TCP vs. UDP

## Connection/Connectionless

**TCP**

An app that uses tcp creates a connection and then streams data over it with the other side. This will require the app to listen, wait and accept a connection from the other side. That’s a lot of time.

Bytes are sent in a stream. This creates two problems:

1. How to determine the end of a message and the start of another.
2. How to reconstruct a single long message from multiple streams.

We make use of a character that we don’t use to separate the text messages, (e.g. \n, which is a non-printing character). So the sender will put this character at the end of a message and the receiver will look for this specific character to mark the end of a message and the start of the next.

This connection will go on until we tell the server to close it.

**UDP**

Apps don’t need a connection. Neither do they listen nor accept a connection, they just send and receive. Just throw the message and hope it reaches the other side. This is faster and more efficient than tcp.

Bytes are sent in a datagram, which is the smallest unit of packet switching. It means every send order will be sent in a packet and every packet will reach the other side using a different paths. That’s why they may reach out of order. Thus we don’t need to terminate the message with character because every message will be received alone in one packet not in a stream.

**Reliability**

Are we guaranteed that or data will reach the other side and if so, will it be in correct order or not.

**Tcp** is reliable because it adds extra overheads of 64bytes to the original message to synchronize the data and it will ask the sender to resend a specific part of the message based on the order written on those overheads.

**Udp** is unreliable because it doesn’t care if the message reaches or not or is it even in correct order. It’s faster and more efficient that video call uses it because it can cope with potential data loss and out of order arrival.

**Error checking**

Tcp will ask the other side to resend a lost part of the data based on the overheads. In udp we have to develop an error detection mechanism.

In most cases, udp can’t reconstruct a message if it has detected an error. Routers, switches and other devices on the network can send a message called “automatic repeat request” to the sender and most modern protocols are built using a mixture of tcp and udp to acquire reliability and efficiency.

**Socket**

It is a tool for apps to communicate through the network with each other. Mostly, a socket refers to tcp/ip model in the transport layer.

Tcp server

1. Create
2. Give ip and port of an interface
3. Listen
4. Wait and accept
5. Create new socket to manage session
6. Send/receive
7. Close

|  |
| --- |
| import socket server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) server.bind(("0.0.0.0", 8080)) server.listen(0) print("waiting") connect, add = server.accept()  print("connected") message = "thank you for connecting"  data = message.encode()  connect.send(data)  connect.close()  server.close() |

This connects and sends the message. Encode the message with default uft-8 which is suitable for text.

AF\_INET means IPv4. SOCK\_STREAM means tcp streaming

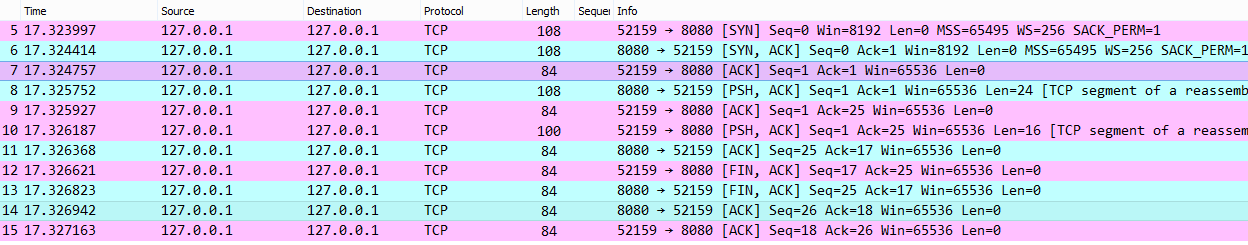
Listen(0): the 0 is the number of unaccepted connections the system will allow before refusing new ones. it will accept every request.

Create dummy tcp client to verify

|  |
| --- |
| import socket client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) client.connect(('127.0.0.1', 8080)) print("connected") data = client.recv(1024)  message = data.decode()  client.close() |

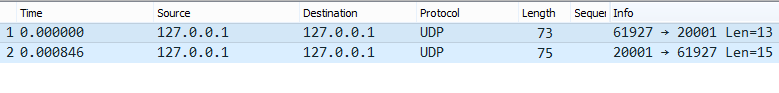
**Wireshark**

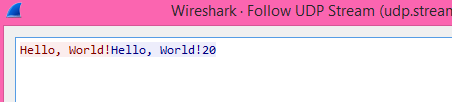
**TCP**



* Color green is sent from port 8080(server to receiver). Color red is the opposite.
* At the very right, First three packets are the normal three way handshake of tcp.
* Forth packet server sends the message “thank you for connecting”
* Fifth packet the client acknowledges that with ACK
* Sixth packet client will send the dummy message “client message”
* Then server acknowledges that using ACK
* They send FIN, ACK packets to tell the other side that the connection will close.
* Messages with length 0 are management packets. Otherwise, they contain data.
* Every message has a sequence that ensures ordering and resending requests.

**UDP**



Just one message to send and one to receive. Tcp needed 11 messages. udp just throws the messages to the other side.

This is not encrypted.

Right click on a packet> follow udp/tcp