

Pipelined Reliable Data Transfer Protocols

Protocol rdt3.0 is a functionally correct protocol, but it is doubtful that anyone would be happy with its performance, especially in today's high-speed networks. At the heart of rdt3.0's performance problem is the fact that it is a stop-and-wait protocol.

To appreciate the performance impact of this stop-and-wait behavior, look at an idealized case of two hosts, one located on the West Coast of the United States and the other located on the East Coast, as shown in Figure 2. The speed-of-light round-trip propagation delay between these two end systems, RTT, is approximately 30 milliseconds. Suppose that they are connected by a channel with a transmission rate, R , of 1 Gbps (10⁹ bits per second). With a packet size, L , of 1,000 bytes

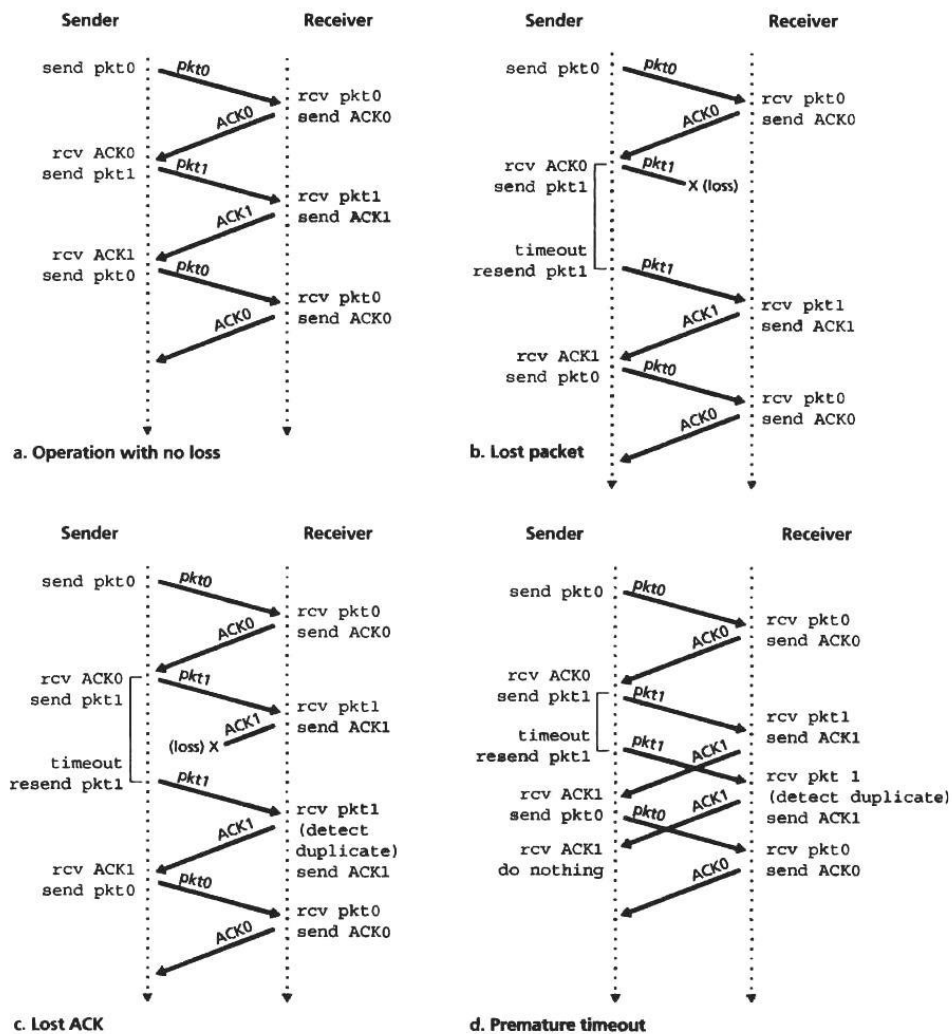


Figure 1. Operation of rdt3.0, the alternating-bit protocol

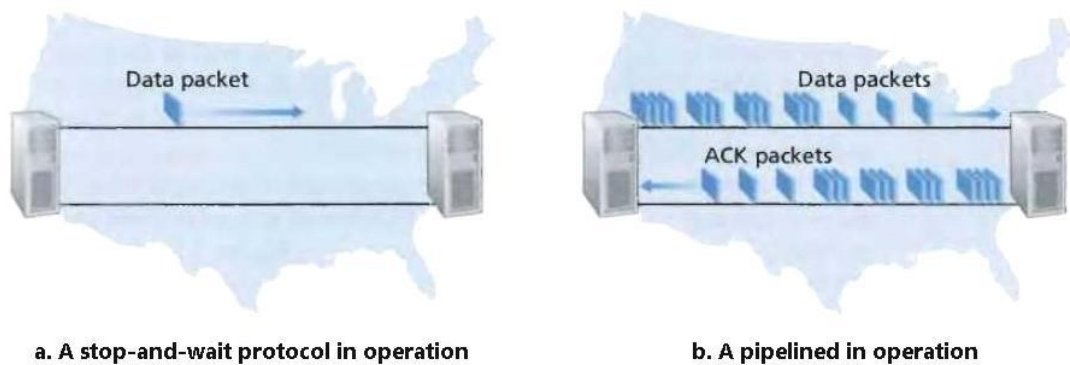


Figure 2. Stop-and-wait versus pipelined protocol

(8,000 bits) per packet, including both header fields and data, the time required to actually transmit the packet into the 1 Gbps link is

$$d_{trans} = \frac{L}{R} = \frac{8000 \text{ bits/packet}}{10^9 \text{ bits/sec}} = 8 \text{ microseconds}$$

"Go-Back-N (GBN)" Figure 1(a) illustrates that with our stop-and-wait protocol, if the sender begins sending the packet at $t = 0$, then at $t = L/R = 8$ microseconds, the last bit enters the channel at the sender side. The packet then makes its 15-msec cross-country journey, with the last bit of the packet emerging at the receiver at $t = RTT/2 + L/R = 15.008$ msec. Assuming for simplicity that ACK packets are very small (so that we can ignore their transmission time) and that the receiver can send an ACK as soon as the last bit of a data packet is received, the ACK emerges back at the sender at $t = RTT + L/R = 30.008$ msec. At this point, the sender can now transmit the next message. Therefore, in 30.008 msec, the sender was sending for only 0.008 msec. If we define the utilization of the sender (or the channel) as the fraction of time the sender is actually busy sending bits into the channel, the analysis in "Go-Back-N (GBN)" Figure 1(a) illustrates that the stop-and-wait protocol has a rather dismal sender utilization, U_{sender} , of

$$U_{sender} = \frac{L/R}{RTT + L/R} = \frac{.008}{30.008} = 0.00027$$

That is, the sender was busy only 2.7 hundredths of one percent of the time. Viewed another way, the sender was able to send only 1,000 bytes in 30.008 milliseconds, an effective throughput of only 267 kbps - even though a 1 Gbps link was available. Imagine the unhappy network manager who just paid a fortune for a gigabit capacity link but manages to get a throughput of only 267 kilobits per second. This is a graphic example of how network protocols can limit the capabilities provided by the underlying

network hardware. Also, we have neglected lower-layer protocol-processing times at the sender and receiver, as well as the processing and queuing delays that would take place at any intermediate routers between the sender and receiver. Including these effects would serve only to further increase the delay and further accentuate the poor performance.

The solution to this specific performance problem is simple: Rather than operate in a stop-and-wait manner, the sender is allowed to send multiple packets without waiting for acknowledgments, as shown in Figure 2(b). "Go-Back-N (GBN)" Figure 1(b) shows that if the sender is allowed to transmit three packets before having to wait for acknowledgments, the utilization of the sender is essentially tripled. Since the many in-transit sender-to-receiver packets can be visualized as filling a pipeline, this technique is known as 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 required at the receiver.
- 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.**

You should implement the Pipelined Reliable Data Transfer Protocols by using:

1. Go-Back-N and
2. Selective repeat.
3. In all the cases, give the new FSM.