**컴네 3장**

**연결 관리**

데이터를 교환 하기 전에 -> 핸드쉐이킹 발생

ESTAB -> 설정된 연결된 이라는 뜻

2단계 핸드쉐이크의 문제점:

지연 변동

메시지 손실로 인한 재전송

메시지 순서 변경

상대방의 상태를 정확히 파악 불가능

**3단계 핸드 쉐이크**

1단계: (->) SYNbit = 1, Seq=x

2단계: (<-) SYNbit = 1, Seq=y

(<-) ACKbit = 1, ACKnum=x+1

3단계: (->) ACKbit = 1, ACKnum=y+1

**혼잡 제어**

혼잡: 여러 발신지에서 **많은 데이터를 전송**하는 것

-> 패킷 손실, 패킷 지연이 나타날 수 있음 -> 이를 제어하는 것이 혼잡 제어

기존의 데이터와 함께 송신자의 재전송까지 합쳐지게 되면

-> 더욱 람다in은 커질 것이다. -> 손실 발생

또한 목적지에 잘 도착한 패킷이라도 특정 시간 내에 회신이 없으면

-> 재전송 실시 -> 불필요한 재전송

또한 여러 개의 경로에서 한 라우터를 놓고 경쟁

-> 불평등한 처리율로 인해 버려진 패킷은 처리율 0

-> 패킷이 버려질 경우 그 지점까지 전송된 패킷에 할당된 용량 역시 버려짐

종단간의 혼잡 제어: 혼잡 제어에 대한 종단간의 접근법 -> 타임 아웃 또는 3중 중복 확인을 -> 혼잡 발생으로 간주

네트워크 지원 혼잡 제어: 네트워크 계층 구성 요소가 직접 송신자에게 자신의 혼잡 사태를 피드백 하는 것

-> 이 두가지 방법 중 하나로 혼잡 제어 한다.

-> **TCP**는 이 두 방법 중 **종단간 혼잡 제어를 사용**한다.

-> 혼잡이 감지되지 않을 때 전송률을 높이고 혼잡이 감지되면 전송률 낮춤

cwnd(혼잡 윈도우, congestion window): TCP의 혼잡 제어를 위한 추가적인 변수

->혼잡이 감지 안되면 cwnd를 늘려가다가 혼잡 감지시 cnwd를 급격히 줄인다.

**혼잡 제어 방법**

슬로우 스타트

MSS: 최대 세그먼트 크기

RTT: 왕복 시간

전송률이 초기 -> cwnd = 1MMS로

-> 매 RTT마다 cwnd 2배로 증가

-> 전송률이 초기에는 낮게 시작하지만 점차 지수적으로 빠르게 증가

이때 타임 아웃에 의해 손실을 감지

-> 3개의 중복 ACK 수신을 손실로 인식

-> RENO: cwnd를 절반으로 줄임 -> 이후 선형으로 증가

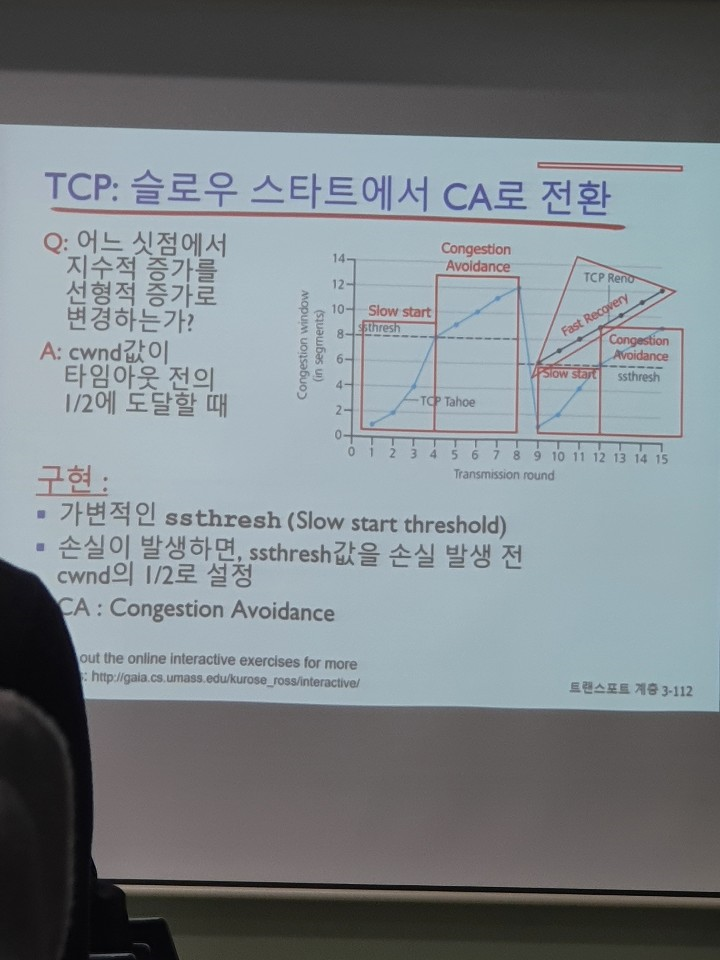
-> TAHOE: cwnd를 1로 설정하고 슬로우 스타트부터 다시 반복

* TCP가 휴식을 취하는 방법

알고리즘 : AIMD(가법적 증가/승법적 감소를 뜻함), Slow start, Fast Recovery, 선형적 증가

정책: Tahoe(AIMD, Slow start),

Reno(AIMD, Slow start, Fast Recovery)



ssthresh : 임계점, 임계치

임계점을 구현하는 방법:

Tahoe 진행 순서

슬로우 스타트 -> ssthresh 임계치(16의 절반) -> 선형적 증가 ->타임아웃-> **임계치를 혼잡 상황이 발생한 12의 절반으로 변경** -> cwnd 1MMS로 설정-> 슬로우 스타트 다시 반복

Reno 진행 순서

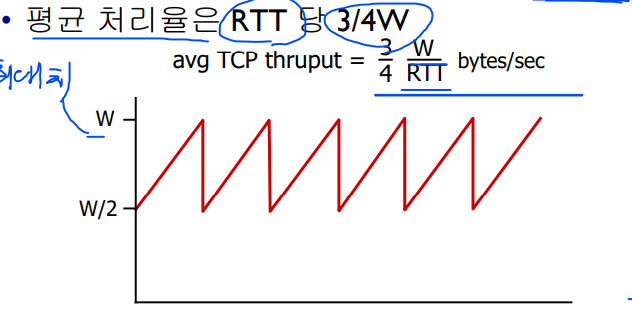
슬로우 스타트 -> ssthresh -> 선형적 증가 -> 타임아웃 -> 타임아웃 당시 cwnd의 절반

-> 다시 선형적 증가

**TCP 처리율**

윈도우 크기와 RTT함수로서 평균적인 TCP 성능 구하기

-> RTT가 작을수록 W가 클수록 평균 처리율이 높아짐



가법적 증가: 손실 감지될 떄까지 매 RTT 마다 1 MMS 단위로 cwnd 증가

승법적 감소: 손실 발생 되면 cwnd를 절반으로 감소

**4장**

**라우팅과 포워딩**

포워딩: **라우터 입력 포트에 도착한 패킷을 해당 라우터의 적절한 출력 포트로 이동 시키는 것** (하드웨어에서 짧은 시간에 실행)

라우팅: 패킷 단위로 출발지부터 목적지까지 경로를 결정 (소프트웨어에서 상대적으로 긴시간에 실행) -> 이 과정에서 포워딩 테이블이 생성됨

**둘 중에 더 중요한 것은 라우팅**

데이터 영역: 포워딩

제어 영역: 라우팅

전통 적인 라우팅 방법: 라우터에서 직접 제어영역에서 커뮤니케이션 하여 **로컬 포워딩 테이블**을 작성하여 사용한다. -> 라우터에 부하가 생긴다.

-

중앙 집중 화된 제어 영역: 라우터 들이 존재하고 원격 제어 장치로부터 로컬 플로우 테이블 받는다. -> 라우터에 부하를 덜어준다.

-> 원격 제어 장치는 제어 장치에 존재

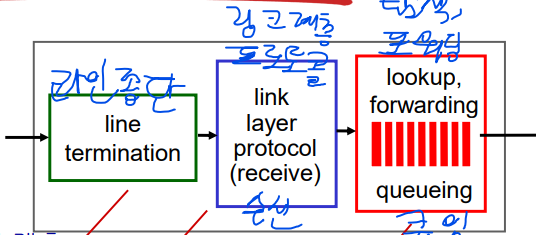
* **원격 제어 장치를 사용하는 이유**
* **-> 전통적 방식은 끊어진 정보를 알고 우회로를 찾는데 시간이 매우 많이 걸리기 때문에**
* **-> 긴급 상황이 발생했을 때 더 유연하게 대처 가능**
* 잊어버리면 안되는 회사명 -> cisco
* 화웨이 -> 중국의 시스코

**라우터 내부 구조**



**라우터 입력 포트, 고속 스위칭 패브릭, 라우터 출력 포트가 있다.**

**라우터 입력 포트**



-> 라인 종단, 링크계층 프로토콜, **탐색 포워딩**으로 구성

**탐색 포워딩에서 큐잉 존재**

->큐잉: 도착하는 데이터그램의 수가 스위치 패브릭으로 전달되는 수 보다 많을 때

분산 스위칭: **헤더 필드 값을 사용**하여 입력 포트 메모리에 존재하는 **포워딩 테이블을 참조**하여 **출력 포트를 찾는 것**

**-> 목적지 기반 포워딩: 전통적 방식, 목적지 IP 기반으로 포워딩**

**-> 일반적인 포워딩: 헤더 필드 값의 설정 내용을 기반으로 다양한 포워딩**

**목적지 기반 포워딩**

-> 목적지를 정하고 출구를 정한다.

-> 이걸 빨리하기 위한 기술 -> **최장 프리픽스 대응 규칙** 4-16

**최장 프리픽스 대응 규칙**

주어진 목적지 주소에 대한 포워딩 테이블 엔트리를 찾을 때 목적지 주소와 가장 길게 매칭 되는 프리픽스를 선택하여 0,1,2,3 같이 매칭해주는 것

**스위칭 패브릭**

**스위칭 패브릭**: 입력 버퍼의 패킷을 출력 버퍼로 전달해주는 역할

-> 메모리, 버스, 크로스 바의 3가지 유형

**메모리 경유 스위칭**

**->** 1세대 라우터, 가장 단순

-> **CPU가 제어**하는 전통적 컴퓨터

-> **패킷은 시스템 메모리에 복사**

-> **메모리 대역폭으로 속도 제한**

**버스를 통한 스위칭**

-> **입력포트**에 도착한 패킷이 **공유 버스를** 경유해서 **출력 포트 메모리**로 이동

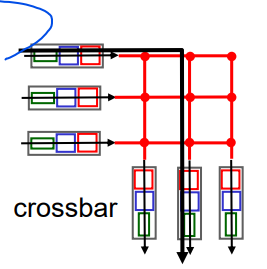
-**버스 충돌**

-> 버스 속도에 제한, 대역폭에 의해서 스위칭 속도가 제한될 수 있다.

-> 패킷이 경쟁하면 -> 여러 명 일수록 속도는 떨어진다.

**크로스바를 통한 스위칭**

-> 멀티 프로세서를 연결하기 위해 개발된 banyan networks, crossbar 등 다양한 연결 방법 사용



**스위칭 패브릭**

**스위칭 패브릭은 입력포트보다 속도가 느리다.**

-> **입력 버퍼에서 오버 플로우로 인한 큐잉 지연과 손실이 발생**

**-> HOL** **블로킹 (Head-of-the-Line , HOL): 큐 전단에서 위치한 데이터 그램이 큐 안의 다른 데이터그램의 이동을 제한하는 것**

**출력 포트(매우 중요)**

**-> 큐잉 지연은 입력포트와 출력포트에서 발생한다.**

**패킷 버퍼 자리에 온후 여기에서 기다린다. -> 아무도 없으면 바로 가지만 만약 도착했는데 여러 사람이 기다린다. -> 기다리다가 -> 순번이 오면 간다.**

**-> 여기서 기다리는 시간이 큐잉 지연**

**여러 개의 데이터들이 패킷 버퍼에 모인다 나가는 출구는 하나**

**-> 병목 현상 발생 -> 버퍼링 필요**

**도착한 것들 중에 누구를 먼저 보낼 것인가 -> 정책 -> 스케줄링 규칙 -> 최대 성능, 네트워크 중립성 감안**

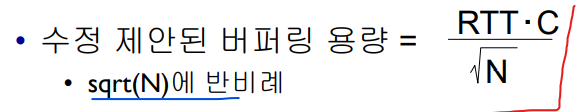
**출력 포트 큐잉**

스위치 경유 도착율이 출력 회선 속도보다 높을 경우 -> 버퍼링 생긴다.

(출력 포트에서 나가는 거보다 들어오는 게 많으면 버퍼링이 생긴다는 뜻)

**출력 포트 버퍼 오버 플로우로 큐잉 지연과 손실 발생**

**라우터 송신 버퍼링 용량 산정**

****

**(1개 회선 내에서 N개의 TCP흐름이 통과할 경우), C(회선 용량)**

**스케줄링 메커니즘**

**-> 폐기 정책이 여러가지 존재**

**FIFO 스케쥴링: 큐에 도착하는 순서대로 전송**

꼬리 자르기 -> 최근에 도착한 패킷을 폐기

**우선순위**: 우선 순위에 따라 버리거나 삭제 (주로 우선 순위가 많음)

임의: 임의로 선택하여 버리거나 삭제

**스케줄링 정책: 우선순위 스케줄링**

**최고 우선 순위 패킷을 먼저 전송하는 것 -> 단 우선순위 높은 패킷이 들어와도 우선순위 낮은 패킷이 전송 중일 때에는 대기한다.**

**라운드 로빈 스케줄링: 패킷을 클래스로 나눈다. ->클래스 간에 서비스를 번 갈아서 제공**

**가중치 공정 큐잉 WFQ: (가장 일반적인 방법) 라운드 로빈과 유사**

**-> 라운드 로빈처럼 각 클래스로 나눈 후 각 클래스마다 다른 양의 서비스 시간(가중치)를 부여한다.**

**IPv4**

헤더 길이: 대체로 IPv4 데이터 그램 헤더는 20바이트

**생존 시간(time to live)** -> 앞에 라우터가 몇 개 있다고 가정할 때->라우터를 거칠 때마다 하나씩 숫자를 깐다 -> **패킷이 갈 수 있는 거리를 제한한다.** -> **패킷이 갈 수 있는 거리를 말한다(데이터 그램이 무한히 순환하는 것을 막기 위함)**

헤더 채크섬: IP 데이터 그램의 오류를 탐지

**IP 단편화(조각내기), 재결합**

**-> 큰 하나가 작은 3개로 분할된 걸 단편화 라고 한다.**

**-> 하나로 완성하는 것이 재결합**

**-> 왜 갈라지는가? MTU 때문에**

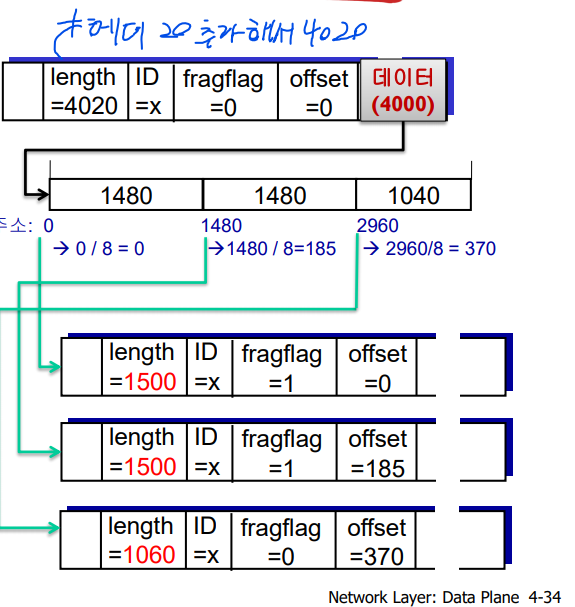
**-> MTU는 통신사가 보낼 수 있는 데이터의 최대 길이를 정한 것**

**특이한거 -> 라우터를 통해 분할하지만 -> 라우터에서 재결합을 못하게 하였다.**

**-> 라우터는 매우 바빠서 부담을 덜어주기 위해서 -> 컴퓨터가 재결합을 책임진다.**

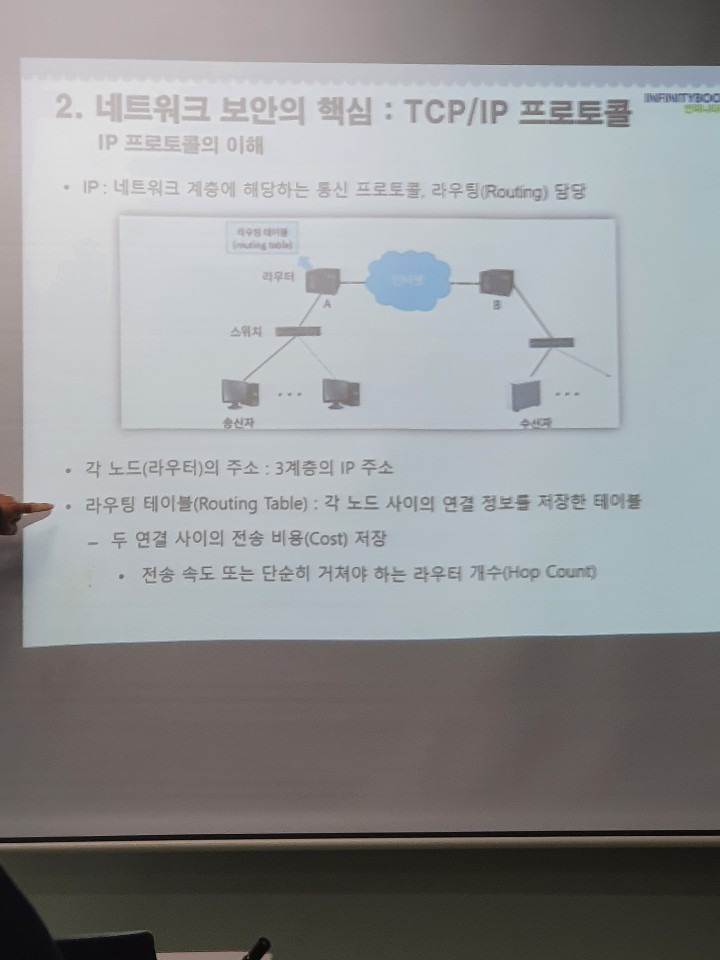
**MTU 1500 -> 보내는 쪽에서는 이 숫자를 맞춰서 보내야 한다.**

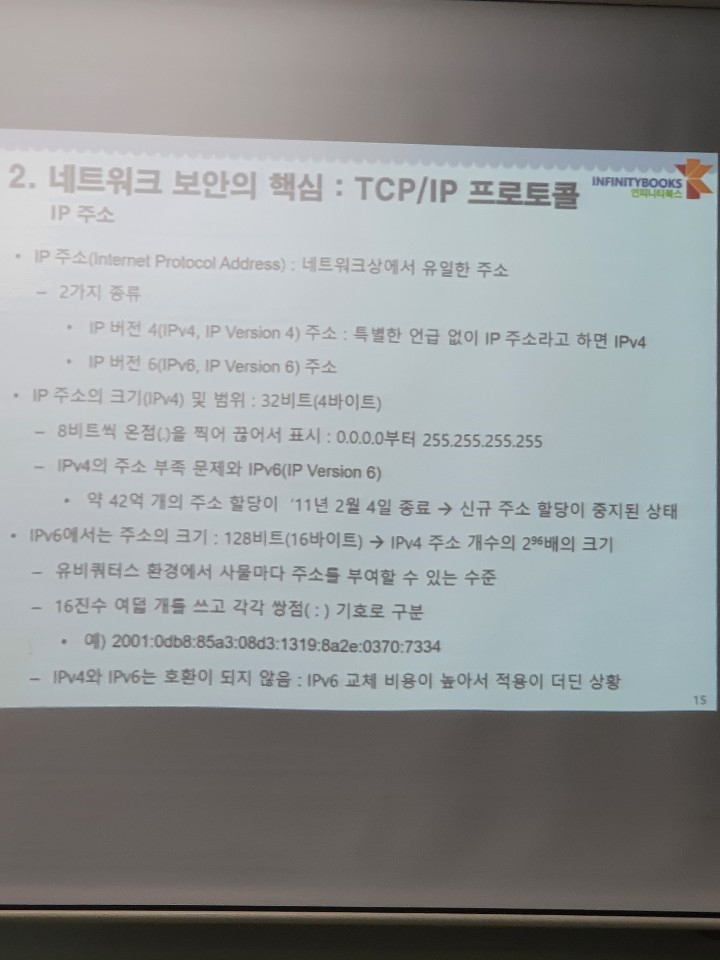
**-> 아니면 짤리기 때문**

****

**8로 나누는 이유 -> fragment offset 값**

**IP 주소**

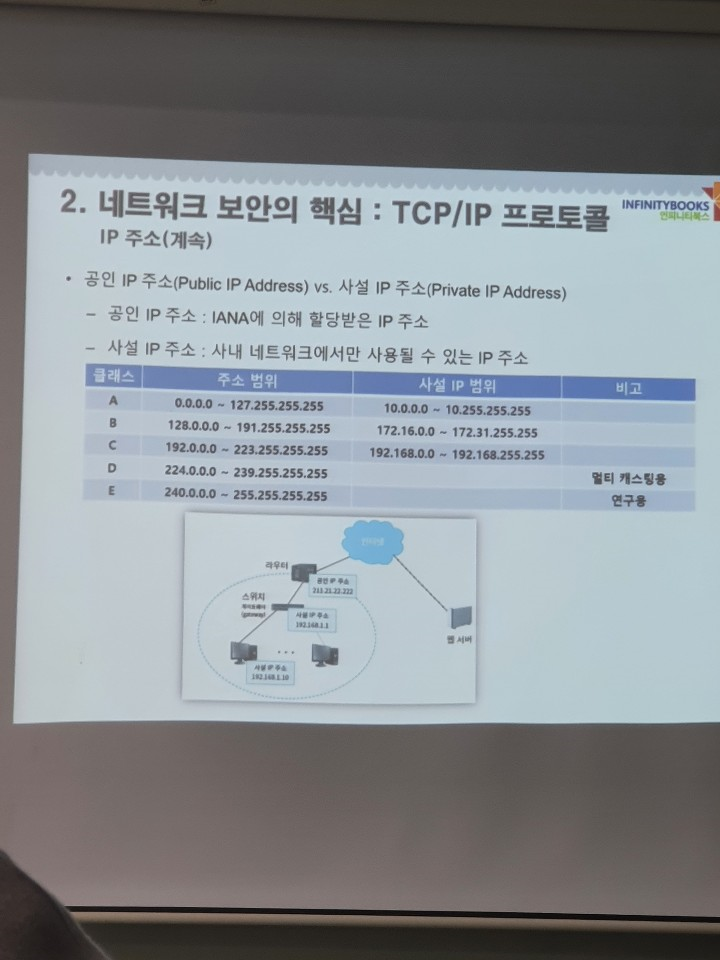
****



**IPv6으로 전환이 어려운 이유**

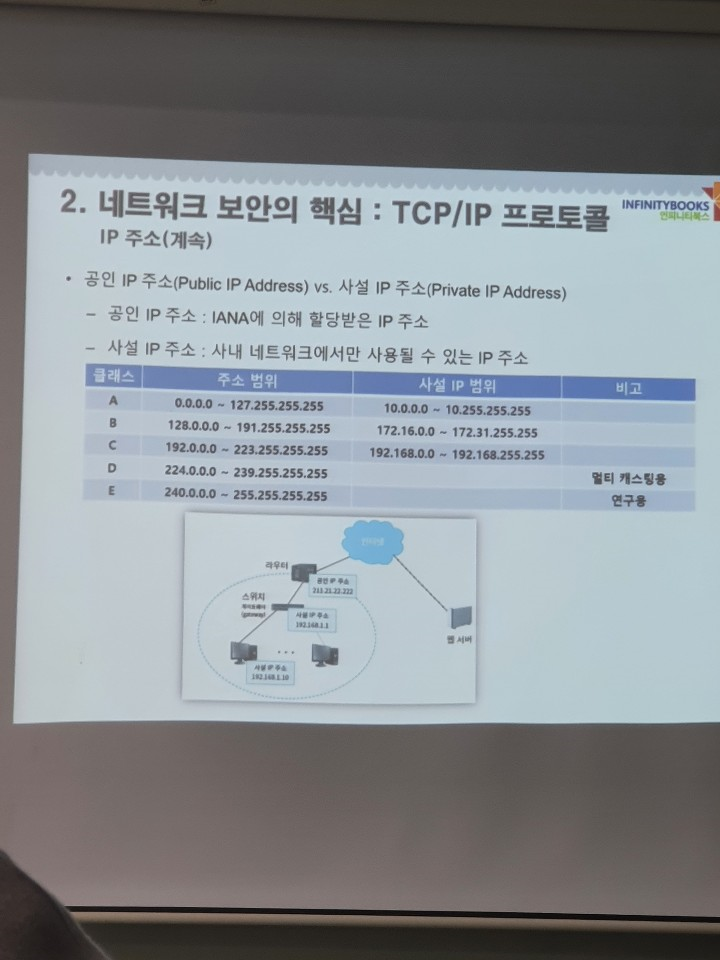
**->초창기 라우터들은 v4로만 설정되는 건데 v6이 오면 버리게 되어 버린다.**

**->이 장치들을 다 걷어 내기엔 돈이 너무 들어서 못 바꾸고 있다.**

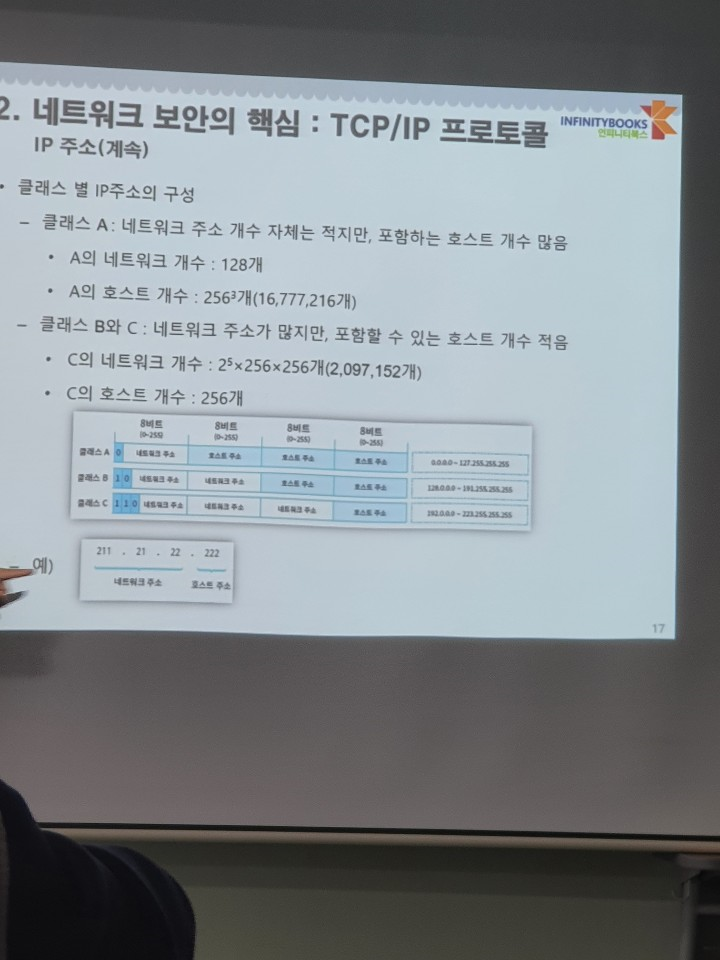
****

**클래스의 의미 -> 주소가 있는데 주소의 달라붙을 수 있는 컴퓨터의 개수**

**클래스 ABC 범위는 암기 할 것 A: 0~127/B: 128~191/C: 192~223**

****

* **공인 주소 -> 외부에서 돈주고 사온 고정 주소**
* **-> 보통 서버들은 공인 주소를 쓴다.**
* **-> 아래의 라우터는 외부에 연결 되어있어야 해서 공인주소를 가짐**
* **-> 안쪽에는 사설 주소를 가짐**
* **사설 주소 -> 자동 DHCP 프로토콜에 의해서 사설 주소 사용**
* **내 노트북이 만약에 저 원 밖에 있다면 난 사설주소 이기 때문에 라우터에서 컷 당한다.(사설 주소의 한계)**
* **만약 공인 주소라면 들어오기 가능**

****

**192.168.142.23 (클래스 C의 경우)**

**-> 앞에 3자리는 네트워크 주소**

**-> 뒤에 1자리가 호스트 주소**

**-> c 클래스**

**-> 192.168.142가 네트워크 주소**

**-> 23이 호스트 주소**

**0~127 128~191 192~223**

**사설 주소**

**첫째로 주소의클래스 파악**

**둘째로 어디까지가 네트워크, 어디까지가**

**호스트 주소인지 알아야 한다.**

**(A-> 1/3,B-> 2/2,C-> 3/1)(네트워크 주소/호스트 주소)**

* **무선 -> 특정한 프로토콜을 주고받아야 한다. DHCP**
* **서브넷: 호스트나 라우터에서 각 인터페이스를 분리하여 고립된 네트워크**

**어떨 때는 서브넷 주소만 해당되지만**

**어떨 땐 네트워크 주소 + 서브넷 주소까지 해당된다.**

* **끝자리가 0이면 -> 네트워크 주소**

**-> 0번을 부여받은 컴퓨터는 존재하지 않는다. 컴퓨터는 무조건 1부터 부여받는다.**

* **사이더 -> 서브넷 주소가 고정이 아닌 변동일 경우 표기 방법**

**DHCP**

* **동적 호스트 구성 프로토콜(DHCP)**
* **동적으로 임시IP를 할당해준다.**

**주소는 임대 -> 대부분 1시간**

**주소 재사용 가능**

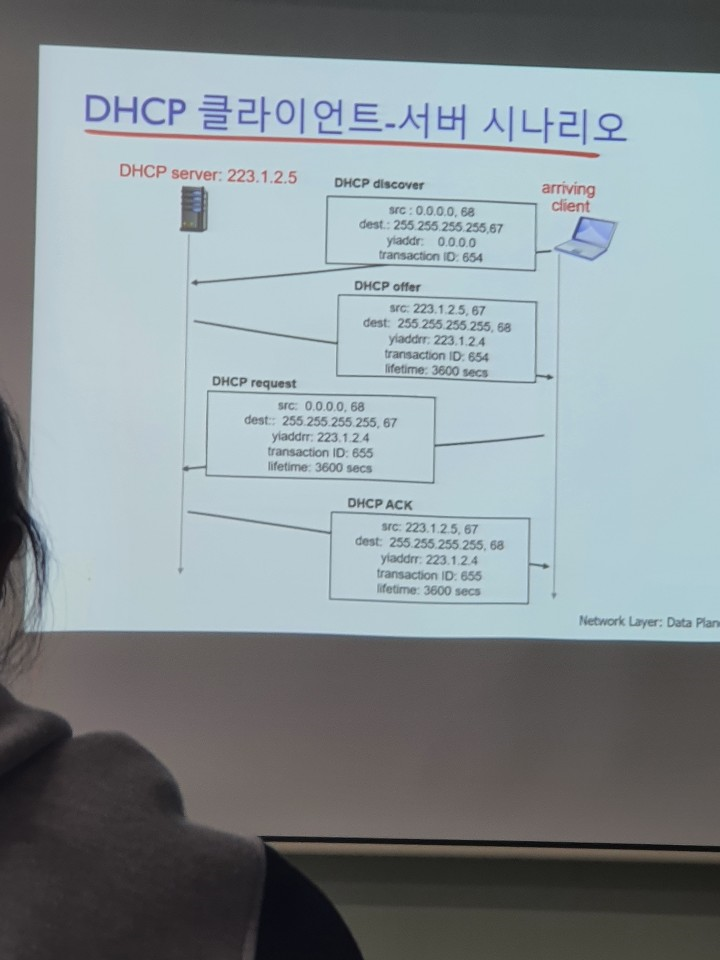
* **데스크탑은 고정을 쓴다 -> 동적보다 성능이 잘나와서**

**그 외의 서비스 중요**

**-> 클라이언트가 사용할 default gateway 제공**

**-> DNS 서버의 이름과 IP 주소 제공**

**-> 네트워크 마스크(네트워크 주소와 호스트 주소를 구분하기 위한 서브넷 마스크) 제공**

****

* **dest는 목적지**
* **3번째에는 yiaddrr이 바뀜**
* **그림 설명**

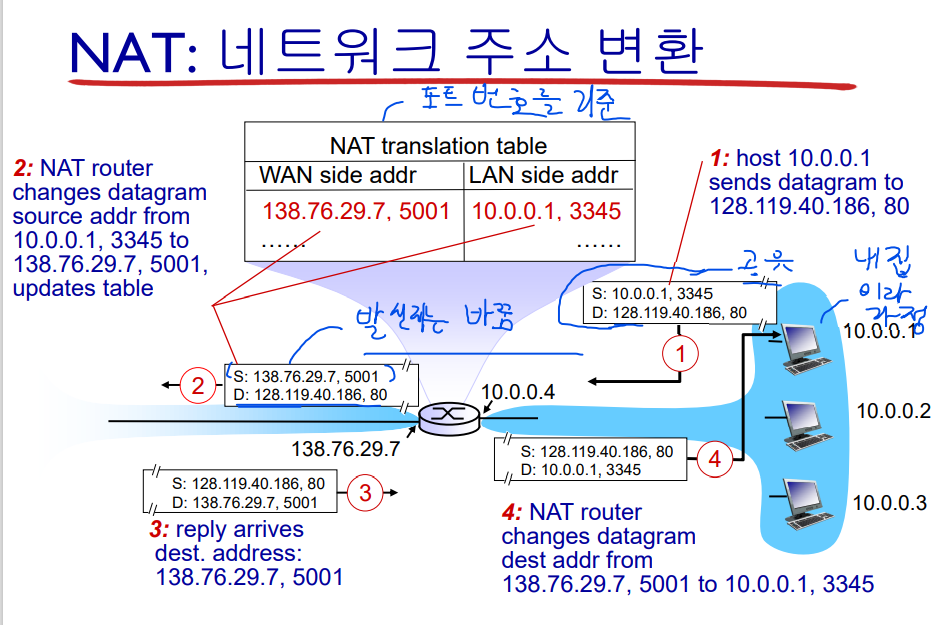
**1단계: DHCP 서버 발견 -> 호스트가 DHCP 발견 메시지를 브로드케스팅**

**2단계: DHCP 서버 제공 -> 발견 메시지를 받은 DHCP서버가 DHCP제공 메시지를 클라이언트로 응답 -> 이 역시 브로드케스팅**

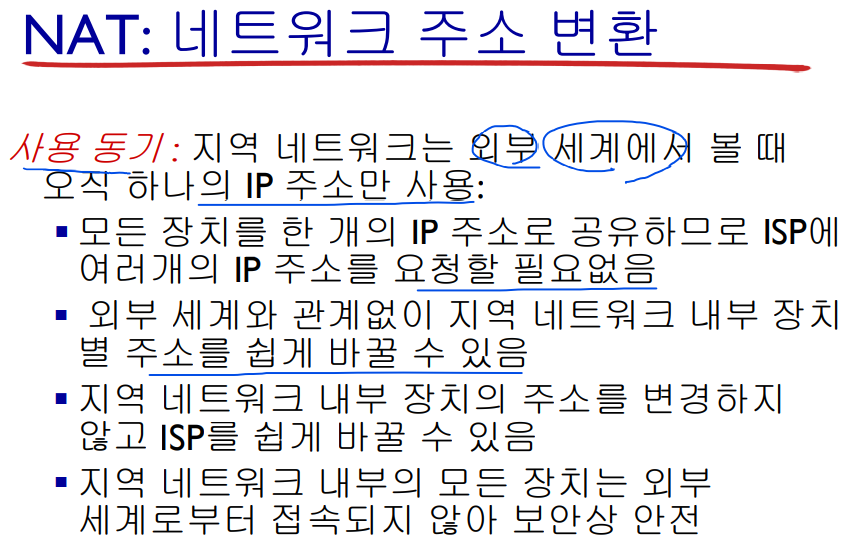
**3단계: DHCP 요청 -> 호스트가 선택한 서버 제공자에게 DHCP 요청 메시지**

**4단계: DHCP ACK -> DHCP ACK 메시지로 응답 -> IP주소 할당**

* **NAT -> 공인과 사설을 왔다 갔다(변환) 하는 기술**
* **사설IP 주소대역:** 사설IP 주소는 다음 3가지 주소대역으로 고정된다.
* Class A : **10**.0.0.0 ~ **10**.255.255.255
* Class B : **172.16**.0.0 ~ **172.31**.255.255
* Class C : **192.168**.0.0 ~ **192.168**.255.255

****

**10 00 설명 듣고 반드시 이해 및 외우기 8 00**

****

**IPv6**

* **v4는 옵션때문에 크기가 고정이 아님**

**-> v6은 고정으로 만듬**

* **V6는 40바이트르로 고정**
* **21 40 v6는 단편화 사용 금지**
* **터널링**

**-> IPv4 에서 IPv6로 전환하는 방법 -> 터널링**

**-> 비밀 얘기를 하고 싶을 때 보내기 전에 암호화를 한다.**

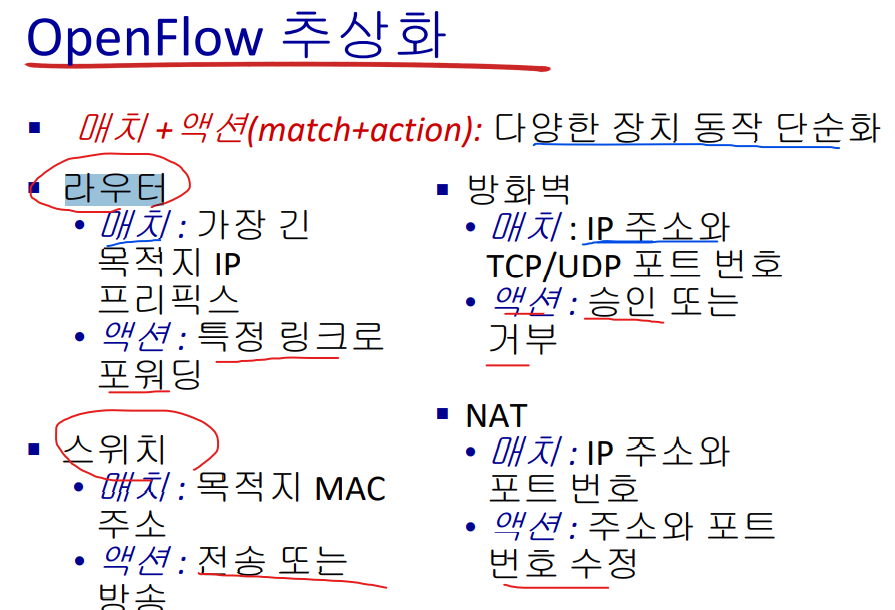
**SDN**

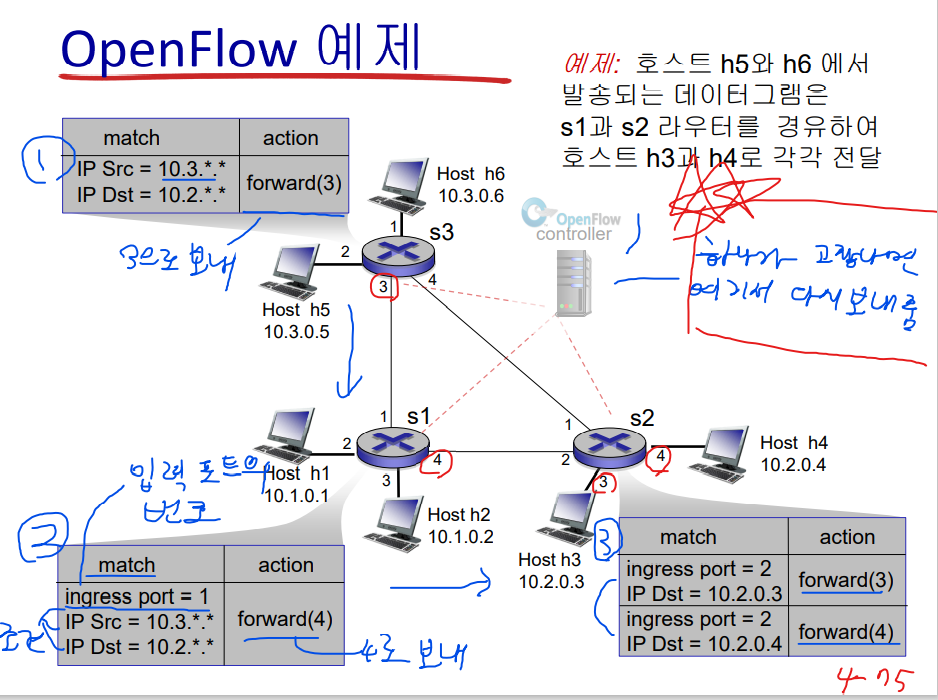
* **SDN -> 제어 영역과 데이터 영역으로(구분) 나눈 것**
* **제어 영역은 주로 라우팅, 데이터 영역은 위에서 완성된 로컬 flow table 활용해서 포워딩**
* **이렇게 프로토콜에 의해 생성된 테이블이 데이터 영역으로 내려오는 것 -> open flow (중요하니까 암기)**
* **원격 제어 장치를 통해 라우터는 로컬 플로우 테이블을 받는다.**
* **오픈 플로우 : 네트워크 컨트롤러가 스위치망을 통해 네트워크 패킷의 경로를 정의하는 SDN 프로토콜**

**오픈 플로우**

**매치: 들어오는 정보들의 조건**

**액션: 조건에 부합하면 하는 행동**

****

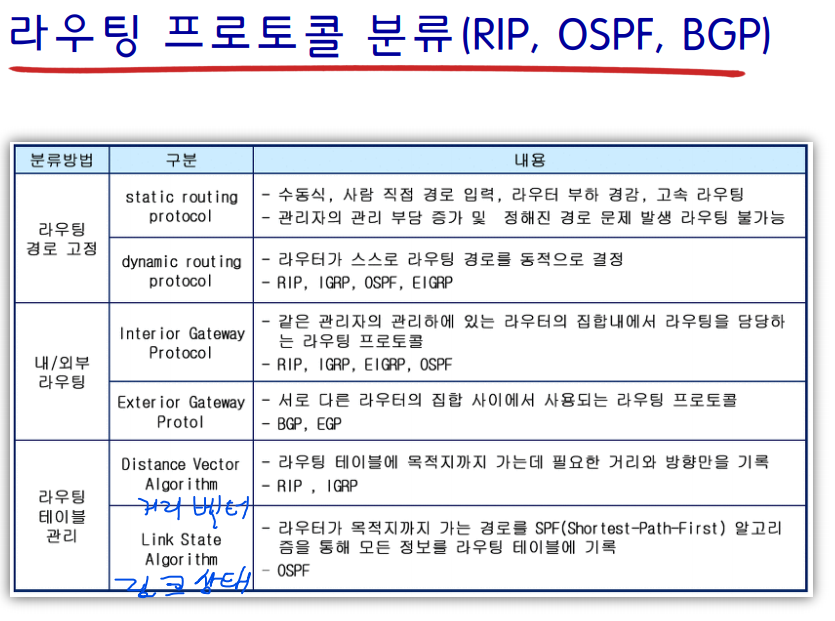
****

**5장**

**라우팅 프로토콜의 목적: 네트워크상의 라우터를 경유하여 출발지에서 목적지까지 갈 수 있는 최적의 경로를 결정하는 것**

**수신 윈도우 크기 -> 수신자가 송신자에게 내가 받을 수 있는 용량을 알려주는 것**

**반드시 기억해야 하는 3가지**

****

**알고리즘**

**중앙 집중형 알고리즘**

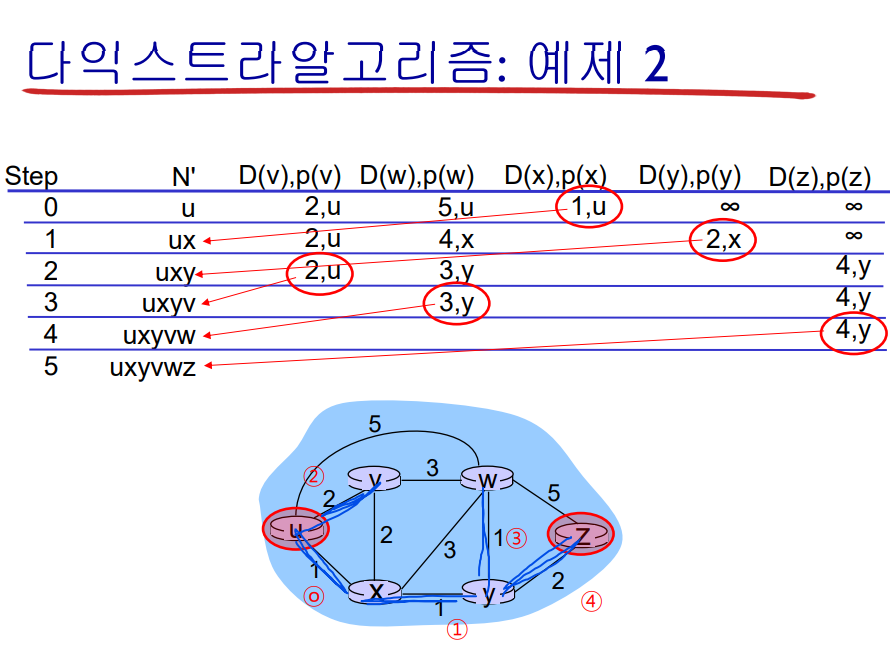
**-> 링크 상태 알고리즘**

**-> 링크 상태 패킷을 다른 모든 노드로 브로드캐스팅 한다.**

**다익스트라 알고리즘**

**-> 내 모든 다른 노드로의 최소 비용 경로를 계산**

**-> 시간 복잡도 최악 -> O(n제곱)**

****

**분산형 알고리즘**

**거리 벡터: 벨만 포드 알고리즘**

**링크 비용 변경 :**

**-> 노드는 연결 링크 비용 감소 감지**

**-> 라우팅 정보 수정하여 이웃 노드에 전달**

**재계산**

**-> 거리벡터 변경되면 이웃에 통보**

**-> 좋은 소식은 빠르게, 나쁜 소식은 느리게 전파(무한 계수 문제)**

**-> min{50, 6+1} = 7 -> 이 짓을 50 넘을 때까지 반복**

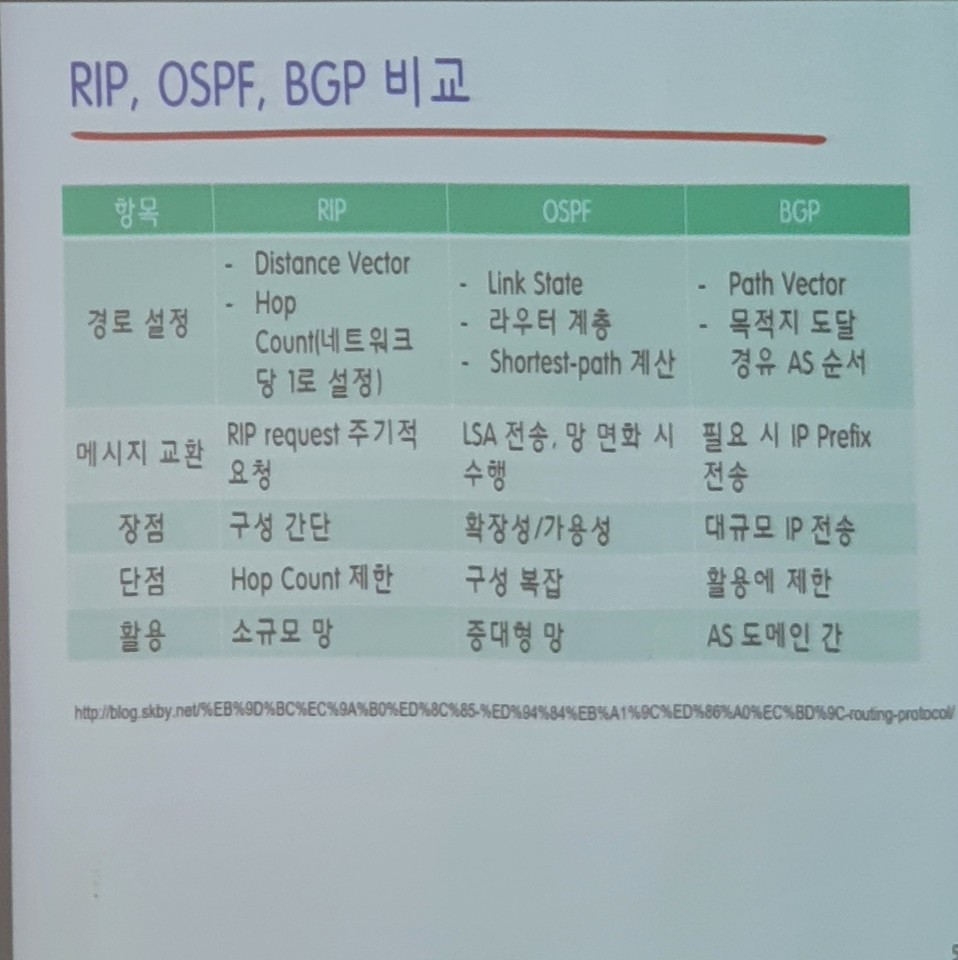
**-> 50의 숫자가 더 커진다면 무한 계수 문제 발생**

**-> 무한 계수 문제 해결 방안 -> 포이즌 리버스**

**-> 비용 높은 대로 가려고 할 때 그 거리를 무한대라고 통보하는 것**

**-> 세 개 이상의 노드를 포함한 루프는 감지 불가능하다는 단점 있음**

****

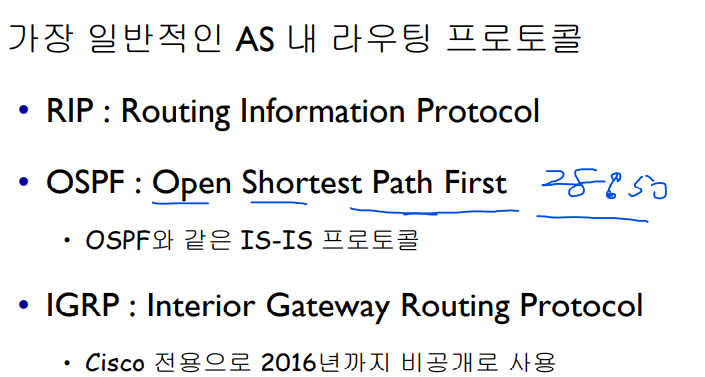
****

**자율 시스템(AS)**

**라우팅 프로토콜 -> 라우팅 테이블을 만들어 내는 작업이 중요**

**이때 AS 내 라우팅/ AS 간 라우팅 구분**

**AS 내 라우팅(IGP)**

****

**OSPF**

**-> AS의 라우터들을 상중하로 나누어 역할을 분리시켜 체계적으로 관리하는 프로토콜**

**-> 공개적으로 사용 가능**

**-> 링크 상태 알고리즘 사용함**

**-> 다익스트라 알고리즘 사용**

**-> 라우터는 OSPF링크 상태 정보를 AS내 모든 라우터에 플러딩 한다.**

**플러딩(flooding)(영어로 홍수를 뜻함)**

**-> 하나를 받아서 다른 주변에 똑같이 다 전달하는 것**

**OSPF 개선 사항**

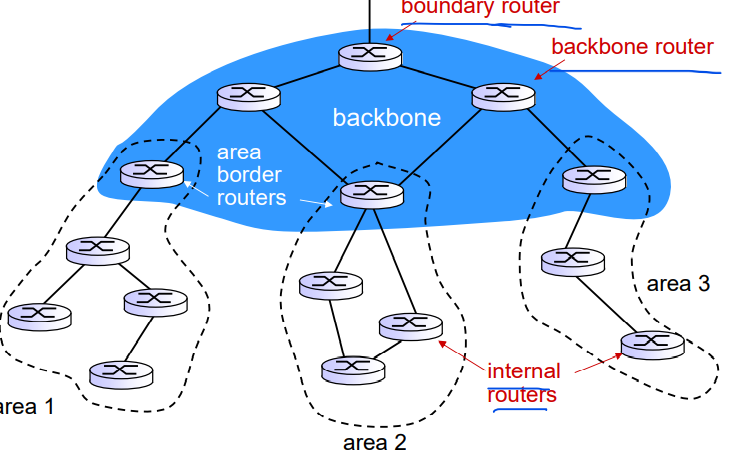
**-> 보안: 보안을 위해 모든 메시지 인증됨**

**-> 다수의 동일한 비용 경로 허용**

**-> 유니 및 멀티 캐스트 통합 지원 -> 라우터는 브로드 케스트는 지원 안함**

**통신 할 때 1대 1로(유니 케스트) 할 경우, 1대 무한대(브로드 캐스트, (방송)), 1 대 n(멀티 케스트)으로 할 때가 있다. 이때 n은 그룹원들만 해당**

**백본 영역과 지역 영역**

****

**영역 경계 라우터 -> 영역 내 네트워크 정보를 요약하여 다른 as라우터로 방송**

**백본 라우터 -> 라우터들의 중추, 여기가 망가지면 심각한 오류**

**-> 백본에 한하여 OSPF 라우팅 실행**

**경계 라우터 -> 다른 AS와 연결**

**AS 내 라우팅**

**-> AS내 모든 라우터는 동일한 라우팅 프로토콜 수행**

**-> 다른 AS 간에는 다른 라우팅 프로토콜 수행**

**-> 게이트웨이 라우터 : AS의 끝단에 위치하여 다른 AS의 라우터와 연결한다.**

**-> OSPF 대표적인 AS 내부 라우팅 프로토콜**

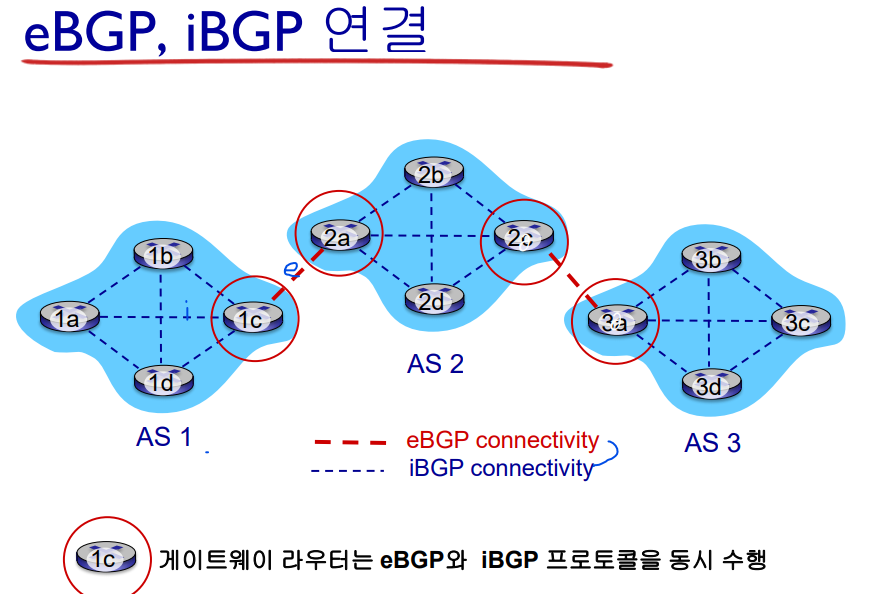
**AS간 라우팅**

**-> BGP -> AS 간 라우팅 프로토콜**

**-> 게이트웨이 라우터가 AS 내 라우팅 뿐만 아니라 도메인간 라우팅을 실행**

**AS간 라우팅 -> BGP**

**도메인 간 라우팅 프로토콜의 사실상 표준**

****