

-컴퓨터 네트워크-

Layer 3 : Routing

2022 Spring
Kyungseop Shin

Course Outline

- IP layer에서의 forwarding table 생성을 위한 기법에 대해 이해
- Routing algorithm 기본 개념에 대해 이해
- Link-state algorithm, Distance-Vector algorithm에 대한 개념 이해
- Internet에서의 routing 메커니즘에 대해 이해

Routing Control Plane

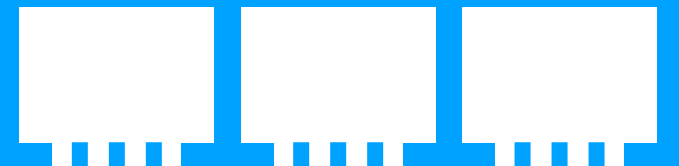
- Router 내의 data/control plane
 - 동작 성격 (Data forwarding / routing)에 따라 기능이 나뉘어짐

Data plane

- IP header 분석
- Datagram forwarding
- Fragmentation

Control plane

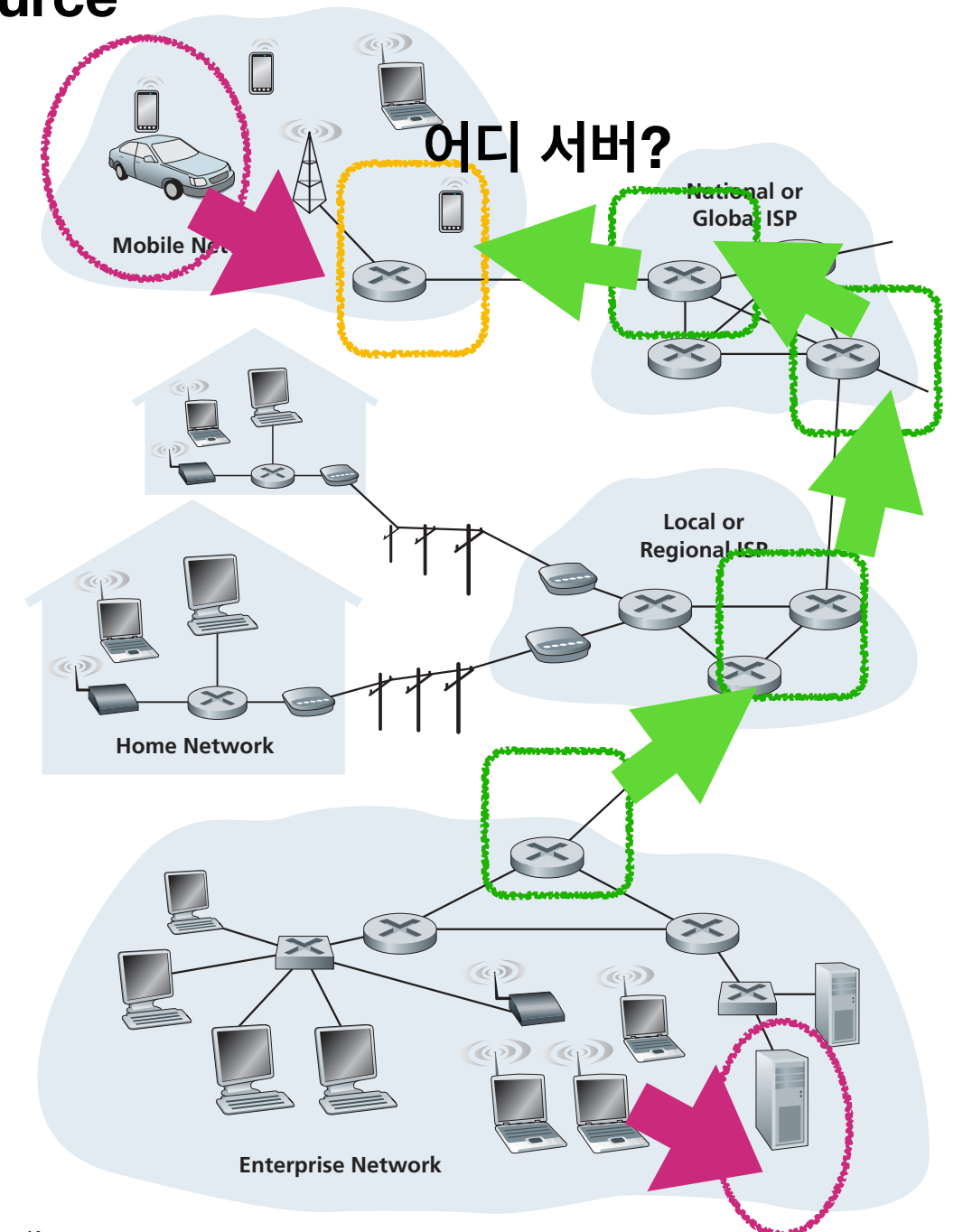
- Routing algorithm
- ICMP



Routing Algorithm

- Router의 forwarding table
- 입력datagram에 대한 출력 port를 정하기 위해 각 router가 보유한 표
- Routing algorithm
- Routing table을 만들기 위한 각 router의 동작
- decentralized

source

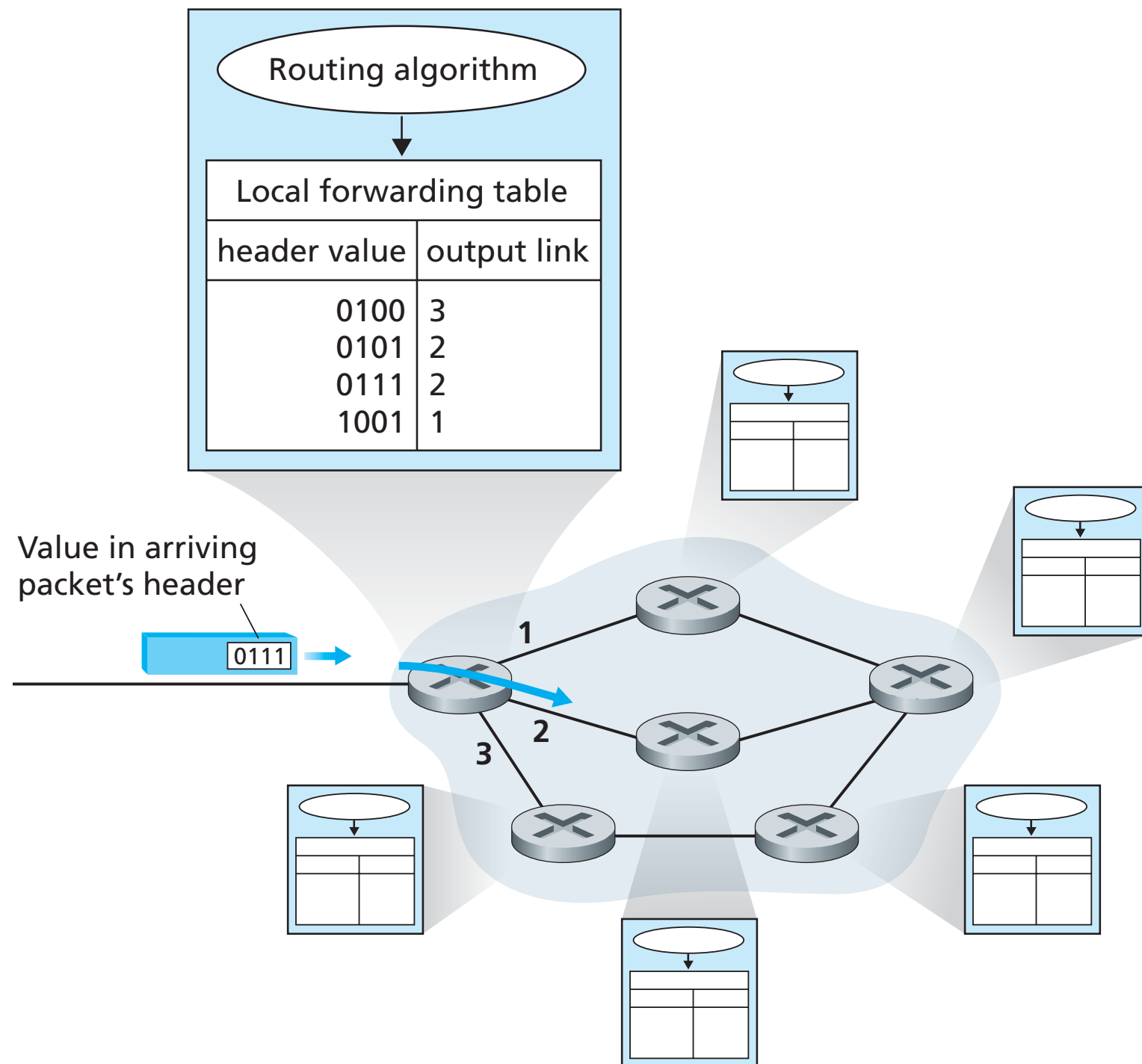


Key:



Routing Algorithm

- Control plane (routing algorithm)에 의해 forwarding table 생성
- Data plane에서는 forwarding table을 이용해서 동작
- 이 IP address 목적지는 어느 port 로 내보내야 궁극적으로 도달하는가?



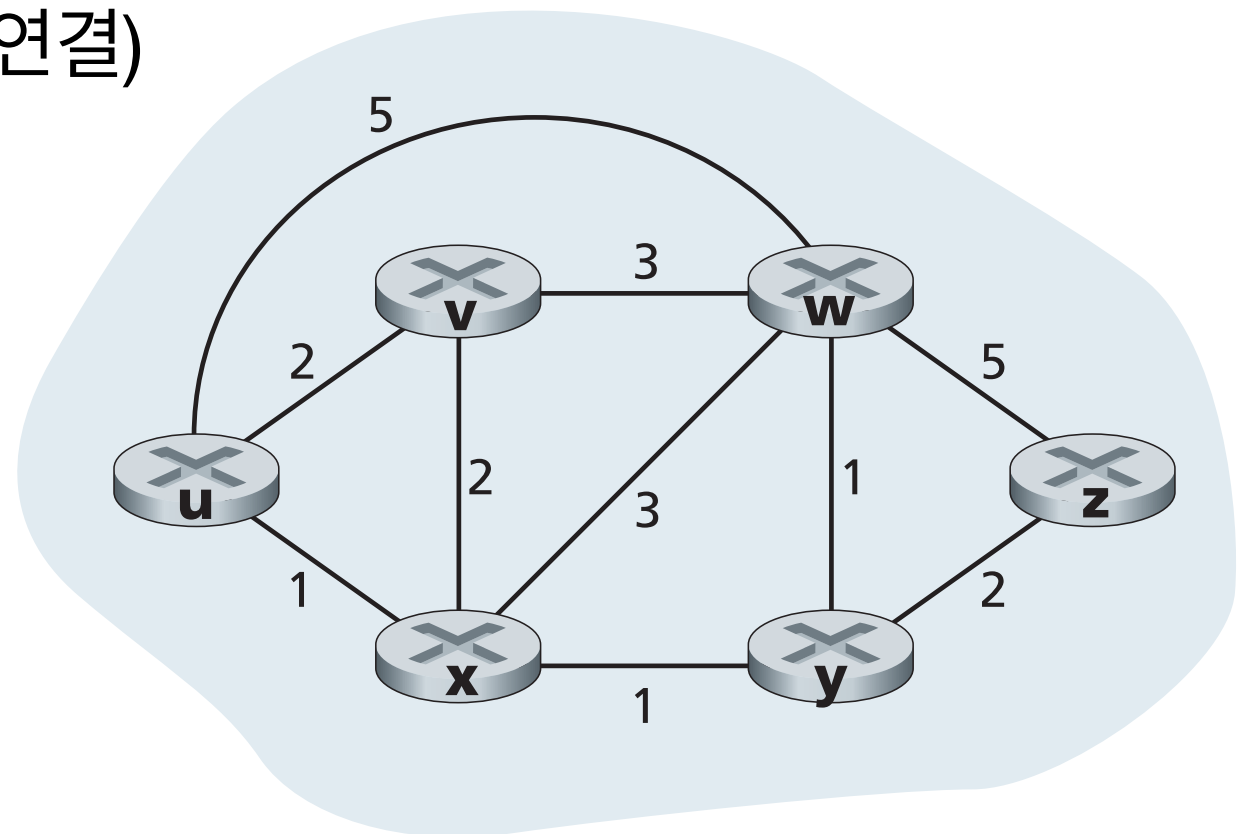
Routing Algorithm

- Routing algorithm의 최종 목적
 - Source router - destination router 간 가장 최적의 경로를 찾는 것



Background

- Routing algorithm은 graph theory를 기본적으로 활용
 - Graph $G = (N, E)$
 - N : node들의 집합 (router)
 - E : edge들의 집합 (router간 연결)
 - Weight : cost = $c(x, y)$
 - x, y 는 Neighbor
 - Path : (x_1, x_2, \dots, x_p)

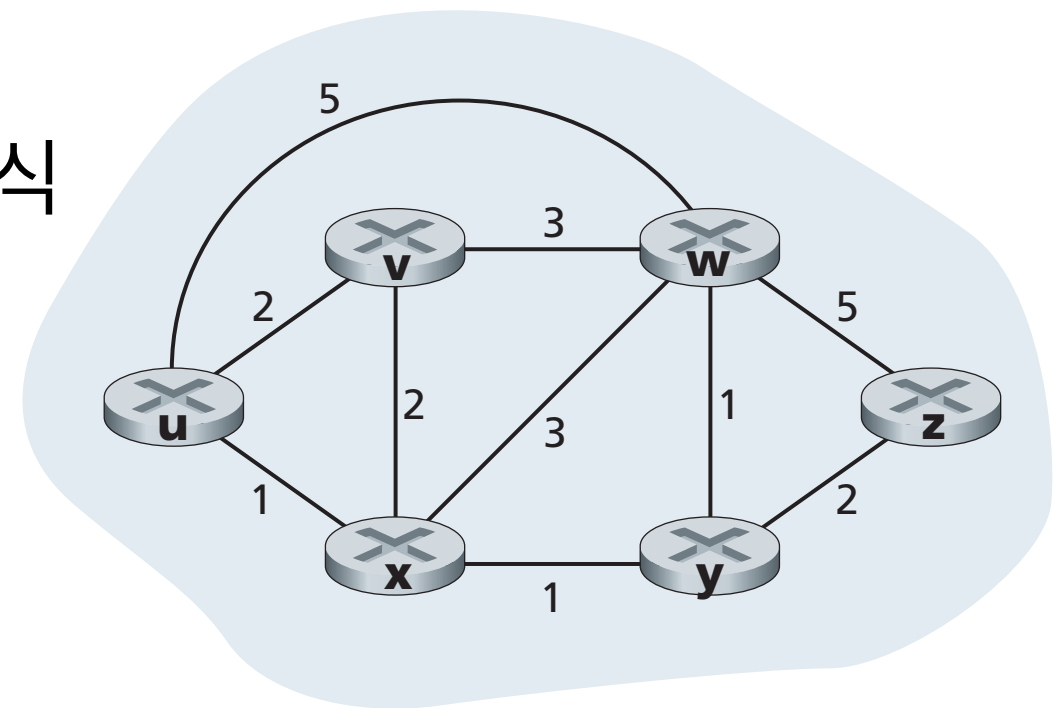


Routing Path

- Least-cost path
 - cost의 합이 최소인 경로 $c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$.
 - 앞의 그림에서 u-w의 경우 u-x-y-w
- Shortest path
 - cost를 고려하지 않고 가장 짧은 path
 - 모든 cost가 같은 경우의 least-cost path

Routing Path

- u - z 간 least-cost path를 구하고자 하면
 - 17가지 모든 경우의 수에 대해 해보면 답을 구할 수 있음
 - = Centralized routing algorithm
 - One location, complete information
 - Decentralized : router들이 하는 방식



Routing Algorithm 종류 :

Global vs. Decentralized

- Global routing algorithm
 - complete/global knowledge about network
 - connectivity, link에 대한 정보
 - Link-state(LS) algorithm
- Decentralized routing algorithm
 - Iterative, distributed manner
 - 특정 node가 모든 link에 대한 정보를 완벽하게 가지지 않으며, 주로 자신과 연결된 link에 대한 정보만 가지고 단계적으로 동작
 - Distance-vector(DV) algorithm

Routing Algorithm 종류 :

Static vs. Dynamic

- Static routing algorithms
 - Routing table이 시간에 따라 변하지 않거나 거의 변하지 않음
 - 주로 사람의 조작에 의해서만 변함
- Dynamic routing algorithms
 - Traffic load 및 topology 변화에 의해 자동으로 routing table 이 변함
 - 잘못하면 Routing loop, oscillation 문제가 발생할 여지가 있음

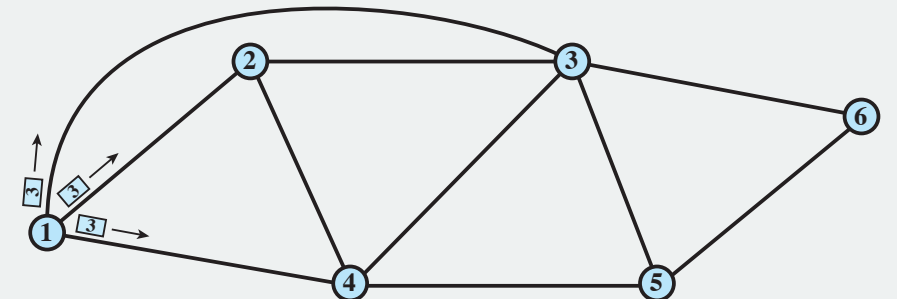
Fixed Routing

- Permanent route : 고정으로 routing 규칙을 지정
 - IP address 특정 대역 등에 대해 out-port 지정
- 활용 시점
 - 전체 관리가 사람에 의해 이루어질 때
 - 부분적으로 고정된 규칙을 적용하고 싶을 때
- 간단하지만 flexibility가 적고 수고가 많이 듦
 - Network failure, congestion 등의 dynamic한 상황에 취약함

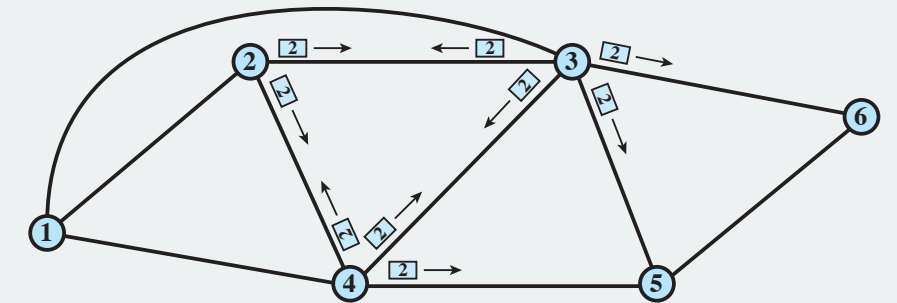


Flooding

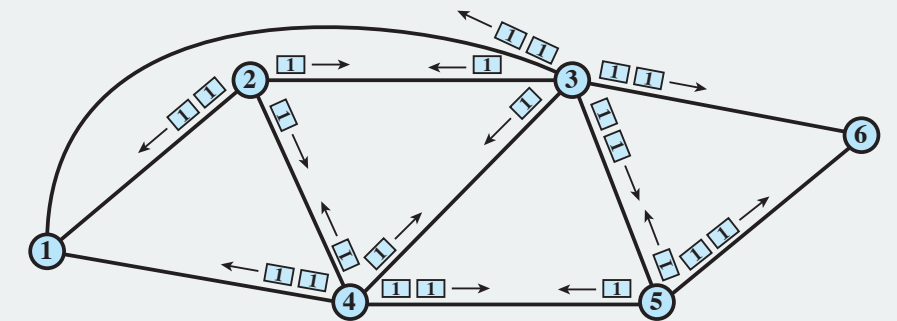
- 모든 neighbor들에게 packet을 무조건 뿌림
- 궁극적으로 destination은 multiple copy 형태로 수신
 - Sequence number를 통해 duplication 방지 가능
- 단순하면서 무식한 방법
 - Routing protocol이나 table등이 필요 없으므로, control plane관점에서 simple함
 - traffic load가 불필요하게 너무 크며 재전송에 대한 제어가 필요



(a) First hop



(b) Second hop



(c) Third hop

Figure 19.3 Flooding Example (hop count = 3)

Link State Routing

- Topology 및 모든 link cost를 파악하고 routing table 생성
 - Link-state broadcast algorithm을 통해 활용
 - 각자의 Identity와 link 별 cost 정보를 neighbor에게 전달
 - 모든 node가 동일한 routing 정보를 공유할 수 있음
- Dijkstra's algorithm : least-cost path를 iterative하게 파악
 - $D(v)$: cost of the least-cost path from the source node to destination v as of this iteration of the algorithm.
 - $p(v)$: previous node (neighbor of v) along the current least-cost path from the source to v .
 - N' : subset of nodes; v is in N' if the least-cost path from the source to v is definitively known.

Link State Routing

- 순차적으로 node를 넣고 least-cost path를 갱신

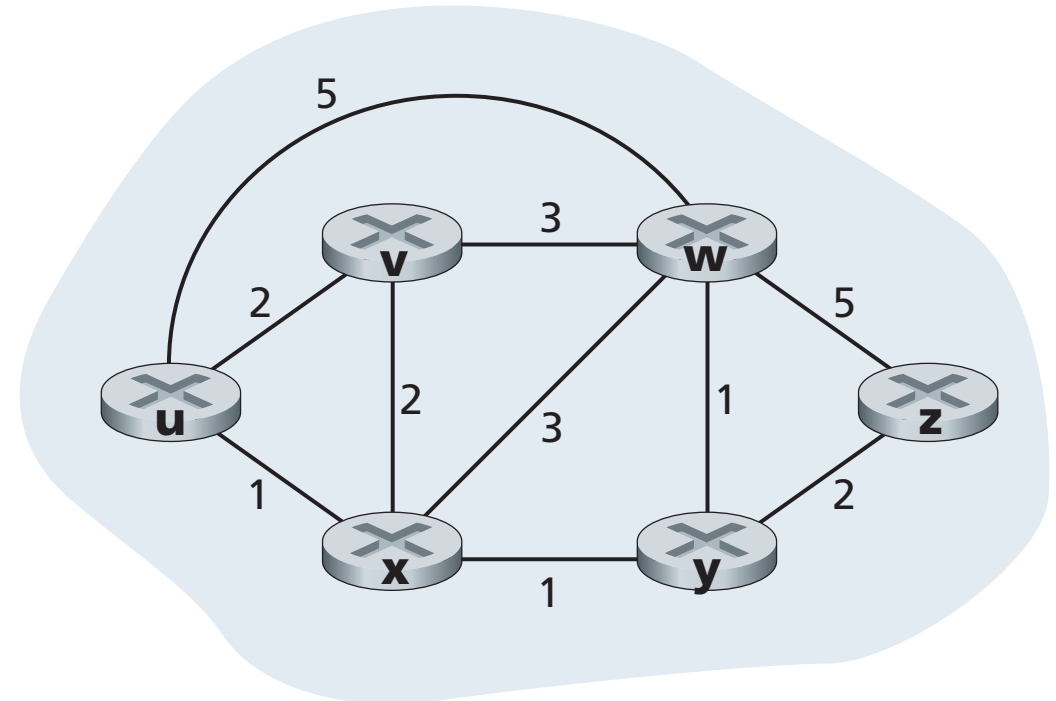
Link-State (LS) Algorithm for Source Node u

```
1  Initialization:
2       $N' = \{u\}$ 
3      for all nodes  $v$ 
4          if  $v$  is a neighbor of  $u$ 
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 each neighbor  $v$  of  $w$  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        least path cost to  $w$  plus cost from  $w$  to  $v$  */
15 until  $N' = N$ 
```

Link State Routing

- 예시

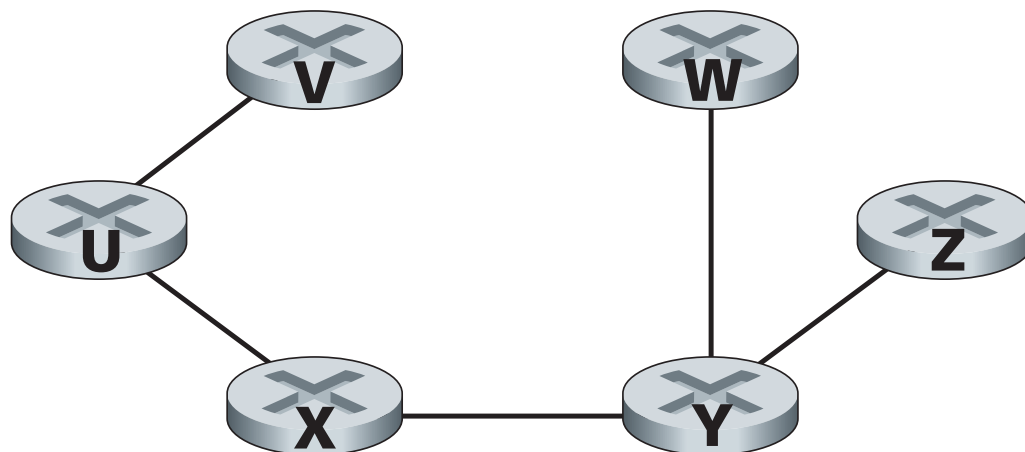
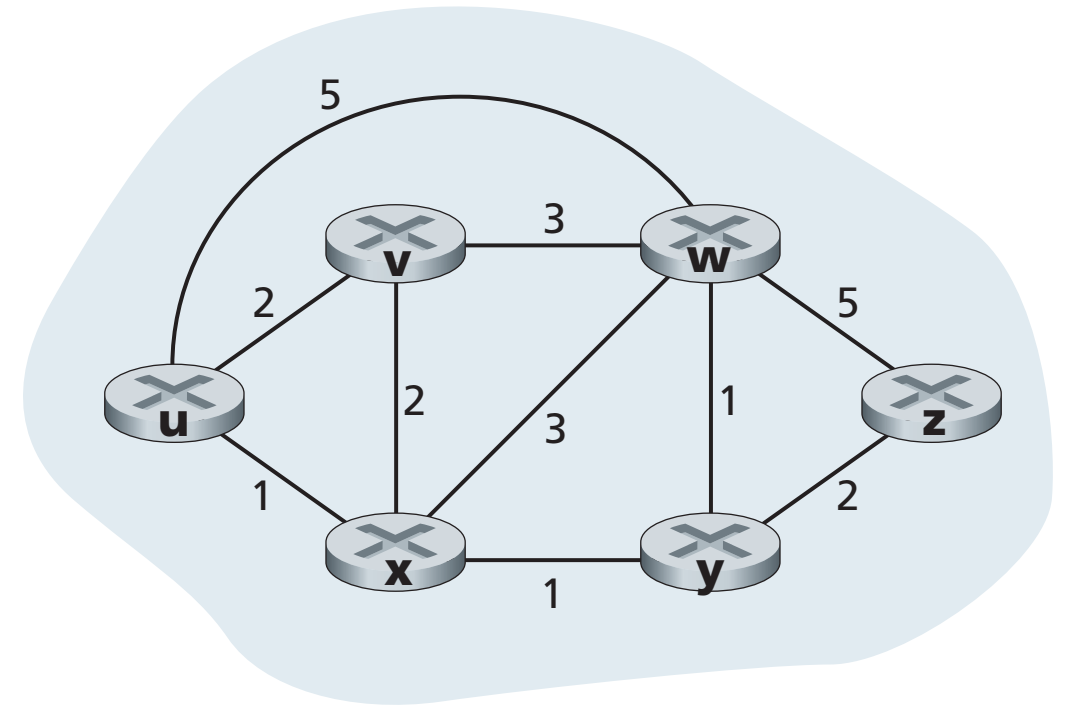
- node가 늘어날수록 $D()$ 값이 줄어
듬
- p 를 추적하면 path를 찾을 수 있음



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					

Link State Routing

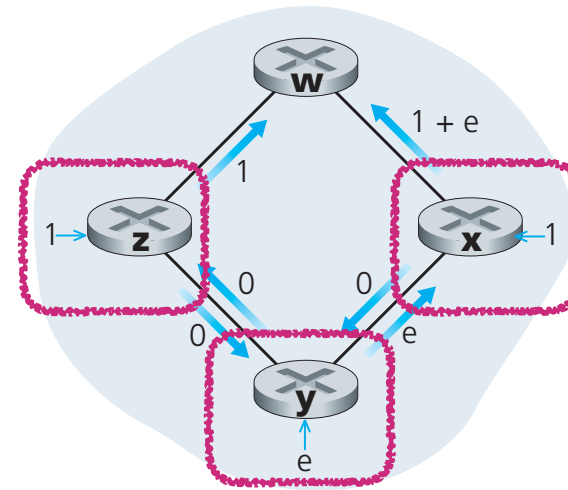
- Forwarding table 결과
- Routing path를 고려하여 destination 별로 out-port 지정



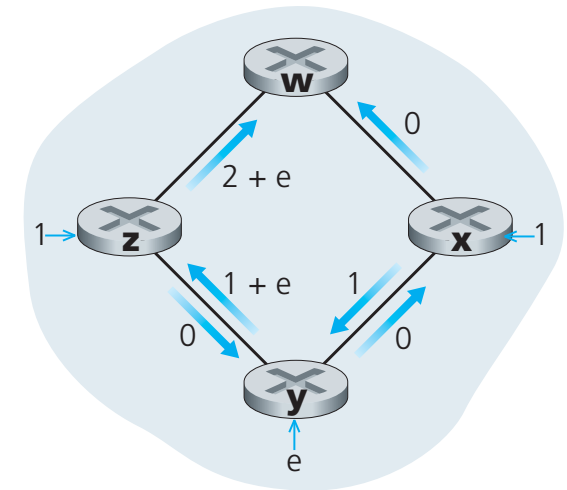
Destination	Link
v	(u, v)
w	(u, x)
x	(u, x)
y	(u, x)
z	(u, x)

Link State Routing

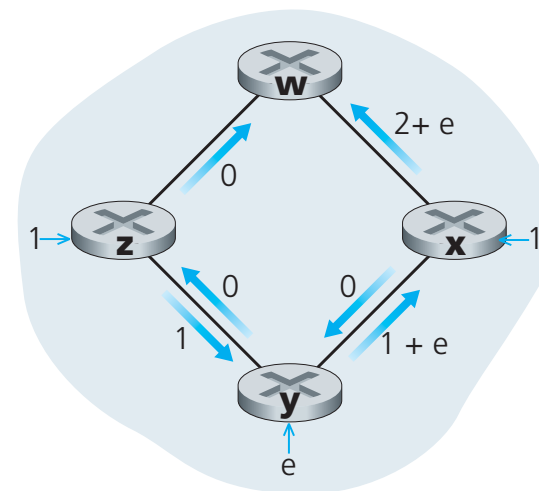
- Congestion-sensitive routing
과정에서의 oscillation
- Link cost를 traffic load로
가정하면
- Traffic 상태에 따라 Routing
경로가 안정적이지 못하고 왔다
다갔다 할수 있음



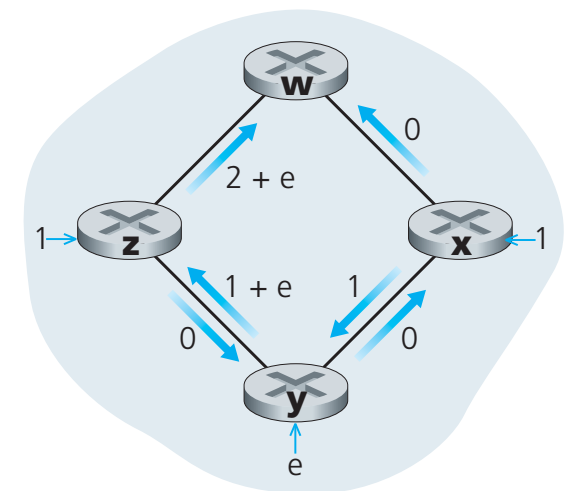
a. Initial routing



b. x, y detect better path to w, clockwise



c. x, y, z detect better path to w, counterclockwise



d. x, y, z, detect better path to w, clockwise

Distance-Vector Routing

- LS routing과는 다르게 asynchronous / distributed 함
 - Neighbor node끼리만 정보를 주고 받으면서 routing 계산
 - 더이상 neighbor node끼리 정보를 주고 받을 일이 없을 때 까지 iterative하게 정보 교환 / 계산을 반복
 - 각 node가 알아서 정보 교환을 하고 계산 수행
- Bellman-Ford equation 활용

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

v 는 x 의 neighbor

Node y 에 대한 packet의 forwarding table element

Distance-Vector Routing

- 기본적인 routing algorithm 방향
 - 각 node x 는 $D_x(y)$ 값을 특정 초기값으로 적절히 설정하고
 - Distance vector \mathbf{D}_x 에 대해 아래 정보를 통해 지속적으로 update
- For each neighbor v , the cost $c(x, v)$ from x to directly attached neighbor, v
- Node x 's distance vector, that is, $\mathbf{D}_x = [D_x(y): y \text{ in } N]$, containing x 's estimate of its cost to all destinations, y , in N
- The distance vectors of each of its neighbors, that is, $\mathbf{D}_v = [D_v(y): y \text{ in } N]$ for each neighbor v of x

Distance-Vector Routing

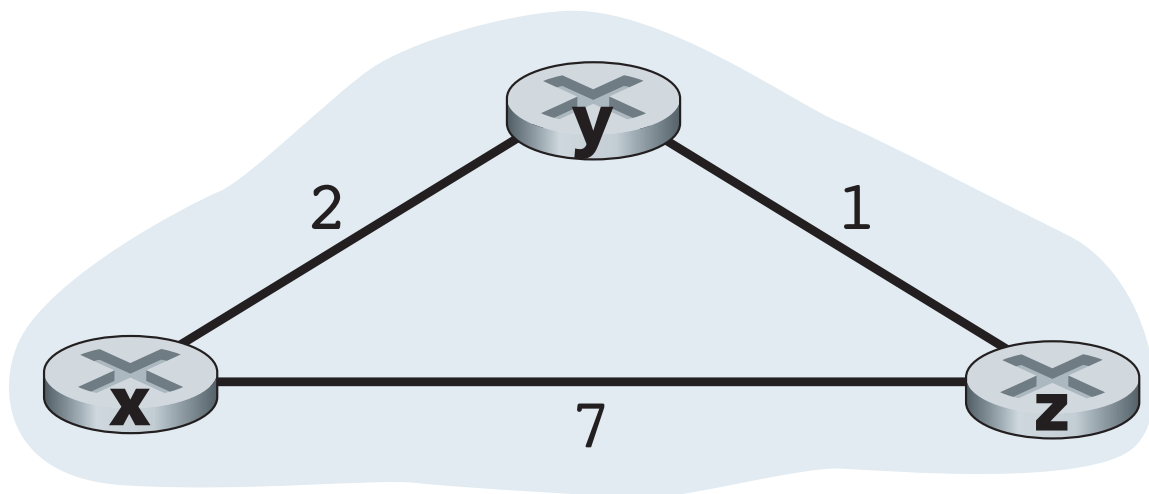
- Neighbor node v 로부터 DV를 받으면
 - Bellman-Ford equation에 의해 자신의 DV를 갱신

```
1  Initialization:
2      for all destinations  $y$  in  $N$ :
3           $D_x(y) = c(x,y)$  /* if  $y$  is not a neighbor then  $c(x,y) = \infty$  */
4      for each neighbor  $w$ 
5           $D_w(y) = ?$  for all destinations  $y$  in  $N$ 
6      for each neighbor  $w$ 
7          send distance vector  $\mathbf{D}_x = [D_x(y): y \text{ in } N]$  to  $w$ 
8
9  loop
10     wait (until I see a link cost change to some neighbor  $w$  or
11           until I receive a distance vector from some neighbor  $w$ )
12
13     for each  $y$  in  $N$ :
14          $D_x(y) = \min_v \{c(x,v) + D_v(y)\}$ 
15
16     if  $D_x(y)$  changed for any destination  $y$ 
17         send distance vector  $\mathbf{D}_x = [D_x(y): y \text{ in } N]$  to all neighbors
18
19 forever
```

Distance-Vector Routing

- 이웃의 DV정보만 이용하며
DV 정보 교환을 각 node가 알아서 asynchronous하게 함

- 예시



Node x table

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

Node y table

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

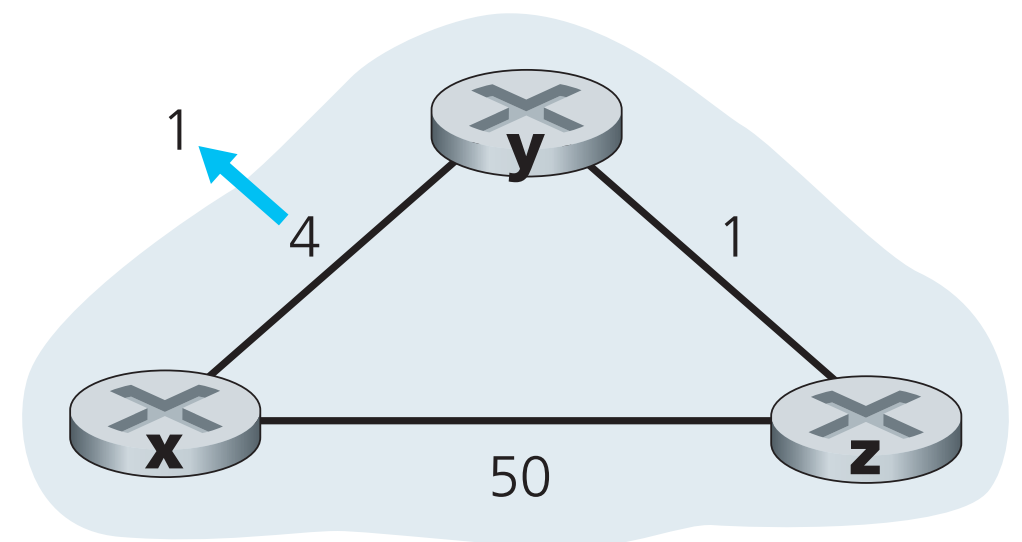
Node z table

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

Time

Distance-Vector Routing

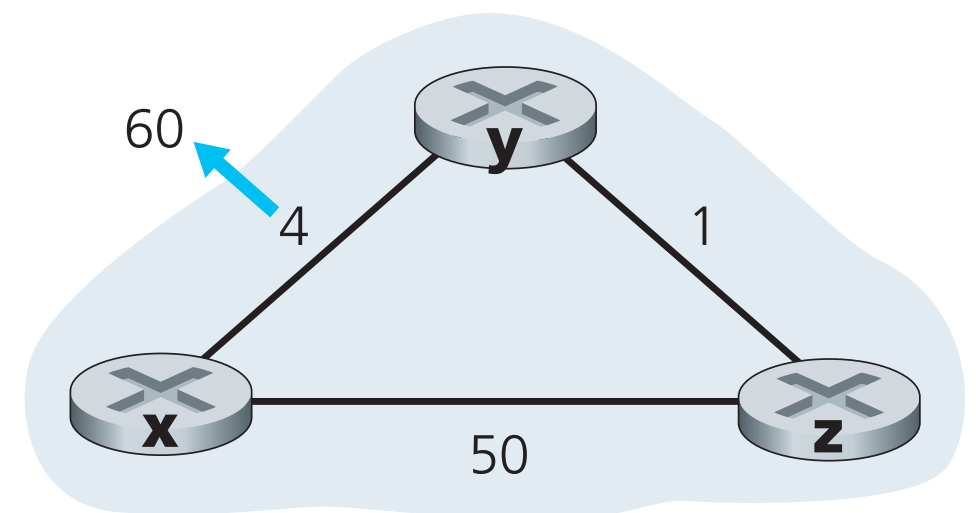
- Link-cost change 1 : cost reduced
 - t_0 : y detects cost change (4 \rightarrow 1)
 - t_1 : z receives update ($D_z(x)$: 5 \rightarrow 2)
 - t_2 : y receives update (no change)



a.

Distance-Vector Routing

- Link-cost change 2 : cost increased , $D_y(x) = 4$, $D_y(z) = 1$, $D_z(y) = 1$, and $D_z(x) = 5$
 - t_0 : y detects (4 \rightarrow 60), $D_y(x) = D_y(z) + D_z(x) = 6$
 - t_1 : z receives update ($D_z(x)$: 5 \rightarrow 7)
 - Routing loop
 - t_2 : y receives update ($D_y(x)=8$)
 - 이후 link cost가 50이 될때까지 2씩 증가
 - \Rightarrow To much iterations



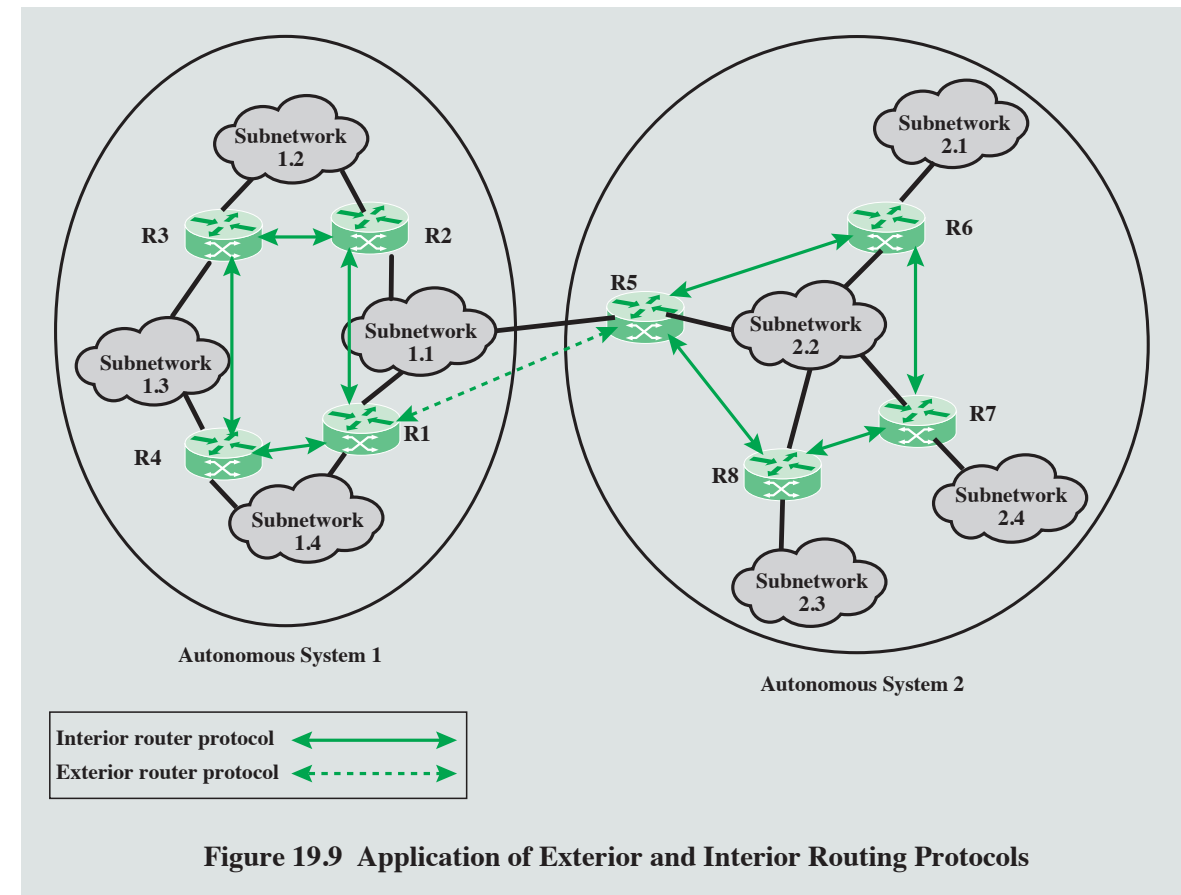
b.

Link State vs. Distance-Vector

- DV : 인접 node끼리만 정보 교환, 정보 내용은 모든 node에 대한 cost
- LS : 모든 node와 정보 교환, 정보 내용은 인접 node에 대한 cost
- Message complexity
 - LS 는 많은 메시지 전송이 필요, DV는 link cost 변화 빈도에 따라 메시지 전송 횟수 증가
- Speed of convergence : DV가 convergence 속도가 느림
- Robustness : LS는 각자 routing을 계산하는 방식이라 더 안정적임

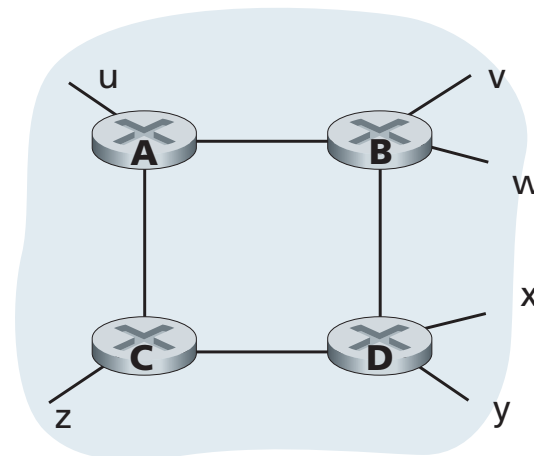
Hierarchical Routing

- 실제 Internet은 sub-network와 gateway router 형태로 계층적으로 네트워크를 형성
 - Subnet간 packet routing은 gateway router를 통해서만 이루어짐
- Autonomous System (AS) : 한 기관이 관리하는 router 및 network 모음
 - 동일한 routing protocol에 의해 밀접하게 동작
- Intra vs. inter-AS routing



Routing Information Protocol (RIP)

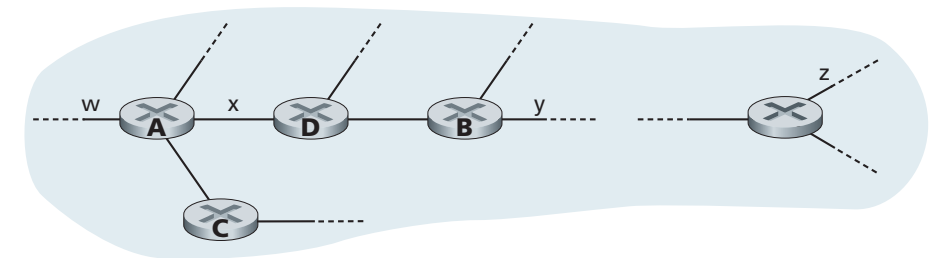
- intra-AS routing에 주로 활용되는 routing algorithm
- Distance-vector protocol에 속하며, link cost가 모두 1
 - Hop count가 cost metric임 (max cost : 15)
- RIP response/advertisement message (30초에 한번 전송)



Destination	Hops
u	1
v	2
w	2
x	3
y	3
z	2

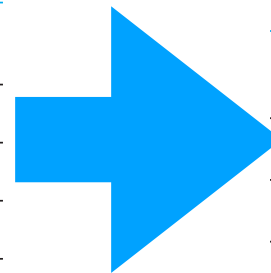
Routing Information Protocol (RIP)

- Routing table : distance vector 및 forwarding table
- Routing advertisement message 수신 후
 - Routing 정보 갱신



Destination Subnet	Next Router	Number of Hops to Destination
z	C	4
w	—	1
x	—	1
....

Destination Subnet	Next Router	Number of Hops to Destination
w	A	2
y	B	2
z	B	7
x	—	1
....



Destination Subnet	Next Router	Number of Hops to Destination
w	A	2
y	B	2
z	A	5
....

Routing Information Protocol (RIP)

- 180초 이상 routing advertisement message 수신이 없으면
 - No longer reachable이라 판단
 - Routing table 을 수정하고 advertisement를 보냄
- RIP request message : 특정 destination에 대한 neighbor's cost를 요청 (UDP, port 520)

Open Shortest Path First (OSPF)

- RIP의 다음 버전이며, 여러가지 개선된 기능을 가짐
- Dijkstra least-cost path algorithm 기반이며, link state에 대한 flooding 활용
 - Complete topological map을 기반으로 모든 subnet에 대한 shortest-path 계산
 - Link cost는 network administrator가 지정하기 나뉘며, 모든 link cost를 1로 설정하면 minimum-hop routing이 됨
- 모든 router에게 link state information을 broadcasting함
 - robustness를 위해서 link change가 없어도 최소한 30분에 한번 수행

Border Gateway Protocol (BGP)

- Internet에서의 Exterior router protocol
 - AS 간 routing을 위한 protocol
- 아래 세가지 역할 수행
 - 인접 AS간 subnet reachability information 교환
 - Reachability 정보에 대해 AS내 router에게 전달
 - subnet에 대한 routing 결정

Key Design Issue

- Routing performance
 - Correctness, Optimality
- Low control overhead
 - Simplicity, Efficiency
- Robustness, Stability, Fairness