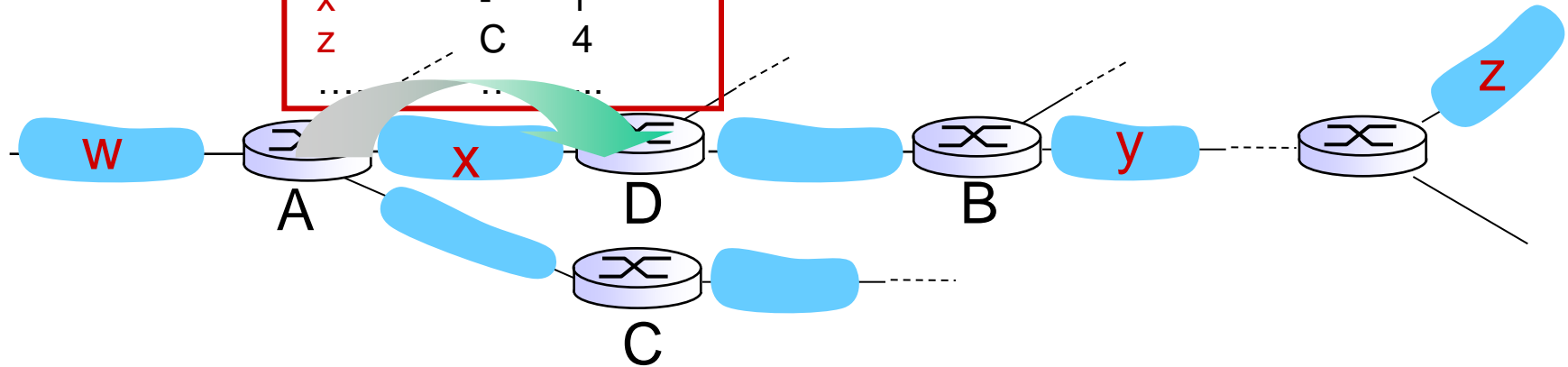
routing table in router D

destination subnet	next router	# hops to dest
W	A	2
y	B	2
z	B	7
x	--	1
....

RIP: example

A-to-D advertisement

dest	next	hops
W	-	1
X	-	1
Z	C	4
...



routing table in router D

destination subnet	next router	# hops to dest
W	A	2
y	B	2
Z	B → A	7 → 5
X	--	1
....

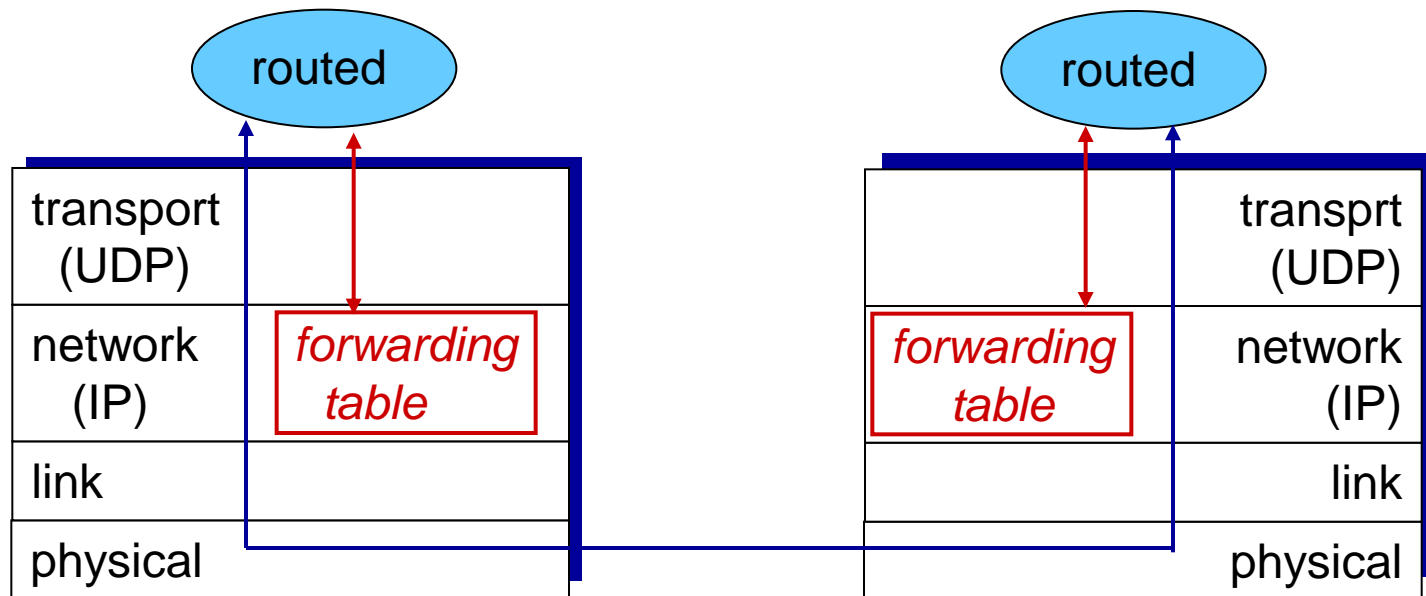
RIP: link failure, recovery

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

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

RIP table processing

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



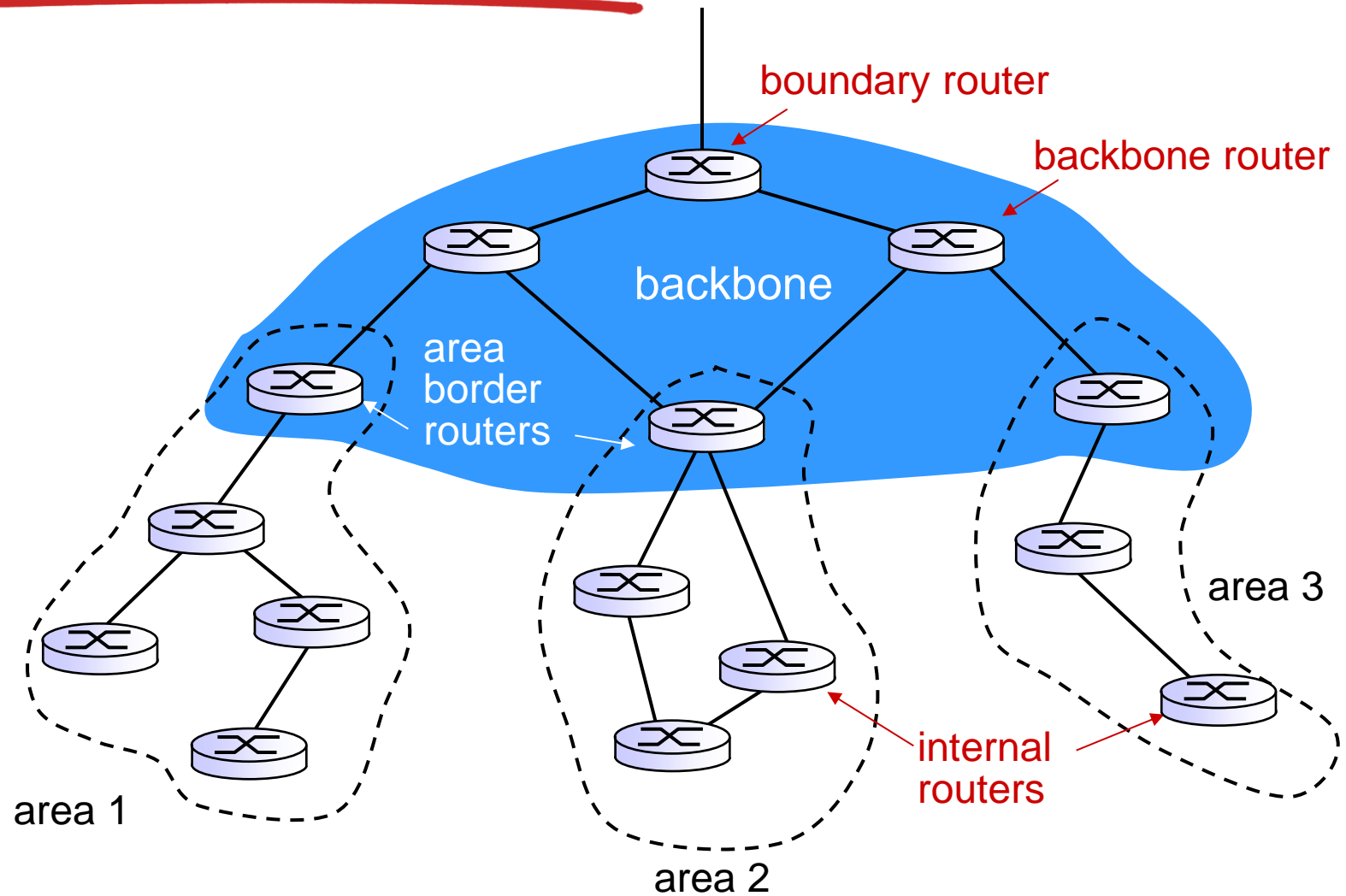
OSPF (Open Shortest Path First)

- ❖ เปิดให้ใช้งานได้อย่างสาธารณะ
- ❖ ใช้อัลกอริทึมแบบ Link State
 - ทำการส่ง LS packet แบบกระจายทั่วไปในเครือข่าย
 - เก็บข้อมูล topology map ของทุกๆ node
 - คำนวณเส้นทางโดยใช้ Dijkstra's algorithm
- ❖ OSPF ประกาศข้อมูล 1 entry ต่อเพื่อนบ้าน 1 จุด
- ❖ ทำการประกาศข้อมูลไปยัง AS ทั้งหมด
 - ส่งข้อมูล OSPF messages โดยตรงผ่านทาง IP (มากกว่าที่จะใช้ TCP หรือ UDP)
- ❖ *IS-IS routing* protocol: มีรูปแบบคล้ายคลึงกับ OSPF

OSPF “advanced” features (not in RIP)

- ❖ *security*: ข้อมูล OSPF messages ทั้งหมดต้องมีการ authenticated (เพื่อป้องกันการบุกรุกที่ไม่ประสงค์ดี)
- ❖ ยินยอมให้มีเส้นทางที่มี cost เท่ากันได้หลายเส้นทาง (RIP จะยินยอมให้มีได้เพียงเส้นทางเดียว)
- ❖ ในแต่ละ link , สามารถมี cost metrics ได้หลายชนิด ขึ้นอยู่กับประเภทของบริการที่ต้องการ (ตัวอย่างเช่น , link ผ่านดาวเทียม กำหนด cost “low” สำหรับบริการแบบ best effort; high สำหรับบริการแบบ real time)
- ❖ รองรับการทำงานร่วมกับ uni- และ *multicast*:
 - Multicast OSPF (MOSPF) ใช้งานข้อมูล topology รูปแบบเดียวกับ OSPF
- ❖ *hierarchical* OSPF นิยมใช้ใน domain ขนาดใหญ่.

Hierarchical OSPF



Hierarchical OSPF

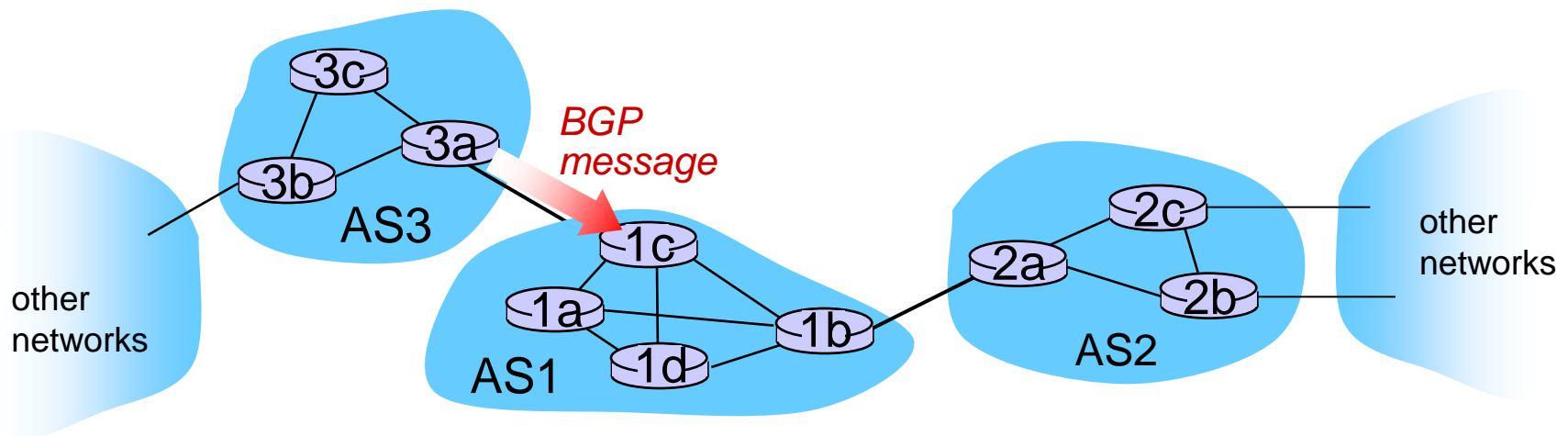
- ❖ *two-level hierarchy*: local area, backbone.
 - การประกาศข้อมูล link-state ทำอย่างมีขอบเขต
 - แต่ละ nodes จะมีข้อมูลรายละเอียด topology ในเขตของตนเอง; และจะมีเพียงข้อมูลเส้นทางที่ใกล้ที่สุดสำหรับพื้นที่นอกเขต.
- ❖ *area border routers*: “summarize” ระยะทางในเขตพื้นที่ของตนเอง, และประกาศข้อมูลไปยัง Area Border routers อื่นๆ.
- ❖ *backbone routers*: ใช้ OSPF routing ให้บริการจำกัดเฉพาะกลุ่ม backbone.
- ❖ *boundary routers*: เชื่อมต่อไปยัง AS อื่นๆ.

Internet inter-AS routing: BGP

- ❖ BGP (Border Gateway Protocol): เป็น *protocol* ที่ใช้งานจริงในกลุ่ม inter-domain routing protocol ในปัจจุบัน
 - “เชื่อมโยง Internet เข้าด้วยกัน”
- ❖ BGP ให้บริการแต่ละ AS ดังนี้:
 - eBGP: รับข้อมูลการเข้าถึง subnet จาก AS เพื่อนบ้านต่างๆ.
 - iBGP: ส่งข้อมูลการเข้าถึงไปยัง AS-internal routers ทั้งหมด.
 - กำหนดเส้นทางที่ดีสำหรับ network อื่นๆ โดยอ้างอิงจากข้อมูลการเข้าถึงและนโยบาย.
- ❖ ยินยอมให้ subnet ประกาศข้อมูลแสดงถึงการมีอยู่ของตนเองไปทั่วทั้งเครือข่าย Internet: “*I am here*”

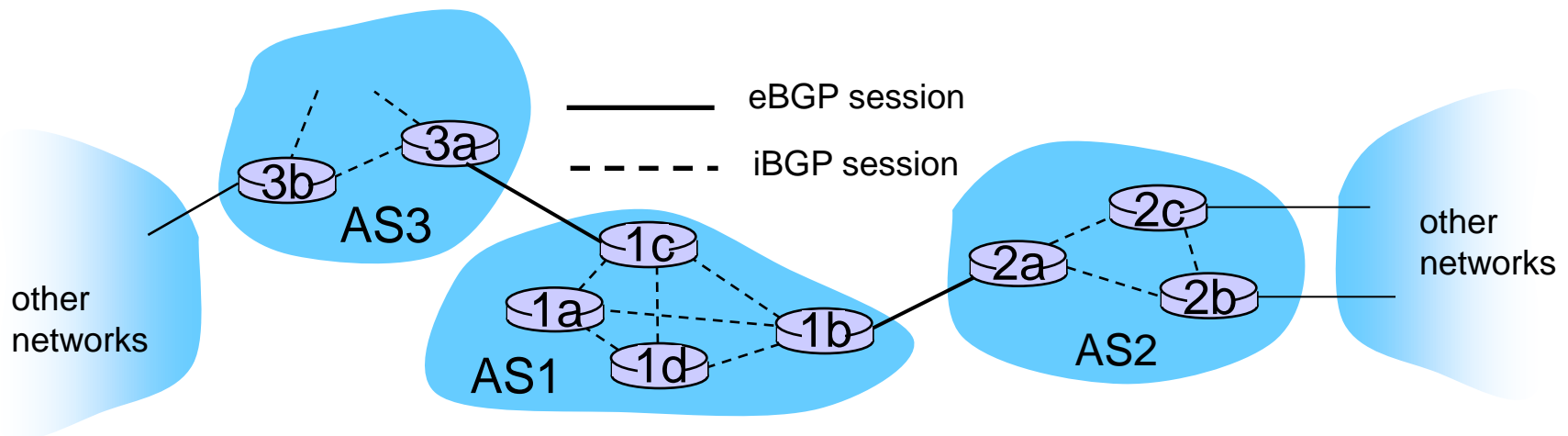
BGP basics

- ❖ **BGP session:** BGP routers 2 ตัว (“peers”) เพื่อแลกเปลี่ยนข้อมูล BGP messages:
 - ประกาศข้อมูล *เส้นทาง* ไปยัง Network ปลายทางที่มี Prefix แตกต่างกัน (“path vector” protocol)
 - ทำการแลกเปลี่ยนข้อมูลผ่านทาง semi-permanent TCP connections
- ❖ เมื่อ AS3 ประกาศข้อมูล prefix ไปยัง AS1:
 - AS3 *ให้สัญญาว่า* จะส่งต่อ Datagram ไปยัง prefix นั้น
 - AS3 สามารถรวบรวมข้อมูล Prefix หลายๆ Prefix เข้าด้วยกันในการประกาศข้อมูลเดียว



BGP basics: distributing path information

- ❖ การใช้งาน eBGP session ระหว่าง 3a และ 1c, AS3 ส่งข้อมูล prefix reachability ไปยัง AS1.
 - 1c จะสามารถใช้ iBGP เพื่อส่งข้อมูล prefix ใหม่ไปยัง routers ทุกตัวใน AS1
 - 1b จะสามารถประกาศข้อมูลการเข้าถึง อีกครั้งไปยัง AS2 ผ่านทาง 1b-ไปยัง-2a eBGP session
- ❖ เมื่อ router รับรู้ prefix ใหม่, จะทำการสร้าง entry สำหรับ prefix ใน forwarding table ของตนเอง.



Path attributes and BGP routes

- ❖ advertised prefix includes BGP attributes
 - prefix + attributes = “route”
- ❖ two important attributes:
 - **AS-PATH**: contains ASs through which prefix advertisement has passed: e.g., AS 67, AS 17
 - **NEXT-HOP**: indicates specific internal-AS router to next-hop AS. (may be multiple links from current AS to next-hop-AS)
- ❖ gateway router receiving route advertisement uses **import policy** to accept/decline
 - e.g., never route through AS x
 - *policy-based* routing

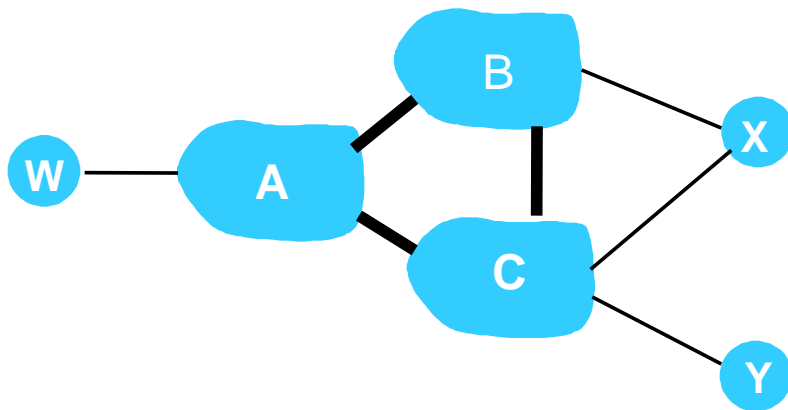
BGP route selection

- ❖ router จะคำนวณเส้นทางจากต้นทางไปยังปลายทาง ซึ่งจะคำนวณเส้นทางโดยดูจากข้อมูลต่อไปนี้ :
 1. local preference value attribute: policy decision
 2. shortest AS-PATH
 3. closest NEXT-HOP router: hot potato routing
 4. additional criteria


BGP messages

- ❖ ข้อความ BGP ถูกแลกเปลี่ยนระหว่างเพียร์ บนการเชื่อมต่อแบบ TCP
- ❖ ข้อความ BGP :
 - **OPEN:** เปิดการเชื่อมต่อ TCP ไปยังเพียร์และตรวจสอบผู้ส่ง
 - **UPDATE:** แจ้งช่องทางใหม่หรือยกเลิกอันเก่า
 - **KEEPALIVE:** รักษาการเชื่อมต่อให้คงอยู่เมื่อขาด UPDATES รวมถึงการร้องขอ ACKs OPEN
 - **NOTIFICATION:** รายงานข้อผิดพลาดในข้อความที่ผ่านมารวมถึงการใช้เพื่อปิดการเชื่อมต่อ

BGP routing policy

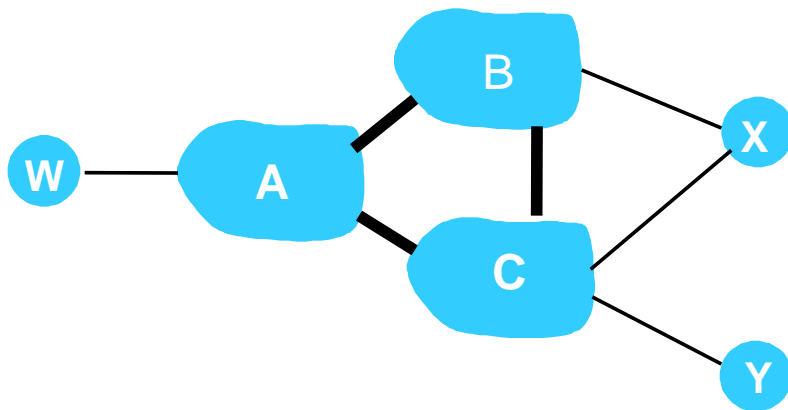




สัญลักษณ์ :  เครือข่ายของผู้ให้บริการ

 เครือข่ายของลูกค้า

- ❖ A,B,C คือ เครือข่ายของผู้ให้บริการ
- ❖ X,W,Y คือลูกค้าของผู้ให้บริการแต่ละราย
- ❖ X คือ *dual-homed*: เชื่อมต่อกับสองเครือข่าย
 - X ไม่ต้องการให้ B เชื่อมต่อกับ C โดยผ่าน X
 - .. ดังนั้น X จึงไม่บอก B ว่าเชื่อมต่อกับ C ได้

BGP routing policy (2)



สัญลักษณ์ :  เครือข่ายของผู้ให้บริการ
 เครือข่ายของลูกค้า

- ❖ A ประกาศการเส้นทาง Aw ไปยัง B
- ❖ B ประกาศการเส้นทาง BAw ไปยัง X
- ❖ B ควรจะประกาศเส้นทาง BAw ไปยัง C หรือไม่?
 - ไม่มีทาง! เพราะ B ไม่ได้รายได้จากการเชื่อมต่อ CBAw เนื่องจาก w และ C ไม่ใช่ลูกค้าของ B
 - B ต้องการบังคับให้ C เชื่อมต่อไปยัง w ผ่าน A
 - B ต้องการให้บริการเฉพาะลูกค้าตัวเอง

Why different Intra-, Inter-AS routing ?

policy:

- ❖ inter-AS: แอดมินต้องการควบคุมเส้นทางและมีหลายคนจึงต้องมีนโยบายในการตัดสินใจ
- ❖ intra-AS: มีแอดมินคนเดียวจึงไม่มีความต้องการนโยบายในการตัดสินใจ

scale:

- ❖ การจัดเส้นทางแบบเป็นลำดับชั้นประหยัดเนื้อที่ในการเก็บข้อมูล จึงทำให้ลดปริมาณข้อมูลอัปเดต

performance:

- ❖ intra-AS: มีประสิทธิภาพ
- ❖ inter-AS: นโยบายอาจส่งผลต่อประสิทธิภาพ

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

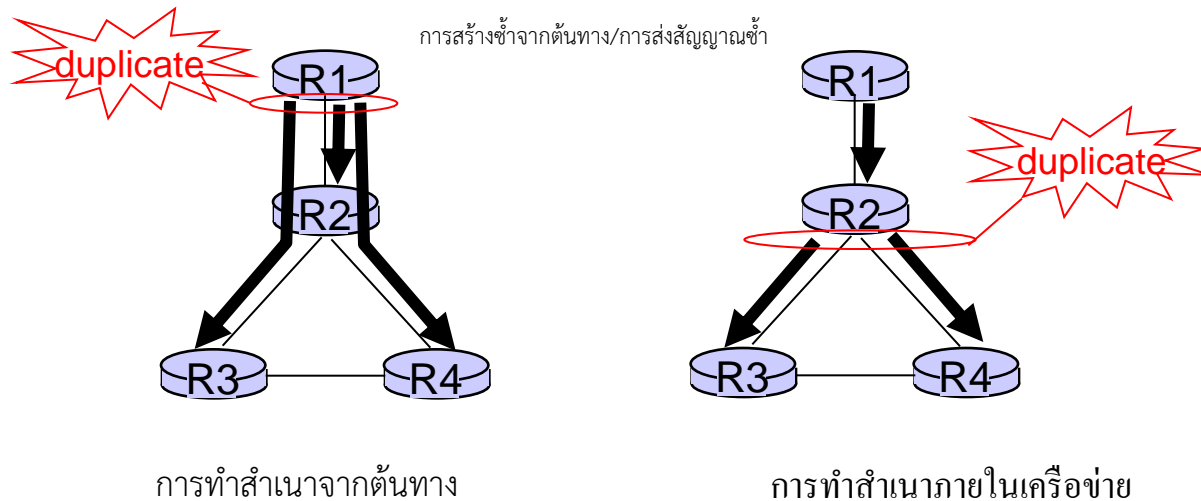
4.6 routing in the Internet

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Broadcast routing

- ❖ ส่งแพ็คเกจจากต้นทางไปยังทุกๆ โหนดอื่น
- ❖ การทำสำเนาจากต้นทางจะไม่มีประสิทธิภาพ:



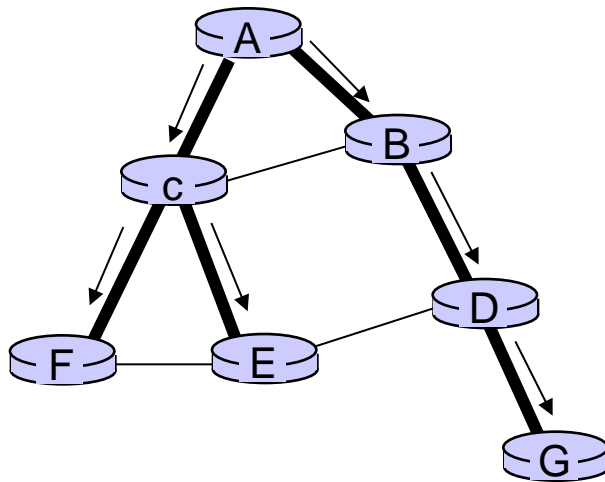
- ❖ การทำให้สำเนาจากต้นทาง: ต้นทางจะกำหนดที่อยู่ผู้รับปลายทางได้อย่างไร ?

In-network duplication

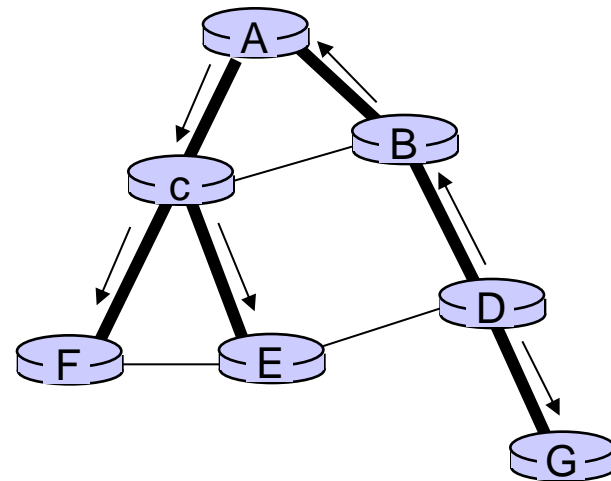
- ❖ *flooding*: เมื่อโหนดได้รับการกระจายแพ็คเก็ต, ก็จะส่งสำเนาไปยังจุดที่เชื่อมต่ออื่นๆกันด้วย
 - ปัญหาที่พบ: รอบของการส่งและการกระจายการส่งอย่างหนัก
- ❖ *controlled flooding*: โหนดแค่กระจายแพ็คเก็ตถ้ามันยังไม่ได้กระจายแพ็คเก็ตนั้นมาก่อน
 - โหนดคอยตามจับไอดีของแพ็คเก็ตที่กระจายมาแล้วเท่านั้น
 - หรือ reverse path forwarding (RPF): ส่งต่อแพ็คเก็ตเฉพาะในเส้นทางที่สั้นที่สุดระหว่างโหนดกับต้นทาง
- ❖ *spanning tree*:
 - แต่ละโหนดจะไม่ได้รับแพ็คเก็ตที่ซ้ำกัน

Spanning tree

- ❖ สร้าง spanning tree มาก่อน
- ❖ โหนดส่งต่อ/ทำซ้ำไปตามทาง spanning tree



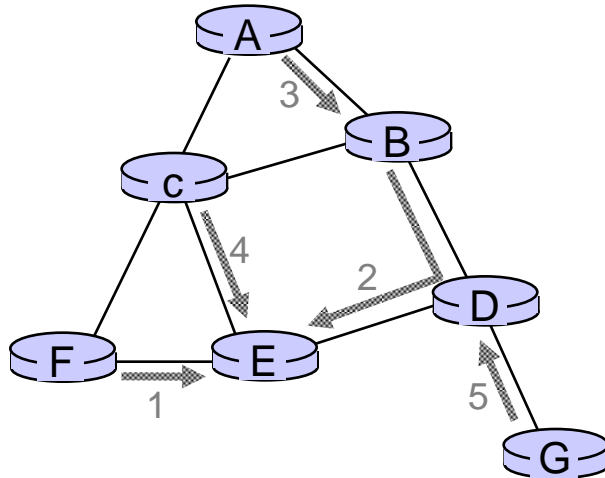
(a) กระจายสัญญาณโดยเริ่มที่ A



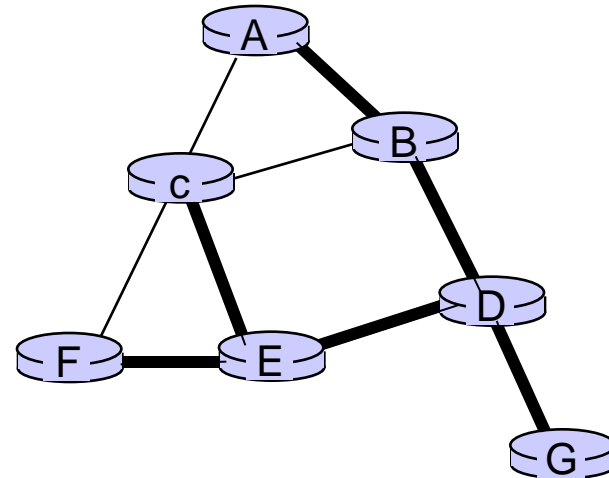
(b) กระจายสัญญาณโดยเริ่มที่ D

Spanning tree: creation

- ❖ โหนดหลัก
- ❖ แต่ละโหนดส่ง unicast รวมข้อความไปถึงโหนดหลัก
 - ข้อความจะถูกส่งไปจนกระทั่งมันไปถึงแต่ละโหนดของ spanning tree แล้ว



(a) ขั้นตอนของการสร้าง spanning tree (โหนดหลักที่: E)

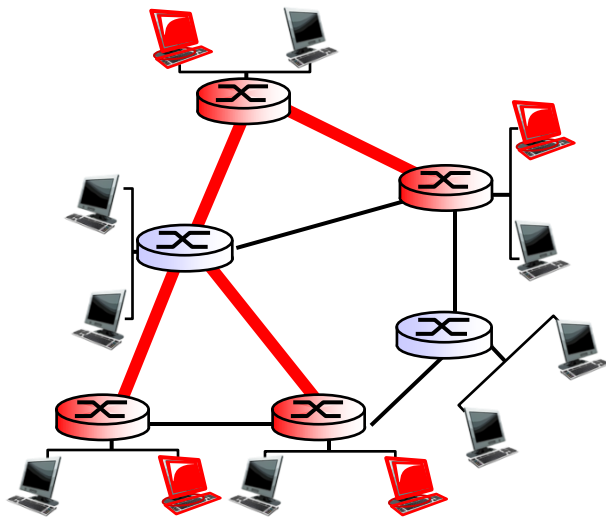


(b) spanning tree ที่ถูกสร้างแล้ว

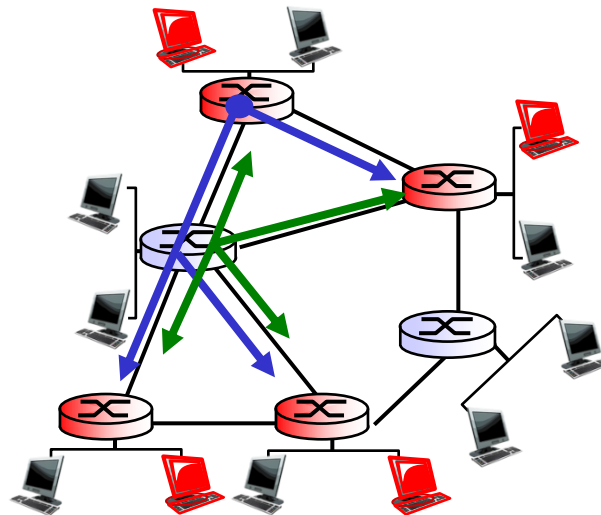
Multicast routing: problem statement

เป้าหมาย: ค้นหา tree หรือ trees ที่ router กำลังเชื่อมต่อกันเพื่อกระจายสัญญาณให้กับกลุ่มผู้ใช้

- ❖ *tree:* ไม่ได้ใช้ทุกเส้นทางที่ routers เชื่อมต่อกัน
- ❖ *shared-tree:* same tree ถูกใช้โดยทุกคนในกลุ่มผู้ใช้
- ❖ *source-based:* different tree จากแต่ละผู้ส่งไปยังผู้รับ



shared tree



source-based trees

legend



กลุ่มผู้ใช้



นอกกลุ่มผู้ใช้



Router กลุ่มผู้ใช้



Router
นอกกลุ่มผู้ใช้

Approaches for building mcast trees

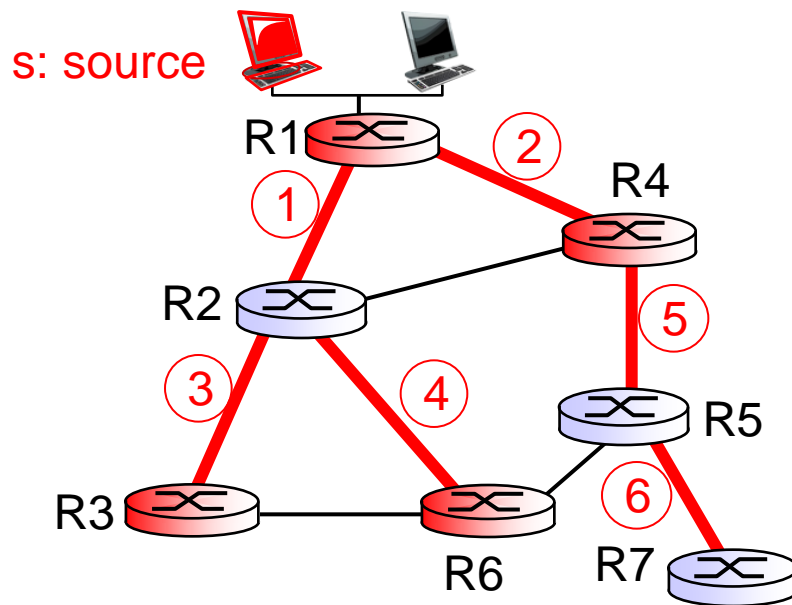
วิธีการทำ:

- ❖ *source-based tree*: หนึ่งฝั่งต่อแหล่ง
 - ฝั่งที่มีเส้นทางที่สั้นที่สุด
 - ย้อนเส้นทางกลับในการส่งต่อ
- ❖ *group-shared tree*: จัดกลุ่มต่อการใช้งานหนึ่งฝั่ง
 - ระยะทางสั้น/น้อย (Steiner)
 - ฝั่งรวมศูนย์

...ให้เราดูวิธีการเบื้องต้นก่อน จากนั้นค่อยระบุโปรโตคอลที่เหมาะสมกับวิธีการเหล่านี้

Shortest path tree

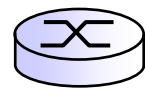
- ❖ mcast การส่งต่อ tree: tree ที่สั้นที่สุด เส้นทาง routes จากแหล่งต้นทางไปสู่ผู้รับทั้งหมด
 - Dijkstra's algorithm



LEGEND



router ที่มีสมาชิกกลุ่ม



router ที่ไม่มีสมาชิกกลุ่ม



link ที่ใช้ในการส่งต่อ,
order link จะทำงานด้วยการใช้ algorithm

Reverse path forwarding

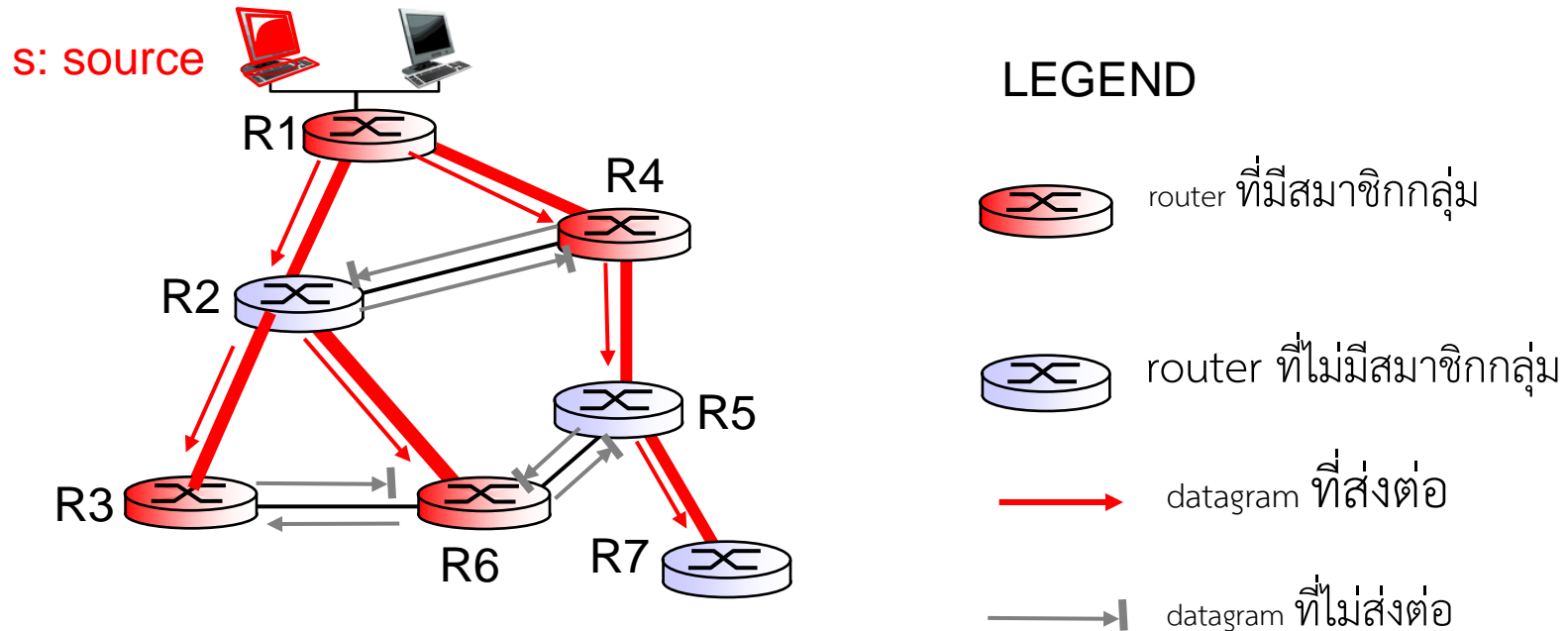
- ❖ ฟังพาข้อมูลจากเราเตอร์ในการหาเส้นทางที่สั้นที่สุดจากเราเตอร์ไปยังผู้ส่ง
- ❖ แต่ละ router มีพฤติกรรมการส่งต่ออย่างง่าย :

if (mcast datagram ที่ได้รับจาก link ขาเข้า เมื่อเดินทางกลับจะกลับเส้นทางที่สั้นที่สุดไปสู่ center)

และ flood datagram ไปสู่ links ขาออกทั้งหมด

else ignore datagram

Reverse path forwarding: example

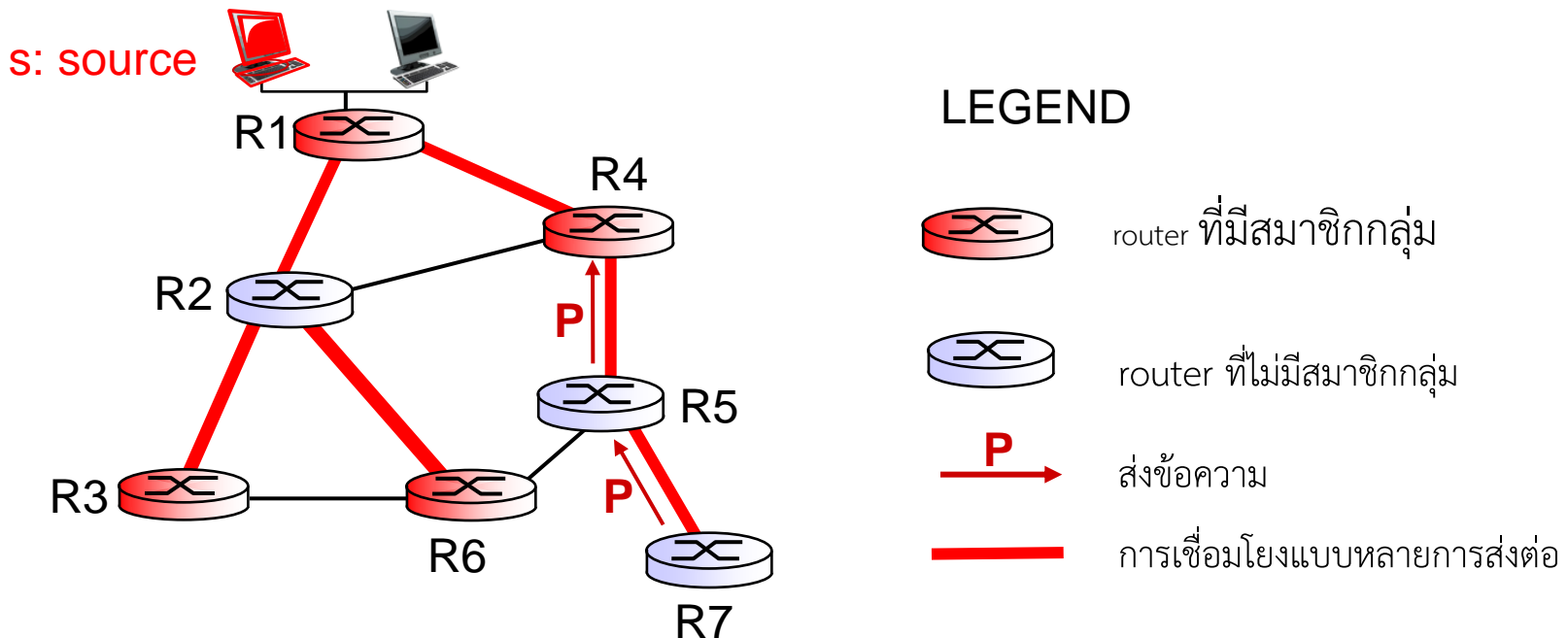


❖ ผลคือจะย้อนกลับไปที่ SPT

- บางทีอาจจะเป็นทางเลือกที่ไม่ดีที่การเชื่อมโยงไม่สมมาตร

Reverse path forwarding: pruning

- ❖ forwarding tree ประกอบด้วย subtrees กับไม่มีสมาชิกกลุ่ม mcast
 - ไม่ต้องการ เพื่อส่งต่อ datagrams down subtree
 - “prune” msgs ส่ง upstream โดย router กับ ไม่ downstream สมาชิกกลุ่ม



Shared-tree: steiner tree

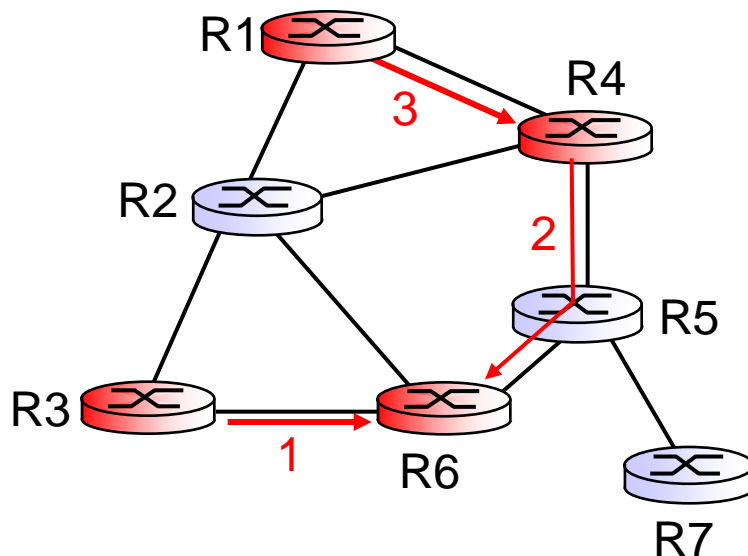
- ❖ *steiner tree*: minimum cost tree connecting all routers with attached group members
- ❖ problem is NP-complete
- ❖ excellent heuristics exists
- ❖ not used in practice:
 - computational complexity
 - information about entire network needed
 - monolithic: rerun whenever a router needs to join/leave

Center-based trees

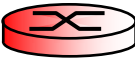


- ❖ single delivery tree shared by all
- ❖ one router identified as “*center*” of tree
- ❖ to join:
 - edge router sends unicast *join-msg* addressed to center router
 - *join-msg* “processed” by intermediate routers and forwarded towards center
 - *join-msg* either hits existing tree branch for this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

Center-based trees: example

suppose R6 chosen as center:



LEGEND

-  router with attached group member
-  router with no attached group member
-  path order in which join messages generated

Internet Multicasting Routing: DVMRP

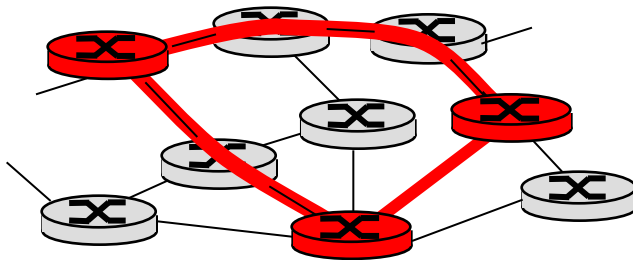
- ❖ **DVMRP**: distance vector multicast routing protocol, RFC1075
- ❖ *flood and prune*: reverse path forwarding, source-based tree
 - RPF tree based on DVMRP's own routing tables constructed by communicating DVMRP routers
 - no assumptions about underlying unicast
 - initial datagram to mcast group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

DVMRP: continued...

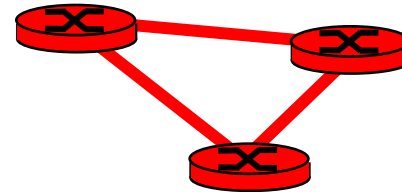
- ❖ *soft state*: DVMRP router periodically (1 min.) “forgets” branches are pruned:
 - mcast data again flows down unpruned branch
 - downstream router: reprune or else continue to receive data
- ❖ routers can quickly regraft to tree
 - following IGMP join at leaf
- ❖ odds and ends
 - commonly implemented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

Tunneling

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



physical topology



logical topology

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

PIM: Protocol Independent Multicast

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

dense:

- ❖ group members densely packed, in “close” proximity.
- ❖ bandwidth more plentiful

sparse:

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

Consequences of sparse-dense dichotomy:

dense

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

sparse:

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

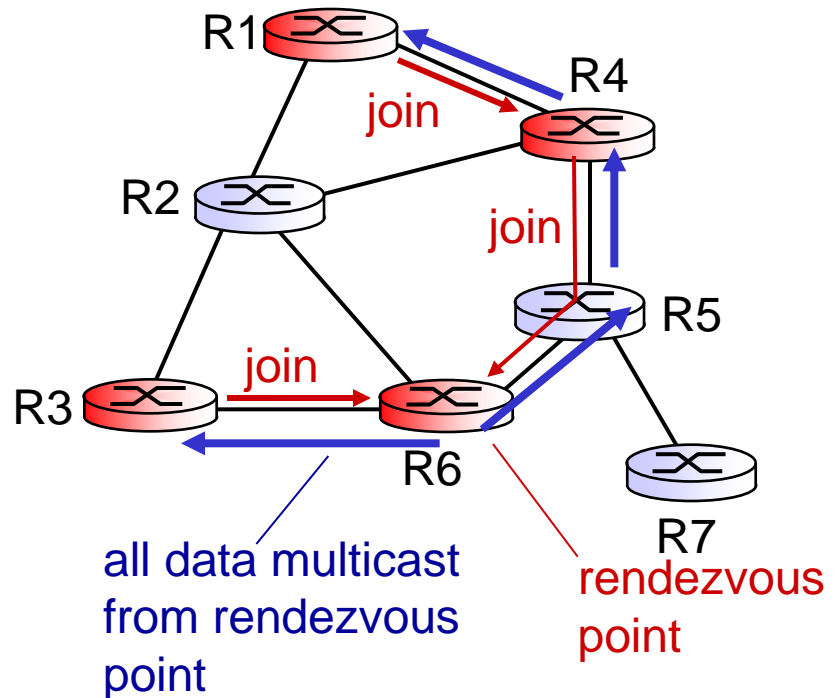
PIM- dense mode

flood-and-prune RPF: similar to DVMRP but...

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

PIM - sparse mode

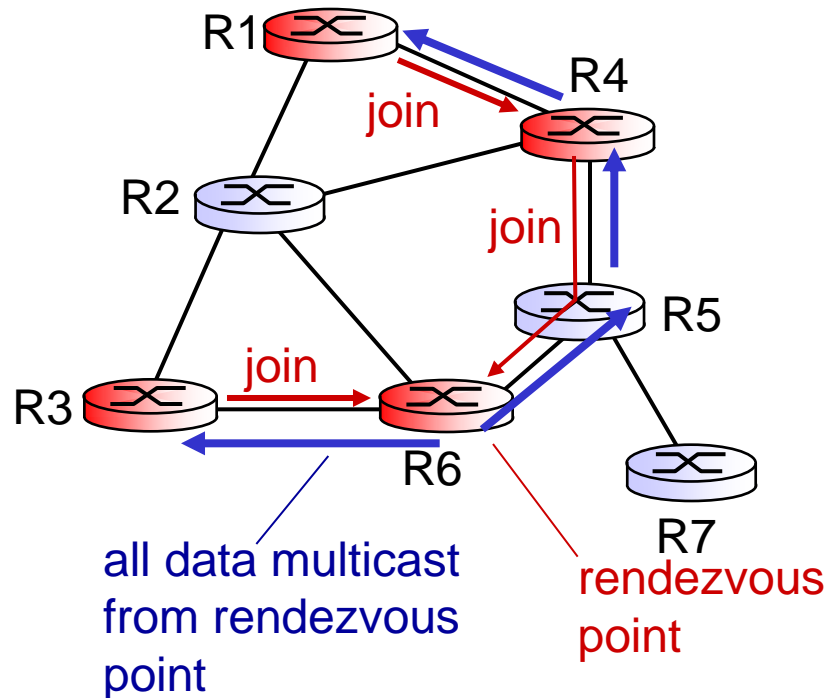
- ❖ center-based approach
- ❖ router sends *join* msg to rendezvous point (RP)
 - intermediate routers update state and forward *join*
- ❖ after joining via RP, router can switch to source-specific tree
 - increased performance: less concentration, shorter paths



PIM - sparse mode

sender(s):

- ❖ unicast data to RP, which distributes down RP-rooted tree
- ❖ RP can extend mcast tree upstream to source
- ❖ RP can send *stop* msg if no attached receivers
 - “no one is listening!”



Chapter 4: *done!*

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol

- datagram format, IPv4 addressing, ICMP, IPv6

4.5 routing algorithms

- link state, distance vector, hierarchical routing

4.6 routing in the Internet

- RIP, OSPF, BGP

4.7 broadcast and multicast routing

- ❖ understand principles behind network layer services:
 - network layer service models, forwarding versus routing
 - how a router works, routing (path selection), broadcast, multicast
- ❖ instantiation, implementation in the Internet