



Lembar Jawaban

Hands-On Tutorial - H01

Introduction to Socket Programming

Nama : Alden Luthfi
NPM : 2206028932

[30 Poin] Refleksi 1 : Connectionless vs Connection-Oriented Socket

[10 Poin] R1-1. Personalisasi

Penjelasan:

Pada bagian ini, kita perlu mengubah konfigurasi server dan client untuk bisa saling komunikasi. Di kedua protokol, server harus “mendengar” dari ip yang tepat di port yang tepat. Begitu juga dari client, harus juga mengirim ke ip yang tepat di port yang tepat. Untuk itu, kita perlu mengubah serverIP menjadi **internal** ip dari VM GCP yang ingin dijadikan server, kita juga perlu mengubah port sesuai dengan kebutuhan. Di lain sisi, client juga perlu mengubah serverIP menjadi **external** ip dari VM GCP dan port yang sesuai dengan port dimana server “mendengar”. Internal ip memberikan cara bagi server untuk mengidentifikasi identitasnya sendiri, sedangkan eksternal ip memberikan cara untuk perangkat lain untuk mengetahui identitas suatu server, sehingga ketika perangkat lain mengirimkan sesuatu ke external ip suatu server, server mendengarkan pesan tersebut melalui internal ip (yaitu dirinya sendiri).

Tangkapan Layar:

tcpServer.go

```
const (  
    serverIP   = "10.128.0.3"  
    serverPort = "8932"  
    serverType = "tcp4"  
    bufferSize = 2048  
)
```

udpServer.go

```
const (  
    serverIP   = "10.128.0.3"  
    serverPort = "8932"  
    serverType = "udp4"  
    bufferSize = 2048  
)
```

tcpClient.go

```
const (  
    serverIP   = "35.194.45.88"  
    serverPort = "8932"  
    serverType = "tcp4"  
    bufferSize = 2048  
)
```

udpClient.go

```
const (  
    serverIP   = "35.194.45.88"  
    serverPort = "8932"  
    serverType = "udp4"  
    bufferSize = 2048  
)
```

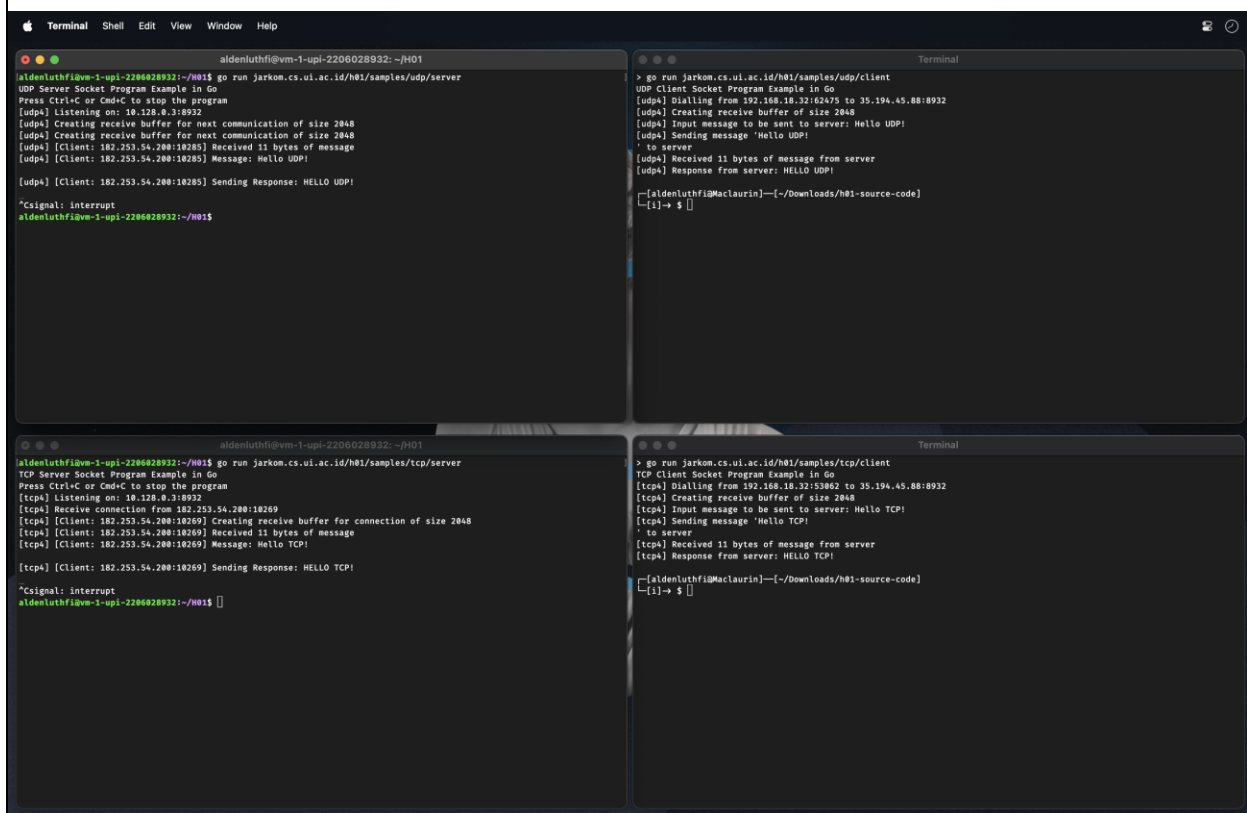
[10 Poin] R1-2. Perbandingan Lalu Lintas

Penjelasan:

Terdapat perbedaan antara keduanya, TCP berusaha untuk memberikan koneksi yang aman sehingga TCP menggunakan Three-Way Handshake (SYN, SYN-ACK, ACK). Hal ini menyebabkan koneksi TCP memberikan banyak sekali frame dan bisa dibilang koneksi TCP lebih lambat dari UDP. Sedangkan, koneksi udp memprioriaskan kecepatan sehingga memerlukan sedikit waktu untuk memberikan koneksi. Terlihat pada tangkap layar Wireshark bahwa koneksi TCP memberikan jauh lebih banyak entri daripada UDP.

Tangkapan Layar:

Setup Client dan Server:



The image displays four terminal windows arranged in a 2x2 grid, showing the execution of network programs. The top-left terminal shows a UDP server running 'jarkom.cs.ui.ac.id/h01/samples/udp/server'. It listens on port 10228, receives a message 'Hello UDP!' from client 182.253.54.200:10285, and sends a response 'HELLO UDP!'. The top-right terminal shows a UDP client running 'jarkom.cs.ui.ac.id/h01/samples/udp/client'. It connects to the server, sends 'Hello UDP!', and receives 'HELLO UDP!'. The bottom-left terminal shows a TCP server running 'jarkom.cs.ui.ac.id/h01/samples/tcp/server'. It listens on port 10229, receives a connection from client 182.253.54.200:10269, receives a message 'Hello TCP!', and sends a response 'HELLO TCP!'. The bottom-right terminal shows a TCP client running 'jarkom.cs.ui.ac.id/h01/samples/tcp/client'. It connects to the server, sends 'Hello TCP!', and receives 'HELLO TCP!'. All terminals are running on a Mac OS environment, as indicated by the window title bars.

Data TCP yang dikirim client:

Capturing from Wi-Fi: en0

ip.addr==35.194.45.88 && tcp

No.	Time	Source	Destination	Protocol	Length	Info
191	0.538797	192.168.18.32	35.194.45.88	TCP	78	53862 → 8932 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=1494237592 TSecr=0 SACK_PERM
196	0.578564	35.194.45.88	192.168.18.32	TCP	74	8932 → 53862 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1412 SACK_PERM TSval=1706856334 TSecr=1494237592 WS=128
197	0.578723	192.168.18.32	35.194.45.88	TCP	66	53862 → 8932 [ACK] Seq=1 Ack=1 Win=131584 Len=0 TSval=1494237824 TSecr=1706856334
200	0.795006	35.194.45.88	192.168.18.32	SSH	246	Server: Encrypted packet (len=180)
201	0.796289	192.168.18.32	35.194.45.88	TCP	66	53843 → 22 [ACK] Seq=1 Ack=181 Win=2048 Len=0 TSval=1286164412 TSecr=1706856563
614	23.661374	192.168.18.32	35.194.45.88	TCP	54	[TCP Keep-Alive] 53862 → 8932 [ACK] Seq=0 Ack=1 Win=131584 Len=0
619	23.888176	35.194.45.88	192.168.18.32	TCP	66	[TCP Window Update] 8932 → 53862 [ACK] Seq=1 Ack=1 Win=65280 Len=0 TSval=1706871656 TSecr=1494237824
625	24.171782	35.194.45.88	192.168.18.32	TCP	66	[TCP Keep-Alive] 8932 → 53862 [ACK] Seq=0 Ack=1 Win=65280 Len=0 TSval=1706871948 TSecr=1494237824
626	24.171960	192.168.18.32	35.194.45.88	TCP	66	[TCP Keep-Alive ACK] 53862 → 8932 [ACK] Seq=1 Ack=1 Win=131584 Len=0 TSval=1494253425 TSecr=1706871656
627	24.329182	35.194.45.88	192.168.18.32	SSH	174	Server: Encrypted packet (len=108)
628	24.329418	35.194.45.88	192.168.18.32	SSH	382	Server: Encrypted packet (len=236)
629	24.329726	192.168.18.32	35.194.45.88	TCP	66	53841 → 22 [ACK] Seq=1 Ack=345 Win=2048 Len=0 TSval=2675330129 TSecr=1706872096
630	24.331253	192.168.18.32	35.194.45.88	TCP	77	8932 → 53862 [ACK] Seq=1 Ack=12 Win=65280 Len=0 TSval=1706871656 TSecr=1494253425
646	28.467551	35.194.45.88	192.168.18.32	TCP	66	8932 → 53862 [ACK] Seq=1 Ack=12 Win=65280 Len=0 TSval=1706876063 TSecr=1494253425
647	28.467868	35.194.45.88	192.168.18.32	TCP	77	8932 → 53862 [PSH, ACK] Seq=1 Ack=12 Win=65280 Len=11 TSval=1706876064 TSecr=1494253425
648	28.468083	192.168.18.32	35.194.45.88	TCP	66	53862 → 8932 [ACK] Seq=12 Ack=12 Win=131584 Len=0 TSval=1494257721 TSecr=1706876064
649	28.468235	192.168.18.32	35.194.45.88	TCP	66	53862 → 8932 [FIN, ACK] Seq=12 Ack=12 Win=131584 Len=0 TSval=1494257721 TSecr=1706876064
650	28.468298	35.194.45.88	192.168.18.32	TCP	66	8932 → 53862 [FIN, ACK] Seq=12 Ack=12 Win=65280 Len=0 TSval=1706876064 TSecr=1494257721
651	28.468291	35.194.45.88	192.168.18.32	SSH	382	Server: Encrypted packet (len=236)
652	28.468341	192.168.18.32	35.194.45.88	TCP	66	[TCP Retransmission] 53862 → 8932 [FIN, ACK] Seq=12 Ack=13 Win=131584 Len=0 TSval=1494257721 TSecr=1706876064
653	28.468418	192.168.18.32	35.194.45.88	TCP	66	53843 → 22 [ACK] Seq=1 Ack=417 Win=2048 Len=0 TSval=1286184803 TSecr=1706876064
657	28.492272	35.194.45.88	192.168.18.32	TCP	66	8932 → 53862 [ACK] Seq=13 Ack=13 Win=65280 Len=0 TSval=1706876460 TSecr=1494257721
1498	91.243453	192.168.18.32	35.194.45.88	SSH	182	Client: Encrypted packet (len=36)
1499	91.267871	192.168.18.32	35.194.45.88	SSH	182	Client: Encrypted packet (len=36)
1500	91.290872	192.168.18.32	35.194.45.88	SSH	182	Client: Encrypted packet (len=36)
1501	91.314382	192.168.18.32	35.194.45.88	SSH	182	Client: Encrypted packet (len=36)
1502	91.338364	192.168.18.32	35.194.45.88	SSH	182	Client: Encrypted packet (len=36)
1503	91.360374	192.168.18.32	35.194.45.88	SSH	182	Client: Encrypted packet (len=36)
1504	91.383283	192.168.18.32	35.194.45.88	SSH	182	Client: Encrypted packet (len=36)
1505	91.405046	192.168.18.32	35.194.45.88	SSH	182	Client: Encrypted packet (len=36)
1506	91.429221	192.168.18.32	35.194.45.88	SSH	182	Client: Encrypted packet (len=36)
1508	91.448725	192.168.18.32	35.194.45.88	SSH	182	Client: Encrypted packet (len=36)
1509	91.469888	35.194.45.88	192.168.18.32	SSH	126	Server: [TCP Previous segment not captured], Encrypted packet (len=60)
1510	91.469977	35.194.45.88	192.168.18.32	TCP	182	[TCP Out-Of-Order] 22 → 53841 [PSH, ACK] Seq=345 Ack=37 Win=483 Len=36 TSval=1706939235 TSecr=2675397842

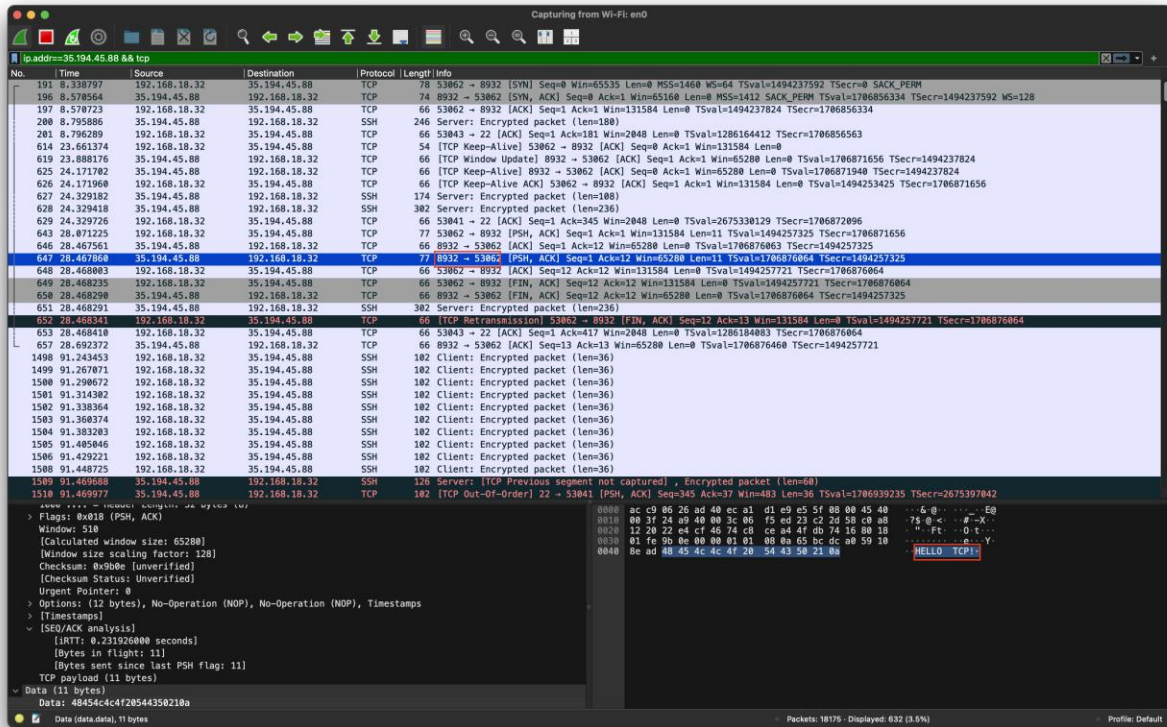
ec a1 d1 e9 e5 5f ac c9 06 26 ad 40 08 00 45 00 & @ E
0020 2d 58 cf 46 22 e4 4f db 74 0b 74 c8 ce a4 80 18 -X P O t t
0030 00 00 66 07 00 00 01 01 08 0a 59 10 0e ad 65 bc V e
0040 cb 60 80 63 6c 6c 6f 20 34 43 30 21 8a Hello TCP!

Data (data.data), 11 bytes

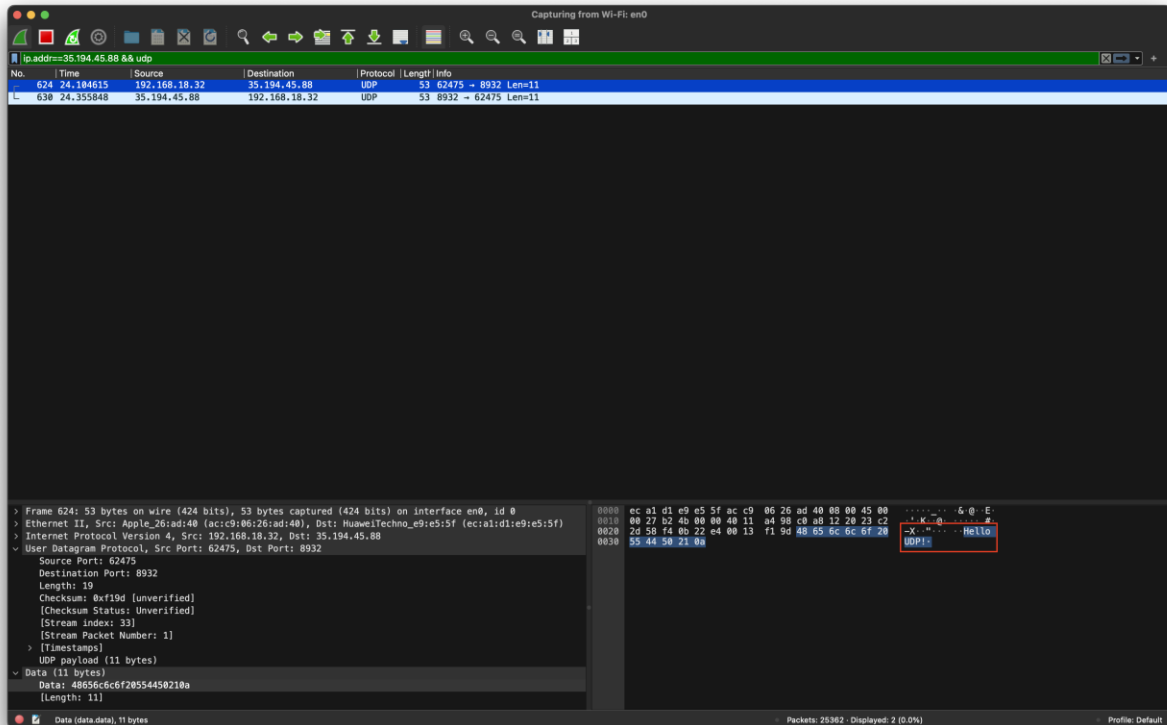
Packets: 11088 - Displayed: 632 (5.3%)

Profile: Default

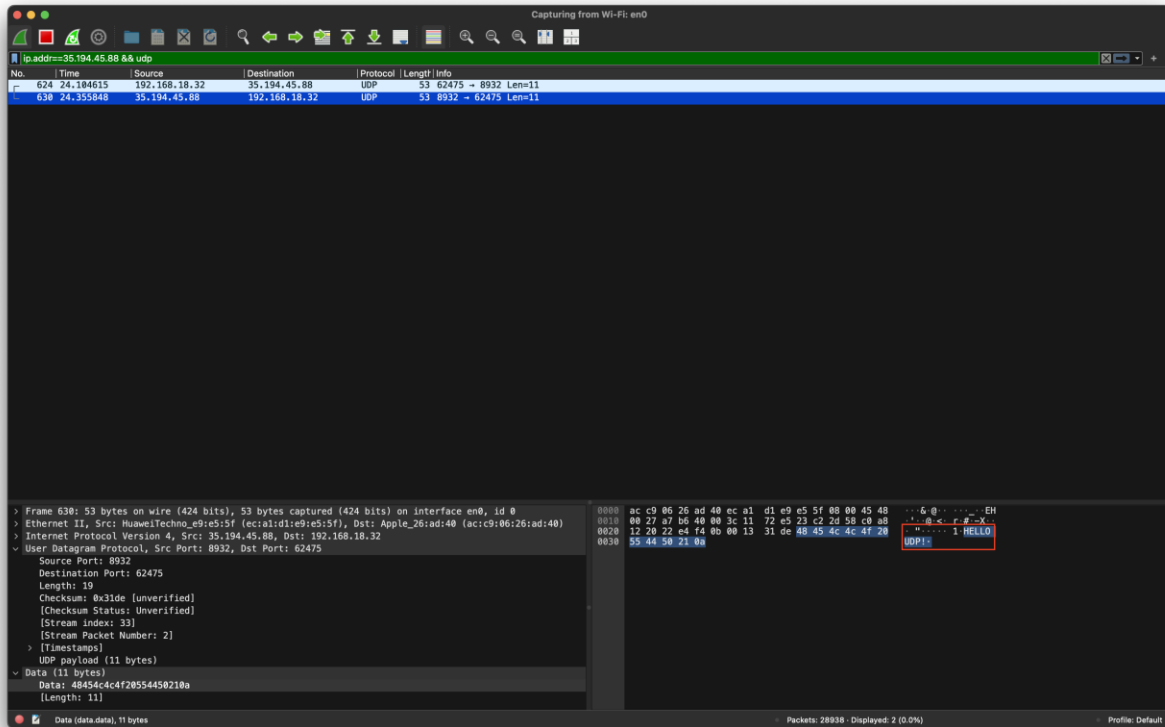
Data TCP yang dikembalikan server:



Data UDP yang dikirim client:



Data UDP yang dikembalikan server:



[10 Poin] R1-3. Bagaimana Jika Server Tidak Ada?

Penjelasan:

Jika tidak ada server, maka tentunya koneksi tidak dapat dibuat. Namun, titik dimana kedua server memberikan pesan error berbeda. Pada koneksi UDP, saat client menyadari bahwa server tidak ada, maka pesan error akan segera ditampilkan. Sedangkan pada koneksi TCP, pesan error baru akan ditampilkan setelah client mengirim pesan lalu menyadari bahwa tidak ada server yang mendengarkan. Pada TCP client mengirimkan SYN (untuk inisiasi koneksi TCP) namun server tidak merespon dengan paket SYN/ACK tetapi respons dari server adalah RST/ACK, hal ini menandakan server tidak menerima koneksi.

Tangkapan Layar:

Setup kedua client:

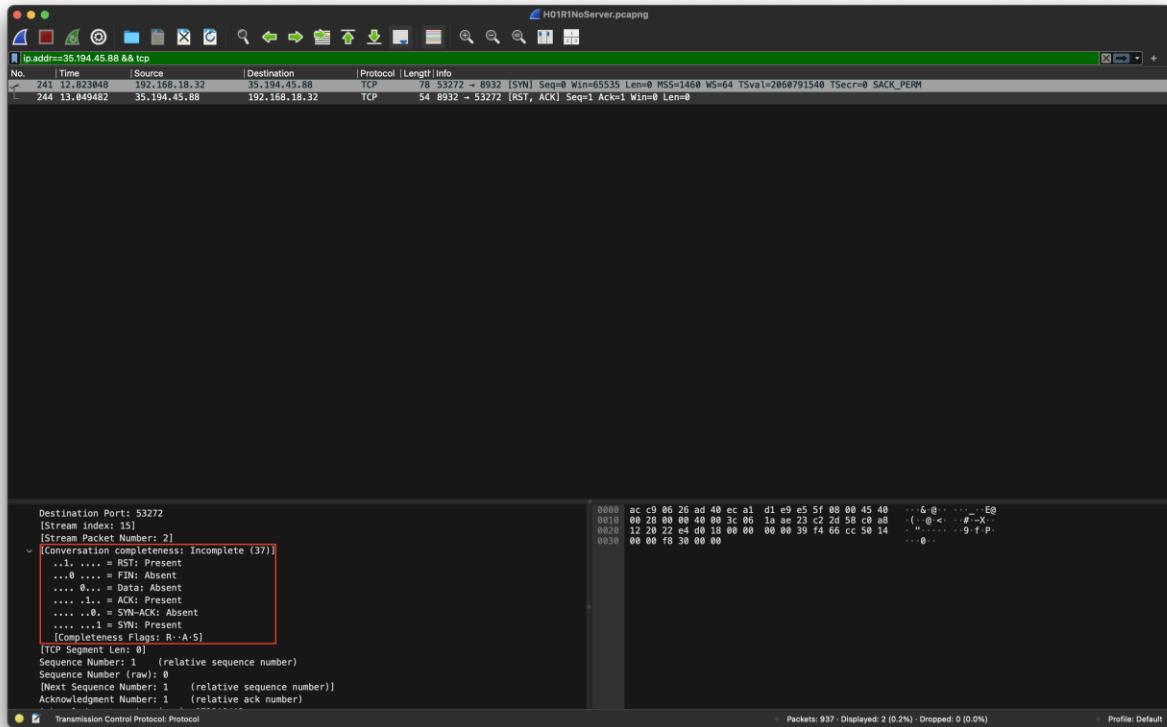
```
Terminal Shell Edit View Window Help

> go run jarkom.cs.ui.ac.id/h01/samples/udp/client
UDP Client Socket Program Example in Go
[udp4] Dialling from 192.168.18.32:59344 to 35.194.45.88:8932
[udp4] Creating receive buffer of size 2048
[udp4] Input message to be sent to server: Hello World!
[udp4] Sending message 'Hello World!'
      to server
2024/09/21 00:48:43 read udp4 192.168.18.32:59344->35.194.45.88:8932: read: connection refused
exit status 1
[aldenluthfi@MacLaurin] ~/Downloads/h01-source-code
[{}]-> $

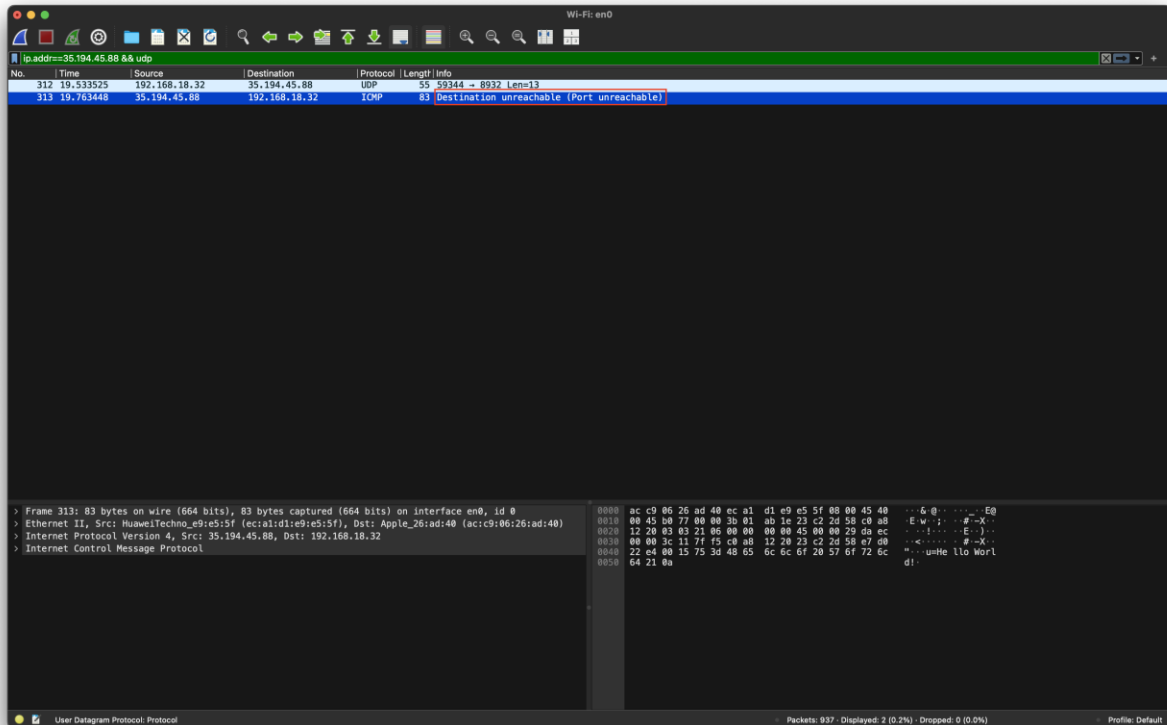
Terminal

> go run jarkom.cs.ui.ac.id/h01/samples/tcp/client
2024/09/21 00:48:36 dial tcp4 35.194.45.88:8932: connect: connection refused
exit status 1
[aldenluthfi@MacLaurin] ~/Downloads/h01-source-code
[{}]-> $ []
```

Pesan error TCP:



Pesan error UDP:



[20 Poin] Refleksi 2 : QUIC Internalization

Penjelasan:

Kita bisa memodifikasi fungsi `handleConnection()` untuk membuka dua stream. Fungsi tersebut memproses panggilan dari fungsi `main` sehingga akan terbuat 2 stream sekaligus ketika terjadi komunikasi dan terjadi parallelism. Dari sisi client perlu ada pembukaan stream menggunakan `for` loop, sehingga client dapat mengirim dan menerima data secara parallel dari server.

Tangkapan Layar:

quicClient.go

```
const (
    serverIP      = "127.0.0.1"
    serverPort    = "54321"
    serverType    = "udp4"
    bufferSize    = 2048
    appLayerProto = "jarkom-quic-sample-upi"
    sslKeyLogFileName = "ssl-key.log"
)
```

```
func main() {
    sslKeyLogFile, err := os.Create(sslKeyLogFileName)
    if err != nil {
        log.Fatalf(err)
    }
    defer sslKeyLogFile.Close()

    net.Printf("QUIC Client Socket Program Example in Go\n")

    tlsConfig := &tls.Config{
        InsecureSkipVerify: true,
        NextProtos:         []string{appLayerProto},
        KeyLogWriter:        sslKeyLogFile,
    }
    connection, err := quic.DialAddr(context.Background(), net.JoinHostPort(serverIP, serverPort), tlsConfig, &quic.Config{})
    if err != nil {
        log.Fatalf(err)
    }

    defer connection.CloseWithError(0, "No Error")

    net.Printf("[quic] Dialling from %s to %s\n", connection.LocalAddr(), connection.RemoteAddr())

    net.Printf("[quic] Creating receive buffer of size %d\n", bufferSize)
    receiveBuffer := make([]byte, bufferSize)

    net.Printf("[quic] Input message to be sent to server: ")
    message, err := bufio.NewReader(os.Stdin).ReadString('\n')
    if err != nil {
        log.Fatalf(err)
    }

    for i := 0; i < 2; i++ {
        stream, err := connection.OpenStreamSync(context.Background())
        if err != nil {
            log.Fatalf(err)
        }
        net.Printf("[quic] Opened bidirectional stream %d to %s\n", stream.StreamID(), connection.RemoteAddr())

        net.Printf("[quic] [Stream ID: %d] Sending message '%s'\n", stream.StreamID(), message)
        _, err = stream.Write([]byte(message))
        if err != nil {
            log.Fatalf(err)
        }
        net.Printf("[quic] [Stream ID: %d] Message sent\n", stream.StreamID())

        receiveLength, err := stream.Read(receiveBuffer)
        if err != nil {
            log.Fatalf(err)
        }
        net.Printf("[quic] [Stream ID: %d] Received %d bytes of message from server\n", stream.StreamID(), receiveLength)

        response := receiveBuffer[:receiveLength]
        net.Printf("[quic] [Stream ID: %d] Received message: '%s'\n", stream.StreamID(), response)
    }
}
```

quicServer.go

```
const (
    serverIP      = ""
    serverPort    = "54321"
    serverType    = "udp4"
    bufferSize    = 2048
    appLayerProto = "jarkom-quic-sample-upi"
)
```

```
func handleConnection(connection quic.Connection) {
    fmt.Printf("[quic] Receiving connection from %s\n", connection.RemoteAddr())

    stream, err := connection.AcceptStream(context.Background())
    if err != nil {
        log.Fatalf(err)
    }

    go handleStream(connection.RemoteAddr(), stream)

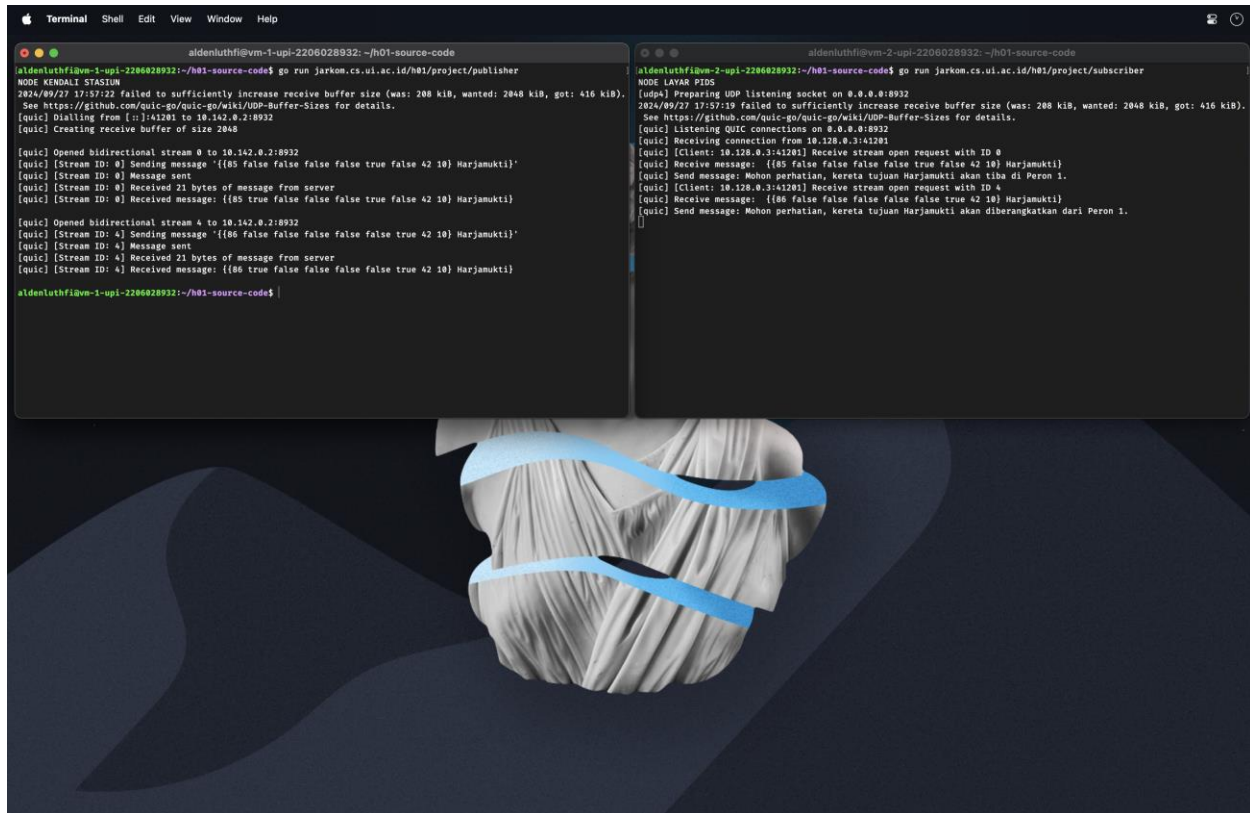
    stream2, err := connection.AcceptStream(context.Background())
    if err != nil {
        log.Fatalf(err)
    }

    go handleStream(connection.RemoteAddr(), stream2)
}
```

[50 Poin] Hands-On Simple Project

[20 Poin] Pengujian Mandiri

Tangkapan Layar:



[30 Poin] Pengujian dengan Autograder

Tidak ada yang perlu dikumpulkan pada bagian ini. **Pastikan kode dikumpulkan agar bisa dinilai.**