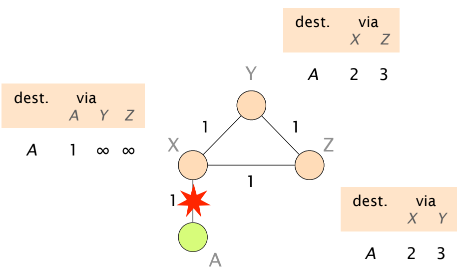
Communication Networks

Prof. Laurent Vanbever

Solution: Exercise 6 – Convergence Process in Link-State and Distance-Vector Routing

Convergence

6.1 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	∞	∞

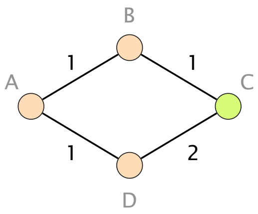
Add the distance vectors to this table

6.2 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.