Reliable Packet Transmission with UDP

Yuan-Ping Chang | Software Engineer

I - Introduction

UDP (User Datagram Protocol) is an unconnected transport layer protocol in the OSI (Open System Interconnection) reference model, providing a transaction-oriented simple unreliable information transfer service. It is called "unreliable" because there is no handshake or verification of the data transfer.

It is not a connection type protocol, so it has the advantages of low resource consumption and fast processing speed. Therefore, audio, video, and normal data usually use UDP when transmitting, because losing even one or two data packets occasionally is not that big of a deal. We barely notice if a packet drops from a video stream or call.

UDP does not have a TCP handshake, acknowledgment, window, retransmission, congestion control, etc. UDP is a stateless transport protocol, so it is very fast when passing data. Without the mechanisms of TCP, UDP is less vulnerable to exploits than TCP.

II - Background

Due to the unreliability of UDP, it has been limited in some applications. Therefore, we have proposed a reliable UDP transmission mechanism to solve this problem. It can also be applied to other applications besides high-speed data transmission, such as Point-to-point technology (P2P), firewall penetration, multimedia data transmission, and so on.

III - How Reliable Packet Transmission With Udp Works

The sending and receiving parties are divided into a connection initiator and a receiver, and data transmission is performed through Reliable Packet Transmission with UDP.

The connection initiator first sends a UDP packet with a packet type of HELLO to the receiving end, and the source sequence number is a non-repeating sequence number generated by the connection initiator, which is only applicable to the conversation. The packet sequence number is 0.

When receiving the UDP packet whose packet type is HELLO, the receiving end replies with a UDP packet whose packet type is HELLO ACK. The source sequence number is the sequence number sent by the connection initiator, and the packet sequence number is 0.

After receiving the UDP packet with the packet type HELLO ACK, the connection initiator sends a reply to the UDP packet of the same content, and the connection is established. The connection initiator can start sending the UDP packet with the packet type DATA.

At the end of the connection, either end can send a UDP packet with a packet type of BYE. The source sequence number is the sequence number of the session phase, and the packet sequence number is 0. The other party will reply to the same content UDP packet after receiving it, and both parties can end the dialogue phase. This is illustrated in Fig.1.

Data Transmission Phase

Both the originating end and the receiving end can transmit the data packet to the other party. The UDP packet of the packet type is DATA, the source serial number is the sequence number of the dialogue phase, the packet length is the length of the data to be transmitted, and the packet sequence number starts from 0—incrementally in order. After receiving the data packet transmitted by the other party, it must reply to the UDP packet whose packet type is DATA ACK. The source sequence number is the sequence number of the session phase, and the packet sequence number is the packet sequence number of the data packet received at the time.

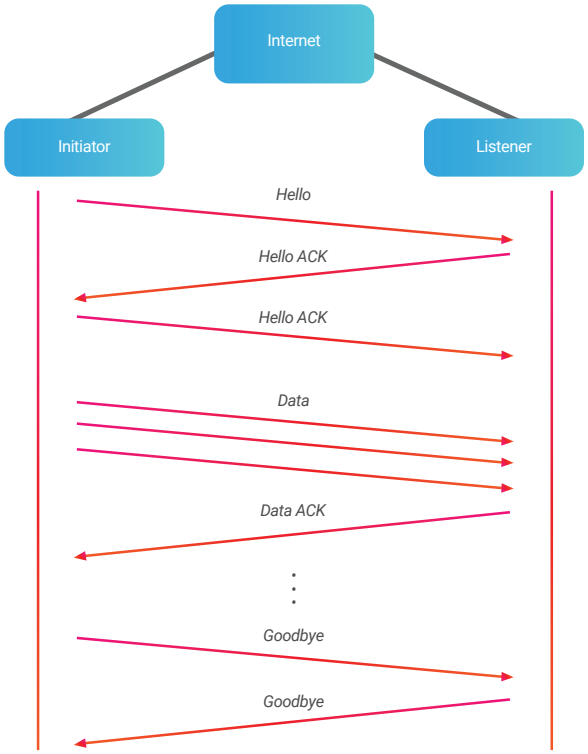

Fig.1.

ERROR Detection and Correct

In the data transmission phase, after the data packet is sent out, if the data packet is not received, the DATA ACK packet for the data packet must be resent to avoid the loss of the data packet during the transmission. The data receiving end shall establish a queue. After receiving the UDP packet of the packet type DATA, the data is temporarily stored in the queue according to the packet serial number. If the serial number is found to be misaligned, the data may be adjusted and then read sequentially. This is how to avoid data errors caused by different data packet arrival orders due to network delay.

Send Window

In the data transmission phase, the data transmitter defines the Send Window Size, and sends multiple data packets in sequence. After receiving the DATA ACK packet from the receiving end, it continues to transmit multiple data packets within the Send Window Size range instead of transmitting a single data packet and waiting for a DATA ACK to be sent before sending the next data packet. When the network is delayed, the Send Window can greatly reduce the time wasted by the data sender waiting for the DATA ACK packet. This is illustrated in Fig.2.

Slow ACK

n the data transmission phase, the data receiving end does not return the DATA ACK packet for each data packet, but after a fixed time, the DATA ACK packet is sent back and forth to the last data packet received, and the Send Window can reduce the transmission of a large amount of data.

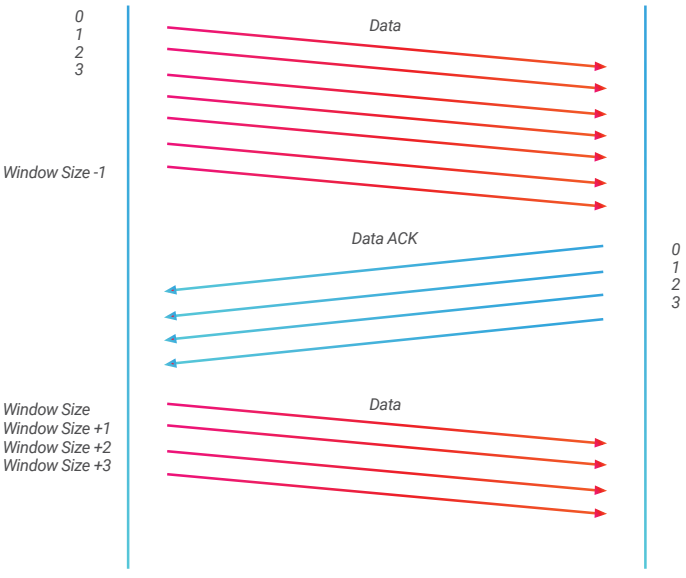

Fig. 2.

The DATA ACK packet traffic caused by the time. This is illustrated in Fig 3..

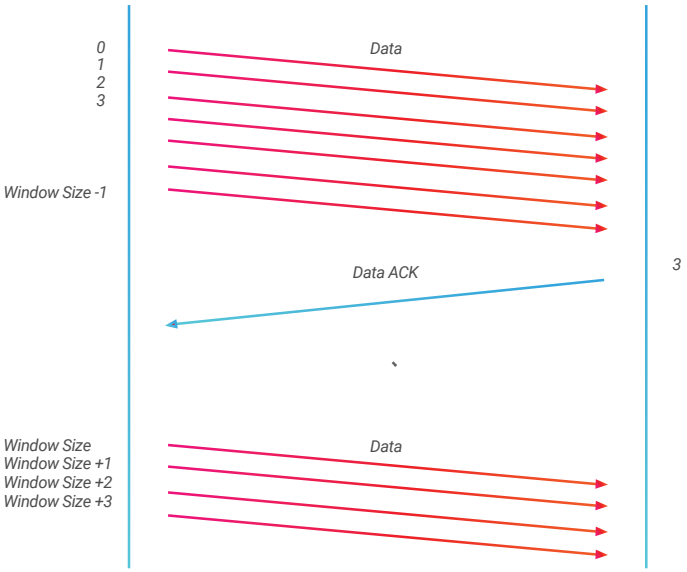

Fig. 3.

Slow Start

The data sender's Send Window Size will be adjusted to the appropriate size according to the network conditions. At the beginning of the data transfer, the data sender's Send Window Size will increase from the defined minimum value to the maximum value over time, but the Send Window Size will be moderately reduced when the data transfer party detects that a packet loss occurs. To avoid the network congestion, the data transmitter continues to send a large number of data packets to make the network situation worse.

V - Conclusion

Different technologies can be selected according to different transmission scenarios. It is possible to choose a loose congestion mode or a specific retransmission mode, but no matter how you choose, it is based on Expense, Latency, and Quality. The trade-offs between the three can help developers find a better solution by combining the scenarios and balancing the triangular balance.

EnGenius Technologies | 1580 Scenic Ave. Costa Mesa, CA 92626

Email: partners@engeniustech.com | Website: engeniustech.com

Version: 1.00 04/2020