Rapport du TP1 : Systèmes concurrents

Les moniteurs

Myriam ROBBANA

Octobre 2024

Table des matières

Table des figures

1 Lecteurs/rédacteurs : priorité rédacteurs

Cette stratégie privilégiant les rédacteurs dans un fichier commun. En effet, ici, un rédacteur en attente bloque l'accès aux lecteurs, garantissant l'exclusivité des écritures. La variable *EC* indique si une écriture est en cours, tandis que *nblect* et *nbRatt* comptent les lecteurs actifs et les rédacteurs en attente. Les conditions *EcrireOK* et *LireOK* suspendent respectivement les lecteurs et rédacteurs lorsque l'accès n'est pas possible. Si aucun rédacteur n'est en attente après une écriture, les lecteurs peuvent alors accéder, sinon la priorité reste aux rédacteurs.

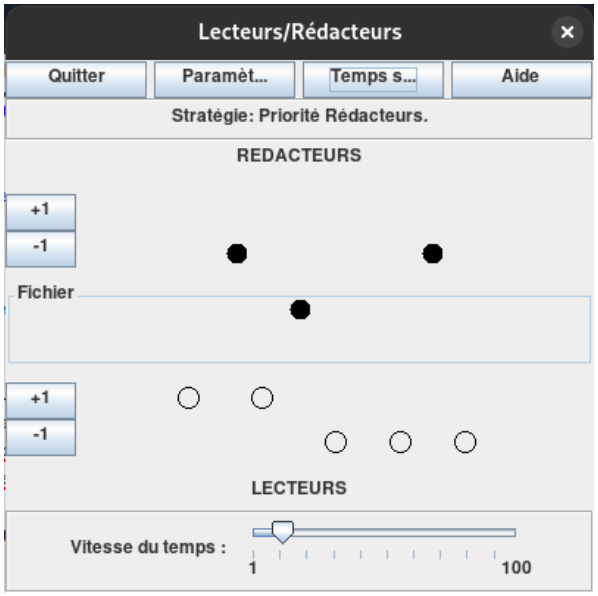

FIGURE 1 – Lecteurs/rédacteurs : priorité rédacteurs : Rédacteur dans le fichier.

On observe bien dans la *Figure*1 que si un rédacteur rédige dans le fichier, aucun lecteur ne peut accéder au fichier. Les lecteurs demandeurs doivent donc attendre que le rédacteur finisse de rédiger.

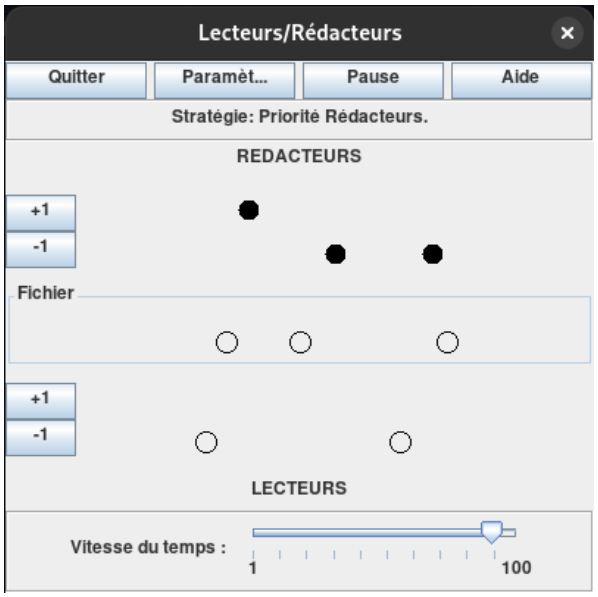

FIGURE 2 – Lecteurs/rédacteurs : priorité rédacteurs : Lecteurs dans le fichier.

Cette fois, on observe bien dans la *Figure*2 que dès que le rédacteur sort du fichier, les lecteurs demandant à lire peuvent tous à la fois accéder au fichier pour lire.

Le fichier fourni pour cette partie est : `LectRed_PrioRedacteur.java`.

2 Lecteurs/rédacteurs : Stratégie FIFO

Cette stratégie FIFO dans la gestion des lecteurs et des rédacteurs nous permet d'accorder l'accès aux ressources dans l'ordre d'arrivée des demandes. Ainsi, les processus sont mis en attente dans une file d'attente, et c'est toujours le premier arrivé qui est servi en premier. Les lecteurs et les rédacteurs attendent donc leur tour sans que personne n'ait de priorité, mais les rédacteurs ne peuvent accéder aux ressources que lorsque tous les lecteurs ont terminé. La gestion se fait via un *sas*, qui bloque les nouvelles lectures lorsqu'une écriture est en attente.

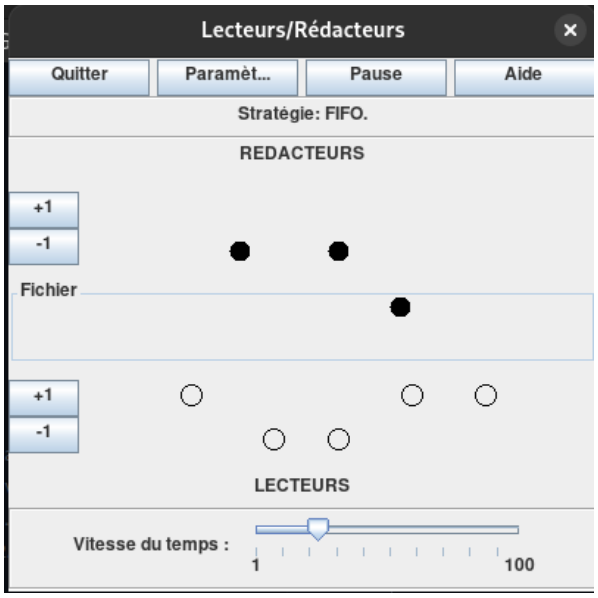

FIGURE 3 – Lecteurs/rédacteurs : Stratégie FIFO

Le fichier fourni pour cette partie est : `LectRed_FIFO.java`.

3 Allocateurs

Dans cette partie, on s'interesse au problème de l'allocation de ressources. Ainsi, dans ce modèle, le nombre de ressources demandées est inférieur au nombre total de ressources. De plus, le nombre de ressources libérées correspond au nombre de ressources précédemment obtenues. Et enfin, une activité ne demande pas de ressources si elle en a déjà, ni n'en libère sans en avoir.

3.1 Allocateurs : priorité aux petits demandeurs

Cette partie du rapport traite de la stratégie d'allocation de ressources, où la priorité est donnée aux petits demandeurs. L'objectif de cette stratégie est de permettre un accès plus rapide aux ressources pour les activités qui en demandent une faible quantité. En privilégiant les petites demandes, on favorise une répartition équitable et une utilisation optimale des ressources disponibles.

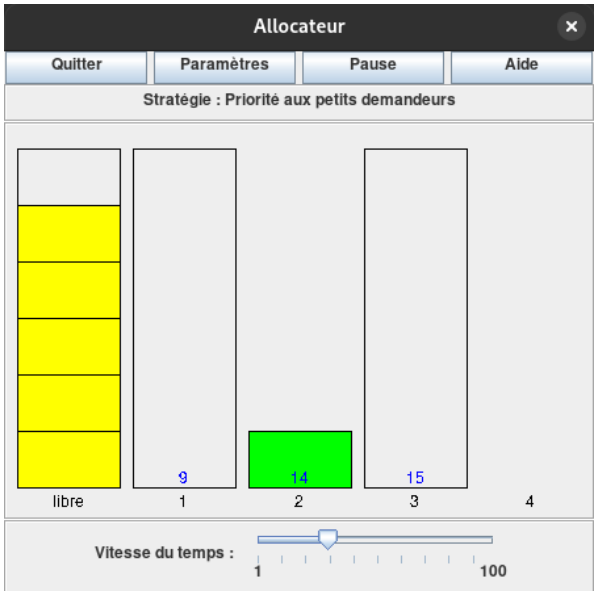

FIGURE 4 – Allocateurs : priorité aux petits demandeurs

Le fichier fourni pour cette partie est : `Allocateur_Petits.java`.

3.2 Allocateurs : Stratégie variante BestFit

La stratégie BestFit débloque en priorité les plus gros demandeurs pouvant être satisfaits par les ressources disponibles. À chaque libération de ressources, elle vérifie les demandes de la plus grande à la plus petite pour allouer au plus grand demandeur possible.

Cette approche optimise l'utilisation des ressources mais nécessite d'ajuster la simulation pour limiter les grandes demandes, afin d'éviter le blocage prolongé.

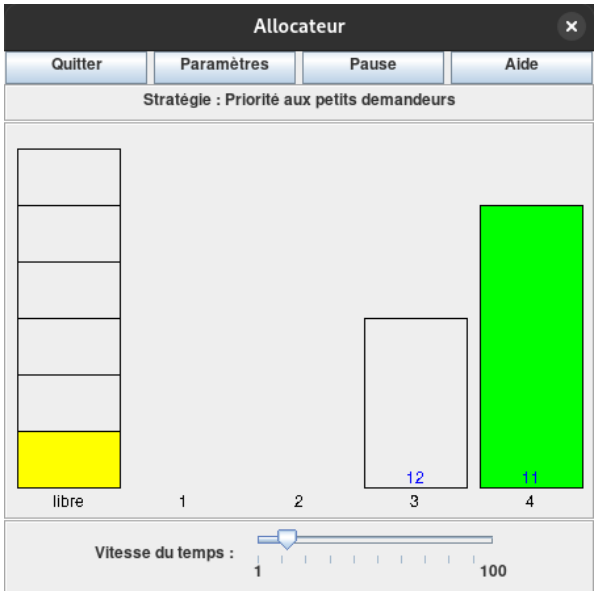

FIGURE 5 – Allocateurs : Stratégie variante BestFit

Le fichier fourni pour cette partie est : `Allocateur_BestFit.java`.