Homework week 3

Dr. Leon Reznik -- Winter Quarter 2002
Rochester Institute of Technology -- Department of Computer Science

Review questions

1. Consider a TCP connection between host A and host B. Suppose that the TCP segments traveling from host A to host B have source port number x and destination port number y. What are the source and destination port numbers for the segments traveling from host B to host A?
2. Is it possible for an application to enjoy reliable data transfer even when the application runs over UDP? If so, how?
3. Suppose host A sends two TCP segments back to back to host B over a TCP connection. The first segment has sequence number 90; the second has sequence number 110.
    a) How much data is in the first segment?
    b) Suppose that the first segment is lost but the second segment arrives at B. In the acknowledgement that host B sends to host A, what will be the acknowledgement number?
4. Suppose two TCP connections are present over some bottleneck link of rate R bps. Both connections have a huge file to send (in the same direction over the bottleneck link). The transmissions of the file start at the same time. What is the transmission rate that TCP would like to give to each of the connections?

Problems

---

5. [Kurose & Ross, chapter 3, problem 5] In protocol rdt 3.0. the ACK packets flowing from the receiver to the sender do not have sequence numbers (although they do have an ACK field that contains the sequence number of the packet they are acknowledging). Why is it that our ACK packets do not require sequence number?
6. [Kurose & Ross, chapter 2, problem 8] Consider a channel that can lose packets but has a maximum delay that is known. Modify protocol rdt2.1 to include sender timeout and retransmit. Informally argue why your protocol can communicate correctly over this channel.

Solutions

1. Source port number y and destination port number x.
2. Yes. The application developer can put reliable data transfer into the application layer protocol. This would require a significant amount of work and debugging, however.
3. a) 20 bytes
   b) ack number = 90
4. R/2
5. To best answer this question consider why we needed sequence numbers in the first place. We saw that the sender needs sequence numbers so that the receiver can tell if a data packet is a duplicate of an already received data packet. In the case of ACKs, the sender does not need this info (i.e. a sequence number on an ACK) to tell detect a duplicate ACK. A duplicate ACK is obvious to the rdt3.0 receiver, since when it has received the original ACK it transitioned to the next state. The duplicate ACK is not the ACK that the sender needs and hence is ignored by the rdt3.0 sender.
6. We add a timer here, whose value is greater than the known round-trip propagation delay. We add a timeout event to the “wait for ACK or NAK0” and “Wait for ACK or NAK1” states. If the

timeout event occurs, the most recently transmitted packet is retransmitted. Let us see why this protocol will still work with the rdt2.1 receiver

Suppose the timeout is caused by a lost data packet, i.e. a packet on the sender-to-receiver channel. In this case, the receiver never received the previous transmission and, from the receiver's viewpoint, if the timeout retransmission is received, it looks exactly the same as if the original transmission is received.

Suppose now that an ACK is lost. The receiver will eventually retransmit the packet on a timeout. But a retransmission is exactly the same action that is taken if an ACK is garbled. Thus the sender's reaction is the same with a loss, as with a garbled ACK. The rdt2.1. receiver can already handle the case of a garbled ACK.