Computer Network

Name : Antuley Aman Siraj.

Roll No. : 23CO25.

Batch : 01

Experiment - 04

Aim : To implement Stop-and-Wait Automatic Repeat Request (ARQ) protocol in Python using UDP sockets, ensuring reliable data transmission between a sender and receiver over an unreliable network.

Theory :

The Stop-and-Wait ARQ is a simple flow control protocol used in computer networks.

- The sender transmits one frame and waits for an acknowledgment (ACK) before sending the next frame.
- If the ACK is not received within a timeout period, the sender retransmits the same frame.
- The receiver sends an ACK for each correctly received frame.
- Sequence numbers (0 and 1) are used to identify duplicate frames.

This ensures reliable data transfer even over an unreliable channel.

Example :

Messages to send:
```
["Hello", "World", "This", "Is", "Stop-and-Wait"]
```

Expected behavior:

- **Sender sends a message with a sequence number.**
- **Receiver acknowledges the message.**
- **If acknowledgment is lost or delayed, sender retransmits.**

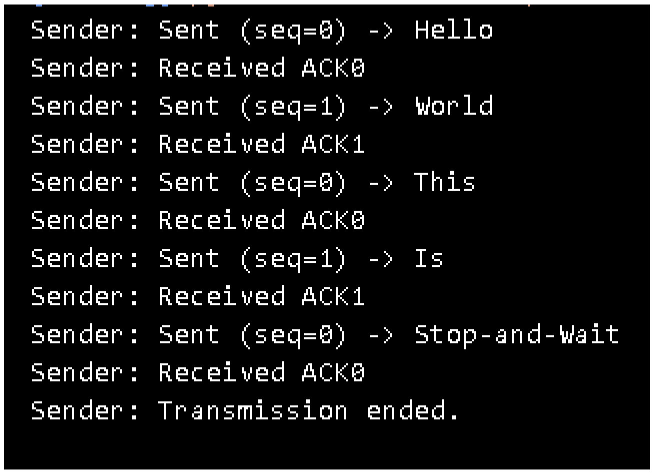
Code : (Sender)

```python
import socket

import time

# Create UDP socket

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

sock.settimeout(2)  # 2 seconds timeout

server_address = ('127.0.0.1', 12345)

messages = ["Hello", "World", "This", "Is", "Stop-and-Wait"]

seq_num = 0

for msg in messages:

    while True:

        packet = f"{seq_num}:{msg}"

        sock.sendto(packet.encode(), server_address)

        print(f"Sender: Sent (seq={seq_num}) -> {msg}")

        try:

            data, _ = sock.recvfrom(1024)

            ack = data.decode()

            if ack == f"ACK{seq_num}":

                print(f"Sender: Received {ack}")

                seq_num = 1 - seq_num

                break

            else:

                print(f"Sender: Unexpected ACK {ack}, resending...")
        except socket.timeout:

            print("Sender: Timeout, resending...")

# End transmission

sock.sendto(b"END", server_address)

print("Sender: Transmission ended.")

sock.close()
```

(Reciver)

```
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

sock.bind(('127.0.0.1', 12345))

print("Receiver ready...")

expected_seq = 0

while True:

    data, addr = sock.recvfrom(1024)

    message = data.decode()

    if message == "END":

        print("Receiver: Transmission ended by sender.")

        break

    try:

        seq_num, payload = message.split(":", 1)

        seq_num = int(seq_num)
        if seq_num == expected_seq:
            print(f"Receiver: Got (seq={seq_num}) -> {payload}")
            ack = f"ACK{seq_num}"
            sock.sendto(ack.encode(), addr)
            expected_seq = 1 - expected_seq
        else:
                print(f"Receiver: Unexpected seq={seq_num}, resending last ACK.")
            ack = f"ACK{1 - expected_seq}"
            sock.sendto(ack.encode(), addr)
    except ValueError:
        print("Receiver: Invalid packet format")
sock.close()
```

Output :



```
Sender: Sent (seq=0) -> Hello
Sender: Received ACK0
Sender: Sent (seq=1) -> World
Sender: Received ACK1
Sender: Sent (seq=0) -> This
Sender: Received ACK0
Sender: Sent (seq=1) -> Is
Sender: Received ACK1
Sender: Sent (seq=0) -> Stop-and-Wait
Sender: Received ACK0
Sender: Transmission ended.
```

Conclusion : The Stop-and-Wait ARQ protocol was successfully implemented in Python using UDP sockets. The system ensures reliable delivery by retransmitting packets upon timeout and uses sequence numbers to avoid duplicate frame acceptance.