1971226 김인우 네트워크 스터디 week2

2장 29 ~ 49 페이지

애플리케이션 계층의 역할

- 사용자가 직접 사용하면서 체감할 수 있는 서비스를 제공

- 애플리케이션 프로그램 간의 통신을 정의

사용자가 직접 사용하는 프로토콜

- HTTP: 웹 클라이언트와 웹 서버 사이에서 웹 페이지 데이터를 주고받는다.

- POP, SMTP, IMAP: 메일을 송수신, 보관

- SMB, AFP: LAN 안에서 파일 공유

- FTP: 서버를 통해 파일을 주고받음

- Telnet, SSH(Secure SHell): 원격에서 서버를 제어

사용자가 간접적으로 사용하는 프로토콜

- 인터넷이나 LAN의 원활한 사용을 위해 사용자에게 보이지 않는 영역에서 동작

- DNS(Domain Name System): 도메인명과 IP 주소의 정보를 서로 변환할 때 사용

- SSL(Secure Sockets Layer) / TLS(Transport Layer Security) : 통신 데이터를 암호화하여 주요 정보를 안전하게 주고받을 때 사용

- NTP(Network Time Protocol): 네트워크에 연결된 장비들의 시스템 시간을 동기화할 때 사용

- LDAP(Lightweight Directory Access Protocol): 네트워크에 연결된 자원(사용자, 장비들)의 통합 관리에 필요한 디렉터리 서비스를 제공할 때 사용

HTTPS(HyperText Transfer Protocol Secure) 프로토콜

- HTTP의 보안이 강화된 버전의 프로토콜

- 소켓 통신에서 일반 텍스트를 이용하는 대신 SSL이나 TLS 프로토콜을 통해 세션 데이터를 암호화

  - SSL: 서명된 인증을 통해 암호화

  - TLS: 인증서 뿐만 아니라 통신 암호 기법을 사용해서 암호화

- 데이터 가로채기, 엿듣기 등으로부터 보호를 위해 데이터 전송 시에 암호화한다.

  - 통신하는 두 명은 약속된 인증 기관이 존재해서 서로의 데이터를 복호화 할 수 있다.

HTTP 개요

- HyperText Transfer Protocol

- 인터넷에서 사용자가 웹 애플리케이션 서비스를 요청하고 웹 서버는 사용자로부터 요청 받은 해당 서비스를 제공해주는 과정을

  원활하게 수행하기 위한 제반 요건을 표준화한 프로토콜을 의미

- client : 객체를 (HTTP 프로토콜로) 요청하고 받아서 "표시"하는 브라우저

- server : 요청에 의해 객체를 (HTTP 프로토콜로) 보내주는 웹 서버

- 클라이언트가 서버에게 HTTP request를 보내면 서버에서 클라이언트에게 HTTP response를 보낸다.

  -> 요청과 응답을 주고받으면서 객체를 가져다 실행한다.

- 파이어폭스와 사파리는 다른 프로그램이지만 똑같이 HTTP를 사용하기 때문에 같은 서버와 통신이 가능하다.

- HTTP는 TCP를 사용함 - 예시

  - 클라이언트가 (소켓을 만들고) 서버 80번 포트에 TCP 연결 요청

  - 서버는 TCP 연결 요청을 수락

  - 브라우저 (HTTP클라이언트)와 웹 서버(HTTP서버) 사이에 HTTP메시지 (애플리케이션 계층 프로토콜 메시지) 교환

  - TCP 연결을 닫음

- HTTP는 무상태(stateless) 프로토콜

  - 서버는 클라이언트의 과거 요청에 대한 정보를 유지하지 않는다.

- 상태(state)를 유지하는 프로토콜은 복잡하다

  - 과거 이력(상태) 관리 필요

  - 서버나 클라이언트가 다운되면 상태에 대한 정보가 일관성을 잃게 된다.

  -> 알아보는 경우 HTTP프로토콜 때문이 아니라 쿠키를 사용하기 때문.

웹 페이지를 구성하는 주요 파일

- HTML: 웹 페이지의 내용을 기술

- CSS: 웹 페이지 화면 표시 방법이 정의된 디자인 파일.

- JS: 사용자와 상호작용하는 동적인 웹 페이지를 만들 때 사용되는 자바스크립트로 작성된 파일.

- JPEG, PNG, GIF 등 이미지 파일, 동영상 파일

HTTP의 연결

- 비지속 연결(non-persistent) HTTP

  - 한 TCP 연결을 통해 최대 한 객체만 전송 -> 그리고 연결 해제

  - 여러 객체를 받으려면 여러 연결이 필요

  - 객체당 2RTT 필요

  - 각 TCP 연결에 따른 OS의 과부하

  - 브라우저는 참조된 객체들을 가져오기 위해 TCP연결 여러 개를 병렬적으로 열 수 있음

  - RTT(round-trip-time) : 작은 패킷이 클라이언트에서 서버까지 갔다가 돌아오는 데에 걸리는 시간

  - HTTP 응답 시간

  -> 2RTT + 파일 전송 시간 // 2RTT = TCP연결설립(RTT) + 요청과 응답의 처음 몇 바이트가 돌아오는데 걸리는 시간(RTT)

  -> 낭비가 심하다.

- 지속 연결(persistent) HTTP

  - 한 TCP연결을 통해 여러 객체를 전송.

  - 비지속 연결보다 편하다.

  - 서버는 응답을 보내고도 연결을 닫지 않는다.

  - 같은 클라이언트 / 서버 사이에서 한 번 통신한 이후의 HTTP 메시지들은 이 연결을 통해 전송

  - 모든 참조 객체들에 대해 RTT정도만 필요.

  - HTTP 1.1 기준으로 달리 명시되어 있지 않으면 모든 연결은 지속연결로 간주된다. 하지만 각 서버마다 타임아웃(TIEMOUT)이 존재한다.

  - 현재 대부분 사용 중 이다.

HTTP 요청 메시지

- 두 종류의 HTTP 메시지 : 요청(request), 응답(respone)

- HTTP 요청 메시지 : ASCII 포멧, URL 사용

- 헤더의 요청 라인(request line)

  - 예) GET/sample/index.html HTTP/1.1

  - GET :

    - 원하는 데이터를 달라는 요청. 별도의 메시지 바디가 없다.

    - URL의 '?' 뒤에 입력 폼의 내용이 붙는다. -> 웹 브라우저의 '이전 페이지' 버튼을 누르면 바로 앞 페이지가 표시되어 정적 웹 페이지처럼 다룰 수 있다.

  - POST :

    - 서버에게 데이터를 보냄(submit 등.) // 정보 저장, 변경 시 사용

    - 전송한 메시지를 해석하면 내용이 노출되기 때문에 보안상 안전하지 않다.

  - HEAD : 바디 없이 헤드만 있음.

  - GET/POST/HEAD 뒤에 디렉터리와 파일명, 프로토콜의 버전 정보가 기술됨

URL(Uniform Resource Locater)

- 예시) http://www.sample.co.kr/sampleD/index.html에서 분석:

- http: 스키마(schema) / 사용하는 프로토콜 명시

- www: 호스트(host) / 서버의 이름 혹은 역할 명시

- sample.co.kr: 도메인(domain) / 서버를 운영하는 조직 명시

- sampleD: 디렉터리(directory) / 서버 내의 디렉터리 명시

- index.html: 파일(file) / 해당 디렉터리 내의 파일을 명시

HTTP 헤더

- HOST: 서버의 도메인 명과 서버가 리스닝 중인 TCP 포트 번호 명시

- User-Agent : 클라이언트에서 사용중인 브라우저의 정보

- Accept : 포켓, 랭귀지 형태, 인코딩, 캐릭터 셋 등.

- Keep-Alive : 지속 연결을 사용한다. 얼마동안 할지 시간, 최대 몇 개의 요청을 받을 수 있는 지 표시

- Connection : 지속/비지속 중 무엇으로 하는지 설정

- 각 라인 끝날 땐(헤더라인의 끝 포함) ₩r₩n들어감. 한 라인이 끝났다는 의미이다.

- 헤더만 전송되는 경우도 있고, 그 아래 바디도 있는 경우도 있음.

방식의 종류

- HTTP/1.0

  - GET / POST / HEAD : 요청된 객체의 헤더만 전송(서버 정보 확인, 상태 체크, 버전, 최종 수정 일자 확인 등 관리 목적)

- HTTP/1.1

  - GET, POST, HEAD / PUT(데이터 추가 업로드), PATCH(일부분만 업데이트) / DELETE(URL 필드에 지정된 파일 지움) / TRACE, CONNECT, OPTIONS

HTTP 응답

- 응답 정보 행: 요청에 대한 응답 상태 표시를 위한 상태 코드가 들어간다. // 예) HTTP/1.1 200 OK

- 헤더: 파일 갱신 날짜나 크기 등 정보가 들어간다. // 예) Date: Mon, 3 April 2023 19:38:15 GMT

- 메시지 바디: HTML 파일 내용이 들어간다.

HTTP 응답 메시지

- 상태 라인(status line): 응답 코드

  - 100 Continue : 서버가 헤더는 받았고 바디가 올 것을 기다리고 있음

  - 101 Switching Protocols: 클라이언트가 서버에게 프로토콜을 바꾸자고 제안했고 서버도 수락한 상태

  - 200 OK: 요청이 성공했고, 요청된 객체가 이 메시지 뒤에 보내짐

  - 201 Created: 요청이 성공했고, 새로 만들어진 URL을 응답으로 보냈음. 웹 서비스나 웹 어플리케이션에서 사용된다.

  - 301 Moved Permanently: 요청된 객체가 이동되었고, 새 위치는 이 메시지 뒤에 표시됨(Location:)

  - 302 Found: 요청한 내용이 다른 경로로 옮겨진다. 임시로 옮겨졌기 때문에 이후에도 동일 경로로 요청해야 하는 상태.

  - 304 Modified: 요청한 내용은 갱신되지 않았음.

  // 400번대는 주로 클라이언트의 문제

  - 400 Bad Request: 서버가 요청을 이해할 수 없다 - 요청에 문제가 있음.

  - 403 Forbidden: 요청한 내용은 접근 금지됨

  - 404 Not Found: 요청한 문서가 서버에 없다.

  - 505 HTTP Version Not Supported // 500번대는 주로 서버 쪽 문제

- 100번대는 정보, 200번대는 성공, 300번대는 경로 전환, 400번대는 에러를 의미.

웹 서비스와 웹 페이지

- 브라우저가 서버에 요청하고 결과를 응답 받아 처리한다. 동작 방식상 큰 차이가 없다.

- 웹 서비스는 웹 서버가 결과를 미리 만들어 놓지 않고 서버 프로그램이 HTML 데이터를 동적으로 만들어 응답한다.

- 웹 페이지는 웹 서버가 미리 만들어 둔 정적인 HTML 파일의 데이터로 응답한다.

AJAX(Asynchronous JavaScript and XML)

- 웹 브라우저가 아닌 자바스크립트로 작성된 프로그램이 웹 서버와 통신

- 자바스크립트가 웹 페이지의 특정 부분에만 응답 받은 내용이 갱신되도록 처리

- 사용자 입장에서 전체 페이지를 다시 조회하지 않아도 되고, 페이지 새로 고침 없이 서버의 응답을 바로 받아 상호작용 할 수 있다. // 비동기성.

  -> 자연스러운 서비스 이용과 향상된 사용자 경험 제공.

- 사용 예: 구글 검색의 입력 필드는 입력 도중 검색 키워드와 유사한 키워드 후보를 표시함.

쿠키(cookies)

- 여러 건의 요청 처리를 동일한 사용자 접속 세션으로 인식할 수 있도록 하는 기술.

- 웹 브라우저는 HTTP 응답에 'Set-Cookie:' 문자열이 있는지 확인하고, 있다면 로컬 디스크에 쿠키 형태로 저장한다.

- 보안 문제

  - 웹 브라우저는 쿠키 정보의 악용으로 인한 보안 문제를 막기 위해 방어책을 자체적으로 구현하고 있다. // 쿠키 유효기간 등.

  - 보안에 문제될 만한 정보는 클라이언트 PC에 저장하지 말아야 한다.

  - 기본적으로 각종 정보는 서버 쪽에 저장하는 것이 원칙.

  - 동일 사용자 식별을 위한 세션 ID 정보만 클라이언트에 쿠키로 저장하도록 제한 필요.

- 네 가지 요소

- HTTP 응답 메시지의 쿠키 헤더 라인

- 응답 메시지의 쿠키를 저장했다가 다음 HTTP 요청 메시지의 쿠키 헤더 라인에 저장된 데이터를 함께 전송함.

- 사용자 호스트에 저장된 쿠키 파일 (웹 브라우저가 관리) // 쿠키가 저장되면 나의 이력을 알아볼 수 있음

- 웹 사이트 백엔드의 데이터베이스(각 클라이언트의 이력을 저장해 놓음)

- 예시 : 클라이언트가 특정 사이트에 처음 접속하면, 고객에게 고유 ID 부여, 이 ID에 대한 백엔드 데이터베이스의 엔트리 생성

다음번에 클라이언트가 접속하면, 서버는 데이터베이스에 있는 쿠키에 해당하는 동작을 수행하여 응답.

SMTP

- Simple Mail Transfer Protocol

- 전자메일을 신뢰성 있게 전송하기 위해 TCP 채용. 25번 포트 사용

- 클라이언트 PC가 메일 서버로 메일 보낼 때, 발신자 메일 서버에서 수신자 메일 서버로 메일 중계할 때, 메일 서버 간 메일 중계에 사용됨.

- Stateful 프로토콜이므로 전송 종료 명령이 보내져야 통신을 종료한다.

- POP와 같은 사용자 인증 체계가 없어서 스팸 메일 등 종종 악용된다.

- POP 서버의 인증 기능을 활용하거나, 다른 네트워크로부터의 SMTP 접근을 제한하는 대안을 사용하기도 한다.

- SMTP Auth: SMTP에 사용자 인증 기능이 추가된 확장 프로토콜.

- 전송의 세 단계

- handshaking(인사) // TCP 커넥션을 열기 위한 단계

- 메시지 전송

- 닫기 // TCP 커넥션을 닫음

- 명령/응답 상호작용 (HTTP에서처럼)

- 명령 : ASCII text

  - 응답 : 상태 코드와 문장

- 메시지는 7-bit ASCII

POP

- Post Office Protocol

- 메일 서버에 저장된 메일 확인할 때 사용된다.

- 메일 수신, 수신한 메일 건수, 용량 확인, 메일 삭제 등 처리

- 현재 사용되는 버전은 3이고, 하위 버전과 차이가 크기 때문에 버전 혼용을 예방하기 위해 프로토콜을 언급할 때 POP3라고 명시한다.

IMAP

- Internet Messaging Access Protocol

- POP 프로토콜과 달리 클라이언트 PC가 메일을 수신하더라도 메일 서버에서 수신한 메일을 지우지 않고 보관

P2P

- peer to peer

- 특별히 공유할 서버를 준비할 필요 없고 공유할 컴퓨터끼리 네트워크에 접속하면 파일 공유가 가능하다.

- 중앙에서 관리하는 서버가 없다.

- 공유 상대 찾기

  - 새로 네트워크에 연결되는 컴퓨터는 다른 모든 컴퓨터에게 자신의 존재를 알린다.

  - 통보 받은 다른 컴퓨터는 자신의 정보를 응답으로 알려주고, 결과적으로 네트워크 전체에서 공유 가능한 컴퓨터들을 서로 식별할 수 있게 된다.

파일 공유 기능

- 운영체제마다 서로 프로토콜이 다른다.

- 윈도우: SMB(Server Message Block)

- 맥: AFP(Apple Filing Protocol)

질문자: 김인우

1. HTTP 요청에서 GET 요청인 경우 메시지 바디가 있다. O / X

2. 웹 페이지를 구성하는 주요 파일 중 JS는 사용자와 상호작용하는 정적인 웹 페이지를 만들 때 사용된다. O / X

3. HTTP 응답의 응답 정보 행에는 상태 코드가 포함된다. O / X

4. http://search.service.com/serch?q=TCP%2FIP 요청은 POST방식을 사용한 URL이다. O / X

5. AJAX에서 페이지 새로 고침 없이 서버의 응답을 바로 받아 상호작용 할 수 있다. O / X

6. 쿠키는 보안에 문제될 만한 정보는 반드시 클라이언트 PC에 저장해야 한다. O / X

7. 파일 공유 프로토콜은 운영체제마다 서로 다르다. O / X

답

1. X

2. X

3. O

4. X

5. O

6. X

7. O

질문자: 최지은

1. HTML은 웹 페이지의 내용을 담은 텍스트 파일이다 (o/x)

2. URL은 HTTP 요청을 보낼 때 사용하는 문자열이다 (o/x)

3. HTML 요청을 보낼 때 사용하는 GET방식은 POST방식보다 더 많은 데이터를 전송할 수 있다 (o/x)

4. 쿠키 정보는 악용될 경우 보안 문제가 발생할 수 있다 (o/x)

5. 이메일에서 사용되는 애플리케이션 계층 프로토콜 중 발신할 때는 POP을 사용한다 (o/x)

6. SMTP 프로토콜은 스테이트풀 프로토콜이다 (o/x)

7. POP3라고 버전을 언급하는 이유는 다른 버전과 혼용될 경우 오동작의 발생을 방지하기 위함이다 (o/x)

답

1. O

2. O

3. X

4. O

5. X

6. O

7. O