Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2018 Prof. R. Wattenhofer

Computer Engineering II

Solution to Exercise Sheet Chapter 3

1 Quiz Questions

a) Yes, it can if F uses the edge e twice because of a cycle in the flow. But if we require F to be cycle-free, then we have $F(e) \leq F$ for all edges e.

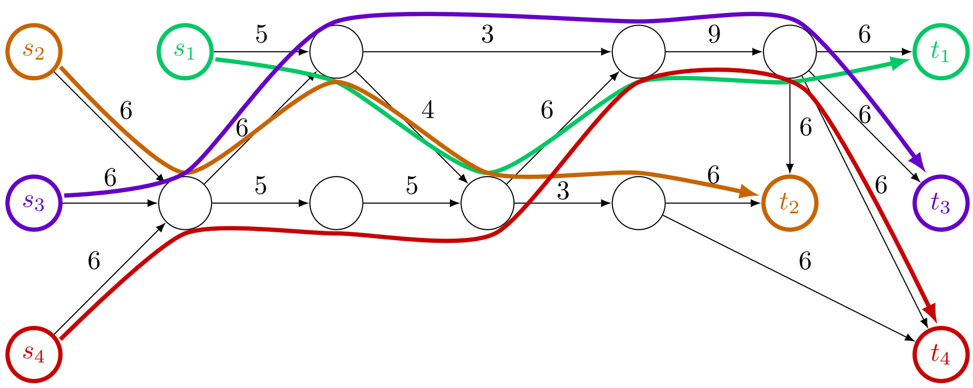
b) No, e.g., you could have two flows on an edge e where one flow has a small rate because it has to get through a low-capacity edge somewhere else and the other one can get all the remaining bandwidth of e. So you can have an arbitrary factor between the allocated bandwidths on an edge (if you design the network appropriately).

c) There are arguments you could make for TCP, e.g., the correct ordering of the packets, but for realtime applications, the latency is more important in most cases. Thus, you rather should choose UDP.

d) When the ACK for a packet arrives, the increase of the congestion window size by one packet (according to slow-start) can be seen as doubling the "packet slot" (of the acknowledged packet) in the congestion window. Since this is the case for any sent packet and since the ACK arrives roughly one RTT after sending a packet, doubling the size of the congestion window roughly takes one RTT.

2 Flows and Allocations

a) Yes.



b) $F_1 = 2$, $F_2 = 2$, $F_3 = 3$, $F_4 = 4$. The throughput is 11.

c) $F_1 = 1$, $F_2 = 3$, $F_3 = 3$, $F_4 = 5$. The throughput is 12.

3 UDP and TCP

a) On the one hand, the UDP application might not adjust its rate, meaning that the congestion will persist if TCP is not dropped. On the other hand, dropping TCP will incur a hefty multiplicative decrease, while the UDP application might be able to handle some lost packets (e.g., video streaming). There are more factors to this, but the real life answer is: It depends (and also: it's complicated).

b) A sender finds out if a router dropped one of its packets due to the missing ACK from the receiver after a timeout. If a router informs the sender directly when it drops its packet then the sender would not have to wait for the timeout. This is not done in practice because routers need to be fast, so we try to keep them simple. Also there might be a problem if the direct message of the router got lost, so we need a timeout no matter what!

c) The first thing to notice is that every 12 ms (take, for instance, the time from $t = 0$ to $t = 11$) 13 packets arrive at the router while only 12 are forwarded. After 12 ms the whole process repeats, but now the buffers are filled with one packet. Thus, the buffer utilization grows by one packet every 12 ms, resulting in a buffer utilization of x at time $t = 12x - 1$.

Now, let's have a look at the change in the buffer utilization during a time period of 12 ms, let's say from $t = 0$ to $t = 11$. At $t = 0$ the utilization increases by 2 packets, at $t = 1$ it decreases by 1, and so on. The buffer utilization is never larger than 2 packets until and including $t = 11$, and similarly, the buffer utilization is never larger than $x + 2$ during the time period from $t = 12x$ to $t = 12x + 11$ (recall the buffer utilization at $t = 12x - 1$). Moreover, the buffer utilization is $x + 2$ for the first time at $t = 12x$.

The router will drop the first packet when it reaches a buffer utilization of 11 packets, exceeding its capacity of 10 packets. Following our considerations above, this is the case for the first time at $t = 12 \cdot 9 = 108$. Thus, the first packet is dropped after 108 ms.

At $t = 108$, packets from all three clients arrive at the router. According to its dropping preferences, it will drop the packet from client C_1.

d) As the amount of packets arriving at R per time unit is only slightly larger than the amount of packets leaving R, the router only has to drop a packet every 12 ms. Since the congestion occurs always at a time which is a multiple of 12 ms, the router will always drop a packet from C_1, due to his dropping preferences. Thus, even if C_1 never notices that he loses packets, C_2 anad C_3 will never lose any packets.

e) Again, depending on the precise scenario, you could perhaps find arguments for both sides. We go with the following: Dropping packets from a close-by client is preferable since such a client will realize faster that it lost packets and therefore the congestion will be remedied faster. Dropping packets from a client sending with a large rate has the advantage that the probability that the congestion is actually removed is larger since halving the rate of a larger flow "removes more congestion". Also, you do not want to utilize routers at their buffers' limit since that would induce undesired latencies.

4 LPs

a) The vertices are $(0, 0)$, $(0, 2)$, $(1, 2)$, $(2, 0)$ and $(2, 1)$. The vertex with the largest objective function value and thus the solution to the LP is $(1, 2)$. Depending on if you go clockwise or counterclockwise, the simplex algorithm takes 2 or 3 steps, respectively. Figure 1 illustrates how the algorithm can procede.

b) Let the multi-commodity flow be denoted by $\mathcal{F} = (F_1, \ldots, F_k)$ where F_i has source s_i and destination t_i.

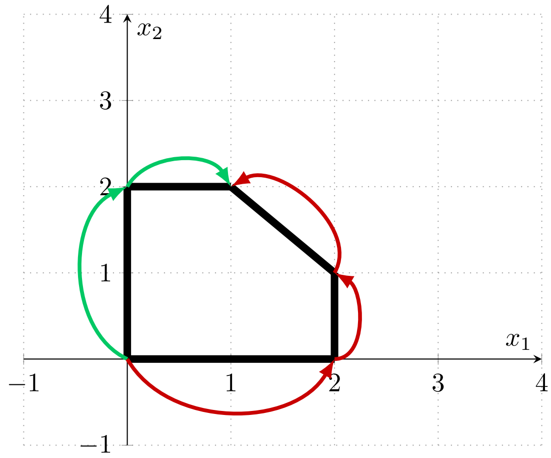


Figure 1: The five-sided polygon (and its interior) is the feasible region, i.e. set of points that satisfy the constraints of the LP. The objective function has a higher value at both $(0, 2)$ and $(2, 0)$ than at $(0, 0)$, so we can choose either one as the first step to take. Depending on that, it takes us two or three steps to find the optimum at $(1, 2)$.

Maximize $f(\mathbf{x}) = \sum_{i=1}^{k} \sum_{e \in \texttt{out}(s_i)} x_{ei}$
subject to

(a) $x_{ei} \geq 0$ for all $e \in E$ and all $1 \leq i \leq k$

(b) $\sum_{i=1}^{k} x_{ei} \leq c(e)$ for all $e \in E$

(c) $(\sum_{e \in \texttt{in}(v)} x_{ei} = \sum_{e \in \texttt{out}(v)} x_{ei}$ for all $v \in V \setminus \{s_i, t_i\})$ for all $1 \leq i \leq k$

(d) $\sum_{e \in \texttt{in}(s_i)} x_{ei} = 0$ for all $1 \leq i \leq k$

---

c) Take the solution for a) and add constraints that ensure that the flow rates are bounded by the d_i:

Maximize $f(\mathbf{x}) = \sum_{i=1}^{k} \sum_{e \in \texttt{out}(s_i)} x_{ei}$
subject to

(a) $x_{ei} \geq 0$ for all $e \in E$ and all $1 \leq i \leq k$

(b) $\sum_{i=1}^{k} x_{ei} \leq c(e)$ for all $e \in E$

(c) $(\sum_{e \in \texttt{in}(v)} x_{ei} = \sum_{e \in \texttt{out}(v)} x_{ei}$ for all $v \in V \setminus \{s_i, t_i\})$ for all $1 \leq i \leq k$

(d) $\sum_{e \in \texttt{in}(s_i)} x_{ei} = 0$ for all $1 \leq i \leq k$

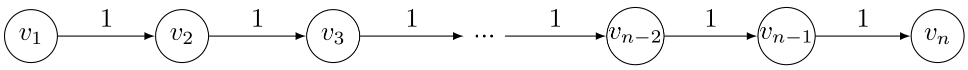
(e) $\sum_{e \in \texttt{out}(s_i)} x_{ei} \geq d_i$ for all $1 \leq i \leq k$

A multi-commodity flow $\mathcal{F}$ as desired exists if and only if there is a solution to the LP. Thus, you infer a YES answer from the existence of a solution and a NO answer from the non-existence, i.e., if there exists no solution to the LP. Note that it is irrelevant which linear function you choose to be maximized in the above LP as the existence of a solution to the LP only depends on if there is a point in the vector space that satisfies all of the inequalities. In other words, if the polytope is non-empty, then there exists a solution for any linear objective function and the answer is YES.

Mastery

5 Fairness vs. Efficiency

Consider the following graph:



Now, let there be $n-1$ flows $F_1, \ldots, F_{n-1}$, with source v_1 and destination v_n each. Additionally, let there be $n-1$ flows $F'_1, \ldots, F'_{n-1}$ where each flow F'_i has source v_i and destination v_{i+1}.

In the max-min-fair allocation, each flow has a rate of $1/n$ since each edge divides its bandwidth equally between the n flows using the edge. Thereby, a throughput of $(2n-2)/n$ is achieved. However, the maximal throughput is achieved by allocating a bandwidth of 1 to each F'_i and a bandwidth of 0 to each F_i. The resulting throughput is $n-1$. We obtain an efficiency of $2/n$ for the max-min-fair allocation, which tends to 0 for $n \to \infty$. Thus, we showed that the efficiency of a max-min-fair allocation can be arbitrarily small.