Simulation Serveur Web

Jean-Marc Celesti
Yohann Hako Moukam

30 Novembre 2015

1 Exercice 1

1.1 Question 1

Justifier pourquoi le temps moyen de réponse est d'au moins 49ms, indépendamment de λ.

Soit X la variable aléatoire qui estime le temps de réponse d'un utilisateur. On peut donc écrire $X = X_1 + X_2$ avec :
- X_1 le temps d'attente avant que la requête soit traitée par le serveur (dépendant de λ).
- X_2 le temps de traitement de la requête (type 1 et 2) par le serveur.

Ce qui nous donne :

$$E[X] = E[X_1] + E[X_2] = f(\lambda) + E[X_2]$$

avec :

$$E[X_2] = 2 * RTT + E[R_1] + E[R_2]$$

- R_1 suit une loi uniforme sur (0 ; 30) : $E[R_1] = 15$
- R_2 suit une loi exponentielle de paramètre $\lambda = 0.1$. On a donc $E[R_2] = \frac{1}{\lambda} = 10$

d'ou :

$$E[X_2] = 2 * 24 + 15 + 10 = 49$$

Par conséquent :

$$E[X] = E[X_1] + 49 = f(\lambda) + 49 \geq 49$$

1.2 Question 2

Programmer un simulateur à évènement discret et le décrire brièvement dans votre compte rendu

Le principe du simulateur est le suivant : Tout d'abord une phase d'initialisation, où des dates d'arrivée sont générées via une loi exponentielle de paramètre λ pour modéliser un processus de Poisson. Ces évènements (date d'arrivée,R_1) sont ensuite placés dans une file de priorité.

Les évènements R_1 sont décrits par la classe *Type1*. Ils possèdent comme attribut principal la date de départ. Les évènements R_2 (classe *Type2*) possèdent également une date de départ, en plus de conserver le temps écoulé depuis le début de la première requête qui l'a engendrée.

Le simulateur conserve une horloge qui est mise à jour après chaque évenement. Si au moment de retirer un évenement de la file, sa date théorique d'exécution est inférieure à celle de l'horloge (ce qui signifie donc que l'évènement est resté en attente avant d'être traité), on met à jour le temps d'attente de l'utilisateur. Au contraire, si la valeur de l'horloge est inférieure, on met à jour l'horloge (à la date d'execution de l'évènement) avec un temps d'attente nul.

Lorsqu'un évènement est retiré de la file, si il est de *Type1*, on détermine le temps de la requête (sans oublier le RTT) et on ajoute à la file d'attente un évènement de *Type2* avec en mémoire le temps écoulé. Si il est de *Type2*, on calcule le temps de réponse de l'utilisateur.

Une liste des utilisateurs connectés est mise à jour à chaque itération. En effet, le simulateur regarde dans la file d'attente toutes les arrivées qui sont inférieures à la date actuelle. A la fin d'un évènement de *Type2*, l'utilisateur est supprimé de la liste.

1.3 Question 3

Pour quelles valeurs de λ [10 ; 100] requêtes/secondes le système vous semble stable ?

La valeur $\lambda = 30$ semble être une valeur de lambda stable. En dessous, le système commence à saturer et le nombre d'utilisateurs en attente augmente de manière exponentielle. Les figures ci-dessous nous montrent la différence entre un système stable et un système instable.

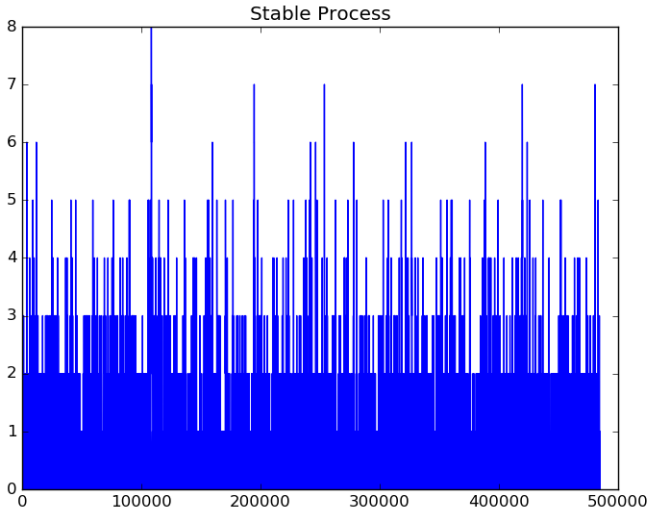

Figure 1: $\lambda = 100$, $users = 5000$

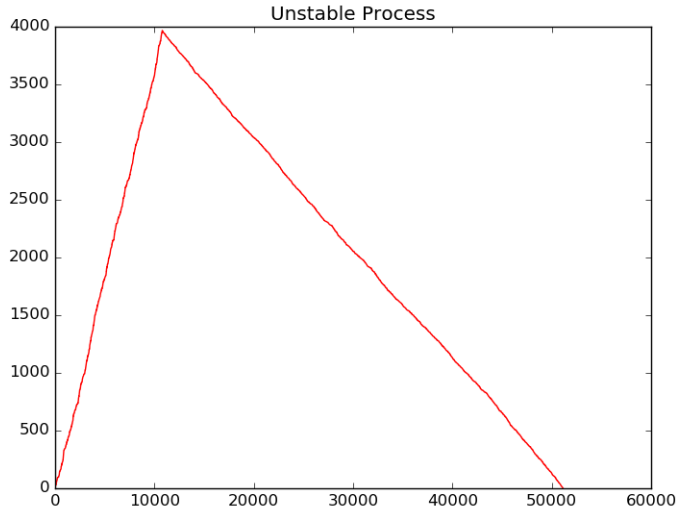

Figure 2: $\lambda = 10$, $users = 5000$

La différence est logique et s'explique par le fait que plus il y a d'espacement entre les différentes arrivées (donc plus λ est grand), plus un utilisateur attend avant que sa requête ne soit traitée,car les arrivées s'enchâinent plus vite que le traitement des requêtes par le serveur. Lorsque λ est faible (≤ 20), les arrivées sont très proches dans le temps et le système commence à diverger.

1.4 Question 4