8.WebSocket

Socket

- 파이프의 끝, 파이프 이음쇠
- 전구 소켓
- CPU소켓(Intel이 주도, CPU업글 메인보드를 바꿔야 하는 이유)
- 네트워크 소켓

IP Address

- IPv4 (32비트 주소 0xFF 0xFF 0xFF 0xFF)
- Subnet Mask
- GateWay
- DNS Server / DDNS Server

IP Address

- IPv6 -> 128bit주소
- 127.0.0.1 / localhost / loopback
- 10.0.0.* / 192.168.0.* / 172.16.0.*
- Apple appstore리젝사유

Port

- 항구?
- Server Port(같은 Port를 사용할 수 없음)
  - 0~1023 / 1024 ~ 49151 / 49152 ~ 65535
  - Client Port
  - FTP(21) SSH(22) Telnet(23) Web(80) SSL(443)

기본포트(숨겨진포트)

- http://www.lucidcore.co.kr:80
- http://www.lucidcore.co.kr
- https://www.lucidcore.co.kr:443
- http://www.lucidcore.co.kr:8080
- ssh 10.0.0.0 / ssh 10.0.0.0 -p 8080

Socket vs HTTP

- Stateful / Stateless
- 동시접속가능자수의 제한
- 10K 동시접속자 구현
- HTTP tunneling
- Push / Polling

Socket Example

서버 접속
let rs: Int32 = c_ytcpsocket_connect(self.address, port: Int32(self.port), timeout: Int32(timeout))

데이터 전송
var buff = [Byte](repeating: 0x0,count: data.count)
(data as NSData).getBytes(&buff, length: data.count)
let sendsize: Int32 = c_ytcpsocket_send(fd, buff: data, len: Int32(data.count))

데이터 수신
var buff = [Byte](repeating: 0x0,count: ExpectLen)
let readLen = c_ytcpsocket_pull(fd, buff: &buff, len: Int32(expectlen), timeout: Int32(timeout))

분할 송수신

- MTU (Maximum Transmission Unit) 최대 프로토콜 데이터 단위의 크기 - 네트워크에 따라서 다름.
- 데이터는 Packet단위로 전송/수신됨.(보통 8K이상 전송/수신되지 않음 더 적을 수도 있음) 데이터 송/수신시 받아야 하는 크기 만큼 반복해서 송신하거나 수신해야 한다.
- 모든 Input/Output Stream형태는 동일하게 반복해야 함.

WebSocket

- 브라우저에서 Socket통신을 통해 Push를 지원받기 위해 제공
- 브라우저에서 지원하지 않으면 사용할 수 없음(과거 ActiveX로 지원한 사례는 있음)
- 일반소켓에 보안기능을 추가적으로 지원

WebSocket

- 프로토콜이므로 일반앱에서도 사용가능하다.
- ws://… / wss://…

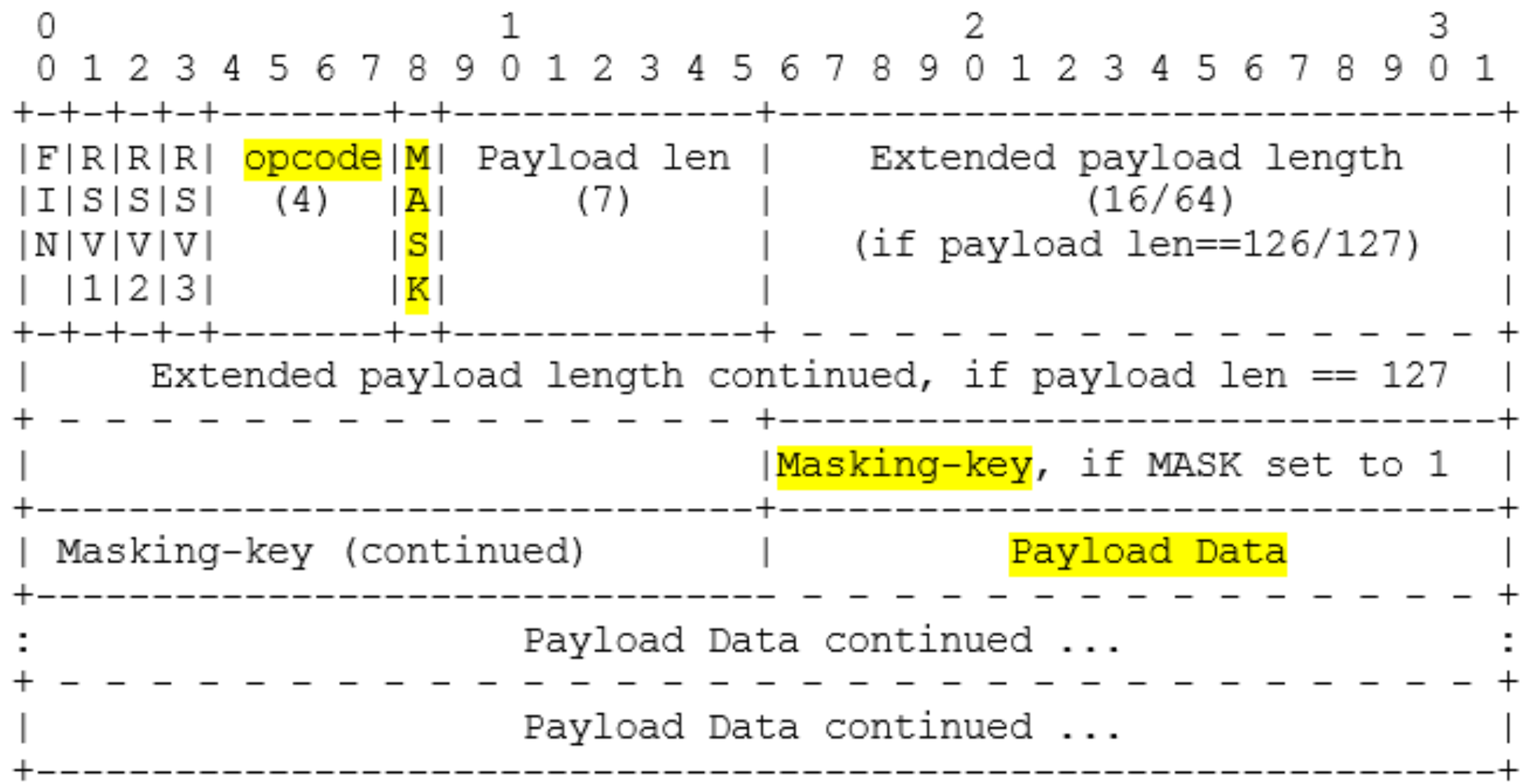
요청

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

응답

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

WebSocket Data Frame Format



WebSocket Library

```
@objc func onConnect() {
	__socketClient = SRWebSocket(url: URL(string: "wss://
pineplus.co.kr:8443/ws")!)
	__socketClient.delegate = self
	DispatchQueue.global(qos: DispatchQoS.QoSClass.background).async {
		self.__socketClient.open()
	}
}

do {
	try self.__socketClient.send(string: s)
} catch {
	print(error)
}

func webSocket(_ webSocket: SRWebSocket, didReceiveMessage message:
Any) {
	writeOutput(String(format: "%@", message as! CVarArg))
}
```

WebSocket Library

```
__socketClient = new WebSocketClient(uri) {
    @Override
    public void onMessage(final String message) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                writeOutput(message);
            }
        });
    }
}
__socketClient.connect();

__socketClient.send(inputEdit.getText().toString());
```

STOMP

- Simple/Stream Text Oriented Message Protocol
- Pub/Sub기반의 메시지 브로커
- Messaging Queue (카프카/모스키토 등등)
- 미래에셋이 이 것을 기반으로 웹소켓 프로그램 진행중이며 거의 다 되었다고 함.(서버때문에 선택했을 것으로 추정)