M|M|c

Corsetti, Ornela Milagros[a], Golzman, Gabriel[a], Bengoechea, Guadalupe María[a]

[a] *Universidad Tecnológica Nacional - Facultad Regional de Rosario*

Enunciado

En este trabajo práctico realizaremos la simulación de un modelo compuesto de una cola inicial que llega a 3 servidores (los cuáles poseen la misma tasa), seguido de tres colas que conectan con otros tres servidores, estos si con tasa de servicio distintas cada uno.

Palabras Clave:
corrida,simulación,servidor,cola,tasa de arribo, tasa de servicio, aleatorio,

Índice

1. Marco Teórico **2**
1.1. Definición . . . 2
1.2. Características . . . 2
1.3. Medidas de Rendimiento . . . 2
1.3.1. Número promedio de clientes en Cola . 2
1.3.2. Utilización promedio del servidor. . . . 2
1.3.3. Tiempo promedio de servicio . . . 2

2. Metodología **2**
2.1. Modelado . . . 2
2.2. Planteamiento . . . 3
2.2.1. Clase Simulación . . . 3
2.2.2. Clase Servidor . . . 3
2.2.3. Clase Cola . . . 3
2.3. Métodos . . . 3
2.3.1. Run . . . 3
2.3.2. Inicialización . . . 3
2.3.3. Tiempos . . . 3
2.3.4. Arribo . . . 3
2.3.5. Partida . . . 4
2.3.6. Algoritmos de Selección de Clientes en Cola . . . 4
2.3.7. Reporte, Analítica y Graficar. . . . 4
2.4. Hipótesis . . . 5

3. Casos de estudio **5**
3.1. Cola inicial con Algoritmo FIFO . . . 5
3.1.1. Sin aplicación de mejora en la segunda columna de colas . . . 5
3.2. Con aplicación de mejoras en la segunda columna de colas . . . 8
3.3. Cola inicial con Algoritmo LIFO . . . 11
3.3.1. Sin aplicación de mejora en la segunda columna de colas . . . 11
3.4. Con aplicación de mejoras en la segunda columna de colas . . . 13
3.5. Cola inicial con Algoritmo Prioridades . . . 16
3.5.1. Sin aplicación de mejora en la segunda columna de colas . . . 16
3.5.2. Con aplicación de mejora en la segunda columna de colas . . . 19

4. Conclusiones **22**
4.1. Comportamiento General de la Simulación. . . 22
4.2. Variación de Algoritmos de Selección en la primer cola. . . . 22
4.3. Mejora Aplicada. . . . 22

Correos Electrnicos: ocorsetti@frro.utn.edu.ar (Corsetti, Ornela Milagros), ggolzman@frro.utn.edu.ar (Golzman, Gabriel), guadabengoechea@gmail.com (Bengoechea, Guadalupe María)

1. Marco Teórico

1.1. Definición

En la *Teoría de Colas*, una disciplina dentro de la *Teoría matemática de la Probabilidad*, una cola M|M|c, representa la longitud de la cola en un sistema que posee varios servidorer, donde los arribos están determinados por un proceso de *Poisson* y los tiempos de servicio tienen una distribución exponencial.

1.2. Características

- Se tiene un sistema de llegadas que se producen según un proceso de Poisson de razón λ, donde los tiempos entre llegadas estarán distribuidos exponencialmente Exp(λ) o donde λ es el número medio de llegadas por unidad de tiempo.
- Los tiempos entre servicios son distribuidos de manera exponencial, Exp(μ) o donde μ es el número medio de paquetes que el servidor es capaz de atender por unidad de tiempo.
- El número de servidores en el sistema se denotará con la constante c.
- La capacidad del sistema puede o no ser infinita.
- La disciplina del sistema sera tanto FIFO como LIFO y un sistema de prioridades (en la primera cola del sistema, en las consecuentes 3 se aplicará solo FIFO)

1.3. Medidas de Rendimiento

1.3.1. Número promedio de clientes en Cola

- Representa el tamaño promedio de la cola a lo largo de toda la simulación.
- Se calcula a partir del área bajo $Q(t)$ dividido por el tiempo final de la simulación o de la corrida.
- $Q(t)$ es una función que indica la cantidad de clientes en cola en el tiempo t.
- $Q(t)$ se implementa en el algoritmo como una variable que se incrementa en 1 cuando entra un cliente en cola y se disminuye en 1 cuando se quita un cliente en cola.
- Esta variable evalúa el sistema desde el punto de vista de la atención al cliente.

$$L_q = \frac{\rho^{c+1}}{(c-1)!} * \frac{1}{c-\rho^2} * P_0$$

1.3.2. Utilización promedio del servidor.

- Se calcula como la sumatoria de los tiempos de servicio incurridos en atención de clientes, dividido por el tiempo final de la simulación corrida.
- Se calcula para cada servidor en el sistema.
- Se puede ver como el área bajo $B(t)$ dividida por el tiempo de simulación.
- $B(t)$ es una función que asume el valor 1 desde el instante en que el servidor esta ocupado y 0 cuando esta desocupado.
- Esta variable evalúa el sistema desde el punto de vista de maximizar la utilización de los recursos sobre los que se ha invertido.

$$U = 1 - \left[P_0 + \frac{c-1}{c} * P_1 + \frac{c-2}{c} * P_2 + ... + \frac{1}{c} * P_{c-1} \right]$$

1.3.3. Tiempo promedio de servicio

Refiere al tiempo promedio desde que el cliente ingresa a la primer cola (cola 1) hasta que sale de alguno de los servidores finales (servidor 4, 5 o 6).
Se calcula como la sumatoria de los tiempos en servicio individuales sobre la duración de la simulación.

$$TPS = \frac{\sum TS\,i}{TS\,total}$$

2. Metodología

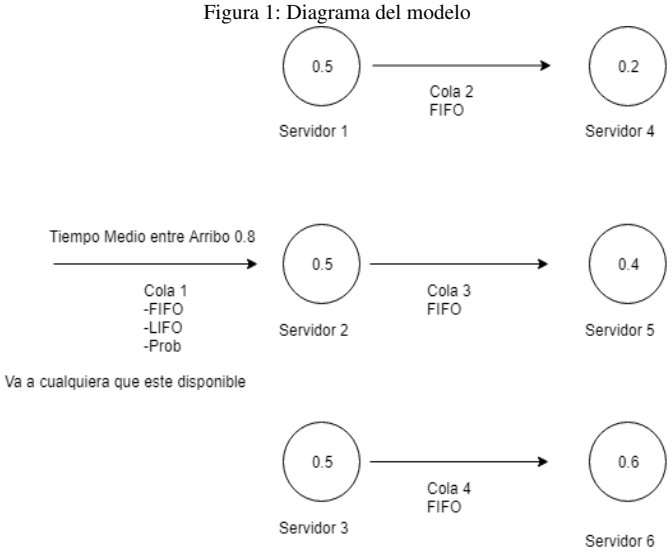
2.1. Modelado

Nuestro modelo de simulación esta compuesto por una cola inicial (denominada *cola 1*), la cual posee una tasa media de arribos de 0,8 que dirige a tres servidores, los cuales poseen todos una tasa de servicio de 0,5.

De estos tres servidores, sale una cola (denominada *cola 2, cola 3* y *cola 4* que se dirige cada una a un servidor distinto, habiendo entonces tres servidores más al final del modelo, con la particularidad que estos poseen una tasa de servicio distinta (siendo estas 0,2, 0,4, y 0,6 respectivamente).

En una etapa inicial, en todas las colas el algoritmo de asignación de servicio es regido por FIFO (*first in, first out*), la asignación de la primer columna de servidores se realiza de forma aleatoria uniformemente, y la asignación de la segunda columna de servidores depende exclusivamente del servidor por el cual obtuvo servicio el cliente.

Figura 1: Diagrama del modelo



2.2. *Planteamiento*

Consideramos programar la simulación de manera orientada a objetos, de forma de que se nos simplifique el manejo de tiempos de manera de tener un seguimiento más sencillo de los mismos durante la programación de los distintos métodos.

2.2.1. *Clase Simulación*

La clase simulación es la clase que representa a cada una de las simulaciones (que comienza con el método *Run*) y luego efectúa el método *análisis* que analiza las medidas de rendimiento para esas corridas.

Los métodos invocados por la clase Simulación son los métodos convencionales del M|M|1, además de métodos extras para los tipos de algoritmos de prioridad de colas y el método reporte, que hace los cálculos de las medidas de rendimiento.

En lo que respecta de sus atributos, la clase tiene una colección con los seis instancias de Servidor con su respectiva tasa y una colección con los cuatro instancias de clase Cola.

Además, el atributo reloj , la colección bidimensional *ProxEvento*,que es donde guardaremos el tiempo de arribo y tiempo de partida menores y una colección Cola.Arribo, donde iremos guardando los arribos de los clientes.

2.2.2. *Clase Servidor*

La clase Servidor tendrá como atributos su tasa correspondiente,los acumuladores utilización promedio del servidor y área de $B(t)$, el tiempo de entrada al servidor y la instancia del cliente que se puede llegar a encontrar en el servidor.

2.2.3. *Clase Cola*

Se declaran los acumuladores Número promedio de clientes en cola y área de $Q(t)$, además de el valor del Último Evento y una colección, llamada cola de espera, que contendrá las instancias de clientes que estén esperando a obtener servicio.

2.3. *Métodos*

2.3.1. *Run*

Este método se encarga de invocar al método Inicialización, y luego entra a un bucle que correrá el método tiempos siempre y cuando el reloj no supere el valor de 10000, caso contrario, sale del bucle e invoca el método Reportes.

2.3.2. *Inicialización*

Inicializa el arreglo de próximos eventos, generando un arribo con el calculo exponencial con la tasa media de arribos y asignándole un valor muy alto (99999, por ejemplo)a el tiempo de servicio para lograr forzar que el primer evento sea un **arribo**.

2.3.3. *Tiempos*

Este método analiza cual es el siguiente evento al cual se le va a asignar el reloj (comparando cual de los dos tiempos es menor).

En el caso de que sea un arribo, invoco el método arribo enviándole como parámetro su cola correspondiente. En el caso de que sea una partida, invoco el método partida con el servidor que le corresponde.

2.3.4. *Arribo*

Mediante el método Evaluar.Servidor que toma como parámetro la cola, puedo obtener a que servidor le corresponde ese arribo. En el caso de que no allá servidores, devuelvo una bandera para notificar eso. Sino, devuelvo el número de servidor correspondiente.

Si el servidor esta ocupado:

- Calculo el área $Q(t)$ como especifica la sección 1.3.1.
- Le asigno al atributo Ultimo. Evento de la cola el reloj actual.
- En el caso de que la cola es la 1, quito el tiempo de arribo correspondiente de la colección .ªrribos".
- Por último, agrego a la cola de espera la instancia del Cliente que produjo ese arribo.

Si el servidor no esta ocupado:

- Genero la próxima partida de la instancia Cliente.
- Agrego cliente al servidor.
- Le asigno al tiempo de entrada del servidor el valor actual del reloj.
- Si la cola es la primera, también le quito el tiempo de arribos de la colección de arribos para ese cliente.
  También si la cola es la primera, nos encargamos de generar un nuevo arribo y guardarlo en dicha colección.

- Se invoca la función *BuscaMin()* que se encarga de analizar cual es la próxima partida con menor tiempo y devuelve el servidor correspondiente.
- Se genera la próxima partida del sistema, guardando los valores en la colección *ProxEvento*.

2.3.5. Partida

Recibe por parámetro el servidor donde se esta generando la partida y la instancia del cliente que se encontraba en el mismo. Inicialmente evaluamos si la partida proviene de la primera linea de servidores, puede ser del *Servidor 1*, *Servidor 2* o *Servidor 3*. De ser así, evalúo a que cola irá el cliente que acá de salir del servidor con cualquiera de estos dos métodos:

- **Evaluar Cola Sig:** Este método retorna el número de la cola a la que irá el cliente según el criterio de que una partida en cada servidor genera un arribo en su cola inmediata, en al caso del Servidor 1 será la cola 2, Servidor 2 la cola 3 y servidor 3 la cola 4. Como se puede observar la cola siempre será *servidor + 1*, por lo tanto toma como parámetro el servidor de que se genera la partida y lo envía a la cola especificada.
- **Evaluar Cola Mej:** Esta es una de las mejoras que aplicaremos al algoritmo, consiste en evaluar primero si alguno de los servidores está disponible y en caso de estarlo mas de uno, prioriza el de menor tiempo de tasa de servicio.
  En caso de que ninguno este disponible, envía al cliente a la cola que posea menos gente esperando.
  Para realizar esto el método controla los estados de cada servidor y deriva al disponible de estarlo alguno. De lo contrario, calcula los largos en las listas de espera de cada cola y retorna el numero de cola que posea el menor largo.

Una vez obtenida la cola a la que se deriva al cliente se invoca el arribo correspondiente.

En el caso de estar en la segunda linea de servidores, simplemente debemos calcular el tiempo demora total en la simulación de dicho cliente, con el tiempo de partida en cuestión y el tiempo de ingreso del cliente almacenado al momento de generarlo. Dicho tiempo se almacena en una lista que lleva el control de todos los tiempos de la simulación.

Una vez terminada esta bifurcación, se calcula el *areaBdeT* con el tiempo de entrada del cliente al servidor y el reloj actual (tiempo de partida), esto se acumula en la instancia del servidor correspondiente.

Posteriormente se evalúa con el método *Evaluar.Cola* la cola de la que ingresan los clientes a dicho servidor. Con el valor de la cola se analiza el largo de la cola de espera en cuestión. De aquí hay dos posibles situaciones:

- **La cola se encuentra vacía:** En este caso se vacía el servidor asignando en el campo del cliente el valor *None*.
- **La cola tiene gente esperando:** En dicho caso, primero se evalúa a que cliente de los que se encuentran en la cola se le dará servicio utilizando algunos de los algoritmos de selección codificados en los métodos *FIFO*, *LIFO* o *Prioridad*. Los cuales devuelven la posición del primer, último o cliente con prioridad(de haberlo) respectivamente y serán explicados en detalle en la seccion 2.3.6.
  Una se que se posee la posición del cliente en cuestión, hay que generar su ingreso al servidor. Para esto, lo asignamos al servidor, calculamos su partida y guardamos el tiempo de entrada al servidor (reloj actual).
  Finalizado esto se calcula el *areaQdeT* de la cola en cuestión, se sobre escribe con el reloj el último momento en el que se calculó la misma para dicha cola y se quita al cliente que ingreso al servidor de la cola de espera. Todo se realiza en el orden descripto para calcular correctamente las medidas.

Ya finalizado esto, debemos controlar que la partida generada (en caso de haberlo hecho) no tenga un tiempo menor que la partida que se encuentra actualmente en *Prox.Evento* o que no se hayan vaciado todos los servidores del sistema.
Para esto volvemos a invocar *BuscaMin*, según los valores que retorne, vamos a asignarle el cliente con el mínimo tiempo de partida y su respectivo servidor a *Prox.Evento* o forzamos un arribo generando un cliente con tiempo 99999 en caso de devolver la bandera que indica que todos los servidores están vacíos.

2.3.6. Algoritmos de Selección de Clientes en Cola

Hay 3 posibilidades para seleccionar los clientes en cola en la *Cola 1*. Estos son:

- **FIFO:** Consiste en devolver la posición del primer cliente que llegó a la cola, en este caso será siempre 0.
- **LIFO:** Consiste en devolver la posición del último cliente que llegó a la cola, se le envía por parámetro la cola en cuestión y retorna el largo de la cola - 1.
- **Prioridad:** Utiliza los valores random generados para cada cliente cuando ingresa al sistema y comprueba si alguno de ellos es menor que 0.03 (3 % de la población) devuelve el que menor prioridad tenga, y de lo contrario devuelve la posición 0 (FIFO).

Todos estos algoritmos se aplican solo en el caso que provengan de la *Cola 1*, de lo contrario devuelve un 0, ya que el de selección de las colas 2, 3 y 4 es FIFO.

2.3.7. Reporte, Analítica y Graficar.

En este método, inicialmente, se calculan las medidas de rendimiento propias de la simulación, estas son:

- **Numero Promedio de Clientes en Cola (NPCC):** Se define de la siguiente manera:

$$\frac{areaQdeT}{reloj}$$

Este valor se agrega a una lista donde se mantienen las medidas extraídas de todas las simulaciones para cada cola por separado. Posteriormente, se calcula el NPCC promedio, que utiliza todos los valores capturados hasta el momento y saca el promedio de los mismos para cada cola.

- **Utilización Promedio de Servidores (UPS):** Se calcula de la siguiente manera:

$$\frac{areaBdeT}{reloj}$$

Este valor se agrega a una lista donde se mantienen las medidas extraídas de todas las simulaciones para cada servidor por separado. Posteriormente, se calcula el UPS promedio, que utiliza todos los valores capturados hasta el momento y saca el promedio de los mismos para cada Servidor.

- **Tiempo Promedio de Simulación (TPS):** Se calcula de la siguiente manera:

$$\frac{\sum_{i=1}^{NroClientes} TiempoSimu_i}{NroClientes}$$

Este valor se agrega a una lista donde se mantienen las medidas extraídas de cada simulación. Posteriormente, se calcula el TPS promedio, que utiliza todos los valores capturados hasta el momento y saca el promedio de estos.

Todos los promedios se calculan con la función *sum* y *len* y se guardan en una variable global que posteriormente se aplica en el método *Analítica*, el cual se encarga de invocar al método *Graficar* que lo que hace es utilizar la librería *MatPlotLib* de Python para graficar los promedios de los datos (enviados como parámetro) para ver gráficamente como estos convergen a un valor.

2.4. *Hipótesis*

Nuestra hipótesis radica en la afirmación que, debido al uso de las fórmulas de número de promedio de clientes en cola enunciada en 1.3.1, el valor estacionario que debería tender en la cola 1 es de : **0.007**.

Observando este valor de cantidad de clientes que va a haber en la cola 1, podemos afirmar que la aplicación de algoritmos distintos de selección de clientes no generará grandes cambios a los valores finales estacionarios. Esto sucede ya que al no encontrarse gente en cola en la mayoría de la simulación, los distintos algoritmos de selección de clientes en cola se igualan

También deducimos que la utilización promedio de servicio de los 3 primeros servidores tendrá valores similares entre sí, el cual sera **0.2**, calculado utilizando las fórmulas de la sección 1.3.2, ya que poseen iguales tasas y son seleccionados de manera uniforme y aleatoria. Pero en cambio, los últimos 3 servidores tendrán una utilización promedio dispar, debido a sus diferentes tasas.

La mejora del algoritmo de asignación de cola para la segunda fila de servidores debería bajar el tiempo promedio de servicio, ya que al seleccionar primero la cola con el servidor disponible de menor tiempo y de estar todo ocupado la cola que menos clientes presente, mejora la medida de rendimiento del sistema modelado.

Al ejecutar la simulación 1000 veces y analizar los valores promedios acumulados en cada una de las corridas, deberíamos poder notar como los valores promedios convergen a un estado estacionario en todas las medidas extraídas. En el caso de la *NPCC Cola 1* y *UPS 1, 2 Y 3* deberán tender a los valores calculados anteriormente.
En cuanto a las demás colas y servidores, no tenemos cálculos exactos de los valores a los cuales convergen, sin embargo deberíamos poder notar el comportamiento descripto en cada una de ellas.

3. Casos de estudio

3.1. *Cola inicial con Algoritmo FIFO*

3.1.1. *Sin aplicación de mejora en la segunda columna de colas*

El comportamiento de la simulación en este caso de estudio se ve reflejado en las siguientes gráficas. Los resultados graficados son los promedios de promedios calculados en cada una de la 1000 corridas.

Figura 2: Número promedio de clientes en cola para la primera cola

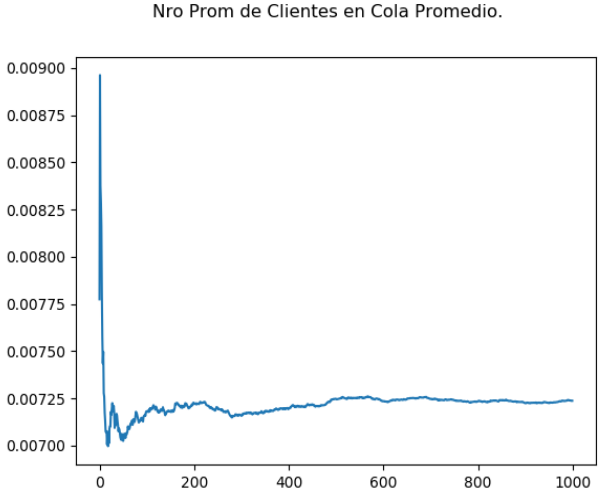


Figura 3: Número promedio de clientes en cola para la segunda cola

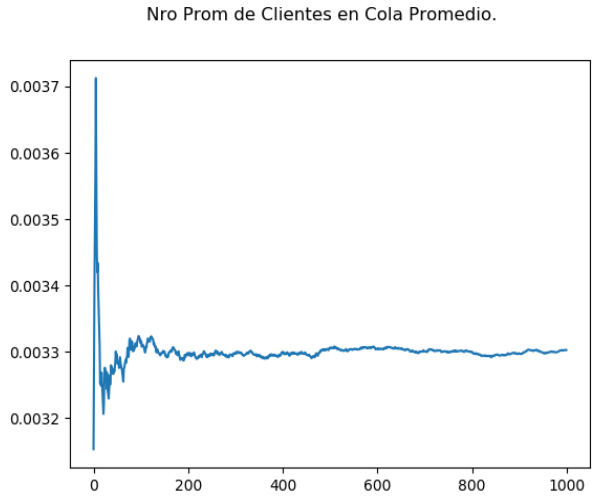


Figura 4: Número promedio de clientes en cola para la tercera cola

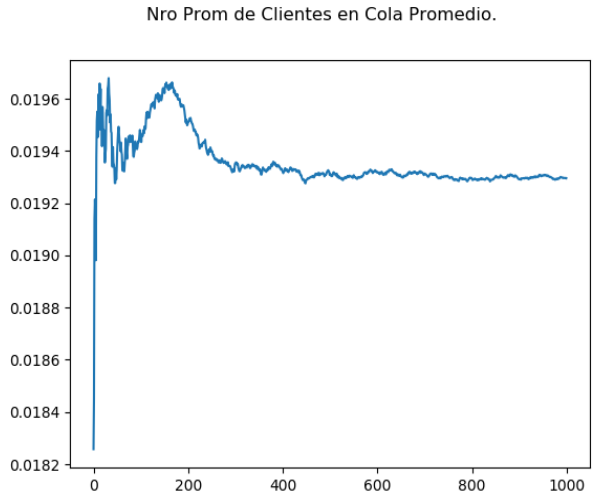


Figura 5: Número promedio de clientes en cola para la cuarta cola con algoritmo FIFO

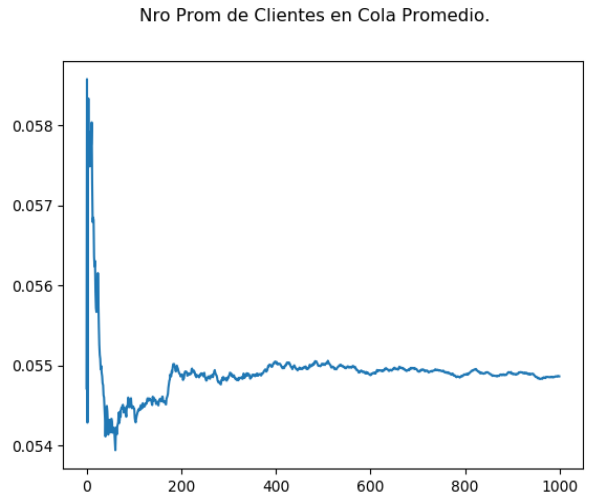


Figura 6: Utilización promedio del servidor 1 con tasa 0.5

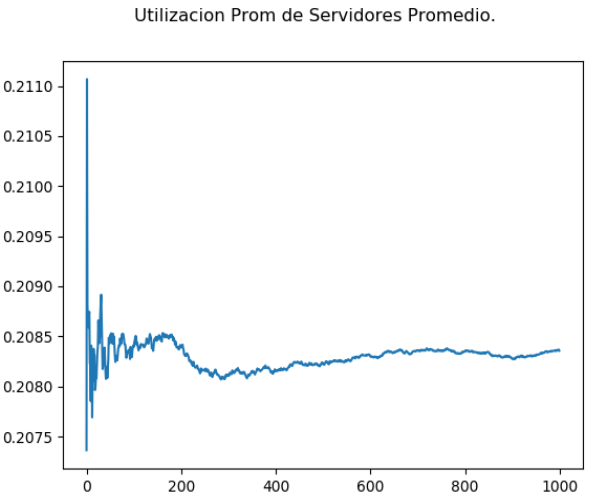


Figura 7: Utilización promedio del servidor 2 con tasa 0.5

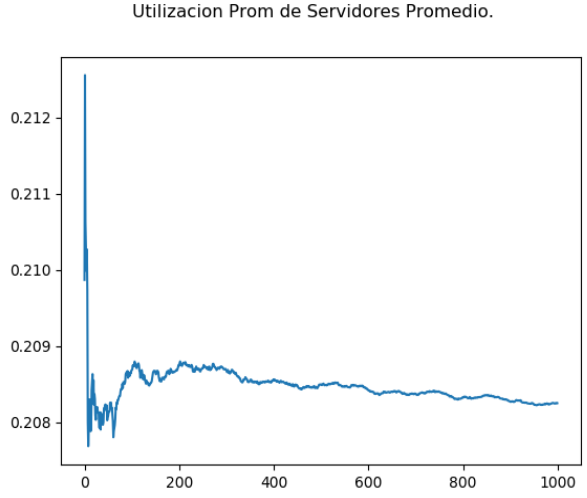


Figura 8: Utilización promedio del servidor 3 con tasa 0.5

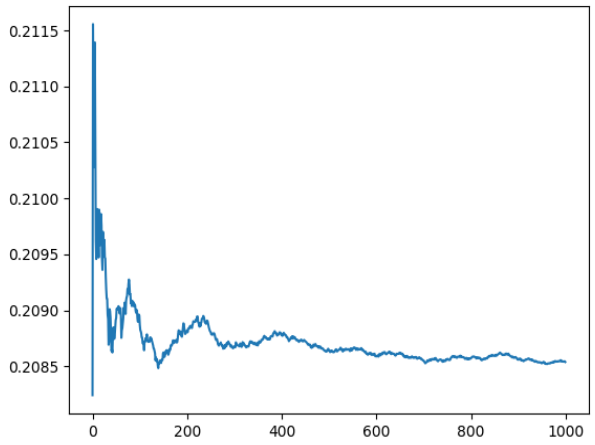


Figura 9: Utilización promedio del servidor 4 con tasa 0.2

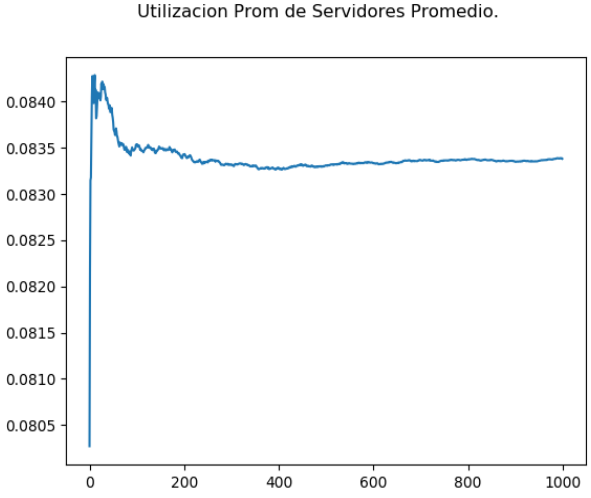


Figura 10: Utilización promedio del servidor 5 con tasa 0.4

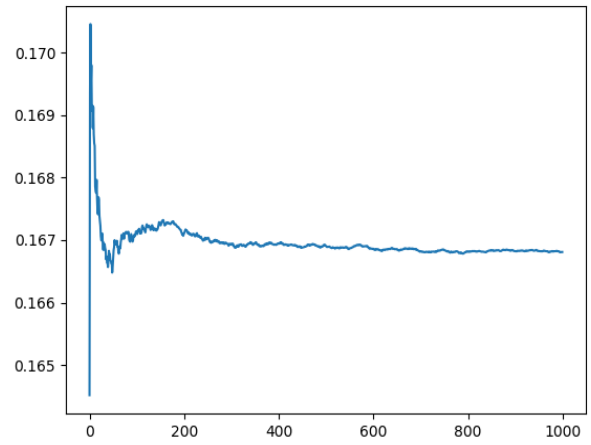


Figura 11: Utilización promedio del servidor 6 con tasa 0.6

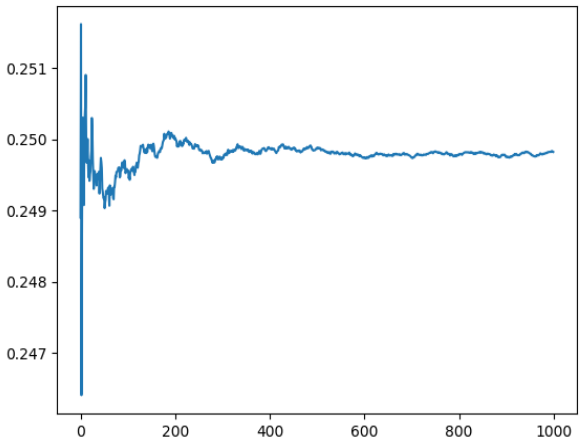


Figura 12: Tiempo promedio de servicio

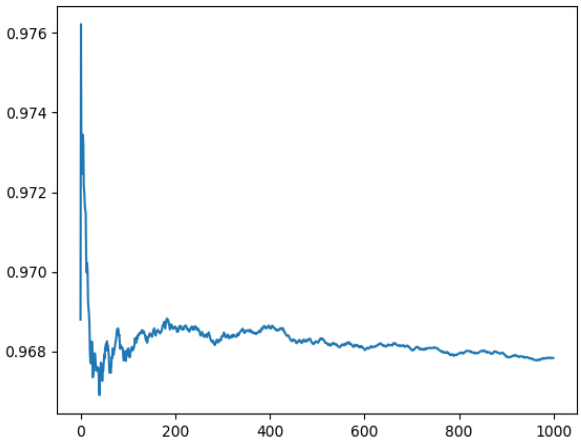


Figura 13: Numero promedio de clientes en cola 1

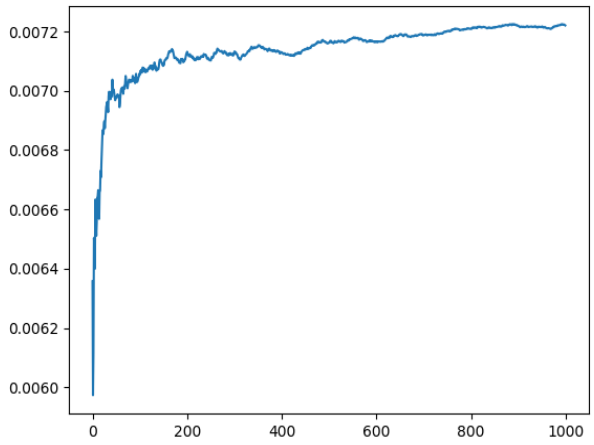
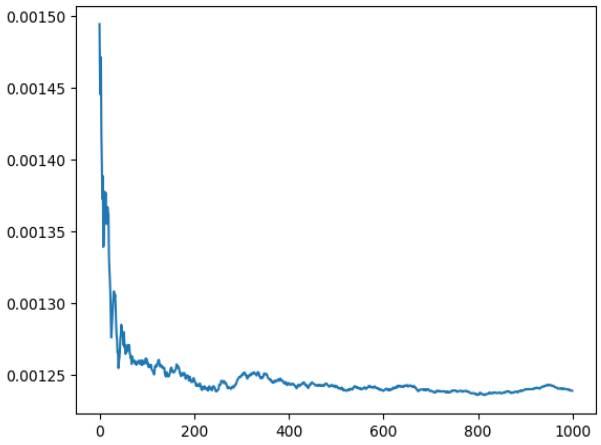


Figura 14: Numero promedio de clientes en cola 2



3.2. *Con aplicación de mejoras en la segunda columna de colas*

El comportamiento de la simulación en este caso de estudio se ve reflejado en las siguientes gráficas. Los resultados graficados son los promedios de promedios calculados en cada una de la 1000 corridas.

Figura 15: Numero promedio de clientes en cola 3

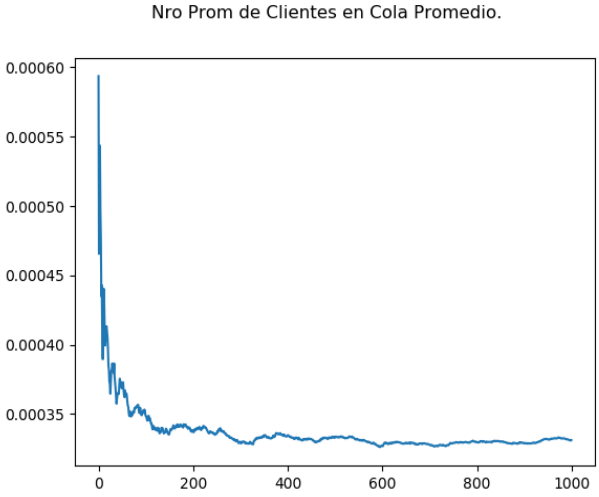


Figura 16: Numero promedio de clientes en cola 4

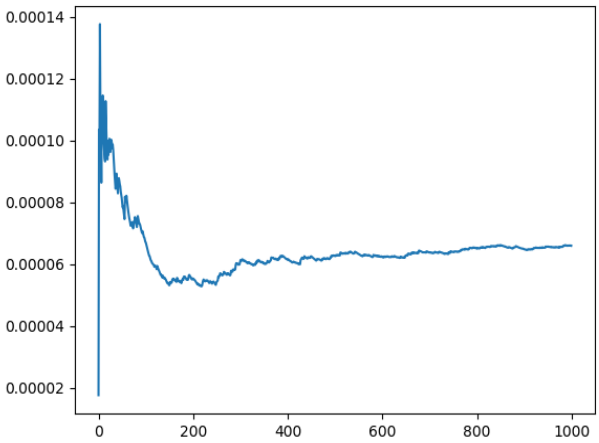


Figura 17: Utilización promedio del servidor 1 con tasa 0.5

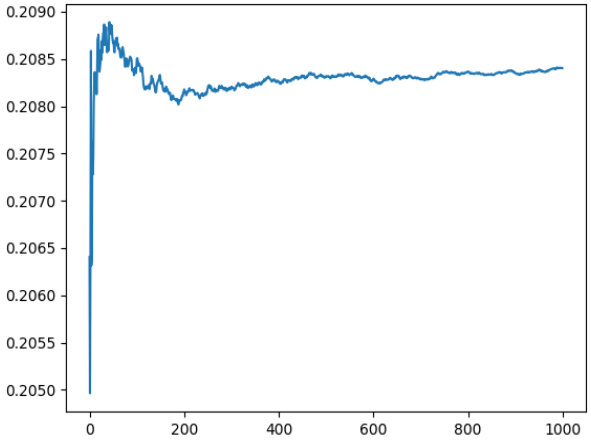


Figura 18: Utilización promedio del servidor 2 con tasa 0.5

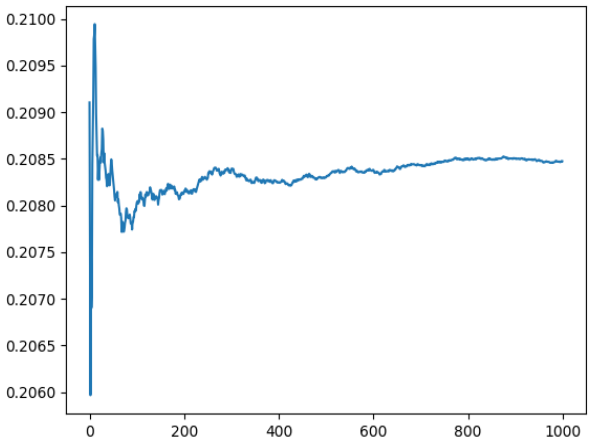


Figura 19: Utilización promedio del servidor 3 con tasa 0.5

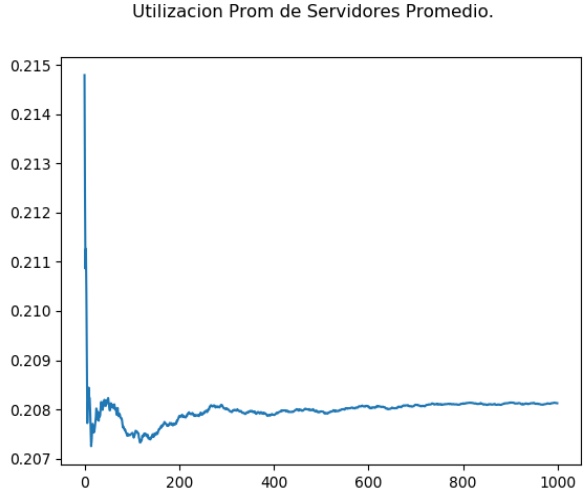


Figura 20: Utilización promedio del servidor 4 con tasa 0.2

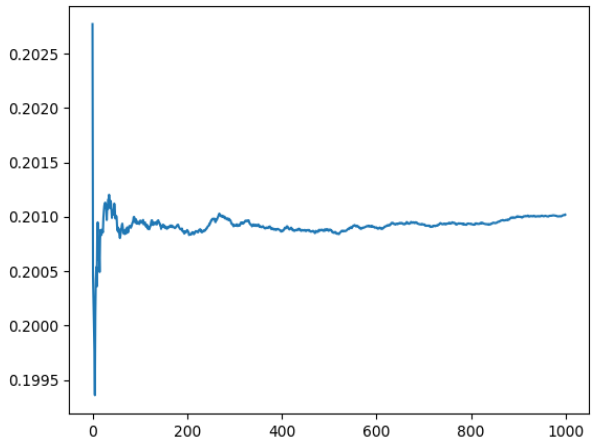


Figura 21: Utilización promedio del servidor 5 con tasa 0.4

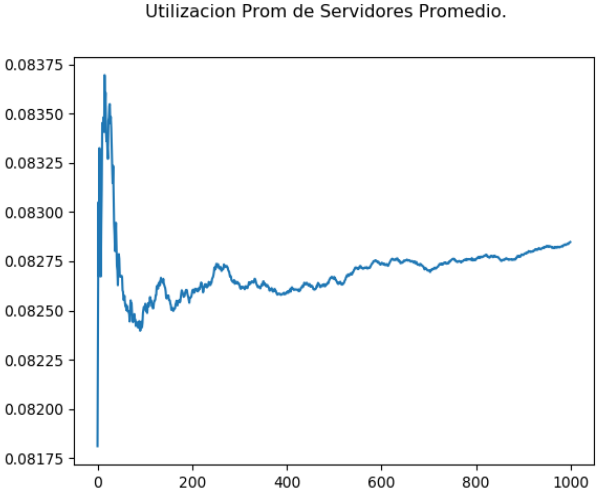


Figura 22: Utilización promedio del servidor 6 con tasa 0.6

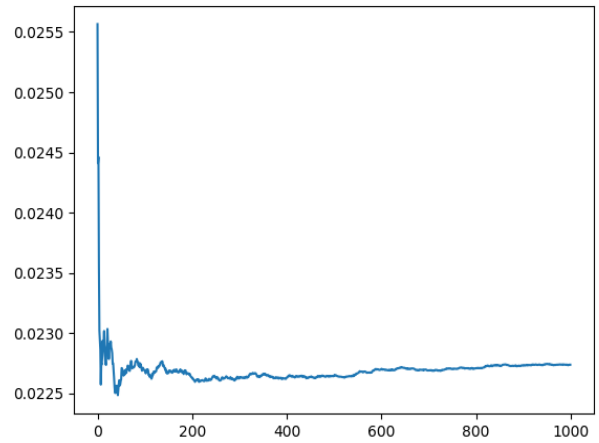


Figura 23: Tiempo promedio de servicio

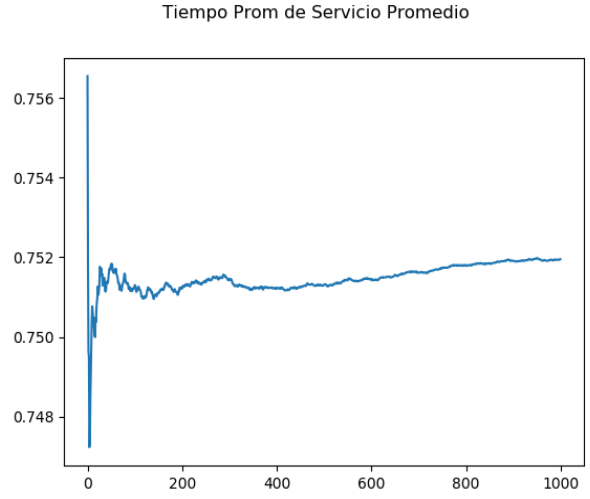
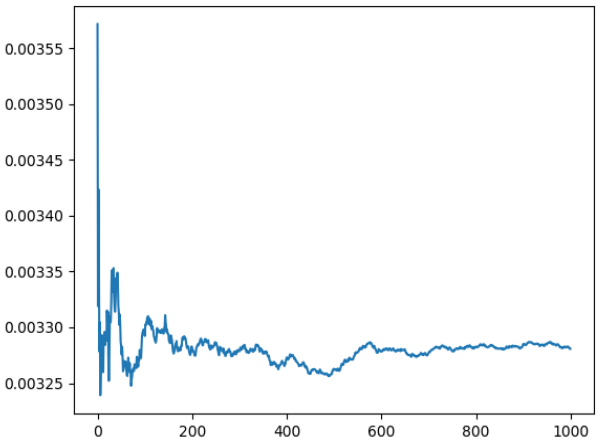


Figura 25: Número promedio de clientes en cola para la segunda cola



3.3. *Cola inicial con Algoritmo LIFO*

3.3.1. *Sin aplicación de mejora en la segunda columna de colas*

El comportamiento de la simulación en este caso de estudio se ve reflejado en las siguientes gráficas. Los resultados graficados son los promedios de promedios calculados en cada una de la 1000 corridas.

Figura 24: Número promedio de clientes en cola para la primera cola

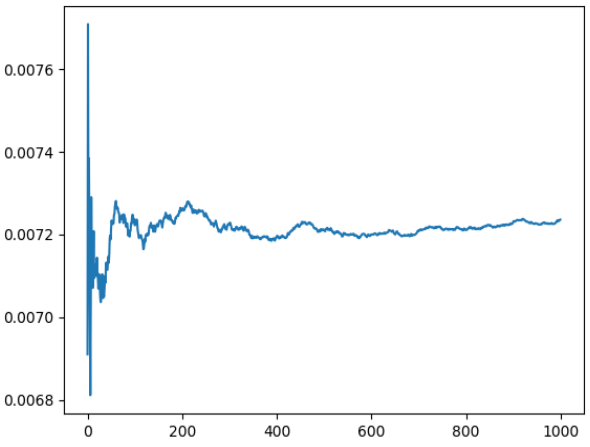


Figura 26: Número promedio de clientes en cola para la tercera cola

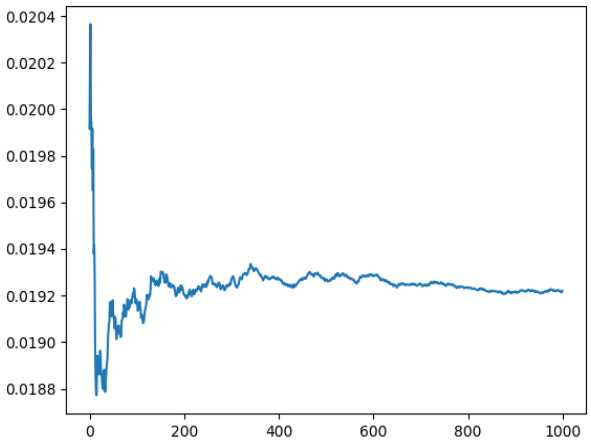


Figura 27: Número promedio de clientes en cola para la primera cola

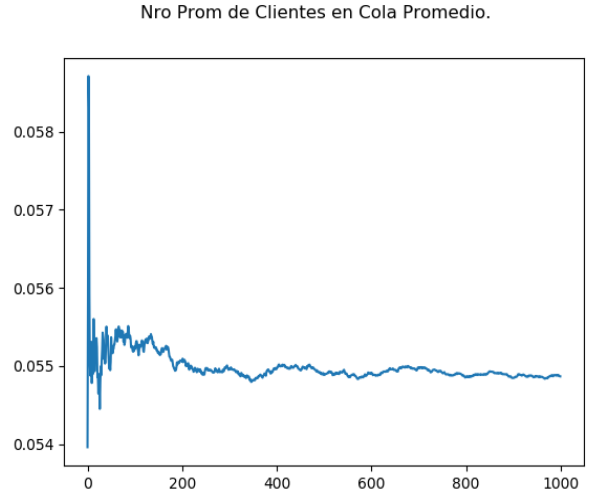


Figura 28: Utilización promedio del servidor 1 con tasa 0.5

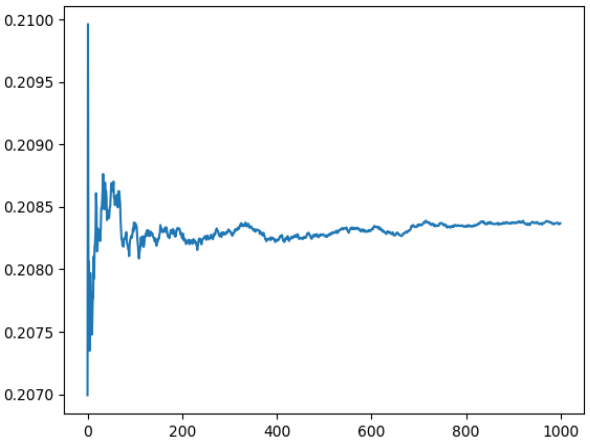


Figura 29: Utilización promedio del servidor 2 con tasa 0.5

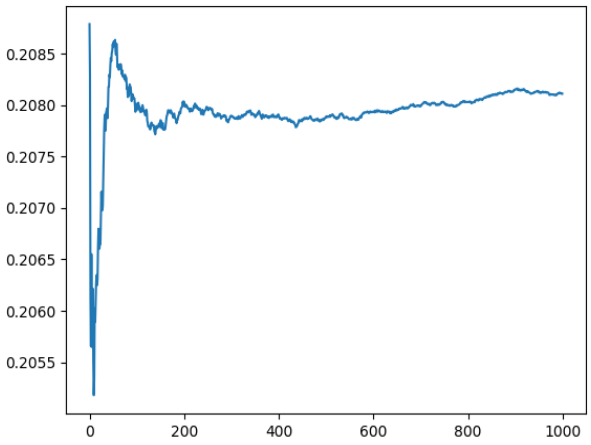


Figura 30: Utilización promedio del servidor 3 con tasa 0.5

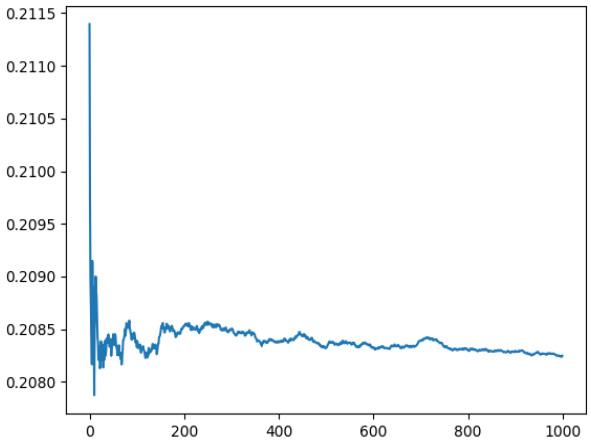


Figura 31: Utilización promedio del servidor 4 con tasa 0.2

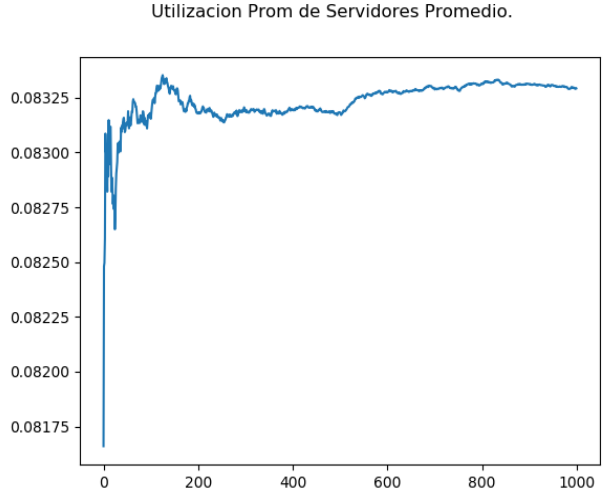


Figura 32: Utilización promedio del servidor 4 con tasa 0.4

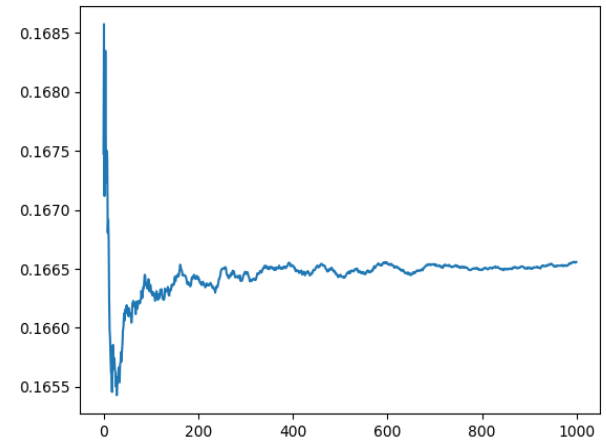


Figura 33: Utilización promedio del servidor 5 con tasa 0.6

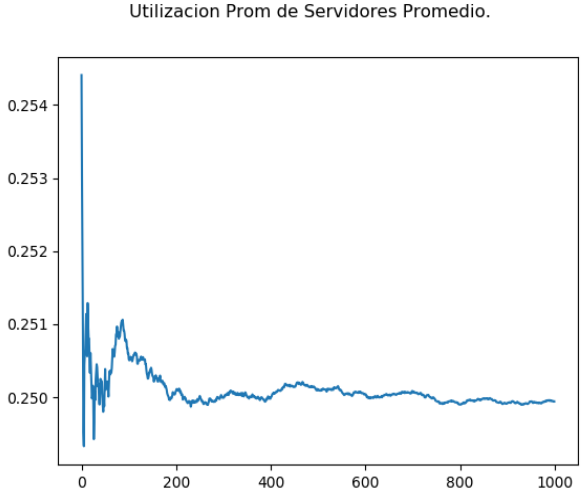
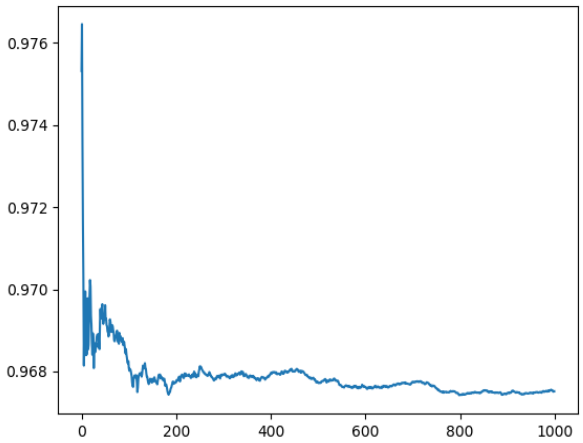


Figura 34: Tiempo promedio de servicio



3.4. Con aplicación de mejoras en la segunda columna de colas

El comportamiento de la simulación en este caso de estudio se ve reflejado en las siguientes gráficas. Los resultados graficados son los promedios de promedios calculados en cada una de la 1000 corridas.

Figura 35: Número promedio de clientes en cola para la primera cola

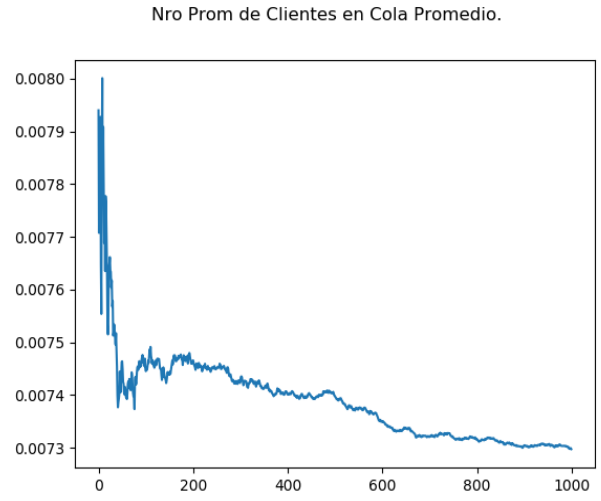


Figura 36: Número promedio de clientes en cola para la segunda cola

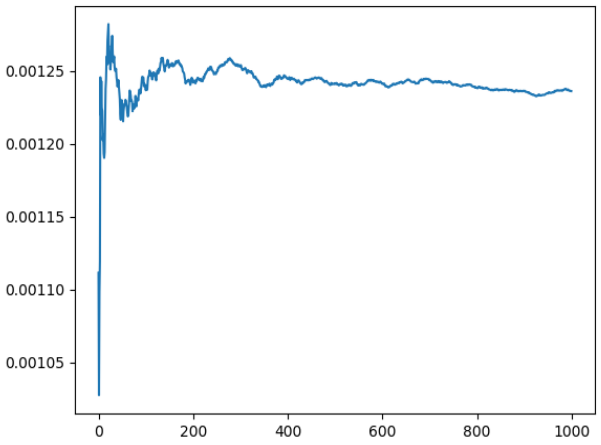


Figura 37: Número promedio de clientes en cola para la tercera cola

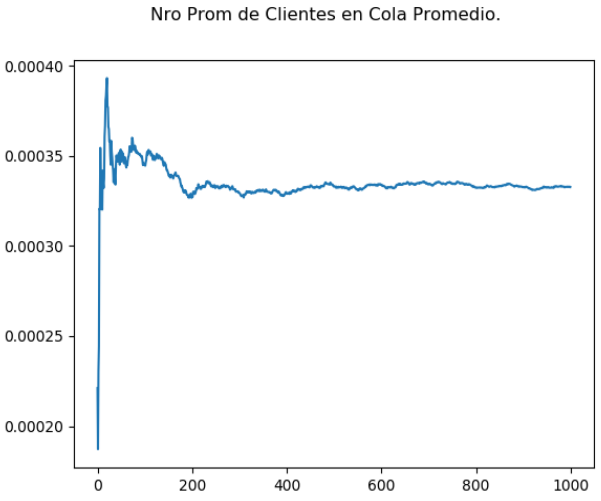


Figura 38: Número promedio de clientes en cola para la cuarta cola

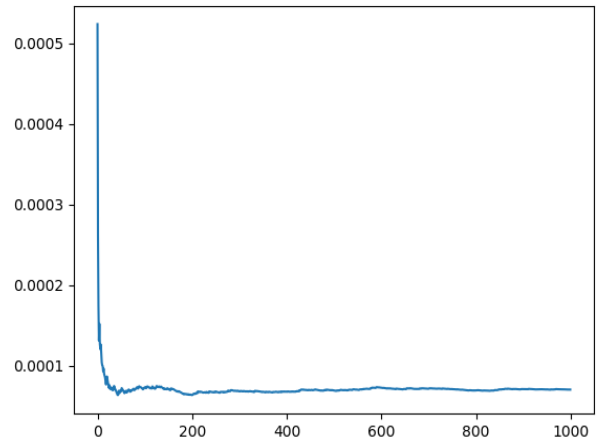


Figura 39: Utilización promedio del servidor 1 con tasa 0.5

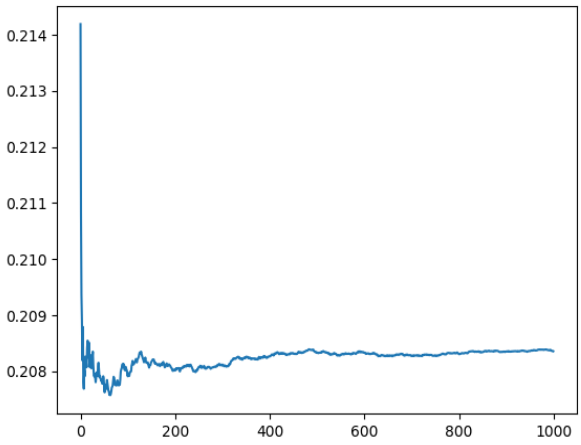


Figura 40: Utilización promedio del servidor 2 con tasa 0.5

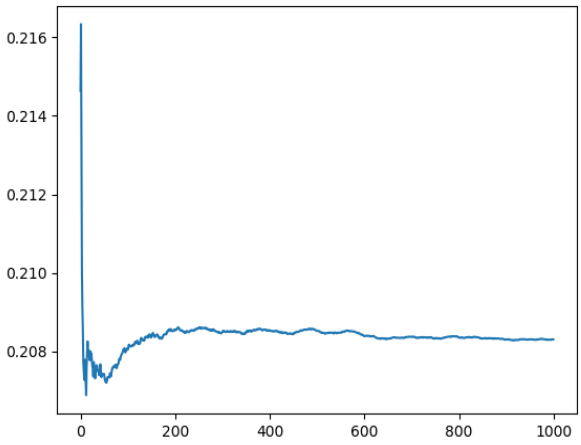


Figura 41: Utilización promedio del servidor 3 con tasa 0.5

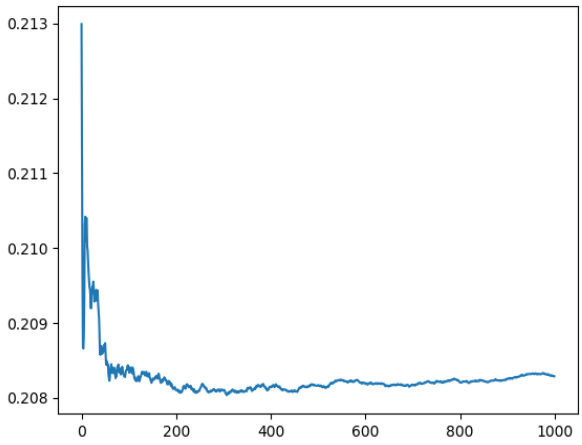


Figura 42: Utilización promedio del servidor 4 con tasa 0.2

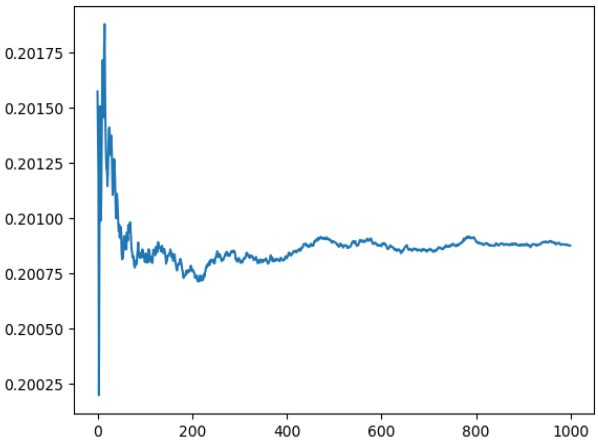


Figura 43: Utilización promedio del servidor 5 con tasa 0.4

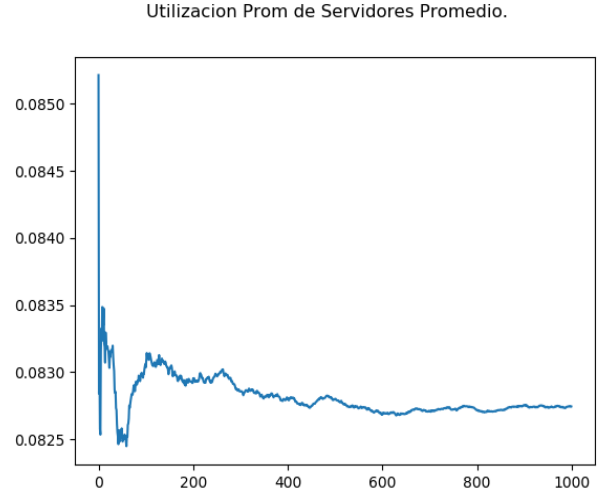


Figura 44: Utilización promedio del servidor 6 con tasa 0.6

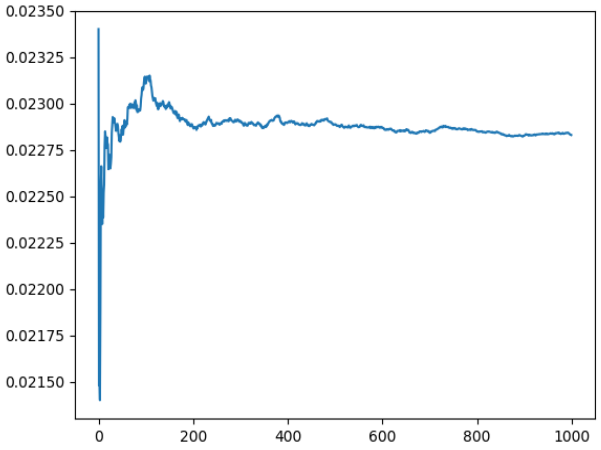
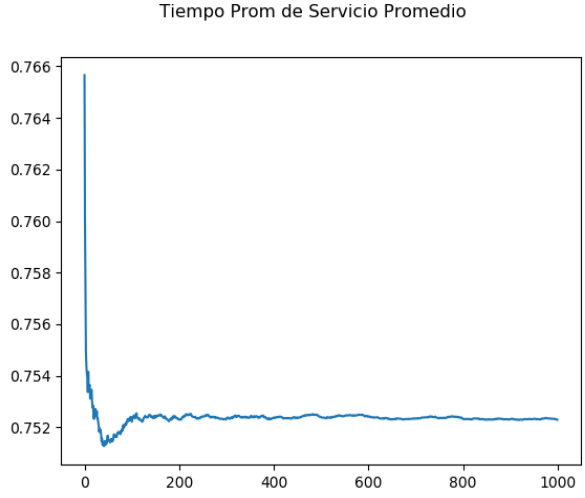


Figura 45: Tiempo promedio de servicio



3.5. *Cola inicial con Algoritmo Prioridades*

3.5.1. *Sin aplicación de mejora en la segunda columna de colas*

El comportamiento de la simulación en este caso de estudio se ve reflejado en las siguientes gráficas. Los resultados graficados son los promedios de promedios de calculados en cada una de la 1000 corridas.

Figura 46: Número promedio de clientes en cola para la primera cola

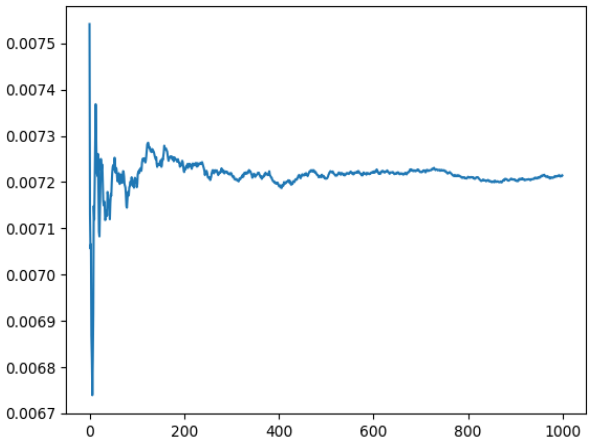


Figura 47: Número promedio de clientes en cola para la segunda cola

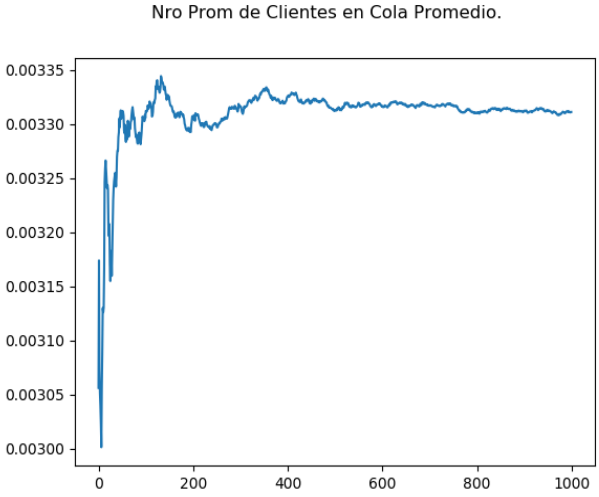


Figura 48: Número promedio de clientes en cola para la tercera cola

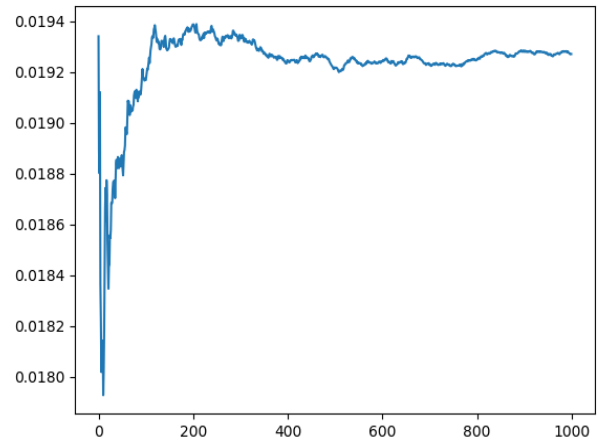


Figura 49: Número promedio de clientes en cola para la cuarta cola

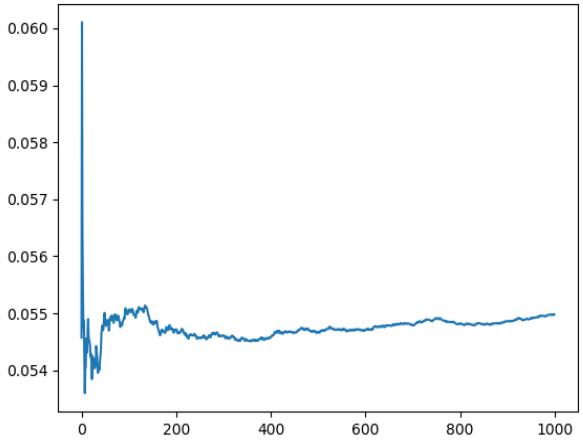


Figura 50: Utilización promedio del servidor 1 con tasa 0.5

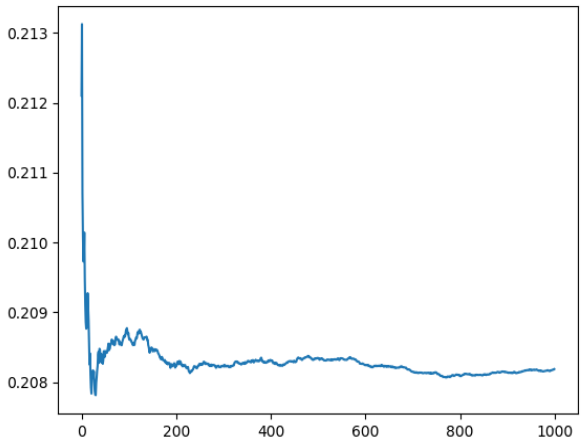


Figura 51: Utilización promedio del servidor 2 con tasa 0.5

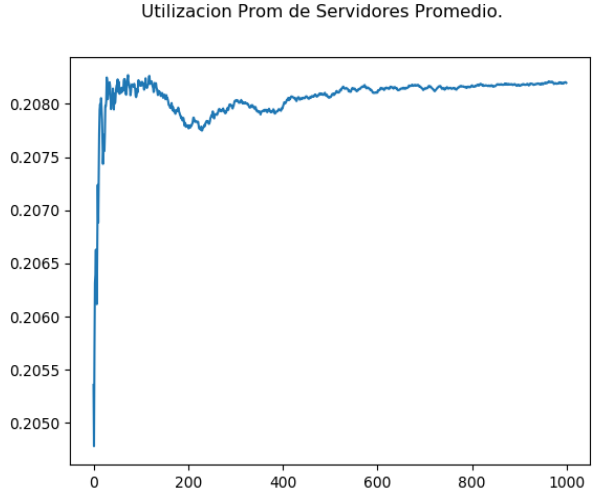


Figura 52: Utilización promedio del servidor 3 con tasa 0.5

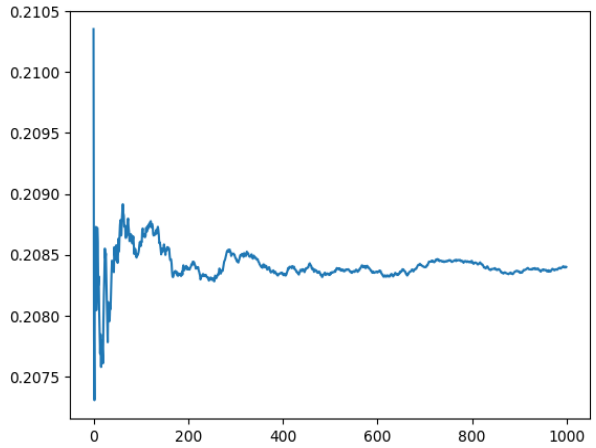


Figura 53: Utilización promedio del servidor 4 con tasa 0.2

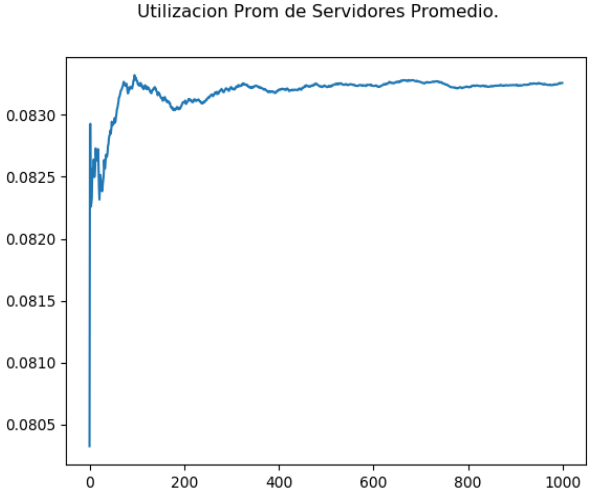


Figura 54: Utilización promedio del servidor 5 con tasa 0.4

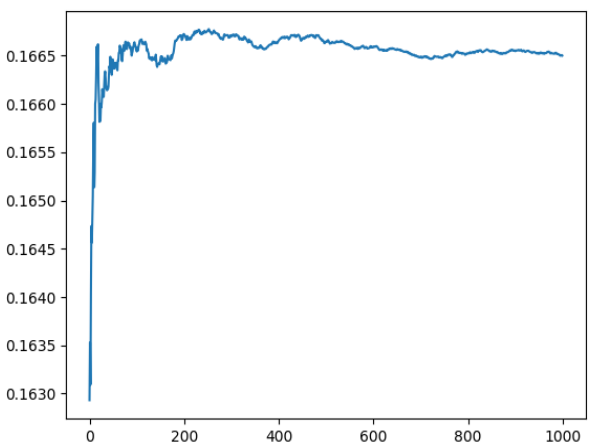


Figura 55: Utilización promedio del servidor 6 con tasa 0.6

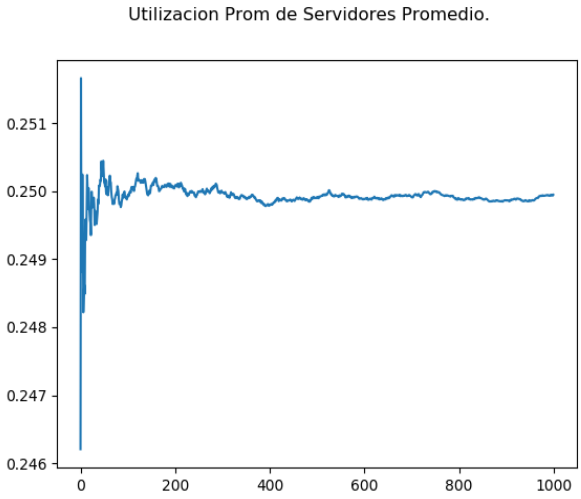


Figura 56: Tiempo promedio de servicio

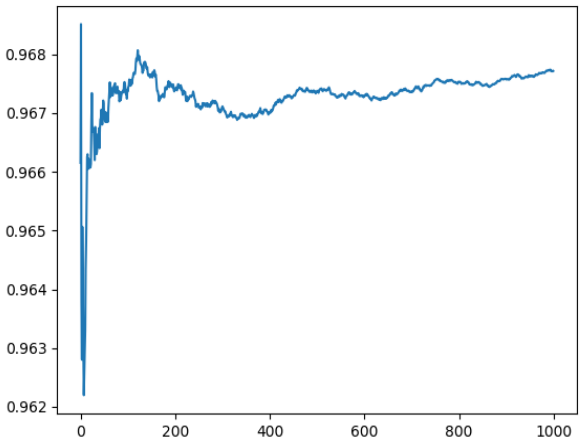


Figura 57: Número promedio de clientes en cola para la primera cola

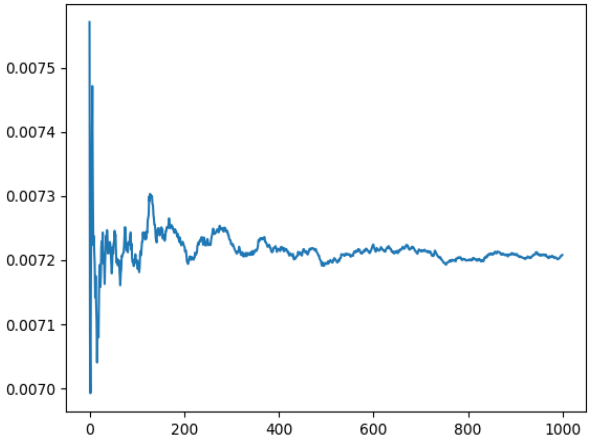
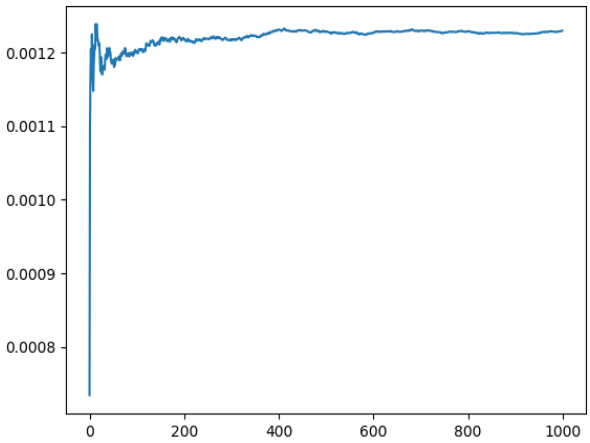


Figura 58: Número promedio de clientes en cola para la segunda cola



3.5.2. *Con aplicación de mejora en la segunda columna de colas*

El comportamiento de la simulación en este caso de estudio se reflejado en las siguientes gráficas. Los resultados graficados son los promedios de calculados en cada una de la 1000 corridas.

Figura 59: Número promedio de clientes en cola para la tercera cola

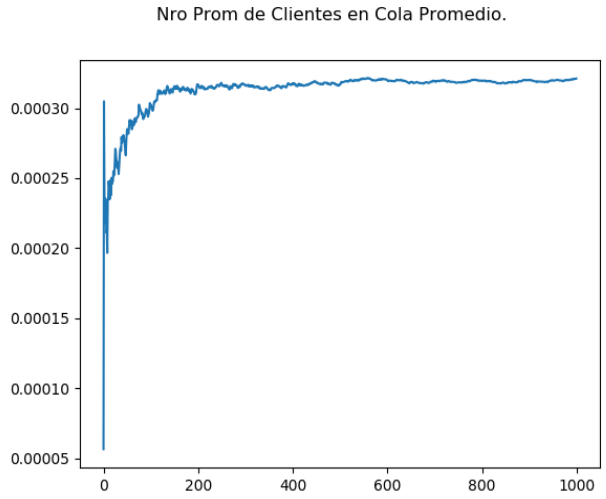


Figura 60: Número promedio de clientes en cola para la cuarta cola

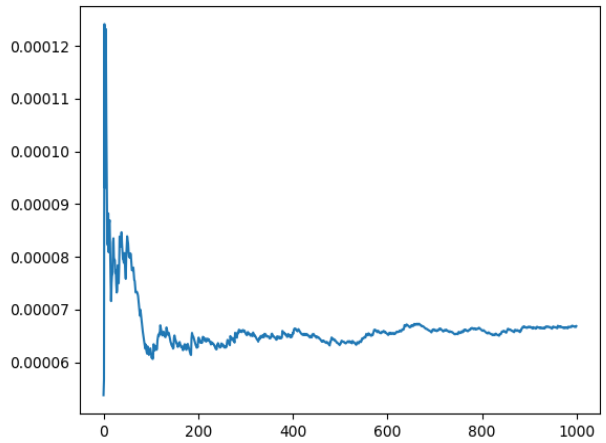


Figura 61: Utilización promedio del servidor 1 con tasa 0.5

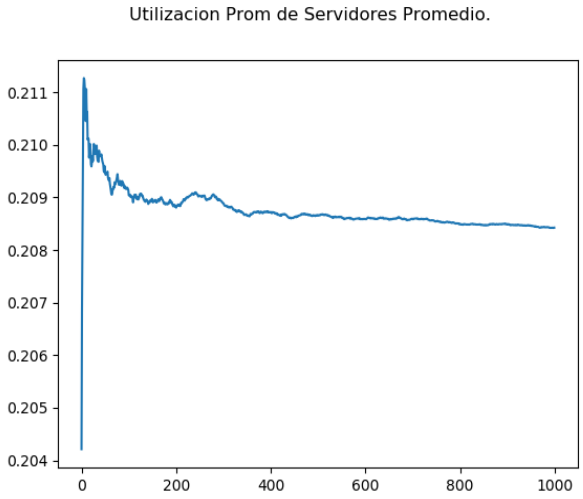


Figura 62: Utilización promedio del servidor 2 con tasa 0.5

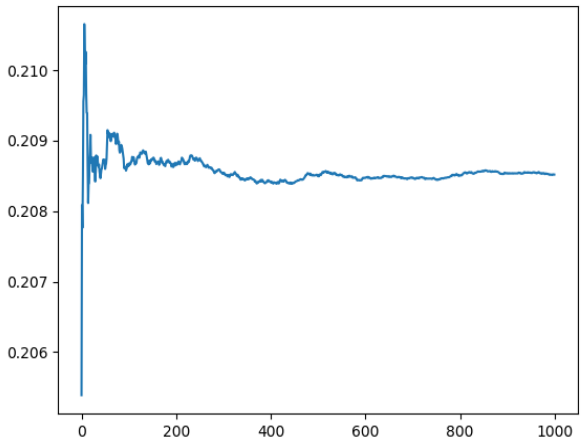


Figura 63: Utilización promedio del servidor 3 con tasa 0.5

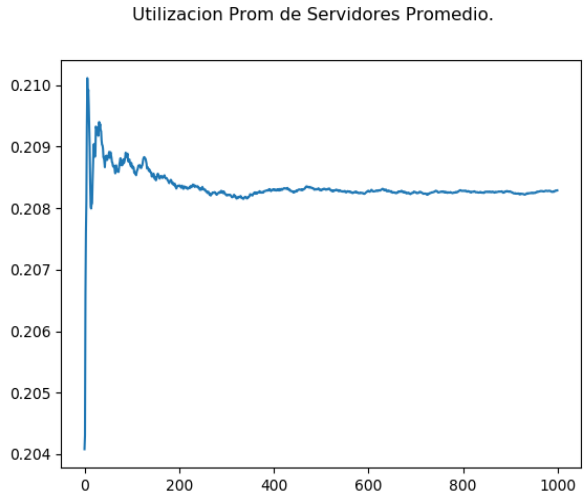


Figura 64: Utilización promedio del servidor 4 con tasa 0.2

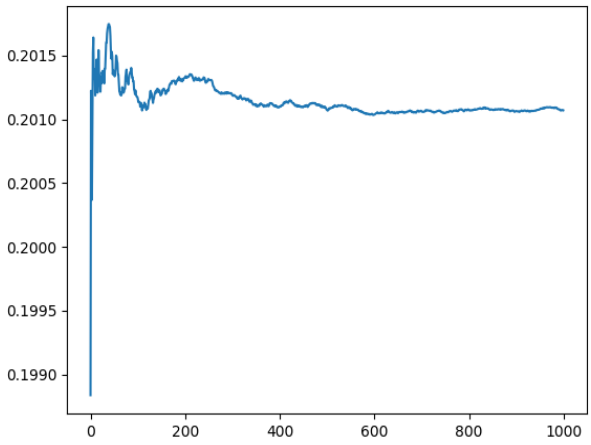


Figura 65: Utilización promedio del servidor 5 con tasa 0.4

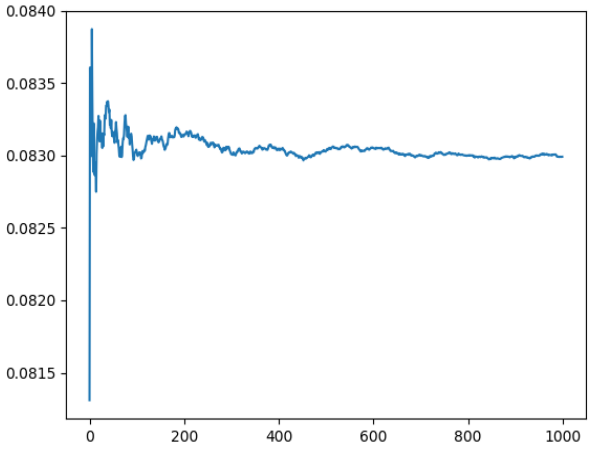


Figura 66: Utilización promedio del servidor 6 con tasa 0.6

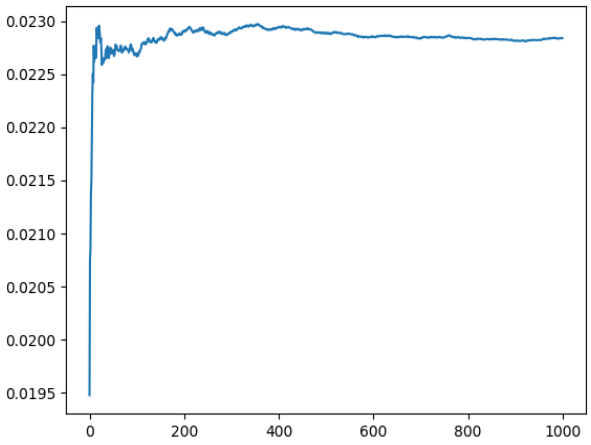
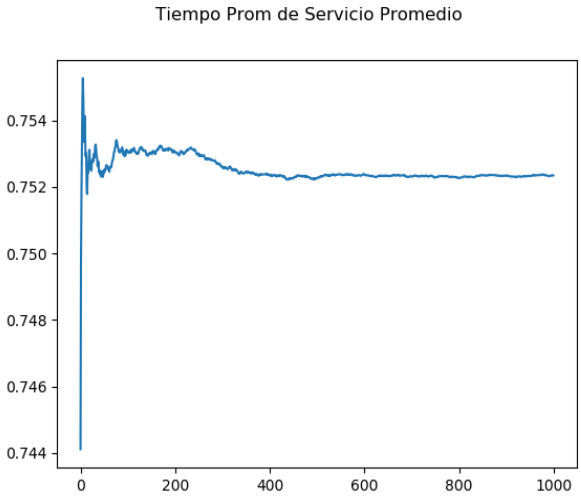


Figura 67: Tiempo promedio de servicio



4. Conclusiones

Las conclusiones elaboradas son las siguientes.

4.1. Comportamiento General de la Simulación.

Como podemos ver los valores de la gráfica convergen a 0,007 y se cumple la hipótesis ‘especificada para esta sección.
En cuanto a la primera linea de servidores al tener la mismas tasas y ser seleccionados de manera aleatoria uniformemente, podemos ver que los valores de utilización promedio de los tres convergen aproximadamente a los mismos valores.
En lo que respecta a la segunda linea de colas se puede notar que los valores de la primer cola son considerablemente mas chicos. Esto sucede porque es la cola que corresponde al servidor con la menor tasa de servicio, y por ende tendrá menos gente esperando a ser atendida.
Por lo tanto, la *NPCC* de la segunda cola será mayor que la primera y la de la tercera mayor que las otras dos.
La Utilización de la segunda linea de servidores también varía según las tasas de estos, siendo los que mayor tasa tengan los que más tiempo estarán utilizándose ya que demoran mas en dar el servicio.

4.2. Variación de Algoritmos de Selección en la primer cola.

El cambio de algoritmos de selección de clientes en la primer cola no presenta variaciones significativas en los resultados obtenidos.
Esto se debe a que la primer cola se mantiene casi todo el tiempo de la simulación vacía (*NPCC* 0,007) y por lo tanto al no haber gente en cola la forma de seleccionar los clientes de ella no influye.

4.3. Mejora Aplicada.

Al seleccionar no solo la cola que menos gente tenga (En este caso,la que esta mayor tiempo vacía es la primera), sino también, la que menos tasa de servicio tenga, se puede ver que el promedio de gente en cola es mayor para la primer cola que posee el servidor de 0,2 debido a que selecciona a la gente para pasar por dicha cola, provocando una baja en los promedios de las otras dos.
La utilización de los servidores varía de la misma manera, ya que al seleccionar continuamente la primera cola de la segunda linea se mantiene el cuarto servidor mucho mas ocupado que los demás.
Al trasladar al cliente siempre por la ruta mas rápida, baja considerablemente el tiempo de servicio promedio total de la simulación.

Consideramos, para finalizar, que un cambio significativo en el valor de las tasas lograría obtener valores que representarían mejores datos estadísticas en lo que respecta a el número promedio de clientes en cola.

```
import numpy as np
from matplotlib import pyplot
import random


class Cliente(object):
    def __init__(self, tiempo):
        self.tiempo = tiempo
        self.ingreso = tiempo
        self.prioridad = random.random()

class Servidor(object):
    def __init__(self, tasa):
        self.tasa = tasa
        self.cli = None
        self.tiempo_entrada = 0
        self.areaBdeT = 0
        self.UPS = 0


class Cola(object):
    def __init__(self):
        self.colaEsp = []
        self.areaQdeT = 0

        self.ultimoEvento = 0
        self.NPCC = 0


class Simulacion(object):
    def __init__(self):
        """Constructor que inicializa las variables de la simulacion."""
        self.TMArribo = 0.8  # Tiempo medio de arribo cola 0
        self.servidores = [Servidor(0.5), Servidor(0.5), Servidor(0.5), S
ervidor(0.2), Servidor(0.4),
                           Servidor(0.6)]  # Arreglo para los tiempos de
partidas de cada servidor
        self.colas = [Cola(), Cola(), Cola(), Cola()]  # Arreglo para gua
rdar los arribos en cada cola
        self.reloj = 0
        self.relof_final = 0
        self.proxEvento = []
        self.tiempoSimu = []
        self.colaArr = []
        self.TPS = 0

    def exponential(self, tasa):
        """Devuelve un valor aleatorio de la funcion exponencial con la t
asa indicada"""
```

```
        return np.random.exponential(tasa)

    def Inicializacion(self):
        """Inicializa el arreglo con los primeros eventos"""
        self.proxEvento = [[Cliente(self.exponential(self.TMArribo)), 0],
 [Cliente(99999), 0]]  # Fuerza el primer arribo
        self.colaArr.append(self.proxEvento[0][0])

    def Tiempos(self):
        """Analiza cual es el siguiente evento al cual le da el reloj"""

        if self.proxEvento[0][0].tiempo <= self.proxEvento[1][0].tiempo:
 # Compara si va un arribo o una partida
            self.reloj = self.proxEvento[0][0].tiempo  # Reasigno el relo
j
            self.Arribo(self.proxEvento[0][0], self.proxEvento[0][1])  #
Invoco un arribo con la cola a la que pertenece
        else:
            self.reloj = self.proxEvento[1][0].tiempo  # Reasigno el relo
j
            self.Partida(self.proxEvento[1][0], self.proxEvento[1][1])  #
 Invoco partida con el servidor al que pertenece
        ult_tiempo = min(self.proxEvento[0][0].tiempo,self.proxEvento[1][
0].tiempo)
        if ult_tiempo>100:
            self.reloj_final = self.reloj

    def Arribo(self, cli, cola):
        ser = self.Evaluar_Servidor(cola)
        if ser == 31:  # Servidor Ocupado
            self.colas[cola].areaQdeT += len(self.colas[cola].colaEsp) *
(self.reloj - self.colas[cola].ultimoEvento)
            self.colas[cola].ultimoEvento = self.reloj
            if cola == 0:
                self.colaArr.pop(0)  # Saco al cliente de la cola de arri
bos
            self.colas[cola].colaEsp.append(cli)  # Asigno cliente en col
a de espera
        else:

            cli.tiempo = self.reloj + self.exponential(self.servidores[se
r].tasa)  # Genero nueva partida y cambio la variable de la clase
            self.servidores[ser].cli = cli  # Agrego cliente al servidor
            self.servidores[ser].tiempo_entrada = self.reloj



            if cola == 0:
                self.colaArr.pop(0)
```

```
            pos = self.BuscaMin()

            self.proxEvento[1] = [self.servidores[pos].cli, pos]
        if cola == 0:
            cli = Cliente(self.reloj + self.exponential(self.TMArribo))
            self.colaArr.append(cli)  # Busco el prox arribo
            self.proxEvento[0] = [self.colaArr[0], 0]  # Cargo el proximo
 arribo

    def Partida(self, cli, servidor):
        if servidor == 0 or servidor == 1 or servidor == 2: #analiza si e
s la primer linea de servidores o la segunda

            col = self.Evaluar_Cola_sig(servidor)
            self.Arribo(cli, col) #genera arribo para cola 1, 2 o 3
        else:
            self.tiempoSimu.append(self.reloj - cli.ingreso) #Acumula tie
mpo
        self.servidores[servidor].areaBdeT += self.reloj - self.servidore
s[servidor].tiempo_entrada
        cola = self.Evaluar_Cola(servidor) #Evalua la cola anterior de la
 que vino el cliente
        if len(self.colas[cola].colaEsp) == 0:
            self.servidores[servidor].cli = None
        else:
            pos = self.FIFO()
            self.servidores[servidor].cli = self.colas[cola].colaEsp[pos]
            self.servidores[servidor].cli.tiempo = self.reloj + self.expo
nential(self.servidores[servidor].tasa)
            self.servidores[servidor].tiempo_entrada = self.reloj
            self.colas[cola].areaQdeT += len(self.colas[cola].colaEsp) *
(self.reloj - self.colas[cola].ultimoEvento)
            self.colas[cola].ultimoEvento = self.reloj
            self.colas[cola].colaEsp.pop(pos)
        ser = self.BuscaMin()

        if ser == 31:
            self.proxEvento[1] = [Cliente(99999), 0] #Fuerza arribo si no
 hay nadie en el sistema
        else:
            self.proxEvento[1] = [self.servidores[ser].cli, ser]

    def Run(self):
        self.Inicializacion()
        while True:
            self.Tiempos()

            if  self.reloj > 500:
```

```
                break
        self.reloj = self.reloj_final
        self.Reporte()



    def FIFO(self):
        return 0

    def LIFO(self, cola):
        if cola == 0:
            return len(self.colas[0].colaEsp) - 1
        else:
            return 0

    def Prioridad(self, cola):
        if cola == 0:
            priori = []
            for c in self.colas[0].colaEsp:
                priori.append(c.prioridad)
            if min(priori) <= 0.03:
                return priori.index(min(priori))
            else:
                return 0
        else:
            return 0

    def Reporte(self):
        i = 0
        for c in self.colas:
            c.NPCC = c.areaQdeT / self.reloj #Numero promedio de clientes
 en cola

            NPCCTot[i].append(c.NPCC)
            npcprom = sum(NPCCTot[i])/len(NPCCTot[i])
            NPCCProm[i].append(npcprom)
            i += 1
        i = 0
        for s in self.servidores:
            s.UPS = s.areaBdeT / self.reloj #Utilizacion promedio del ser
vidor

            UPSTot[i].append(s.UPS)
            UPSProm[i].append(sum(UPSTot[i])/len(UPSTot[i]))
            i += 1
        self.TPS = sum(self.tiempoSimu) / len(self.tiempoSimu) #Tiempo pr
omedio de la simulacion x cliente
        TPSTot.append(self.TPS)
        TPSProm.append(sum(TPSTot)/len(TPSTot))
```

```
    def Analisis(self):
        for i in range(4):
            self.Graficar(NPCCProm[i], "Nro Prom de Clientes en Cola Prom
edio.")
            print(NPCCProm[i][-1])
        for i in range(6):
            self.Graficar(UPSProm[i], "Utilizacion Prom de Servidores Pro
medio.")
            print(UPSProm[i][-1])
        self.Graficar(TPSProm, "Tiempo Prom de Servicio Promedio")
        print(TPSProm[-1])

    def Graficar(self, valor, tit):
        pyplot.plot(valor)
        pyplot.suptitle(tit)
        pyplot.show()

    def Evaluar_Servidor(self, cola):
        if cola == 0:
            dispo = []
            for i in range(3):
                if self.servidores[i].cli is None:
                    dispo.append(i)
            if len(dispo) != 0:
                return random.choice(dispo)
            """for i in range(3):
                if self.servidores[i].cli is None:
                    return i"""

        elif self.servidores[cola + 2].cli is None:
            return cola + 2
        return 31

    def BuscaMin(self):
        """Analiza la proxima partida con el menor tiempo y devuelve el s
ervidor"""
        tiempo = []
        for i in range(6):
            if self.servidores[i].cli is not None:
                tiempo.append(self.servidores[i].cli.tiempo)
            else:
                tiempo.append(99999)
        if tiempo.count(99999) != 6:
            return tiempo.index(min(tiempo))
        else:
            return 31
```

```
    def Evaluar_Cola(self, servidor):  # Segunda Linea de Colas
        if servidor == 0 or servidor == 1 or servidor == 2:
            return 0
        else:
            return servidor - 2

    def Evaluar_Cola_sig(self,servidor):
        return  servidor + 1

    def Evaluar_Cola_Mejorado(self):  # Segunda Linea de Colas
        disp = [] #Primero analizo si algun servidor esta disponible, por
 orden prioriza los mas rapidos
        for i in range (3):
            if self.servidores[i+3].cli == None:
                disp.append(0)
            else:
                disp.append(1)
        if disp.count(1) != 3:
            for i in range(3):
                if disp[i] == 0:
                    return i+1
        gente_colas = []

        for c in range(3):
            cant_gente = len(self.colas[c+1].colaEsp)
            gente_colas.append(cant_gente)
            #if(cant_gente != 0):
             #   band = False
        #if band == True:
         #   return random.randint(1,3)
        #else:
        return gente_colas.index(min(gente_colas)) +1



NPCCTot = [[],[],[],[]]
UPSTot = [[],[],[],[],[],[]]
TPSTot = []
UPSProm = [[],[],[],[],[],[]]
TPSProm = []
NPCCProm = [[],[],[],[]]

for i in range(1000):
    sim = Simulacion()
    sim.Run()
sim.Analisis()
```