TCP And UDP Socket Programming Using Unix Sockets

By

Recep Ecem

25 December 2021

Abstract

TCP and UDP are two very different transport layer protocols. TCP is connection-oriented and secure. In the other hand UDP is connectionless and non-secure. At first glance, UDP may look like unnecessary, but safety brings some burden to our shoulders. TCP header files and its technologies cost more time than UDP. Briefly, A programmer must abnegate safety or speed.

Contents

Introduction

People had to find link their own personal computers to each other. But there were too many companies and different devices. We needed to create a reference model for everyone. Then OSI (Open Systems Interconnection) reference model has published by ISO (International Organization for Standardization). OSI reference model has 7 layers, and each layer has different properties. These layers have their own header files, and these headers may be added or removed layer by layer, so it means OSI is a bidirectional model.

There is more for OSI, but we want to focus on transport layer which contains TCP and UDP protocols that we want to examine.

UNIX stands for local machine which works on Loopback address. You can use this IP address and any port that IANA (Internet Assigned Numbers Authority) did not reserve.

Layer			Protocol Data Unit
Host Layers	Application	7	Data
	Presentation	6	
	Session	5	
	Transport	4	Segment, Datagram
Media Layers	Network	3	Packet
	Data link	2	Frame
	Physical	1	Bit

Table 1: OSI reference model

Transport Layer

Transport Layer is the fourth layer of the OSI reference model. This layer has many protocols, there is two well-known protocols that are TCP and UDP. You can communicate with under layer or top layer using these protocols. I would like to express Transport layer with an analogy.

The Household Analogy:

Consider two houses, one on the European side and the other on the Asian side of the Istanbul, with each house being home to a dozen kids. The kids in the European side household are cousins with the kids in the Asian side households. Each kid writes each cousin every week, with each letter delivered by the traditional postal service in a separate envelope. In each of the households there is one kid, Alice in the European side and Bob in the Asian side, responsible for mail collection and distribution. Each week Alice visits all her siblings, collects the mail, and gives the mail to a postal-service mail person. Alice also has the job of distributing the mail to her siblings. Bob has a similar job in the Asian side.

- Hosts: Houses
- Processes: Cousins
- Application messages: Letters in envelopes
- Network layer protocol: Postal service
- Transport layer protocol: Alice and Bob

Transmission Control Protocol (TCP)

TCP is one of the most popular transport layer protocols. TCP is reliable protocol because the receiver sends either positive or negative acknowledgment about the data packet.

Offsets	Octet	0								1								2								3							
Octet	Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	Source port																Destination port																
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size														
16	128	Checksum																	Urgent pointer (if URG set)														
20	160	Options (if *data offset* > 5. Padded at the end with "0" bits if necessary.)																															
⋮	⋮																																
60	480																																

Table 2: TCP segment header

TCP has three phases:

1. **Connection establishment**

   TCP uses a three-way handshake. Before a client attempts to connect a server, it listens the channel and if it could get a reply then phase 2 starts. Three-way handshake also has 3 steps, these are:

   a) Client sends TCP-SYN segment to server

      i) Specifies a ‘random’ initial sequence number

      ii) Has no data

   b) Server receives SYN, replies with SYN-ACK segment

      i) Server allocates buffers

   c) Client receives SYN-ACK, replies to ACK.

      i) May contain data

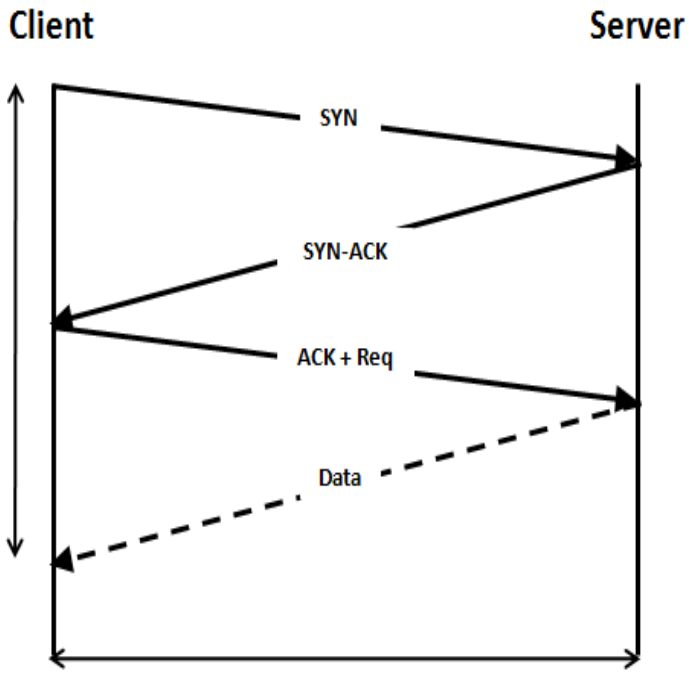

Figure 1: TCP's 3-way handshake

2. User data transmission

This step has lots of features such as bandwidth management, Error control, Flow control, Congestion control, etc. We just need to know that TCP settles everything for us, and it provides a lot of convenience to the programmer.

3. Disconnection

TCP uses a four-way handshake to terminate the connection. When a node wishes to terminate its connection the four-way handshake starts. The four-way handshake has 4 steps, these are:

a) Client sends FIN segment to server
    i) Server waits ‘CLOSE_WAIT’ milliseconds
b) Server receives FIN, replies with ACK
    i) Client waits ‘FIN_WAIT’ milliseconds
c) Also, server sends its FIN segment to the client
    i) Client waits ‘TIME_WAIT’ milliseconds
d) Client receives ACK and FIN segments, replies with ACK
    i) Connection terminated

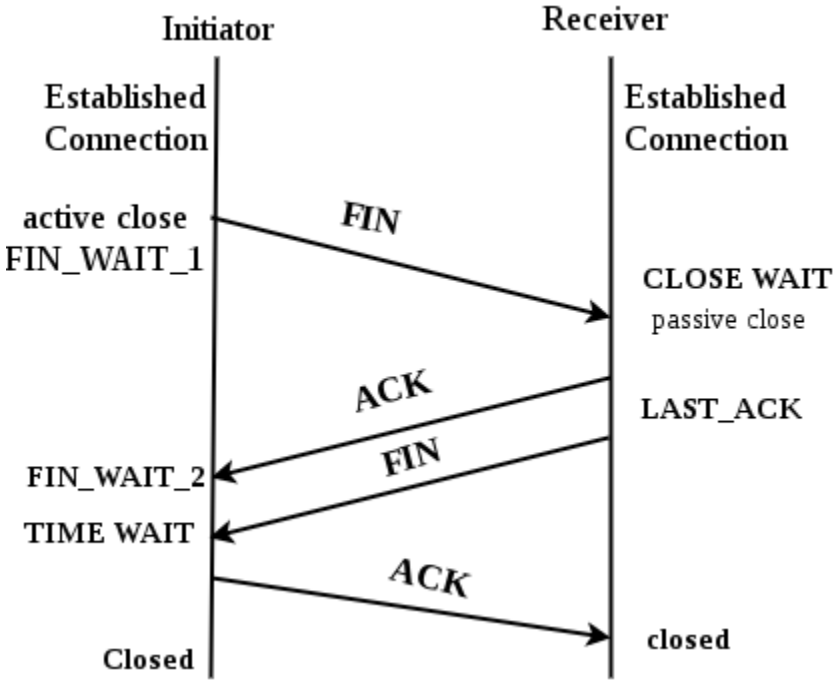

Figure 2: TCP’s 4-way handshake

User Datagram Protocol (UDP)

UDP is also one of the most popular transport layer protocols. UDP is very simple and cheap way to send any data. Despite its ease it is not secure. We can say that UDP is a best-effort service, but you can add some of TCP's features by hand. Such as DNS, SNMP protocols uses UDP with extra in-app security adjustments.

Bits	0 - 15	16 - 31
0	Source port	Destination port
32	Length	Checksum
64	Data	

Table 3: UDP datagram header

UNIX Socket

Unix Socket (aka. Inter-process communication socket) is an endpoint for transfer data between process on the same computer operating system. UNIX sockets works on loopback IP address which is ‘127.0.0.1’. Within local personal computer any program may use this address to communicate with any other program that exist on the local computer or with itself. There is two well accepted UNIX domain, these are:

1) SOCK_STREAM

This domain uses TCP as a transmission protocol. If you use SOCK_STREAM, you do not need to struggle thanks to TCP. When you use this domain, it needs some other functions to work properly. (I will touch these functions later)

2) SOCK_DGRAM

This domain uses UDP as a datagram protocol. If you use SOCK_DGRAM, you must pay attention to connection because UDP is connectionless, best-effort transport layer protocol. Generally, programmers add extra features on the UDP for extra security and reliability.

There is also another UNIX socket, SOCK_SEQPACKET which uses SCTP. But it is obsolete.

I have written two separate programs. First one is ‘Server Side’ and other one is ‘Client Side’

The Server Side

The server program uses UNIX sockets and both TCP and UDP protocols. I have used C as a programming language. I am going to use a pseudo code to explain this.

Program: Server-side TCP and UDP programming using UNIX sockets

Pseudo code:

```
If argument counter != 3
  Exit failure
If argument vector 2 == “tcp”
  socket(AF_UNIX, SOCK_STREAM, IPPROTO)

  Address.family = AF_UNIX
  Address = INADDR_ANY
  Address.port = htons(port)

  Bind(SocketFileDescriptor, address, sizeof(address))
  Listen(SocketFileDescriptor, clientNo)
  Accept(SocketFileDescriptor, address, addressLenght)

  While(true)
    Read(buffer)
    Print(buffer)
    Scan(message)
    Send(message)

If argument vector 2 == “udp”
  Socket(AF_UNIX, SOCK_DGRAM, IPPROTO)

  Address.family = AF_UNIX
  Address = IANDDR_ANY
  Address.port = htons(port)

  Bind(SocketFileDescriptor, address, sizeof(address))

  While(true)
    Recievefrom(buffer)
    Print(buffer)
    Scan(message)
    Sendto(message)
```

The Client Side

The server program uses UNIX sockets and both TCP and UDP protocols. I have used C as a programming language. I am going to use a pseudo code to explain this.

Program: Client-side TCP and UDP programming using UNIX sockets

Pseudo code:

```
If argument counter != 3
  Exit failure
If argument vector 2 == “tcp”
  socket(AF_UNIX, SOCK_STREAM, IPPROTO)

  Address.family = AF_UNIX
  Address.port = htons(port)

  Connect(SocketFiledescriptor, address, sizeof(address))



  While(true)
    Scan(message)
    Send(message)
    Read(buffer)
    Print(buffer)

If argument vector 2 == “udp”
  Socket(AF_UNIX, SOCK_DGRAM, IPPROTO)

  Address.family = AF_UNIX
  Address.port = htons(port)

  While(true)
    Scan(message)
    Sendto(message)
    Recievefrom(buffer)
    Print(buffer)
```

Conclusion

There is plenty of ways to communicate among the computers. We especially dealt with UNIX (local) sockets using TCP and UDP protocols. When we use UNIX sockets, we do not need to define any IP addresses and converting it to machine under stable form. There are some differences between TCP and UDP. If you would like to use TCP, there are some settlements that you have to do sake of 3-way handshake. In the other case UDP needs to some manual adjustments such as various of flags.

References

- W.Ross, James F. Kurose. (2021 December 25). Transport Layer Services and Principles. http://www2.ic.uff.br/~michael/kr1999/3-transport/3_01-transport_layer.htm
- Tutorialspoint. (2021 December 26). Transmission Control Protocol. https://www.tutorialspoint.com/data_communication_computer_network/transmission_control_protocol.htm
- Vskills. (2021 December 26). TCP Connection Establish and Terminate. https://www.vskills.in/certification/tutorial/tcp-connection-establish-and-terminate/
- Wikipedia. (2021 December 26). Unix Domain Socket. https://en.wikipedia.org/wiki/Unix_domain_socket
- Table 2: wikipedia. (2021 December 26). Transmission Control Protocol. https://en.wikipedia.org/wiki/Transmission_Control_Protocol
- Figure 1: Bifulco, Roberto. (2021 December 26). 3 Way Handshake. https://www.researchgate.net/profile/Roberto-Bifulco-2/publication/305081696/figure/fig2/AS:643173193879554@1530355847765/3-way-handshake-3-way-handshake-and-Proxy-Early-SYN-Forwarding.png
- Figure 2: geeksforgeeks. (2021 December 26). 4 Way handshake. https://media.geeksforgeeks.org/wp-content/uploads/CN.png