Experimentación y Análisis

G2 - Modelos Estocásticos y Simulación

Primer Parcial, Quinto Punto

Juan Esteban Alarcón Bravo
David Sneider Ovalle Pineda
Gustavo Alberto Puentes Romero

A. Compilación y corrección inicial

Inicialmente se realizaron correcciones menores al código original. Primero, se corrigió un error tipográfico en el archivo `lcgrand.cpp`. En la línea 46 de la imagen, entre el 7 y el 1 había dos barras verticales, pero el libro de Law muestra solo una:

```cpp
double lcgrand(int num) {
    long zi, lowprd, hi31;

    zi     = zrng[num];
    lowprd = (zi & 65535) * MULT1;
    hi31   = (zi >> 16) * MULT1 + (lowprd >> 16);
    zi     = ((lowprd & 65535) - MODLUS) +
             ((hi31 & 32767) << 16) + (hi31 >> 15);
    if (zi < 0) zi += MODLUS;
    lowprd = (zi & 65535) * MULT2;
    hi31   = (zi >> 16) * MULT2 + (lowprd >> 16);
    zi     = ((lowprd & 65535) - MODLUS) +
             ((hi31 & 32767) << 16) + (hi31 >> 15);
    if (zi < 0) zi += MODLUS;
    zrng[num] = zi;
    return (zi >> 7 | 1) / 16777216.0;
}
```

```cpp
float lcgrand(int stream)
{
    long zi, lowprd, hi31;

    zi     = zrng[stream];
    lowprd = (zi & 65535) * MULT1;
    hi31   = (zi >> 16) * MULT1 + (lowprd >> 16);
    zi     = ((lowprd & 65535) - MODLUS) +
             ((hi31 & 32767) << 16) + (hi31 >> 15);
    if (zi < 0) zi += MODLUS;
    lowprd = (zi & 65535) * MULT2;
    hi31   = (zi >> 16) * MULT2 + (lowprd >> 16);
    zi     = ((lowprd & 65535) - MODLUS) +
             ((hi31 & 32767) << 16) + (hi31 >> 15);
    if (zi < 0) zi += MODLUS;
    zrng[stream] = zi;
    return (zi >> 7 | 1) / 16777216.0;
}
```

Las semillas del generador de números aleatorios **no** se cambiaron.

Al archivo principal de código se le cambió el nombre por `main.cpp` y se ejecutó con los parámetros del primer ejemplo contemplado en el libro:

```
result.txt
Sistema de Colas Simple

Tiempo promedio de llegada      1.000 minutos

Tiempo promedio de atencion          0.500 minutos

Numero de clientes          1000


Espera promedio en la cola      0.430 minutos

Numero promedio en cola     0.418

Uso del servidor          0.460

Tiempo de terminacion de la simulacion    1027.915 minutos
```



```
Single-server queueing system

Mean interarrival time      1.000 minutes

Mean service time           0.500 minutes

Number of customers          1000


Average delay in queue      0.430 minutes

Average number in queue     0.418

Server utilization          0.460

Time simulation ended    1027.915 minutes
```

FIGURE 1.19
Output report, queueing model.

Podemos apreciar que los resultados de la simulación son los mismos del libro, por lo que entendemos que el código funciona adecuadamente hasta este punto.

B. Verificación de módulos

Los módulos del programa están referenciados en la siguiente tabla:

MÓDULO	FUNCIÓN QUE LO REPRESENTA	NOTAS
Inicialización	`82 > void inicializar(void)`	Funcionalidades ya incluidas y puestas en el módulo con la excepción de las funciones *fopen()* y *fscanf()* que estaban en *main()* y fueron reubicadas.
Manejo de Espacio-Tiempo	`122 > void controltiempo(void)`	Funcionalidades ya incluidas y ubicadas en el módulo.
Eventos	`149 > void llegada(void)` `197 > void salida(void)`	Los dos eventos básicos de un sistema de colas, *llegada()* y *salida()*. Funcionalidades ya incluidas y ubicadas en el módulo.

Función Percentil	`float percentil(float param_poblacional) // Función Percentil`
	`{`
	`  return tiempo_simulacion + expon(param_poblacional);`
	`}`
		La función percentil fue una nueva implementación en el código. Se definió como $ts + Exp(p)$, en donde ts es el tiempo de la simulación, $Exp()$ es la función exponencial ya implementada y p es el parámetro poblacional.

Generador de Reporte	`230 > void reportes(void)`	Funcionalidades ya incluidas y ubicadas en el módulo.
Simulador Principal	`47 > int main(void)`	Funcionalidades ya incluidas en el programa. Llama las funciones *inicializar()*, *controltiempo()* y *reporte()* y las funciones de eventos *llegada()* y *salida()*.

C. Verificación del simulador

El archivo de Excel incluido con el código traía dos columnas de datos: Tiempo entre llegadas y tiempo de atención. En esta sección se mostrará la fiabilidad del simulador para reproducir estos datos.

Primero, para poder usar los datos del archivo en el simulador, se obtuvo la media de las dos columnas de datos:

	A	B	C
1	Cliente i-ésimo	Tiempo entre la llegada del i-ésimo y el (i-1)-ésimo cliente (segundos)	Tiempo atención del i-ésimo cliente (segundos)
2		*Media: 5,141*	*Media: 2,932*
3	1	14,51578699	6,024561883
4	2	5,157072108	3,949211335
5	3	5,845257374	6,274187291
6	4	5,804763104	2,641817026

Estas medias, que están en segundos, deben pasarse a minutos para ser usada como parámetros del simulador:

$$5{,}141\ seg \approx 0{,}086\ min \qquad 2{,}932\ seg \approx 0{,}049\ min$$

Se introducen estos datos en el archivo de parámetros del simulador:

param.txt
```
0.086
0.049
1000
```

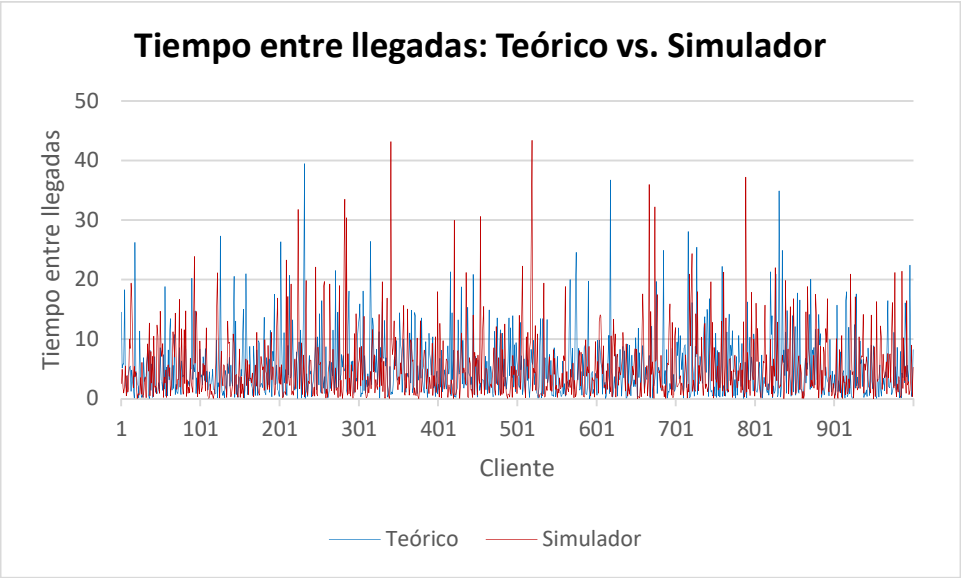
En el archivo principal del simulador se agregó una línea de código que imprime en el archivo `datosLlegada.txt` el valor del tiempo entre llegadas que se va almacenando con cada iteración de la simulación:

```
fprintf (datosLlegada, "%f\n",(tiempo_sig_evento[1]-tiempo_anterior)*60);
```

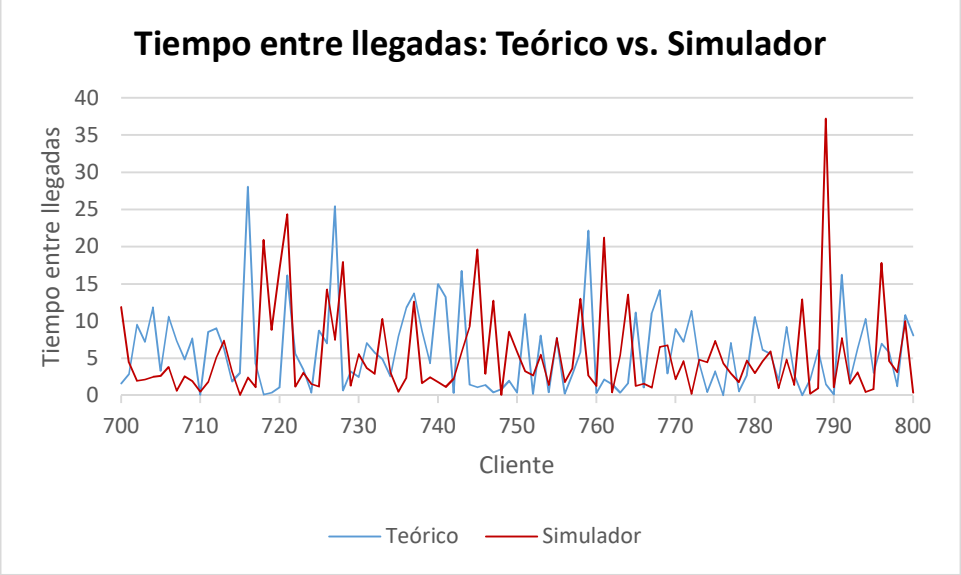
El resultado son estos 1000 datos:

datosLlegada.txt
```
2,520329
4,952023
1,773655
0,939066
2,252969
5,443547
0,485991
3,849190
1,124455
1,843253
```

Transcribiendo esos datos a Excel y comparándolos con los datos originales, obtenemos la siguiente gráfica:



Ampliando una sección arbitraria (intervalo 700-800), podemos ver en mejor detalle los resultados:



Podemos entonces observar que el simulador puede representar de manera aproximada los datos teóricos del archivo de datos proporcionado.

Estos son los resultados del generador de reportes:

result.txt

```
Sistema de Colas M/M/1

Tiempo promedio de llegada      0.086 minutos

Tiempo promedio de atencion           0.049 minutos

Numero de clientes          1000



Espera promedio en la cola      0.070 minutos

Numero promedio en cola     0.803

Uso del servidor          0.544

Tiempo de terminacion de la simulacion      86.868 minutos
```

D. Modificación para modelos con intervalos uniformes

Para poder utilizar el simulador en estos nuevos modos de operación, se creó la variable `modo_operacion`. Esta variable modifica el flujo del programa para que pueda utilizar distintos valores de entrada. En este caso, para el "tiempo promedio de llegada" no se utilizará la función exponencial con media aleatoria que estaba implementada por defecto, sino que se usará un intervalo uniforme que hace uso del generador de números aleatorios para escoger un valor de llegada:

```
// Inicializa la lista de eventos.
// Ya que no hay clientes, el evento salida (terminacion del servicio) no se tiene en cuenta
switch(modo_operacion){
    case 1: // Modo M/M/1
        tiempo_sig_evento[1] = tiempo_simulacion + expon(media_entre_llegadas);
    break;
    case 2: // Modo U/M/1
        tiempo_sig_evento[1] = tiempo_simulacion + uniform(a1,b1);
    break;
    case 3: // Modo U/U/1
        tiempo_sig_evento[1] = tiempo_simulacion + uniform(a1,b1);
    break;
    case 4: // Modo M/M/n
        tiempo_sig_evento[1] = tiempo_simulacion + expon(media_entre_llegadas);
    break;
};
```

En vez de usar la función expon() para ajustar el tiempo de la siguiente llegada, se usa uniform(), por ejemplo.

```
float uniform(float a, float b) //
{
    return a + lcgrand(1) * (b-a);
}
```

```
float uniform(float a, float b)  /* Uniform variate generation function. */
{
    /* Return a U(a,b) random variate. */

    return a + lcgrand(1) * (b - a);
}
```

FIGURE 1.43
C code for function uniform.

Detalle de la función uniform(), copiada del libro de Law.

Estas dos nuevas implementaciones nos ayudan también a resolver los siguientes dos casos, en donde se implementa un modelo U/M/1 y un modelo U/U/1.

4.1. Modelo $Uni(a_1, b_1)/M/1$

Nuevamente vamos a comparar los datos teóricos con los prácticos, pero para este modelo necesitamos definir valores adecuados para usar en el intervalo $[a_1, b_1]$. Para esto, calculamos ahora también la desviación estándar de los datos:

Tiempo entre la llegada del i-ésimo y el (i-1)-ésimo cliente (segundos)
Media: 5,141
Desv Est: 5,222
14,51578699
5,157072108
5,845257374
5,804763104
18,24595104
2,165952654
1,621634599
0,892162039

El intervalo entonces lo definimos arbitrariamente como $[m - \sigma, m + \sigma]$ (en minutos, nuevamente). Estos parámetros los pasamos al simulador, lo ejecutamos y obtenemos los siguientes resultados:

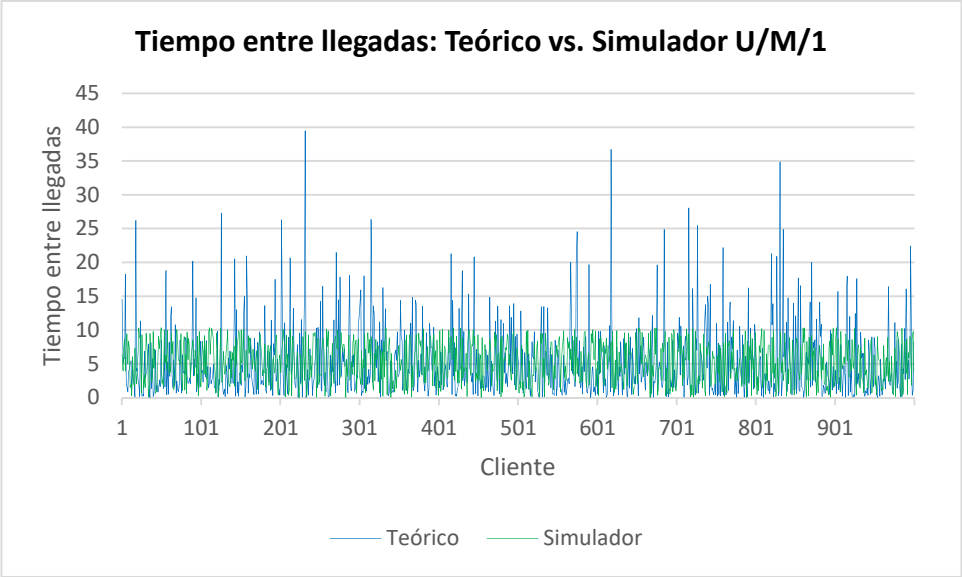
result.txt

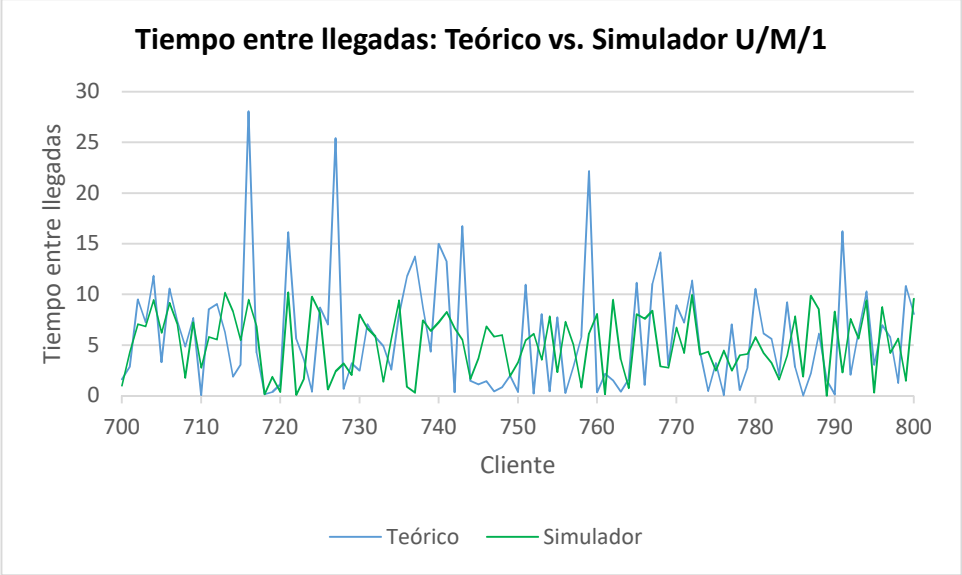
```
Sistema de Colas U/M/1

a1      0.000000

b1      0.173000

Tiempo promedio de atencion          0.049 minutos

Numero de clientes          1000


Espera promedio en la cola      0.032 minutos

Numero promedio en cola     0.365

Uso del servidor          0.546

Tiempo de terminacion de la simulacion      88.092 minutos
```

Nota: el valor mínimo del intervalo lo establecimos como 0 porque $m - \sigma$ es un número negativo, y esto en términos de tiempo no es posible

Lo primero que vemos es que el tiempo de espera y el número promedio de personas en la cola se redujo a aproximadamente la mitad que en el caso anterior (M/M/1). Por otro lado, el uso del servidor y el tiempo de terminación de la simulación son similares.

Veamos ahora el comportamiento del tiempo entre llegadas:





Lo que más resalta de esta simulación es la uniformidad de los datos: parecen bien distribuidos en un intervalo aproximado de [0,10]. Esto era de esperarse, ya que la función uniforme no genera tantos datos atípicos como una distribución exponencial, incluso cuando ambas hacen uso del generador de números aleatorios.

La desviación estándar también es menor en este caso, aunque la media es similar:

Tiempo entre llegadas (Resultados del Simulador U/M/1)
Media: 5,285
Desv Est: 3,023
6,369016
3,975644
7,360657
8,652871
3,898226
7,925817
9,446994

4.2. Modelo $Uni(a_1, b_1)/Uni(a_2, b_2)/1$

Aplicando el mismo procedimiento del modelo anterior, calculamos la media y la desviación estándar de las dos variables:

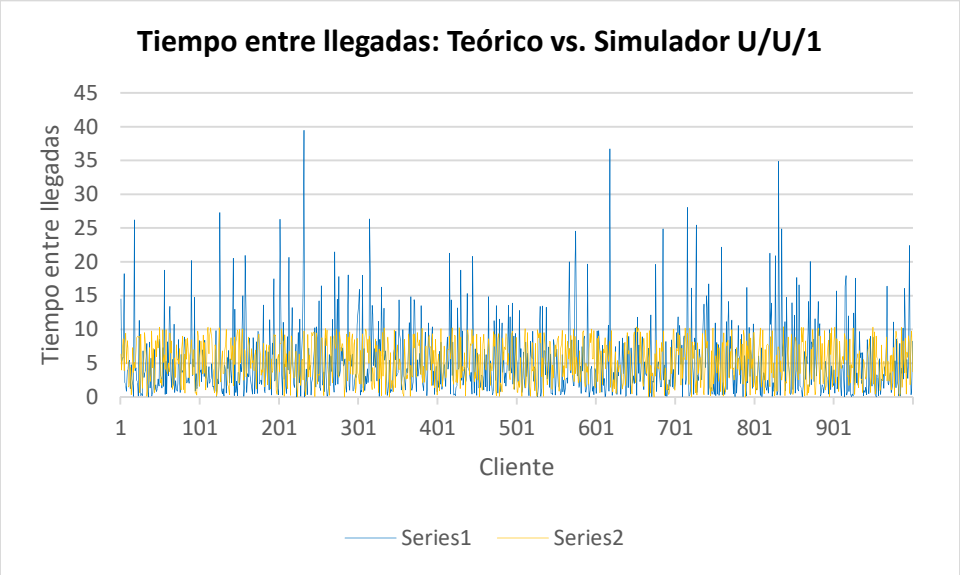
Tiempo entre la llegada del i-ésimo y el (i-1)-ésimo cliente (segundos)	Tiempo atención del i-ésimo cliente (segundos)
Media: 5,141	*Media: 2,932*
Desv Est: 5,222	*Desv Est: 2,826*
14,51578699	6,024561883
5,157072108	3,949211335
5,845257374	6,274187291
5,804763104	2,641817026

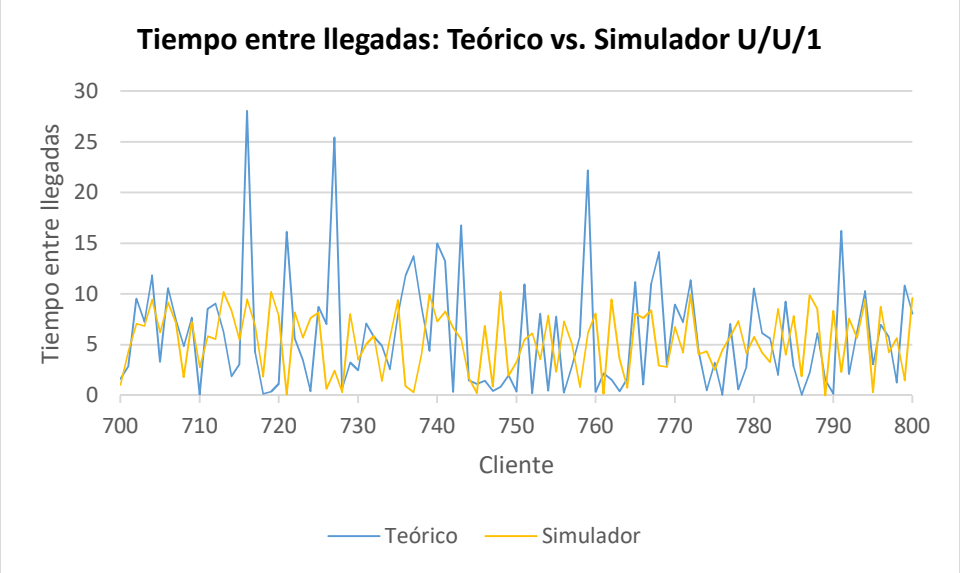
Convertimos estos valores a minutos y los ingresamos en el simulador:

result.txt

```
Sistema de Colas U/U/1

a1      0.000000

b1      0.173000

a2      0.002000

b2      0.096000

Numero de clientes          1000


Espera promedio en la cola      0.018 minutos

Numero promedio en cola     0.207

Uso del servidor          0.565

Tiempo de terminacion de la simulacion      88.091 minutos
```

Nuevamente los valores de espera promedio y número promedio en la cola se redujeron a la mitad de la simulación anterior (y, por ende, a una cuarta parte de la simulación por defecto M/M/1).





Nuevamente vemos una distribución de datos uniforme, muy similar a la anterior.

E. Modificación para modelos con múltiples servidores (M/M/n)

Para generalizar el simulador a todos los casos M/M/n desde su estado actual (que es un caso particular de este modelo, donde $n = 1$), debemos hacer ajustes en el control de la variable `estado_servidor`. Originalmente, esta tenía dos estados definidos así:

```
#define OCUPADO 1 // Indicador de Servidor Ocupado
#define LIBRE 0 // Indicador de Servidor Libre
```

Esto solo permitía caracterizar un servidor con el estado "libre" u "ocupado", pero al cambiar esta variable para que utilice más valores, podemos hacer que sea interpretada como "el número de servidores disponibles".

Por ejemplo, en la función `llegada()`, la línea 224 fue reemplazada por la línea 225: Ahora, ya no se revisa si "el servidor está ocupado" sino que se revisa si "en número de servidores disponibles es menor o igual al número total de servidores", o, en palabras más sencillas, "¿hay servidores disponibles?":

```
224     // if (estado_servidor == OCUPADO) {
225     if (estado_servidor >= num_servidores) {
```

La variable `num_servidores` es un nuevo parámetro de entrada que se refiere a la n del modelo M/M/n que queremos trabajar. Si esta fuera dejada con un valor de 1, por ejemplo, obtendríamos los mismos resultados que en el caso trivial del modelo M/M/1.

5.1. Modelo $M/M/n$

Ya con el simulador modificado, probamos su funcionalidad con diferentes valores de n. Específicamente, se probó con $n = 2,3,5,10,50$.

result.txt

```
Sistema de Colas M/M/n

Tiempo promedio de llegada      0.086 minutos

Tiempo promedio de atencion          0.049 minutos

Numero de servidores             2

Numero de clientes          1000


Espera promedio en la cola      0.070 minutos

Numero promedio en cola     0.801

Uso del servidor          1.541

Tiempo de terminacion de la simulacion      86.847 minutos
```

$n = 2$

result.txt

```
Sistema de Colas M/M/n

Tiempo promedio de llegada      0.086 minutos

Tiempo promedio de atencion          0.049 minutos

Numero de servidores             3

Numero de clientes          1000


Espera promedio en la cola      0.070 minutos

Numero promedio en cola     0.801

Uso del servidor          2.538

Tiempo de terminacion de la simulacion      86.847 minutos
```

$n = 3$

result.txt

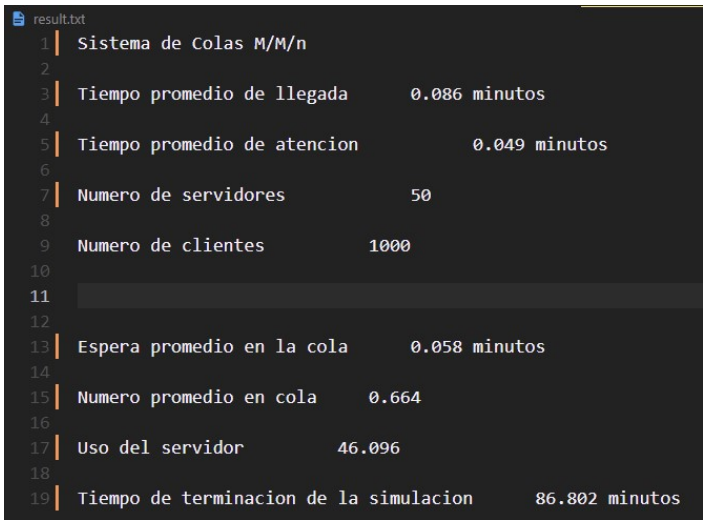
```
Sistema de Colas M/M/n

Tiempo promedio de llegada      0.086 minutos

Tiempo promedio de atencion          0.049 minutos

Numero de servidores             5

Numero de clientes          1000


Espera promedio en la cola      0.069 minutos

Numero promedio en cola     0.800

Uso del servidor          4.529

Tiempo de terminacion de la simulacion      86.847 minutos
```

$n = 5$

result.txt

```
Sistema de Colas M/M/n

Tiempo promedio de llegada      0.086 minutos

Tiempo promedio de atencion          0.049 minutos

Numero de servidores            10

Numero de clientes          1000


Espera promedio en la cola      0.062 minutos

Numero promedio en cola     0.708

Uso del servidor          9.429

Tiempo de terminacion de la simulacion      87.698 minutos
```

$n = 10$



$n = 50$

Vemos entonces que el uso del servidor ahora representa la cantidad promedio de servidores que estuvieron en uso durante la simulación. Este dato refleja fidedignamente el parámetro de entrada n, lo que significa que el sistema está usando todos los servidores en la simulación.