Socket Programming

Kinley

Written on June, 2022

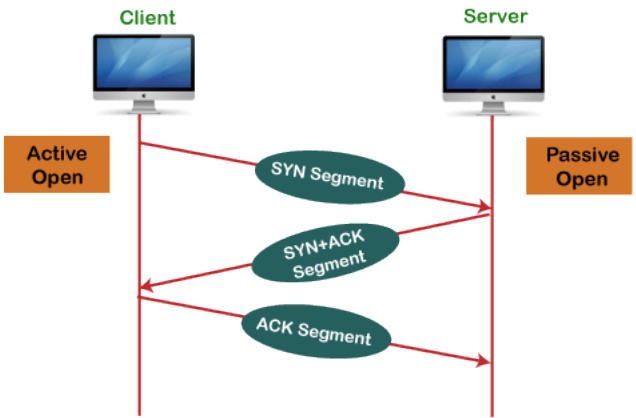
SINTEF

What is TCP?

- Transmission Control Protocol (TCP) is connection-oriented, meaning once a connection has been established, data can be transmitted in two directions.
- TCP has built-in systems to check for errors and to guarantee data will be delivered in the order it was sent, making it the perfect protocol for transferring information like still images, data files, and web pages
- But while TCP is instinctively reliable, its feedback mechanisms also result in a larger overhead, translating to greater use of the available bandwidth on your network.

SINTEF

Working of the TCP protocol



SINTEF

What is UDP?

- User Datagram Protocol (UDP) is a simpler, connectionless Internet protocol wherein error-checking and recovery services are not required.
- With UDP, there is no overhead for opening a connection, maintaining a connection, or terminating a connection
- It will sends a checksum along with the packet so that the receiver can check if the packet is damaged
- Data is continuously sent to the recipient, whether or not they receive it.
- It thus leads to a much faster process than TCP does.
- Although UDP isn't ideal for sending an email, viewing a webpage, or downloading a file, it is largely preferred for real-time communications like broadcast or multitask network transmission.

SINTEF

What is the Difference Between TCP and UDP?

- In general, TCP requires an established connection between server and client when UDP does not.
- TCP's use of extensive error-checking and retransimission mechanisms makes it more reliable than UDP.
- Although UDP isn't ideal for sending an email, viewing a webpage, or downloading a file, it is largely preferred for real-time communications like broadcast or multitask network transmission.
- https://www.privateinternetaccess.com/blog/tcp-vs-udp-understanding-the-difference/

SINTEF

What is Socket Programming?

It's really easy!

- Write a program
- Connect two nodes on a network to communicate with each other
- Several important concepts:
  - Socket
  - Port
  - IP
  - Packet

SINTEF

Important Concepts

Socket

- A network socket is a software structure within a network node of a computer network
- It serves as an endpoint for sending and receiving data across the network

Port

- Each port is a communication endpoint that identifies a specific process or a type of network service
- The port number is a 16-bit unsigned and ranges from 0 to 65335
- It is identified by different transport protocols (TCP or UDP) and addresses

SINTEF

Important Concepts

IP

- Internet Protocol
- Different versions have different length and expression
- For one device, each network interface will have an assigned IP address when it is connected to a network
- The device connects to the Ethernets and WiFis with separate interfaces

Packet

- It is a protocol data units
- When sending a bunch of data (like a file, video and etc.), transport protocol will break it down to small packets and transmit one by one to the destination

SINTEF

Socket Programming in Python

- *socket* is a useful Python library
- You do not need to simulate the mechanism of the protocols (e.g. three handshake)
- Only TCP and UDP processes are required today
- Two devices, *EACH* serves as a client and a server
- Server *listens* to the client; client *sends data* to the server
- Thus, you will need two files that simulate different statuses
  - For example, TCP_client.py and **TCP_server.py**

SINTEF

socket library: functions

- You can refer to https://docs.python.org/3/library/socket.html for official description.
- It has many built-in constants used to represent different socket families (like IP version, protocol type, booleans...)

Create a socket

socket.socket(family=AF_INET, type=SOCK_STREAM)

- *AF_INET*: IPv4
- *AF_INET6*: IPv6
- *SOCK_STREAM*: TCP
- *SOCK_DGRAM*: UDP

SINTEF

socket library: functions

Bind the socket to your device

socket.bind((_HOST, _PORT))

Server

socket.listen(), socket.accept()

socket.recv(_BufferSize)

socket.recvfrom(_BufferSize)

Client

socket.sendall(_data)

socket.sendto(_data, (_HOST, _PORT))

SINTEF

socket library: functions

Other useful functions

socket.close()

data.decode(encoding='utf-8')

data.encode(encoding='utf-8')

General Structure

- **Server**

1. Bind to a host and a port
2. Listen to the incoming messages
3. Accept/receive the data and print out the message if you want
4. Close the socket

- **Client**

1. Bind to a host and a port
2. Send messages to the destination server

SINTEF

Good Luck!

- Enough for an introduction! You should know enough by now
- Please CAREFULLY read the instructions
- If you have more questions, try to search on Google or Baidu BEFORE asking others

SINTEF

Technology for a better society