# 방 입장하는 화면과 tcp.port == 8080 client 어떻게 들어오는지

**<1. 개요>**

온라인 수업으로 인해 따로 모임이 불가능해 서버, 클라이언트 노트북 1대로 시행

홈페이지를 따로 만들었다: 주소 http://114.204.208.162:8080/home)

게임시작 링크를 따로 두었다: (http://114.204.208.162:8080/game)

**초성게임을 만들었다**

**: 세부 rule)**

**: 중복 단어의 경우 퇴장**

**: 국립국어원 표준국어대사전 api 참조해 존재하지 않는 단어인 경우 퇴장**

**: 초성 불일치시 퇴장**

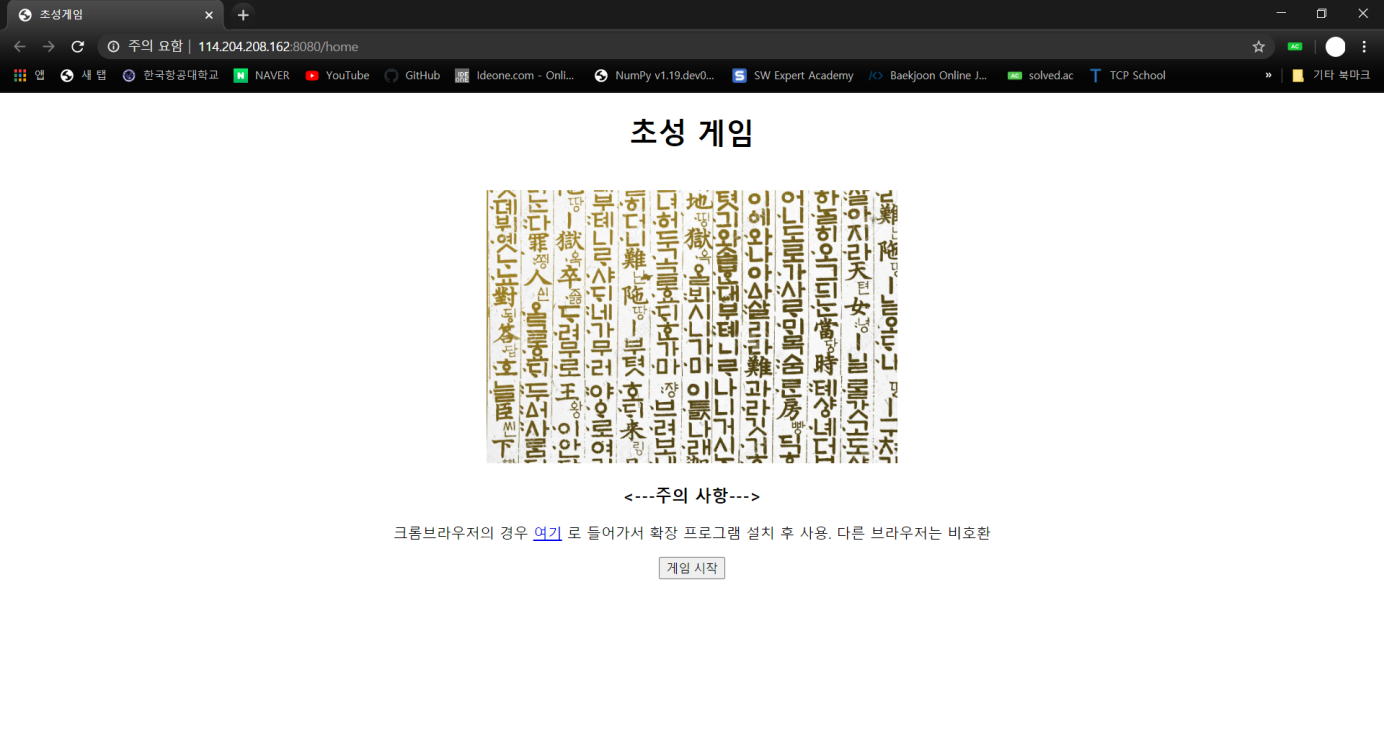
**: 게임에 접속하는 순서대로 자동 이름 부여**

wireshark 프로그램을 사용했다

**다른 브라우저는 안되는 이유:**

국어사전 api를 참조할 때, 교차 출처 리소스 공유(Cross-origin resource sharing)의 문제로 Access-Control-Allow-Origin 헤더를 추가해야 했다. 그러나 Vanilla JavaScript를 사용했기 때문에 외부 라이브러리를 사용할 수 없어 브라우저 내에서 해결해야 했다. 이를 위해 크롬브라우저를 사용해 확장프로그램을 설치하고 사용하기로 했으며, 파이어폭스의 경우도 확장프로그램을 설치하면 사용이 가능하나 불안정하다는 것까지 확인했다.

<2. 패킷 전송 및 수신 과정>



http://114.204.208.162:8080/home : client 프로그램(초성게임) 시작화면

게임 시작화면인 http://114.204.208.162:8080/home으로 접속하면 나오는 화면으로, javascript ajax 크로스 도메인을 요청\*한 후, 게임시작 버튼을 누르면 초성게임 방(http://114.204.208.162:8080/game)으로 입장한다.

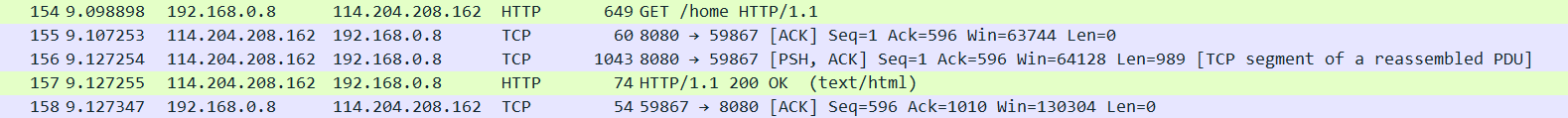
\* CORS 크롬 확장 프로그램 설명(일단 설명은 해봤는데, 패킷분석은 아니라 빼도 됩니다.)

자바스크립트 엔진 표준 스팩의 동일 출처 정책(same-origin policy)에 의해서 웹페이지의 스크립트는 그 페이지와 같은 서버에 있는 주소로만 ajax 요청을 할 수 있다. CORS(Cross-Origin Resource Sharing)는 서버에서 외부 요청을 허용할 경우 ajax요청이 가능해지게 한다. 화면에서 설치를 요구한 것은 크롬에서 발생하는 모든 http 요청을 가로채서 응답에 위 header를 추가해주는 플러그인이다. 서버에서 받은 요청의 응답에 특정 header(Access-Control-Allow-Origin: \*)만 추가하면 웹 브라우저가 요청이 가능한 사이트로 인식해서 요청이 가능함을 이용한 것이다.

**2-1방 입장\_client1**

**2-1-1. 시작화면 입장**

먼저 client가 게임 시작화면(http://114.204.208.162:8080/home)으로 들어간 후, 게임시작 버튼을 클릭하여 초성게임 방(http://114.204.208.162:8080/game)으로 입장한다.



<그림 1> client1이 초성게임 시작화면으로 들어가는 과정의 패킷

|  |  |  |  |
| --- | --- | --- | --- |
| no.154 | client1 → server | HTTP | GET /home HTTP/1.1 |
| 서버에게 home.html 파일을 요청(GET)하고, HTTP 프로토콜 버전 1.1로 통신하기를 원한다는 의미의 패킷으로, 게임 시작화면으로 연결할 것을 요청한 것이다.    **<그림 2>**  Packet detail을 보면, frame의 프로토콜이 eth:ethertype:ip:tcp:http임을 알 수 있다. Ethernet II 에서 MAC 주소가 Src: ASUSTekC\_e0:7a:0e (38:d5:47:e0:7a:0e), Dst: EFMNetwo\_b2:35:dc (88:36:6c:b2:35:dc) 임을 확인할 수 있다.  Network layer는 IPv4를 사용한다. 각각의 IP가 Src: 192.168.0.8, Dst: 114.204.208.16 이며, Flags:는 0x4000으로 fragment되지 않았다. offset은 0이며 TTL은 128이다.  TCP(Transmission Control Protocol)를 보면 Src Port: 59867, Dst Port: 8080 이고, Flag는 PSH, ACK이고, Window size는 513이다.  HTTP(Hypertext Transfer Protocol)에서 응답 받을 브라우저 정보와 사용자 인증정보 등의 부가적인 정보가 담겨있으며, http://114.204.208.162:8080/home로 연결할 것을 요청했다. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| no.155 | server → client1 | TCP | 8080->59867 [ACK] |
| 154번 패킷에 대한 확인(ACK) 세그먼트. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| no.156 | server → client1 | TCP | 8080->59867 [PSH, ACK]  [TCP segment of a reassembled PDU] |
| 157번 패킷과 재결합된 PDU의 세그먼트. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| no.157 | server → client1 | HTTP | HTTP/1.1 200 OK |
| HTTP 프로토콜 버전 1.1 기준으로 응답하겠으며, 154번 패킷에서의 요청이 성공적으로 처리되었다는 의미이다.    <그림 3>  TCP(Transmission Control Protocol)에서 156번 segment와 해당 패킷인 157번 segment가 reassambled 되었음을 알 수 있다.  Line-based text data: text/html (21 lines)에서 home.html 내용을 확인할 수 있다. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| no.158 | client1 → server | TCP | 59867->8080 [ACK] |
| 157번 패킷에 대한 확인(ACK) 세그먼트. | | | |

**1-1-2. 초성게임 방 입장**

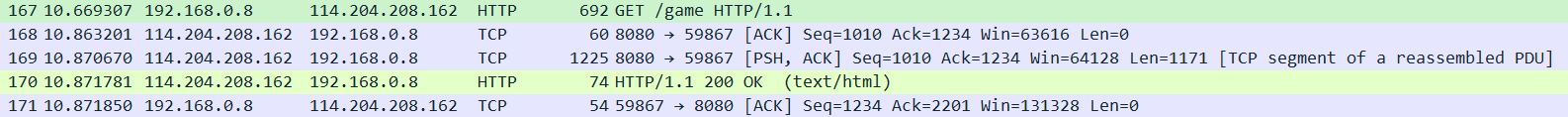


그림 3 client1이 초성게임 방으로 들어가는 과정

|  |  |  |  |
| --- | --- | --- | --- |
| no.167 | client1 → server | HTTP | GET /home HTTP/1.1 |
| 서버에게 game.html 파일을 요청(GET)하고, HTTP 프로토콜 버전 1.1로 통신하기를 원한다는 의미의 패킷으로, 게임 방으로 연결할 것을 요청한 것이다.    마찬가지로 frame의 프로토콜이 eth:ethertype:ip:tcp:http임을 알 수 있다. MAC 주소는 Src: ASUSTekC\_e0:7a:0e (38:d5:47:e0:7a:0e), Dst: EFMNetwo\_b2:35:dc (88:36:6c:b2:35:dc)이고, IP 주소는 Src: 192.168.0.8, Dst: 114.204.208.16, 포트는 Src Port: 59867, Dst Port: 8080으로 전과 동일하다.  IPv4에서 Flags: 0x400으로 fragment가 되지않았고, offset은 0, TTL은 128, Header checksum은 0x5307임을 알 수 있었다.  TCP(Transmission Control Protocol)에서의 Flags는 PSH,ACK였고, Window size는 506, checksum은 0x1a8b였고, TCP payload에는 638bytes가 있었다.  HTTP(Hypertext Transfer Protocol)에서 응답 받을 브라우저 정보와 사용자 인증정보 등의 부가적인 정보가 담겨있으며, http://114.204.208.162:8080/game으로 연결할 것을 요청했다. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| no.168 | server → client1 | TCP | 8080->59867 [ACK] |
| 167번 패킷에 대한 확인(ACK) 세그먼트이다. TTL은 52이고, window size는 497이다. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| no.169 | server → client1 | TCP | 8080->59867 [PSH, ACK]  [TCP segment of a reassembled PDU] |
| 170번 패킷과 재결합된 PDU의 세그먼트. TTL은 52이고, window size는 501이다. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| no.170 | server → client1 | HTTP | HTTP/1.1 200 OK |
| HTTP 프로토콜 버전 1.1 기준으로 응답하겠으며, 154번 패킷에서의 요청이 성공적으로 처리되었다는 의미이다.    TCP(Transmission Control Protocol)에서 169번 segment와 해당 패킷인 170번 segment가 reassambled 되었음을 알 수 있다.    Line-based text data: text/html (36 lines)에서 game.html 내용을 확인할 수 있으며, 해당 게임방의 랜덤초성은 “ㅈㅁ”이 나왔다. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| no.171 | client1 → server | TCP | 59867->8080 [ACK] |
| 170번 패킷에 대한 확인(ACK) 세그먼트. TTL은 128이고, window size는 513이다. | | | |

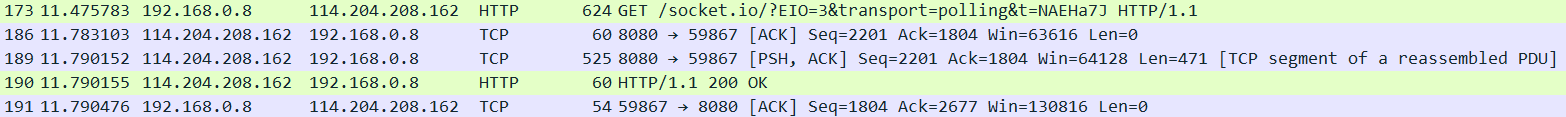


그림 4 초성게임방에 입장한 후, client1이 몇번째 user로 connected되었는지 화면에 표시하는 과정.

|  |  |  |  |
| --- | --- | --- | --- |
| no.173 | client1 → server | HTTP | GET/socket.io/?EIO=3&transport=polling&t=NAEHa7J HTTP/1.1 |
| http://114.204.208.162:8080/socket.io/?EIO=3&transport=polling&t=NAEHa7J]에 대해서 HTTP 프로토콜 버전 1.1로 통신하기를 원한다는 의미의 패킷으로, 이후 화면에 user가 connected되었다는 문구가 나오게 할 것이다. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| no.186 | server → client1 | TCP | 8080->59867 [ACK] |
| 173번 패킷에 대한 확인(ACK) 세그먼트이다. TTL은 52이고, window size는 497이다. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| no.189 | server → client1 | TCP | 8080->59867 [PSH, ACK]  [TCP segment of a reassembled PDU] |
| 190번 패킷과 재결합된 PDU의 세그먼트. TTL은 52이고, window size는 501이다.    TCP segment data 471 byte 중 일부에 user name과 data에 대한 정보가 있다. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| no.190 | server → client1 | HTTP | HTTP/1.1 200 OK |
| HTTP 프로토콜 버전 1.1 기준으로 응답하겠으며, 173번 패킷에서의 요청이 성공적으로 처리되었다는 의미이다.    TCP(Transmission Control Protocol)에서 189번 segment와 해당 패킷인 190번 segment가 reassambled 되었음을 알 수 있다. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| no.191 | client1 → server | TCP | 59867->8080 [ACK] |
| 190번 패킷에 대한 확인(ACK) 세그먼트. TTL은 128이고, window size는 256이다. | | | |

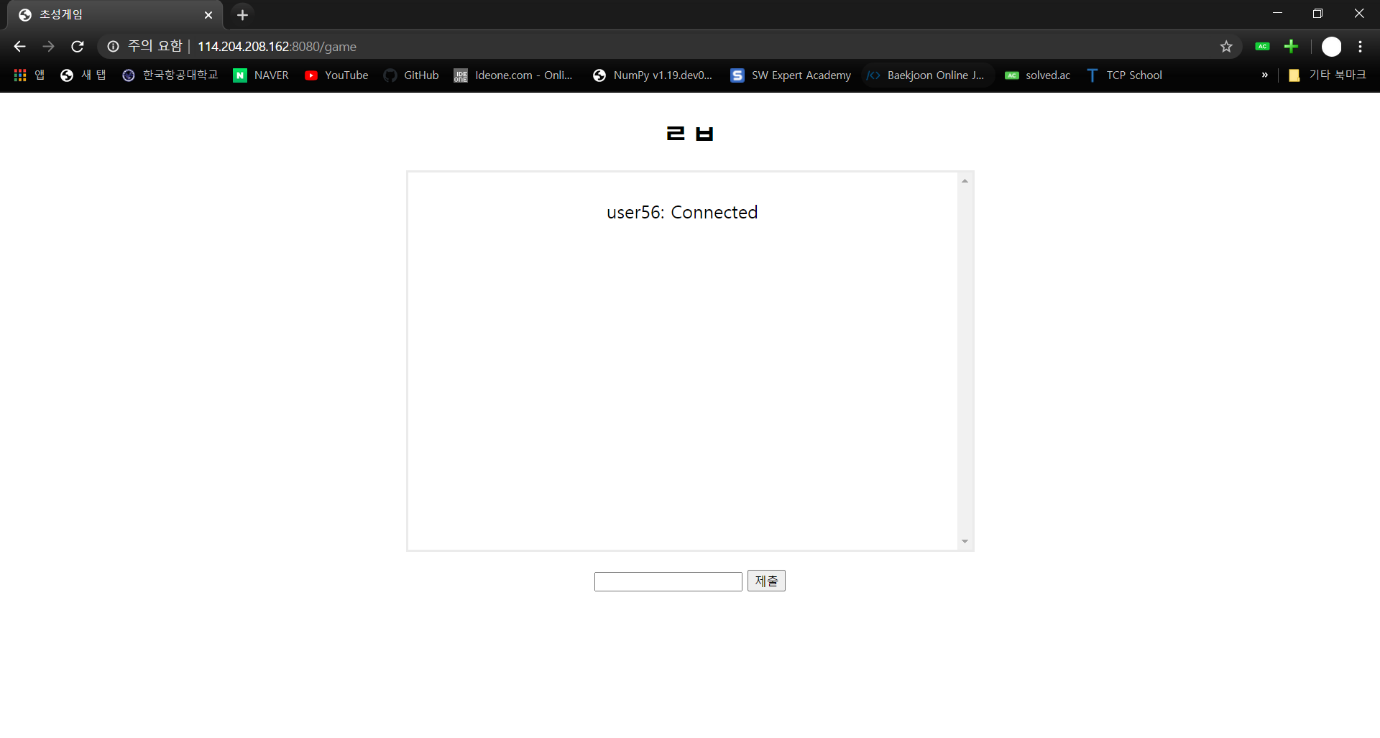


그림 5 http://114.204.208.162:8080/game : 초성게임 방에 들어온 화면. user56으로 connected되었으며, 캡쳐 당시의 랜덤초성은 “ㄹㅂ”이었다.

**2-2. 초성게임 방 입장\_client2**

**1-2-1. 시작화면 입장**

client2 또한 client1과 동일한 입장 과정을 거치며, 소스의 MAC주소: QuantaCo\_14:0a:7e (d8:c4:97:14:0a:7e), IP주소: 192.168.0.25, TCP Port: 49966로 client1과 주소와 포트에서 차이가 있다.

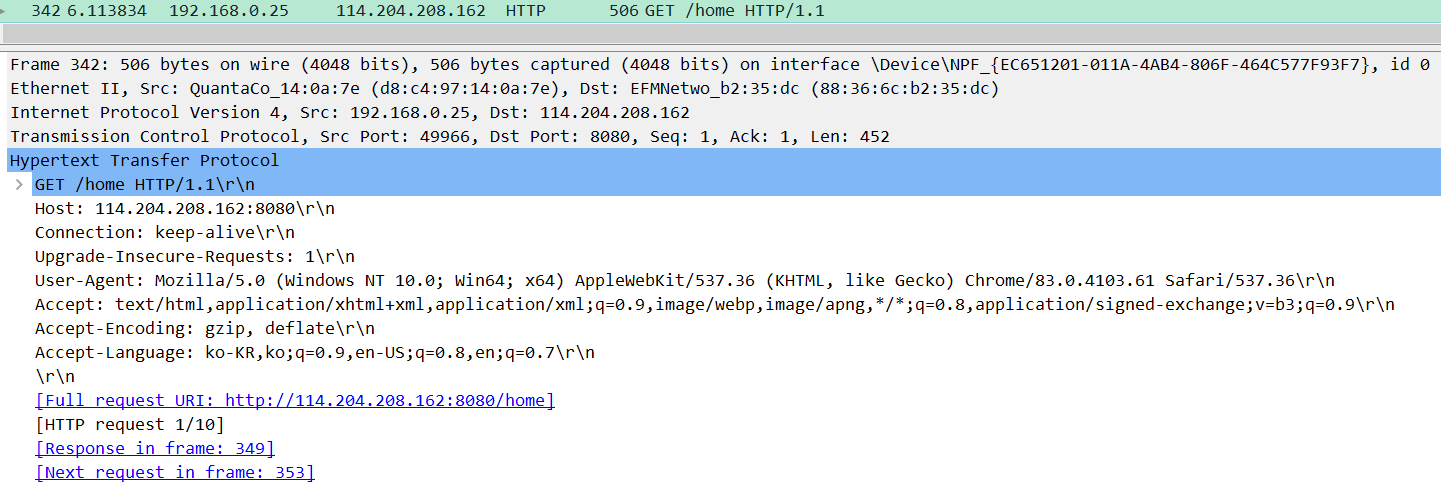


그림 6 client2가 게임시작 화면으로 들어가는 과정의 첫번째 패킷

다만 client2는 /home과 /game사이에 /favicon.ico로 연결을 시도하는데, HTTP 상태코드 404 NOT FOUND로 존재하지 않는 파일이라는 패킷을 받는다.

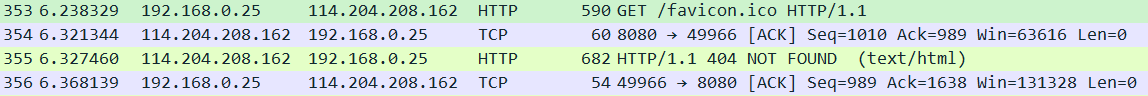
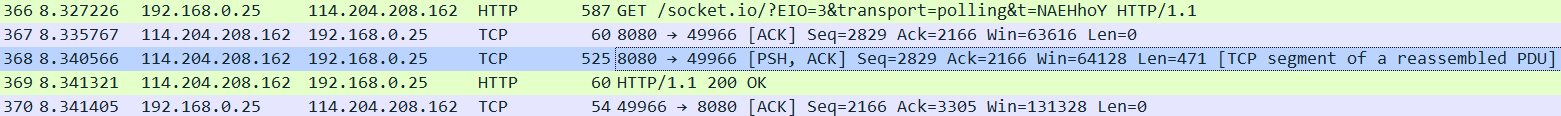


그림 7 no.353~no.356의 HTTP 연결 실패 과정

**2-2-2. 초성게임 방 입장**

client1과 마찬가지로 GET /game HTTP/1.1로 연결 성공한 후, 366번 패킷의 GET /socket.io /?EIO=3&transport=polling&t=NAEHhoY HTTP/1.1 을 통해 user name “user8”로 connected 문구를 표시한다.



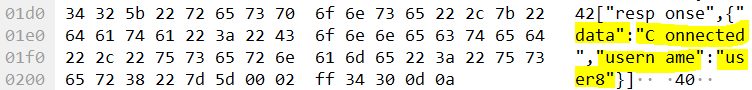


그림 8 client2가 초성게임방에 입장한 후, user8 connected로 화면에 표시하는 과정.

/\*

주석 부분은 제거하고 보고서에 추가하시면 됩니다.

아파치 서버를 이용한 application layer의 HTTP/DNS/TCP 패킷 분석

Client1 - port number: 59867 / filter : tcp.port==59867 || DNS / mac : (38:d5:47:e0:7a:0e) / ip : 192.168.0.8

Client2 - port number: 49966 / filter : tcp.port == 49966 || DNS / mac : (d8:c4:97:14:0a:7e) / ip : 192.168.0.25

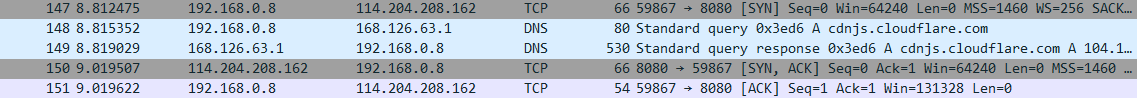
Server port number: 8080 / mac : (88:36:6c:b2:35:dc) / ip : 114.204.208.162

두 클라이언트의 캡쳐를 따로 진행하였기 때문에 필터링은 각 클라이언트의 port number+dns로만 진행하였습니다.

\*/

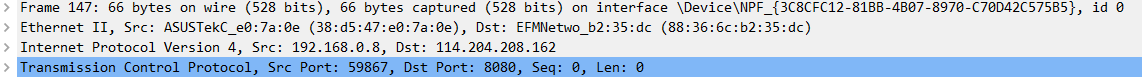
**Client 1**

1. Client 1 – server간의 3way-handshaking을 보여주는 부분을 캡쳐한다.



<그림 1> Client 1과 서버의 3way-handshaking을 통한 연결을 보여준다.

1. [3way-handshaking] Client가 통신을 하기 위해 패킷 번호(Client 1에선 59861)을 지정한 후 Client 1은 포트번호를 포함한 SYN 패킷을 서버에 전송하여 연결을 요청한다.



<그림 2> Client 1에서 서버로 SYN을 보내는 것을 보여주는 147번 패킷

* **147번 패킷 상세 분석**

**Frame**

해당 패킷의 Frame number가 147번이고, interface id는 0이다. 패킷의 Encapsulation type이 Ethernet임을 알 수 있다.

**Ethernet 2 Frame header**

source(Client1)의 mac 주소는 (38:d5:47:e0:7a:0e)이고, destination(Server)의 mac 주소는 (88:36:6c:b2:35:dc)이다. Type은 IPv4이다.

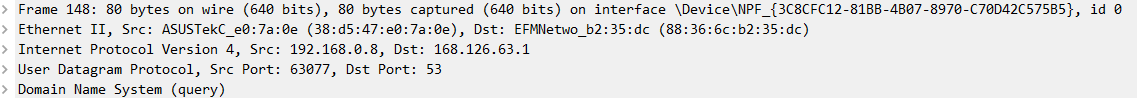
**IPv4 (Internet protocol)**

source(Client 1)의 IP주소는 192.168.0.8, destination(Server)의 IP주소는 114.204.208.162이다. Fragment Flag로 인해 Fragment 되지 않았고, Header checksum이 0x557e임을 알 수 있다.

**TCP(Transmission Control Protocol)**

Source(Client 1)의 port는 59867, destination(Server)의 port는 8080이다. Flag를 통해 해당 패킷이 SYN이고, TCP checksum은 0xe576임을 확인할 수 있다.

1. [DNS] Client 1이 초성게임 URL(“http://114.204.208.162:8080/home”)을 입력하면, Client 1은 Local DNS에 해당 URL의 IP주소를 요청한다.



<그림 3> Client 1에서 Local DNS에게 Query를 보내는 것을 보여주는 148번 패킷

* **148번 패킷 상세 분석**

**IPv4**

해당 패킷의 Source IP는 Client 1의 IP주소인 192.168.0.8이지만, destination IP주소는 Local DNS의 IP주소인 168.126.63.1이다. 또한 DNS는 UDP를 사용하며, TTL은 128, Header checksum은 0xfcfe이다.

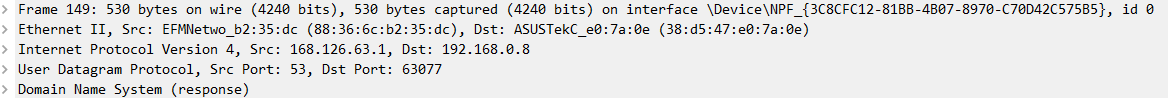
**UDP(User Datagram Protocol)**

source port는 Client 1의 임의의 포트인 63077이고, destination port는 Local DNS 지정 port인 53이다. UDP checksum은 0x6320임을 알 수 있다.

**DNS(Domain Name System)**

DNS query와 response를 묶어주는 Transaction ID는 0x3ed6이고, Flag를 통해 해당 패킷이 query임을 알 수 있다. 또한 query의 response가 149번임을 알려준다. Queries에 IP주소를 알고 싶은 URL의 정식 호스트 네임이 저장되어있다.

1. [DNS] Local DNS가 URL의 IP주소를 가지고 있었기 때문에 Root DNS에게 정보를 요청하지 않고 Client 1에게 IP주소를 전달한다.



<그림 4> Local DNS에서 Client 1에게 DNS query에 대한 response를 보내는 것을 보여주는 149번 패킷

* **149번 패킷 상세 분석**

**IPv4**

query와 반대로 source IP는 Local DNS의 IP, destination IP는 Client 1의 IP주소임을 알 수 있다. TTL은 58, Header checksum은 0x173b이다.

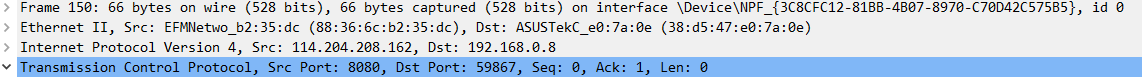
**UDP**

IP주소와 마찬가지로 source port는 Local DNS의 지정 port인 53, destination port는 Client 1의 63077이다. UDP checksum은 0x5a5c임을 알 수 있다.

**DNS**

Transaction ID가 0x3ed6으로 query의 Transaction ID와 일치하고, Flag를 통해 해당 패킷이 query response임을 알 수 있다.

1. 서버가 Client 1의 SYN을 받고 연결 허용을 위해 Client 1에게 SYN+ACK 패킷을 전송한다.



<그림 5> Server에서 Client 1로 SYN+ACK 패킷을 보내는 것을 보여준다.

* **150번 패킷 상세 분석**

**Ethernet 2 Frame header**

source의 mac 주소는 Server의 mac주소이고, destination의 mac 주소는 Client 1의 mac주소임을 알 수 있다.

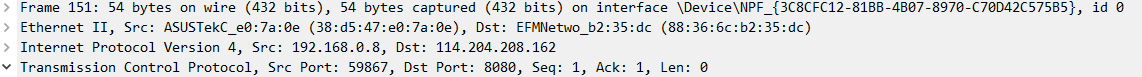
**IPv4**

IP주소 또한 Source IP는 Server의 IP, destination IP는 Client 1의 IP이다. Header checksum이 0x43a5임을 알 수 있다.

**TCP**

Source의 port는 Server의 port, destination의 port는 Client의 port인 59867이다. Flag를 통해 해당 패킷이 SYN, ACK임을 알 수 있다. Sequence number는 0이고, 다음 Sequence number가 1로 지정되어있다. TCP checksum은 0xfca9이다.

1. 서버가 보낸 SYN+ACK을 받은 Client 1이 연결 허용을 위해 서버에게 ACK를 보내어 3way-handshaking을 마치고 Client 1과 Server의 TCP연결이 성립된다.



<그림 6> Client 1이 Server로 ACK를 보내는 것을 보녀주는 151번 패킷

* **151번 패킷 상세 분석**

**Ethernet 2 Frame header**

source의 mac 주소는 Client 1의 mac 주소이고, destination의 mac 주소는 Server의 mac 주소이다.

**IPv4 (Internet protocol)**

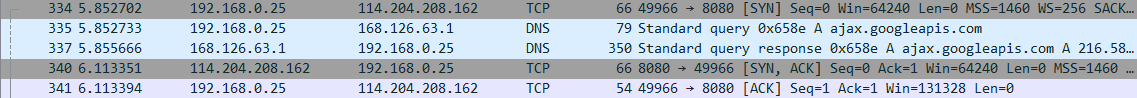
source의 IP 주소는 Client의 IP 주소이고, destination의 IP주소는 Server의 IP 주소이다. Header checksum이 0x5589임을 알 수 있다.

**TCP(Transmission Control Protocol)**

Source의 port는 Client 1의 port인 59867, destination의 port는 Server의 port인 8080이다. Flag를 통해 해당 패킷이 ACK이고, TCP checksum은 0x366c임을 확인할 수 있다. Sequence number가 1이므로 150번 패킷에 대한 응답임을 알 수 있다.

**Client 2**

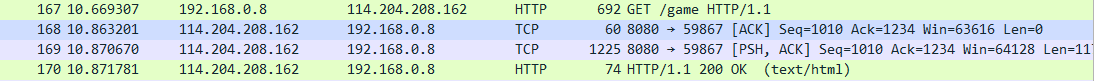
* 패킷 캡쳐를 통해 Client 2 또한 Client 1과 마찬가지로 DNS를 통해 IP주소를 알아내고, 3way-handshaking을 통해 TCP연결이 성립하는 것을 확인할 수 있다.

****

<그림 7> Client 2와 서버의 3way-handshaking을 통한 연결을 보여준다.

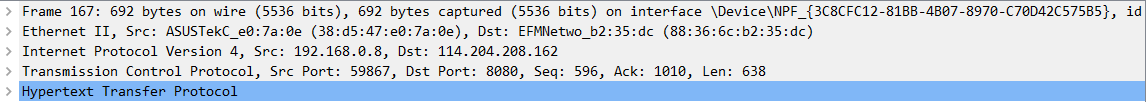
**게임 입장**

**Client 1**



<그림 1> HTTP 프로토콜을 이용한 게임 접속 (Client 1)

**167번 패킷 :**



<그림 2> HTTP request를 통해 게임 URI를 요청하는 167번 패킷

* **167번 패킷 상세 분석**

**Frame** : 해당 패킷의 Frame number가 167번이고, Frame length는 74byte, interface id는 0이다. 패킷의 Encapsulation type이 Ethernet임을 알 수 있다.

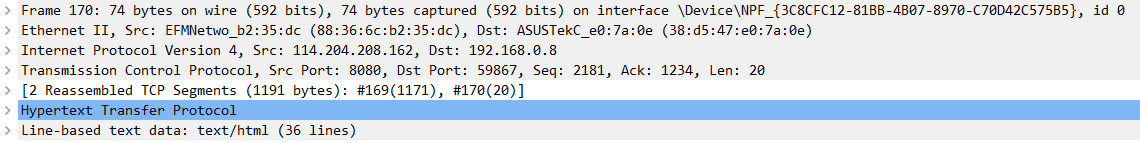
**Ethernet 2** : source(Client 1)의 mac 주소는 (38:d5:47:e0:7a:0e)이고, destination(Server)의 mac 주소는 (88:36:6c:b2:35:dc)이다. Type은 IPv4이다.

**IPv4 (Internet protocol)** : source(Client 1)의 IP주소는 192.168.0.8, destination(Server)의 IP주소는 114.204.208.162이다. Fragment Flag를 확인해보면 Fragment 되지 않았음을 알 수 있고, Header checksum이 0x5307, TTL은 128임을 알 수 있다.

**TCP(Transmission Control Protocol)** : Source(Client 1)의 port는 59867, destination(Server)의 port는 8080이다. Flag를 통해 해당 패킷이 PSH, ACK이고, TCP checksum은 0x1a8b임을 확인할 수 있다. 또한 Window size value는 509이다.

**HTTP(Hypertext Transfer Protocol) :** message format을 확인해보면, http request 형식임을 알 수 있다. 또한 GET method를 이용하였으며, URI로 /game을 호출하였고 HTTP 버전은 1. 1을 사용함을 알 수 있다. 또한 Connection이 keep-Alive 상태인 것이 확인되고, 따라서 tcp 연결 설정이 계속 유지되고 있음을 알 수 있다.

**170번 패킷 :**



<그림 3> 167번 패킷에 응답하는 HTTP response, 170번 패킷

* **170번 패킷 상세 분석**

**Frame**

해당 패킷의 Frame number가 170번이고, Frame length는 74byte, interface id는 0이다. 패킷의 Encapsulation type이 Ethernet임을 알 수 있다.

**Ethernet 2**

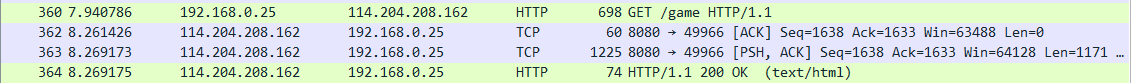
source(Client 1)의 mac 주소는 (88:36:6c:b2:35:dc)이고, destination(Server)의 mac 주소는 (38:d5:47:e0:7a:0e)이다. Type은 IPv4이다.

**IPv4 (Internet protocol)**

source(Client 1)의 IP주소는 114.204.208.162, destination(Server)의 IP주소는 192.168.0.8이다. Fragment Flag를 확인해보면 Fragment 되지 않았음을 알 수 있고, Header checksum이 0x23bf, TTL은 52임을 알 수 있다.

* **TCP(Transmission Control Protocol)** : Source(Client 1)의 port는 8080, destination(Server)의 port는 59867이다. Flag를 통해 해당 패킷이 PSH, ACK이고, TCP checksum은 0xdc66임을 확인할 수 있다. 또한 Window size value는 501이다.
* **HTTP(Hypertext Transfer Protocol) :** message format을 확인해보면, http response 형식임을 알 수 있다. 또한 200 OK 응답이 온 것으로 보아 GET 요청이 성공적으로 받아들여졌음을 알 수 있다. Content를 확인해보면 webpage를 구성하는 html형식의 실제 데이터를 볼 수 있다.

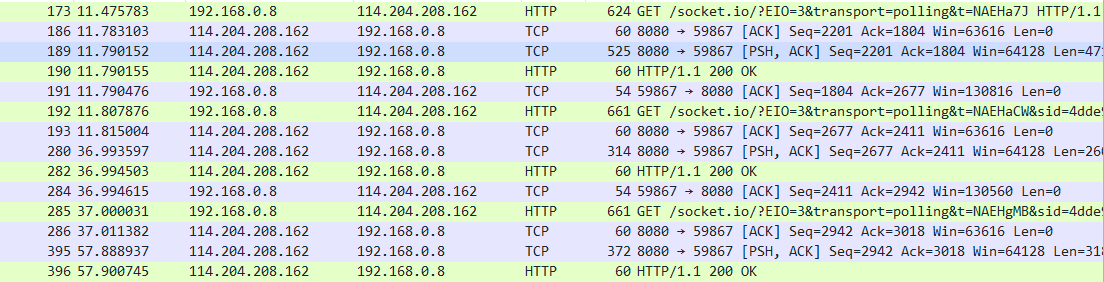
**Client 2**



<그림 4> HTTP 프로토콜을 이용한 게임 접속 (Client 2)

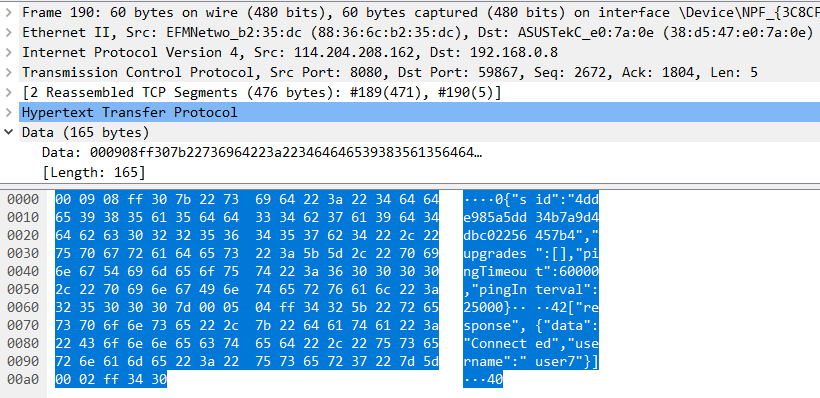
그림 4를 보면 클라이언트 1의 접속과정과 동일하게 HTTP 프로토콜이 동작하고 있음을 확인할 수 있다.

**3. 게임 진행**



<그림 5> HTTP 프로토콜을 이용한 게임 진행

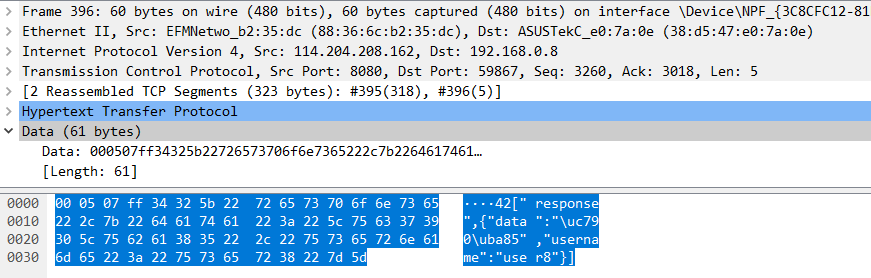
게임 진행에서 캡처된 패킷도 게임 입장 부분의 패킷 분석과 같이 link layer, network layer, transport layer, application layer(http 패킷)를 확인할 수 있으며, mac, ip, port, data내용을 알 수 있고, checksum, ttl, window size 등을 확인 할 수 있다.



<그림 6> HTTP 프로토콜을 이용한 게임 진행 - “user 7: connected”

게임에 접속하면 화면에 “user 7: connected” 문구가 나온다. (client 2의 화면에서는 “user 8: connected”라고 나온다.) 위 그림은 173번 request의 response인 190번 패킷이다. Data 부분을 확인하면 아래쪽에 user7 : connected 문구를 확인할 수 있다.

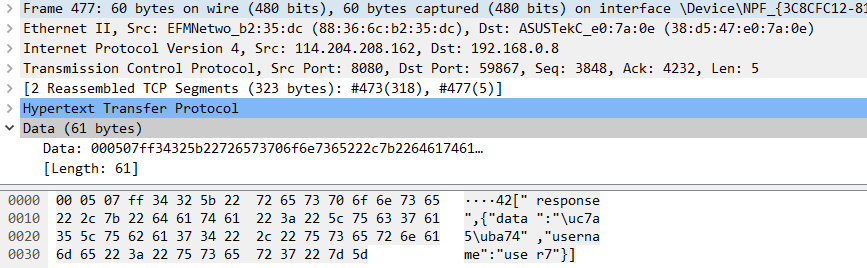
IPv4 Checksum: 0x1221, TCP Checksum: 0xe0d6, TTL: 52



<그림 7> HTTP 프로토콜을 이용한 게임 진행 – “user 8 : 단어”

User8(클라이언트 2)이 먼저 “ㅈㅁ”의 초성에 맞는 “자명”을 입력하였고 웹페이지에 “user 8 : 자명” 문구가 추가되었다. Data의 아랫부분에서 해당 형식을 확인할 수 있다.

IPv4 Checksum: 0x4000, TCP Checksum: 0xd9cc, TTL: 52

<그림 7> HTTP 프로토콜을 이용한 게임 진행 – “user 7 : 단어

User7(클라이언트 1)이 “장면”을 입력하여 웹페이지에 “user 7 : 장면” 문구가 추가되었다.

Data의 아랫부분에서 해당 형식을 확인할 수 있다.

IPv4 Checksum: 0x4000, TCP Checksum: 0xd2c2, TTL: 52

이런 식으로 모든 플레이어가 탈락될 때까지 게임이 진행된다.