Routing

Dept. of Computer Science and Engineering
Sogang University

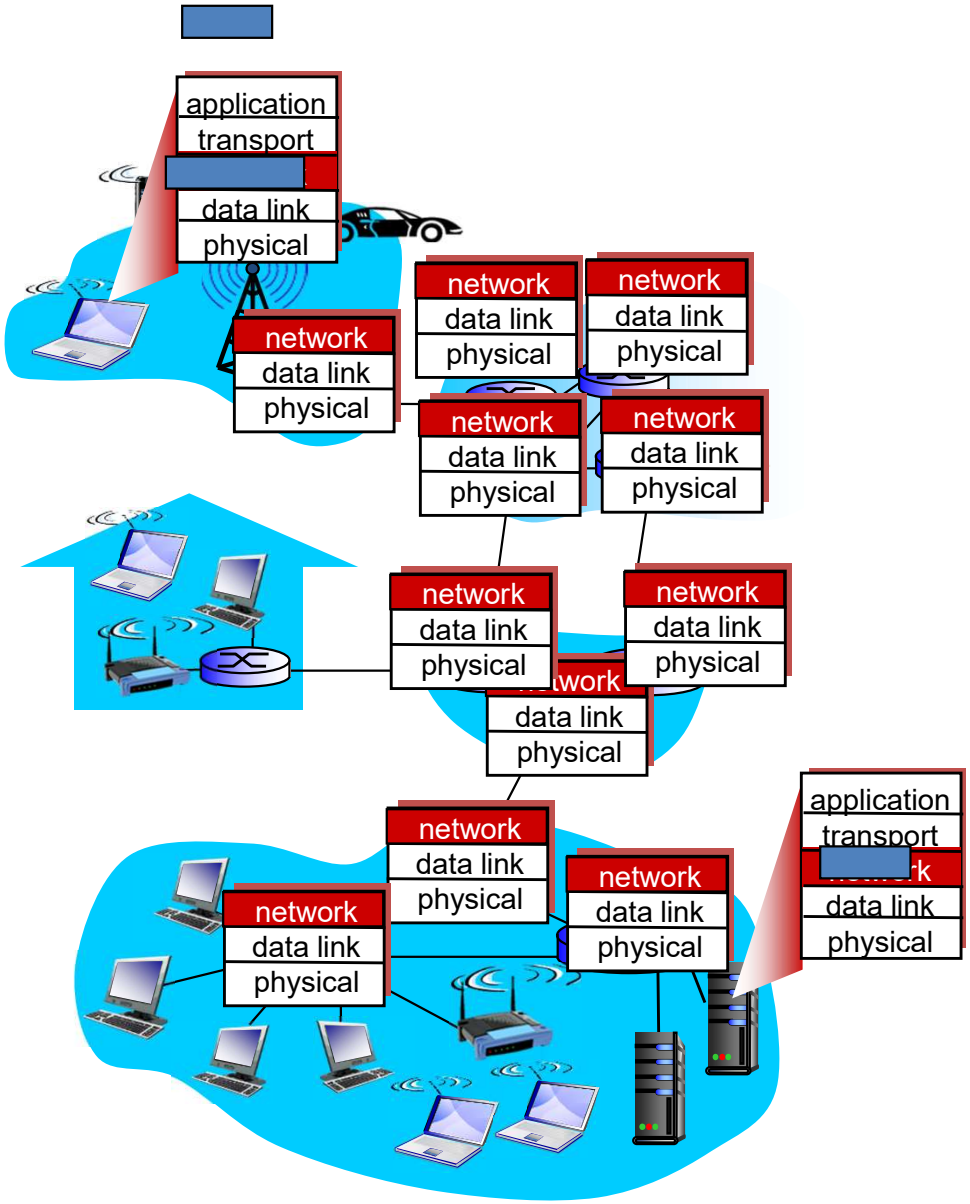
Overview

Network Layer

- Transports segments 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

Application
Transport
Network
Link
Physical

Network Layer



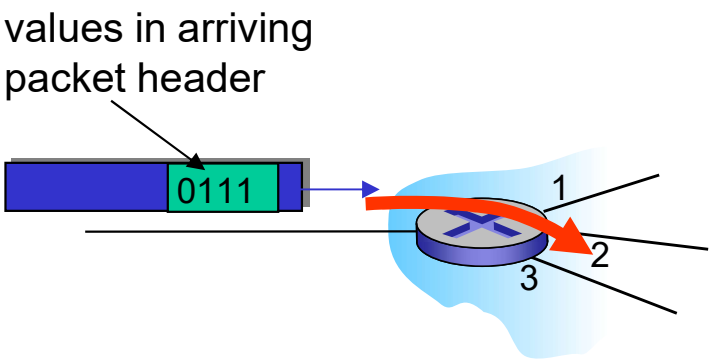
Network Layer Functions

- Forwarding: move packets from router's input to appropriate router output
  - analogy: process of getting through a single interchange
  - **DATA PLANE**

- Routing: determine route taken by packets from source to destination
  - analogy: process of planning trip from source to destination
  - **CONTROL PLANE**

Network Layer: Data Plane

- Local, per-router function
- Determines how datagram arriving on router input is forwarded to router output port
- Forwarding function



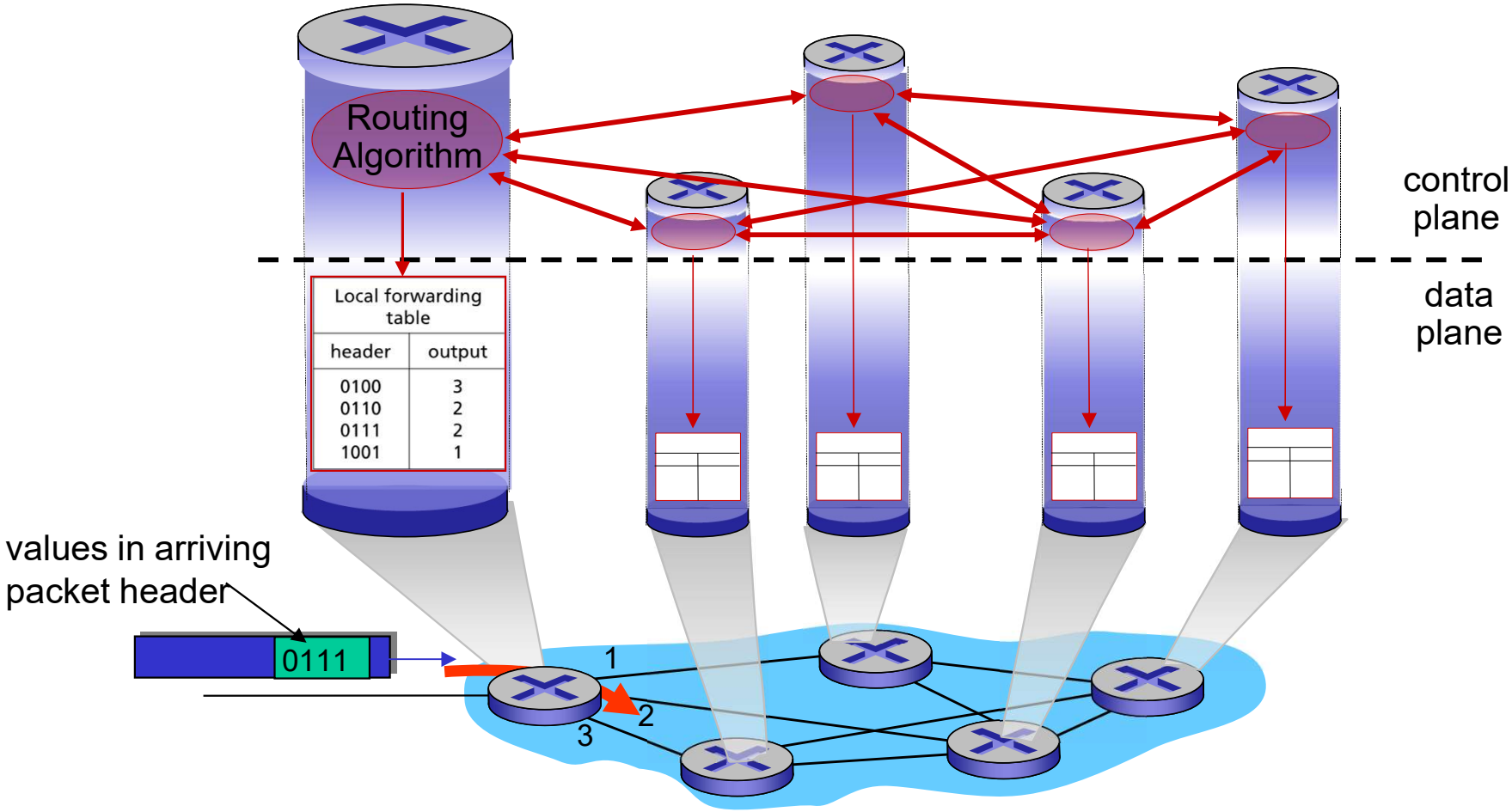
Network Layer: Control Plane

- Network-wide logic
- Determines how datagram is routed among routers along end-to-end path from source host to destination host

- Two control-plane approaches
  - traditional routing algorithms: implemented in routers
  - software-defined networking (SDN): implemented in (remote) servers

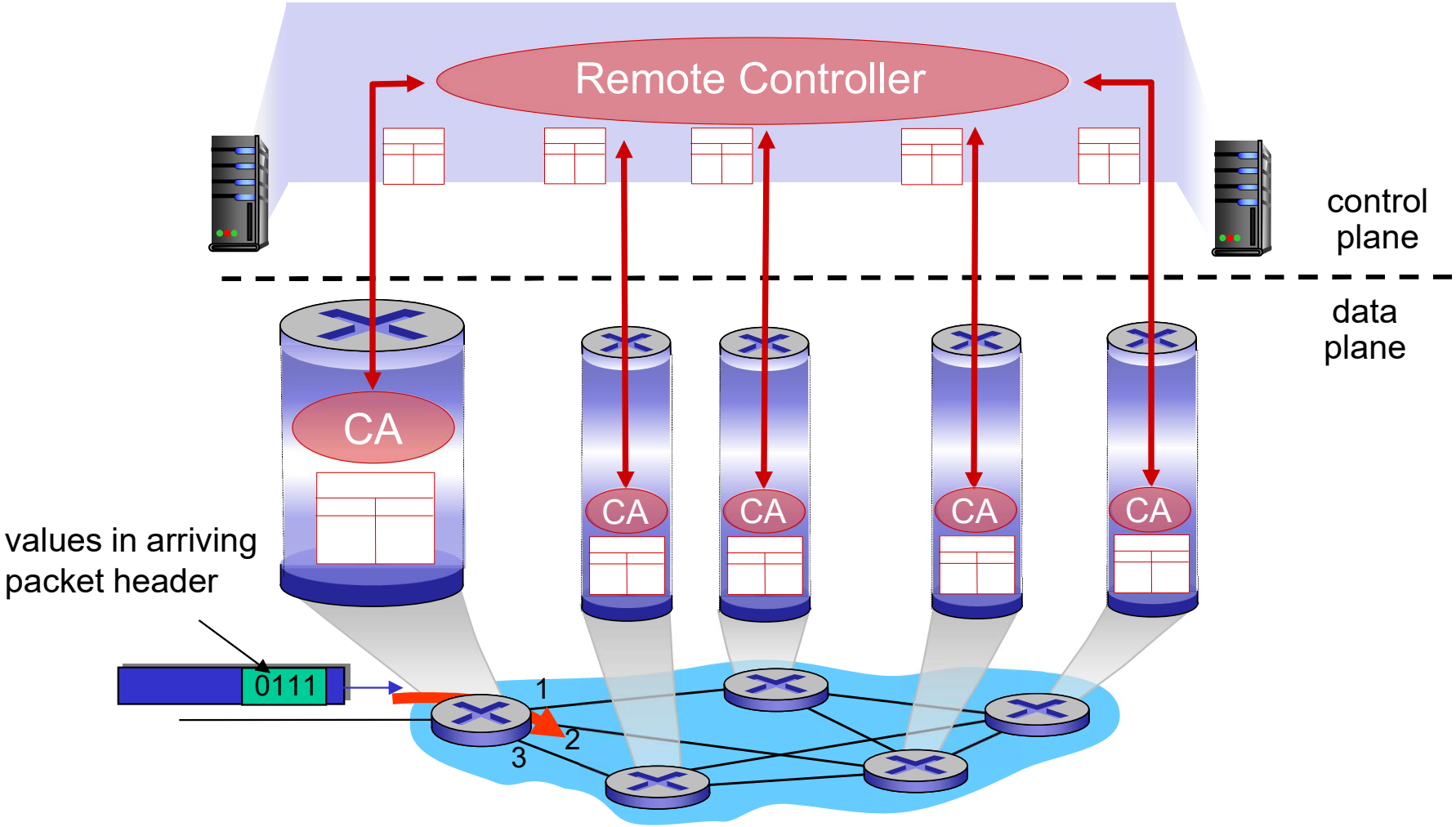
Per-Router Control Plane

- Individual routing algorithm components in each and every router interact in the control plane



Logically Centralized Control Plane

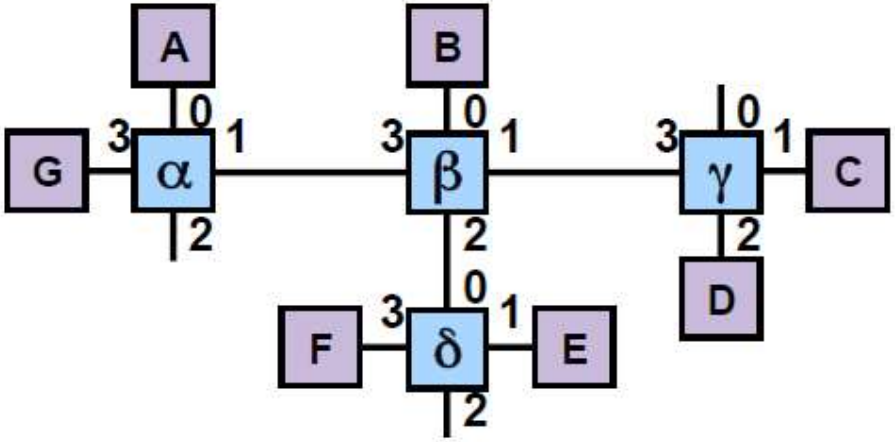
- A distinct (typically remote) controller interacts with local control agents (CAs)



Routing

Routing

- Deciding path on which packets are forwarded
- Method for routers to construct the <u>forwarding table</u>



α's Table

A	0
B	1
C	1
D	1
E	1
F	1
G	3

β's Table

A	3
B	0
C	1
D	1
E	2
F	2
G	3

γ's Table

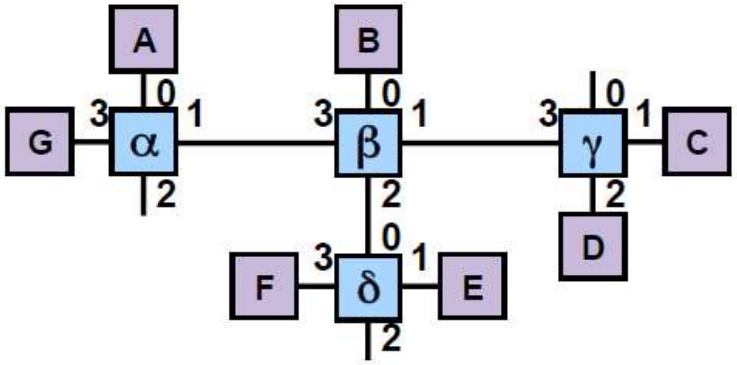
A	3
B	3
C	1
D	2
E	3
F	3
G	3

δ's Table

A	0
B	0
C	0
D	0
E	1
F	3
G	0

Routing

- Delivering a packet from A to F
  - Router α sends packet to output port 1
  - Router β sends packet to output port 2
  - Router δ sends packet to output port 3



α's Table

A	0
B	1
C	1
D	1
E	1
F	1
G	3

β's Table

A	3
B	0
C	1
D	1
E	2
F	2
G	3

γ's Table

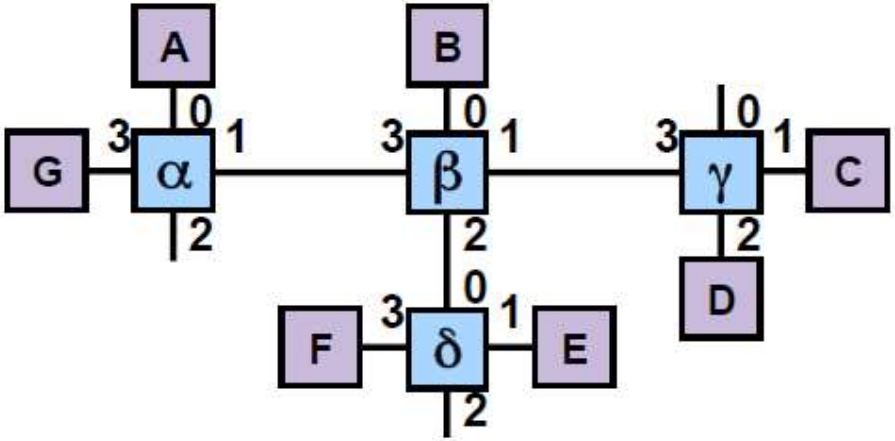
A	3
B	3
C	1
D	2
E	3
F	3
G	3

δ's Table

A	0
B	0
C	0
D	0
E	1
F	3
G	0

Routing

- How does switch α know that the packet should go through output port 1 to reach F?



α's Table

A	0
B	1
C	1
D	1
E	1
F	1
G	3

β's Table

A	3
B	0
C	1
D	1
E	2
F	2
G	3

γ's Table

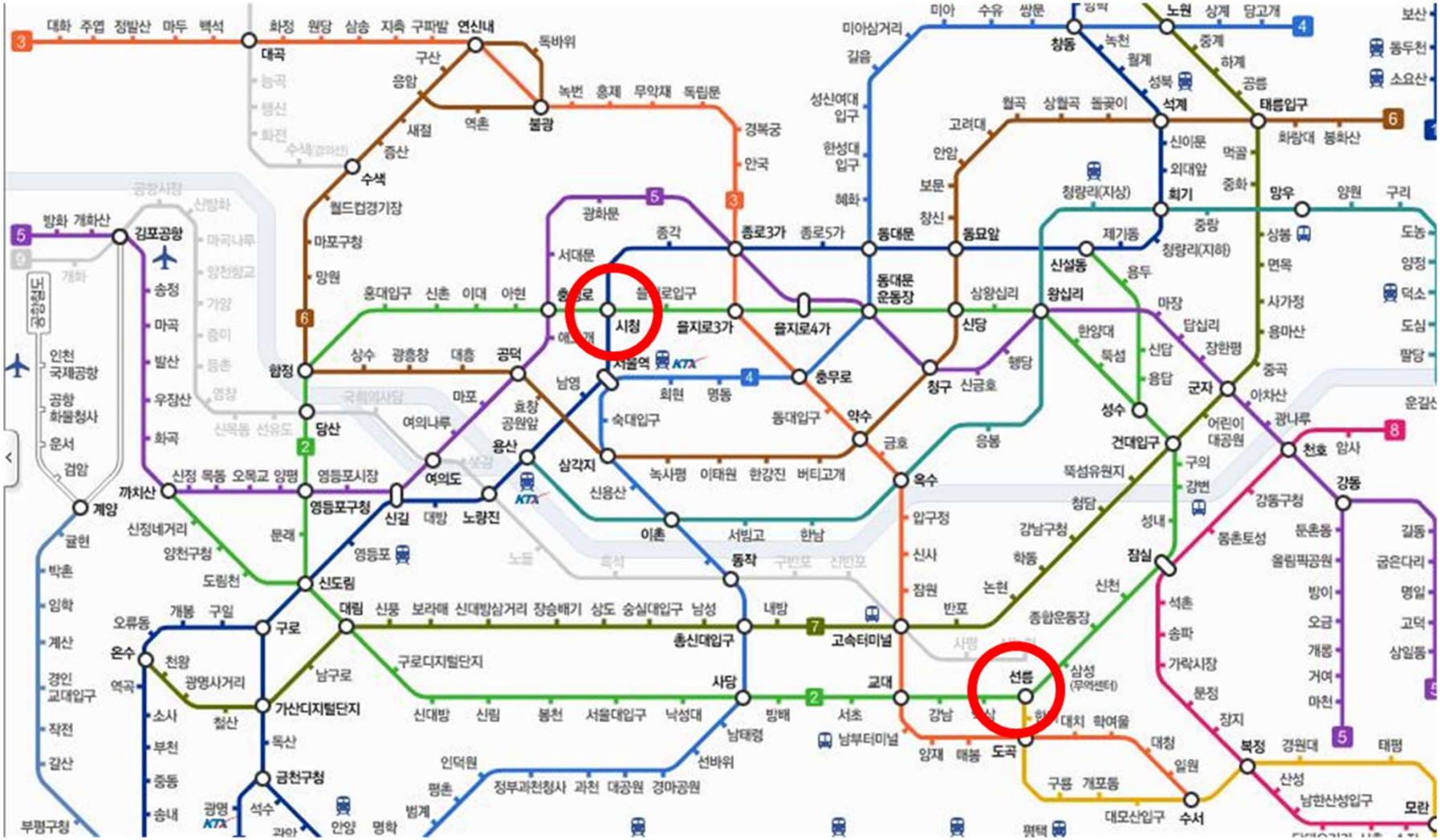
A	3
B	3
C	1
D	2
E	3
F	3
G	3

δ's Table

A	0
B	0
C	0
D	0
E	1
F	3
G	0

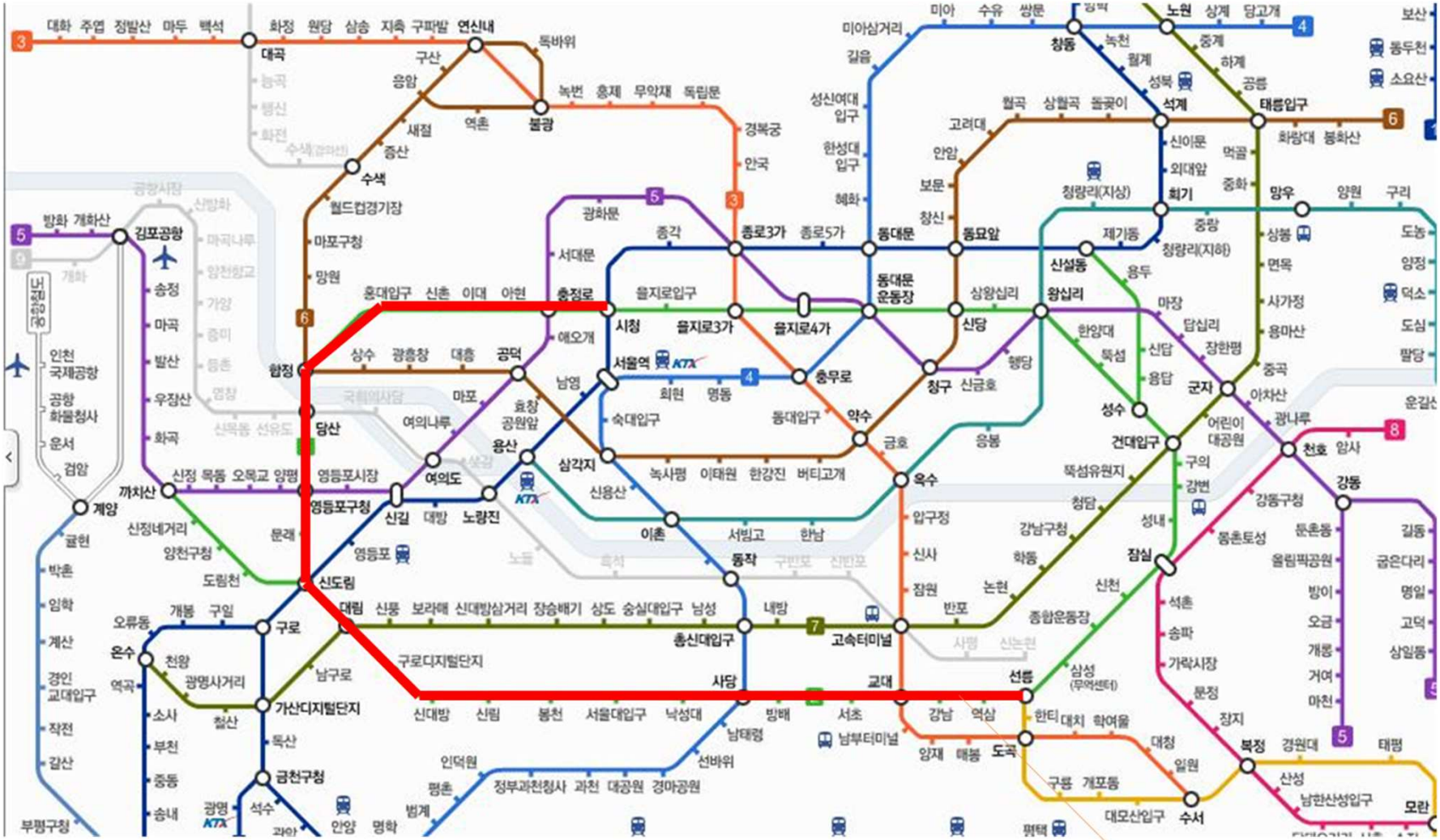
Routing

- I'm at 선릉 station. I want to go to 시청 station.


시청
선릉
종로3가
을지로3가
을지로4가
충무로
동대문
동대문운동장
신당
왕십리
성수
건대입구
잠실
강남
교대
서초
사당
방배
총신대입구
고속터미널
신도림
구로
가산디지털단지
금천구청
여의도
용산
삼각지
이촌
동작
노량진
신길
영등포구청
당산
합정
홍대입구
공덕
서울역
숙대입구
동대입구
약수
금호
옥수
압구정
신사
잠원
반포
논현
학동
강남구청
청담
뚝섬유원지
구의
강변
성내
천호
강동
길동
둔촌동
올림픽공원
방이
오금
개롱
거여
마천
명일
고덕
상일동
굽은다리
강동구청
몽촌토성
석촌
송파
가락시장
문정
장지
복정
경원대
태평
산성
남한산성입구
모란
수서
대모산입구
구룡
개포동
대치
학여울
도곡
양재
매봉
남부터미널
삼성(무역센터)
종합운동장
신천

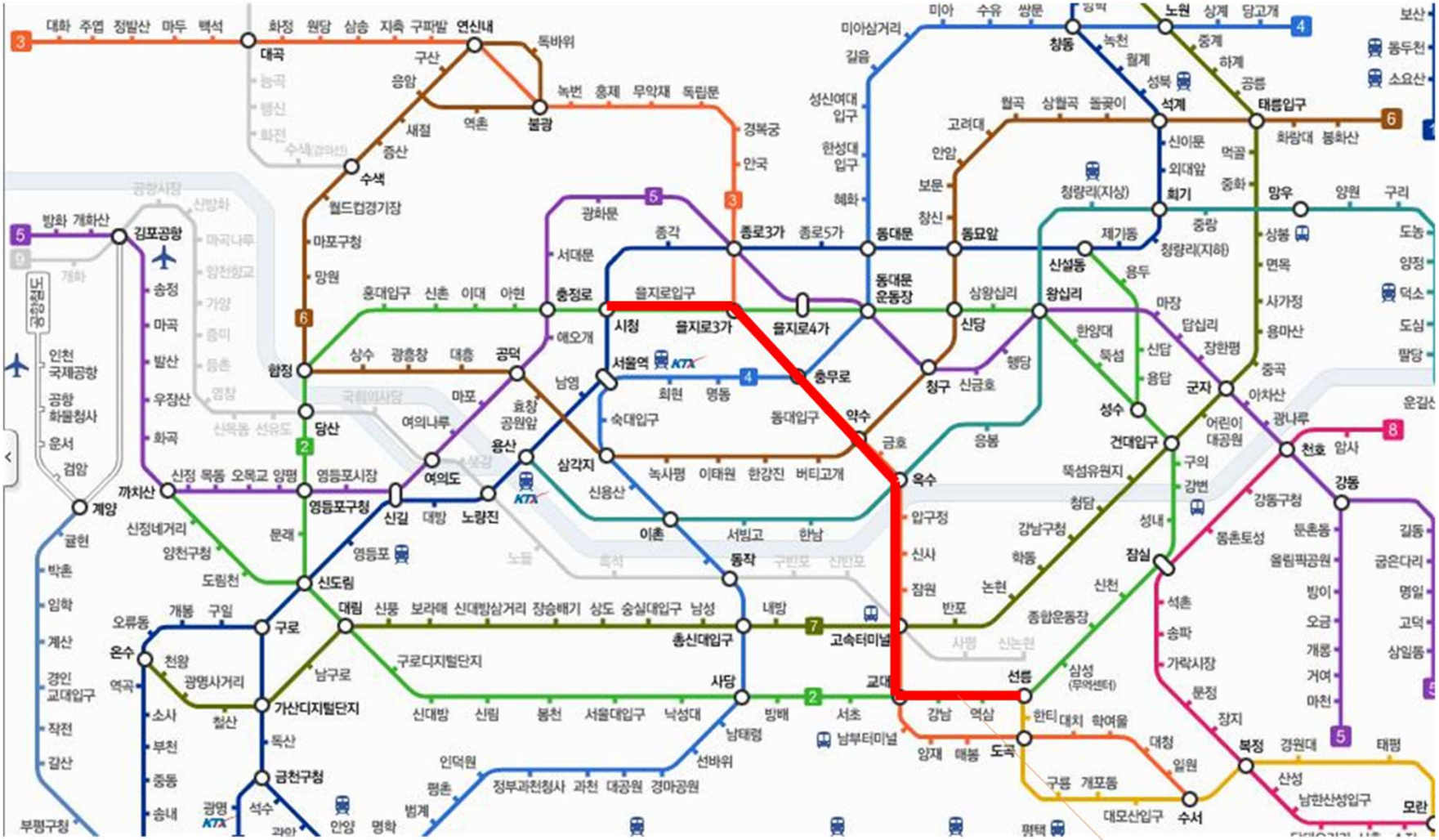
Routing

- 선릉 → 시청



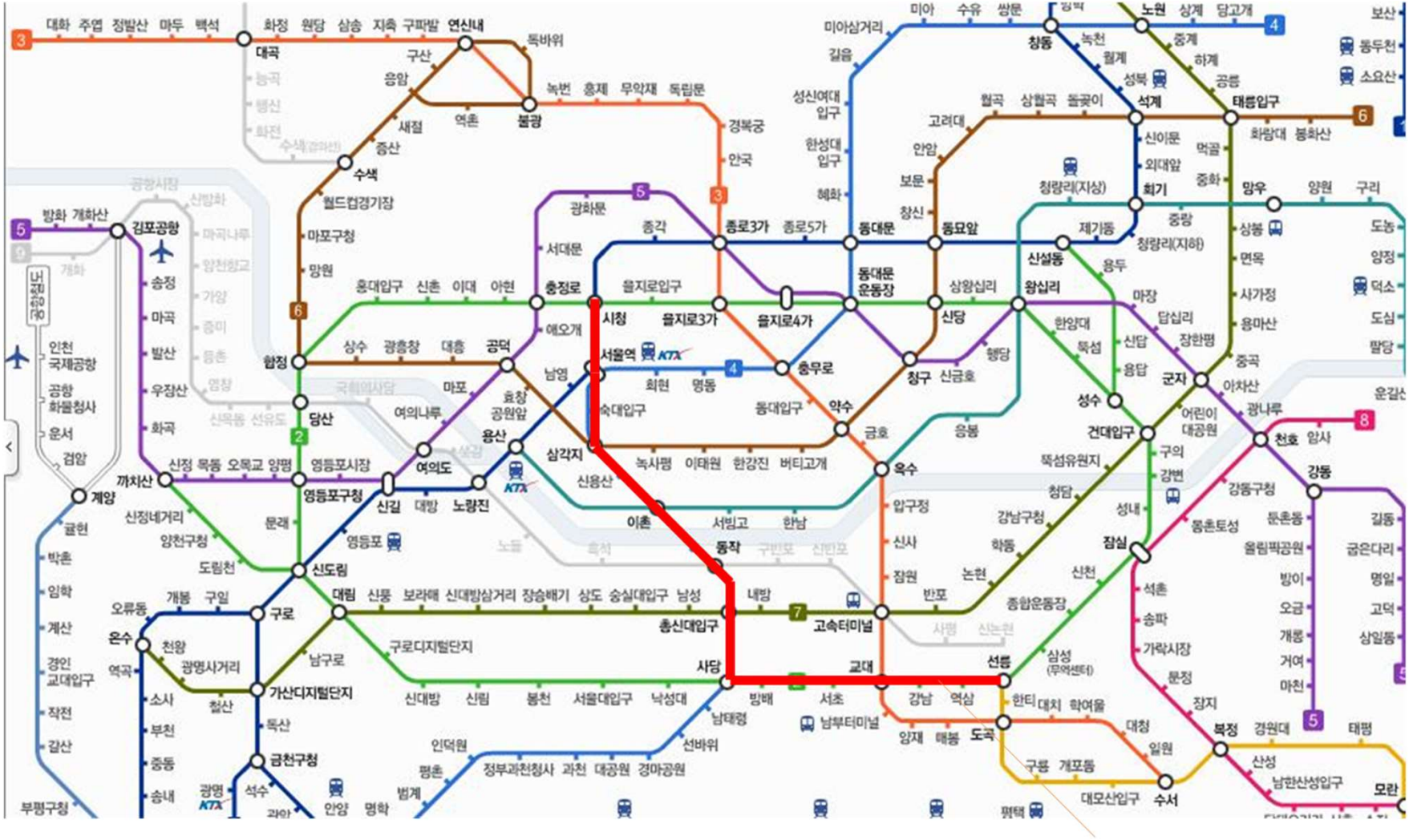
Routing

- 선릉 → 교대 → 을지로3가 → 시청


시청
을지로3가
을지로4가
종로3가
종로5가
을지로입구
충무로
동대문
동대문운동장
동묘앞
신당
상왕십리
왕십리
약수
금호
옥수
압구정
신사
잠원
반포
고속터미널
교대
서초
남부터미널
강남
역삼
선릉
삼성(무역센터)
한티
대치
학여울
도곡
매봉
양재
구룡
개포동
대모산입구
수서
경복궁
안국
광화문
서대문
충정로
경희궁
아현
이대
신촌
홍대입구
공덕
효창공원앞
용산
삼각지
숙대입구
서울역
회현
명동
남영
녹사평
이태원
한강진
버티고개
이촌
서빙고
한남
동작
총신대입구(사당)
사당
내방
신림
봉천
서울대입구
낙성대
남태령
선바위
인덕원
정부과천청사
과천
대공원
경마공원
범계
평촌
안양
명학
금천구청
석수
광명
가산디지털단지
독산
구로디지털단지
남구로
구로
대림
신도림
문래
영등포구청
영등포시장
영등포
신길
대방
노량진
여의도
여의나루
마포
당산
합정
홍대입구
상수
광흥창
대흥
공덕
마포구청
월드컵경기장
수색
증산
새절
응암
역촌
불광
연신내
구산
녹번
홍제
무악재
독립문
독바위
대곡
화정
원당
삼송
지축
구파발
김포공항
송정
마곡
발산
우장산
화곡
까치산
신정
목동
오목교
양평
계양
귤현
박촌
임학
계산
경인교대입구
작전
갈산
부평구청
온수
천왕
광명사거리
철산
소사
부천
중동
송내
역곡
개봉
구일
오류동
방화
개화산
개화
공항철도
인천국제공항
공항화물청사
운서
검암
신정네거리
양천구청
도림천
신도림
창동
녹천
월계
성북
석계
태릉입구
노원
상계
당고계
중계
하계
공릉
화랑대
봉화산
먹골
중화
망우
양원
구리
도농
양정
덕소
도심
팔당
보산
동두천
소요산
미아삼거리
미아
수유
쌍문
길음
성신여대입구
한성대입구
혜화
고려대
안암
보문
창신
신이문
외대앞
회기
청량리(지상)
청량리(지하)
중랑
제기동
신설동
용두
답십리
장한평
군자
아차산
광나루
천호
암사
강동
길동
굽은다리
명일
고덕
상일동
둔촌동
올림픽공원
방이
오금
개롱
거여
마천
강동구청
몽촌토성
잠실
석촌
송파
가락시장
문정
장지
복정
경원대
태평
산성
남한산성입구
모란
성내
천호
광나루
어린이대공원
건대입구
구의
강변
뚝섬유원지
청담
성수
뚝섬
한양대
신답
용답
상월곡
돌곶이
월곡
석계
신설동
용두
마장
사가정
용마산
중곡
군자
장한평
답십리
왕십리
상왕십리
신당
청구
신금호
행당
응봉
동대문운동장
동묘앞
창신
보문
안암
고려대
혜화
한성대입구
성신여대입구
길음
미아삼거리
미아
수유
쌍문
창동
노원
상계
당고개
중계
하계
공릉
태릉입구
화랑대
봉화산
먹골
중화
망우
양원
구리
도농
양정
덕소
도심
팔당
상봉
면목
사가정
용마산
중곡

Routing

- 선릉 → 사당 → 서울역 → 시청

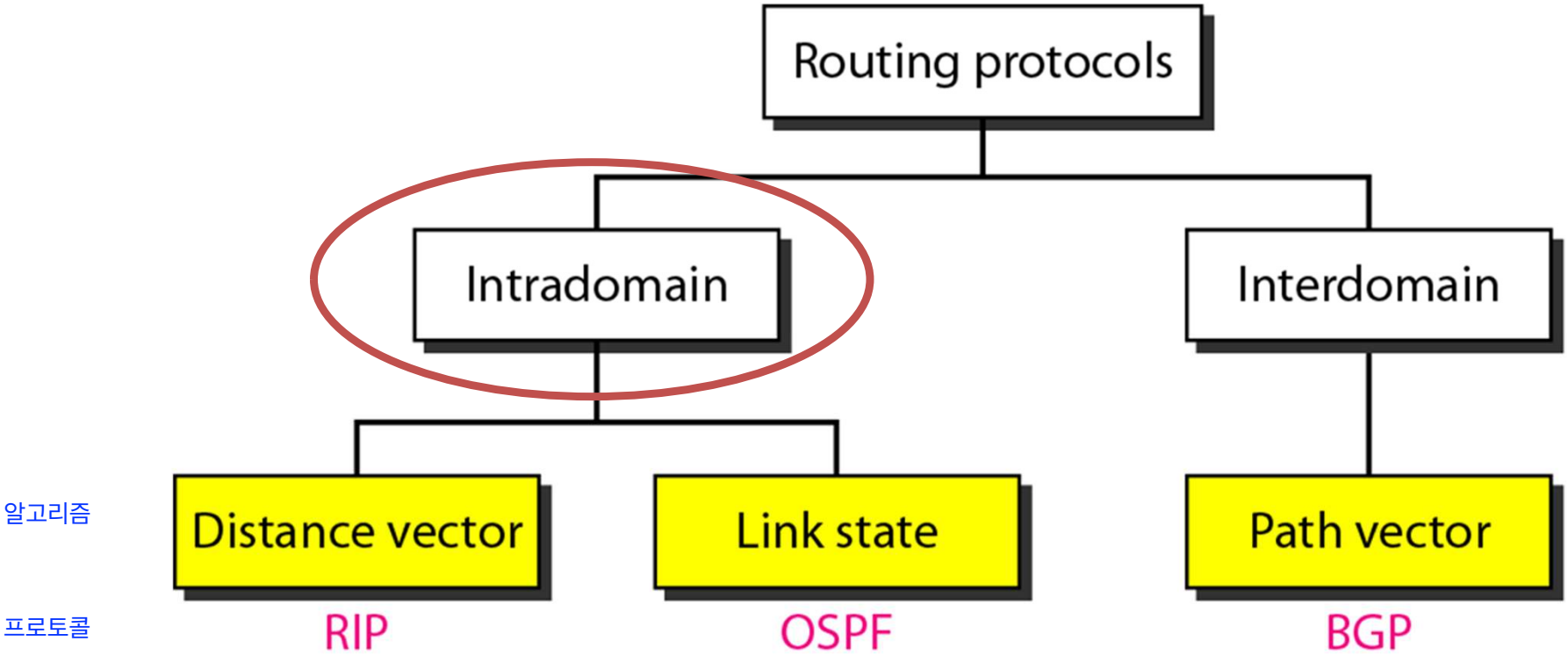

시청
을지로3가
을지로4가
종로3가
종로5가
충정로
서울역
충무로
동대문
동대문운동장
동묘앞
신당
상왕십리
왕십리
약수
금호
옥수
창구
신금호
행당
응봉
성수
건대입구
뚝섬유원지
청담
강남구청
학동
논현
반포
잠실
종합운동장
삼성(무역센터)
한티
대치
학여울
도곡
대청
일원
수서
대모산입구
구룡
개포동
교대
서초
강남
역삼
선릉
남부터미널
양재
매봉
고속터미널
사평
신논현
총신대입구
내방
사당
방배
남태령
선바위
낙성대
서울대입구
봉천
신림
신대방
구로디지털단지
남구로
가산디지털단지
독산
금천구청
광명
석수
안양
명학
범계
평촌
인덕원
정부과천청사
과천
대공원
경마공원
대림
신풍
보라매
신대방삼거리
장승배기
상도
숭실대입구
남성
구로
신도림
문래
영등포구청
영등포
영등포시장
신길
대방
노량진
여의도
샛강
여의나루
마포
공덕
효창공원앞
용산
삼각지
숙대입구
녹사평
이태원
한강진
버티고개
이촌
서빙고
한남
동작
구반포
신반포
노들
흑석
합정
당산
홍대입구
신촌
이대
아현
애오개
서대문
광화문
종각
을지로입구
시청
을지로3가
을지로4가
충무로
동대입구
약수
금호
옥수
압구정
신사
잠원
반포
논현
신천
석촌
송파
가락시장
문정
장지
북정
경원대
태평
산성
남한산성입구
모란
강동
천호
암사
길동
굽은다리
명일
고덕
상일동
둔촌동
올림픽공원
방이
오금
개롱
거여
마천
강동구청
몽촌토성
성내
광나루
아차산
어린이대공원
구의
강변
군자
중곡
용마산
사가정
답십리
장한평
청량리(지하)
회기
중랑
상봉
면목
망우
양원
구리
도농
양정
덕소
도심
팔당
운길산
동두천
소요산
보산
창동
녹천
월계
성북
석계
태릉입구
화랑대
봉화산
신이문
외대앞
먹골
중화
하계
공릉
노원
상계
당고개
미아
수유
쌍문
미아삼거리
길음
성신여대입구
한성대입구
혜화
보문
안암
고려대
월곡
상월곡
돌곶이
창신
제기동
용두
신설동
청량리(지하)
한양대
뚝섬
신답
용답
성수
건대입구

Routing

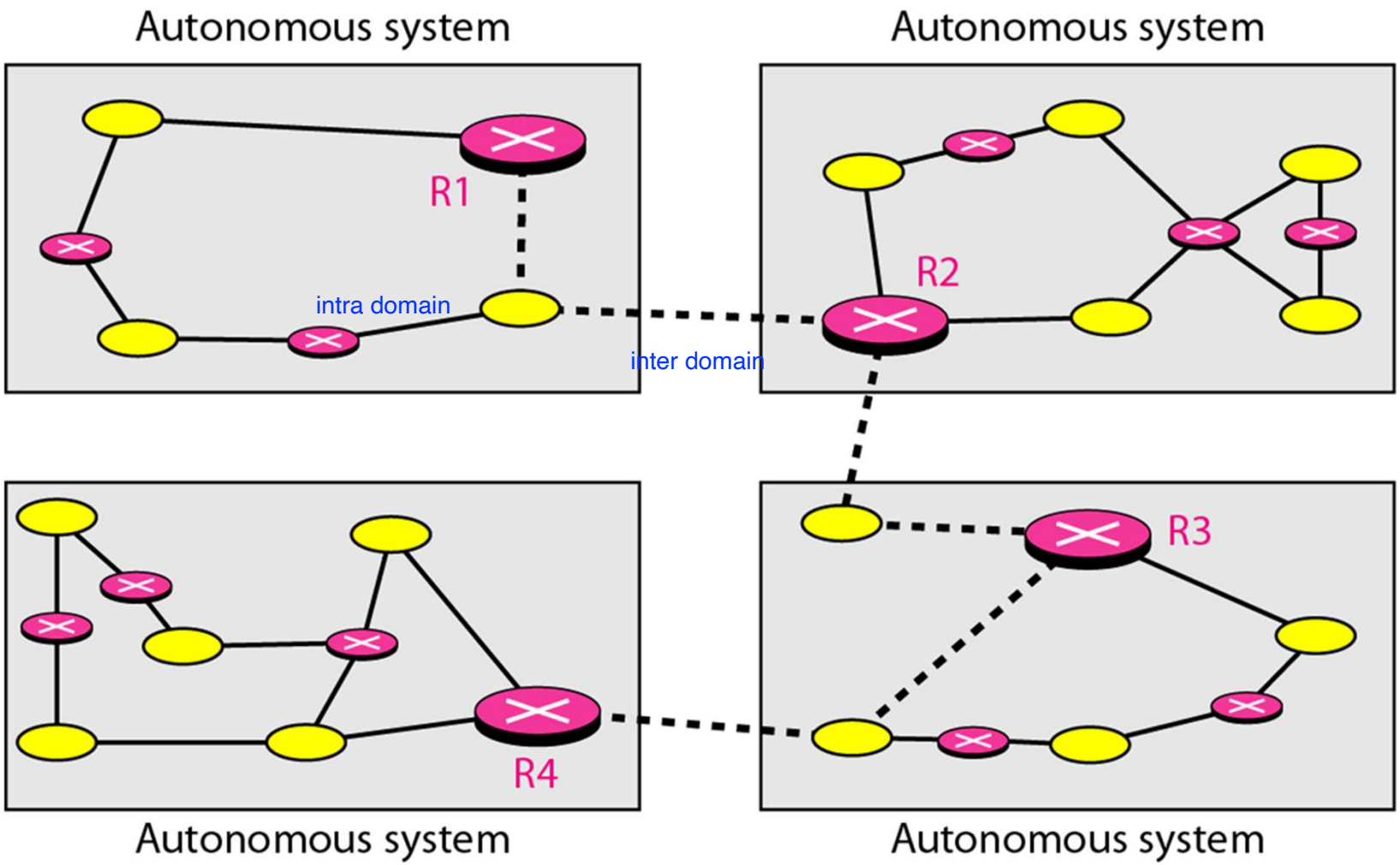
- How should we decide our path?
  - The fastest
  - The cheapest
  - Possibility of getting a seat
  - Shortest walking distance
  - etc.

- If we choose “the fastest path”, what should we consider?
  - Number of stations
  - Time duration between stations
  - Time for transfer
  - ...

Routing protocols taxonomy



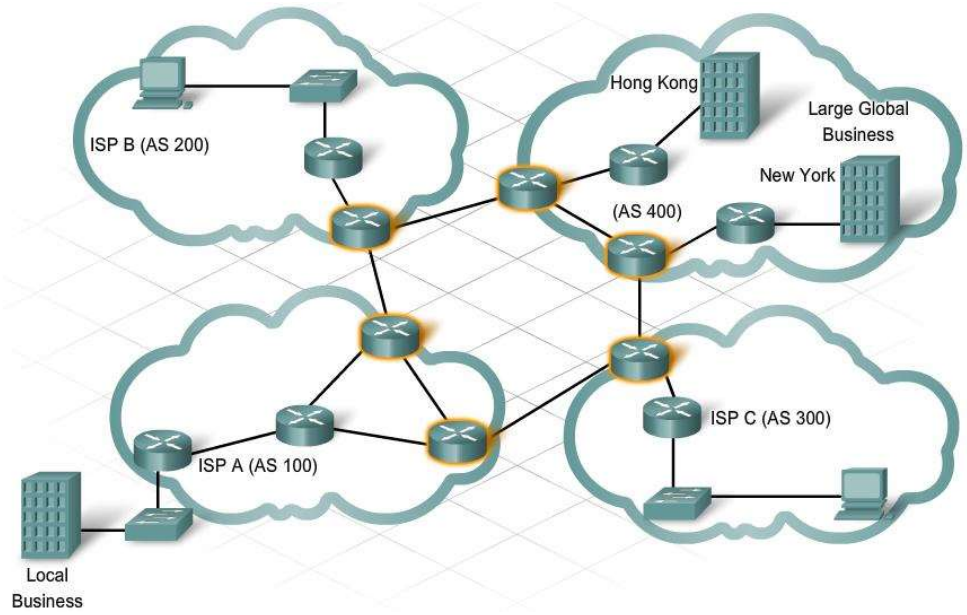
Intra-domain vs. Inter-domain



Autonomous Systems (AS)

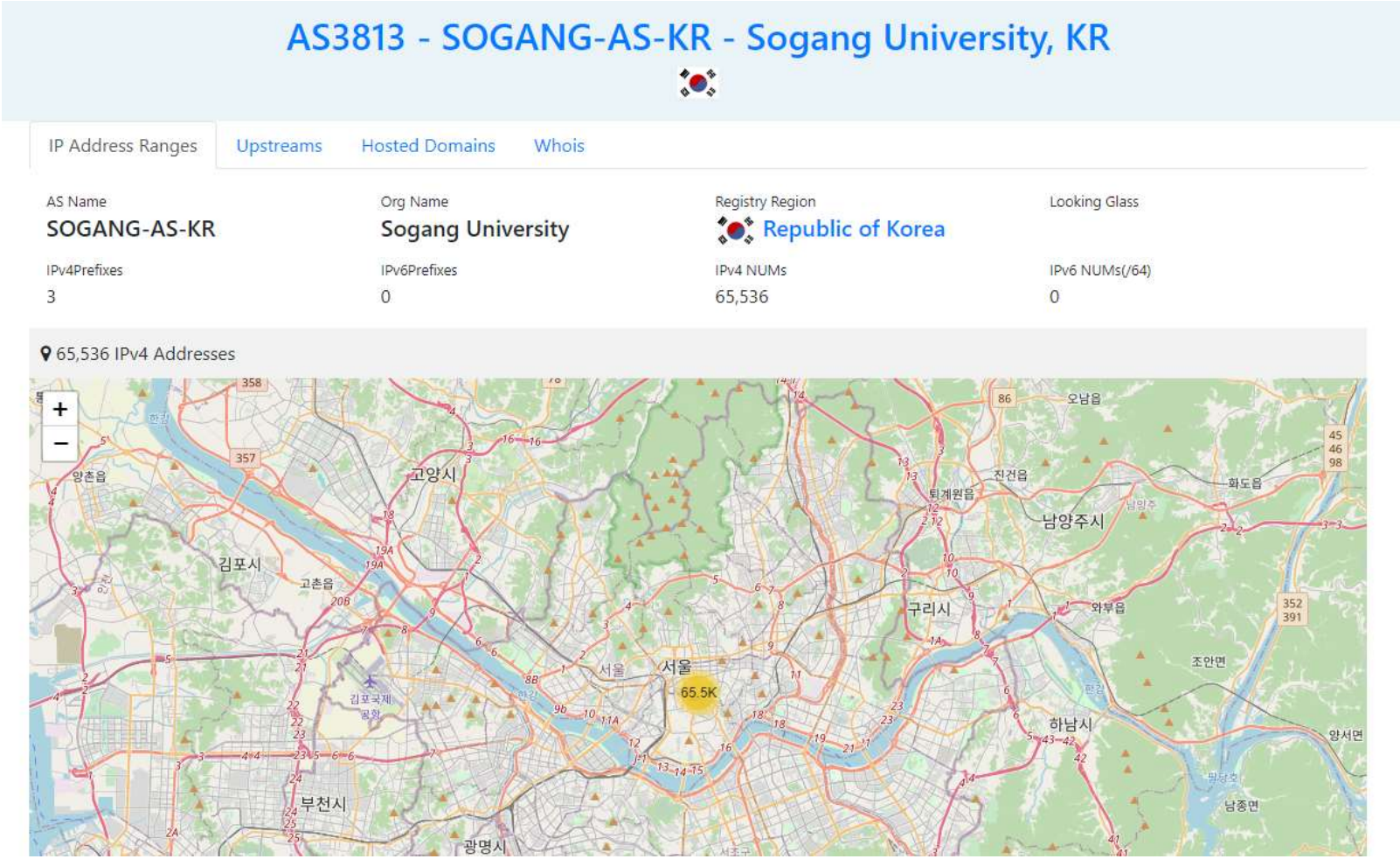
- Internet: Network of Networks
- Autonomous System: a large network or a group of networks that has a unified routing policy
- Controlled by ISP or an organization
- Each AS has a unique ASN (AS number)
- 100,424 AS in the world (as of April 2021)
- 1,036 AS in Korea

하나의 AS는 intradomain 망이다
다른 AS와의 통신이 interdomain 통신



Autonomous Systems (AS)

- AS3813 - Sogang University



Intra vs. Inter-domain Routing

- Intra-domain (Intra-AS) routing
  - routing among hosts, routers in same AS ("network")
  - all routers in AS must run the same intra-domain protocol
  - routers in different AS can run different intra-domain routing protocol
  - gateway router: at "edge" of its own AS, has link(s) to router(s) in other AS'es gateway router는 intra, inter다 해야함
- Inter-domain (Inter-AS) routing
  - routing among AS'es
  - gateways perform inter-domain routing (as well as intra-domain routing)

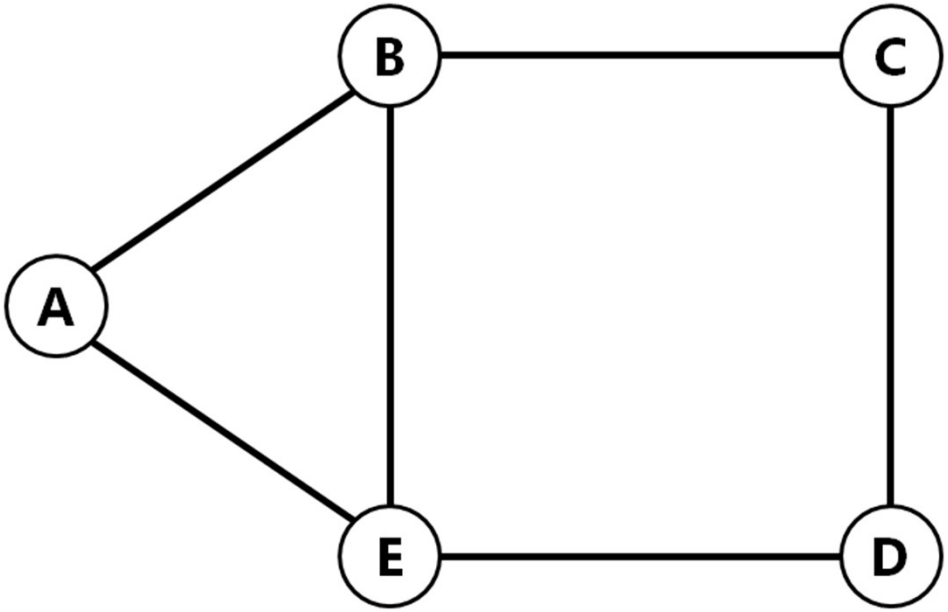
Routing Algorithms & Protocols

Distance vector routing

- Record the next node for each destination
- Exchange tables with neighbors, and update my table using exchanged information
- Information stored at routing tables
  - (destination, distance, next node) → distance vector

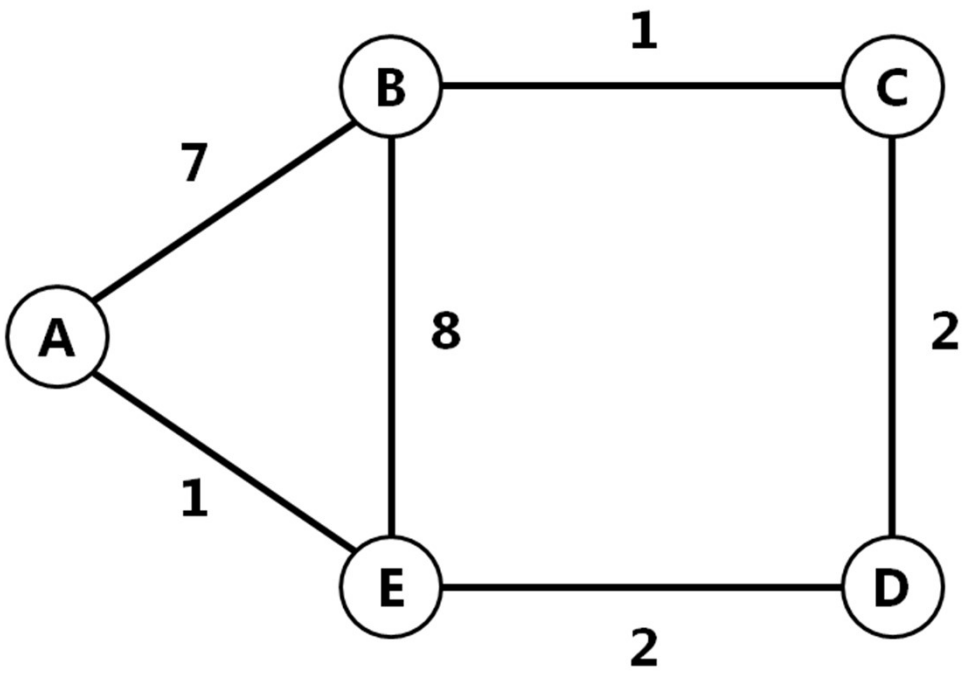
Distance vector routing

- What is the <u>shortest path</u> from A to C?



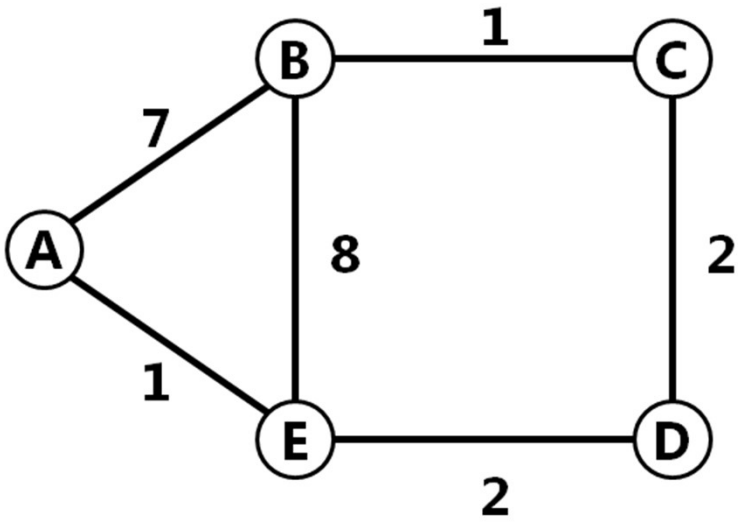
Distance vector routing

- First we need to know the “distance” of each link
  - Distance can be physical distance, but it can also mean other things (e.g. cost)



Distance vector routing

- In the routing table, a "distance vector" is stored for each destination
  - distance vector = (destination, distance, next node)



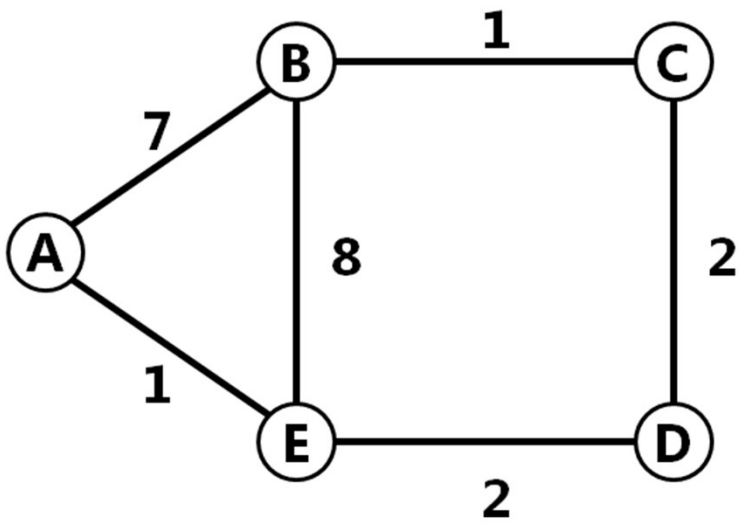
A의 Table		
목적지	거리	다음
A	0	-
B	7	B
C	-	-
D	-	-
E	1	E

trigger based - table이 업데이트 되면 업데이트 된 내용으로 고쳐서 뿌림
timer based - 일정 시간동안 특정 table을 보내고, 시간이 지나면 가장 최근 table을 뿌림

Distance vector routing

- Initially, we know the paths to the direct neighbors
  - A to B: distance 7: next node B
  - A to E: distance 1: next node E

- Store this information in the routing table

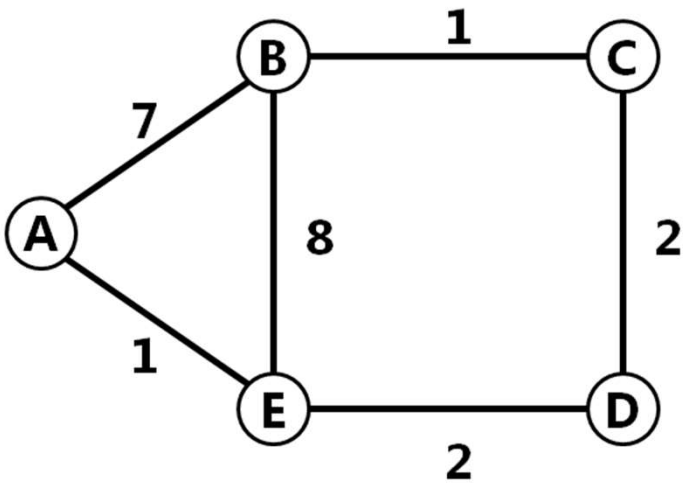


A의 Table		
목적지	거리	다음
A	0	-
B	7	B
C	-	-
D	-	-
E	1	E

Distance vector routing

- Initial state of routing tables for all routers

A와 인접한 B,E가 자기와 연결된 라우터들을 알려줌



A의 Table		
목적지	거리	다음
A	0	-
B	7	B
C	-	-
D	-	-
E	1	E

B의 Table		
목적지	거리	다음
A	7	A
B	0	-
C	1	C
D	-	-
E	8	E

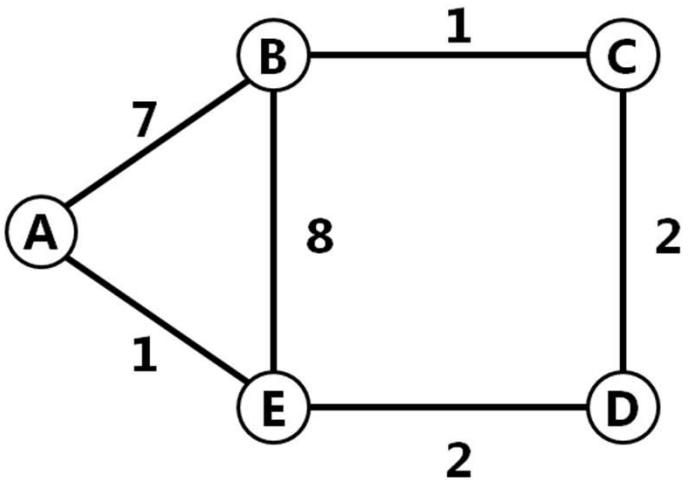
C의 Table		
목적지	거리	다음
A	-	-
B	1	B
C	0	-
D	2	D
E	-	-

D의 Table		
목적지	거리	다음
A	-	-
B	-	-
C	2	C
D	0	-
E	2	E

E의 Table		
목적지	거리	다음
A	1	A
B	8	B
C	-	-
D	2	D
E	0	-

Distance vector routing

- Now, <u>nodes exchange tables with direct neighbors</u>
  - Each node sends its routing table to all its direct neighbors



A의 Table		
목적지	거리	다음
A	0	-
B	7	B
C	-	-
D	-	-
E	1	E

B의 Table		
목적지	거리	다음
A	7	A
B	0	-
C	1	C
D	-	-
E	8	E

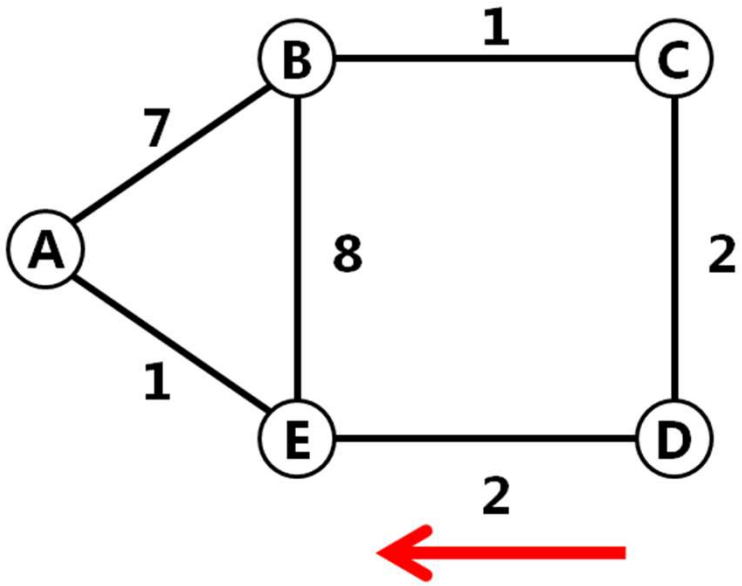
C의 Table		
목적지	거리	다음
A	-	-
B	1	B
C	0	-
D	2	D
E	-	-

D의 Table		
목적지	거리	다음
A	-	-
B	-	-
C	2	C
D	0	-
E	2	E

E의 Table		
목적지	거리	다음
A	1	A
B	8	B
C	-	-
D	2	D
E	0	-

Distance vector routing

- Node D sends its table to node E
  - Information sent: (destination, distance)
  - Next node should be the sending node itself



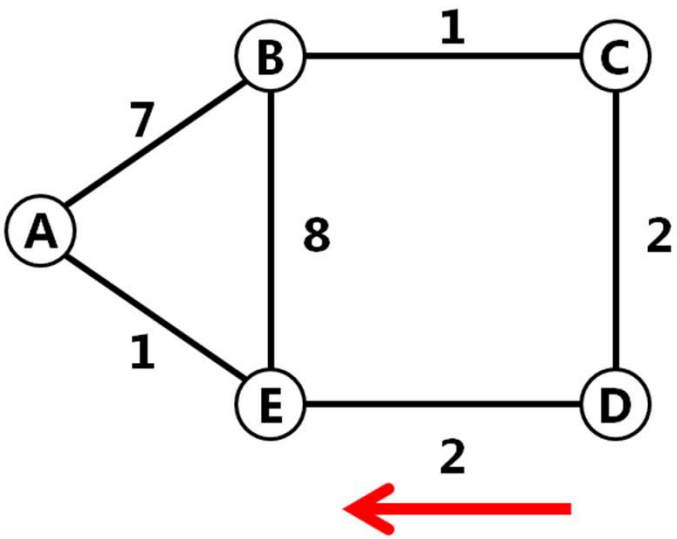
C	2
D	0
E	2

D의 Table		
목적지	거리	다음
A	-	-
B	-	-
C	2	C
D	0	-
E	2	E

E의 Table		
목적지	거리	다음
A	1	A
B	8	B
C	-	-
D	2	D
E	0	-

Distance vector routing

- On receiving table from D, node E updates its routing table
  - E didn't know the path to C, but now it knows that it can reach C through node D, and the distance will be 4.
    - 2 (E→D) + 2 (D→C)



C	2
D	0
E	2

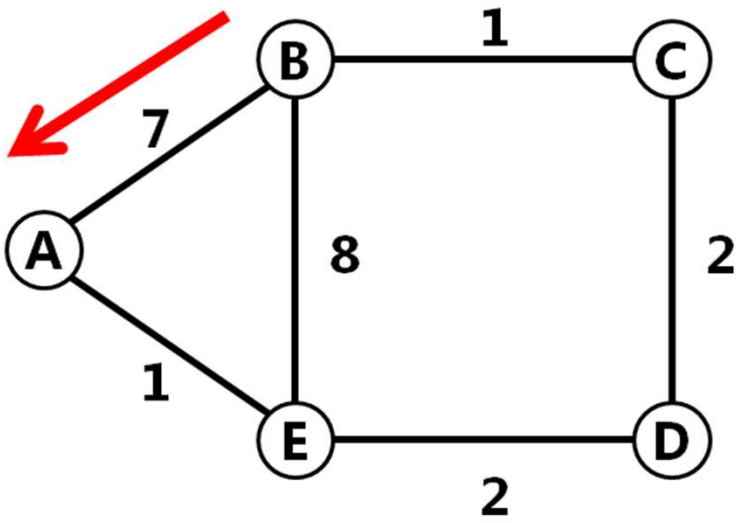
D의 Table		
목적지	거리	다음
A	-	-
B	-	-
C	2	C
D	0	-
E	2	E

E의 Table		
목적지	거리	다음
A	1	A
B	8	B
C	4	D
D	2	D
E	0	-

Distance vector routing

- Node B sends its table to node A

A	7
B	0
C	1
E	8



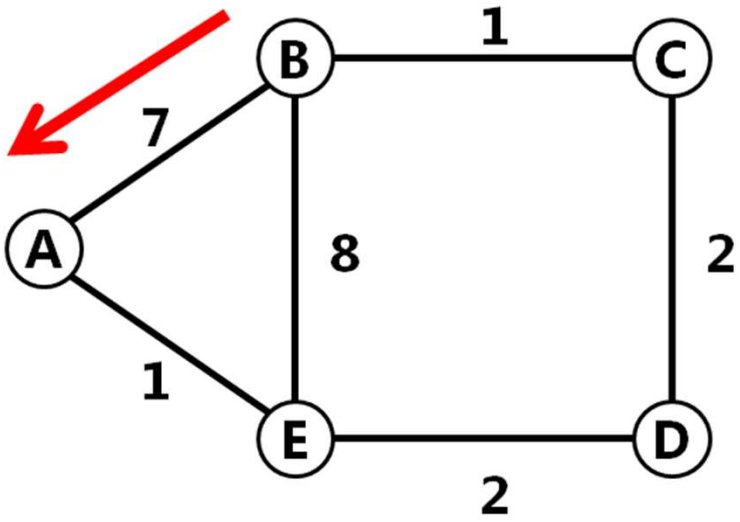
A의 Table		
목적지	거리	다음
A	0	-
B	7	B
C	-	-
D	-	-
E	1	E

B의 Table		
목적지	거리	다음
A	7	A
B	0	-
C	1	C
D	-	-
E	8	E

Distance vector routing

- A updates its routing table
  - A knows it can reach C through B, with distance 8.

A	7
B	0
C	1
E	8

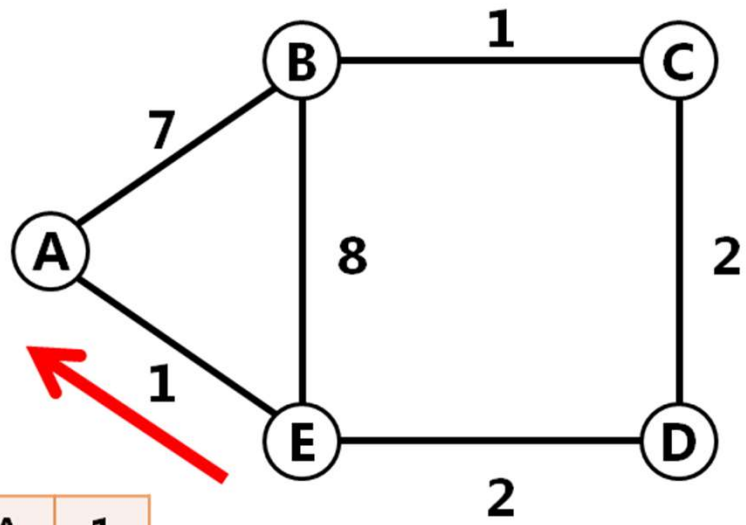


A의 Table		
목적지	거리	다음
A	0	-
B	7	B
C	8	B
D	-	-
E	1	E

B의 Table		
목적지	거리	다음
A	7	A
B	0	-
C	1	C
D	-	-
E	8	E

Distance vector routing

- Node E sends its table to node A



A	1
B	8
C	4
D	2
E	0

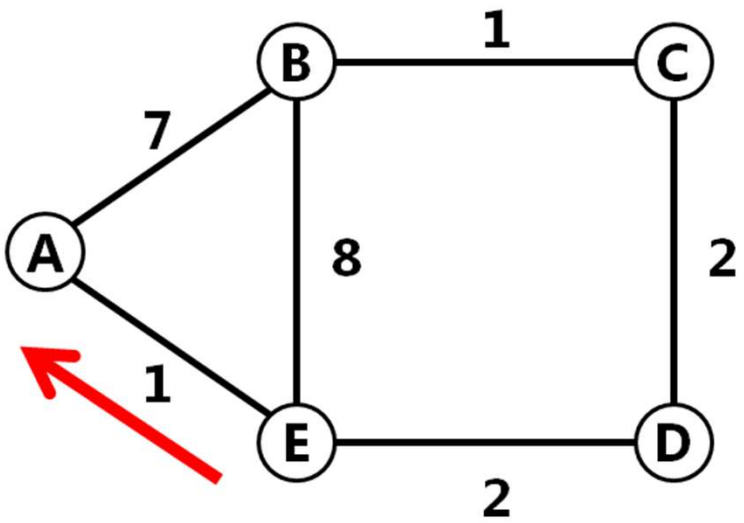
A의 Table		
목적지	거리	다음
A	0	-
B	7	B
C	8	B
D	-	-
E	1	E

E의 Table		
목적지	거리	다음
A	1	A
B	8	B
C	4	D
D	2	D
E	0	-

E를 통해서 C로 가면 1 + 4 -> 8보다 적음

Distance vector routing

- Node A updates its table
  - Since distance from E to C is 4, E can reach C with distance 5.
  - Now, node A knows a path to C with distance 8.
  - Since <u>the path through E is shorter</u>, node A updates its path to C to go through E.
    - If the current path is shorter or equal, it is not updated.



A의 Table		
목적지	거리	다음
A	0	-
B	7	B
C	5	E
D	3	E
E	1	E

E의 Table		
목적지	거리	다음
A	1	A
B	8	B
C	4	D
D	2	D
E	0	-

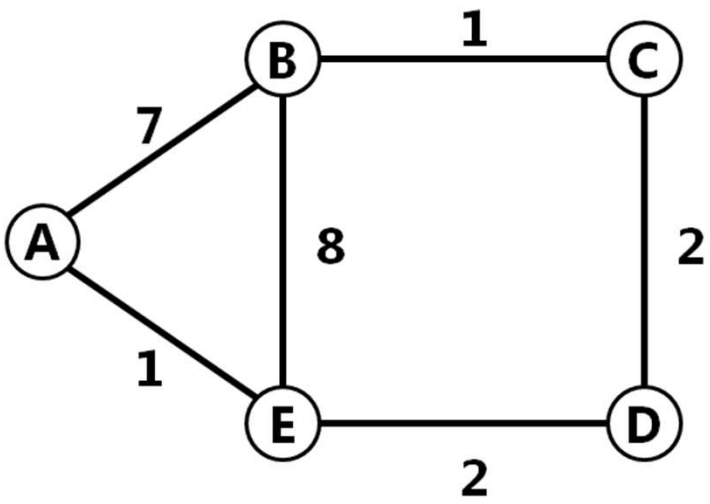
Distance vector routing

- All nodes exchange tables
  - Send my table to all my neighbors
  - If my table is updated, send my table again
  - Also, nodes send its table periodically
    - node state or link distance might have changed
- After all exchanges have been finished…

Distance vector routing

더이상 업데이트가 안되는 상황 - 모든 라우터가 모든 dest에 대해서 shortest path를 얻음

- The final state of routing tables



A의 Table		
목적지	거리	다음
A	0	-
B	6	E
C	5	E
D	3	E
E	1	E

B의 Table		
목적지	거리	다음
A	6	C
B	0	-
C	1	C
D	3	C
E	5	C

C의 Table		
목적지	거리	다음
A	5	D
B	1	B
C	0	-
D	2	D
E	4	D

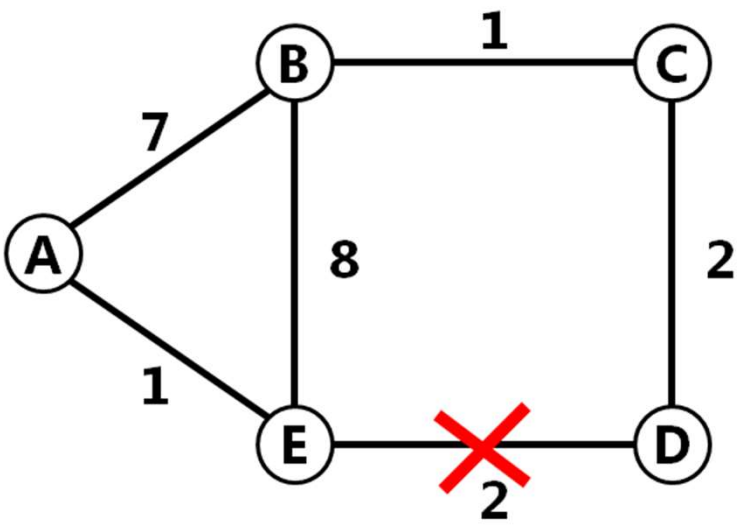
D의 Table		
목적지	거리	다음
A	3	E
B	3	C
C	2	C
D	0	-
E	2	E

E의 Table		
목적지	거리	다음
A	1	A
B	5	D
C	4	D
D	2	D
E	0	-

Distance vector routing

- Suppose the link between D and E breaks
  - D and E detect this event

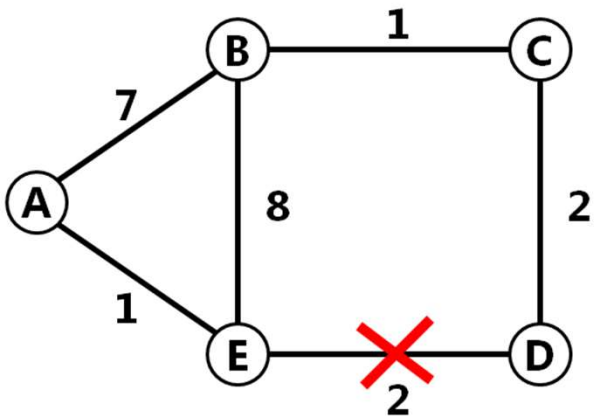
중간에 길이 끊기는 일이 발생



Distance vector routing

길이 끊어진 걸 반영해서 converge할 수 있다

- D and E sends the updated routing table to their neighbors, and nodes update their routing table



A의 Table		
목적지	거리	다음
A	0	-
B	7	B
C	8	B
D	10	B
E	1	E

B의 Table		
목적지	거리	다음
A	7	A
B	0	-
C	1	C
D	3	C
E	8	E

C의 Table		
목적지	거리	다음
A	8	B
B	1	B
C	0	-
D	2	D
E	9	B

D의 Table		
목적지	거리	다음
A	10	C
B	3	C
C	2	C
D	0	-
E	11	C

E의 Table		
목적지	거리	다음
A	1	A
B	8	B
C	9	B
D	11	B
E	0	-

Distance vector routing

- Nodes periodically exchange routing tables with direct neighbors

- If there is a change in the network, the routing table will be updated to adopt the change

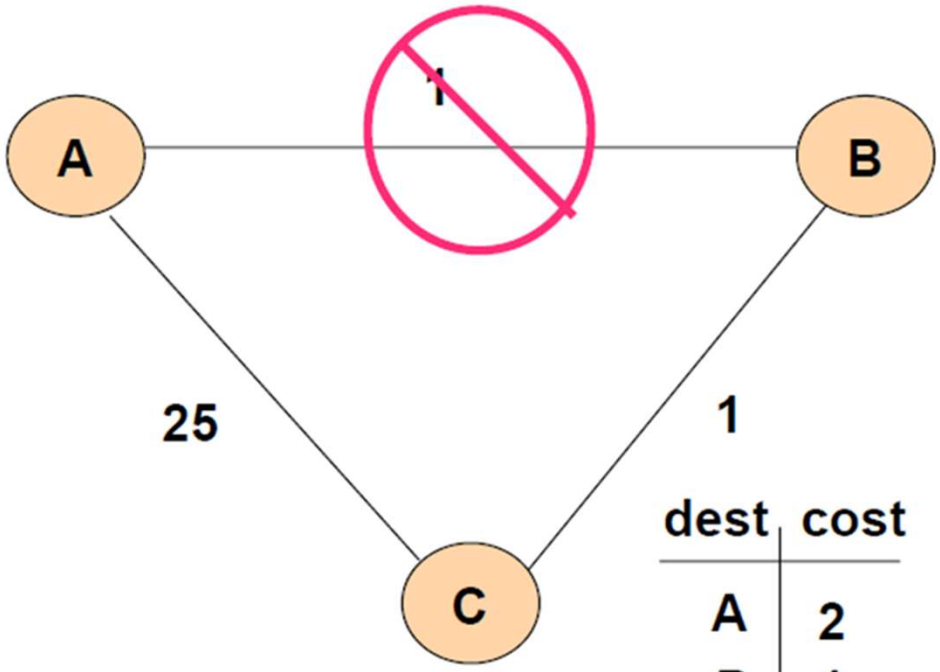
타이머가 지나도 저쪽 라우터에서 응답이 없으면 죽었다고 생각해서 업데이트하고 계속한다

- Each route information has a timer. If the route is not updated for a certain duration, it is invalidated.
  - Suppose A has a route to C via B. If node A does not get periodic updates from B, the route to C will be removed after some time.

Count-to-infinity problem

- A problem in distance vector routing
- Consider the following network. After route table is finalized, the link between A and B breaks

dest	cost
B	1
C	2



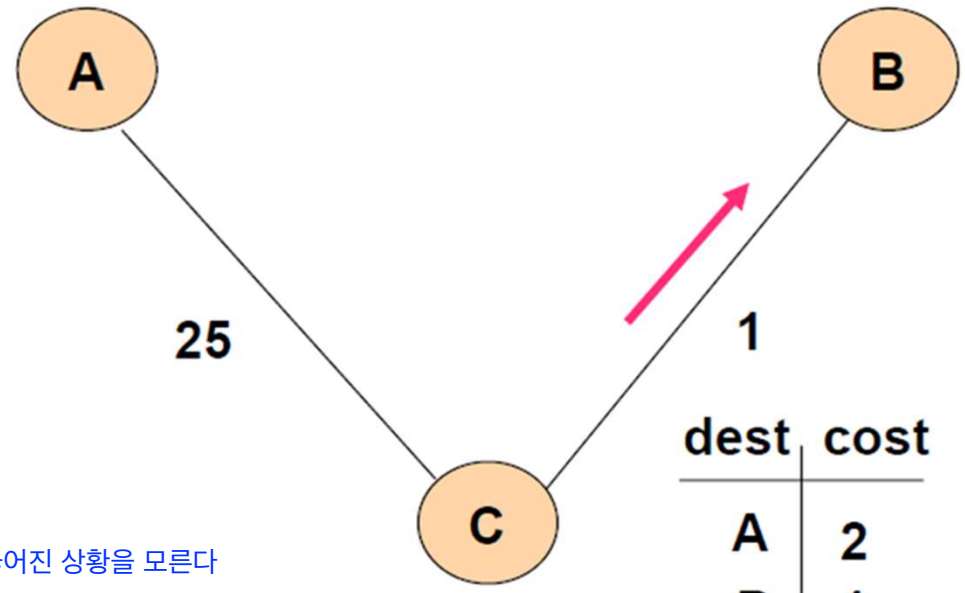
dest	cost
A	1
C	1

dest	cost
A	2
B	1

Count-to-infinity problem

- B does not know how to reach A
- C sends its routing table to B

dest	cost
B	1
C	2



dest	cost
A	~
C	1

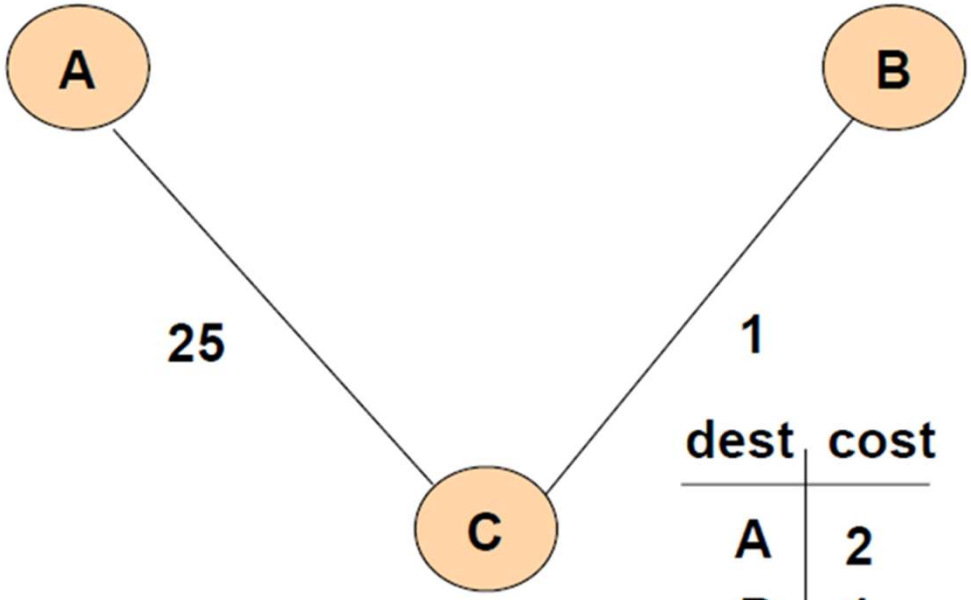
dest	cost
A	2
B	1

c는 a-b가 끊어진 상황을 모른다

Count-to-infinity problem

- B thinks it can reach A via C with distance 3.
  - However, this path does not exist. (B→C→B→A)

dest	cost
B	1
C	2



dest	cost
A	3
C	1

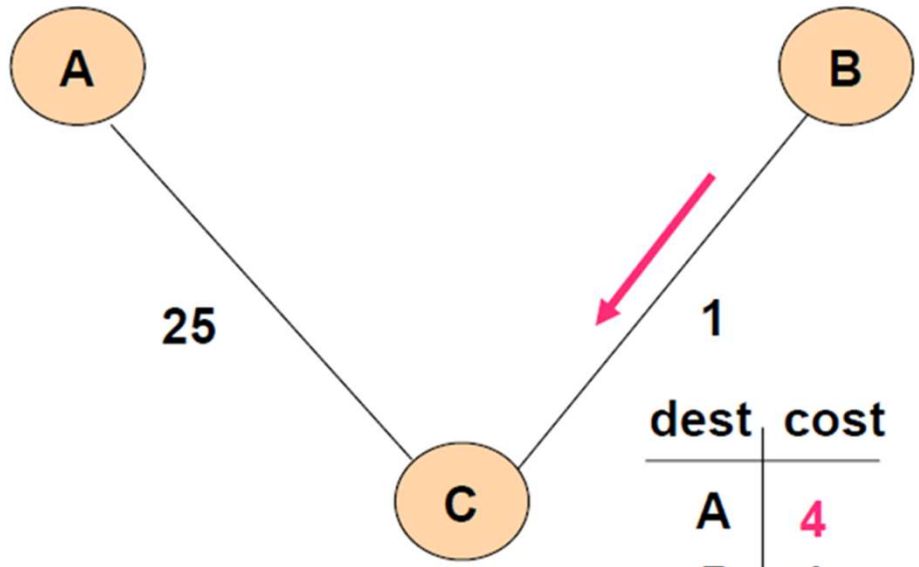
dest	cost
A	2
B	1

Count-to-infinity problem

- Now B sends its table to C and C updates its table
  - The distance to C has changed.

c에서 원래 가던 길이 b->a 2었다
업데이트가 된 b가 c에게 자길 거쳐서 가면 3이다. c는 업데이트를 해서 3+1 , a로 가는 길을 4로 업데이트

dest	cost
B	1
C	2



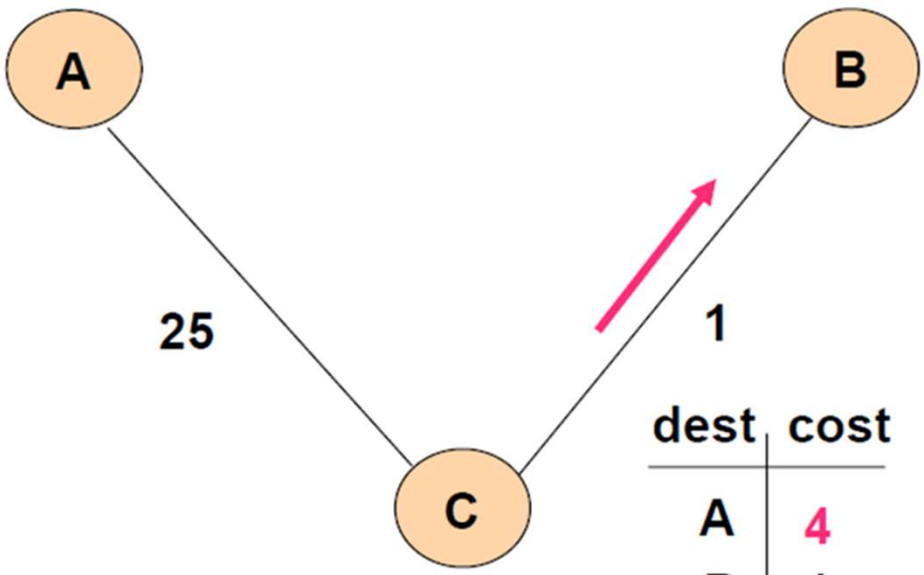
dest	cost
A	3
C	1

dest	cost
A	4
B	1

Count-to-infinity problem

- C sends its table to B
- B updates its table

dest	cost
B	1
C	2



dest	cost
A	5
C	1

dest	cost
A	4
B	1

Count-to-infinity problem

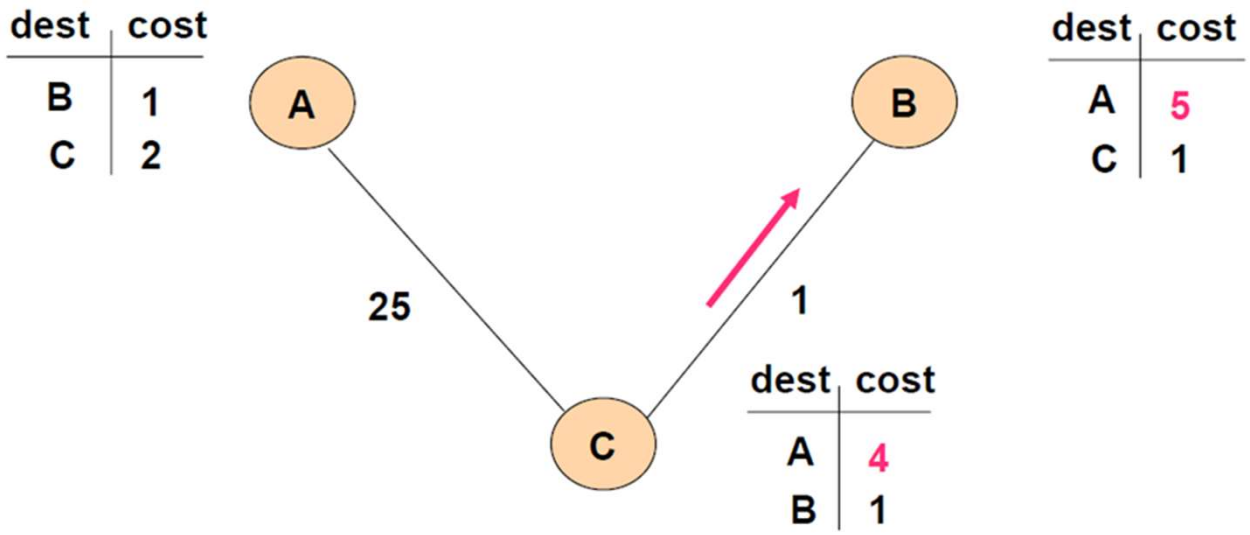
- C will eventually change its path to A (direct path), but too much time is needed until C changes its path.

- Also, too much route information is exchanged
  - waste of network resource

Count-to-infinity problem

문제는 처음 c가 알려준 길은 b를 통해서 a로 간 것.
근데 길의 경로를 알려주지 않음.
나를 통해서 가는 길을 취하면 안됨.

- Cause of the problem
  - When B receives route information from C, B does not know whether the route <u>goes through itself</u>.
  - This problem arises because distance vector only records the next node (not the whole path)



Count-to-infinity problem

- Solutions to problem
  - split horizon
  - split horizon with poison reverse
  - path vector routing

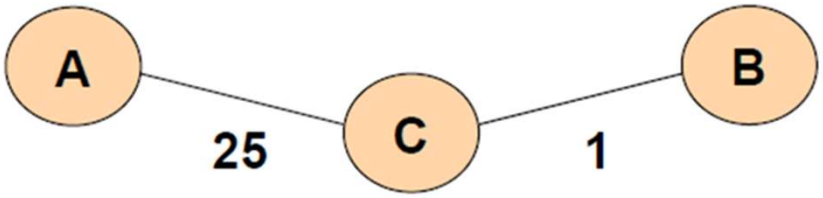
Count-to-infinity problem

- Split horizon
  - When C sends its route table to B, do not include routes that go through B

c가 b로 table을 보낼 때
b가 next stop인 entry는 빼고 보낸다

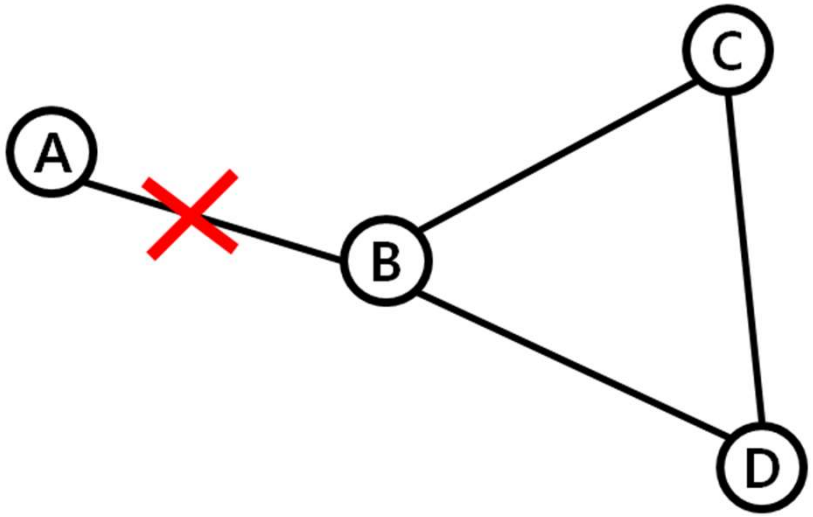
- Split horizon with poison reverse
  - When C sends its route table to B, include routes that go through B, but make their distance to be infinity

b가 next stop인 entry를 보내는데 distance를 무한으로
잘못된 길을 없애버림



Count-to-infinity problem

- Split horizon with poison reverse: limitation
  - 3-hop loop cannot be avoided



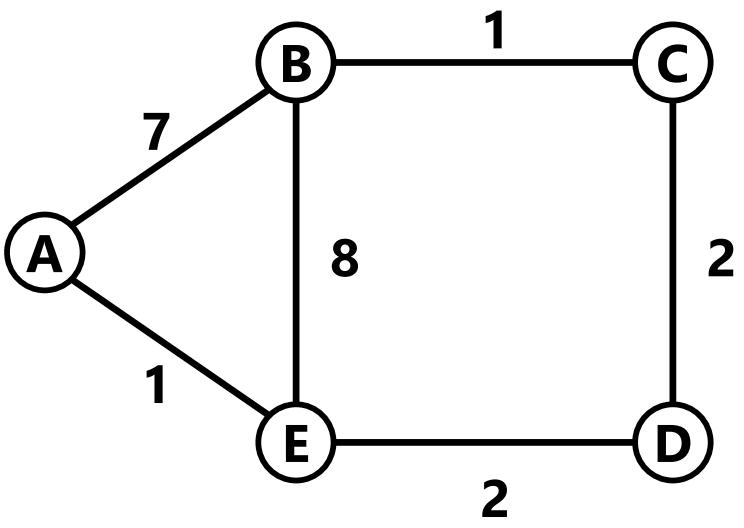
Count-to-infinity problem

- Path vector routing
  - record the whole path (not just the next node)
  - loop can be detected as soon as the table is received
  - used in BGP (Border Gateway Protocol)

Path vector routing

- record distance and the whole path

next stop 뿐만 아니라 완전한 경로를 적는다
원천적인 문제를 해결할 수 있지만 거치는 모든 라우터를 기억하는 게 쉽지 않음
interdomain bgp에서는 이걸 채용하고 있음



A의 Table		
목적지	거리	경로
A	0	-
B	6	E D C B
C	5	E D C
D	3	E D
E	1	E

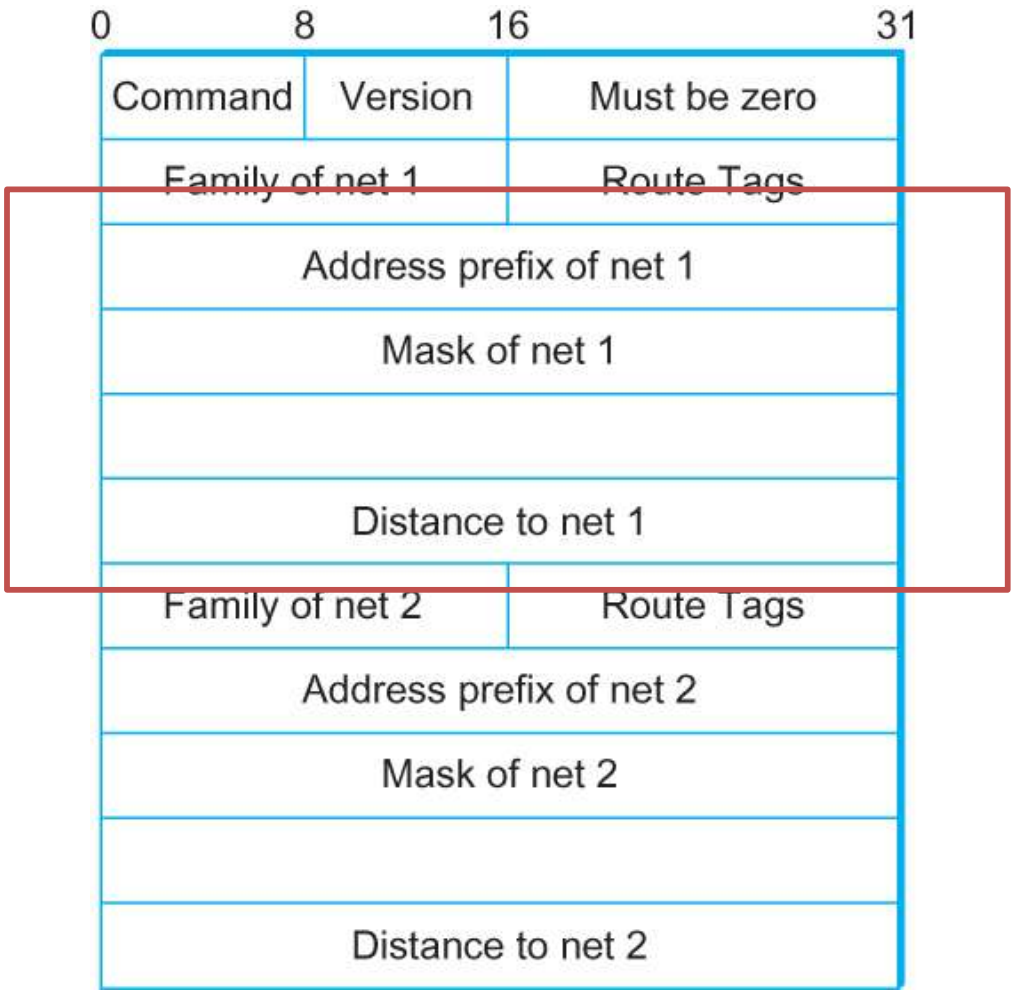
Routing Information Protocol (RIP)

- A widely used routing protocol based on distance vector routing

- Each router sends advertisement messages
  - Every 30 seconds, or
  - If routing table is updated

Routing Information Protocol (RIP)

- RIPv2 Packet Format



Link-state routing

- Send my "link state" information to <u>all nodes in the network</u>
  - Link state packet (LSP)

네트워크 모든 애들한테 줌

- Link state packet
  - ID of the sender
  - (ID, distance) for each direct neighbor of the sender
  - Sequence number (to find out if it is the newest info)

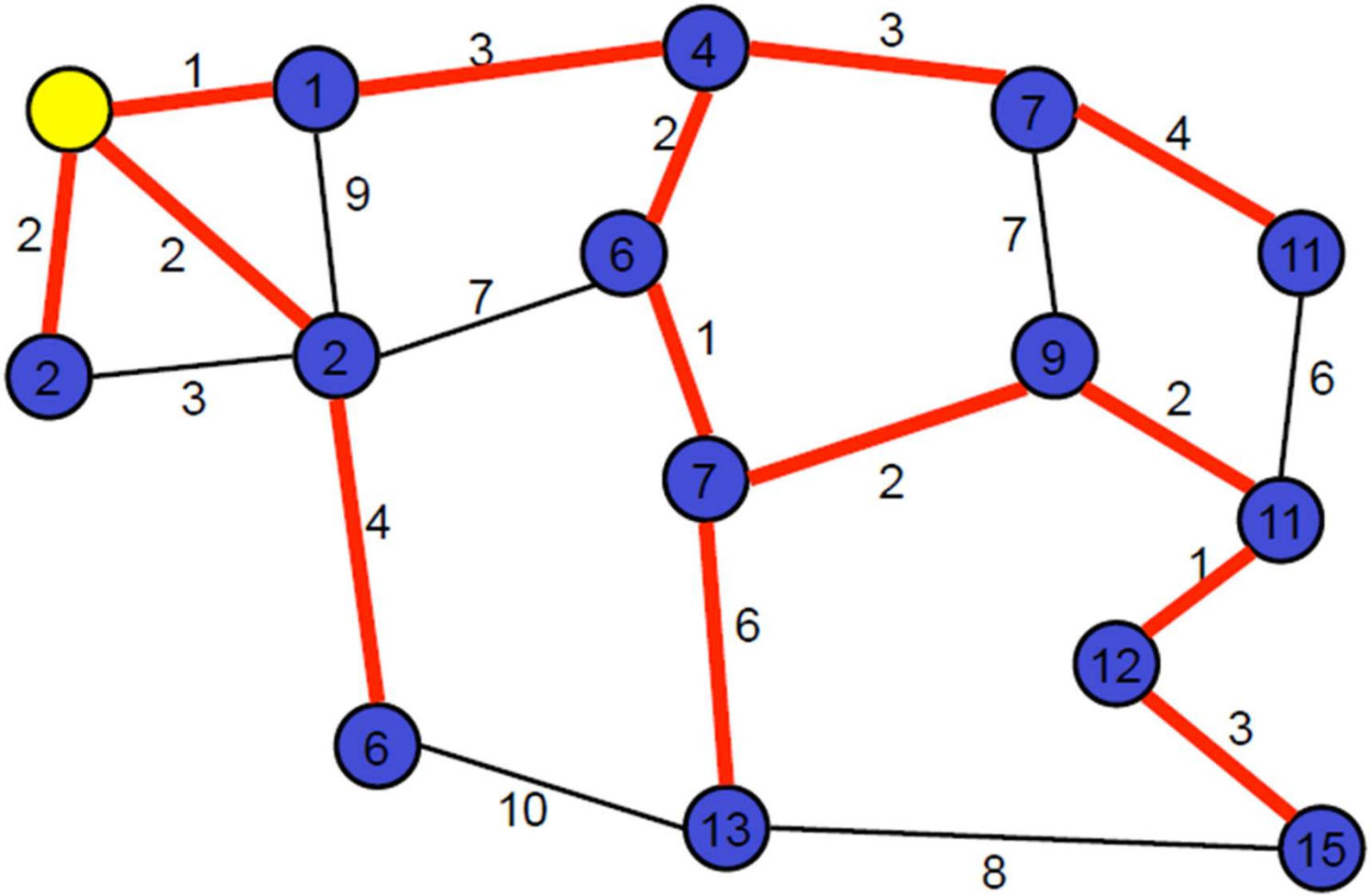
- Once a node receives LSP from other nodes, it uses the Dijkstra algorithm to find out the shortest paths

Dijkstra algorithm

- Input
  - A graph with edge weights (distance)

- Output
  - Shortest paths from one node to all others

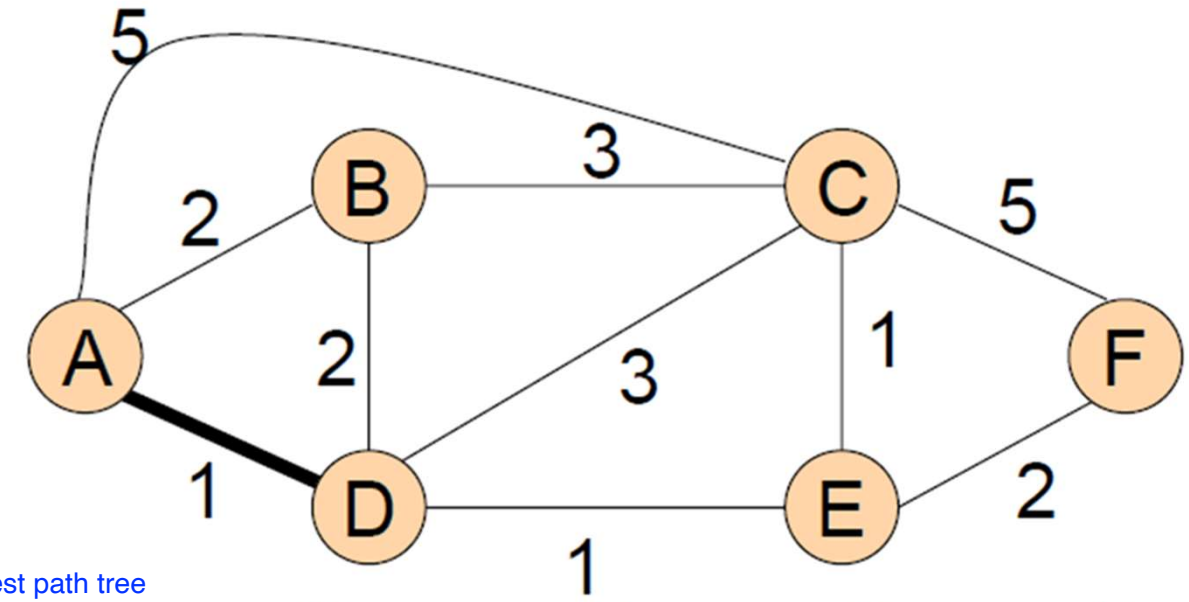
Dijkstra algorithm



Dijkstra algorithm

- Initially **C** = {source}
- Loop N-1 times
  - Find node M that is not included in **C** and is the closest to the source node
  - Include M in **C**

Dijkstra algorithm: example



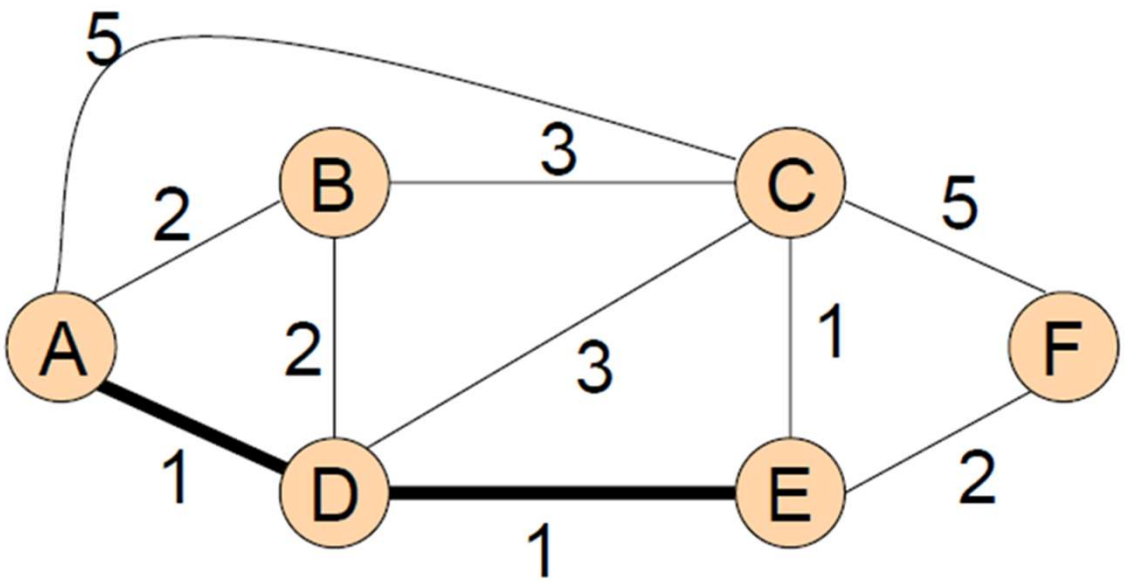
shortest path tree
맨 처음에는 나만 포함

		B	C	D	E	F
step	SPT	D(b), P(b)	D(c), P(c)	D(d), P(d)	D(e), P(e)	D(f), P(f)
0	A	2, A	5, A	**1, A**	~	~
1	AD	2, A	4, D		2, D	~

가장 작은 값을 shortest path가 되도록 선택

A랑 가장 가까운 D를 spt에 넣음

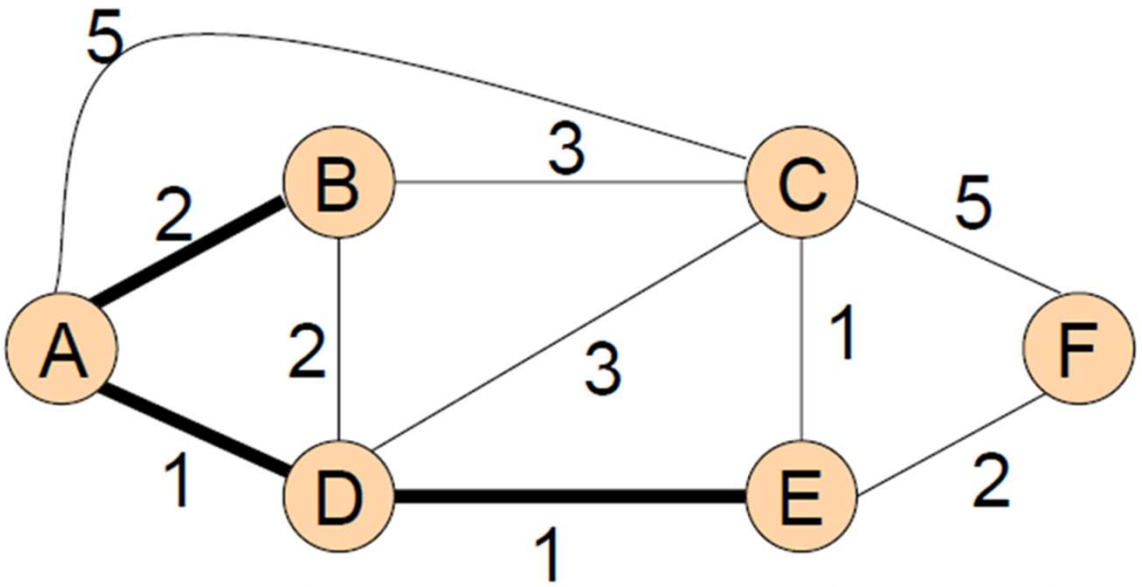
Dijkstra algorithm: example



D로 가쳐서 가는 길이
A에서 바로 간 것 보다 좋은지 판단

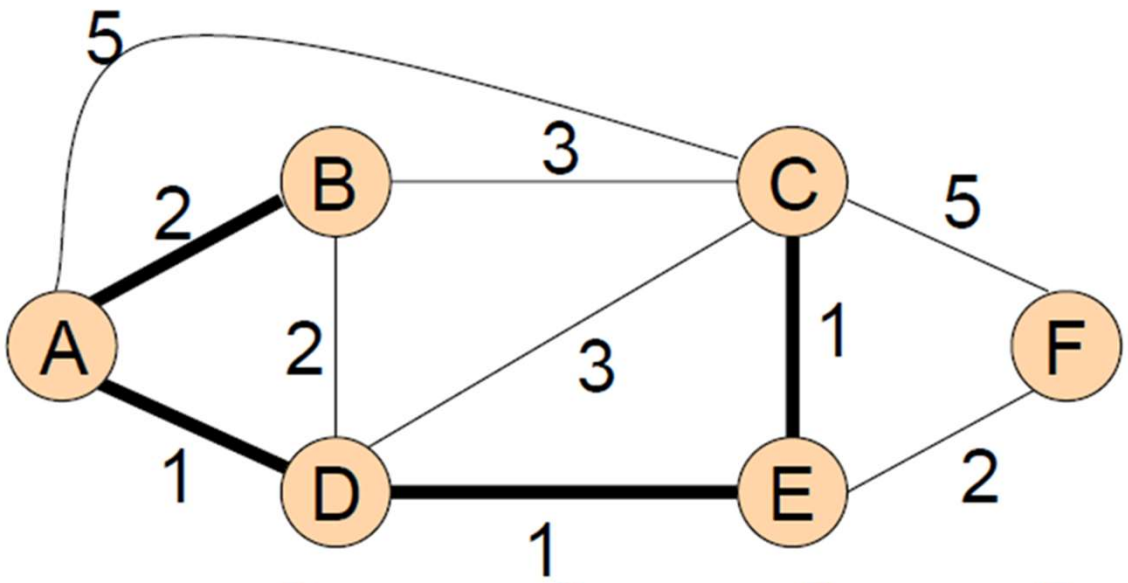
		B	C	D	E	F
step	SPT	D(b), P(b)	D(c), P(c)	D(d), P(d)	D(e), P(e)	D(f), P(f)
0	A	2, A	5, A	**1, A**	~	~
1	AD	2, A	4, D D를 거쳐서 E로 가면 4로 더 빠름		**2, D**	~
2	ADE	2, A	3, E			4, E

Dijkstra algorithm: example



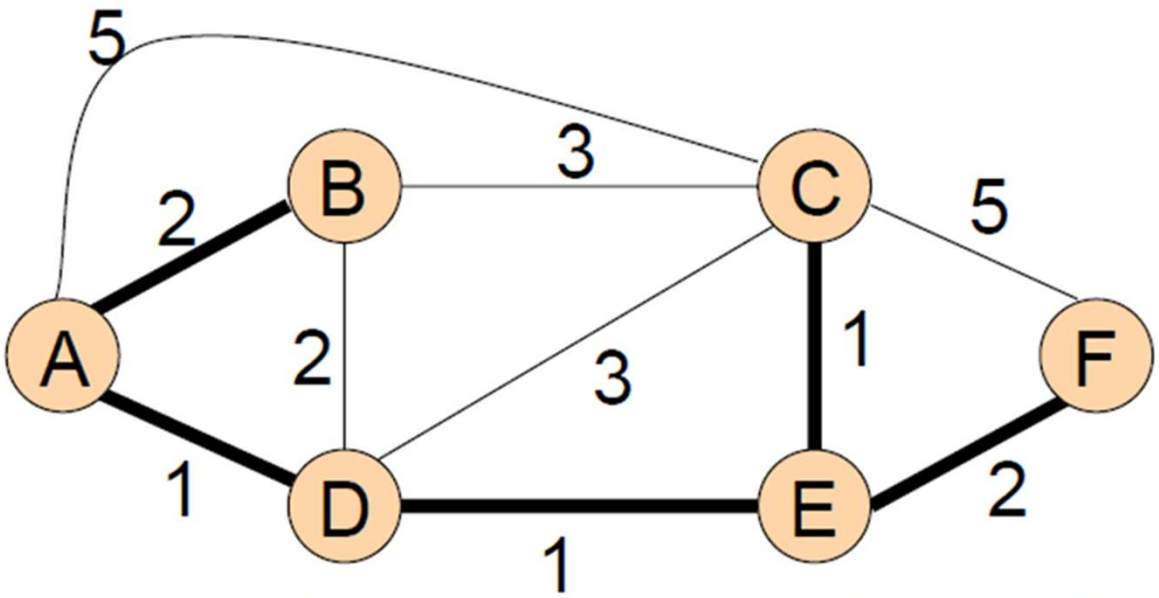
		B	C	D	E	F
step	SPT	D(b), P(b)	D(c), P(c)	D(d), P(d)	D(e), P(e)	D(f), P(f)
0	A	2, A	5, A	**1, A**	~	~
1	AD	2, A	4, D		**2, D**	~
2	ADE	**2, A**	3, E			4, E
3	ADEB		3, E			4, E

Dijkstra algorithm: example



		B	C	D	E	F
step	SPT	D(b), P(b)	D(c), P(c)	D(d), P(d)	D(e), P(e)	D(f), P(f)
0	A	2, A	5, A	**1, A**	~	~
1	AD	2, A	4, D		**2, D**	~
2	ADE	**2, A**	3, E			4, E
3	ADEB		**3, E**			4, E
4	ADEBC					4, E

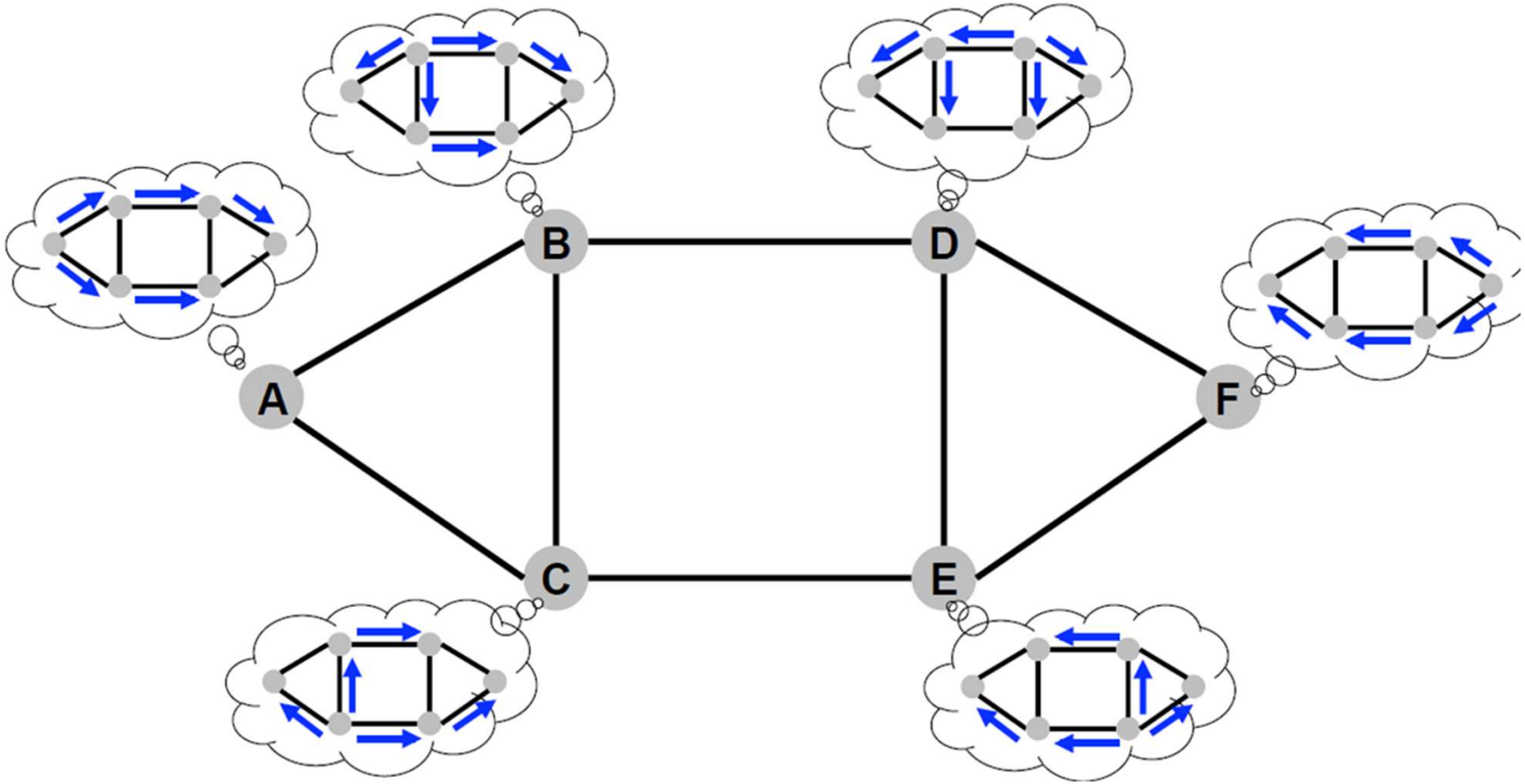
Dijkstra algorithm: example



		B	C	D	E	F
step	SPT	D(b), P(b)	D(c), P(c)	D(d), P(d)	D(e), P(e)	D(f), P(f)
0	A	2, A	5, A	**1, A**	~	~
1	AD	2, A	4, D		**2, D**	~
2	ADE	**2, A**	3, E			4, E
3	ADEB		**3, E**			4, E
4	ADEBC					**4, E**

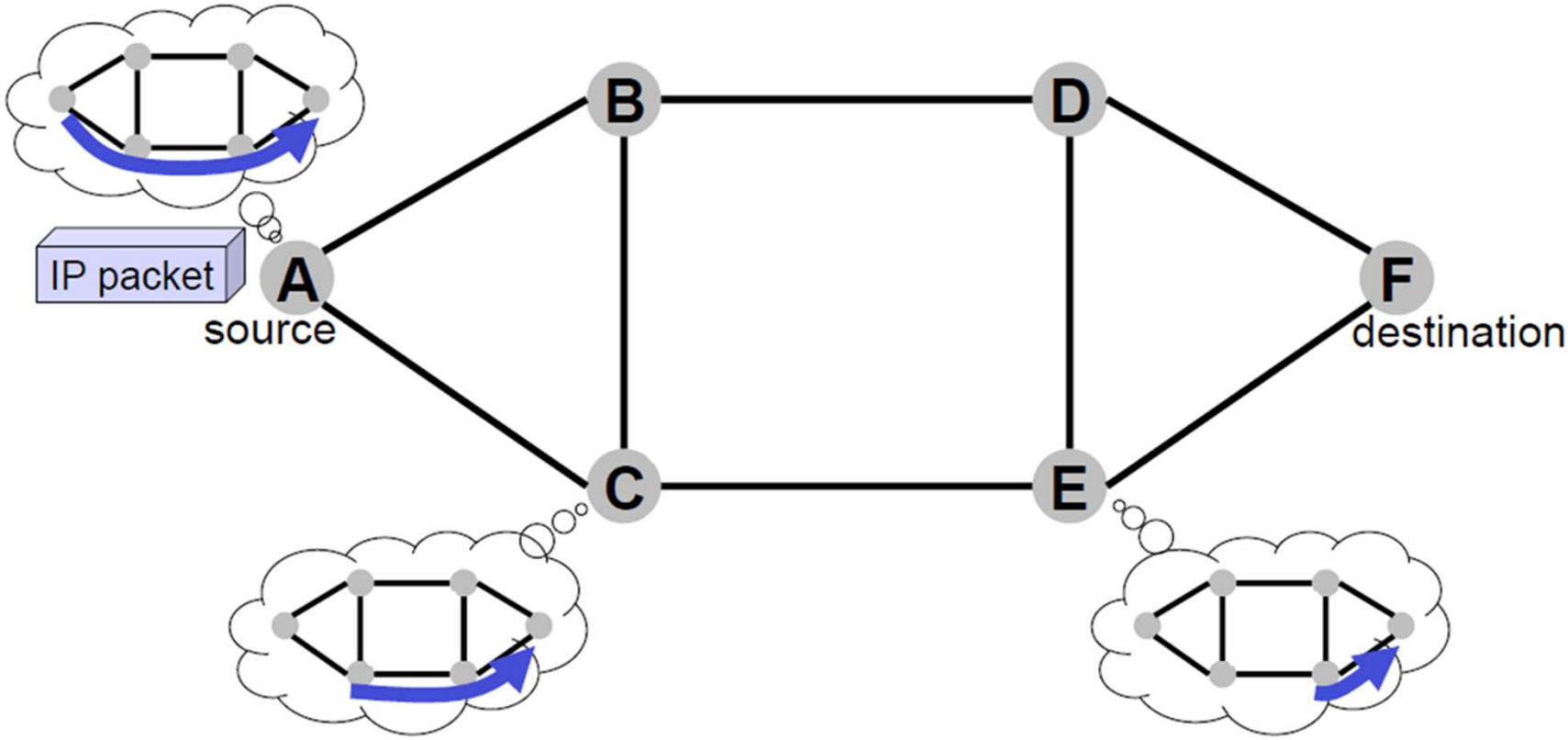
Link-state routing

- Each node uses Dijkstra algorithm to find the shortest paths to destinations



Link-state routing

- packet forwarding



Link-state routing

- Link-state packet forwarding
  - send LSP to <u>all nodes in the network</u>
    - compare: distance vector sends only to direct neighbors
  - Network-wide flooding

Flooding

- A sends LSP to all its neighbors
- Suppose node B receives A's LSP
- B checks the sequence number of A's LSP
  - If it is the newest,
    - B updates its routing table, resends the LSP to all its neighbors excluding node A
  - If it is not the newest
    - Discard the LSP

OSPF

- Open Shortest Path First
- A routing protocol based on link state routing
- Each router sends Link State Advertisement (LSA) packets
  - Periodically
  - On updates

OSPF

- LSA packet format

LS Age		Options	Type=1
나의 주소 Link-state ID			
Advertising router			
LS sequence number			
LS checksum		Length	
0	Flags	0	Number of links
Link ID			
Link data			
Link type	Num_TOS	Metric	
Optional TOS information			
More links			

ID of the router Sending LSA

ID of the router at th end of this link

Disambiguate paralle links

Cost of the link

Summary

- Link state
  - Information exchanged: (neighbor ID, distance to neighbor)
  - Information sent to: all nodes in the network
  - Protocol: OSPF
- Distance vector
  - Information exchange: (destination, distance to destination)
  - Information sent to: direct neighbors
  - Protocol: RIP

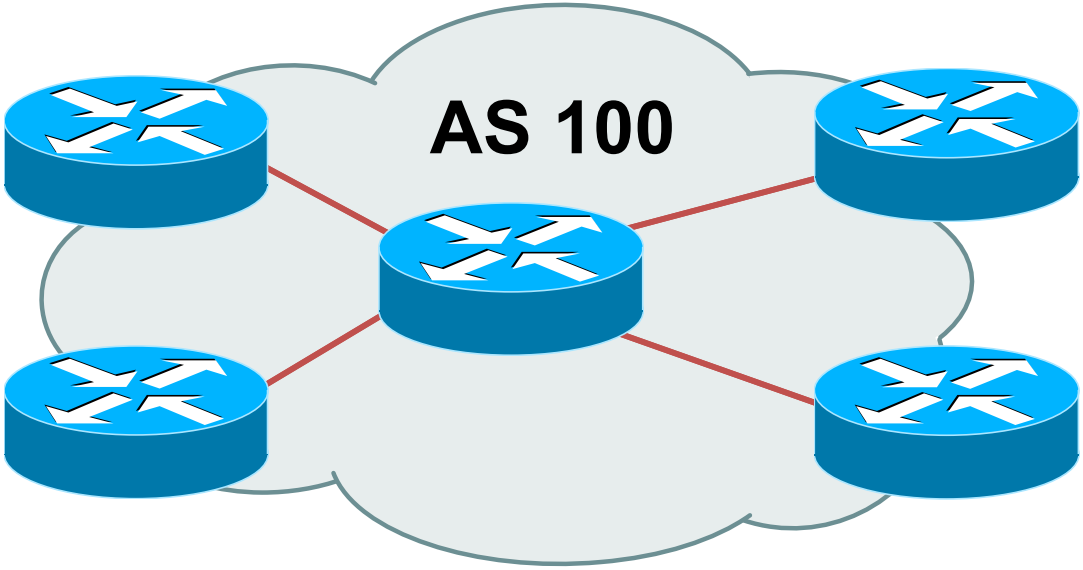
Border Gateway Protocols (BGP)

Flat vs. Hierarchical Networks

- Flat
  - No hierarchy
  - Not scalable
- Hierarchical
  - Network divided into “areas”
  - Scalable
- Internet is hierarchical

Automonous System (AS)

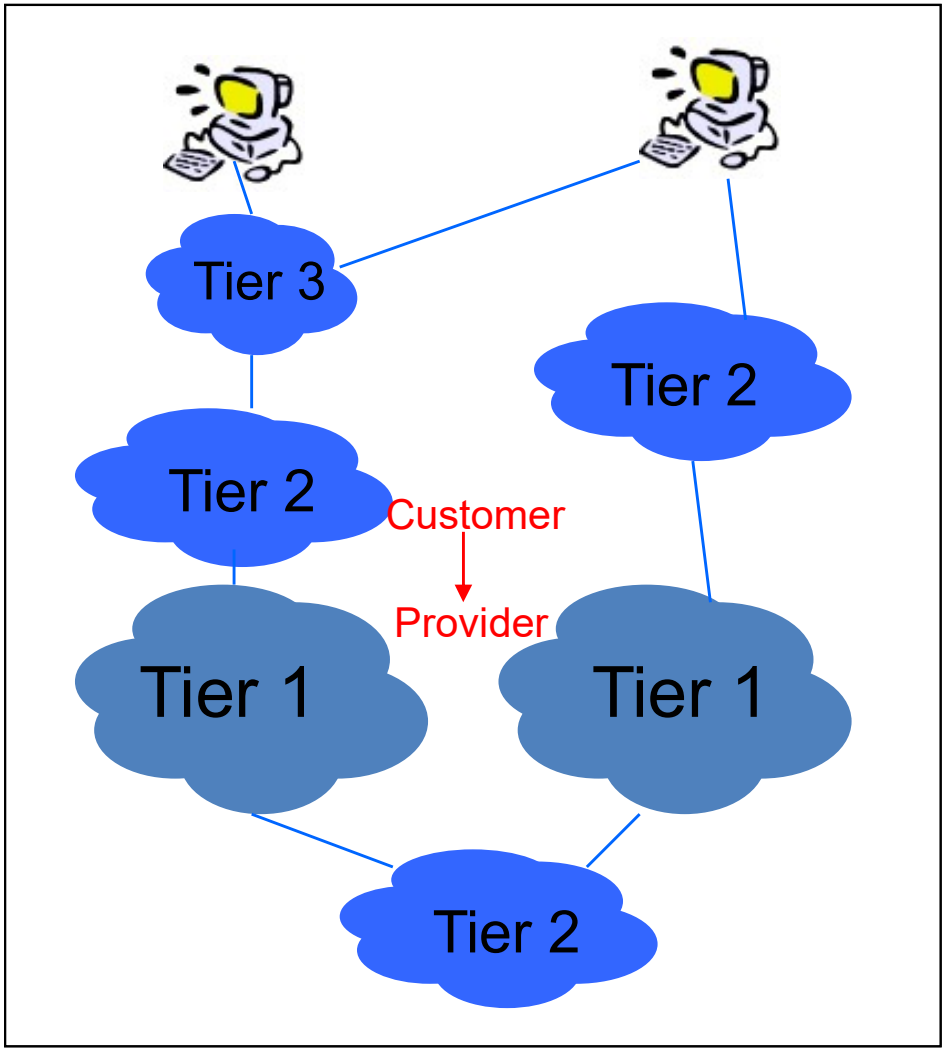
- Collection of networks with same routing policy
- Single routing protocol
- Single administrative control



Hierarchical Structure

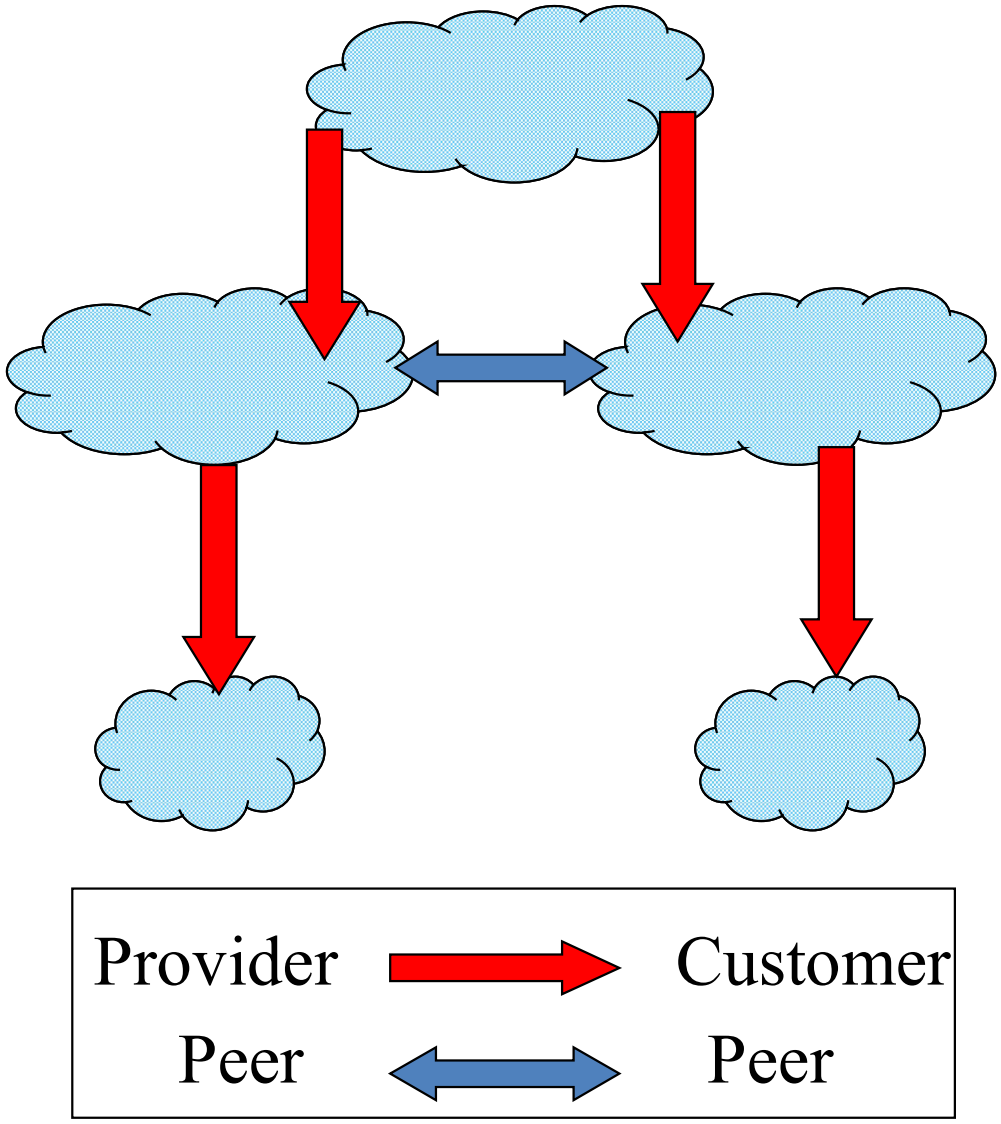
- Tiers
  - Tier 1 AS: global reachability
  - Tier 2 AS: regional/country
  - Tier 3 AS: local

- AS relationships
  - provider/customer 다른 티어 끼리
  - peers 같은 티어 끼리



Business Model

- Customer pays provider for service
- Peers: mutual cooperation
- Multi-homing
  - An AS may have multiple providers

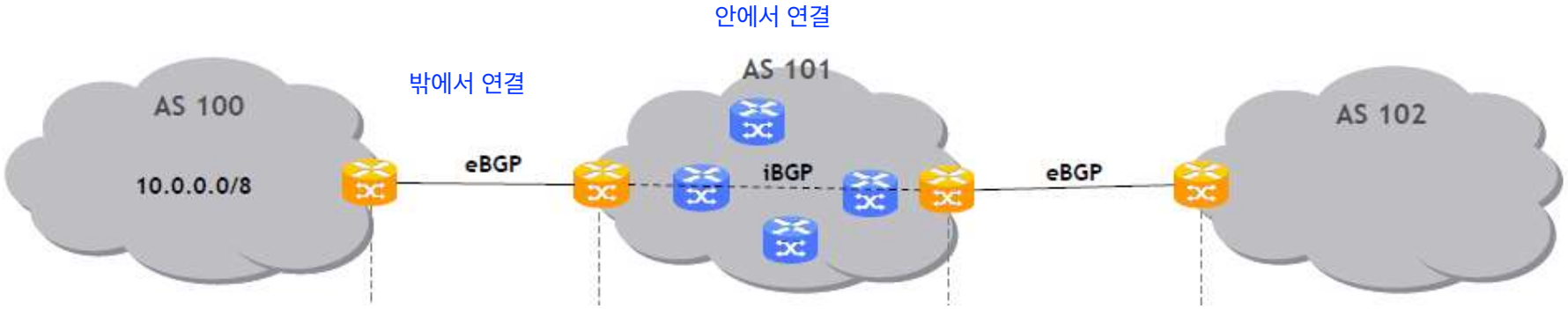


Routing

- Intra-domain (inside a single AS)
  - RIP (Distance Vector)
  - OSPF (Link State)
- RIP/OSPF do not scale well
  - Count-to-infinity problem (RIP)
  - Routing information flooded to all nodes (OSPF)
- Inter-domain needs a different routing scheme
  - BGP (Border Gateway Protocol)

BGP

- Routing between "border" routers
- Border routers
  - Routers that have links to other AS
- eBGP (E-BGP, external BGP)
  - Routing between border routers of different domains
- iBGP (I-BGP, internal BGP)
  - Routing between border routers of the same domain



BGP routing

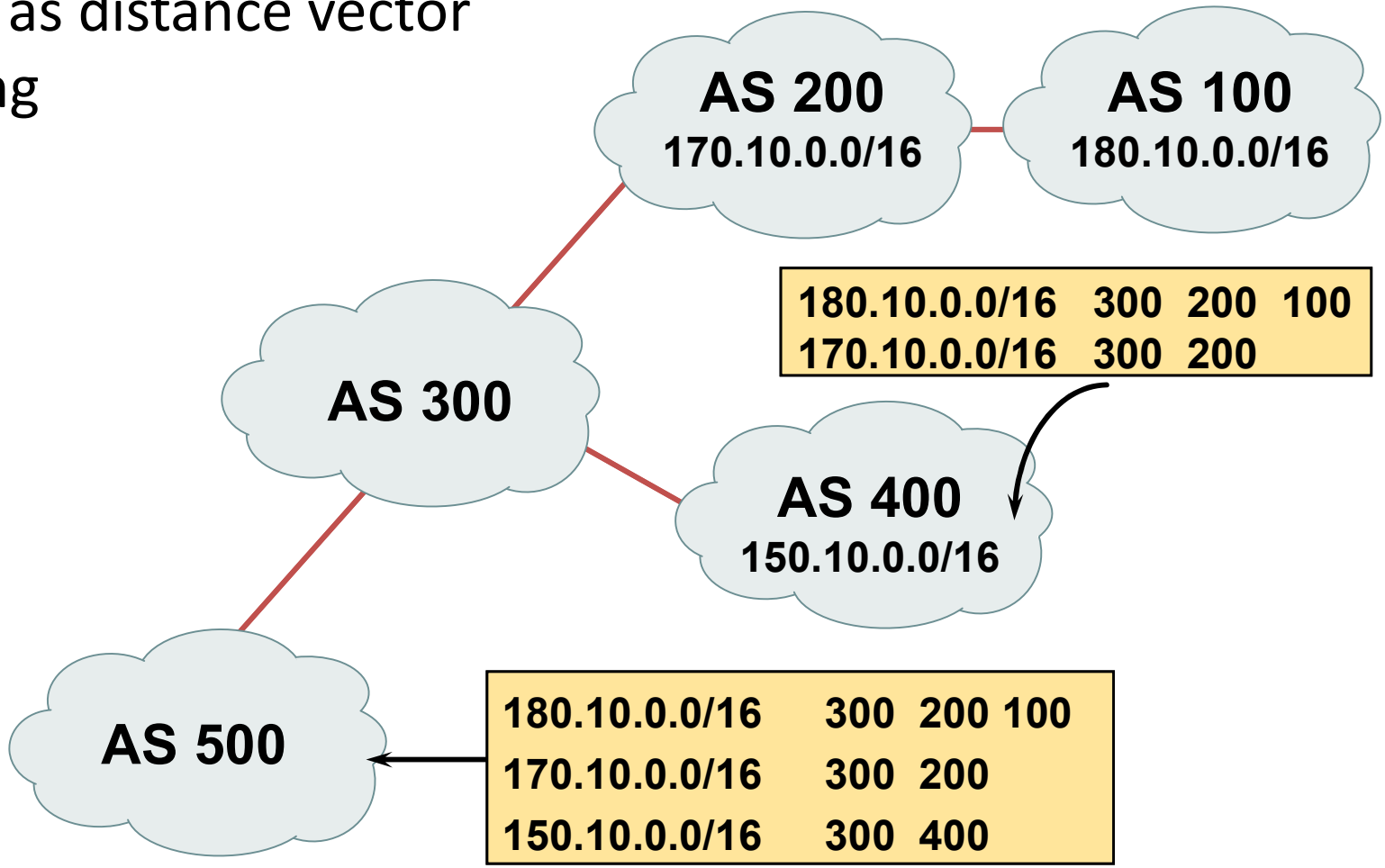
- Distance vector with paths (Path vector routing)

- BGP routing table

Prefix	Origin AS	Path
128.32.0.0/16 128.32.0.0 - 128.32.255.255 는 123번 as에 있는 거	123	14 56 123 길을 두 개를 만듦
	123	34 101 203 123
128.32.101.0/24	15	50 15 ~~15~~

BGP routing

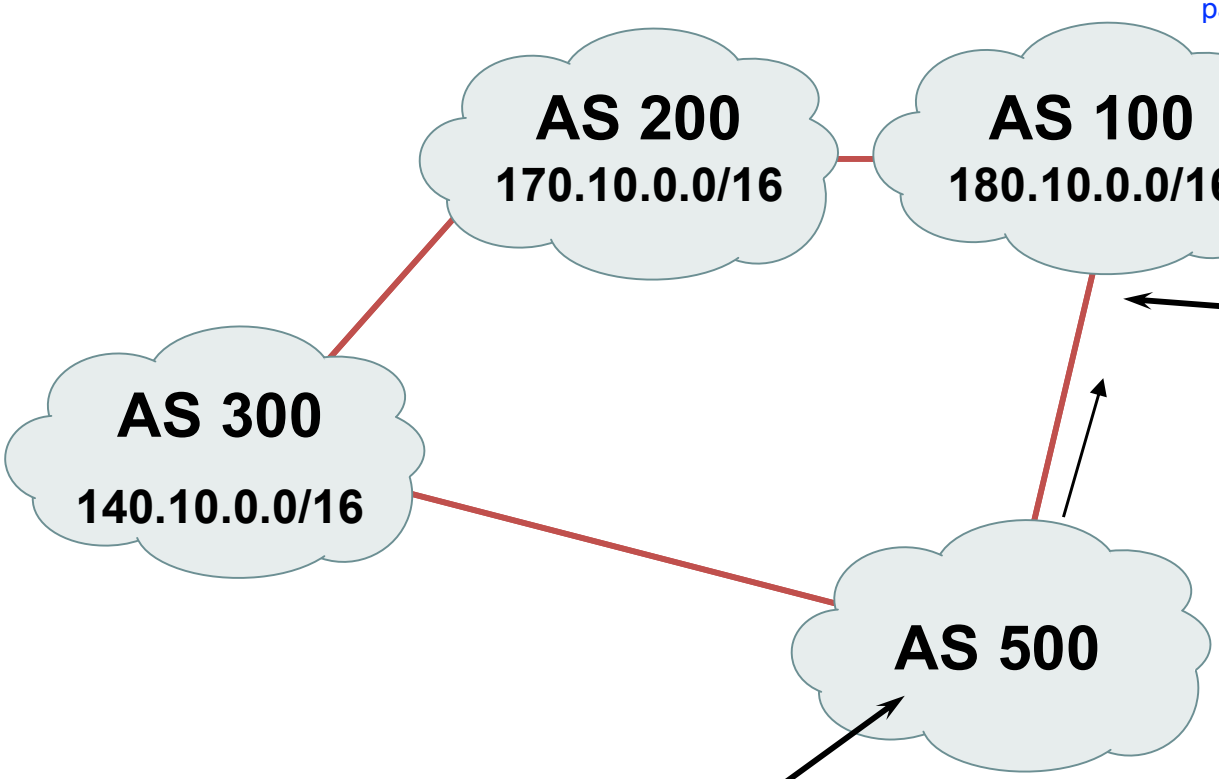
- BGP routers (also called BGP speakers) send route advertisements to other BGP routers
  - Same as distance vector routing



BGP routing

- Using path information, loop is detected

path에 있는 라우터를 다 적기 때문에 loop detect



140.10.0.0/16	500 300
170.10.0.0/16	500 300 200

180.10.0.0/16	300 200 100
170.10.0.0/16	300 200
140.10.0.0/16	300

180.10.0.0/16 is not announced to AS100 as AS500 sees that it is originated from AS100, and that AS100 is the neighbouring AS – loop detection in action

iBGP

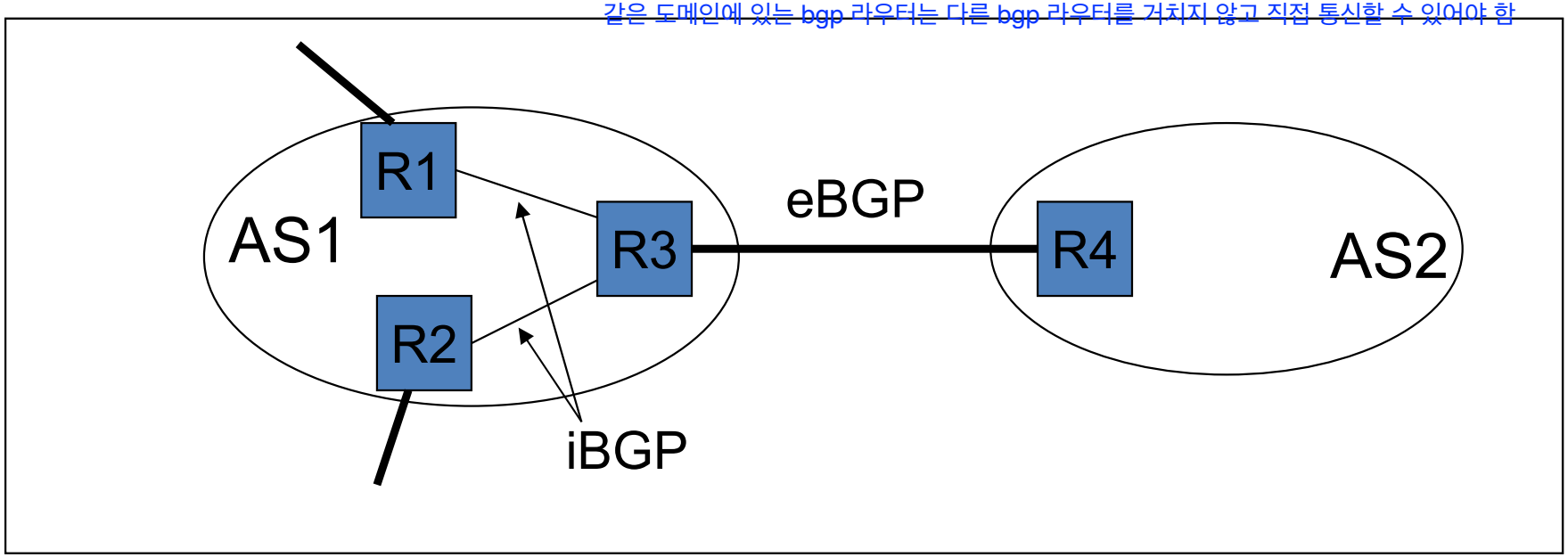
- Same as eBGP
- Routes learned from eBGP can be advertised to iBGP neighbors
- Routes learned from iBGP neighbor cannot be advertised to other iBGP neighbors
  - Avoid loops
  - AS-path information not carried inside an AS
- All iBGP routers inside an AS must be fully connected

iBGP

- R3 can tell R1 and R2 routes from R4
- R3 can tell R4 routes from R1 and R2
- R3 cannot tell R2 routes from R1
  - R2 must directly hear routes from R1 (not through R3)
  - iBGP routers must be fully connected

r3는 r1,r2에게 날 통하면 r4로 간다는 걸 알려줌
r4에게도 r1,r2로 갈 수 있다는 걸 알려줌

r1, r2, r3는 서로 바로 연결되어 있어야 함
같은 도메인에 있는 bgp 라우터는 다른 bgp 라우터를 거치지 않고 직접 통신할 수 있어야 함



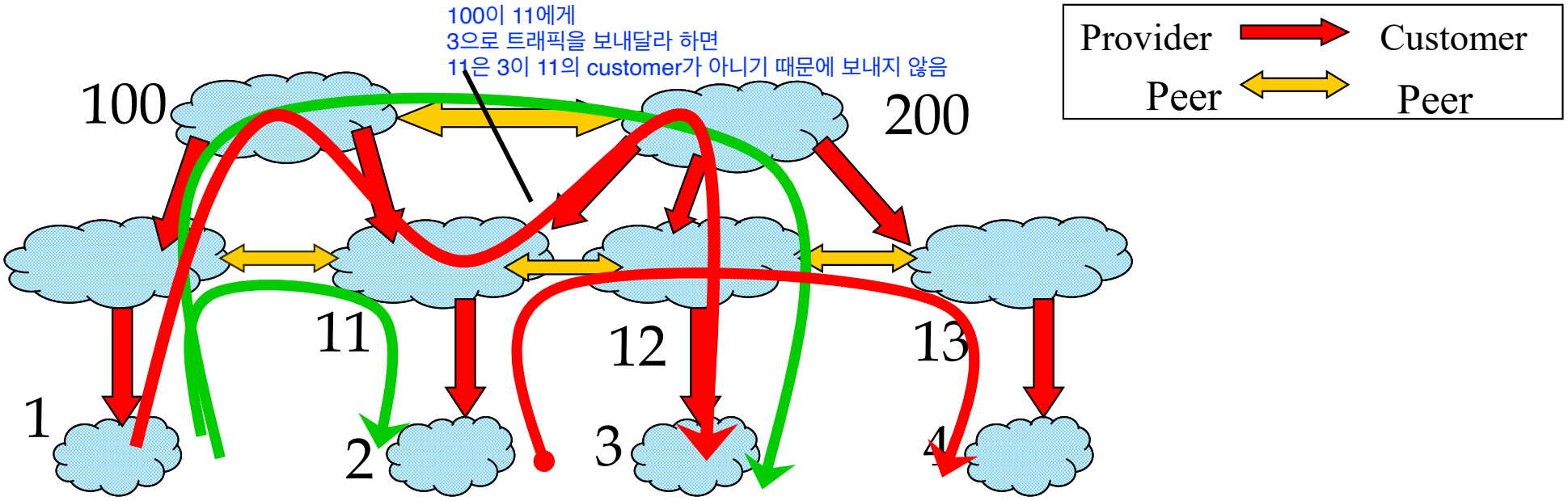
Policies

- In BGP, shortest path is often not the best path
  - Money involved
- Path selected based on policies
  - customer > peer > provider
- Policies on transit traffic
  - Transit traffic: traffic that does not go to my customers 내 customer로 안 가는 traffic
  - Providers accept transit traffic
  - Customers and Peers do not accept transit traffic
- Other policies may apply

Policies affect routing

- Green: valid routes
- Red: invalid routes
- Summary: traffics go up and then down

provider, customer 관계에 의해서 갈 수 있는 경로가 있고 없는 경로가 있다



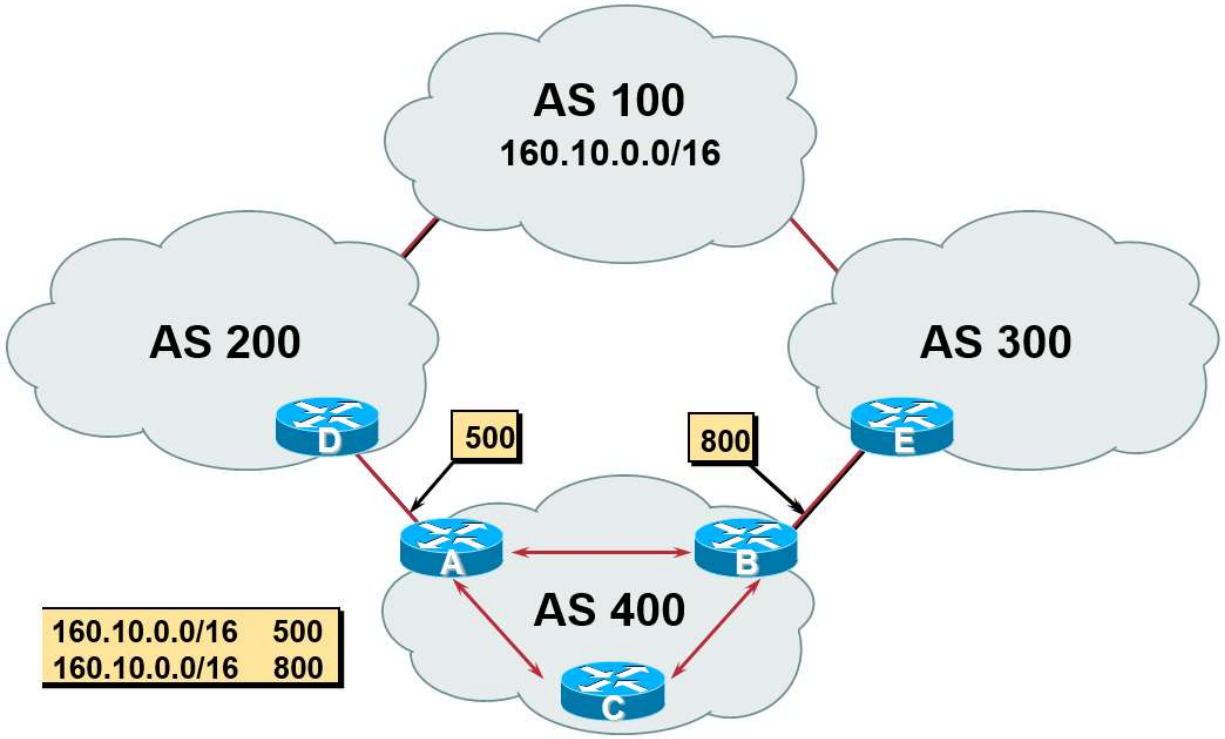
12랑 13이 peer 계약을 맺지 않으면
12는 12->13->4로 보내는 일을 하지 않는다
대신 12 -> 200 ->13 ->4 로 200으로 올릴 수 있다

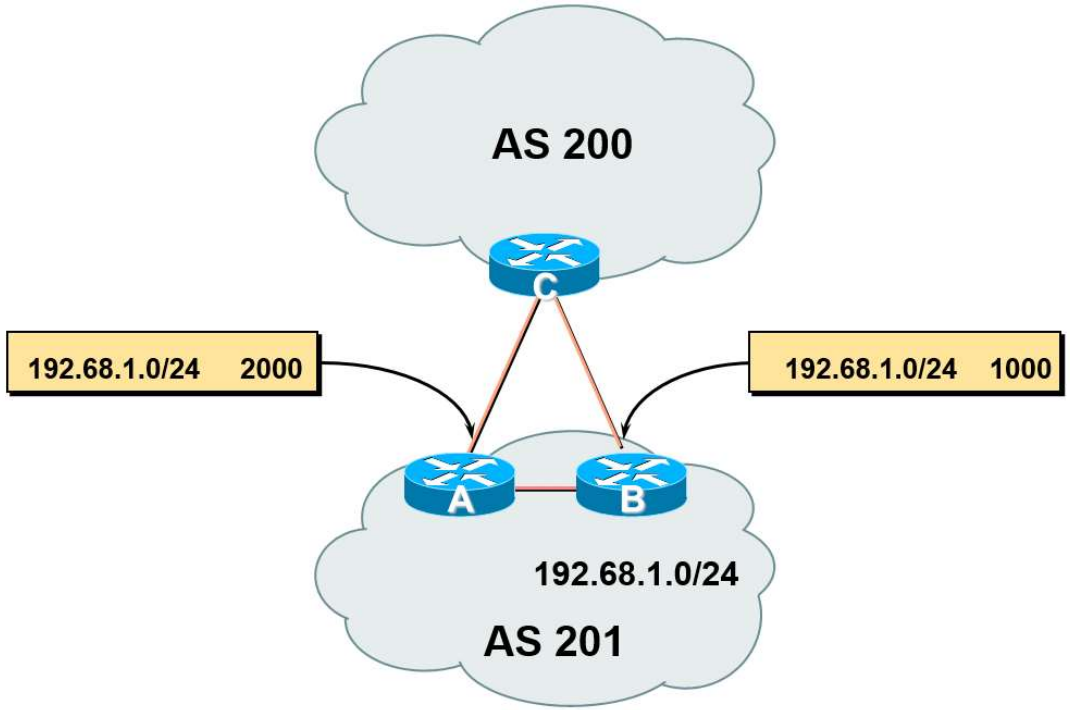
Route advertisements

- To customer link
  - Advertise all routes
- To provider link
  - Advertise customers only
- To peer link
  - Advertise customers only

Local Preference

- Multi-homed AS
  - Has multiple providers
  - Preference on which provider to use
  - Higher local preference value wins



Multi-Exit Discriminator (MED)

- AS201 wants traffic from AS200 to be directed to B rather than A
- B advertises smaller MED compared to A
- Smaller MED wins