데이터통신과 네트워킹

Data Communication & Networking Ch. 10

동아대학교 컴퓨터AI 공학부
S03 301-01호실 (이양민 교수 연구실)
2024.11 (Kakao ID: yanwenry)

CHAPTER

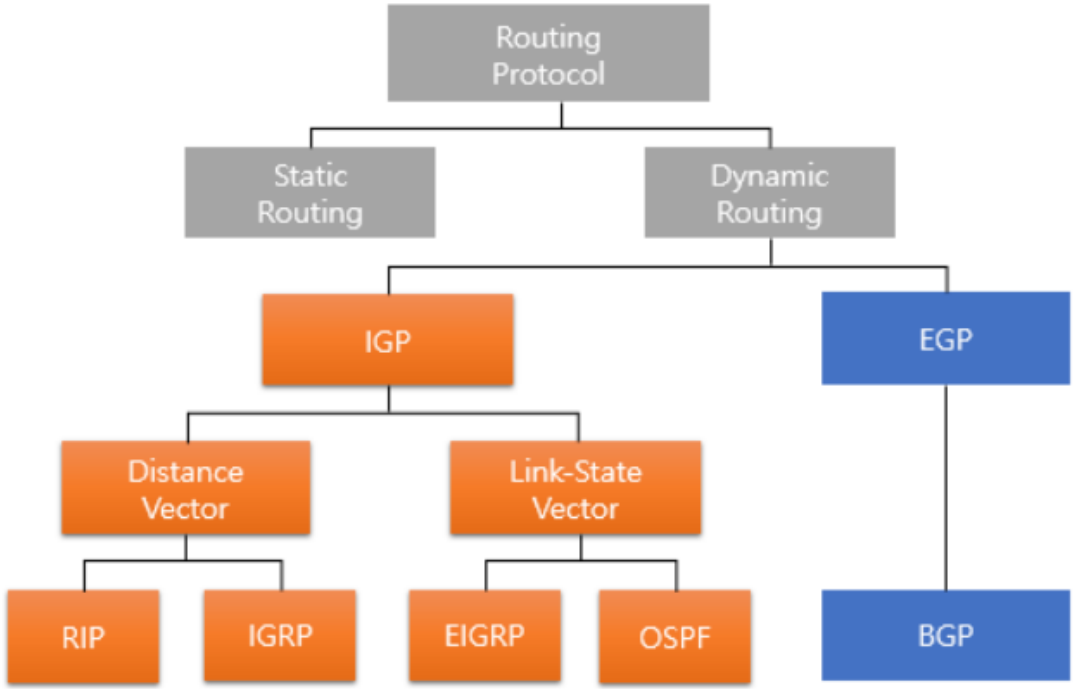
10

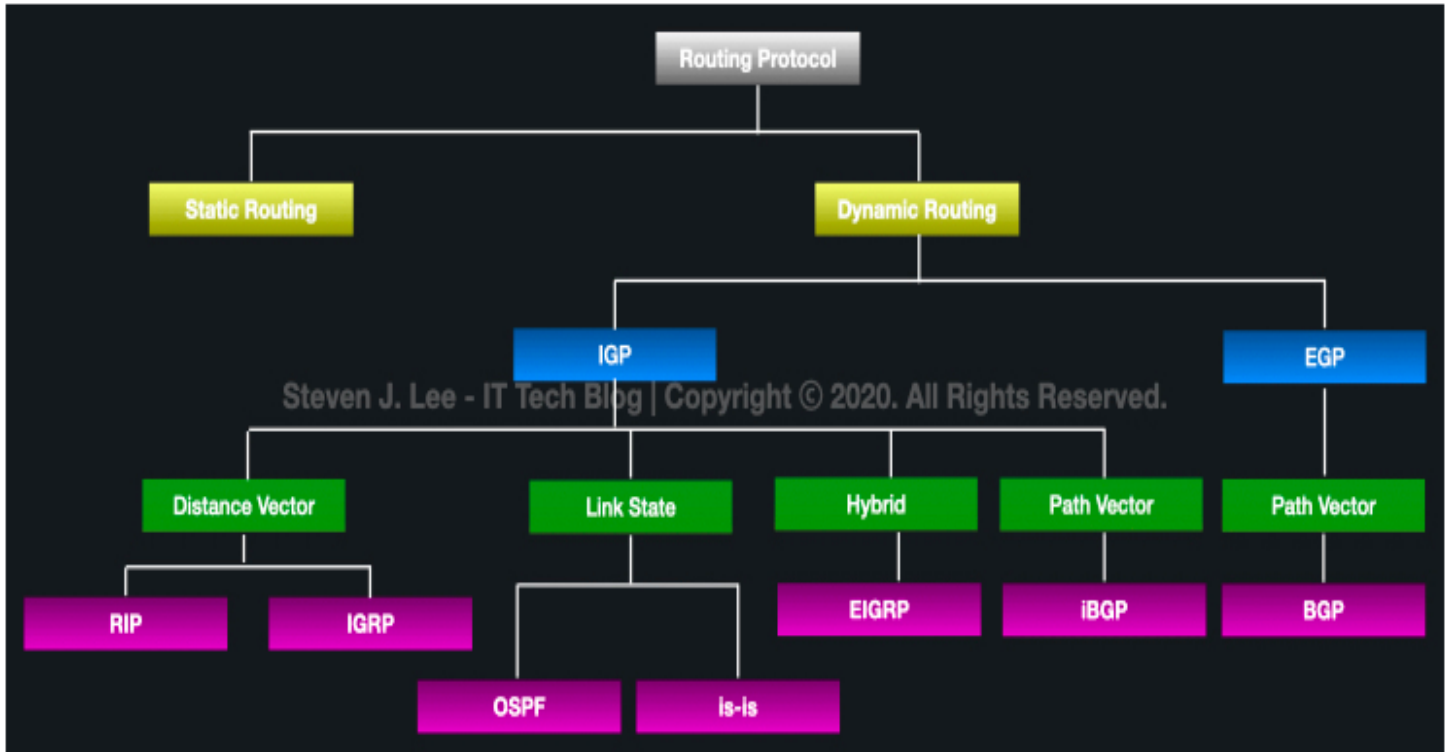
라우팅 알고리즘

Section

SECTION 01

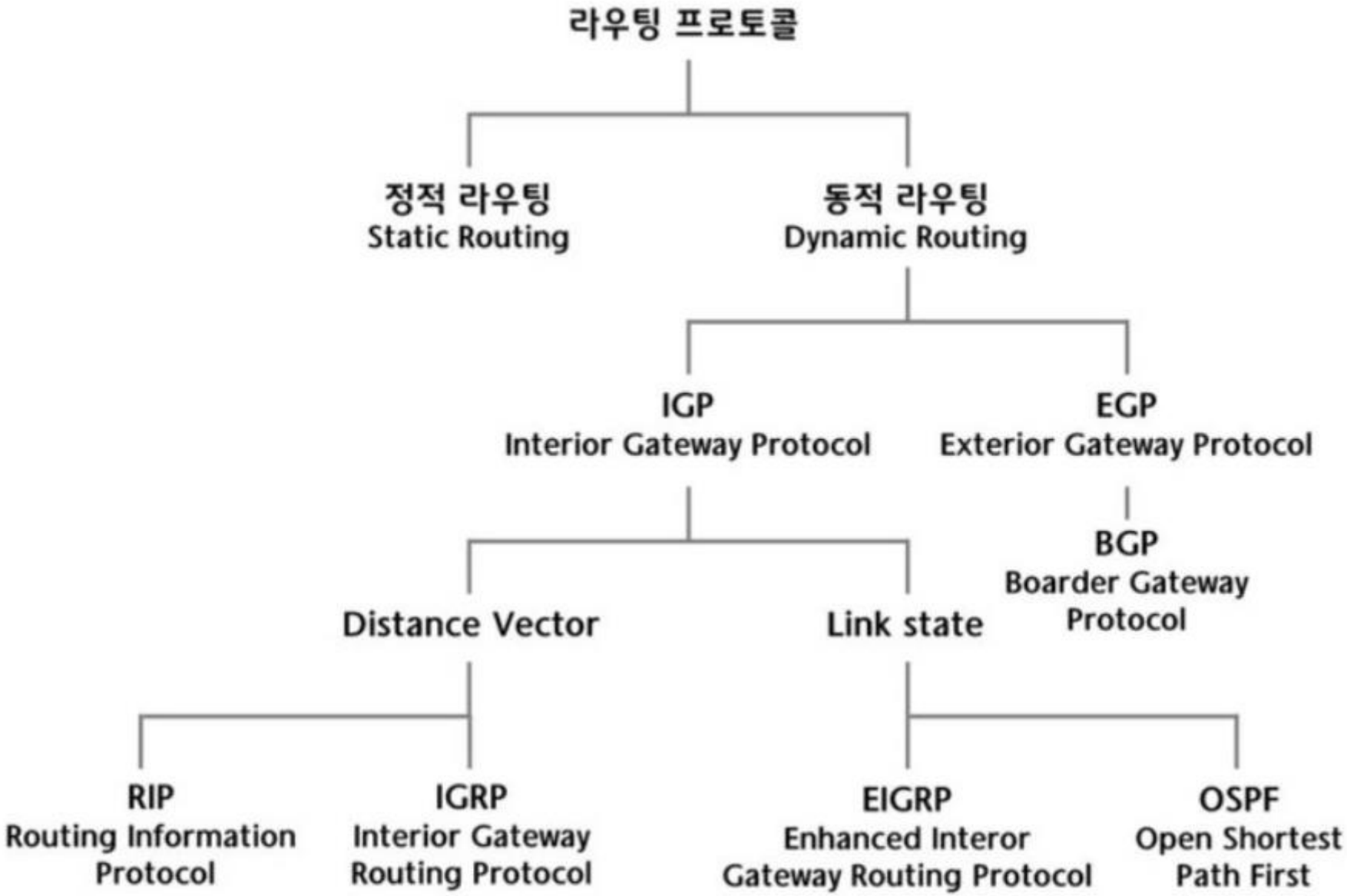
라우팅 알고리즘 분류





SECTION 01

라우팅 알고리즘 분류



SECTION 01

정적 라우팅 알고리즘

1. 라우팅 알고리즘의 개요

- 네트워크 계층에 있는 IP의 가장 중요한 작업은 출발지에서 목적지까지 패킷을 전달하는 것 -> 이때 사용하는 것이 **라우팅 알고리즘**routing algorithm
- 정적 라우팅 알고리즘은 현재의 상태에서 어떤 곳으로 패킷을 보내야 목적지에 도달하는지를 찾는 알고리즘.
- 시시각각 라우터의 상태를 감지하여 경로를 변경하는 알고리즘이 동적 라우팅 알고리즘.

표 10-1 라우팅 알고리즘 분류

	정적 라우팅 알고리즘	동적 라우팅 알고리즘
특징	현재 상태 정보를 반영하지 않음	상태 정보를 반영하여 경로 변경
알고리즘	최단경로, 플러딩	거리벡터 라우팅, 연결상태 라우팅, 계층적 라우팅

SECTION 01

정적 라우팅 알고리즘

- 라우터들과 통신선의 연결을 그래프graph로 표현.
  - 그래프는 노드node와 선edge의 집합 -> 노드는 라우터를, 선은 통신선을 의미.
  - 네트워크에서 통신은 양방향이기 때문에 방향성이 없는 선.
  - 선 위에 적혀있는 숫자들은 패킷을 전송하는데 걸리는 시간 혹은 거리를 의미 -> 여기서는 숫자가 밀리초(msec)를 의미.
- 라우팅 알고리즘의 가장 중요한 목적은 각 라우터로 가는 가장 빠른 경로를 찾는 것.
- 라우팅 알고리즘이 최단경로(최소시간 경로)를 결정하면 결과는 트리tree(사이클이 없는 그래프).

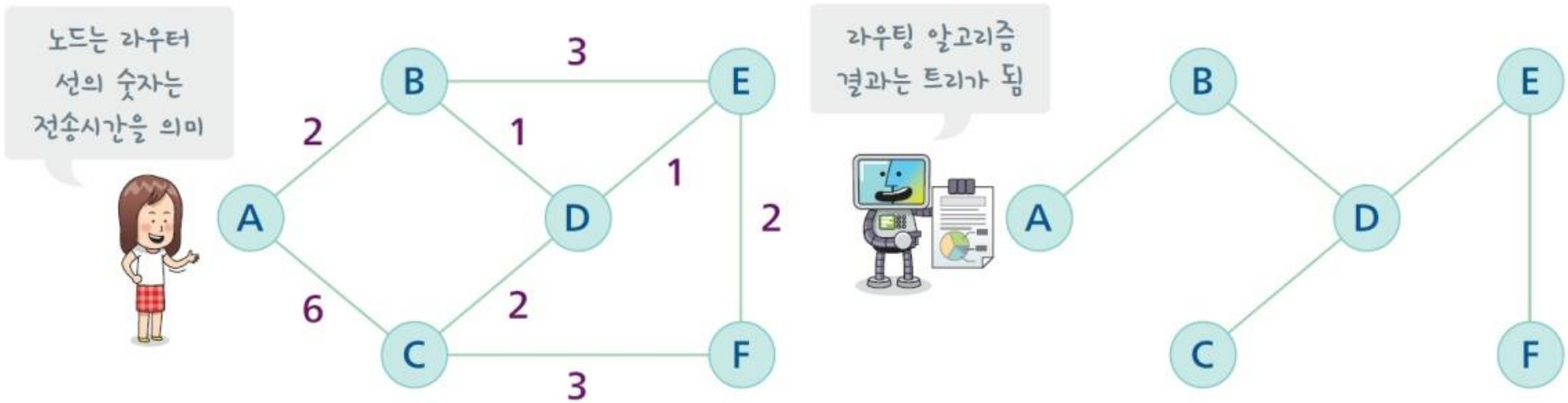


그림 10-1 네트워크의 그래프 표현

SECTION 01

정적 라우팅 알고리즘

2. 최단경로(shortest pach) 알고리즘

- **최단경로 알고리즘**Shortest Path Algorithm은 대표적인 정적 라우팅 알고리즘 -> 다익스트라 알고리즘.
- 최단경로 알고리즘은 집합의 개념을 사용.
  - 초기에는 출발지점의 라우터(노드)가 집합에 들어감.
  - 새로운 라우터가 집합에 들어오면, 집합에 속한 라우터들에서 보이는 경로 중 가장 짧은 경로를 찾아 집합에 넣는 방식.
  - 모든 라우터가 집합에 다 들어갈 때 까지 반복

표 10-2 최단경로 알고리즘

```
초기값 => 시작 노드를 집합에 넣음
모든 노드가 집합에 들어갈 때까지 다음을 반복
{
    ① 집합에 있는 모든 노드에서 바로 보이는 노드의 거리 계산
    ② 집합에 있는 노드를 제외한 가장 작은 값을 가진 노드 선택
    ③ 선택된 노드를 집합에 삽입
}
```

정적 라우팅 알고리즘

- 각 노드(라우터)들은 두 개의 초기 값을 가짐. 출발 노드에서부터 해당 노드까지 도달하는데 걸리는 시간(msec)과 어디에서부터 오는 경로인지를 나타내는 경로정보가 저장 -> (시간, 경로정보)의 쌍으로 표시, 초기값은 (∞, -).
- 모든 노드의 초기값이 만들어지면 출발 노드인 A를 집합에 넣고 최단경로 알고리즘을 반복 -> A 노드가 집합에 들어갔다는 것은 {A}로 표시.
- 집합 {A}에서 한 번에 갈 수 있는 모든 노드의 시간을 계산 -> A에서 한 번에 갈 수 있는 노드는 B와 C이다. B 노드의 값은 (2, A)로 변경, C 노드는 (6, A)로 변경.
- B의 (2, A)가 가장 작은 값 -> B 노드가 집합에 들어가고 집합은 {A, B}가 됨.

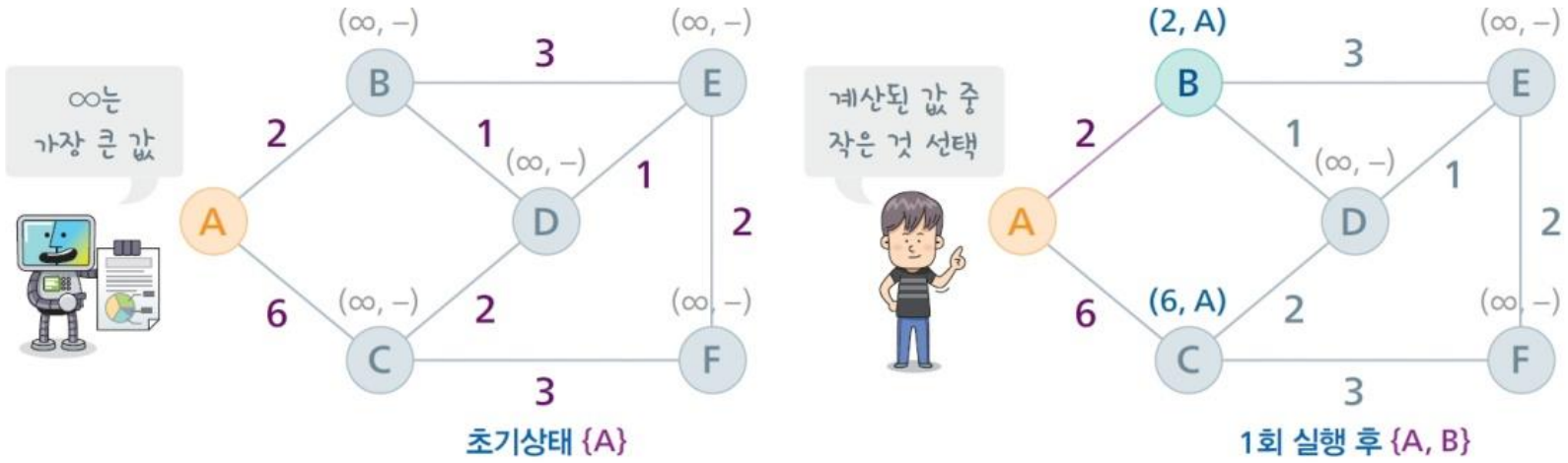
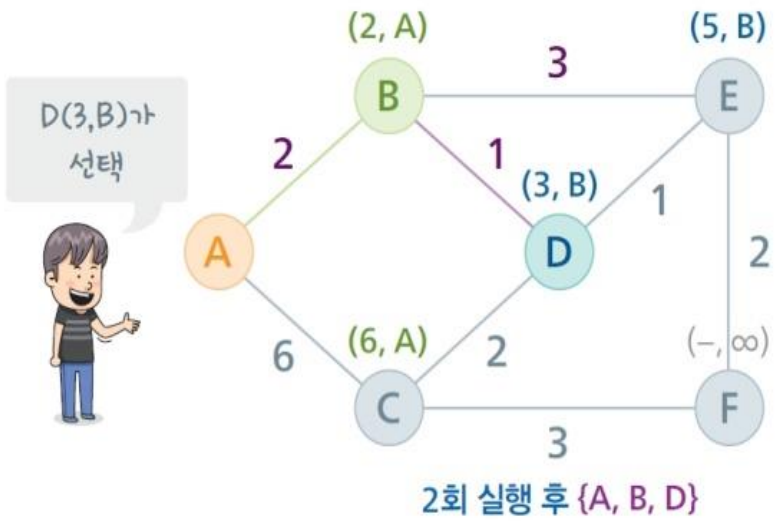


그림 10-2 최단경로(shortest path) 알고리즘의 초기값과 1회 실행 후

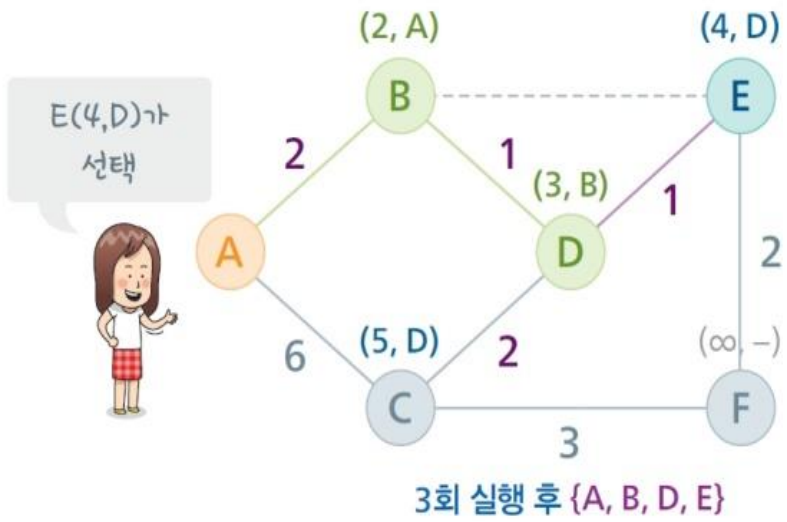
정적 라우팅 알고리즘

- 집합 {A, B}에 대하여 한 번에 갈 수 있는 모든 노드의 값을 새로 계산.
- D와 E 노드의 값이 갱신 -> A에서 B 노드까지 2msec가 걸리고, B에서 D 노드까지 1msec가 걸림으로 D 노드의 값은 둘을 합한 3이 되며, B 라우터를 거쳐서 도달하기 때문에 (3, B)가 됨. E 노드의 값은 (5, B)가 됨.
- 2회 실행 후에서는 가장 작은 값 (3, B)을 가진 D가 집합에 들어가서, 집합은 {A, B, D}가 됨.

- C는 (5, D)로 E는 (4, D)로 변경, E 노드의 (4, D)가 가장 작은 값임으로 E가 집합에 들어감.
- B-D-E의 경로가 최단 경로, B-E의 경로는 더 이상 사용되지 않음.



2회 실행 후 {A, B, D}



3회 실행 후 {A, B, D, E}

정적 라우팅 알고리즘

- E 노드가 새로 들어오고 F의 값은 (6, E)로 바뀜 -> C 노드가 집합에 들어감 -> 눈으로 보면 A-C가 빠를 것 같지만, 실제로는 A-B-D-C가 빠르다는 것을 최단경로 알고리즘이 찾음.

- 5회 실행에서 F 노드의 (6, E)가 채택되고 C-F 경로는 더 이상 사용되지 않음 -> 모든 노드가 집합에 들어 왔기 때문에 최단경로 알고리즘은 종료.
- 사용하지 않은 점선을 제외하면 그래프가 트리(싸이클이 없는 그래프)로 바뀜.

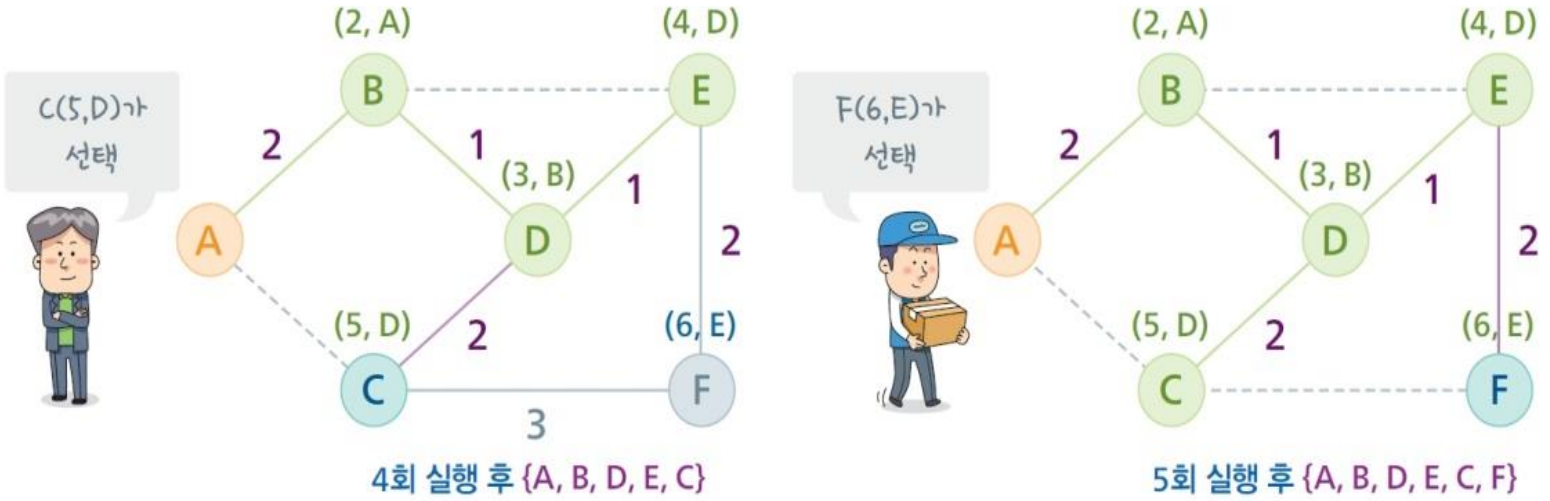


그림 10-3 최단경로 알고리즘의 실행과정

정적 라우팅 알고리즘

3. 플러딩(flooding) 알고리즘

- 플러드flood는 홍수를 의미하며, **플러딩**flooding 알고리즘은 네트워크에 홍수를 일으켜 가장 빠른 길을 찾는 정적 라우팅 방식.
- 라우터들은 패킷이 들어온 선을 제외한 모든 선에 패킷을 복사하여 보냄 -> 패킷에는 지나온 라우터들을 적어 놓음 -> 가장 먼저 도착한 패킷에 적혀 있는 경로가 가장 빠른 경로.
- 플러딩은 알고리즘이 단순하여 구현하기 싶다는 장점이 있지만 많은 패킷이 폭주하여 네트워크의 정체를 유발하는 단점이 있음.

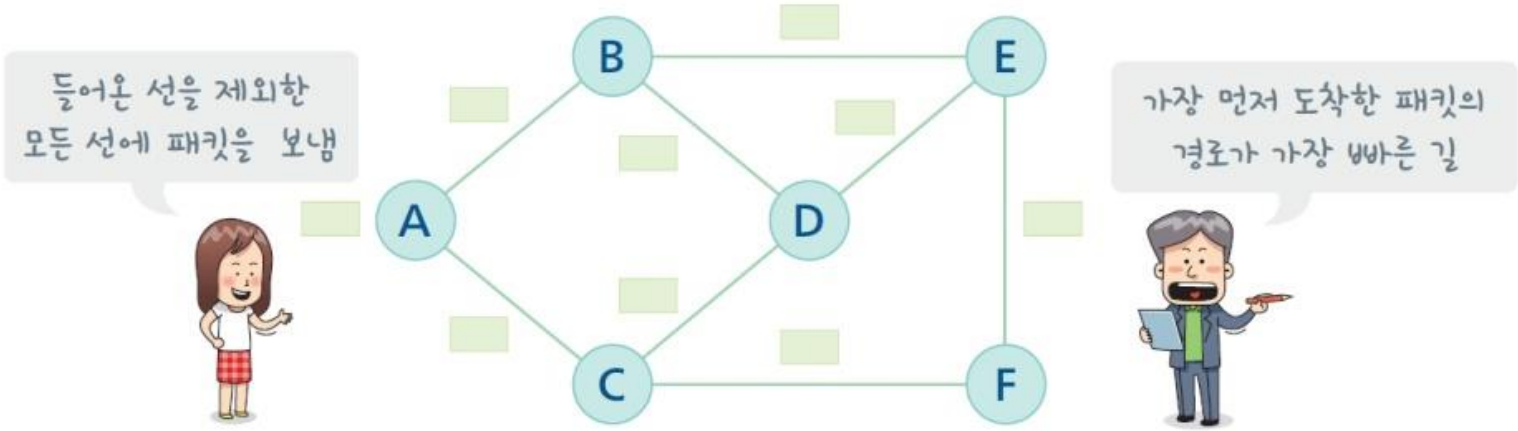
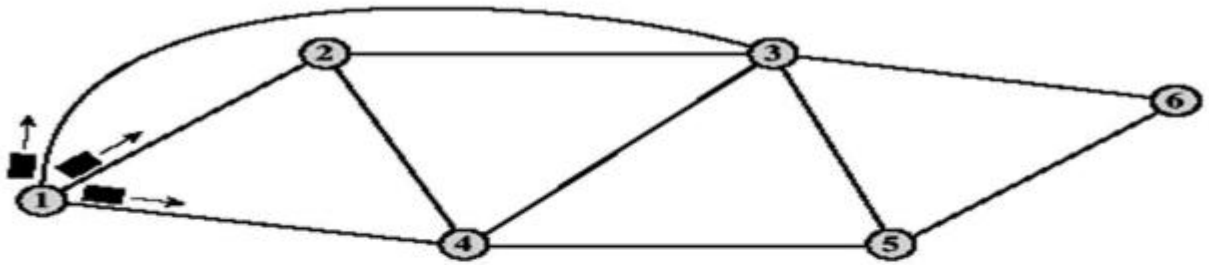
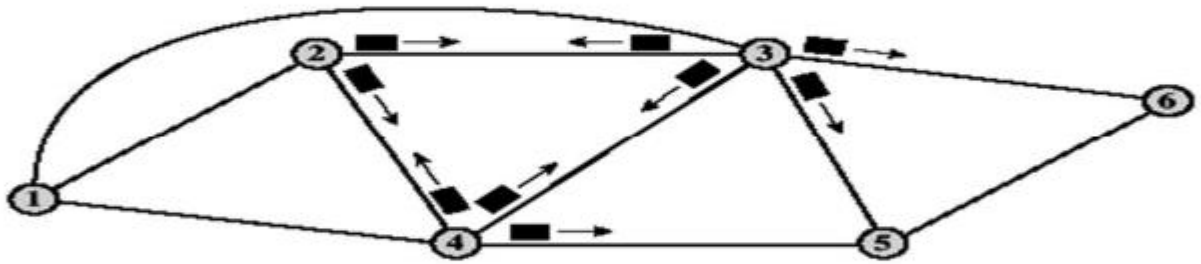


그림 10-5 플러딩(flooding) 알고리즘

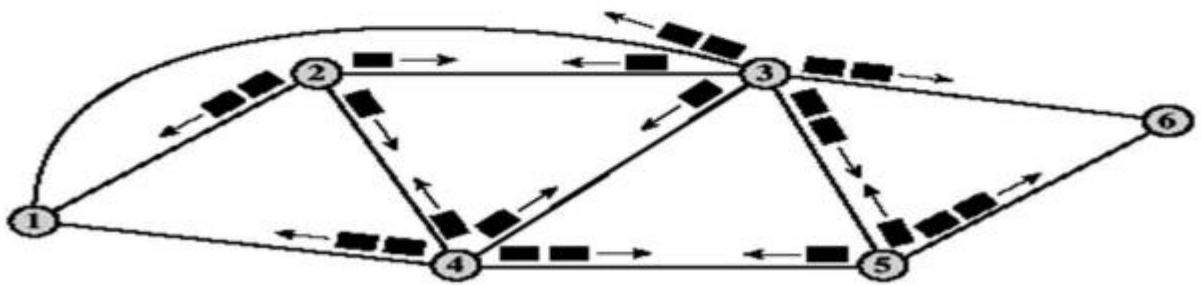
Flooding Example



(a) First hop



(b) Second hop



(c) Third hop

SECTION 02

동적 라우팅 알고리즘

1. 거리벡터 라우팅(distance vector routing) 알고리즘

- **거리벡터 라우팅**distance vector routing은 알파넷 개발 초기에 많이 사용하던 동적 알고리즘.
  - 각 라우터들은 주기적으로 라우팅 테이블을 주고 받는데, 테이블에는 자신의 기준에서 다른 라우터까지 가는데 걸리는 시간이 명시되어 있음. 라우터까지의 거리에 대한 연속적인 값(벡터)이기 때문에 거리벡터 라우팅.
  - 자신만의 라우팅 테이블을 만들고, 이를 다시 주변 라우터에게 전송하는 방식.

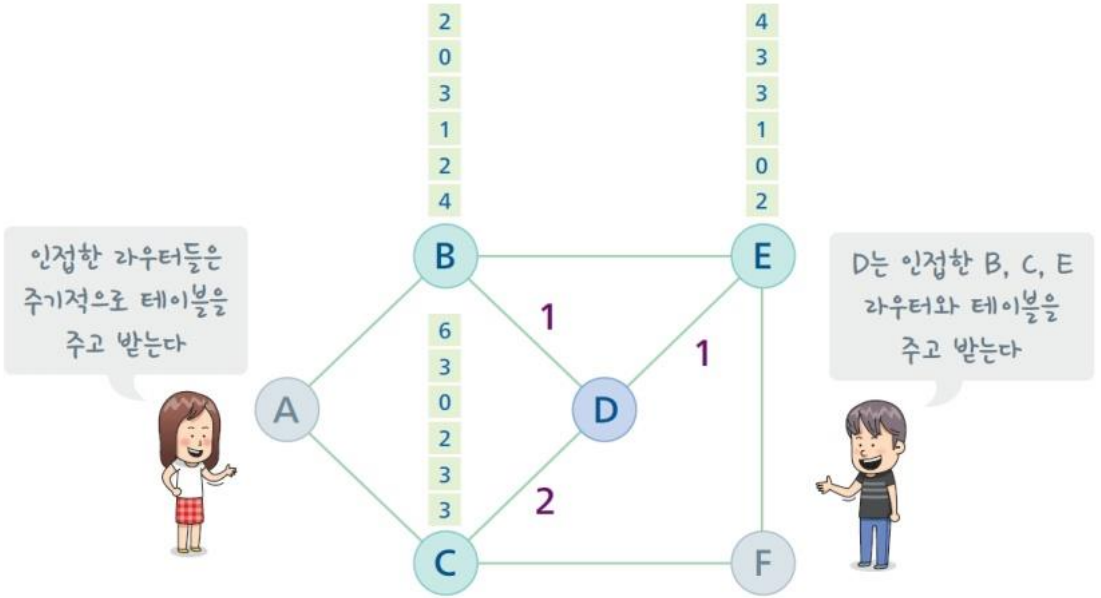


그림 10-6 거리벡터(distance vector) 라우팅 알고리즘

동적 라우팅 알고리즘

- B 테이블은 B 라우터가 생각하는 다른 라우터까지의 시간을 명시.
- 'D까지의 걸린 시간'에서 +1의 의미는 B 라우터가 D에게 테이블을 보내어 도착할 때까지 1msec가 걸렸다는 의미. C는 2msec, E는 1msec가 걸렸음 -> **D까지 걸린 시간이 추가되어서 계산.**
- A줄의 시간 중 가장 작은 시간을 선택 -> B테이블의 2를 골라서 D의 테이블에 A줄에는 (3, B)라 적는다. 이는 A에게 보낼 패킷이 있다면 B 라우터에게 보내고, 시간은 3msec가 걸린다는 의미.
- A에서 F까지 같은 방법으로 계산.

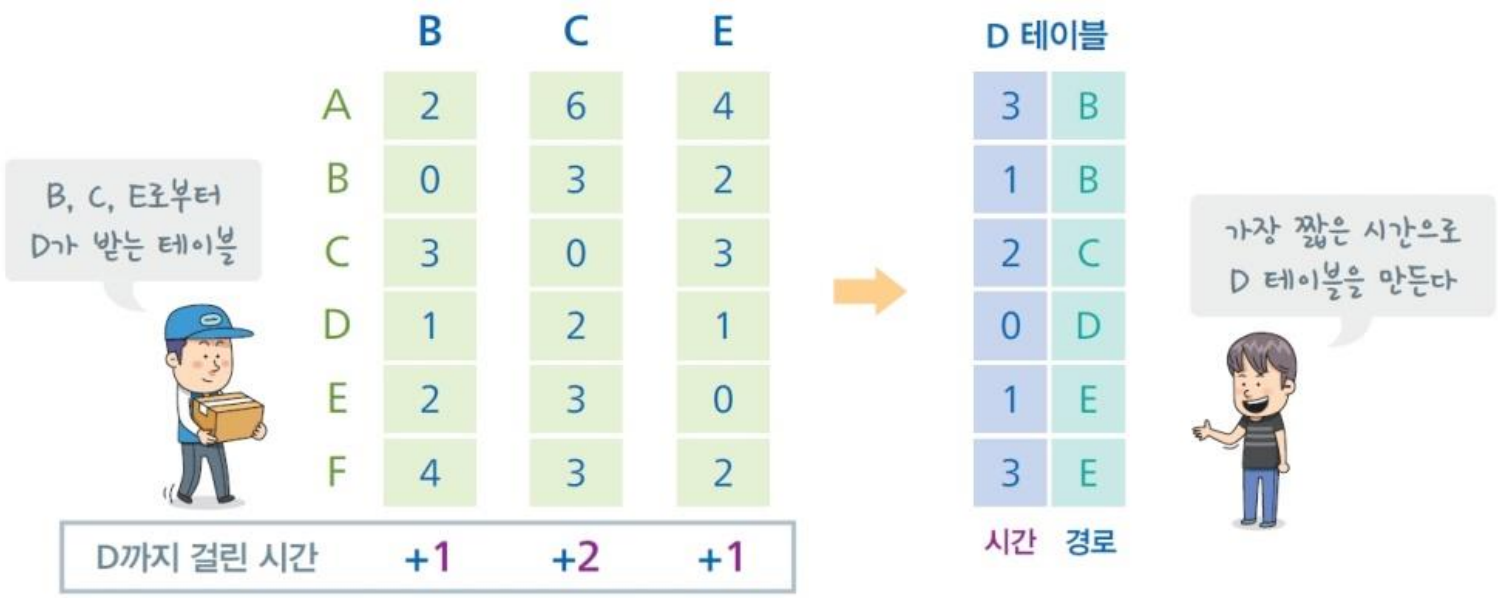


그림 10-7 거리벡터 테이블 만드는 법

동적 라우팅 알고리즘

- 무한 숫자세기count-to-infinite 문제.
  - 한 줄로 연결된 라우터가 있고 각각 패킷을 보내는데 1msec 만큼 걸린다고 가정 -> 좋은 뉴스는 문제없이 전달 됨.
  - A 라우터가 고장 나는 경우 -> B 라우터는 A로부터 테이블을 받지 못했는데, C 라우터로부터 받은 테이블을 보니 A = 2, B = 1, C = 0, D = 1이라 써 있음 -> B는 C와 A가 다른 길로 연결되어 있다고 착각 -> A = (3, C)라고 적음.
  - 이렇게 잘못된 정보가 계속 전달됨 -> 시간은 늘어나지만 A가 고장났다는 사실을 모름.

그림 10-8 무한 숫자세기(count-to-infinite) 문제

동적 라우팅 알고리즘

2. 연결상태 라우팅(link state routing) 알고리즘

- 거리벡터 라우팅의 문제를 개선한 알고리즘이 **연결상태 라우팅**link state routing
- 연결상태 라우팅은 자신에게 연결된 라우터 정보만을 보내고, 최단경로 알고리즘을 사용.
- 일련번호sequence number와 나이age를 추가하여 잘못된 정보가 도착하는 것을 막고, 특정 라우터가 고장나는 것을 확인할 수 있도록 하였음.

① 인접한 라우터(이웃 라우터)들 파악

자신 이웃 라우터neighbor router(자신과 직접 연결된 라우터)가 무엇인지를 파악하는 것.

라우터가 처음 시작되면 자신의 주변에 HELLO 패킷을 보내고, HELLO 패킷을 받은 라우터는 ACK 패킷을 보냄으로서 서로 연결된 것을 확인. 또한 HELLO 패킷이 전송된 시간을 계산.

표 10-3 연결상태 라우팅 알고리즘

① 인접한 라우터(이웃 라우터)들 파악
② 라우팅 테이블을 주기적으로 모든 라우터에게 보냄(플러딩)
③ 최단경로 알고리즘을 사용하여 라우팅 테이블을 만듦
④ 라우터 구조와 라우팅 테이블 값을 지속적으로 업데이트

동적 라우팅 알고리즘

② 라우팅 테이블을 주기적으로 모든 라우터에게 보냄 (플러딩).

이웃 라우터들이 파악된 후 라우터들은 주기적으로 테이블을 주고 받음. 연결상태 라우팅 알고리즘에서 주고받는 테이블에는 라우터 이름, 일련번호, 나이, 자신에게 연결된 이웃 라우터의 거리 정보가 적혀있음.

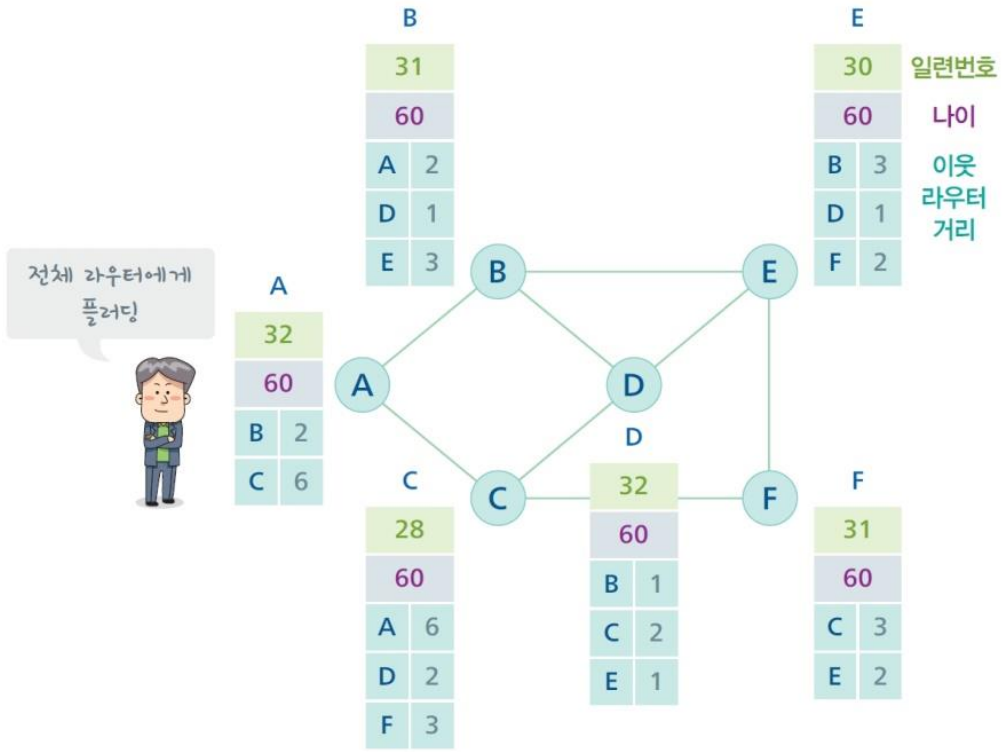


그림 10-9 연결상태(link state) 라우팅에서 전송되는 테이블

동적 라우팅 알고리즘

③ 최단경로 알고리즘을 사용하여 라우팅 테이블 만듬

연결상태 라우팅에서 모든 라우터들로부터의 정보가 모아지면, 각 라우터들은 최단경로 알고리즘을 사용하여 모든 노드까지 가는데 걸리는 최소의 시간을 계산하여 라우팅 테이블을 만듬.

- 관리 테이블의 일련번호는 해당 라우터로부터 가장 최근에 받은 테이블의 일련번호를 의미 -> 늦게 도착한 일련번호 테이블은 폐기.
- 나이는 다음 테이블의 도착예정시간 -> 패킷이 도착한 후에 나이가 계속 줄어들다가 0이 될 때까지 다음 테이블이 도착하지 않으면 해당 라우터가 고장난 것으로 판단 -> 일련번호도 0으로 만듦.

	일련번호	나이	SEND flag	ACK flag
A	32	60		
B	31	60		
C	28	59		
E	30	58		
F	31	60		

D 라우터의 관리 테이블

그림 10-10 연결상태 라우팅의 관리 테이블

동적 라우팅 알고리즘

3. 계층적 라우팅

- 라우터들은 계층 구조를 가짐 -> 계층적 라우팅(hierarchical routing).
  - 일정한 지역을 하나씩 묶어서 대표 라우터를 선정 -> 지역끼리 패킷을 주고받을 때에는 대표 라우터끼리 통신. 지역 안에서는 각자 지역에서 알아서 라우팅 -> 지역 안에 속한 라우터들은 내부 라우터라 부르고, 지역끼리 통신하는 라우터를 외부 라우터라 부름.
  - 내부 라우터의 경우 시작점에서 목적지까지 가장 빠르게 패킷을 전송할 수 있도록 라우팅 테이블을 유지, 외부 라우터는 도달 가능성을 확인하는 것이 가장 중요한 목표.

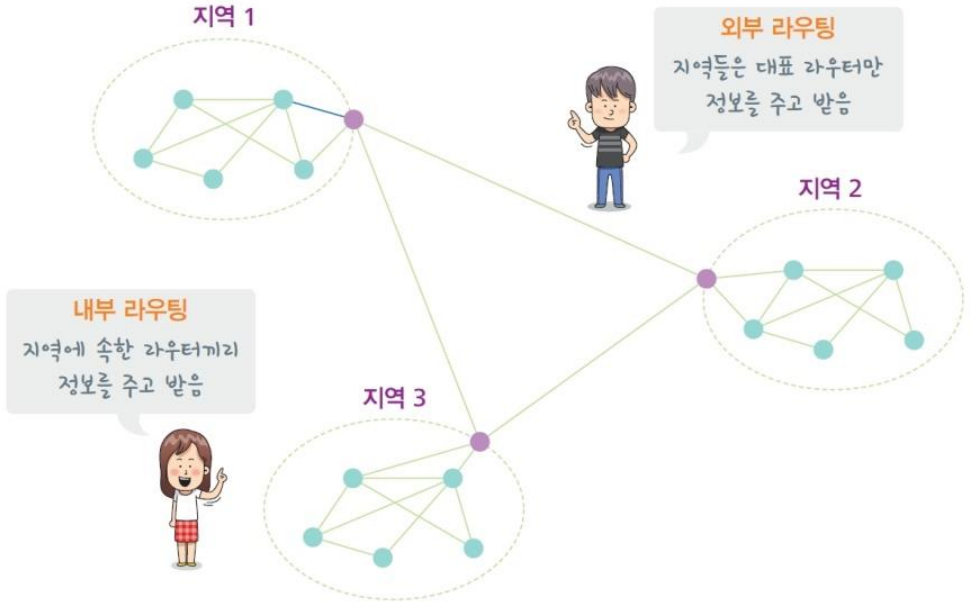


그림 10-11 계층적 라우팅 알고리즘

동적 라우팅 알고리즘

4. 거리벡터와 연결상태 라우팅 알고리즘 비교

표 10-4 거리벡터와 연결상태 알고리즘 비교

	거리벡터 알고리즘	연결상태 알고리즘
경로설정 방식	거리와 방향	전체 구조를 고려함
거리계산 방식	라우터 거리 합산	전체 라우터 최단경로 계산
사용 알고리즘	벨만-포드(여기서 잠깐 참조)	다익스트라 최단경로
테이블 전달 범위	인접	전체 라우터
테이블 전달 주기	일정 주기	변화 발생 시
테이블 정보	라우팅 테이블	링크 상태 테이블
테이블 전송 정보	전체 테이블	변경된 일부 테이블
장점	알고리즘 간단 구성이 단순 쉽게 구현 가능	네트워크 변화 빠르게 반영 네트워크 트래픽 적음 무한루프 발생 안함 대규모 네트워크에 적합
단점	네트워크 변화가 느리게 반영 무한루프 발생가능 주기적으로 많은 트래픽 발생 대규모 네트워크 부적합	CPU 부하 높음 계산에 많은 시간과 메모리 소모
사용 범위	외부 라우팅 프로토콜	내부 라우팅 프로토콜
주요 프로토콜	BGP	OSPF