|  |
| --- |
| 1. **Purpose of experiment** 2. Learn to use socket socket programming to write stop and wait protocol, understand the working principle and characteristics of stop and wait protocol 3. Understand the working principle of the stop protocol based on UDP socket programming, and learn to use the UDP architecture to write the server and client of the stop protocol 4. **Experimental principle**   Stop-and-wait protocol is the simplest Automatic Repeat request (ARQ) protocol. It is a connection-oriented transport protocol based on reliable transmission. This experiment will build the server (sender) and client (receiver) of the stop and wait protocol based on UDP.  .  The basic idea of the stop-and-wait protocol is that after the sender sends a data packet, it stops sending and waits for an acknowledgment from the receiver. After receiving the acknowledgement, the next data packet is sent. If the acknowledgment is lost or timed out, the sender retransmits the previous data packet and does not request the next data packet from the sender until the receiver has correctly received the previous data packet. The stop-and-wait protocol has strict requirements on the order of receiving data packets, and does not allow data packets to be sent and received out of order.  The advantage of the stop-wait protocol is that it is simple to implement and easy to understand and program. The disadvantages are low efficiency, low utilization, sensitive to delay, and unable to fully utilize the bandwidth of the channel.  停等协议的性能指标有：   1. Throughput: The effective amount of data sent per unit time, related to the size of the data packet and the Round-trip time (RTT). 2. Delay: the time from the sender sending the data packet to the receiver receiving the acknowledgement, which is related to the propagation delay and processing delay of the channel. 3. Packet loss rate: the proportion of data packets sent by the sender that are not correctly received by the receiver, which is related to the reliability of the channel and the timeout setting   **(Because the packet loss rate is small in the real situation, which is not conducive to the observation of the experiment, this experiment manually simulated packet loss by generating random numbers.)**   1. **Content**   **Server：**  **(1) Creating TCP/UDP socket**  **(2) Receiving data from client**  **(3) Return Ack, you can define an Ack manually**  **Client：**  **(1) Creating TCP/UDP socket**  **(2) Sending a series of packet to server, and the number of packet can be assigned**  **(3) Receiving Ack from server and counting**  **(4) Setting a timeout**  **(5) If timeout, the packet should resend packet**  **(6) After all packet are sent, print time and paket loss rate**   1. **communication in “localhost”** 2. **Sending 10 packets, packet loss rate is set to 0.5** 3. **Sending 100 packets, packet loss rate is set to 0.2** 4. **Sending 100 packets, packet loss rate is set to 0.5** 5. **Communication with partner**   **1.client and server 10 packets ，loss rate 0.5**    Fig1. Set the loss rate to 0.5    Fig2.The result of transmitting 10 packets  **2.client 和 server 100 packet loss rate 0.2**    Fig3.Set the loss rate to 0.2    Fig4. The result of transmitting 100 packets with 0.2 loss rate  **2.client 和 server 100 packet loss rate 0.5**    Fig5.Set the loss rate to 0.5    Fig6. The result of transmitting 100 packets with 0.5 loss rate |
| 1. **Conclusion and discussion**   Through this experiment, we have learned the basic principle and workflow of the stop protocol, mastered the basic methods and skills of socket programming in the python environment, and analyzed the working principle and characteristics of the stop protocol.  We find that the stop-and-wait protocol is inefficient, sensitive to delay and cannot fully utilize the bandwidth of the channel due to its high requirement on the order of data packets. In the actual network environment, the stop-and-wait protocol is not suitable for the transmission of a large number of data or long distance communication, but the stop-and-wait protocol is suitable for the transmission of a small amount of data, short distance and high data integrity and accuracy requirements of the communication.  In order to improve the efficiency and reliability of transmission, we can use other protocols, such as sliding window protocol, selective retransmission protocol, etc. These protocols can improve the utilization and throughput of the channel by allowing the sender to send multiple data packets without waiting for the acknowledgement of each packet. In the actual network transmission, the stop-and-wait protocol is mixed with a variety of different transmission protocols, according to the size of the data transmission, distance and network delay and other elements to select the most suitable transmission method, to achieve the fastest speed, maximum efficiency and can use the bandwidth of the channel as much as possible for transmission. |