Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Networked Systems
ETH Zürich — seit 2015

Communication Networks

Prof. Laurent Vanbever

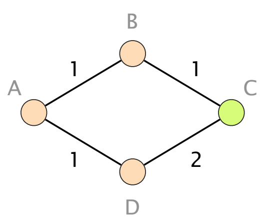
Solution: Exercises week 7 – Internet Routing

Convergence (Exam Style Question)

Consider this simple network running OSPF as link-state routing protocol. Each link is associated with a weight that represents the cost of using it to forward packets. Link weights are bi-directional.

Assume that routers A, B and D transit traffic for an IP destination connected to C and that link (B, C) fails. Which nodes among A, B and D could potentially see their packets being stuck in a transient forwarding loop? Which ones would not?

Solution: Nodes A and B could see their packets stuck in a forwarding loop if B updates its forwarding table before A, which is likely to happen as B would be the first to learn about an adjacent link failure. On the other hand, D would not see any loop as it uses its direct link with C to reach any destination connected beyond it.



Loopy or not?

Assume now that the network administrator wants to take down the link (B, C), *on purpose*, for maintenance reasons. To avoid transient issues, the administrator would like to move away all traffic from the link *before* taking it down and this, without creating any transient loop (if possible). What is the minimum sequence of increased weights setting on link (B, C) that would ensure that *no packet* destined to C is dropped?

Solution: One example of a minimum sequence of weight settings is [1, 3, 5].

Note: The problem highlighted above happens because B shifts traffic to A before A shifts traffic to D, hence creating a forwarding loop. By setting the (B, C) link weight to 3, (only) A shifts from using (A, B, C) to using (A, D, C). Once A has shifted, it is safe to shift B by setting the link weight to 5 (or higher). Once B has shifted has well, the link can be safely torn down.

Exam Question

For the following statements, decide if they are *true* or *false*. Motivate your decision. These questions are directly taken from the Communication Networks final exam of 2016.

a) Consider a positively weighted graph G. Applying the Bellman-Ford (used by distance-vector protocols) or Dijkstra (used by link-state protocols) algorithm on G would lead to the same forwarding state.

Solution: True. Both solve the shortest-path problem.

b) Link-state protocols (such as OSPF) are guaranteed to compute loop-free forwarding state as long as the link-state databases are consistent on all routers.

Solution: True. However, they can experience transient loops while it isn't the case.

c) Link-state protocols (such as OSPF) require routers to maintain less state than distance-vector protocols (such as RIP).

Solution: False. Link-state protocols require routers to maintain the entire topology in memory (Link-State database). Distance-vector protocols only need to maintain the costs to reach each prefix.

d) Poisoned reverse solves the problem of count-to-infinity.

Solution: False. The problem is still there it is mitigated by having a small infinity value.

e) Consider a positively weighted graph G. Multiplying all link weights by 2 would change the all-pairs shortest paths computed by the Dijkstra algorithm on G.

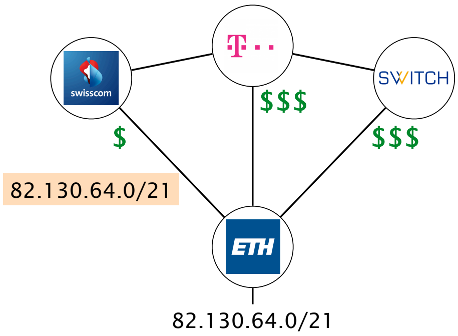
Solution: False. Multiplying by a constant factor keeps the ranking between the paths constant.

f) Consider a positively weighted graph G. Adding 1 to all link weights would change the all-pairs shortest paths computed by the Dijkstra algorithm on G.

Solution: True. Longer paths will see a bigger increase than shorter ones.

Traffic Engineering

Assume that ETH has only one prefix: 82.130.64.0/21. As depicted on the left, the ETH network is connected to three providers (Swisscom, Deutsche Telekom and Switch) and the providers are interconnected with each other. The contract with Swisscom is the cheapest one (indicated by the dollar symbols). For this reason, ETH wants to receive all the incoming traffic over the Swisscom link and therefore announces its prefix only to Swisscom.



ETH is connected to three providers with different costs.

a) Do you think that is a good configuration? What happens if the link between ETH and Swisscom fails?

Solution: Not a good solution. If the link fails, ETH will no longer receive any traffic. ETH is no longer reachable from other networks.

b) To improve the connectivity in case of a link failure between ETH and Swisscom, ETH wants to optimize its announcements. Write down the prefixes which ETH announces to Swisscom, Deutsche Telekom and Switch. During normal operation (no link failure) ETH should still receive all incoming traffic over the Swisscom link.

Solution:

To Swisscom: 82.130.64.0/22 and 82.130.68.0/22 (other splits are also possible)

To Deutsche Telekom: 82.130.64.0/21

To Switch: 82.130.64.0/21

c) After further investigations, ETH decides that only traffic towards 82.130.68.0/23 has to be received over the Swisscom link. All the other traffic can enter over any of the providers. Which prefixes do you have to announce to achieve this traffic distribution?

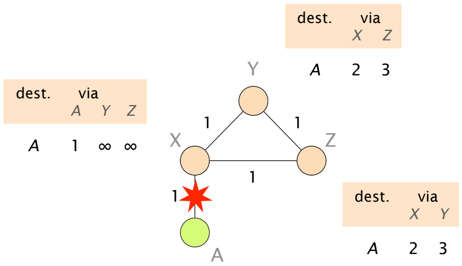
Solution:

To Swisscom: 82.130.68.0/23 and 82.130.64.0/21

To Deutsche Telekom: 82.130.64.0/21

To Switch: 82.130.64.0/21

Convergence with Poisoned Reverse



Consider the network on the left which uses distance vector routing with poisoned reverse. Each link is associated with a weight that represents the cost of using it to forward packets. Link weights are bi-directional.

Assume that the link between X and A fails (as shown in the figure) and use the table below to show the first 8 steps of the convergence process. How many steps does it take until the network has converged to a new forwarding state? Explain your observations.

Solution: The network does not converge as the maximum link weight is increased by one in each round ("count to infinity problem"). Poisoned reverse does not solve the problem of counting to infinity if three or more nodes are involved. One possible workaround is to define ∞ as a small value (e.g. ∞ := 16).

Solution:

		X			**Y**		**Z**	
	dst=A	via A	via Y	via Z	via X	via Z	via X	via Y
$t = 0$	before the failure	1	∞	∞	2	3	2	3
$t = 1$	after X sends its vector	★	∞	∞	∞	3	∞	3
$t = 2$	after Y sends its vector	★	4	∞	∞	3	∞	∞
$t = 3$	after Z sends its vector	★	4	∞	∞	∞	∞	∞
$t = 4$	after X sends its vector	★	4	∞	∞	∞	5	∞
$t = 5$	after Y sends its vector	★	∞	∞	∞	∞	5	∞
$t = 6$	after Z sends its vector	★	∞	∞	∞	6	5	∞
$t = 7$	after X sends its vector	★	∞	∞	∞	6	∞	∞
$t = 8$	after Y sends its vector	★	7	∞	∞	6	∞	∞

Add the distance vectors to this table