**전체 과정**

* https://naver.com 라는 URL을 분류해보면

**1.1 통신 규약 (Protocol)**

https:// 는 통신 프로토콜 이다. 이 스키마는 브라우저에 전송 계층 보안 (TLS=Transfer Layer Secure)을 사용하여 서버에 연결하도록 지시한다. HTTPS를 사용하면 암호와 같이 위험한 정보들이 브라우저와 서버가 데이터를 교환할 때 암호화 된다.

ftp:// 나 file:// 은 브라우저가 다른 방식으로 연결하도록 하는 프로토콜들이다.

**1.2 도메인 (Domain)**

naver.com 은 웹 사이트의 도메인 이름이다. 기억하기 쉬운 주소로 특정 서버의 IP 주소를 가리킨다.

**2. 웹 브라우저가 도메인명으로 IP 주소 조회**

브라우저는 URL을 가지고 인터넷에서 연결할 서버를 파악하기 위해 입력한 도메인을 사용하여 웹 사이트를 호스팅하는 서버의 IP주소를 알아낸다. DNS를 사용해서 이 작업을 수행한다.

DNS는 복잡하고 매우 빨라야 하기 때문에 DNS 데이터는 웹 브라우저 사이의 서로 다른 계층과 인터넷의 다양한 위치에 임시로 저장된다.

이를 캐시(Cache)라고 부르며 웹 브라우저는 고유한 캐시 → 운영 체제 캐시 → 라우터의 로컬 네트워크 캐시 → 회사 네트워크 또는 인터넷 서비스 제공업체(ISP)의 DNS 캐시를 확인한다.

만약, 웹 브라우저가 캐시 계층에서 IP주소를 찾을 수 없는 경우 회사 네트워크 또는 ISP의 DNS 서버가 재귀적 DNS 조회를 수행한다.

재귀적으로 인터넷에 있는 여러 DNS 서버를 요청하며, 검색될 때까지 많은 DNS 서버에 요청한다.

웹 브라우저가 IP 주소로 DNS 레코드를 가져오면 인터넷에서 서버를 찾아 연결을 설정한다.

특정 웹 브라우저는 사용자가 링크를 따라가기 전에 미리 도메인 네임을 확인하는 DNS Prefetch 기능을 가지고 있다.

웹 페이지 내에 도메인명이 미리 확인되면 DNS 확인 시간으로 인한 지연이 발생하지 않는다.

\* DNS Prefetch가 도움이 될 수 있는 경우는 검색 결과 페이지와 같이 다양한 도메인명의 링크가 있는 페이지를 보고 있을 때이다.

**3. 웹 브라우저가 서버와의 TCP 연결 시작**

인터넷에 연결된 웹 브라우저 요청 패킷은 일반적으로 TCP/IP 전송 제어 프로토콜을 사용하여 라우터 장비, 인터넷 서비스 제공회사 교환기를 통해 이동되어 통신 회사간 경로인 라우팅 테이블을 따라서 연결할 IP 주소가 있는 웹 서버를 찾는다.

하지만 웹 서버에 직접 도달하는 방법은 가끔 효율적이지 않을 수 있어서 요즘에는 직접 서버에 연결하지 않고 콘텐츠 전송 네트워크(CDN)을 사용하여 정적 및 동적 컨텐츠를 웹 브라우저에 가까이 위치 시킨다.

웹 브라우저가 인터넷에서 서버를 찾으면 웹 서버와 TCP 연결을 설정하고, HTTP를 통해 통신을 시작한다.

하지만 요새는 대부분이 HTTPS를 사용하고, HTTPS는 주고 받는 데이터의 암호화를 위한 TLS 핸드셰이크 과정을 수행한다.

이는 암호화를 할 상호 대상을 확인하는 것이다.

HTTP 통신이 진행될 때, 웹 브라우저와 웹 서버 간의 통신이 어떻게 일어나는지 알아보기 위해서는 웹 브라우저의 개발자 도구에서 확인이 가능하다.

* init connection 부분을 보면, 특정 리소스를 가져오는데 필요한 DNS를 찾는데 걸린 시간, 연결 시간 등을 확인할 수 있다.
* 웹 브라우저가 서버와의 연결을 설정했으면 다음에는 리소스 또는 페이지를 가져오기 위해 HTTP 요청을 전송한다.

**4. 웹 브라우저가 HTTP 요청을 서버로 전송**

웹 브라우저가 서버에 연결되면(TCP), HTTPS 프로토콜에 대한 통신 규칙을 따른다.

웹 브라우저가 페이지의 콘텐츠를 요청하기 위해 HTTP 요청을 전송하는데, HTTP 요청에는 요청 라인, 헤더(또는 요청에 대한 메타데이터) 및 본문이 포함된다.

요청 라인에는 클라이언트(브라우저)가 수행하려는 작업을 서버가 확인하는 데 사용하는 정보가 포함되어 있다.

|  |
| --- |
| <요청 라인>  GET /tables/user HTTP/1.1  - GET, POST, PUT, DELETE 등 요청 메서드 : GET  - 요청된 리소스를 가리키는 경로 : /tables/user  - 통신할 HTTP 버전 : HTTP/1.1 |

요청 헤더(Request Header) 는 요청을 라우팅하는데 도움이 되는 추가 정보를 전달하고, 어떤 유형의 클라이언트(사용자 에이전트)가 요청을 수행했는지 나타내며, 블루/그린 배포나 카나리 배포를 지원하는데 사용할 수 있다.

헤더는 아래와 같은 키-값 페어이다.

|  |
| --- |
| User-Agent: Mozilla/5.0  Accept: \*/\* |

마지막은 본문이다. 본문은 대부분 GET 요청일 때는 비어있고, POST, PUT 또는 PATCH와 같이 리소스를 조작하는 경우 생성하거나 업데이트할 데이터를 포함한다. 요청 본문의 형태를 서버는 요청 헤더인 Content-Type을 기반으로 이해한다.

Content-Type는 application/json, application/x-www-form-urlencoded, multipart/form-data 등 다양하다.

웹 서버가 클라이언트로부터 요청을 받으면 서버는 요청을 처리하고 응답을 다시 전송한다.

**5. 웹 서버가 요청을 처리하고 응답을 전송**

웹 서버는 요청을 받고 요청 라인, 헤더 및 본문의 정보를 기반으로 요청을 처리한다.

컨텐츠를 가져오고 응답을 생성하여 클라이언트로 다시 전송을 하며 응답에는 다음이 포함된다.

- 클라이언트에게 요청 상태를 알려주는 상태 라인 (200, 404 등)

- 브라우저에 응답 처리 방법을 알려주는 응답 헤더 (text/html, application/json 등)

- 해당 경로에서 요청된 리소스 (이미지 파일과 같은 콘텐츠 또는 데이터, HTML, CSS, JS)

**6. 웹 브라우저가 콘텐츠 렌더링**

웹 브라우저가 서버로부터 응답을 받으면 응답 헤더를 검사하여 렌더링 하는 방법에 대한 정보를 확인한다. (Content-Type)

위 예제에서 Content-Type 헤더는 png 파일을 수신했음을 알려준다.

전체 브라우저를 띄우는 관점에서 보면,

GET 요청은 페이지의 구조인 HTML을 반환한다.

웹 브라우저 개발 도구를 사용하여 페이지의 HTML을 검사하면 다른 Javascript, CSS, 이미지 리소스를 참조하고 웹 페이지를 설계된 대로 렌더링하기 위해 추가 데이터를 요청하는 것을 확인할 수 있다.

브라우저가 HTML을 파싱하고 렌더링 할 때 JS, CSS, 이미지 및 데이터를 가져오라는 추가 요청을 하는데 항상 병렬로 수행할 수 있는 것은 아니다.

HTML은 CSS나 JS 파일 리소스를 참조하는데 웹 브라우저가 이미지 리소스를 요청하면 Content-Type 헤더가 브라우저에 이미지임을 알려주고 그에 따라 랜더링 하도록 지시한다.

**요약**

1. 웹 브라우저에 URL을 입력한다
2. 웹 브라우저가 도메인의 IP주소를 조회한다 (먼저 캐시를 찾고, 그 다음 DNS를 검색한다.)
3. 웹 브라우저가 찾은 IP주소를 기반으로 서버와의 TCP 연결을 시작한다.
4. 웹 브라우저가 HTTP(S) 요청을 서버로 전송한다
5. 웹 서버가 요청을 처리하고 응답을 다시 웹 브라우저로 전송한다.
6. 웹 브라우저가 전송 받은 컨텐츠를 렌더링한다.

출처 : https://aws.amazon.com/ko/blogs/korea/what-happens-when-you-type-a-url-into-your-browser/