2021-09-09

<네트워크>

네트워크는 Connection-Oriented service의 방식으로 data를 전달한다.

이러한 방법에는 TCP와 UDP 2가지가 있다.

TCP : Transmission Control Protocl. /. UDP : User Datagram Protocol

TCP는 신뢰성이 있다. 내가 보낸 그대로 데이터가 전달되며 순서를 지키면서 전달된다.

또한 보내는 쪽이 받는 쪽의 속도에 맞춰서 전달한다. 또한 둘 사이의 네트워크 상황을 고려하여 전달한다.

따라서 대부분은 TCP로 이루어져 있다.

이와 반대로 UDP는 신뢰성이 필요없는 경우에 사용한다. 관계가 없고 내 마음대로 보내는 것.

Protocol이란 기계들의 대화 약속이다. 중요한 메세지를 주고 받기 위한 서로의 준비 동작.

같은 Protocol이어야 통신이 된다. TCP-TCP는 되는데 TCP-UDP는 안된다.

Router(라우터) : 패킷의 위치를 추출하고 최적 경로를 지정하며 그 경로를 따라 데이터 패킷을 다음 장치로 전달하는 장치

라우터가 데이터를 출발지부터 목적지까지 어떤 방식으로 전달할까?

크게 2가지 방법이 있다.

1. Circuit switching과
2. Packet-switching

이 중 인터넷에서 사용하는 방식은 Packet-switching이다.

Circuit Switching이란 출발지에서 목적지까지 가는 길을 미리 에약해놓고 특정 사용자만을 위해 사용하는 방식이다.

대표적인 예시로 전화기가 있다. 누군가가 전화를 걸고 전화를 받으면 그 때부터 통화가 이루어지고 다른 사람과 전화를 할 수 없다. 발신자와 수신자가 회선을 사용하기 때문에 circuit Switching이다.

Packet Switching이란 유저가 보내는 Packet을 Packet 단위로 받아서 그때 그때 올바른 방향으로 전송하는 방식이다.

Packet은 패킷 방식의 컴퓨터 네트워크가 전달하는 데이터의 형식화된 블록이다. 패킷 지원하지 않는 컴퓨터는 단순히 바이트, 문자열, 비트를 독립적으로 연속하여 데이터를 전송한다고 한다.

예를 들어 초당 10M비트 크기의 데이터를 한번에 보낼 수 있는 크기를 가진 케이블이 있을 때

Circuit Switching 방식은 각 유저들의 속도가 1Mbps라고 하면 10명까지 가능하다는 것이다.

Packet Switching 방식은 상관없다. 들어오는 대로 다 받아서 보내니까 몇 명까지 이런 것이 없다. 따라서 인터넷은 Packet-Switching 방식을 사용한다.

조금 더 자세히 풀어서 설명하면 사용자가 인터넷을 사용할 때 버튼을 클릭하면 데이터가 왔다갔다 한다. 그 과정 속에서 사용자는 끊임없이 버튼을 누르는 것이 아니다. 버튼을 누르지 않을 때 다른 사용자의 버튼을 받으면 되기 때문에 사용자가 많아도 분산시켜서 받을 수가 있다. 따라서 Packet Switching 방식이 인터넷으로 적합하다.

Packet 받는 것부터 전달하는 것까지의 과정을 좀 더 자세하게 살펴보자.

1.nodal processing : 처음에 CPU로부터 packet을 받게 되면 최종 목적지가 어디인지 확인한다. 목적지에 따라 알맞은 outgoing edge로 보내야 한다.

그 과정 속에서 내보내려 할 때 나가는 데이터보다 들어오는 데이터의 양이 많아지면 느려지기 때문에 그것을 임시로 저장해 놓을 공간이 필요하다. 그것을 대비하기 위한 것이 Queue(버퍼)이다.

Queue라는 저장장소는 라우터에 존재한다.

Packet이 delay되는 방법은 4가지가 있다.

1. Nodal processing : 패키징 검사 시간
2. Queueing : queue에서 기다리는 시간
3. Transmission delay : 첫번째 비트부터 마지막 비트가 나가는데까지 걸리는 시간
4. Propagation delay : 출발해서 다음 라우터까지 도착하는데 걸리는 시간

이 중 1,3,4는 성능 개선 또는 변경하여 delay를 줄일 수 있는데 queueing delay는 어떻게 할 수가 없다.

패키징 검사 시간을 processing delay라고 부른다.

Queue에서 기다리는 시간을 Queueing delay라고 부른다.

Queue에 있다가 이제 나갈 차레가 되면 나가야 하는데 나가는 순간에 걸리는 delay가 있다.

첫번째 비트가 나가는 순간부터 마지막 비트가 완전히 나가는데 까지 걸리는 시간을 transmission delay 라고 한다.

Processing delay같은 경우 라우터를 좋은걸 사거나 (성능 개선하면) 좋아진다.

-네트워크 상황에서의 delay는 대부분 queueing delay에서 비롯된다. 만약 queue가 넘치면 packet loss(패킷 손실)이 일어난다. 인터넷에서 발생하는 packet loss 90% 이상은 queue에서 일어난다고 한다.

-queue가 존재하는데 [대가히고 빠져나가고]를 반복하다보면 그 사이사이에서 delay가 생긴다.

Ex) 새벽시간에는 다 비어있으니까 대기 없이 쭉 가는 것이고

사람이 많은 시간에는 대기를 하게 되는데 Queuer 저장 공간이 가득차게 되면 drop(loss) 즉 packet loss가 일어나게 된다.

이러한 과정 때문에 원하는 결과를 다 보내기 위해서는 결국 재전송을 할 수밖에 없다.

재전송은 처음부터 (TCP)부터 재전송을 해주거나 직전 라우터가 재전송을 해주는 경우 2가지가 있는데 인터넷은 처음부터 (TCP부터) 재전송을 하게 된다.

라우터는 대기하고 빠져나가는 것만으로도 할 일이 많기 때문에 단순 작업을 극대화 시키고 재전송은 처음부터 하게 된다.

Transmission delay같은 경우는 케이블을 늘려 bandwidth를 늘리면 delay가 줄어들게 된다.

중간 정리를 해보자면 packet이라는 비트들의 집합을 묶음 단위로 전달하고 packet은 출발지에서 목적지까지 라우터를 거쳐가며 묶음 형태로 게속 이동하게 된다.

네트워크 앱을 만들 때 여러 계층이 있고 그 게층 안에는 다양한 프로토콜들이 존재한다.

웹 브라우저는 네트워크 기능이 있는 process

네트워크 계층은 client server안에만 존재한다.

레이어를 보면

Application

Transport

Network

Data link

Physical

이렇게 구성되어 있다.

IP주소는 인터넷상의 존재하는 컴퓨터를 지칭하는 주소이다.

컴퓨터에서도 process가 정말 많기 때문에 특정 process를 지칭하기 위해서 Port를 사용한다.

[port의 역할 : 많은 process들 중에서 어떤 process인지, 더 자세히는 그 포트의 소켓이 어떤 것인지 알려주는 역할]

포트의 소켓의 주소를 알아야 뭔가를 할 수 있다. 따라서 그 소켓의 주소를 인덱싱하는 뭔가가 필요하다.

* IP address와 port의 커뮤니케이션

예시를 들어보자

Web browser를 실행하면 그 프로세스가 돌아갈테고 네이버 홈페이지를 실행하려고 하면 네이버 프로세스에 연결을 해주어야 한다.

그때 네이터 프로세스에 해당하는 소켓 주소를 입력해야 연결이 된다.

[www.naver.com](http://www.naver.com)

실제로는 어떻게 연결하는가?

네이버 컴퓨터의 IP주소와 네이버 웹 서버 process의 그 소켓에 해당하는 포트넘버 80을 입력해야 연결된다.

사람들을 위해서 [www.naver.com](http://www.naver.com) 이렇게 입력하면 내부에서 DNS(도메인 네임 시스템)를 통해 네이버의 ip 주소로 변환된다. 이런 방식으로 접근을 하게 되는 것.

원래는 포트넘버 80 즉 [www.naver.com:80](http://www.naver.com:80) 이렇게 입력해야 하는데 그것도 귀찮기 때문에 [www.naver.com](http://www.naver.com)만 입력해도 :80을 알아서 넣어준다.

www.naver.com 과 [www.naver.com:80](http://www.naver.com:80) 똑같이 [www.naver.com](http://www.naver.com) 으로 들어가진다.(당연한 것)

\*DNS : 호스트 도메인 이름을 호스트의 네트워크 주소로 바꾸거나 그 반대의 변환을 수행할 수 있도록 하는 것

대부분의 서비스가 공통적으로 포트넘버 80을 사용한다. 이는 왜 그럴까?

서비스는 24시간 켜져 있어야 하고 서비스의 주소는 항상 일치해야 한다.

네이버, 다음, 구글 등의 주소가 다 다르니까 그 주소를 해석해 주는 것이 DNS이고

DNS는 단순히 IP 주소만 번역해준다. 포트넘버까지 필요한데 다르면 엄청 복잡해지니까 포트넘버는 동일하게 사용하기로 하였고 그렇게 정한 것이 :80 이다.

(공통적으로 80을 사용하지 않았다면 DNS에서 포트까지도 다 만들어 주었어야 했다.)

계층이란 개념은 하위계층에서 상위계층으로 어떠한 기능을 제공해준다. 즉 기능을 사용할 수 있게 서비스를 제공하는 것을 말한다.

Application 내의 protocol들은 그 하위계층인 transport protocol에서 제공하는 서비스를 받는 것이다. 그렇게 되니까 transport layer에서 제공하는 기능이 어떤 것인지가 중요하다.

제공해주지 않는 기능은 못쓰는 것이고 제공해주는 기능은 쓸 수 있는 것이다.

\*개발자 입장에서 transport 층에서 제공해주었으면 하는 기능들

1. data integrity (100% 완전하게 보내는것)

2. throughput (용량)

3. timing (시간)

4. security (보안)

=> 이 4가지 중 실제로 transport 층에서 제공해주는 기능은 data integrity 하나.

따라서 남은 3가지는 상위 계층인 application layer에서 진행된다.

HTTP : Hyper Text Transfer Protocol

한마디로 텍스트인데 Link가 있다는 뜻. 즉 다른 텍스트를 지칭하는 링크들.

이런 텍스트들을 전송하는 Protocol이 HTTP이다.

HTTP는 매우 간단하다. 요청하고 전달받는다. (Http is stateless) 그거 외엔 안한다.

내가 요청하는 웹페이지의 Hyper text를 요청하면 (Hyper Text 파일이름) 그것을 받아 전달해준다.

HTTP 서비스는 TCP 서비스를 사용하니까 요청하고 전달받기 전에 TCP connection을 먼저 해주어야 한다.

따라서 HTTP가 TCP connection을 사용하는 방법에 따라 2가지로 나뉘어진다

[TCP connection을 persistent하게 사용할 것인지, 아닌지로 나눌 수 있다.]

1. 메세지를 주고 받고 끊으면 non-persistent
2. 끊지 않고 유지하면서 재사용하게 되면 persistent이다.

Non-persistent 같은 경우 끊게 되면 또 새로운 것을 받아올 때 다시 처음부터 받아와야 한다.

현재 웹 브라우저에서는 persistent HTTP를 사용한다.

<코딩테스트>

이것이 코딩테스트다 – 상하좌우

L R U D 4가지 경우를 if로 나눠서 해결하려고 했다.

책에 나온 답안은 리스트 형태로 한번에 묶어서 보기 좋게끔 구성하였다. => 깨달았다.

오늘은 네트워크 공부를 파면서 HTTP3는 왜 TCP를 쓰지 않고 UDP를 사용했는지에 대해 찾아보고 깨달으면서 하루를 마무리 한다.