

ネットワーク2(ソケット通信) レポート (3144 吉高僚真)

目的

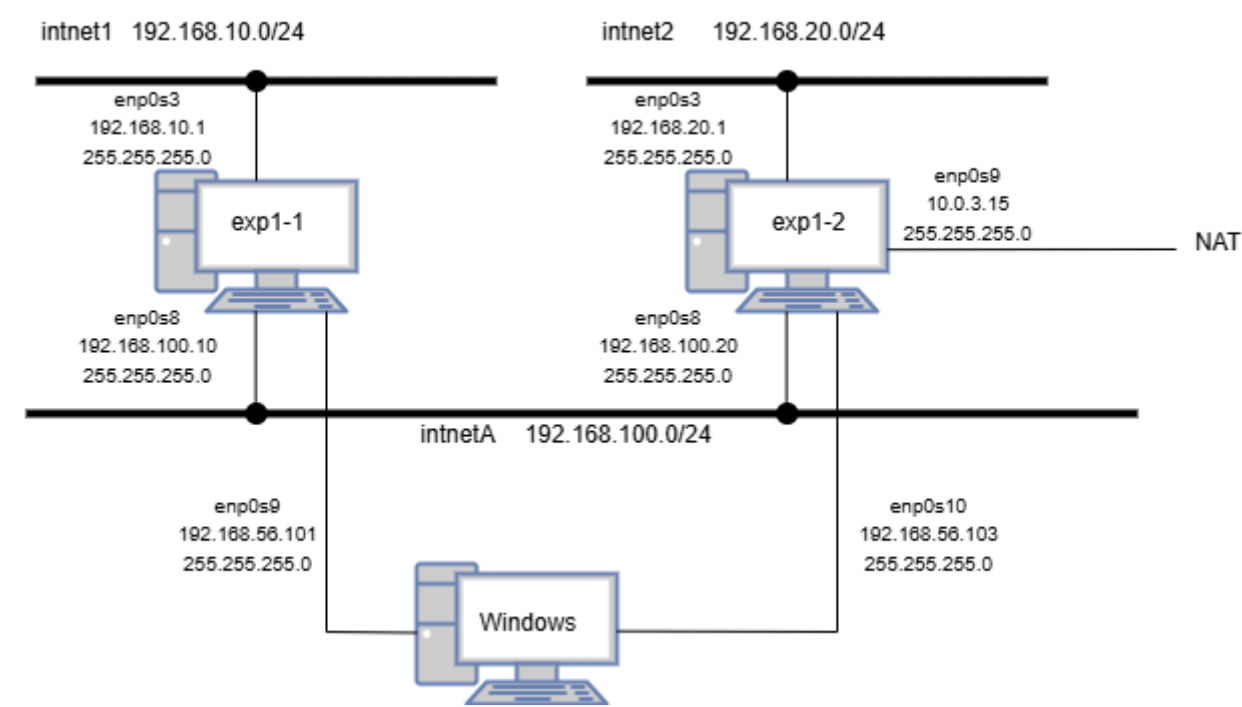
- VirtualBox上のLinuxでサーバ・クライアントの実行環境を構築する
- ssh/sftpを使いリモート環境でプログラムを実行する方法を理解する
- pythonでソケット通信を使った簡単なプログラムを作成する
- パケットキャプチャによってTCPパケットの構造を理解する

環境構築

IPアドレス

- exp1-1 192.168.56.101
- exp1-2 192.168.56.103

ネットワーク図



ルーティングテーブル

[exp1-1]

ネットワーク	ネクストホップ	メトリック
192.168.10.0/24	直接接続	0
192.168.100.0/24	直接接続	0
192.168.20.0/24	192.168.100.20	1

ネットワーク	ネクストホップ	メトリック
192.168.56.0/24	直接接続	0
10.0.4.0/24	192.168.100.20	1

[exp1-2]

ネットワーク	ネクストホップ	メトリック
192.168.20.0/24	直接接続	0
192.168.100.0/24	直接接続	0
192.168.10.0/24	192.168.100.10	1
192.168.56.0/24	直接接続	0
10.0.4.0/24	直接接続	0

プログラム変更差分

```
diff --git a/client/client.py b/client/client.py
index 440ff22..d26ecab 100644
--- a/client/client.py
+++ b/client/client.py
@@ -9,6 +9,9 @@ class EchoClient():
     port = 55556          # サーバのポート番号
     recv_size = 1024     # 受け取るデータの最大サイズ

+   def __init__(self, host=host):
+       self.host = host
+
     def send(self, msg):
         server_address = (self.host, self.port) # サーバのホスト名(IPアドレス)とポート番号

@@ -23,11 +26,14 @@ class EchoClient():
     def main():
         import argparse # コマンドラインパーサを導入
         parser = argparse.ArgumentParser(description='Echo client.')
+
         parser.add_argument('msg', metavar='message', help='message to server') # コマ
         # コマンドライン引数の文字列をmsgという変数に格納
+
         parser.add_argument('--addr', dest='addr', default='127.0.0.1', help='IP
         address')
         args = parser.parse_args() # コマンドラインをパースして、解析し、適宜変数に値を入れる
-        # print(args.msg) # 今回の場合、args.msg にコマンドラインで指定した文字列が格納される
+        # print(args.msg) # 今回の場合、args.msg にコマンドラインで指定した文字列が格納される
+        # print(args.addr)

-        c = EchoClient()
+        c = EchoClient(host=args.addr)
         ret = c.send(args.msg)
```

```
print(ret.decode('utf-8'))

diff --git a/server/server.py b/server/server.py
index b19ad04..de0c078 100644
--- a/server/server.py
+++ b/server/server.py
@@ -19,6 +19,9 @@ class EchoServer:
     port = 55556          # サーバのポート番号
     recv_size = 1024     # 一回に受信するデータの最大バイト数

+   def __init__(self, host=host):
+       self.host = host
+
     def start(self):
         with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
# socket() ソケット作成
            server_socket.bind((self.host, self.port)) # bind() 待ち受け用に設定
@@ -40,8 +43,16 @@ class EchoServer:
            server_socket.close()

def main():
-   s = EchoServer()
+   import argparse # コマンドラインパーサを導入
+   parser = argparse.ArgumentParser(description='Echo client.')
+
+   parser.add_argument('--addr', dest='addr', default='127.0.0.1', help='IP
address')
+   args = parser.parse_args() # コマンドラインをパースして、解析し、適宜変数に値を入れる
+   #print(args.addr)
+
+   s = EchoServer(host=args.addr)
+   s.start()
+

# エントリーポイント
```

動作確認の結果

server(exp1-1)のIP アドレス	説明	接続可否 (192.168.100.10)	接続可否 (192.168.10.1)
192.168.100.10	intnetA側のアダプタのアドレス	可能	不可
192.168.10.1	intnet1側のアダプタのアドレス	不可	不可
127.0.0.1	ループバックアドレス	不可	不可
0.0.0.0	ローカルマシンの全てのIPアドレス	可能	不可

考察

- 192.168.100.10で接続できなかったのは、前回の実験で行った設定ではexp1-1からexp1-2にアクセスをするような設定は行い、exp1-1から`ping -c 5 192.168.20.1`が動くようになったが、exp1-2からexp1-1にアクセスする設定をしていなかったためだと考えられる。そのため、今回の課題ではclient側がexp1-2でserverがexp1-1なのでアクセスできなかった。
- 127.0.0.1で起動してしまうとexp1-1内からしかアクセスできないため動作しなかった。
- 192.168.100.10ではexp1-2のIPアドレスも192.168.100.20となっていて同じネットワークにあるためアクセスできる。0.0.0.0で起動するとローカルマシンのすべてのIPアドレスで待つことになるので、同様に192.
- 168.100.10からアクセスすることができる。

課題2

条件

- wireshark
 - クライアント側(exp1-2)
 - enp0s10
- server
 - 192.168.100.10で起動
 - 0.0.0.0(でき)

シーケンスごとのパケット

1.クライアントからサーバーへの接続要求

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.100.20	192.168.100.10	TCP	74	35532 → 55556 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3524877132 TSecr=0
2	0.000419318	192.168.100.10	192.168.100.20	TCP	74	55556 → 35532 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2092238237
3	0.000478436	192.168.100.20	192.168.100.10	TCP	66	35532 → 55556 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3524877133 TSecr=2092238237
4	0.000670524	192.168.100.20	192.168.100.10	TCP	71	35532 → 55556 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=5 TSval=3524877133 TSecr=2092238237
5	0.001009428	192.168.100.10	192.168.100.20	TCP	66	55556 → 35532 [ACK] Seq=1 Ack=6 Win=65200 Len=0 TSval=2092238238 TSecr=3524877133
6	0.002571162	192.168.100.10	192.168.100.20	TCP	71	55556 → 35532 [PSH, ACK] Seq=1 Ack=6 Win=65200 Len=5 TSval=2092238239 TSecr=3524877133
7	0.002581586	192.168.100.20	192.168.100.10	TCP	66	35532 → 55556 [ACK] Seq=6 Ack=6 Win=64256 Len=0 TSval=3524877135 TSecr=2092238239
8	0.002898439	192.168.100.20	192.168.100.10	TCP	66	35532 → 55556 [FIN, ACK] Seq=6 Ack=6 Win=64256 Len=0 TSval=3524877135 TSecr=2092238239
9	0.003594109	192.168.100.10	192.168.100.20	TCP	66	55556 → 35532 [FIN, ACK] Seq=6 Ack=7 Win=65200 Len=0 TSval=2092238240 TSecr=3524877135
10	0.003602573	192.168.100.20	192.168.100.10	TCP	66	35532 → 55556 [ACK] Seq=7 Ack=7 Win=64256 Len=0 TSval=3524877136 TSecr=2092238240

Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface enp0s8, id 0
Ethernet II, Src: PcsCompu_aa:f7:3f (08:00:27:aa:f7:3f), Dst: PcsCompu_84:b8:a7 (08:00:27:84:b8:a7)
Internet Protocol Version 4, Src: 192.168.100.20, Dst: 192.168.100.10
0100 = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 60
Identification: 0xa757 (42839)
Flags: 0x4000, Don't fragment
Fragment offset: 0
Time to live: 64
Protocol: TCP (6)
Header checksum: 0x49f5 [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.100.20
Destination: 192.168.100.10
Transmission Control Protocol, Src Port: 35532, Dst Port: 55556, Seq: 0, Len: 0
Source Port: 35532
Destination Port: 55556
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number)
Sequence number (raw): 843590168
[Next sequence number: 1 (relative sequence number)]
Acknowledgment number: 0
Acknowledgment number (raw): 0
1010 = Header Length: 40 bytes (10)
Flags: 0x002 [SYN]
Window size value: 64240
[Calculated window size: 64240]
Checksum: 0x499e [unverified]
[Checksum Status: Unverified]

0000 08 00 27 84 b8 a7 08 00 27 aa f7 3f 08 00 45 00 ..?...!...?..E..
0010 00 3c a7 57 40 00 00 06 49 f5 c0 a8 64 14 c0 a8 <..W..@..I...d..
0020 64 0a 8a cc d9 04 32 48 2a 18 00 00 00 00 a0 02 d....2H*.....
0030 fa f0 49 9e 00 00 02 04 05 b4 04 02 08 0a d2 19 ..I.....
0040 5b 4c 00 00 00 00 01 03 03 07 [L.....

- 対応する部分
 - client.py 19行目

```
client_socket.connect(server_address) # connect() 接続
```

- server.cpp 47行目

```
retCode = listen(fdAccept, backlog);
```

- サーバー側で接続要求を受け取る

2.サーバーからクライアントへの接続許可

The image shows a Wireshark packet capture analysis of a TCP connection. The top pane displays a list of packets, with packet 2 selected. The middle pane shows the packet details for the selected packet, including Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The bottom pane shows the raw packet data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.100.20	192.168.100.10	TCP	74	35532 → 55556 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3524877132 TSecr=0...
2	0.000419318	192.168.100.10	192.168.100.20	TCP	74	55556 → 35532 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2092238237...
3	0.000478436	192.168.100.20	192.168.100.10	TCP	66	35532 → 55556 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3524877133 TSecr=2092238237
4	0.000670524	192.168.100.20	192.168.100.10	TCP	71	35532 → 55556 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=5 TSval=3524877133 TSecr=2092238237
5	0.001009428	192.168.100.10	192.168.100.20	TCP	66	55556 → 35532 [ACK] Seq=1 Ack=6 Win=65280 Len=0 TSval=2092238238 TSecr=3524877133
6	0.002571162	192.168.100.10	192.168.100.20	TCP	71	55556 → 35532 [PSH, ACK] Seq=1 Ack=6 Win=65280 Len=5 TSval=2092238239 TSecr=3524877133
7	0.002581586	192.168.100.20	192.168.100.10	TCP	66	35532 → 55556 [ACK] Seq=6 Ack=6 Win=64256 Len=0 TSval=3524877135 TSecr=2092238239
8	0.002808439	192.168.100.20	192.168.100.10	TCP	66	35532 → 55556 [FIN, ACK] Seq=6 Ack=6 Win=64256 Len=0 TSval=3524877135 TSecr=2092238239
9	0.003594109	192.168.100.10	192.168.100.20	TCP	66	55556 → 35532 [FIN, ACK] Seq=6 Ack=7 Win=65280 Len=0 TSval=2092238240 TSecr=3524877135
10	0.003602573	192.168.100.20	192.168.100.10	TCP	66	35532 → 55556 [ACK] Seq=7 Ack=7 Win=64256 Len=0 TSval=3524877136 TSecr=2092238240

Frame 2: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface enp0s8, id 0
 Ethernet II, Src: PcsCompu_84:b8:a7 (08:00:27:84:b8:a7), Dst: PcsCompu_aa:f7:3f (08:00:27:aa:f7:3f)
 Internet Protocol Version 4, Src: 192.168.100.10, Dst: 192.168.100.20
 0100 = Version: 4
 0101 = Header Length: 20 bytes (5)
 Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 Total Length: 60
 Identification: 0x0000 (0)
 Flags: 0x4000, Don't fragment
 Fragment offset: 0
 Time to live: 64
 Protocol: TCP (6)
 Header checksum: 0xf14c [validation disabled]
 [Header checksum status: Unverified]
 Source: 192.168.100.10
 Destination: 192.168.100.20
 Transmission Control Protocol, Src Port: 55556, Dst Port: 35532, Seq: 0, Ack: 1, Len: 0
 Source Port: 55556
 Destination Port: 35532
 [Stream index: 0]
 [TCP Segment Len: 0]
 Sequence number: 0 (relative sequence number)
 Sequence number (raw): 4275229138
 [Next sequence number: 1 (relative sequence number)]
 Acknowledgment number: 1 (relative ack number)
 Acknowledgment number (raw): 843590169
 1010 = Header Length: 40 bytes (10)
 Flags: 0x012 (SYN, ACK)
 Window size value: 65160
 [Calculated window size: 65160]
 Checksum: 0xbf66 [unverified]
 [Checksum Status: Unverified]

```

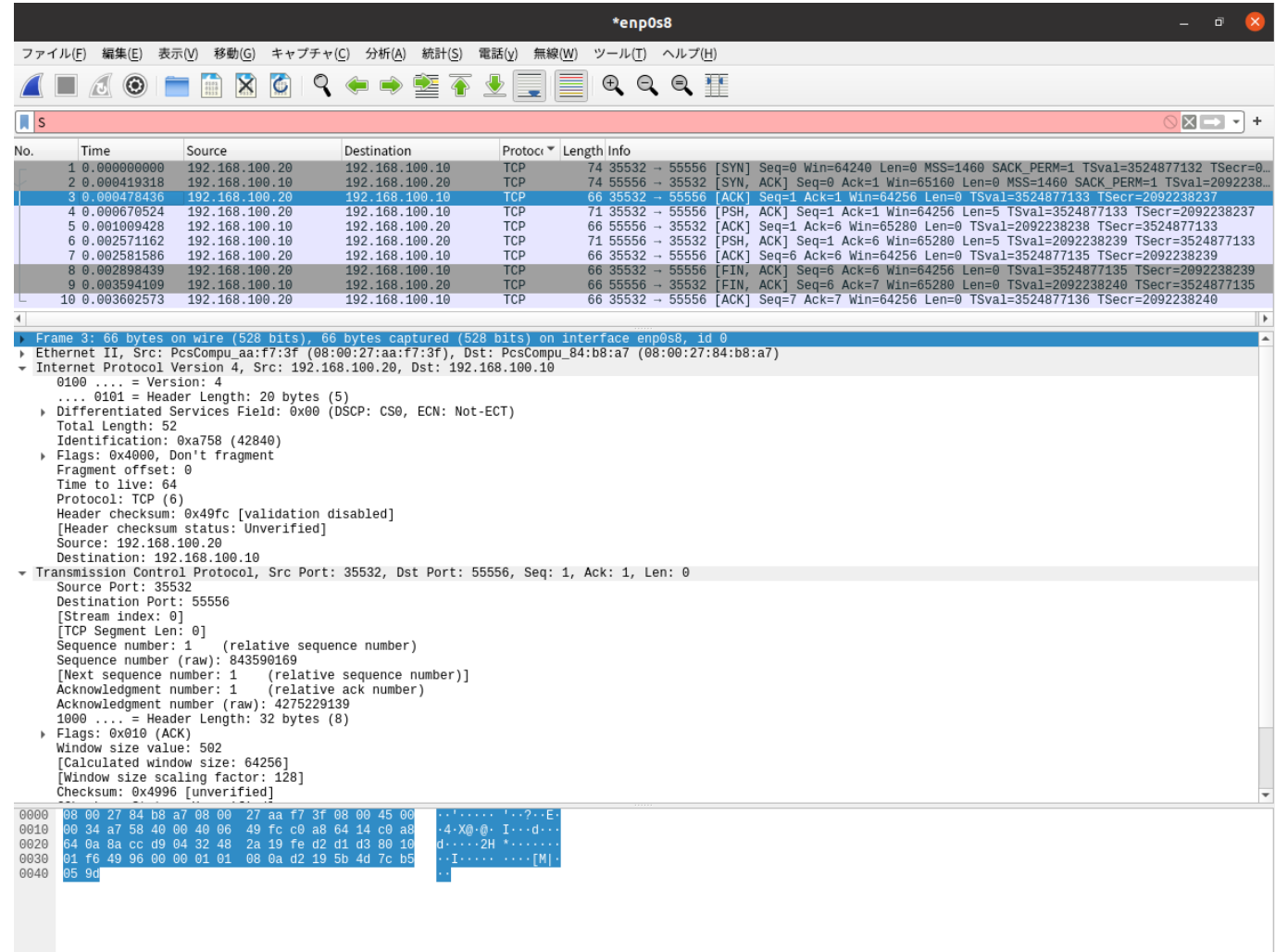
0000 08 00 27 aa f7 3f 08 00 27 84 b8 a7 08 00 45 00  ..'.?..'....E.
0010 00 3c 00 00 40 00 40 06 f1 4c c0 a8 64 0a c0 a8  <...@...L..d..
0020 64 14 d9 04 8a cc fe d2 d1 d2 32 48 2a 19 a0 12  d.....2H*...
0030 fe 88 bf 66 00 00 02 04 05 b4 04 02 08 0a 7c b5  ...f.....|..
0040 05 9d d2 19 5b 4c 01 03 03 07  ....[L...

```

- 対応する部分
 - server.cpp 56行目

```
fd0ther = accept(fdAccept, (struct sockaddr *)&sin_client,
&socklen);
```

3.接続確認



- サーバーと正しく接続できたことをクライアントから確認してる部分。

4. メッセージの送信

*enp0s8

ファイル(F) 編集(E) 表示(V) 移動(G) キャプチャ(C) 分析(A) 統計(S) 電話(y) 無線(W) ツール(T) ヘルプ(H)

SS

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.100.20	192.168.100.10	TCP	74	35532 → 55556 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3524877132 TSecr=0
2	0.000419318	192.168.100.10	192.168.100.20	TCP	74	55556 → 35532 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2092238238
3	0.000478436	192.168.100.20	192.168.100.10	TCP	66	35532 → 55556 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3524877133 TSecr=2092238237
4	0.000678524	192.168.100.20	192.168.100.10	TCP	71	35532 → 55556 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=5 TSval=3524877133 TSecr=2092238237
5	0.001009428	192.168.100.10	192.168.100.20	TCP	66	55556 → 35532 [ACK] Seq=1 Ack=6 Win=65280 Len=0 TSval=2092238238 TSecr=3524877133
6	0.002571162	192.168.100.10	192.168.100.20	TCP	71	55556 → 35532 [PSH, ACK] Seq=1 Ack=6 Win=65280 Len=5 TSval=2092238239 TSecr=3524877133
7	0.002581586	192.168.100.20	192.168.100.10	TCP	66	35532 → 55556 [ACK] Seq=6 Ack=6 Win=64256 Len=0 TSval=3524877135 TSecr=2092238239
8	0.002898439	192.168.100.20	192.168.100.10	TCP	66	35532 → 55556 [FIN, ACK] Seq=6 Ack=6 Win=64256 Len=0 TSval=3524877135 TSecr=2092238239
9	0.003594109	192.168.100.10	192.168.100.20	TCP	66	55556 → 35532 [FIN, ACK] Seq=6 Ack=7 Win=65280 Len=0 TSval=2092238240 TSecr=3524877135
10	0.003602573	192.168.100.20	192.168.100.10	TCP	66	35532 → 55556 [ACK] Seq=7 Ack=7 Win=64256 Len=0 TSval=3524877136 TSecr=2092238240

Frame 4: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface enp0s8, id 0

Ethernet II, Src: PcsCompu_aa:f7:3f (08:00:27:aa:f7:3f), Dst: PcsCompu_84:b8:a7 (08:00:27:84:b8:a7)

Internet Protocol Version 4, Src: 192.168.100.20, Dst: 192.168.100.10

0100 = Version: 4

... 0101 = Header Length: 20 bytes (5)

Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 57

Identification: 0xa759 (42841)

Flags: 0x4000, Don't fragment

Fragment offset: 0

Time to live: 64

Protocol: TCP (6)

Header checksum: 0x49f6 [validation disabled]

[Header checksum status: Unverified]

Source: 192.168.100.20

Destination: 192.168.100.10

Transmission Control Protocol, Src Port: 35532, Dst Port: 55556, Seq: 1, Ack: 1, Len: 5

Source Port: 35532

Destination Port: 55556

[Stream index: 0]

[TCP Segment Len: 5]

Sequence number: 1 (relative sequence number)

Sequence number (raw): 843590169

[Next sequence number: 6 (relative sequence number)]

Acknowledgment number: 1 (relative ack number)

Acknowledgment number (raw): 4275229139

1000 = Header Length: 32 bytes (8)

Flags: 0x018 (PSH, ACK)

Window size value: 502

[Calculated window size: 64256]

[Window size scaling factor: 128]

Checksum: 0x499b [unverified]

0000 08 00 27 84 b8 a7 08 00 27 aa f7 3f 08 00 45 00 ..?..E..

0010 00 39 a7 59 40 00 40 06 49 f6 c0 a8 64 14 c0 a8 ..9.Y@.I..d..

0020 64 0a 8a cc d0 04 32 48 2a 19 fe d2 d1 d3 80 18 d.....2H.....

0030 01 f6 49 9b 00 00 01 01 08 0a d2 19 5b 4d 7c b5 ..I.....[M]..

0040 95 9d 68 65 6c 6c 6fhello

- メッセージとして送ったhelloがあることIPアドレスからクライアントからサーバーへ送信している部分とわかる。
- 対応する部分
 - client.py 20行目

```
client_socket.send(msg.encode('utf-8')) # send() 送信
```

5. 送信確認

*enp0s8

ファイル(F) 編集(E) 表示(V) 移動(G) キャプチャ(C) 分析(A) 統計(S) 電話(Y) 無線(W) ツール(T) ヘルプ(H)

SS

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.100.20	192.168.100.10	TCP	74	36958 → 55556 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3525414010 TSecr=0...
2	0.000768953	192.168.100.10	192.168.100.20	TCP	74	55556 → 36958 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2092775...
3	0.001012480	192.168.100.20	192.168.100.10	TCP	66	36958 → 55556 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3525414011 TSecr=2092775115
4	0.001291537	192.168.100.20	192.168.100.10	TCP	71	36958 → 55556 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=5 TSval=3525414011 TSecr=2092775115
5	0.002517651	192.168.100.10	192.168.100.20	TCP	66	55556 → 36958 [ACK] Seq=1 Ack=6 Win=65280 Len=0 TSval=2092775116 TSecr=3525414011
6	0.002517712	192.168.100.10	192.168.100.20	TCP	71	55556 → 36958 [PSH, ACK] Seq=1 Ack=6 Win=65280 Len=5 TSval=2092775117 TSecr=3525414011
7	0.002530459	192.168.100.20	192.168.100.10	TCP	66	36958 → 55556 [ACK] Seq=6 Ack=6 Win=64256 Len=0 TSval=3525414012 TSecr=2092775117
8	0.002815271	192.168.100.20	192.168.100.10	TCP	66	36958 → 55556 [FIN, ACK] Seq=6 Ack=6 Win=64256 Len=0 TSval=3525414013 TSecr=2092775117
9	0.003359589	192.168.100.10	192.168.100.20	TCP	66	55556 → 36958 [FIN, ACK] Seq=6 Ack=7 Win=65280 Len=0 TSval=2092775118 TSecr=3525414013
10	0.003367557	192.168.100.20	192.168.100.10	TCP	66	36958 → 55556 [ACK] Seq=7 Ack=7 Win=64256 Len=0 TSval=3525414013 TSecr=2092775118

Frame 5: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface enp0s8, id 0

Ethernet II, Src: PcsCompu_84:b8:a7 (08:00:27:84:b8:a7), Dst: PcsCompu_aa:f7:3f (08:00:27:aa:f7:3f)

Internet Protocol Version 4, Src: 192.168.100.10, Dst: 192.168.100.20

0100 = Version: 4

... 0101 = Header Length: 20 bytes (5)

Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 52

Identification: 0xc453 (50259)

Flags: 0x4000, Don't Fragment

Fragment offset: 0

Time to live: 64

Protocol: TCP (6)

Header checksum: 0x2d01 [validation disabled]

[Header checksum status: Unverified]

Source: 192.168.100.10

Destination: 192.168.100.20

Transmission Control Protocol, Src Port: 55556, Dst Port: 36958, Seq: 1, Ack: 6, Len: 0

Source Port: 55556

Destination Port: 36958

[Stream index: 0]

[TCP Segment Len: 0]

Sequence number: 1 (relative sequence number)

Sequence number (raw): 1472883988

[Next sequence number: 1 (relative sequence number)]

Acknowledgment number: 6 (relative ack number)

Acknowledgment number (raw): 3665610756

1000 = Header Length: 32 bytes (8)

Flags: 0x010 (ACK)

Window size value: 510

[Calculated window size: 65280]

[Window size scaling factor: 128]

Checksum: 0x4865 [unverified]

[Checksum Status: Unverified]

Urgent pointer: 0

Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps

[SEQ/ACK analysis]

[Timestamps]

0000 08 00 27 aa f7 3f 08 00 27 84 b8 a7 08 00 45 00 ...?...'...E...

0010 00 34 c4 53 40 00 40 0c 2d 01 c0 a8 64 0a c0 a8 ...4.SQ.0...-...d...

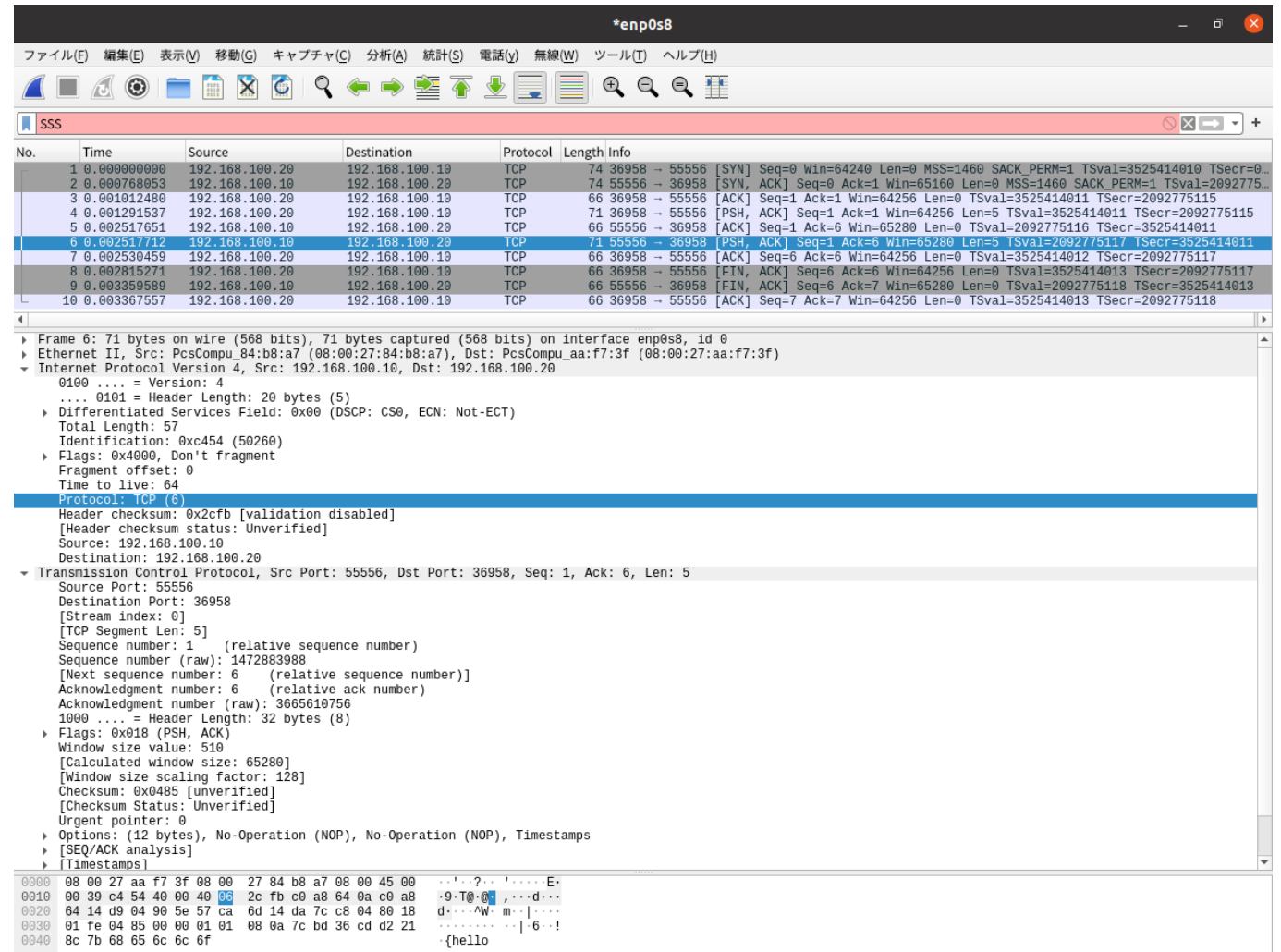
0020 64 14 d9 04 90 5e 57 ca 6d 14 da 7c c8 04 80 10 ...d...AW...m...|....

0030 01 fe 48 65 00 00 01 01 08 0a 7c bd 36 cc d2 21 ...He.....|'6...!

0040 8c 7b ...{

- サーバー側がクライアントからのメッセージを受け取ったことを確認するため送り返している。

6. メッセージの受信



- メッセージとして送ったhelloがあることとIPアドレスからサーバーからクライアントへ送信している部分とわかる。
- 対応する部分
 - server.cpp 72行目

```
if (send(fdOther, buf, recvMsgSize, 0) != recvMsgSize){
```

7. 受信確認

*enp0s8

ファイル(F) 編集(E) 表示(V) 移動(G) キャプチャ(C) 分析(A) 統計(S) 電話(Y) 無線(W) ツール(T) ヘルプ(H)

SSS

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.100.20	192.168.100.10	TCP	74	36958 → 55556 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3525414010 TSecr=0
2	0.000768053	192.168.100.10	192.168.100.20	TCP	74	55556 → 36958 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2092775115 TSecr=3525414010
3	0.001012480	192.168.100.20	192.168.100.10	TCP	66	36958 → 55556 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3525414011 TSecr=2092775115
4	0.001291537	192.168.100.20	192.168.100.10	TCP	71	36958 → 55556 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=5 TSval=3525414011 TSecr=2092775115
5	0.002517651	192.168.100.10	192.168.100.20	TCP	66	55556 → 36958 [ACK] Seq=1 Ack=6 Win=65280 Len=0 TSval=2092775116 TSecr=3525414011
6	0.002517712	192.168.100.10	192.168.100.20	TCP	71	55556 → 36958 [PSH, ACK] Seq=1 Ack=6 Win=65280 Len=5 TSval=2092775117 TSecr=3525414011
7	0.002530459	192.168.100.20	192.168.100.10	TCP	66	36958 → 55556 [ACK] Seq=6 Ack=6 Win=64256 Len=0 TSval=3525414012 TSecr=2092775117
8	0.002815271	192.168.100.20	192.168.100.10	TCP	66	36958 → 55556 [FIN, ACK] Seq=6 Ack=6 Win=64256 Len=0 TSval=3525414013 TSecr=2092775117
9	0.003359589	192.168.100.10	192.168.100.20	TCP	66	55556 → 36958 [FIN, ACK] Seq=6 Ack=7 Win=65280 Len=0 TSval=2092775118 TSecr=3525414013
10	0.003367557	192.168.100.20	192.168.100.10	TCP	66	36958 → 55556 [ACK] Seq=7 Ack=7 Win=64256 Len=0 TSval=3525414013 TSecr=2092775118

Frame 7: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface enp0s8, id 0

Ethernet II, Src: PcsCompu_aa:f7:3f (08:00:27:aa:f7:3f), Dst: PcsCompu_84:b8:a7 (08:00:27:84:b8:a7)

Internet Protocol Version 4, Src: 192.168.100.20, Dst: 192.168.100.10

0100 = Version: 4

.... 0101 = Header Length: 20 bytes (5)

Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 52

Identification: 0x65e9 (26089)

Flags: 0x4000, Don't fragment

Fragment offset: 0

Time to live: 64

Protocol: TCP (6)

Header checksum: 0x8b6b [validation disabled]

[Header checksum status: Unverified]

Source: 192.168.100.20

Destination: 192.168.100.10

Transmission Control Protocol, Src Port: 36958, Dst Port: 55556, Seq: 6, Ack: 6, Len: 0

Source Port: 36958

Destination Port: 55556

[Stream index: 0]

[TCP Segment Len: 0]

Sequence number: 6 (relative sequence number)

Sequence number (raw): 3665610756

[Next sequence number: 6 (relative sequence number)]

Acknowledgment number: 6 (relative ack number)

Acknowledgment number (raw): 1472883993

1000 = Header Length: 32 bytes (8)

Flags: 0x010 (ACK)

Window size value: 502

[Calculated window size: 64256]

[Window size scaling factor: 128]

Checksum: 0x4996 [unverified]

[Checksum Status: Unverified]

Urgent pointer: 0

Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps

[SEQ/ACK analysis]

[Timestamps]

0000 08 00 27 84 b8 a7 08 00 27 aa f7 3f 08 00 45 00 ..?..E.

0010 00 34 65 e9 40 00 40 00 8b 6b c0 a8 64 14 c0 a8 .4e-@.0.k..d...

0020 64 0a 90 5e d9 04 da 7c c8 04 57 ca 6d 19 80 10 d..^...|..W.m...

0030 01 f6 49 96 00 00 01 01 08 0a d2 21 8c 7c 7c bd .I.....!||..

0040 36 cd 6.

- クライアント側がサーバー側からのメッセージを受け取ったことを確認するため送り返している。

8. 接続終了要求

*enp0s8

ファイル(F) 編集(E) 表示(V) 移動(G) キャプチャ(C) 分析(A) 統計(S) 電話(V) 無線(W) ツール(T) ヘルプ(H)

SSSS

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.100.20	192.168.100.10	TCP	74	36958 → 55556 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3525414010 TSecr=0...
2	0.000768053	192.168.100.10	192.168.100.20	TCP	74	55556 → 36958 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2092775...
3	0.001012480	192.168.100.20	192.168.100.10	TCP	66	36958 → 55556 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3525414011 TSecr=2092775115
4	0.001291537	192.168.100.20	192.168.100.10	TCP	71	36958 → 55556 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=5 TSval=3525414011 TSecr=2092775115
5	0.002517651	192.168.100.10	192.168.100.20	TCP	66	55556 → 36958 [ACK] Seq=1 Ack=6 Win=65280 Len=0 TSval=2092775116 TSecr=3525414011
6	0.002517712	192.168.100.10	192.168.100.20	TCP	71	55556 → 36958 [PSH, ACK] Seq=1 Ack=6 Win=65280 Len=5 TSval=2092775117 TSecr=3525414011
7	0.002530459	192.168.100.20	192.168.100.10	TCP	66	36958 → 55556 [ACK] Seq=6 Ack=6 Win=64256 Len=0 TSval=3525414012 TSecr=2092775117
8	0.002815271	192.168.100.20	192.168.100.10	TCP	66	36958 → 55556 [FIN, ACK] Seq=6 Ack=6 Win=64256 Len=0 TSval=3525414013 TSecr=2092775117
9	0.003359589	192.168.100.10	192.168.100.20	TCP	66	55556 → 36958 [FIN, ACK] Seq=6 Ack=7 Win=65280 Len=0 TSval=2092775118 TSecr=3525414013
10	0.003367557	192.168.100.20	192.168.100.10	TCP	66	36958 → 55556 [ACK] Seq=7 Ack=7 Win=64256 Len=0 TSval=3525414013 TSecr=2092775118

Frame 8: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface enp0s8, id 0

Ethernet II, Src: PcsCompu_aa:f7:3f (08:00:27:aa:f7:3f), Dst: PcsCompu_84:b8:a7 (08:00:27:84:b8:a7)

Internet Protocol Version 4, Src: 192.168.100.20, Dst: 192.168.100.10

0100 = Version: 4

.... 0101 = Header Length: 20 bytes (5)

Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 52

Identification: 0x65ea (26090)

Flags: 0x4000, Don't fragment

Fragment offset: 0

Time to live: 64

Protocol: TCP (6)

Header checksum: 0x8b6a [validation disabled]

[Header checksum status: Unverified]

Source: 192.168.100.20

Destination: 192.168.100.10

Transmission Control Protocol, Src Port: 36958, Dst Port: 55556, Seq: 6, Ack: 6, Len: 0

Source Port: 36958

Destination Port: 55556

[Stream index: 0]

[TCP Segment Len: 0]

Sequence number: 6 (relative sequence number)

Sequence number (raw): 3665610756

[Next sequence number: 7 (relative sequence number)]

Acknowledgment number: 6 (relative ack number)

Acknowledgment number (raw): 1472883993

1000 = Header Length: 32 bytes (8)

Flags: 0x011 (FIN, ACK)

Window size value: 502

[Calculated window size: 64256]

[Window size scaling factor: 128]

Checksum: 0x4996 [unverified]

[Checksum Status: Unverified]

Urgent pointer: 0

Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps

[Timestamps]

0000 08 00 27 84 b8 a7 08 00 27 aa f7 3f 08 00 45 00 ..'.....?..E.

0010 00 34 65 ea 40 00 40 00 8b 6a c0 a8 64 14 c0 a8 ..4e-@-@-j--d...

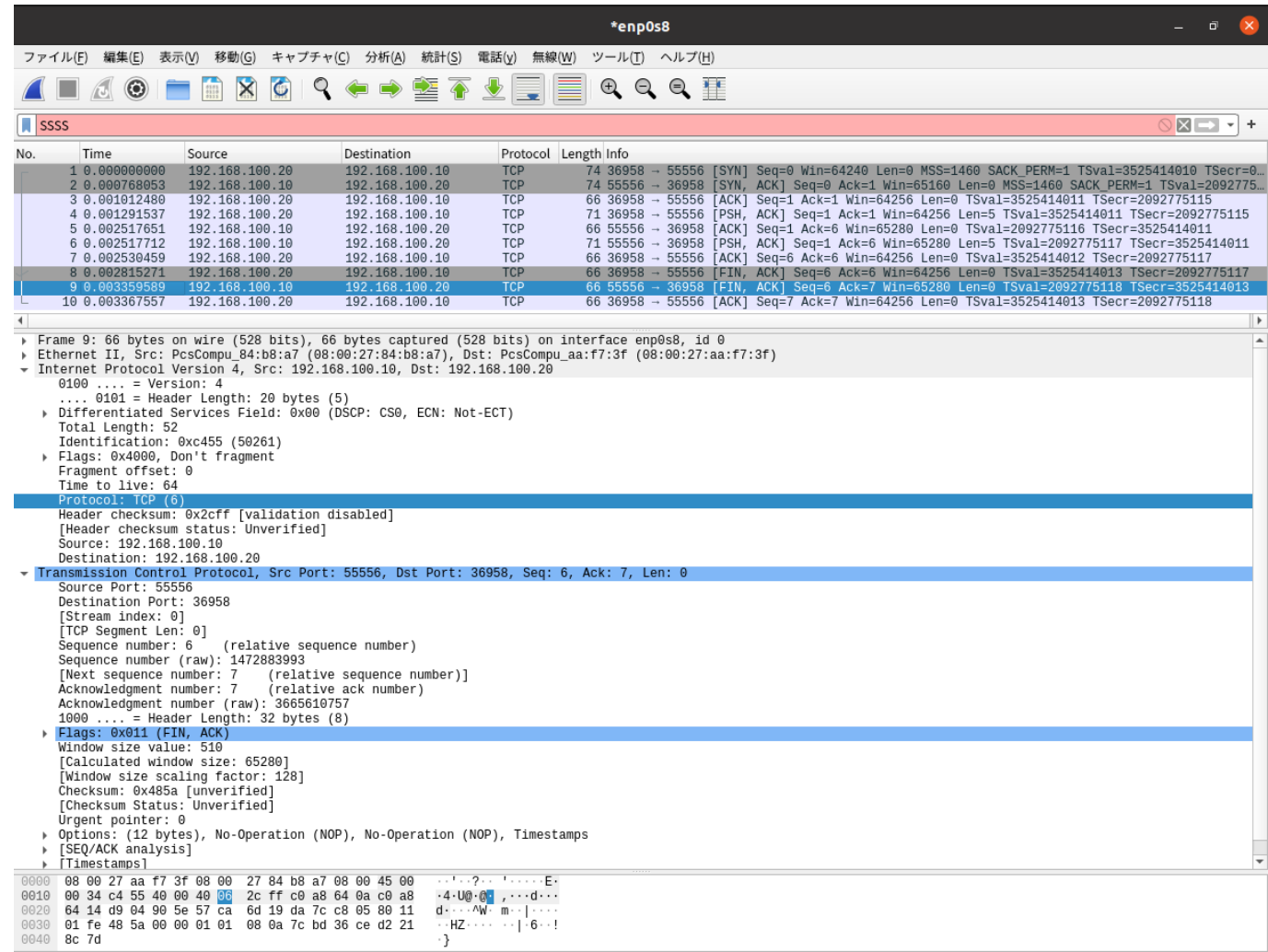
0020 64 0a 90 5e d9 04 da 7c c8 04 57 ca 6d 19 80 11 d-^...|..W.m...

0030 01 f6 49 96 00 00 01 01 08 0a d2 21 8c 7d 7c bd ..I-....-!..}|-

0040 36 cd 6

- 対応する部分
 - プログラムの終了

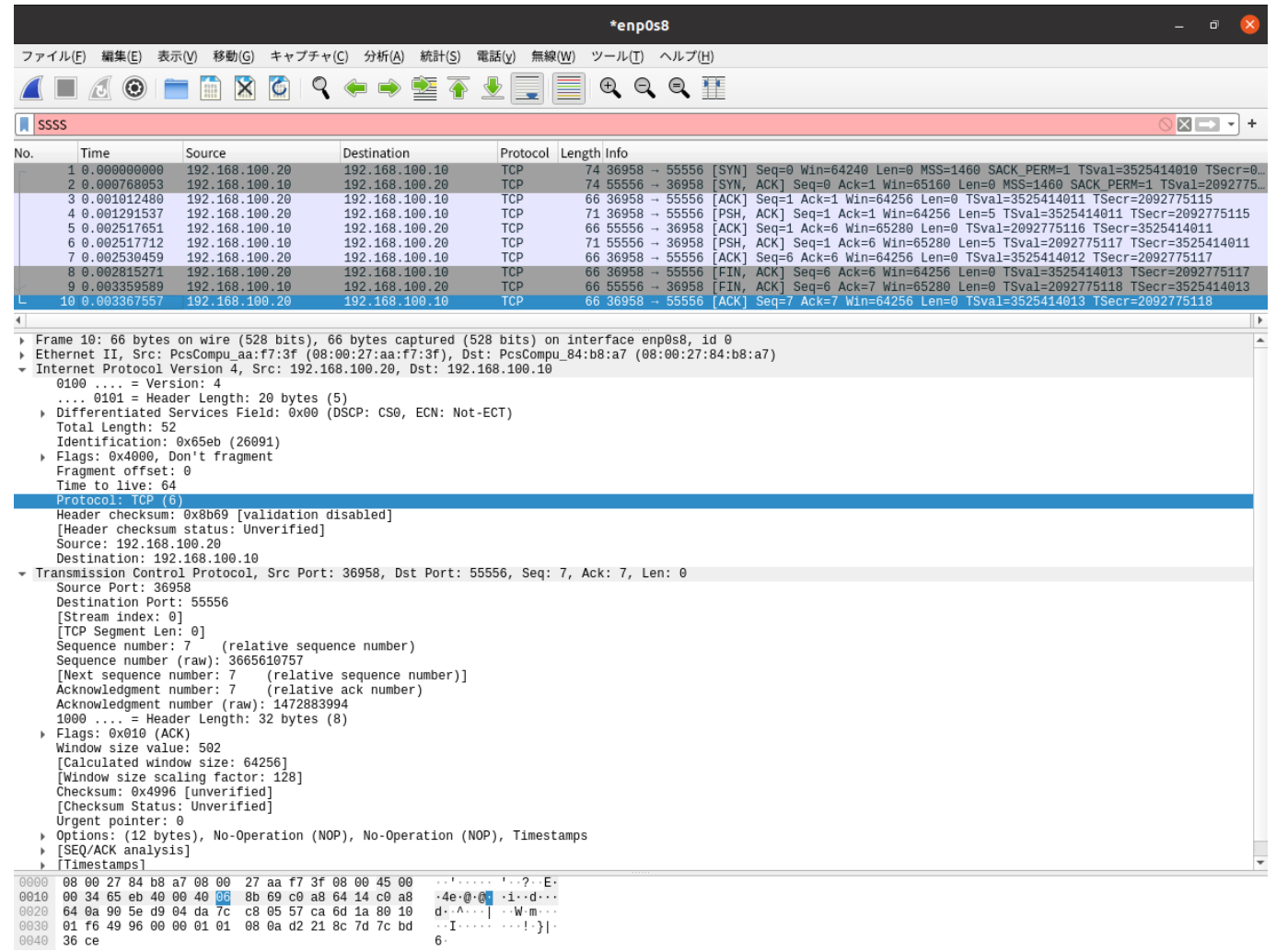
9. 接続終了



- 対応する部分
 - server.py 43行目

```
server_socket.close()
```

10. 接続終了確認



課題3

プログラム変更差分

```
diff --git a/client/client.py b/client/client.py
index d26ecab..9efafe6 100644
--- a/client/client.py
+++ b/client/client.py
@@ -9,8 +9,9 @@ class EchoClient():
     port = 55556          # サーバのポート番号
     recv_size = 1024     # 受け取るデータの最大サイズ

-    def __init__(self, host=host):
+    def __init__(self, host=host, port=port):
         self.host = host
+        self.port = port

     def send(self, msg):
         server_address = (self.host, self.port) # サーバのホスト名(IPアドレス)とポート番号
@@ -29,11 +30,12 @@ def main():

     parser.add_argument('msg', metavar='messege', help='message to server') # コマンドライン引数の文字列をmsgという変数に格納
     parser.add_argument('--addr', dest='addr', default='127.0.0.1', help='IP
```

```

address')
+   parser.add_argument('--port', dest='port', default=55556, type=int,
help='port number')
    args = parser.parse_args() # コマンドラインをパースして、解析し、適宜変数に値を入れる
    #print(args.msg) # 今回の場合、args.msg にコマンドラインで指定した文字列が格納される
    #print(args.addr)

-   c = EchoClient(host=args.addr)
+   c = EchoClient(host=args.addr, port=args.port)
    ret = c.send(args.msg)
    print(ret.decode('utf-8'))

diff --git a/server/server.py b/server/server.py
index de0c078..24c4af5 100644
--- a/server/server.py
+++ b/server/server.py
@@ -19,8 +19,9 @@ class EchoServer:
    port = 55556          # サーバのポート番号
    recv_size = 1024     # 一回に受信するデータの最大バイト数

-   def __init__(self, host=host):
+   def __init__(self, host=host, port=port):
        self.host = host
+        self.port = port

    def start(self):
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
# socket() ソケット作成
@@ -45,12 +46,12 @@ class EchoServer:
    def main():
        import argparse # コマンドラインパーサを導入
        parser = argparse.ArgumentParser(description='Echo client.')
-
        parser.add_argument('--addr', dest='addr', default='127.0.0.1', help='IP
address')
+   parser.add_argument('--port', dest='port', default=55556, type=int,
help='port number')
        args = parser.parse_args() # コマンドラインをパースして、解析し、適宜変数に値を入れる
        #print(args.addr)

-   s = EchoServer(host=args.addr)
+   s = EchoServer(host=args.addr, port=args.port)
    s.start()

```

動作確認の方法と結果

- サーバー側で以下のように設定した。
 - ターミナル1

```
python3 server.py --addr 192.168.100.10 --port 55557
```


- ターミナル2

```
python3 server.py --addr 192.168.100.10 --port 55555
```

- クライアント側で以下のコマンドを実行した。

- ターミナル1

```
python3 client.py --addr 192.168.100.10 --port 55557 hello1
```

- ターミナル2

```
python3 client.py --addr 192.168.100.10 --port 55555 hello2
```

- 実行結果

- サーバー側

- ターミナル1

```
DEBUG:__main__:Connected by (192.168.100.20, 49272)
DEBUG:__main__:Recieve: hello1
DEBUG:__main__:Send: hello1
```

- ターミナル2

```
DEBUG:__main__:Connected by (192.168.100.20, 36660)
DEBUG:__main__:Recieve: hello2
DEBUG:__main__:Send: hello2
```

- クライアント側

- ターミナル1

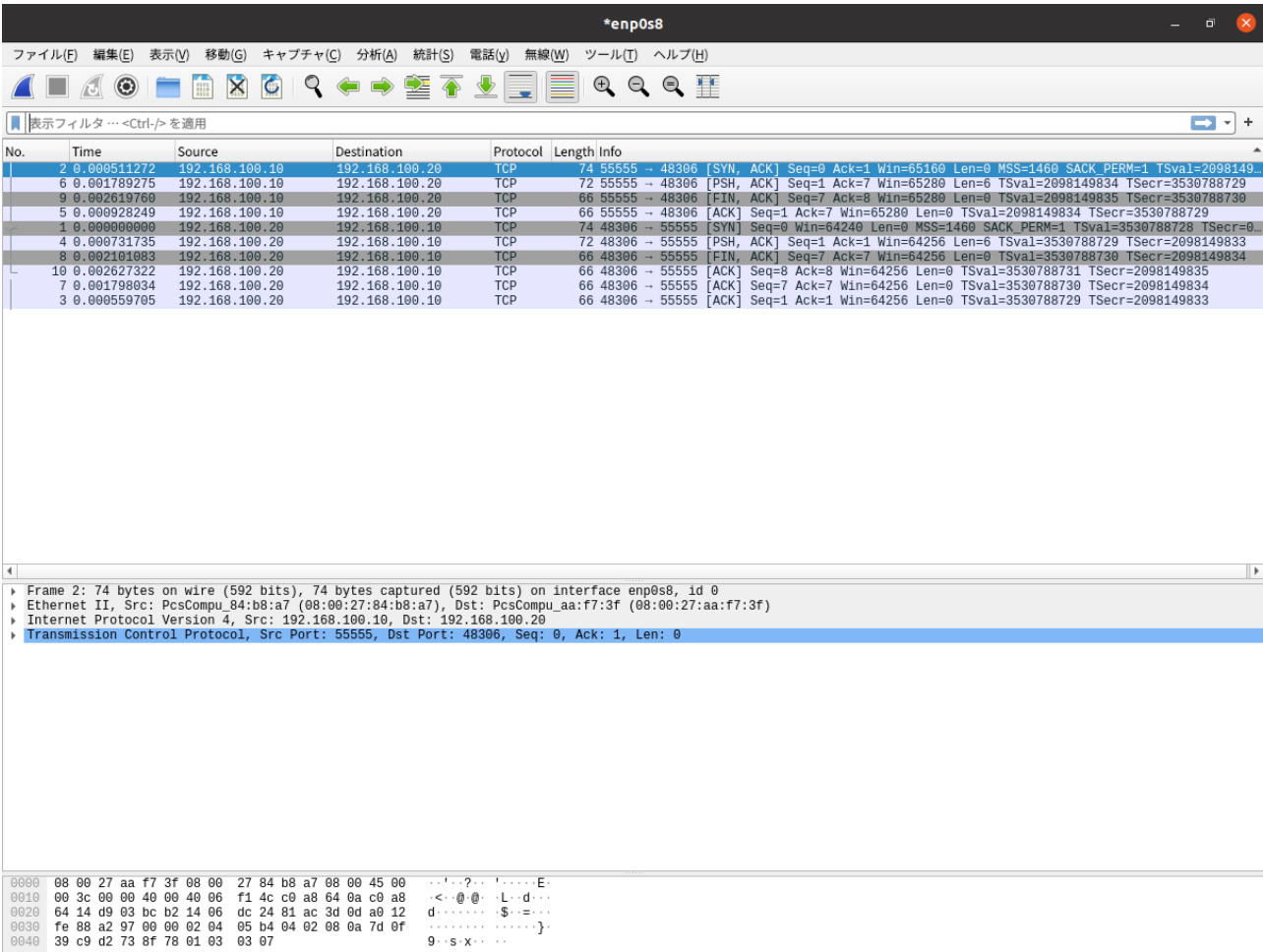
```
hello1
```

- ターミナル2

```
hello2
```

パケットキャプチャの状態と結果

- ターミナル2



考察

ポートを違うもので待ち受けると、ポートごとにそれぞれに別に通信することができることがわかる。また、パケットキャプチャを見ると、TCPのプロトコルの情報にポート番号が正しく記録されていることがわかり、サーバーの送り先のポート番号が異なると、クライアント側も違うポート番号を使用してパケットを送信している。

課題4

プログラム変更差分

```
diff --git a/client/client.py b/client/client.py
index 9efafe6..5655fe8 100644
--- a/client/client.py
+++ b/client/client.py
@@ -9,35 +9,31 @@ class EchoClient():
     port = 55556          # サーバのポート番号
     recv_size = 1024     # 受け取るデータの最大サイズ

-    def __init__(self, host=host, port=port):
-        self.host = host
+    def __init__(self, port=port):
+        self.port = port
```

```

-     def send(self, msg):
+         self.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
+         server_address = (self.host, self.port) # サーバのホスト名(IPアドレス)とポート番号
+         self.client_socket.connect(server_address) # connect() 接続
+
+     def send(self, msg):
+         self.client_socket.send(msg.encode('utf-8')) # send() 送信
+         data = self.client_socket.recv(self.recv_size) # recv() 受信

-         with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client_socket:
# socket()ソケット生成
-             client_socket.connect(server_address) # connect() 接続
-             client_socket.send(msg.encode('utf-8')) # send() 送信
-             data = client_socket.recv(self.recv_size) # recv() 受信
-
-             return data
+         return data

def main():
    import argparse # コマンドラインパーサを導入
    parser = argparse.ArgumentParser(description='Echo client.')

-     parser.add_argument('msg', metavar='messege', help='message to server') # コマ
ンドライン引数の文字列をmsgという変数に格納
-     parser.add_argument('--addr', dest='addr', default='127.0.0.1', help='IP
address')
        parser.add_argument('--port', dest='port', default=55556, type=int,
help='port number')
        args = parser.parse_args() # コマンドラインをパースして、解析し、適宜変数に値を入れる
-     #print(args.msg) # 今回の場合、args.msg にコマンドラインで指定した文字列が格納される
-     #print(args.addr)
-
-     c = EchoClient(host=args.addr, port=args.port)
-     ret = c.send(args.msg)
-     print(ret.decode('utf-8'))
+
+     c = EchoClient(port=args.port)
+     for line in sys.stdin:
+         ret = c.send(line)
+         print(ret.decode('utf-8'))

# エントリーポイント
if __name__ == '__main__':

diff --git a/server/server.py b/server/server.py
index 24c4af5..de0c078 100644
--- a/server/server.py
+++ b/server/server.py
@@ -19,9 +19,8 @@ class EchoServer:
    port = 55556 # サーバのポート番号
    recv_size = 1024 # 一回に受信するデータの最大バイト数

```

```

-     def __init__(self, host=host, port=port):
+     def __init__(self, host=host):
            self.host = host
-            self.port = port

            def start(self):
                with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
# socket() ソケット作成
@@ -46,12 +45,12 @@ class EchoServer:
    def main():
        import argparse # コマンドラインパーサを導入
        parser = argparse.ArgumentParser(description='Echo client.')
+
        parser.add_argument('--addr', dest='addr', default='127.0.0.1', help='IP
address')
-    parser.add_argument('--port', dest='port', default=55556, type=int,
help='port number')
        args = parser.parse_args() # コマンドラインをパースして、解析し、適宜変数に値を入れる
        #print(args.addr)

-    s = EchoServer(host=args.addr, port=args.port)
+    s = EchoServer(host=args.addr)
        s.start()

```

動作確認の結果

- サーバー側

```

ubuntu@ubuntu-VirtualBox:~/server$ python3 server.py --addr 192.168.100.10 --port
55555
DEBUG:__main__:Connected by (192.168.100.20, 37094)
DEBUG:__main__:Recieve: abcdefg

DEBUG:__main__:Send: abcdefg

DEBUG:__main__:Recieve: ABCDEFG

DEBUG:__main__:Send: ABCDEFG

```

- クライアント側

```

ubuntu@ubuntu-VirtualBox:~/client$ python3 client.py --port 55555
abcdefg
abcdefg

ABCEFG
ABCEFG

```

以上から正しく行ごとに送受信ができてることがわかる。

考察

この課題から、ソケットを生成してからソケットを閉じるまでならいくつパケットを送信しても正しく動作することがわかった。

git履歴

```
commit 62e42f33f064a6fbd9f9e789c999530b68584ff9 (HEAD -> main, origin/main, origin/HEAD)
```

```
Author: r05i42 <r05i42@ed.cc.suzuka-ct.ac.jp>
```

```
Date: Tue Jul 1 17:06:17 2025 +0900
```

```
tag: release-ex4
```

```
commit 24a72fe280957eca3b1468368dbd249ce17dfa36
```

```
Author: r05i42 <r05i42@ed.cc.suzuka-ct.ac.jp>
```

```
Date: Tue Jul 1 16:17:19 2025 +0900
```

```
tag: release-ex3
```

```
commit 56a57bac1fc7c1761a2aa3967808a1b8adc80ad9 (tag: release-ex1)
```

```
Author: r05i42 <r05i42@ed.cc.suzuka-ct.ac.jp>
```

```
Date: Tue Jul 1 13:55:44 2025 +0900
```

```
tag: release-ex1
```

```
commit 8941b90ad8d00b724365f6aa7d0bf6cb4e866be3
```

```
Author: github-classroom[bot] <66690702+github-classroom[bot]@users.noreply.github.com>
```

```
Date: Mon Jun 30 09:45:55 2025 +0000
```

```
Initial commit
```

今回の実験で理解したこと 開発するパソコンで行った変更を実際に動作するサーバーに送るには、VS Codeの拡張機能を使うと便利なのが分かった。今までこのような作業をするときは、FTPでプログラムを送るために別のアプリを立ち上げて、同期してみたいなことをしていたのですが、これを使ったらすごく開発効率が上がると思います。gitの機能としてlogやtagがあることとその使い方がよくわかりました。特にプログラムを変更していくような実験や課題があるときは、diffで差分を見れるのは便利だと思います。わざわざbranchを作って管理しなくてもtagで状態を戻せたりするのもすごく便利で今後使っていきたいです。サーバーとクライアントのTCPパケット通信がどのように行われているのか理解することができました。