Projet Réseaux

Simulation d'un protocole de routage à vecteur de distances

- Rapport, groupe 2 -

Abdelkrim ESSAFSYFY, matricule : 191856

Paul ONDAFE MATOCK, matricule : 202048

Année académique 2020-2021

Table des matières

1 Exercice 2.2

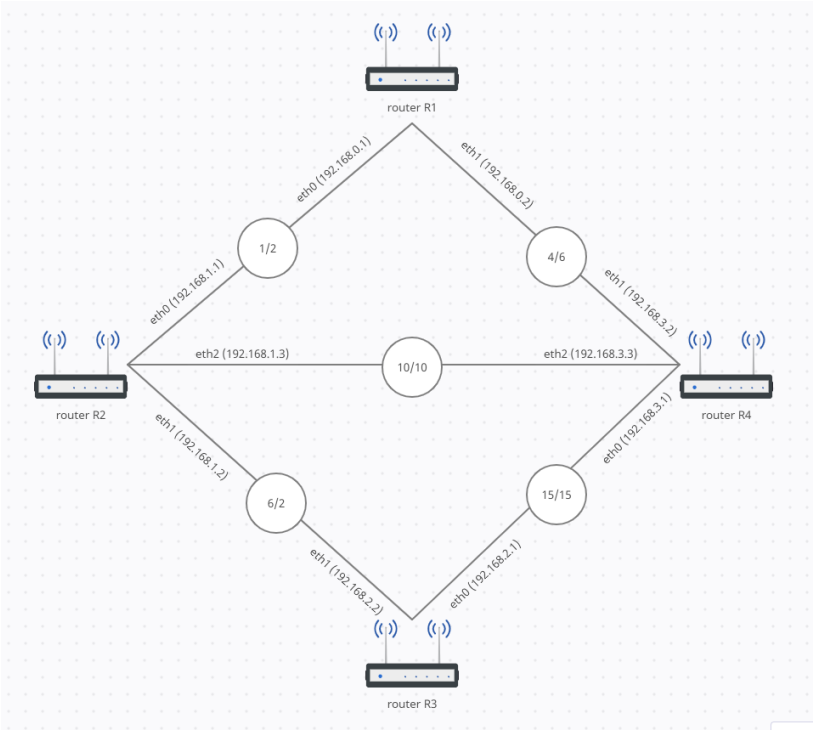

FIGURE 1 – Topologie de 4 nœuds et 5 liens

Step	D_{R1}(R2)	D_{R2}(R2)	D_{R3}(R2)	D_{R4}(R2)
init	∞	0	∞	∞
1	1 [R2]		2 [R2]	10 [R2]
2				7 [R1]
3	14 [R4]		25 [R4]	17 [R3]
4	11 [R4]		22 [R4]	

FIGURE 2 – Tableau des routes calculées par les routeurs vers R2

La Figure 1 représente une topologie contenant 4 nœuds et 5 liens ainsi que les coûts et les interfaces par lesquelles ils sont connectés. La figure 2 quant à elle, représente les routes calculées à partir de chacun des routeurs vers une unique destination *(R2)*.

2 Exercice 2.3

La Figure 3 représente la topologie résultant en un comptage à l'infini, le nombre 60 en rouge indique le changement nécessaire pour provoquer le comptage à l'infini.

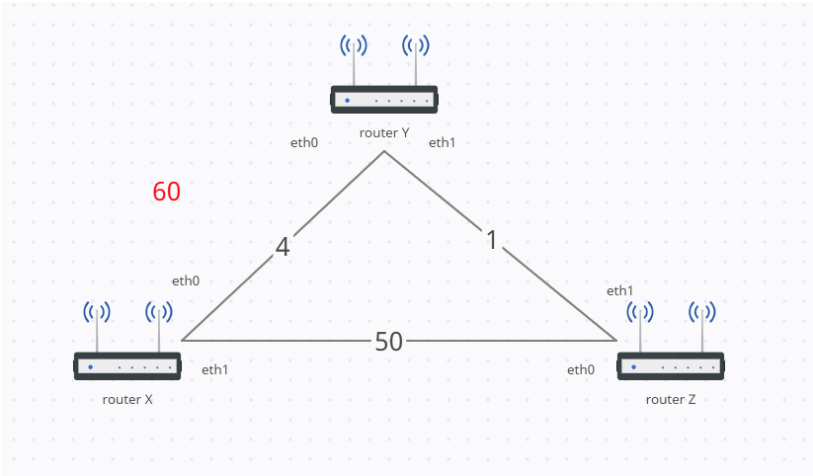

FIGURE 3 – Topologie résultant en un comptage à l'infini

Simulation	Lien X-Y	Lien Y-Z	Lien X-Z	Nombre d'itérations (temps en ms)
#1	60	1	50	3350
#2	40	1	50	2602
#3	70	1	60	4022

FIGURE 4 – Tableau contenant 3 simulations différentes

Le nombre de messages échangés par les routeurs depuis le changement de métrique et jusqu'à la nouvelle convergence est de 24 messages.

Les 2 assignations de coûts de liens qui résultant en une convergence plus courte et une autre plus longue sont renseignées dans la Figure 4.

3 Exercice 2.4

La nom de la solution au comptage à l'infini vue dans le cours est *Poisoned reverse*. Elle consiste à ne pas indiquer le coût réel vers la destination aux voisins (en leur envoyant $+\infty$ comme coût) qui passent par ce même nœud afin d'éviter de progressivement incrémenter de 1 les coûts. Dans la topologie donnée dans l'exercice 2.3, Y doit renseigner $+\infty$ à Z — son voisin — puisque celui-ci passe pas Y afin d'atteindre X ; sa destination.

4 Exercice 2.5

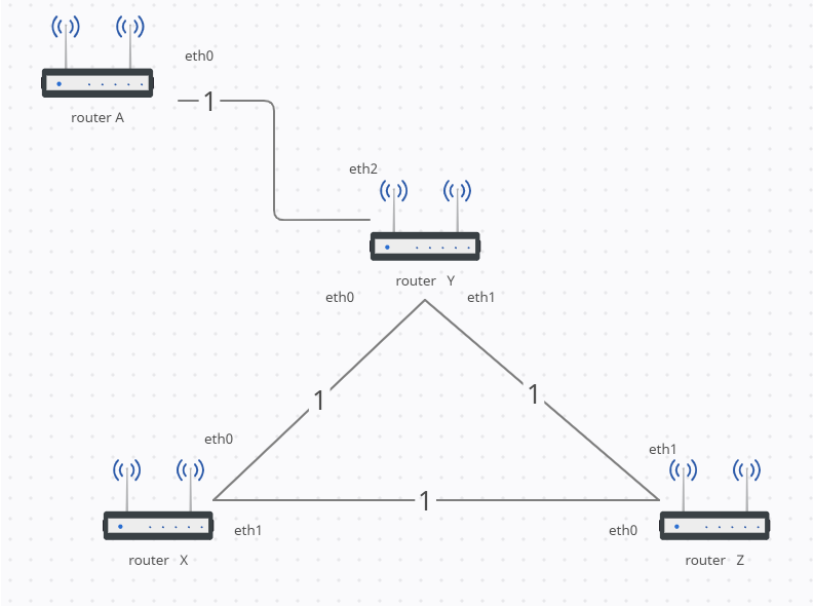

FIGURE 5 – Topologie où Poisoned reverse ne règle pas le problème de comptage à l'infini

La Figure 5 contient une topologie où, même avec application de la méthode Poison reverse, le problème de comptage à l'infini persiste. Dans cette topologie, le lien entre le routeur A et le routeur Y n'est plus fonctionnel. Il est donc impossible d'accéder au routeur A.

4.1 Persistance du problème de comptage à l'infini

Le problème de comptage à l'infini persiste car, malgré le fait que le routeur Y informe X et Z que le coût vers A est égal à l'infini, le premier routeur recevant cette information — supposons dans ce cas qu'il s'agit de Z — remarque qu'il est toujours possible d'accéder au routeur A via X. Ceci est dû au fait que X n'est pas au courant du changement des coûts des liens ($D_y(a) = +\infty$). Nous remarquerons donc un comptage à l'infini entre X et Z, qui continuent d'incrémenter les coûts.

4.2 Solution au comptage à l'infini

Une solution à ce problème est appelée *Split Horizon*. Elle consiste à interdire à un routeur de partager les coûts des liens dans la table de routage avec le routeur (interface) à partir duquel il

a appris ceux-ci. Concrètement, dans la Figure 5, quand le routeur Y informe Z — et plus tard X — que le routeur A n'est plus accessible, le routeur Z n'enverra pas à Y les coûts associés à ce routeur Y. De cette façon, Y conservera $D_y(a) = +\infty$ et le comptage à l'infini sera évité.

5 Exercice 2.6

Nombre de liens	Temps de convergence (en ms)
10	1008
20	1344
30	2016
50	2016
100	2016

FIGURE 6 – Tableau de convergence avec différents liens

La Figure 6 représente le temps convergence obtenus pour des topologies de 10, 20, 30, 50 et 100 liens.

5.1 Fonctionnement de l'algorithme

Le principe utilisé se décline de la manière suivante : création des différents nœuds qui seront utilisés dans le réseau. En suite, initialisation du réseau par deux nœuds avec un lien. Finalement, ajout d'un nœud dans le réseau existant en s'appuyant sur le model de Barabási, le choix de rattacher le nouveau nœud à un des nœuds du réseau selon le degré de ce dernier. On réalise une itération sur ce point tout vérifiant que la condition de la densité est toujours respectée.

5.2 Description du lancement du programme

L'algorithme a été conçu en Java. Le package contient quatre classes. La classe Main.java, qui permet de lancer la simulation, une classe Node.java qui simule un noeud (notamment un routeur), une classe Link.java qui elle simule un lien entre deux noeuds (ou, encore une fois, deux routeurs) et finalement une classe Generator.java qui crée la topographie et la stocke dans un fichier texte nommé 'graph.txt'.