Chapter 02

웹 브라우저

1. HTTP 리퀘스트 메시지

URL 입력

- 우리가 웹 브라우저를 통해 인터넷에 접속할 때에는 웹 브라우저의 URL(Uniform Resource Locator) 창에 주소를 입력.
- 예를 들어 네이버에 접속하고자 한다면, http://www.naver.com 이라고 입력을 하면 네이버에 접속할 수가 있다.
- URL은 일반적으로 http(s):로 시작하는 것 같지만 사실은 http:뿐만 아니라 ftp:로 시작하는 것, file: 로 시작하는 것, mailto: 로 시작하는 것 등 여러 가지가 있다.
- 다양한 URL이 준비되어 있는 것은 이유가 있다.
- 브라우저는 웹 서버에 접속하는 클라이언트로 사용하는 경우가 많지만 브라우저의 기능은 그뿐만이 아니다.
- 파일을 다운로드/업로드하는 FTP(File Transfer Protocol)의 클라이언트 기능이나 메일의 클라이언트 기능도 가지고 있다.
- 브라우저는 몇 개의 클라이언트 기능을 겸비한 복합적인 클라이언트 소프트웨어라고 할 수 있다.
- 그렇기 때문에 웹 서버에 접속할 때는 http:, FTP 서버라면 ftp: 라는 식으로 여러 종류의 URL이 준비되어 있는 것.

1. HTTP 리퀘스트 메시지

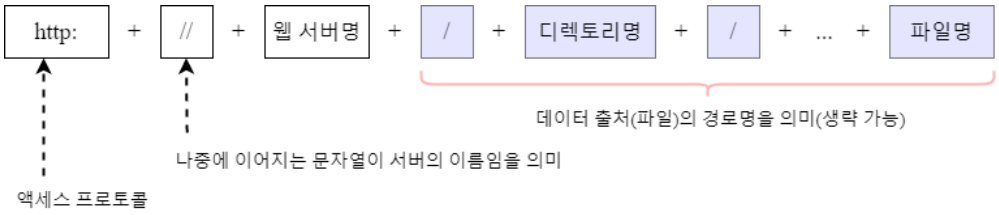
URL 입력

- 이와 같이 쓰는 방법은 다양하지만 모든 URL에는 하나의 공통점이 있다.
- URL의 맨 앞에 있는 문자열, 즉 http:, ftp:, file:, mailto: 라는 부분에서 접속할 때의 프로토콜 종류를 나타낸다는 점이다.
- 접속 대상이 웹 서버라면 HTTP(HyperText Transfer Protocol)라는 프로토콜을 사용하여 접속하고 FTP 서버라면 FTP라는 프로토콜을 사용하는 식이다.
- 그 후에 계속되는 부분을 쓰는 방법도 다양하지만, 맨 앞부분에 따라 그 뒤에 계속해서 쓰는 방법이 결정된다.

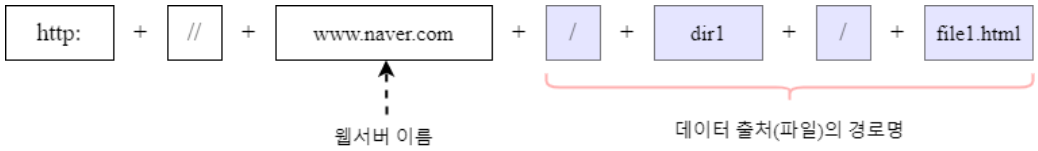
1. HTTP 리퀘스트 메시지

URL 구조

- 브라우저가 처음 하는 일은 웹 서버에 보내는 요청 메시지를 작성하기 위해 사용자가 입력한 URL을 분석하는 것.
- (a) URL의 요소



- (b) URL의 예
  - http://www.naver.com/dir1/file1.html
- (c) (a)의 요소에 맞춰 (b)를 분석한 예



1. HTTP 리퀘스트 메시지

URL 구조

- 브라우저가 처음 하는 일은 웹 서버에 보내는 요청 메시지를 작성하기 위해 사용자가 입력한 URL을 분석하는 것.
- URL은 그림 (a)와 같이 몇 개의 요소를 나열한 것이다.
- 그래서 분해한 요소를 나열한 방법을 살펴보면 URL의 의미를 알 수 있다.
- 예를 들어 (b)의 URL을 분해한 것이 (c)이다.
- 분해한 결과인 (C)를 보면 웹 서버의 이름을 나타내는 부분에 'www.naver.com'이라는 이름이 있고, 파일의 경로명에 해당하는 부분에 '/dir1/file1.html'이라는 문자열이 있다.
- 이것을 해석한다면, (b)의 URL은 'www.naver.com' 이라는 웹 서버에 있는 '/dir1/'이라는 디렉토리의 아래에 있는 'file1.html' 파일에 접근한다는 의미이다.
- 그림 (b)는 http: 로 시작하는 URL의 대표적인 예이다.
- 이것과는 조금 달리 다음과 같이 /로 끝나는 URL을 볼 수 있다.

```
http://www.naver.com/dir/
```

1. HTTP 리퀘스트 메시지

URL 구조

- 이것은 다음과 같이 생각할 수 있다.
- 끝이 /로 끝난 것은 /dir/의 다음에 써야 할 파일 이름을 쓰지 않고 생략한다는 것이다.
- URL의 규칙에는 이와 같이 파일 이름을 생략해도 좋다.
- 하지만 파일 이름을 쓰지 않으면 웹 서버는 웹 브라우저가 어느 파일을 요청하는지 알 수 없다.
- 그래서 파일 이름을 생략할 때를 대비하여 웹 서버는 요청에 대한 기본 파일을 미리 설정해 둔다.
- 이 설정은 서버에 따라 다르지만 대부분의 서버가 'index.html' 또는 'default.htm' 이라는 파일을 설정해 둔다.
- 그러므로 위와 같이 파일 이름을 생략하면 '/dir/index.html' 또는 '/dir/default.htm' 이라는 파일에 접근한다.
- 다음과 같이 웹 서버의 도메인 이름만 쓴 URL을 볼 수 있는데, 이것도 파일 이름을 생략한 것이다.

```
http://www.naver.com/
```

1. HTTP 리퀘스트 메시지

URL 구조

- 끝에 /가 있으므로 /라는 디렉토리가 지정되고 파일 이름은 생략된 것입니다. 따라서 '/index.html' 또는 '/default.htm'이라는 파일에 접근하는 것입니다.

```
http://www.naver.com
```

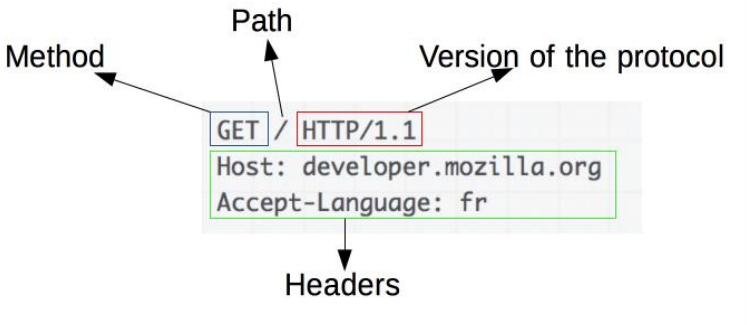
- 이번에는 끝의 /까지 생략되어 있다.
- 이와 같이 디렉토리 이름까지 생략해 버리면 무엇을 요청하고 있는지 알지 못하게 되므로 너무 지나친 생략일 수 있다.
- 하지만 이렇게 쓰는 방법도 인정되고 있다.
- 경로 이름이 아무 것도 없는 경우에는 루트 디렉토리의 아래에 있는 미리 설정된 파일 즉 '/index.html' 또는 '/default.htm'이라는 파일에 액세스하면 혼란이 없기 때문이다.
- 단, 다음과 같은 예는 미묘하다.

```
http://www.naver.com/whatisthis
```

1. HTTP 리퀘스트 메시지

HTTP 요청 메시지 생성

- **URL을 해독하고 웹 서버와 파일명을 판단하면 브라우저는 이것을 바탕으로 HTTP의 요청 메시지를 생성.**
- **실제 HTTP 메시지는 쓰는 방법, 즉 포맷이 결정되어 있으므로 브라우저는 이 포맷에 맞게 요청 메시지를 생성.**



- **HTTP 메시지의 메소드를 통해 웹 브라우저는 웹 서버에게 어떻게 할 것인지를 전달.**
  - URL 입력 상자에 URL을 입력하면 해당 페이지를 표시하는데, 이 경우 GET 메소드를 사용.
  - 하이퍼링크를 클릭한 경우도 마찬가지로 GET 메소드를 사용.
  - 폼의 경우 폼 부분의 HTML 소스 코드에 어느 메소드를 사용하여 요청 메시지를 보낼 것인지를 지정한 후 GET과 POST를 구분하여 사용.

1. HTTP 리퀘스트 메시지

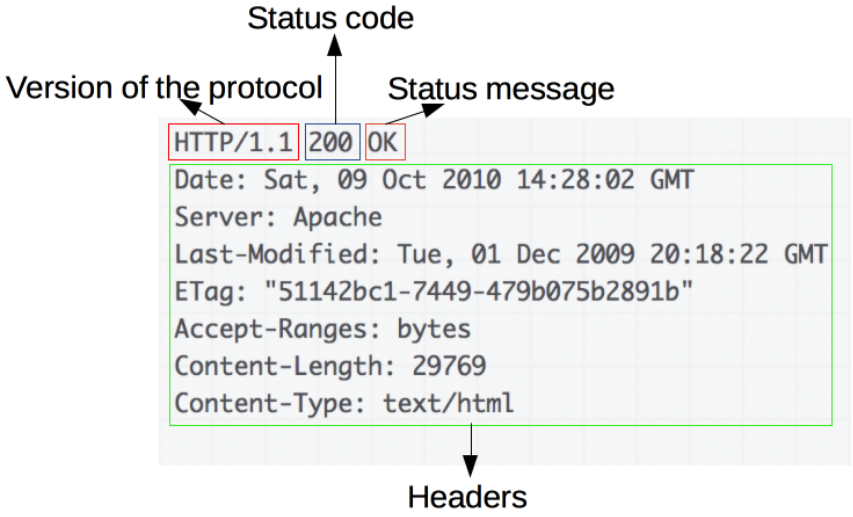
HTTP 요청 메시지 생성

- **메소드를 썼으면 한 칸 띄운 다음에 URI를 기록.**
  - 경로 이름은 보통 URL에 포함되어 있으므로 URL에서 경로 이름을 추출하여 복사
- **첫 번째 행의 끝에 메시지가 HTTP 프로토콜의 버전 번호를 써서 첫 번째 행을 완료**
- **두 번째 행부터는 메시지 헤더라는 행이 이어진다.**
  - 첫 번째 행에서 리퀘스트의 내용을 대략 알 수 있지만 부가적인 자세한 정보가 필요한 경우도 있는데, 이것을 써 두는 것이 메시지 헤더의 역할.
  - 날짜, 클라이언트측이 취급하는 데이터의 종류, 언어, 압축 형식, 클라이언트나 서버의 소프트웨어 명칭과 버전, 데이터의 유효 기간이나 최종 변경 일시 등 다수의 항목이 사양으로 정해져 있다.
- **메시지 헤더를 쓰면 그 뒤에 아무 것도 쓰지 않은 하나의 공백 행을 넣고, 그 뒤에 송신할 데이터를 기록.**
  - 이 부분을 메시지 본문이라 하며, 이것이 메시지의 실제 내용이 된다.
  - 단, 메소드가 GET인 경우에는 메소드와 URI만으로 웹 서버가 무엇을 할지 판단할 수 있으므로 메시지 본문에 쓰는 송신 데이터는 아무것도 없다.
  - 따라서 메시지 헤더가 끝난 곳에서 메시지는 종료.
  - 메시지가 POST인 경우에는 폼에 입력한 데이터 등을 메시지 본문 부분에 기록.
- **이로써 요청 메시지 작성 동작이 완료.**

1. HTTP 리퀘스트 메시지

HTTP 응답 메시지 수신

- 요청 메시지를 보내면 웹 서버에서 응답 메시지가 되돌아온다.
- 응답 메시지의 포맷도 기본적인 개념은 요청 메시지와 같다.
- 단, 첫 번째 행이 다르다.



- 응답의 경우는 정상 종료했는지, 아니면 오류가 발생했는지, 즉 요청의 실행 결과를 나타내는 스테이터스 코드와 응답 문구를 첫 번째 행에 써야 한다.
- 이들 둘은 내용이 같지만 용도가 다르다.
  - 스테이터스 코드는 숫자로 쓴 것이며, 주로 프로그램 등에 실행 결과를 알려주는 것이 목적
  - 이에 비해 응답 문구쪽은 문장으로 쓰여있으며 사람에게 실행 결과를 알리는 것이 목적

1. HTTP 리퀘스트 메시지

HTTP 응답 메시지 수신

- 응답 메시지가 되돌아오면 그때부터 데이터를 추출한 후 화면에 표시하여 웹 페이지를 눈으로 볼 수 있다.
- 페이지가 문장으로만 되어 있으면 이것으로 끝이지만, 영상 등이 포함되어 있는 경우에는 계속 내용이 있다.
- 영상 등을 포함한 경우에는 문장 안에 영상 파일을 나타내는 태그라는 제어 정보가 포함되어 있으므로 브라우저는 화면에 문장을 표시할 때 태그를 탐색.
- 그리고 영상을 포함하고 있는 의미의 태그를 만나면 그곳에 영상용 공백을 비워두고 문장을 표시.
- 이후 다시 한 번 웹 서버에 접속하여 태그에 쓰여있는 영상 파일을 웹 서버에서 읽어와서 방금 전에 비워 둔 공백에 표시.
- 이 경우 문장 파일을 읽을 때와 마찬가지로 URI 부분에 영상 파일의 이름을 쓴 요청 메시지를 만들어 전송.
- 요청 메시지에 쓰는 URI는 한 개만으로 결정되어 있으므로 파일을 한 번에 한 개씩만 읽을 수 있기 때문에 파일을 따로따로 읽어야 한다.

1. HTTP 리퀘스트 메시지

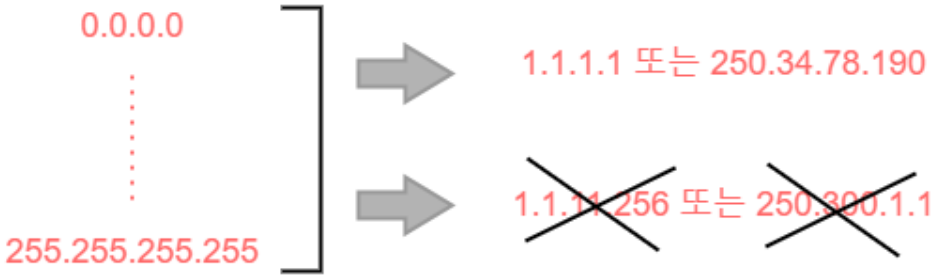
HTTP 응답 메시지 수신

- 이렇게 해서 필요한 파일을 판단하고 이것을 읽은 후 레이아웃을 정하여 화면에 표시하는 상태로 전체의 동작을 조정하는 것도 브라우저의 역할이다.
- 웹 서버측은 이러한 사정을 모릅니다. 4회의 요청이 한 개의 페이지인지, 아니면 각각 별도의 페이지인지 전혀 신경쓰지 않는다.
- 단순히 한 개의 요청에 대해 한 개의 응답만 돌려보낼 뿐이다.
- 이것이 브라우저와 웹 서버의 대화의 전체 모습이다.

2. IP 주소 조회

IP 주소의 기본

- **HTTP의 메시지를 만들면 다음에는 이것을 OS에 의뢰하여 웹 서버에게 송신.**
  - 브라우저는 URL을 해석하거나 HTTP 메시지를 만들지만, 메시지를 네트워크에 송출하는 기능은 없으므로 OS에 의뢰하여 송신하는 것이다.
- **이때 URL 안에 쓰여있는 서버의 도메인명에서 IP(Internet Protocol) 주소를 조사해야 한다.**
  - OS에 송신을 의뢰할 때는 도메인명이 아니라 IP 주소로 메시지를 받을 상대를 지정해야 하기 때문이다.
- **IP 주소는 우편 주소와 마찬가지로 중복과 누락을 허용하지 않는다.**
- **그림과 같이 IP 주소 범위는 0.0.0.0 ~ 255.255.255.255 이다.**

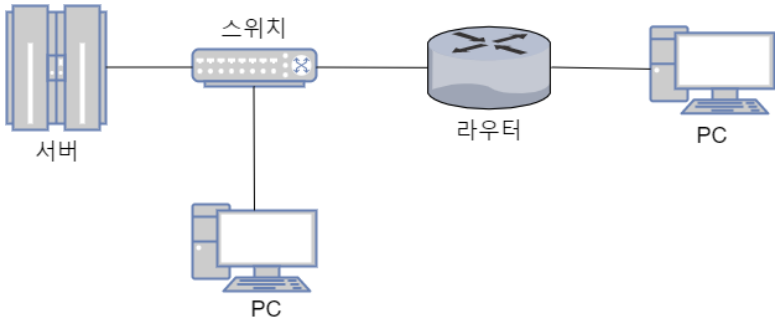


2. IP 주소 조회

IP 주소의 기본

- 그림 35에서 각 장치의 인터 페이스에 할당된 IP 주소는 2.3.6.1과 163.35.7.126이다.
- 이 주소들은 IP 주소 범위에 속하고, 누락되지도 중복되지도 않는다.

- 일반적인 네트워크는 단말들끼리 직접 연결하는 대신, 그림과 같이 스위치나 라우터와 같은 네트워크 장치들을 통해 연결한다.

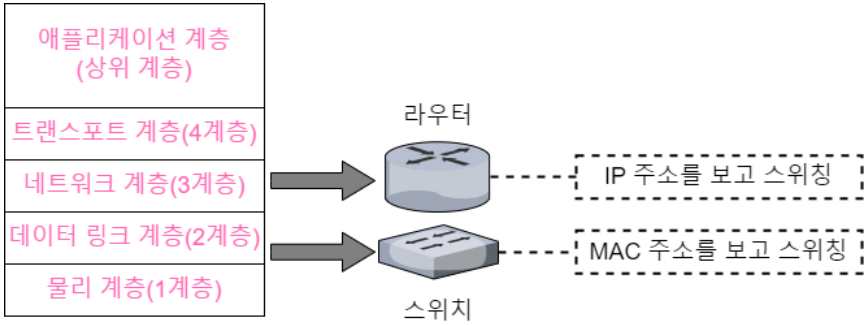


- 라우터와 스위치는 네트워크를 구성하는 핵심 장치이다.

2. IP 주소 조회

IP 주소의 기본

- 그렇다면 라우터와 스위치는 어떤 역할을 할까?
- 라우터와 스위치는 모두 스위칭(switching)을 한다.
- 스위칭 이란, 도착한 패킷의 목적지 주소를 보고 '몇 번 포트로 보낼 것인가?'를 결정한 후 패킷을 그 포트로 보내는 것을 말한다.
- 스위치와 라우터의 유일한 공통점이 스위칭이다.
- 그러면 스위치와 라우터의 차이점은 무엇일까?
- 그림과 같이 첫 번째 차이점은 스위치는 데이터 링크 계층(2계층) 주소 즉 MAC 주소를 보고 스위칭한다는 것이고, 라우터는 TCP/IP 모델을 기준으로 네트워크 계층(3계층) 주소, 즉 IP 주소를 보고 스위칭한다는 것이다.



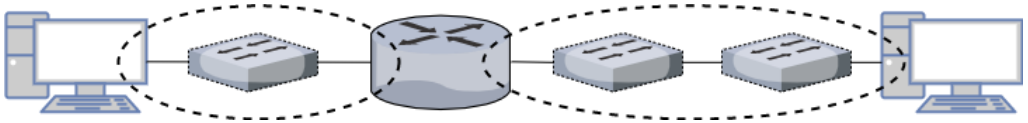
2. IP 주소 조회

IP 주소의 기본

- 아래 그림을 보면, 1개의 네트워크가 보인다.

- 라우터는 네트워크를 분할하는 장치이다.
- 따라서 아래 그림에서 네트워크 수는 2개이다.

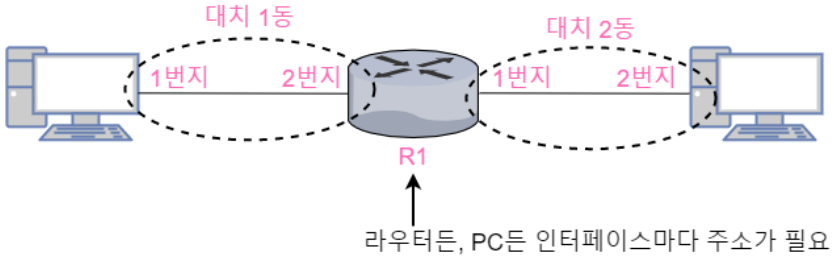
- 반면, 스위치는 네트워크를 분할하지 못한다.
- 따라서 아래 그림과 같이 스위치를 추가해도 네트워크 수는 변함없다.



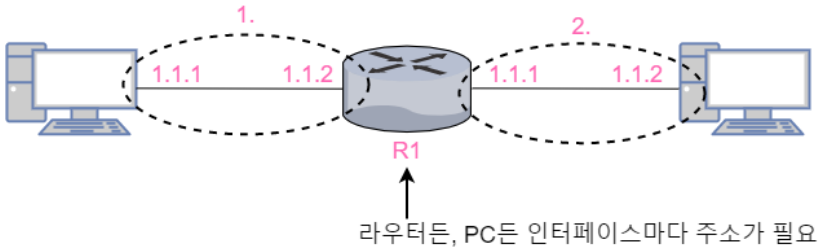
2. IP 주소 조회

IP 주소의 기본

- IP 주소는 우편 주소와 유사
- 차이점은 우편 주소는 집(네트워크로 말하면, 장치)마다 할당하는 반면, IP 주소는 장치의 인터페이스(집으로 말하자면, 문)마다 할당한다는 것이다.



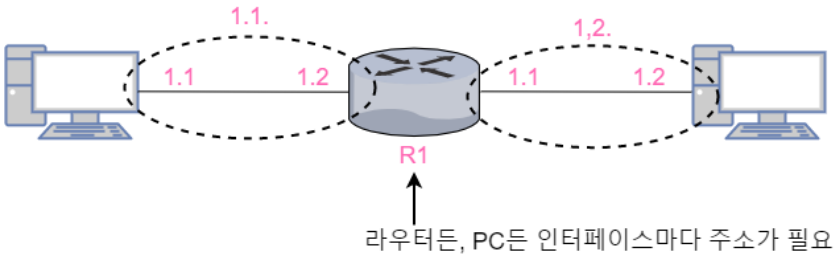
- IP 주소의 4개 옥텟들 중 첫 번째 옥텟만 네트워크를 구분하는 자리 즉 네트워크 자리이다.
- IP 주소의 두 번째, 세 번째, 네 번째 옥텟은 네트워크 내부에서 (장치의)인터페이스를 구분하는 자리, 즉 호스트 자리이다.



2. IP 주소 조회

IP 주소의 기본

- 그런데 IP 주소에서 네트워크와 호스트 자리의 경계는 고정된 것이 아니다.



- 이와 같이 네트워크 자리와 호스트 자리의 경계는 유동적일 수 있다.
- 하지만, 이처럼 경계가 유동적이라면 어떻게 네트워크와 호스트의 자리를 구분할 수 있을까?
- 이 문제를 해결하기 위한 수단이 바로 서브넷 마스크(subnet mask)이다.
- 서브넷 마스크는 IP 주소를 따라다니면서 네트워크 자리와 호스트 자리의 경계를 표시한다.

2. IP 주소 조회

IP 주소의 기본

- 서브넷 마스크의 다른 예

IP 주소/서브넷 마스크	네트워크 자리	호스트 자리
100.101.102.103/255.0.0.0	100	101.102.103
100.101.102.103/255.255.0.0	100.101	102.103
100.101.102.103/255.255.255.0	100.101.102	103

- 그런데 서브넷 마스크는 표와 같이 세 가지 방식으로 표시할 수 있다.

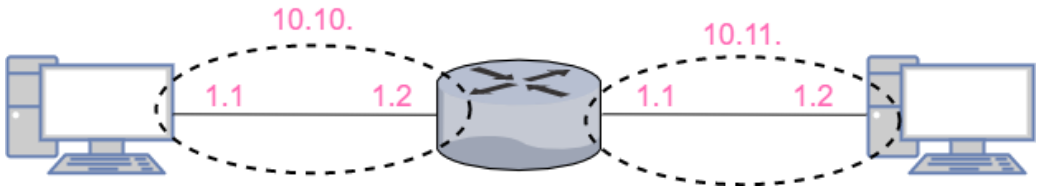
서브넷 마스크 (십진수)	서브넷 마스크 (이진수)	서브넷 마스크 (프리픽스 길이)
255.0.0.0	11111111.00000000.00000000.00000000	/8
255.255.0.0	11111111.11111111.00000000.00000000	/16
255.255.255.0	11111111.11111111.11111111.00000000	/24

2. IP 주소 조회

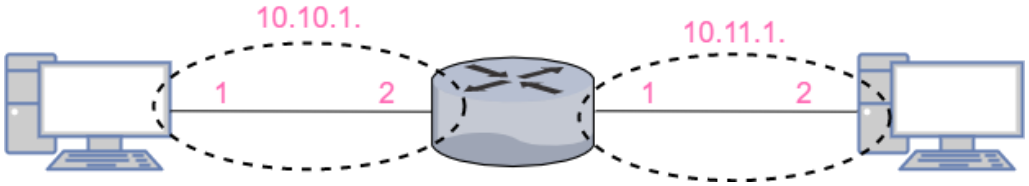
IP 주소의 기본

할당된 IP 주소와 서브넷 마스크의 예

서브넷 마스크가 255.255.0.0이라면



서브넷 마스크가 255.255.255.0이라면



2. IP 주소 조회

IP 주소의 기본

- IP에는 디폴트 서브넷 마스크(Default Subnet Mask) 라는 것이 있다.
- 일반적으로는 IP 주소의 네트워크와 호스트 자리의 경계를 표시하기 위해 서브넷 마스크가 따라다닌다.
- 그러나 서브넷 마스크를 별도로 표기하지 않았을 때는 디폴트(기본적인) 서브넷 마스크를 활용한다.
- IP 주소는 표와 같이 5개의 클래스로 구분된다.

클래스	첫 옥텟 (십진수)	첫 옥텟 (이진수)	디폴트 서브넷 마스크
A	1~126	0xxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx	255.0.0.0
B	128~191	10xxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx	255.255.0.0
C	192~223	110xxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx	255.255.255.0
D	224~239	1110xxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx	해당 없음
E	240~255	1111xxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx	해당 없음

2. IP 주소 조회

도메인 이름과 IP 주소

- TCP/IP의 네트워크는 이 IP 주소로 통신 상대를 지정하므로 IP 주소를 모르면 상대에게 메시지를 전달할 수 없다.
- 그러므로 OS의 메시지 송신을 의뢰할 때는 IP 주소를 조사해야 한다.
- 그렇다면 URL 안에는 서버명이 아니라 IP 주소를 쓰면 좋을 것이라고 생각할 수도 있다.
- 실제 서버명 대신에 IP 주소를 써도 올바르게 작동한다.
- 그러나 전화번호를 기억하기가 어려운 것과 마찬가지로 숫자를 나열한 IP 주소는 기억하기 어렵다.

2. IP 주소 조회

도메인 이름과 IP 주소

- 그렇다면 역으로 IP 주소대신 이름으로 상대를 지정하여 통신하는 방법을 채택하는 것은 어떨까?
- 실행 효율이라는 관점에서 보면 이것은 좋은 방법이라고 할 수 없다.
- IP 주소는 32비트, 즉 4바이트에 해당하는 개수 밖에 없지만, 도메인 이름은 적어도 수십 바이트부터 최대 255바이트나 있다.
- IP 주소라면 4바이트분의 수치만 취급하면 되지만, 도메인 이름이라면 수십 바이트에서 255바이트까지의 문자를 취급해야 한다.
- 그러면 그만큼 라우터가 부하가 걸리게 되고 데이터를 운반하는 동작에 더 많은 시간이 걸리면서 네트워크의 속도가 느려진다.
- 이러한 이유로, 사람은 이름을 사용하고 라우터는 IP 주소를 사용한다는 방법이 고안되었고, 현재 이 방법이 정착되어 있다.
- 이름을 알면 IP 주소를 알 수 있다거나 IP 주소를 알면 이름을 알 수 있다는 원리를 사용하여 양쪽의 차이를 해소하면 모두 좋아지는데, 그 원리가 DNS(Domain Name System)이다.

2. IP 주소 조회

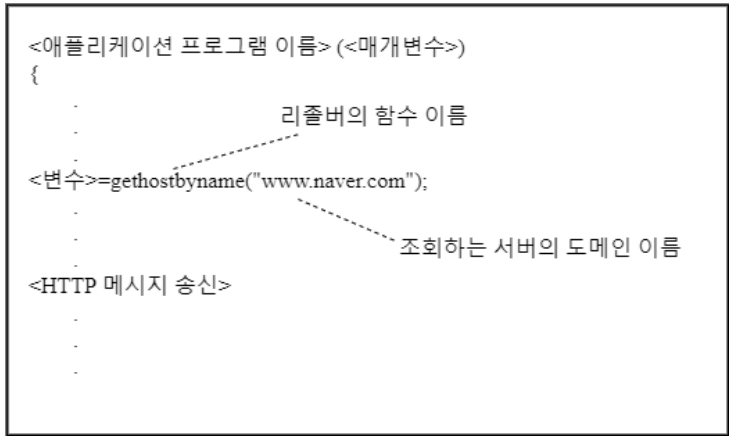
소켓 라이브러리

- IP 주소를 조사하는 방법은 간단하다.
- 즉 가장 가까운 DNS 서버에 'www.naver.com이라는 서버의 IP 주소를 가르쳐 주세요.' 라고 질문하는 것이다.
- 그러면 DNS 서버가 '그 서버의 IP 주소는 xxx.xxx.xxx.xxx 입니다.' 라는 식으로 가르쳐 준다.
- 여기까지는 간단하지만 브라우저는 어떻게 해서 DNS 서버를 조회할 수 있을까?
- DNS 서버에 조회한다는 것은 DNS 서버에 조회 메시지를 보내고, 거기에서 반송되는 응답 메시지를 받는다는 것과 같다.
- 이것은 DNS 서버에 대해 클라이언트로 동작한다고도 말할 수 있다.
- 이 DNS 클라이언트에 해당하는 것을 DNS 리졸버 또는 단순히 리졸버라고 부른다.
- 그리고 DNS의 원리를 사용하여 IP 주소를 조사하는 것을 네임 리졸루션(name resolution, 이름 확인) 이라고 하는데, 이 리졸루션을 실행하는 것이 리졸버(resolver)이다.
- 리졸버의 실체는 Socket 라이브러리에 들어있는 함수이다.
- Socket 라이브러리 속에는 데이터를 송 · 수신할 때 사용하는 함수 이외에 다수의 함수들이 포함되어 있다.

2. IP 주소 조회

DNS 서버 조회

- 브라우저 등의 애플리케이션 프로그램을 만들 때 그림과 같이 리졸버의 함수 이름(gethostbyname)과 웹 서버의 이름(www.naver.com)을 쓰기만 하면 리졸버를 호출할 수 있다.

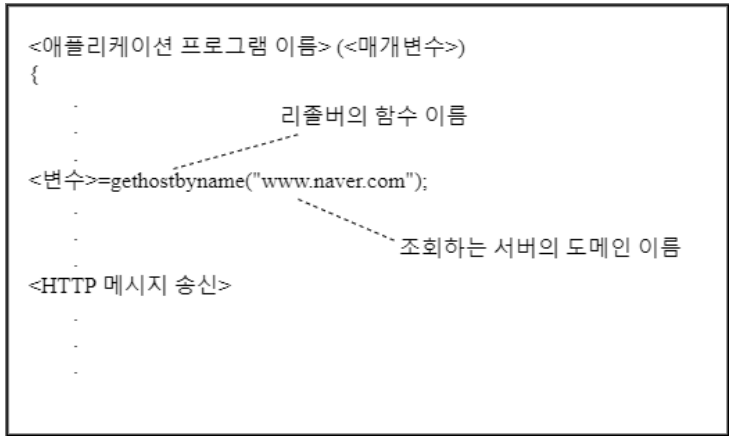


- 이렇게 해서 리졸버를 호출하면 리졸버가 DNS 서버에 조회 메시지를 보내고, DNS 서버에서 응답 메시지가 돌아온다.
- 이 응답 메시지 속에 IP 주소가 포함되어 있으므로 리졸버는 이것을 추출하여 브라우저에서 지정한 변수에 저장한다.
- 리졸버 함수를 실행하면 이 작업이 실행되면서 IP 주소를 조회하는 동작은 끝난다.

2. IP 주소 조회

DNS 서버 조회

- 브라우저 등의 애플리케이션 프로그램을 만들 때 그림과 같이 리졸버의 함수 이름(gethostbyname)과 웹 서버의 이름(www.naver.com)을 쓰기만 하면 리졸버를 호출할 수 있다.



- 이렇게 해서 리졸버를 호출하면 리졸버가 DNS 서버에 조회 메시지를 보내고, DNS 서버에서 응답 메시지가 돌아온다.
- 이 응답 메시지 속에 IP 주소가 포함되어 있으므로 리졸버는 이것을 추출하여 브라우저에서 지정한 변수에 저장한다.
- 리졸버 함수를 실행하면 이 작업이 실행되면서 IP 주소를 조회하는 동작은 끝난다.

2. DNS 서버

DNS 서버의 기본 동작

- **DNS 서버의 기본 동작은 클라이언트에서 조회 메시지를 받고 조회의 내용에 응답하는 형태로 정보를 회답하는 일이다.**
- **조회 메시지에는 다음의 세 가지 정보가 포함되어 있다.**
  - 이름
    - 서버나 메일 배송 목적지(메일 주소에서 @ 뒷부분의 이름)와 같은 이름이다.
  - 클래스
    - DNS의 구조를 고안했을 때 인터넷 이외에도 네트워크에서의 이용까지 검토하여 이것을 식별하기 위해 클래스라는 정보를 준비했다.
    - 그러나 지금은 인터넷 이외의 네트워크는 소멸되었으므로 클래스는 항상 인터넷을 나타내는 'IN' 이라는 값이 된다.
  - 타입
    - 이름에 어떤 타입(종류)의 정보가 지원되는지를 나타냅니다. 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.

2. DNS 서버

DNS 서버의 기본 동작

- DNS 서버에는 이들 세 가지 정보에 대응하여 클라이언트에 회답하는 항목을 등록해 두었다.
- 표와 같은 형태로 이 등록 내용에서 조회 메시지에 해당하는 것을 찾아 클라이언트에게 회답하는 것이다.

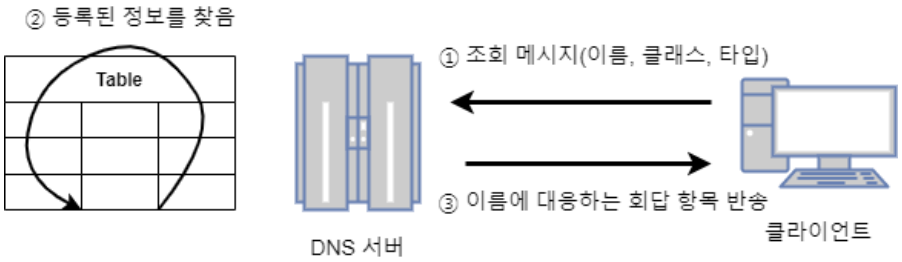
이름	클래스	타입	클라이언트에 회답하는 항목
www.naver.com	IN	A	192.0.2.127
naver.com	IN	MX	10 mail.naver.com
mail.naver.com	IN	A	192.0.2.128
...	...	...	...

- 그러면 DNS 서버는 등록된 정보를 찾아서 이름, 클래스, 타입의 세 가지가 일치하는 것을 찾는다.

2. DNS 서버

DNS 서버의 기본 동작

- 가령 그림의 경우 세 항목 모두 조회 메시지와 일치하는 맨 위의 행이 해당한다.
- 이 들 세 가지 항목이 일치하면 여기에 등록되어 있는 192.0.2.127이라는 값을 클라이언트에 회답한다.
- 또한 웹 서버에는 www.naver.com과 같이 www로 시작하는 이름을 붙인 것이 많지만, 모두 이런 규칙이 있는 것은 아니다.
- 최초에 웹의 구조를 만들 때 'www' 라는 이름으로 웹 서버를 등록한 것이 많았기 때문에 이것이 관례가 된 것뿐이다.
- WebServer1이라고 해도 되고, MySrv라고 해도 되므로 마음에 드는 이름을 붙이고 A라는 타입으로 DNS 서버에 등록하면 그것이 웹 서버의 이름이 된다.



2. DNS 서버

도메인의 계층

- 앞에서는 조회 메시지를 받은 DNS 서버에 이름과 IP 주소가 등록되어 있는 경우를 가정하고 설명했다.
- 사내 네트워크처럼 웹 서버나 메일 서버의 수가 제한되어 있으면 정보를 전부 한 대의 DNS 서버에 등록할 수 있으므로 이 설명처럼 움직일 것이다.
- 그러나 인터넷에는 막대한 수의 서버가 있으므로 이것을 전부 1대의 DNS 서버에 등록하는 것은 불가능하다.
- 그렇기 때문에 조회 메시지를 받은 DNS 서버에 정보가 등록되어 있지 않은 경우도 생긴다.
- 이 경우 DNS 서버가 어떻게 작동할까?
- 결론부터 말하면 정보를 분산시켜서 다수의 DNS 서버에 등록하고, 다수의 DNS 서버가 연대하여 어디에 정보가 등록되어 있는지를 찾아내는 구조이다.

2. DNS 서버

도메인의 계층

- DNS에서 취급하는 이름은 www.naver.com처럼 점으로 구분되어 있는데, 이 점이 계층을 구분한다.
- 회사 조직과 같이 '사업부'에서 '부'라는 호칭을 사용하지 않고 점으로 구분했을 뿐이다.
- 그리고 오른쪽에 위치한 것이 상위의 계층을 나타낸다.
- 그러므로 www.naver.com 이라는 이름은 회사 조직의 이름을 흉내내어 'com사업부 naver부의 www'라는 이름과 같다고 생각하면 된다.
- 그리고 하나의 부서에 해당하는 것을 도메인이라고 한다.
- 그러므로 com이라는 도메인의 아래에 naver라는 도메인이 있으며, 그 도메인의 안에 www라는 이름이 있는 셈이 된다.
- 이렇게 계층화된 도메인의 정보를 서버에 등록하는데, 이때 하나의 도메인을 일괄적으로 취급한다.

2. DNS 서버

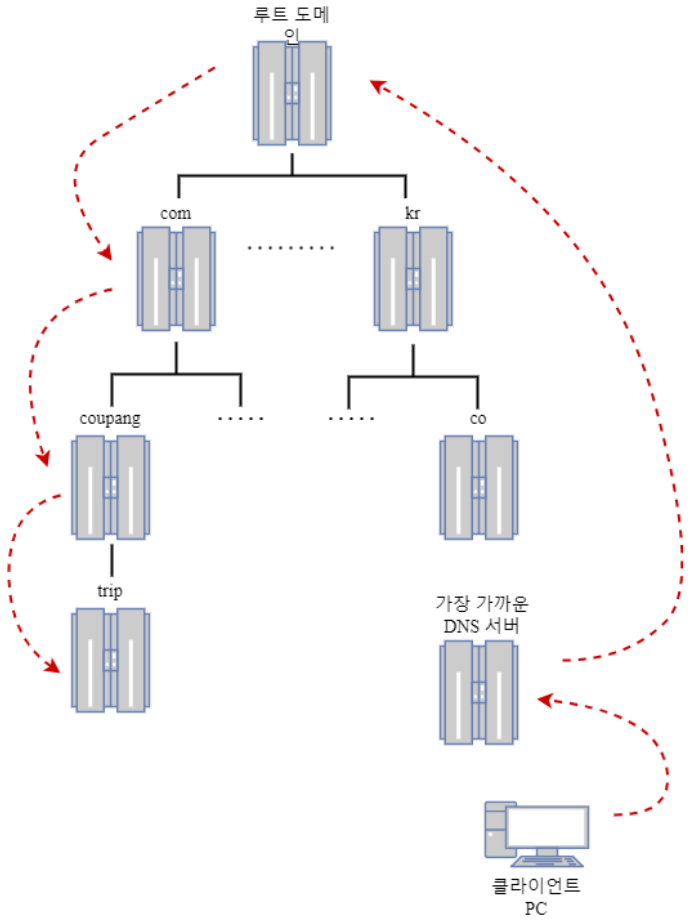
DNS 서버에서 IP 주소 조회

- 다음은 DNS 서버에 등록한 정보를 찾아내는 방법이다.
- 여기에서 중요한 것은 접속하려는 웹 서버가 어느 DNS 서버에 등록되어 있는지를 찾아내는 방법이다.
- 인터넷에는 DNS 서버가 수만 대나 있으므로 닥치는 대로 뒤지면서 다닐 수는 없다.
- 그래서 다음과 같은 방법을 고안했다.
- 먼저 하위의 도메인을 담당하는 DNS 서버의 IP 주소를 그 상위의 DNS 서버에 등록한다.
- 그리고 상위의 DNS 서버를 또 그 상위의 DNS 서버에 등록하는 식으로 차례대로 등록한다.
- 즉 trip.coupang.com이라는 도메인을 담당하는 DNS 서버를 coupang.com의 DNS 서버에 등록하고, coupang.com의 DNS 서버를 com 도메인의 DNS 서버에 등록하는 식이다.
- 이렇게 하면서 상위의 DNS 서버에 가면 하위의 DNS 서버의 IP 주소를 알 수 있고, 거기에서 조회 메시지를 보낼 수 있다.
- 위의 설명에서는 com이나 kr라는 도메인(이것을 최상위 도메인이라 함)의 DNS 서버에 하위의 DNS 서버를 등록한 곳에서 끝나는 것처럼 보이지만, 사실은 그렇지 않다.

2. DNS 서버

DNS 서버에서 IP 주소 조회

- 인터넷의 도메인은 com이나 kr의 상위에 또 하나의 루트 도메인이라는 도메인이 있다.
- 루트 도메인에는 com이나 kr라는 도메인명이 없으므로 보통 도메인을 쓸 때는 그것을 생략한다.
- 하지만 루트 도메인을 명시적으로 써야 하는 경우에는 www.coupang.com. 처럼 끝에 마침표를 찍어서 루트 도메인을 나타내지만 보통은 그렇게 쓰지 않으므로 루트 도메인의 존재를 알아차리지 못한다.
- 이 경우 루트 도메인이 존재하므로 DNS 서버에 com이나 co의 DNS 서버를 등록한다.
- 이렇게 해서 하위의 DNS 서버를 상위의 DNS 서버에 등록하여 루트 도메인에서 차례로 아래쪽으로 거슬러 내려갈 수 있다.



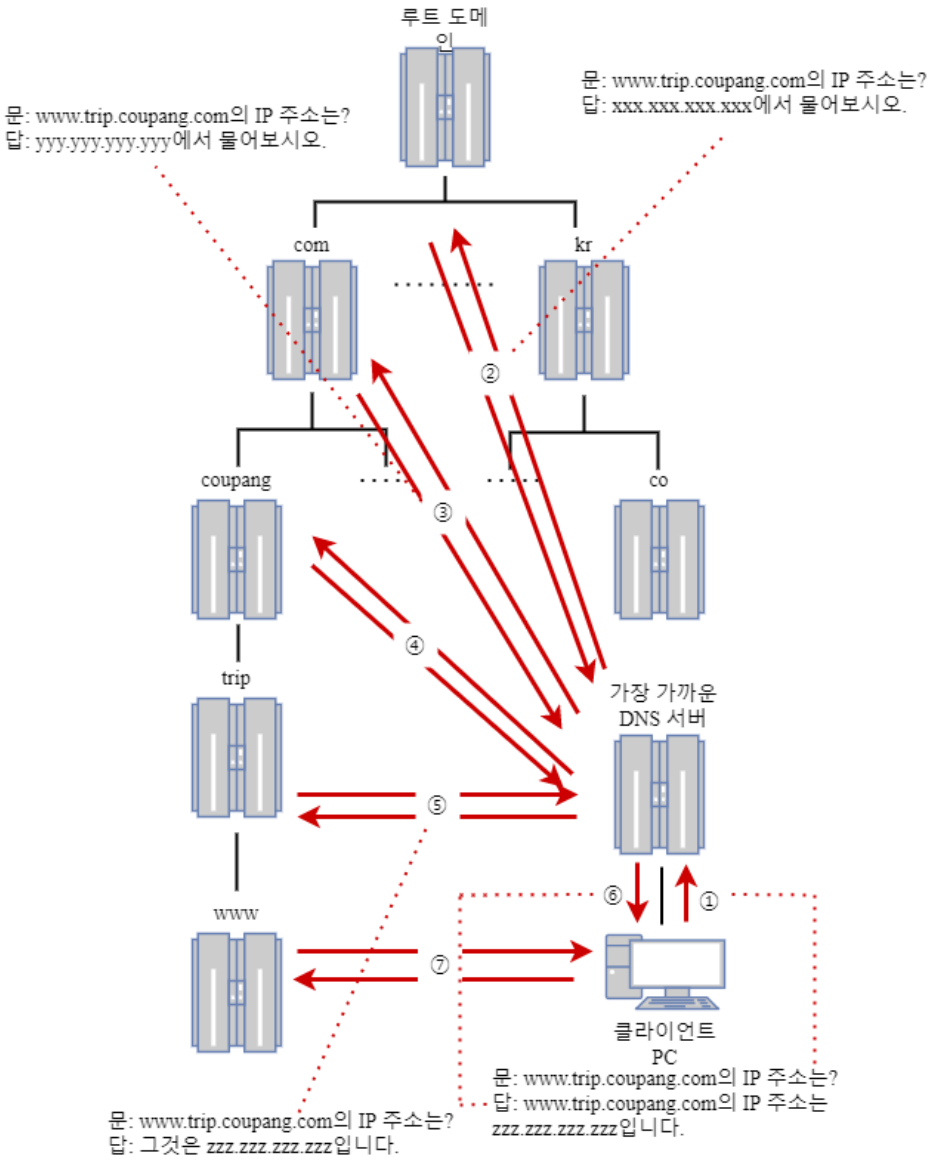
2. DNS 서버

DNS 서버에서 IP 주소 조회

- 등록 작업은 한 가지가 더 있다.
- 루트 도메인의 DNS 서버를 인터넷에 존재하는 DNS 서버에 전부 등록하는 것이다.
- 이렇게 해서 어느 DNS 서버도 루트 도메인에 접속할 수 있게 된다.
- 그 결과 클라이언트에서 어딘가의 DNS 서버에 접속하면 여기에서부터 루트 도메인을 경유하여 도메인의 계층 아래로 찾아가서 최종적으로 원하는 DNS 서버에 도착한다.
- 실제로 루트 도메인의 DNS 서버에 관한 정보는 DNS 서버 소프트웨어와 함께 설정 파일로 배포되어 있으므로 DNS 서버 소프트웨어를 설치하면 자동으로 등록이 완료된다.

2. DNS 서버

- **DNS 서버에서 IP 주소 조회**
  - DNS 서버들의 조회 동작



2. DNS 서버

DNS 서버의 캐시 기능

- 앞의 그림은 기본이 되는 동작을 나타낸 것이므로 현실의 인터넷과는 동작이 다른 부분이 있다.
- 현실의 인터넷에서는 한 대의 DNS 서버에 복수 도메인의 정보를 등록할 수 있으므로 각 도메인에 한 대씩 DNS 서버가 존재한다고 단정할 수 없다.
- 그림에서는 도메인마다 따로 DNS 서버를 썼지만 현실에는 상위와 하위의 도메인을 같은 DNS 서버에 등록하는 경우도 있다.
- 이 경우 상위의 DNS 서버에 조회하면 하위 DNS 서버를 한 개 건너뛰고 다시 그 아래의 DNS 서버에 관한 정보가 돌아온다.
- 최상위 루트 도메인에서 차례대로 따라간다는 원칙대로 움직이지 않을 수도 있다.
- DNS 서버는 한 번 조사한 이름을 캐시에 기록할 수 있는데, 조회한 이름에 해당하는 정보가 캐시에 있으면 그 정보를 회답하기 때문이다.
- 그러면 그 위치에서 계층 구조를 아래로 향하여 찾을 수 있다.
- 루트 도메인부터 찾기 시작하는 것보다 이 방법이 더 편리하다.
- 조회한 이름이 도메인에 등록되어 있지 않은 경우에는 이름이 존재하지 않는다는 회답이 돌아오지만 그것을 캐시에 보존할 수도 있다.
- 이렇게 해서 이름이 존재하지 않는 경우에도 빠르게 회답할 수 있다.

2. DNS 서버

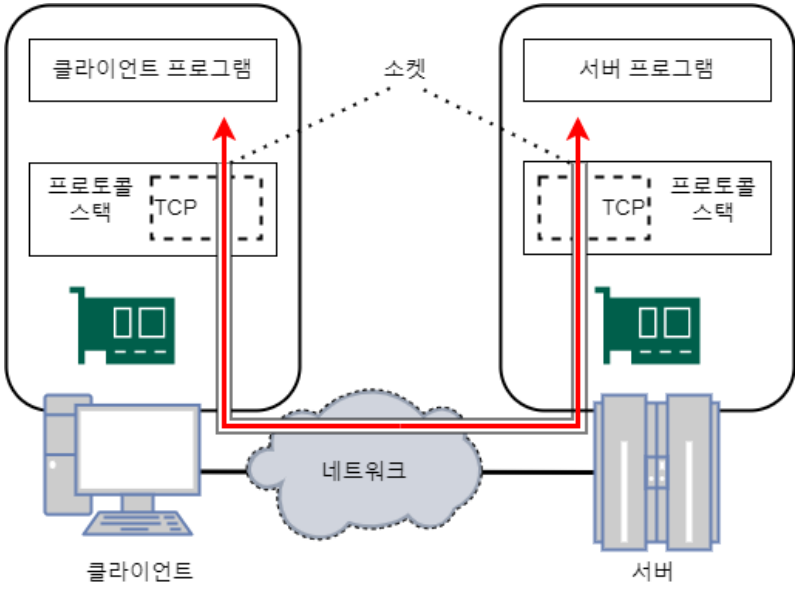
DNS 서버의 캐시 기능

- 이 캐시의 원리에는 한 가지 주의할 점이 있다.
- 캐시에 정보를 저장한 후 등록 정보가 변경되는 경우도 있으므로 캐시 안에 저장된 정보는 올바르다고 단언할 수 없다.
- 따라서 DNS 서버에 등록하는 정보에는 유효 기한을 설정하고, 캐시에 저장한 데이터의 유효 기간이 지나면 캐시에서 삭제한다.
- 또한 조회에 회답할 때 정보가 캐시에 저장된 것인지, 아니면 등록처 DNS 서버에서 회답한 것인지 알려준다.

3. 프로토콜 스택에서의 메시지 송신

데이터 송 · 수신 동작의 개요

- IP 주소를 조사했으면 IP 주소의 상대 여기에서는 액세스 대상 웹 서버에 메시지를 송신하도록 OS의 내부에 있는 프로토콜 스택에 의뢰한다.
- 메시지를 송 · 수신하는 동작은 브라우저뿐만 아니라 네트워크를 이용하는 애플리케이션 전체에 공통이다.
- 소켓 라이브러리를 이용한 데이터 송 · 수신 동작은 그림과 같다.



3. 프로토콜 스택에서의 메시지 송신

데이터 송 · 수신 동작의 개요

- 데이터를 송 · 수신할 때의 개념은 이것뿐이지만 중요한 것이 한 가지 더 있다.
- 이 그림만 보면 최초부터 파이프가 존재하는 것처럼 보이지만, 사실은 송 · 수신 동작을 하기 전에 송 · 수신하는 양자 사이를 파이프로 연결하는 동작이 필요하다는 점이다.
- 이 부분의 요점은 파이프의 양끝에 있는 데이터의 출입구이다.
- 이 출입구를 소켓이라고 부르는데, 우선 이 소켓을 만들고 연결한다.
- 실제로는 먼저 서버 측에서 소켓을 만들고, 소켓에 클라이언트가 파이프를 연결하기를 기다린다.
- 이렇게 해서 서버측이 기다리고 있는 동안에 클라이언트측에서 파이프를 연결한다.
- 데이터 송 · 수신 동작은 이와 같이 몇 단계로 나누어져 있는데, 다음의 네 단계로 요약할 수 있다.

① 소켓을 만듭니다(소켓 작성 단계).

② 서버측의 소켓에 파이프를 연결합니다(접속 단계).

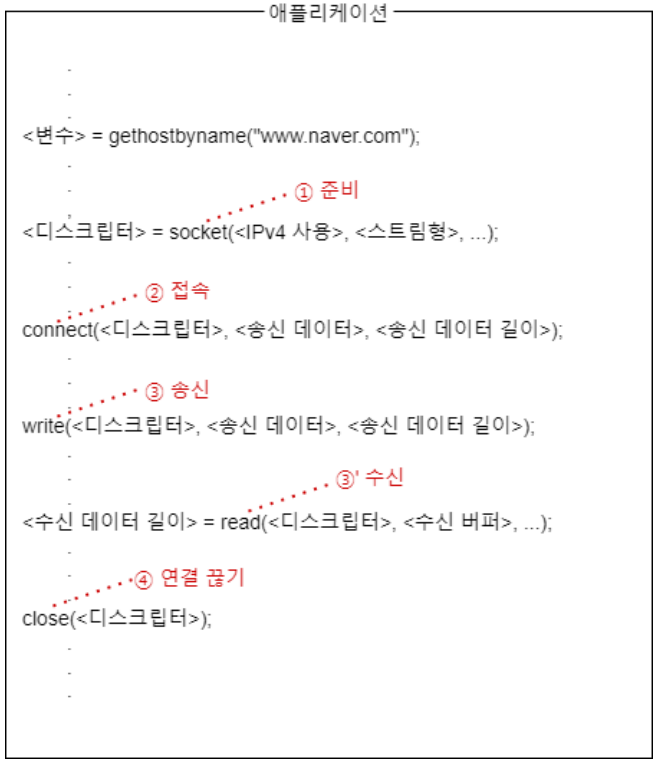
③ 데이터를 송 · 수신합니다(송 · 수신 단계).

④ 파이프를 분리하고 소켓을 말소합니다(연결 끊기 단계).

3. 프로토콜 스택에서의 메시지 송신

소켓의 작성 단계

- 데이터 송 · 수신을 의뢰하는 애플리케이션 프로그램(브라우저)의 동작은 그림과 같다.



3. 프로토콜 스택에서의 메시지 송신

파이프를 연결하는 접속 단계

- **만든 소켓을 서버측의 소켓에 접속하도록 프로토콜 스택에 의뢰한다.**
- **애플리케이션은 소켓 라이브러리의 connect라는 함수를 호출하여 이 의뢰 동작을 실행한다.**
- **여기에서의 요점은 connect를 호출할 때 지정하는 디스크립터, 서버의 IP 주소, 포트 번호라는 세 가지 값이다.**
  - 디스크립터
    - 최초의 디스크립터는 소켓을 만들 때 돌아온 디스크립터이다.
    - 여기에서 지정한 디스크립터는 connect가 프로토콜 스택에 통지한다.
    - 그리고 프로토콜 스택이 통지받은 디스크립터를 보고 어느 소켓을 서버측의 소켓에 접속할지 판단하여 접속 동작을 실행한다.
  - IP 주소
    - 다음으로 IP 주소는 DNS 서버에 조회하여 조사한 액세스 대상 서버의 IP 주소이다.
  - 포트 번호
    - IP 주소로 지정할 수 있는 것은 네트워크의 어느 컴퓨터인가 하는 것까지이다.
    - 접속 동작은 상대측의 소켓에 대해 이루어지므로 소켓을 지정해야 하는데, IP 주소로는 소켓을 지정할 수 없다.
    - 전화를 걸었을 때 "00 님 계십니까?"라고 말하여 통화할 상대를 바꿔달라는 중간 과정이 필요한데, 이 중간과정이 포트 번호이다.
    - IP 주소와 포트 번호의 두 가지를 지정해야 어느 컴퓨터의 어느 소켓과 접속할지를 분명히 지정할 수 있다.

3. 프로토콜 스택에서의 메시지 송신

파이프를 연결하는 접속 단계

- 포트 번호가 접속 상대측에서 소켓을 지정하기 위해 사용하는 것이라면 서버에서도 클라이언트측의 소켓의 번호가 필요할 텐데 이 부분은 어떻게 되어 있을까?
- 이것은 다음과 같이 된다.
- 먼저 클라이언트측의 소켓의 포트 번호는 소켓을 만들 때 프로토콜 스택이 적당한 값을 골라서 할당한다.
- 그리고 이 값을 프로토콜 스택이 접속 동작을 실행할 때 서버측에 통지한다.
- connect를 호출하면 프로토콜 스택이 접속 동작을 실행한다.
- 그리고 상대와 연결되면 프로토콜 스택은 연결된 상대의 IP 주소나 포트 번호 등의 정보를 소켓에 기록한다.
- 이로써 데이터 송 · 수신이 가능한 상태가 된다.

3. 프로토콜 스택에서의 메시지 송신

메시지를 주고받는 송 · 수신 단계

- 애플리케이션은 송신 데이터를 메모리에 준비한다.
- 사용자가 입력한 URL을 바탕으로 만든 HTTP의 리퀘스트 메시지가 여기에서 말하는 송신 데이터이다.
- 그리고 소켓 라이브러리의 write를 호출할 때 디스크립터와 송신 데이터를 지정한다.
- 그러면 프로토콜 스택이 송신 데이터를 서버에게 송신한다.
- 소켓에는 연결된 상대가 기록되어 있으므로 디스크립터로 소켓을 지정하면 연결된 상대가 판명되어 그곳을 향해 데이터를 송신한다.
- 이 때 송신 데이터는 네트워크를 통해 전부 그대로 액세스 대상의 서버에 도착한다.
- 그러면 서버는 수신 동작을 실행하여 받은 데이터의 내용을 조사하고 적절한 처리를 실행하여 응답 메시지를 반송한다.

3. 프로토콜 스택에서의 메시지 송신

메시지를 주고받는 송 · 수신 단계

- 이 메시지가 돌아오면 이번에는 메시지를 수신하는 동작이다.
- 수신할 때는 소켓 라이브러리의 read라는 함수를 통해 프로토콜 스택에 수신 동작을 의뢰한다.
- 이 때 수신한 응답 메시지를 저장하기 위한 메모리 영역을 지정하는데 이 메모리 영역을 수신 버퍼라고 부른다.
- 그러면 응답 메시지가 돌아올 때 read가 받아서 수신 버퍼에 저장한다.
- 수신 버퍼는 애플리케이션 프로그램의 내부에 마련된 메모리 영역이므로 수신 버퍼에 메시지를 저장한 시점에서 메시지를 애플리케이션에 건네준다.

3. 프로토콜 스택에서의 메시지 송신

연결 끊기

- 브라우저가 데이터 수신을 완료하면 송 · 수신 동작은 끝난다.
- 그 후 소켓 라이브러리의 close라는 함수를 호출하여 연결 끊기 단계로 들어가도록 의뢰한다.
- 그러면 소켓 사이를 연결한 파이프와 같은 것이 분리되고 소켓도 말소된다.
- 이때의 동작은 다음과 같이 된다.
- 웹에서 사용하는 HTTP 프로토콜에서는 본래 응답 메시지의 송신을 완료했을 때 웹 서버 측에서 연결 끊기 동작을 실행하므로 먼저 웹 서버 측에서 close를 호출하여 연결을 끊는다.
- 그러면 이것이 클라이언트측에 전달되어 클라이언트의 소켓은 연결 끊기 단계로 들어간다.
- 그리고 브라우저가 read로 수신 동작을 의뢰했을 때 read는 수신한 데이터를 건네주는 대신 송 · 수신 동작이 완료되어 연결이 끊겼다는 사실을 브라우저에 통지한다.
- 이로써 송 · 수신이 종료되었다는 것을 알 수 있으므로 브라우저에서도 close를 호출하여 연결 끊기 단계에 들어간다.

3. 프로토콜 스택에서의 메시지 송신

연결 끊기

- 이것이 HTTP의 본래 동작이다.
- HTTP 프로토콜은 HTML 문서나 영상 데이터를 하나하나 별도의 것으로 취급하여 1개의 데이터를 읽을 때마다 접속, 요청 메시지 송신, 응답 메시지 수신, 연결 끊기라는 동작을 반복한다.
- 따라서 하나의 웹 페이지에 영상이 많이 포함되어 있으면 접속, 송 · 수신, 연결 끊기 동작을 여러 번 반복한다.
- 그러나 같은 서버에서 복수의 데이터를 읽을 때 접속과 연결 끊기를 반복하는 것은 비효율적이므로 한 번 접속한 후 연결을 끊지 않고 복수의 요청과 응답 주고받기를 실행하는 방법도 나중에 마련되었다.
- HTTP의 버전 1.1에서 이 방법을 사용할 수도 있다.
- 이 경우 요청해야 할 데이터가 없어진 상태에서 브라우저에서 연결 끊기 동작에 들어갈 수 있다.

Thank You