Faculty of Arts and Science
University of Toronto
CSC 358 - Introduction to Computer Networks

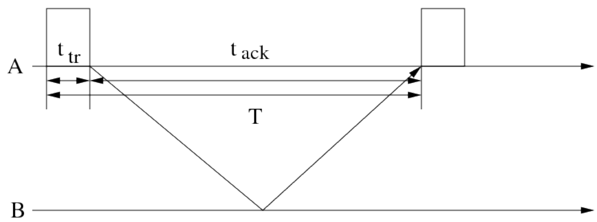
Solutions for Tutorial 3

Topic

In this tutorial we discuss how we can use ARQ protocols to implement a congestion control algorithm (as used in TCP), as well as ARQ protocols under different assumptions on the communication channel.

Question 1: Efficiency and Congestion Control
Two peer processes A (sender) and B (receiver) use stop-and-wait ARQ to send packets over a single link with capacity C. All packets have the same length of 100 bits. The round-trip time (which is the time until A receives an acknowledgment for a sent packet) is equal to 2 seconds. Assume that no packets or ACK's are dropped and that all packets and ACK's arrive error-free. Furthermore, assume that the capacity C is equal to 100,000 bits per second.

(a) Find the average (transmission) rate (in bits per seconds) with which process A sends data to process B?



The transmission delay t_{tr} for sending the packet is equal to

$$t_{tr} = \frac{L}{C} = \frac{1}{1000} \text{ seconds.}$$

Let t_{ack} be the time until A receives an acknowledgment for a sent packet. In the time interval $T = t_{tr} + t_{ack}$ we have that A sends a single packet of length L. The rate x at which A sends packets to B is then given by

$$x = \frac{L}{T} = \frac{100}{\frac{1}{1000} + 2} \text{ bps} = \frac{100,000}{2001} \text{ bps} = 49.97 \text{ bps}.$$

(b) What is the link utilization?

The link utilization (the portion of the time that the link is used for transmitting a packet) is given by

$$\frac{t_{tr}}{T} = \frac{\frac{1}{1000}}{\frac{1}{1000} + 2} = \frac{1}{2001} = 0.0005.$$

(c) Assume that A and B do not implement stop-and-wait ARQ, but that A have up to n unacknowledged packets until it has to stop and wait for a ACK (this is called go-back n ARQ). For this case, express the average (transmission) rate (in bits per seconds) with which process A sends data to process B as a function of n.

Let

$$N_0 = \frac{t_{ack}}{t_{tr}} = 2000$$

be the maximum number of packets A can send while waiting for an ACK. The rate $x(n)$ at which A sends packets to B as a function of n is given by

$$x(n) = \begin{cases} \frac{nL}{T} = \frac{100n}{\frac{1}{1000}+2} \text{ bps}, & n \leq N_0 + 1 \\ C = 100,000 \text{ bps}, & n > N_0 + 1 \end{cases}$$

(d) Find the link utilization as a function n? For $n \leq N_0 + 1$, the link utilization is equal

to

$$\frac{nt_{tr}}{T},$$

and for $n \geq N_0 + 1$ the link utilization is equal to 1.

(e) Assume that we 200 processes (each generating packets of length 100 bits) share the single link. How should we choose n to avoid link congestion? (Note: this question shows how we can use Go-Back-n ARQ to implement a congestion avoidance algorithm. However, one additional difficulties we have to deal with is that in practice, we do not know the number of sessions sharing a link and we have to design an (adaptive) algorithm to tune n. TCP uses Go-Back-n to implement congestion control, as we will discuss in more details later in the course).

We need that

$$200x(n) \leq C,$$

or

$$200\frac{100n}{\frac{1}{1000} + 2} \leq 100,000.$$

This means that

$$n \leq 10.05$$

and we have to use $n \leq 10$ to avoid link congestion.

Question 2: A variant of the stop-and-wait protocol
Consider a channel that can lose packets but has a maximum delay that is known. Design a stop-and-wait protocol that can communicate reliably over this channel.

We distinguish three different cases based on different assumption regarding packet loss.
No Loss, but possibly errors in Data Packets or ACK's: When there is no loss (that means neither data packets, or ACK's, are lost), then we have the following result. Because the channel has a known maximum delay for delivering a packet, there is a maximum delay for receiving an ACK when neither the packet nor the ACK is lost. Let the maximum delay for receiving an ACK be t_{out}. In this case, we don't need to number the packets (SN) or ACK's (RN); but the sender simply resends a packet when it has not received an ACK within t_{out}. The receiver just sends immediately (!) an ACK for every error-free packet and ignores packets with an error.
ACK's are never lost: When data packets can get lost, but ACK's are never lost, then the same protocol as above works. That is, we don't need to number the packets (SN) or ACK's (RN); but the sender simply resends a packet when it has not received an ACK within t_{out}. The receiver just sends immediately (!) an ACK for every error-free packet and ignores packets with an error.
Data Packets and ACK's can get lost: When both data packets and ACK's can get lost, then the above protocol will not work properly. To see this, note that when a ACK for a given data packet (that has been received without an error) gets lost, then the sender will time-out and resend the same packet. However, in this case, the receiver won't be able to distinguish whether this is a new packet, or a retransmission of the earlier packet. To avoid this ambiguity, the sender has to use a sequence number (SN). However, because there is a maximum delay for receiving an ACK, we the receiver does not have to number the ACK's (no RN is required).