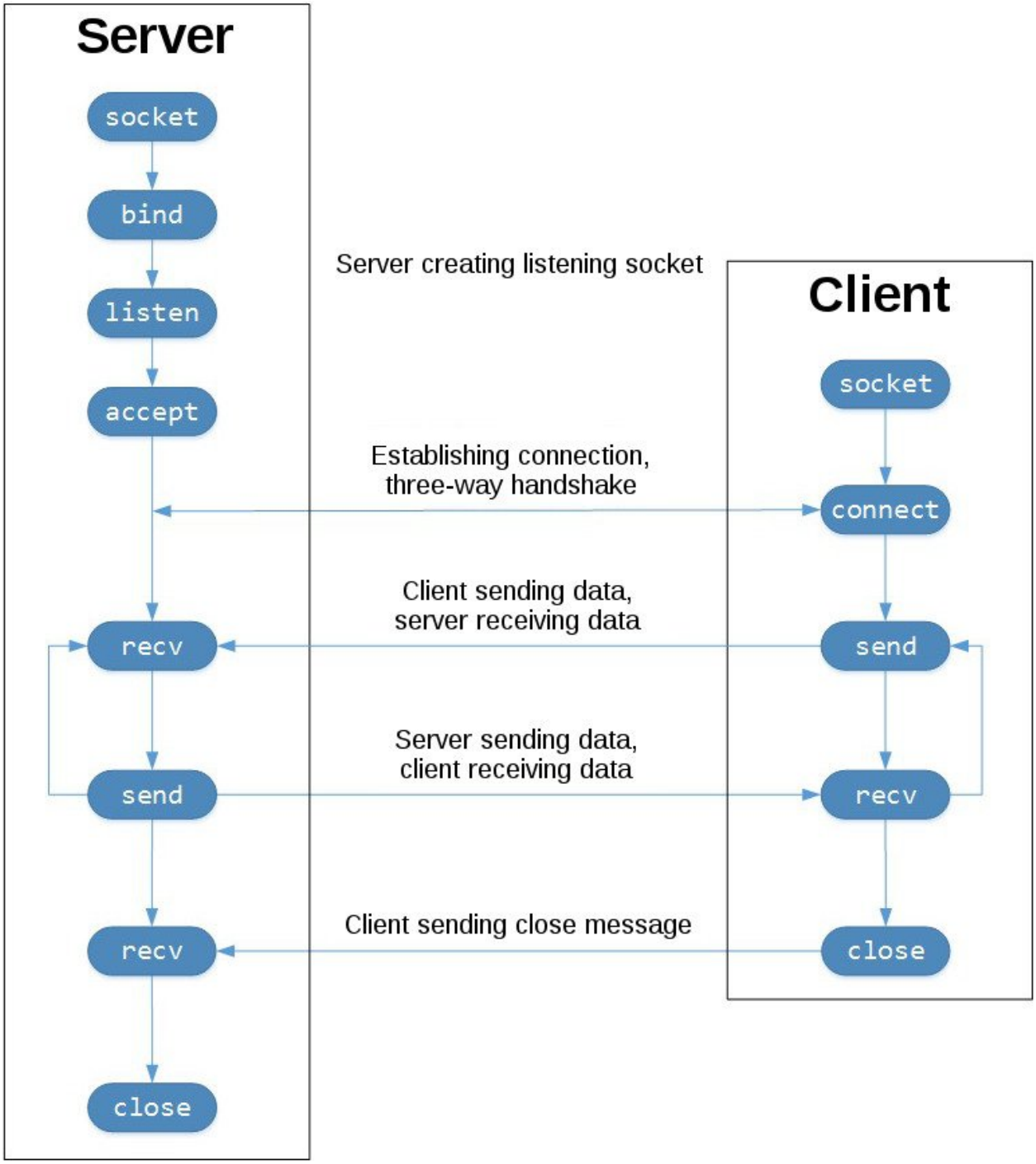
SOCKET PROGRAMMING IN PYTHON:



Methods	Description
socket.socket()	used to create sockets (required on both server as well as client ends to create sockets)
socket.accept()	used to accept a connection. It returns a pair of values (conn, address) where conn is a new socket object for sending or receiving data and address is the address of the socket present at the other end of the connection
socket.bind()	used to bind to the address that is specified as a parameter
socket.close()	used to mark the socket as closed
socket.connect()	used to connect to a remote address specified as the parameter
socket.listen()	enables the server to accept connections

Example program:

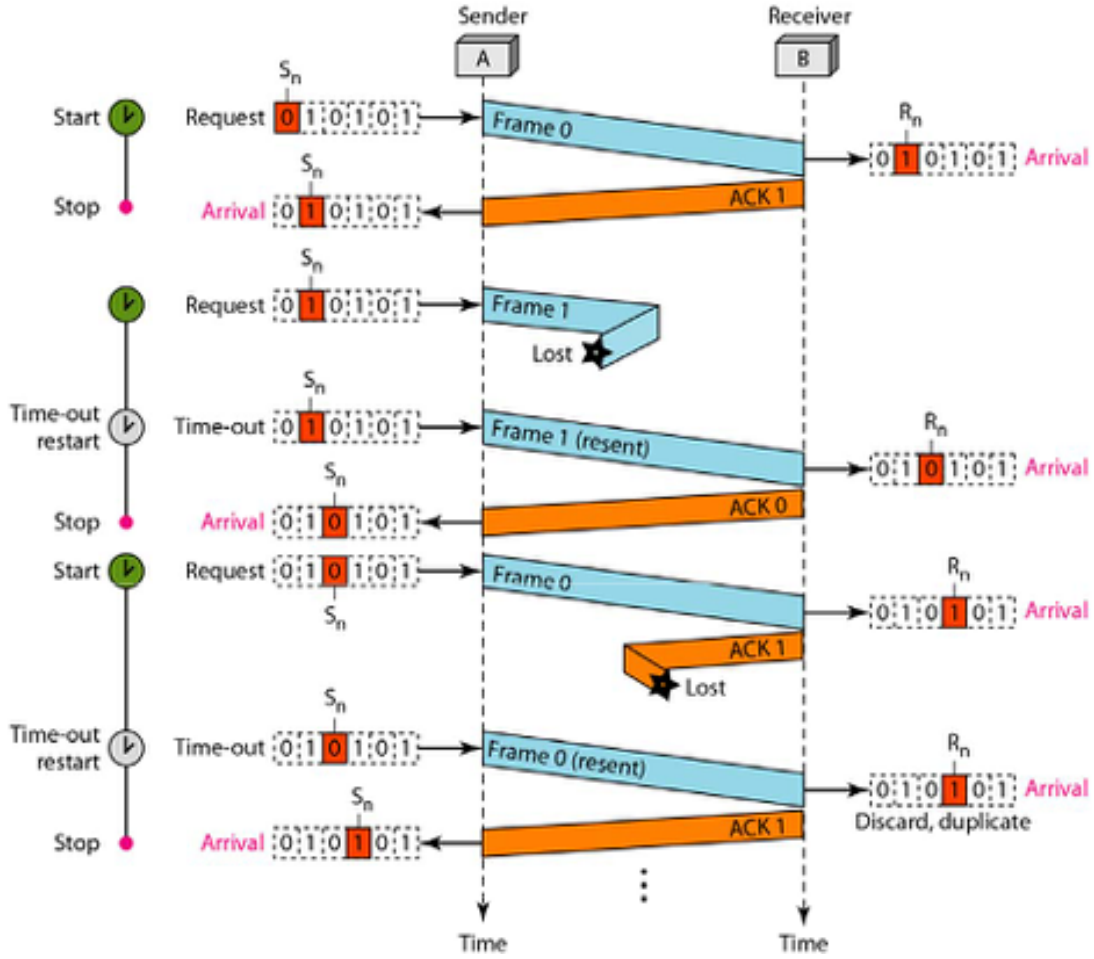
SERVER SIDE:

```
import socket
s = socket.socket()
s.bind(("localhost", 8038))
s.listen(5)
c, adr = s.accept()
print(str(adr) + "connection established successfully")
c.send(str("hello Client").encode())
print(c.recv(1024).decode())
c.close()
```

CLIENT SIDE:

```
import socket
s=socket.socket()
s.connect(("localhost", 8038))
print("s.recv(1024).decode())
s.send(str("hello server").encode())
s.close()
```

8. Write a Program to implement Stop and Wait Protocol.



Implementation at Sender side (server.py):

```
import socket
import time
import random
s=socket.socket()
s.bind(("localhost", 8020))
s.listen(5)
c, adr = s.accept()
print("connection to " + str(adr) + " established")
a=int(input("enter total number of frames"))
x = 0
print("sending -->", x)
c.send(str(x).encode())
while( a > 1 ):
```

```
    timer = 5
    t=random.randint(1,7)
    msg = c.recv(1).decode()
    if( timer > t):
        time.sleep(3)
        print("ack-->", msg)
        x=int(msg)
        print("sending -->", str(x))
        c.send(str(x).encode())
    else:
        time.sleep(3)
        print("timeout")
        print("sending again-->", x)
        c.send(str(x).encode())
        a=a+1
    a = a-1
```

Implementation at Receiver side (client.py):

```
import socket
import time
s=socket.socket()
s.connect(("localhost", 8020))
while(1):
    msg=s.recv(1).decode()
    print("Received --> ", msg)
    x=int(msg)
    if(x==0):
        x=x+1
        s.send(str(x).encode())
    else:
        x=x-1
        s.send(str(x).encode())
```

Output:

Note: first run server.py and then client.py

SERVER SIDE:

connection to ('127.0.0.1', 55894) established
enter total number of frames 5
sending --> 0
ack--> 1
sending --> 1
timeout
sending again--> 1
timeout
sending again--> 1
timeout
sending again--> 1
ack--> 0
sending --> 0
ack--> 1
sending --> 1
timeout
sending again--> 1
timeout
sending again--> 1
timeout
sending again--> 1
ack--> 0
sending --> 0

Process finished with exit code 0

CLIENT SIDE:

Received --> 0
Received --> 1
Received --> 1
Received --> 1
Received --> 1

Received --> 0
Received --> 1
Received --> 1
Received --> 1
Received --> 1
Received --> 0
Process finished with exit code 1

6. Write a Program to implement Sliding window protocol for Goback N.

SENDER SIDE:

```
import socket
import random
import time

s = socket.socket()
s.bind(("localhost", 1450))
s.listen(5)
c, adr = s.accept()
print(str(adr))
n = int(input("Enter number of frames: "))
N = int(input("Enter window size: "))
seq = 1  # is used to keep track of the window starting
frame = 1  # frame to send starts with 1

# send first N window size frames
for i in range(N):
    print('Frames sent ->', frame)
    c.send(str(frame).encode())
    frame += 1
    time.sleep(2)

timer = 5

# will start with acknowledgement frame of 1
while frame <= n:
    t = random.randint(1, 7)
    msg = c.recv(1).decode()
    msg = int(msg)

    if (msg != seq):
        # here we try to discard the already sent frames after failed frame
        continue
```

```
    if (timer > t):
        # if the timer is greater than random number be consider it as ack
        print("acknowledgement received")
        print('Frames sent ->', str(frame))
        # we will send next frame
        c.send(str(frame).encode())
        seq += 1
        frame += 1
        time.sleep(2)
    else:
        # if timer is less than the random number we consider as not received ack
        print('acknowledgement not received')
        frame = seq
        # we will again send the frames from window starting i.e seq
        for i in range(N):
            print('Frames sent ->', frame)
            c.send(str(frame).encode())
            frame += 1
            time.sleep(2)
```

RECEIVER SIDE:

```
import socket
import time
s=socket.socket()
s.connect(("localhost", 1450))
while 1:
    msg=s.recv(2).decode()
    print("Received --> ",int(msg))
    s.send(str(msg).encode())
    time.sleep(1)
```

7. Write a Program to implement Sliding window protocol for Selective repeat.

SENDER SIDE:

```
import socket
import random
import time
s = socket.socket()
s.bind(("localhost",8038))
s.listen(5)
c, adr = s.accept()
```

```
print(str(adr))
n  = int(input("Enter number of frames: "))
N = int(input("Enter window size: "))
seq = 1 # is used to keep track of the window starting
frame = 1  # frame to send starts with 1

# send first N window size frames
for i in range(N):
    print('Frames sent ->',frame)
    c.send(str(frame).encode())
    frame += 1
    time.sleep(2)

timer = 5

# will start with acknowledgement frame of 1
while frame <= n :
    t = random.randint(1,7)
    msg = c.recv(1).decode()
    msg = int(msg)
    print("Frame ", msg)
    if(timer > t):
        # if the timer is greater than random number be consider it as ack
        print("acknowledgement received")
        print('Frames sent ->',str(frame))
        # we will send next frame
        c.send(str(frame).encode())
        seq += 1
        frame += 1
        time.sleep(2)
    else:
        # if timer is less than the random number we consider as not received ack
        print('acknowledgement not received')
        # we will again send the frames from window starting i.e seq
        print('Frames sent ->',seq)
        c.send(str(seq).encode())
        time.sleep(2)
```

RECEIVER SIDE:

```
import socket
import time
s=socket.socket()
s.connect(("localhost", 8038))
while 1:
    msg=s.recv(2).decode()
    print("Received --> ",int(msg))
    s.send(str(msg).encode())
    time.sleep(1)
```