**컴퓨터 네트워크 2조**

**팀프로젝트 – 묵찌빠 게임 및 패킷 분석**

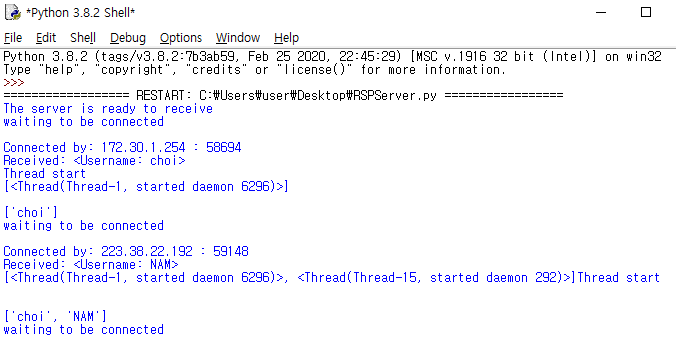
****

|  |  |
| --- | --- |
| **학번** | **이름** |
| **2018125080** | **김필재** |
| **2018125083** | **변재환** |
| **2018125084** | **임예랑** |
| **2019125013** | **김연주** |
| **2019125021** | **남서아** |
| **2019125073** | **최환효** |

**게임 방법 및 규칙**

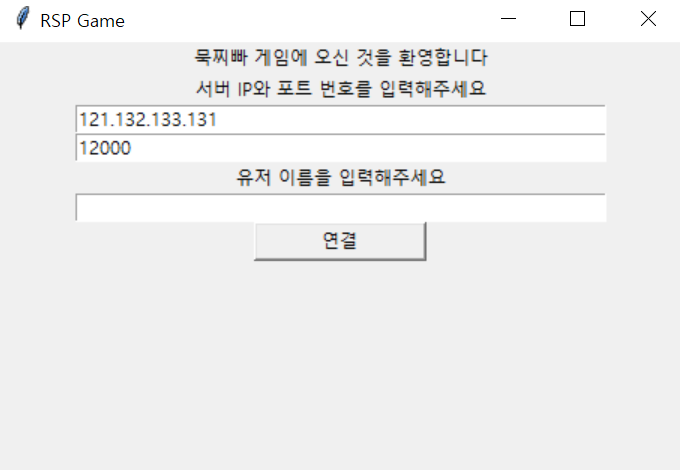
**1) 게임 방법**

1. 게임 시작을 위해서 서버를 연다.

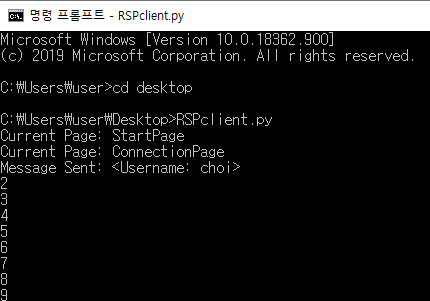


2. 서버가 열리면 클라이언트를 실행하여 GUI 창을 띄운다.

3. GUI 창이 나오면 서버의 IP와 접속할 포트번호 그리고 유저 이름을 입력하고 연결을 누른다.

.

4. 연결 중일 때 클라이언트 창에서 시간이 얼마나 흐르는지를 초단위로 표시한다.



5. 게임 접속에 성공하면 다음과 같은 GUI로 넘어가며 여기서 자신이 무엇을 낼지를 정하고, 본격적인 묵찌빠 게임을 시작한다.



6. 묵찌빠 게임이 끝나면 최종 결과가 나타나며 게임은 종료 된다.

**2) 게임 규칙**

기본적으로 5판 3선 승제로 1판을 round로 칭한다.

게임이 시작되면 일정 시간 이내로 자신의 패(가위, 바위, 보)를 선택하여 버튼을 눌러야 한다.

만일 시간 이내로 버튼을 누르지 않으면 time out으로 기권 처리된다.

각 round에는 step 1과 step 2가 있다.

step 1은 선후공 결정, step 2는 round의 승패를 결정한다.

step 1, step 2모두 가위바위보를 통하여 공격자를 정한다.

step 1은 선후공을 결정하며 여기서 공격자가 누구인지 결정되고, 공격자가 결정이 될 때까지

가위바위보를 시행한다.

step 1에서 선공자(공격자)가 정해지면, step 2로 넘어간다.

step 2는 round의 승패를 결정합니다. 서로 같은 패가 나올 때까지 계속해서 가위바위보를 하고,

가위바위보의 결과에 따라 공격자가 수시로 바뀌게 된다.

ex) A가 공격자일 때, B가 가위바위보를 하여 A를 이기면, 공격자는 A에서 B로 바뀌게 된다.

ex) A가 공격자일 때, A와 B 모두 같은 패를 내었을 경우(가위-가위, 바위-바위, 보-보), round의 승자는 A가 되며 각 클라이언트에 자신과 상대편의 round에서 승리한 숫자가 표시된다.

round를 최소 3번 최대 5번 반복하여 먼저 round를 3번 이긴 사람이 최종적으로 승리하게 된다.

**묵찌빠 게임 실행 환경**

서버, 클라이언트 1: 최환효, 클라이언트 2: 남서아

서버는 Python IDLE의 shell에서 수행, 클라이언트는 cmd창에서 수행하였다.

서버와 클라이언트 1의 IPv4는 172.30.1.28, 기본 게이트웨이는 172.30.1.254, 외부 IP는121.132.133.131이고, 클라이언트 2의 IP는 192.168.43.228이다.

사용한 포트 번호는 12000을 사용하였으며, 프로토콜은 TCP를 사용했다.

**묵찌빠 게임 – 플레이어 모집**

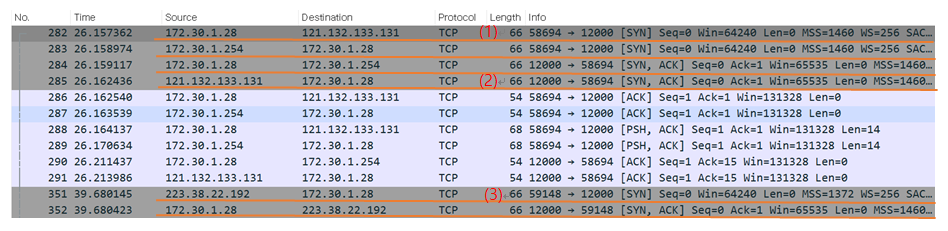
서버 오픈 – 클라이언트 2개 연결 – 게임실행 순으로 시행된다.

1. 서버 오픈

서버코드에 HOST와 PORT에 각각 서버의 IPv4와, 사용할 포트번호를 입력하고 실행 시킨다.

2. 클라이언트 2개 연결

서버가 정상적으로 실행되었다면, 클라이언트를 연결한다.



서버 – 클라이언트 연결

1) No. 282 : 서버의 내부에서 외부로 패킷을 보내기 위해 내부(ip : 172.30.1.28)에서 외부(ip : 121.132.133.131)로 SYN 보냄

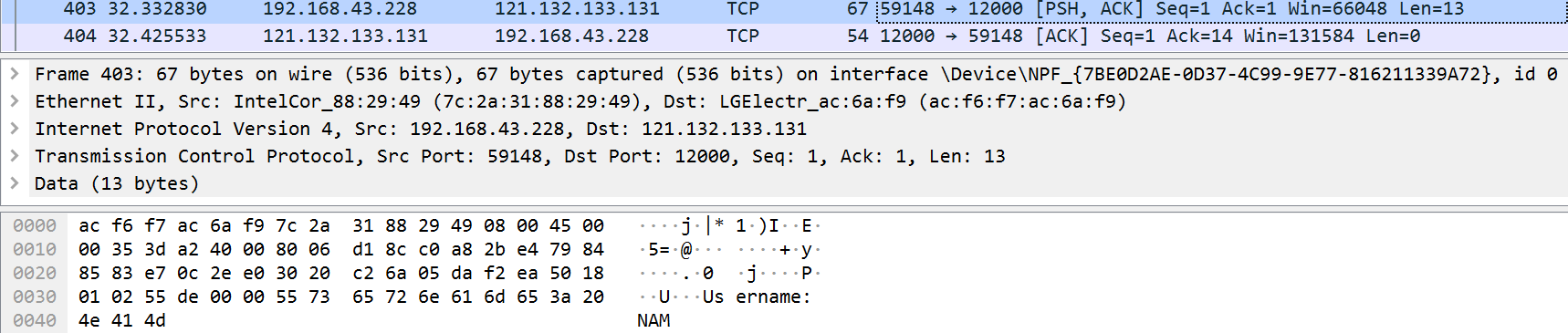
-> No. 285 : 외부에서도 내부로 이에 대한 SYN과 ACK보냄.

2) No. 283에서 클라이언트 1(ip : 172.30.1.254)에서 서버로 SYN보냄. Sequence num은 0으로 설정하며 TCP에서 세션 성립.

-> No. 284에서 서버(ip : 172.30.1.28)에서 클라이언트 1으로 SYN,ACK보냄.

3) 클라이언트 2(ip : 223.38.22.192)에서 서버로 SYN 보냄. 여기서도 마찬가지로 Sequence num = 0으로 설정하며 TCP에서 세션 성립.

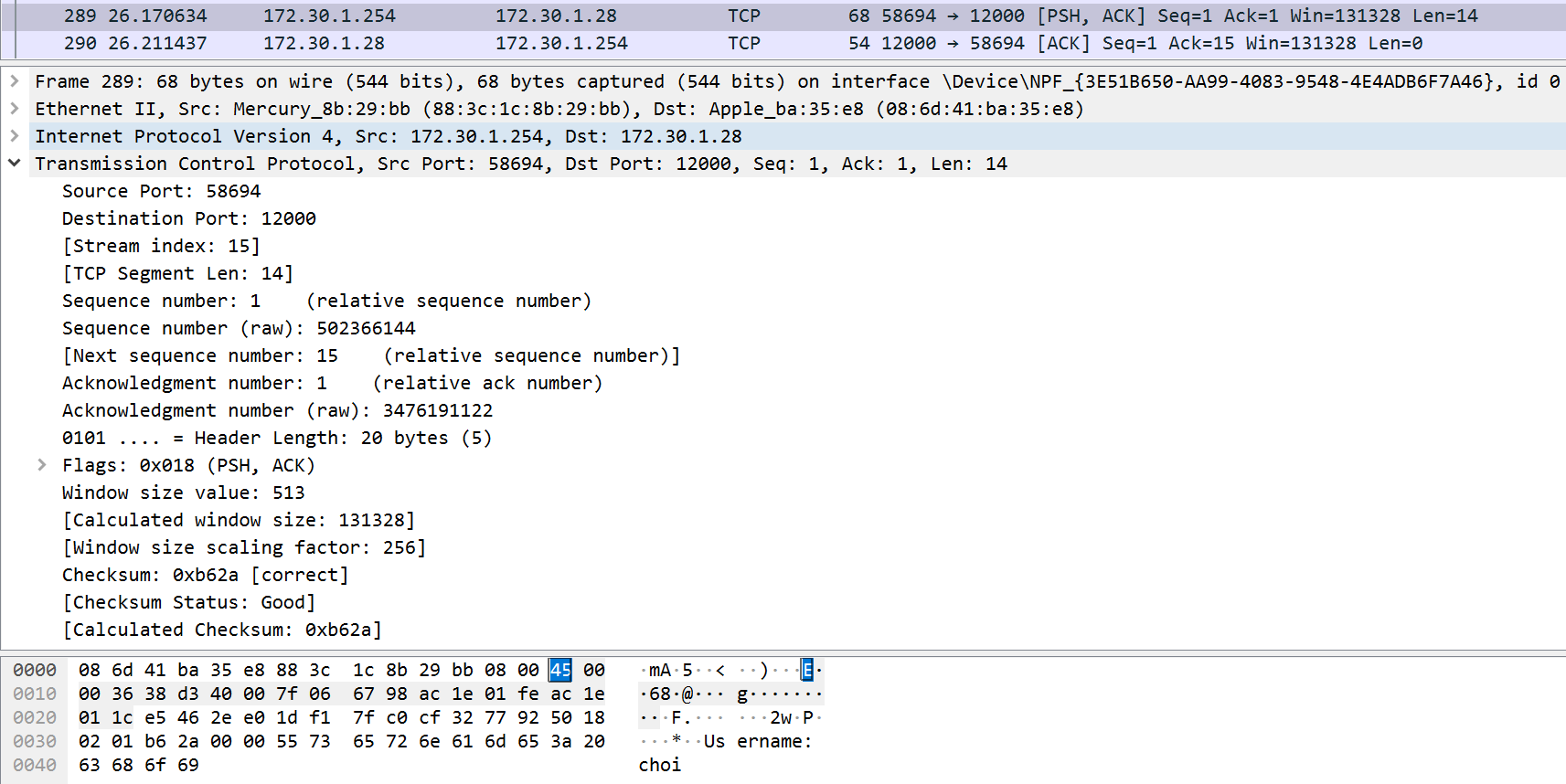
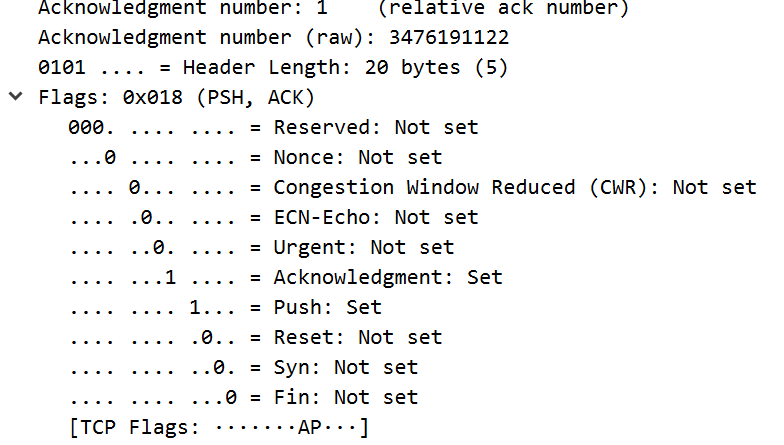
**묵찌빠 게임 – 유저 이름 설정**



위의 그림은 남서아(클라이언트2)의 패킷을 캡쳐한 것이다.

No. 403 : 각 클라이언트에게 이름을 입력하라는 메시지를 받았고, 위 패킷은 그에 대한 클라이언트의 메시지이다. 위에서는 서아가 username을 NAM으로 받았기 때문에 0040에 NAM으로 표기되었다. 여기서 sequence number가 1이고 데이터의 길이가 13이기 때문에 이 메시지가 전송되고 나서 받는 ack는 Seq=1 과 Len=13을 더한 14여야 한다.

따라서 no 404를 보았을 때, 서버에서 클라이언트에 보내주는 ack가 14라는 것을 확인할 수 있다. 또한 서아의 wndsize가 66048이며, 환효 서버의 wndsize는 131584라는 것을 알 수 있다.



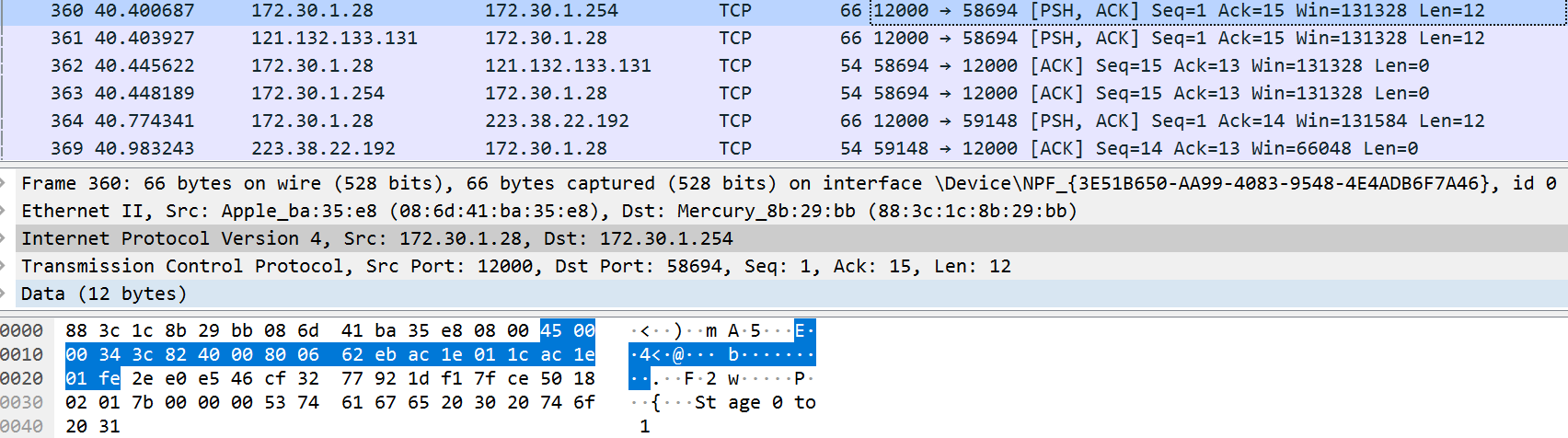
다음은 환효의 서버에서 캡쳐한 패킷이다.

No. 289에서 172.30.1.254의 ip를 가진 환효의 클라이언트가 172.30.1.28의 ip를 가진 서버에 push를 하고 있는 것을 확인할 수 있다. 위에서 서아의 클라이언트가 했던 것과 마찬가지로 username을 서버에 전달하고 있는 모습이다. Sequence number가 1이고, len이 14이니 이 전송된 데이터에 대한 ack의 값은 1+14인 15라는 것을 추론할 수 있다. 또한 Ipv4 버전을 사용하고 있으며, Ipv4이기에 checksum을 이용해 error를 detective 해볼 수 있다. 실제 checksum의 값이 0xb62a이고 계산된 checksum의 값 또한 0xb62a이므로 데이터가 유실 없이 잘 전송되었다. 또한 위에서는 그저 ip address를 알 수 있었지만 밑의 자료를 보면 mac주소 또한 알 수 있다. 환효 클라이언트의 mac 주소는 88:3c:1c:8b:29:bb이며 환효 서버의 mac주소는 08:6d:41:ba:35:38이다. 또한 flag는 push와 ack를 포함하기 때문에 0x018이 된다. 또한 그 밑의 사진은 289번 패킷의 세부내용을 캡쳐한 것이다. ACK. PSH FLAG가 SET이 된 상태이며 보낸 문자열이 CHOI이니 data가 4이다.

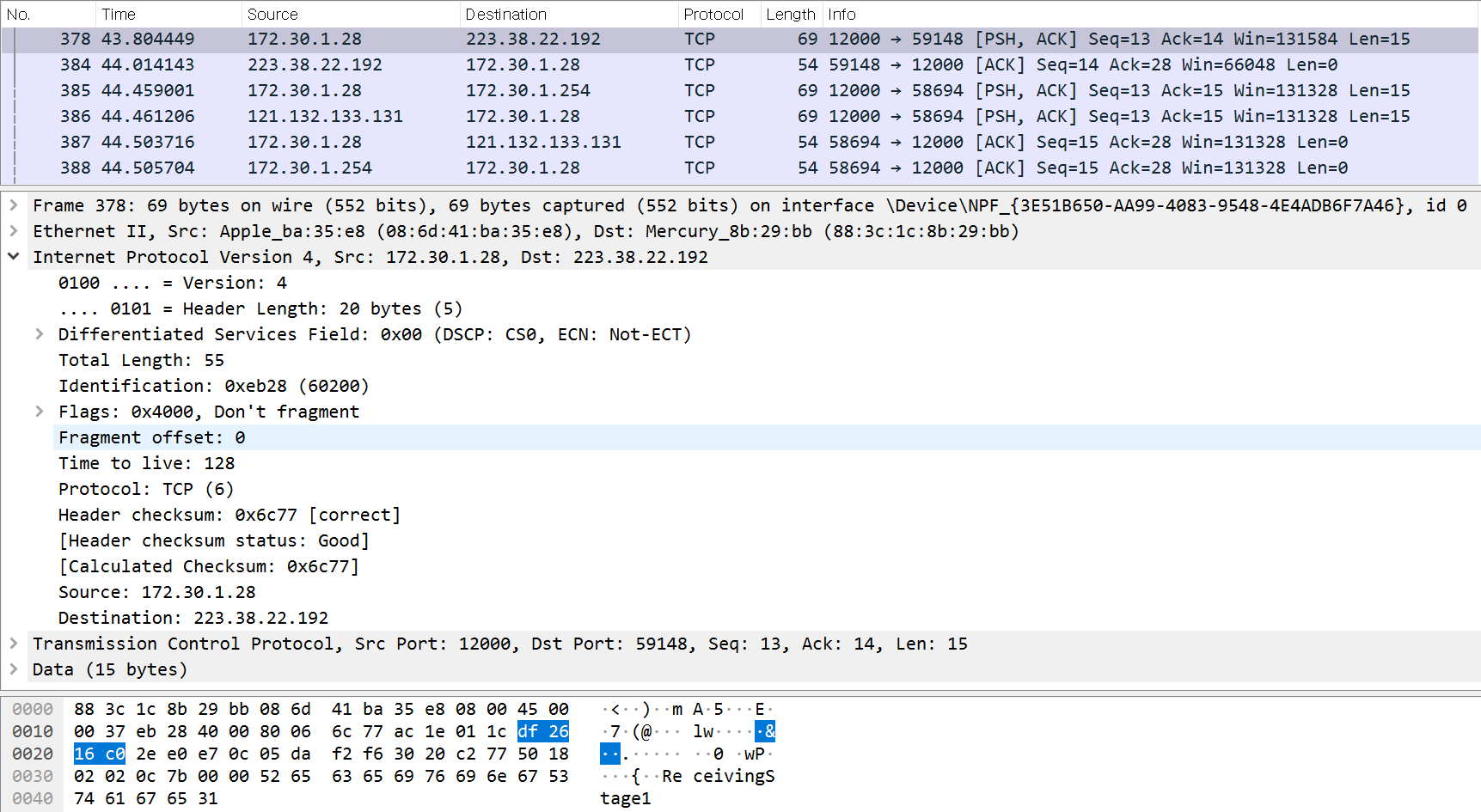
No. 290에서 ack의 값을 확인할 수 있다. Ack의 값이 15이므로 우리가 앞에서 한 추론은 맞다는 것을 확인해 볼 수 있었다.

이제 2대의 클라이언트가 무사히 서버에 접속했고, username을 알려주었으니 서버에서 게임을 시작할 준비가 되었다는 것을 알리고 게임을 시작해야 한다.

**묵찌빠 게임 – 서버에서 게임 시작을 알림 round 1**



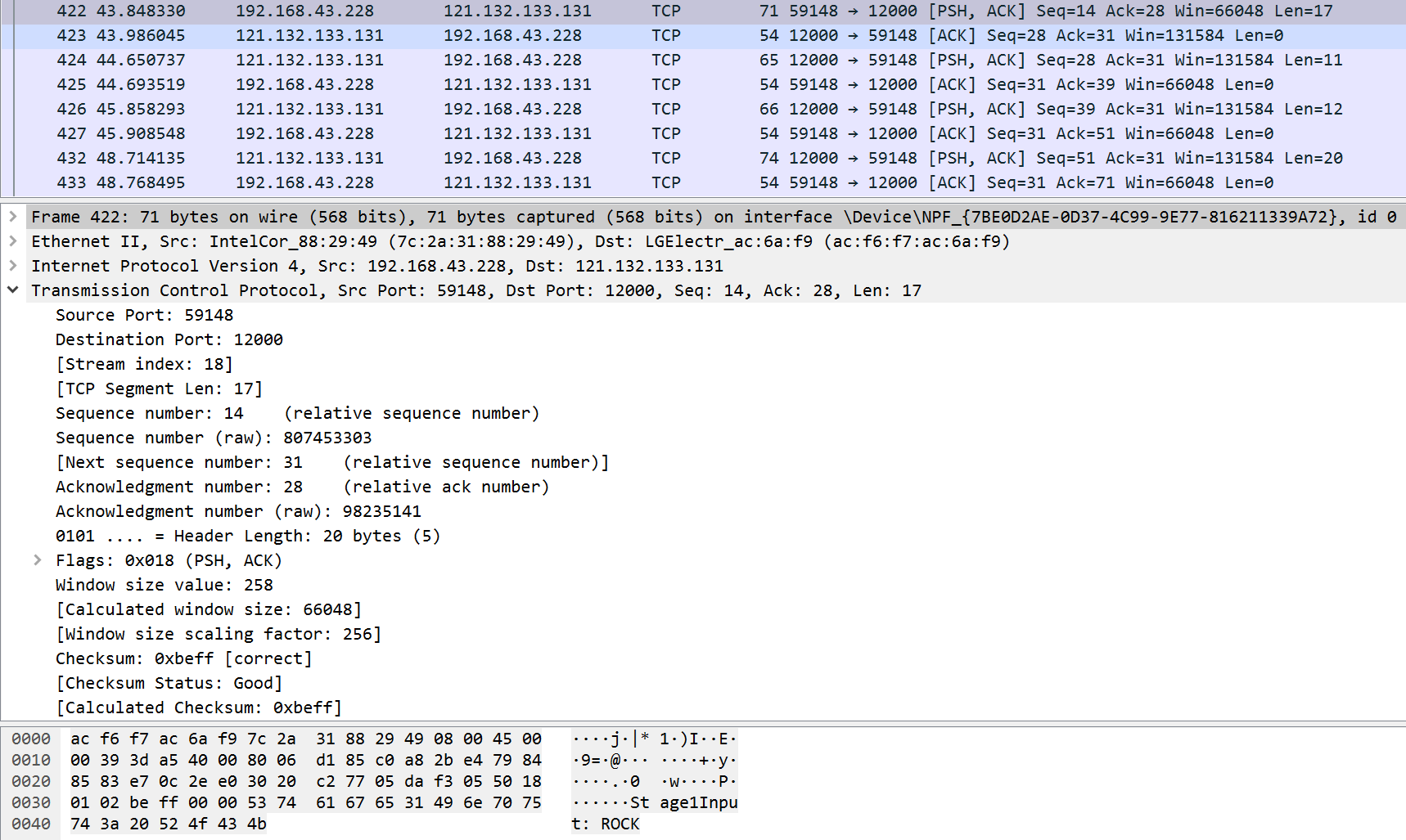
위의 캡쳐본을 보면, 서버의 내부 아이피에서 클라이언트1인 환효의 내부 아이피에 push한 것을 볼 수 있고, 마찬가지로 클라이언트 2인 서아의 내부 아이피에도 push를 했다. 메시지의 내용은 stage0 to 1이며, stage0는 유저 2명이 접속한 후 유저 네임을 입력하는 것이었으므로 이 단계가 끝나 실제 공수를 정하는 stage1을 시행하는 것이다.



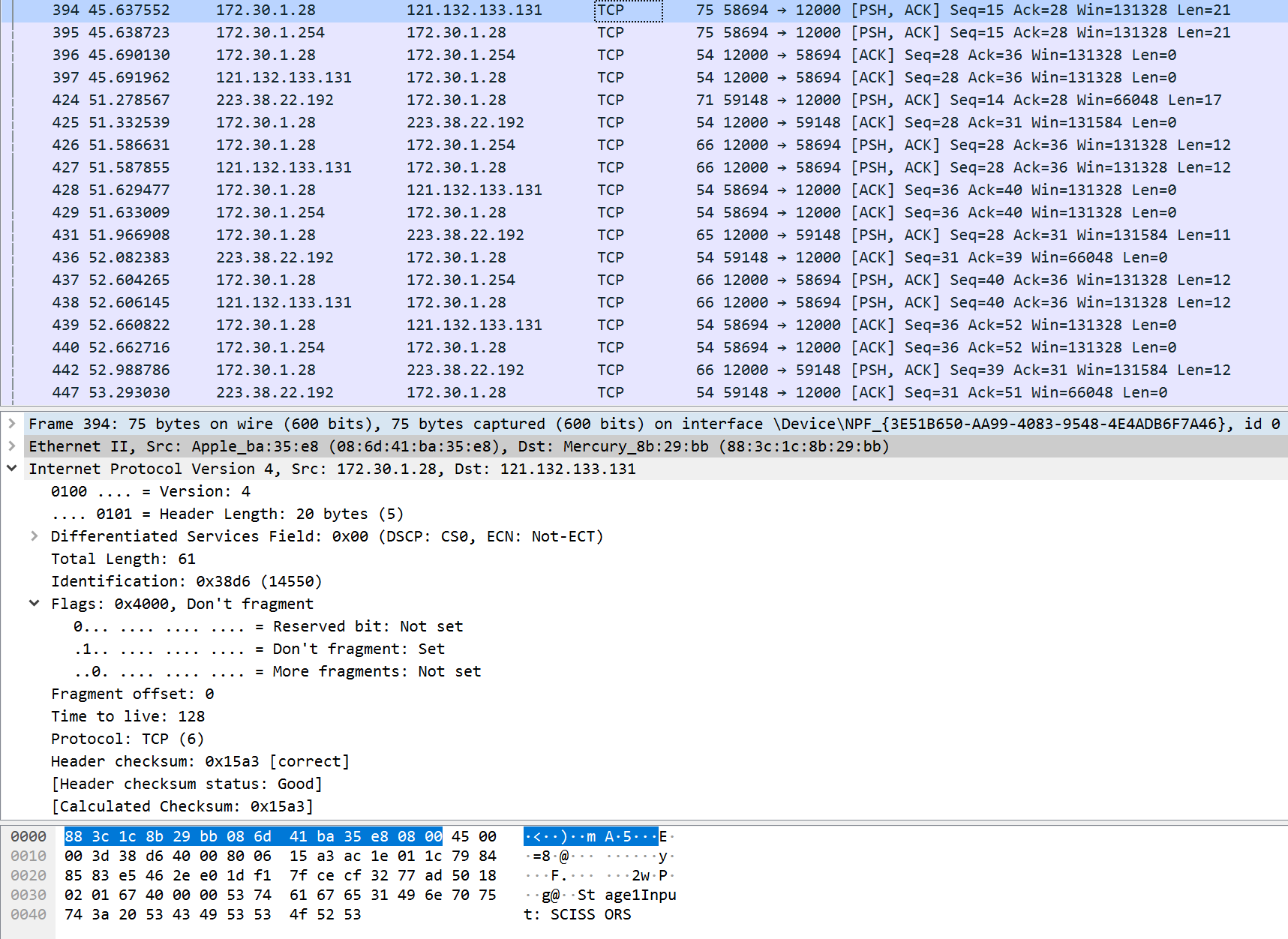
위의 캡쳐본은 서버가 클라이언트들에게 stage1을 받을 준비가 되었다고 메시지를 보내는 것이다. 각 push마다 그에 따른 ack가 오고, 그 ack값은 push에서의 sequence + len이다. No. 378부터 no. 388까지는 push와 그에 따른 ack를 의미한다. No.378은 환효 서버의 내부 ip에서 외부 ip로 메시지를 전송하는 것이며, no.385와 no.386은 각각 클라이언트1과 클라이언트 2에게 receiving stage1이라는 메시지를 보내는 패킷이다. 각 전송마다 그에 따른 ack 패킷이 온다. 그림의 checksum값과 calculated checksum값이 같으므로 loss가 일어나지 않았다.

**묵찌빠 게임 – 가위바위보 공수 결정 step 1**

이제는 묵찌빠의 공수를 정하기 위해 가위바위보를 할 차례이다. 코드에서는 이를 stage1이라 일컬으며, 클라이언트에서 보낸 패를 서버가 받고 이에 따른 승부의 결과를 다시 클라이언트들에게 보낸다. 우리는 각각의 서버와 클라이언트의 패킷을 분석해서 어떤 number의 패킷이 어떻게 동작하는지를 알기 위해 패킷 2개를 비교할 것이다.



위 사진은 클라이언트 2(남서아)의 패킷을 캡쳐한 것이다.



위 사진은 클라이언트 1과 서버(최환효)의 패킷을 캡쳐한 것이다.

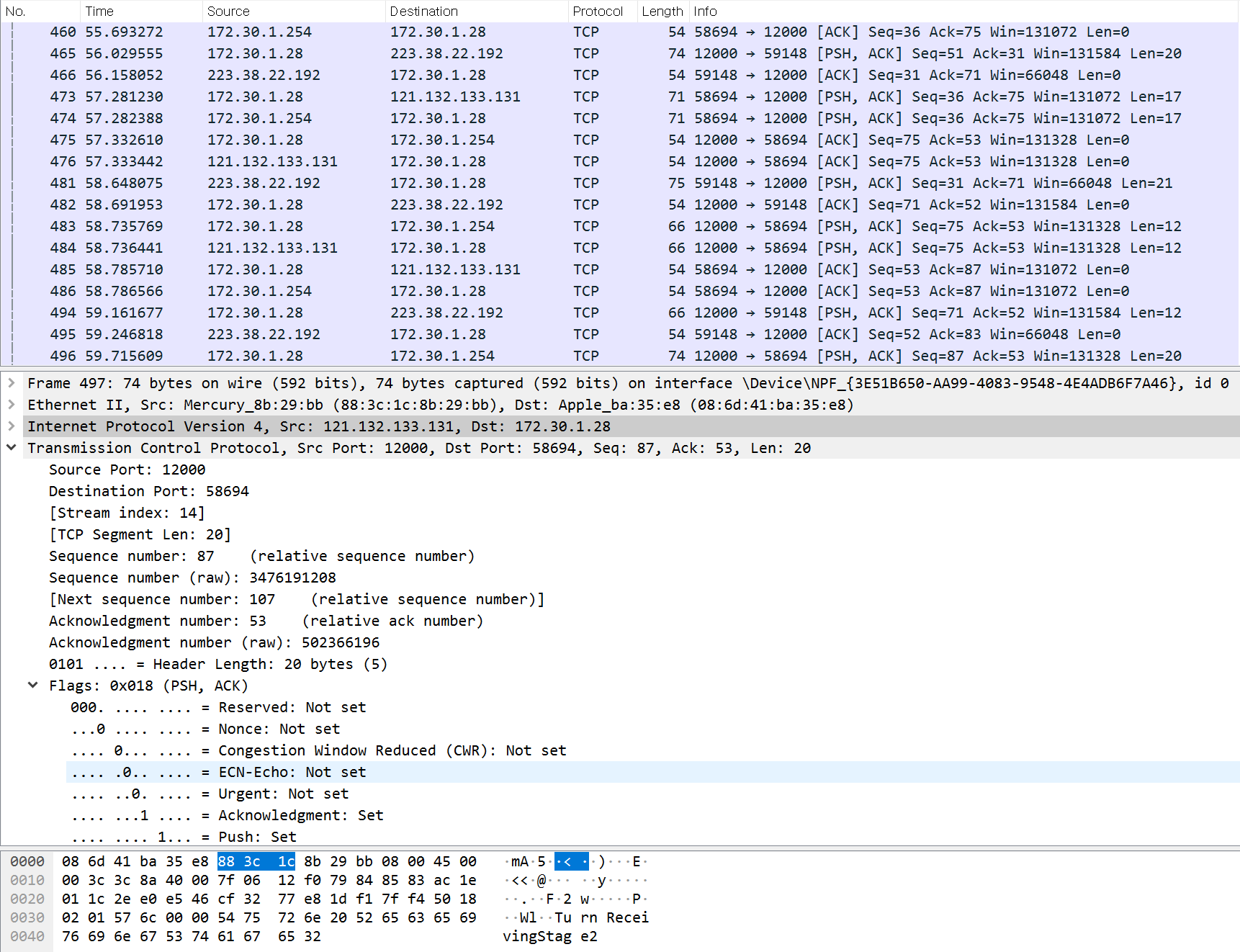
환효는 cmd창에서 서버와 클라이언트 각각을 켜서 게임을 실행한 것이기 때문에 와이어샤크 캡쳐본에서는 서버와 클라이언트1의 패킷이 같이 분석된다.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 패킷 number | 패킷 | 내용 |
| 서버&클라이언트 1 | 394 | 내부 ip -> 외부 ip  가위 냈다고 전송 | Checksum = 0x15a3  TTL = 128  Checksum correct |
| 서버&클라이언트 1 | 395 | 클라이언트 1 -> 내부 ip  가위 냈다고 전송 | Checksum = 0x678e  TTL = 127  Checksum correct |
| 서버&클라이언트 1 | 424 | 클라이언트 2-> 서버  바위 냈다고 전송 | Checksum = 0xa897  TTL = 113  Checksum correct |
| 클라이언트 2 | 422 | Checksum = 0xd185  TTL = 128  Checksum correct |
| 서버&클라이언트 1 | 426 | 내부 ip -> 클라이언트 1  Lose 전송 | Checksum = 0x62e8  TTL = 128  Checksum correct |
| 서버&클라이언트 1 | 427 | 외부 ip -> 내무 ip  Lose 전송 | Checksum = 0x12fd  TTL = 127  Checksum correct |
| 서버&클라이언트 1 | 431 | 서버 -> 클라이언트 2  Win 전송 | Checksum = 0x6c79  TTL = 128  Checksum correct |
| 클라이언트 2 | 424 | Checksum = 0x1d31  TTL = 114  Checksum correct |
| 서버&클라이언트 1 | 437 | 내부 ip -> 클라이언트 1  Stage 1 to 2 | Checksum = 0x62e7  TTL = 128  Checksum correct |
| 서버&클라이언트 1 | 438 | 외부 ip -> 내부 ip | Checksum = 0x12fc  TTL = 127  Checksum correct |
| 서버&클라이언트 1 | 442 | 서버 -> 클라이언트 2 | Checksum = 0x6c77  TTL = 128  Checksum correct |
| 클라이언트 2 | 426 | Checksum = 0x1d30  TTL = 114  Checksum correct |

이처럼 승패가 정해진 후에는 각각의 클라이언트에 승패를 알리고 stage 1에서 stage 2로 넘어가라는 메시지를 전달한다. 이 표를 통해 알 수 있는 점은, 서버에서 클라이언트 1으로 전송할 때(즉, 같은 컴퓨터에서 돌아가는 서버에서 클라이언트로 메시지 전송할 때)는 내부 ip에서 클라이언트1으로 먼저 전송하고, 그 후에 외부 ip로 전송한다는 것을 알 수 있었다. 또한 서버에서 클라이언트 2로 전송할 때(즉, 서버에서 다른 컴퓨터에 있는 클라이언트로 메시지 전송할 때)는 외부 ip에서 클라이언트 2로 전송한다는 것을 관찰할 수 있었다. 이를 통해 서버와 클라이언트 2에서 캡쳐했을 때, 왜 ip주소가 다르게 나타나는지를 확인해 볼 수 있었다. 외부 네트워크로 나갈 수 있는 통로가 되어주는 기기의 주소가 외부 ip인데, 여기서는 와이파이의 ip이다. 그러므로 인터넷망에 접속해서 통신하기 때문에 이렇게 ip가 달라지는 것이다. 여기서는 PUSH만 분석하고 ACK는 생략했다. 이제 묵찌빠를 하는 stage2로 넘어가 보자.

**묵찌빠 게임 – 묵찌빠 게임 시작 step 2**

묵찌빠 게임을 시작하는 데 앞서, 만약 공수가 결정된 상태에서 똑 같은 패를 내지 않고 다른 패를 냈다고 가정해 보자. 그렇다면 승부는 나지 않고 공수가 바뀔 것이다. 우리는 이 경우를 draw라고 일컬으며 이 draw상태일 때는 서버가 누가 공인지 수인지 알려주지 않고 그저 draw라는 결과만 각 클라이언트로 전송하는 것으로 했다. 그 결과는 서버에 currentplayer로 따로 저장되어 있다. 또한 draw가 너무 많이 반복되는 관계로 각각의 checksum 값이나 TTL값은 생략하고 클라이언트 창을 비교해 2명의 클라이언트가 무엇을 패로 냈는지만 분석하도록 하겠다.

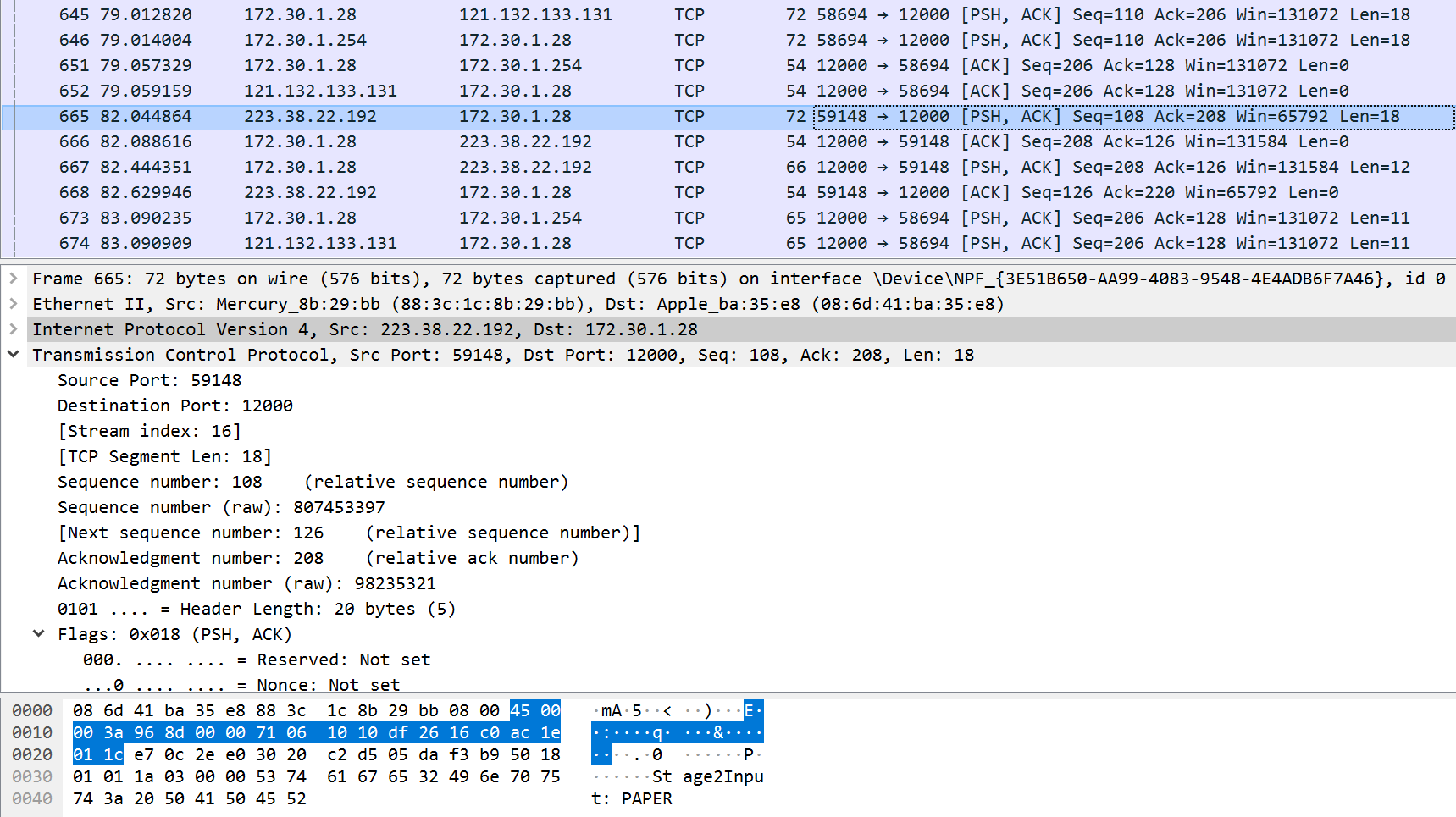


위의 사진은 서버에서의 캡쳐 내용이다.

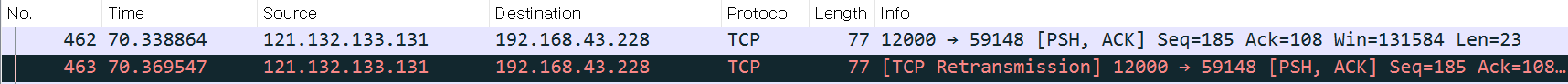
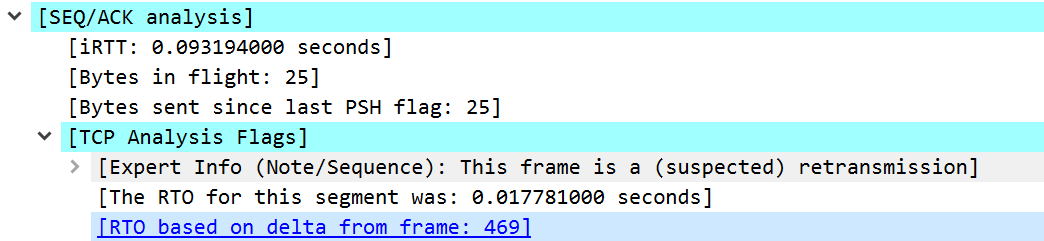
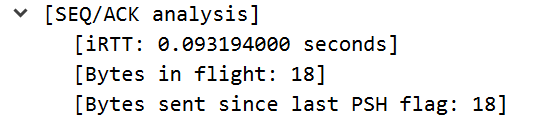
No. 465에서는 turn receiving stage2라는 메시지를 서버에 보내고 있으며, no.473, no.474, no.481은 각각 서버의 내부 ip가 외부 ip에게, 클라이언트1이 서버의 외부 ip에 rock을 냈다는 것과 클라이언트 2에서 서버의 내부 ip에 scissors를 냈다는 것을 의미한다. 그러나 앞전의 가위바위보에서는 클라이언트 2(남서아)가 공격권을 쥐고 있었으나 이번 판에 결판이 나지 않았고, 클라이언트1(최환효)가 클라이언트2(남서아)를 묵을 내서 이겼으므로 공격권은 클라이언트1이 쥐게 된다. 그러나 결판이 나지 않았으므로 no.483, 484, 494에서 보는 것처럼 draw를 보낸다. 이후로 굉장히 많은 draw가 있었으므로 각각의 패킷을 캡쳐하는 것은 생략하고 각 시도마다 클라이언트가 어떤 패를 냈는지 명시하도록 하겠다. 또한 공격권을 가진 사람은 붉은 색으로 표시한다.

|  |  |  |  |
| --- | --- | --- | --- |
| 시도횟수 | 클라이언트 1(환효) | 클라이언트 2(서아) | 승패결과 |
| 1 | 바위 | 가위 | Draw |
| 2 | 가위 | 보 | Draw |
| 3 | 보 | 가위 | Draw |
| 4 | 보 | 바위 | Draw |

이 다음번에 승부가 난다. 4번째 시도에서 환효가 공격권을 쥐고 있는 상태이다.



No.645 646, 665를 보면 클라이언트 1과 2 모두 보를 낸 상태이다. 바로 직전의 시도에서 클라이언트1(환효)가 공격권을 쥔 상태에서 같은 패를 냈으므로 round1의 승자는 클라이언트 1이다. 서버는 이 결과를 각 클라이언트에 전송하는데, 클라이언트 1에는 win을, 클라이언트2에는 lose를 전송한다. 이 모든 push에서 checksum과 calculated checksum은 같아 loss가 일어나지 않은 것을 확인할 수 있었다. 그렇지만 이 과정에서 TCP retransmission이 2번 일어났다는 것이 와이어샤크에 캡쳐되었다.

**※ TCP Retransmission**

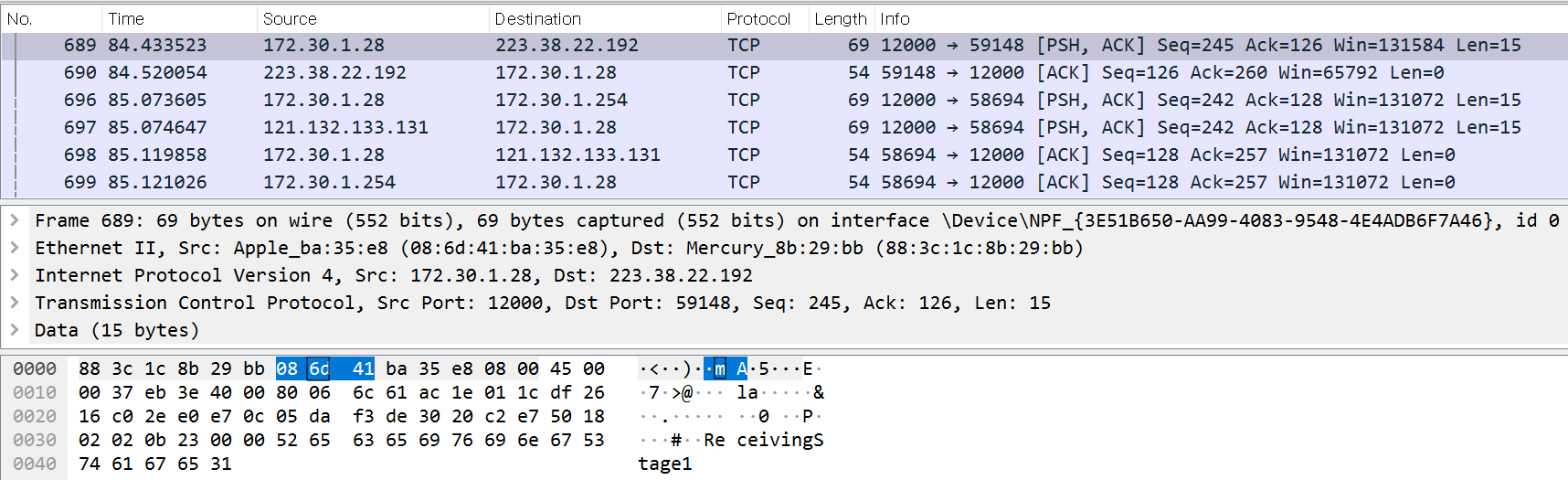
위 사진은 TCP Retransmission이 일어난 패킷의 정보이다.

그 아래의 사진은 TCP Retransmission이 일어나지 않은 일반적인 패킷의 정보이다

-> TCP Retransmission이 일어난 패킷에는 TCP Analysis Flags에 대한 정보가 있는 반면, 재전송이 일어나지 않은 패킷에는 TCP Analysis Flags에 대한 정보가 없다. 위의 RTO는 sender가 너무 많은 acknowledgments를 잃어버려 time out을 결정하거나 보내는 것을 멈출 때 나타난다고 한다. 즉 TCP Retransmission이 일어난 패킷은 acknowledgments를 많이 일어버려 재전송을 시작한 것이다.

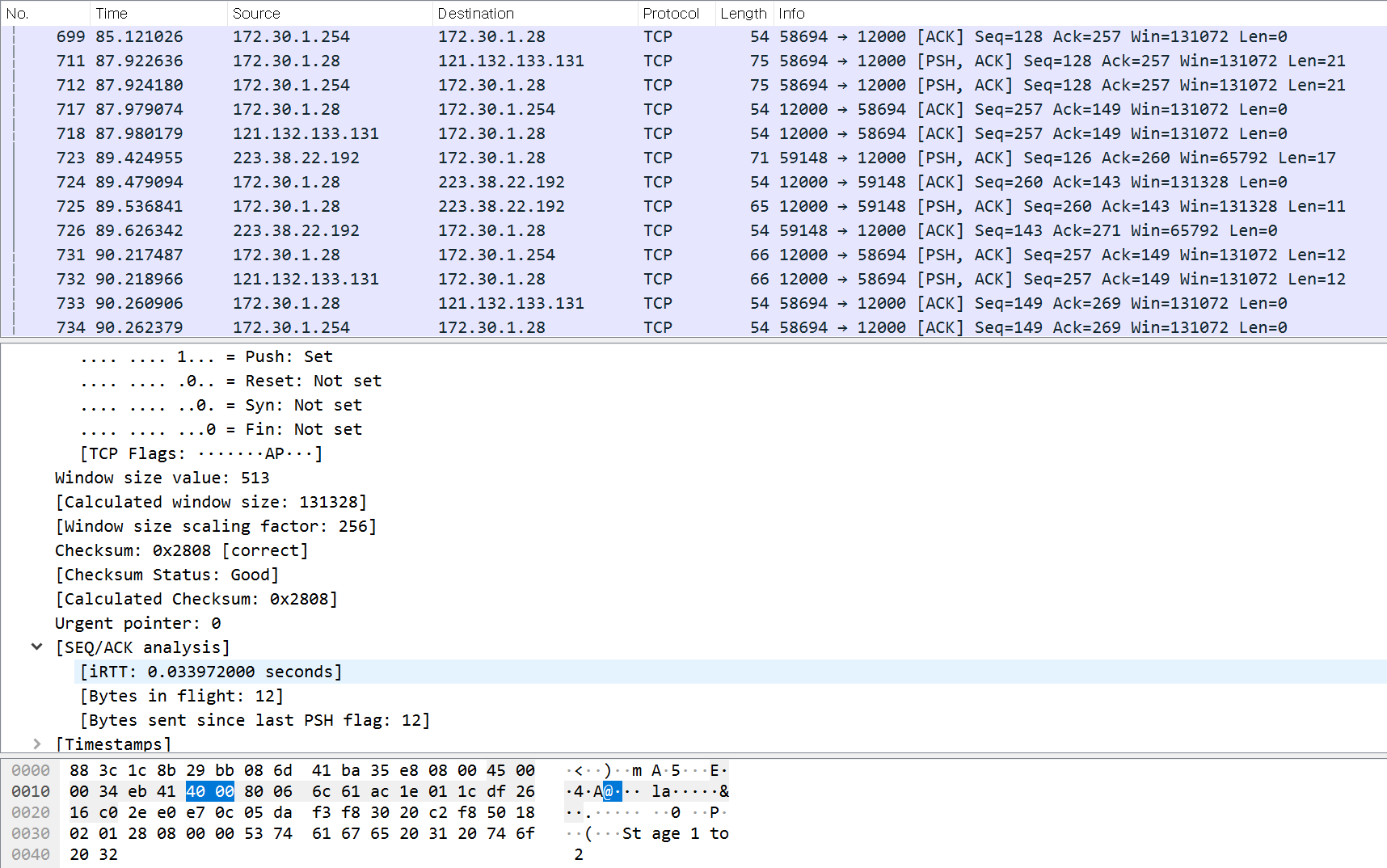
추가적으로 아래 일반적인 패킷의 캡쳐 결과에서는 RTT가 약 0.09초이며 데이터를 보냈지만 ACK가 오지 않은, 즉 in flight상태의 데이터가 총 18B라는 것을 알 수 있다.

**묵찌빠 게임 – 서버에서 게임 시작을 알림 round 2**

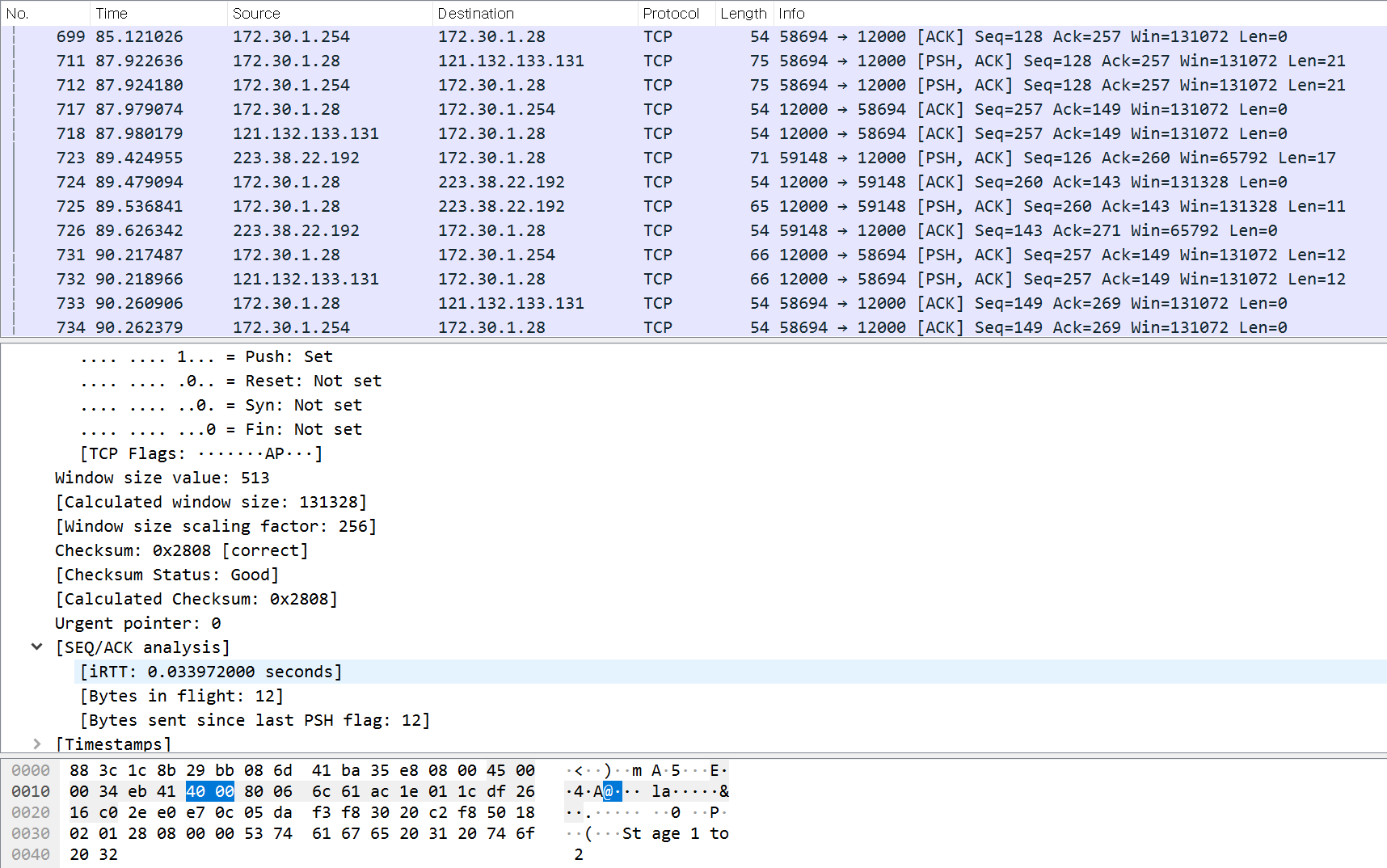


위의 캡쳐본은 서버와 클라이언트1의 와이어샤크 캡쳐본이다. No.689에서는 서버 내부 ip에서 외부 ip에 receiving stage1의 메시지를 전송하고, no.696, no. 697은 각각 서버 내부 ip에서 클라이언트 1에게, 서버 외부 ip에서 클라이언트 2에게 receiving stage 1의 메시지를 전송한다. 이제 가위바위보를 통해 공수를 정해야 한다.

**묵찌빠 게임 – 가위바위보 공수 결정 step 1**



위는 서버에서의 와이어샤크 캡쳐 결과이다.



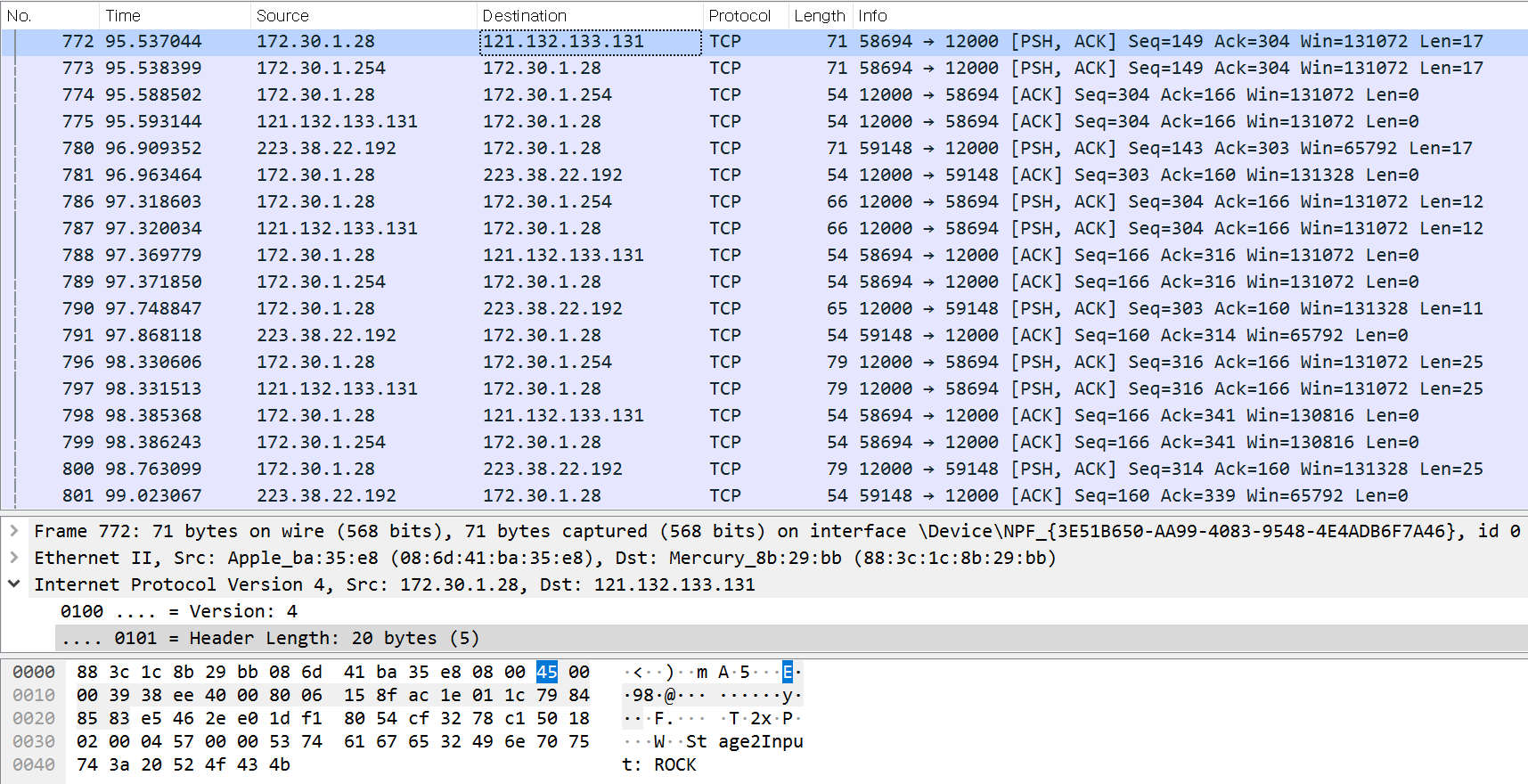
위는 클라이언트 2(서아)캡쳐 사진이다.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 패킷 number | 패킷 | 내용 |
| 서버&클라이언트 1 | 712 | 클라이언트 1 -> 내부 ip  가위 냈다고 전송 | Checksum = 0x677a  TTL = 127  Checksum correct |
| 서버&클라이언트 1 | 723 | 클라이언트 2 -> 서버  바위 냈다고 전송 | Checksum = 0xb20e  TTL = 113  Checksum correct |
| 클라이언트 2 | 478 | Checksum = 128  TTL = 0xd171  Checksum correct |
| 서버&클라이언트 1 | 725 | 내부 ip -> 클라이언트 1  Lose 전송 | Checksum =0x6c63  TTL = 128  Checksum correct |
| 클라이언트 2 | 480 | 서버 -> 클라이언트 2  Win 전송 | Checksum =0x1d31  TTL = 114  Checksum correct |

진행 과정은 round1의 진행 과정과 거의 동일하기에 stage1→stage2로 가는 과정 등은 생략하겠다.

공격자가 클라이언트2(서아)로 정해졌으므로 stage2로 이동

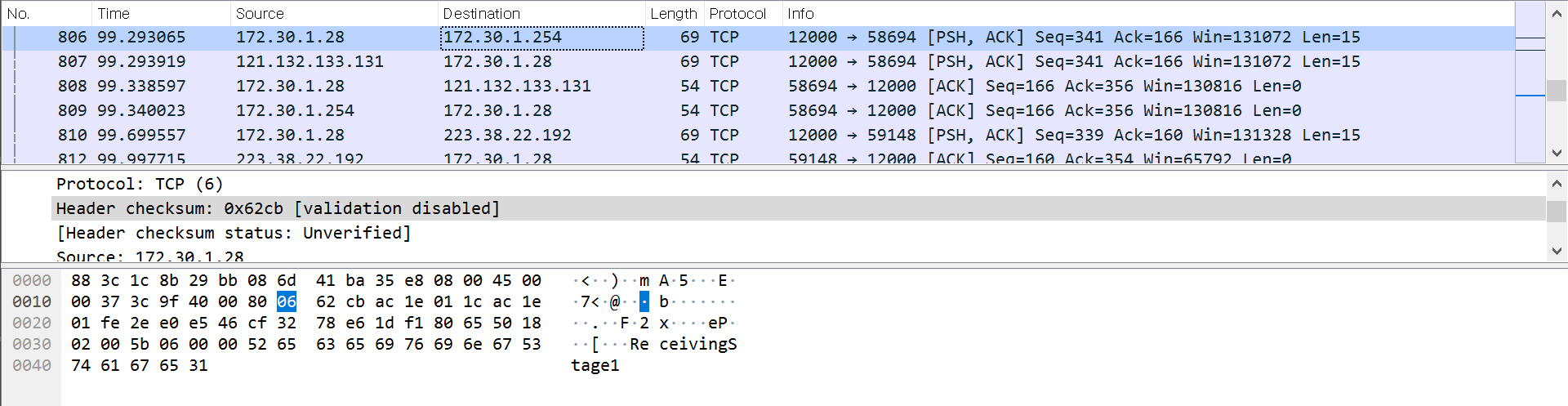
**묵찌빠 게임 – 묵찌빠 게임 시작 step 2**



위는 서버에서 캡쳐한 사진이다.

No.772에서 서버 내부 ip에서 서버 외부 ip로 묵을 냈다고 전송한 뒤 no.773에서 클라이언트 1에서 서버 내부 ip로 묵을 냈다고 전송한다. 그 뒤 no.780에서 클라이언트2가 서버 외부 ip로 묵을 냈다고 전송했다. 그 결과 바로 결판이 나게 된다. ROUND1과는 다르게 이번에는 draw없이 한번에 결판이 났고, 이전 가위바위보 공수 정할 때 공격자가 클라이언트 2(서아)였으므로 이번에는 서아가 승리하게 되었다. Round 1에서는 클라이언트 1(환효)가 이겼었는데 이번엔 환효가 졌으므로 score는 1:1로 동점이다.

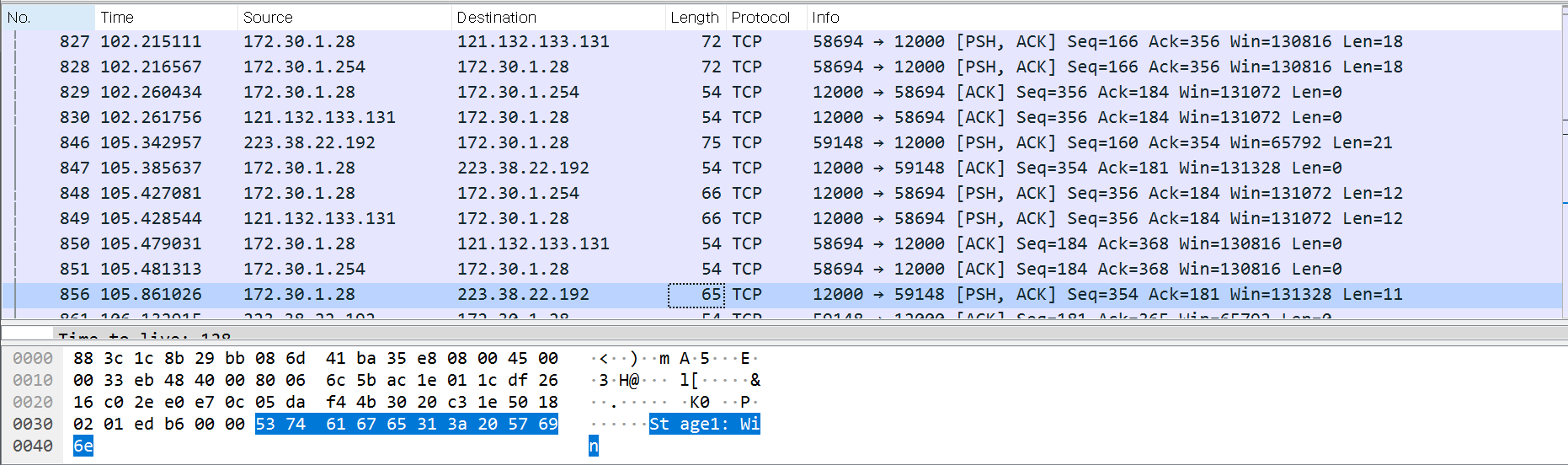
**묵찌빠 게임 – 서버에서 게임 시작을 알림 round 3**



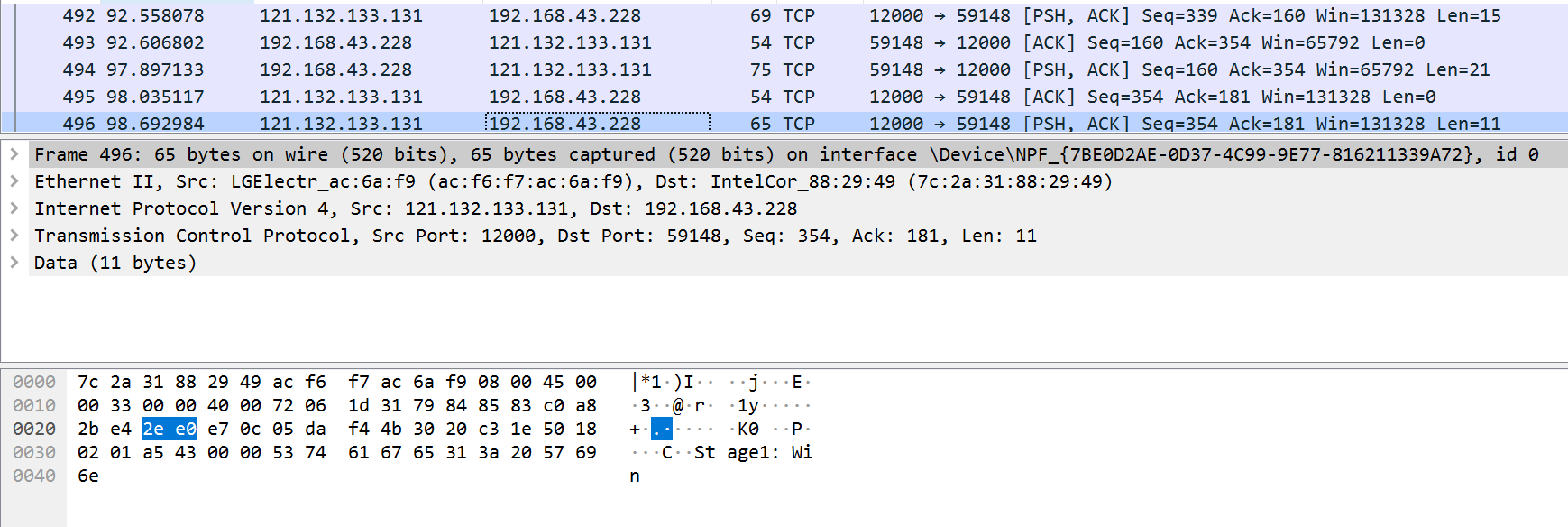
위 캡쳐본은 클라이언트1과 서버(환효)의 캡쳐본이다. 여기에서는 서버 내부ip에서 환효의 클라이언트로 receiving stage1의 메시지가 전송된다.

또한 외부ip에서 서아의 클라이언트로도 동일한 receiving stage1의 메시지를 전송한다.

**묵찌빠 게임 – 가위바위보 공수 결정 step 1**



환효의 서버&클라이언트1을 캡쳐한 사진



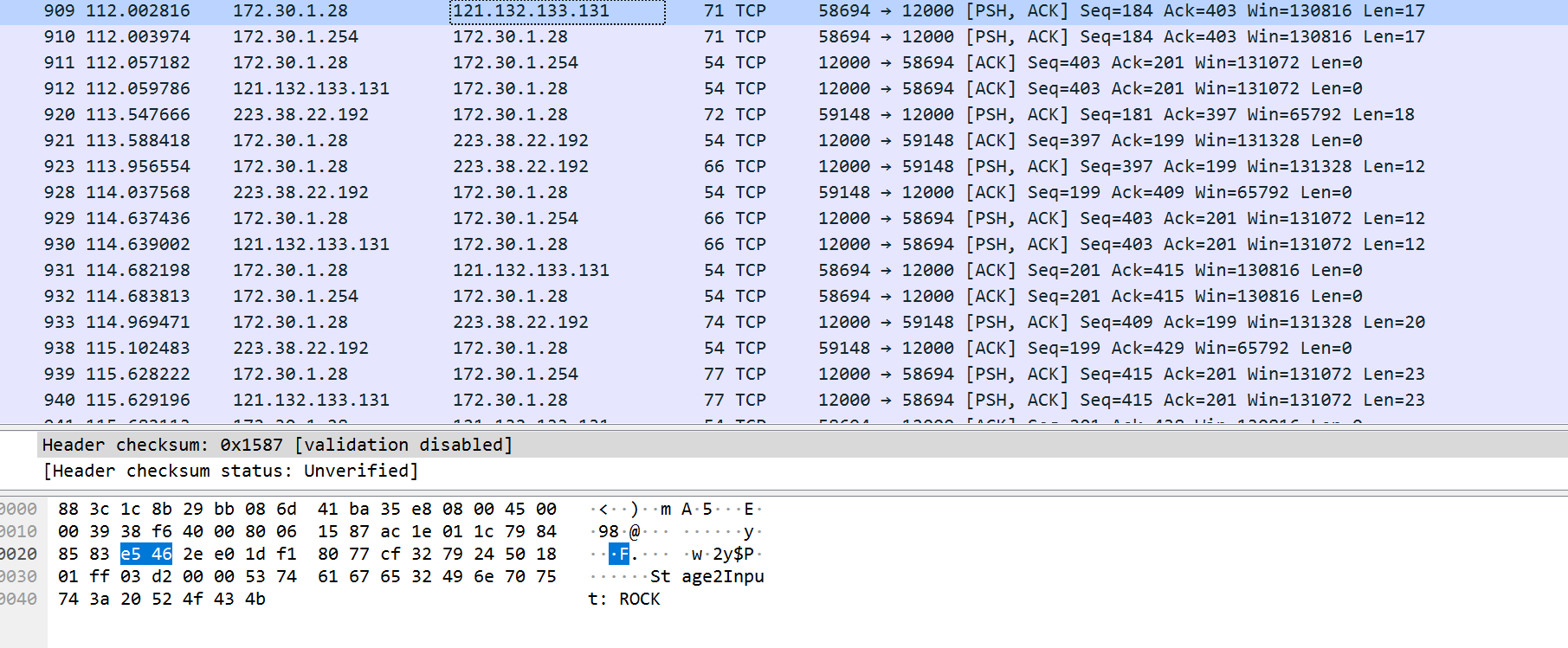
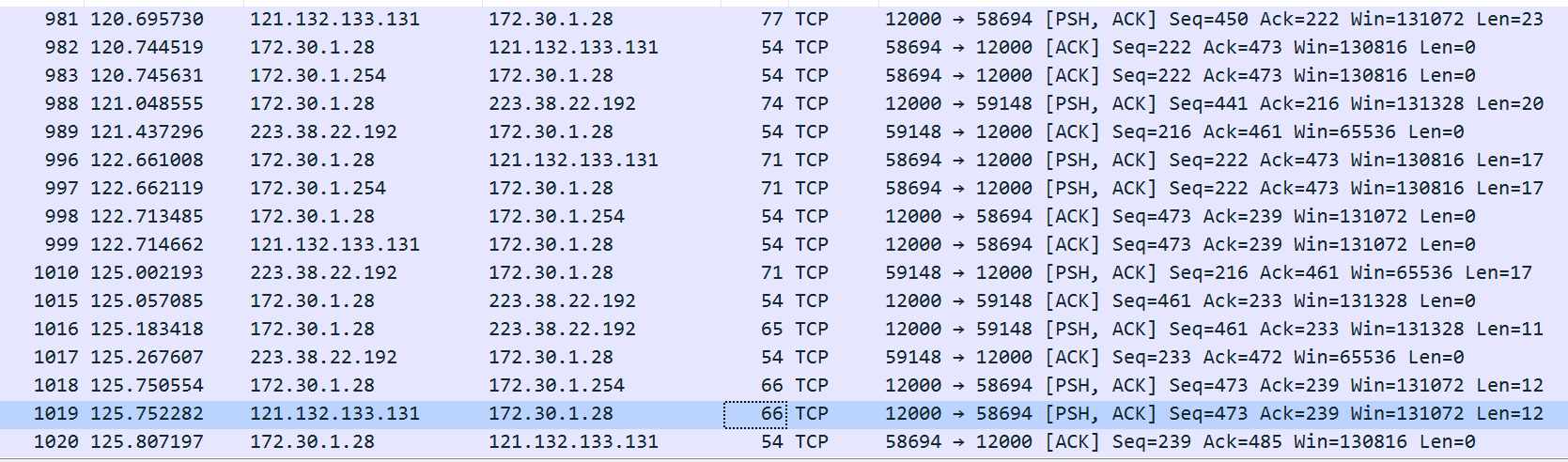
서아의 클라이언트2를 캡쳐한 사진

|  |  |  |  |
| --- | --- | --- | --- |
|  | 패킷 number | 패킷 | 내용 |
| 서버&클라이언트 1 | 827 | 내부 ip -> 외부 ip  보 냈다고 전송 | Checksum = 0x17b6  TTL = 128  Checksum correct |
| 서버&클라이언트 1 | 846 | 클라이언트 2 -> 서버  가위 냈다고 전송 | Checksum = 0x9bbb  TTL = 113  Checksum correct |
| 클라이언트 2 | 494 | Checksum =0xd165  TTL = 128  Checksum correct |
| 서버&클라이언트 1 | 848 | 내부 ip -> 클라이언트 1  Lose 전송 | Checksum =0x62cc  TTL = 128  Checksum correct |
| 클라이언트 2 | 496 | 서버 -> 클라이언트 2  Win 전송 | Checksum =0x1d31  TTL = 114  Checksum correct |

클라이언트1(환효)이 보, 클라이언트2(서아)이 가위를 내어 서아가 이긴다. 따라서 클라1에게는 Lose, 클라2에게는 Win의 메시지가 전송되며 공격권은 서아에게 가게 된다.

**묵찌빠 게임 – 묵찌빠 게임 시작 step 2**

다음 사진들은 서버에서의 캡쳐내용이다.

No.909, 910에서는 내부 ip에서 외부ip로, 클라이언트1에서 내부ip로 클라이언트1이 바위를 입력했다는 메시지가 전송된다.

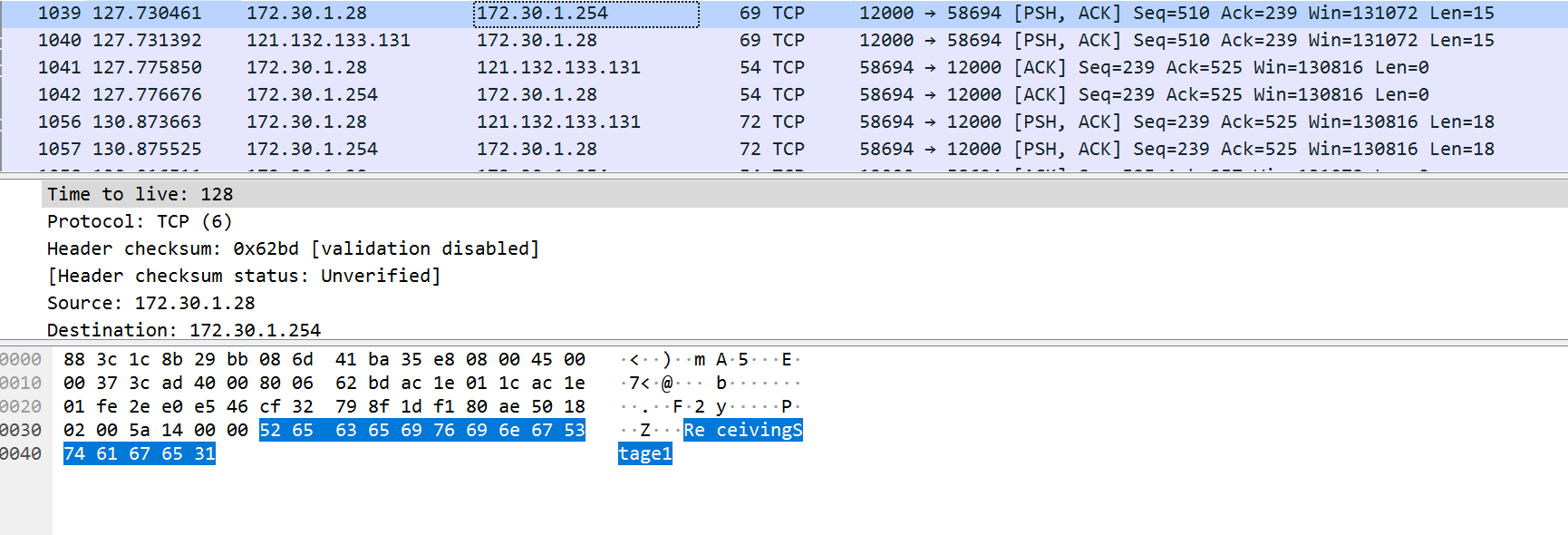
No.920에서는 클라이언트2가 보를 입력했다는 것이 서버의 내부 ip에 전송된다.

가위바위보에서 먼저 클라2가 공격권을 가지고 시작했으나, 묵찌빠 1회차에서 클라1은 바위, 클라2는 보를 입력하였다. 따라서 공격권이 클라1(환효)에게 넘어가게 된다.

묵찌빠 2회차에서는 No.948, 949를 통해 클라1이 가위를 입력했고, No.967을 통해 클라2는 바위를 입력했음을 알 수 있다. 이때 다시 공격권이 클라2(서아)에게 넘어가게 된다.

묵찌빠 3회차에서는 No. 996, 997을 통해 클라 1이 바위를 입력한 것을, 그리고 No. 1010을 통해 클라1도 바위를 냈음을 알 수 있다. 따라서 공격권을 가진 서아가 이긴다. 이때 점수는 클라1(환효): 클라2(서아) = 1 : 2 이다.

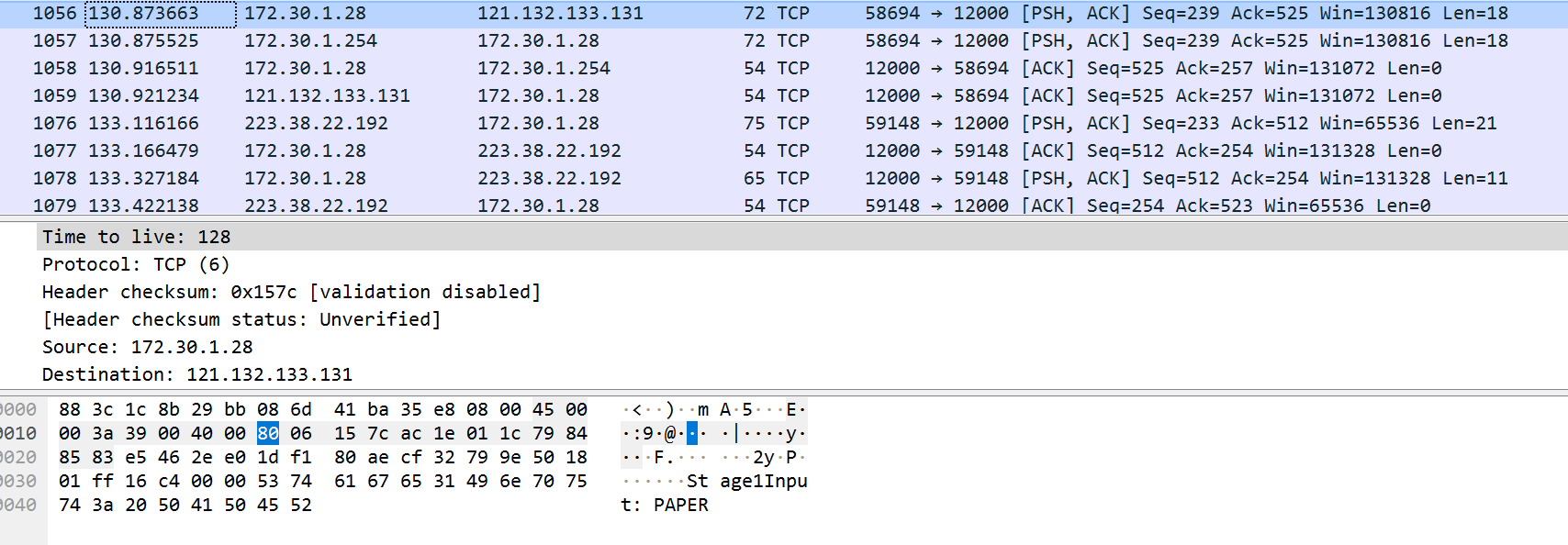
**묵찌빠 게임 – 서버에서 게임 시작을 알림 round 4**



위 사진은 서버&클라이언트1을 캡쳐한 것이다.

No.1039와 No.1040에서 Receiving stage1의 메시지를 확인할 수있다. no.1039는 서버 내부ip에서 클라1로 전송한 메시지이고, no.1040은 내부ip가 외부ip에 전송한 메시지이다.

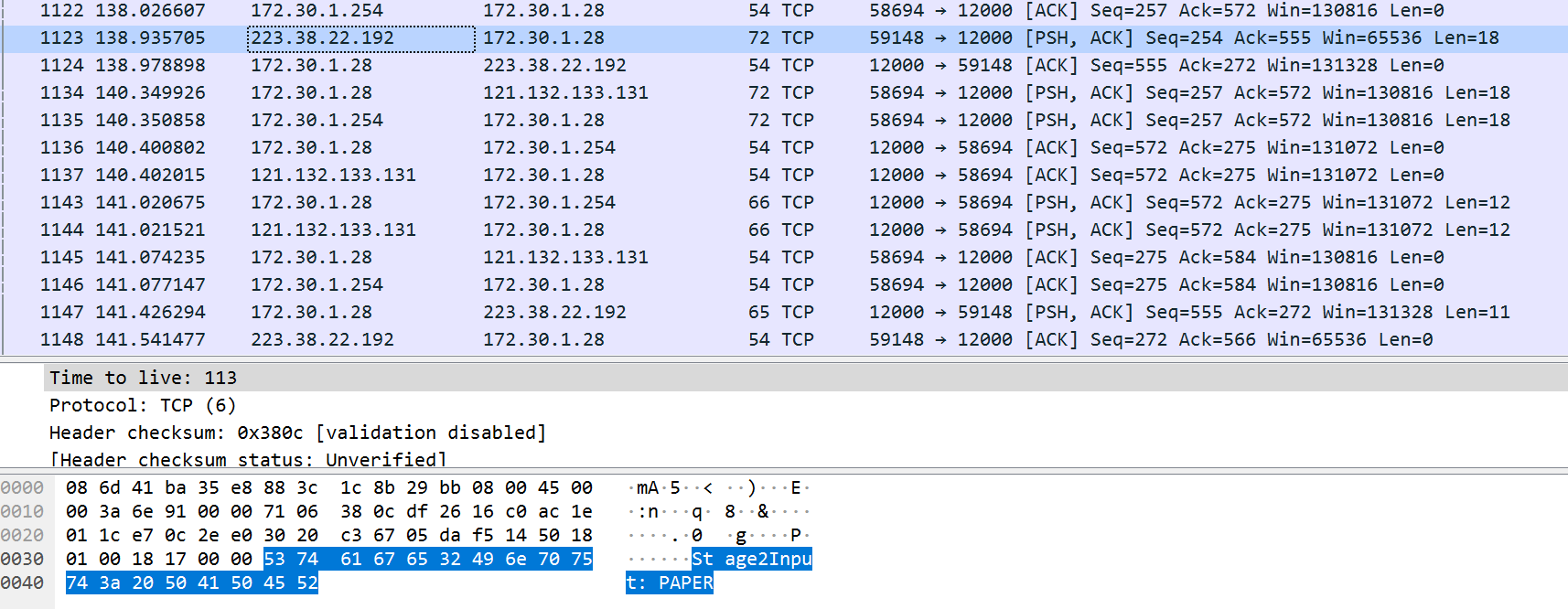
**묵찌빠 게임 – 가위바위보 공수 결정 step 1**



Round4의 가위바위보 결과는 Round3과 동일하다.

**묵찌빠 게임 – 묵찌빠 게임 시작 step 2**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 패킷 number | 패킷 | 내용 |
| 서버&클라이언트 1 | 1056 | 내부 ip -> 외부 ip  보 냈다고 전송 | Checksum = 0x157c  TTL = 128  Checksum correct |
| 서버&클라이언트 1 | 1076 | 클라이언트 2 -> 서버  가위 냈다고 전송 | Checksum = 0x9bd0  TTL = 113  Checksum correct |
| 클라이언트 2 | 526 | Checksum =0xd157  TTL = 128  Checksum correct |
| 서버&클라이언트 1 | 1080 | 내부 ip -> 클라이언트 1  Lose 전송 | Checksum =0x62be  TTL = 128  Checksum correct |
| 클라이언트 2 | 528 | 서버 -> 클라이언트 2  Win 전송 | Checksum =0x1d31  TTL = 114  Checksum correct |



위 사진은 서버&클라이언트1의 화면을 캡쳐한 것이다.

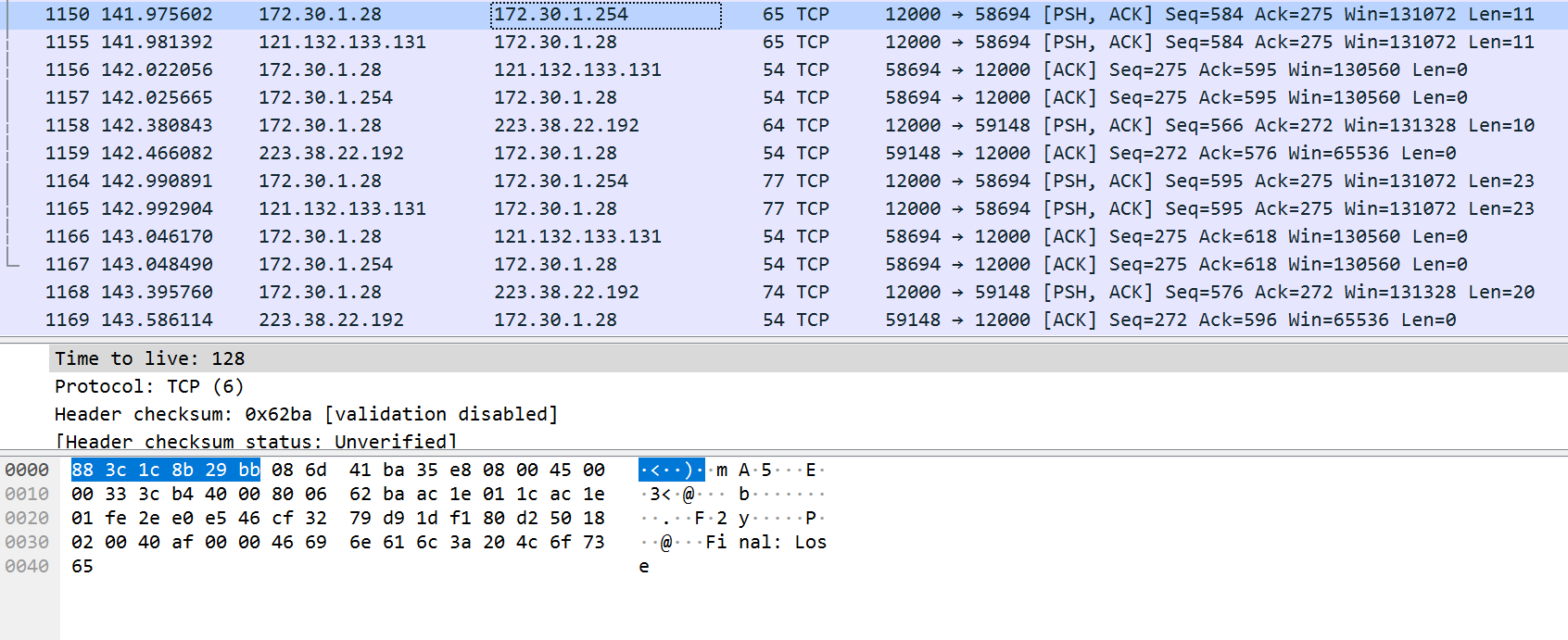
가위바위보에서 클라2(서아)가 이겨서 공격권을 가지게 된다. 이때 묵찌빠의 값은 둘다 보가 입력된다.

(No.1123에서 클라2에서 서버 내부ip로 보가 입력되어 전송됨을 알 수 있으며, No.1134, 1135를 통해 서버 내부ip에서 외부ip로, 클라1에서 서버 내부ip로 보가 입력되어 전송됨을 알 수 있다.)

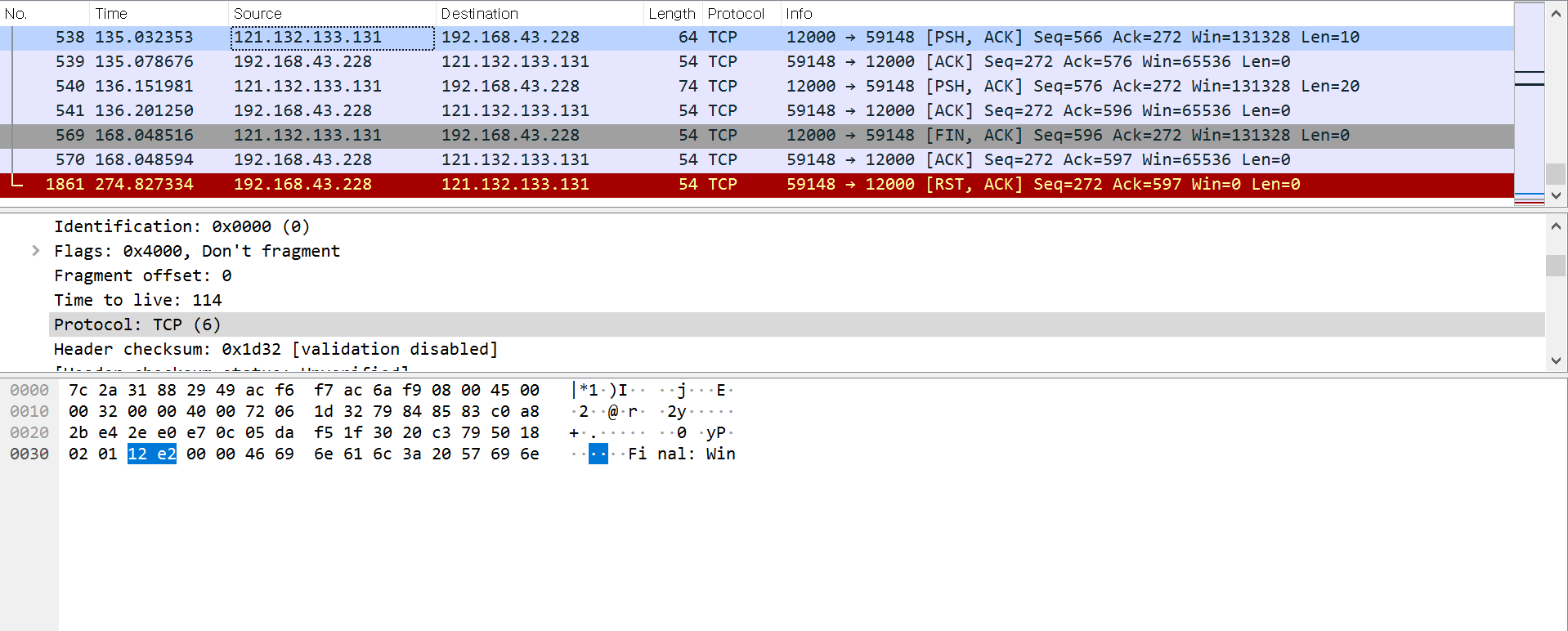
그러므로 클라2가 이겨서 점수를 얻는다.

이때 점수는 클라1(환효): 클라2(서아) = 1 : 3이다.

**묵찌빠 게임 – 묵찌빠 게임 종료**



서버&클라이언트1에서 Final 점수를 출력 후 종료하기까지의 화면



클라이언트2에서 Final 점수를 출력 후 종료하기까지의 화면

한쪽의 점수가 3점이 되었으므로 게임이 종료된다. 이때 Final 결과를 Win/Lose로 출력하여 각각의 클라이언트에 전송한다. 그다음 클라이언트 2의 화면처럼, FIN과 RST플래그를 받아 연결을 종료한다 . FIN은 연결종료 요청 플래그로, 더 이상 전송할 데이터가 없음을 나타내며 세션연결을 종료시킬 때 사용한다. RST은 연결종료 플래그이다. 패킷을 보내는 곳이 현재 접속하고 있는 곳과 즉시 연결을 끊고자 할 때 사용한다.