Report of Project4

Visualizing Sliding Window and Flow Control for TCP Via Python

Qiong Yang

1 Principles of Reliable Data Transfer

rt3.0 is good, but it's performance is very bad. The key of bad performance is stop-and-wait protocol. To solve this problem, people think out of a simple solution: givie up the stop-and-wait protocol and admit sending the multiple packages at the same time without waiting ack. This technique is called pipelining. Pipelining has the following consequences for reliable data transfer protocols:

- The range of sequence numbers must be increased, since each in-transit packet (not counting retransmissions) must have a unique sequence number and there may be multiple, in-transit, unacknowledged packets.

- The sender and receiver sides of the protocols may have to buffer more than one packet. Minimally, the sender will have to buffer packets that have been transmitted but not yet acknowledged. Buffering of correctly received packets may also be needed at the receiver, as discussed below.

- The range of sequence numbers needed and the buffering requirements will depend on the manner in which a data transfer protocol responds to lost, corrupted, and overly delayed packets. Two basic approaches toward pipelined error recovery can be identified: Go-Back-N and selective repeat.

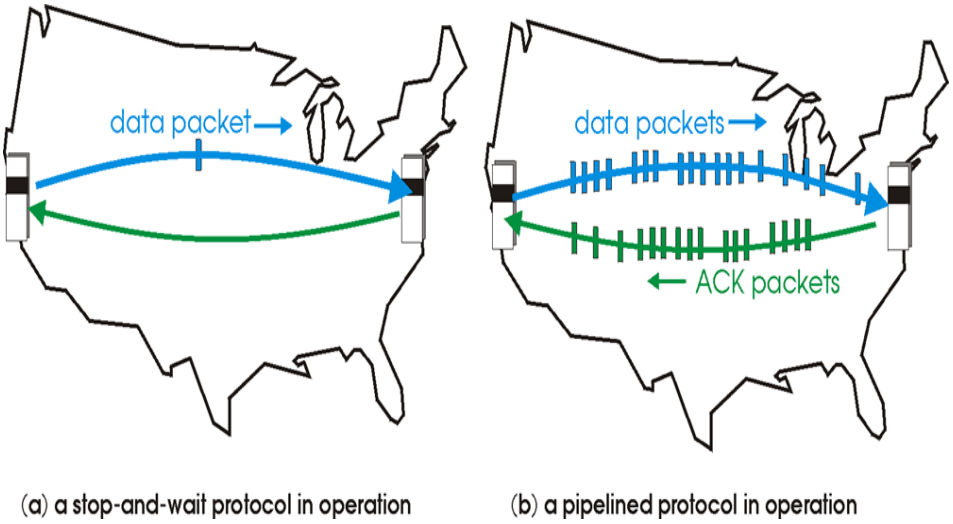

Figure 1: Stop-and-wait and pipelined protocol

Next, we will analyse the principles of Go-Back-N and Selective repeat.

1.1 Go-Back-N

In a Go-Back-N (GBN) protocol, the sender is allowed to transmit multiple packets (when available) without waiting for an acknowledgment, but is constrained to have no more than some maximum allowable number, N, of unacknowledged packets in the pipeline.
In GBN, we define textbfbase to be the sequence number of the oldest unacknowledged packet and textbfnextseqnum to be the smallest unused sequence number (that is, the sequence number of the next packet to be sent) We can devide the range of sequence numbers into four intervals:[0,base-1],[base,nextseqnum-1],[nextsequem,base+N-1] and [base+N,default],where

- interval [0,base-1]:correspond to packets that have already been transmitted and acknowledged
- interval [base,nextseqnum-1]:corresponds to packets that have been sent but not yet acknowledged
- interval [nextsequem,base+N-1]:be used for packets that can be sent immediately
- interval [base+N,default]:cannot be used until an unacknowledged packet currently in the pipeline (specifically, the packet with sequence number base) has been acknowledged.

Figure 2 shows the senders view of the range of sequence numbers in a GBN protocol.

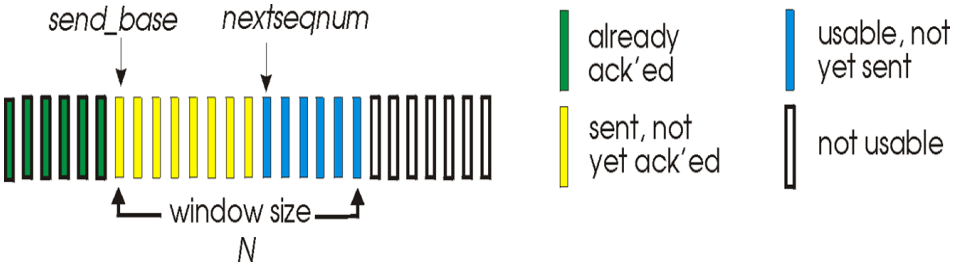

Figure 2: Sender's view of sequence numbers in Go-Back-N

1.2 Simulation Result

Firstly, I use Fig.3 to explain the meaning of rectangulars in simulation result.

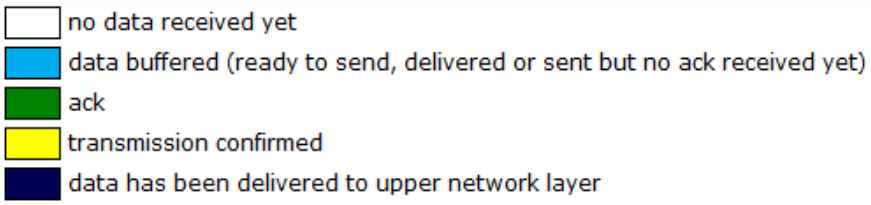

Figure 3: the meaning of rectangulars in simulation result

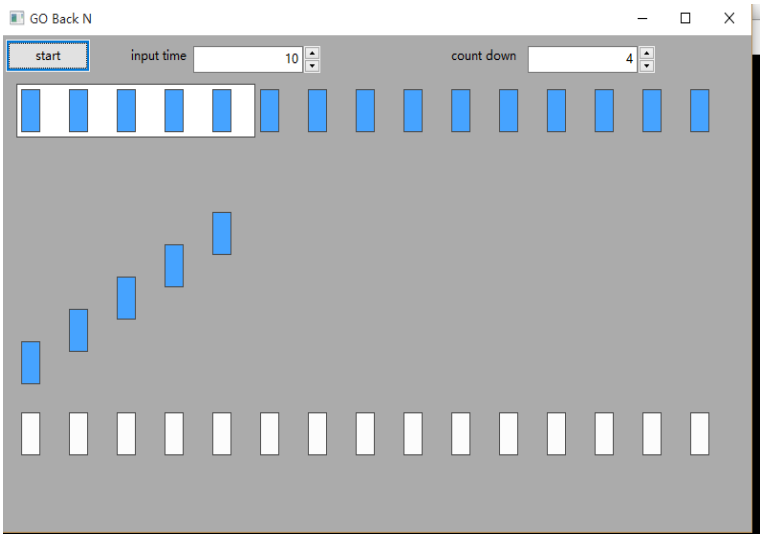

Figure 4: simulation result of Go-Back-N

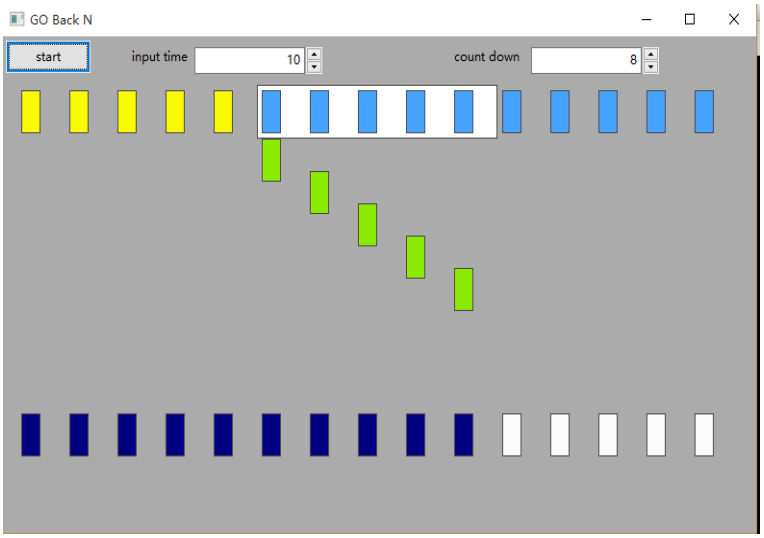

Figure 5: simulation result of Go-Back-N

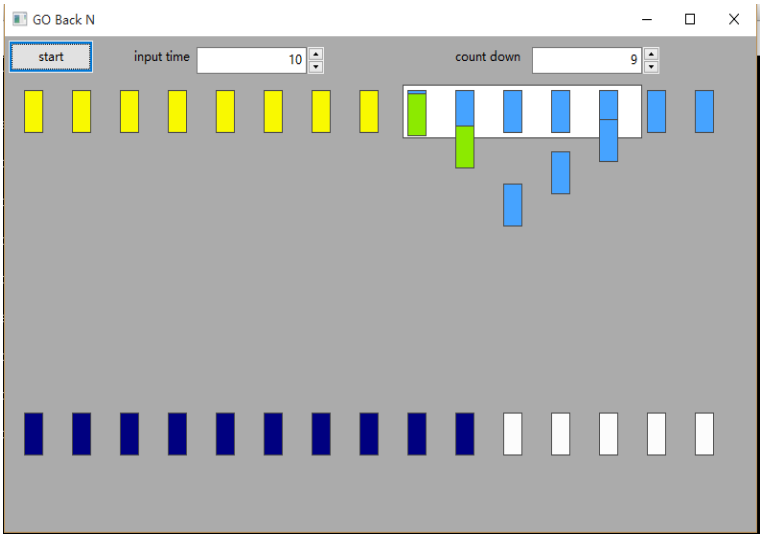

Figure 6: simulation result of Go-Back-N

2 Flow Control

We know that the hosts on each side of a TCP connection set aside a receive buffer for the connection. When the TCP connection receives bytes that are correct and in sequence, it places the data in the receive buffer. The associated application process will read data from this buffer, but not necessarily at the instant the data arrives. Indeed, the receiving application may be

busy with some other task and may not even attempt to read the data until long after it has arrived. If the application is relatively slow at reading the data, the sender can very easily overflow the connections receive buffer by sending too much data too quickly.
TCP provides a flow-control service to its applications to eliminate the possibility of the sender overflowing the receivers buffer. Flow control is thus a speed-matching servicematching the rate at which the sender is sending against the rate at which the receiving application is reading. As noted earlier, a TCP sender can also be throttled due to congestion within the IP network; this form of sender control is referred to as congestion control.

2.1 Simulation Result

For convenience of show the change of buffer , I assume that there are three buffers and applications in sender and receiver. However, in reality, three buffers respectively are the same.

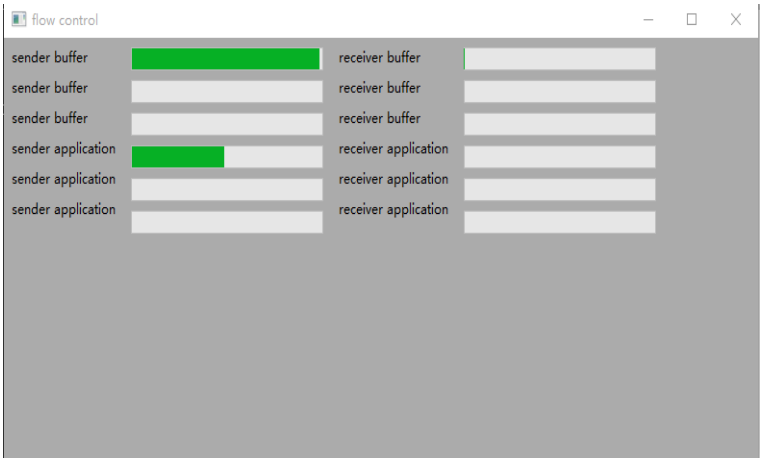

Figure 7: Sender puts 2048 bites to buffer

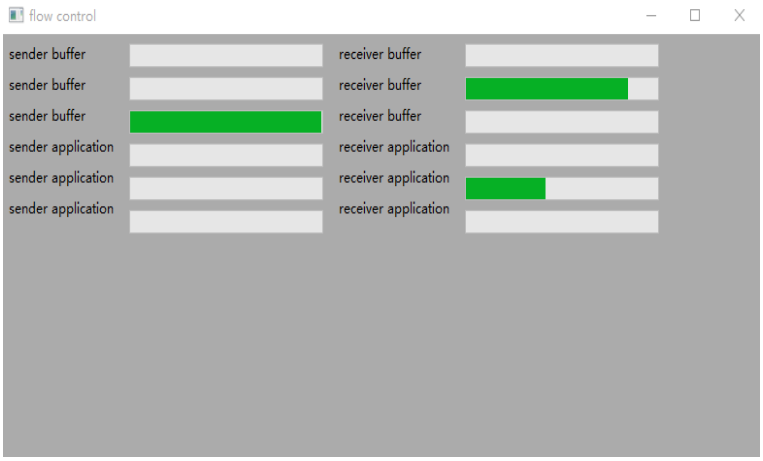

Figure 8: Sender puts 2048 bites to buffer again and app consumes 2048 bites

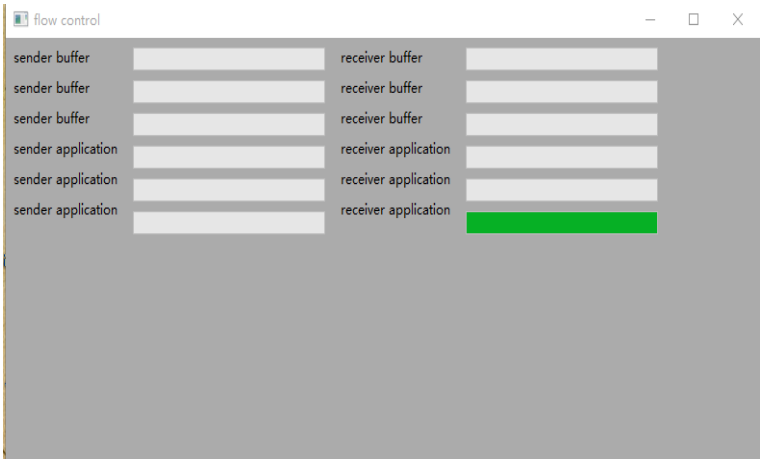

Figure 9: App consumes 2048 bites again

3 Conclusion

By the project, I learn more about the principle of Go Back N and flow control. At the same time, My skill of programming has been enforced.