2주차

목차

Story3 전세계 DNS서버 연대

Story4 프로토콜 스택에 메시지 송신을 의뢰

Story3 전세계 DNS서버 연대

1.데이터 송.수신 동작의 개요

조회 메시지에는 세가지 정보가 포함되어 있다.

(a)이름

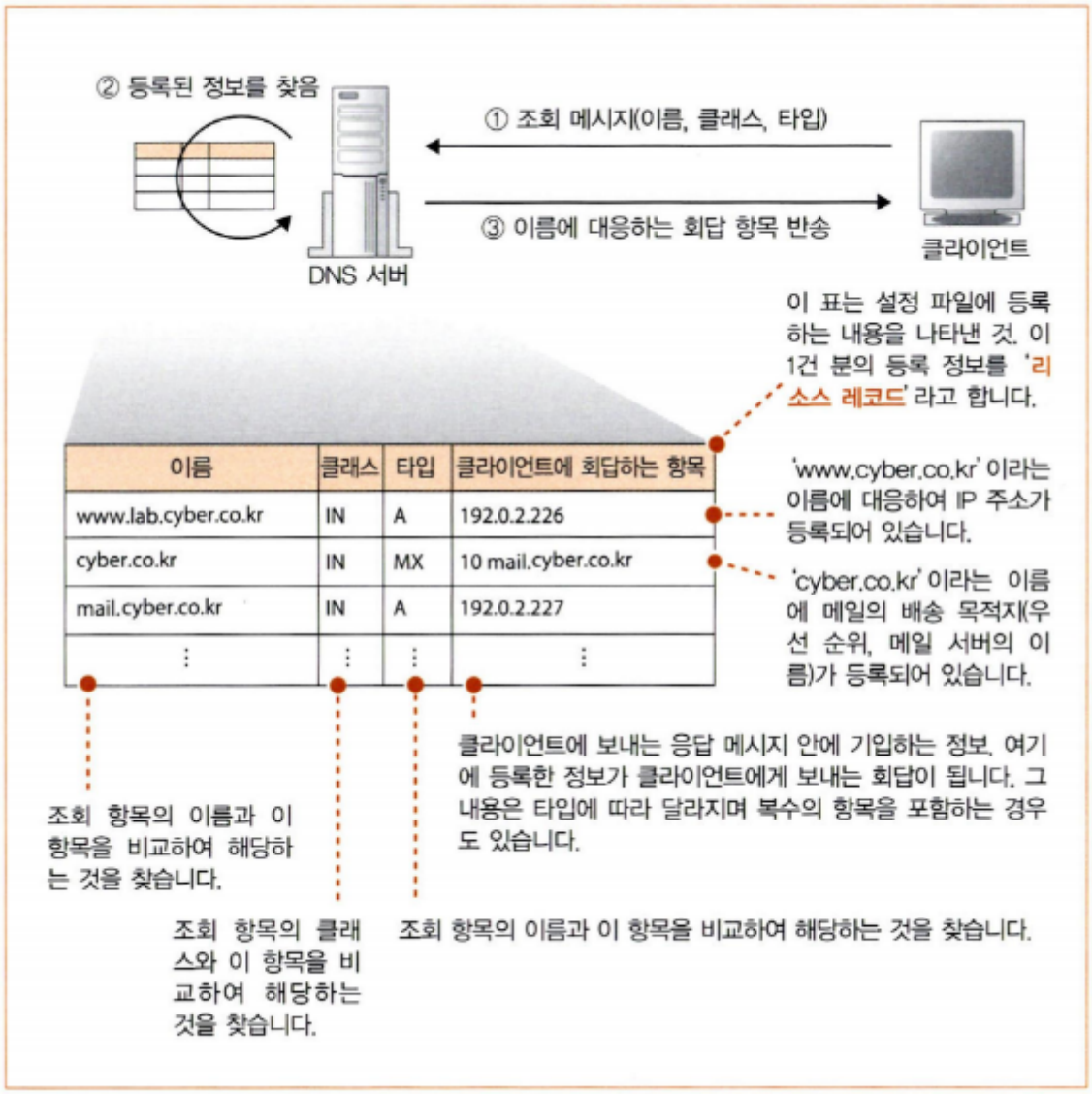
서버나 메일 배송 목적지(메일주소에서 @ 뒷부분의 이름)와 같은 이름

(b)클래스

DNS의 구조를 고안했을때 인터넷 이외에도 네트워크에서의 이용까지 검토하여 이것을 식별하기 위해 클래스라는 정보를 준비하였다. 그러나 지금은 인터넷 이외의 네트워크는 소멸되었으므로 클래스는 항상 인터넷을 나타내는 “IN”이라는 값이 된다.

(c)타입

이름에 어떤 타입의 정보가 지원되는지를 나타낸다. 예를 들어 타입이 A이면 이름에 IP 주소가 지원되는 것을 나타내며 MX이면 이름에 메일 배송 목적지가 지원된다는 것을 나타낸다. 또한 이 타입에 따라 클라이언트에 회답하는 정보의 내용이 달라진다.



위의 그림은 DNS서버의 기본동작

DNS서버에는 이들 세가지 정보에 대응하여 클라이언트에 회답하는 항목을 등록해 두었다가 조회메시지에 해당하는 것을 찾아 클라이언트에게 회답하는 것이다.

EX)

1)

a)이름= www.lab.cyber.co.kr

b)클래스= IN

c)타입=A

→192.0.2.226을 클라이언트에게 반환

2)

a)이름= cyber.co.kr

b)클래스= IN

c)타입= MX

→10(메일서버의 우선순위), mail.cyber.co.kr(메일서버의 이름),192.0.2.277 을 클라이언트에게 반환

즉 A(조회), MX(메일)등 타입에 따라 반환되는 값이 다름.

2 도메인의 계층

인터넷에는 막대한 수의 서버가 있기 때문에 1대의 DNS에 옮겨 놓는것은 불가능 하다.

따라서 정보를 분산시켜, 다수의 DNS서버에 등록을 하고 다수의 DNS서버가 연대하여 어디에 정보가 등록되어 있는지를 찾아내는 구조이다.

ex)

www.cyber.co.kr

→

kr은 대한민국 할당된 도메인

co는 국내의 도메인 분리를 위해 사용(회사를 의미)

cyber회사에 할당된 도메인

www서버의 이름

3 담당 DNS 서버를 찾아 IP주소를 가져옴

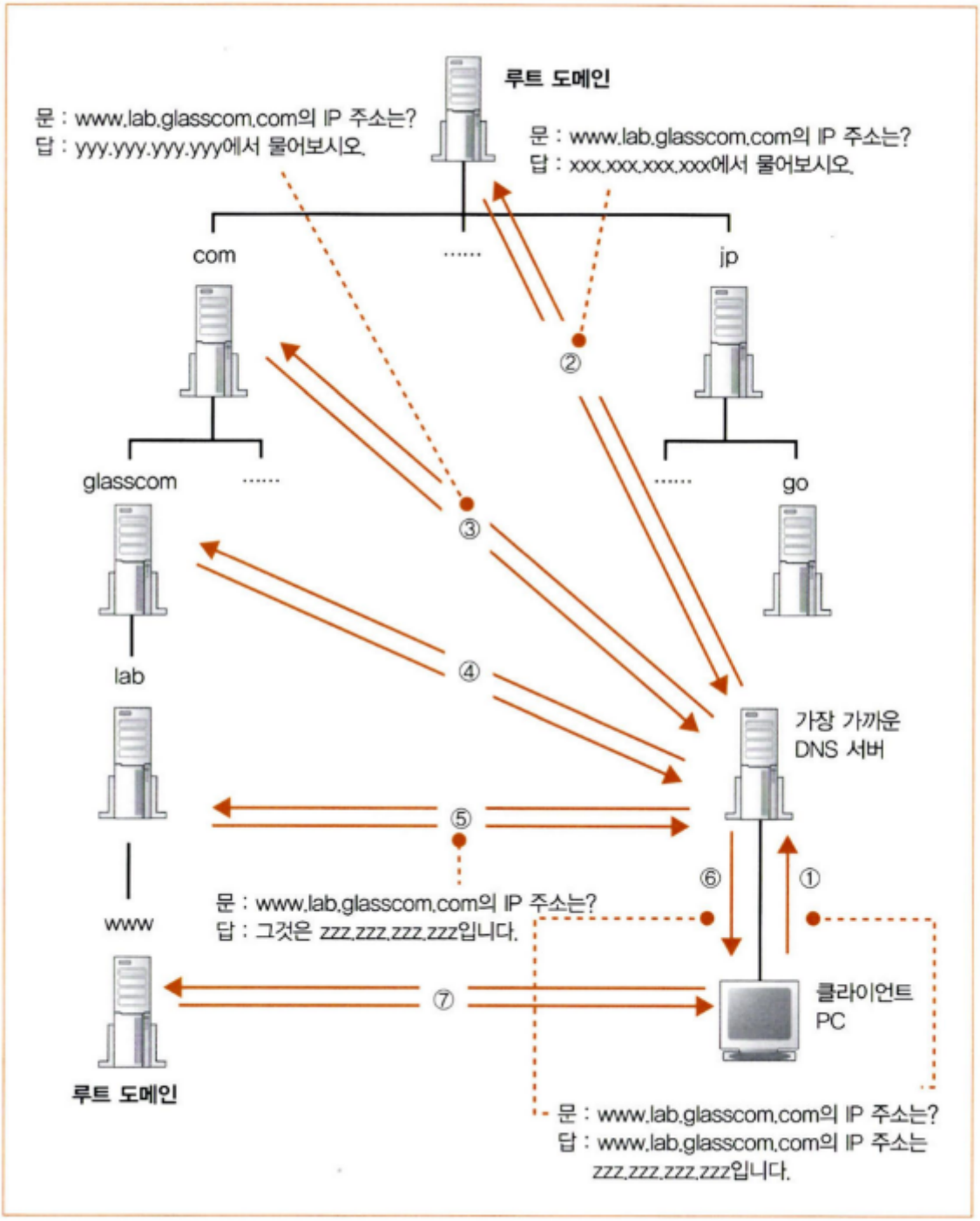
DNS서버가 무수히 많기 때문에 하위 도메인을 담당하는 DNS서버의 IP주소를 상위의 DNS서버에 등록하는 방식으로 사용.

이렇게 되면 최상이 도메인 com이나 kr에 하위의 도메인을 등록하는것이냐면 그것은 아님

그 위에 **루트도메인**이 존재함.

또한 루트도메인의 DNS서버를 인터넷에 존재하는 DNS서버에 전부 등록하여 어느 DNS서버도 루트 도메인에 엑세스 할 수 있도록 한다.

(현재 루트 도메인의 DNS서버에 할당된 IP주소는 13개이다.)



위와 같이 DNS서버에 있지 않으면 상위의 DNS서버로 이동하고 그 과정을 반복하다 보면 루트까지 넘어갈 수 있다.

그 이후 루트DNS에서부터 차례로 값을 알 수 있다.

7번의 과정을 통해 ip주소를 얻을 수 있는 것이다.

DNS서버는 캐시 기능으로 빠르게 회답할 수 있다.

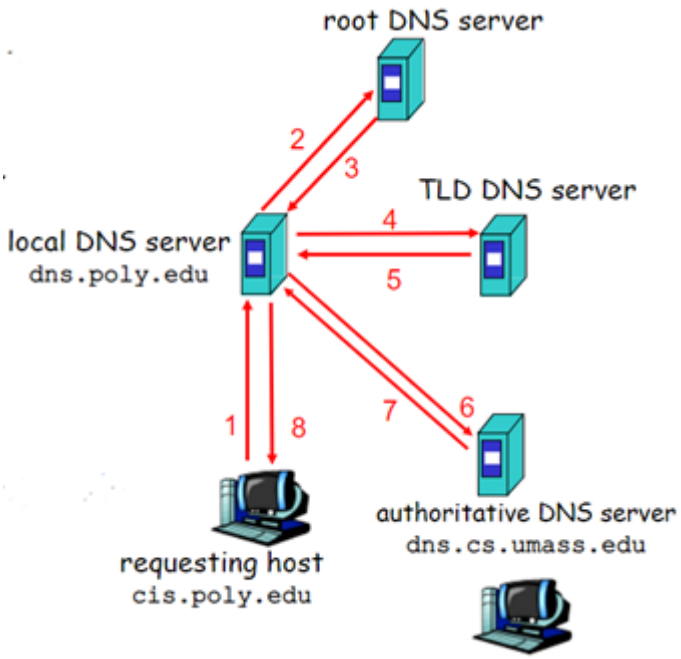
물론 상위와 하위의 도메인이 같은 DNS서버에 등록되는 경우와 같이 구조가 위와 다를 수 있다.

또한 최상위 루트 도메인에서 차례로 따라간다는 원칙 또한 DNS서버의 캐시에 기록되어 원칙대로 움직이지 않을 수 있다.(캐시 방법이 더 유리)

다만 캐시의 값이 항상 올바를 수는 없기에 유효기간을 정하고 지나면 캐시에서 삭제함.

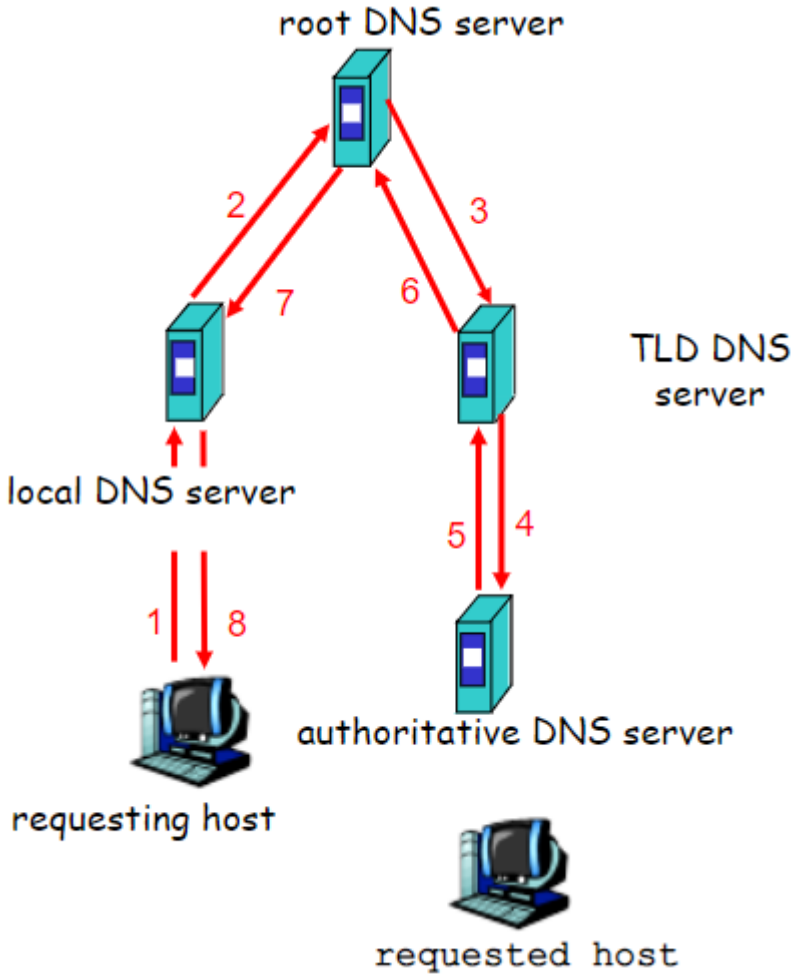
DNS의 두가지 방법

Iterated query



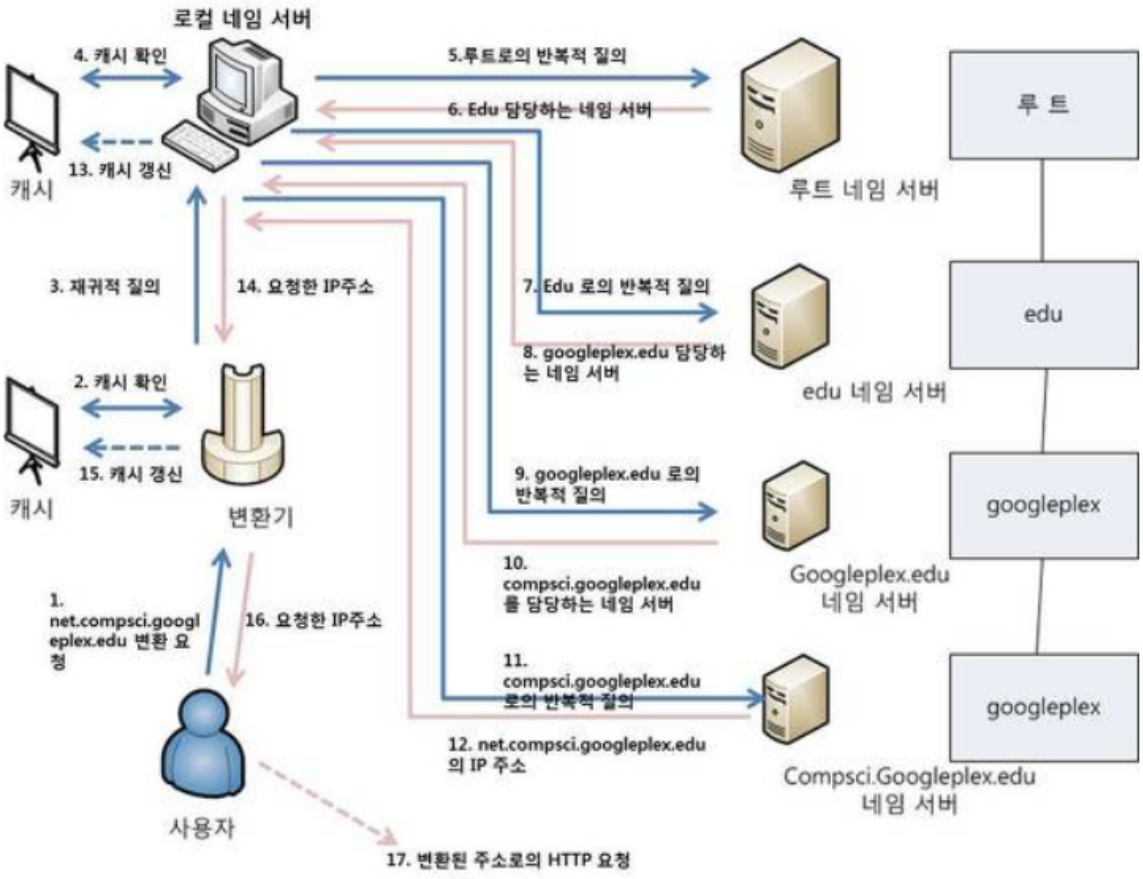
host가 naver.com에 대해 query를 보내면 Local DNS서버가 rootnameServer에 query를 보내 com담당 server의 주소를 return받고 다시 com담당 server에 query를 보내 naver.com이 반환요청을 보낸다. 이렇게 최종IP주소를 받을때까지 요청&응답을 계속해서 localnameserver가 반복하는 방법이다.

Recursive query



localHost가 naver.com에 대해 query를 보내면 LocalDNS server가 root name server에 query를 보내고, root server는 자신의 server 에 등록되어 있는지 검사한 다음 없으면 com담당 서버에 요청을 한다. recursive하게 실제 domain name을 가지고 있는 server까지 query가 이동하여 IP주소를 얻는 방법이다.

실제 DNS server의 동작방식



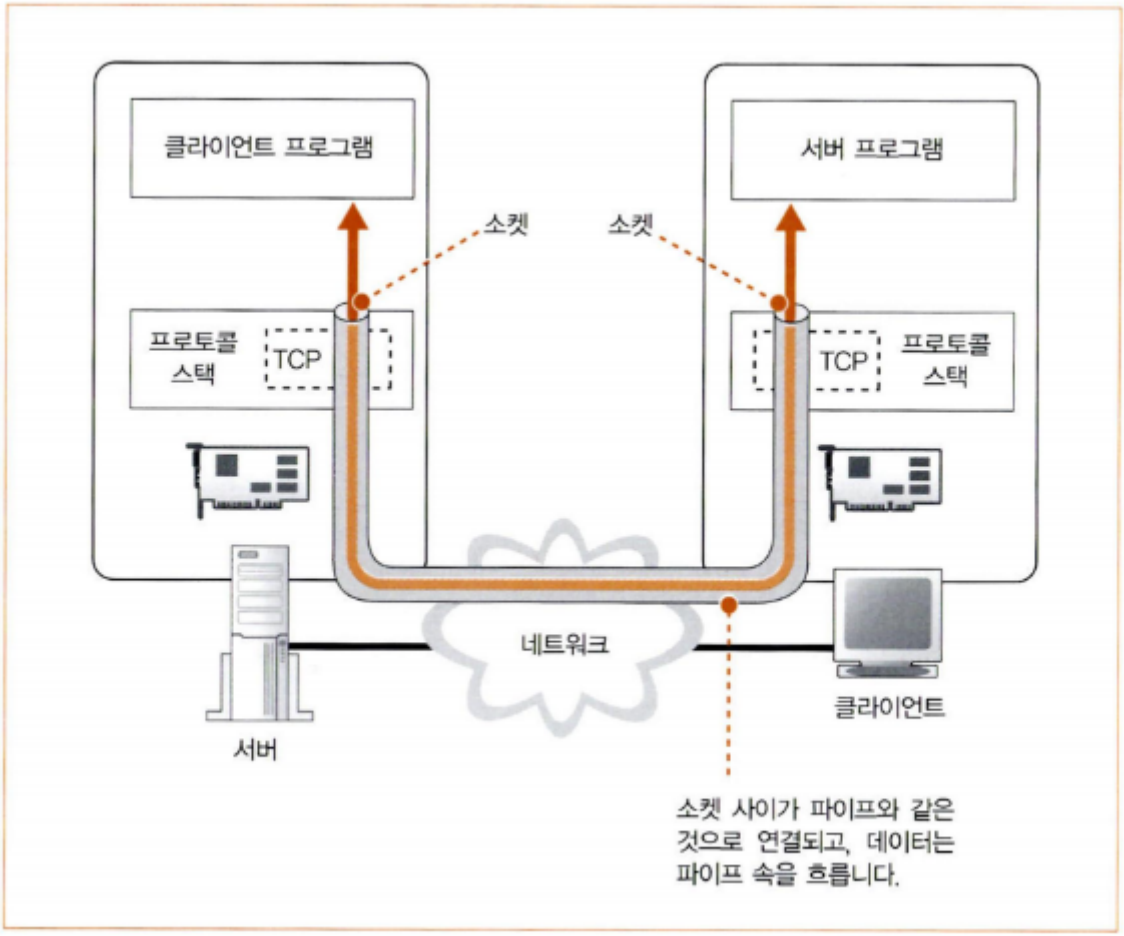
두가지를 함께 사용함으로서 효율성을 높인다.

Story4 프로토콜 스택에 메시지 송신 의뢰

데이터 송*수신 동작의 개요

IP주소를 조사한 이후 IP주소의 상대(엑세스 대상 웹서버)에 메시지를 송신하도록 **프로토콜 스택**에 의뢰한다. 웹서버에 보내는 HTTP의 메시지는 디지털 데이터이므로 디지털 데이터를 송신하도록 의뢰한다고 보면 된다.

여기에서도 Socket라이브러리에 들어있는 프로그램을 이용한다.



파이프로 연결하는 출입구 같은 것을 **소켓**으로 보며 되는데, 서버측에서 소켓을 만들고, 소켓에 클라이언트가 파이프를 연결하기를 기다린후 클라이언트측에서 파이프를 연결한다.

데이터 송*수신 동작의 단계

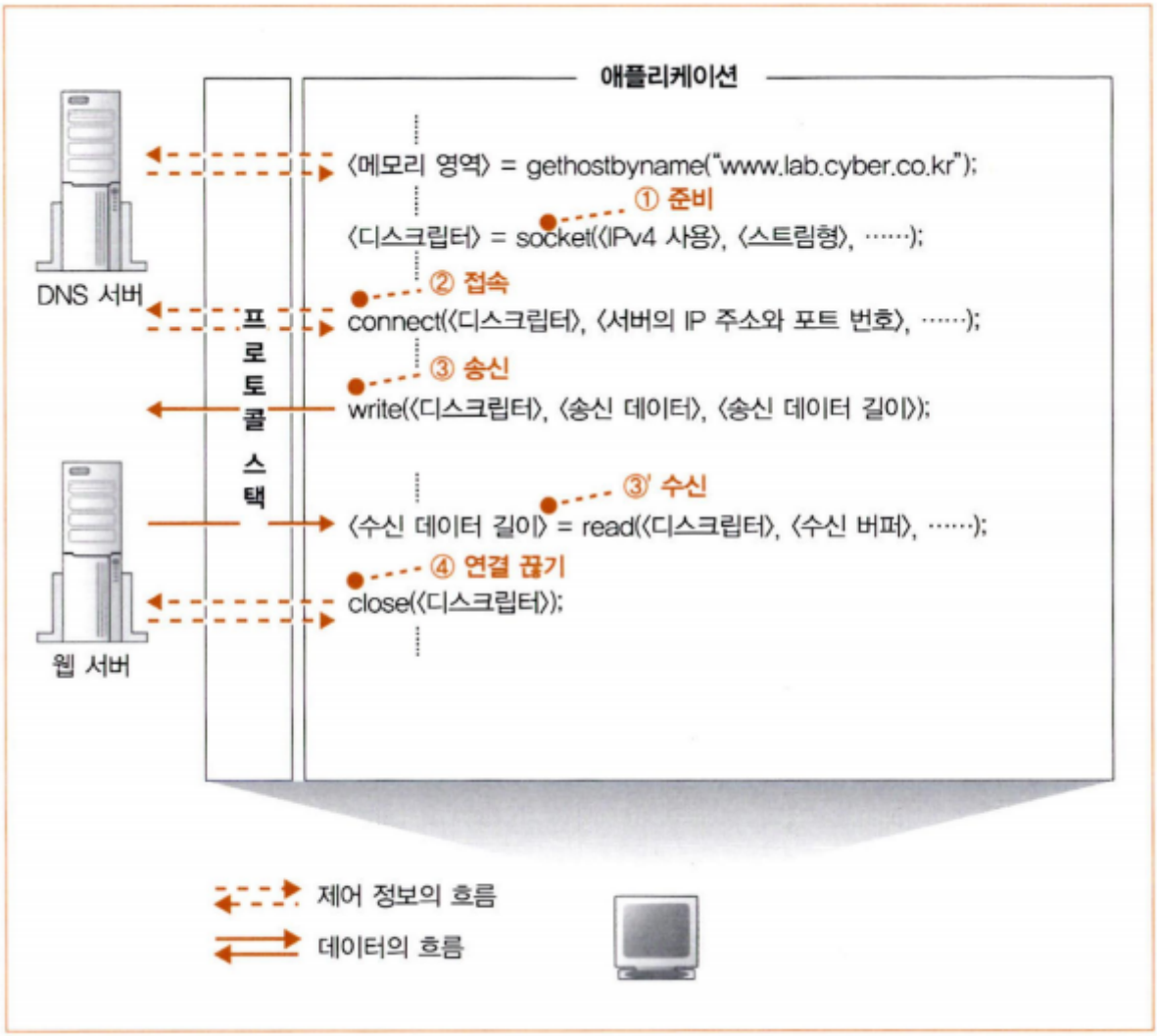
(1)소켓을 만든다(소켓 작성 단계)

(2)서버측의 소켓에 파이프를 연결한다(접속단계)

(3)데이터를 송*수신 한다(송*수신 단계)

(4)파이프를 분리하고 소켓을 말소한다(연결 끊기 단계)

이제 위의 동작들을 하나씩 알아보도록 하겠다.



소켓의 작성 단계

1번 동작을 보면 socket이라는 프로그램 부품만 호출(socket내부에 제어가 넘어가 소켓을 만듦)

소켓이 생기면 **디스크립터**(소켓을 식별하기 위해 사용) 라는 것이 돌아오는데 애플리케이션은 이것을 받아 메모리에 기록해둔다.

디스크립터는 복수의 데이터 송*수신 동작이 진행될때 필요하다.

파이프를 연결하는 접속단계

만든 소켓을 서버측의 소켓에 접속하도록 프로토콜 스택에 의뢰하는데, 애플리케이션은 Socket라이브러리의 connect라는 프로그램 부품을 호출하여 실행한다.

connect를 할 때 지정하는 **디스크립터, 서버의 IP주소,포트번호** 세가지 이다.

소켓을 만들때 돌아온 디스크립터는 connect가 프로토콜 스택에 통지받은 디스크립터를 보고 어느 소켓을 서버측에 접속할지 판단하여 접속 동작을 실행한다.

IP주소는 DNS서버에 조회하여 조사한 액세스 대상 서버의 IP주소이다. 송수신 하는 상대의 IP주소를 프로토콜 스택에 알릴 필요가 있다.

정리하자면

- 디스크립터: 애플리케이션이 소켓을 식별하는 것
- IP주소와 포트번호: 클라이언트와 서버 간에 상대의 소켓을 식별하는 것

IP주소로 지정할수 있는 것은 네트워크의 어느 컴퓨터인가 까지만 이므로 접속동작은 상대측의 소켓에 대해 이루어지므로 소켓을 지정해야 하는데, 여기서 포트번호 또한 필요하다.

서버측의 포트번호는 애플리케이션의 종류에 따라 미리 결정된 값을 사용한다.

접속단계를 마치면 데이터 송*수신이 가능한 상태가 된다.

메시지를 주고받는 송*수신 단계

데이터를 송신할 때 write, 수신할때 read 라는 프로그램 부품을 사용하는데

write를 호출 할때 디스크립터와 송신데이터를 지정한다.

read를 통해 수신동작을 프로토콜스택에 의뢰하면 수신한 응답 메시지를 저장하기 위한 메모리 영역을 지정하는데, 이 메모리 영역을 수신 버퍼라고 한다.

연결 끊기 단계에서 송*수신 종료

Socket에서 close라는 프로그램부품을 호출하면 연결 끊기 단계로 넘어간다.

그러면 소켓 사이를 연결한 파이프와 같은 것이 분리되고 소켓도 말소된다.

여기서 1개의 데이터를 읽을때마다 위와 같은 작업을 반복해야 하므로 비효율 적이다.

그리하여 등장한 것이 HTTP1.1버젼이다.

http 1.1 vs 2.0 vs 3.0

Http pipelining

을 참고하여 Http의 변천사를 확인.

출처:

성공과 실패를 결정하는 1%의 네트워크 원리

컴퓨터 네트워크 하향식접근