UNIVERSIDAD DE ALCALÁ
ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA TELEMÁTICA

Trabajo Fin de Grado
Desarrollo de un simulador de eventos discretos en Python

Autor: Alejandro Cabello Fernández

Director/es: José Manuel Giménez Guzmán

TRIBUNAL

Presidente **Miguel Ángel López Carmona**

Vocal 1º **Enrique de la Hoz de la Hoz**

Vocal 2º **José Manuel Giménez Guzmán**

CALIFICACIÓN:

FECHA:

Tabla de contenido

Índice de ilustraciones

Índice de tablas

Resumen en castellano

Hay muchos tipos de simuladores de eventos discretos disponibles en la actualidad. El problema es que la mayoría son demasiado complejos, incluso para realizar una simulación sencilla.

Realizaremos un simulador inspirado en SMPL descrito por MacDougall en "*Simulating Computer Systems: Tecniques and Tools*". Este simulador consiste en una serie de funciones en lenguaje C para modelar y simular eventos discretos.

El objetivo de este trabajo es realizar un simulador de eventos discretos, programado en Python, cuyo uso sea sencillo para el usuario, y, a su vez, con una mayor modularidad y productividad.

Realizaremos el simulador también en Visual Basic y C# para hacer una comparativa de usabilidad y tiempos de ejecución.

Summary in English

There are many kinds of discrete event simulators available nowadays. The problem is that most of them are too complicated, even for making a simple simulation.

We are going to make a simulator inspired on SMPL and described by MacDougall in his book "Simulating Computer Systems: Tecniques and Tools". That simulator consists of a serie of functions in C language in order to modelate and simulate discrete events.

The aim of this project is to make a discrete event simulator, programmed in Python, whose use would be easy for the user and also should have more modularity and productivity.

We are going to make the simulator in Visual Basic and C# with the purpose of comparing the usability as well as their running times.

Palabras clave

Eventos
Tiempo discreto
Simulador
Python
Probabilidad
M/M/1
WiFi

Resumen extendido

La simulación se ha convertido en una herramienta clave para el diseño y análisis de sistemas, como pueden ser las redes de comunicaciones o las redes de colas. Gracias a la simulación, podemos optimizar los sistemas, o elegir un sistema u otro en función de los resultados obtenidos tras la simulación.
Podemos obtener la carga máxima que aceptará un sistema, pudiendo ajustarlo de manera bastante aproximada antes de implantar el sistema.
Se pueden prevenir errores gracias a la simulación, que, de otra manera solo se producirían con el sistema ya implantado.

En resumen el objetivo del simulador consiste en medir las prestaciones o el rendimiento de un determinado sistema. En este proyecto, además queremos que un simulador también sea sencillo de usar, polivalente y actual, y por estos motivos hemos decidido realizar este proyecto, "Desarrollo de un simulador de eventos discretos en Python".

Inspirándonos en el simulador SMPL descrito por MacDougall en "*Simulating Computer Systems: Tecniques and Tools*", desarrollado en C, le hemos quitado alguna funcionalidad para hacerlo más sencillo de utilizar, sin que pierda la funcionalidad básica.
SMPL es un simulador con muchos años de historia, ya que fue creado en 1987, y a día de hoy gracias a su eficiencia y simplicidad, todavía se usa.
Es una evidencia que desde el año 1987, los lenguajes de programación que se utilizan, y el tipo de programas que se desarrollan actualmente, han cambiado de manera notable, por lo que creemos que desarrollar SMPL en diferentes lenguajes de programación cuya utilización ha crecido tanto con el paso de los años, es una idea acertada.

La elección de desarrollarlo en Python, es por su versatilidad y por su rápida expansión en los tiempos actuales, ya que cada vez está siendo más usado. Para dar más facilidades aún, hemos desarrollado el simulador también en C# y Visual Basic para que se pueda utilizar en ámbitos diferentes, ya que Python puede ser usado en Linux, si se desea programar en lenguajes propietarios como por ejemplo C# o Visual Basic que en este caso son propiedad de Microsoft.

Se incluirán manuales de usuario para aprender a usar el simulador en todos los lenguajes anteriormente dichos, así como también el código completo.
Para probar los simuladores realizaremos dos modelos con comportamientos aleatorios. Estos modelo serán un modelo de simulación de una cola M/M/1 y un modelo de simulación del consumo energético de WiFi en un terminal móvil. Explicaremos teóricamente ambos modelos e incluiremos los códigos completos de ejemplo, para mostrar cómo se utiliza el simulador.

Realizaremos varias ejecuciones de los modelos, en los diferentes lenguajes, cambiando en cada una de las ejecuciones la semilla que se utiliza para generar los tiempos aleatorios. Cambiaremos a su vez, en el modelo M/M/1 el numero de servidos, y en el modelo WiFi, la cantidad de kbits máximos de recepción de datos.
Recogeremos los tiempos que tardarán los modelos en ejecutarse en las distintas repeticiones para poder obtener las medias de tiempos de ejecución, así como las probabilidades de estado en el modelo M/M/1. De esta manera podremos hacer una comparativa del simulador en los diferentes lenguajes elegidos.

Introducción

En esta memoria se recoge la información necesaria para la realización y utilización de un simulador de eventos discretos en Python con una ampliación para realizarlo en C# y Visual Basic.

Se detallará el presupuesto detallado junto con un diagrama de tiempos del proceso de la realización de este proyecto. Incluye información de todos los equipos utilizados y todo el software necesario para realizar los simuladores.

La memoria contiene un capítulo en el que se encuentran unos manuales de instrucciones detalladas y el código completo de todos los simuladores, así como un ejemplo para generar eventos discretos siguiendo un modelo M/M/1 y consumo energético en WiFi.
Podrán ver resultados de ejecución de los simuladores con datos y gráficas incluidos en la memoria. También se han realizado intervalos de confianza de los resultados obtenidos en los distintos simuladores y una comparación entre ellos.

Base teórica

Simulación de eventos discretos

Un simulador de eventos discretos es una técnica informática para modelar sistemas dinámicos que nos permite controlar el tiempo para avanzar a intervalos variables, para poder planificar eventos futuros en tiempos conocidos.

Es un programa que se ejecuta en un ordenador y que nos permite emular el comportamiento un sistema. Es decir, es un modelo que al ser ejecutado con los mismos estímulos que el sistema real, nos da una respuesta "suficientemente" próxima a la realidad del sistema que estamos simulando.

Con estos simuladores logramos la estimación del valor de un conjunto de parámetros de mérito que se eligen para evaluar las prestaciones de un sistema.

Cuando empleamos un simulador de eventos discretos, es deseable que cumpla con ciertos requisitos y campos de aplicación como por ejemplo:

- Aprendizaje sencillo: es deseable que la simulación no sea costosa, y que el tiempo necesario para realizar una primera simulación sea lo más corto posible.
- Facilidad de uso: el simulador debe ser fácil de usar para todos los niveles y requerir un esfuerzo bajo.
- Fácil instalación: el simulador se realizará para que no sea necesario instalar ningún software y necesitar el menor número de dependencias externas que complique su uso.
- Modular: el objetivo es que el simulador pueda ser construido modularmente, de tal manera que se puedan reutilizar bloques de código de forma sencilla.
- Lenguaje de programación sencillo y actual: la elección de un lenguaje de programación como Python, nos permite la utilización de un lenguaje sencillo y conocido por la comunidad de programadores. Gracias a Python también podremos generar de forma sencilla muestras de variables aleatorias que sigan ciertas distribuciones estadísticas.

Para ello lo primero que necesitamos es construir un modelo. Este modelo se compondrá de un conjunto de parámetros que son de nuestro interés y un conjunto de reglas que componen el funcionamiento del sistema.
Es importante que el modelo sea lo más exacto y detallado, sin omitir detalles que puedan dar a resultados erróneos, pero sin aumentar la complejidad y el tiempo de ejecución del mismo. Hay que encontrar un equilibrio entre la calidad de los resultados, y el tiempo de procesamiento.

Cuando en un sistema de espera, el espacio de estados es discreto, hablamos de un sistema generado con "eventos discretos". Llamaremos evento a cualquier cambio de estado del sistema. Por ejemplo, la llegada o salida de un cliente a una estación de servicio.

Generación de eventos

Un evento es un suceso que al ocurrir, cambia las variables de estado del sistema. El tiempo de simulación durante este procesamiento del evento permanece fijo.
Dicho espacio de tiempo, es el tiempo que avanza en el sistema a una velocidad diferente al tiempo real.

Es esencial que las variables del sistema no cambien su comportamiento en los intervalos de simulación. El simulador contiene un estado interno global que cambia cuando ocurre un evento. Estos eventos, que se almacenan en un "contenedor", son ejecutados por uno o varios procesos. El tiempo de simulación va aumentando conforme se van atendiendo los eventos, y estos, se van eliminando.

Los eventos tienen marcado su tiempo de generación, por lo que no puede coincidir el tiempo de generación con el de ejecución. La ejecución de un evento puede generar otros eventos futuros. Un sistema simulado, se caracteriza por entidades que en su conjunto conforman el sistema.

En el caso que presentamos, utilizamos por ejemplo un sistema M/M/1 que se caracteriza porque las variables aleatorias como el tiempo entre llegadas de cliente al sistema y el tiempo de servicio demandado por cada cliente, están distribuidas exponencialmente.

De este modo, cada vez que queramos simular una llegada nueva de un cliente, lo único que tendremos que hacer es llamar a una función que nos entregue el instante futuro en el que llegará el próximo cliente, basándose en una distribución exponencial.
Por otra parte, cuando queremos que un cliente ocupe el servidor, llamando a la misma función obtendremos el instante futuro y aleatorio, en el que habrá que sacar al cliente del sistema.
En ambos casos, decimos que estamos planificando eventos.

A continuación mostramos un diagrama básico del funcionamiento del simulador al que nos referimos.

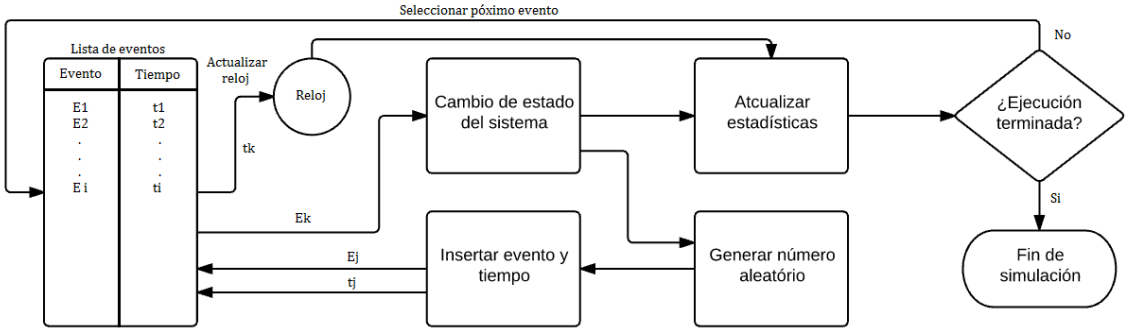


Ilustración 1: Diagrama del simulador

Intervalos de confianza

En estadística, se le llama intervalo de confianza a un par o pares de números, entre los cuales se estima que se encontrará un resultado que se obtendrá, con una probabilidad de acierto.

A partir de varias muestras, se calcula un intervalo en el que estimaremos la probabilidad de que una muestra futura de las mismas características, esté dentro de ese intervalo.
El cálculo de intervalos de confianza, son técnicas que nos permiten hacer estimaciones sobre qué valores podemos esperar para un parámetro.

El intervalo de confianza dependerá de:

- Lo estimado en la muestra, ya que el intervalo de confianza estará formado por valores ligeramente mayores y menores de la aproximación ofrecida por nuestra muestra.
- El tamaño muestral, por ello, cuantos más datos hayan participado en el cálculo, más pequeño esperamos que sea la diferencia entre el valor estimado y el valor real.
- La probabilidad, es el nivel de confianza con el que el método nos dará una respuesta correcta. Lo más habitual es que el nivel confianza esté entre el 95% y 99%.

En estadística, la probabilidad de éxito en una estimación se representa con (1- α) o nivel de confianza., siendo "α" el error aleatorio, esto es, la media de la probabilidad de fallar en la estimación.

Esta probabilidad de error, dependerá de la amplitud del intervalo de confianza. A mayor amplitud, menor probabilidad de error habrá, por lo que cuanto menor sea la amplitud, mayor será la probabilidad de error.

Para conseguir un intervalo de confianza, necesitamos saber cuál es la distribución que sigue el parámetro que estamos estimando.
En resumen, la expresión de un intervalo de confianza al "1- α" por ciento, para una probabilidad "Θ" que sigue una distribución de probabilidad "P" del tipo $[\Theta_1, \Theta_2]$ seria:

$$P[\Theta_1 \leq \Theta \leq \Theta_2] = 1 - \alpha$$

Aplicando lo anterior, ponemos un ejemplo con la distribución t (de Student) que es una distribución de probabilidad, que surge del problema de estimar la media de una población normalmente distribuida, cuando el tamaño de la muestra es pequeño.

La función de probabilidad P es la siguiente:

$$P = Z\sqrt{\frac{v}{V}}$$

Donde:

- Z es una variable aleatoria distribuida según una normal de media nula y varianza 1.
- V es una variable aleatoria que sigue una distribución X^2 con v grados de libertad
- Z y V deben ser independientes.

Simuladores de eventos discretos

SMPL

Es una biblioteca de C que proporciona todas las funciones necesarias para crear pequeños simuladores de sucesos discretos.

En SMPL se distinguen tres entidades: estaciones de servicios, marcas o tokens y eventos.

Las estaciones de servicios vienen caracterizadas por un conjunto de servidores y su correspondiente cola para alojar a los clientes que esperan ser atendidos. SMPL ofrece funciones para definir estaciones de servicio, usarlas y preguntar por su estado.

Cuando un nuevo cliente llega al sistema, se le asigna una estructura de datos propia, que le acompañará desde su llegada hasta su marcha. Esta estructura de datos, contiene, por ejemplo el tiempo en el que el cliente llegó al sistema, para poder comprobar a su salida, cuanto tiempo esperó en el sistema, también un identificador de tipo, para poder clasificar al cliente. No es necesaria una clasificación individual (aunque podría hacerse), sino una clasificación grupal por tipos.

Un evento es un cambio de estado en el sistema, SMPL nos da funciones para registrar eventos y seleccionarlos en función de su tiempo de ocurrencia. Estos eventos suelen estar identificados por el tiempo en el que ocurrieron, y el tipo al que pertenecen.

Para usar el simulador SMPL se debe iniciar, para después poder usar sus recursos compartidos y controlarlo.

Inicialización de SMPL

Para poder utilizar el simulador lo primero que hay que hacer es iniciarlo con la función **smpl(s)** donde **s** es un puntero a un conjunto de caracteres que describen el modelo. Esta función inicializa las estructuras de datos, inicia el reloj de simulación y el intervalo de medida.

Definición de los recursos compartidos y su control

- **facility(s, n),** esta función crea y nombra una estación de servicio, donde 's' es un puntero al nombre de la estación y 'n' es el número de servidores de la estación. Esta función devuelve un entero '**f**' que identifica a la estación de servicio.

- **request(f, tkn, pri),** esta función solicita a la estación de servicio con identificador 'f', que atienda al cliente con identificador 'tkn', con una prioridad 'prl'. Si la estación de servicio está totalmente ocupada, el cliente es puesto en cola por orden de prioridad.

- **preempt(f, tkn, pri),** esta función solicita a la estación de servicio con identificador 'f', que atienda al cliente con identificador 'tkn', con una prioridad 'pr'. Si la estación de servicio está totalmente ocupada, se busca el servidor que esté atendiendo al cliente con menos prioridad, si este cliente tiene menor prioridad que el anterior, se procede al desalojo para atender al nuevo cliente con mayor prioridad, en caso contrario, se podrá al cliente en cola.

- **release(f, tkn),** esta función libera de la estación de servicio con identificador 'f', ocupada por la marca 'tkn'. Al producirse la liberación, se produce a la recogida de datos estadísticos y a la reubicación de los clientes pendientes en el servidor.

- **inq(f),** se utiliza para obtener el número de marcas en la cola de la estación de servicio con el identificador 'f', sin contar los que están siendo servidos.

- **status(f),** devuelve '1' si la estación de servicio esta totalmente ocupada o '0' si hay al menos 1 servidor libre.

- **u(f),** devuelve la utilización media de la estación de servicio 'f', es decir, el número medio de servidores ocupados.

- **b(f),** devuelve el factor de utilización, es decir, el porcentaje de tiempo que han sido utilizados los servidores de la estación 'f'.

- **Lq(f),** esta función devuelve la longitud media de la cola de la estación de servicio 'f' durante el intervalo de medida.

- **reset,** nos permite iniciar un nuevo intervalo de medida.

Generación y registro de eventos

- **schedule(ev, te, tkn),** esta función planifica un evento con número 'ev', en la tabla de eventos, 'te' es el tiempo incremental futuro, (a partir del instante actual) en que ocurrirá el evento, y 'tkn' es la marca asociada a dicho evento.
  La función ordenará de manera ascendente el evento en función del 'te' dentro de la lista de eventos.

- **cause(var ev, tkn),** esta función extrae el siguiente evento de la tabla de eventos, avanza el reloj hasta la siguiente marca de tiempo, que es, el instante que se debe producir el evento, y devuelve el número de evento y la marca.

- **cancel(ev),** se utiliza para eliminar de la tabla de eventos el evento 'ev'. Si lo encuentra lo elimina y devuelve la marca, y sino devuelve '-1'. Solo elimina, en caso

de encontrar eventos, el que tenga la marca más próxima, es decir, el que primero se atendería de todos.

- **time,** esta función retorna el valor actual del reloj de simulación.

Generación de números aleatorios

- **random(i, n),** esta función retorna números aleatorios dentro del rango [i – n]
- **expntl(x),** esta función retorna números aleatorios distribuidos exponencialmente y con media |x|.
- **uniform(a, b)**, genera números aleatorios generados por una distribución uniforme acotada entre a y b.
- **erlang(x, s)**, genera números aleatorios generados por una distribución erlang de media |x| y desviación estándar s.
- **hyperx(x, s)**, genera números aleatorios generados por una distribución hiperexponencial con media |x| y desviación estándar s.
- **normal(x, s)**, genera números aleatorios generados por una distribución normal con media |x| y deviación estándar s.

GPSS

General Purpose Simulation System es un lenguaje de programación de propósito general de simulación en tiempo discreto.

GPSS se distribuye en bloques, cada bloque se ocupa de una parte de la simulación.
Todo empieza con un START, que pone en marcha el simulador, hay que informar a este bloque el número de transacciones que el usuario quiere simular.

El siguiente bloque que tiene que ser ejecutado es el GENERATE.
En el bloque GENERATE los eventos se inyectan al simulador, de esta manera los eventos son incluidos en la cadena de eventos futuros, con la información del instante del futuro nacimiento.
GENERATE admite varios valores que los enumeraremos de la A a la E de la siguiente manera:

GENERATE A,B,C,D,E

Donde:
A: es la tasa promedio con la que se crean las transacciones.
B: es la dispersión en el tiempo de creación promedio de las transacciones.
C: es para establecer el tiempo de simulación en el que llega la primera transacción al modelo.
D: número límite de transacciones creadas.
E: prioridad de la transacción.

Otro bloque es el llamado ADVANCE cuya labor es generar eventos futuros pero informando del instante de la terminación de la tarea.
ADVANCE admite dos valores de la siguiente manera:

ADVANCE A,B

Donde:
A: es el tiempo de retardo para la transacción.
B: es el intervalo de dispersión alrededor de A.

GPSS detiene el reloj y simula todo lo que le toca en ese determinado instante, cuando no tiene nada más que simular, mira en la lista de eventos pendientes, y obtiene el próximo, cambiando el reloj al instante del evento que se va a atender.
Este proceso se repite hasta que se encuentra un TERMINATE.

GPSS es un lenguaje que persiste en el tiempo, ya que fue un diseño muy avanzado para la época en la que fue desarrollado. Es muy usado en la enseñanza de los eventos discretos.

Simpy

Simpy es un módulo que permite crear modelos de simuladores de eventos discretos. Está programado en Python y fue comercializado por GNU GPL.

Su aplicación se centra en simular y modelar epidemias, tráfico, vigilancia aérea de aviones, ingeniería industrial, estudios de rendimiento del hardware de las computadoras, funcionamientos de modelado, estudios de volumen de trabajo, optimización de procesos industriales y enseñanza de la metodología de la simulación.

El paquete está formado por un conjunto de módulos escritos en Python los cuales son:

Simulation: Es el módulo que tiene implementada las definiciones de las clases y métodos para los objetos Process (Procesos) y Resources (Recursos), empleados en los modelos de simulación.

Monitor: Módulo para la compatibilidad de versiones anteriores a SimPy.

SimulationTrace: Módulo que implementa las trazas para los eventos.

SimulationRT: Módulo para controlar la velocidad de la simulación a través de la sincronización de los eventos.

SimulationStep: Módulo para realizar la simulación paso a paso a través del seguimiento de los eventos.

SimPlot: Módulo que permite realizar gráficas de estadísticas durante la simulación. Está basado en la librerías Tk/Tkinter.

SimGui: Este módulo provee las herramientas para implementar el modelo de simulación con interfaces gráficas de usuario (GUI). Al igual que el módulo SimPlot, SimGUI está basado en las librerías Tk/Tkinter.

Lister: Módulo empleado para darle un mejor formato de impresión a los objetos SimPy empleados por el modelo.

OMNeT ++

OMNeT++ es una herramienta de modelado y simulación pública, basado en componentes modulares, con un ambiente de simulación de arquitectura abierta y con soporte de GUI.

OMNeT fue creado en 2003 en la Universidad Técnica de Budapest. Es un simulador de eventos discretos. Su área de aplicación es la simulación de redes de comunicación, y debido a su arquitectura genérica y flexible, ha sido utilizada exitosamente en redes basadas en colas de espera y arquitectura de hardware.

Ya que OMNeT++ provee una arquitectura modular, estos componentes (módulos) están programados en C++, después son ensamblados en componentes y modelos más grandes utilizando un lenguaje de alto nivel.
Gracias a su soporte de GUI, junto con su arquitectura modular, las simulaciones pueden ser incluidas fácilmente en aplicaciones propias, por lo que OMNeT++ actualmente está ganando popularidad como una plataforma de simulación de redes en la comunidad científica así como la industria.

OMNeT++ corre para las plataformas Linux, Unix-like y Win32 (Windows 2000 y XP).

Los componentes de OMNeT son:

- Compilador para la topología de descripción de lenguaje NED (nedc)
- GUI para la ejecución del simulador, enlaces dentro de las simulaciones ejecutables (Tkenv) .
- Interfaz al usuario de las líneas de comando de la simulación ejecutada (Cmdenv)
- Herramienta de graficación por vectores de gráficas de resultados (Plove)
- Herramienta de visualización escalar de gráficas de resultados (Scalars)
- Herramienta de documentación de modelo (opp_neddoc)
- Utilidades (herramienta de generación de semillas de números aleatorios, herramienta de creación de archivos, etc.)
- Documentación, simulaciones de muestra, etc.

OPNet

Opnet es un programa con un lenguaje de programación que os permite simular redes de comunicaciones. Proporciona acceso al código fuente lo que ofrece la posibilidad de personalización del simulador.

Nos permite simular gran variedad de redes de comunicaciones, gracias a sus librerías y nos facilita el estudio del desarrollo de los modelos mediante la posibilidad de hacer diferentes conexiones entre los nodos de la red.

Nos ofrece la posibilidad de crear modelos de procesos. Estos procesos son representados por estados, y transiciones entre ellos. Estas operaciones o transiciones entre estados, se programan en C++, a este tipo de simulación se le denomina DES (Discrete Event Simulation).

A continuación mostramos un ejemplo de la representación de dos nodos y sus procesos asociados.

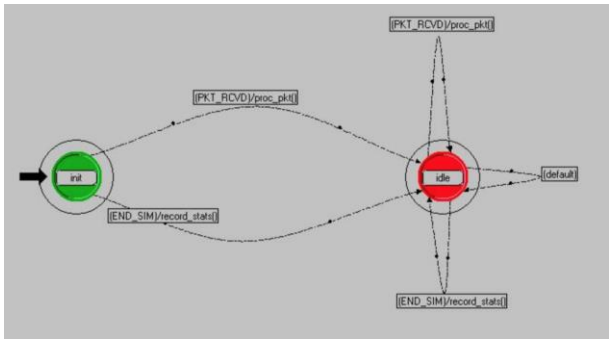


Ilustración 2: Nodos y procesos en Opnet

En la simulación con Opnet podemos elegir qué estadísticas queremos recopilar. Como por ejemplo, estadísticas globales de nodos, de enlaces, de atributos e incluso tráfico que se genera en las rutas.

Las estadísticas pueden ser representadas en gráficos como el que se muestra a continuación, en el que podemos ver la representación del rendimiento de una simulación.

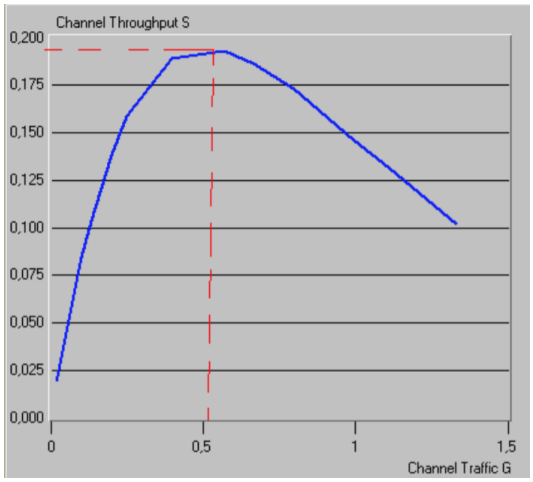


Ilustración 3: Gráfico de rendimiento en Opnet

Opnet es una herramienta muy completa que nos permite de forma gráfica construir y simular redes LAN de forma sencilla y detallada. El ejemplo que veremos a continuación es una red LAN con nodos como servidores, dos switches de 3COM un router catalyst de Cisco y diferentes equipos conectados a la red.

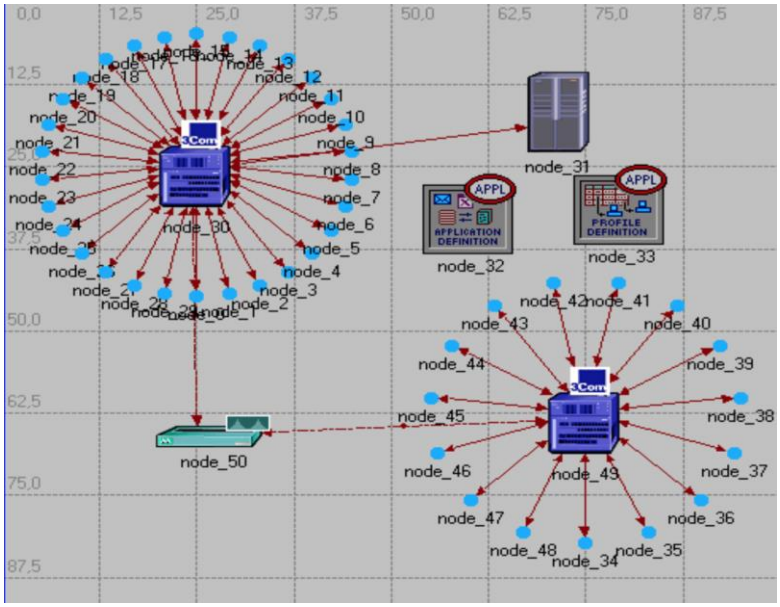


Ilustración 4: Ejemplo de una red LAN en Opnet

Como hemos visto, Opnet permite de forma clara y visual construir una red de comunicaciones de forma sencilla, pero que nos da acceso al código fuente para poder configurar la simulación y los elementos que la forman de manera más compleja. Tras la simulación nos ofrece una gran variedad de estadísticas e información de la simulación.

Si se desea simular una red de comunicaciones Opnet es una de las soluciones más completas e ilustrativas. Es muy utilizado en la enseñanza de eventos discretos y simulación de redes de comunicaciones.

Objetivos a conseguir

Una vez explicado sin entrar en profundidad los diferentes simuladores más utilizados, podemos explicar el porqué de este proyecto, y los objetivos que queremos conseguir con la realización del mismo.

El principal objetivo de este proyecto, es constituir un simulador, basado en el descrito por MacDougall en "*Simulating Computer Systems: Tecniques and Tools*" llamado SMLP, con la particularidad de hacerlo lo más sencillo posible para el usuario.

En los principales simuladores descritos anteriormente, es necesario documentarse y hacer un estudio previo y completo del simulador para poder utilizarlos. Esto requiere un tiempo y esfuerzo extra, que, con la realización de este proyecto se quiere evitar.
Por ello queremos hacer un simulador polivalente, fácil de utilizar y con las funcionalidades más importantes y básicas de los simuladores habituales.

Para conseguir esa polivalencia, haremos el simulador en lenguajes como Python, C# y Visual Basic.

Para que el simulador sea fácil de usar, le dotaremos de la funcionalidad básica, la cual será:

- Introducir eventos en el simulador.

- Extraer eventos del simulador.
- Eliminar eventos de un tipo.
- Imprimir la lista de eventos.
- Eliminar la lista de eventos.
- Obtener el tiempo de simulación.
- Imprimir el tiempo de simulación.

Los eventos se introducirán con los siguientes parámetros:

- Tiempo
- Tipo
- Parámetro opcional

Se incluirá un manual de instrucciones, corto y sencillo, con ejemplos de códigos de modelos que utilizan nuestro simulador. Estos ejemplos serán un modelo de simulación de una cola M/M/1 y un modelo de simulación del consumo energético de WiFi en terminales móviles.

A continuación explicaremos qué es un modelo de simulación y los diferentes modelos de que utilizaremos en el proyecto.

Modelos de simulación

Para poder comprobar el correcto funcionamiento del simulador, realizaremos modelos de generación de eventos que utilizarán el simulador para su funcionamiento.

Antes de explicar los modelos elegidos, vamos a resumir conceptos importantes para entender los que es un modelo de simulación.

A continuación citaremos las definiciones de sistema, modelo y simulación, relacionándolas entre sí, estas definiciones son del economista Minsky y el ingeniero y matemático Shannon.

Sistema: Es un conjunto de objetos o ideas que están interrelacionados entre sí como una unidad para la consecución de un fin (Shannon). También se puede definir como la porción del Universo que será objeto de la simulación.

Modelo: Un objeto X es un modelo del objeto Y para el observador Z, si Z puede emplear X para responder cuestiones que le interesan acerca de Y (Minsky).

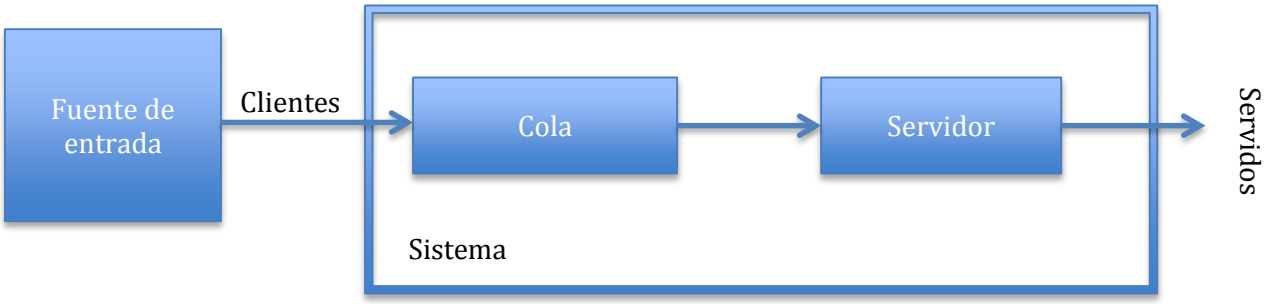
Simulación: Simulación es el proceso de diseñar un modelo de un sistema real y llevar a cabo experiencias con él, con la finalidad de aprender el comportamiento del sistema o de evaluar diversas estrategias para el funcionamiento del sistema (Shannon).

Los modelos de simulación elegidos son el modelo M/M/1 y un modelo del consumo energético de un terminal móvil con la tecnología WiFi.

Modelo M/M/1

El modelo M/M/1 es un modelo perteneciente a la Teoría de Colas. La teoría de colas es parte de la investigación que estudia el comportamiento de sistemas cuyos elementos incluyen líneas de espera o colas.

Un ejemplo de cola con 1 servidor sería la siguiente.



Los elementos más importantes en un sistema de colas son los clientes y el servicio.

Los clientes se caracterizan por los intervalos de tiempo que separan sus llegadas.

El servicio se caracteriza por el tipo y tiempo de servicio, además de por el numero de servidores.
El tipo de servicio o disciplina representa el orden en el que los clientes se seleccionan de la cola. Las llegadas de clientes pueden ser deterministas o aleatorios. Los tiempos de servicio también pueden ser deterministas o aleatorios.

Dos elementos importantes en las colas son la tasa de llegadas y el tiempo medio entre llegadas.
La tasa de llegadas o λ, es el número medio de clientes que llegan al sistema por unidad de tiempo.
El tiempo medio entre llegadas se define como 1/λ.

Es usual suponer que los tiempos entre llegadas y los de servicio se distribuyan de forma exponencial, cuya función de densidad de probabilidad es:

$$f(t) = \lambda e^{-\lambda t} \ para\ t \geq 0$$

Donde λ es la tasa de llegadas.

Una distribución exponencial de los tiempos entre llegadas implica un proceso de Poisson para las llegadas, el cual describe la probabilidad de que lleguen n clientes en las siguientes t unidades de tiempo:

$$P(Xt = n) = e^{-\lambda t}\frac{(\lambda t)^n}{n!} \ para\ n = 0, 1 \ldots$$

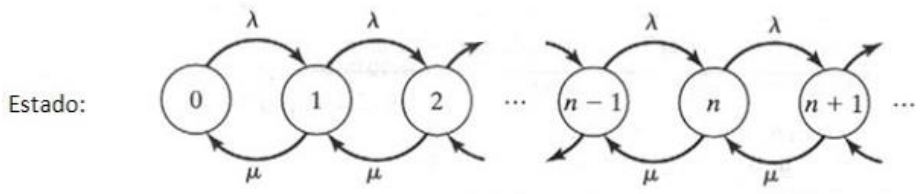
En la práctica, se habla de llegadas de Poisson y tiempos de servicio exponencial. En general se supone que el sistema se encuentra en estado estacionario (estabilidad independiente del tiempo).

Los parámetros más importantes de las colas M/M/1 son los siguientes.

λ ≡ tasa de llegadas.
μ ≡ tasa de servicio.
s ≡ número de servidores

Empezaremos realizando el diagrama de estados de una cola M/M/1, definido por:

- Nodo: o también llamado estados o vértices
- Arcos: representan la transición entre estados o nodos.



$Pk \equiv Probablidad\ de\ estar\ en\ el\ estado\ k$
En régimen permanente se cumple el equilibrio de flujos de un corte vertical .
Haciendo sucesivos cortes verticales tenemos:

$$\lambda P_0 = \mu P_1 \rightarrow P_1 = \rho P_0$$
$$\lambda P_1 = \mu P_2 \rightarrow P_2 = \rho P_1 = \rho^2 P_0$$
$$\lambda P_2 = \mu P_3 \rightarrow P_3 = \rho P_2 = \rho^3 P_0$$

$$\lambda P_{k-1} = \mu P_k \rightarrow P_k = \rho P_{k-1} = \rho^k P_0$$

La suma de todas las probabilidades es igual a 1, por lo que:

$$\sum_{K=0}^{\infty} P_K = 1$$

$P_0\ (1 + \rho + \rho^2 + \rho^3 + \ldots) = 1$ Suma geométrica de razón ρ $(\rho < 1)$

$$S = \frac{a_1 - a_n r}{1 - r} = \frac{1}{1 - \rho}$$

$$P_0 \frac{1}{1-\rho} = 1 \quad \text{como } P_0 = 1 - \rho$$

$$P_k = (1 - \rho)\rho^k \quad \forall_k > 0$$

Con las probabilidades de estado podemos calcular diferentes parámetros:

- P (sistema vacío) = $P_0 = 1 - \rho$
- P (haya alguien en el sistema) = 1 - $P_0 = \rho$
- Utilización= 1 - $P_0 = \rho$

- Número medio de clientes en el sistema

$$r = E[r] = \sum_{K=0}^{\infty} KP_K = \sum_{K=0}^{\infty} K(1-\rho)\rho^k = (1-\rho)\rho \sum_{K=0}^{\infty} \frac{d\rho}{\rho}\rho^k = (1-\rho)\rho \frac{d\rho}{\rho} \sum_{K=0}^{\infty} \rho^k$$

Como $\sum_{K=0}^{\infty} \rho^k$ es la suma de una serie geométrica

$$= (1-\rho)\rho \frac{d\rho}{\rho} \frac{1}{1-\rho} = (1-\rho)\rho \frac{1}{(1-\rho)^2} = \frac{\rho}{1-\rho}$$

A continuación vamos a hacer una comparación de los códigos del modelo M/M/1 en Python, C# y Visual Basic.

Cola M/M/1 en Python

```
from Imports import *
import random

YES = 1
NO = 0
ERROR = 1

nuevaLlegada = 0
servicio = 1
finServicio = 2
labda = 1.0
mu = 2.0
seed_n = 0
cola=0
seguir=YES
servidor_ocupado=NO
servidos=0

random.seed("texto")

init_simulation()

Schedule(event(nuevaLlegada,0.0,0))

while servidos < 10000:
   eventoRecibido = Cause()

   if eventoRecibido.type == nuevaLlegada:
      Schedule(event(servicio,0.0,0))
```

```
    Schedule(event(nuevaLlegada,random.expovariate(labda),0))

  elif eventoRecibido.type == servicio:
    if servidor_ocupado==NO:
      servidor_ocupado=YES
      Schedule(event(finServicio,random.expovariate(mu),0))
    else:
      cola = cola + 1


  elif eventoRecibido.type == finServicio:
    servidos = servidos + 1
    if cola>0:
      cola = cola - 1
      Schedule(event(finServicio,random.expovariate(mu),0))
    else:
      servidor_ocupado=NO
```

Cola M/M/1 en C, C# y Visual Basic

El código de la cola M/M/1 en los diferentes lenguajes usados, se encuentra en la parte de los anexos.

Modelo del consumo energético de WiFi en un terminal móvil

Dado que en la sociedad actual, la dependencia que tenemos del móvil va en aumento, este se ha convertido en una herramienta básica en nuestras vidas. La necesidad de estar 100% conectados y disponibles, ahora es una realidad.

Para ello, nuestros terminales móviles utilizan conexiones WiFi que permiten estar conectados a internet.
Vamos a centrarnos en el estudio del consumo energético de un terminal móvil cuando trasmite y recibe datos a través de la conexión WiFi.

Descripción del modelo energético de WiFi

El modelo energético de WiFi se define fácilmente con un diagrama de estados, sus respectivos tiempos de transición de estados y la potencia necesaria para conseguir conmutar de un estado a otro.

Para verlo mejor de manera gráfica, incluimos el siguiente diagrama que describe de forma resumida el modelo energético de WiFi.

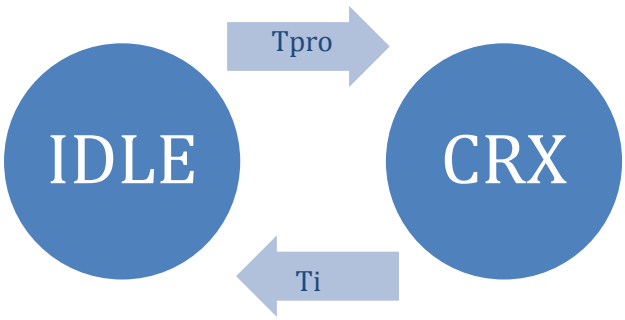


Ilustración 5: Diagrama de estados del modelo energético de WiFi

Los estados por los que el modelo energético pasa durante la simulación son:

- CRX: es el estado en el que el sistema se encuentra en recepción continua
- IDLE: o también llamado estado de reposo, como su nombre indica es un estado en el que el sistema se encuentra "dormido" a la espera de tener que recibir o enviar datos.

Los tiempos que se muestran en el diagrama se explicarán a continuación en la siguiente sección.

Funcionamiento del modelo energético WiFi

Descripción gráfica

Vamos a mostrar un ejemplo gráfico del funcionamiento del modelo energético WiFi, para, a continuación poder explicarlo de forma teórica de manera más sencilla.

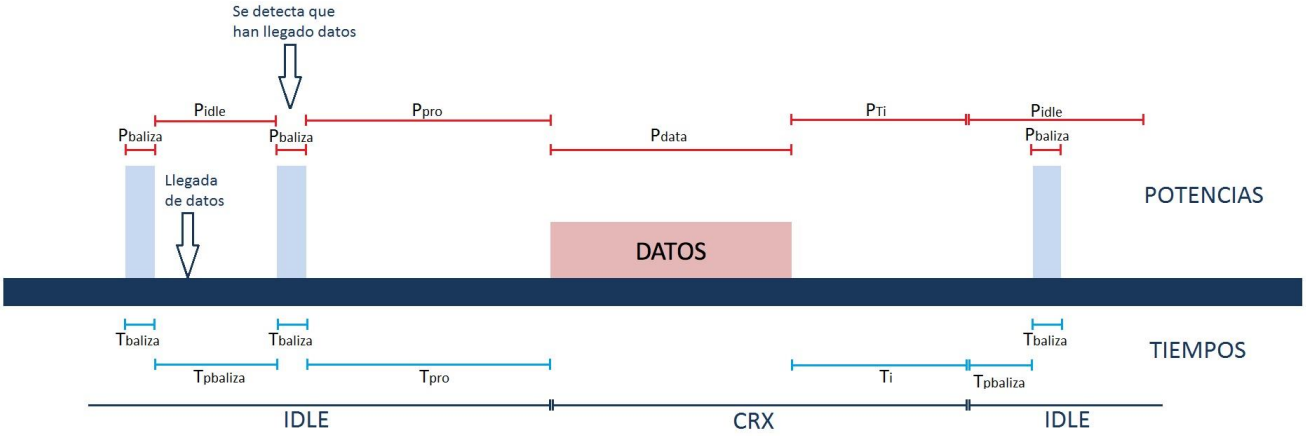


Ilustración 6: Ejemplo de modelo de consumo energético de WiFi

Para poder continuar con la explicación es necesario explicar los tiempos y potencias que se muestran en el ejemplo anterior.

Tiempos

Los tiempos utilizados en el modelo energético de WiFi son los siguientes:

- $T_{pbaliza}$: es el periodo con el que se producen tramas baliza.
- T_{baliza}: es el tiempo de duración de una trama baliza.
- T_{pro}: tiempo de promoción, es el tiempo usado para asignar recursos de radio al terminal, tras el cual se cambia del estado IDLE al CRX.
- T_i: es el tiempo de inactividad, cuando un paquete es enviado o recibido, se empieza a decrementar este tiempo, si durante este tiempo no se tiene que recibir, ni enviar otro paquete, se pierden los recursos radio y se vuelve al estado IDLE.

En la siguiente tabla se muestran los valores elegidos en esta simulación para los diferentes tiempos.

Variable	Duración (ms)
$T_{pbaliza}$	300.6
T_{baliza}	7.6
T_{pro}	79.1
T_i	328.1

Tabla 1: Tiempos del modo energético de WiFi

Potencias

Las potencias indicadas en el ejemplo anterior son las siguientes:

- P_{baliza}: es la potencia consumida para procesar la trama baliza.
- P_{idle}: potencia empleada por el terminal cuando no está operativo.
- P_{pro}: es la potencia requerida para la fase de promoción.
- P_{data}: potencia consumida para procesar los datos.
- P_{Ti}: es la potencia consumida durante el transcurso del temporizador Ti.

En la siguiente tabla se muestran los valores elegidos en esta simulación para las diferentes potencias.

Variable	Valor (mW)
P_{baliza}	77.2
P_{idle}	11.4
P_{pro}	124.4
P_{Ti}	119.3

Tabla 2: Potencias del modelo energético WiFi

Una vez explicado cada uno de los tiempos y las potencias, y con un ejemplo gráfico del modelo energético WiFi, procedemos a explicar de forma más teórica lo que sucede en el proceso.

Descripción teórica

Cuando el terminal se encuentra en reposo, es decir, sin transmitir ni recibir datos, se encuentra en el estado IDLE. En este estado, cada $T_{pbaliza}$ se comprueba si hay datos que recibir o enviar, esta comprobación dura T_{baliza}. En este proceso se consume una potencia en la espera y en la comprobación, igual a la suma de las potencias P_{idle} y P_{baliza}.

Si hay datos que enviar o recibir, se procede a la transición al estado CRX. En este momento, se comienza un tiempo Tpro, consumiendo una potencia Ppro. Pasado este tiempo nos encontramos en el estado CRX en el cual podemos empezar a recibir o enviar los datos.

Para enviar los datos es necesario consumir una potencia Pdata y un tiempo t diferente en cada caso, en función de la cantidad de datos que se envíen o reciben. Durante este proceso seguimos en el estado CRX.

Cuando se termina de enviar o recibir los datos, se espera un tiempo Ti el cual tiene un coste de potencia para mantenerse en el estado CRX igual a PTi. Si durante este tiempo se reciben datos nuevos, o por el contrario hay que enviar más datos, se repite el proceso del párrafo anterior, pero si no hay que enviar ni recibir más datos, cambiamos de estado del CRX al IDLE.
Tras esta transición volvemos a la situación del primer párrafo de esta sección, en el que debe transcurrir un tiempo Tpbaliza consumiendo una potencia Pidle, para repetir todo el proceso explicado de nuevo.

Consumo energético de WiFi en terminales móviles

Una vez explicado el proceso del modelo, vamos a analizar como calcular la energía consumida por nuestro dispositivo en todo el proceso.

Transferencia de datos

En este apartado vamos a hablar de la potencia que se necesita en la transmisión o recepción de datos.

En el proyecto de Laura Sánchez (ver referencia en la bibliografía), toma como ejemplo un artículo llamado Huang 2013, el cual toma como referencia los siguientes parámetros específicos para subida y bajada de datos.

αu [mW/Mbps]	αd [mW/Mbps]	β[mW]	αu/αd
283.17	137.01	132.86	2.07

Tabla 3: Parámetros de subida y bajada en el modelo WiFi

Los parámetros de la tabla son los siguientes:

- α_u: Potencia consumida en mW/Mbps en régimen de subida.
- α_d: Potencia consumida en mW/Mbps en régimen de descarga.
- β: Energía fija independiente del proceso de subida o bajada.
- t_u: Régimen binario de subida en Mbps.
- t_d: Régimen binario de bajada en Mbps.

Observando la tabla propuesta la relación α_u / α_d nos indica que el consumo es mayor durante la subida que la bajada.

Durante la transferencia de datos, con los factores anteriores, el nivel de energía se determinará con la potencia multiplicada por el tiempo de duración de la transmisión. Vamos a explicar cada una de las potencias:

- Potencia de subida: Es la consumida por el terminal cuando se produce una transferencia de subida de datos.

$$P_u = \alpha_u \times t_u + \beta \text{ [mW]}$$

- Potencia de bajada: Es la que requiere el terminal durante la transferencia de datos de bajada.

$$P_d = \alpha_d \times t_d + \beta \text{ [mW]}$$

- Potencia de transmisión: Potencia total necesaria para la transferencia.

$$P_{data} = P_u + P_d \text{ [mW]}$$

Energía consumida por el terminal

Para calcular la energía, se debe tener en cuenta todas las contribuciones, relacionadas con cada uno de los procesos, tanto los cambios de estado, como las transferencias de datos.

Gracias a relación existente entre la Potencia y le Energía, podemos calcular cual sería el consumo de la batería del móvil en Julios.

$$P = \frac{E}{T}$$

A continuación vamos a explicar cada una de las energías que contribuyen dentro de cada uno de los procesos.

- **Estado IDLE**: en el estado IDLE o reposo, tendremos las siguientes energías.
    - Energía baliza: es la consumida cada vez que se recibe una traza baliza.

$$E_{baliza} = P_{baliza} \times T_{baliza} \times N$$

* Donde N es el número de balizas recibidas durante el proceso

    - Energía en reposo: es consumida mientras el terminal no esta operativo.

$$E_{off} = P_{idle} \times T_{idle}$$

    - Energía de promoción: es la requerida para preparar el dispositivo para una transmisión de datos, cambiando del estado IDLE al estado CRX.

$$E_{promoción} = P_{pro} \times T_{pro} \times N$$

* Donde N es el número de promociones que suceden durante la comunicación.

    - Energía IDLE: que es la energía que consume el terminal en total durante este estado.

$$E_{IDLE} = E_{baliza} + E_{off} + E_{promoción}$$

- **Estado CRX**: es el estado en el que se produce la transmisión o recepción de los paquetes, cuya aportación de energía se explica a continuación.

    - Energía transmisión: es la consumida cada vez que se recibe o envía un paquete.

$$Edata = Pu \times tsub + Pd \times tbaj$$

* Donde tsub tiempo de subida de datos, y tbaj es el tiempo de bajada de datos.

    - Energía inactividad: es consumida mientras el terminal está inactivo, esperando una nueva trasmisión o recepción.

$$ETi = PTi \times tTi$$

* Donde tTi es el tiempo que se encuentra en inactividad.

    - Energía de recepción continua: es la energía total consumida en el estado CRX.

$$ECRX = Edata + ETi$$

Para calcular le energía total, tan solo tendremos que sumar las dos energías totales calculadas anteriormente.

$$ETOTAL = EIDLE + ECRX$$

Modelo energético de WiFi en Python

A continuación incluimos el código del modelo energético WiFi en python.

```
from Imports import *
import random, time

kbits_remaining=100000000.0

init_simulation()

debug=False
YES=1
NO=0
ERR=1
DOWN=0
UP=1
user_to_CRX=0
user_to_IDLE_OFF=1
user_to_IDLE_ON=2
new_packet=3
end_flow=4

CRX=0
IDLE_ON=1
IDLE_OFF=2

T_PRO=79.1
T_ON_IDLE=7.6
```

```
T_OFF_IDLE=300.6
T_i=238.1

P_PRO=124.4
P_Ti_BASE=119.3
P_ON_IDLE=77.2
P_IDLE_BASE=11.4
alfapow_u=0.28317
alfapow_d=0.13701
betapow=132.86

e_pro=0.0
e_data=0.0
e_idle=0.0
e_Ti=0.0

Rb_down=40000.0
Rb_up=4000.0
Rb=0.0

follow=1
time_a=0.0
state=0
Ti_running=0.0
init_Ti=0.0

promoting=0
summed_p_IDLE=0
entry_to_IDLE_ON=0.0
just_entry_to_IDLE_ON=0
PS=0.0
interpacket=0.0
UP_DOWN=0
kbits_remainingInicio=0.0
tk_app2=0.0


tk=0
aux_real=0.0
PSize=0.0
PS=1000.0
interpacket = 100.0
kbits_remainingInicio= kbits_remaining
UP_DOWN = DOWN

alfa=0.0
power=0.0
elapsed_time=0.0

def empty_apps():
  global   debug, YES, NO, ERR, P_PRO, T_i, T_OFF_IDLE, T_ON_IDLE, T_PRO, IDLE_OFF,   IDLE_ON,
           CRX, end_flow, new_packet, user_to_IDLE_ON, user_to_IDLE_OFF, user_to_CRX, UP,
           DOWN, Ti_running, state, time_a, follow, Rb, Rb_up, Rb_down, e_Ti, e_idle, e_data, e_pro,
           betapow, alfapow_d, alfapow_u, P_ON_IDLE, P_Ti_BASE, P_IDLE_BASE, interpacket, PS,
           PSize, aux_real, tk, tk_app2, kbits_remaining, UP_DOWN, interpacket,
           just_entry_to_IDLE_ON, promoting, init_Ti, summed_p_IDLE, entry_to_IDLE_ON, PS,
           tk_app2, UP_DOWN, kbits_remainingInicio, alfa, power, elapsed_time

  elapsed_time = 0.0
```

```
    power = 0.0
    alfa = 0.0

    elapsed_time=timeSim() - time_a
    time_a = timeSim()


    if tk_app2 > 0.0 and state == CRX:

        if (UP_DOWN == DOWN):
            Rb = Rb_down
            alfa = alfapow_d
        else:
            Rb = Rb_up
            alfa = alfapow_u

        tk_app2 = int(round(tk_app2 - round(elapsed_time * Rb * 0.001)))

        if (int(round(tk_app2)) == -1):
            tk_app2 = 0.0

        if (tk_app2 == 0.0):
            Schedule(event(user_to_IDLE_OFF, T_i, 2))
            summed_p_IDLE=YES

            now = timeSim()

            init_Ti = now

            Ti_running = YES

        power = alfa * Rb + betapow
        e_data = e_data + power * elapsed_time * 0.000001



def end_reschedule():
    global    debug, YES, NO, ERR, P_PRO, T_i, T_OFF_IDLE, T_ON_IDLE, T_PRO, IDLE_OFF, IDLE_ON,
              CRX, end_flow, new_packet, user_to_IDLE_ON, user_to_IDLE_OFF, user_to_CRX, UP,
              DOWN, Ti_running, state, time_a, follow, Rb, Rb_up, Rb_down, e_Ti, e_idle, e_data, e_pro,
              betapow, alfapow_d, alfapow_u, P_ON_IDLE, P_Ti_BASE, P_IDLE_BASE, interpacket, PS,
              PSize, aux_real, tk, tk_app2, kbits_remaining, UP_DOWN, interpacket,
              just_entry_to_IDLE_ON, promoting, init_Ti, summed_p_IDLE, entry_to_IDLE_ON, PS,
              tk_app2, UP_DOWN, kbits_remainingInicio, alfa, power, elapsed_time

    leaving = 0
    aux_real= 0.0

    while leaving != -1:
        leaving = deleteEventType(end_flow)

    if (state == CRX):
        if (UP_DOWN == DOWN):
            Rb = Rb_down
        else:
            Rb = Rb_up

        aux_real = tk_app2 / Rb * 1000.0
```

```
    if (aux_real > 0.0):
      Schedule(event(end_flow, aux_real, 1))

#Main

Schedule(event(new_packet, 0.0, 1))

time_a = timeSim()

Schedule(event(user_to_CRX, 0.0, 2))


while follow == YES:

  evento = Cause()

  opcion = evento.type

  if (kbits_remaining == 0):
    follow = NO

  empty_apps()

  if opcion == user_to_CRX:
    state = CRX

    promoting = NO

  if opcion == user_to_IDLE_OFF:
    if (promoting == NO):
      if (Ti_running == YES):
        e_Ti = e_Ti + P_Ti_BASE * (timeSim() - init_Ti) * 0.000001

      Ti_running = NO
      state = IDLE_OFF

      if (summed_p_IDLE == NO):
        e_idle = e_idle + P_ON_IDLE * (timeSim() - entry_to_IDLE_ON) * 0.000001

      Schedule(event(user_to_IDLE_ON, T_OFF_IDLE, 2))

      entry_to_IDLE_ON = timeSim() + T_OFF_IDLE
      summed_p_IDLE = NO
      just_entry_to_IDLE_ON  = YES
      e_idle = e_idle + P_IDLE_BASE * T_OFF_IDLE * 0.000001


  if opcion == user_to_IDLE_ON:

    state = IDLE_ON

    if (int(round(tk_app2)) > 0 and promoting == NO):
      Schedule(event(user_to_CRX, T_PRO, 2))
      promoting = YES
      e_pro = e_pro + P_PRO * T_PRO * 0.000001

      if (just_entry_to_IDLE_ON == YES):
        aux_real = 1.0
      else:
```

```
            aux_real = timeSim() - entry_to_IDLE_ON

        e_idle = e_idle + P_ON_IDLE * aux_real * 0.000001
        summed_p_IDLE = YES

      else:
        if (just_entry_to_IDLE_ON == YES):
          Schedule(event(user_to_IDLE_OFF, T_ON_IDLE, 2))

      just_entry_to_IDLE_ON = NO

    if opcion == new_packet:
      PSize = random.expovariate(1.0/PS)

      if (PSize <= kbits_remaining):
        tk_app2 = tk_app2 + PSize
        kbits_remaining = kbits_remaining - PSize
      else:
        tk_app2 = tk_app2 + kbits_remaining
        kbits_remaining = 0.0

      if (state == CRX):
        leaving = 0
        while (leaving != -1):
          leaving = deleteEventType(user_to_IDLE_OFF)

        if (Ti_running == YES):
          e_Ti = e_Ti + P_Ti_BASE * (timeSim() - init_Ti) * 0.000001

        Ti_running = NO

      if (state == IDLE_ON):
        just_entry_to_IDLE_ON = NO
        Schedule(event(user_to_IDLE_ON, 0.0, 2))

      Schedule(event(new_packet,random.expovariate(1.0/interpacket), tk))


    end_reschedule()

print(e_pro, e_Ti, e_data, e_idle, (e_pro + e_Ti + e_data + e_idle))
```

Modelo energético de WiFi en C# y Visual Basic

Los códigos del modelo energético de WiFi en C# y Visual Basic se incluyen en el Anexo IX y Anexo X respectivamente.

Descripción experimental

Modelo M/M/1

Probabilidades de estado

Hemos realizado unos cálculos de 8 probabilidades de estado, desde P0 hasta P7. El experimento se ha realizado 5 veces cambiando la semilla que se utiliza para generar los números aleatorios y fijando el valor de los servidos igual a 1.000.000. Tras realizar los experimentos hemos hecho la media y comparado el valor entre los distintos lenguajes y el valor teórico.

C	AVG	STD	CI	Valor teórico
P0	0,499226	0,001006104	0,001249244	0,5
P1	0,2500874	0,000638726	0,000793082	0,25
P2	0,1253572	0,000351588	0,000436554	0,125
P3	0,0627114	0,000328567	0,00040797	0,0625
P4	0,031384	0,000225159	0,000279572	0,03125
P5	0,0156152	0,000165329	0,000205283	0,015625
P6	0,007794	0,000153979	0,00019119	0,0078125
P7	0,003933	7,18157E-05	8,9171E-05	0,00390625

Tabla 4: Probabilidades de estado del modelo M/M/1 en C

Python	AVG	STD	CI	Valor teórico
P0	0,49949272	0,000372346	0,000462329	0,5
P1	0,25021556	0,000566389	0,000703264	0,25
P2	0,125174043	0,00040089	0,00049777	0,125
P3	0,062580461	7,31576E-05	9,08372E-05	0,0625
P4	0,03130695	0,000171671	0,000213158	0,03125
P5	0,015675432	0,000163813	0,0002034	0,015625
P6	0,007810208	9,84921E-05	0,000122294	0,0078125
P7	0,003901885	0,000110442	0,000137132	0,00390625

Tabla 5: Probabilidades de estado del modelo M/M/1 en Python

Visual Basic	AVG	STD	CI	Valor teórico
P0	0,50004146	0,000865527	0,001074694	0,5
P1	0,250102156	0,000358859	0,000445583	0,25
P2	0,124787728	0,000255133	0,00031679	0,125
P3	0,062541166	0,00020771	0,000257906	0,0625
P4	0,031091874	0,000249309	0,000309558	0,03125
P5	0,01566189	0,000214159	0,000265914	0,015625
P6	0,007855433	0,000100312	0,000124554	0,0078125
P7	0,00393474	6,65761E-05	8,26651E-05	0,00390625

Tabla 6: Probabilidades de estado del modelo M/M/1 en Visual Basic

C#	AVG	STD	CI	Valor teórico
P0	0,499658669	0,000511948	0,000635668	0,5
P1	0,250213982	0,000418344	0,000519442	0,25
P2	0,125056936	0,000296294	0,000367898	0,125
P3	0,062565483	0,00013483	0,000167414	0,0625
P4	0,03121438	0,000140172	0,000174046	0,03125
P5	0,015643645	9,1885E-05	0,00011409	0,015625
P6	0,007854679	4,72074E-05	5,86157E-05	0,0078125
P7	0,003908063	5,44566E-05	6,76169E-05	0,00390625

Tabla 7: Probabilidades de estado del modelo M/M/1 en C#

Podemos comprobar como no hay apenas diferencias en los resultados de los diferentes lenguajes, ya que todos se ajustan al valor teórico de manera muy parecida, por lo que podemos validar tanto el modelo MM1 como los diferentes simuladores desarrollados.

Tiempos de ejecución

A continuación incluimos unas tablas para las cuales hemos realizado 10 ejecuciones para 1.000, 10.000, 100.000 y 1.000.000 de servidos en cada uno de los lenguajes. Hemos calculado el valor medio (avg), desviación estándar (std) y los intervalos de confianza, (CI) para luego hacer las medias de cada uno.

Todos los valores temporales, están expresados en **milisegundos**.

	C			C#		
Servidos	**avg**	**std**	**CI**	**avg**	**std**	**CI**
1000	0,90	0,32	0,23	3,12	6,58	4,71
10000	3,40	0,52	0,37	9,36	8,06	5,76
100000	28,40	0,52	0,37	87,36	8,06	5,76
1000000	277,10	1,10	0,79	804,96	8,06	5,76

	VB			Python		
Servidos	**avg**	**std**	**CI**	**avg**	**std**	**CI**
1000	7,80	8,22	5,88	16,77	0,25	0,18
10000	28,08	6,58	4,71	136,49	0,28	0,20
100000	276,12	7,54	5,39	1333,35	2,09	1,50
1000000	2552,16	8,06	5,76	13286,10	22,77	16,29

Tabla 8: Tiempos de ejecución del modelo M/M/1

Hemos representado gráficamente los resultados para poder ver las comparaciones de mejor manera, incluyendo los intervalos de confianza al 95%.

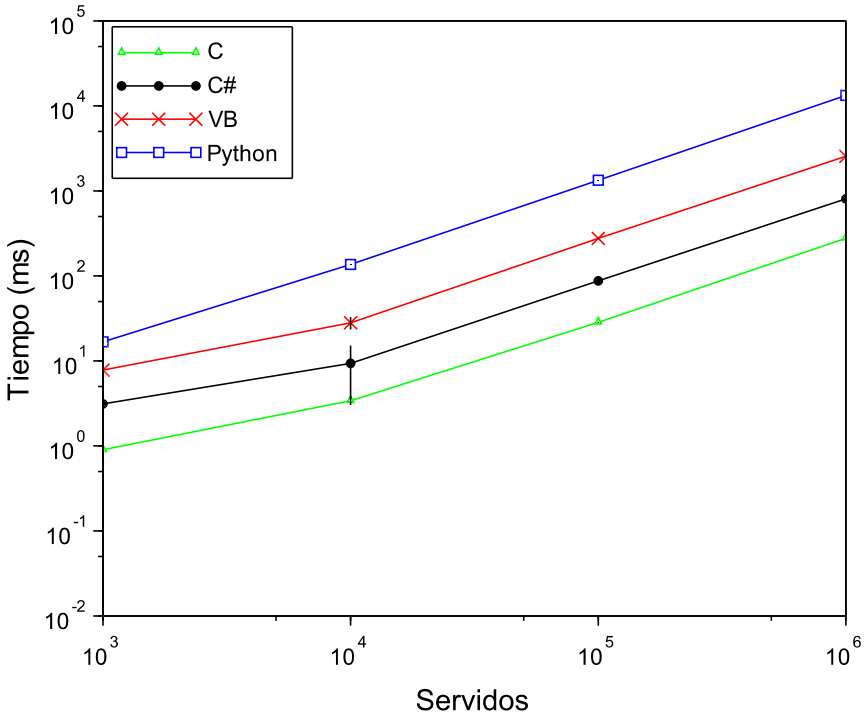


Ilustración 7: Tiempos de ejecución M/M/1

Podemos ver en la gráfica anterior, como C es el lenguaje más rápido, seguido de C#, Visual Basic y finalmente Python.

En la siguiente gráfica haremos una comprobación de los lenguajes utilizados (Python, C# y Visual Basic), pero comparándolos en función de los tiempos de C.

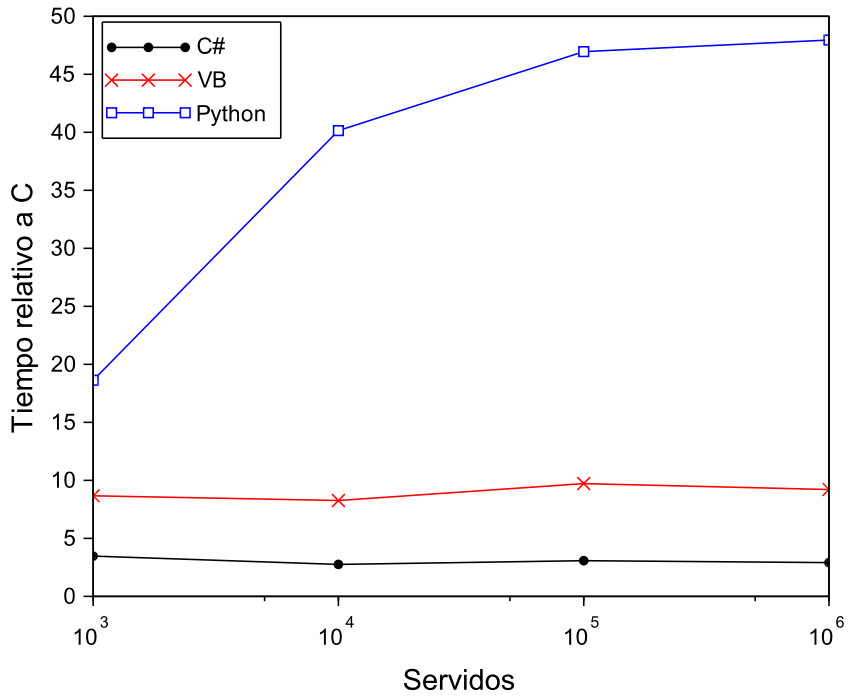


Ilustración 8: Tiempos relativos a C en M/M/1

Cómo podemos apreciar, C# es aproximadamente 3 veces más lento que C y Visual Basic es 8,5 veces más lento que C, y ambos se mantienen más o menos constantes conforme aumentan los servidos. Sin embargo Python empieza siendo unas 18 veces más lento que C y crece con mucha pendiente en un principio, haciéndose más constante conforme aumentan los servidos.

Modelo energético de WiFi en terminales móviles

Energías

Hemos realizado unos cálculos de las energías. El experimento se ha realizado 7 veces cambiando la cantidad de kits desde 10^3 hasta 10^9, en los diferentes lenguajes.

	C				
	Epro	**Eti**	**Edata**	**Eidle**	**Etotal**
10^3	0	0,015	0,1403	0	0,1553
10^4	0,0197	0,1076	1,0599	0,007	1,1942
10^5	0,0886	0,8333	14,0026	0,0315	14,956
10^6	0,6002	7,0495	140,3309	0,2297	148,2103
10^7	5,5793	64,361	1403,2145	2,0914	1475,2463
10^8	53,6774	642,7838	14033,1554	20,2042	14749,8209
10^9	535,5639	6379,8654	140331,3183	201,6184	147448,366

Tabla 9: Energías del modelo energético de WiFi en C

	Python				
	Epro	**Eti**	**Edata**	**Eidle**	**Etotal**
10^3	0,00000	0,02153	0,14033	0,00000	0,16186
10^4	0,00000	0,04707	1,32956	0,00000	1,37663
10^5	0,04920	0,61565	14,03360	0,01752	14,71597
10^6	0,51168	6,14025	140,33048	0,19425	147,17666
10^7	5,26442	64,29415	1402,33785	1,95883	1473,85525
10^8	53,47078	637,88567	14032,78540	20,08326	14744,22511
10^9	534,99313	6368,43620	140331,47071	201,21680	147436,11684

Tabla 10: Energías del modelo energético de WiFi en Python

	Visual Basic				
	Epro	**Eti**	**Edata**	**Eidle**	**Etotal**
10^3	0,00000	0,00746	0,14033	0,00000	0,14779
10^4	0,00984	0,11563	1,19195	0,00693	1,32435
10^5	0,05904	0,65689	13,79620	0,02102	14,53315
10^6	0,61992	6,73306	140,33024	0,22878	147,91200
10^7	5,18570	63,09910	1403,31705	1,94295	1473,54480
10^8	52,25061	636,25838	14033,15577	19,68873	14741,35349
10^9	538,60443	6372,67301	140331,49517	202,37387	147445,14648

Tabla 11: Energías del modelo energético de WiFi en Visual Basic

	C#				
	Epro	**Eti**	**Edata**	**Eidle**	**Etotal**
10^3	0,00000	0,00000	0,00953	0,00000	0,00953
10^4	0,00984	0,04298	0,87941	0,00350	0,93573
10^5	0,05904	0,51222	14,03298	0,02102	14,62527
10^6	0,53136	5,98801	139,84436	0,21283	146,57657
10^7	5,62850	62,79969	1403,31339	2,14084	1473,88243
10^8	52,42773	635,54970	14033,15146	19,76727	14740,89616
10^9	538,46667	6380,48618	140331,50770	202,09710	147452,55766

Tabla 12: Energías del modelo energético de WiFi en C#

A partir de estos resultados, podemos validar el funcionamiento de los diferentes simuladores.

Tiempos de ejecución

A continuación incluimos unas tablas para las cuales hemos realizado 10 ejecuciones entre 10^3 y 10^9 kbits en cada uno de los lenguajes. Hemos calculado el valor medio (avg), desviación estándar (std) y los intervalos de confianza, (CI) para luego hacer las medias de cada uno.

Todos los valores temporales, están expresados en **milisegundos**.

	C			C#		
kbits	**avg**	**std**	**CI**	**avg**	**std**	**CI**
10^3	0,60	0,52	0,37	4,68	7,54	5,39
10^4	0,70	0,48	0,35	4,68	7,54	5,39
10^5	0,80	0,42	0,30	1,56	4,93	3,53
10^6	1,00	0,00	0,00	4,68	7,54	5,39
10^7	4,00	0,47	0,34	14,04	4,93	3,53
10^8	30,30	0,95	0,68	112,32	6,58	4,71
10^9	289,70	1,16	0,83	1087,32	39,64	28,35

Tabla 13: Tiempos de ejecución del modelo energético de WiFi en C y C#

	VB			Python		
kbits	**avg**	**std**	**CI**	**avg**	**std**	**CI**
10^3	3,12	6,58	4,71	0,07	0,02	0,01
10^4	1,56	4,93	3,53	0,22	0,05	0,03
10^5	1,56	4,93	3,53	1,73	0,12	0,08
10^6	7,80	8,22	5,88	15,91	0,58	0,41
10^7	29,64	4,93	3,53	160,46	2,28	1,63
10^8	232,44	13,66	9,77	1608,63	18,17	13,00
10^9	2199,60	270,40	193,43	15966,47	13,04	9,33

Tabla 14: Tiempos de ejecución del modelo energético de WiFi en Visual Basic y Python

En el caso del modelo energético de WiFi en un terminal móvil, Python es el más rápido hasta que los kbits no superen 10^4, que, en ese momento, la línea que representa a Python aumenta su pendiente considerablemente. Siguiendo a Python se encuentran de C, Visual Basic y por último C#.

Conforme aumentan los kbits, cuando el valor es 10^5, observamos que todos los lenguajes tienen un tiempo de ejecución muy parecido, encontrándose todos prácticamente en el mismo punto. Salvo C, que es ligeramente más rápido.

A partir de 10^5, Python sigue creciendo con aproximadamente la misma pendiente, colocándose en el más lento de los 4. Visual Basic intercambia el lugar con C# colocándose en tercer lugar, C# se coloca el segundo más rápido, y una vez más C acaba siendo el más rápido.

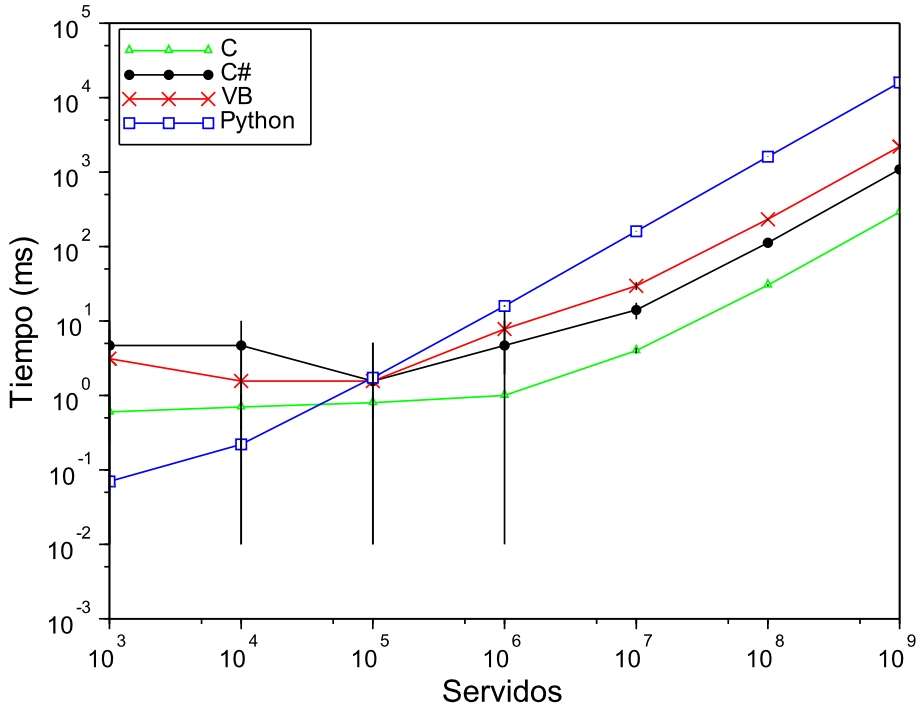


Ilustración 9: Tiempos de ejecución modelo WiFi

A continuación incluimos una gráfica con los tiempos de los diferentes lenguajes, comparados con los tiempos de ejecución en C.

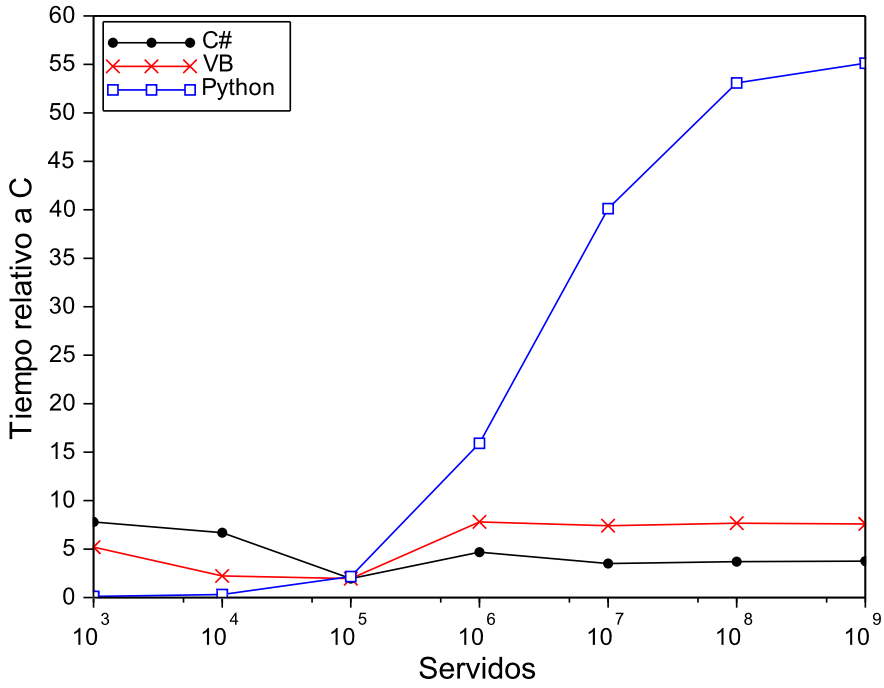


Ilustración 10: Tiempos relativos a C del modelo WiFi

Podemos observar claramente como los lenguajes Visual Basic y C# empiezan siendo entre 5 y 7 veces más lentos que C respectivamente. Sin embargo Python, aunque no se aprecia en la gráfica está por debajo de C.

Como mencionábamos en el apartado anterior, en 10^5 se aprecia como todos los lenguajes prácticamente coinciden en os tiempos, y a partir de ese momento es cuando Python empieza a aumentar sus tiempos de ejecución, mientras que C# y Visual Basic, crecen con una pendiente muy parecida que en C.

Conclusiones

Hemos observado como el lenguaje más rápido sin duda alguna es C tanto en el modelo M/M/1 como en el modelo energético de WiFi.
Sin embargo C# y Visual Basic se mantienen en la zona media de la gráfica, entre los tiempos de C y los de Python, siendo C# más rápido en el modelo WiFi, pero más lento que Visual Basic en el modelo M/M/1.

Python es el lenguaje más lento, y además, su comportamiento en ambos modelos es muy diferente. En el modelo M/M/1 empieza siendo más lento que el resto de lenguajes, y creciendo con una pendiente muy inclinada, bajando la pendiente considerablemente, tendiendo a ser constante conforme avanzan los servidos. Por el contrario, en el modelo WiFi, Python es el lenguaje más rápido hasta que los kbits son iguales a 10^4. A partir de ahí la pendiente crece de forma drástica hasta los 10^8 kbits, que de nuevo empieza a estabilizarse con una pendiente más constante.

Es importante mencionar, que durante todos los resultados los intervalos de confianza permanecen muy bajos, exceptuando en el modelo WiFi, en el que observamos que, en algunos puntos, los intervalos de confianza aumentan considerablemente.

Podemos afirmar, que con los resultados obtenidos, comprobamos la importancia del hecho de que un lenguaje sea compilado, o no, ya que queda reflejado en sus tiempos de ejecución. C es un lenguaje de más bajo nivel, lo cual hace que sus resultados sean excepcionales. Visual Basic y C# tienen comportamientos parecidos, pero más lentos que C, ya que son lenguajes a más alto nivel que C, pero también compilados. Sin embargo Python, es un lenguaje de alto nivel, y no es compilado.
Aunque en estas ejecuciones en Python, el simulador, junto con los archivos importados, han sido compilados, sin embargo el programa principal, (modelo M/M/1 o modelo WiFi), no están compilados, y eso hace que Python tenga la ventaja de ser un lenguaje fácil de usar por su alto nivel, pero sufre una penalización en los tiempos de ejecución.

Esto es algo a tener en cuenta. Si lo que se pretende es que el simulador sea lo más rápido posible, habría que optar por usar el simulador en C, sin embargo, si el tiempo de ejecución no es algo tan importante, Python nos permite una flexibilidad y facilidad de uso que otros lenguajes no nos ofrecen.

Pliego de condiciones

Para la realización del proyecto utilizaremos diferentes herramientas:

- Equipo informático.
- Herramientas de desarrollo.

A continuación las comentamos.

Equipo para desarrollo

Para instalar las herramientas de desarrollo y poder realizar el proyecto en un equipo físico, utilizaremos un ordenador portátil con las siguientes características.

Ilustración 11: OS X El Capitán

El desarrollo completo se realizará en OS X El Capitán, el cual incluye una máquina virtual con Windows 7 y Ubuntu Linux 12 para realizar pruebas de rendimiento del simulador en los sistemas operativos más importantes.

Herramienta de software

PyCharm

Para desarrollar el generador de eventos discretos en PYTHON utilizaremos una herramienta llamada PyCharm CE. Es un IDE (Entorno de Desarrollo Integrado) libre propiedad de JetBrains, basado en IntelliJ IDEA.

Ilustración 12: PyCharm

PyCharm es un IDE o entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica, integración con VCS / DVCS y soporte para el desarrollo web con Django, entre otras bondades.

PyCharm es desarrollado por la empresa JetBrains y debido a la naturaleza de sus licencias tiene dos versiones, la *Community* que es gratuita y orientada a la educación y al desarrollo puro en Python y la *Professional*, que incluye más características como el soporte a desarrollo web.

Las características principales de PyCharm son:

- Autocompletado, resaltador de sintaxis, herramienta de análisis y refactorización.
- Integración con frameworks web como: Django, Flask, Pyramid, Web2Py.
- Frameworks javascripts: jQuery, AngularJS.
- Debugger avanzado de Python y Javascript.
- Integración con lenguajes de plantillas: Mako, Jinja2, Django Template.
- Compatibilidad con SQLAlchemy (ORM), Google App Engine, Cython.
- Soporte para modo VIM (Con plugin).
- Sistemas de control de versiones: Git, CVS, Mercurial.

Scilab

Para generar las gráficas de los intervalos de confianza y las comparaciones de tiempos de ejecución hemos utilizado una herramienta llamada Scilab.

Scilab es una herramienta de análisis numérico. Utiliza un lenguaje de programación de alto nivel para cálculo científico. Incluye visualización 2D y 3D, optimización, análisis estadístico, diseño y análisis de sistemas dinámicos, procesamiento de señales, e interfaces con Fortan, Java, C y C++.

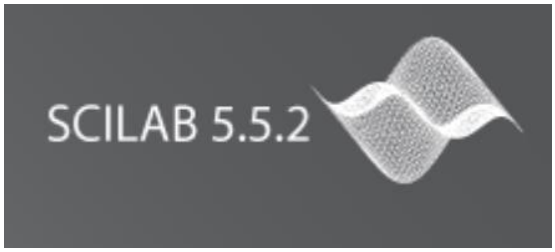


Ilustración 13: Scilab

Herramienta para el desarrollo de la memoria

Word Mac 2011

El software elegido para realizar la memoria es Microsoft Word 2011 para Mac.
Es una aplicación informática para el procesamiento de textos, creado por Microsoft, integrado en el paquete Office.

Ilustración 14: Word Mac 2011

Microsoft Word soporta diferentes formatos como DOC, RTF y contiene plug-ins para entender otros formatos.

Es la aplicación más extendida en el procesamiento de textos, por todos estos motivos es la herramienta seleccionada.

Excel Mac 2011

Para la realización de os presupuestos, diagramas de Gantt, procesado de resultados y otras secciones de la memoria hemos utilizado Microsoft Excel 2011 para Mac.

Ilustración 15: Excel Mac 2011

Microsoft Excel es una aplicación distribuida por Microsoft Office para hojas de cálculo utilizado en tareas financieras y contables.

Lenguajes de programación

Definiremos programación como una estructura que, con una cierta base sintáctica y semántica, imparte distintas instrucciones a un programa de computadora

Es el procedimiento del código fuente de un software, con el cual le decimos a un programa informático, qué tiene que hacer y cómo tiene que hacerlo.

Actualmente el mundo de la programación cada vez está creciendo más, ya que el hardware en muchos aspectos está más avanzado que el software, y necesitamos tener programas más potentes.

A la hora de elegir un lenguaje de programación, hay muchos factores a tener en cuenta. Hay lenguajes más modernos como Python, Objective C, Javascript, y otros algo más antiguos como C, C++, C#, Java, Visual Basic.

Las diferencias entre ellos son diversas como el tipo de sintaxis, los entornos de desarrollo utilizados, etcétera.

La diferencia más significativa entre los lenguajes que vamos a utilizar en nuestro simulador, es que C#, Visual Basic y C son compilados, y Python es interpretado, aunque Python nos permite compilar los archivos que se importan al programa principal, cosa que utilizaremos para mejorar los tiempos del simulador.

Lenguajes interpretados

Un lenguaje interpretado es aquel en el cual sus instrucciones o código fuente, está escrito en un lenguaje de alto nivel.
Este es traducido por el intérprete a un lenguaje entendible para la máquina paso a paso, instrucción por instrucción. El proceso se repite cada vez que se ejecuta el programa el código en cuestión.

Los lenguajes interpretados permiten el tipado dinámico de datos, es decir, no es necesario inicializar una variable con determinado tipo de dato sino que esta puede cambiar su tipo en condición al dato que almacene entre otras características más.

La principal desventaja de estos lenguajes es el tiempo que necesitan para ser interpretados. Al tener que ser traducido a lenguaje máquina con cada ejecución, este proceso es más lento que en los lenguajes compilados, sin embargo, algunos lenguajes poseen una máquina virtual que hace una traducción a lenguaje intermedio con lo cual el traducirlo a lenguaje de bajo nivel toma menos tiempo.

En este proyecto utilizamos Python para realizar el simulador.

Lenguajes compilados

Un lenguaje compilado es aquel cuyo código fuente está escrito en un lenguaje de alto nivel. Este es traducido por un compilador a un archivo ejecutable entendible para la máquina en determinada plataforma.
Con ese archivo se puede ejecutar el programa cuantas veces sea necesario sin tener que repetir el proceso por lo que el tiempo de espera entre ejecución y ejecución es ínfimo.

Principalmente vemos los lenguajes interpretados en el desarrollo de aplicaciones o sitios web que van acompañados de frameworks que facilitan en gran medida su programación.

Esto se da ya que no es necesario que el usuario final posea, en el caso de los lenguajes compilados, el compilador instalado en su ordenador para ejecutar el programa o el archivo objeto que este produce.
Mayoritariamente necesitan un navegador actualizado y conexión a Internet para acceder y usar aplicaciones en línea.

En este proyecto, los lenguajes compilados usados son C, C# y Visual Basic.

Python

Python es un lenguaje de programación creado por Guido van Rossum a finales de los ochenta, y que gracias a sus características ha llegado a ser un lenguaje muy conocido en la actualidad.

Ilustración 16: Python

Python es Open Source, ha sido modificado para poder funcionar en diferentes plataformas como Linux, Windows, Macintosh, Solaris y muchas más.

Es un lenguaje orientado a objetos, combinando datos y funcionalidades. Es un lenguaje de alto nivel, por lo que no nos debe preocupar los detalles como manejar la memoria, o recolectar la "basura".

Es incrustable pudiéndose insertar el lenguaje Python dentro de un programa C / C++ y gracias a esto, nos ofrece grandes facilidades de scripting.

Contiene extensas librerías, tipos de datos, funciones incorporadas en el propio lenguaje que ayudan a realizar tareas comunes sin necesidad de tener que programarlas nosotros.

Gracias a estas librerías podemos realizar tareas como expresiones regulares, generar documentos, evaluar unidades, pruebas, procesos, interactuar con bases de datos, servicios ftp, web, correo electrónico, etcétera.

Contiene una sintaxis clara y visual gracias a la identación (con márgenes), de carácter obligatorios, que hace que el código esté, muy organizado y que todo programador pueda entender de forma rápida y sencilla los códigos que herede en Python, ya que todos los programas tienen un aspecto bastante similar.

C

C es un lenguaje de programación orientado a la implementación de sistemas operativos. Fue desarrollado originariamente por Dennis M.Ritchie hacia el 1970. Es una evolución de su predecesor lenguaje B.
C es el lenguaje más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.

Ilustración 17: C

Es un lenguaje de medio nivel, pero dispone de construcciones que permiten controlar a muy bajo nivel, posibilitando mezclar código ensamblador con código en C, accediendo así directamente a memoria gracias a los punteros o a los periféricos.

Tiene un núcleo de lenguaje simple, con funciones añadidas como operaciones matemáticas y manejo de archivos.

En las funciones los parámetros se realizan por valor, aunque podemos pasarle la dirección de la variable en memoria pasándole el argumento por referencia.

Se puede lograr polimorfismo y encapsulado utilizando punteros a funciones.

Como carencia podríamos decir que no tiene un recolector de basura, y hay que controlar por ejemplo la eliminación de los punteros utilizados en nuestro programa.

Visual C#

C# es un lenguaje de programación orientado a objetos, desarrollado y estandarizado por Microsoft como parte de su plataforma .NET.
C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

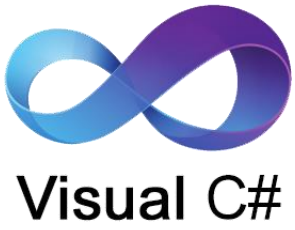


Ilustración 18: Visual C#

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

C# nos permite crear aplicaciones seguras gracias a la seguridad de tipos, ejecutándose en plataformas .NET Framework. Se pueden crear aplicaciones de Windows, servicios Web, aplicaciones cliente-servidor, aplicaciones de bases de datos.

La potencia de C# reside, aparte de lo anterior, en las bibliotecas de clases que nos ofrece Microsoft con funcionalidades muy completas.
Al ser un lenguaje orientado a objetos nos permite crear clases, objetos, usar herencias, y polimorfismos.

Como novedades más significativas tenemos métodos encapsulados llamados delegados, los cuales habilitan notificaciones de eventos con seguridad de tipos. Podemos usar propiedades para describir el acceso a las variables de miembro privadas.

Una gran utilidad de C# que nos proporciona funciones de consulta integradas, tanto para manejar listas, incluso para generar queries para interactuar con gestores de bases de datos a través de contextos previamente configurados.

Visual Basic

Visual Basic fue diseñado para la creación de aplicaciones de manera productiva con seguridad de tipos y orientada a objetos, pudiendo crear aplicaciones Windows, Web e incluso aplicaciones para dispositivos móviles.

Ilustración 19: Visual Basic

Al igual que C#, explicado en el punto anterior, las aplicaciones creadas con Visual Basic están creadas en .NET Framework.

Es un lenguaje de programación dirigido por eventos, desarrollado por Allan Cooper para Microsoft. Es un dialecto de BASIC.

Contiene un entorno de desarrollo integrado (IDE) que integra un editor de textos para editar el código fuente, un compilador, un depurador y un editor de interfaces gráficas (GUI).

Plan de implantación

Para la realización de este proyecto se ha seguido un plan de implantación con los siguientes pasos:

- Recopilación de la documentación necesaria con su lectura y comprensión. Los documentos más importantes para este proyecto han sido:
  - *"Simulating Computer Systems: Tecniques and Tools", MacDougall.*
  - *"Python Tutorial", Guido van Rossum.*
  - *"TFG – Simulación y estudio del consumo energético de terminales móviles en redes WiFi", Laura Sánchez, Universidad de Alcalá.*
- Recopilación del software y hardware necesario para la constitución del proyecto completo.
- Realización del simulador de eventos discretos en Python, Visual Basic y C#.
- Realización del modelo M/M/1 en Python, Visual Basic y C#.
- Realización del modelo WiFi en Python, Visual Basic y C#
- Elaboración de la memoria, presupuesto, pliego de condiciones.

A continuación incluimos el diagrama de Gantt del proyecto:

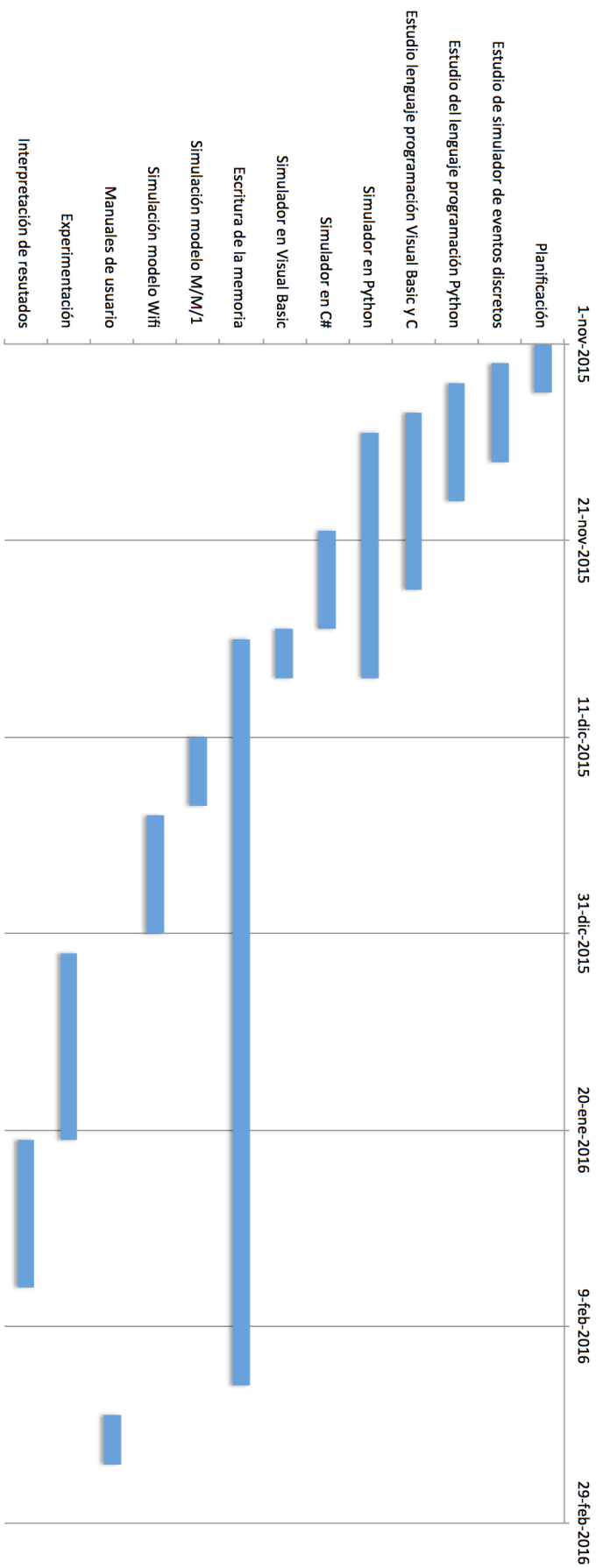


Ilustración 20: Diagrama de Gantt

Presupuesto

En este apartado detallamos cada uno de los gastos que implica la realización del proyecto, incluyendo los materiales necesarios, software utilizado y mano de obra.

Incluiremos los presupuestos detallados y un presupuesto final que resuma todo lo descrito a continuación.

Materiales

Equipo	Total (sin IVA)
MacBook Pro Retina 13'', Intel Core i5 2,6 GHz, 8GB 1600MHz DDR3, Intel Iris Pro 1536 MB	1.163,00 €
Línea ADSL Telefónica Fibra óptica 30Mb – 12,314 €/mes (sin IVA)	49,25 €
Total costes materiales	**1.212,25 €**

Tabla 15: Presupuesto de materiales

Software

Software	Total (sin IVA)
Licencia Office para Mac	69,08 €
Licencia PyCharm (1 año)	164,46 €
Licencia Scilab	0,00 €
Licencia Windows 7 profesional 64 bits	24,71 €
Licencia Microsoft Visual Studio 2012	451,24 €
Total costes software	**709,49 €**

Tabla 16: Presupuesto de software

Personal

Concepto	Coste por hora	Total horas	Total (sin IVA)
Diseño de ingeniería	61,98 €	340 horas	21.073,20 €
Documentación del proyecto	15,70 €	120 horas	1.884,00 €
Total costes de personal			**22.957,20 €**

Tabla 17: Presupuesto de personal

Presupuesto final

Concepto	Coste sin IVA	Coste con IVA (21%)
Materiales		1.212,25 €
Software	709,49 €	858,48 €
Personal	22.957,20 €	27.778,21 €
Totales		**29.848,94 €**

Tabla 18: Presupuesto final

Manual de usuario

Simulador en C#

A continuación vamos a explicar cómo utilizar el simulador en C#.

Lo primero que debemos hacer es declarar un objeto de tipo Simulador.

```
Simulador Simulador = new Simulador();
```

De esta manera tendremos acceso a toda la funcionalidad que nos presenta el simulador,

El simulador debe ser iniciado por lo que debemos llamar a la función InitSimulator() al principio de nuestro programa.

```
Simulador.init_Simulation();
```

Para poder utilizar eventos para introducir en el simulador debemos utilizar la clase "Event". Esta clase contiene 3 propiedades:

- Type: es una propiedad de tipo entero para fijar el tipo de evento en el caso de que usemos varios tipos.
- Time: es una propiedad de tipo Double para poder informar del tiempo en que se planifica el evento.
- Optional: es una propiedad de tipo string, para poder introducir parámetros extra, pudiendo introducir un array de datos.

En el siguiente ejemplo, creamos un evento cuyos parámetros serán:

- Evento de tipo 1
- Tiempo = 0.4 s
- Texto opcional = "optional"

```
Evento evento = new Evento(1, 0.4, "option");
```

A partir de ahora podremos introducir eventos en el simulador con la función "Schedule". Esta función acepta como único parámetro un evento.

```
Simulador.Schedule(new Evento(1, 0.0, "0"));
```

Esta función introducirá en una lista de eventos, el evento ordenado de forma ascendente, basándose en la marca de tiempo.

Podremos llamar a la función Schedule hasta que queramos extraer algún evento.

Para extraer los eventos de uno en uno se utilizará la función "Cause".

```
Evento eventoRecibido = Simulador.Cause();
```

La función Cause devuelve el primer evento de tipo Evento, y fijará el tiempo de simulación con el tiempo del evento devuelto, para tenerlo en cuenta en eventos próximos.

Otra funcionalidad del simulador es poder obtener información. Para ello varias funciones como "EventList" que devuelve la lista actual de eventos.

```
Simulador.EventList();
```

Si necesitamos saber el tiempo de simulación actual, podremos usar la función "TimeSim" que nos devolverá el tiempo de simulación.

```
Simulador.TimeSim();
```

El simulador nos da la opción también de eliminar eventos de la lista. Para ello se puede utilizar la función "deleteEventType". Esta función acepta un parámetro de tipo entero, que será el tipo de los eventos que queremos eliminar.

```
Simulador.deleteEventType();
```

Tenemos la opción si fuese necesario de vaciar la lista de eventos. Para realizarlo, el simulador nos ofrece la función "deleteAllEvents".

```
Simulador.deleteAllEvents();
```

Simulador en Visual Basic

Visual Basic está diseñado para la creación de aplicaciones de manera productiva con seguridad de tipos y orientado a objetos. Visual Basic permite a los desarrolladores centrar el diseño en Windows, la web y dispositivos móviles. Como ocurre con todos los lenguajes destinados a Microsoft .NET Framework, los programas escritos en Visual Basic se benefician de la seguridad y la interoperabilidad de los lenguajes.
Esta generación de Visual Basic continúa la tradición de ofrecerle una manera rápida y fácil de crear aplicaciones basadas en .NET Framework

A continuación vamos a explicar como utilizar el simulador en Visual Basic.

Lo primero que debemos hacer es declarar un objeto de tipo Simulador.

```
Dim Simulador As New Simulador
```

De esta manera tendremos acceso a toda la funcionalidad que nos presenta el simulador,

El simulador debe ser iniciado por lo que debemos llamar la función "InitSimulation" al principio de nuestro programa.

```
Simulador.init_simulation()
```

Para poder utilizar eventos para introducir en el simulador debemos utilizar la clase "Event". Esta clase contiene 3 propiedades:

- Type: es una propiedad de tipo entero para fijar el tipo de evento en el caso de que usemos varios tipos.
- Time: es una propieda de tipo Double para poder informar del tiempo en que se planifica el evento.
- Optional: es una propiedad de tipo string, para poder introducir parámetros extra, pudiendo introducir un array de datos.

En el siguiente ejemplo, creamos un evento cuyos parámetros serán:

- Evento de tipo 1
- Tiempo = 0.4 s
- Texto opcional = “optional”

```
Dim evento As New Evento(1, 0.4, "option")
```

A partir de ahora podremos introducir eventos en el simulador con la función “Schedule”. Esta función acepta como único parámetro un evento.

```
Simulador.Schedule(new Evento(1, 0.0, "0"))
```

Esta función introducirá en una lista de eventos, el evento ordenado de forma ascendente, basándose en la marca de tiempo.

Podremos llamar a la función Schedule hasta que queramos extraer algún evento.

Para extraer los eventos de uno en uno se utilizará la función “Cause”.

```
Dim eventoRecibido As Evento = Simulador.Cause()
```

La función Cause devuelve el primer evento de tipo Evento, y fijará el tiempo de simulación con el tiempo del evento devuelto, para tenerlo en cuenta en eventos próximos.

Otra funcionalidad del simulador es poder obtener información. Para ello varias funciones como “EventList” que devuelve la lista actual de eventos.

```
Simulador.EventList()
```

Si necesitamos saber el tiempo de simulación actual, podremos usar la función “TimeSim” que nos devolverá el tiempo de simulación.

```
Simulador.TimeSim()
```

El simulador nos da la opción también de eliminar eventos de la lista. Para ello se puede utilizar la función “deleteEventType”. Esta función acepta un parámetro de tipo entero, que será el tipo de los eventos que queremos eliminar.

```
Simulador.deleteEventType()
```

Tenemos la opción si fuese necesario de vaciar la lista de eventos. Para realizarlo, el simulador nos ofrece la función “deleteAllEvents”.

```
Simulador.deleteAllEvents()
```

Simulador en Python

Python es un lenguaje de programación creado por Guido van Rossum a finales de los ochenta, y que gracias a sus características ha llegado a ser un lenguaje muy conocido en la actualidad.

Python es un lenguaje open source, está orientado a objetos, también es incrustable, tiene librerías extensas y una sintaxis clara.

A continuación explicamos como utilizar el simulador.

Primeramente es necesario importar la librería de funciones que nos ofrece el simulador. Para ello importamos la librería "Imports" de la siguiente manera:

```
from Imports import *
```

El simulador debe ser iniciado por lo que debemos llamar la función "InitSimulation" al principio de nuestro programa.

```
init_simulation()
```

Para poder utilizar eventos para introducir en el simulador debemos utilizar la clase "Event". Esta clase contiene 3 propiedades:

- Type: es una propiedad para fijar el tipo de evento en el caso de que usemos varios tipos.
- Time: es una propiedad para poder informar del tiempo en que se planifica el evento.
- Optional: es una propiedad para poder introducir parámetros extra, pudiendo introducir cualquier tipo de dato.

En el siguiente ejemplo, creamos un evento cuyos parámetros serán:

- Evento de tipo 1
- Tiempo = 0.4 s
- Texto opcional = "optional"

```
evento = event(1, 0.4, "option")
```

A partir de ahora podremos introducir eventos en el simulador con la función "Schedule". Esta función acepta como único parámetro un evento.

```
Schedule(event(1, 0.0, "0"))
```

Esta función introducirá en una lista de eventos, el evento ordenado de forma ascendente, basándose en la marca de tiempo.

Podremos llamar a la función Schedule hasta que queramos extraer algún evento.

Para extraer los eventos de uno en uno se utilizará la función "Cause".

```
evento = Cause()
```

La función Cause devuelve el primer evento de tipo Evento, y fijará el tiempo de simulación con el tiempo del evento devuelto, para tenerlo en cuenta en eventos próximos.

Otra funcionalidad del simulador es poder obtener información. Para ello varias funciones como "EventList" que devuelve la lista actual de eventos.

```
EventList()
```

Si necesitamos saber el tiempo de simulación actual, podremos usar la función “TimeSim” que nos devolverá el tiempo de simulación.

```
TimeSim()
```

El simulador nos da la opción también de eliminar eventos de la lista. Para ello se puede utilizar la función “deleteEventType”. Esta función acepta un parámetro de tipo entero, que será el tipo de los eventos que queremos eliminar.

```
deleteEventType()
```

Tenemos la opción si fuese necesario de vaciar la lista de eventos. Para realizarlo, el simulador nos ofrece la función “deleteAllEvents”.

```
deleteAllEvents()
```

Bibliografía

- McDougal – "Simulating Computing System Techniques and Tools"
- https://www.wikipedia.org
- TFG - Simulación y estudio del consumo energético de terminales móviles en redes WiFi, Laura Sánchez
- http://www.est.uc3m.es/esp/nueva_docencia/comp_col_leg/ing_info/io/doc_generica/archivos/tc.pdf
- https://www.jetbrains.com/pycharm/documentation/ (accedida 7 Diciembre 2015)
- Python, https://www.python.org/ (accedida 20 Enero 2016)
- https://pastranamoreno.files.wordpress.com/2011/03/intro-al-gpss.pdf (accedida 12 enero 2016)
- https://msdn.microsoft.com/es-es/default.aspx (accedida en diferentes ocasiones durante la realización de los programas en C# y Visual Basic)
- https://omnetpp.org/ (accedida 20 enero 2016)
- http://www.it.uniovi.es/wp-content/uploads/2013/10/Introduccion_OPNET.pdf (accedida 21 enero 2016)
- http://heather.cs.ucdavis.edu/~matloff/156/PLN/DESimIntro.pdf (accedida 28 enero 2016)

ANEXO I : Código del simulador en Python

Python

Simulador.py

```
#_____________________________
#        NO EXPUESTO
#_____________________________


def set_time(time):
   global simulationTime
   simulationTime = time

def printEvent(event):
      if event.optional == '':
         print "Event type", event.type, "in time", event.time
      else:
         print "Event type", event.type, "in time", event.time, ", optional value", event.optional

#_____________________________
#         EXPUESTO
#_____________________________
class event:
   def __init__(self, type, time, optional=''):
      self.type = type
      self.time = time
      self.optional = optional

def Cause():
   global eventList
   if eventList != []:
      eventoPrimero = eventList.pop(0)
      set_time(eventoPrimero.time)
      return eventoPrimero
   else:
      return 0


def init_simulation():
   set_time(0.0)
   global eventList
   eventList = []

def Schedule(event):
   global eventList
   global simulationTime

   insertPos = 0
   tiempo = event.time + simulationTime
   event.time = tiempo
   for ev in eventList:
      if ev.time < event.time:
         insertPos = insertPos + 1
      else:
         break
```

```
    eventList.insert(insertPos, event)


def printEventList():
    global eventList
    if len(eventList):
        for ev in eventList:
            if ev.optional == '':
                print "Event type", ev.type, "in time", ev.time
            else:
                print "Event type", ev.type, "in time", ev.time, ", optional value", ev.optional
    else:
        print "No events"

def printSimulationTime():
    global simulationTime
    print "Simulation time:", simulationTime, "s"

def timeSim():
    global simulationTime
    return simulationTime


def deleteEventType(type):
    global eventList
    flag = 0
    for ev in eventList:
        if ev.type == type:
            eventList.remove(ev)
            flag=1
            break
    if flag==1:
        return 0
    else:
        return -1

def deleteAllEvents():
    global eventList
    for i in range(0,len(eventList)):
        eventList.pop(0)
    print "List of events deleted"
```

Imports.py

```
from Simulador import init_simulation, Schedule, Cause, \
    event, printEvent, printSimulationTime, printEventList, \
    deleteEventType, deleteAllEvents, timeSim
```

ANEXO II: Ejemplo de cola MM1 en Python

MM1.py

```
from Imports import *
import random, time

t1 = time.clock()

def grabartxt():
   archi=open('ResultadosPythonMM1.txt','a')

   textoInicio = str("\n") + "Servidos= " + str(servidos)
   texto0 =str("\n") + "P0= " + str(tiempo_0/timeSim()) + " Teorico: " + str(1.0-rho)
   texto1 =str("\n") + "P1= " + str(tiempo_1/timeSim()) + " Teorico: " + str(1.0-rho*rho)
   texto2 =str("\n") + "P2= " + str(tiempo_2/timeSim()) + " Teorico: " + str(1.0-rho*rho*rho)
   texto3 =str("\n") + "P3= " + str(tiempo_3/timeSim()) + " Teorico: " + str(1.0-rho*rho*rho*rho)
   texto4 =str("\n") + "P4= " + str(tiempo_4/timeSim()) + " Teorico: " + str(1.0-
rho*rho*rho*rho*rho)
   texto5 =str("\n") + "P5= " + str(tiempo_5/timeSim()) + " Teorico: " + str(1.0-
rho*rho*rho*rho*rho*rho)
   texto6 =str("\n") + "P6= " + str(tiempo_6/timeSim()) + " Teorico: " + str(1.0-
rho*rho*rho*rho*rho*rho*rho)
   texto7 =str("\n") + "P7= " + str(tiempo_7/timeSim()) + " Teorico: " + str(1.0-
rho*rho*rho*rho*rho*rho*rho*rho)

   archi.write(textoInicio)
   archi.write(texto0)
   archi.write(texto1)
   archi.write(texto2)
   archi.write(texto3)
   archi.write(texto4)
   archi.write(texto5)
   archi.write(texto6)
   archi.write(texto7)
   archi.close()

YES = 1
NO = 0
ERROR = 1

nuevaLlegada = 0
servicio = 1
finServicio = 2


labda = 1.0
mu = 2.0

tiempo_0 = 0.0
tiempo_1 = 0.0
tiempo_2 = 0.0
tiempo_3 = 0.0
tiempo_4 = 0.0
tiempo_5 = 0.0
tiempo_6 = 0.0
```

```
tiempo_7 = 0.0

evento = 0
marca = 0
seed_n = 0
cola=0
seguir=YES
servidor_ocupado=NO
servidos=0
tiempo_anterior= 0.0
rho = labda / mu


random.seed("seed")

init_simulation()

Schedule(event(nuevaLlegada,0.0,0))

while servidos < 1000000:
   eventoRecibido = Cause()

   if (servidor_ocupado==NO):
      tiempo_0 = tiempo_0 + (timeSim() -tiempo_anterior)

   if (servidor_ocupado==YES and cola==0):
      tiempo_1 = tiempo_1 + (timeSim()-tiempo_anterior)

   if (servidor_ocupado==YES and cola==1):
      tiempo_2 = tiempo_2 + (timeSim()-tiempo_anterior)

   if (servidor_ocupado==YES and cola==2):
      tiempo_3 = tiempo_3 + (timeSim()-tiempo_anterior)

   if (servidor_ocupado==YES and cola==3):
      tiempo_4 = tiempo_4 +(timeSim()-tiempo_anterior)

   if (servidor_ocupado==YES and cola==4):
      tiempo_5= tiempo_5 +(timeSim()-tiempo_anterior)

   if (servidor_ocupado==YES and cola==5):
      tiempo_6 = tiempo_6 + (timeSim()-tiempo_anterior)

   if (servidor_ocupado==YES and cola==6):
      tiempo_7= tiempo_7 + (timeSim()-tiempo_anterior)

   tiempo_anterior=timeSim()


   if eventoRecibido.type == nuevaLlegada:
      #print "Nueva llegada. Servidor: ", timeSim(), "Cola: ",servidor_ocupado,cola
      Schedule(event(servicio,0.0,0))
      Schedule(event(nuevaLlegada,random.expovariate(labda),0))

   elif eventoRecibido.type == servicio:
      if servidor_ocupado==NO:
         servidor_ocupado=YES
         Schedule(event(finServicio,random.expovariate(mu),0))
      else:
```

```
        cola = cola + 1


    elif eventoRecibido.type == finServicio:
        #print "Fin servicio. Servidor:", timeSim(), "Cola:",servidor_ocupado,cola
        servidos = servidos + 1
        if cola>0:
            cola = cola - 1
            Schedule(event(finServicio,random.expovariate(mu),0))
        else:
            servidor_ocupado=NO


    else:
        print "ERROR en switch. Evento no definido."


rho=labda/mu
print "P0=", tiempo_0/timeSim(), "Teorico:", 1.0-rho
print "P1=", tiempo_1/timeSim(), "Teorico:", (1.0-rho)*rho
print "P2=", tiempo_2/timeSim(), "Teorico:", (1.0-rho)*rho*rho
print "P3=", tiempo_3/timeSim(), "Teorico:", (1.0-rho)*rho*rho*rho
print "P4=", tiempo_4/timeSim(), "Teorico:", (1.0-rho)*rho*rho*rho*rho
print "P5=", tiempo_5/timeSim(), "Teorico:", (1.0-rho)*rho*rho*rho*rho*rho
print "P6=", tiempo_6/timeSim(), "Teorico:", (1.0-rho)*rho*rho*rho*rho*rho*rho
print "P7=", tiempo_7/timeSim(), "Teorico:", (1.0-rho)*rho*rho*rho*rho*rho*rho*rho

t2 = time.clock()

grabartxt()
```

ANEXO III: Código del simulador en C

El código que se muestra a continuación no ha sido desarrollado para este proyecto, es el original de SMPL.

smpl.c

```
#include "smpl.h"

#define nl 40960     /* element pool length              */
#define ns 4096     /* namespace length                 */
#define pl 58       /* printer page length   (lines used   */
#define sl 23       /* screen page length    by 'smpl')   */
#define FF 12       /* form feed                        */

static FILE
   *display, *opf;
 //*display=stdout,   /* screen display file              */
 //*opf=stdout;      /* current output destination        */

static int
 event,          /* current simulation event          */
 token,          /* last token dispatched             */
 blk,           /* next available block index         */
 avl,           /* available element list header      */
 evl,           /* event list header                  */
 fchn,          /* facility descriptor chain header    */
 avn,           /* next available namespace position   */
 tr,            /* event trace flag                   */
 mr,            /* monitor activation flag            */
 lft=sl;        /* lines left on current page/screen   */

static real
 clock,          /* current simulation time           */
 start,          /* simulation interval start time     */
 tl;            /* last trace message issue time      */

static int
 l1[nl],
 l2[nl],         /*     facility descriptor,        */
 l3[nl];         /*         queue, &               */
static real       /*        event list              */
 l4[nl],         /*       element pool             */
 l5[nl];

static char
 name[ns];       /* model and facility name space      */

/*---------------  INITIALIZE SIMULATION SUBSYSTEM  -------------------*/
void smpl(int m, char *s)
   {
    int i;
    static int rns=1;
    display=opf=stdout;
    blk=1; avl=-1; avn=0;      /* element pool & namespace headers */
    evl=fchn=0;         /* event list & descriptor chain headers  */
```

```
    clock=start=tl=0.0;   /* sim., interval start, last trace times */
    event=tr=0;           /* current event no. & trace flags  */
    for (i=0; i<nl; i++)  {l1[i]=l2[i]=l3[i]=0; l4[i]=l5[i]=0.0;}
    i=save_name(s,50);            /* model name -> namespace */
    rns=stream(rns); rns=++rns>15? 1:rns;  /* set random no. stream */
    mr=(m>0)? 1:0;                /* set monitor flag */
  /* if (mr) then {opf=display; init_mtr(1);} */
   }

/*----------------------  RESET MEASUREMENTS  ----------------------*/
void reset(void)
 {
  resetf(); start=clock;
 }

/*---------------------------  SAVE NAME  ---------------------------*/
static int save_name(char *s, int m)
  {
   int i,n;
   n=strlen(s); if (n>m) then n=m;
   if (avn+n>ns) then error(2,NULL); /* namespace exhausted */
   i=avn; avn+=n+1; strncpy(&name[i],s,n);
   if (n==m) then name[avn++]='\0';
   return(i);
  }

/*-------------------------  GET MODEL NAME  -------------------------*/
char *mname(void)
 {
  return(name);
 }

/*------------------------  GET FACILITY NAME  ------------------------*/
char *fname(int f)
  {
   return(&name[l3[f+1]]);
  }

/*---------------------------  GET BLOCK  ---------------------------*/
static int get_blk(int n)
  {
   int i;
   if (blk==0) then error(3,NULL); /* block request after schedule */
   i=blk; blk+=n;
   if (blk>=nl) then error(1,NULL);     /* element pool exhausted */
   return(i);
  }

/*--------------------------  GET ELEMENT  --------------------------*/
static int get_elm(void)
 {
  int i;
  if (avl<=0) then
   {
    if (avl==0) then error(1,NULL);    /* empty element list */
   /* if (mr && !tr) then init_mtr(2); */
     /* build the free element list from the block of elements */
     /* remaining after all facilities have been defined      */
     for (i=blk; i<(nl-1); i++) l1[i]=i+1;
```

```
      avl=blk; blk=0;
    }
  i=avl; avl=l1[i];
  return(i);
 }

/*------------------------- RETURN ELEMENT -------------------------*/
static void put_elm(int i)
  {
   l1[i]=avl; avl=i;
  }

/*------------------------- SCHEDULE EVENT -------------------------*/
void schedule(int ev,real te,int tkn)
  {
   int i;
   if (te<0.0) then error(4,NULL); /* negative event time */
   i=get_elm(); l2[i]=tkn; l3[i]=ev; l4[i]=0.0; l5[i]=clock+te;
   enlist(&evl,i);
   if (tr) then msg(1,tkn,"",ev,0);
  }

/*-------------------------- CAUSE EVENT --------------------------*/
void cause(int *ev,int *tkn)
  {
   int i;
   if (evl==0) then error(5,NULL);      /* empty event list  */
   i=evl; *tkn=token=l2[i]; *ev=event=l3[i]; clock=l5[i];
   evl=l1[i]; put_elm(i);  /* delink element & return to pool */
   if (tr) then msg(2,*tkn,"",event,0);
  /* if (mr && (tr!=3)) then mtr(tr,0); */
  }

/*-------------------------- RETURN TIME --------------------------*/
real time(void)
 {
  return(clock);
 }

/*-------------------------- CANCEL EVENT --------------------------*/
int cancel(int ev)
  {
   int pred,succ=evl,tkn;
   while((succ!=0) && (l3[succ]!=ev)) {pred=succ; succ=l1[pred];}
   if (succ==0) then return(-1);
   tkn=l2[succ]; if (tr) then msg(3,tkn,"",l3[succ],0);
   if (succ==evl)
     then evl=l1[succ];             /* unlink  event */
     else l1[pred]=l1[succ];         /* list entry &  */
   put_elm(succ);                  /* deallocate it */
   return(tkn);
  }

/*-------------------------- SUSPEND EVENT --------------------------*/
static int suspend(int tkn)
  {
   int pred,succ=evl;
   while((succ!=0) && (l2[succ]!=tkn)) {pred=succ; succ=l1[pred];}
   if (succ==0) then error(6,NULL);  /* no event scheduled for tkn */
```

```
    if (succ==evl)
     then evl=l1[succ];      /* unlink  event */
     else l1[pred]=l1[succ];  /* list entry    */
    if (tr) then msg(6,-1,"",l3[succ],0);
    return(succ);
   }

/*--------------  ENTER ELEMENT IN QUEUE OR EVENT LIST  --------------*/
static void enlist(int *head,int elm)
   { /* 'head' points to head of queue/event list */
    int pred,succ; real arg,v;
    arg=l5[elm]; succ=*head;
    while(1)
     { /* scan for position to insert entry:  event list is order- */
       /* ed in ascending 'arg' values, queues in descending order */
       if (succ==0)
        then break;  /* end of list */
        else
         {
          v=l5[succ];
          if (*head==evl)
           then
            { /* event list  */
             if (v>arg) then break;
            }
           else
            { /* queue: if entry is for a preempted token-    */
              /* (l4, the remaining event time, >0), insert   */
              /* entry at beginning of its priority class;    */
              /* otherwise, insert it at the end              */
              if ((v<arg) || ((v==arg) && (l4[elm]>0.0)))
                then break;
            }
         }
       pred=succ; succ=l1[pred];
     }
    l1[elm]=succ; if (succ!=*head) then l1[pred]=elm; else *head=elm;
   }

/*-------------------------  DEFINE FACILITY  -------------------------*/
int facility(char *s, int n)
   {
    int f,i;
    f=get_blk(n+2); l1[f]=n; l3[f+1]=save_name(s,(n>1 ? 14:17));
    if (fchn==0)
     then fchn=f;
     else {i=fchn; while(l2[i+1]) i=l2[i+1]; l2[i+1]=f;}
    l2[f+1]=0;
    if (tr) then msg(13,-1,fname(f),f,0);
    return(f);
   }

/*---------------  RESET FACILITY & QUEUE MEASUREMENTS  --------------*/
static void resetf(void)
 {
  int i=fchn,j;
   while(i)
    {
     l4[i]=l4[i+1]=l5[i+1]=0.0;
```

```
      for (j=i+2; j<=(i+l1[i]+1); j++) {l3[j]=0; l4[j]=0.0;}
      i=l2[i+1];  /* advance to next facility */
     }
 }

/*------------------------  REQUEST FACILITY  ------------------------*/
int request(int f,int tkn, int pri)
  {
   int i,r;
   if (l2[f]<l1[f])
    then
     { /* facility nonbusy - reserve 1st-found nonbusy server    */
      for (i=f+2; l1[i]!=0; i++);
      l1[i]=tkn; l2[i]=pri; l5[i]=clock; l2[f]++; r=0;
     }
    else
     { /* facility busy - enqueue token marked w/event, priority */
      enqueue(f,tkn,pri,event,0.0); r=1;
     }
   if (tr) then msg(7,tkn,fname(f),r,l3[f]);
   return(r);
  }

/*-------------------------  ENQUEUE TOKEN  -------------------------*/
static void enqueue(int f, int j,int pri,int ev,real te)
  {
   int i;
   l5[f+1]+=l3[f]*(clock-l5[f]); l3[f]++; l5[f]=clock;
   i=get_elm(); l2[i]=j; l3[i]=ev; l4[i]=te; l5[i]=(real)pri;
   enlist(&l1[f+1],i);
  }

/*------------------------  PREEMPT FACILITY  ------------------------*/
int preempt(int f,int tkn,int pri)
  {
   int ev,i,j,k,r; real te;
   if (l2[f]<l1[f])
    then
     { /* facility nonbusy - locate 1st-found nonbusy server     */
      for (k=f+2; l1[k]!=0; k++); r=0;
      if (tr) then msg(8,tkn,fname(f),0,0);
     }
    else
     { /* facility busy - find server with lowest-priority user  */
      k=f+2; j=l1[f]+f+1;  /* indices of server elements 1 & n  */
      for (i=f+2; i<=j; i++) if (l2[i]<l2[k]) then k=i;
      if (pri<=l2[k])
       then
        { /* requesting token's priority is not higher than   */
         /* that of any user: enqueue requestor & return r=1 */
         enqueue(f,tkn,pri,event,0.0); r=1;
         if (tr) then msg(7,tkn,fname(f),1,l3[f]);
        }
       else
        { /* preempt user of server k.  suspend event, save   */
         /* event number & remaining event time, & enqueue   */
         /* preempted token.  If remaining event time is 0   */
         /* (preemption occurred at the instant release was  */
         /* to occur, set 'te' > 0 for proper enqueueing     */
```

```
            /* (see 'enlist').  Update facility & server stati- */
            /* stics for the preempted token, and set r = 0 to  */
            /* reserve the facility for the preempting token.   */
            if (tr) then msg(8,tkn,fname(f),2,0);
            j=l1[k]; i=suspend(j); ev=l3[i]; te=l5[i]-clock;
            if (te==0.0) then te=1.0e-99; put_elm(i);
            enqueue(f,j,l2[k],ev,te);
            if (tr) then
              {msg(10,-1,"",j,l3[f]); msg(12,-1,fname(f),tkn,0);}
            l3[k]++; l4[k]+=clock-l5[k];
            l2[f]--; l4[f+1]++; r=0;
           }
       }
    if (r==0) then
     { /* reserve server k of facility */
      l1[k]=tkn; l2[k]=pri; l5[k]=clock; l2[f]++;
     }
    return(r);
  }

/*------------------------  RELEASE FACILITY  ------------------------*/
void release(int f, int tkn)
  {
    int i,j=0,k,m; real te;
    /* locate server (j) reserved by releasing token */
    k=f+1+l1[f];     /* index of last server element */
    for (i=f+2; i<=k; i++) if (l1[i]==tkn) then {j=i; break;}
    if (j==0) then error(7,NULL); /* no server reserved */
    l1[j]=0; l3[j]++; l4[j]+=clock-l5[j]; l2[f]--;
    if (tr) then msg(9,tkn,fname(f),0,0);
    if (l3[f]>0) then
     { /* queue not empty:  dequeue request ('k' =  */
       /* index of element) & update queue measures */
       k=l1[f+1]; l1[f+1]=l1[k]; te=l4[k];
       l5[f+1]+=l3[f]*(clock-l5[f]); l3[f]--; l4[f]++; l5[f]=clock;
       if (tr) then msg(11,-1,"",l2[k],l3[f]);
       if (te==0.0) then
         then
          { /* blocked request:  place request at head of event   */
            l5[k]=clock; l1[k]=evl; evl=k; m=4;
          }
         else
          { /* return after preemption:  reserve facility for de- */
            /* queued request & reschedule remaining event time   */
            l1[j]=l2[k]; l2[j]=(int)l5[k]; l5[j]=clock; l2[f]++;
            if (tr) then msg(12,-1,fname(f),l2[k],0);
            l5[k]=clock+te; enlist(&evl,k); m=5;
          }
       if (tr) then msg(m,-1,"",l3[k],0);
     }
  }

/*------------------------  GET FACILITY STATUS  ------------------------*/
int status(int f)
  {
    return(l1[f]==l2[f]? 1:0);
  }

/*--------------------  GET CURRENT QUEUE LENGTH  --------------------*/
```

```
int inq(int f)
  {
   return(l3[f]);
  }

/*--------------------  GET FACILITY UTILIZATION  --------------------*/
real U(int f)
  {
   int i; real b=0.0,t=clock-start;
   if (t>0.0) then
    {
     for (i=f+2; i<=f+l1[f]+1; i++) b+=l4[i];
     b/=t;
    }
   return(b);
  }

/*----------------------  GET MEAN BUSY PERIOD  ----------------------*/
real B(int f)
  {
   int i,n=0; real b=0.0;
   for (i=f+2; i<=f+l1[f]+1; i++) {b+=l4[i]; n+=l3[i];}
   return((n>0)? b/n:b);
  }

/*--------------------  GET AVERAGE QUEUE LENGTH  --------------------*/
real Lq(int f)
  {
   real t=clock-start;
   return((t>0.0)? (l5[f+1]/t):0.0);
  }

/*-----------------------  TURN TRACE ON/OFF  -----------------------*/
void trace(int n)
  {
   switch(n)
    {
     case 0: tr=0; break;
     case 1:
     case 2:
     case 3: tr=n; tl=-1.0; newpage(); break;
     case 4: end_line(); break;
    default: break;
    }
  }

/*--------------------  GENERATE TRACE MESSAGE  ----------------------*/
static void msg(int n,int i,char *s,int q1,int q2)
  {
   static char *m[14] = {"",  "SCHEDULE", "CAUSE", "CANCEL",
    "   RESCHEDULE","   RESUME", "   SUSPEND", "REQUEST", "PREEMPT",
    "RELEASE", "   QUEUE", "   DEQUEUE", "   RESERVE", "FACILITY" };
   if (clock>tl)    /* print time stamp (if time has advanced) */
    then {tl=clock; fprintf(opf,"  time %-12.3f  ",clock);}
    else fprintf(opf,"%21s",m[0]);
   if (i>=0)       /* print token number if specified */
    then fprintf(opf,"--  token %-4d  -- ",i);
    else fprintf(opf,"--            -- ");
   fprintf(opf,"%s %s",m[n],s);  /* print basic message */
```

```
    switch(n)
     { /* append qualifier */
       case 1:
       case 2:
       case 3:
       case 4:
       case 5:
       case 6:  fprintf(opf," EVENT %d",q1); break;
       case 7:
       case 8:  switch(q1)
                {
                  case 0: fprintf(opf,":  RESERVED"); break;
                  case 1: fprintf(opf,":  QUEUED  (inq = %d)",q2);
                        break;
                  case 2: fprintf(opf,":  INTERRUPT"); break;
                  default: break;
                }
               break;
       case 9:  break;
       case 10:
       case 11: fprintf(opf," token %d  (inq = %d)",q1,q2); break;
       case 12: fprintf(opf," for token %d",q1); break;
       case 13: fprintf(opf,":  f = %d",q1); break;
       default: break;
     }
    fprintf(opf,"\n"); end_line();
  }

/*-------------------------  TRACE LINE END  -------------------------*/
static void end_line(void)
 {
   if ((--lft)==0) then
    { /* end of page/screen.  for trace 1, advance page if print- */
      /* er output;  screen output is free-running.  for trace 2, */
      /* pause on full screen;  for trace 3, pause after line.    */
      switch(tr)
       {
         case 1: if (opf==display)
                   then lft=sl;
                   else endpage();
                 break;
         case 2: if (mr)
                   then {putchar('\n'); lft=sl; pause();}
                   else endpage();
                 break;
         case 3: lft=sl; break;
       }
    }
   if (tr==3) then pause();
 }

/*----------------------------  PAUSE  ----------------------------*/
void pause(void)
 { /* pause execution via 'mtr' call (if active) */
 /* if (mr) then mtr(tr,1); else */ getchar();
 }

/*------------------  DISPLAY ERROR MESSAGE & EXIT  ------------------*/
void error(int n,char *s)
```

```
  {
   FILE *dest;
   static char
      *m[8]= { "Simulation Error at Time ",
            "Empty Element Pool",
            "Empty Name Space",
            "Facility Defined After Queue/Schedule",
            "Negative Event Time",
            "Empty Event List",
            "Preempted Token Not in Event List",
            "Release of Idle/Unowned Facility" };
   dest=opf;
   while(1)
    { /* send messages to both printer and screen */
     fprintf(dest,"\n**** %s%.3f\n",m[0],clock);
     if (n) then fprintf(dest,"    %s\n",m[n]);
     if (s!=NULL) then fprintf(dest,"    %s\n",s);
     if (dest==display) then break; else dest=display;
    }
   if (opf!=display) then report();
  /* if (mr) then mtr(0,1); */
   exit(0);
  }

/*---------------------- GENERATE REPORT ----------------------*/
void report(void)
 {
  newpage();
  reportf();
  endpage();
 }

/*-------------------- GENERATE FACILITY REPORT --------------------*/
void reportf(void)
 {
  int f;
  if ((f=fchn)==0)
   then fprintf(opf,"\nno facilities defined:  report abandoned\n");
   else
    { /* f = 0 at end of facility chain */
     while(f) {f=rept_page(f); if (f>0) then endpage();}
    }
 }

/*---------------------- GENERATE REPORT PAGE ----------------------*/
static int rept_page(int fnxt)
  {
   int f,i,n; char fn[19];
   static char *s[7]= {
    "smpl SIMULATION REPORT", " MODEL: ", "TIME: ", "INTERVAL: ",
    "MEAN BUSY    MEAN QUEUE      OPERATION COUNTS",
    " FACILITY       UTIL.    ",
    " PERIOD      LENGTH    RELEASE   PREEMPT   QUEUE" };
   fprintf(opf,"\n%51s\n\n\n",s[0]);
   fprintf(opf,"%-s%-54s%-s%11.3f\n",s[1],mname(),s[2],clock);
   fprintf(opf,"%68s%11.3f\n\n",s[3],clock-start);
   fprintf(opf,"%75s\n",s[4]);
   fprintf(opf,"%s%s\n",s[5],s[6]);
   f=fnxt; lft-=8;
```

```
    while(f && lft--)
     {
      n=0; for (i=f+2; i<=f+l1[f]+1; i++) n+=l3[i];
      if (l1[f]==1)
       then sprintf(fn,"%s",fname(f));
       else sprintf(fn,"%s[%d]",fname(f),l1[f]);
      fprintf(opf," %-17s%6.4f %10.3f %13.3f %11d %9d %7d\n",
       fn,U(f),B(f),Lq(f),n,(int)l4[f+1],(int)l4[f]);
      f=l2[f+1];
     }
    return(f);
   }

/*---------------------------  COUNT LINES  --------------------------*/
int lns(int i)
   {
    lft-=i;  if (lft<=0) then endpage();
    return(lft);
   }

/*---------------------------  END PAGE  ----------------------------*/
void endpage(void)
 {
   int c;
   if (opf==display)
    then
     { /* screen output: push to top of screen & pause */
      while(lft>0) {putc('\n',opf); lft--;}
      printf("\n[ENTER] to continue:");  getchar();
    /* if (mr) then clr_scr(); else */ printf("\n\n");
     }
    else if (lft<pl) then putc(FF,opf);
   newpage();
 }

/*---------------------------  NEW PAGE  ----------------------------*/
void newpage(void)
 { /* set line count to top of page/screen after page change/screen  */
   /* clear by 'smpl', another SMPL module, or simulation program    */
   lft=(opf==display)? sl:pl;
 }

/*-----------------------  REDIRECT OUTPUT  -------------------------*/
FILE *sendto(FILE *dest)
   {
    if (dest!=NULL) then opf=dest;
    return(opf);
   }
```

smpl.h

```
#include <stdio.h>
#include <string.h>
#include <math.h>

typedef double real;
```

```
#define then

/* ----------  PROTOTYPES  ---------------------------------------- */
/* ----------  for functions in smpl.c  -------------------------- */

static int save_name(char *s, int m);
static int get_blk(int n);
static int get_elm(void);
static void put_elm(int i);
static int suspend(int tkn);
static void enlist(int *head, int elm);
static void resetf(void);
static void enqueue(int f, int j, int pri, int ev, real te);
static void msg(int n, int i, char *s, int q1, int q2);
static void end_line(void);
static int rept_page(int fnxt);
void smpl(int m, char *s);
void reset(void);
void schedule( int ev, real te, int tkn);
void cause(int *ev, int *tkn);
void release(int f, int tkn);
void trace(int n);
void pause(void);
void error(int n, char *s);
void report(void);
void reportf(void);
void endpage(void);
void newpage(void);
real time(void);
real U(int f);
real B(int f);
real Lq(int f);
char *fname(int f);
char *mname(void);
int facility(char *s, int n);
int request(int f, int tkn, int pri);
int cancel (int ev);
int preempt(int f, int tkn, int pri);
int status(int f);
int inq(int f);
int lns(int i);
FILE * sendto(FILE *dest);

/* ----------  PROTOTYPES  ---------------------------------------- */
/* ----------  for functions in bmeans.c  ------------------------ */

int obs(real y);
real Z(real p);
real T(real p,int ndf);
void civals(real *mean,real *hw, int *nb);
void init_bm(int m0,int mb);


/* ----------  PROTOTYPES  ---------------------------------------- */
/* ----------  for functions in rand.c  -------------------------- */

int stream(int n);
int seed(int lk, int n);
int random(int i, int n);
```

```
real ranf(void);
real uniform(real a, real b);
real expntl(real x);
real erlang(real x, real s);
real hyperx(real x, real s);
real normal(real x, real s);
```

ANEXO IV: Código del simulador en Visual Basic

Simulador.vb

```
Imports System.IO

Public Class Simulador
  Private Shared eventList As List(Of Evento)
  Private Shared simulationTime As Double

  Public Sub init_simulation()
    eventList = New List(Of Evento)
    simulationTime = 0.0
  End Sub

  Public Function Cause() As Evento
    If eventList.Count > 0 Then

      Dim eventoPrimero = eventList(0)
      eventList.Remove(eventoPrimero)
      set_time(eventoPrimero._time)
      Return eventoPrimero
    Else
      Return Nothing
    End If
  End Function


  Public Sub Schedule(ev As Evento)
    Dim insertPos As Integer = 0

    Dim tiempo = ev._time + simulationTime
    ev._time = tiempo

    For Each temp As Evento In eventList
      If temp._time < ev._time Then
        insertPos = insertPos + 1
      Else
        GoTo end_of_for
      End If
    Next

end_of_for:
    eventList.Insert(insertPos, ev)
  End Sub

  Public Sub prrintEventList()
    If eventList.Count > 0 Then
      For Each temp As Evento In eventList
        Console.WriteLine(String.Format("Event type: {0} in time {1}", temp._type, temp._time))
      Next
    Else
      Console.WriteLine("No events")
    End If
  End Sub

  Public Sub printSimulationTime()
    Console.WriteLine("Simulation time: " + simulationTime)
```

```
    End Sub

    Public Function deleteEventType(type As String) As Integer
        Dim contador = 0
        For Each temp As Evento In eventList
            If eventList(contador)._type.Equals(type) Then
                eventList.Remove(eventList(contador))
                GoTo end_of_for
            End If
            contador += 1
        Next
        Return -1
end_of_for:
        Return 0
    End Function

    Public Sub deleteAllEvents()
        If eventList.Count > 0 Then
            eventList.Clear()
        Else
            Console.WriteLine("No events")
        End If
    End Sub

    Private Shared Sub set_time(time As Double)
        simulationTime = time
    End Sub

    Public Function expo_variate(media As Double) As Double
        Return -media * Math.Log(Rnd())
    End Function

    Public Function timeSim() As Double
        Return simulationTime
    End Function

    Public Function Cancel(type As Integer) As Integer
        Dim contador As Integer = 0
        For Each temp In eventList
            If temp._type = type Then
                eventList.Remove(eventList(contador))

                contador = contador + 1

                Return contador - 1
            End If
        Next
        Return -1
    End Function
End Class
```

event.cs

```
Public Class Evento
    Public Property _type As String
```

```
    Public Property _time As Double
    Public Property _optional As String

    Public Sub New(ByVal type As String, ByVal time As Double, ByVal opcional As Integer)
        MyBase.New()
        _type = type
        _time = time
        _optional = opcional
    End Sub

End Class
```

ANEXO V: Ejemplo de cola MM1 en Visual Basic

MM1.vb

```
Imports Simulador
Imports System.Random
Module MM1

  Sub Main()
    Dim time2 As TimeSpan
    Dim time1 As New TimeSpan(DateTime.Now.Ticks)

    Dim Simulador As Simulador = New Simulador

    Dim YES = 1
    Dim NO = 0
    Dim ERR = 1

    Dim nuevaLlegada = 0
    Dim servicio = 1
    Dim finServicio = 2

    Dim lambda = 1.0
    Dim mu = 2.0

    Dim tiempo_0 As Double = 0.0
    Dim tiempo_1 As Double = 0.0
    Dim tiempo_2 As Double = 0.0
    Dim tiempo_3 As Double = 0.0
    Dim tiempo_4 As Double = 0.0
    Dim tiempo_5 As Double = 0.0
    Dim tiempo_6 As Double = 0.0
    Dim tiempo_7 As Double = 0.0

    Dim evento As Integer = 0
    Dim marca As Integer = 0

    Dim seed_n As Integer = 0
    Dim cola As Integer = 0
    Dim seguir As Integer = YES
    Dim servidor_ocupado As Integer = NO
    Dim servidos As Integer = 0
    Dim tiempo_anterior As Double = 0.0
    Dim rho As Double = 0.0

    Randomize()
    Rnd(Timer)

    Simulador.init_simulation()
    Simulador.Schedule(New Evento(nuevaLlegada, 0.0, 0))

    Dim lista As New List(Of Integer)

    While servidos < 1000000


      Dim eventoRecibido = Simulador.Cause()
```

```
        If servidor_ocupado = NO Then
            tiempo_0 = tiempo_0 + (Simulador.timeSim() - tiempo_anterior)
        End If

        If servidor_ocupado = YES And cola = 0 Then
            tiempo_1 = tiempo_1 + (Simulador.timeSim() - tiempo_anterior)
        End If

        If servidor_ocupado = YES And cola = 1 Then
            tiempo_2 = tiempo_2 + (Simulador.timeSim() - tiempo_anterior)
        End If

        If servidor_ocupado = YES And cola = 2 Then
            tiempo_3 = tiempo_3 + (Simulador.timeSim() - tiempo_anterior)
        End If

        If servidor_ocupado = YES And cola = 3 Then
            tiempo_4 = tiempo_4 + (Simulador.timeSim() - tiempo_anterior)
        End If

        If servidor_ocupado = YES And cola = 4 Then
            tiempo_5 = tiempo_5 + (Simulador.timeSim() - tiempo_anterior)
        End If

        If servidor_ocupado = YES And cola = 5 Then
            tiempo_6 = tiempo_6 + (Simulador.timeSim() - tiempo_anterior)
        End If

        If servidor_ocupado = YES And cola = 6 Then
            tiempo_7 = tiempo_7 + (Simulador.timeSim() - tiempo_anterior)
        End If


        tiempo_anterior = Simulador.timeSim()


        If eventoRecibido._type = nuevaLlegada Then
            'Console.WriteLine("Nueva llegada. Servidor: " + Simulador.timeSim().ToString + "Cola: " +
servidor_ocupado.ToString + cola.ToString)
            Simulador.Schedule(New Evento(servicio, 0.0, 0))
            Simulador.Schedule(New Evento(nuevaLlegada, Simulador.expo_variate(1 / lambda), 0))

        ElseIf eventoRecibido._type = servicio Then
            If servidor_ocupado = NO Then
                servidor_ocupado = YES
                Simulador.Schedule(New Evento(finServicio, Simulador.expo_variate(1 / mu), 0))
            Else
                cola = cola + 1
            End If

        ElseIf eventoRecibido._type = finServicio Then
            'Console.WriteLine("Fin servicio. Servidor:" + Simulador.timeSim().ToString + "Cola:" +
servidor_ocupado.ToString + cola.ToString)
            servidos = servidos + 1
            If cola > 0 Then
                cola = cola - 1
                Simulador.Schedule(New Evento(finServicio, Simulador.expo_variate(1 / mu), 0))
            Else
                servidor_ocupado = NO
```

```
            End If

        Else
            Print("ERROR en switch. Evento no definido.")
        End If

    End While

    rho = lambda / mu
    Console.WriteLine("P0= " + (tiempo_0 / Simulador.timeSim()).ToString + " Teorico: " + (1.0 -
rho).ToString)
    Console.WriteLine("P1= " + (tiempo_1 / Simulador.timeSim()).ToString + " Teorico: " + ((1.0 -
rho) * rho).ToString)
    Console.WriteLine("P2= " + (tiempo_2 / Simulador.timeSim()).ToString + " Teorico: " + ((1.0 -
rho) * rho * rho).ToString)
    Console.WriteLine("P3= " + (tiempo_3 / Simulador.timeSim()).ToString + " Teorico: " + ((1.0 -
rho) * rho * rho * rho).ToString)
    Console.WriteLine("P4= " + (tiempo_4 / Simulador.timeSim()).ToString + " Teorico: " + ((1.0 -
rho) * rho * rho * rho * rho).ToString)
    Console.WriteLine("P5= " + (tiempo_5 / Simulador.timeSim()).ToString + " Teorico: " + ((1.0 -
rho) * rho * rho * rho * rho * rho).ToString)
    Console.WriteLine("P6= " + (tiempo_6 / Simulador.timeSim()).ToString + " Teorico: " + ((1.0 -
rho) * rho * rho * rho * rho * rho * rho).ToString)
    Console.WriteLine("P7= " + (tiempo_7 / Simulador.timeSim()).ToString + " Teorico: " + ((1.0 -
rho) * rho * rho * rho * rho * rho * rho * rho).ToString)

    time2 = New TimeSpan(DateTime.Now.Ticks)

    Dim file As System.IO.StreamWriter
    Dim filePath As String
    filePath = System.IO.Path.Combine(My.Computer.FileSystem.SpecialDirectories.MyDocuments,
"MM1-VB.txt")

    file = My.Computer.FileSystem.OpenTextFileWriter(filePath, True)

    file.WriteLine("Time: " + (time2.Subtract(time1).TotalMilliseconds).ToString)
    file.WriteLine("P0= " + (tiempo_0 / Simulador.timeSim()).ToString + " Teorico: " + (1.0 -
rho).ToString)
    file.WriteLine("P1= " + (tiempo_1 / Simulador.timeSim()).ToString + " Teorico: " + ((1.0 - rho) *
rho).ToString)
    file.WriteLine("P2= " + (tiempo_2 / Simulador.timeSim()).ToString + " Teorico: " + ((1.0 - rho) *
rho * rho).ToString)
    file.WriteLine("P3= " + (tiempo_3 / Simulador.timeSim()).ToString + " Teorico: " + ((1.0 - rho) *
rho * rho * rho).ToString)
    file.WriteLine("P4= " + (tiempo_4 / Simulador.timeSim()).ToString + " Teorico: " + ((1.0 - rho) *
rho * rho * rho * rho).ToString)
    file.WriteLine("P5= " + (tiempo_5 / Simulador.timeSim()).ToString + " Teorico: " + ((1.0 - rho) *
rho * rho * rho * rho * rho).ToString)
    file.WriteLine("P6= " + (tiempo_6 / Simulador.timeSim()).ToString + " Teorico: " + ((1.0 - rho) *
rho * rho * rho * rho * rho * rho).ToString)
    file.WriteLine("P7= " + (tiempo_7 / Simulador.timeSim()).ToString + " Teorico: " + ((1.0 - rho) *
rho * rho * rho * rho * rho * rho * rho).ToString)

    file.Close()
    Console.ReadLine()

  End Sub

End Module
```

ANEXO VI: Código del simulador en C#

C#

Simulador.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Simulador
{
  class Simulador
  {
    private double simulationTime;
    private List<Evento> eventList;

    Random rand;


    public void init_Simulation()
    {
      rand = new Random();
      simulationTime = 0.0;
      eventList = new List<Evento>();
    }


    public Evento Cause()
    {
      if (eventList.Count > 0)
      {
        Evento eventoPrimero = eventList[0];
        eventList.Remove(eventoPrimero);
        set_time(eventoPrimero.Time);

        return eventoPrimero;

      }
      else { return null; }

    }

    public void Schedule(Evento ev)
    {
      int insertPos = 0;
      double tiempo = ev.Time + simulationTime;

      ev.Time = tiempo;

      if (eventList.Count > 0)
      {
        foreach (Evento temp in eventList)
        {
          if (temp.Time < ev.Time)
          {
```

```
                insertPos = insertPos + 1;
              }
            }
            eventList.Insert(insertPos, ev);
        }
        else
            eventList.Insert(0, ev);
    }

    public List<Evento> EventList()
    {
        return eventList;
    }

    public int deleteEventType(int type)
    {
        int contador = 0;

        foreach (Evento temp in eventList)
        {
            if (eventList[contador].Type == type)
            {
                eventList.Remove(eventList[contador]);
                goto end_of_for;
            }
            contador = contador + 1;

        }
        return -1;
    end_of_for:
        return 0;
    }

    public void deleteAllEvents()
    {
        if (eventList.Count > 0)
            eventList.Clear();
        else
        { Console.WriteLine("No events"); }

    }

    private void set_time(double time)
    {
        simulationTime = time;
    }

    public double expo_variate(double media)
    {
        var rnd = new Random(rand.Next());
        return -media * Math.Log(rnd.NextDouble());
    }

    public double TimeSim()
    {
        return simulationTime;
    }

}
```

```
}
```

Evento.cs

```
using System;

namespace Simulador
{
  class Evento
  {
    private int _type;
    private Double _time;
    private string _optional;

    public Evento(int Type, double Time, string Optional)
    {
      _type = Type;
      _time = Time;
      _optional = Optional;
    }

    public int Type
    {
      get { return _type; }
      set { _type = value; }
    }

    public Double Time
    {
      get { return _time; }
      set { _time = value; }
    }

    public string Optional
    {
      get { return _optional; }
      set { _optional = value; }
    }
  }
}
```

ANEXO VII: Ejemplo de cola MM1 en C#

MM1.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Simulador
{
  class Program
  {
    static void Main(string[] args)
    {
      TimeSpan time2;
      TimeSpan time1 = new TimeSpan(DateTime.Now.Ticks);
      Simulador Simulador = new Simulador();

      int YES = 1;
      int NO = 0;
      int ERR = 1;

      int nuevaLlegada = 0;
      int servicio = 1;
      int finServicio = 2;

      double lambda = 1.0;
      double mu = 2.0;

      double tiempo_0 = 0.0;
      double tiempo_1 = 0.0;
      double tiempo_2 = 0.0;
      double tiempo_3 = 0.0;
      double tiempo_4 = 0.0;
      double tiempo_5 = 0.0;
      double tiempo_6 = 0.0;
      double tiempo_7 = 0.0;

      int seed_n = 0;
      int cola = 0;
      int seguir = YES;
      int servidor_ocupado = NO;
      int servidos = 0;
      double tiempo_anterior = 0.0;
      double rho = 0.0;

      Evento evento = new Evento(1, 0.4, "option");


      Simulador.init_Simulation();
      Simulador.Schedule(new Evento(nuevaLlegada, 0.0, "0"));


      while (servidos < 10000000)
```

```
{
  Evento eventoRecibido = Simulador.Cause();

  if (servidor_ocupado == NO)
    tiempo_0 = tiempo_0 + (Simulador.TimeSim() - tiempo_anterior);

  if (servidor_ocupado == YES && cola == 0)
    tiempo_1 = tiempo_1 + (Simulador.TimeSim() - tiempo_anterior);

  if (servidor_ocupado == YES && cola == 1)
    tiempo_2 = tiempo_2 + (Simulador.TimeSim() - tiempo_anterior);

  if (servidor_ocupado == YES && cola == 2)
    tiempo_3 = tiempo_3 + (Simulador.TimeSim() - tiempo_anterior);

  if (servidor_ocupado == YES && cola == 3)
    tiempo_4 = tiempo_4 + (Simulador.TimeSim() - tiempo_anterior);

  if (servidor_ocupado == YES && cola == 4)
    tiempo_5 = tiempo_5 + (Simulador.TimeSim() - tiempo_anterior);

  if (servidor_ocupado == YES && cola == 5)
    tiempo_6 = tiempo_6 + (Simulador.TimeSim() - tiempo_anterior);

  if (servidor_ocupado == YES && cola == 6)
    tiempo_7 = tiempo_7 + (Simulador.TimeSim() - tiempo_anterior);

  tiempo_anterior = Simulador.TimeSim();


  if (eventoRecibido.Type == nuevaLlegada)
  {
    Simulador.Schedule(new Evento(servicio, 0.0, "0"));
    Simulador.Schedule(new Evento(nuevaLlegada, Simulador.expo_variate(1 / lambda),
"0"));
  }
  else if (eventoRecibido.Type == servicio)
  {
    if (servidor_ocupado == NO)
    {
      servidor_ocupado = YES;
      Simulador.Schedule(new Evento(finServicio, Simulador.expo_variate(1 / mu), "0"));
    }
    else
    {
      cola = cola + 1;
    }
  }
  else if (eventoRecibido.Type == finServicio)
  {
    servidos = servidos + 1;
    if (cola > 0)
    {
      cola = cola - 1;
      Simulador.Schedule(new Evento(finServicio, Simulador.expo_variate(1 / mu), "0"));
    }
    else
      servidor_ocupado = NO;
  }
```

```
            else
                Console.WriteLine("ERROR en switch. Evento no definido.");
        }

        rho = lambda / mu;
        Console.WriteLine(string.Format("P0= {0} Teórico: {1}", tiempo_0 / Simulador.TimeSim(),
(1.0 - rho)));
        Console.WriteLine(string.Format("P1= {0} Teórico: {1}", tiempo_1 / Simulador.TimeSim(),
(1.0 - rho) * rho));
        Console.WriteLine(string.Format("P2= {0} Teórico: {1}", tiempo_2 / Simulador.TimeSim(),
(1.0 - rho) * rho * rho));
        Console.WriteLine(string.Format("P3= {0} Teórico: {1}", tiempo_3 / Simulador.TimeSim(),
(1.0 - rho) * rho * rho * rho));
        Console.WriteLine(string.Format("P4= {0} Teórico: {1}", tiempo_4 / Simulador.TimeSim(),
(1.0 - rho) * rho * rho * rho * rho));
        Console.WriteLine(string.Format("P5= {0} Teórico: {1}", tiempo_5 / Simulador.TimeSim(),
(1.0 - rho) * rho * rho * rho * rho * rho));
        Console.WriteLine(string.Format("P6= {0} Teórico: {1}", tiempo_6 / Simulador.TimeSim(),
(1.0 - rho) * rho * rho * rho * rho * rho * rho));
        Console.WriteLine(string.Format("P7= {0} Teórico: {1}", tiempo_7 / Simulador.TimeSim(),
(1.0 - rho) * rho * rho * rho * rho * rho * rho * rho));

        time2 = new TimeSpan(DateTime.Now.Ticks);

        String filePath = "";
        filePath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

        System.IO.StreamWriter file = new
System.IO.StreamWriter(string.Format("{0}\\MM1C#.txt", filePath), true);

        file.WriteLine(String.Format("Time:  " + (time2.Subtract(time1).TotalMilliseconds)));

        file.WriteLine(string.Format("P0= {0} Teórico: {1}", tiempo_0 / Simulador.TimeSim(), (1.0 -
rho)));
        file.WriteLine(string.Format("P1= {0} Teórico: {1}", tiempo_1 / Simulador.TimeSim(), (1.0 -
rho) * rho));
        file.WriteLine(string.Format("P2= {0} Teórico: {1}", tiempo_2 / Simulador.TimeSim(), (1.0 -
rho) * rho * rho));
        file.WriteLine(string.Format("P3= {0} Teórico: {1}", tiempo_3 / Simulador.TimeSim(), (1.0 -
rho) * rho * rho * rho));
        file.WriteLine(string.Format("P4= {0} Teórico: {1}", tiempo_4 / Simulador.TimeSim(), (1.0 -
rho) * rho * rho * rho * rho));
        file.WriteLine(string.Format("P5= {0} Teórico: {1}", tiempo_5 / Simulador.TimeSim(), (1.0 -
rho) * rho * rho * rho * rho * rho));
        file.WriteLine(string.Format("P6= {0} Teórico: {1}", tiempo_6 / Simulador.TimeSim(), (1.0 -
rho) * rho * rho * rho * rho * rho * rho));
        file.WriteLine(string.Format("P7= {0} Teórico: {1}", tiempo_7 / Simulador.TimeSim(), (1.0 -
rho) * rho * rho * rho * rho * rho * rho * rho));
        file.Close();

        Console.ReadLine();
    }
  }
}
```

ANEXO VIII: Ejemplo de cola MM1 en C

MM1

```
#include "smpl.h"
#include <stdio.h>

#define YES (1)
#define NO  (0)
#define ERROR (1)

//Tipos de eventos
#define nueva_llegada      (0)
#define servicio           (1)
#define fin_servicio       (2)

unsigned int main(int argc, char* argv[])
{
   real lambda=1.0;
   real mu=2.0;

        unsigned int evento,marca;
   unsigned long int seed_n;
   unsigned int cola=0;
   unsigned int seguir=YES;
   unsigned int servidor_ocupado=NO;
   unsigned int servidos=0;

   real tiempo_0=0.0;
   real tiempo_1=0.0;
   real tiempo_2=0.0;
   real tiempo_3=0.0;
   real tiempo_4=0.0;
   real tiempo_5=0.0;
   real tiempo_6=0.0;
   real tiempo_7=0.0;

   real tiempo_anterior=0.0;
   real rho;

        smpl(0,"mm1");

        stream(1); //Emplea la semilla del stream 1
        seed_n=atoi(argv[1]);
        seed(seed_n,1); //Se introduce la semilla deseada en el stream1
   schedule(nueva_llegada,0.0,0);

 while(servidos<1000000)
        {
        cause(&evento, &marca);

        //pause();

        if (servidor_ocupado==NO) tiempo_0+=(time()-tiempo_anterior);
        if (servidor_ocupado==YES && cola==0) tiempo_1+=(time()-tiempo_anterior);
        if (servidor_ocupado==YES && cola==1) tiempo_2+=(time()-tiempo_anterior);
        if (servidor_ocupado==YES && cola==2) tiempo_3+=(time()-tiempo_anterior);
```

```
        if (servidor_ocupado==YES && cola==3) tiempo_4+=(time()-tiempo_anterior);
        if (servidor_ocupado==YES && cola==4) tiempo_5+=(time()-tiempo_anterior);
        if (servidor_ocupado==YES && cola==5) tiempo_6+=(time()-tiempo_anterior);
        if (servidor_ocupado==YES && cola==6) tiempo_7+=(time()-tiempo_anterior);
        tiempo_anterior=time();

        switch(evento){
    case nueva_llegada:
     //printf("%f Nueva llegada. Servidor: %d. Cola: %d\n", time(),servidor_ocupado,cola);
           schedule(servicio,0.0,0);
           schedule(nueva_llegada,expntl(1.0/lambda),0);

           break;

         case servicio:
     if (servidor_ocupado==NO){
        servidor_ocupado=YES;
                    schedule(fin_servicio,expntl(1.0/mu),0);
           }else{
                    cola++;
     }

           break;

    case fin_servicio:
     //printf("%f Fin servicio. Servidor: %d. Cola: %d\n", time(),servidor_ocupado,cola);
     servidos++;
     if (cola>0){
        cola--;
                    schedule(fin_servicio,expntl(1.0/mu),0);
     }else{
        servidor_ocupado=NO;
     }

           break;

         default:
           printf("ERROR en switch. Evento no definido.\n");
     printf("Pulse una tecla \n");
     while(getchar()==EOF){};
     exit(ERROR);
           break;
         }//switch
        }//while()

rho=lambda/mu;
printf("P0=%f. Teorico: %f\n",tiempo_0/time(),(1.0-rho));
printf("P1=%f. Teorico: %f\n",tiempo_1/time(), (1.0-rho)*rho);
printf("P2=%f. Teorico: %f\n",tiempo_2/time(), (1.0-rho)*rho*rho);
printf("P3=%f. Teorico: %f\n",tiempo_3/time(), (1.0-rho)*rho*rho*rho);
printf("P4=%f. Teorico: %f\n",tiempo_4/time(), (1.0-rho)*rho*rho*rho*rho);
printf("P5=%f. Teorico: %f\n",tiempo_5/time(), (1.0-rho)*rho*rho*rho*rho*rho);
printf("P6=%f. Teorico: %f\n",tiempo_6/time(), (1.0-rho)*rho*rho*rho*rho*rho*rho);
printf("P7=%f. Teorico: %f\n",tiempo_7/time(), (1.0-rho)*rho*rho*rho*rho*rho*rho*rho);

seed_n=seed(0,1); //Devuelve la semilla de stream1
return(seed_n);
}
```

ANEXO IX: Código en C# del modelo energético WiFi en un terminal móvil.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.VisualBasic;

namespace Simulador
{
  class Program
  {

    static Simulador Simulador = new Simulador();

    static bool debug = false;

    static int YES = 1;
    static int NO = 0;
    static int ERR = 1;
    static int DOWN = 0;
    static int UP = 1;
    static int user_to_CRX = 0;
    static int user_to_IDLE_OFF = 1;
    static int user_to_IDLE_ON = 2;
    static int new_packet = 3;
    static int end_flow = 4;

    static int CRX = 0;
    static int IDLE_ON = 1;
    static int IDLE_OFF = 2;

    static double T_PRO = 79.1;
    static double T_ON_IDLE = 7.6;
    static double T_OFF_IDLE = 300.6;
    static double T_i = 238.1;

    static double P_PRO = 124.4;
    static double P_Ti_BASE = 119.3;
    static double P_ON_IDLE = 77.2;
    static double P_IDLE_BASE = 11.4;
    static double alfapow_u = 0.28317;
    static double alfapow_d = 0.13701;
    static double betapow = 132.86;

    static double e_pro = 0.0;
    static double e_data = 0.0;
    static double e_idle = 0.0;
    static double e_Ti = 0.0;

    static double Rb_down = 40000.0;
    static double Rb_up = 4000.0;
```

```
static double Rb = 0.0;

static int follow = 1;
static double time_a = 0.0;
static int state = 0;
static int Ti_running = 0;
static double init_Ti = 0.0;

static int promoting = 0;
static int summed_p_IDLE = 0;
static double entry_to_IDLE_ON = 0.0;
static int just_entry_to_IDLE_ON = 0;
static int leaving = 0;
static double PS = 0.0;
static double interpacket = 0.0;
static int UP_DOWN = 0;
static double kbits_remaining = 0;
static double kbits_remainingInicio = 0;

static double tk_app2 = 0.0;

static void Main(string[] args)
{
   TimeSpan time2;
   TimeSpan time1 = new TimeSpan(DateTime.Now.Ticks);

   Simulador.init_Simulation();
   int tk = 0;
   double aux_real = 0.0, PSize= 0.0;


   PS = 1000.0;
   interpacket = 100.0;
   kbits_remaining = 10000000000;
   kbits_remainingInicio = kbits_remaining;
   UP_DOWN = DOWN;

   tk_app2 = 0;

   Simulador.Schedule(new Evento(new_packet, 0.0, "1"));

   time_a = Simulador.TimeSim();

   Simulador.Schedule(new Evento(user_to_CRX, 0.0, "2"));

   while (follow == YES)
   {
      Evento evento = Simulador.Cause();

      int opcion = evento.Type;

      if (kbits_remaining == 0)
      {
         follow = NO;
      }

      empty_apps();

      switch (opcion)
```

```
        {
          case 0: //user_to_CRX
            state = CRX;
            if (debug)
            {
              Console.WriteLine(String.Format("t = {0} CRX", Simulador.TimeSim()));
            }
            promoting = NO;
            break;

          case 1: //user_to_IDLE_OFF
            if (promoting == NO)
            {
              if (Ti_running == YES)
              {
                e_Ti = e_Ti + P_Ti_BASE * (Simulador.TimeSim() - init_Ti) * 0.000001;
              }

              Ti_running = NO;
              state = IDLE_OFF;

              if (debug)
                Console.WriteLine(String.Format("t = {0} IDLE_OFF", Simulador.TimeSim()));

              if (summed_p_IDLE == NO)
                e_idle = e_idle + P_ON_IDLE * (Simulador.TimeSim() - entry_to_IDLE_ON) *
0.000001;

              Simulador.Schedule(new Evento(user_to_IDLE_ON, T_OFF_IDLE, "2"));

              entry_to_IDLE_ON = Simulador.TimeSim() + T_OFF_IDLE;
              summed_p_IDLE = NO;
              just_entry_to_IDLE_ON = YES;

              e_idle = e_idle + P_IDLE_BASE * T_OFF_IDLE * 0.000001;

            }
            break;

          case 2: //user_to_IDLE_ON

            state = IDLE_ON;

            if (debug)
              Console.WriteLine(String.Format("t = {0} IDLE_ON", Simulador.TimeSim()));

              if (tk_app2 > 0 && promoting == NO)
              {
                Simulador.Schedule(new Evento(user_to_CRX, T_PRO, "2"));
                promoting = YES;

                e_pro = e_pro + P_PRO * T_PRO * 0.000001;
                if (just_entry_to_IDLE_ON == YES)
                  aux_real = 1.0;
                else
                  aux_real = Simulador.TimeSim() - entry_to_IDLE_ON;

                e_idle = e_idle + P_ON_IDLE * aux_real * 0.000001;
                summed_p_IDLE = YES;
```

```
                }
                else
                {
                    if (just_entry_to_IDLE_ON == YES)
                        Simulador.Schedule(new Evento(user_to_IDLE_OFF, T_ON_IDLE, "2"));
                }

                just_entry_to_IDLE_ON = NO;
                break;

            case 3: //new_packet

                PSize = Simulador.expo_variate(PS);

                if (PSize <= kbits_remaining)
                {
                    tk_app2 = tk_app2 + Convert.ToInt32(PSize);
                    kbits_remaining = kbits_remaining - Convert.ToInt32(PSize);
                }
                else
                {
                    tk_app2 = tk_app2 + kbits_remaining;
                    kbits_remaining = 0;
                }

                if (debug)
                    Console.WriteLine(String.Format("t = {0} New pkt: {1}  tk_app2 = {2}.  State = {3}",
                                      Simulador.TimeSim(), Convert.ToInt32(PSize), tk_app2, state));

                if (state == CRX)
                {
                    leaving = 0;
                    while (leaving != -1)
                        leaving = Simulador.deleteEventType(user_to_IDLE_OFF);


                    if (Ti_running == YES)
                        e_Ti = e_Ti + P_Ti_BASE * (Simulador.TimeSim() - init_Ti) * 0.000001;

                    Ti_running = NO;
                }

                if (state == IDLE_ON)
                {
                    just_entry_to_IDLE_ON = NO;
                    Simulador.Schedule(new Evento(user_to_IDLE_ON, 0.0, "2"));

                }

                Simulador.Schedule(new Evento(new_packet, Simulador.expo_variate(interpacket),
Convert.ToString(tk)));
                break;

            case 4: //end_flow

                break;
        }

        end_reschedule();
```

```
        }

        Console.WriteLine(string.Format("{0}, {1}, {2}, {3}, {4}", e_pro, e_Ti, e_data, e_idle, (e_pro +
e_Ti + e_data + e_idle)));

        time2 = new TimeSpan(DateTime.Now.Ticks);

        String filePath = "";
        filePath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

        System.IO.StreamWriter file = new System.IO.StreamWriter(string.Format("{0}\\testC#.txt",
filePath), true);

        file.WriteLine(String.Format("{0}, {1}, {2}, {3}, {4} TIEMPO: {5} ms, kbits = {6}", e_pro, e_Ti,
e_data, e_idle, (e_pro + e_Ti + e_data + e_idle), (time2.Subtract(time1).TotalMilliseconds),
kbits_remainingInicio));
        file.Close();

        Console.ReadLine();


    }

    static void empty_apps()
    {
      double elapsed_time, power, alfa = 0;

      elapsed_time = Simulador.TimeSim() - time_a;

      time_a = Simulador.TimeSim();

      if (tk_app2 > 0 && state == CRX)
      {
        if (debug)
          Console.WriteLine(String.Format("t = {0} Empty: tk_app2 = {1}", Simulador.TimeSim(),
tk_app2));

          if (UP_DOWN == DOWN)
          {
            Rb = Rb_down;
            alfa = alfapow_d;
          }
          else
          {
            Rb = Rb_up;
            alfa = alfapow_u;
          }

        tk_app2 = tk_app2 - Convert.ToInt32(Math.Round(elapsed_time * Rb * 0.001));

       if(debug)
        Console.WriteLine(String.Format("{0} = {1} con Rb = {2}",
Convert.ToInt32(Math.Round(elapsed_time * Rb * 0.001)), tk_app2, Rb));

        if (tk_app2 == -1)
          tk_app2 = 0;

        if (tk_app2 == 0)
```

```
            {
                Simulador.Schedule(new Evento(user_to_IDLE_OFF, T_i, "2"));
                summed_p_IDLE = YES;

                var now = Simulador.TimeSim();

                if (debug)
                    Console.WriteLine(String.Format("Start Ti at t = {0}", now));

                init_Ti = now;

                Ti_running = YES;
            }

            power = alfa * Rb + betapow;
            e_data = e_data + power * elapsed_time * 0.000001;

        }
    }
    static void end_reschedule()
    {
        double aux_real = 0;
        int leaving = 0;

        while (leaving != -1)
        {
            leaving = Simulador.deleteEventType(end_flow);
        }

        if (state == CRX)
        {
            if (UP_DOWN == DOWN)
                Rb = Rb_down;
            else
                Rb = Rb_up;

            aux_real = Convert.ToDouble(tk_app2 / Rb * 1000.0);

            if (aux_real > 0.0)
                Simulador.Schedule(new Evento(end_flow, aux_real, "1"));
        }
    }
  }
}
```

ANEXO X: Código en Visual Basic del modelo energético WiFi en un terminal móvil.

```
Imports System.Random
Imports System.Math

Module Module1

    Dim debug As Boolean = False

    Dim Simulador As Simulador = New Simulador

    Dim YES As Integer = 1
    Dim NO As Integer = 0
    Dim ERR As Integer = 1
    Dim DOWN As Integer = 0
    Dim UP As Integer = 1
    Dim user_to_CRX As Integer = 0
    Dim user_to_IDLE_OFF As Integer = 1
    Dim user_to_IDLE_ON As Integer = 2
    Dim new_packet As Integer = 3
    Dim end_flow As Integer = 4

    Dim CRX As Integer = 0
    Dim IDLE_ON As Integer = 1
    Dim IDLE_OFF As Integer = 2

    Dim T_PRO As Double = 79.1
    Dim T_ON_IDLE As Double = 7.6
    Dim T_OFF_IDLE As Double = 300.6
    Dim T_i As Double = 238.1

    Dim P_PRO As Double = 124.4
    Dim P_Ti_BASE As Double = 119.3
    Dim P_ON_IDLE As Double = 77.2
    Dim P_IDLE_BASE As Double = 11.4
    Dim alfapow_u As Double = 0.28317
    Dim alfapow_d As Double = 0.13701
    Dim betapow As Double = 132.86

    Dim e_pro As Double = 0.0
    Dim e_data As Double = 0.0
    Dim e_idle As Double = 0.0
    Dim e_Ti As Double = 0.0

    Dim Rb_down As Double = 40000.0
    Dim Rb_up As Double = 4000.0
    Dim Rb As Double = 0.0

    Dim follow As Integer = YES
    Dim time_a As Double = 0.0
    Dim state As Integer = 0
    Dim Ti_running As Integer = NO
    Dim init_Ti As Double = 0.0

    Dim promoting As Integer = NO
```

```
Dim summed_p_IDLE As Integer = NO

Dim entry_to_IDLE_ON As Double = 0.0
Dim just_entry_to_IDLE_ON As Integer = 0
Dim leaving As Integer = 0

Dim PS As Double = 0.0
Dim interpacket As Double = 0.0
Dim UP_DOWN As Integer = 0

Dim kbits_remaining As Double = 0

Dim tk_app2 As Double = 0.0

Sub Main()
   'Dim time1 = Date.Now.Ticks
   Dim time2 As TimeSpan
   Dim time1 As New TimeSpan(DateTime.Now.Ticks)

   Simulador.init_simulation()
   Randomize()
   Rnd(Timer)

   Dim tk As Integer
   Dim aux_real, PSize As Double

   PS = 1000.0
   interpacket = 100.0
   kbits_remaining = 1000000000.0
   Dim kbits_remainingInicio = kbits_remaining

   UP_DOWN = DOWN

   tk_app2 = 0

   Simulador.Schedule(New Evento(new_packet.ToString, 0.0, 1))

   time_a = Simulador.timeSim()

   Simulador.Schedule(New Evento(user_to_CRX.ToString, 0.0, 2))

   While follow = YES

      'Simulador.prrintEventList()

      Dim evento As Evento = Simulador.Cause()

      Dim opcion As Integer = 10
      If evento Is Nothing Then
         Console.WriteLine("LISTA DE EVENTOS VACÍA")
         Console.ReadLine()
      Else
         opcion = CType(evento._type, Integer)
      End If

      'Console.WriteLine("Kbits = " & kbits_remaining)
      If kbits_remaining = 0 Then
         follow = NO
      End If
```

```
empty_apps()

Select Case opcion

  Case user_to_CRX
    state = CRX
    If debug Then
      Console.WriteLine(String.Format("t = {0} CRX", Simulador.timeSim()))
    End If
    promoting = NO


  Case user_to_IDLE_OFF
    If promoting = NO Then
      If Ti_running = YES Then
        e_Ti = e_Ti + P_Ti_BASE * (Simulador.timeSim() - init_Ti) * 0.000001
      End If

      Ti_running = NO
      state = IDLE_OFF

      If debug Then
        Console.WriteLine(String.Format("t = {0} IDLE_OFF", Simulador.timeSim()))
      End If

      If summed_p_IDLE = NO Then
        e_idle = e_idle + P_ON_IDLE * (Simulador.timeSim() - entry_to_IDLE_ON) * 0.000001
      End If

      Simulador.Schedule(New Evento(user_to_IDLE_ON.ToString, T_OFF_IDLE, 2))

      entry_to_IDLE_ON = Simulador.timeSim() + T_OFF_IDLE
      summed_p_IDLE = NO
      just_entry_to_IDLE_ON = YES

      e_idle = e_idle + P_IDLE_BASE * T_OFF_IDLE * 0.000001
    End If

  Case user_to_IDLE_ON

    state = IDLE_ON

    If debug Then
      Console.WriteLine(String.Format("t = {0} IDLE_ON", Simulador.timeSim()))
    End If

    If tk_app2 > 0 And promoting = NO Then
      Simulador.Schedule(New Evento(user_to_CRX.ToString, T_PRO, 2))
      promoting = YES

      e_pro = e_pro + P_PRO * T_PRO * 0.000001

      If just_entry_to_IDLE_ON = YES Then
        aux_real = 1.0
      Else
        aux_real = Simulador.timeSim() - entry_to_IDLE_ON
      End If
      e_idle = e_idle + P_ON_IDLE * aux_real * 0.000001
```

```
            summed_p_IDLE = YES
          Else
            If just_entry_to_IDLE_ON = YES Then
              Simulador.Schedule(New Evento(user_to_IDLE_OFF.ToString, T_ON_IDLE, 2))
            End If
          End If

          just_entry_to_IDLE_ON = NO

        Case new_packet
          'Console.WriteLine("New packet, kbit = " & kbits_remaining)
          ' Simulador.prrintEventList()
          PSize = Simulador.expo_variate(PS)

          If PSize <= kbits_remaining Then
            tk_app2 = tk_app2 + CType(PSize, Integer)
            kbits_remaining = kbits_remaining - CType(PSize, Integer)
          Else
            tk_app2 = tk_app2 + kbits_remaining
            kbits_remaining = 0
          End If

          If debug Then
            Console.WriteLine(String.Format("t = {0} New pkt: {1}  tk_app2 = {2}.  State = {3}",
                              Simulador.timeSim(), CType(PSize, Integer), tk_app2, state))
          End If

          If state = CRX Then
            leaving = 0
            While leaving <> -1
              leaving = Simulador.deleteEventType(user_to_IDLE_OFF)
            End While

            If Ti_running = YES Then
              e_Ti = e_Ti + P_Ti_BASE * (Simulador.timeSim() - init_Ti) * 0.000001
            End If

            Ti_running = NO
          End If

          If state = IDLE_ON Then
            just_entry_to_IDLE_ON = NO
            Simulador.Schedule(New Evento(user_to_IDLE_ON.ToString, 0.0, 2))

          End If

          Simulador.Schedule(New Evento(new_packet.ToString,
Simulador.expo_variate(interpacket), tk))

        Case end_flow

      End Select

      end_reschedule()
    End While


    Console.WriteLine(String.Format("{0}, {1}, {2}, {3}, {4}", e_pro, e_Ti, e_data, e_idle, (e_pro + e_Ti
+ e_data + e_idle)))
```

```
        'Dim time2 = Date.Now.Ticks

        'Dim restaTiempo = time2 - time1
        time2 = New TimeSpan(DateTime.Now.Ticks)

        Dim file As System.IO.StreamWriter
        Dim filePath As String
        filePath = System.IO.Path.Combine(My.Computer.FileSystem.SpecialDirectories.MyDocuments,
"test.txt")

        file = My.Computer.FileSystem.OpenTextFileWriter(filePath, True)
        file.WriteLine(String.Format("{0}, {1}, {2}, {3}, {4} TIEMPO: {5} ms, kbits = {6}", e_pro, e_Ti,
e_data, e_idle, (e_pro + e_Ti + e_data + e_idle), (time2.Subtract(time1).TotalMilliseconds),
kbits_remainingInicio))
        file.Close()
        Console.ReadLine()

    End Sub


    Public Sub empty_apps()

        Dim elapsed_time, power, alfa As Double

        elapsed_time = Simulador.timeSim() - time_a

        time_a = Simulador.timeSim()

        If tk_app2 > 0 And state = CRX Then
            If debug Then
                Console.WriteLine(String.Format("t = {0} Empty: tk_app2 = {1}", Simulador.timeSim(),
tk_app2))
            End If

            If UP_DOWN = DOWN Then
                Rb = Rb_down
                alfa = alfapow_d
            Else
                Rb = Rb_up
                alfa = alfapow_u
            End If

            tk_app2 = tk_app2 - CType(Round(elapsed_time * Rb * 0.001), Integer)

            If debug Then
                Console.WriteLine(String.Format("{0} = {1} con Rb = {2}", CType(Round(elapsed_time * Rb
* 0.001), Integer), tk_app2, Rb))
            End If

            If tk_app2 = -1 Then
                tk_app2 = 0
            End If

            If tk_app2 = 0 Then
                Simulador.Schedule(New Evento(user_to_IDLE_OFF.ToString, T_i, 2))
                summed_p_IDLE = YES

                Dim now = Simulador.timeSim()
```

```
            If debug Then
              Console.WriteLine(String.Format("Start Ti at t = {0}", now))
            End If
            init_Ti = now

            Ti_running = YES
          End If
          power = alfa * Rb + betapow
          e_data = e_data + power * elapsed_time * 0.000001 'J
        End If
      End Sub

      Public Sub end_reschedule()
        Dim aux_real As Double
        Dim leaving As Integer = 0

        While leaving <> -1
          leaving = Simulador.deleteEventType(end_flow)
        End While

          If state = CRX Then
            If UP_DOWN = DOWN Then
              Rb = Rb_down
            Else
              Rb = Rb_up
            End If
            aux_real = CType(tk_app2 / Rb * 1000.0, Double)

            If aux_real > 0.0 Then
              Simulador.Schedule(New Evento(end_flow.ToString, aux_real, 1))
            End If
          End If
      End Sub

    End Module
```