Devoir Evaluation de Performance de OM2M basé sur des modèles de files d'attentes

Question 1 : On fixe $N = 1,\ \lambda_H = 2,\ \mu_H = 1,\ \mu_R = 1,\ \mu_D = 1\ et\ C = 1$

a - i Fixons $p = 1$

Pour le calcul analytique commençons par calculer les différents γ_i

$\gamma_H = \gamma_T = p * \gamma_R$ et $\gamma_D = (1 - p) * \gamma_R$

On pose γ_R = 1 et on obtient :
$\gamma_T = \gamma_H = 1$ et $\gamma_D = 0$

Des expressions de γ_i on peut déduire les expressions de ρ_i

$\rho_i = \frac{\gamma_i}{\mu_i}$

donc $\rho_H = \rho_R = 1\ et\ \rho_D = 0\ et\ \rho_T = \frac{1}{2}$

Calculons maintenant les valeurs de $G_4(N)$ et $\Phi_R(n_R)$

$$G_4(N) = \sum_{n_T+n_H+n_R+n_D=N} \rho_T^{n_T} * \rho_H^{H} * \frac{\rho_R^{n_R}}{\Phi_R(n_R)} * \rho_D^{n_D} = \frac{1}{2} + 1 + 1 + 0 = \frac{5}{2}$$

Car comme N=1 on a 4 cas possible où un seul des $n_i = 1$ et les autres sont nuls

$$\Phi_R(n_R) = \prod_{j=1}^{n_R} min(j, C) = 1 \quad \forall\, n_R \in \{0,\ 1\}$$

on sait que $n_R \leq 1$ car $N = 1$

Nous pouvons maintenant calculer les différents $\pi(n_T, n_H, n_R, n_D)$

$$\pi(1,0,0,0) = \frac{2}{5} * (\frac{1}{2})^1 * 1^0 * \frac{1^0}{1} * 0^0 = \frac{1}{5}$$

$$\pi(0,1,0,0) = \frac{2}{5} * (\frac{1}{2})^0 * 1^1 * \frac{1^0}{1} * 0^0 = \frac{2}{5}$$

$$\pi(0,0,1,0) = \frac{2}{5} * (\frac{1}{2})^0 * 1^0 * \frac{1^1}{1} * 0^0 = \frac{2}{5}$$

$$\pi(0,0,0,1) = \frac{2}{5} * (\frac{1}{2})^0 * 1^0 * \frac{1^0}{1} * 0^1 = 0$$

On peut maintenant calculer le throughput des différentes node
$\theta_i = (1 - \pi_i(0)) * \mu_i$ car on a un seul serveur.

avec $\pi_T(n) = \sum_{n_H+n_R+n_D=N-n} \pi(n, n_H, n_R, n_D)$, la formule étant analogue pour les autres stations.
On a donc:

$$\theta_T = (1 - \pi(1,0,0,0)) * 2 = \frac{2}{5}$$

$$\theta_H = (1 - \pi(0,1,0,0)) * 1 = \frac{2}{5}$$

$$\theta_R = (1 - \pi(0,0,1,0)) * 1 = \frac{2}{5}$$

$$\theta_D = (1 - \pi(0,0,0,1)) * 1 = 1$$

Calculons maintenant le nombre moyen de requête par système pour le serveur HTTP ainsi que les ressources:

$$\overline{A_i} = \sum_n \pi_i(n) * n$$

$$\overline{A_R} = \pi_R(1) = \pi(0,0,1,0) = \frac{2}{5}$$

$$\overline{A_H} = \pi_H(1) = \pi(0,1,0,0) = \frac{2}{5}$$

On sait que le temps de Sojourn moyen est

$$\overline{B} = \overline{B_H} + \frac{\overline{B_R} + (1-p) * \overline{B_D}}{p} = \overline{B_H} + \overline{B_R} \; car \; p = 1$$

avec $\overline{B_i} = \frac{\overline{A_i}}{\overline{\theta_i}}$

donc $\overline{B} = \frac{\frac{2}{5}}{\frac{2}{5}} + \frac{\frac{2}{5}}{\frac{2}{5}} = 2$

On a donc que le temps de Sojourn moyen est de 2

Pour la probabilité de réjection κ on a la formule suivante

$$\kappa = \sum_{n_H+n_R+n_D=N} \pi(0, n_H, n_R, n_D) = \pi(0, 1, 0, 0) + \pi(0, 0, 1, 0) + \pi(0, 0, 0, 1)$$

$$\kappa = \frac{2}{5} + \frac{2}{5} + 0 = \frac{4}{5}$$

Donc la probabilité de réjection est de $\frac{4}{5}$

ii - En faisant run la simulation du réseau fermé avec un nombre d'itération de 100000, on obtient les résultats suivants :

$\overline{B} = 2.010930842489235$
$\kappa = 0.800897641428358$

On retrouve ici les mêmes valeurs (à une petite approximation prêt) que les résultats trouvés avec la méthode analytique.

b.i -
Si notre objectif est de réduire la probabilité de réjection, nous devons doubler la capacité du serveur HTTP ou bien du serveur de ressource R dépendamment de la valeur de p.

- En effet, si p=1, on a avec la formule de κ que si l'on double μ_R *ou bien* μ_H cela revient au même. De plus, avec une approche logique cela peut se confirmer car comme le circuit est déterministe et fait:

Server Thread -> Serveur HTTP -> Serveur Ressource -> Server Thread

et bien si on augmente la vitesse du serveur HTTP ou celui de Ressource on arrivera plus vite au serveur Thread donc la probabilité de réjection diminuera.

- Si p=0, la probabilité de réjection tendra vers 1 comme seule la première requête pourra être traiter. Cette requête ne pourra jamais rendre son thread car elle sera bloquée entre le serveur de Ressource et le serveur de la Database.

- Si 0<p<1 alors on a qu'il vaut mieux doubler la capacité du serveur de Ressource car c'est le serveur qui aura le plus gros ρ donc doubler sa vitesse permettra de le diminuer plus fortement et donc d'avoir une valeur minimale pour κ (probabilité de réjection). En effet en prenant $\gamma_R = 1$:

  $\rho_R = \frac{\gamma_R}{\mu_R} = \frac{1}{\mu_R}, \rho_H = \frac{\gamma_H}{\mu_H} = \frac{p}{\mu_H}, \rho_D = \frac{\gamma_D}{\mu_D} = \frac{(1-p)}{\mu_D}, \rho_T = \frac{\gamma_T}{\mu_T} = \frac{p}{\mu_T}$

b.ii -

Si notre objectif est de réduire le mean sojourn time alors les réponses ne seront pas forcément les mêmes.

Comme $\overline{B}$ dépend directement de p on peut en déduire que:

-Si p = 1 alors $\overline{B_H}$ et $\overline{B_R}$ auront le même coefficient dans le calcule de $\overline{B}$ donc là encore, on peut doubler indépendamment la vitesse sur serveur HTTP ou bien du serveur Ressource pour réduire le temps de Sojourn.

-Si p=0 alors $\overline{B} \rightarrow +\infty$ donc le système ne converge pas. On passe un temps infini dans le système comme on ne peut pas sortir de la boucle serveur Ressource -> serveur Database.

-Si 0<p<1 alors on a que le coefficient de $\overline{B_H}$ est plus important que celui de $\overline{B_R}$ (comme il est divisé par p). Il en va de même pour $\overline{B_D}$. On en déduit donc qu'il vaut mieux doubler la vitesse du serveur HTTP.

Question 2:

a- N=15 et p=0.5

Résultat de simulation:

Closed Loop:

Rejection probability : 0.0
Mean number request for T: 12.72573000247775 H: 0.11199832603467917
R: 2.050412367514408 D: 0.11185930397317503
Mean Sojourn time : 2.248870892651803

Opened Loop:

Rejection probability : 0.0
Mean number request for T: 12.74801500170324 H: 0.11099089208659774
R: 2.027983641799895 D: 0.1130104644101243
Mean Sojourn time : 2.250176958752671

Approximation infinite threads :

Mean number request for H: 0.1111111111111112 R: 2.848484848484849
D: 0.1111111111111112
Mean sojourn time : 3.0707070707070714

Approximation infinite threads single resource :

Mean number request for H: 0.1111111111111112 R: 0.6666666666666667
D: 0.1111111111111112
Mean sojourn time : 0.8888888888888891

Approximation infinite threads infinite resource :

Mean number request for H: 0.1111111111111112 R: 1.0
D: 0.1111111111111112
Mean sojourn time : 2.2222222222222223
---

b- N=15, p= 0.22

Résultat de simulation:

Closed Loop:

Rejection probability : 0.030735015445297345
Mean number request for T: 7.624571742274459 H: 0.10397590989727358
R: 6.76805232563991 D: 0.503400022188367
Mean Sojourn time : 7.73610437130879

Opened Loop:

Rejection probability : 0.047619047619047616
Mean number request for T: 7.446467367087596 H: 0.10643152577691747
R: 6.9327943636081315 D: 0.5143067435274242
Mean Sojourn time : 7.938571488685573

Approximation infinite threads :

Mean number request for H: 0.11111111111111112 R: 13.36178959600828
D: 0.5492957746478873
Mean sojourn time : 14.022196481767276

Approximation infinite threads single resource :

Mean number request for H: 0.11111111111111112 R: 9.992017771250847
D: 0.5492957746478873
Mean sojourn time : 10.652424657009842

Approximation infinite threads infinite resource :

Mean number request for H: 0.11111111111111112 R: 1.0
D: 0.5492957746478873
Mean sojourn time : 5.205861431213544
---

c- N=30, p=0,5

Résultat de simulation:

Closed Loop:

Rejection probability : 0.0
Mean number request for T: 27.73910853076271 H: 0.11163245080465979
R: 2.0400778074766626 D: 0.10918121095555387
Mean Sojourn time : 2.2672220833132823

Opened Loop:

Rejection probability : 0.0
Mean number request for T: 27.75179021283652 H: 0.11144978994983915
R: 2.028464914449845 D: 0.10829508276454229
Mean Sojourn time : 2.265923094071457

Approximation infinite threads :

Mean number request for H: 0.11111111111111112 R: 2.848484848484849 D:
0.11111111111111112
Mean sojourn time : 3.0707070707070714

Approximation infinite threads single resource :

Mean number request for H: 0.11111111111111112 R: 0.6666666666666667
D: 0.11111111111111112
Mean sojourn time : 0.8888888888888891

Approximation infinite threads infinite resource :

Mean number request for H: 0.11111111111111112 R: 1.0
D: 0.11111111111111112
Mean sojourn time : 2.2222222222222223
---

d- N=30, p=0,22

Résultat de simulation:

Closed Loop:

Rejection probability : 0.006354816823191739
Mean number request for T: 18.82638605891073 H: 0.10946565880192194
R: 10.514903734134426 D: 0.5492445481531946
Mean Sojourn time : 11.234295954645182

Opened Loop:

Rejection probability : 0.01248992747784045
Mean number request for T: 18.098010732850433 H: 0.1099774937762361
R: 11.23303247159304 D: 0.5589793017803747
Mean Sojourn time : 11.832548413100042

Approximation infinite threads :

Mean number request for H: 0.11111111111111112 R: 13.36178959600828
D: 0.5492957746478873
Mean sojourn time : 14.022196481767276

Approximation infinite threads single resource :

Mean number request for H: 0.11111111111111112 R: 9.992017771250847
D: 0.5492957746478873
Mean sojourn time : 10.652424657009842

Approximation infinite threads infinite resource :

Mean number request for H: 0.11111111111111112 R: 1.0
D: 0.5492957746478873
Mean sojourn time : 5.205861431213544

En comparant le nombre moyen de requêtes dans le réseau, le nombre N et la qualité des approximations nous pouvons conclure les choses suivantes.

On peut juger de la charge d'un réseau par la présence du nombre moyen de requête dans le serveur de Thread. Plus ce nombre est proche de N, plus le système n'est pas très chargé. A contrario, si ce nombre tend vers 0, cela signifie que le système est très chargé.

-Dans les cas b et d, l'approximation des threads infinis avec une seule ressource est la meilleure. Cela peut s'expliquer par le fait que comme p est faible alors beaucoup de requêtes font une boucle entre le serveur de Ressource et le Serveur de la Database ce qui induit une charge de réseau importante. Or le modèle avec l'approximation des threads infini avec une seule ressource permet de modéliser ce genre de réseau chargé avec une probabilité de réjection faible.

-Pour ce qui est des cas a et c, comme le réseau n'est pas très chargé comme la probabilité p est plus importante alors c'est le modèle avec l'approximation des threads infini et des ressources infinis qui donnent les meilleurs résultats.

-On peut noter que le modèle des thread infini n'est pas excellent, mais il reste toujours bon en moyenne alors que pour les deux autres approximations, les résultats sont soit excellents soit médiocre. Cela peut s'expliquer par le fait que l'approximation des thread infinis ne fait une hypothèse que sur une probabilité de réjection faible mais aucune sur la charge du réseau.