Universidad Politécnica de Cartagena

Análisis de teletráfico mediante una herramienta de simulación gráfica

Grado en Ingeniería Telemática

Trabajo Fin de Grado

Alumno: Sergio Noël Moreno Pérez
Tutor: José María Malgosa Sanahuja

ÍNDICE DE CONTENIDOS

1. CONCEPTOS BASICOS DEL TELETRAFICO

1.1. Introducción

En lo referente a este trabajo, cuando nos refiramos al concepto de teletráfico entenderemos que estamos hablando de todo tipo de tráfico, ya sea de comunicaciones de datos o de tráfico telefónico.

La Teoría del Teletráfico consiste en la aplicación de la teoría de probabilidades y la teoría de estadísticas para la solución de problemas relacionados con la planificación, la evaluación de prestaciones, la operación y el mantenimiento de sistemas de telecomunicación.

En base a esto, definimos el objetivo de esta teoría como:

> *El diseño de un sistema efectivo en costes, que cumpla con un conjunto de características estructurales, medidas de desempeño y criterios de optimización predefinidos, con el fin de satisfacer las demandas futuras.*

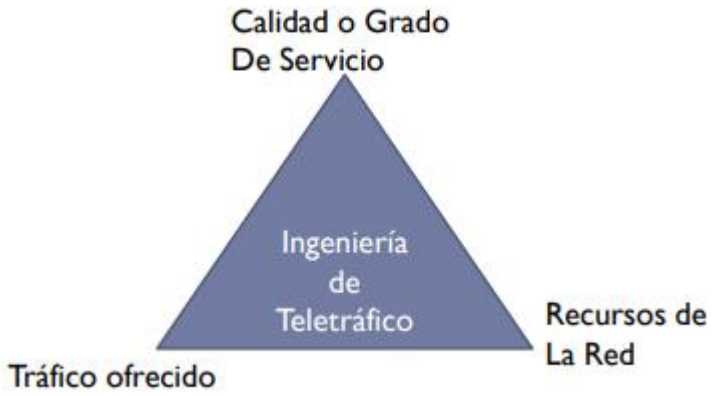

***Figura 1**. Relación entre los objetivos, el QoS y los recursos*

Para cumplir con estos fines necesitamos de métodos de pronóstico de demandas, métodos para calcular la capacidad del sistema y la construcción de medidas de desempeño con el fin de medir el grado de servicio.

Para ello utilizamos una serie de herramientas, las cuales son:

- Procesos estocásticos: Sirven para caracterizar una sucesión de variables aleatorias que evolucionan con el tiempo.

- Simulación por computador: Programa informático cuyo fin es crear un modelo abstracto de un determinado sistema, lo cual nos permite comprender mejor su funcionamiento mediante test del tipo prueba y error.

- Mediciones de tráfico: Aunque el tráfico real es aleatorio, podemos estimar la cantidad de tráfico que habrá por una línea gracias a los hábitos y costumbres de la población, es

decir, medir el uso que se hace de los recursos de comunicaciones durante la jornada laboral, durante las horas nocturnas, etc.

- Teoría de colas: Es el estudio matemático de sistemas de espera. Se utiliza para estudiar factores como el tiempo medio de espera en las colas o la capacidad de trabajo del sistema sin que llegue a colapsar.

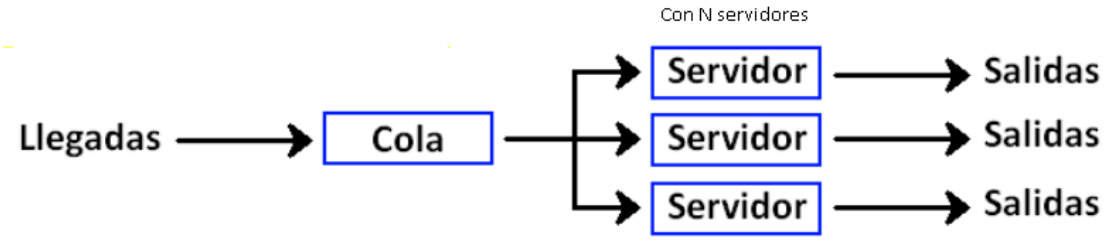

***Figura 2**. Representación de una cola simple M/M/N*

A continuación se describirán una serie de parámetros que en su conjunto constituyen la teoría del teletráfico. Dichos parámetros se describen de forma intuitiva y su definición matemática se simplifica cuando se asumen comportamientos estadísticos tales como ergodicidad, estabilidad en régimen permanente y la ley de los grandes números, todos ellos sucesos habituales en sistemas de telecomunicación.

A partir de ahora se asumirá que nuestro sistema de comunicaciones puede modelarse por un sistema de colas parecido al ilustrado en la figura 2.

1.2. Tasa cursada, rechazada y ofrecida

Definimos tasa como el número de sucesos *n* que suceden en un intervalo de tiempo de observación (Tobs). A partir de esta definición diferenciamos los tres tipos de tasas:

- Tasa Cursada: Número de llamadas a las cuales damos servicios (#C) en un instante tiempo (Tobs).

$$\lambda c = \frac{\#C}{Tobs}$$

- Tasa Ofrecida: Número total de llamadas generadas por la población (#O), a las cuales se le puede dar servicio en un instante tiempo (Tobs).

$$\lambda o = \frac{\#O}{Tobs}$$

- Tasa Perdida: Número de llamadas que son rechazadas (#R) en un instante tiempo (Tobs). El rechazo se produce cuando el sistema de colas no dispone de suficientes recursos para almacenar temporalmente la petición ofrecida.

$$\lambda R = \frac{\#R}{Tobs}$$

1.3. Trafico cursado, rechazado y ofrecido

Usamos tráfico para denotar la Intensidad del flujo de paquetes que circulan en una red de telecomunicaciones. La intensidad de tráfico instantánea en un sistema de telecomunicaciones compuesto por un conjunto de servidores es el número de servidores ocupados en un instante de tiempo dado. La intensidad instantánea se denota por la función de *n(t).* Gracias a esto podemos definir la Intensidad media del tráfico (cuya unidad de medida se denomina como Erlang) en un periodo como:

$$I = \frac{1}{T} \cdot \int_0^T n(t)dt = \frac{V}{T}$$

El bloqueo ocurre cuando el sistema no tiene recursos disponibles para atender una petición entrante.

A partir de las definiciones aportadas procedemos a definir los diferentes tipos de tráfico

- **El Tráfico Cursado** (TC) se define como el número medio de servidores ocupados en el intervalo T. Cuando el sistema de colas solo tiene un servidor, también representa la utilización o carga de la línea. El volumen de tráfico (V) coincide con la suma de todos los tiempos de servicio.

- **El Tráfico Ofrecido** (TO), corresponde al tráfico que sería cursado si no hubiera solicitudes rechazadas bajo el supuesto de un sistema de capacidad ilimitada. Gracias a la ley de Little, también se puede definir como la tasa ofrecida por la población multiplicada por el tiempo medio de servicio. Matemáticamente se expresa como:

$$TO = \lambda \cdot \bar{t_s}$$

- **El Trafico Rechazado** (TL), corresponde a la diferencia entre el Tráfico Ofrecido y el Tráfico Cursado. Es decir:

$$TP = TO - TC$$

Para disminuir el tráfico rechazado la solución más obvia seria incrementar la capacidad del sistema, aunque ello no siempre es posible.

A continuación, vamos a proceder a definir las diferentes formas en las que se cursa el tráfico en función de la hora de ocupación y del bloqueo.

Y es que varios estudios demuestran que, si se conocen el contexto, la estructura y la actividad principal a la cual se dedica un sistema, el comportamiento del tráfico presenta patrones regulares a pesar de ser estocástico.

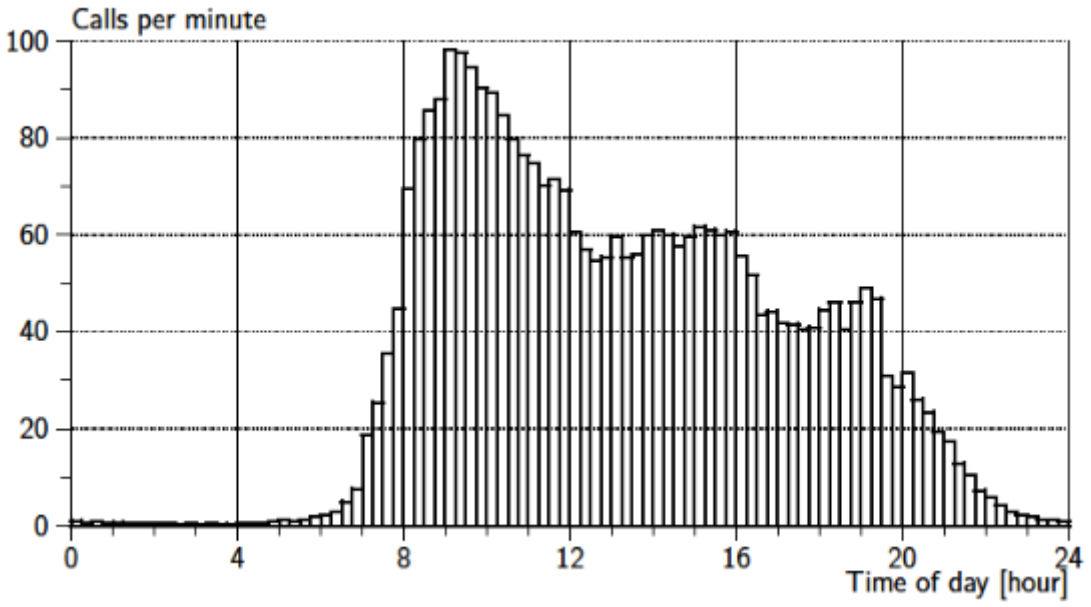

***Figura 3**. Ejemplo de comportamiento de un sistema de teletráfico*

Por ejemplo, la figura 3 muestra la ocupación a lo largo de las 24 horas de un día laboral de un sistema telefónico. Podemos observar que por la noche apenas hay actividad y conforme avanza el día ésta aumenta hasta que vuelve a reducirse por la noche. Los picos de tráfico que se producen a las 8 a.m. y a las 16 p.m. se deben a la actividad laboral, mientras que el pequeño pico que se produce entorno a las 17:30 p.m. se debe al tráfico telefónico hogareño.

También se define otro término a partir de estas gráficas, y es la hora cargada. La hora cargada es el periodo del día el cual recibe el mayor número de solicitudes en todo el día.

Debido a estas variaciones del teletráfico según la forma en que se trate la demanda distinguimos entre sistemas con pérdidas y sistemas de espera.

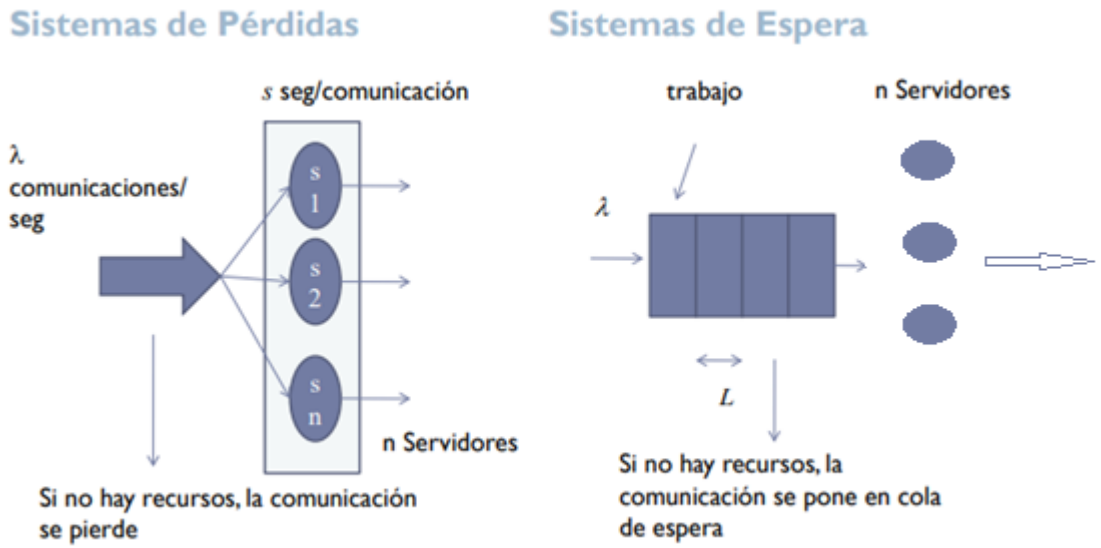

***Figura 4**. Sistema de Perdidas/ Sistema de Espera*

1.4. Probabilidad de perdida

En un sistema de perdidas, se produce una perdida cuando se rechaza una petición ofrecida si la comunicación deseada no se puede establecer inmediatamente debido a una situación de bloqueo.

Entendemos pues que la probabilidad de pérdida es la fracción (con respecto al total de llamadas que se ofrecen) de los intentos de llamadas que observan todos los servidores ocupados.

$$PL = \frac{\#LlamadasRechazadas}{\#LlamadasOfrecidas}$$

1.5. Probabilidad de demora

En un sistema de espera al contrario que en un sistema de pérdidas, cuando no se puede cursar una petición ofrecida debido a una situación de bloqueo, la petición queda en cola a la espera de que se resuelva el bloqueo y se pueda satisfacer la demanda.

Entendemos pues que la probabilidad de bloqueo es la fracción de tiempo en que todos los recursos del sistema (servidores y cola) están ocupados.

$$\boldsymbol{PB = \frac{T_{todos\_los\_recursos\_ocupados}}{Tob}}$$

Teniendo en cuenta los sistemas de perdida y demora, en función de cómo lo implementemos nos hayamos con sistemas mixtos los cuales combinan ambas técnicas, tanto la de perdida como la de demora y los sistemas puros los cuales solo implementan una de las dos técnicas.

1.6. Ejemplo Red Telefónica

Vamos a estudiar como ejemplo un sistema de telefonía móvil. Un sistema de telefonía móvil es aquel en el que los teléfonos están conectados a las antenas transmisoras y receptoras por medio de ondas electromagnéticas.

En este sistema la red está formada por un conjunto de antenas repartidas por la superficie terrestre y por los terminales los cuales permiten el acceso a esta red.

Los operadores se encargan de cubrir el área de cobertura usando células hexagonales, creando una inmensa red de hexágonos. Cada hexágono constara de una estación base la cual se encarga de dar cobertura a esa área

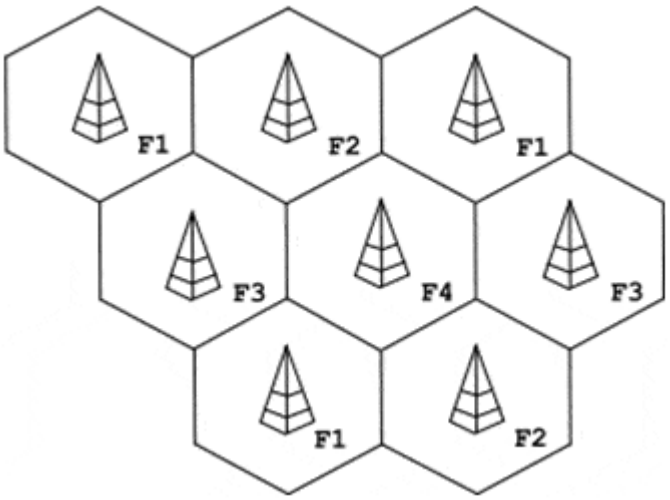

***Figura 5**. Representación de las células de una operadora 9telefónica*

Cada célula utiliza varios canales por los cuales se puede emitir una llamada. La diferencia entre los canales es la frecuencia a la que transmite. El que cada canal transmita a diferente frecuencia es lo que permite que haya varias llamadas simultáneas. Hay que tener en cuenta que las frecuencias pueden ser reutilizadas, ya que la misma frecuencia puede ser usada en un área distinta para una transmisión distinta, para ello se definirá el factor de reutilización de frecuencia, que indica cada cuanto se puede utilizar una misma frecuencia en una red. Por ejemplo como podemos observar en la figura 5, nos fijamos en la celda que utiliza la frecuencia F1, ninguna celda adyacente puede utilizar esa misma frecuencia.

El proceso de conexión empieza cuando un terminal intenta conectar con otro; para ello el terminal manda la petición de conexión a la estación base la cual recibe la petición e intenta conectar con el destinatario mediante la conexión con la central base a la que pertenezca el destinatario. Aquí se puede dar tres casos:

1. El destinatario está disponible, en este caso la centralita transmite la petición de conexión al destinatario el cual ya se encarga de decidir si responde o no. En caso de que conteste transmitirá la confirmación a la centralita y ésta terminará de establecer la conexión con el cliente

2. El destinatario no se encuentra disponible, en este caso la centralita rechazará la petición y le informara al cliente que el destinatario no está disponible para la comunicación.

3. No hay ningún canal disponible, este caso, aunque es el más extraño, no es imposible. Básicamente la centralita está saturada de peticiones por lo cual es imposible acceder al servicio ya que no hay ningún canal disponible.

En todo momento las centrales de conmutación tienen un registro de las llamadas que se están realizando ya que las estaciones base les informan periódicamente de las llamadas a las que están dando soporte.

2. MODELADO DE SISTEMAS TELEFÓNICOS

Como ya mencionamos anteriormente el uso de variables aleatorias es de gran importancia a la hora de tratar con el teletráfico, ya que son las bases de los modelos que vamos a estudiar.

Dependiendo del tipo de variable aleatoria que utilicemos estamos tratando con un modelo de:

- Exponencial: Se utilizan para medir los tiempos entre llegadas, los tiempos de servicios así como la longitud de paquetes.

- Poisson: Se utiliza para medir el número de peticiones que ocurren en las distribuciones temporales exponenciales.

- Earlang-B: Se usa para describir la probabilidad de pérdida en un grupo de circuitos. Suponemos una población infinita.

- Engset-B: Es una derivación de Erlang-B, pero a diferencia de ésta, suponemos una población finita.

2.1 Distribución estadística exponencial

Al hablar de distribución exponencial en el teletráfico, hablamos del uso de variables aleatorias no-negativas las cuales utilizamos para medir el tiempo de servicio, la duración de la congestión, tiempos de espera, etc. Podemos usar cualquier tipo de función de distribución, pero la distribución exponencial es la óptima para el uso práctico y analítico ya que es la única distribución estadística que satisface la condición *memoryless* (sin memoria), por la cual el comportamiento futuro del proceso que modela la exponencial no depende de lo que haya ocurrido en el pasado, sino solo del estado en que se encuentre el sistema en el presente.

Se caracteriza por un parámetro único denominada tasa (λ).

Matemáticamente la distribución exponencial se expresa como:

$$F(t) = 1 - e^{-\lambda t}, \qquad \lambda > 0, \qquad t \geq 0$$

$$f(t) = \lambda e^{-\lambda t}, \qquad \lambda > 0, \qquad t \geq 0$$

En cuanto a los parámetros que caracterizan una distribución exponencial tenemos:

$$\text{Mean Value} \quad m_1 = \frac{1}{\lambda}$$

$$\text{Second Moment} \quad m_2 = \frac{2}{\lambda^2}$$

$$\text{Variance} \quad \sigma^2 = \frac{1}{\lambda^2}$$

En cuanto a sus propiedades, tenemos que esta función es muy apropiada para intervalos de tiempo y que tiene la propiedad de *memoryless*. Esta propiedad nos indica que la probabilidad de que ocurra un evento después de un cierto tiempo no depende de ninguno de los sucesos ocurridos anteriormente

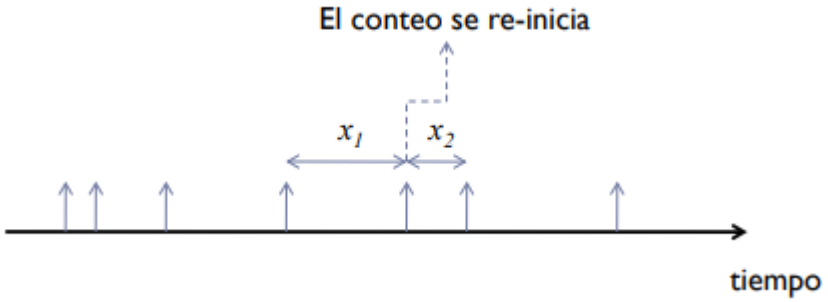

***Figura 6**. Representación de la propiedad Falta de memoria*

Procedemos a demostrarlo matemáticamente:

$$X \sim \text{Exp}(a) \Rightarrow P(X > t + s | X > s) = P(X > t), \forall s, t > 0$$

En primer lugar, partimos de que si t > 0, entonces:

$$P(X > t) = 1 - P(X \leq t) = 1 - FX(x) = 1 - (1 - e^{-at}) = e^{-at}$$

Por tanto, dado t, s > 0, se verifica que:

$$P(X > t + s | X > s) = \frac{P(X > t + s, X > s)}{P(X > s)} = \frac{P(X > t + s)}{P(X > s)} = \frac{e^{-a(t+s)}}{e^{-as}} = e^{-at} = P(X > t$$

2.2 Distribución estadística de Poisson

Para entender la distribución estadística de Poisson empezaremos explicando qué es un proceso de conteo.

Un proceso de conteo es un proceso estocástico que caracteriza la sucesión de eventos a lo largo del tiempo. En resumen, mide para cada instante de tiempo, el número de sucesos que se han dado en ese determinado tiempo.

Un proceso de Poisson es por lo tanto un proceso en el que fijado un tiempo de observación (T), el número de sucesos que ocurren siguen una variable aleatoria de Poisson. Hay tres formas de definir un proceso de Poisson.

1. Un proceso de Poisson N(t) con tasa λ es un proceso de conteo en el que no hay eventos antes del instante inicial. El número de eventos entre el instante inicial y el instante de tiempo T sigue una variable aleatoria de Poisson de media λT. Y el número de eventos en cualquier intervalo de longitud t sigue la misma distribución independiente del inicio de tiempo. Esta descripción es la más común y en la que basaremos nuestro estudio.

2. Un proceso de Poisson N(t) con tasa λ es un proceso de conteo en el que el tiempo entre eventos es independiente y se distribuye según una variable aleatoria exponencial de media $\frac{1}{\lambda}$

$$P_T(\mathrm{n}) = \frac{(\lambda T)^n}{n!} e^{-\lambda T}$$

2.3 Modelo de Earlang-B

El modelo de Earlang-B hace uso de las distribuciones estadística de Poisson y exponencial, ya que éstas se encarga de definir el comportamiento de la población (generación de llamadas) y la duración de las mismas.

Además, para este modelo, estamos suponiendo que la población de origen es infinita. Por lo tanto, cuando se produce una situación de bloqueo, las llamadas que no pueden ser atendidas se rechazan.

A la hora de asignar los recursos el sistema se puede configurar de diferentes maneras:

1. Aleatoriamente: Cuando se realice una llamada esta se asignará de manera aleatoria entre los canales disponibles

2. Secuencialmente: Los canales son numerados desde 1 hasta n y se buscará el canal vacío en orden empezando siempre desde el numero 1

3. Cíclicamente: Sigue el mismo procedimiento que la manera secuencial con la diferencia de que este empieza a buscar desde el ultimo canal que se utilizó.

La fórmula de Earlang-B se expresa como:

$$\boldsymbol{PB = B(A,m) = \frac{\frac{A^m}{m!}}{\sum_{i=0}^{m} \frac{A^i}{i!}}}$$

Donde PB es la probabilidad de bloqueo, m es el número de recursos o enlaces de los que se disponen y A es la cantidad total de tráfico (tráfico ofrecido). Como las llegadas siguen un proceso de Poisson, podemos calcular el tráfico ofrecido como $A = \frac{\lambda}{\mu}$

Podemos modelar el modelo de Erlang-B mediante el uso de cadenas de Márkov, En el cual definimos el estado del sistema como el número de recursos utilizados.

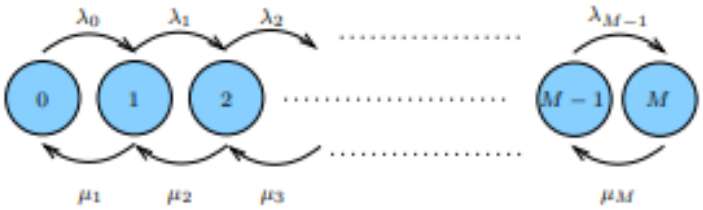

***Figura 7**. Cadena Márkov en Earlang-B*

Para pasar a un estado superior se debe producir una llamada, esto es la tasa de nacimiento, mientras que para pasar a un estado inferior debe morir (ser cursada) una llamada, lo que se denomina como tasa de muerte o simplemente tasa cursada. En el caso que nos ocupa, la tasa de nacimiento coincide con la tasa ofrecida λ y la tasa de muerte es nμ (siendo n el estado y μ la tasa de servicio de un servidor).

Para obtener las probabilidades de estado, utilizaremos ecuaciones de balance de flujo parciales, donde:

$$\lambda \cdot P(i-1) = i\mu \cdot P(i), \qquad i = 1,2, \dots \dots$$

Y con la restricción de normalización:

$$\sum_{i=0}^{M} P(i) = 1, \qquad P(i) \geq 0$$

Podemos sacar que la probabilidad de estado es:

$$P(i) = \frac{\frac{A^i}{i!}}{\sum_{j=0}^{M} \frac{A^j}{j!}}$$

Las características fundamentales de este modelo son:

- Es un modelo válido para cualquier distribución de tiempo.
- Tanto el tráfico que se cursa como el que se rechaza no son exponenciales. Ello tiene implicaciones importantes cuando se estudian sistemas de comunicaciones homogéneos conectados en cascada.
- Finalmente, al ser la generación de nuevas llamadas independientes de la ocupación del sistema, las probabilidades de pérdidas y de bloqueo coinciden.

2.4 Modelo de Engset-B

Para este tipo de modelo partiremos de todo lo ya descrito en el modelo de Erlang-B con la diferencia de que en este modelo supondremos una población compuesta por un número finito de personas (S).

También necesitamos definir un nuevo parámetro que es el factor de actividad (a); éste representa el tiempo que un servidor estaría siendo ocupado bajo el supuesto de ausencia de pérdidas. Matemáticamente se expresa como:

$$a = \frac{1/\mu}{1/\mu + \gamma} = \frac{\beta}{1 + \beta}$$

Donde $\beta = \gamma/\mu$ y representa el tráfico cursado por cada usuario sin la presencia de llamadas rechazadas.

Al igual que el caso anterior, podemos modelar el modelo Engset-B mediante el uso de cadenas de Márkov, en el cual definimos el estado del sistema como el número de recursos utilizados.

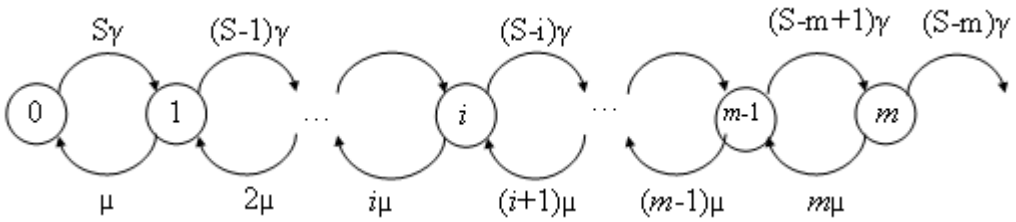

***Figura 8**. Cadena Márkov en Engset-B*

En este caso, la tasa de nacimiento es proporcional al número de usuarios libres y la tasa de muerte sigue siendo $n\mu$ (siendo n el estado y μ la tasa de servicio de un servidor).

Para obtener las probabilidades de estado, utilizaremos ecuaciones de balance de flujo parciales, donde:

$$(S - n + 1) * \gamma * P(n-1) = n * \mu * P(n)$$

En cuanto a sus ecuaciones características para su estudio tenemos:

Ecuación de normalización.

$$\sum_{i=0}^{m} P(n) = 1$$

Ecuación de probabilidad de estado.

$$P(i) = \frac{\binom{S}{i} \cdot \beta^i}{\sum_{j=0}^{m} \binom{S}{j} \cdot \beta^j}$$

En cuanto a la probabilidad de bloqueo y la probabilidad de pérdida, en el modelo de Engset son distintas y por tanto necesitamos definirlas por separado.

La probabilidad de bloqueo se expresa como:

$$PB = \frac{\binom{S}{m} \cdot \beta^m}{\sum_{j=0}^{m} \binom{S}{j} \cdot \beta^j}$$

Mientras que la probabilidad de perdida se expresa como:

$$PL = \frac{\binom{S-1}{n} \cdot \beta^n}{\sum_{j=0}^{n} \binom{S-1}{j} \cdot \beta^j} = P_{B|_{s-1}}$$

3. GRÁFICAS 2D EN LA WEB

3.1. Preliminares: JavaScript

En primer lugar, tenemos JavaScript, es un lenguaje de programación que permite la implementación de funciones complejas en las páginas web, es decir, una función en JavaScript no es compilada, sino que es interpretada por el propio navegador web, por eso se dice que JavaScript es un lenguaje de programación interpretado. Su principal ventaja es que al ser el navegador el que ejecuta la función en vez del servidor, el navegador del usuario reaccionas más rápidamente a las peticiones.

En cuanto a su uso gráfico, principalmente se usa para gráficos 2D, ya que no necesita de mucho código para su implementación. En nuestro caso podría haber valido pero no era el más óptimo ya que nuestro diseño es evolutivo, es decir queremos que la gráfica se vaya dibujando a medida que avanzamos en la simulación.

3.2. Cascading Style Sheets (CSS)

CSS es el lenguaje de diseño gráfico que define y crea la presentación de documentos estructurados como es el caso de los HTML o XHTML. Junto con HTML y JavaScript, CSS se utiliza para el diseño de páginas web visualmente atractivas e intuitivas.

Pero el problema que plantea CSS, el cual es el motivo por el que no lo hemos utilizado para el diseño de la gráfica, es que su tecnología se basa en los gráficos de mapa de bits, los cuales exportan una imagen y la cargan directamente. Esto da como resultado una imagen tipo fotográfica, la cual no es escalable ni permite ciertas modificaciones.

Debido a este motivo solo hemos utilizado CSS para hacer que el menú de la simulación quede atractivo, es decir, el posicionamiento de los contenedores así como la alineación y que el diseño web sea *responsive* (se adapte a distintos dispositivos tales como tabletas o móviles), y quede visualmente agradable.

3.3. Canvas

Es un elemento de HTML que permite la representación de gráficos usando scripts, los cuales pueden ser configurados para dar un resultado acorde a las necesidades planteadas. El método que se utiliza en los documentos HTML para la declaración de este elemento es mediante las etiquetas <canvas> </canvas>.

La razón por la cual no hemos utilizado este método es que sigue siendo un mapa de bits y plantea los mismos problemas de CSS, pero también que canvas necesita de Scripts para la representación gráfica, es decir, necesita de un tercero para su funcionamiento. Mientras que la siguiente opción que vamos a describir, el SVG, funciona por si solo y resuelve los problemas que nos plantea tanto el uso de CSS como el de Canvas.

3.4. SVG

El formato SVG es un formato vectorial que ofrece una gran flexibilidad así como una buena calidad grafica

Los gráficos aparte de ser escalables pueden ser tanto estáticos como dinámicos. El método que se utiliza en los documentos HTML para su declaración es <svg> </svg>. Su funcionamiento es muy simple: aparte de definir la anchura y altura del contenedor, lo único que se necesita para su representación son las coordenadas de inicio y fin. Con este sistema podemos representar distintos tipos de figuras a partir de la representación de elementos básicos como líneas, rectángulos, círculos, etc.

El motivo de la utilización de este método es que gracias a su sencillez y a lo cómodo que es un sistema vectorial para una representación gráfica, solo es necesario pasarle los resultados del simulador a través de un archivo de texto, y con SVG utilizamos estos resultados como coordenadas para su representación gráfica. Además, que al ser vectorial es escalable y nos soluciona el problema que teníamos con los mapas de bits, que si los ampliábamos se perdía resolución.

3.5. Gráficos 3D en la web: WebGL

WebGL (Web Graphics Library) es la tecnología más actual que permite la visualización de contenido 3D en los navegadores web. Se basa en la aceleración grafica de su ordenador para la representación de los modelos 3D con una gran calidad.

Podríamos haberla utilizado perfectamente pero como no necesitábamos de una representación en 3D al final hemos elegido SVG, el cual es óptimo para las representaciones graficas en 2D como ya hemos indicado.

4. APLICACIÓN DESARROLLADA

4.1. Diagrama de bloques

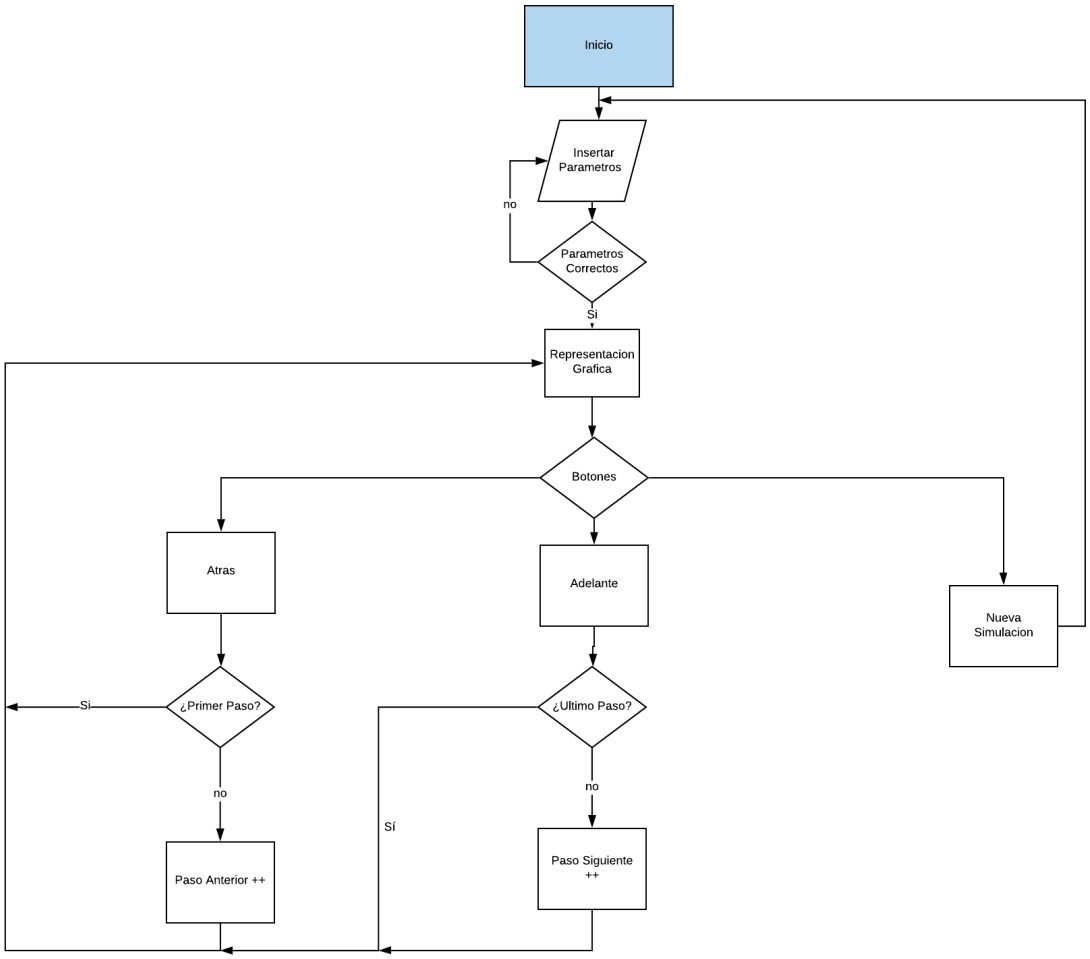

***Figura 9**. Diagrama de bloques de la web*

4.2. Vistas de la web

Empezaremos hablando por la primera vista de la aplicación, la cual aparece al entrar en la web y corresponde al fichero index.php

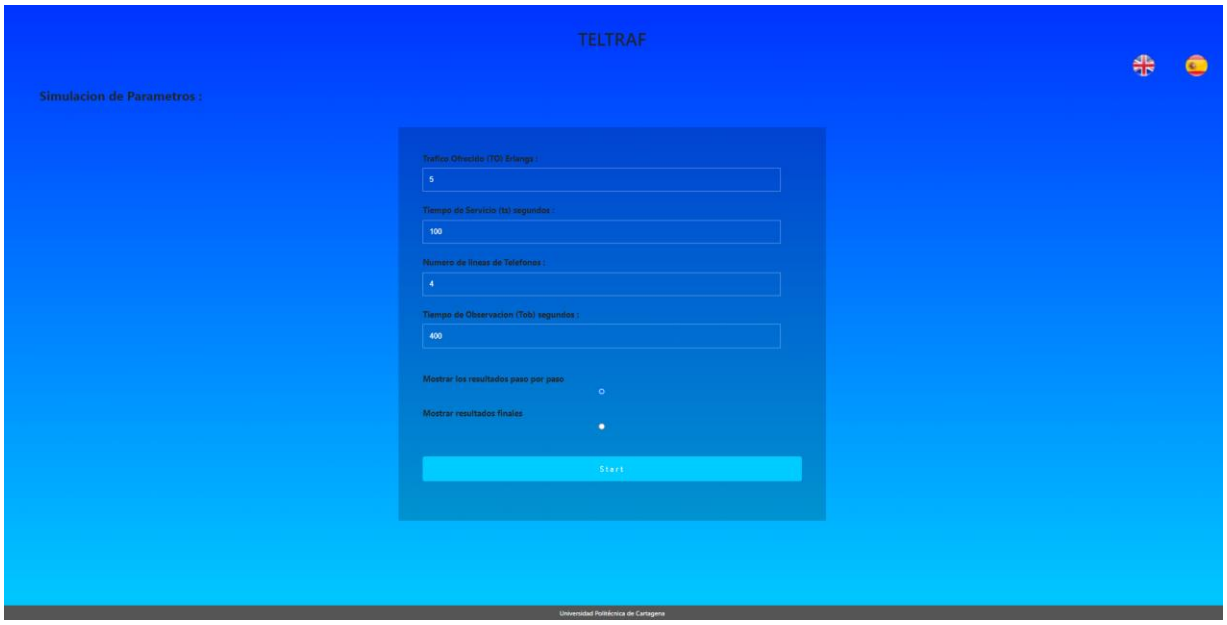

***Figura 10**. Vista de "index.php" en la web*

Como podemos ver en la figura, consiste en un formulario en el cual introduciremos los parámetros de la simulación:

- Trafico Ofrecido (TO): la cantidad de tráfico total que simularemos ofrecer.
- Tiempo de Servicio (ts): el tiempo en que está en uso una línea telefónica cuando se produce una llamada.
- Número de líneas telefónicas: el número total de canales disponibles para cursar una llamada.
- Tiempo de observación (Tobs): Tiempo total que durara nuestra simulación.

También podemos observar una serie de características que nos ofrece el programa, como el cambio de idioma entre español e inglés, o el mostrar el resultado paso a paso o directamente visualizar el resultado final.

Por otro lado, hemos establecido que los parámetros del formulario deben ser rellenados, así evitamos que se intente ejecutar la simulación en caso de que falten datos.

Cabe destacar que el estilo que se le ha dado al index.php se debe al fichero styleReg.css, el cual no solo sirve para dar color y posicionar los elementos, sino que también lo hemos usado para un diseño *responsive*, de modo que, si la aplicación se ejecutara desde un móvil, todo se ajuste a las medidas de la pantalla.

A continuación, vamos a hablar de la simulación, la cual se produce cuando le damos a “Start”. Empezaremos hablando de la simulación paso a paso. En esta opción, podemos ver en la parte superior se nos genera la gráfica con los resultados de la simulación además de tres botones,

los cuales sirven para movernos por la simulación o para ejecutar una nueva mientras que en la parte de abajo podemos visualizar cuatro cuadros de texto los cuales nos indican los resultados en cada paso de la simulación.

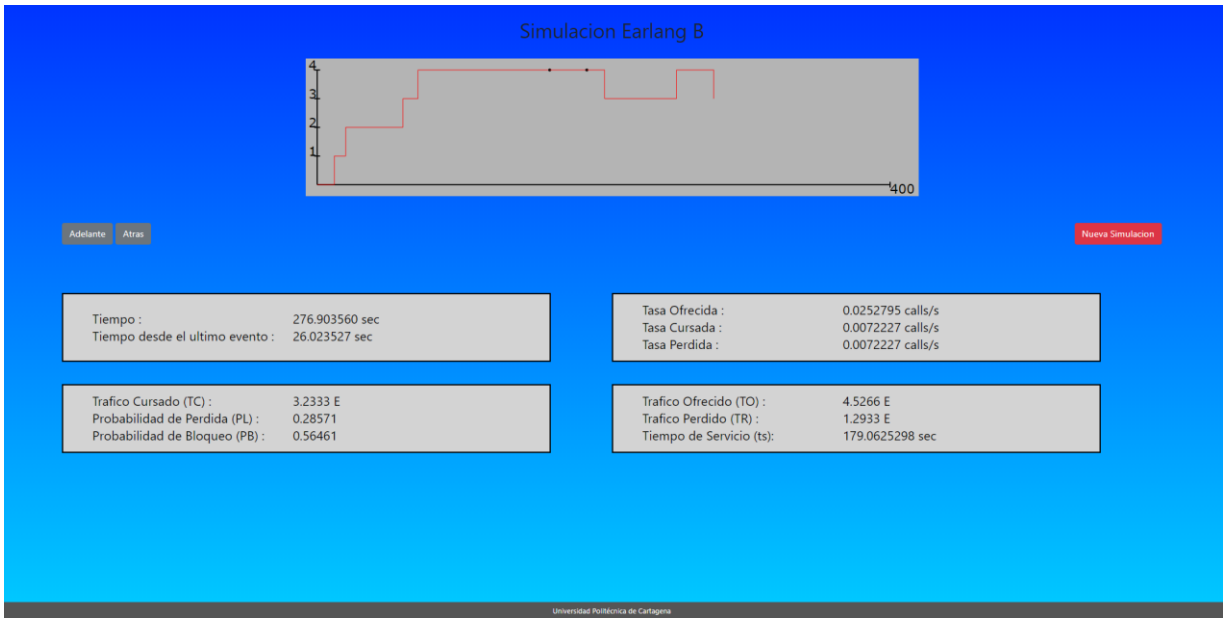

***Figura 11**. Vista de "procesar.php" en la web*

Como podemos visualizar en la imagen, el número de líneas telefónicas se representan como los escalones del eje Y, mientras que el Tiempo de observación indica el límite del eje X.

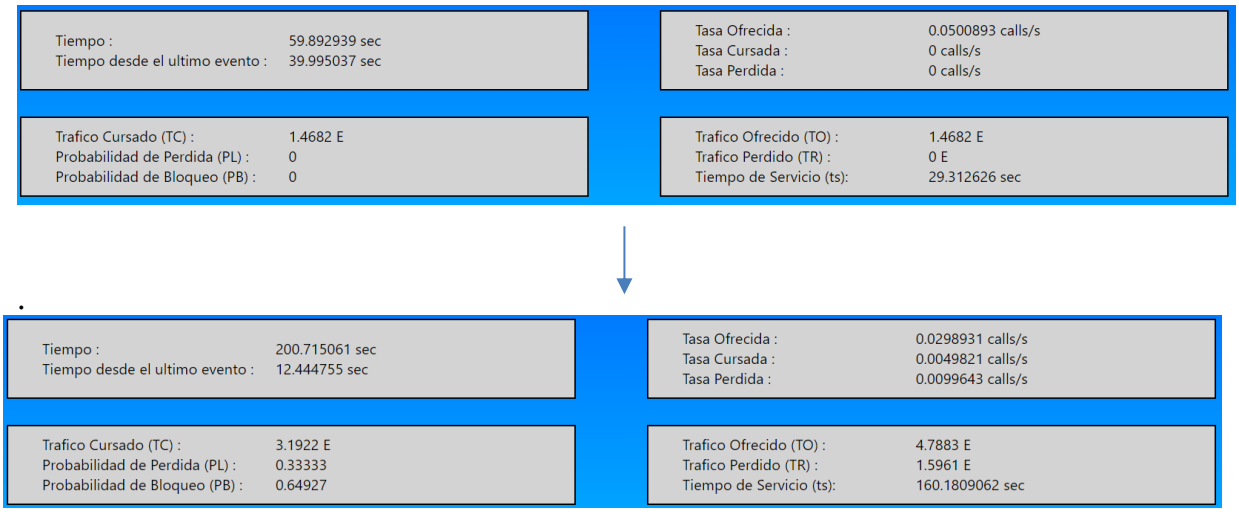

***Figura 12 y 13**. Variaciones de los cuadros de texto*

Como ya hemos indicado, si avanzamos o retrocedemos en la simulación, los valores obtenidos van variando en los cuadros de texto acorde al momento en el que se encuentra la simulación.

Cabe destacar que al haber hecho la gráfica y los cuadros de texto en SVG, su tamaño se ajusta acorde a la pantalla por lo que no hay problemas en ejecutar la aplicación desde un móvil, por ejemplo, solo hemos usado CSS por medio del archivo style.css para que quedara más ameno a la vista.

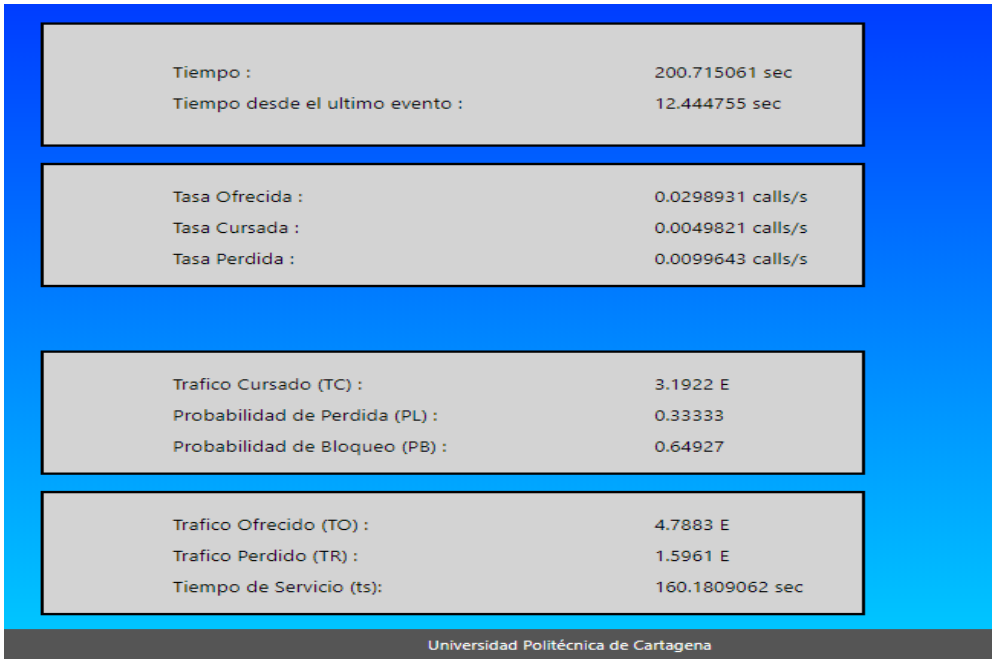

***Figura 14**. Vista desde el móvil*

Para la opción de mostrar el resultado final, directamente se nos mostrará lo mismo que en la opción paso a paso, pero con la simulación acabada, es decir, cuando hayan pasado el tiempo de observación establecido.

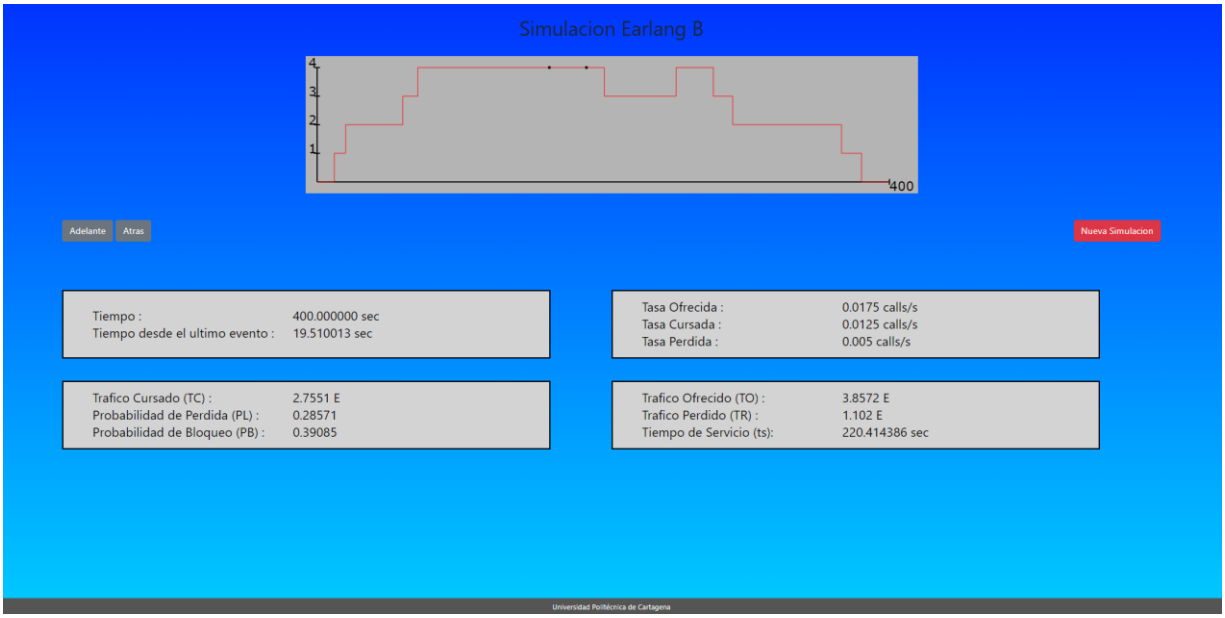

***Figura 15**. Opción mostrar resultado final*

4.3. Diseño del gráfico

Como ya hemos indicado a lo largo de este proyecto, hemos utilizado SVG a la hora de la representación gráfica. En este caso estaremos hablando de 5 elementos SVG, la gráfica y los 4 cuadros de texto.

Empezaremos hablando primero del gráfico.

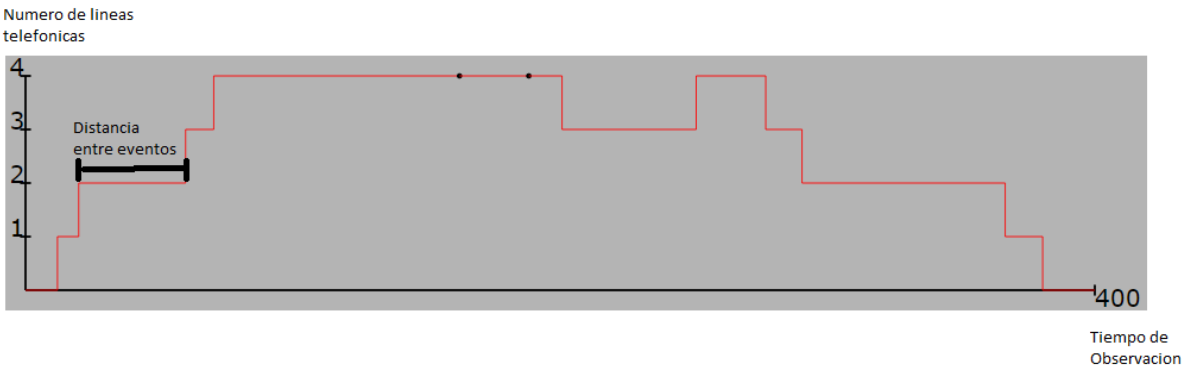

***Figura 16**. Gráfico SVG*

Vamos a describir el proceso que dibuja el grafico en nuestro programa. Primero dibuja los ejes. El eje Y divide su tamaño entre el número de líneas telefónicas que pasamos a nuestra simulación por lo que la distancia entre cada escalón será igual:

$$Distancia\_LineaTelefonica = \frac{Tamaño\_Eje\_Y}{Nº\ Lineas\_Telefonicas}$$

Por otro lado, al eje X se le asigna que el valor máximo va a ser el tiempo de observación establecido para nuestra simulación.

El funcionamiento del simulador consiste en que cada vez que se produce una llamada o finaliza una existente, se registra el nuevo estado del sistema. En base a esto el grafo se va desarrollando en función de la diferencia de tiempo entre un suceso y su anterior y el cambio de utilización de las líneas telefónicas.

Como mención especial decir que cuando todas las líneas telefónicas están en uso, si se produce un evento de llamada se produce una pérdida, el cual es representado con un punto negro en la gráfica.

A continuación, vamos a hablar de los elementos SVG que son los cuadros de texto. En este caso tenemos dos tipos: Tipo A: Solo hay un cuadro de texto que use este tipo de cuadro y es el primero, ya que es el único que tiene dos líneas de datos.

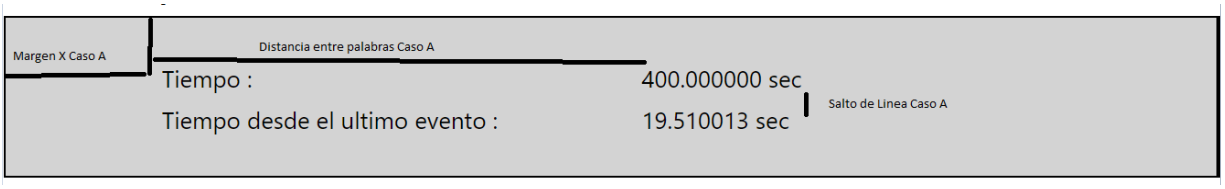

***Figura 17**. Caja de Texto A*

Estas cajas de texto son construidas en función del margen respecto al eje X e Y y a la distancia entre la línea 1 y la línea 2.

El motivo de introducir por medio de valores la distancia entre palabras es debido no solo a que los resultados son leídos de un fichero, sino que debido a la función multilenguaje de la cual hablaremos posteriormente, las palabras también son leídas de un fichero en función del idioma seleccionado, así que establecer la distancia por medio de valores numéricos resultaba óptimo para el encuadrado.

Por último, tenemos el caso de los cuadros de texto Tipo B: los cuales singuen el mismo funcionamiento que los Tipo A, pero con la diferencia de que estos tienen tres líneas y al ser los 4 cuadrados del mismo tamaño hay que cambiar los valores con el fin de que queden bien centrados.

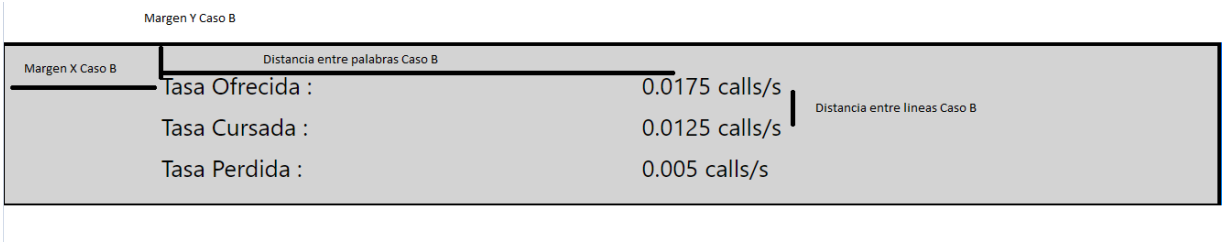

***Figura 18**. Caja de Texto B*

4.4. Realización de la web

Por último, vamos a hablar de cómo hemos utilizado las diferentes tecnologías que componen esta aplicación.

Empezaremos hablando de HTML5 el cual es la base de la web, es el elemento más básico para la construcción de la web ya que no solo aporta una estructura sobre la que empezar sino también facilita la conexión entre las distintas tecnologías implementadas, que son JavaScript, PHP, CSS y SVG.

De JavaScript no hay que comentar mucho ya que su utilización ha sido mínima, solo ha servido para ayudar a hacer el diseño *responsive* de la web.

Por otro lado, PHP ha sido el lenguaje más importante ya que se encarga de la mayoría de las funciones del programa. Este lenguaje se caracteriza por ser muy simple ya que no requiere de complicadas configuraciones para que funcione en HTML5, solo hay que incrustarla por medio de las etiquetas <?php y se cierra con?>.

Nuestro programa PHP se encarga de la recogida de parámetros del formulario, los cuales son almacenados en variables, para posteriormente ser enviadas al simulador.

```
// Only if it is this first time, generate the simulation results.
if ($run == 1) {
   system("teltraf_MMmm --to ".$TO." --ts ".$ts." --max ".$max_servers." --tob ".$Tob." > /tmp/".$fileID."_teltraf_MMmm.txt");
   $run = 0;
}
```

***Figura 19**. Ejecución del Simulador*

También se encarga de hacer los cálculos necesarios para posteriormente poder dibujar los gráficos SVG

```php
// Drawing the x axis
$line_x1 = $Mx;
$line_y1 = $My+$axis_y_length;
$line_x2 = $Mx+$axis_x_length;
$line_y2 = $My+$axis_y_length;
echo "<line x1='".$line_x1."' y1='".$line_y1."' x2='".$line_x2."' y2='".$line_y2."' stroke='black' stroke-width='2' />";

// Drawing the y axis
$line_x1 = $Mx;
$line_y1 = $My+$axis_y_length;
$line_x2 = $Mx;
$line_y2 = $My;
echo "<line x1='".$line_x1."' y1='".$line_y1."' x2='".$line_x2."' y2='".$line_y2."' stroke='black' stroke-width='2' />";

// Drawing the y marks
for ($i=1; $i<=$max_servers; $i++){
    $text_x1 = $Mx-15;
    $text_y1 = $My+$axis_y_length-$i*$Dy;
    $line_x1 = $Mx-5;
    $line_y1 = $My+$axis_y_length-$i*$Dy;
    $line_x2 = $Mx+5;
    $line_y2 = $My+$axis_y_length-$i*$Dy;
    echo "<text x='".$text_x1."' y='".$text_y1."' font-family='Verdana' font-size='10'>$i</text>";
    echo "<line x1='".$line_x1."' y1='".$line_y1."' x2='".$line_x2."' y2='".$line_y2."' stroke='black' stroke-width='2' />";
}
```

***Figura 20**. Ejemplo uso PHP en cálculos*

Y para el establecimiento del sistema multilenguaje, el cual nos permite cambiar el idioma de la web por medio de una sola variable.

Este sistema esta codificado en un archivo aparte, Lang.php y es incrustado posteriormente en el programa.

```php
<?php

session_start();

// Comprobamos la variable get

if(isset($_GET['idioma'])){
//si pasamos la variable por medio de la barra de direcciones o un link
$idioma = $_GET['idioma'];
//creamos la variable de session "lang" con el valor recibido mediante la barra de direcciones o el link
$_SESSION['idioma'] = $idioma;
//si ya existe la variable lang en nuestra sesion asignamos su valor a la ariable local $lang
}elseif(isset($_SESSION['idioma'])){
$idioma = $_SESSION['idioma'];
//de lo contrario asignamos una variable local por defecto para llamar a nuestro idioma principal o por defecto
}else{
$idioma = 'es';
}


switch ($idioma) {
case 'en':
$texto = explode("\n", file_get_contents("english.txt"));
break;
case 'es':
 $texto = explode("\n", file_get_contents("spain.txt"));
break;
default:
 $texto = explode("\n", file_get_contents("spain.txt")); //en caso de no pasar o no existir la indicacion del idioma cargar el por defecto (espanol)
break;
}

?>
```

***Figura 21**. Sistema Multilenguaje*

Consiste en establecer una sesión en PHP para mantener viva la variable del idioma, y en función de la opción señalada, gracias al comando explode ("\n", file_get contents ("")), obtenemos todas las palabras de texto del archivo correspondiente, y las introducimos por medio de variables PHP en nuestro documento, así evitamos duplicar código para cada opción de idioma.

El comando explode también es utilizado para obtener los resultados devueltos por el simulador e introducirlos en nuestra web por medio de variables PHP. Un ejemplo del fichero devuelto por el simulador seria:

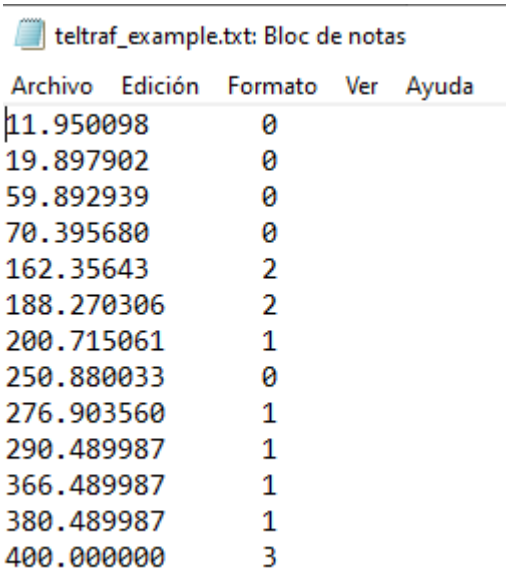

***Figura 22**. Ejemplo Txt devuelto por el simulador*

Donde la primera columna hace referencia al tiempo en el que ocurre un evento y la segunda al número de líneas telefónicas ocupadas (denotar que en nuestro simulador la primera línea ocupada al ser un array empieza en 0 pero al dibujarla en la Web lo transformamos en 1).

Por último, tenemos el archivo delete.php, el cual es invocado al dar a nueva simulación, este proceso PHP se encarga del borrado del archivo de texto generado por la simulación anterior, el motivo es eliminar todos los archivos temporales con el fin de que no se llene el disco duro de archivos zombi.

La siguiente tecnología usada en el proyecto ha sido CSS, que se utiliza para definir el aspecto base del HTML y las características del SVG. Este lenguaje está diseñado principalmente para marcar la separación del contenido del documento así como la forma de presentación, sus características como las capas o layouts, los colores y las fuentes.

Debido a la utilización de selectores y clases podemos dar diferentes características en cada parte del archivo lo cual facilita el hacerlo visiblemente agradable.

Hay diferentes formas de declarar un estilo CSS en un documento HTML, pero en este proyecto nos hemos centrado en la utilización de dos formas. La forma externa, la cual consiste en que usando archivos externos donde se incluyen los estilos agregarlos a la web por medio de “<link rel="stylesheet" href="css/styleReg.css">” y la forma interna que consiste en declarar dentro de los elementos web su diseño CSS por medio de la declaración. style=” …”

Hemos usado dos archivos .css para hacer los estilos generales: styleReg.css se encarga de implantar las bases visuales de index.html, mientras que style.css se encarga de teltraf.php.

Una vez establecidas estas bases, hemos ido acomodando cada elemento de la Web para que quedara mejor por medio de la declaración interna style=” …” ajustando visualmente los distintos elementos de la página.

Por último, tenemos SVG, el cual ya hemos ido comentando a lo largo de todo el proyecto que su utilización iba a ser para la parte gráfica, ya que la utilización de un sistema vectorial para

hacer dibujo en 2D resultaba de lo más óptima para nuestro caso. Se ha utilizado para la realización del gráfico de la red, así como los textos que representan los resultados.

4.5. Caso de Uso

Vamos a ejemplificar el uso de este programa a modo de ejemplo de los conceptos básicos del teletráfico, el régimen estacionario y población infinita.

- Conceptos Básicos: Para este caso, procedemos a configurar el simulador con:

    - Trafico ofrecido: 5 Erlangs
    - Tiempo medio de servicio: 100 segundos
    - Número de líneas: 4
    - Tiempo de observación: 400 segundos

  Con estas opciones podemos ver el comportamiento básico de un sistema de teletráfico, observamos que el máximo número de llamadas va a ser igual al número de líneas y en caso de que se dé una llamada esta seria bloqueada. También podemos calcular y comprobar que coinciden con el simulador las diferentes tasas y perdidas.

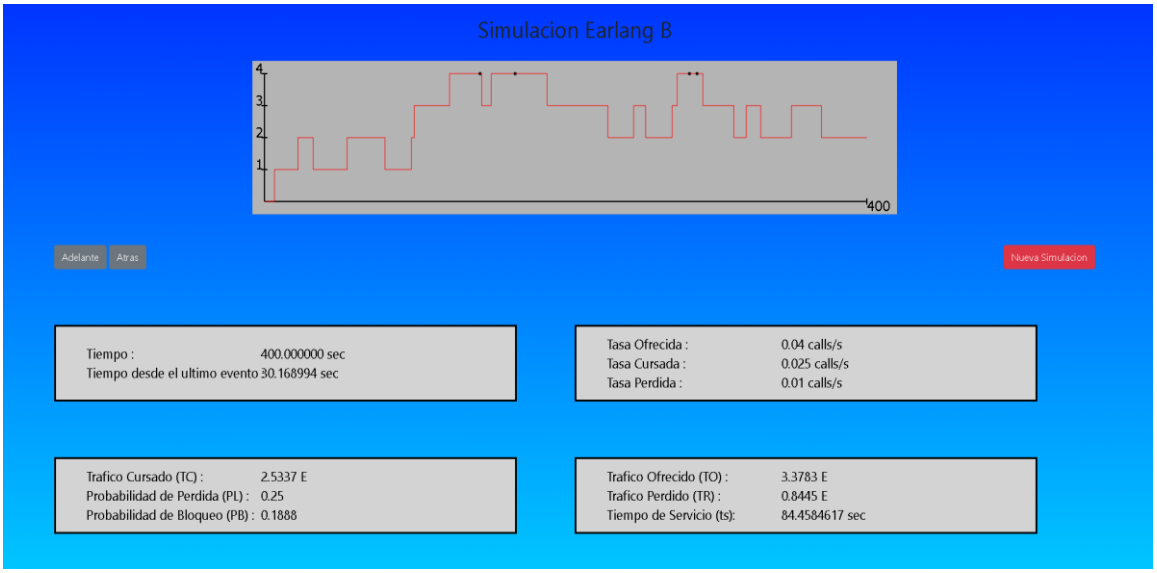

***Figura 23**. Simulación Conceptos básicos*

Para este ejemplo el tráfico cursado no puede ser nunca mayor de 4, ya que el número de líneas máximas que tenemos disponible es 4. Cuando están ocupadas todas las líneas y se produce una llamada ésta se rechaza. La probabilidad de perdida y la probabilidad de bloqueo no tienen por qué coincidir.

Procedemos a calcular el Trafico Cursado a partir de su fórmula para comprobar si el resultado coincide con el de la simulación.

$$TC = \frac{1}{T}\int_0^T n(t)dt = [(15.5 + 22.55 + 17.6) + 2 * (10.2 + 25.04 + 1.7 + 17.4 + 17.59 + 8.21 + 20.44 + 30.16) + 3 * (23.46 + 6.34 + 40.31 + 7.96 + 3.41 + 20.58 + 9.52 + 19.87) + 4 * (20.32 + 0.98 + 15.95 + 21.23 + 8.14 + 4.99 + 3.88)]/400 = 2.53 \text{ E}$$

Como podemos observar coincide con el resultado de la simulación.

- Régimen Estacionario: Modificaremos con respecto al apartado anterior el tiempo de observación para observar si el sistema se estabiliza a lo largo del tiempo. Pondremos un tiempo de observación a 3000 segundos para observar si se cumple este caso, también podemos calcular las tasas y las pérdidas para ver si coinciden con las del simulador.

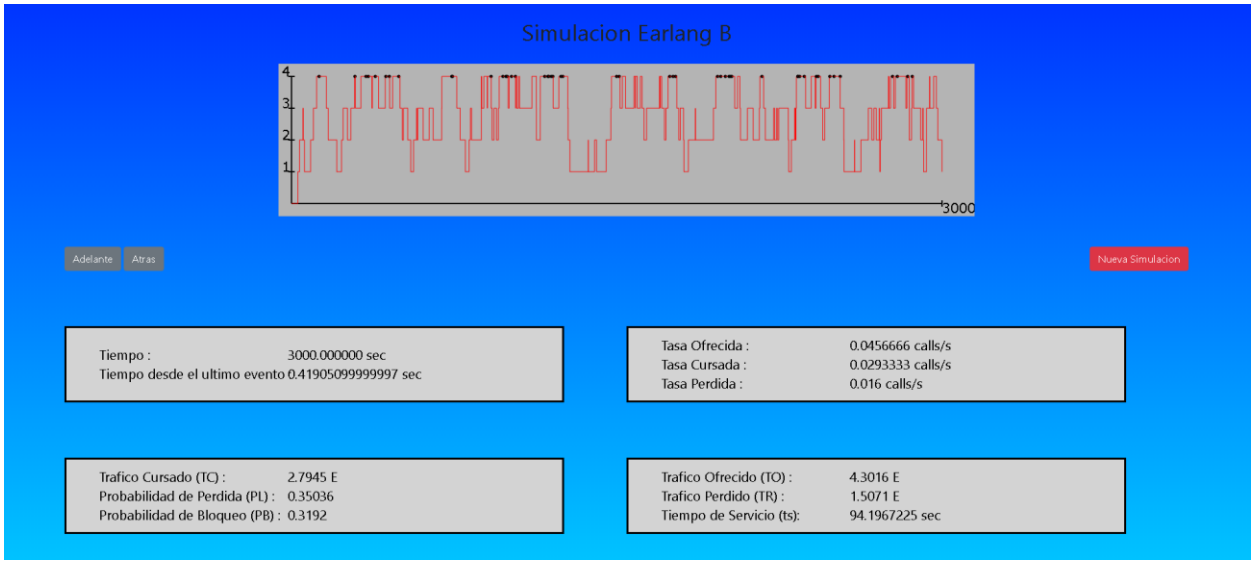

***Figura 24**. Simulación Régimen Estacionario*

Para este caso podemos comprobar que a lo largo de la simulación los valores obtenidos para las distintas tasas se van normalizando.

A su vez podemos ver que la Probabilidad de Bloqueo tiende a acercarse a la Probabilidad de Perdida.

También confirmamos

$$\text{TO} = \text{TC} + \text{TP}$$
$$\text{TP} = \text{TO} * \text{Pl}$$
$$\text{TC} = \text{TO} * (1 - \text{Pl})$$

- Aumentaremos el tiempo a 30000 segundos con respecto al apartado anterior para este estudio y marcamos la opción que nos permite hacer la simulación de un tirón.

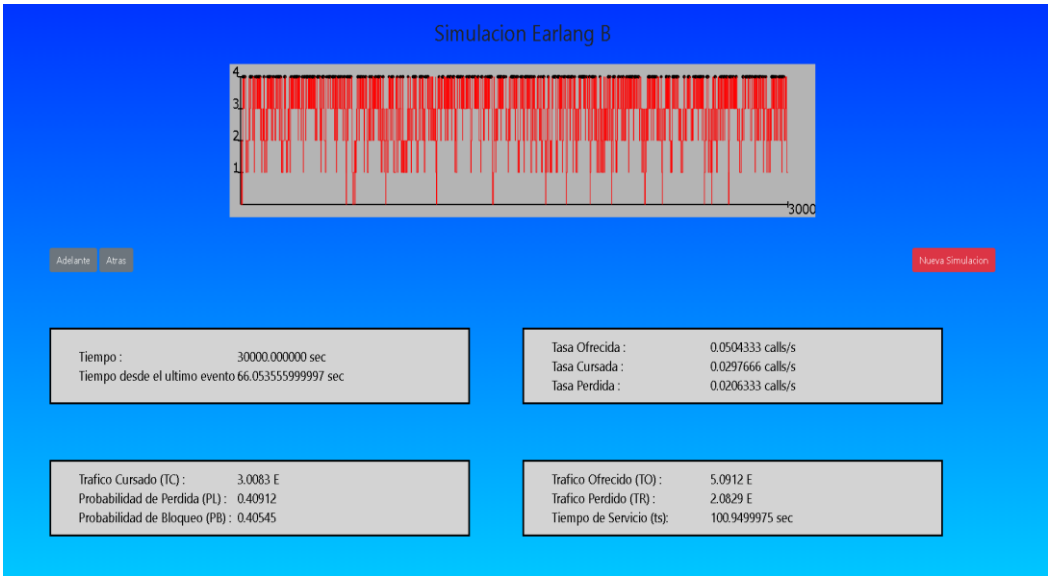

***Figura 25**. Simulación Población Infinita*

Con respecto al apartado anterior podemos ver que las probabilidades de perdida y de bloqueo se asemejan aún más.

A diferencia de los apartados anteriores el Tráfico Ofrecido y el tiempo de servicio alcanzan los valores deseados de 5E y 100 sec.

Podemos modificar el número de líneas también con el fin de comprobar si la probabilidad de bloqueo coincide con el de las tablas de Earlang-B.

5. CONCLUSIONES

En lo referente a este trabajo podemos concluir que hemos cumplido con todos los puntos propuestos del trabajo.

De cara a ciertas modificaciones algunas funciones podrían haber sido modificados para que funcionaran de forma diferente, por ejemplo, se podría haber hecho que en vez de borrar directamente la simulación una vez le damos a “Nueva Simulacion” hubiera una opción de guardarla de manera que se pudiera hacer comparaciones entre distintos casos.

Como ya vimos en los apartados teóricos tenemos distintos tipos de Modelos de Sistemas Telefónicos, nuestro programa utiliza un modelo de Erlang-B, pero debido a la facilidad de los distintos tipos de tecnologías utilizadas, se podría reprogramar en un futuro para otros modelos distintos usando como base este proyecto.

En cuanto al desarrollo del código, la parte más complicada fue la programación del simulador en lenguaje C, pero una vez desarrollado éste, el resto de la Web fue bastante más fácil.

De hecho, lo más laborioso no sería la utilización de variables en PHP ni el establecer el sistema multilenguaje. Lo que más tiempo requirió fue el aspecto gráfico, no solo el hecho de realizar los gráficos SVG sino también establecer las características por medio de CSS y hacer que la Web sea *responsive*, permitiendo abrirla desde un móvil si fuera necesario.

6.BIBLIOGRAFÍA

- Apuntes de la asignatura de Conmutación de la Universidad Politécnica de Cartagena

- Apuntes Departamento de Ingeniería de Sistemas Telemáticos UPM https://docplayer.es/40097711-Tema-5-introduccion-al-teletrafico-y-a-la-teoria-de-colas.html.

- Como hacer una página Web Multilenguaje https://www.jimdo.com/es/blog/crear-pagina-web-multilenguaje/

- Apuntes Introducción a la Teoría de Teletráfico Universidad Nacional De Colombia https://aprenderly.com/doc/3418415/introducci%C3%B3n-a-la-teor%C3%ADa-de-teletr%C3%A1fico.

- Proceso Estocástico, https://es.wikipedia.org/wiki/Proceso_estoc%C3%A1stico.

- Teoría de Colas, https://es.wikipedia.org/wiki/Teor%C3%ADa_de_colas

- Simulación por Computadora https://es.wikipedia.org/wiki/Simulaci%C3%B3n_por_computadora#:~:text=Una%20simulaci%C3%B3n%20por%20computadora%2C%20un,abstracto%20de%20un%20determinado%20sistema.

- Apuntes Ingeniería de Trafico de Telecomunicaciones Escuela Politécnica Nacional. https://bibdigital.epn.edu.ec/bitstream/15000/12048/1/Ingenier%C3%ADa%20de%20Tr%C3%A1fico%20HWCR.pdf

- Apuntes Teletráfico Universidad de Cantabria https://ocw.unican.es/pluginfile.php/1955/course/section/2254/Tema2_Trafico.pdf.

- Información sobre Telefonía Móvil, https://www.areatecnologia.com/telefonia-movil.htm

- Red de Celdas, https://es.wikipedia.org/wiki/Red_de_celdas.

- Unidad y Fórmula de Erlang https://es.wikipedia.org/wiki/Unidad_Erlang#F%C3%B3rmula_Engset.

- Distribuciones de Probabilidad https://www.sergas.es/Saudepublica/Documents/1899/Ayuda_Epidat_4_Distribuciones_de_probabilidad_Octubre2014.pdf.

- Comportamiento JavaScript Gráficos 2D https://www.purosoftware.com/desarrollo-web-scripts-graficos-estadisticos/06-javascript-chart.html

- JavaScript, https://es.wikipedia.org/wiki/JavaScript

- Hoja de Estilos en Cascada
  https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada

- Tutorial de Canvas
  https://developer.mozilla.org/es/docs/Web/API/Canvas_API/Tutorial

- Qué es y qué ventajas tiene SVG
  https://graffica.info/formato-svg-ventajas/

- Qué es WebGL
  http://learn.sculpteo.com/qu%C3%A9-es-webgl-c%C3%B3mo-se-activa-webgl.

- Adaptar CSS https://www.solucionex.com/blog/adaptar-un-texto-un-contenedor-con-css

- Selectores y Pseudoclases
  https://www.yunbitsoftware.com/blog/2017/12/14/selectores-y-pseudo-clases-css/.

- Diseño Responsive https://www.genbeta.com/desarrollo/responsive-design-estructura-adaptable.

7. ANEXO

7.1. Código web: Index.php

```
<!DOCTYPE html>
<?php include("lang.php"); ?>
<html>
<head>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
      integrity="sha384-
      Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
      crossorigin="anonymous">
    <link rel="stylesheet" href="css/styleReg.css">
    <link rel="stylesheet" href="css/bootstrap.min.css">
   <meta charset="UTF-8">
   <meta name="escala" content="width=device-width, initial-scale=1.0">

</head>
<body>

   <div class="main-w3layouts wrapper">
     <header>
      <h2 style="text-align: center; font-size: 35px">
         TELTRAF<br>
      </h2>
         <ul style="list-style:none;">
            <li><a href="index.php?idioma=en"><img
      src="icon/united_kingdom_round_icon_64.png"></a></li>
            <li><a href="index.php?idioma=es"><img src="icon/spain_round_icon_64.png"></a></li>
         </ul>

         <h3>
         <br>
         <br>
         <b style="font-size:25px; margin: 75px"><?php echo $texto[0]; ?>:</b></h3>
     </header>
      <div class="main-agileinfo">
         <div class="agileits-top">
            <form action="teltraf.php?" method="GET">

               <?php

               if (isset($_GET['TO'], $_GET['ts'], $_GET['max_servers'], $_GET['Tob'])){

                  $TO = $_GET['TO'];
                  $ts = $_GET['ts'];
                  $max_servers = $_GET['max_servers'];
                  $Tob = $_GET['Tob'];


               }
               else {

                  $TO = 5;
                  $ts = 100;
                  $max_servers = 4;
                  $Tob = 400;
```

```
                }

                echo '<b><label for="TO">'.$texto[1].':</label></b>';
                echo '<input name="TO" type="text" id="TO" min="0" value="'.$TO.'" size="3" required>
            <br>';

                echo '<b><label for="ts">'.$texto[2].':</label></b>';
                echo '<input name="ts" type="number" id="ts" min="0" value="'.$ts.'" size="4" required><br>';



                echo '<b><label for="max_servers">'.$texto[3].':</label></b>';
                echo '<input name="max_servers" type="number" id="max_servers" min="1"
            value="'.$max_servers.'" size="2" required><br>';

                echo '<b><label for="Tob">'.$texto[4].':</label></b>';
                echo '<input name="Tob" type="number" id="Tob" min="1" value="'.$Tob.'" size="5"
            required><br>';

                echo "<br>";

                echo '<b><label for="show_step">'.$texto[5].'</label></b>';
                echo '<input type="radio" name="show" id="show_step" value="step" checked><br>';

                echo '<b><label for="show_final">'.$texto[6].'</label></b>';
                echo '<input type="radio" name="show" id="show_final" value="final"><br>';
              ?>
                <input value="Start" class="btn btn-primary" type="submit">
            <input name="step" class="form-control" type="hidden" value="0">
            <input name="run" class="form-control" type="hidden" value="1">
            <?php
            $fileID=rand();
            echo '<input name="fileID" type="hidden" value="'.$fileID.'">';
            ?>
                  </form>
                </div>
            </div>
          </div>
        </body>
        <footer class="container-fluid text-center" style="bottom:0px ;
        background-color: #555;width:100%; color: white; padding: 7px;position:fixed;
        max-height:50px; z-index:999 ; zoom:79%">
        <p class="d-inline">Universidad Politécnica de Cartagena</p>
         </footer>
</html>
```

7.2. Código web: teltraf.php

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
<link rel="stylesheet" type="text/css" media="all" href="css/style.css" />
<link rel="stylesheet" href="css/bootstrap.min.css">
<meta charset="UTF-8">
<meta name="escala" content="width=device-width, initial-scale=1.0">


</head>
<body style="margin:0;">
   <?php include("lang.php"); ?>
   <br>
          <div class="wrapper"><?php echo $texto[7]; ?></div>
   <br>

<?php
$step = $_GET['step'];
$TO = $_GET['TO'];
$ts = $_GET['ts'];
$max_servers = $_GET['max_servers'];
$fileID = $_GET['fileID'];
$Tob = $_GET['Tob'];
$run = $_GET['run'];
$show = $_GET['show'];

// Number of offered, carried and lost calls
$offered = 0;
$carried = 0;
$lost    = 0;

// offered, carried and lost rates
$rate_o = 0;
$rate_c = 0;
$rate_r = 0;

// offered, carried and lost measured traffics
$TC_m = 0;
$TO_m = 0;
$TR_m = 0;

// Volume of traffic
$Vol = 0;

// Measured service time
$ts_m = 0;

// Blocking and Lost probability
$blocking_time = 0;
$PB = 0;
$PL = 0;


// system state. From 0 up to max_servers
$state = 0;
```

```
// x and y axes margins
$Mx = 20;
$My = 20;

// x axis step
$Dy = 50;

// Position of box
$Bx = 50;
$By = 35;

// Only if it is this first time, generate the simulation results.
if ($run == 1) {
   system("teltraf_MMmm --to ".$TO." --ts ".$ts." --max ".$max_servers." --tob ".$Tob." >
/tmp/".$fileID."_teltraf_MMmm.txt");
   $run = 0;
}


// SVG root element
$axis_x_length = 1000;
$axis_y_length = $Dy*$max_servers;
$svg_width = 2*$Mx+$axis_x_length+30;
$svg_height = 2*$My+$axis_y_length;
echo "<svg id='teltraf' x='0' y='0' width='".$svg_width."' height='".$svg_height."'>";
echo "<rect x='0' y='0' width='".$svg_width."' height='".$svg_height."' fill='rgb(180, 180, 180)'/>";

// Drawing the x axis
$line_x1 = $Mx;
$line_y1 = $My+$axis_y_length;
$line_x2 = $Mx+$axis_x_length;
$line_y2 = $My+$axis_y_length;
echo "<line x1='".$line_x1."' y1='".$line_y1."' x2='".$line_x2."' y2='".$line_y2."' stroke='black' stroke-width='2'
/>";

// Drawing the y axis
$line_x1 = $Mx;
$line_y1 = $My+$axis_y_length;
$line_x2 = $Mx;
$line_y2 = $My;
echo "<line x1='".$line_x1."' y1='".$line_y1."' x2='".$line_x2."' y2='".$line_y2."' stroke='black' stroke-width='2'
/>";

// Drawing the y marks
for ($i=1; $i<=$max_servers; $i++){
   $text_x1 = $Mx-15;
   $text_y1 = $My+$axis_y_length-$i*$Dy;
   $line_x1 = $Mx-5;
   $line_y1 = $My+$axis_y_length-$i*$Dy;
   $line_x2 = $Mx+5;
   $line_y2 = $My+$axis_y_length-$i*$Dy;
   echo "<text x='".$text_x1."' y='".$text_y1."' font-family='Verdana' font-size='10'>$i</text>";
   echo "<line x1='".$line_x1."' y1='".$line_y1."' x2='".$line_x2."' y2='".$line_y2."' stroke='black' stroke-
width='2' />";
}

// Drawing the x marks
$text_x1 = $Mx + $axis_x_length;
$text_y1 = $My + $axis_y_length + 15;
```

```
$line_x1 = $Mx + $axis_x_length;
$line_y1 = $My + $axis_y_length + 5;
$line_x2 = $line_x1;
$line_y2 = $My + $axis_y_length - 5;
echo "<text x='".$text_x1."' y='".$text_y1."' font-family='Verdana' font-size='10'>$Tob</text>";
echo "<line x1='".$line_x1."' y1='".$line_y1."' x2='".$line_x2."' y2='".$line_y2."' stroke='black' stroke-width='2'
/>";


// Drawing the graph
$line_x1 = $Mx;
$line_y1 = $My + $axis_y_length;
$line_x2 = $Mx;
$line_y2 = $My + $axis_y_length;

$sim_results = explode("\n",file_get_contents("/tmp/".$fileID."_teltraf_MMmm.txt"));
$num_events = sizeof($sim_results)-1;

if ($show == "final") $step = $num_events-1;

for ($i=0; $i <= $step; $i++){
   $tmp = explode("\t", $sim_results[$i]);
   $time  = $tmp[0];
   $event = $tmp[1];

   $line_x2 = $Mx + $time*$axis_x_length/$Tob;
   echo "<line x1='".$line_x1."' y1='".$line_y1."' x2='".$line_x2."' y2='".$line_y2."' stroke='red' stroke-width='1'
/>";
   $time_since_last_event = ($line_x2 - $line_x1)*$Tob/$axis_x_length;
   if ($state == $max_servers) $blocking_time += $time_since_last_event;
   $Vol = $Vol + $state*$time_since_last_event;

   $line_x1 = $line_x2;
   if ($event == 0) {
     $offered++;
     if ($state < $max_servers) $state++;
     $line_y2 = $line_y2 - $Dy;
     echo "<line x1='".$line_x1."' y1='".$line_y1."' x2='".$line_x2."' y2='".$line_y2."' stroke='red' stroke-
width='1' />";
   }
   elseif ($event == 1) {
     $carried++;
     if ($state > 0) $state--;
     $line_y2 = $line_y2 + $Dy;
     echo "<line x1='".$line_x1."' y1='".$line_y1."' x2='".$line_x2."' y2='".$line_y2."' stroke='red' stroke-
width='1' />";
   }
   elseif ($event == 2){
     $offered++;
     $lost++;
     echo '<circle cx='.$line_x2.' cy='.$line_y2.' r="2" stroke="black" stroke-width="1" fill="black"/>';
   }
   $line_y1 = $line_y2;

}
if ($step == 0){
  $step_f = 1;
  $step_b = 0;
}
elseif ($step == ($num_events-1)) {
```

```
  $step_f = $step;
  $step_b = $step - 1;
}
else {
  $step_f = $step + 1;
  $step_b = $step - 1;
}

// Printing results
$rate_o = $offered/$time;
$rate_c = $carried/$time;
$rate_r = $lost/$time;
$ts_m = $Vol/($offered-$lost);
$TC_m = $Vol/$time;
$TO_m = $rate_o*$ts_m;
$TR_m = $rate_r*$ts_m;
$PL = $lost/$offered;
$PB = $blocking_time/$time;
$tmp = $time_since_last_event;

//Axis for text

//Case 2 phrases

$margin_x1 = 50;
$distance_word = 350;
$margin_y1 = 50;
$distance_lane = 30;


$lanex1 =  $margin_x1;
$lanex2 =  $margin_x1 + $distance_word;
$laney1 =  $margin_y1 = 50;
$laney2 =  $margin_y1 = 50 + $distance_lane = 30;

//Case 3 phrases

$margin_y3 = 35;
$distance_lane2 = 30;

$laney3 = $margin_y3;
$laney4 = $margin_y3 + $distance_lane2;
$laney5 = $margin_y3 + 2*$distance_lane2;




echo "</svg>";
echo "<br>";
echo "<br>";
echo '<div style="margin-left:5%;margin-right:5%;">';
echo "<a
href='teltraf.php?TO=".$TO."&ts=".$ts."&max_servers=".$max_servers."&step=".$step_f."&Tob=".$Tob."&run
=".$run."&fileID=".$fileID."&show=step'>";
echo '<button type="button" class="btn btn-secondary">'.$texto[8].'</button></a>';
echo "  ";

echo "<a
```

```
href='teltraf.php?TO=".$TO."&ts=".$ts."&max_servers=".$max_servers."&step=".$step_b."&Tob=".$Tob."&run
=".$run."&fileID=".$fileID."&show=step'>";
echo '<button type="button" class="btn btn-secondary">'.$texto[9].'</button></a>';
echo "  ";

echo "<a
href='index.php?TO=".$TO."&ts=".$ts."&max_servers=".$max_servers."&Tob=".$Tob."&fileID=".$fileID."'
style='float:right;'>";
echo '<button type="button" class="btn btn-danger">'.$texto[10].'</button></a>';
echo "</div>";

echo "<br>";
   echo "<br>";
   echo "<br>";
echo "<svg width='40%' height='120' style='background-color: #D3D3D3; margin-left:5%; margin-top:15;
border: 3px solid black;'>";
echo "<text x='".$lanex1."' y='".$laney1."'>".$texto[11].":</text><text x='".$lanex2."' y='".$laney1."'> $time sec
</text>";
echo "<text x='".$lanex1."' y='".$laney2."'>".$texto[12].":</text><text x='".$lanex2."' y='".$laney2."'> $tmp sec
</text> ";
echo"</svg>";

$rate_o = floor($rate_o*10000000)/10000000;
$rate_c = floor($rate_c*10000000)/10000000;
$rate_r = floor($rate_r*10000000)/10000000;

echo "<svg width='40%' height='120' style='background-color: #D3D3D3; margin-left:5%; margin-top:15;
border: 3px solid black;'>";
echo "<text x='".$lanex1."' y='".$laney3."'>".$texto[13].":</text><text x='".$lanex2."' y='".$laney3."'> $rate_o
calls/s</text>" ;
echo "<text x='".$lanex1."' y='".$laney4."'>".$texto[14].":</text><text x='".$lanex2."' y='".$laney4."'> $rate_c
calls/s</text>" ;
echo "<text x='".$lanex1."' y='".$laney5."'>".$texto[15].":</text><text x='".$lanex2."' y='".$laney5."'> $rate_r
calls/s</text>" ;
echo"</svg>";

$TC_m = floor($TC_m*10000)/10000;
$TO_m = floor($TO_m*10000)/10000;
$TR_m = floor($TR_m*10000)/10000;

echo "<br>";
echo "<br>";

echo "<svg width='40%' height='120' style='background-color: #D3D3D3; margin-left:5%; margin-top:15;
border: 3px solid black;'>";
echo "<text x='".$lanex1."' y='".$laney3."'>".$texto[16].":</text><text x='".$lanex2."' y='".$laney3."'> $TC_m E
</text>";
$PL = floor($PL*100000)/100000;
$PB = floor($PB*100000)/100000;
echo "<text x='".$lanex1."' y='".$laney4."'>".$texto[17].":</text><text x='".$lanex2."' y='".$laney4."'>
$PL</text>";
echo "<text x='".$lanex1."' y='".$laney5."'>".$texto[18].":</text><text x='".$lanex2."' y='".$laney5."'>
$PB</text>";
echo"</svg>";

echo "<svg width='40%' height='120' style='background-color: #D3D3D3; margin-left:5%; margin-top:15;
border: 3px solid black;'>";
echo "<text x='".$lanex1."' y='".$laney3."'>".$texto[19].":</text><text x='".$lanex2."' y='".$laney3."'> $TO_m E
</text>";
echo "<text x='".$lanex1."' y='".$laney4."'>".$texto[20].":</text><text x='".$lanex2."' y='".$laney4."'> $TR_m E
```

```
</text> ";
$ts_m = floor($ts_m*10000000)/10000000;
echo "<text x='".$lanex1."' y='".$laney5."'>".$texto[21].":</text><text x='".$lanex2."' y='".$laney5."'> $ts_m
sec</text> ";
echo"</svg>";
echo "<br>";
echo "<br>";
?>


</body>
<footer class="container-fluid text-center" style="bottom:0px ;
background-color: #555;width:100%; color: white; padding: 7px;position:fixed;
max-height:50px; z-index:999 ; zoom:79%">
<p class="d-inline">Universidad Politécnica de Cartagena</p>
 </footer>
</html>
```

7.3. Código web: Lang.php

```
<?php

session_start();

// Comprobamos la variable get

if(isset($_GET['idioma'])){
//si pasamos la variable por medio de la barra de direcciones o un link
$idioma = $_GET['idioma'];
//creamos la variable de session "lang" con el valor recibido mediante la barra de direcciones o el link
$_SESSION['idioma'] = $idioma;
//si ya existe la variable lang en nuestra sesion asignamos su valor a la ariable local $lang
}elseif(isset($_SESSION['idioma'])){
$idioma = $_SESSION['idioma'];
//de lo contrario asignamos una variable local por defecto para llamar a nuestro idioma principal o por
    defecto
}else{
$idioma = 'es';
}



switch ($idioma) {
case 'en':
$texto = explode("\n", file_get_contents("english.txt"));
break;
case 'es':
 $texto = explode("\n", file_get_contents("spain.txt"));
break;
default:
 $texto = explode("\n", file_get_contents("spain.txt")); //en caso de no pasar o no existir la indicacion del
    idioma cargar el por defecto (espanol)
break;
}

?>
```

7.4. Código web: delete.php

```
<?php
        $fileID = $_GET['fileID'];
        $TO = $_GET['TO'];
        $ts = $_GET['ts'];
        $max_servers = $_GET['max_servers'];
        $Tob = $_GET['Tob'];

        system("rm /tmp/".$fileID."_teltraf_MMmm.txt");
        header("Location: ./index.php?TO=".$TO."&ts=".$ts."&max_servers=".$max_servers."&Tob=".$Tob."");
?>
```

7.5. Código web: styleReg.css

```
header{
   width: 100%;
}

ul{
   margin: 0;
   list-style: none;
   padding: 0;
   display: flex;
   float: right;
}

li a{
   padding: 20px;
   text-align: right;
}

 body {
   background: #00CCFF;
   background: -webkit-linear-gradient(to top, #00CCFF, #0033FF);
   background: -moz-linear-gradient(to top, #00CCFF, #0033FF);
   background: -o-linear-gradient(to top, #00CCFF, #0033FF);
   background: linear-gradient(to top, #00CCFF, #0033FF);
   background-size: cover;
   background-attachment: fixed;
   font-family: 'Roboto', sans-serif;
 }

 h1 {
   font-size: 3em;
   text-align: center;
   color: #fff;
   font-weight: 100;
   text-transform: capitalize;
   letter-spacing: 4px;
   font-family: 'Roboto', sans-serif;
 }

 /*-- DENTRO DE BODY --*/
 .main-w3layouts {
   padding: 3em 0 1em;
 }

 .main-agileinfo {
   width: 35%;
   margin: 3em auto;
   background: rgba(0, 0, 0, 0.18);
   background-size: cover;
 }

 .agileits-top {
   padding: 3em;
 }

 form ul li {
     color: #fff;
     font-weight: 100;
     font-size: 0.9em;
```

```
}

input[type="text"], input[type="number"], input[type="radio"] {
  font-size: 0.9em;
  color: #fff;
  font-weight: 100;
  width: 94.5%;
  display: block;
  border: none;
  padding: 0.8em;
  border: solid 1px rgba(255, 255, 255, 0.37);
  -webkit-transition: all 0.3s cubic-bezier(0.64, 0.09, 0.08, 1);
  transition: all 0.3s cubic-bezier(0.64, 0.09, 0.08, 1);
  background: -webkit-linear-gradient(top, rgba(255, 255, 255, 0) 96%, #fff 4%);
  background: linear-gradient(to bottom, rgba(255, 255, 255, 0) 96%, #fff 4%);
  background-position: -800px 0;
  background-size: 100%;
  background-repeat: no-repeat;
  color: #fff;
  font-family: 'Roboto', sans-serif;
}

input.text, input.radio, input.number{
  margin: 2em 0;
}
 input[type="submit"] {
  font-size: .9em;
  color: #fff;
  background: #00CCFF;
  outline: none;
  border: 1px solid #00CCFF;
  cursor: pointer;
  padding: 0.9em;
  -webkit-appearance: none;
  width: 100%;
  margin: 2em 0;
  letter-spacing: 4px;
}

input[type="submit"]:hover {
  -webkit-transition: .5s all;
  -moz-transition: .5s all;
  -o-transition: .5s all;
  -ms-transition: .5s all;
  transition: .5s all;
  background: #0033FF;
}

/*-- diseño responsive --*/
@media(max-width:768px) {

    body{
        height: auto;
        width: auto;
    }
    label{
        font-size: calc(0.60em + 0.60vw);
    }

}
```

7.6. Código web: style.css

```
 body {
   background: #00CCFF;
   background: -webkit-linear-gradient(to top, #00CCFF, #0033FF);
   background: -moz-linear-gradient(to top, #00CCFF, #0033FF);
   background: -o-linear-gradient(to top, #00CCFF, #0033FF);
   background: linear-gradient(to top, #00CCFF, #0033FF);
   background-size: cover;
   background-attachment: fixed;
   font-family: 'Roboto', sans-serif;
     margin: 0;
     padding: 0;
 }

.wrapper {
  max-width: 1440px;
  margin: auto;
  font-size: 35px;
  position: relative;
  text-align: center;
  float: auto;

}
.left{
    float: left;

}
.right{
    float: right;

}
text{
  max-width: 1440px;
  margin: auto;
  font-size: 20px;
  position: relative;
  text-align: center;
  float: auto;
}
.btn btn-danger{
    text-align: right;
    }
#teltraf{
 display:flex;
```

```
 position: relative;
 vertical-align: middle;
   margin: auto;

}

svg text{
   font-size: calc(0.60em + 0.60vw);
}

rect{
   background-color: inherit;
}

  @media(max-width:1440px) {

     body{
        height: auto;
        width: auto;
     }
     text{
        display: calc(0.60em + 0.60vw);
        height: auto;
        width: auto;
     }
     #teltraf{
        width: 100%;
     }
     #teltraf text line{
       height: auto;
       width: auto;
       font-size: calc(0.60em + 0.60vw);
     }

     svg{
      margin:0px;
      padding:0px;
      padding-left:5%;
      display:block;
      float:none;
      text-align:left;
      width: 70%;
     }

}
```

7.7. Codigo Simulador: teltraf_MMmm.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#include "rand.h"

void help(void);

int main(int argc, char** argv)
{
   int n=0, max_m, type;
   double lambda, mu, rate, tob, tobmax, p, to, ts;

   if (semillas(MAX_SEEDS) != 0) exit (-1);

   to = 5;
   ts = 60;
   max_m = 4;
   tobmax = 400;
   tob= 0;

   while(1){
    int c;
    static struct option long_options[] =
    {
      {"help", no_argument, NULL, 'h'},
      {"to", required_argument, NULL, 'o'},
      {"ts", required_argument, NULL, 's'},
      {"max", required_argument, NULL, 'm'},
      {"tob", required_argument, NULL, 't'},
      {0, 0, 0, 0}
    };
    c = getopt_long(argc, argv, "ho:s:m:t:", long_options, (int *)0);

    if (c == EOF) break;
      switch(c){
        case 'h': help();
        case 'o': to = atof(optarg);
                  break;
        case 's': ts = atof(optarg);
                  break;
        case 'm': max_m = atoi(optarg);
                  break;
```

```
        case 't': tobmax = atof(optarg);
                break;
        default : fputs("An error ocurred, try --help\n", stderr);
                exit(1);
      }
   }

   lambda = to/ts;
   mu = 1/ts;

   while (tob < tobmax){
     rate = lambda + n*mu;
     p = lambda/(lambda + n*mu);
     tob += expo(1/rate, 0);


     if (ranf(1) < p) type = 0;
     else type =1;

     if (n == max_m && type == 0) type = 2;

     if (tob < tobmax) printf("%lf\t%d\n", tob, type);
     else printf("%f\t3\n", tobmax);

     if (type == 0 && n<max_m) n++;
     if (type == 1 && n>0)     n--;
   }
   return 0;
}


void help(void)
{
  fprintf(stderr, "--help/-h  help\n");
  fprintf(stderr, "--to offered traffic (default 120 E)\n");
  fprintf(stderr, "--ts service time (default 100 seconds)\n");
  fprintf(stderr, "--max number of servers (default 6)\n");
  fprintf(stderr, "--tob observation time (default 400 seconds)\n");

  exit(-1);
       }
```