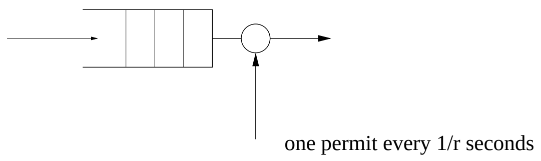
Faculty of Arts and Science
University of Toronto
CSC 358 - Introduction to Computer Networks

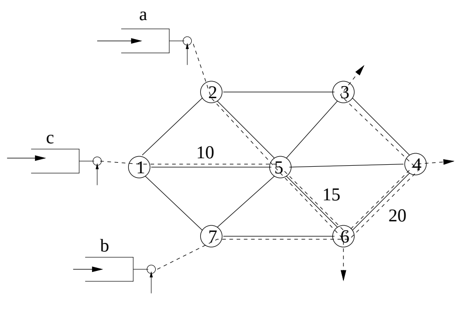
Solutions for Tutorial 8

Question 1

In class, we introduced the leaky bucket as a way to police the transmission rate of a session. One can do very clever things using the leaky bucket - in particular, one can do call admission control (resource allocation) to provide a good quality-of-service (to all sessions). In this problem, we get a glimpse of how this could be done.
First, consider the simple leaky bucket that we introduced in class, where a session receives permits (tokens) at the rate r (the rate allocated to the session). The session needs a permit (token) in order to transmit a packet, and unused permits are lost. We assume that all packets have the same length of $L = 10$ bits.



(a) Consider the network below. There are already 2 sessions, a (with the route 2-5-6) and b (with the route 7-6-4-3), using the network. The rates (of the leaky bucket) allocated to the two sessions are $r_a = 0.3$ packets per second, and $r_b = 1.7$ packets per second, respectively. Assume that session c wants to use the route 1-5-6-4; where the capacity of the link between router 1 and 5 is $C_{1,5} = 10$ bits per second, the capacity of the link between router 5 and 6 is $C_{5,6} = 15$ bits per second, and the capacity of the link between router 6 and 4 is $C_{6,4} = 20$ bits per second. What is the maximal rate r_c of the leaky bucket that we can allocate to session c without causing congestion in the network?

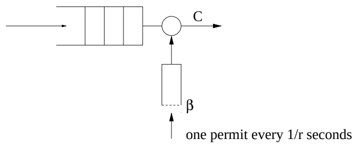


The solutions is

$$r_c = 0.3 \text{ packets per second}$$

The leacky bucket controls the average transmission rate of a sessions. In the network above, the bottleneck for session c is then the link 6-4, where sessions b has a tranmission rate of 17 bits per second. This leaves a tranmission rate of 3 bits per second (or 0.3 packets per second) for session c.

(b) Next consider the leaky bucket where (up to β) unused permits can be stored and used later by the session. Again we assume that all packets have the same length $L = 10$ bits.



To model this case, we use a fluid-flow model (kitchen-sink model) and assume that the transmission capacity C of the link between the leaky bucket and the network is equal to ∞ (very fast). We then say that the total traffic (fluid) $A(t,\tau)$ that a session with the above leaky bucket can submit in the interval $[t,\tau]$ is is upper-bounded by

$$A(t,\tau) \leq Lr(t-\tau) + L\beta.$$

Note that the term Lr captures the average transmission rate and $L\beta$ captures the maximal burst size (where β is the maximal number of saved permits).

Consider the situation below, where three sessions (a, b, and c) access a single link with transmission capacity $C = 10$ bits per second and space (in buffer or in service) for $B = 100$ bits. The amount of traffic (fluid) $B(t,\tau)$ that the link can transmit in the interval $[t,\tau]$ is then given by,

$$B(t,\tau) = C(t-\tau).$$

The parameters for the first two sessions are as follows,

$$r_a = 0.2 \text{ packets per second}, \qquad \beta_a = 5,$$

$$r_b = 0.5 \text{ packets per second}, \qquad \beta_b = 2,$$

What is the maximal rate r_c and the maximal number of saved permits β_c that we can allocate to session c so that there will never be a buffer overflow (packet loss)?

We have the following condition for the average tranmission rate,

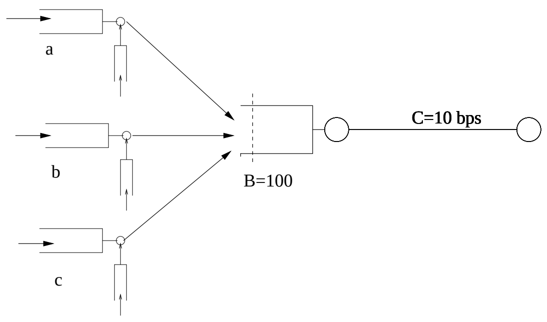
$$L(r_a + r_b + r_c) \leq C = 10.$$

The parameter β_c controls the burst size of a sessions, which has to be absorbed by the buffer for the link to avoid packet loss, and we obtain the followig condition

$$L(\beta_a + \beta_b + \beta_c) \leq B = 100.$$

The solutions then is

$$r_c = 0.3 \text{ packets per second}, \qquad \beta_c = 3.$$



The analysis that we did here is very simple, but it should give you a flavor of how the leaky bucket can be used to prevent congestion and provide a good quality-of-service. Note however that this comes at a price: for each new session we have to decided on the rate and maximal number of saved permits that we allocate to the sessions. Now think about the cost of doing this (for each session) in the global Internet! As we will see, the TCP congestion control mechanism is less expensive to implement (and therefore can be done in a global network), but it does not provide any quality-of-service guarantees.