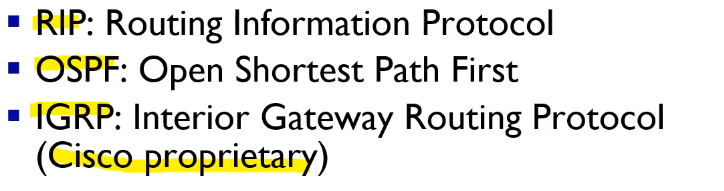
# 9주차 - Chapter 6

### 6.8.1 Routing in the Internet: RIP

p.6-4

RIP는 Intra-AS로, AS 내부에서 사용하는 라우팅 프로토콜이다. RIP는 Cisco에서 이름을 붙인 것으로, interior gateway protocols (IGP) 라고도 부른다. Intra-AS에는 세가지가 있음.

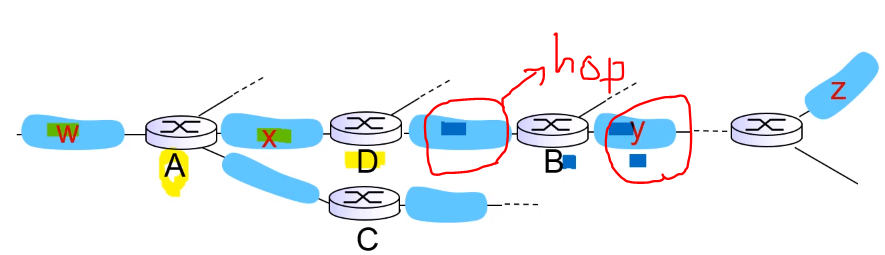


p.6-5

RIP는 distance vector algorithm을 기반으로한 프로토콜임. distance metric은 최대 15개의 hop을 거친다. distance vector는 30초에 한번씩 인접 node로 보내지는데, 이것을 advertisement라고 부른다. 각각의 advertisement는 최대 25개의 목적지 서브넷을 가진다.

p.6-6

D노드가 가지고있는 라우팅 테이블 예시이다. w에 가기 위해서는 A노드를 거쳐야하고, hop의 수는 2개이다. y에 가기 위해서는 B를 거쳐야하고, hop의 수가 2개이다. z에 가기 위해서는 B를 거쳐야하고, hop의 수는 7개이다 (중간에 생략되어있음).



p.6-7

이러한 상태에서, A가 D에게 advertisement를 보냈다고 가정하자. A입장에서 z를 가기위해서는 C를 거쳐야하고 4개의 hop을 지나야한다는 정보가 있다. 이러한 정보를 D에게 알려주는 것이다 (advertisement 하는것).

다시 라우팅 테이블 D로 돌아와서, z로 가는 방법을 A에게 전달받은 정보로 업데이트 해준다. 기존에 B를 거치면 7개의 홉스를 지났는데, A를 거쳐서 가니 5번만에 가게됨.

p.6-8

중간에 어떤 link가 고장이 났다면?

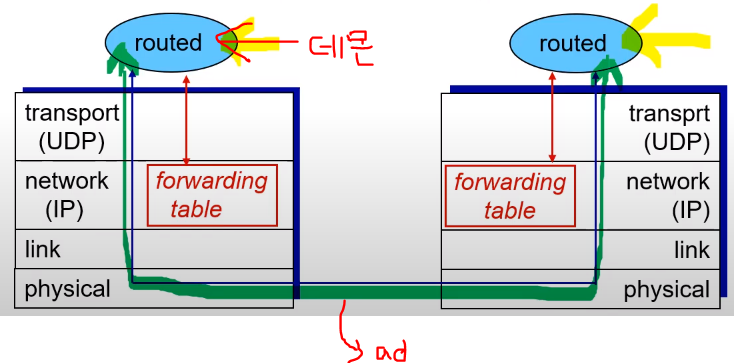
인접하는 노드로부터 180초 동안 advertisement가 오지 않으면 인접하는 link(또는 router)가 죽었다고 판단하고, 새로운 distance vector값을 만들어서 나머지 인접하는 노드들에게 보내주고, 나머지 인접한 노드들은 이 정보들을 새롭게 업데이트한다. 이렇게되면, 해당 라우터가 고장이 났다는 사실이 모든 네트워크에 퍼진다.

p.6-9

라우팅 테이블에서 RIP가 어떻게 돌아갈까? (RIP table 처리방법에 대해)

forwarding table을 업데이트시키기 위한 라우터 어플리케이션을 ‘daemon’이라고 부른다.

모든 라우터끼리 주기적으로 advertisement를 보낸다.



그때의 전송 프로토콜은 UDP가 담당한다. 주기적으로 업데이트되는 distance vector 정보들을 IP계층에 forwarding table로 내려준다.

### 6.8.2 routing in the Internet: OSPF

p.6-10

OSPF는 open(공개된) 프로토콜이며, link state 알고리즘을 기반으로 하는 프로토콜이다.

라우터내의 모든 AS들로 advertisement가 전달되어 flooded 되었다고 한다. OSPF message는 Layer 4를 거치지 않고, Layer 3위에 바로 올라가게 된다. 보안측면에서 가짜 메시지가 전달되지 않도록 방지하고 있기도 한다.

p.6-11

w에서 y로 가는 cost가 8이다. [w, y, 8] 이라는 정보를 모든 AS로 전파시킨다 flood!

u가 y로 가려면? x 또는 v, w를 거쳐서 갈 수있다. 이처럼 OSPF에서는 동일한 cost의 여러 경로가 있을 수 있다.

p.6-12

보통 패킷에서는 TCP와 UDP 세그먼트 위에 data가 실리게 되는데, OSPF에서 advertisement data는 IP 바로 위에 올라간다는 차이점.

### 6.8.3 routing in the Internet: BGP

p.6-13

BGP 프로토콜은 inter-AS 프로토콜이다. (AS간의 프로토콜이라는 것)

de facto 표준이면서 inter-domain 라우팅 프로토콜이다. (de facto는 사람들이 쓰다가 편해서 표준이된 것이라는 의미)

“glue that holds the Internet together” - 서로 떨어져있는 인터넷들을 연결하게 만드는 접착제 역할을 하는것이 BGP이다.

1. eBGP - 인접한 AS와 gateway 라우터 사이에서는 subnet reachability 정보를 얻기 위해 eBGP라는 세션이 만들어진다.
2. iBGP - eBGP세션을 통해 얻은 subnet reachability 정보를 자신의 AS 내부 라우터 모두에게 알려주는데, 이때 알려주는 경로를 iBGP세션 이라고 한다.
3. AS 내에있는 모든 internal 라우터들은 다른 네트워크로 가기 위해 경로를 결정하는데, subnet reachability과 정책적 판단에 따라 경로를 결정한다.

>> 이 세가지가 이루어지려면, 서브넷들은 가장 먼저 ‘I am here’라는 정보를 발생시켜서 서브넷에 인접한 모든 네트워크들에게 advertise 즉, 광고를 해야한다. 이 광고를 시작하는 것도 BGP가 주관하는 일이다.

p.6-14

# 10주차 - Chapter 6

### 6.9 Broadcast and multicast routing

p.6-4

패킷을 소스로부터 모든 다른 노드들에게 전달해주는 라우팅이다. 소스는 하나이고, 수신측은 여러가지 이다.

1. Source duplication - 소스에서 일일이 라우터의 목적지를 알고 전송하는 기법. 소스에서 duplication이 일어나는 방식이다.

문제점: 어떻게 소스가 모든 수신자의 주소를 알까? 사용하지 말자.

>> in-network duplication 방식이 솔루션이래.

1. in-network duplication - 처음 R1에서 한번만 보내주면, R2가 또 다른 라우터들에 전송하는 방식 즉, 네트워크에서 duplication이 일어나는 방식이다.

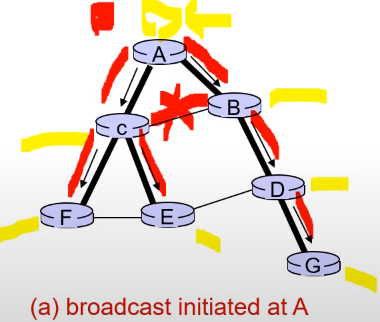
p.6-5

1. flooding 방식 - 어떤 노드가 브로드캐스트 패킷을 받으면 무조건 자기 인접으로 다시 복사해서 넘겨준다. 간단하지만, 사이클 및 broadcast storm 현상이 발생할 수 있다. 데이터 홍수.. 느낌
2. controlled flooding - 한번 받은 데이터는 받지 않는 방식이다. 패킷 id를 기억하고있다가, 과거에 이미 broadcast한 노드는 무시해버린다. 그리고 RPF라고 reverse path forwarding 방식이 있는데, 브로드캐스트 패킷이 들어오면, 그 패킷이 소스로부터 나에게 최단거리로 들어왔는지 검토하고 그럴경우에만 인접노드로 받아서 다음 노드에 넘겨준다.
3. spanning tree - RPF보다 intelligent하다. tree구조를 만들어서 좀더 간단히 구조화 시킨 방식. 소스로부터 모든 노드 각각에 spanning tree를 일일이 만들어야한다는 문제점이 있다.

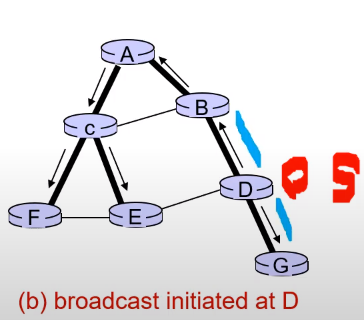
p.6-6

Spanning Tree

* 스패닝 트리를 만들자.



이 경우는 A가 소스노드인 경우이다. 스패닝 트리의 구조는 나무를 거꾸로 세워놓은 모양인데, 가지가 쭉쭉 뻗어나가는 모습. (볼드 선 !!) 예를들어 B와 C가 서로 연결될 수 없다.

D가 소스노드인 경우는 다음과 같다. 

p.6-7

Reverse path forwarding

현재 들어온 패킷이 들어오는데, 최단경로로 들어왔는지 검토한다. 만약 아니라면, datagram을 무시한다.

p.6-9

example

reverse path forwarding이 스패닝 트리보다는 쓸데없는 패킷을 잘 내보낸다.

p.6-10

pruning

어떻게 하면 reverse path 포워딩도 쓸데없는 패킷을 줄일 수 있을까 하는것이 pruning 기법이다. 소스로부터 방송을 듣기 원하는 노드가 없다는 prune 메시지를 보낸다. 결과적으로, prune이라는 것은 필요없는 정보들을 가지치기해서 없애는 과정을 말한다.

p.6-11

지역에 있는 각각의 라우터들은 특정한 멀티캐스트 참여를 원하는 호스트들이 있는지 알아내야한다. 특정한 멀티캐스트 그룹은 동일한 multicast address를 가지고 표현한다. 예를들어 class D classification은 모두 1110으로 시작한다.

p.6-12

IGMP는 로컬 라우터가 내가 관할하고있는 호스트중에서 특정한 멀티캐스트 방송을 수신하기 원하는 호스트가 있는지 파악하는 프로토콜이다.

* 각각의 로컬 라우터 입장에서는 자신이 관할하고있는 호스트들에게 query를 보낸다. 특정 멀티캐스트 참여를 원하는 호스트가 있는가 물어보는것. 그에대한 응답으로, IGMP Reply를 보낸다. header의 upper layer가 2라는 것은, 현재 이 패킷은 IGMP쿼리 혹은 리플라이가 실려있다는 뜻이다.

p.6-13

IGMP는 query와 reply방식으로 이루어진다.

* 로컬 라우터가 쿼리를 보내고, 호스트가 reply를 보내준다. 먼저 로컬라우터가 쿼리를 보내는데, 어떤 특정한 멀티캐스트 주소를 사용하길 원하는 호스트가 있는지 알기위해 보내는것이고, class D 주소를 포함하고 있다.
* 호스트는 그 특정한 멀티캐스트 주소를 내가 계속 사용하길 원하면 reply 메시지를 보낸다.
* 만약에 호스트가 더이상 특정 멀티캐스트 그룹에 속하지 않는다면 reply 메시지는 corresponding주소를 적어서 더이상 원하지 않는다고 전송한다.

p.6-14

* DVMRP는 한마디로, RPF이고 tree based 이다. RPF 방식은 flood와 prune방식을 합해놓은 것과 같다.
* initial datagram은 처음에 RPF를 통해서 flood가 이루어지게 된다.

p.6-15

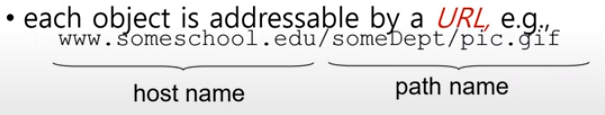
* DVMRP를 soft state라고 한다. DVMRP라우터는 prune 되었다는 사실을 1분만 기억하고, 그 후에는 잊어버린다.

# 11주차 - HTTP 어플리케이션 및 JAVA 웹서버 과제

### 9.1 Web and HTTP

p.4-2

* Web이란? : 컴퓨터를 통해 사람들이 정보를 공유할 수 있는 전세계적인 정보 공간을 말한다.
* 인터넷에서 HTML 프로토콜, 하이퍼텍스트, HTML형식 등을 사용하여 그림과 문자들을 교환하는 전송방식을 말하기도 한다.
* 주소는 URL로 표현한다.



p.4-3

HTTP overview

클라이언트와 서버 사이에 통신이 어떻게 이루어질까?

HTTP: hypertext transfer protocol (html로 이루어진 파일들을 어떻게 전송하는가에 대한 프로토콜)

* client: 크롬 같은 브라우저. http를 화면에 display하는 것이 목적이다.
* server: 고객이 요청한 request file에 대해 응답으로 보내주는 역할을 하고, 각 파일에 대한 url을 만들어서 디렉토리안에 차곡차곡 저장해둔다.

p.4-4

HTTP는 전달계층 프로토콜로 TCP를 이용한다.

클라이언트가 먼저 TCP를 요청한다.

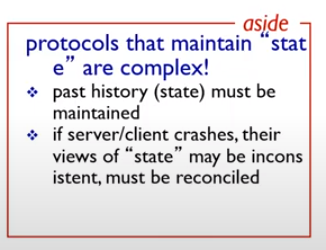
서버의 웹 어플리케이션은 port #80에서 항상 돌아가도록 규정되어있다.

클라이언트와 서버 사이에 HTTP 메시지를 서로 주고받는데, 이때 html문서를 주고받는다.

클라이언트와 서버 사이에 연결이 끊어지게 된다.

HTTP는 stateless다.

* 한번 TCP가 연결이 되었다가, 클라이언트가 요청하는 정보를 보내주고난 뒤에는, 바로 TCP connection을 끊어버리고, 서버는 그 전의 기록들을 절대로 남기지 않는다.



서버가 state를 가지고있으면 너무 복잡해지기 때문에 stateless인 것이다.

p.4-5

HTTP연결은 두가지 형태로 나누어진다.

1. non-persistent HTTP

* 하나의 object를 가져오기 위해서는 한번의 TCP를 연결하고, 그 연결을 끊는 과정이 필요하다.
* stateless 개념과 유사하다.

1. persistent HTTP

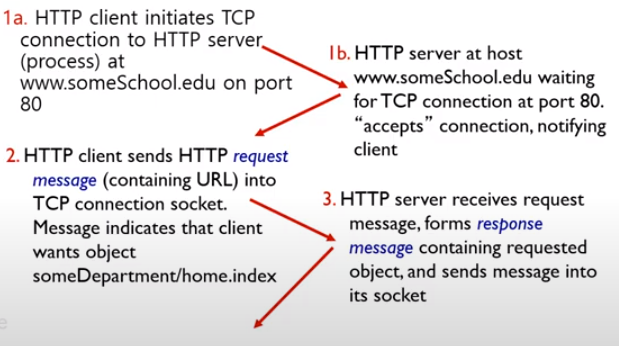
* 한번 TCP연결을 시켜놓고, 관련된 object들 여러개를 동일한 TCP connection을 통해 가져온다. 그리고, TCP 연결을 끊는다.

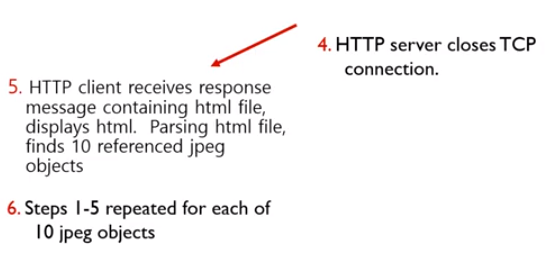
p.4-6

non-persistent HTTP

예시) URL이 [www.someSchool.edu](http://www.someschool.edu) 이고, 디렉토리가 somDepartment/home.index 인 HTTP를 가져오고자 한다. 이 안에는 10개의 이미지가 들어있다.



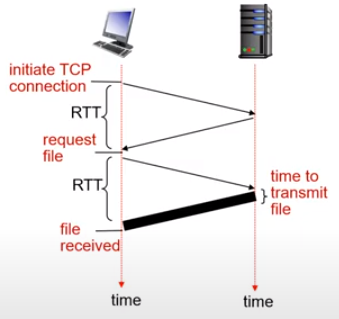




1. HTTP 클라이언트가 포트넘버 80번으로 TCP연결을 시도한다.
2. 서버가 TCP연결을 허가해주고, 클라이언트에 허가되었다고 알려준다.
3. 클라이언트는 url을 포함하는 HTTP요청을 TCP connection을 통해 보낸다.
4. HTTP request message를 수신한 서버 입장에서는 그에 맞는 file을 찾아서 응답을 하게되는 것이다.
5. TCP연결을 끊는다.
6. 클라이언트는 html파일로부터 그 안에 어떤 object가 들어있는지 찾는다.
7. 각 오브젝트들에 대해 TCP연결하고, HTTP 요청 메시지 주고받고 를 반복한다. (위의 1번부터 6번까지의 과정)

p.4-8

non-persistent HTTP 에서 file을 가져와 display하는데 까지 걸리는 시간은 얼마나 될까.



* 클라이언트가 서버에게 TCP연결을 시도한다.
* 서버가 TCP연결을 허가해준다.
* 이때, TCP연결을 위해 걸리는 시간을 RTT라고 한다. (Round Trip Time)
* TCP가 연결되면, HTTP request를 보내기 시작. 짧은 길이의 메시지다.
* 하지만, 서버가 보내는 메시지에는 파일이 들어있기 때문에 transmit file time 만큼의 시간이 더 걸리게 된다.
* 따라서 하나의 메시지를 수신하는데는 하나의 RTT와 파일 전송시간이 추가로 걸리며, 하나의 object파일을 전송받기 위해서는 2개의 RTT와 하나의 파일 전송시간이 소요된다.

p.4-9

non - persistent HTTP에서 하나의 object를 받기 위해 object당 2개의 RTT가 요구된다고 했다. OS입장에서는 object마다 각각의 TCP 커넥션을 처리하는데 있어 서버에 오버헤드가 생긴다.

persistent HTTP에서 서버 입장에서는 response를 보내고 나서도 계속해서 connection을 유지한다.

p.4-10

클라이언트에서 요청하는 HTTP request message구조

GET: 달라! / 디렉토리 / 버전 ….

p.4-11

URL이라는 것은 주소부분은 없다!

cr: 캐리지

lf: 라인피드..?

p.4-12

서버가 보내는 response message구조

클라이언트가 요청하는 것을 잘 찾았다면, 200을 보낸다.

Content-Type: 텍스트인지, 이미지, 영상인지가 여기에 들어있다.

p.4-13

HTTP response에서 중요한 status codes

200: 정상적으로 서버가 동작을 하면 200 ok

301: 다른곳으로 갔다 error

400: 잘못된 요청이다.

404: 파일이 없다

505: 버전이 맞지 않음

# 12주차

### 7.1 Constraints in high-speed IP forwarding

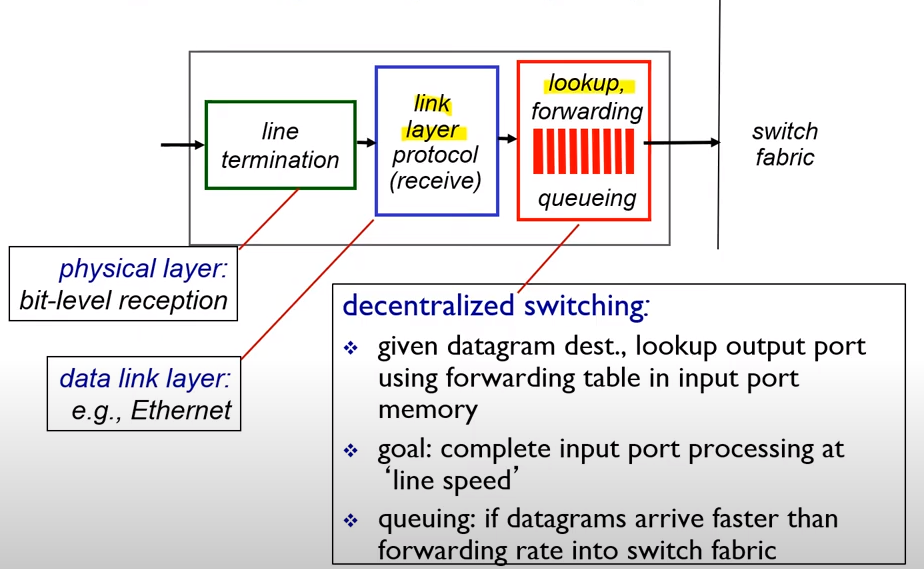
p.7-5

라우팅 테이블을 lookup하는 시간을 어떻게하면 줄일수 있을까가 가장 포인트이다.

queueing delay .. 이렇게 두가지 관점으로 인해 IP forwarding 기술이 나온것이라 생각하면 된다.

p.7-7

Queueing delay에서 입력포트 구조



들어오는(입력되는) 패킷마다 라우팅 테이블을 lookup하고, 원하는 방향으로 내보내는 과정을 반복한다.

p.7-8

Queueing delay에서 출력포트 구조

아웃풋 버퍼가 처리할 수 있는 데이터의 양보다 많은 트래픽이 몰리면, 큐잉이 발생할 수 밖에 없다.

p.7-9

virtual circuit n.w 복습

패킷이 목적지까지 거쳐가는 경로가 마치 전화망에서의 circuit switch 처럼 생겼다 하여 불리는 말이다.

p10

VC implementation

# 13주차

# 8.1 Transport - layer services

### Transport service & protocols

1. logical communication : 서로 다른 두개의 호스트 (브라우저와 웹서버) 사이에서 어플리케이션이 돌아갈 수 있도록 논리적인 커뮤니케이션을 제공한다는 의미.
2. transport 프로토콜은 end 시스템에서만 돌아간다.

* send side: 전송측. 메시지가 큰 경우, segment로 잘라서 네트워크 layer로 보내준다.
* rcv side: reassemble 세그먼트는 pass to application layer라고 한다. segment들을 다시 message로 재조합 하고 application layer로 보냄

1. 애플리케이션 들에게 제공이 가능한 transport 레이어는 internet에서는 TCP와 UDP가 있다.

### Transport vs network layer

1. network layer: logical 커뮤니케이션이다. logical 커뮤니케이션이란, 통신은 통신인데, 간접적인 통신방식을 말한다. IP주소를 가진 네트워크 계층을 호스트라고 한다. 정리하면, source 네트워크와 목적지 네트워크 계층 사이에서 logical 커뮤니케이션을 구성하는 것이 network layer이다.
2. transport layer: 전송계층 역시 logical 커뮤니케이션이고, 프로세스 간에 logical communication을 담당한다. 전송계층 logical 커뮤니케이션은 network layer logical communication에 의존한다.

### Internet transport-layer protocols

1. TCP

* reliable (신뢰가능한)
* in-order delivery (보내는 순서대로 목적지에 도착함)
* congestion control (네트워크 상황이 현재 얼마나 복잡한지 확인후, 제어하는 역할)
* flow control (수신측 입장을 고려하여 받을준비가 안되어있는 경우 조금 나중에 보내는 등의 제어역할)
* connection setup (connection하기 위해 수신측과 송신측 사이에서 setup하는것)

1. UDP

* reliable, in-order 둘다 해당사항 없다. TCP보다 하는일이 별로 없다.
* no-frills extension of ‘best-effort’ IP (최대한 성심성의껏 전달서비스를 해준다.)

# 8.2 Multiplexing and Demultiplexing

### Multiplexing / Demultiplexing

* socket과 port number의 관계: socket에 port number가 적혀있음.

1. P3 process

* transport header에는 source 포트넘버와, destination 포트넘버가 들어있다.
* network에는 버스가 달려있고, 버스에는 source IP주소와 destination IP주소가 들어있다.

1. P1, P2 process

* Multiplexing: sender측에서 일어나는 것으로 여러 socket에서 오는 데이터에 transport header를 추가하여 보낸다.
* Demultiplexing: segment에서의 header정보를 이용해 적절한 socket들로 보낸다.

### How demultiplexing works

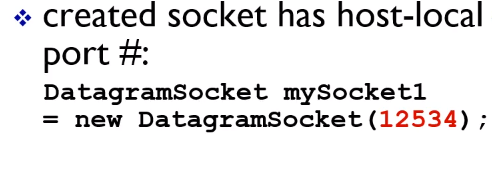
1. host receives IP datagrams

* 각 datagram은 source IP주소와, destination IP주소를 가지고있다.

1. host는 적절한 소캣에 세그먼트를 보내기위해서 IP주소와 포트넘버를 사용하는 것이다.

### Connectionless demultiplexing

1. 소캣을 생성하는 자바코드



1. 12534에 데이터를 보내고자 한다면?

* 목적지 IP주소와 목적지 포트넘버를 가지고 datagram을 만들어야 한다.

1. when host receives UDP segment: UDP demultiplexing (1대다 통신이 가능하다)

### Connection-oriented demultiplexing

* TCP demultiplexing은 connection oriented 방식이다.

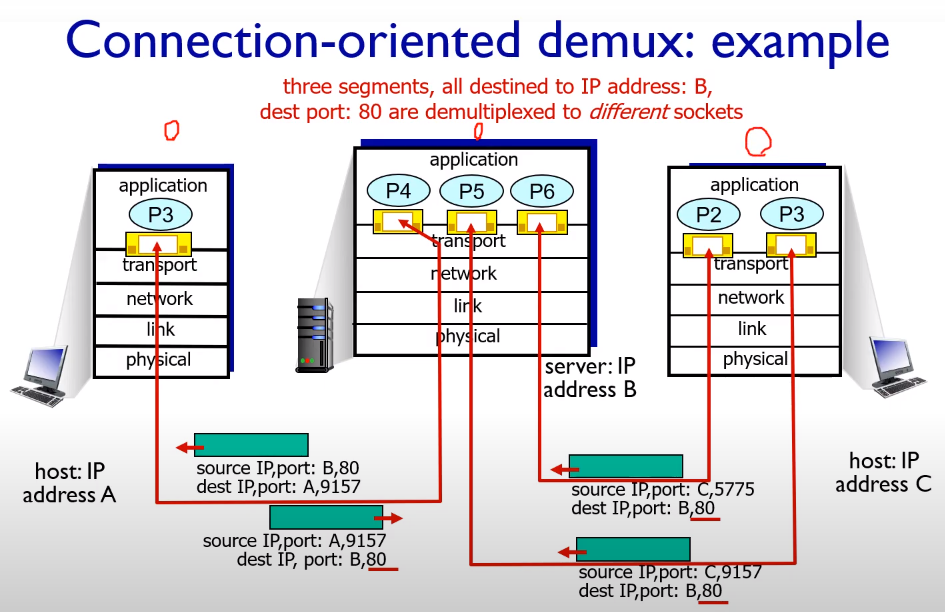
1. TCP socket: identified by 4-tuple (다음 4가지가 있어야 TCP socket이 만들어진다.)

* source IP address
* source port number
* dest IP address
* dest port number

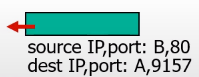
1. demux: 위의 네가지 정보가 맞아야 demultiplexing이 가능하다.
2. server host may support many simultaneous TCP sockets
3. web servers는 여러개의 서로다른 소캣을 만든다.

* non-persistent HTTP를 보면, 서버가 여러개의 서로다른 소캣을 가지고있어서 여러명에게 서비스가 가능한 것이다.

### TCP demux example

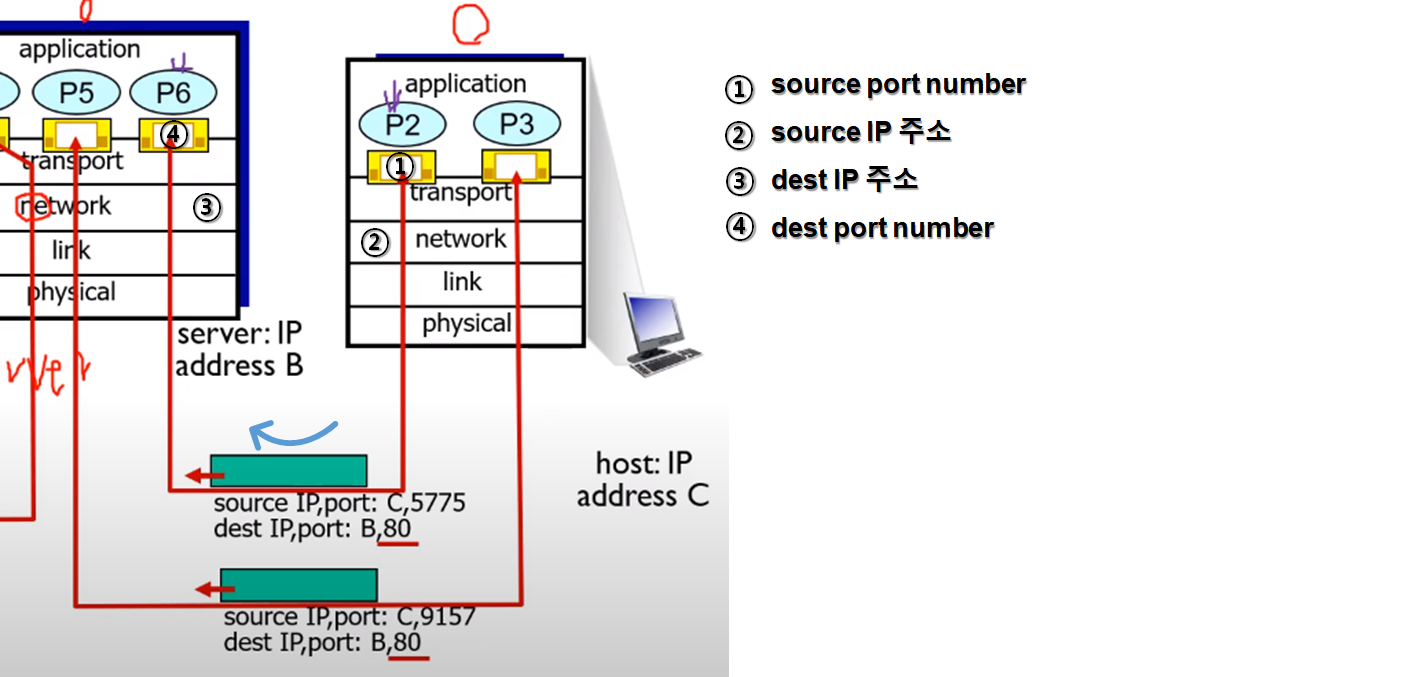


P4, P5, P6: server 담당



이처럼 source IP address, source port number, dest IP address, dest port number가 맞아떨어져야 소캣이 만들어지는 것이다.

TCP에서 demux가 일어나는 과정



# 

# 8.3 Connectionless transport: UDP

### UDP에 대해

1. no frills, bare bones(몸에 꼭맞는)인 인터넷 전송 프로토콜이다.
2. best effort service: UDP 세그먼트는 중간 라우터에서 분실될 수도있고, 순서에 맞지 않게 도착할 수도있는 서비스이다. 자신이 할수있는 수준에서 최대한으로 잘 서비스만 해주지, 앞서 발생하는 문제들에 대해 책임을 지지는 않는다.
3. connectionless이다.

* no handshaking
* UDP는 독립적으로 하나하나의 세그먼트를 처리한다.

1. UDP use:

* 멀티미디어 서비스에 많이 이용한다. (많은 신호를 보내는데 조금 분실되어도 괜찮아서! 이러한것을 loss tolerant 라고 한다.) (rate sensitive는 예를들어 음성데이터를 패킷으로 만들면 보내는쪽에서 20개의 패킷을 보낸다. 목적지에 잘 도착할지 여부를 고려하지 않고 무조건 초당 20개를 보낸다는 의미다.)
* DNS (domain name system): DNS는 단일 패킷으로 이루어진다. 보내고자하는 메시지가 패킷 하나에 들어간다. 이렇게 간단한 경우에 UDP를 사용하는 것이다.
* SNMP (simple network management protocol)

1. reliable transfer over UDP (UDP를 신뢰성있게 사용하는 방법)

* add reliability at application layer (응용계층에서 loss를 감지하고 다시 보낼수 있는 프로그램을 만들면 된다.)
* application-specific error recovery

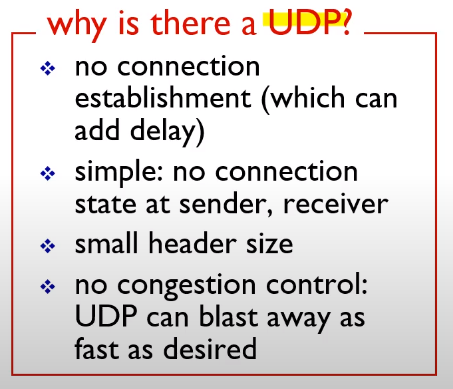
### UDP datagram from application (중요)

* 각각의 application은 UDP 전송계층과 접촉하게 되는데, 오직 하나의 UDP 데이터그램만 받는다.

Application이 오직 하나의 UDP datagram을 만든다.

네트워크 계층으로 내려준다. 이 3계층에서는 앞에 IP header를 붙인다. 여기서 중요한것은 application 데이터가 하나의 세그먼트가 된다는 것이다. 그게 그대로 3계층으로 내려와 또 하나의 패킷을 이룬다. 

### UDP: segment header

64bit로 이루어져있다. 

1. 연결이 없다
2. 간단하다
3. 헤더 사이즈가 작다
4. no congestion control : blast away이다 내가 원하는 만큼 데이터를 내보낸다는 의미!

### UDP checksum

목표: detect ‘errors’ in transmitted segment

1. sender입장:

* treat segment contents, including header fields, as sequence of 16-bit integers. (헤더필드를 포함하는 세그먼트 내용이 16비트 integer로 연결되어있는걸로 보고 여기서 체크섬을 구현해낸다..?)
* 체크섬: one’s complement를 구현해내고
* 그것을 UDP 체크섬필드의 체크섬 value에 집어넣어 목적지로 보낸다.

1. receiver입장:

* 받은 세그먼트에대해 체크섬 값을 계산한다. 이 계산된 값과 지금 도착한 체크섬필드의 값을 비교한다. 같으면, no error, 다르면, error.

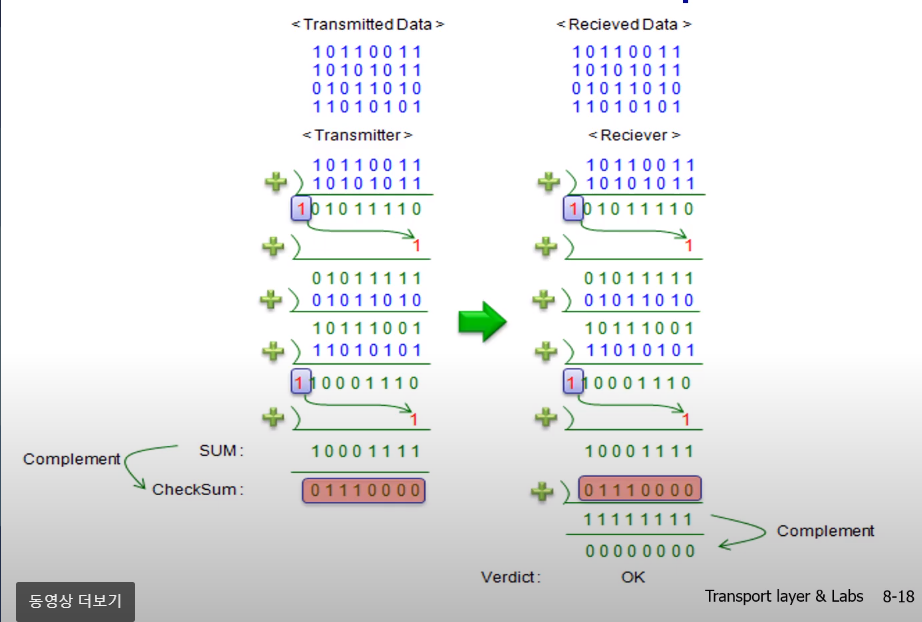
체크섬 방법

* 송신측에서 전송할 데이터들을 16비트 word 단위로 구분한다.
* 1의 보수를 취하고 그 합에대한 결과를 전송.
* 수신측에서 같은 합을 해보고 오류를 검출하는 방식이다.

특징

* 간단하지만, 워드의 순서가 바뀌는 오류가 발생하면 검출이 안됨
* 데이터링크 계층에서는 점차 사용하지 않고 여기서는 CRC가 더 많이 사용중.
* IPv6 헤더에는 체크섬 필드가 사라졌다.
* 전송계층의 TCP헤더나, UDP헤더 등 일부에서 사용한다.

### Internet 체크섬 example



* 16비트로 표현된 입력 데이터가 있다.
* 한줄씩 서로 더하고, carry가 생기면 그걸 다시더해서 8비트를 맞춰준다.
* 마지막에 나온 결과에서 1의 보수를 취한것이 바로 checksum이다.
* 수신측에서도 1의 보수를 취하는 과정까지는 동일하고, 여기서 나온 결과에 전송측에서 구한 checksum을 더해준다.
* 더해준 결과를 다시 1의 보수를 취했을때 0000 0000 결과가 나온다면 no error 즉, ok 이다.