

2023-08-13 (Chapter 01)

- 1장. 웹 브라우저가 메세지를 만든다_웹 브라우저의 내부 탐험
 - 브라우저의 동작을 처음 추적.
 - 브라우저에 URL을 입력. (ex: <http://www.lab.cyber.co.kr/sample1.html>)
 - 브라우저는 입력된 URL을 결정된 규칙에 따라 URL의 의미를 조사한 후 그 의미에 따라 리퀘스트 메세지를 만든다.
 - <http://www.lab.cyber.co.kr/sample1.html>
 - sample1.html이라는 파일에 저장된 페이지의 데이터를 요구.
 - 이런 의미의 리퀘스트 메세지를 만들고, 웹 서버에 보낸다.
 - but 브라우저 자체가 메세지를 보내는 것이 아니다.
 - 메세지를 보내는 것은 디지털 데이터를 운반하는 구조의 역할.
(OS에 내장된 네트워크 제어용 소프트웨어에 의뢰하여 메세지를 서버측까지 도착하게 함)
 - 1장에서는 ‘의뢰하는 동작’까지 추적.
- ▼ Story.01 HTTP 리퀘스트 메세지를 작성한다.
 - 출발점은 브라우저에 URL을 입력. URL을 해석하는 곳부터 브라우저의 동작이 시작됨.
 - URL의 의미에 따라 리퀘스트 메세지를 만든다.
 - 브라우저는 리퀘스트 메세지에 따라 웹 서버에 무엇을 하려는지 전달.
 - 구체적인 메세지의 모습과 의미를 알면 웹 서버에 액세스할 때 사용하는 HTTP 프로토콜 학습.
- ▼ 1. 탐험 여행은 URL 입력부터 시작한다.
 - URL에는 http:, ftp:, file:, mailto: (메일 소프트 웨어를 설치해야함) 등 여러 가지 종류가 있다.

- 다양한 URL이 있는 이유는 브라우저가 웹 서버에 액세스하는 기능뿐 아니라, 파일 다운로드, 업로드, 메일의 클라이언트 기능도 가지고 있기 때문.
- 브라우저는 몇 개의 클라이언트 기능을 결합한 복합적인 클라이언트 소프트웨어라고 할 수 있다.
- 따라서 몇 개의 기능 중 어느 것을 사용할지 기능 선택에 URL을 이용.

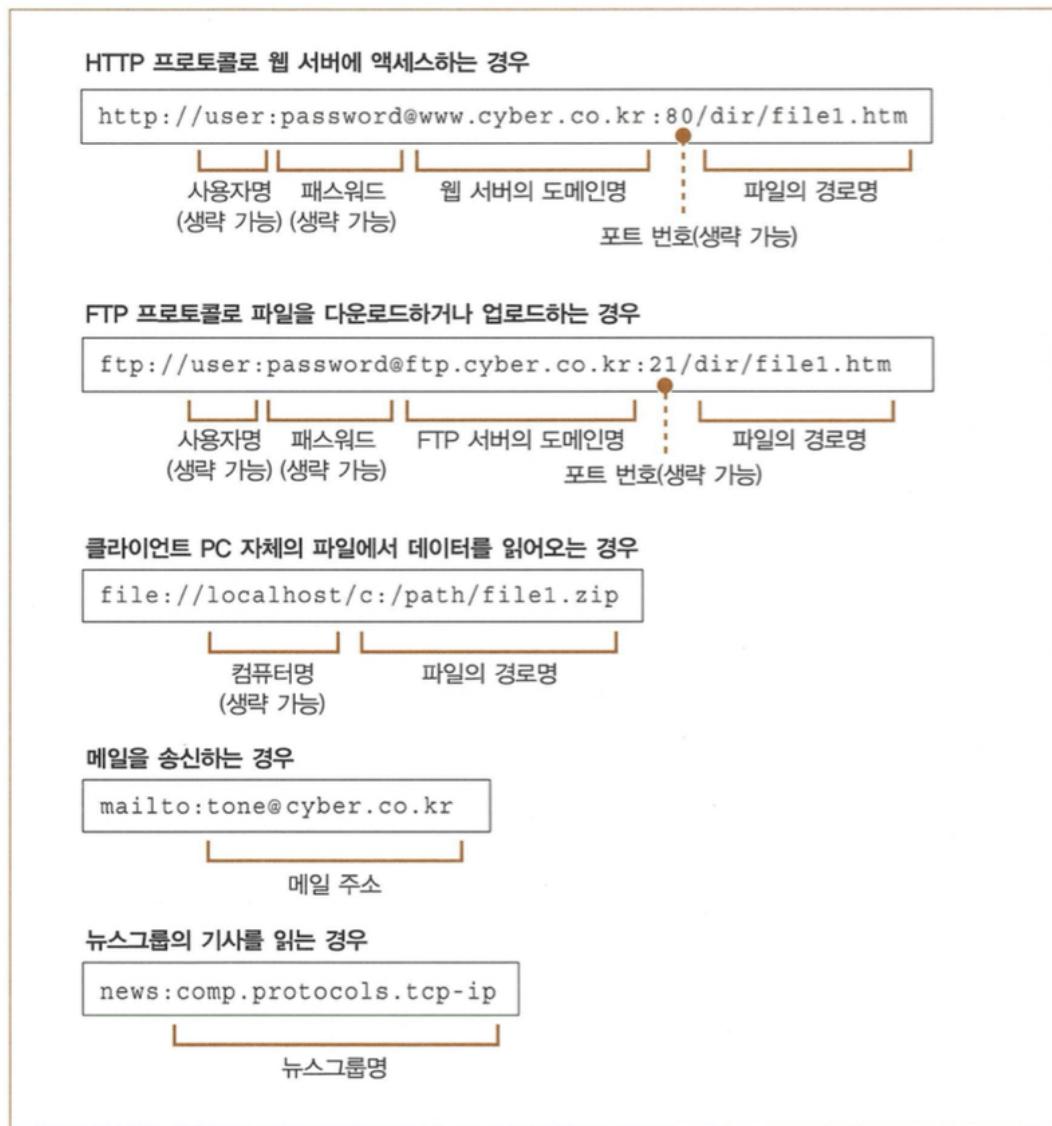


그림 1-1 각종 URL의 형식

- 모든 URL이 맨 앞 문자열 (http:, ftpL, file:, mailto...) 부분에 액세스 방법을 나타냄.
- 각 Http, ftp 프로토콜을 사용하여 액세스하지만 file은 네트워크를 사용하지 않는 것도 있으므로 무조건 프로토콜이라기 보단, 액세스 방법이라는 식으로 생각하는 것이 낫다.

▼ 2. 브라우저는 먼저 URL을 해독한다.

- 브라우저가 처음 하는 일은 웹 서버에 보내는 리퀘스트의 메세지를 작성하기 위한 URL 해석.
 - URL 형식은 프로토콜에 따라 다르므로 여기서는 웹 서버에 액세스하는 경우.

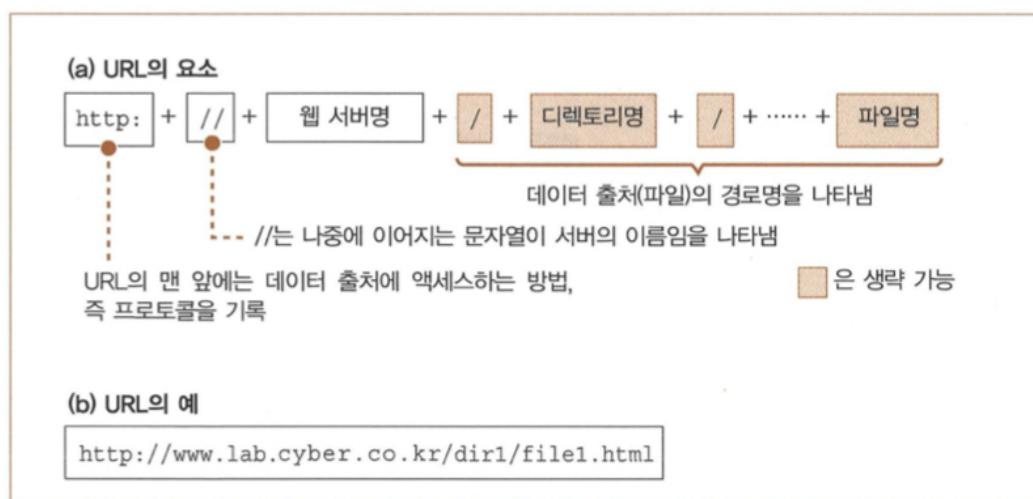


그림 1-2 웹 브라우저가 URL을 해독하는 흐름

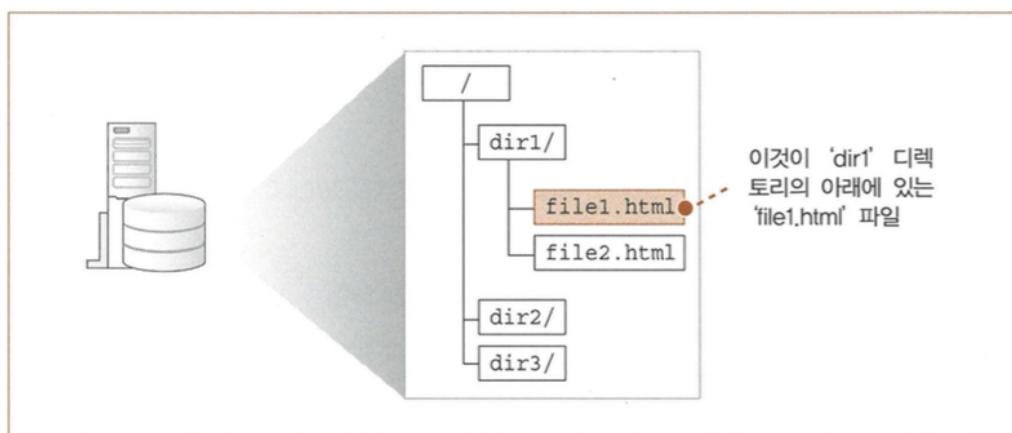


그림 1-3 '/dir1/file1.html'이라는 경로명의 파일

- URL은 결국 몇 개의 요소를 나열한 것.
- 때문에 URL을 해독할 때는 먼저 요소를 따로따로 분해한다. (c)
- (c)를 보면 URL은 (웹 서버 명 + 리소스 경로)로 볼 수 있다.

▼ 3. 파일명을 생략한 경우

- HTTP URL 예제.
 - `http://www.lab.cyber.co.kr/dir1/file1.html` (a)
 - `http://www.lab.cyber.co.kr/dir/` (b)
 - `http://www.lab.cyber.co.kr(/)` (c)
 - `http://www.lab.cyber.co.kr/whatisthis` (d)
- (b)와 같이 끝이 /로 끝난 것은 /dir/ 다음에 써야 할 파일명을 쓰지 않고 생략함.
- URL의 규칙에는 이와 같이 파일명을 생략해도 좋다.
- but 파일명을 쓰지 않으면 어느 파일에 액세스 할 지 모름.
그래서 파일명을 생략할 때를 대비해서 파일명을 미리 서버측에 설정해둠.
(대부분 `index.html`, `default.html`) → `/dir/index.html`, `/dir/default.html`
- (c)의 경우 끝에 /가 있는 경우 루트 디렉토리를 의미 `/index.html`, `/default.html` 파일 액세스. (c)에서 /가 없는 경우도 마찬가지
- (d)의 경우 `whatisthis`는 파일 명, 디렉토리 명, 둘 다 가능하다. 웹 서버에 `whatisthis`라는 파일이 있으면 파일명으로 보고, 디렉토리가 있으면 디렉토리 명으로 본다.

▼ 4. HTTP의 기본 개념

- URL을 해독하면 액세스 위치가 판명됨.
- HTTP 프로토콜은 클라이언트 서버가 주고받는 메세지의 내용이나 순서를 정한 것.
- 클라이언트에서 서버를 향해 리퀘스트 메세지를 보냄. (무엇, 어떻게.. 내용)

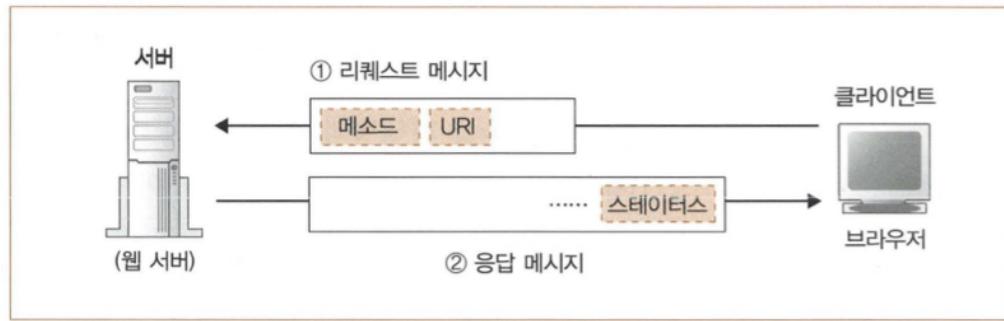


그림 1-4 HTTP의 기본 개념

- 무엇이라는 내용은 URI라고 하는데, 보통 페이지 데이터를 저장한 파일의 이름이나 CGI 프로그램의 파일명을 URI로 쓴다.
- 어떻게 해서에 해당하는 것은 메소드. 어떤 동작을 하고 싶은지를 전달한다.
- 리퀘스트 메세지가 웹 서버에 도착하면 웹 서버는 내용을 해독. 무엇을, 어떻게 를 해독 판단한 후 요구에 따라 동작하고, 결과 데이터를 응답 메세지에 저장한다.

코드값	설명
1xx	처리의 경과 상황 등을 통지합니다.
2xx	정상 종료
3xx	무언가 다른 조치가 필요함을 나타냅니다.
4xx	클라이언트측의 오류
5xx	서버측의 오류

표 1-3 HTTP의 스테이터스 코드의 개요

스테이터스 코드는 첫 번째 행에 개요를 나타내고, 두 번째와 세 번째 행에 상세한 상황을 나타냅니다. 표는 첫 번째 행의 의미를 정리한 것입니다.

- 응답 메세지의 맨 앞부분에는 실행 결과가 정상 종료되었는지 이상이 발생했는지를 나타내는 스테이터스 코드가 있다.
- 이런 응답 메세지를 클라이언트에 반송, 도착하여 브라우저가 메세지의 안에서 데이터를 추출하여 화면에 표시하면 HTTP의 동작이 끝남.

메소드	HTTP의 버전		의미
	1.0	1.1	
GET	○	○	URI로 지정한 정보를 도출합니다. 파일의 경우 해당 파일의 내용을 되돌려 보내고, CGI 프로그램의 경우 해당 프로그램의 출력 데이터를 그대로 반송 합니다.
POST	○	○	클라이언트에서 서버로 데이터를 송신합니다. 폼에 입력한 데이터를 송신하는 경우에 사용합니다.
HEAD	○	○	GET과 거의 같습니다. 단 HTTP 메시지 헤더만 반송하고 데이터의 내용을 돌려보내지 않습니다. 파일의 최종 생성 일시와 같은 속성 정보를 조사할 때 사용합니다.
OPTIONS		○	통신 옵션을 통지하거나 조사할 때 사용합니다.
PUT	△	○	URI로 지정한 서버의 파일을 치환합니다. URI로 지정한 파일이 없는 경우에는 새로 파일을 작성합니다.
DELETE	△	○	URI로 지정한 서버의 파일을 삭제합니다.
TRACE		○	서버측에서 받은 리퀘스트 라인과 헤더를 그대로 클라이언트에 반송합니다. 프록시 서버 등을 사용하는 환경에서 리퀘스트가 치환되는 상태를 조사할 때 사용합니다.
CONNECT		○	암호화한 메시지를 프록시로 전송할 때 이용하는 메소드입니다.

● 표 1-1 HTTP의 주요 메소드-버전 1.0은 RFC1945이며, 버전 1.1은 RFC2626으로 기술된 내용을 기초로 합니다.

○ : 각 버전에서 '사용'으로 정의되어 있는 것

△ : 정식으로는 사양이 아니라 부가 기능으로 사양의 부록에 기술되어 있는 것

- **Get** : 웹 서버에 액세스하여 페이지의 데이터를 읽을 때 사용.
- 일반적으로 액세스 동작은 먼저 리퀘스트 메세지의 메소드에 Get, URI에는 /dir1/file1.html과 같이 페이지의 데이터를 저장한 파일의 이름을 씀.
(get + /dir1/file1.html 이라는 파일의 데이터를 읽으라는 의미)
- **Post** : 폼에 데이터를 사용해서 웹 서버에 송신하는 경우 사용.
- 입력 필드 부분이 폼 형태. 폼에 입력된 데이터를 전송해 응답 메세지를 받음.

▼ 5. HTTP 리퀘스트 메세지를 만든다.

- URL을 해독하고 웹 서버와 파일명을 판단하면 브라우저는 이것을 바탕으로 HTTP의 리퀘스트 메세지를 만듦.

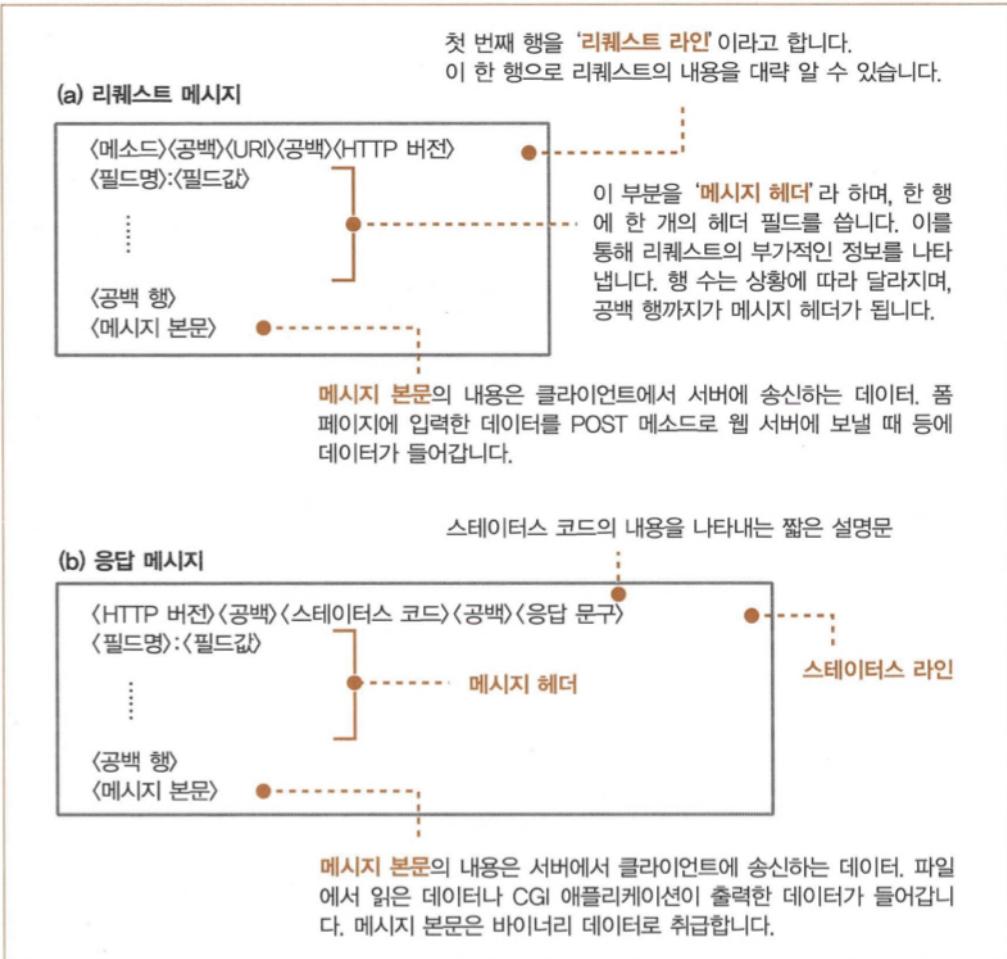


그림 1-5 HTTP 메시지의 포맷

브라우저나 웹 서버는 이 포맷에 맞게 메시지를 만듭니다.

- HTTP는 포맷이 결정되어 있으므로 브라우저는 이 포맷에 맞게 리퀘스트 메세지를 만듭니다.

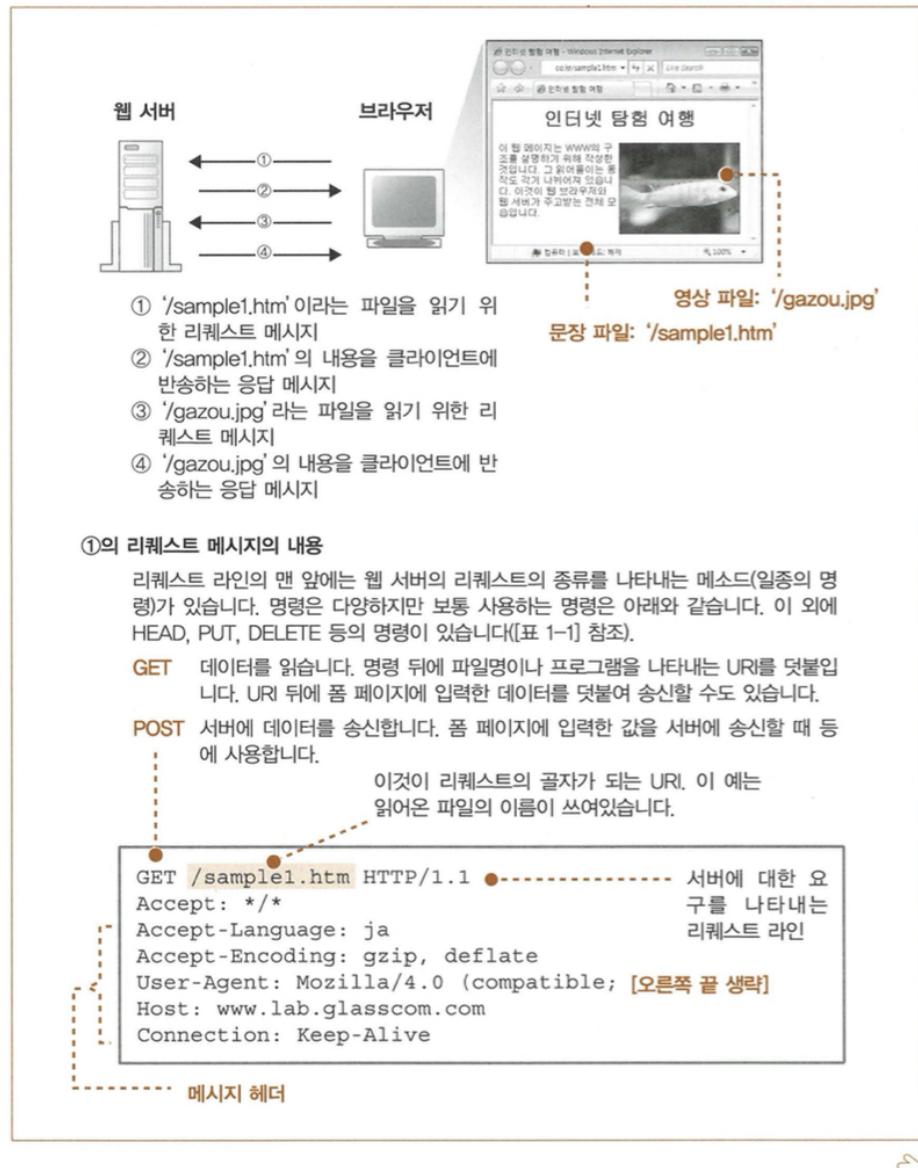
헤더 필드의 종류	HTTP의 버전		설명
	1.0	1.1	
제너럴 헤더 : 리퀘스트와 응답 양쪽에 모두 사용하는 헤더 필드			
Date	<input type="radio"/>	<input type="radio"/>	리퀘스트나 응답이 작성된 날짜를 나타냅니다.
Pragma	<input type="radio"/>	<input type="radio"/>	데이터의 캐시를 허용할지의 여부를 나타내는 통신 옵션을 지정합니다.
Cache-Control		<input type="radio"/>	캐시를 제어하기 위한 정보
Connection		<input type="radio"/>	응답 송신 후에 TCP에 계속 접속할지, 아니면 연결을 끊을지를 나타내는 통신 옵션을 지정합니다.
Transfer-Encoding		<input type="radio"/>	메시지 본문의 인코딩 방식을 나타냅니다.
Via		<input type="radio"/>	도중에 경유한 프록시나 게이트웨이를 기록합니다.
리퀘스트 헤더 : 리퀘스트의 부가 정보로 사용하는 헤더 필드			
Authorization	<input type="radio"/>	<input type="radio"/>	사용자 인증용 데이터
From	<input type="radio"/>	<input type="radio"/>	리퀘스트 발신자의 메일 주소
If-Modified-Since	<input type="radio"/>	<input type="radio"/>	지정한 날짜 이후 정보가 갱신된 경우에만 리퀘스트를 실행하려고 필드값으로 이 날짜를 지정합니다. 보통 클라이언트측에서 캐시에 저장한 정보를 비교하고, 이것이 오래된 경우에 새 정보를 받으려고 할 때 사용합니다.
Referer	<input type="radio"/>	<input type="radio"/>	하이퍼링크를 거쳐 어느 페이지를 읽은 경우 등에 링크 대상인 URI를 나타냅니다.
User-Agent	<input type="radio"/>	<input type="radio"/>	클라이언트 소프트웨어의 명칭이나 버전에 관한 정보
Accept	<input type="triangle"/>	<input type="radio"/>	클라이언트측이 Content-Type으로 받는 데이터의 종류. MIME 사양의 데이터 타입으로 표현한 것입니다.
Accept-Charset	<input type="triangle"/>	<input type="radio"/>	클라이언트측이 받은 문자 코드 세트
Accept-Encoding	<input type="triangle"/>	<input type="radio"/>	클라이언트측이 Content-Encoding으로 받은 인코딩 방식. 보통 데이터 압축 형식을 나타냅니다.
Accept-Language	<input type="triangle"/>	<input type="radio"/>	클라이언트측이 받는 언어의 종류. 한국어는 ko, 영어는 en입니다.
Host		<input type="radio"/>	리퀘스트를 받는 서버의 IP 주소와 포트 번호
If-Match		<input type="radio"/>	Etag 참조
If-None-Match		<input type="radio"/>	Etag 참조
If-Unmodified-Since		<input type="radio"/>	지정한 날짜 이후 갱신되지 않은 경우에만 리퀘스트를 실행합니다.
Range		<input type="radio"/>	데이터의 전체가 아니라 일부만 읽을 때 해당 범위를 지정합니다.



헤더 필드의 종류	HTTP의 버전		설명
	1.0	1.1	
응답 헤더 : 응답의 부가 정보로 사용되는 헤더 필드			
Location	<input type="radio"/>	<input checked="" type="radio"/>	정보의 정확한 장소를 나타냅니다. 리퀘스트의 URI가 상대 이름 (relative name)으로 지정된 경우 절대 이름으로 했을 때의 정보의 위치를 통지하기 위해 사용합니다.
Server	<input type="radio"/>	<input checked="" type="radio"/>	서버 소프트웨어의 명칭이나 버전에 관한 정보
WWW-Authenticate	<input type="radio"/>	<input checked="" type="radio"/>	리퀘스트한(요청한) 정보에 대한 액세스가 제한되어 있는 경우 사용자 인증용 데이터(챌린지 등)를 반송합니다.
Accept-Ranges		<input checked="" type="radio"/>	데이터의 일부만 리퀘스트하는 Range를 지정한 경우 서버가 해당 기능을 가지고 있는지 여부를 클라이언트에 통지합니다.
엔티티 헤더 : 엔티티(메시지 본문)의 부가 정보로 사용하는 헤더 필드			
Allow	<input type="radio"/>	<input checked="" type="radio"/>	지정한 URI로 사용 가능한 메소드를 나타냅니다.
Content-Encoding	<input type="radio"/>	<input checked="" type="radio"/>	메시지 본문에 압축 등의 인코딩 처리가 되어 있는 경우 해당 방식을 나타냅니다.
Content-Length	<input type="radio"/>	<input checked="" type="radio"/>	메시지 본문의 길이를 나타냅니다.
Content-Type	<input type="radio"/>	<input checked="" type="radio"/>	메시지 본문이 어떤 데이터인지 종류를 나타냅니다. MIME 사양으로 정의된 데이터 타입으로 데이터의 종류를 나타냅니다.
Expires	<input type="radio"/>	<input checked="" type="radio"/>	메시지 본문의 유효 기간을 나타냅니다.
Last-Modified	<input type="radio"/>	<input checked="" type="radio"/>	정보를 최종 변경한 일시
Content-Language		<input checked="" type="radio"/>	메시지 본문의 언어를 나타냅니다. 한국어의 경우 ko, 영어의 경우 en입니다.
Content-Location		<input checked="" type="radio"/>	메시지 본문이 서버의 어디에 놓여 있는지 위치를 URI로 나타냅니다.
Content-Range		<input checked="" type="radio"/>	데이터의 전체가 아니라 일부가 리퀘스트된 경우 메시지 본문에 어느 범위의 데이터가 포함되어 있는지를 나타냅니다.
Etag		<input checked="" type="radio"/>	갱신 처리 등에서 이전 리퀘스트의 응답을 바탕으로 한 갱신 데이터를 다음 리퀘스트에서 송신하는 경우가 있는데, 이때 이전 응답과 다음 리퀘스트를 관련시키기 위해 사용하는 정보입니다. 이전 응답에서 서버가 Etag에 따라 고유한 값을 클라이언트에 전네주고, 다음 번 리퀘스트의 'If-Match', 'If-None-Match', 'If-Range' 필드에서 값을 서버에 통지하면 서버는 이전 회의 계속이라고 인식합니다. 쿠키라는 필드와 역할이 같은데, 쿠키는 넷스케이프의 독자적인 시양이며, Etag는 이것을 표준화한 것입니다.

- 두번째 행 부터는 헤더 행.
- 첫 번째 행에서 리퀘스트 내용에 부가적인 정보가 필요한 경우 써 두는 것이 헤더의 역할.
- 헤더 뒤에 공백을 넣고 송신할 데이터를 씀. 이것이 실제 내용.
- 단 메소드가 Get인 경우 메소드와, URI만으로 웹 서버가 무엇을 할지 판단할 수 있으므로 메세지 본문에 쓰는 데이터는 없어도 됨.
- Post인 경우 폼에 입력한 데이터 등을 메세지 본문 부분에 씀.

▼ 6. 리퀘스트 메세지를 보내면 응답이 되돌아온다.



② '/sample1.htm'의 내용을 클라이언트에 반송하는 응답 메시지

서버 소프트웨어의 종류
서버 소프트웨어의 종류

```
HTTP/1.1 200 OK
Date: Wed, 21 Feb 2008 09:19:14 GMT
Server: Apache
Last-Modified: Mon, 19 Feb 2008 12:24:51 GMT
ETag: "5a9da-279-3c726b61"
Accept-Ranges: bytes
Content-Length: 632
Connection: close
Content-Type: text/html

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>인터넷 탐험 여행</title>
</head>

<body>
<h1 align="center">인터넷 탐험 여행</h1>


```

이 웹 페이지는 www의 구조를 설명하기 위해 작성한 것으로, 읽어들이는 동작도 서로 나누어져 있습니다. 이것이 웹 브라우저와 웹 서버가 주고받는 전체 모습입니다.

</body>
</html>

MIME 사양으로 데이터의 형식을 정의한 것으로, text/html은 HTML 문서를 나타냅니다. JPEG 형식의 영상 데이터인 경우에는 이 곳이 image/jpeg가 됩니다.

이것이 포함된 영상 파일의 이름. 다음 리퀘스트에서 이 파일을 서버에서 읽습니다.

그림 1-7 HTTP 메시지의 예

③ '/gazou.jpg'를 읽기 위한 리퀘스트 메시지

```
GET /gazou.jpg HTTP/1.1
Accept: */*
Referer: http://www.lab.cyber.co.kr/sample1.htm
Accept-Language: ja
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; [오른쪽 끝 생략])
Host: www.lab.cyber.co.kr
Connection: Keep-Alive
```

④ '/gazou.jpg'의 내용을 클라이언트에 반송하는 응답 메시지

```
HTTP/1.1 200 OK
Date: Wed, 21 Feb 2008 09:19:14 GMT
Server: Apache
Last-Modified: Mon, 19 Feb 2008 13:50:32 GMT
ETag: "5a9d1-1913-3aefa236"
Accept-Ranges: bytes
Content-Length: 6419
Connection: close
Content-Type: image/jpeg
```

image/jpeg는 JPEG 형식의 영
상 데이터임을 나타냅니다.

[여기부터 영상 데이터가 시작되지만, 바이너리 데이터이므로 생략]

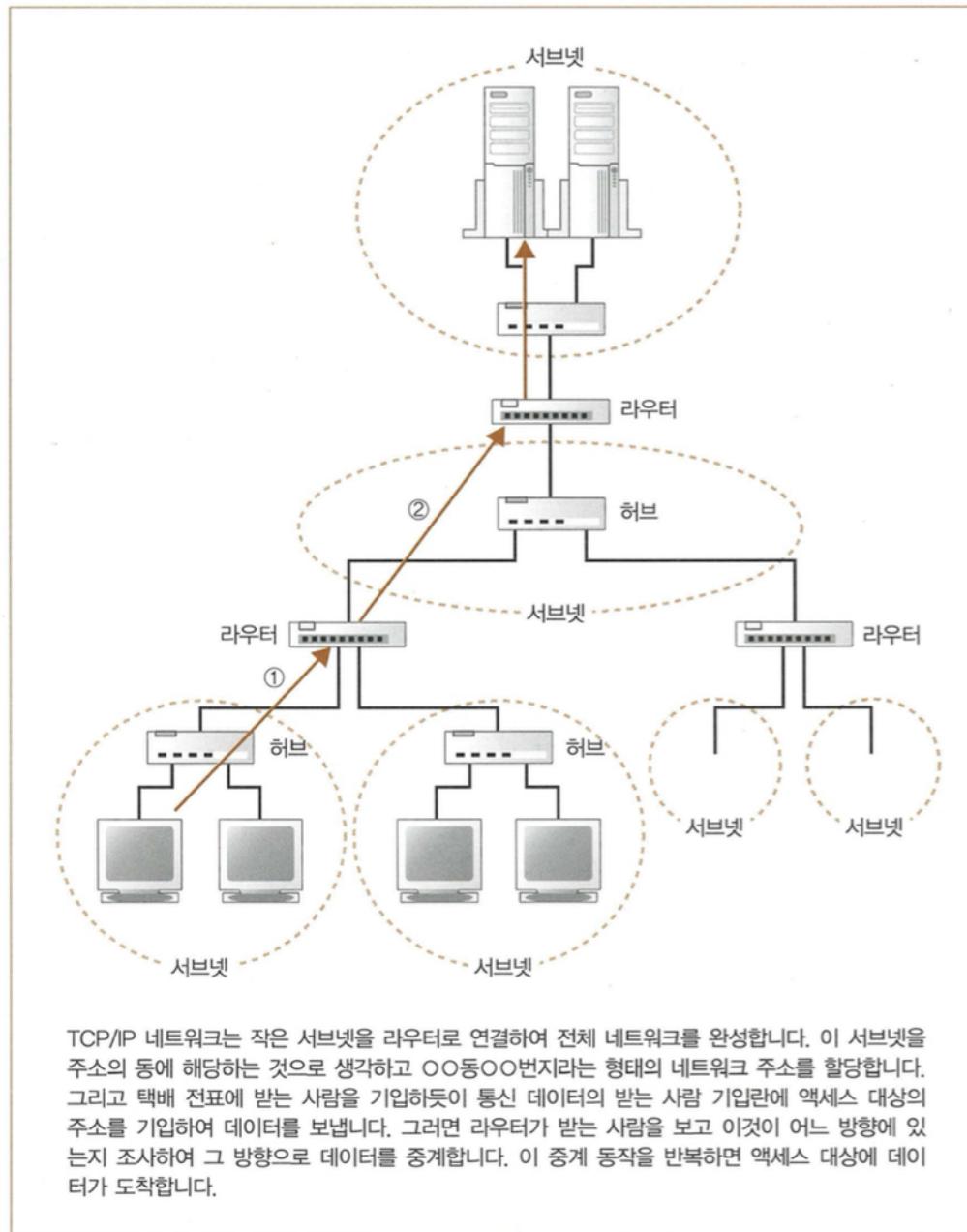
- 응답의 경우 정상 종료, 오류 발생등의 실행 결과를 나타내는 스테이터스 코드와 응답 문구를 첫 번째 행에 써야 함.
- 스테이터스 코드는 숫자로 쓴 것, 실행 결과를 알려주는 것이 목적.
- 응답 메세지가 오면, 데이터를 추출한 후 화면에 표시하여 웹 페이지를 눈으로 볼 수 있다.
- 영상들을 포함하는 경우 문장 안에 영상 파일을 나타내는 태그(제어 정보)가 포함되어 있으므로 브라우저는 화면에 문장을 표시할 때 태그를 탐색.
- 영상을 포함하는 의미의 태그를 만나면 공백을 비워두고 문장을 표시. 이후 다시 웹 서버에 액세스하여 태그에 쓰여있는 영상 파일을 웹 서버에서 읽어와서 공백에 표시한다.
- 한 문장에 3개의 영상이 포함되어 있다면 문장을 읽는 리퀘스트, 영상 파일을 읽는 리퀘스트 (3), 총 4회의 리퀘스트 메세지를 웹 서버에 보낸다.

▼ Story.02 웹 서버의 IP 주소를 DNS 서버에 조회한다.

- 메세지를 만들면 OS에 의뢰해서 웹 서버에 송신.
- 이때 메세지를 넘기는 상대의 IP 주소를 OS에 통지해야 한다.
- 여기에서 브라우저는 웹 서버의 IP 주소를 조사.
- URL 안에는 웹 서버의 도메인명이 쓰여 있고, 그 이름을 DNS 서버에 조회하여 IP 주소를 조사함.

▼ 1. IP 주소의 기본

- HTTP 메세지를 만들면 OS를 통해 액세스 대상의 웹 서버에 송신.
- 브라우저는 URL 해독, HTTP 메세지 만들기만 가능 네트워크 송출은 OS가 함.



- TCP/IP는 서브넷이라는 작은 네트워크를 라우터로 접속하여 전체 네트워크로 만들어 진다고 생각하면 됨. (서브넷 : 허브에 몇 대의 PC가 접속된 것이라고 생각해도 됨)
- IP 주소 : 네트워크 번호 + 호스트 번호
- 액세스 대상의 서버까지 메세지를 운반할 때는 이 IP 주소를 따라 판단하고 운송.
- 송신측이 메세지를 보내면 서브넷 안에 있는 허브가 운반, 송신측에서 가장 가까운 라우터까지 도착.

- 라우터가 메세지를 보낸 상대를 확인하여 다음 라우터를 판단하고, 거기에 보내도록 지시하여 송신 동작을 실행한 후 다시 서브넷의 허브가 라우터까지 메세지를 보냄.
- 라우터를 반복적으로 방문하고 최종적으로 상대의 데이터가 도착한다는 원리.

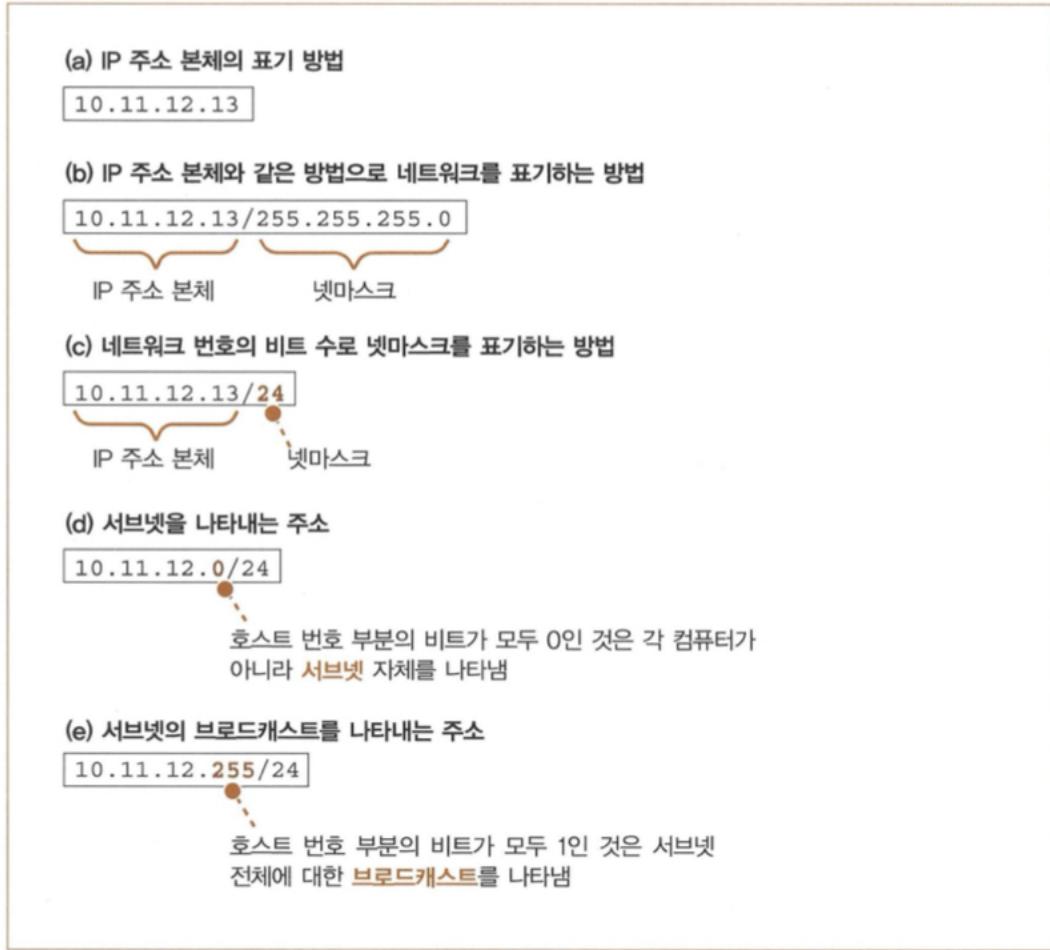


그림 1-9 IP 주소의 표기 방법

- 실제 IP 주소는 32비트의 디지털 데이터. 8비트(1바이트)씩 점으로 구분하여 10진수 표기
- but 이것만으로는 어느 부분이 네트워크 번호, 호스트 번호인지 알 수 없다.
- IP 주소의 규칙에서는 네트워크 번호, 호스트 번호 2가지를 합쳐서 32비트로 한다는 것이 결정 내역은 결정되어 있지 않다.
- 이 내역은 직접 결정할 수 있고, 나타내는 정보를 IP 주소에 덧붙이는데, ‘넷마스크’라 함.

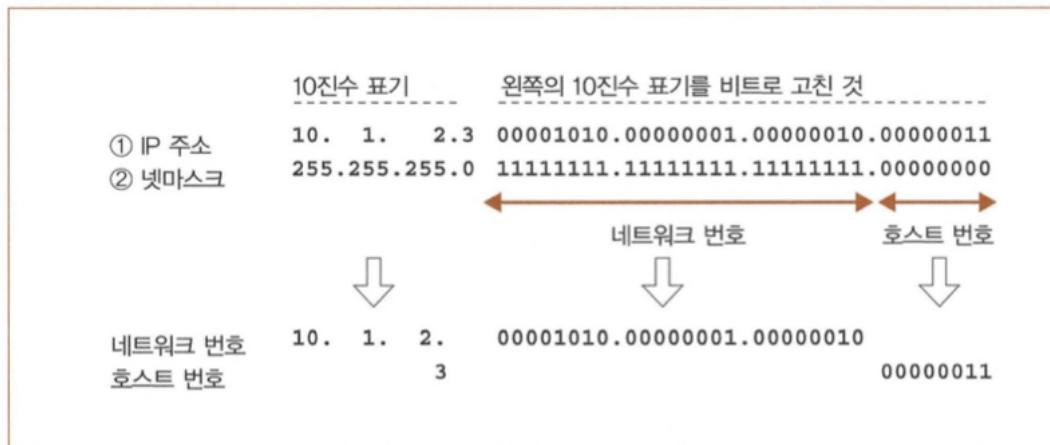


그림 1-10 IP 주소의 구조

넷마스크로 네트워크 번호와 호스트 번호의 경계를 나타내는데, 이 예는 경계가 바이트의 경계와 일치합니다. 즉 마침표 위치와 일치하고 바이트의 도중에 경계를 두어도 상관없습니다.

- 넷마스크는 IP주소에서 32비트 부분의 디지털 데이터. 왼쪽에 1이 나열되고 오른쪽에 0이 나열된 값이 됨.
- 넷마스크가 1인 부분은 네트워크 번호, 0인 부분은 호스트 번호를 나타냄.
- 모두 0 → 서브넷 자체를 나타냄.
- 모두 1 → 서브넷에 있는 기기 전체에 보냄 (브로드캐스트)

▼ 2. 도메인명과 IP 주소를 구분하여 사용하는 이유

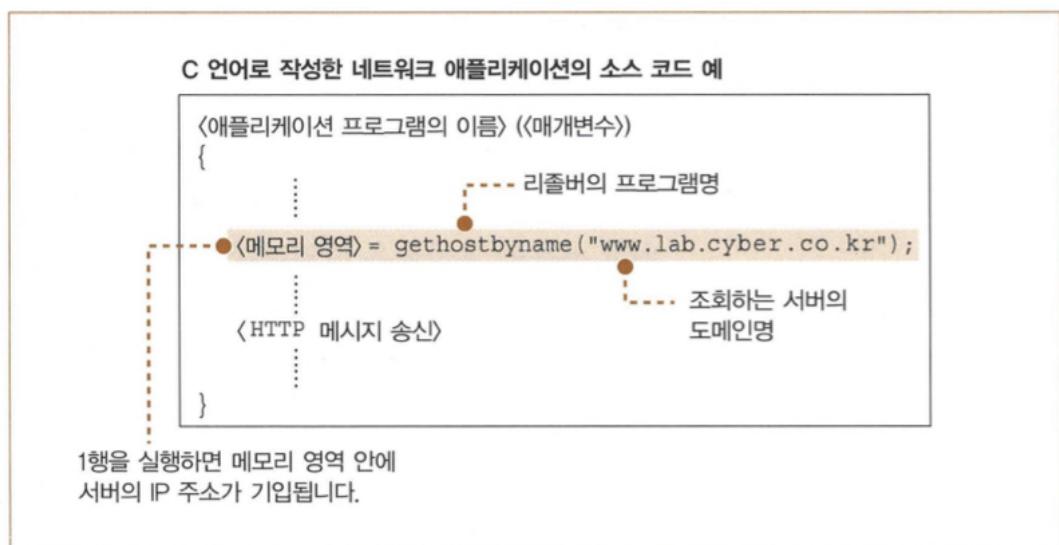
- TCP/IP 네트워크는 IP 주소를 사용해 메세지 전달.
- URL 주소는 IP주소 보다 기억하기 쉬워서 사용.
- but IP주소는 32비트, 4바이트에 해당하는 개수 밖에 없지만, 도메인명은 수십 바이트부터 최대 255바이트나 있다.
- 이런 만큼 라우터가 부하되어 데이터를 운반하는 동작에 더 많은 시간이 걸리면서 네트워크의 속도가 느려짐.
- 도메인 명으로 IP주소를 알아 사용하는 원리가 DNS

▼ 3. Socket 라이브러리가 IP 주소를 찾는 기능을 제공한다.

- IP주소를 조사하는 법. → 가장 가까운 DNS 서버에 도메인 명 (www.lab.cyber.co.kr)이라는 서버의 IP주소를 요청.
- 그러면 DNS 서버는 IP 서버를 알려줌.

- DNS 서버에 조회는 DNS 서버에 조회 메세지를 보내고, 응답 메세지를 받는 것임.
- 이 DNS 클라이언트에 해당하는 것을 DNS 리졸버 (리졸버)라고 부른다.
- DNS의 원리를 사용하여 IP 주소를 조사하는 것은 네임 리졸루션.
- 리졸버는 그냥 Socket 라이브러리에 있는 프로그램.

▼ 4. 리졸버를 이용하여 DNS 서버를 조회한다.



● 그림 1-11 리졸버를 호출하는 방법

애플리케이션 안에 위와 같이 1행을 쓰면 리졸버가 호출되어 DNS 서버에 대한 IP 주소 조회 동작을 실행 합니다.

- 리졸버를 호출하면 리졸버가 DNS 서버에 조회 메세지를 보내고, 응답 메세지가 돌아옴.
- 리졸버는 응답메세지의 IP주소를 추출하여 메모리 영역에 써넣는다.
- DNS를 통한 IP주소 조회가 끝나면 메모리 영역의 IP주소를 추출해 Http 리퀘스트 메세지와 OS 송수신 함.

▼ 5. 리졸버 내부의 작동

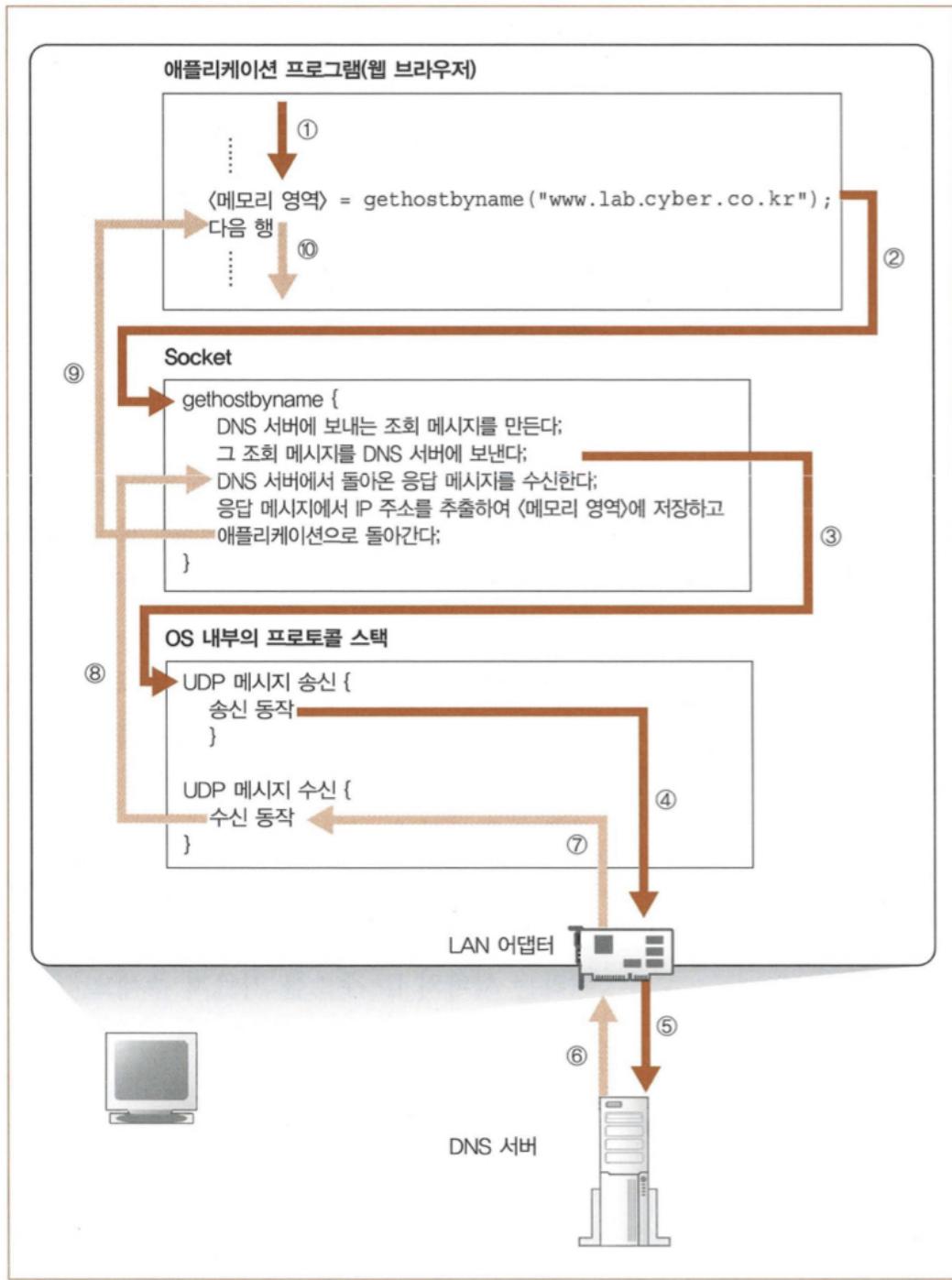


그림 1-12 리졸버를 호출할 때 PC 내부의 움직임
복수의 프로그램들을 차례로 처리해서 실행하여 데이터가 송신됩니다.

- 위 동작이 리졸버를 통한 IP 주소 조회 절차이다.
- DNS 서버에 메세지를 송신할 때도 DNS 서버의 IP주소가 필요.
- 이것들은 TCP/IP 설정 항목으로 컴퓨터에 미리 설정되어 있다.

▼ Story.03 전 세계의 DNS 서버가 연대한다.

- 브라우저에서 IP 주소에 관한 조회를 받은 DNS 서버가 IP주소를 조사하는 장면으로 이어짐.
- DNS 서버는 전 세계에 수만 대가 있고, 그것들이 연대하여 IP주소를 조사함.

▼ 1. DNS 서버의 기본 동작

- DNS 서버의 기본 동작은 클라이언트에서 조회 메세지를 받고 조회의 내용에 응답하는 형태로 정보를 회답.
- 조회 메세지에는 다음의 세 가지 정보가 포함.
 - 이름 :
 - 서버나 메일 배송 목적지(메일 주소에서 @ 뒷부분의 이름)와 같은 이름
 - 클래스 :
 - DNS 구조를 고안했을 때 인터넷 이외에도 네트워크에서의 이용까지 검토하여 식별하기 위해 클래스 정보를 준비.
 - 그러나 지금은 인터넷 이외의 네트워크는 소멸해서 클래스는 항상 인터넷을 나타내는 IN이라는 값이 됨.
 - 타입 :
 - 이름에 어떤 타입의 정보가 지원되는지를 나타냄.
 - ex)
 - 타입이 A이면 이름에 IP주소가 지원되는 것을 나타냄.
 - MX이면 이름에 메일 배송 목적지가 지원되는 것을 나타냄.
 - 타입에 따라 클라이언트에 회답하는 정보의 내용이 달라짐

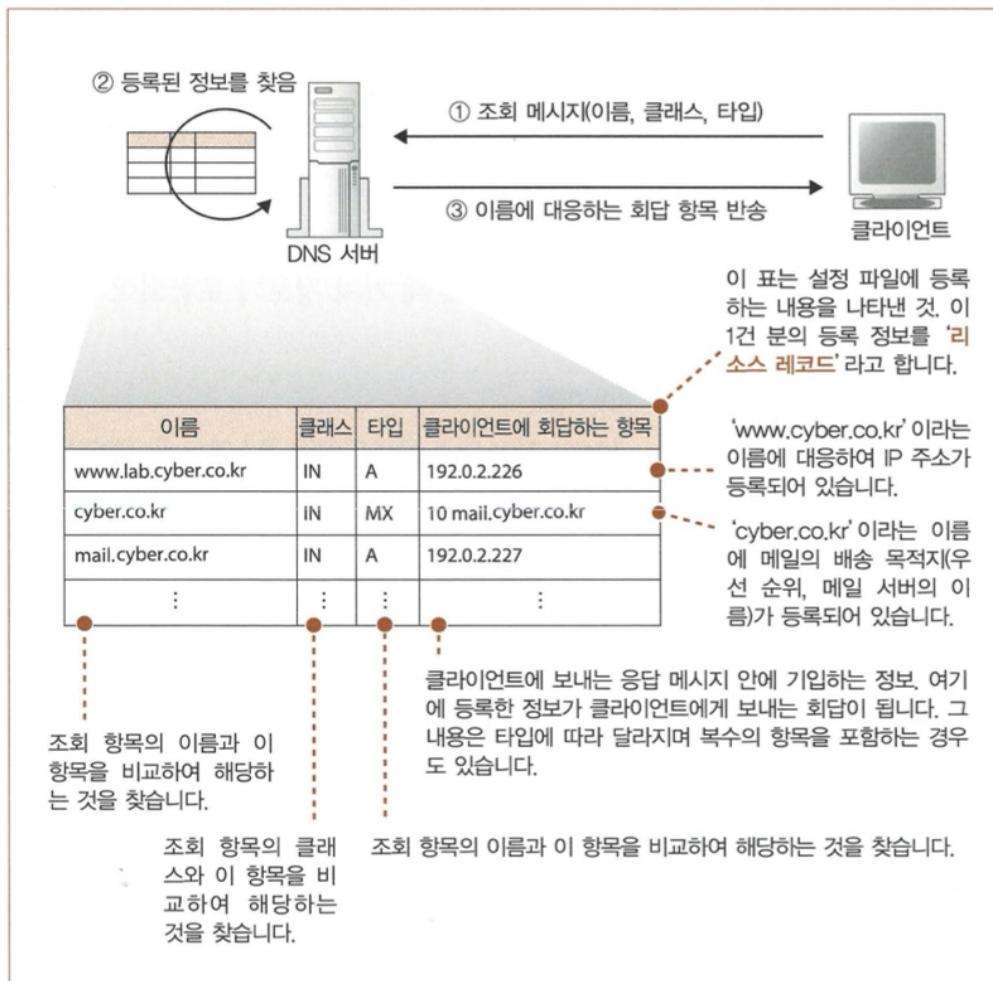


그림 1-14 DNS 서버의 기본 동작

- 이름, 클래스, 타입 (www.lab.cyber.co.kr, IN, A) 정보의 조회 메세지를 DNS 서버에 보내면,
 - 위 이미지에서 192.0.2.226이라는 값을 클라이언트에 회답.
- IP주소를 조회할 때는 A타입, MX는 메일 배송 목적지를 조회.
 - 이름, 클래스, 타입 (cyber.co.kr, IN, MX)
 - tone@cyber.co.kr주소가 있는데 배송 목적지 메일 서버를 조사할 경우. @뒤에 이름이 메일 배송 목적지가 되서 그 이름을 조회.
 - 위 그림에서는 10, mail.cyber.co.kr 두 개의 항목에 회답(우선 순위, 메일 서버의 이름)
 - mail.cyber.co.kr이 등록되 있기 때문에 등록된 192.0.2.227이 회답.
- 이 외에도 IP주소에서 이름을 조사하는 PTR 타입, 이름에 닉네임을 붙이기 위한 CNAME,

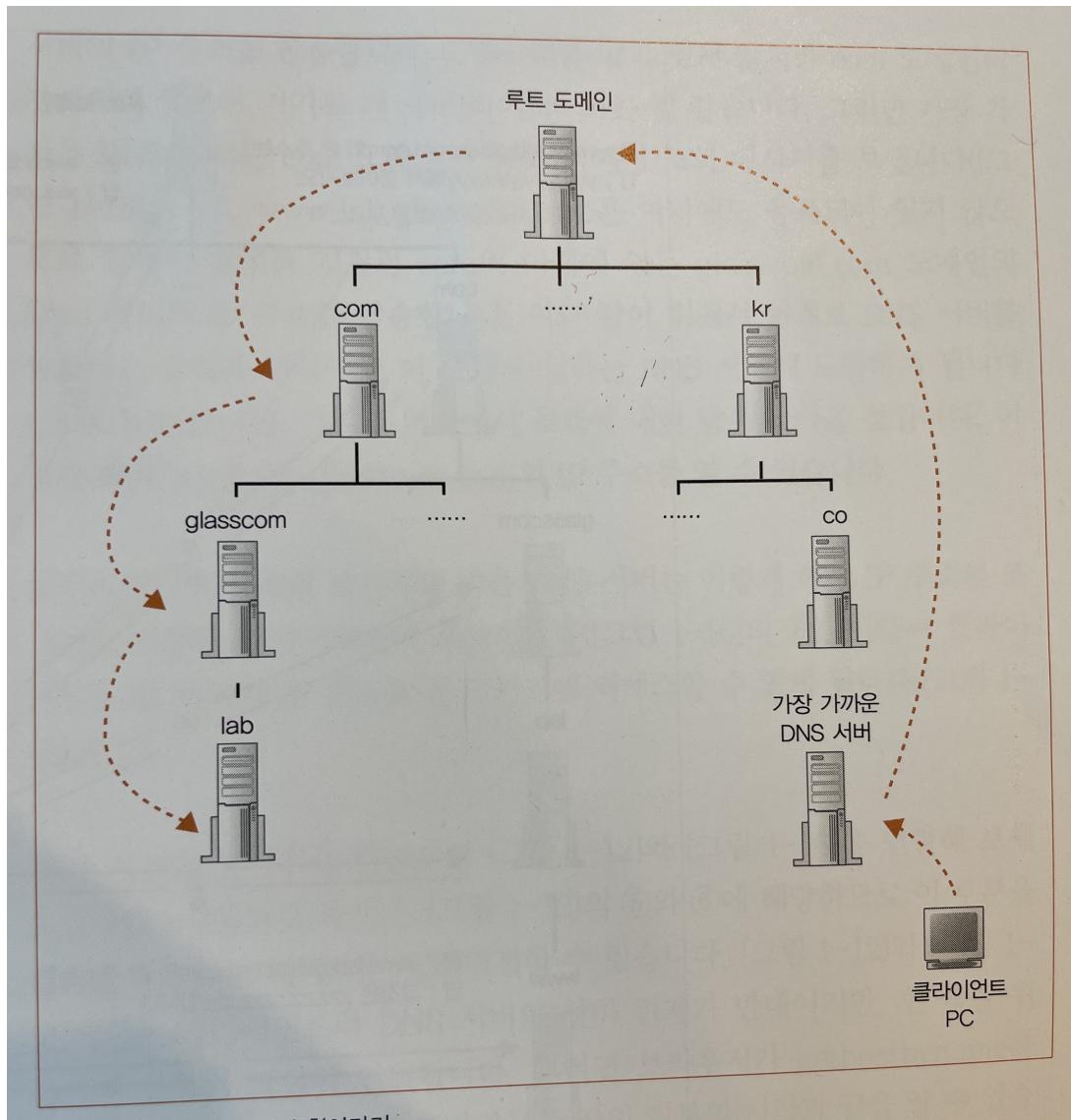
DNS 서버의 IP주소를 등록하는 NS,
도메인 자체의 속성 정보를 등록하는 SOA등이 있다.

▼ 2. 도메인의 계층

- 위 경우는 DNS에 이름과 IP주소가 등록되어 있는 경우임.
 - DNS 서버에 등록되어 있지 않은 경우.
 - 정보를 분산시켜서 다수의 DNS 서버에 등록하고, 다수의 DNS 서버가 연대하여 어디에 정보가 등록되어 있는지 찾아내는 구조.
-
- DNS 취급 이름은 www.lab.cyber.co.kr 처럼 점(.)으로 구분되어 있는데, 이걸로 계층을 구분. → www, lab, cyber, co, kr
 - 한 개의 도메인 정보를 일괄적으로 DNS 서버에 등록하고 도메인 한 대의 정보를 분할하여 복수의 DNS 서버에 등록하는 것은 불간으.
 - 단 DNS 서버와 도메인은 항상 1대1이 아니라 한 대의 DNS 서버에 복수 도메인의 정보를 등록 가능.
 - 즉 한 대의 DNS서버에 도메인 한 대를 등록한다고 생각.

▼ 3. 담당 DNS 서버를 찾아 IP 주소를 가져온다.

- DNS 서버에 등록한 정보를 찾아내는 방법.
- 중요한 것은 액세스 대상의 웹 서버가 어느 DNS서버에 등록되어 있는지 찾아내는 방법.
- 수 만대의 DNS 서버에서 찾지 않고, 먼저 하위의 도메인을 담당하는 DNS 서버의IP주소를 상위의 DNS 서버에 등록, 그리고 상위의 DNS 서버를 또 그 상위의DNS 서버에 등록하는 방식.
 - lab.glasscom.com → glasscom.com → com 등록
- com, kr (최상위 도메인) DNS 서버에 하위 DNS 서버를 등록한 곳에서 끝나지 않고 루트 도메인이 존재.
- 이 루트 도메인의 DNS 서버를 인터넷에 존재하는 DNS 서버에 전부 등록.
- 루트 도메인 부터 계층적으로 액세스하게 된다.



- 클라이언트는 가장 가까이에 있는 DNS 서버에 웹서버 정보 조회
(ex: www.lab.glasscom.com)
- 가장 가까운 DNS 서버는 루트 도메인 DNS 서버가 등록되어 있어 클라이언트로부터 받은 조회 메세지를 전송

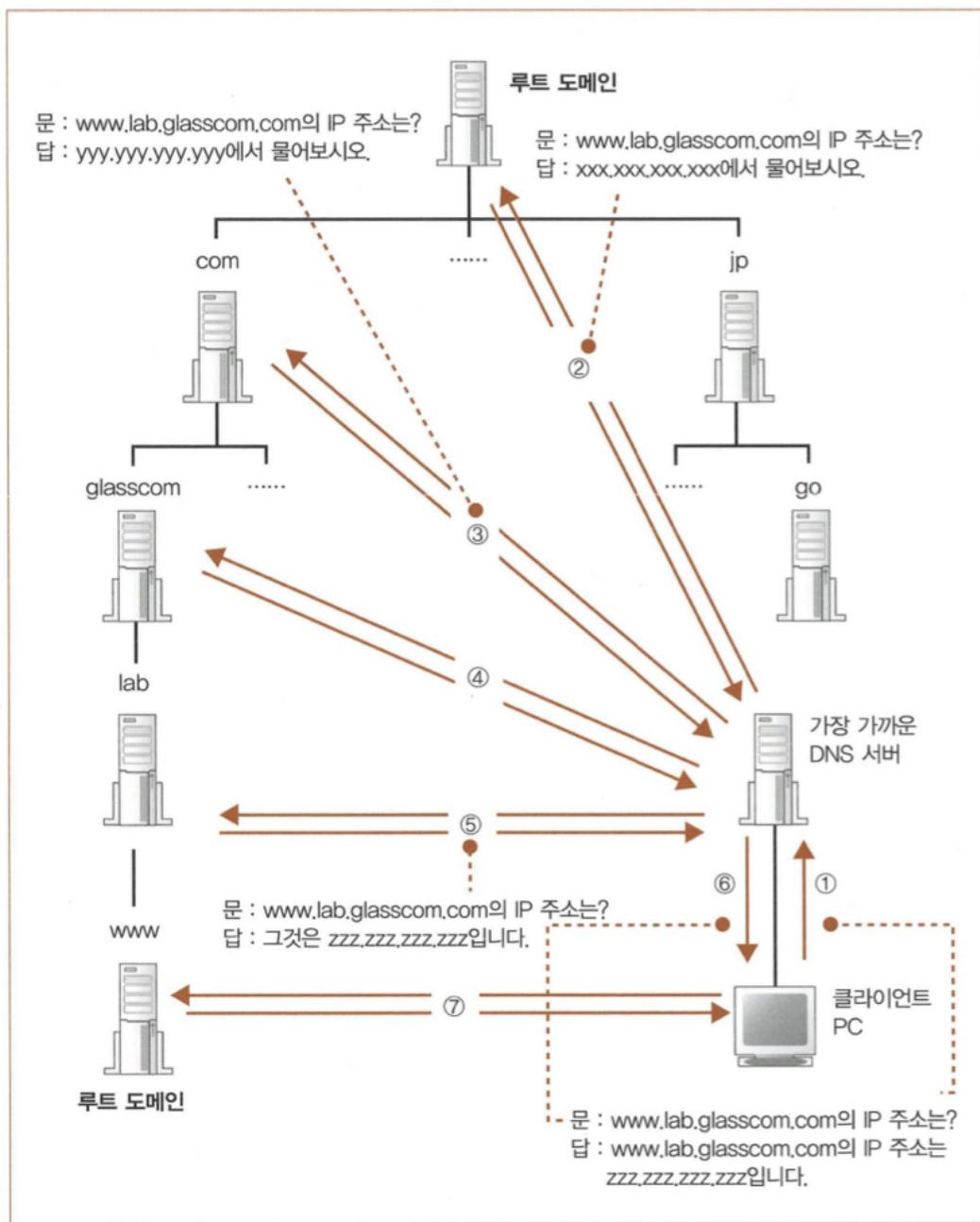


그림 1-16 DNS 서버들의 조회 동작

- com → glasscom.com → lab.glasscom.com → www.lab.glasscom.com 정보 얻음.

▼ 4. DNS 서버는 캐시 기능으로 빠르게 회답할 수 있다.

- 현실의 인터넷에서는 한 대의 DNS 서버에 복수 도메인의 정보를 등록할 수 있어 각 도메인에 한 대씩 DNS 서버가 존재한다고 단정할 수 없다.

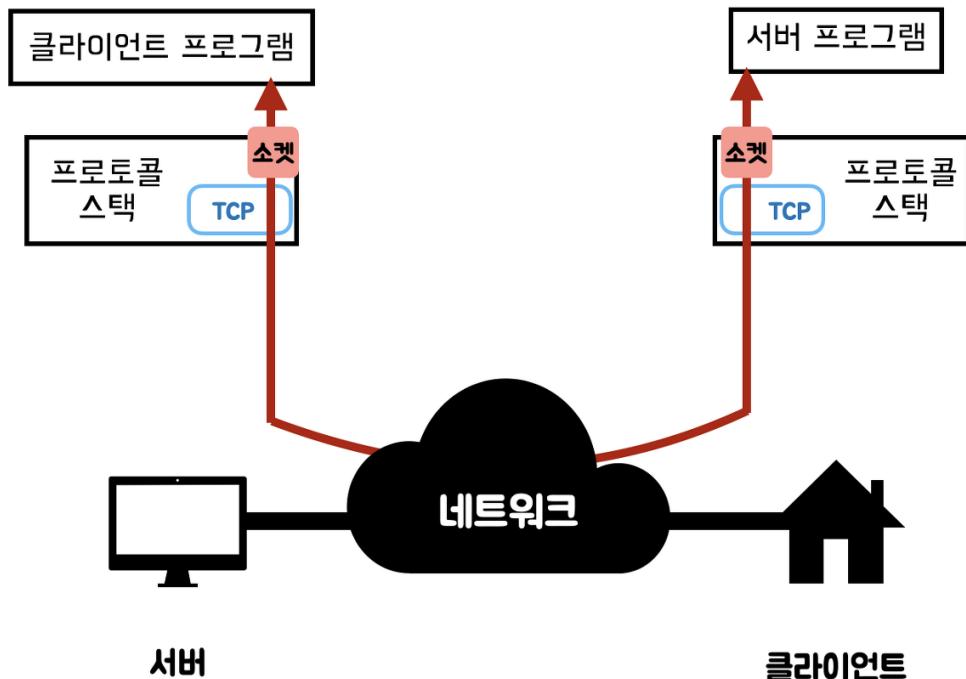
- 위 예제에서는 도메인마다 따로 DNS서버를 셋지만 현실에서는 상위, 하위의 도메인을 같은 DNS 서버에 등록한느 경우도 있다.
- 위 예시 처럼 계층으로 차례대로 움직이지 않고 DNS 서버에 한 번 조회 했나 이름을 캐시에 기록할 수 있다.
- 캐시에 저장한 정보가 등록 정보가 변경되는 경우도 있어 항상 올바르다고 단언할 수 없다.
- 유효 기간을 설정하고, 캐시에서 삭제함.
- 또한 조회에 회답의 정보가 캐시에 저장된 것인지, 등록처 DNS서버에서 회답한 것인지 알려줌.

▼ Story.04 프로토콜 스택에 메세지 송신을 의뢰한다.

- IP 주소를 조사하면 메세지를 웹 서버에 송신하도록 OS에 의뢰.
- OS에 의뢰할 때 규칙이 있으나 정통할 필요는 없다.
- 규칙의 분위기만 알고 있으면 큰 도움이 된다. OS에 의뢰할 때 규칙을 알면 어떤 방식으로 의뢰하면 무엇을 해 주는지를 알게 되기 때문.

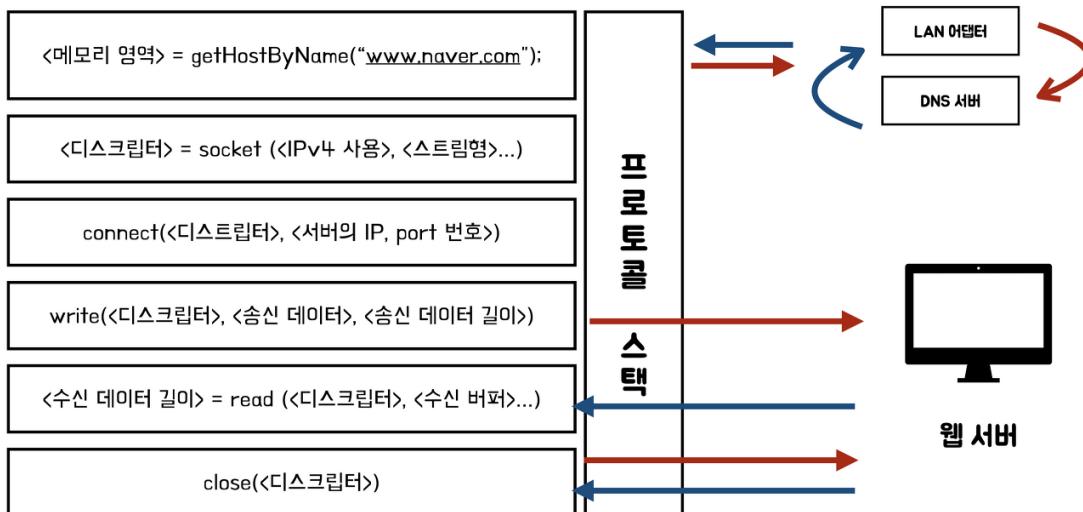
▼ 1. 데이터 송.수신 동작의 개요

- IP 주소를 알면 OS 내부에 이쓴ㄴ 프로토콜 스택에 메세지를 송신하도록 의뢰.



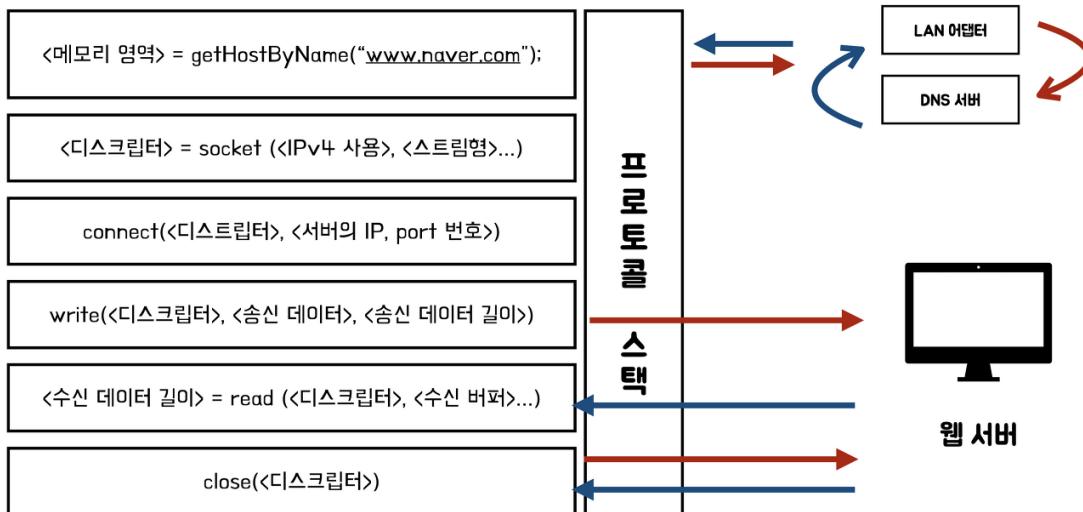
- 데이터를 송,수신하는 컴퓨터 사이에 데이터의 통로 같은 것이 있고, 이것을 통해 데이터가 흐르면서 상태측에 도착, 통로는 파이프와 같은 것으로 생각.
- 이 파이프의 출입구를 소켓이라고 함.
- 데이터 송,수신 동작은 몇 단계로 나누어져 있다. 네 단계로 요약.
 - 소켓을 만듦 (소켓 작성 단계)
 - 서버측의 소켓에 파이프를 연결 (접속 단계)
 - 데이터를 송,수신 (송,수신 단계)
 - 파이프를 분리하고 소켓을 말소 (연결 끊기 단계)
- 위 네 가지 동작을 실행하는 것은 OS 내부의 프로토콜 스택.
- 브라우저 등의 어플리케이션은 자체에서 파이프 연결, 데이터 송, 수신을 하지 않고, 프로토콜 스택에 의뢰해서 파이프를 연결하거나 데이터를 송, 수신.
- 중개역을 역할을 하는 Socket 라이브러리를 통하여 동작한다.

▼ 2. 소켓의 작성 단계



- DNS 서버에 조회를 보낼 때 Socket 라이브러리 사용과 같이 사용.
- 소켓을 생성. 소켓이 생기면 디스크립터라는 것이 돌아옴. 애플리케이션은 이 것을 받아 메모리에 기록한다.
- 디스크립터는 소켓을 식별하기 위해 사용. (다수의 소켓에서 식별하기 위해 사용)

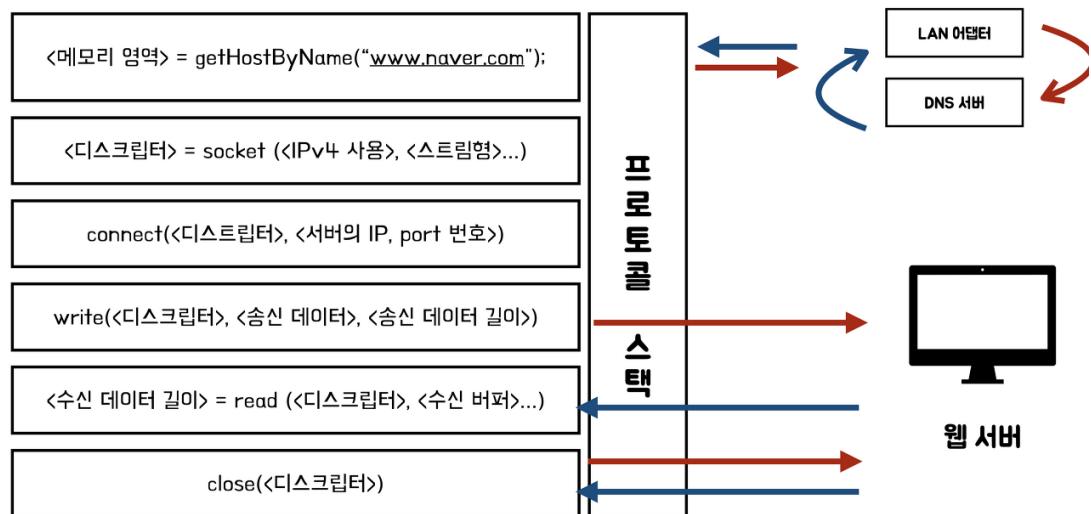
▼ 3. 파일을 연결하는 접속 단계



- connect에서 지정 디스크립터, 서버의 IP주소, 포트 번호가 필요함.
- 최초의 디스크립터는 소켓을 만들 때 돌아온 디스크립터.

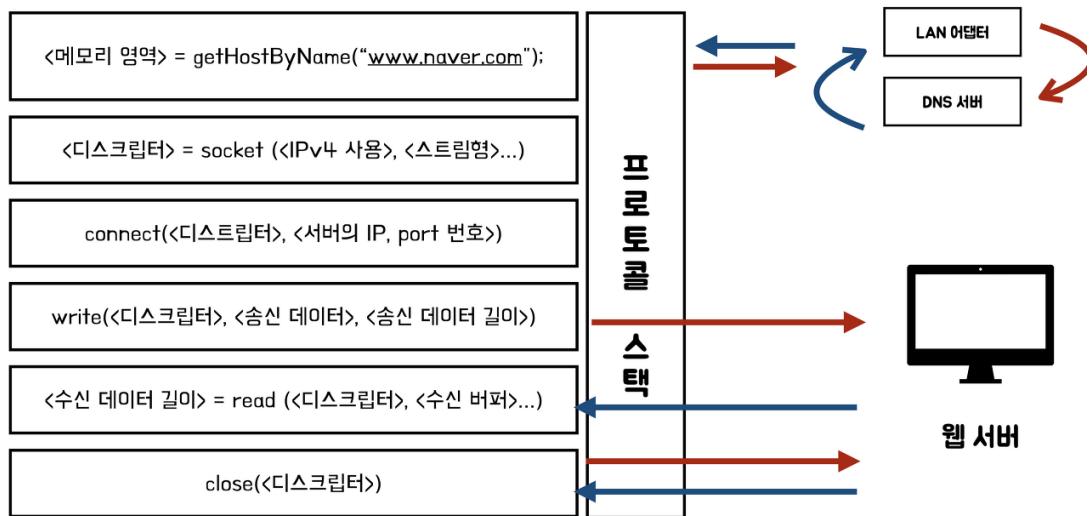
- DNS 서버로 조회한 IP 주소.
- 서버의 어느 소켓과 접속할지 지정은 소켓 번호.
- 서버는 미리 지정되었는 소켓을 사용할 수 있다 (80, 443, ..)
- 서버측에서 클라이언트측의 소켓 번호가 필요한데, 클라이언트 측의 소켓의 포트 번호는 소켓 생성시 프로토콜 스택이 적당히 골라 할당.
- 그 값을 접속 동작을 실행할 때 서버측에 통지 한다.

▼ 4. 메세지를 주고받는 송.수신 단계



- 애플리케이션은 송신 데이터를 메모리에 준비.
(사용자가 입력한 URL을 바탕으로 만든 HTTP의 리퀘스트 메세지)
- write 호출시 디스크립터와 송신 데이터를 지정. 프로토콜 스택이 송신 데이터를 서버에게 송신 한다.
- 보낸 메세지가 돌아오면 수신하는 동작이다.
- 수신할 때는 Socket 라이브러리의 read를 통해 프로토콜 스택에 수신 동작을 의뢰.
- 이때 수신한 응답 메세지를 저장하기 위한 메모리 영역을 지정, 이 메모리 영역을 수신 버퍼라 부른다.
- 수신 버퍼에 메세지를 저장한 시점에서 메세지를 애플리케이션에 건네 준다.

▼ 5. 연결 끊기 단계에서 송.수신이 종료된다.



- 브라우저가 데이터 수신을 완료하면 송,수신 동작은 끝남.
- close는 연결 끊기 단계로 들어가 파이프와 같은 것이 분리되고 소켓 말소.
- HTTP 프로토콜에서는 응답 메세지의 송신을 완료했을 때 웹 서버측에서 연결 끊기 동작을 실행하므로 먼저 웹 서버측에서 close를 호출하여 연결을 끊음.
- 그러면 클라이언트측에 전달되어 클라이언트의 소켓은 연결 끊기로 들어감. 브라우저에 통지.

- HTTP 프로토콜은 HTML 문서나 영상 데이터를 하나하나 별도의 것으로 취급 해 1개의 데이터를 읽을 때마다 접속, 송수신, 끊기 동작을 반복.
- 따라서 하나의 웹 페이지에 영상이 많이 포함되어 있으면 위 동작을 여러번 반복.
- HTTP 1.1 버전부터는 리퀘스트해야할 데이터가 없어진 상태에서 연결 끊기 동작이 들어감.