INF3610 –Systèmes embarqués

Automne 2021

TP No. 1

Groupe 2

2020847 – Guillaume Rochon-Vear

2028284 – Florence Cloutier

Soumis à : M. Guy Bois

22 septembre 2021

Question 1

Expliquez la structure de la fonction TaskForwarding. Pourquoi un if then else et pas un switch ?

On veut aller chercher les paquets dans l'ordre High – Medium – Low priorités des files. Le premier if est fait sur le message d'erreur retourné en essayant d'aller chercher dans la file de haute priorité. S'il y a une erreur, on va chercher dans la file de priorité moyenne. Finalement, s'il y a encore une erreur, on va chercher dans la file de priorité basse. Un switch est utilisé quand une variable prend plusieurs valeurs différentes. Dans ce cas-ci, la condition des if reste toujours pareille, car on vérifie seulement s'il y a une erreur ou non.

Expliquez. Également, pourquoi l'option OS_OPT_PEND_NON_BLOCKING dans OSQPend ?

L'option OS_OPT_PEND_NON_BLOCKING est utilisée dans OSQPend pour éviter d'attendre à l'infini dans la file de haute priorité pour un paquet, tandis qu'il y a peut-être des paquets dans les files moins prioritaires. Si cette option n'était pas utilisée, on n'attendrait que des paquets dans la file de priorité élevée.

Question 2

Justifiez le choix des priorités de chaque tâche (lignes 24 à 28 de routeur.h). Auriez-vous fait différemment ? Au besoin, proposez des alternatives (que vous pouvez rapidement valider).

Justification choix priorités TP :

TaskStats a la plus haute priorité parce qu'on veut qu'il affiche les statistiques à toutes les 30 secondes, sans avoir à attendre après une autre tâche.

TaskGenerate est le deuxième plus prioritaire, car c'est la tâche d'entrée. TaskGenerate simule le routeur qui reçoit les paquets. On ne veut pas qu'il perde de paquets, donc, pour éviter le sous-échantillonage, on lui donne une priorité élevée. De plus, dans TaskGenerate, il y a un boolean isGenPhase qui permet de sortir de cette tâche quand il atteint un maximum de 255 paquets. Donc, même si cette tâche est prioritaire aux subséquentes, elle laissera éventuellement sa place aux tâches moins prioritaires, si elle atteint 255 paquets générés. Ce qui évite la famine dans les tâches moins prioritaires.

TaskOutputPort est le 3^{e} plus prioritaire, car il n'a pas de FIFO. Pour éviter qu'il perde des paquets quand il les reçoit, on lui donne une priorité élevée.

Finalement, TaskForwarding est moins prioritaire que TaskComputing, car TaskForwarding n'est pas bloquant, donc il roulerait tout le temps si sa priorité était plus élevée que les autres tâches. TaskComputing est donc l'avant dernier en termes de priorités et TaskForwarding est le dernier plus prioritaire.

Alternative proposée:

Nous aurions mis en ordre du plus prioritaire au moins prioritaire :

TaskStats > TaskOutputPort > TaskForwarding > TaskComputing > TaskGenerate

Avec cet ordre de priorité, on évite la famine, sans avoir à utiliser de booléen isGenPhase dans TaskGenerate. De plus, dans le cadre du TP, cette solution fonctionne, car TaskGenerate génère les paquets et ne les reçoit pas. Donc il ne va jamais en perdre, contrairement à si c'était réellement l'entrée d'un routeur qui recevait les paquets.

TaskStats reste le plus prioritaire, car on souhaite qu'il s'exécute tous les 30 secondes sans avoir à attendre après une autre tâche. Ensuite, si TaskOutputPort, la tâche la plus prioritaire après TaskStats, n'a rien à output, la prochaine tâche la plus prioritaire est TaskForwarding. Si TaskForwarding n'a rien à forward, la prochaine tâche la plus prioritaire est TaskComputing. Si TaskComputing n'a aucun paquet à "compute", alors TaskGenerate pourra s'exécuter. Cette ordre de priorité permet de print les statistiques tous les 30 secondes et évite aussi la famine de certaines tâches.

Question 3

Présentez vos résultats (4 captures d'écran) d'analyse de performance réalisés plus haut et comparez les 2 résultats de recherche binaire (333Hz et 1000Hz) en tentant de voir ce qui peut expliquer les similitudes et différences.

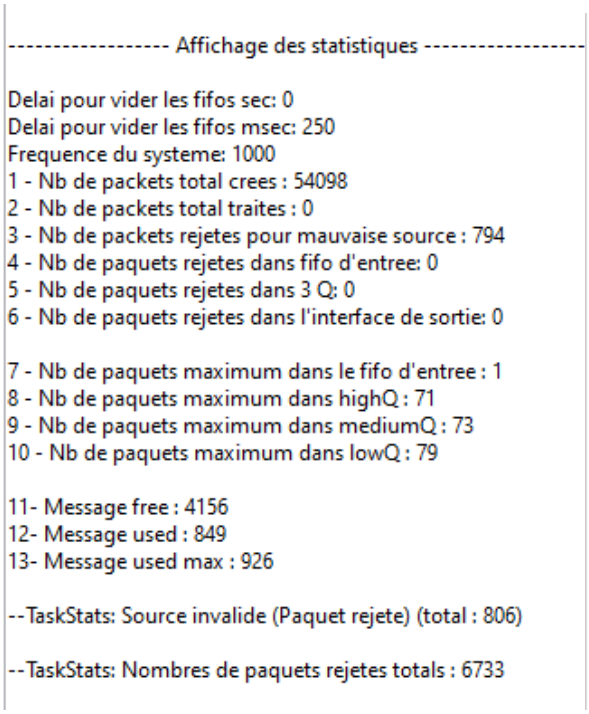

Figure 1 : Fréquence 1000Hz juste avant que le fifo déborde

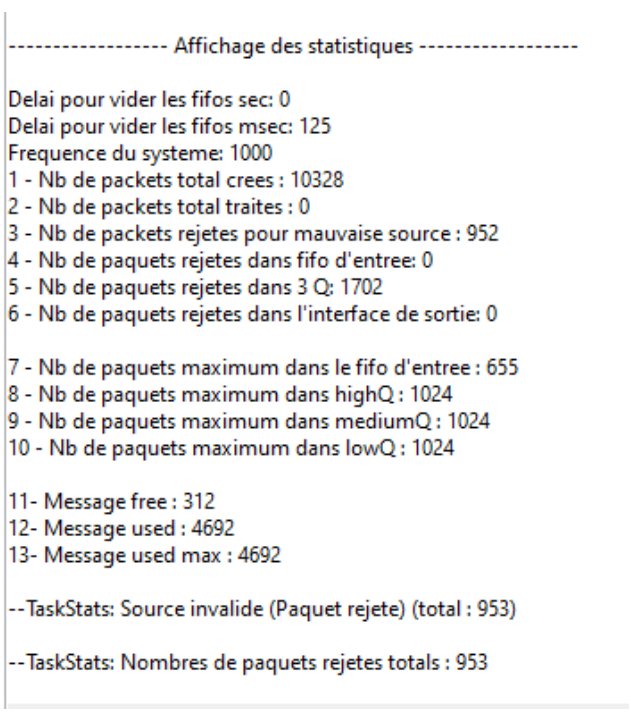

Figure 2 : Fréquence 1000Hz après que le fifo déborde

Comme nous pouvons observer sur les captures d'écrans ci-haut, lorsque le délai pour vider les fifos est à 250ms, que l'attente active pour les ticks est de 3 ticks et que la fréquence des ticks est de 1000Hz, toutes les fifos sont à moins de 50% de capacité, puisque les statistiques 8, 9 et 10 sont plus bas que 50% de 1024 (512), qui est la taille des fifos. De plus, selon la statistique 5, aucun paquet n'est rejeté dans les 3 files lorsque la fréquence des ticks est de 1000Hz et que le délai pour vider les fifos est de 250ms. Cependant, quand le délai pour vider les fifos baisse à 125ms avec une attente active de 3 ticks et une fréquence de 1000Hz, nous pouvons observer dans la statistique 5 que les files débordent, entraînant une perte de paquets. De plus, le nombre maximum de paquet excède 50% de 1024, qui est la taille des fifos. En fait, dans ce cas-ci, les fifos sont à capacité maximale.

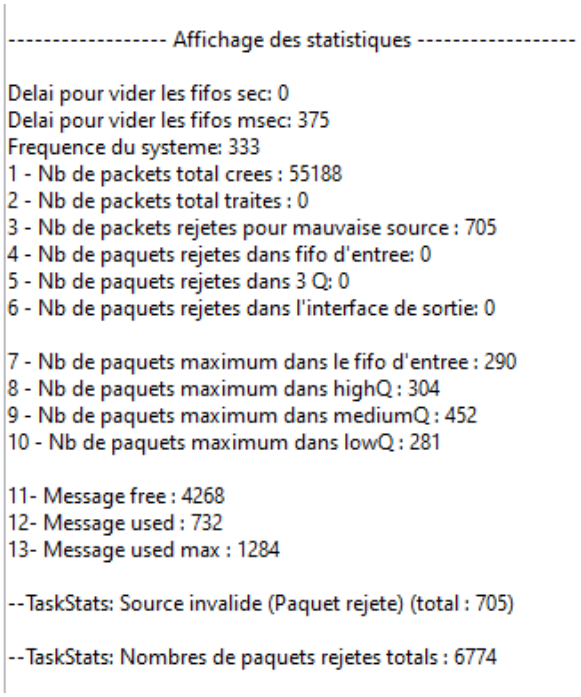

Figure 3 : Fréquence 333Hz juste avant que le fifo déborde

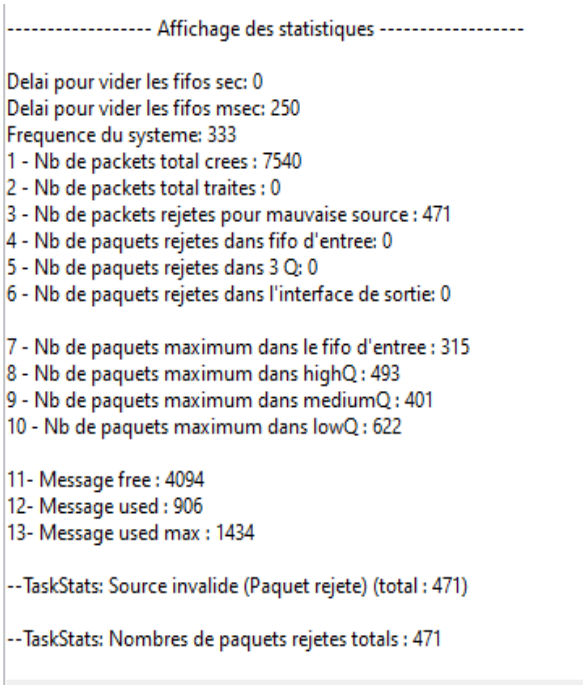

Figure 4 : Figure 333Hz après que le fifo déborde

Comme nous pouvons observer sur les captures d'écrans ci-haut, lorsque le délai pour vider les fifos est à 375ms, que l'attente active pour les ticks est de 1 tick et que la fréquence des ticks est de 333Hz, toutes les fifos sont à moins de 50% de capacité, puisque les statistiques 8, 9 et 10 sont plus bas que 50% de 1024. De plus, selon la statistique 5, aucun paquet n'est rejeté dans les 3 files lorsque la fréquence des ticks est de 333Hz et que le délai pour vider les fifos est de 375ms. Cependant, quand le délai pour vider les fifos baisse à 250ms avec une attente active de 1 tick et une fréquence de 333Hz, nous pouvons observer que le nombre maximum de paquet excède 50% de 1024, qui est la taille des fifos. En effet, dans la file de plus basse priorité, le nombre de paquets est 622, ce qui est plus grand que 512, 50% de 1024.

Il y a une différence entre le moment que les fifos débordent dans le cas que la fréquence d'horloge est à 1000Hz avec un WAITFORComputing à 3 ticks et que la fréquence d'horloge est à 333Hz avec un WAITFORComputing à 1 tick à cause de comment OSTimeDly est implémenté. En effet, le délai est arrondi

au prochain tick, donc avec une fréquence de 333Hz, les délais sont en fait plus long que si on a le même délai avec une fréquence de 1000Hz, car le tick survient plus rapidement.

Question 4

Expliquez la nécessité de faire un passage de paramètres lors de la création des tâches TaskOutputPortTCB (lignes 591 à 594). Et pourquoi ici utiliser un tableau de tâches?

Il y a un passage de paramètres pour permettre de différencier les différents TaskOutputPort, du fait que les 3 TaskOutputPort exécutent la même tâche. De plus, on utilise un tableau pour les 3 différents TaskOutputPorts pour éviter la duplication de code.

---------- Affichage des statistiques ----------

Delai pour vider les fifos sec: 0
Delai pour vider les fifos msec: 250
Frequence du systeme: 1000
1 - Nb de packets total crees : 54098
2 - Nb de packets total traites : 0
3 - Nb de packets rejetes pour mauvaise source : 794
4 - Nb de paquets rejetes dans fifo d'entree: 0
5 - Nb de paquets rejetes dans 3 Q: 0
6 - Nb de paquets rejetes dans l'interface de sortie: 0

7 - Nb de paquets maximum dans le fifo d'entree : 1
8 - Nb de paquets maximum dans highQ : 71
9 - Nb de paquets maximum dans mediumQ : 73
10 - Nb de paquets maximum dans lowQ : 79

11- Message free : 4156
12- Message used : 849
13- Message used max : 926

--TaskStats: Source invalide (Paquet rejete) (total : 806)

--TaskStats: Nombres de paquets rejetes totals : 6733

---------- Affichage des statistiques ----------

Delai pour vider les fifos sec: 0
Delai pour vider les fifos msec: 250
Frequence du systeme: 1000
1 - Nb de packets total crees : 54098
2 - Nb de packets total traites : 0
3 - Nb de packets rejetes pour mauvaise source : 794
4 - Nb de paquets rejetes dans fifo d'entree: 0
5 - Nb de paquets rejetes dans 3 Q: 0
6 - Nb de paquets rejetes dans l'interface de sortie: 0

7 - Nb de paquets maximum dans le fifo d'entree : 1
8 - Nb de paquets maximum dans highQ : 71
9 - Nb de paquets maximum dans mediumQ : 73
10 - Nb de paquets maximum dans lowQ : 79

11- Message free : 4156
12- Message used : 849
13- Message used max : 926

--TaskStats: Source invalide (Paquet rejete) (total : 806)

--TaskStats: Nombres de paquets rejetes totals : 6733