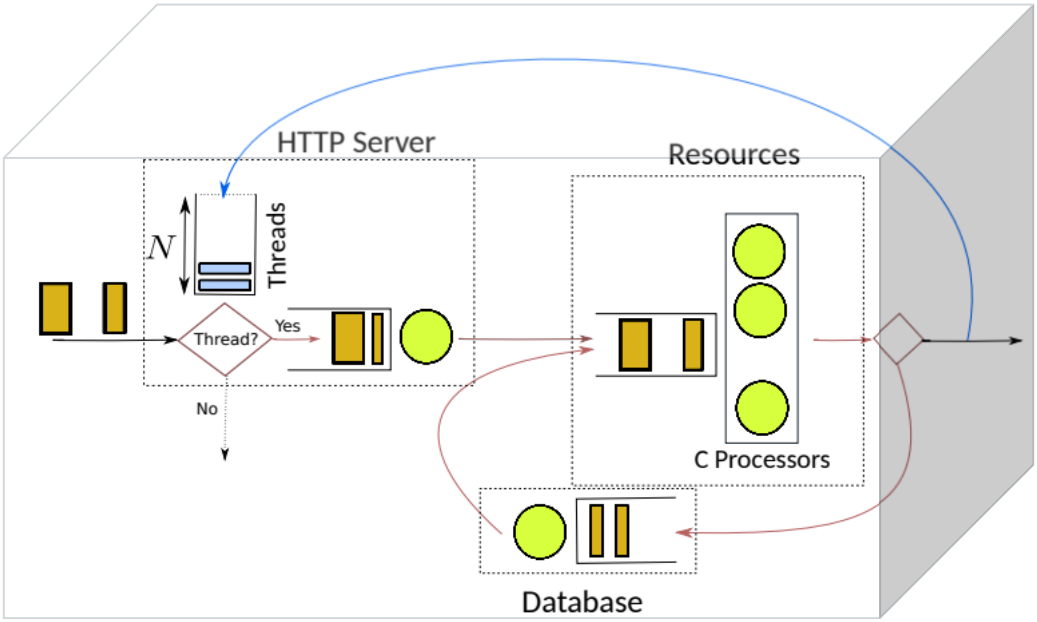
TEO GENEAU

MIDDLEWARE IOT PERFORMANCE EVALUATION

25 janvier 2022



Informations relatives au document

INFORMATIONS GÉNÉRALES

Auteur(s)	Teo Geneau
Lieu	INSA Toulouse
Version	V3.0
Référence	EA1

HISTORIQUE DES MODIFICATIONS

Version	Date	Rédigé par	Modifications
V3.0	25/01/2022	Teo Geneau	Version Finale : ajouts de chapitres et corrections

TABLE OF CONTENTS

TABLE OF CONTENTS

INTRODUCTION

The goal of this project is to get experience with performance evaluation of OM2M Middleware based on queueing models. OM2M is an architecture that serves as a middleware for IoT devices. During these lab exercises we will expand upon some calculations performance measures of interest using the stochastic model of our network that we have built during the course. We will then go through the simulation of a Markov chain that we wil use to compute some features like probability of rejection and the mean sojourn time performance measures.

Part 1 : performance measures for network with one thread and one processor

We will determine the mean sojourn time of requests that are served and the probability of request being rejected. For that, we will proceed with two methods, first, we will analysis the system using queueing theory : we will do an exact analysis using closed queueing network.
In a second time, we will try to validate our results obtained using the analysis with the simulation of Markov chain representing the system as a closed network.

(a) - Analytical expression

We set the features of the system such as :

- $N = 1$
- $\lambda_H = 2$
- $\mu_H = \mu_R = \mu_D = 1$
- $C = 1$

And we set $p = 1$, we will compute the probability of rejection and the mean sojourn time of requests. For that, we will study the system as a closed queueing network model.

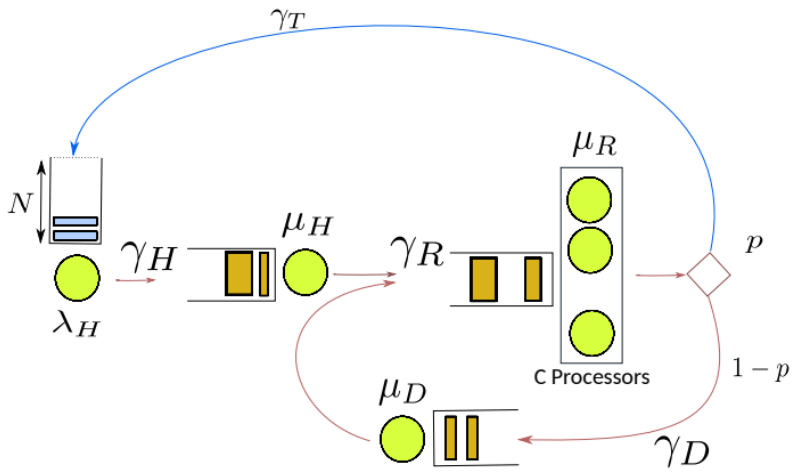


Figure 1 : Closed queueing network model

We have four nodes : Thread(T), HTTP (H), Resources (R) and Database (D). We observe that T, H and D behave like single server queues while R it is just as a C-server queue.

We determine the routing matrix M :

$$M = \begin{array}{c} \\ T \\ H \\ R \\ D \end{array} \begin{array}{c} \begin{array}{cccc} T & H & R & D \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & 0 & 0 & 1-p \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Probability of rejection

To determine the probability of rejection we need first to compute the load on the system ρ :

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

Thus, we need to compute the effective arrival rate in each station i of the system, with γ a solution of $\gamma = \gamma M$, we determine these equations :

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

With p=1 and γ_R=1 we can solve the system :

$$\gamma_H = \gamma_T = \gamma_R = 1$$
$$\gamma_D = (1-p)\gamma_R = 0$$

Thus :

$$\begin{cases} \rho_T = \dfrac{\gamma_T}{\lambda_H} = \dfrac{1}{2} \\ \rho_H = \dfrac{\gamma_H}{\mu_H} = 1 \\ \rho_R = \dfrac{\gamma_R}{\mu_R} = 1 \\ \rho_D = \dfrac{\gamma_D}{\mu_D} = 0 \end{cases}$$

We can deduce ϕ_R :

$$\phi_R(n_R) = \prod_{j=1}^{n_R} \min(j, C) = 1$$

We must consider each case for n_i, knowing that $N = 1$ and $n_T + n_H + n_R + n_D = N$, possible cases are :

$$\begin{matrix} n_T & n_H & n_R & n_D \end{matrix}$$
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} n_T \\ n_H \\ n_R \\ n_D \end{matrix}$$

Then, we obtain the normalization constant $G_4(N)$:

$$G_4(N) = \sum_{n_T+n_H+n_R+n_D=N} \rho^{n_T} \rho_H^{n_H} \frac{\rho_R^{n_R}}{\phi_R(n_R)} \rho_D^{n_D}$$

Knowing that $x^0 = 1$ and that $\phi_R(n_R) = 1$, we have :

$$G_4(N) = \frac{1}{2} + 1 + 1 + 0 = \frac{5}{2}$$

Finally, we can compute the probability of rejection define by the formula :

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

Indeed, a rejection occurs only when $n_T = 0$

We must compute $\pi(0, n_H, n_R, n_D)$ for these possible cases :

$$\begin{matrix} n_T & n_H & n_R & n_D \end{matrix}$$
$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} n_T \\ n_H \\ n_R \\ n_D \end{matrix}$$

Thus :

$$\text{For } n_H: \quad \pi(0,1,0,0) = \frac{1}{G_4(N)} \quad \rho^{n_T}\rho_H^{n_H}\frac{\rho_R^{n_R}}{\phi_R(n_R)}\rho_D^{n_D} = \frac{2}{5}.\rho_H^{n_H} = \frac{2}{5}$$
$$\text{For } n_R: \quad \pi(0,0,1,0) = \frac{1}{G_4(N)} \quad \rho^{n_T}\rho_H^{n_H}\frac{\rho_R^{n_R}}{\phi_R(n_R)}\rho_D^{n_D} = \frac{2}{5}.\rho_R^{n_R} = \frac{2}{5}$$
$$\text{For } n_D: \quad \pi(0,0,0,1) = \frac{1}{G_4(N)} \quad \rho^{n_T}\rho_H^{n_H}\frac{\rho_R^{n_R}}{\phi_R(n_R)}\rho_D^{n_D} = \frac{2}{5}.\rho_D^{n_D} = 0$$

And finally, we compute $P_{rejection}$:

$$P_{rejection} = \sum_{n_H+n_R+n_D=N} \pi(0, n_H, n_R, n_D) = \frac{2}{5} + \frac{2}{5} = \frac{4}{5} = 0.8$$

We have determined $P_{rejection} = 80\%$.

Mean sojourn time

To compute the mean sojourn time of requests we first use Little's Law to compute the mean sojourn time in each node :

$$\overline{B_i} = \frac{\overline{A_i}}{\theta_i} = \frac{\sum_n \pi_i(n)\, n}{\sum_{n\geq 1} \pi_i(n)\, \mu_i}$$

With $\overline{A_i}$ the mean number of requests and θ_i the throughput in each node.

As N=1 and $\overline{B_i} = 0$ for n=0 we can write :

$$\overline{B_i} = \frac{\overline{A_i}}{\theta_i} = \frac{1}{\mu_i}$$

Which gives us :

$$\begin{cases} \overline{B_T} = \frac{1}{2} \\ \overline{B_H} = 1 \\ \overline{B_R} = 1 \\ \overline{B_D} = 0 \end{cases}$$

What is more, we know that the mean sojourn time of request is given by :

$$\overline{B} = \overline{B_H} + \overline{B_{R+D}}$$

With :

$$\overline{B_{R+D}} = \overline{B_R} + (1-p)(\overline{B_D} + \overline{B_{R+D}})$$

As we have set p=1, we obtain :

$$\overline{B_{R+D}} = \overline{B_R}$$

And finally :

$$\overline{B} = \overline{B_H} + \overline{B_R} = 1 + 1 = 2$$

(a) - Simulation

We have simulated the closed network in python and computed the exact performance measures for the network with N=1. We have set the stopping criterion (which is the time stopping time of 1 200 000) in order to obtain convergent performance measures by reaching stationarity.

We can see the results of the simulation in the figure below

```
The Probability of rejection is:  0.8007363670210411

The Mean sojourn time is:  1.9994312600836615
```

Figure 2 : simulation results

We can observe almost overlapping results between the analytical and simulation results for both the probability of rejection and the mean sojourn time.

(b) – Model analysis

In this part, we assume that it is possible to multiply the capacity of one of the servers by a factor of 2. The point is to determine which server should be chosen in order to reduce the probability of rejection and the mean sojourn time depending on the value of p varying from 0 to 1.

First, we use the mean sojourn time of request formula :

$$\overline{B} = \overline{B_H} + \overline{B_{R+D}}$$

With :

$$\overline{B_{R+D}} = \overline{B_R} + (1-p)(\overline{B_D} + \overline{B_{R+D}})$$

$$\overline{B_{R+D}} = \frac{B_R + (1-p)\overline{B_D}}{p}$$

As we can observe, only the servers R and D are involved in the mean sojourn time calculation, we will derive $\overline{B_{R+D}}$ in order to determine which one of those servers must be enhanced to reduce the mean sojourn time :

$$\frac{d\overline{B_{R+D}}}{dB_R} = \frac{1}{p}$$

$$\frac{d\overline{B_{R+D}}}{d\overline{B_D}} = \frac{1-p}{p} = \frac{1}{p} - 1$$

We have : $\frac{1}{p} > \frac{1}{p} - 1$, therefore $\frac{d\overline{B_{R+D}}}{dB_R} > \frac{d\overline{B_{R+D}}}{d\overline{B_D}}$ which means we must multiply the capacity of the server R in order to reduce the mean sojourn time, indeed, it is the coefficient that has the greatest impact on the calculation of the mean sojourn time.

Now, for the probability of rejection if we need to compute the effective arrival rate in each station i of the system, with γ a solution of $\gamma = \gamma M$, we have determined these equations :

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

Regardless of p, we can observe that each term depends on γ_R, the greater γ_R is, the greater the other terms are and the great the probability of rejection is which means that once again, we must multiply the capacity of the server R in order to reduce the probability of rejection.

See appendix 1 for more details on the simulation.

Part 2 : performance measures for network with one thread and one processor

In this part, the main goal is to simulate and compute the mean sojourn time for a closed system, an opened system as well as the three approximations.

We set the features of the system such as :

- $\lambda_H = 1$
- μ_H =10
- μ_R =1
- $\mu_D = 10$
- $C = 5$

And with $p = 0.5\ or\ 0.22$ and N=15 or 30.

See appendix 2 for more details on the simulation.

CONCLUSION

During this exercise, we were able to explore different performance evaluation measures such as the probability of rejection and the mean sojourn time. We were able to obtain a theoretical model for a closed queueing network model and validate it through the simulation which means that we could use the theoretical model and the simulation tool to study in a relevant way the behaviour of a OM2M middleware. Nevertheless, the theoretical model is based on certain assumptions, and it could be improved by considering them like taking account to buffers during transmission of the end devices.

Appendix 1 : Python algorithm for the first part

```
# -*- coding: utf-8 -*-
"""
@author: Teo Geneau
"""

#############################################################################
# Simulation of the closed network question 1
#############################################################################


import numpy as np
import matplotlib.pyplot as plt
import time


#############################################################################
# Parameter declaration
#############################################################################

λ_H = 2
μ_H = 1
μ_R = 1
μ_D = 1
C = 1
p = 1
N = 1

γ_H = 1
γ_T = 1
γ_R = 1
γ_D = 0


ρ_T = 1/2
ρ_H  = 1
ρ_R  = 1
ρ_D  = 0



Xi = np.array([N,0,0,0]) # State matrix [XT , XH , XR, XD] with XT initialized to N=1


CNT = 0 # We count every time we pass by the state XT, that we use to compute the mean
sojourn time


time_Xi = np.array([0.0,0.0,0.0,0.0]) # Array stocking the time passed in each state



while(np.sum(time_Xi) < 120000): #Stopping criterion, sufficiently long to reach
stationarity

  if Xi[0] == 1: # We check that XT > 0

    #Here, the scale parameter Beta=1/λ_H

    time_Xi[0] += np.random.exponential(scale=1/λ_H)

    Xi = [0,1,0,0] # XH < XT after exp(λ_H)
```

```
  elif Xi[1] == 1: # We check that XH > 01

    #Here, the scale parameter Beta=1/μ_H

    time_Xi[1] += np.random.exponential(scale=1/μ_H)

    Xi = [0,0,1,0]  # XR < XH after exp(μ_H)

  elif Xi[2] == 1: # We check that XR > 0

    #Here, the scale parameter Beta=1/μ_R

    time_Xi[2] += np.random.exponential(scale=1/μ_R)


    if(np.random.uniform()<= p) :
      Xi = [1,0,0,0]
      CNT += 1
    else :
      Xi = [0,0,0,1]

  elif Xi[3] == 1: # We check that XD > 0

    #Here, the scale parameter Beta=1/μ_D

    time_Xi[3] +=np.random.exponential(scale=1/μ_D)

    Xi = [0,0,1,0] # XR < XD after exp(μ_D)


##############################################################################
# Performance measures
##############################################################################


#Probability of rejection
#It is equivalent to 1-(Time XT)/Total Time

p_rej = 1-(time_Xi[0])/(np.sum(time_Xi))
print("\nThe Probability of rejection is: ",p_rej)


#Mean Sojourn Time

#Mean Sojourn Time = (Total Time - Initialization Time)/Cycle Number

Mean = (np.sum(time_Xi)-time_Xi[0])/(CNT)
print("\nThe Mean sojourn time is: ",Mean)


print("\nXT",time_Xi[0],"XH",time_Xi[1],"XR",time_Xi[2],"XD",time_Xi[3])
```

Appendix 2 : Python algorithm for the second part

```
# -*- coding: utf-8 -*-
"""
@author: Teo Geneau
"""

#############################################################################
# Simulation of the closed network question 2
#############################################################################

import numpy as np



λ_H = 1
μ_H = 10
μ_R = 1
μ_D = 10
C = 1
p = 0.5
N = 1

γ_H = 1
γ_T = 1
γ_R = 1
γ_D = 0


ρ_T = 1/2
ρ_H  = 1
ρ_R  = 1
ρ_D  = 0

time = 0


Xi = np.array([N,0,0,0]) # State matrix [XT , XH , XR, XD] with XT initialized to N=1


CNT = 0 # We count every time we pass by the state XT, that we use to compute the mean
sojourn time


time_Xi = np.array([0.0,0.0,0.0,0.0]) # Array stocking the time passed in each state



while(np.sum(time_Xi) < 120000): #Stopping criterion, sufficiently long to reach
stationarity
  Tab =
[np.random.exponential(scale=1/λ_H),np.random.exponential(scale=1/μ_H),np.random.expon
ential(scale=1/μ_R),np.random.exponential(scale=1/μ_D)]
  Min=np.argmin(Tab)


  if (Xi[0] > 0 and Min == 0): # We check that XT > 0

    time += Tab[0]
    time_Xi[0] += Tab[0]

    X[0] -= 1
    X[1] += 1
```

```
  elif (Xi[0] == 0 and Min == 0): # We check that XH > 01
    Tab[0] = np.random.exponential(scale=(1/λ_H)

  if (Xi[1] > 0 and Min == 1):

    time += Tab[1]
    time_Xi[1] += Tab[1]

    X[1] -= 1
    X[2] += 1

  elif (Xi[2] > 0 and Min == 2): # We check that XR > 0

    X[2] -= 1

    if(np.random.uniform()<= p) :
    time += Tab[2]
    time_Xi[2] += Tab[2]

    X[0] += 1
    CNT += 1


    else :
     X[3] += 1
     time += Tab[2]
     time_Xi[2] += Tab[2]

  elif (Xi[3] == 1 and Min == 3): # We check that XD > 0

    time += Tab[3]
    time_Xi[3] += Tab[3]

    X[3] -= 1
    X[2] += 1


##########################################################################
# Performance measures
##########################################################################


#Probability of rejection
#It is equivalent to 1-(Time XT)/Total Time

p_rej = 1-(time_Xi[0])/(np.sum(time_Xi))
print("\nThe Probability of rejection is: ",p_rej)


#Mean Sojourn Time

#Mean Sojourn Time = (Time passed in XR + XH)/Cycle Number

Mean = (time_Xi[1]+time_Xi[2])/(CNT)
print("\nThe Mean sojourn time is: ",Mean)


print("\nXT",time_Xi[0],"XH",time_Xi[1],"XR",time_Xi[2],"XD",time_Xi[3])
```