#cs #study

목차

1) 전송 계층 - TCP, UDP

포트

- 포트를 통한 프로세스 식별
  - 패킷의 최종 송수신 대상은 호스트가 실행하고 있는 프로세스
  - 프로세스는 포트(port) 번호를 통해 식별할 수 있다
  - 주소 표기는 IP:PORT로 한다. 예시) 192.168.0.10:8000
  - TCP와 UDP 전부 포트를 통해 프로세스를 식별하기에 둘 다 헤더에 송/수신자 포트 번호가 포함되어 있다.
- 포트 번호 부여 방식
  - 포트 번호는 16비트로 할당할 수 있는 포트 번호 개수는 총 2^16 개
  - 잘 알려진 포트(well known port) : 0 ~ 1023번으로 범용적으로 사용하는 프로토콜이나 가장 대중적으로 사용하는 애플리케이션이 해당 포트를 점유한다.
    - 예시 : SSH(22), HTTP(80), HTTPS(443)
  - 등록된 포트(registered port) : 1024 ~ 49151번으로 흔하게 사용하는 애플리케이션 프로토콜이 점유한다.
    - 예시 : OpenVPN(1194), MySQL(3306), MSSQL(1433)
  - 사설 포트(private port) : 49152 ~ 65535번으로 동적 포트라고도 불리며 사전에 포트가 정해지지 않은 클라이언트 프로그램을 사용할 때 점유한다.
- 공공 IP와 사설 IP 변환
  - NAT(Network Address Translation) : 공인 IP주소와 사설 IP주소를 변환한다. 이 때, 별도의 변환 테이블을 보유한다.
  - NAPT(Network Address Port Translation) : 포트 기반 NAT. 변환 테이블에 포트 번호를 추가하여 공인 IP주소가 동일하더라도 포트를 통해 각 사설 IP주소를 구분할 수 있다.

TCP와 UDP 차이

- 신뢰할 수 있는 통신 : 네트워크를 통해 패킷 교환 시, 신뢰성 보장 기능을 보유하면 신뢰할 수 있는 통신이다. 신뢰성은 네트워크 패킷의 상태 관리, 흐름 제어, 오류 제어, 혼잡 제어가 제공되는 지 여부다. TCP는 이를 지원하며 UDP는 지원하지 않는다.
- 연결성 통신 : 연결 수립과 종료 단계를 거치는 지 여부다. 두 단계를 거치는 통신이 TCP 통신이며 UDP는 거치지 않는다.

- UDP 헤더 : 송/수신지 포트 번호, 길이(UDP 패킷 바이트 크기), 체크섬(데이터그램 훼손 여부)만 있다.
- TCP 헤더 : 송/수신지 포트 번호, 순서 번호, 확인 응답 번호, 제어비트, 체크섬, 긴급 포인터 등이 있다. UDP에 비해 헤더의 크기가 큰 것은 신뢰성과 연결성 보장을 위해서다.
    - 순서 번호(sequence number) : TCP 세그먼트의 올바른 송수신 순서를 보장하기 위해 새겨진 번호다. 현재 주고받는 TCP 세그먼트가 송수신하고자 하는 데이터의 몇 번째 바이트에 해당하는 지 알 수 있다.
    - 확인 응답 번호(acknowledgement number) : 상대 호스트가 보낸 세그먼트에 대한 응답. 현재 수신한 순서 번호 + 1 값이 매겨지며 확인 응답 번호가 포함된 걸 알리기 위해 제어 비트의 ACK 플래그를 1로 설정한다.
    - 제어 비트(control beat) : 현재 세그먼트에 대한 부가 정보를 나타낸다. ACK(세그먼트 승인 여부), SYN(연결 수립 여부), FIN(연결 종료 여부) 등이 있다.

TCP 연결부터 종료까지

- TCP 연결 수립 : 세 단계 악수(3 way handshake)
    - 호스트 A가 B에게 정보를 송신한다.
    - 1단계(active open) : 호스트 A가 B에게 SYN 비트를 1로 올리고 호스트 B에게 전송한다. 이때, 순서 번호에는 호스트 A의 순서 번호가 포함된다.
        - 첫 연결을 시작하는 과정이다. 보통 클라이언트가 위와 같은 역할을 수행.
    - 2단계(passive open) : 호스트 B가 A에게 응답을 전송한다. ACK 비트와 SYN 비트를 1로 하며 세그먼트의 순서 번호에는 호스트 B의 순서 번호가 존재하고 1단계에서 보낸 순서 번호를 참고하여 응답 번호를 구성한다.
        - 연결 요청을 수신하고 그에 대한 응답을 보낸다. 보통 서버가 위와 같은 역할을 수행.
    - 3단계 : 호스트 A가 B에게 ACK 비트를 1로 올리고 호스트 B에게 전송한다. 순서 번호에는 호스트 A의 순서 번호가 포함되며 2단계에서 보낸 순서 번호를 참고하여 응답 번호를 구성한다.
- TCP의 오류, 흐름, 혼잡 제어
    - 재전송을 통한 오류 제어
        - 송수신 과정에 잘못 전송된 세그먼트가 있는 경우 재전송을 실시하며 중복된 ACK 세그먼트 도착 여부 또는 타임아웃이 발생했을 때 오류를 확인한다.
        - 중복된 세그먼트 : 중간에 보낸 요청 또는 응답 데이터가 손실 되어서 받지 못한 호스트 측에서 다시 세그먼트를 보내는 경우이다.
        - 타임아웃 : 세그먼트 전송 시 전송한 호스트 측에서 타이머를 사용하며 시간 내로 ACK 세그먼트를 못 받은 경우, 재전송을 실시한다.
        - 최근에는 파이프라이닝(pipelining) 전송이라고 해서 확인 응답을 받기 전에 미리 여러 세그먼트를 전송하고 나중에 응답을 받는 방식을 사용한다.
    - 흐름 제어(flow control)
        - 수신 호스트가 한 번에 받아 처리할 수 있는 만큼만 전송한다.
        - 처리 가능한 최대 전송량을 위해 TCP 헤더의 윈도우 필드에 있는 수신 윈도우(receiver window)를 확인한다.
    - 혼잡 제어(congestion control)

        - 대량의 트래픽이 발생하여 패킷 처리 속도가 느려지거나 유실되는 상황을 제어하기 위한 기능이다.
        - 혼잡 정도 판단 기준 : 전송 오류가 발생하면 네트워크가 혼잡한 것으로 간주한다.
        - 혼잡 윈도우(congestion window) : 혼잡 상황 없이 전송 가능한 정보량. 혼잡 윈도우와 수신 윈도우 둘 다 커널 내에서 정의된 값이다.
    - 혼잡 제어 알고리즘
        - 혼잡 윈도우 값이 커널에 정의되어 있지만, 상황에 따라 당연하게도 혼잡 정도가 다르고 정도에 따라 처리할 수 있는 정보량 또한 달라진다. 그렇기에 상황에 따라 혼잡 윈도우 값을 정의하는 계산 방식이 필요하다.
        - AIMD(Additive Increase/Multiplicative Decrease) : 가장 기본적인 혼잡 제어 알고리즘. AIMD 세그먼트를 보내서 RTT(Round Trip Time, 패킷 송신에서 수신까지 걸리는 시간)마다 혼잡 상황을 검사한다. AIMD 응답이 오기 전까지 혼잡 상황이 발생하지 않으면 혼잡 윈도우 값을 1씩 더하고 혼잡 상황이 발생하면 혼잡 윈도우 값을 절반으로 떨어트린다.
- TCP 종료
    - 1단계 : 호스트 A가 FIN 비트가 1로 설정된 FIN 세그먼트를 호스트 B에게 전송
    - 2단계 : 호스트 B가 1단계에 대한 응답으로 ACK 세그먼트를 호스트 A에게 전송
    - 3단계 : 호스트 B가 FIN 세그먼트를 호스트 A에게 전송
    - 4단계 : 호스트 A가 3단계에 대한 응답으로 ACK 세그먼트를 호스트 A에게 전송

TCP의 상태

- 상태(state) : 현재 어떤 통신 과정에 있는 지 나타내는 정보
- TCP는 상태를 관리하고 유지하는 프로토콜로 상태는 크게 1. 연결이 수립되지 않은 상황에서 쓰는 상태, 2. 연결 수립 과정에서 쓰는 상태, 3. 연결 종료 과정에서 쓰는 상태 로 나뉘어진다.
- 연결 수립되지 않은 상황에서 쓰는 상태
    - CLOSED : 연결이 없는 상태
    - LISTEN : 연결 대기 상태(3 way handshake의 1단계 대기)
- 연결 수립 과정에서 쓰는 상태
    - SYN-SENT : 액티브 오픈 호스트가 SYN 세그먼트 전송하고 응답인 SYN + ACK 세그먼트 수신 대기 상태
    - SYN-RECEIVED : 패시브 오픈 호스트가 SYN + ACK 세그먼트를 보낸 뒤, 그에 대한 ACK 세그먼트를 기다리는 상태
    - ESTABLISHED : 3 way handshake가 끝난 뒤에 데이터를 송수신할 수 있는 상태
- 연결 종료 상태에서 쓰는 상태
    - FIN-WAIT-1 : 액티브 클로즈 호스트가 FIN 세그먼트로 연결 종료 요청을 보낸 상태
    - CLOSE-WAIT : FIN 세그먼트를 받은 패시브 클로즈 호스트가 그에 대한 응답으로 ACK 세그먼트를 보낸 후에 대기하는 상태
    - FIN-WAIT-2 : FIN-WAIT-1 상태에서 ACK 세그먼트를 받은 상태
    - LAST-ACK : CLOSE-WAIT 상태에서 FIN 세그먼트를 전송하고 대기하는 상태
    - TIME-WAIT : 액티브 클로즈 호스트가 마지막 ACK 세그먼트를 전송한 뒤 접어드는 상태

Quiz

- 철수는 빠르게 정보를 송신하기 위해 UDP 방식으로 소켓 통신을 실시했다. 근데 철수는 그냥 성능에만 만족해서 프로그램 정보량 감시도 하지 않은 채 허송세월 하다가 나중에 이용자가 너무 불어난 나머지, 서버 패킷 전송량이 큰 폭으로 상승하여 메시지 전송에 문제가 생겼다. 이 때, TCP 통신이라면 어떤 방식으로 이 문제를 해결할 수 있을까?
- 철수는 다시 한 번 심기일전하여 TCP 통신으로 소켓 통신을 구현했다. 그런데 무슨 문제가 있는 건지 TCP 상태가 SYS-SENT만 뜨고 있다. 이 때, SYS-SENT는 무엇을 의미하며 원래라면 어떤 흐름으로 이어져야 하는 지 서술하시오.

2) 응용 계층 - HTTP 기초

DNS와 URL

- 도메인 이름과 DNS
  - 도메인 이름(domain name) : 호스트 IP 주소와 대응되는 정보로 문자열로 표기한다. www.example.com 같은 이름이 도메인 이름이다.
  - DNS 서버 : 도메인 네임을 관리하는 서버로 전 세계에 여러 군데 존재한다. 서버에 저장한 도메인 네임은 각각 대응하는 IP주소도 같이 기록되어 있기 때문에 도메인 네임을 통해 IP주소 파악할 수 있다.
  - 도메인 이름 구조 : 도메인 이름은 점(.)을 기준으로 계층적 구조를 띄고 있다. 최상단에 루트 도메인이 존재하며 그 뒤에 최상위 도메인이 존재한다. 즉, 도메인 이름의 단계는 이름의 역순으로 따라가면 된다.
    - 예시) www.example.com. -> com 앞의 점이 루트 도메인. 그리고 com이 최상위 도메인이다. 최상위 도메인의 다른 이름이 1단계 도메인이며 그 뒤의 이름은 점을 기준으로 단계가 하나씩 올라간다. 여기서는 www가 3단계 도메인인 셈이다.
    - 각 단계 도메인을 전부 합친 도메인 이름을 "전체 주소 도메인 이름(FQDN)"이라고 한다.
  - 도메인 이름 시스템(DNS, domain name system)
    - 도메인 이름은 계층적이기 때문에 각 도메인 이름의 단계에 따라 이름 서버도 나뉜다. 즉, . 도메인을 기준으로 com 도메인으로 들어가는 서버가 있고 kr로 들어가는 서버가 있는 식이다
    - 지역 이름 서버(local name server) : 클라이언트와 가장 밀접한 도메인 이름 서버로 해당 서버가 FQDN에 대응하는 IP주소를 가지고 있으면 주소를 반환한다. 없으면 루트 이름 서버부터 시작해서 FQDN을 찾아나간다.
  - DNS 캐시 : 위와 같은 탐색 과정이 빈번하게 발생하면 시간이 오래 걸린다. 그렇기에 기존에 응답받은 결과를 미리 저장하고 동일한 질문이 나오면 저장한 내용물을 불러와 반환한다. 이 때 값을 저장하는 것을 DNS 캐시라고 한다.
  - DNS 레코드 : 도메인 이름과 관련된 설정 정보. 이름 서버에 DNS 레코드를 등록해야 도메인 이름과 IP를 대응할 수 있다. 레코드 유형은 다음과 같다
    - A : 특정 호스트 도메인 이름과 ipv4
    - AA : 특정 호스트 도메인 이름과 ipv6
    - CNAME : 호스트 이름에 대한 별칭 지정

        - NS : 특정 호스트의 IP 주소를 찾을 수 있는 이름 서버
        - MX : 해당 도메인과 연동된 메일 서버
- 자원과 URI/URL
    - 네트워크에서 자원이란 네트워크 상의 메시지를 통해 주고 받는 최종 대상이다. HTML일 지 그림일 지 음악 파일일 지는 그 때마다 다르다.
    - URI(Uniform Resource Identifier) : 웹 상에서 자원을 식별하기 위한 정보. 이름이나 위치 기반으로 식별한다.
        - URN(Uniform Resource Name) : 이름 기반 식별 방식.
        - URL(Uniform Resource Locator) : 위치 기반 식별 방식.
    - URL 구조
        - foo://www.example.com:8042/over/there?name=ferret#nose
        - scheme : 위 주소에 foo에 해당한다. 자원에 접근하는 방법을 나타내며 예를 들어 http가 붙으면 http 프로토콜을 준수하여 자원에 접근해야 한다.
        - authority : www.example.com:8042 부분이 해당한다. 호스트 특정이 가능한 IP주소 또는 도메인 이름을 명시하며 추가로 구분자인 쌍점(:)을 기입하면 포트 번호도 추가할 수 있다.
        - path : /over/there이 해당한다. 자원이 위치하고 있는 경로이며 IP주소를 통해 찾아간 기기의 디렉터리나 파일 경로라고 생각해도 된다.
        - query : ?name=ferret부분이 해당한다. ?는 쿼리가 시작하는 걸 알리는 구분자고 각각의 내용물은 매개변수를 뜻한다. name이 변수명이고 ferret이 변수에 들어가는 값이다. 매개변수를 여러개 사용할 때는 각 매개변수마다 구분자로 앰퍼샌드(&)를 기입한다.
        - fragment : #nose 부분이 해당한다. 자원의 일부분을 가리킬 때 사용한다. 가령 http 파일에서 #nose라는 부분으로 이동하라는 것 같이 특정 부분으로 이동할 때 사용한다.

HTTP의 특징과 메시지 구조

- HTTP : 애플리케이션의 다양한 자원을 네트워크를 통해 송수신하기 위해 만든 프로토콜.
- HTTP 특징
    - 요청 응답 기반 프로토콜
        - 요청(request)과 응답(response)을 주고 받지 않으면 동작하지 않는다.
    - 미디어 독립적 프로토콜
        - 자원의 종류(media type)와 상관 없이 자원을 보낼 수 있다. 그렇기에 상대방이 무슨 자원인 지 알 수 있도록 자원 종류를 표기할 필요가 있다.
        - 표시 : type/subtype. 먼저 크게 자원 종류를 분리하고 그 다음에 세부적으로 분리한다.
            - text : text/plain(평문 텍스트), text/html, text/css, text/javascript
            - image : image/png, image/jpeg. image/webp, image/gif
            - video : video/mp4, video/ogg, video/webm
            - audio : audio/mdi, audio/wav
            - application : 바이너리 형식 데이터. application/octet-stream, application/pdf, application/xml, application/json, application/x-www-form-urlencoded(HTML 입력 양식 데이터)

            - multipart : 종류가 다른 여러 자원을 묶은 것. multipart/form-data(HTML 입력 폼 데이터), multipart/encrypted(암호화된 데이터)
    - 스테이트리스 프로토콜
        - HTML은 상태를 기억하지 않는다. 즉, 클라이언트의 상태를 별도로 기억하지 않기에 여러 클라이언트와 교류를 하더라도 빠르게 정보를 송수신할 수 있다.
    - 지속 연결 프로토콜
        - HTTP 1.0 버전은 매 요청마다 연결을 시작하고 응답이 끝나면 연결을 끊는 구조다. HTTP 1.1 버전은 지속 연결 기능을 제공하여 하나의 TCP 통신이 연결된 상태에서 지속적으로 요청-응답이 가능하다.
    - HTTP 2.0 : 기존 HTTP 버전과 다르게 바이너리 기반 데이터 송수신, 헤더 압축, 서버 푸시(미래 가용 자원 사전 전송), HTTP 멀티플렉싱 기법(다수의 독립스트림을 통해 메시지를 병렬적으로 주고 받음)이 추가되었다.
    - HTTP 3.0 : UDP 기반으로 동작한다.
- HTTP 메시지 구조
    - HTTP 메시지는 시작 라인, 필드 라인, 메시지 본문으로 이루어져 있다.
    - 시작 라인 : 요청 메시지면 요청 라인이 들어가고 응답 메시지면 상태 라인이 들어간다
        - 요청 라인 : 메서드(수행할 작업 종류), 요청 대상(요청을 보낼 서버 자원, 보통은 URL path + query), HTTP 버전
        - 상태 라인 : HTTP 버전, 상태 코드(요청에 대한 결과를 3자리 정수로 표기), 이유 구문(상태 코드에 대한 문자열로 된 설명문)
    - 필드 라인 : HTTP 헤더를 명시한다. 헤더에는 메시지 전송과 관련된 제어 및 부가 정보를 기입한다. 헤더는 쌍점을 기준으로 헤더명과 헤더명에 들어가는 값을 표기한다.
        - 예시) Content-Type: text/plain

HTTP 메서드와 상태 코드

- HTTP 메서드
    - GET : 자원 습득 요청. 서버에서 자원을 받아올 때 사용한다.
    - HEAD : GET과 동일하지만 응답으로 헤더만 받는다.
    - POST : 서버에서 특정 작업을 처리하라고 요청. 보통은 클라이언트가 서버에 신규 자원 생성 요청 시 사용한다.
    - PUT : 자원 대체 요청.
    - PATCH : 자원에 대한 부분 수정 요청.
    - DELETE : 자원 삭제 요청.
- HTTP 상태 코드
    - 100번대 : 정보성 상태 코드
    - 200번대 : 성공 상태 코드. 요청이 성공한 걸 알린다.
        - 200 : OK. 요청 성공.
        - 201 : Created. 요청하였고 신규 자원도 생성.
        - 202 : Accepted : 요청은 받았으나 아직 요청 작업을 끝내지 못했음.
        - 204 : No Content : 요청은 받았으나 메시지 본문으로 표시할 데이터 없음.

- 300번대 : 리다이렉션 코드. 클라이언트 요청 자원이 다른 위치에 있을 때, 해당 위치로 이동한다.
  - 영구적 리다이렉션 : 자원이 완전히 새로운 곳으로 이동하여 경로를 영구 재지정한다. 경로가 바뀌었으므로 기존의 URL을 기억할 필요가 없음.
  - 일시적 리다이렉션 : 자원의 위치가 임시로 변경된 경우, 임시 URL을 전송한다. 임시 경로이므로 여전히 기존 경로인 URL을 기억해야 함.
  - 301 : Moved Permanently. 영구적 리다이렉션으로 재요청 시 메서드 변경 가능
  - 308 : Permanent Redirect. 영구적 리다이렉션으로 재요청 시 메서드 변경 안 됨
  - 302 : Found. 일시적 리다이렉션으로 재요청 시 메서드가 Get으로 변경
  - 307 : Temporary Redirect. 일시적 리다이렉션으로 재요청 시 메서드 변경 없음
  - 304 : Not Modified. 캐시 관련 상태 코드로 자원 변경 없음
- 400번대 : 클라이언트 에러 상태 코드
  - 400 : Bad Request. 요청 메시지 내용이나 형식 자체가 문제 있음.
  - 401 : Unauthorized. 요청한 자원에 대한 유효 인증이 없음.
  - 403 : Forbidden. 요청이 서버에 의해 거부 됨. 권한 부족.
  - 404 : Not Found. 요청 받은 자원을 찾을 수 없음.
  - 405 : Method Not Allowed. 요청한 메서드를 지원하지 않음.
- 500번대 : 서버에서 문제가 발생
  - 500 : Internal Server Error. 서버 문제로 요청 처리 불가.
  - 502 : Bad Gateway. 중간 서버 통신 오류.

HTTP 주요 헤더

- 요청 메시지에서 주로 활용하는 헤더
  - Host : 요청을 보낼 호스트를 명시. 도메인 이름이나 IP 주소로 표기하며 추가로 포트도 들어갈 수 있음.
  - User-Agent : HTTP 요청을 시작하는 클라이언트 측의 프로그램 정보.
  - Referer : 클라이언트가 요청을 보낼 때 머무르던 URL 명시.
- 응답 메시지에서 주로 활용하는 헤더
  - Server : 서버 호스트와 관련된 정보 명시.
  - Allow : 처리 가능한 HTTP 헤더 목록 명시. 처리 불가한 HTTP 헤더를 보내면 405 오류가 발생한다.
  - Location : 클라이언트에게 자원의 위치를 알려주기 위해 사용. 신규 자원 또는 리다이렉션 발생 시에 주로 사용한다.
- 요청과 응답 메시지 모두에서 활용하는 HTTP 헤더
  - Date : 메시지가 생성된 날짜와 시각 기록.
  - Content-Length : 메시지 본문 바이트 길이 표기.
  - Content-Type, Content-Language, Content-Encoding : 메시지 본문 표현 방식을 알려준다.
    - Content-Type : 메시지 본문에 들어간 자원의 종류를 나타낸다.

        - Content-Langauge : 메시지 본문에 어떤 자연어를 사용했는 지 나타낸다. 가로선(-)을 기준으로 태그를 구분하며 흔히 국가코드-언어코드 조합으로 이루어진다.
        - Content-Encoding : 메시지 본문을 압축하거나 변환한 방식을 명시한다. 인코딩 방식이 여러 개인 경우에는 반점(,)을 기준으로 사용한 인코딩 방식을 열거한다.
    - Connection : HTTP 메시지를 송신하는 호스트가 어떤 방식으로 연결하기를 원하는 지 명시한다. 지속 연결을 원하면 'keep-alive', 연결 종료를 원하면 'close' 같은 식으로 표기한다.

Quiz

- 웹소켓은 HTTP 몇 버전을 사용하는가?
- 영희는 로그인 요청을 서버로 보냈다. 이 때, 서버에서 401 오류를 줬다. 아니, 백엔드에서 요청한 토큰도 잘 포함해서 줬는데 이게 무슨 날벼락인가? 그래서 다시 한 번 영희는 자기 자신이 보낸 정보를 잘 확인했고 문제점을 발견했다. 이 때, 어떤 문제점이 있을 것이며 해결 방식은 무엇일 지 생각해보시오.

```
영희가 보낸 정보
customerID, customerPassword, tokenValue
위 정보의 변수명은 서버에서 요구한 대로 모두 정확하게 들어갔다고 가정한다.
```

- 영희는 이제 401 오류에서 벗어났다. 그러나 기쁨도 잠시 서버에서 403 오류를 보내줬다. 이제는 401을 피하니까 403이라니 요택 늪지대에서 허우적거리며 인간극장을 찍던 수나라 병사들이 이런 심정이었을까? 영희는 다시 한 번 자신이 전송한 값들을 천천히 살펴보았고 문제를 발견했다. 이 때, 어떤 문제점이 있을 것이며 해결 방식은 무엇일 지 생각해보시오.

3) 응용 계층 - HTTP 응용

쿠키(Cookie)

- 서버에서 생성되어 클라이언트 측에 <키, 값> 형태로 저장하는 데이터
- 상태 값을 저장하지 않는 HTTP 프로토콜을 보완할 수 있다. 왜냐하면 별도로 값을 클라이언트에서 보유하기 때문에 요청-응답과 상관없이 클라이언트가 저장하고 있으니까.
- 응답 메시지
    - HTTP 헤더에 부가 정보 중 하나로 Set-Cookie 속성이 들어간다.
    - 형식) Set-Cookie: 이름=값; 속성1; 속성2; ... 속성N
        - 속성값은 넣을 지 말 지 선택이다.
- 요청 메시지
    - HTTP 헤더에 기입하는 건 동일하다.
    - 형식) Cookie: 이름1=값1; 이름2=값2; ... 이름N=값N;
- 속성
    - 도메인과 경로 : 각각 이름을 domain, path로 설정하여 해당 쿠키를 전송할 도메인과 경로를 제한할 수 있다.

    - 쿠키 수명 : 이름을 Expires로 설정하여 수명(형식: 요일, DD-MM-YY HH:MM:SS GMT)을 지정하면 쿠키가 해당 수명이 되었을 때 자동으로 삭제된다.
    - Secure : 보안 속성으로 별도의 값 없이 이름만 넣어서 설정하는 값이다. 쿠키가 HTTPS 프로토콜을 준수할 때만 열람할 수 있다.
    - HttpOnly : 자바스크립트를 통한 쿠키의 접근을 제한하고, HTTP 송수신을 통해서만 쿠키에 접근할 수 있다
- 참고) 스토리지 : 쿠키 이외에 클라이언트에서 <키: 값> 형태로 저장하는 방식
    - 지역 저장소(local storage) : 영구적으로 정보를 저장할 수 있음
    - 인증 저장소(session storage) : session이 유지되는 동안에만 저장

캐시(Cache)

- 응답을 통해 받은 자원의 사본을 임시 저장하여 대역폭 낭 및 응답 지연 방지.
- 캐시 또한 쿠키처럼 수명이 있다. 그래서 수명이 끝나면 다시 서버에 자원을 요청할 필요가 있다. 사용하는 속성값은 Expires 또는 Cache-Control이다
- 수명 속성
    - Expires : 쿠키에서 사용하는 Expires 속성과 동일하며 HTTP 헤더에 기입한다
    - Cache-Control : 캐시에 대한 다양한 설정을 할 수 있는 속성. max-age=수명값(sec)을 넣어주면 응답 시간 기준으로 캐시 수명을 설정한다.
- 캐시 신선도(cache freshness) : 사본 데이터가 원본 데이터와 유사한 정도. 클라이언트는 신선도 유지를 위해 캐시 수명이 끝났을 때, 서버에 원본 데이터 변경 여부를 확인한다.
    - 자원 변경 여부에 대한 서버의 응답
        - 자원이 변경됨 : 200(OK)
        - 자원이 그대로임 : 304(Not Modified)
        - 자원이 없음 : 404(Not Found)

콘텐츠 협상

- HTTP 메시지를 통해 주고 받는 것은 송수신 가능한 자원의 형태이다. 즉, 자원은 컴퓨터에게 있어 바이너리 데이터이기에 이를 어떤 형태로 규정하느냐에 따라 원본 데이터가 그 형태에 맞게 전송된다는 것이다.
- 콘텐츠 협상이란 이런 자원에 대한 다양한 표현 중에 클라이언트가 선호하는 자원의 표현을 주고 받기 위해 서버와 대화를 나누는 것이다.
- 콘텐츠 협상 헤더
    - Accept : 선호하는 자원의 종류
    - Accept-Language : 선호하는 언어
    - Accept-Encoding : 선호하는 인코딩 방식
- 선호 자원에 대한 우선순위 표기
    - 콘텐츠 협상 헤더는 자원명1, 자원명2, 자원명3 등으로 표기하여 클라이언트가 선호하는 자원들을 복수로 표현할 수 있다.
    - 각각 우선순위를 지정하기 위해서는 자원명1;q=우선하는 정도(1 ~ 0 사이 소수점 표기)를 표기한다. 즉, q값을 넣어서 각 자원 별로 상대적인 순위를 지정하는 것이다.

보안: SSL/TLS와 HTTPS

- HTTPS : HTTP 프로토콜에 SSL 또는 TLS 보안 프로토콜이 추가된 것을 의미한다.
- SSL 프로토콜은 TLS 프로토콜의 구버전이며 현재는 TLS 1.2 ~ 1.3을 자주 사용한다.
- TLS 기반 메시지 전송 방식
  - TCP 3 way handshake를 거치고 난 다음에 TLS handshake를 거친다
  - TLS handshake 방식
    - ClientHello
      - 지원하는 TLS 버전, 사용 가능한 암호화 알고리즘과 해시 함수, 키를 만들기 위하 사용할 클라이언트의 난수 등이 포함.
    - ServerHello
      - 선택된 TLS 버전, 사용한 암호화 알고리즘과 해시 함수 정보, 키를 만들기 위해 사용할 서버 난수 등이 포함. 추가로 서버가 인증기관(CA)에서 검증된 사이트라는 걸 입증하기 위한 인증서도 첨부한다.
    - Finished

Quiz

- 철수는 서버에서 요청하여 받은 내용을 클라이언트(철수의 컴퓨터)에 저장하였다. 그런데 어느 날 보안 감사에서 보안 과장이 철수의 컴퓨터에 중요한 정보가 삭제되지 않고 남아있다고 하였다. 철수가 분명 요청받은 내용 중에서 중요한 값은 별도 처리를 하라는 팀장 님의 지시를 기억하지 못했기 때문이다. 이 때, 철수가 저장한 내용들은 어디에 저장되었으며 원래라면 어디에 저장하는 것이 좋겠는가?