인터넷에서 사용되는 여러 기술 : HTTP 2

조인준
KBS 미디어기술연구소 차장

지난 편에서는 HTTP 프로토콜의 전반적인 아웃라인에 대한 설명을 드렸습니다. 이번 편에서는 HTTP Request와 Response 메시지의 구체적 구조를 통해서 실제로 HTTP 프로토콜이 어떻게 동작하는지 살펴보겠습니다.

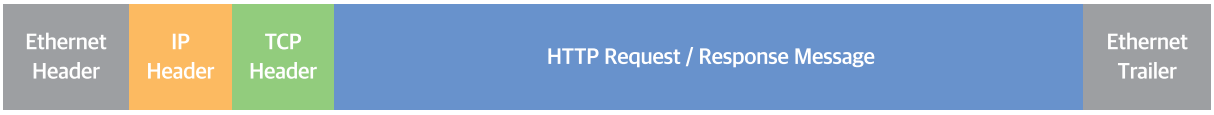

그림 1. HTTP를 사용한 이더넷 프레임 구조

지난 편에서도 말씀드렸듯이 HTTP 프로토콜은 응용계층 프로토콜로써 전송계층 프로토콜인 TCP 프로토콜 위에 올라가며, 이를 네트워크를 통해 전송되는 [그림 1]의 이더넷 프레임 구조로 보면 TCP 헤더 뒤에 HTTP Request 또는 Response 메시지 데이터가 붙는 구조가 됩니다.

[그림 2]는 [그림 1]에서 푸른색으로 표시된 HTTP 프로토콜 블록에 들어가는 Request 메시지와 Response 메시지의 구조를 나타냅니다. Request 메시지가 Request Line으로 시작하고 Response 메시지는 Status Line으로 시작한다는 차이를 빼면 전반적인 구조는 같습니다.

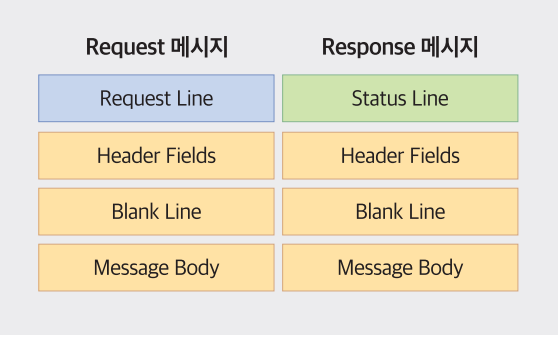

그림 2. HTTP Request 메시지와 Response 메시지 구조

먼저 Request 메시지의 구조가 실제 데이터로는 어떤 모양을 하고 있는지부터 살펴보겠습니다. [표 1]은 브라우저의 주소창에 'tech.kobeta.com'(방송과기술 사이트 주소)을 입력하고 엔터키를 누르면 브라우저가 '방송과기술' 웹서버에 접속하여 보내는 Request 메시지입니다(실제 브라우저에서 보내는 메시지를 설명을 위해 단순화했습니다). HTTP Request 메시지와 Response 메시지는 일반적으로 바이트 단위의 블록으로 정보의 크기와 위치를 고정하여 전달하는 프로토콜들과 달리 [그림 1]의 푸른색 블록에 [표 1]과 같은 텍스트 상태의 정보를 그대로 전달하는 것이 지금까지 다루었던 프로토콜들과의 구별되는 특징입니다.

```
GET / HTTP/1.1
Connection: keep-alive
Host: tech.kobeta.com
Accept: text/html
Accept-Language: ko-KR
User-Agent: Mozilla/5.0 Chrome/115.0.0.0
```

표 1. 방송과기술 시작 웹페이지 요청 예시

[표 2]는 지난 편에서 설명한 HTTP의 개요에 맞춰 [표 1]의 내용을 [그림 2]의 HTTP Request 메시지 구조에 따라 정리한 것입니다. [표 2]를 통해 Request Line은 'GET / HTTP/1.1'이라는 것을 확인하실 수 있습니다. GET은 지난 편에서 말씀드린 요청의 종류를 나타내는 HTTP 메서드 중의 하나로 서버에 데이터를 요청하는 메서드입니다. 예의 GET은 서버로부터 웹페이지의 HTML을 요청하게 됩니다. GET 다음에 공백을 사이에 두고 있는 '/'는 요청 URI(Uniform Resource Identifier)이며 클라이언트가 서버에 어떤 리소스를 요청하는지를 나타냅니다. '/'는 루트 디렉토리를 의미하며, 웹서버에서 루트 디렉토리는 웹사이트에 접속하면 나타나는 초기 웹페이지가 있는 위치와 같습니다. 다른 데이터나 리소스를 요청하려면 원하는 데이터나 리소스의 URI를 '/'의 위치에 적으면 됩니다. '/'와 공백으로 분리되어 마지막으로 나오는 'HTTP/1.1'은 현재 사용하는 HTTP 프로토콜의 버전을 나타냅니다. 'HTTP/1.1'은 HTTP 버전 1.1을 사용하고 있다는 것을 의미하고 이 버전은 현재 가장 널리 사용되는 HTTP 버전입니다. 종합하면 'GET / HTTP/1.1' 요청은 웹서버에 루트 디렉토리의 기본 웹페이지를 'HTTP/1.1'로 보내달라는 요청을 보낸 것입니다.

구분		Request Message
Request Line		GET / HTTP/1.1
Header Fields	General Header	Connection: keep-alive
	Request Header	Host: tech.kobeta.com
		Accept: text/html
		Accept-Language: ko-KR
		User-Agent: Mozilla/5.0 Chrome/115.0.0.0
	Entity Header	-

표 2. 방송과기술 시작 웹페이지 요청 예시의 구조 분류

다음으로 [그림 2]에서 Request 메시지의 Header Fields는 [표 2]와 같이 General Header, Request Header 및 Entity Header로 세분될 수 있습니다. 이 구분에 관해서는 인터넷에 공개된 자료마다 항목에 조금씩 차이가 있고, 어떤 자료는 구분을 안 하는 경우도 있는 것 같습니다. 하지만 Header 구성의 체계적 이해를 위해서 구분에 관해 소개하는 것이 좋을 것 같고, 추후 관련 내용들을 더 알아보고 싶은 독자분들께서도 이 부분에 대해 아시는 것이 도움이 될 것 같아 C군이 찾은 자료들의 교집합에 해당하는 항목 구성으로 Header 구분을 소개해 드립니다.

General Header는 Request 메시지와 Response 메시지에서 공통으로 사용되는 부분입니다. General Header에는 접속 후 현재 연결을 유지할지 또는 종료할지를 나타내는 'Connection' 정보, 메시지를 생성한 날짜와 시간을 알려주는 'Date' 정보 등이 포함될 수 있으며 꼭 필요하지 않은 경우 생략도 가능합니다. [표 2]의 예에서는 'Connection' 정보만을 포함하고 있으며, 이 값이 예와 같이 'keep-alive'이면 이 요청에 응답한 이후에도 연결을 유지하며, 'close'이면 요청에 응답한 이후에 연결을 종료합니다. 'keep-alive'를 사용하는 이유는 HTTP 프로토콜이 주로 사용하는 전송 프로토콜인 TCP의 특성상 빈번한 요청이 있으면 요청마다 연결을 생성하고 끊고를 반복하면 매번 핸드 셰이크를 새로 하는 비효율이 발생하기 때문입니다. 반대로 일회성 연결의 경우 'close'를 사용하여 요청 후에 연결을 끊습니다.

Request Header에는 요청을 받는 서버의 도메인 이름이나 포트 번호를 지정하는 'Host' 정보, 해당 요청이 어떤 클라이언트(브라우저나 애플리케이션)로부터 나온 것인지 식별하기 위한 'User-Agent' 정보, 클라이언트가 처리 가능한 콘텐츠 유형을 알려주는 'Accept' 정보, 클라이언트가 선호하는 언어를 나타내는 'Accept-Language' 정보, 클라이언트가 지원하는 콘텐츠 인코딩 방식을 알려주는 'Accept-Encoding 정보, 클라이언트의 인증 정보 및 서버가 보호된 리소스에 접근할 권한이 있는지 확

인하기 위한 'Authorization' 정보 등이 포함될 수 있습니다. Request Header 또한 때에 따라 필요한 정보만을 포함합니다. [표 2]의 예에서 'Host'는 'tech.kobeta.com'로 되어 있는데 이는 현재 GET 요청이 '방송과기술' 웹서버로 향하고 있음을 나타냅니다. 'Accept'는 'text/html'로 표시되어 있으며, 이는 텍스트로 된 정보인 html을 요청한다는 의미입니다. 'Accept-Language'는 'ko-KR'로 설정되어 있으며 이는 한글로 된 자료를 클라이언트가 선호한다는 의미입니다. 마지막으로 'User-Agent'는 'Mozilla/5.0 Chrome/115.0.0.0'으로 표시되어 있으며 클라이언트 브라우저의 호환성을 나타냅니다.

Entity Header에는 요청이나 응답의 본문에 해당하는 [그림 2]의 Message Body의 미디어 타입을 알려주는 'Content-Type' 정보, 요청 본문의 길이를 나타내는 'Content-Length' 정보 등이 들어갑니다. [표 2]의 예에서는 해당 내용이 없어서 빈칸으로 처리되어 있습니다.

마지막으로 [표 1]의 예에는 [그림 2]와 달리 Blank Line과 Message Body가 없습니다. 이는 GET의 경우 클라이언트가 서버에 제공할 리소스나 데이터가 없기 때문입니다. 서버에 데이터나 리소스 등록을 위한 POST 요청의 경우 서버에 등록할 데이터나 리소스 등에 관한 정보를 담은 Message Body 및 이와 헤더를 구분하기 위한 Blank Line을 포함하여 전송하게 됩니다. 이로써 브라우저 주소창에 'tech.kobeta.com'(방송과기술 사이트 주소)을 입력하면 브라우저가 첫 화면에 보여줄 '방송과기술' 웹페이지를 서버에 요청하는 HTTP Request 메시지가 실제로 어떤 모습을 하고 있는지 알아보았습니다.

다음으로 이 요청을 받은 서버가 응답으로 '방송과기술' 시작 웹페이지의 HTML을 보내주는 Response 메시지가 어떤 모습을 하고 있는지 알아보겠습니다. [표 3]은 [표 1]의 Request 메시지에 대한 '방송과기술' 웹서버의 Response 메시지입니다. 참고로 Blank Line 아래에 <html>..... 등으로 표시한 것은 '방송과기술' 시작 페이지의 HTML을 축약해서 표시한 것입니다. [표 4]는 [표 3]을 [그림 2]의 오른쪽 Response 메시지의 구조로 풀어낸 것입니다.

[표 4]의 맨 위 Status Line인 'HTTP/1.1 200 OK'는 'GET / HTTP/1.1' 요청에 대해 요청이 성공적으로 처리되었다는 것을 알려줍니다. 상태코드인 '200'의 의미는 지난 편에서 설명드렸듯이 성공을 나타냅니다. General Header 부분에는 [표 2]에는 없던 'Date: Mon, 07 Aug 2023 07:11:38 GMT'가 들어있는데 이는 Response 메시지가 생성된 시간을 알려줍니다. 'Connection'은 이전과 동일한 내용이므로 넘어가겠습니다. General Header 다음의 Response Header에도 Request Header의 경우와 같이 여러 종류의 정보가 포함될 수 있고 이 중에 필요한 정보만을 나열할 수 있습니다. 간결한 설명을 위해 [표 4]의 예에 포함된 정보에 대해서만 다루어 보겠습니다. [표 4]의 'Server'는 'Apache'로 표시되어 있고, 이는 웹서버의 종류를 나타냅니다. 'Expires'는 'Thu, 19 Nov 1981 08:52:00 GMT'로 표시되어 있으며 이는 해당 데이터가 유효한 시점이 과거에 이미 만료되었음을 나타냅니다. 그런데 이 부분이 뭔가 좀 이상한 것 같지 않나요? 왜 40여 년 전에 만료되었다고 표시해서 응답을 할까요? 이는 클라이언트인 브라우저에 캐시를 하지 말라는 의미로 서버가 과거의 날짜를 보내는 것이며 이 날짜가 Thu, 19 Nov 1981인 이유는 해

```
HTTP/1.1 200 OK
Date: Mon, 07 Aug 2023 07:11:38 GMT
Connection: Keep-Alive
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Keep-Alive: timeout=5, max=100
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
[Blank Line]
[16진수 데이터 바이트 사이즈]
<html>.....
[16진수 데이터 바이트 사이즈]
.....
[16진수 데이터 바이트 사이즈]
.....
[16진수 데이터 바이트 사이즈]
.....</html>
0
```

표 3. 방송과기술 시작 웹페이지 요청에 대한 응답 예시

구분		Request Message
Status Line		HTTP/1.1 200 OK
Header Fields	General Header	Date: Mon, 07 Aug 2023 07:11:38 GMT
		Connection: Keep-Alive
	Response Header	Server: Apache
		Expires: Thu, 19 Nov 1981 08:52:00 GMT
		Keep-Alive: timeout=5, max=100
		Transfer-Encoding: chunked
	Entity Header	Content-Type: text/html; charset=UTF-8
Blank Line		-
Message Body		[16진수 데이터 바이트 사이즈] <html> · · · · · · · · · [16진수 데이터 바이트 사이즈] · · · · · · · · · [16진수 데이터 바이트 사이즈] · · · · · · · · ·</html> 0

표 4. 방송과기술 시작 웹페이지 요청에 대한 응답 예시의 구조 분류

당 코드를 작성한 개발자인 Sascha Schumann이 자신의 생일을 넣은 것이라고 합니다. 개발자가 자신의 개인적인 내용을 사용한 것이 널리 퍼지고 계속 사용되고 있는 것이 재밌는 것 같습니다.

다시 본론으로 돌아와서 'Keep-Alive'는 'timeout=5, max=100'으로 설정되어 있으며 이는 General Header의 'Connection'이 'Keep-Alive'일 경우 5초를 연결의 타임아웃으로 설정하고 타임아웃 전까지 최대 100개의 리퀘스트까지 처리해 줄 수 있다는 의미입니다. 'Transfer-Encoding'은 'chunked'로 표시되어 있으며 이는 'Transfer-Encoding'이 가질 수 있는 chunked, compress, deflate, gzip 등의 모드 중에 chunked 모드로 HTML을 전송하겠다는 의미입니다.

Entity Header의 'Content-Type'은 'text/html; charset=UTF-8'으로 표시되어 있으며 텍스트 정보인 HTML을 보낸다는 것과 보내는 HTML이 UTF-8 방식으로 인코딩되어 있음을 나타냅니다. 헤더가 끝나면 Blank Line을 사이에 두고 Message Body에 요청한 데이터가 실립니다. 위 예의 경우 'Transfer-Encoding: chunked'로 Message Body를 전송하겠다고 했으므로 [표 4]와 같이 '방송과기술'의 시작 페이지 HTML을 청크(chunk)로 분할해서 각 청크의 바이트 수를 먼저 적고 그다음 줄에 해당 바이트의 html 본문 청크를 연이어 적은 다음 가장 마지막에 0과 빈 줄을 넣어서 데이터가 끝났음을 알려줍니다.

지금까지 HTTP Request 메시지와 Response 메시지에 관해 흔히 접하는 개론보다는 구체적으로 알아보았습니다. 보잘 것 없는 내용이지만, 독자 여러분의 HTTP 이해에 조금이라도 효용이 있으면 좋겠습니다. 다음 편에서는 이번 편에서 알게 된 것들과 연결해서 HTTP Rest API에 관해 알아보도록 하겠습니다.

P.S.

C군이 여러분께 전하는 내용 중 전문적 성격이 짙은 것은 엄밀한 언어를 사용하여 설명하기에는 한계가 있습니다.
본 내용은 설명하는 대상에 대한 전체적 맥락의 이해에만 이용하시고, 그 이상은 권위 있는 전문자료를 참고하시기 바랍니다.