Ch6. Application Protocol

Created	@Jun 17, 2020 6:14 PM
Tags	컴망

응용 단에서, 애플리케이션에서 사용하고 있는 프로토콜에는 무엇이 있는가?

Process communicating

프로세스끼리 통신한다. 프로세스 : 호스트에서 돌아가는 프로그램

(동일한 호스트 내의 프로세스끼리 통신은 IPC)

Socket

프로세스는 소켓을 사용해서 메시지를 보내고 받는다.

Addressing process

프로세스를 어떻게 식별할 것이냐? 호스트는 IP주소, 호스트 내의 프로세스는 포트 번호로 식별.

Application Layer Protocol

- 정의하는 것
  - 메시지 타입, 메시지 문법, semantics (메시지에 들어 있는 정보의 뜻), rules 등
- open protocols들도 있고, proprietary protocol도 있다
- 이런 걸 지켜야 한다
  - Data integrity : 보내는 것과 받는 것이 같아야 한다.
  - Throughput
  - Security
  - Timing

Securing TCP

TCP, UDP는 암호화가 없다.

→ SSL!

- SSL은 애플리케이션 레이어에 있다.
- 애플리케이션들은 SSL 라이브러리를 사용한다.

HTTP

- HTTP도 일종의 클라이언트-서버 연결이다.
- TCP 사용 : Data consistency를 지켜야 한다.
  - UDP도 사용할 수 있다. (QUIC : UDP 위에서 돌아감)
- HTTP is "stateless".
  - 서버는 클라이언트의 이전 요청 기록을 기억하지 않는다.

HTTP Connection

non-persistent HTTP

object를 받을 때마다 TCP conneciton을 새로 맺는다.

persistent HTTP

한 TCP connection에서 여러 object를 받는다.

HTTP Message

HTTP 메시지는 두 타입이 있다 : Request, Response.

HTTP Request

HTTP request message : ASCII 형식.

HTTP/1.0

GET, POST, HEAD

HTTP/1.1

GET, POST, HEAD, PUT, DELETE

HTTP Response

Status codes:

- 200 OK
- 301 Moved Permanently
- 400 Bad Request
- 404 Not Found
- 505 HTTP Version Not Supported (HTTP 버전 불일치)

User-server state : cookies

쿠키 : Keeping "state". 클라이언트 측에 클라이언트의 상태를 저장한다.

Web caches (Proxy server)

프록시 서버에 먼저 접근해서, 프록시 서버에 캐싱된 게 있으면 그걸 받아오고, 없으면 origin server로 간다.

그런데, 문제가 있다.

- origin server의 데이터가 변경되었으면? 클라이언트는 프록시 서버에서 다른 정보(캐싱된 이전 정보)를 가져오게 된다.

Conditional GET

> 💡 Remind: HTTP Response에 Last-Modified 필드가 있다.

프록시 서버에서 origin server에 if-modified-since를 보내서, 바뀌었으면 바뀐 데이터를 받아와서 넘겨주고, 안 바뀌었으면 캐시에 있는 거 그대로 준다.

💡 HOL(Head-Of-Line) Blocking : 먼저 들어온 패킷 때문에 뒷 패킷이 처리되지 못하는 상황.

HTTP/3.0 - Connection ID

IP주소가 바뀌어도 Connection을 새로 만들 필요는 없다.

FTP : File Transfer Protocol

Control connection, Data connection이 따로 있다.

- **TCP control connection ("out of band") : 21번 포트**
- **TCP data connection : 20번 포트**

FTP Commands, FTP Responses 둘 다 ASCII 텍스트이다.

Email

3개의 중요 요소:

- **User agent**
- **메일 서버**
- **SMTP** : Simple Mail Transfer Protocol
  - TCP 사용. 25번 포트. ASCII 사용.
  - Handshaking → 메시지 전달 → Closure

💡 **HTTP는 Pull**(일방적 정보 수신), **SMTP는 Push**(일방적 정보 전송)

💡 **SMTP** : 메일서버에 보내거나, 메일서버끼리 통신할 때 사용.
유저(클라이언트)가 메일서버로부터 Mail을 받아오는 **Mail access protocol**은 3가지가 있다.
- POP
- IMAP
- HTTP (웹메일)

DNS : Domain Name System

네임서버들의 **분산된 계층구조 데이터베이스**

💡 왜 중앙화하지 않는가? 나중에 크기를 더 늘리기 힘들다. Scaling 힘듦. 관리 문제 등.

Local DNS 서버에게 이 도메인에 해당하는 IP주소가 무엇인지 query를 날리는 것.

- **Iterated query** : Local DNS 서버에서 여러 DNS 서버에게 질의를 주는 것. (local 주변에서 BFS)
- **Recursive query** : 알면 넘겨주고, 모르면 다른 DNS 서버 주소를 알려준다. 거기로 가서 다시 물어본다. 거기 가서 물어봐라. (DFS)

Pure P2P Architecture

- NO always-on server
- 임의의 end system들이 직접 연결할 수 있다.

P2P file distribution

서버에 파일을 올리는 게 아니라, **서버에는 파일명, 소유자 등 파일의 메타데이터만 저장한다.**

클라이언트는 이를 보고, 파일을 가지고 있는 클라이언트에게 가서 파일을 가져온다.

- 이는 P2P인가? P2P로 볼 수 있다.
  서버에 파일 자체를 올리는 게 아니라 직접 파일을 주고받는 것이므로.
- 그러나 서버는 있다. 서버에 파일을 누가 가지고 있는지에 관한 메타데이터는 저장한다.

> 💡 서버를 없앨 수도 있다.
> 아예 Peer끼리 원하는 파일에 대한 질의를 쭉 퍼트리고, 파일 가지고 있는 애가 이를 보면 전송해주는 식.

P2P file distribution : BitTorrent

파일을 Chunk로 잘라서, 각자 가지고 있는 거 각자에게 받아온다.

Requesting chunks

파일을 원하는 Peer는 주기적으로 Peer 각자가 가지고 있는 chunk 리스트를 요청해서, **가장 드문(Rare한) Chunk부터 요청**한다.

Sending chunks : tit-for-tat

자신에게 높은 Rate로 Chunk를 주는 Peer에게만 높은 Rate로 전송한다.

DHT : Distributed Hash Table

DHT : 분산된 P2P (key, value)쌍 데이터베이스. Peer는 각자 DHT를 가지고 있다.

- **Peer가 key로 질의하면, DHT가 value를 리턴**해준다.
- **Peer가 (다른 Peer의 DHT에) insert**할 수 도 있다.
  - Key는 Hash function을 통해 integer로 변환되어 저장.
  - 각 Peer마다 번호가 붙어 있으므로, **Key 번호 바로 다음으로 큰 Peer의 DHT에 insert한다.**
  - *ex) Peer가 1, 3, 4, 5, 8, 10, 12, 14 있고, 13을 insert하려면 Peer 14에게 넣으면 된다.*

Circular DHT

각 Peer는 자신의 바로 이전 Peer 번호, 다음 Peer 번호만 알고 있다.

Peer가 key값 받으면, 그냥 나올 때까지 다음 애한테 계속 넘기는 것 → Peer 수 N일 때 O(N)

Circular DHT with shortcuts

각 Peer가 이제 바로 갈 수 있는 shortcurt도 기억한다.

→ **O(logN)**이 되게 만들 수 있다.

Peer churn

Peer가 중간에 들어오거나 나가는 경우가 있을 수 있다.

각 Peer는 주기적으로 자신의 이전 Peer와 다음 Peer를 ping해서 살아 있는지 체크 후, 죽었으면 그 이전/다음 Peer랑 잇는다.

CDN : Content Distribution Network

CDN : 컨텐츠 복사본을 다수의 CDN 노드에 저장해둔다.

→ 요청이 들어오면, 요청 위치로부터 가장 가까운 CDN에서 가져온다.