TCP

Intro

TCP stands for Transmission Control Protocol, it is a connection oriented, reliable, byte stream based transport layer communication protocol.

When we are talking about the layer that TCP belongs to, let's go over the seven layers we have already learnt.

Physic layer, data link layer, net work layer, transmission layer, session layer, presentation layer, application layer.

TCP is designed to provide reliable end-to-end byte stream over unreliable Internet, reminded that, "end-to-end", so that means we should have some special techniques to address the machines and even determine which service you should apply for.

So, together with these restrictions, it's natural for us to think about another well-known protocol: IP-protocol. IP-protocol tells you how to find out the proper machine and port, then the TCP protocol ensure that this connection is stable and all the data transmitted via this connection is intact. Well, we cannot talk too much about it or we will snatch what other groups going to say.

TCP provides connection oriented communication transmission, that means, we have to make some preparations between the two ends before data communication starts. Let's see, how the connection work with the TCP/IP protocol.

When you want to set up a connection, you need three handshakes:

1, The first handshake:

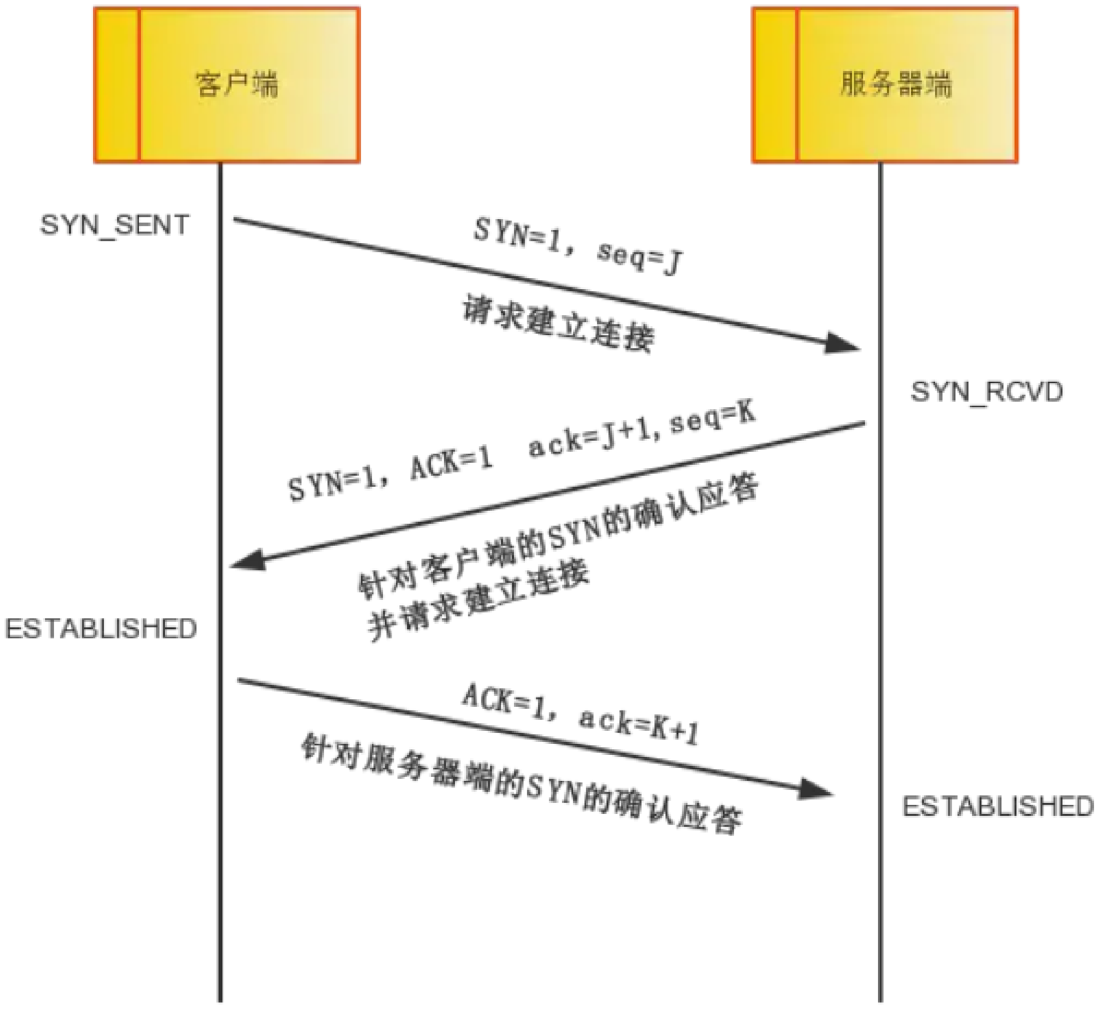
the client sets the flag SYN to 1, randomly generates a value seq = J, and sends the packet to the server, and the client enters SYN_SENT status, waiting for the server to confirm.

2, The second handshake:

after the server receives the packet, the flag bit SYN = 1 knows that the client requests to establish a connection. The server sets the flag bit SYN and ACK to 1, ack = j + 1, randomly generates a value seq = K, and sends the packet to the client to confirm the connection request, and the server enters SYN_ RCVD status.

3, The third Handshake:

after receiving the confirmation, the client checks whether the ack is j + 1 and whether the ACK is 1. If it is correct, set the flag bit ACK to 1 and ack = K + 1, and send the packet to the server. The server checks whether the ack is K + 1 and whether the ACK is 1. If it is correct, the connection is established successfully. The client and the server enter the ESTABLISHED state and complete the handshake for three times Data transfer can begin between the server and the server.



How to transmit data? Obviously, it's by slide window. There are still something interesting in data transmission.

Another key word in TCP is "stable", which means that we must make sure there's no duplicate, no disorder, no loss of data.

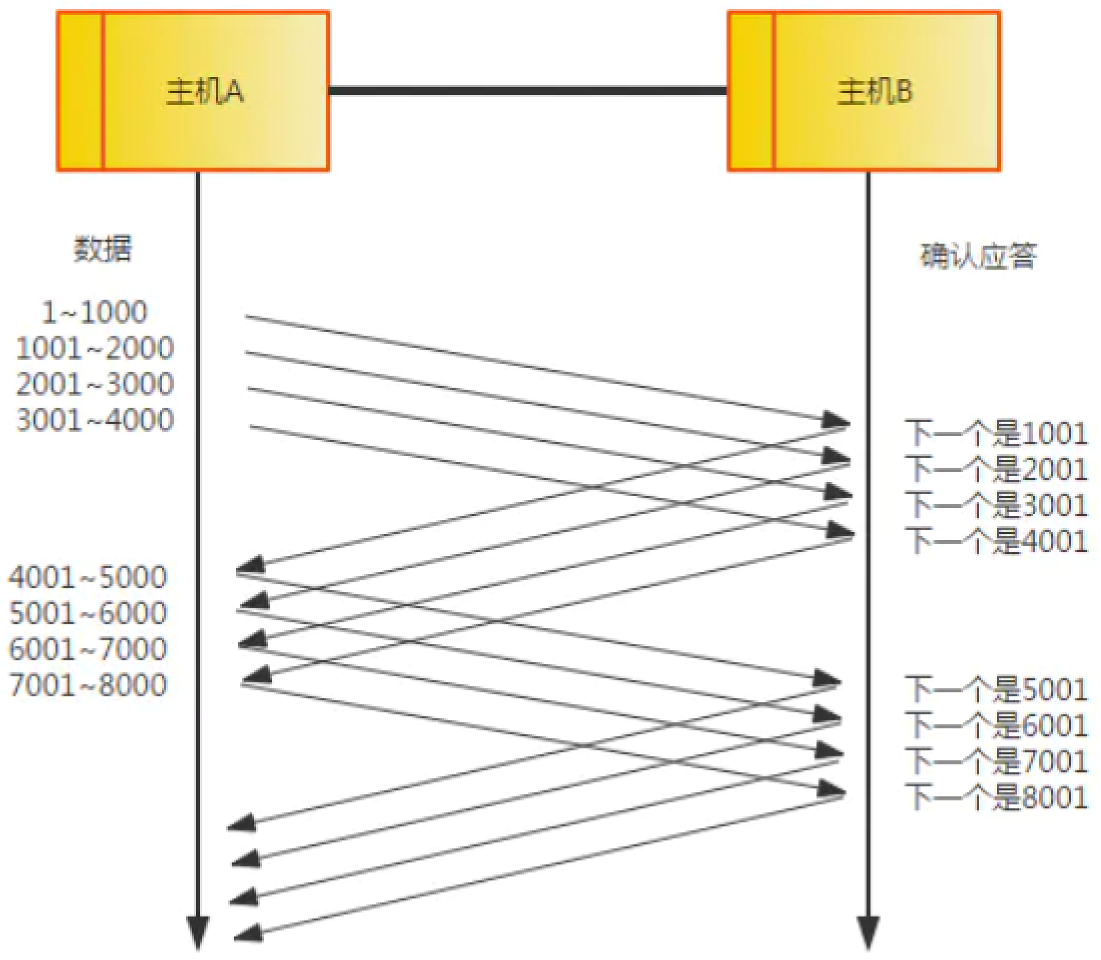
Well, actually, the easiest problem is disorder, since we can solve it by sorting all the posts after receiving all of them.

Similarly, to remove duplicate packs, we just need to add a sequence number in the post, and the server can maintain the posts it has received.

How about...dealing with data loss?

In TCP protocol, the maximum data size is defined as Max message size, which is equal to the size of message in IP protocol that will not be partitioned. Then, the data going to transmit with TCP protocol will be divided into several segments.

Well, if one segment is taken as the unit, and each segment is sent for a confirmation response seems inefficient, so in TCP, The confirmation response is no longer in each segment, but in a larger unit, and the forwarding time will be greatly reduced. In other words, the sending host does not have to wait for a confirmation response after sending a segment, but continues to send. Like this:



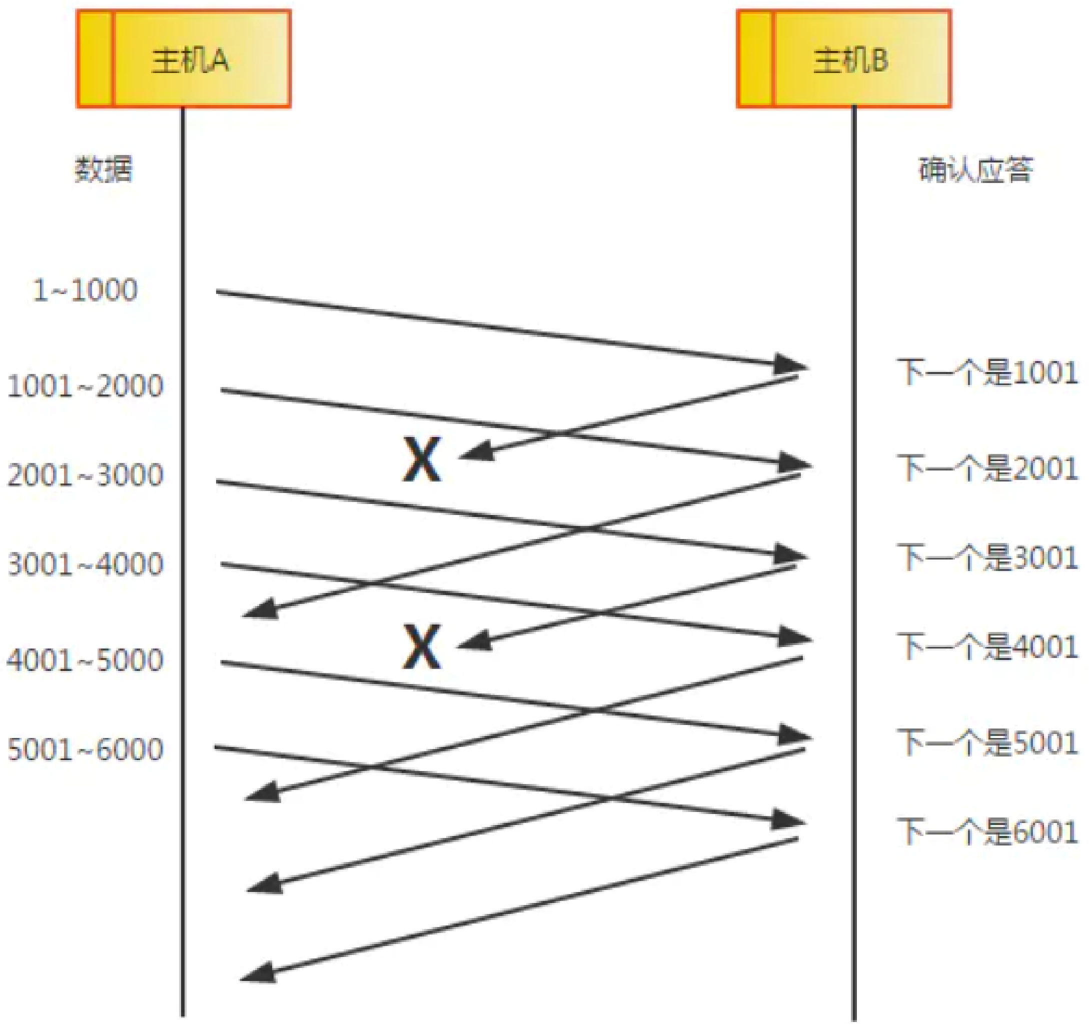
根据窗口为4000字节时返回的确认应答，下一步就发送比这个值还要大4000个序列号为止的数据。这跟前面每个段接收确认应答以后再发送另一个新段的情况相比，即使往返时间变长也不会影响网络的吞吐量。

Since data can be sent even if no acknowledgement is received, however, before the confirmation response of the whole window does not arrive, if some of the data is lost, the sender is still responsible for retransmission. To this end, the sending host needs to set up a cache to keep the data to be retransmitted until it receives their confirmation.

When there's data lost, it has two cases:

1, loss of ACK, actually the data has arrived.

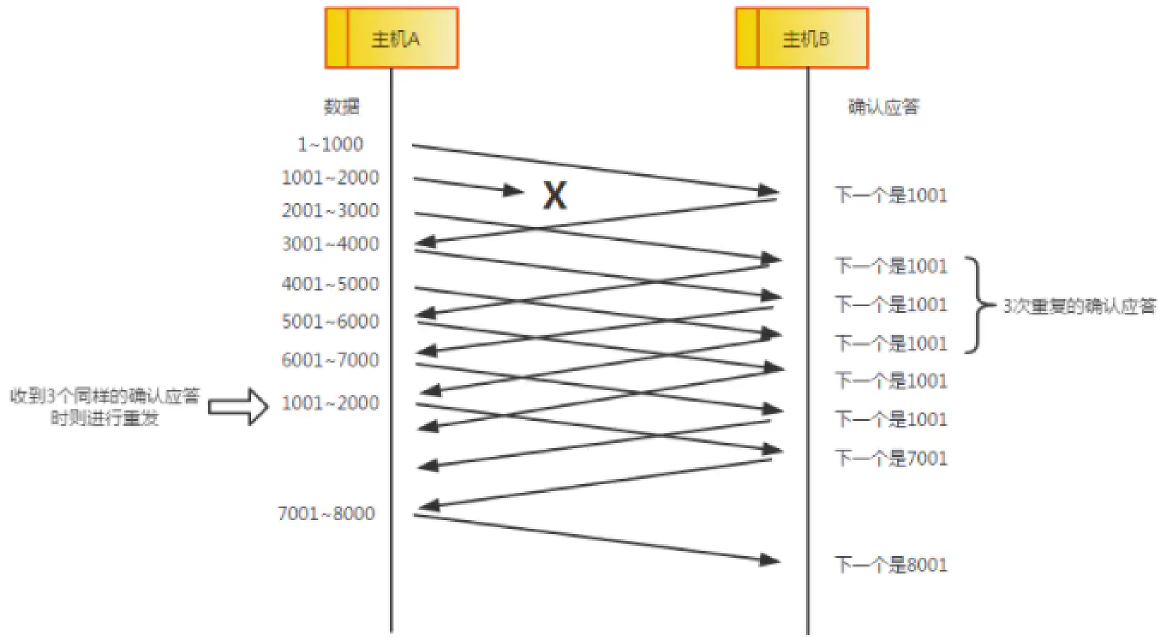
In this case, it has no problem, since we can infer the status from the next ACK, if we loss the ACK of last segment, successfully received the ACK of next segment, the last segment must be sent successfully.



窗口在一定程度上较大时，即使有少部分的确认应答丢失也
不会进行数据重发，可以通过下一个确认应答进行确认。

2, loss of a segment

When a message segment is lost, the sender will always receive the acknowledgement response with the serial number of last success segment. Therefore, when the sliding window is large enough and the message segment is lost, the acknowledgement response with the same serial number will be returned repeatedly. If the sender receives the same acknowledgement for three consecutive times, it will resend the corresponding data. This mechanism is more efficient than the time-out management mentioned before, so it is also called high-speed retransmission control.



接收端在没有收到自己所期望序号的数据时，会对之前收到的数据进行确认应答。发送端则一旦收到某个确认应答后，又连续3次收到同样的确认应答，则认为数据段已经丢失，需要进行重发。

When you want to close a connection, you need four handshakes:

Unlike setting up a connection, the disconnection end can be the client side or the server side. Since the TCP connection is full duplex, each direction must be closed separately. When a terminal completes the data transmission task, it sends a FIN to terminate the connection in this direction. Receiving a FIN only means that there is no data flow in this direction, that is, no data will be received. However, data can still be sent on this TCP connection until Fin was also sent in another direction. The first part to close will perform an active shut down, while the other part will perform a passive shut down.

1, The first handshake:

the client sends a FIN = M, which is used to close the data transmission from the client to the server, and the client enters the FIN_ WAIT_ 1 status. It means "my client has no data to send to you", but if your server still has data to send, you don't need to close the connection in a hurry, you can continue to send data.

2, The second wave:

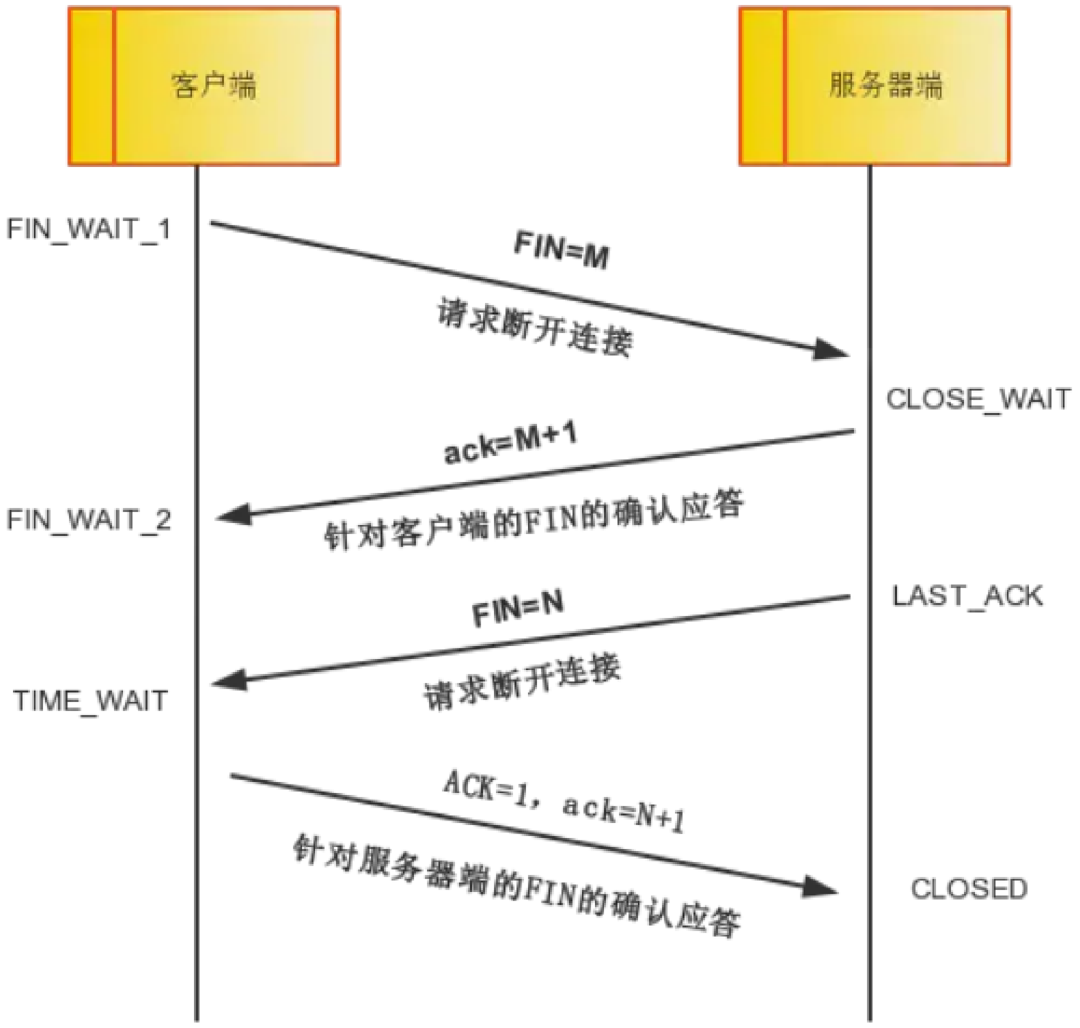
after the server receives FIN, it sends ack = M + 1, and tells the client that I have received your request, but I am not ready. Please continue to wait for my message. At this time, the client enters FIN_ WAIT_ 2 status, continue to wait for the fin message of the server.

3, The third wave:

when the server determines that the data has been sent, it will send the FIN = N message to the client, and tell the client, OK, my side of the data has been sent, and I am ready to close the connection. The server enters LAST_ ACK status.

4, The fourth wave:

after receiving the FIN = N message, the client knows that it can close the connection, but he still doesn't trust the network, for fear that the server does not know to close it, so it sends ack = N + 1 and enters TIME_ WAIT status. If the server does not receive the ACK, it can retransmit. After receiving the ACK, the server knows that it can disconnect. If the client does not receive a reply after waiting for 2MSL(Maximum segment survival), it proves that the server has been shut down normally. Well, my client can also close the connection. Finally, four handshakes were completed.



After we know that how the TCP/IP protocol works, we can bring it to a brief conclusion, TCP has following characteristics:

1, Stream based mode

2, Connection oriented, any transmission rely on this connection.

3, Reliable, make sure that no duplicate, no disorder, no loss.

4, When the network condition is not good, the bandwidth overhead caused by retransmission should be reduced as much as possible

5, Connection maintenance is oriented to the two endpoints of communication, regardless of the intermediate network segment and node.

Giving those characteristics, what kinds of application is it fit for? How about, visiting a website? Or, online communication?