ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Distributed Computing

HS 2015 Prof. R. Wattenhofer

Distributed Systems Part II

Solution to Exercise Sheet 10

1 Pop Quiz

a) If ALL of the edges would have slack, then it would work - else it is quite easy to construct a counter-example: E.g., consider that a small set of flows can only be routed inside of a small part of the network, and they currently use all of these edges' capacity.

b) First, generate the graph composed of all old and new rules. Then, check if this graph has a cycle.

c) First, split up all rules into single-destination rules. These can always be updated. Once all single-destination rules are updates, merge them back together to the prefix rules.

2 Network Updates

a) v_3 can not change before v_2, but v_2 needs to wait for v_1, requiring three steps in total.

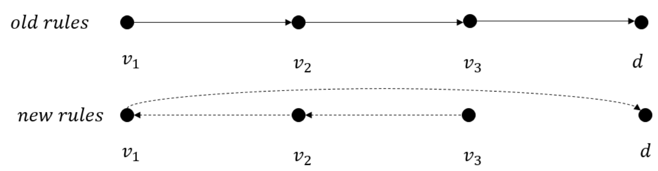

Figure 1: Graph with three rules.

b) Let E' be the set of rules that no longer need to be updated and V' the set of nodes with the property that there exists a path to the destination using only rules from E', with $d \in V'$. Any new rule with the property that it points to a node from V' can not induce a cycle, since all paths from all nodes from V' end at d and d has no outgoing edge. If there are still rules to be updated, then such a rule will always exist, since the set of new rules induces a directed tree with d as its root and all edges in this tree are oriented towards d, meaning at least one new rule will point to a node from V'.

Note: As seen in **c)**, all rules that can be updated might have this property.

c) For a graph with n nodes we use the same concept as in the first item, but with n instead of three vertices v_i. Again, v_i can not change before v_{i-1} for $2 \leq i \leq n$, requiring n steps in total.

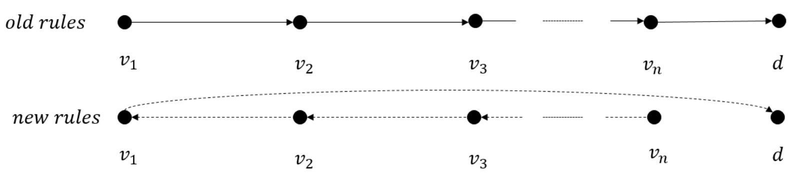

Figure 2: Graph with n rules.

d) Observe that the only cycle that can appear is v_1, v_2, v_3, v_1, which can be seen by overlaying both graphs. As the only part of the cycle contained in the old rules is the outgoing edge from v_3, we need to make sure that at least one of v_1 and v_2 performs its update strictly later than v_3. The possible combinations are

- v_3; v_2; v_1
- v_3; v_1; v_2
- v_3; v_2, v_1
- v_3, v_2; v_1
- v_3, v_1; v_2
- v_1; v_3; v_2
- v_2; v_3; v_1

3 Capacity-Consistent Updates

a) Consider the solid edge in the top middle: If f_p moves before f_b does, then these flows have a combined size of 4, but the capacity of the edge is only 3.

b) f_p: No. f_g: Yes. f_b: Yes. f_r: No.

c) We saw above that we cannot do it in one update. You can do it in 2: First, move f_b and f_g. Then, move f_r and f_p.