Téléinformatique

IFT 3325

Devoir n°3

17 Décembre 2023

Auteurs :

- Léo Jetzer (20070432)
- Luchino Allix-Lastrego (20222844)

Université de Montréal
Département d'informatique et de recherche opérationnelle

Exercice 1 *(10 points)*

a. *(6 points)*

Pour réduire la charge d'un serveur il faut favoriser l'utilisation de Go-Back-N car les acknowledge cumulatifs permettent de diminuer le nombre de réponses par rapport au grand nombre de petits messages.

b. *(4 points)*

S'il y a une erreur lors de la lecture d'une trame et qu'elle est rejetée, il est posssible que le fanion qui était supposé marquer la fin soit interprété comme le début d'une nouvelle trame. On essai alors de lire des trames qui n'existe pas et/ou d'ignorer les vrais trames.

Exercice 2 *(16 points)*

Dijkstra

Dans le tableau ci-dessous, les colonnes indiquent les sommets et les lignes le sommet où l'on est actuellement. Par exemple *E (5)* signifie qu'on se trouve sur le sommet E et que le poid associé pour arriver à ce sommet est de 5. Le croisement entre une ligne et une colonne indique comment faire pour arriver à ce sommet. Par exemple, *5 B* à la ligne *H (4)* et à la colonne *C* indique que pour se rendre en *C* le plus court chemin vaut 5 et passe par *B*. Le symbole ∞ indique que le sommet n'a pas encore pu être atteint et '-' indique que le chemin à déjà été visité.

	A	B	C	D	E	F	G	H	I	J	Z
Départ	0 A	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
A (0)	-	3 A	∞	∞	5 A	∞	∞	4 A	∞	∞	∞
B (3)	-	-	5 B	∞	5 A	10 B	∞	4 A	∞	∞	∞
H (4)	-	-	5 B	∞	5 A	9 H	∞	-	6 H	∞	∞
E (5)	-	-	5 B	∞	-	9 H	∞	-	6 H	∞	∞
C (5)	-	-	-	8 C	-	7 C	11 C	-	6 H	∞	∞
I (6)	-	-	-	8 C	-	7 C	11 C	-	-	12 I	∞
F (7)	-	-	-	8 C	-	-	11 C	-	-	10 F	∞
D (8)	-	-	-	-	-	-	11 C	-	-	10 F	10 D

À la dernière ligne du tableau, on voit que l'on peut arriver en *Z* en venant de D avec un chemin de poid 10. Ceci met fin à l'algorithme car les autres chemins qui n'arrivent pas encore à *Z* sont de poid supérieur ou égal à 10. Pour retrouver le chemin parcouru on remonte le tableau. On arrive en *Z* depuis *D*, on arrive en *D* depuis *C* et ainsi de suite pour obtenir le chemin de poid 10 : *ABCDZ*.

Bellman-Ford

Dans le tableau ci dessous, les colonnes indiquent le sommet et les lignes le nombre maximum de chemin que l'on peut prendre pour arriver au sommet. La logique reste la même concerant les cases. Par exemple *5 B* à la ligne *4* et à la colonne *C* indique que pour se rendre en *C* avec au plus 4 chemins emprunté, le plus court chemin vaut 5 et passe par *B*.Comme précedement ∞ indique que le sommet n'a pas encore pu être atteint.

Itération	A	B	C	D	E	F	G	H	I	J	Z
0	0 A	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
1	0 A	3 A	∞	∞	5 A	∞	∞	4 A	∞	∞	∞
2	0 A	3 A	5 B	∞	5 A	9 E	∞	4 A	6 H	∞	∞
3	0 A	3 A	5 B	8 C	5 A	7 C	11 C	4 A	6 H	12 I	∞
4	0 A	3 A	5 B	8 C	5 A	7 C	11 C	4 A	6 H	10 F	10 D
5	0 A	3 A	5 B	8 C	5 A	7 C	11 C	4 A	6 H	10 F	10 D
6	0 A	3 A	5 B	8 C	5 A	7 C	11 C	4 A	6 H	10 F	10 D
7	0 A	3 A	5 B	8 C	5 A	7 C	11 C	4 A	6 H	10 F	10 D
8	0 A	3 A	5 B	8 C	5 A	7 C	11 C	4 A	6 H	10 F	10 D
9	0 A	3 A	5 B	8 C	5 A	7 C	11 C	4 A	6 H	10 F	10 D
10	0 A	3 A	5 B	8 C	5 A	7 C	11 C	4 A	6 H	10 F	10 D

On remarque que à partir de la ligne 4, plus rien ne change, en effet tous les plus courts chemins depuis le sommet A vers les autres sommets empruntent au plus 4 arrêtes. Pour trouver le chemin le plus court de *A* à *Z* même logique que précedement, ce qui nous donne : *ABCDZ* avec un poid de 10.

Exercice 3 *(12 points)*

a. *(6 points)*

Les deux réseaux ne peuvent pas communiquer car ils ont le même network ID.

$$A = 01100101.01000000.00000000.01100110 \qquad B = 01100101.01000000.00101101.01100110$$
$$M = 11111111.11111111.00000000.00000000 \qquad M = 11111111.11111111.00000000.00000000$$
$$A \wedge M = 01100101.01000000.00000000.00000000 \qquad B \wedge M = 01100101.01000000.00000000.00000000$$

En effet, on remarque que $A \wedge M = B \wedge M$

b. *(6 points)*

Oui il faut modifier cette table de routage car pour se rendre en D, la table indique que la passerelle est 192.168.51.1 alors qu'elle devrait être 192.168.52.2.

Exercice 4 *(10 points)*

a. *(3 points)*

Lors de l'établissement d'une connexion au niveau de la couche transport, on peut négocier :

1. La taille maximale de la fenêtre. Pour améliorer l'efficacité.
2. La taille maximale de segment. Pour permettre de lire correctement les paquets.
3. L'utilisation du timestampes. Ceci permet de facilier l'ordre dans lequel les paquets ont été envoyé.

b. *(4 points)*

— **Length** : Change de la taille total du paquet à la taille du fragment
— **fragFlag/bit More** : Indique s'il s'agit du dernier fragment
— **Offset** : offset du premier bit pertinent du fragment relatif au début du paquet
Bien entendu, les donnés contenues changes aussi.

c. *(3 points)*

— Le routeur doit ré-assembler/désassembler chaque paquet qu'il reçoit et transmet, consommant beaucoup de ressources. Un paquet peut donc être ré-assembler/désassembler plusieurs fois lors de sa transmission. Par contre, la quantité de paquet circulant est optimisée
— Chaque paquet est ré-assembler/désassembler une seule fois, à la source et la destination. Il faut par contre connaître d'avance la plus petite taille de paquet autorisée sur le chemin.

Exercice 5 *(12 points)*

Exercice 6 *(12 points)*

J'ai de la difficulté à comprendre la question, alors voici des solutions selon différentes manière de l'interpréter

T : threshold de la taille de la fenêtre avant de ralentir l'agrandissement

n : nombre de segment à envoyer. $= \left\lceil \frac{M}{N} \right\rceil$

F : Taille maximale de la fenêtre

t : nb de RT nécessaire pour que la fenêtre atteigne la taille T. $= \lceil \lg(T+1) \rceil$

$$q(i) : \text{nb de segments envoyés au RT } i. = \begin{cases} 0 & \text{si } i \leq 0 \\ 2^{i-1} & \text{si } 0 < i \leq t \\ \min(F, 2^{t-1} + i - t) & \text{sinon} \end{cases}$$

$$Q(i) : \text{nb total de segments envoyés après un RTT } i. = \sum_{j=0}^{i} q(j)$$

Envoyer M octets en 1 RT

On veut simplement trouver un i tq. $q(i-1) < n \leq q(i)$

Si $n \leq T$ alors la quantité de RT nécessaire pour envoyer M octets en un RT est $\lceil \lg 2n \rceil$. Sinon, elle est de $\lceil \lg 2T \rceil + n - T$

Envoyer M octets en tout

Pour n segments, on veut donc trouver un i tq. $Q(i-1) < m \leq Q(i)$.

Si $n \leq Q(t)$ (ie. $i \leq t$), alors $Q(i) = 2^i - 1$ donc $i = \lceil \lg(n+1) \rceil$

Exercice 7 *(7 points)*

Non les réseaux à circuits virtuel n'ont pas besoin d'être capable de router des paquets isolé car ils suivent tous un chemin prédéfini. Si on parle d'un paquet qui possède sa destination (qui a été poussé dans le canal quand la connexion à été établie) et qu'il fait parti du circuit virtuel alors aucun routage n'est necessaire, il est dans le circuit et il n'a qu'à le suivre.

Exercice 8 *(10 points)*

La durée d'un RT est de $1 + 3 + 2 + 1 + 3 = 10$ unités. Si on assume qu'il n'y a aucune congestion et que la fenêtre de réception reste de la même taille constamment, il y a 7 segments d'informations transmis chaque 10 unités au maximum.

Exercice 9 *(6 points)*

Avec une connection de 100Mbps et un RTT de 1ms, jusqu'à 125 bytes sont transféré à chaque RT. Pour assurer un flux continue, il nous faut donc une fenêtre d'au moins 125 bytes (ça rentre bien dans un seul segment)

Exercice 10 *(5 points)*

Non les algorithmes de Dijkstra et Bellman-Frod ne produisent pas tout le temps les mêmes résultats car l'algorithme de Dijkstra ne permet pas des arrêtes avec des poids négatifs alors que celui de Bellman-Ford oui. Donc dans un graph avec au moins une arrête de poid négatif, les deux algorithmes ne produiront pas peut être pas le même résultat.