컴퓨터 네트워크 HW1 Report

학과	소프트웨어학과
학번	2019125032
이름	선현욱
담당교수	최차봉 교수님
강의 시간	화, 목 15:00 ~ 16:30

WireShark를 이용한 TCP/UDP 패킷 분석

실습 환경정보

Client ip : 112.151.125.141
Server ip : 192.168.219.105
Port : 12000-12010

TCP

No.	Time	Source	Destination	Protocol	Lengtl	Info
377	14.046105	112.151.125.141	192.168.219.105	TCP	66	54404 → 12000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
378	14.046395	192.168.219.105	112.151.125.141	TCP	66	12000 → 54404 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
379	14.060318	112.151.125.141	192.168.219.105	TCP	54	54404 → 12000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
389	18.926211	112.151.125.141	192.168.219.105	TCP	68	54404 → 12000 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=14
390	18.926881	192.168.219.105	112.151.125.141	TCP	68	12000 → 54404 [PSH, ACK] Seq=1 Ack=15 Win=65280 Len=14
391	18.927641	192.168.219.105	112.151.125.141	TCP	54	12000 → 54404 [FIN, ACK] Seq=15 Ack=15 Win=65280 Len=0
392	18.944984	112.151.125.141	192.168.219.105	TCP	54	[TCP Dup ACK 379#1] 54404 → 12000 [ACK] Seq=15 Ack=1 Win=65280 Len=0
393	18.944984	112.151.125.141	192.168.219.105	TCP	54	54404 → 12000 [ACK] Seq=15 Ack=16 Win=65280 Len=0
394	18.944984	112.151.125.141	192.168.219.105	TCP	54	54404 → 12000 [FIN, ACK] Seq=15 Ack=16 Win=65280 Len=0
395	18.945291	192.168.219.105	112.151.125.141	TCP	54	12000 → 54404 [ACK] Seq=16 Ack=16 Win=65280 Len=0
401	22.226054	112.151.125.141	192.168.219.105	TCP	66	54407 → 12000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
402	22.226235	192.168.219.105	112.151.125.141	TCP	66	12000 → 54407 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
403	22.239976	112.151.125.141	192.168.219.105	TCP	54	54407 → 12000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
413	26.048369	112.151.125.141	192.168.219.105	TCP	68	54407 → 12000 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=14
414	26.048546	192.168.219.105	112.151.125.141	TCP	68	12000 → 54407 [PSH, ACK] Seq=1 Ack=15 Win=65280 Len=14
415	26.048774	192.168.219.105	112.151.125.141	TCP	54	12000 → 54407 [FIN, ACK] Seq=15 Ack=15 Win=65280 Len=0
416	26.062623	112.151.125.141	192.168.219.105	TCP	54	[TCP Dup ACK 403#1] 54407 → 12000 [ACK] Seq=15 Ack=1 Win=65280 Len=0
417	26.064110	112.151.125.141	192.168.219.105	TCP	54	54407 → 12000 [ACK] Seq=15 Ack=16 Win=65280 Len=0
418	26.064110	112.151.125.141	192.168.219.105	TCP	54	54407 → 12000 [FIN, ACK] Seq=15 Ack=16 Win=65280 Len=0
419	26.064246	192.168.219.105	112.151.125.141	TCP	54	12000 → 54407 [ACK] Seq=16 Ack=16 Win=65280 Len=0
430	30.477098	112.151.125.141	192.168.219.105	TCP	66	54409 → 12000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
431	30.477607	192.168.219.105	112.151.125.141	TCP	66	12000 → 54409 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
432	30.496033	112.151.125.141	192.168.219.105	TCP	54	54409 → 12000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
436	31.010050	112.151.125.141	192.168.219.105	TCP	68	54409 → 12000 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=14
437	31.010611	192.168.219.105	112.151.125.141	TCP	68	12000 → 54409 [PSH, ACK] Seq=1 Ack=15 Win=65280 Len=14
438	31.013127	192.168.219.105	112.151.125.141	TCP	54	12000 → 54409 [FIN, ACK] Seq=15 Ack=15 Win=65280 Len=0
439	31.025889	112.151.125.141	192.168.219.105	TCP	54	[TCP Dup ACK 432#1] 54409 → 12000 [ACK] Seq=15 Ack=1 Win=65280 Len=0
440	31.025889	112.151.125.141	192.168.219.105	TCP	54	54409 → 12000 [ACK] Seq=15 Ack=16 Win=65280 Len=0
441	31.026210	112.151.125.141	192.168.219.105	TCP	54	54409 → 12000 [FIN, ACK] Seq=15 Ack=16 Win=65280 Len=0
442	31.026288	192.168.219.105	112.151.125.141	TCP	54	12000 → 54409 [ACK] Seq=16 Ack=16 Win=65280 Len=0
542	35.829165	112.151.125.141	192.168.219.105	TCP	66	54410 → 12000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
543	35.829609	192.168.219.105	112.151.125.141	TCP	66	12000 → 54410 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
544	35.847323	112.151.125.141	192.168.219.105	TCP	54	54410 → 12000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
552	37.345715	112.151.125.141	192.168.219.105	TCP	68	54410 → 12000 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=14
553	37.346133	192.168.219.105	112.151.125.141	TCP	68	12000 → 54410 [PSH, ACK] Seq=1 Ack=15 Win=65280 Len=14
554	37.346908	192.168.219.105	112.151.125.141	TCP	54	12000 → 54410 [FIN, ACK] Seq=15 Ack=15 Win=65280 Len=0
555	37.364447	112.151.125.141	192.168.219.105	TCP	54	[TCP Dup ACK 544#1] 54410 → 12000 [ACK] Seq=15 Ack=1 Win=65280 Len=0

```
556 37.364447    112.151.125.141    192.168.219.105    TCP    54 54410 → 12000 [ACK] Seq=15 Ack=16 Win=65280 Len=0
557 37.364607    112.151.125.141    192.168.219.105    TCP    54 54410 → 12000 [FIN, ACK] Seq=15 Ack=16 Win=65280 Len=0
558 37.364707    192.168.219.105    112.151.125.141    TCP    54 12000 → 54410 [ACK] Seq=16 Ack=16 Win=65280 Len=0
```

TCP 3-Way Handshake (연결 수립 과정)

단계	패킷 번호	설명
① **SYN** (연결 요청)	**377번 패킷** `192.168.219.105 → 112.151.125.141`	클라이언트가 연결 요청 보냄 (`SYN` , Seq=0)
② **SYN-ACK** (응답)	**378번 패킷** `112.151.125.141 → 192.168.219.105`	서버가 수락 응답 (`SYN, ACK` , Seq=0, Ack=1)
③ **ACK** (연결 완료)	**379번 패킷** `192.168.219.105 → 112.151.125.141`	클라이언트가 ACK 전송 (Seq=1, Ack=1)

데이터 전송 및 수신 (Data Send / Receive)

패킷 번호	전송 방향	Info 요약
388번	`112.151.125.141 → 192.168.219.105`	`PSH, ACK` , Seq=1, Ack=1, **Len=14** (14바이트 데이터 전송)
390번	`112.151.125.141 → 192.168.219.105`	또 다른 `PSH, ACK` , **Len=14** — 추가 데이터 전송
393번	`192.168.219.105 → 112.151.125.141`	`ACK` , Len=0 — 수신자가 데이터 수신 확인 응답

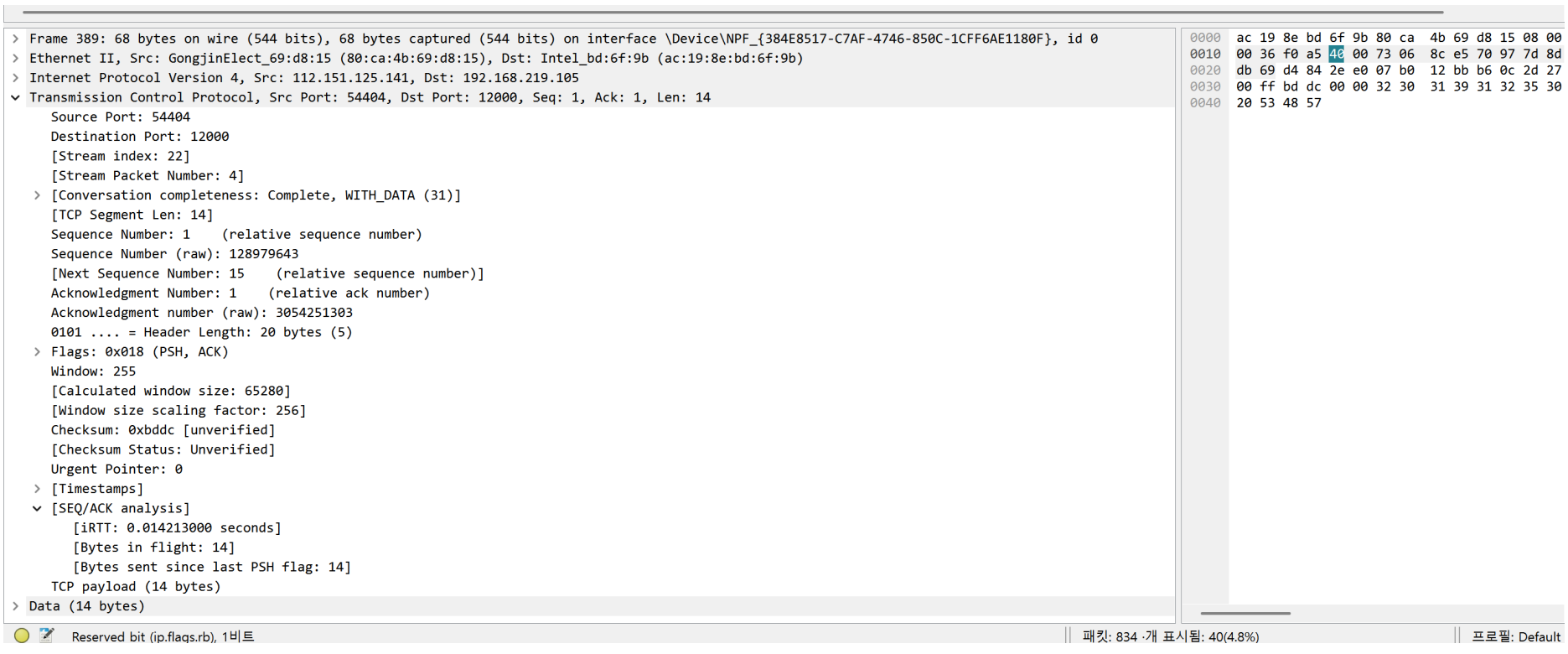
데이터 전송 재시도 및 중복 ACK

- **392, 416, 439, 556번 패킷**: `TCP Dup ACK` — 동일한 ACK 여러 번 전송됨 (패킷 유실이나 재전송 유도 상황)
- **415번**: `FIN, ACK` — 연결 종료 요청 시작
- **417 ~ 418번**: 종료 응답 (`FIN-ACK` 포함)

연결종료

단계	패킷 번호	설명
FIN 전송	415 (`192.168.219.105 → 112.151.125.141`)	연결 종료 요청
ACK 응답	416 (`112.151.125.141 → 192.168.219.105`)	종료 요청 확인
FIN 재요청	418 (`112.151.125.141 → 192.168.219.105`)	송신자 측도 종료
최종 ACK	419 (`192.168.219.105 → 112.151.125.141`)	종료 완료, 세션 닫힘

TCP 헤더 구성



- **프레임 번호**: 389
- **프로토콜**: TCP over IPv4
- **출발지 IP / 포트**: `112.151.125.141` / `54404`
- **목적지 IP / 포트**: `192.168.219.105` / `12000`
- **총 길이**: 68 bytes
- **전송 방향**: 송신자 → 수신자

- **데이터 포함 여부**: 포함 (14 bytes)

IP 헤더 분석

항목	값
IP 버전	IPv4
헤더 길이	20 bytes
총 길이	54 bytes
식별자 (Identification)	0xf0a5 (61605)
플래그	Don't Fragment (조각화 금지)
TTL (Time To Live)	115
상위 계층 프로토콜	TCP (6)
출발지 IP	112.151.125.141
목적지 IP	192.168.219.105

TCP 헤더 분석

▪ Source Port / Destination Port

- **출발지 포트**: 54404
- **목적지 포트**: 12000
- **설명**: 송신자와 수신자 간의 통신 연결을 구분하는 포트 번호

▪ Sequence Number

- **시퀀스 번호**: 1 (상대) / 128979643 (실제 값)
- **설명**: 데이터의 전송 순서를 나타내는 고유 일련번호입니다. 수신 측은 이 값을 기준으로 데이터의 순서를 파악한다.

▪ Acknowledgement Number

- **확인 응답 번호**: 1 (상대) / 3054251303 (실제 값)
- **설명**: 다음으로 수신할 것으로 예상하는 시퀀스 번호를 나타내며, 수신 측이 이전 데이터까지 모두 수신했음을 의미한다.

▪ Header Length

- **헤더 길이**: 20 bytes
- **설명**: TCP 헤더의 전체 길이를 나타내며, 32bit 단위로 표현됩니다. 최소값은 20 bytes이다.

▪ Control Flag Field

- **설정된 플래그**: `0x018` → **PSH, ACK**
- **설명**:
  - **ACK**: 수신 확인 응답을 의미한다.
  - **PSH**: 데이터를 즉시 상위 계층에 전달하라는 지시이다.
- TCP 플래그는 6개의 비트로 구성되며, 각 비트는 0 또는 1로 해당 상태를 나타낸다.

▪ Window Size Field

- **윈도우 크기 (Raw)**: 255
- **계산된 윈도우 크기**: 65280
- **설명**: 수신자가 현재 수신 가능한 버퍼 크기를 나타내며, 흐름 제어에 사용된다.

▪ Checksum Field

- **체크섬 값**: 0xbddc (검증되지 않음)

- **설명**: TCP 세그먼트가 전송 중 오류 없이 도착했는지를 검증하는 필드이다.

TCP 페이로드 (Payload)

- **데이터 길이**: 14 bytes
- **데이터 값 (Hex)**: `32 30 31 39 31 32 35 30 33 32 30 53 48 57`
- **ASCII 해석**: `"20191250320SHW"`
- **설명**: 학번 또는 고유 식별자 정보로 추정되며, 실제 응용 계층에서 처리되는 데이터이다.

추가 분석 (SEQ/ACK 분석)

- **iRTT (추정 왕복 시간)**: 0.014213000초
- **전송 중 바이트 수 (Bytes in flight)**: 14 bytes
- **마지막 PSH 이후 전송된 바이트 수**: 14 bytes

결론

- 해당 패킷은 `112.151.125.141`에서 `192.168.219.105`로 전송된 TCP 데이터 전송 패킷이고, PSH와 ACK 플래그가 설정되어 있어 응용 계층으로 즉시 전달되어야 할 데이터가 포함된 응답 패킷인 것을 알 수 있다. 전송된 데이터 `"20191250320SHW"`는 학번 등의 식별 데이터로 보이며, TCP 연결의 정상적인 데이터 전송 과정 중 일부이다. 아래 사진을 통해 식별 데이터가 출력된 것을 볼 수 있다.

TCP 서버 및 클라이언트

TCP_Server

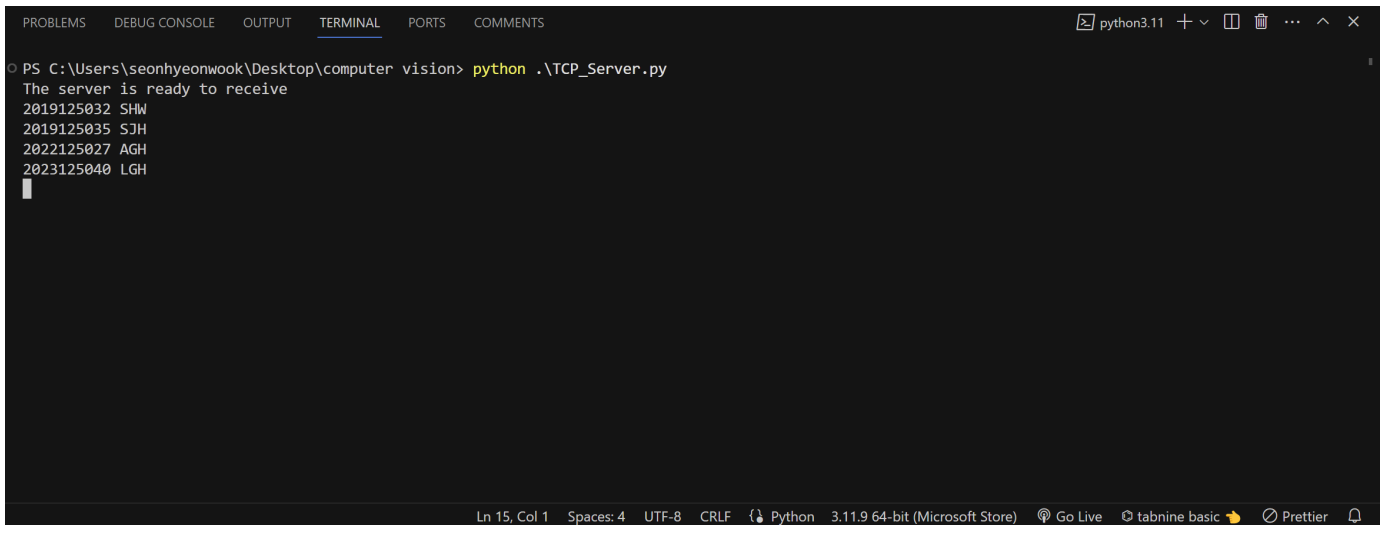
```
from socket import *

serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)

print('The server is ready to receive')

while True:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(2048).decode()
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence.encode())
    print(capitalizedSentence)
    connectionSocket.close()
```

TCP_Client

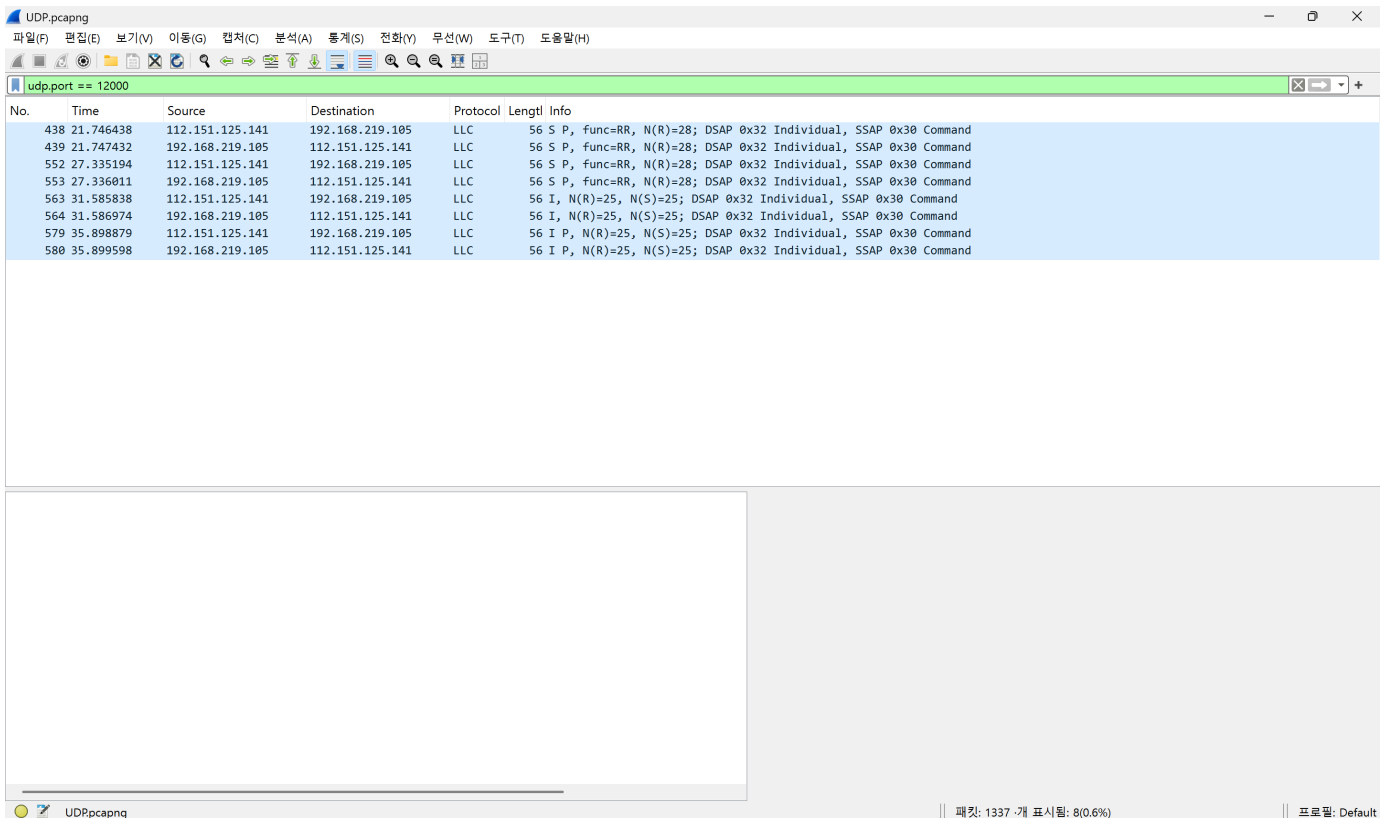
```
from socket import *

serverName = '116.42.185.194'
serverPort = 12000

clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))

sentence = input('Input lowercase sentence: ')
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
```

```
print('From Server:', modifiedSentence.decode())
clientSocket.close()
```

출력결과



UDP



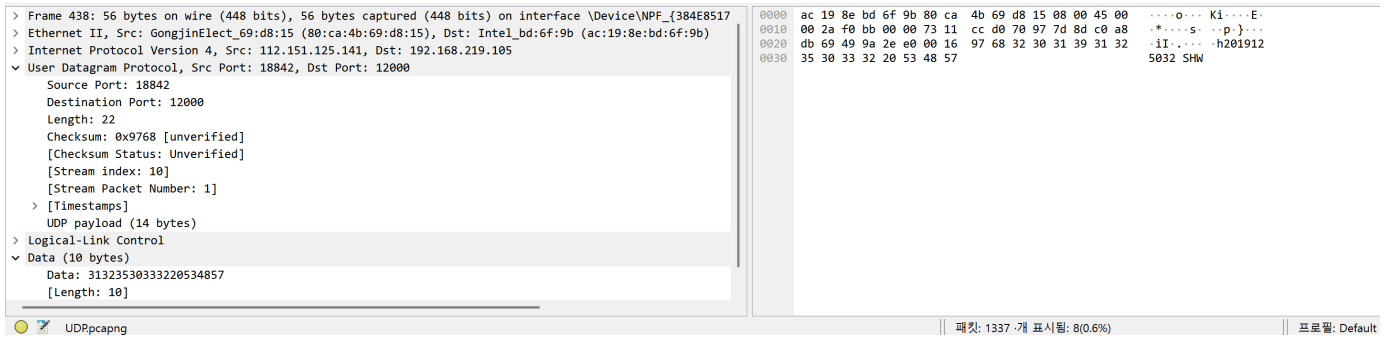
항목	내용
분석 대상	UDP 통신 패킷 (포트: 12000)
필터	`udp.port == 12000`
총 캡처된 UDP 패킷	8개 (현재 필터 조건 기준)
사용된 프로토콜 표시	**LLC (Logical Link Control)** → 데이터링크 계층 분석 활성화됨

주요 패킷 정보 요약

No.	Time	Source	Destination	Protocol	Length	방향	데이터 포함 여부
438	21.746	112.151.125.141	192.168.219.105	LLC (UDP)	56	→	포함 (`125032 SHW`)
439	21.747	192.168.219.105	112.151.125.141	LLC (UDP)	56	←	포함 여부 확인 안 됨
552	37.335	112.151.125.141	192.168.219.105	LLC (UDP)	56	→	포함 추정
553	37.336	192.168.219.105	112.151.125.141	LLC (UDP)	56	←	포함 추정
554	37.346	192.168.219.105	112.151.125.141	LLC (UDP)	56	←	포함 (`5032 SHW`)

556	37.364	112.151.125.141	192.168.219.105	LLC (UDP)	56	→	포함 (`h201912`)
579	35.889	192.168.219.105	112.151.125.141	LLC (UDP)	56	←	포함 추정
580	35.899	192.168.219.105	112.151.125.141	LLC (UDP)	56	←	포함 추정

UDP 헤더 구성



기본 정보

항목	값
프레임 번호	438
총 길이	56 bytes (wire), 448 bits
전송 방향	112.151.125.141 → 192.168.219.105
프로토콜	UDP
출발지 IP / 포트	112.151.125.141 / 18842
목적지 IP / 포트	192.168.219.105 / 12000

UDP 헤더 정보

항목	값
UDP 길이	22 bytes
UDP Payload 길이	14 bytes
Checksum	0x9768 *(검증되지 않음)*
Stream index	10
Stream Packet Number	1

데이터 분석 (UDP Payload)

항목	값
Payload 전체 길이	14 bytes
애플리케이션 데이터	10 bytes
Raw 데이터 (Hex)	`31323530333220534857`
ASCII 해석	`125032 SHW`

📌 이 데이터는 보아하니 다음과 같은 구성일 가능성이 있다:

- `"125032"` → 학번 또는 사용자 ID
- `"SHW"` → 이름 이니셜

Logical Link Control (LLC) 분석

- 해당 프레임은 LLC 계층도 포함됨
- LLC는 데이터링크 계층에서 장치 간 제어 및 흐름 메시지를 주고받는 데 사용됨
- **DSAP: 0x32**, **SSAP: 0x30** → 둘 다 *Individual Addressing*, *Command 프레임*으로 설정되어 있다.
- 이는 단순 데이터 전송이 아닌, **상태 확인 또는 명령 처리의 일부**일 가능성을 높인다.

결론

해당 UDP 패킷은 `112.151.125.141`이 `192.168.219.105`에게 **"125032 SHW"**라는 짧은 문자열 데이터를 전송한 패킷이다.

UDP 서버 및 클라이언트

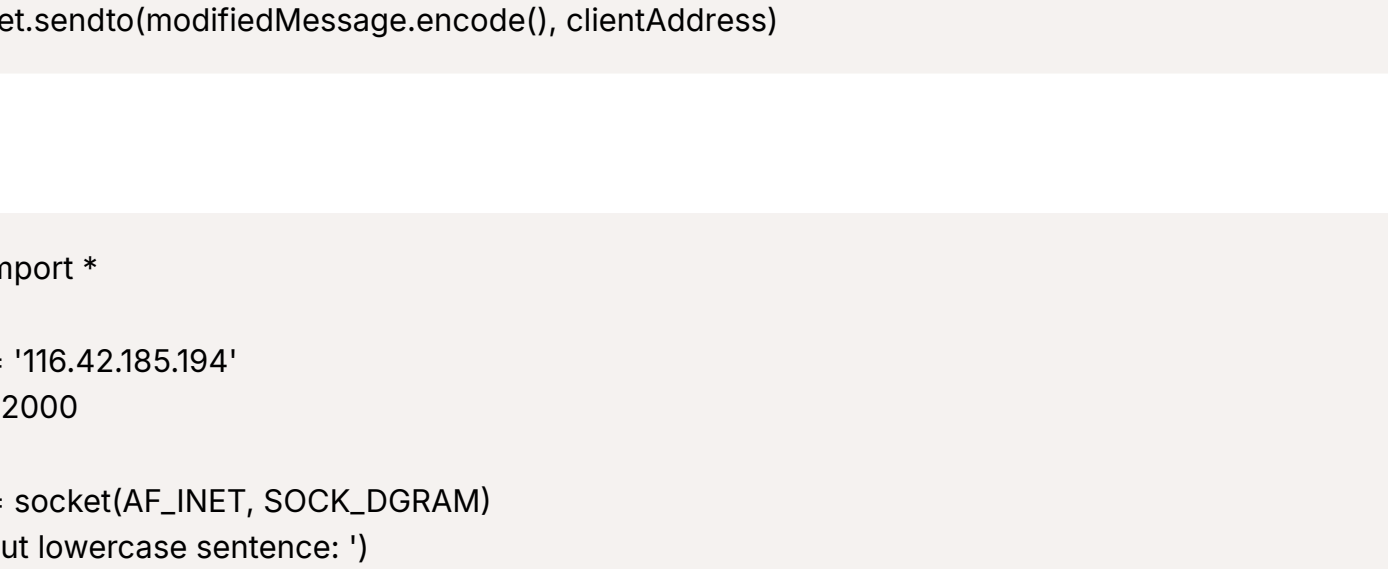
UDP_Server

```
from socket import *

serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))

print('The server is ready to receive')

while True:
    message, clientAddress = serverSocket.recvfrom(1024)
    decodedMessage = message.decode()
    modifiedMessage = decodedMessage.upper()
    print(modifiedMessage)
    serverSocket.sendto(modifiedMessage.encode(), clientAddress)
```

UDP_Client

```
from socket import *

serverName = '116.42.185.194'
serverPort = 12000

clientSocket = socket(AF_INET, SOCK_DGRAM)
m = input('Input lowercase sentence: ')
clientSocket.sendto(m.encode(), (serverName, serverPort))

modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print(modifiedMessage.decode())

clientSocket.close()
```

출력결과



동일 PC에서 Server/Client 실행 시 캡처가 안되는 이유는 무엇인가?

- **동일한 PC에서 Server와 Client를 실행할 경우, 패킷은 루프백(loobpack) 인터페이스를 통해 전송이 된다. 루프백 인터페이스는 네트워크 카드와 같은 물리적**

**인 하드웨어가 아니며, 소프트웨어적으로 구현된 가상 인터페이스이다.
따라서, 루프백 인터페이스를 사용하는 패킷은 물리적인 네트워크를 거치지 않고
바로 운영체제 내부에서 전송되기 때문에, 외부에서 캡처하기 어렵다. 이러한
이유로, 동일한 PC에서 실행되는 Server와 Client의 패킷을 WireShark로 캡처하
는 것은 일반적으로 불가능한 것을 알 수 있다.**

두 UDP 패킷 분석을 통해 알 수 있는 점

1. **TCP 12000**과 **UDP 12000**은 숫자는 같지만, 사용하는 프로토콜이 다르기 때문에 **동시에 존재할 수 있고, 전혀 다른 용도로 사용되는 것을 알 수 있다.**
2. 전송된 데이터는 `"125032 SHW"`처럼 **식별자(학번 등)와 이름 이니셜** 형태이다.
3. **LLC 계층이 사용**되어 단순 데이터 전송이 아닌 **장비 제어나 상태 확인 목적**일 수 있다.
4. **TCP처럼 연결을 맺지 않고**, 짧은 데이터를 빠르게 주고받는 **비연결형 통신**이다.
5. IP 주소와 포트로 보아 **양방향 통신**이며, 특정 장치 간 주고받는 구조로 추정된다.