18131611

**TCP Simulator**

The state class is a superclass of all states in the state machine. It expresses the common attributes found in all states. In the State class there is a trigger() method which enhances the flow of the whole TCP process. It is like an automatic trigger function that implements immediate tasks when the state machine enters a new state.

The transition class is a very important aspect of the state machine since it is responsible for implementing any methods(by displaying an error) that cannot be implemented inside the state that the state machine is currently in. All the states therefore inherit from the transition class.

Initially both the server and the client are at a closed state. In order to open a TCP connection, the server should be listening for incoming connections. To do this, passive\_open() is called that calls on to the listen() method for the creation of a listening socket to accept incoming connections. If a connection is successfully made, the state of the server is updated to LISTEN, otherwise the program exits.

The client also initiates an active open to establish a connection. If a connection is successful, a three-way handshake is commenced by sending a “SYN” to the server by the client then the client updates its state to SYNSENT. The SYN packet is loaded with the client’s sequence number which is random whereby the client is asking the server for permission to speak to it.

Once the packet is received by the server, it sets the acknowledgement number to one more than the sequence number received from the client in the SYN packet. The server then responds with a SYN\_ACK packet containing sequence number which is random and the acknowledgement number. The server then updates its state to SYNRECVD.

The client on receiving the SYN\_ACK packet, checks if the ACK number received is equal to one more than its own sequence number. If it is, the client updates its sequence number to the acknowledgement number received and the acknowledgement number to one more than the sequence number received from the server. The client then sends an ACK packet to the server containing the sequence number and acknowledgement number then it updates its state to ESTABLISHED. If for instance the client waits for too long for information to be received from the server, it timeouts and closes the connection. This is implemented by the use of a try block. Also if the client receives a RST packet in this state, it closes the connection.

The server on receiving the ACK packet, checks that the acknowledgement number received is equal to one more than its own sequence. If it is, it updates its state to Established, and if it is not, it remains in the same state (SYNRECVD).

At this point, both the server and the client have received an acknowledgement of the connection and are ready to transfer data.

The process starts with the client getting some data from the user. As long as the data does not indicate to quit the communication, the client encrypts the data and sends it to the server. The server which continuously receives data from the client, gets the data, decrypts it, converts it to upper case, encrypts it and sends it back to the client. The client on receiving it, decrypts it, displays it and prompts a new message from the user.

The encryption/decryption functionalities make use of the XOR operation to encrypt and decrypt data. This is done by traversing through every character in the data and performing an XOR with the corresponding character of the secret key. The indication to encrypt is represented by a “True” value and the indication to decrypt is represented by a “False” value as function arguments.

Whenever the client or server is done with exchanging data, they can terminate their connection however, they need to let each other know about the termination instead of just dropping the connection. In this case, the client initiates the termination by sending a FIN packet containing a sequence number to the server and updating its state to FINWAIT1. The server uses a try block to check that the data received from the client is a valid JSON string which means that it is a packet in which the client might request to close the connection. If it is, the server updates the acknowledgement number to one more than the sequence number received from the client, breaks the loop, sends an ACK packet containing its sequence number and the acknowledge number which is one more that the sequence number received from the client in the FIN packet and then updates its state to CLOSE\_WAIT.

The client receives the ACK packet and checks to see if the acknowledgement number received is one more than its sequence number. If it is, it sets its sequence number equal to the received acknowledgement number and then sets the acknowledgement number to one more than the received sequence number then updates its state to FINWAIT2 and waits for an ACK from the server. At this point the client does not send anything to the server.

After the server transitioned to CLOSE\_WAIT , it then sends a FIN packet to the client containing its sequence number and the acknowledgement number as updated previously and then updates its state to a LAST\_ACK.

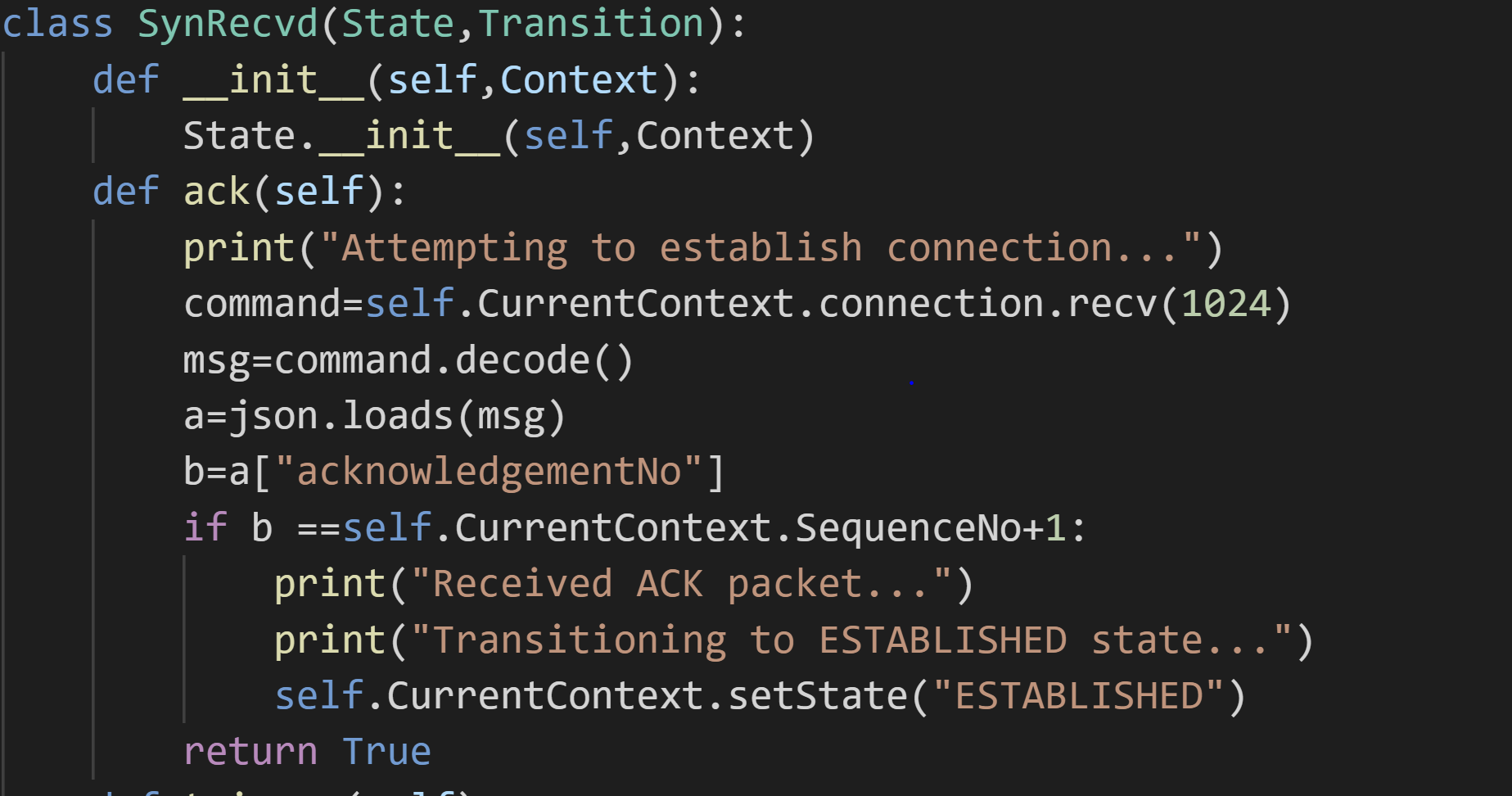
The client who was waiting for a FIN packet, receives it, checks to see if the acknowledgement number equals its own sequence number. If it is then it sets the sequence number to the received acknowledgement number and the acknowledgement number to one more than the received sequence number from the server and then it sends the ACK packet containing its sequence number and acknowledgement number. It then updates its state to TIMEDWAIT whereby it waits for some time and then transition to CLOSED state where it terminates the connection.

The server receives the ACK packet, check to see if the acknowledgement number is equal to one more than its sequence number. If it is then it transitions to CLOSED state where the connection is terminated.

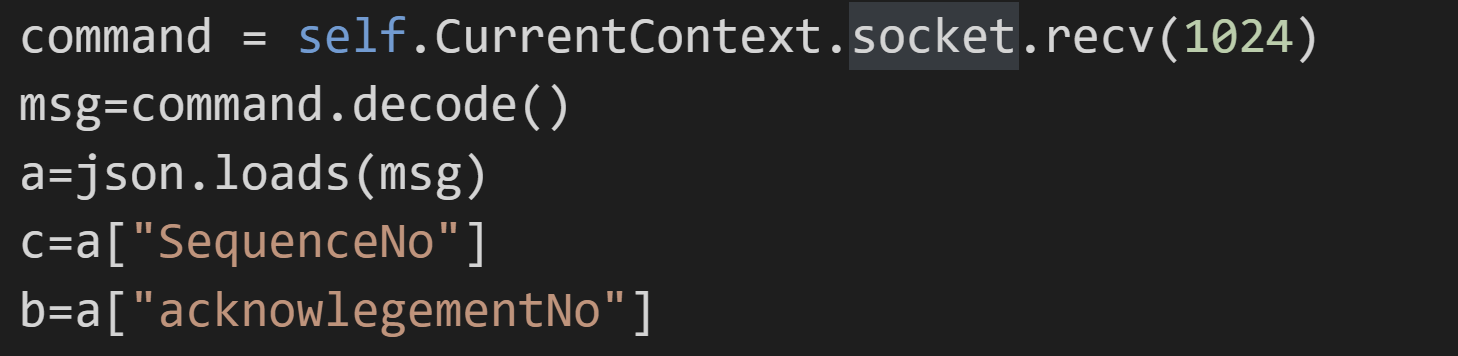
In some states there are certain conditions that have to be met in order for the state machine to transition to a new state. For instance when the server receives an ACK from the client and the acknowledge number does not match its sequence number plus one, the server will remain in the same state.

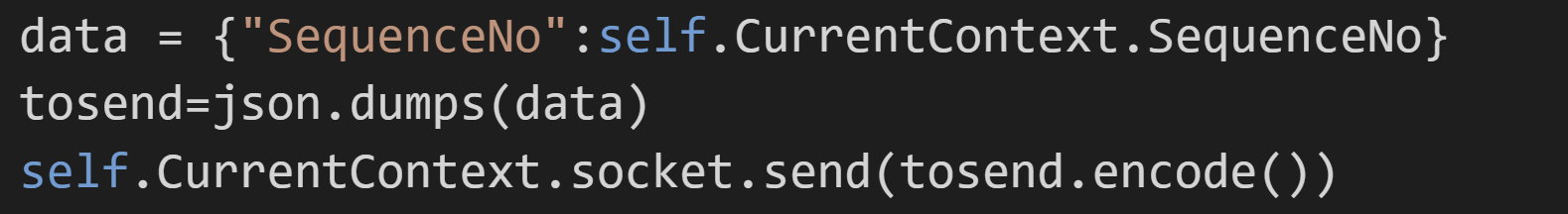
For instance from the image below, the server will remain in the SYNRECVD state if the ‘if’ statement condition is not met.

18131611

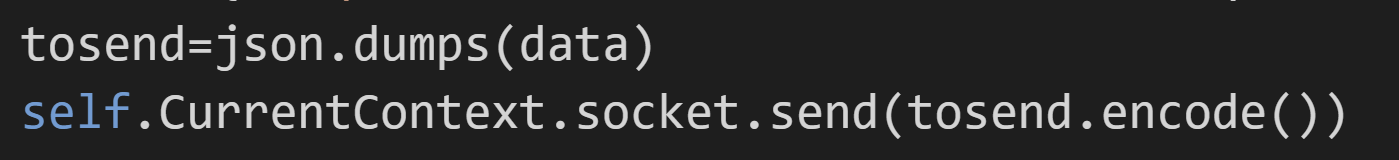


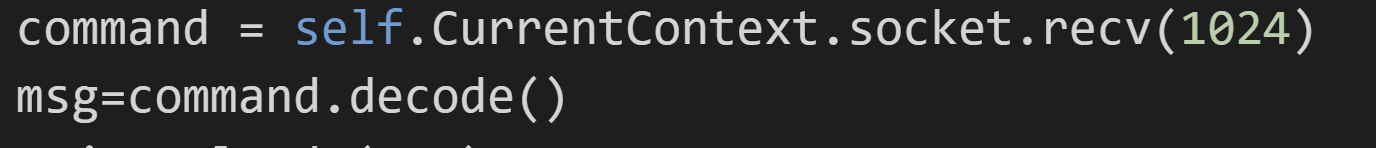
The program uses json format for serializing the structured data which in this case python objects (dictionary) being sent across. The data is first put in a python object then converted to a json string using ‘json.dumps()’. The receiving end then converts the json string back to the python object using ‘json.loads()’ where each attribute can be accessed by specifying its key name.





There is also the use of encode() and decode() methods. The encode() is used when sending data whereby the data string is converted to bytes and decode method is used when data is received to convert the bytes back to a string. The encoding of the strings is necessary since they are being sent over a network.





18131611

The way that the TCP simulation is explained above is that a packet for instance “SYN” packet is sent first then the state machine transitions to a SYNSENT state. However, the way that it is implemented in code is that the transition takes place and then the packet is sent within the new state’s trigger() method.

18131611