UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Desarrollo de un simulador por eventos discretos para el estudio de la gestión de tráfico en redes móviles que implementan network slicing

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR/A: Guijarro Monerris, Alejandro

Tutor/a: Pla Boscà, Vicent

CURSO ACADÉMICO: 2021/2022

Resumen

Una de las características de las redes 5G es la posibilidad de crear diversas redes lógicas sobre una misma infraestructura física. En la jerga de 5G, cada una de las redes lógicas que funcionan sobre la misma infraestructura de red se dice que es una slice de la red. Desde el punto de vista del explotador de la infraestructura de red, el que la infraestructura se comparta por múltiples slices puede permitir un uso más eficiente de los recursos, pero por otra parte, también debe conseguir el aislamiento necesario entre slices y el cumplimiento de los acuerdos de servicio suscritos con los usuarios de las slices. Además, el número de slices y sus características variarán dinámicamente, lo cual complica los mecanismos de provisión de recursos y gestión de tráfico. Esto es especialmente así en la red de acceso radio, pues los recursos físicos, principalmente el espectro radioeléctrico, están intrínsecamente limitados y ligados al lugar en el que se necesitan, mientras que los usuarios finales son móviles. El objetivo de este TFG es desarrollar un simulador por eventos discretos que permita evaluar distintas técnicas de provisión de recursos y gestión de tráfico en una red de acceso celular con usuarios móviles. Como parte del TFG se estudiará también un método aproximado para el dimensionado de la red y el control de admisión de nuevas slices.

Resum

Una de les característiques de les xarxes 5G és la possibilitat de crear diverses xarxes lògiques sobre una mateixa infraestructura física. En l'argot de 5G, cadascuna de les xarxes lògiques que funcionen sobre la mateixa infraestructura de xarxa es diu que és una slice de la xarxa. Des del punt de vista de l'explotador de la infraestructura de xarxa, el que la infraestructura es compartisca per múltiples slices pot permetre un ús més eficient dels recursos, però d'altra banda, també ha d'aconseguir l'aïllament necessari entre slices i el compliment dels acords de servei subscrits amb els usuaris de les slices. A més, el número de slices i les seues característiques variaran dinàmicament, la qual cosa complica els mecanismes de provisió de recursos i gestió de trànsit. Això és especialment així en la xarxa d'accés ràdio, perquè els recursos físics, principalment l'espectre radioelèctric, estan intrínsecament limitats i lligats al lloc en el qual es necessiten, mentre que els usuaris finals són mòbils. L'objectiu d'aquest TFG és desenvolupar un simulador per esdeveniments discrets que permeta avaluar diferents tècniques de provisió de recursos i gestió de trànsit en una xarxa d'accés cel·lular amb usuaris mòbils. Com a part del TFG s'estudiarà també un mètode aproximat per al dimensionament de la xarxa i el control d'admissió de noves slices.

Abstract

One of the characteristics of 5G networks is the possibility of creating several logical networks on the same physical infrastructure. In 5G jargon, each of the logical networks running over the same network infrastructure is said to be a slice of the network. From the network infrastructure operator's point of view, sharing the infrastructure across multiple slices can enable more efficient use of resources, but on the other hand, it has to achieve the necessary isolation between slices and compliance with the service agreements signed with the slices' users. In addition, the number of slices and their characteristics will vary dynamically, which complicates resource provisioning and traffic management mechanisms. This is especially so in the radio access network, as physical resources, mainly radio spectrum, are inherently limited and tied to the location where they are needed, while end users are mobile. The objective of this Bachelor's thesis is to develop a discrete event simulator to evaluate different resource provisioning and traffic management techniques in a cellular access network with mobile users. As part of the Bachelor's thesis, an approximate method for network resource provisioning and admission control of new slices will also be studied.

A mis padres.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El *network slicing* es la técnica en redes móviles que permite crear y configurar redes lógicas dentro de la infraestructura de red, asignando una parte de los recursos de esta a las redes lógicas. Las redes lógicas se conocen como *slices* (nomenclatura que utilizaremos a lo largo del trabajo). La principal ventaja del network slicing es la posibilidad de crear redes lógicas para cada tipo de usuarios con requisitos de servicio distintos (ancho de banda, latencia, etc) sobre una misma infraestructura. De este modo se evita crear redes físicas independientes que sustenten cada una de estas aplicaciones y se maximiza la eficiencia del uso de los recursos de la red física.

El network slicing es la base hoy en día para el estándar de comunicaciones móviles de quinta generación, 5G. El 5G pretende usarse en numerosas aplicaciones con requisitos de servicio distintos, de ancho de banda o de latencia. Esto se puede ejemplificar a través de, por una parte, la telemedicina, que requiere de una latencia muy baja, o por otro lado, el Internet of Everything, el cual no requiere de conexiones con altas tasas de subida o bajada, debido a la baja tasa de información se transmite, pero sí de muchas conexiones simultáneas. Estos servicios se benefician del "slicing", ya que este permite una mayor flexibilidad y dinamismo en la definición y provisión de recursos para cada servicio sobre una misma infraestructura de red.

El uso concurrente de los recursos de la red física por varias slices puede llevar a que situaciones de sobreuso por parte de alguna de ellas degrade el Grado de Servicio (GoS) suscrito por las otras slices. Para solucionar y prevenir este problema, es necesario por parte del operador de la infraestructura conseguir el aislamiento necesario entre slices y el cumplimiento por parte de estas de los acuerdos de servicio suscritos. Además, el número de slices y sus características variarán dinámicamente, lo cual complica los mecanismos de gestión de recursos. Esto es especialmente así en la red de acceso radio, pues los recursos físicos, principalmente el espectro radioeléctrico, están intrínsecamente limitados y ligados al lugar en el que se necesitan, mientras que los usuarios finales son móviles.

A la vista de las dificultades que presenta el network slicing, es necesario realizar estudios de la red que permitan evaluar a priori determinados mecanismos de gestión de recursos que puedan ser implementados por el operador de la infraestructura.

El objetivo de este Trabajo de Fin de Grado es el desarrollo de un simulador por eventos discretos para evaluar distintas técnicas de provisión de recursos y gestión de tráfico en una red de acceso celular que implementa network slicing, permitiendo el estudio de la problemática anteriormente descrita. Además, el simulador permite la prueba y validación de distintos modelos analíticos para

la gestión de recursos. En una primera parte, el trabajo se enfocará en la programación del simulador. A continuación, se evaluará un método aproximado para el dimensionado de la red y el control de admisión de nuevas slices en la misma, como ejemplo de uso del simulador desarrollado.

El trabajo se ha desarrollado durante un año académico completo, desde septiembre 2021 hasta agosto 2022, siendo este compaginado con una intercambio académico Erasmus en "IMT Atlantique" en Francia.

La presente memoria se estructura en 5 capítulos.

En el capítulo 2 se han agrupado todos los conceptos teóricos, desde modelos conceptuales a los resultados analíticos, que más tarde serán utilizados en el desarrollo del trabajo. Este capítulo pretende añadir contexto a la fórmulas utilizadas en el trabajo, pero no pretende ser una referencia de estudio para los distintos conceptos.

En el capítulo 3, se presentará de manera detallada el proceso de desarrollo y verificación del simulador en Python.

Se ha realizado una simulación de un escenario de red móvil y una aproximación para el dimensionado de la red, que ha sido estudiado y validado usando el simulador desarrollado. Este estudio se presenta en el capitulo 4.

Finalmente, tras el desarrollo y posterior uso del simulador, en el capítulo 5 se concluye el trabajo, mencionando las limitaciones del estudio que pueden servir de base para propuestas de trabajo futuro.

Capítulo 2

Marco Teórico

En esta capítulo se pretende dar de forma breve y concisa un repaso de los aspectos generales y de los resultados analíticos más importantes de las distintas herramientas teóricas utilizadas para el desarrollo de este trabajo. Los resultados que aquí se expresan serán retomados a lo largo del desarrollo del trabajo.

1. Procesos de Markov

Una proceso de Markov es un proceso estocástico donde se cumple la propiedad de memoria-nula: la evolución del proceso de Markov solo depende del estado actual y no de los anteriores. Este espacio de estados puede ser infinito. Además, si las transiciones solo se permiten entre estados sucesivos, tendremos un proceso de *nacimiento y muerte*.

Una cadena de Markov se puede resumir en un diagrama de estados y de transiciones, la figura 2.1 representa un proceso de nacimiento y muerte. λ_i corresponde al número medio de veces por unidad de tiempo que se transita a un estado superior, de i a $i+1$, mientras que μ_i corresponde a la tasa de transiciones de un estado i a $i-1$.

Para determinar la solución analítica de un proceso de nacimiento y muerte en régimen permanente, siendo π_n la probabilidad de que el sistema se encuentre en el estado n, resolveremos el siguiente

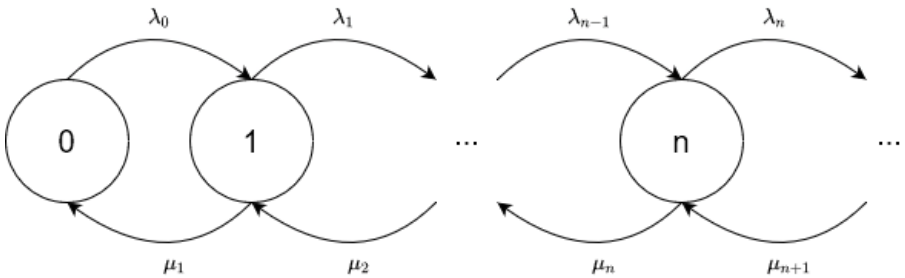


Figura 2.1: Diagrama de estados y de transiciones de un proceso de nacimiento y muerte

sistema de ecuaciones [1, Cap. 11]:

$$\pi_n = \frac{\lambda_{n-1}}{\mu_n}\pi_{n-1}, \quad n = 1, ..., N \tag{2.1}$$

$$\pi_0 = \left(1 + \sum_{n=1}^{N}\prod_{j=1}^{n}\frac{\lambda_{j-1}}{\mu_j}\right)^{-1}$$

2. Teoría de colas

En este apartado se pretende presentar las bases de la Teoría de Colas y los resultados que se obtienen, que se usarán en las siguientes secciones y capítulos.

La Teoría de Colas tiene como objetivo formalizar el estudio de los procesos de espera dentro de un servicio determinado, en el que la naturaleza estocástica de las llegadas al servicio y los tiempos de servicio producen situaciones de espera previamente a ser servido.

2.1. Las colas

En la Teoría de Colas se definen los *usuarios* (a veces referidos por otros nombres, según el contexto), que utilizan los recursos ofrecidos por los *servidores*. Los usuarios podrán ser directamente atendidos, o podrán esperar en una *cola* hasta que tengan la oportunidad de ser servidos. Comúnmente, se refiere al conjunto cola-servidor(es) simplemente como cola.

En una cola las principales característica que la definen son: el número de servidores disponibles, la capacidad de usuarios en espera, la tasa de llegadas, que denotaremos λ, y la tasa de servicio, μ.

Capacidad

La capacidad de una cola corresponde a cuántos usuarios caben dentro del sistema. Por una parte, podrá haber usuarios siendo servidos en los servidores, y por otra parte, podrá haber usuarios que estén a la espera de ser servidos en la cola.
Por ello es pertinente definir el número de servidores del cual dispone el sistema, como el número de plazas en espera que el sistema puede alojar en la cola. Tanto el número de servidores como la cola pueden tener una capacidad infinita.

Tasa de llegadas

La tasa media de llegadas corresponde al número medio de llegadas a la cola que se producen por unidad de tiempo. Equivalentemente, $1/\lambda$ corresponde al tiempo medio entre llegadas.

Tasa de servicio

Similarmente a la tasa de llegadas, la tasa media de servicio corresponde al número medio de usuarios servidos por unidad de tiempo. $1/\mu$ es el tiempo medio de servicio. El tiempo medio de servicio también puede interpretarse desde el punto de vista del usuario, como el tiempo medio que este demora siendo servido.

Las variables aleatorias que describen el tiempo entre llegadas y el tiempo de servicio, se asumen que siguen una distribución exponencial de parámetro λ y μ respectivamente. Esto tiene como consecuencia que el proceso de llegadas seguirá una distribución de Poisson [1, Cap. 11], lo que se asume en los teoremas que presentaremos en la siguiente sección.

2.2. Notación de Kendall

Sea la cola definida como $A/B/C/X/Y/Z$:

A: corresponde a la distribución del tiempo entre llegadas

B: corresponde a la distribución del tiempo de servicio

C: corresponde al número de servidores

X: corresponde a la capacidad del sistema

Y: corresponde al tamaño de la población de clientes

Z: corresponde a la disciplina de servicio

En el desarrollo del trabajo, asumiremos por una parte en un tamaño de población infinito $Y = \infty$ y por otra parte en la disciplina de servicio FCFS. Esta disciplina, la más sencilla de todas, establece que el primero en llegar a la cola será el primero en salir de esta para ser servido, no hay distinción de usuarios dentro de la cola. Es una disciplina determinista.

2.3. Resultados analíticos de colas

En este apartado se expondrán los resultados obtenidos del análisis de ciertos tipos de colas que nos serán útiles más adelante en el desarrollo de nuestro estudio. Estas colas son la $M/M/\infty$ y la $M/M/c/c$.

Las colas se modelan como un proceso de nacimiento y muerte, donde cada estado representa el número de clientes en el sistema (cola + servidor).

2.3.1. La cola $M/M/\infty$

La cola $M/M/\infty$ se caracteriza por tener infinitos servidores, de manera que cada usuario que llega a la cola será servido.
Sea p_n la probabilidad de que haya n usuarios en la cola. La fórmula para determinar esta probabilidad es [1, Cap. 14]

$$p_n = \frac{(\lambda/\mu)^n e^{-\lambda/\mu}}{n!}. \tag{2.2}$$

De la cual deducimos el número de usuarios medio en la cola L:

$$L = \sum_{n=1}^{\infty} n p_n = e^{-\lambda/\mu} \sum_{n=1}^{\infty} n \frac{(\lambda/\mu)^n}{n!} = e^{-\lambda/\mu} \frac{\lambda}{\mu} \sum_{n=1}^{\infty} \frac{(\lambda/\mu)^{n-1}}{(n-1)!} = \frac{\lambda}{\mu} \tag{2.3}$$

2.3.2. La cola $M/M/c/c$

La cola $M/M/c/c$ es un caso especial de la cola $M/M/c/K$, donde tenemos c servidores y una capacidad total de K. En este caso, $K = c$, por lo que no hay usuarios en espera, si un usuario

llega y ve todos los recursos ocupados, este será rechazado. Esto permite simplificar las fórmulas analíticas que se obtendrían para la cola de capacidad total K [1, Cap. 14].

Como para la cola $M/M/\infty$, nos interesaremos por la probabilidad p_n y el número medio de usuarios L. La soluciones encontradas para la cola $M/M/c/c$ son [1, Cap. 14]:

$$p_n = p_0 \frac{(\lambda/\mu)^n}{n!} \tag{2.4}$$

Teniendo en cuenta:

$$\sum_{i=0}^{c} p_i = 1 \tag{2.5}$$

encontramos la expresión para p_0:

$$p_0 = \left[\sum_{n=0}^{c} \frac{(\lambda/\mu)^n}{n!} \right]^{-1} \tag{2.6}$$

Por lo que deducimos la siguiente fórmula para L:

$$L = \sum_{n=0}^{c} n p_n = \sum_{n=1}^{c} \frac{(\lambda/\mu)^n}{(n-1)!} p_0 \tag{2.7}$$

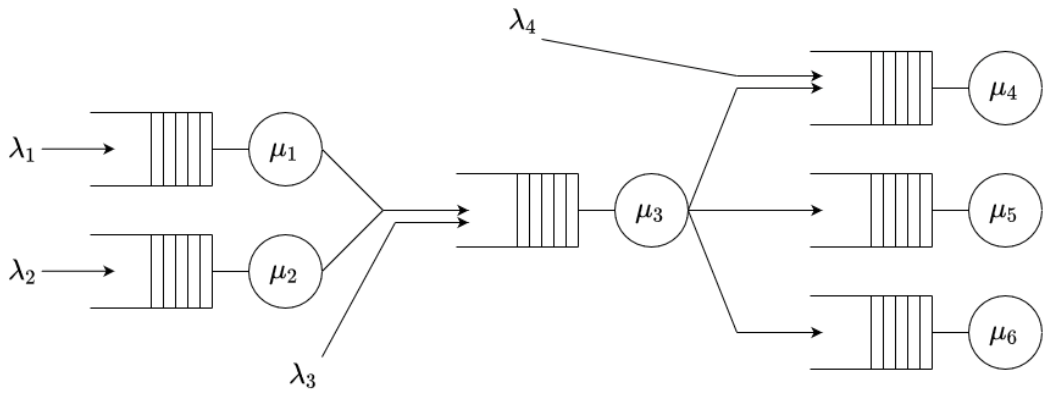


Figura 2.2: Diagrama de agregación y descomposición de procesos de Poisson en una red de colas

3. Redes de colas

Una red de colas se compone de varias colas, definidas en la sección 2, interconectadas entre sí, donde la salida de una se dirige hacia la entrada de una o varias colas de la red. Se presentará los grandes rasgos de una red de colas y los teoremas más relevantes que permiten su estudio.

Existen dos tipos de redes de colas. Nos limitaremos a estudiar las redes de colas abiertas, es decir, cuyos usuarios pueden tanto entrar de la red como salir de ella; a diferencia de las redes cerradas donde un número determinado de usuarios transitan indefinidamente dentro de la red.

3.1. Teorema de Burke

El teorema de Burke nos dice que en una cola $M/M/1$, $M/M/c$ o $M/M/\infty$ alimentada por un proceso de Poisson con tasa de llegadas λ el proceso de salidas también será de Poisson con la misma tasa λ.

Gracias al teorema de Burke se pueden analizar una red de colas, cuyos nodos se suceden unos a otros. En efecto, si las **salidas** de los nodos son de Poisson, también lo serán las **entradas** al nodo siguiente, propiedad necesaria en los resultados analíticos de las colas. Cada nodo se puede analizar por separado, y deducir los resultados de la red conjunta en forma de producto de los resultados previos. Además, teniendo en cuenta que la agregación y la descomposición de procesos de Poisson también son procesos de Poisson, podemos estudiar configuraciones de redes donde un nodo puede tener varios nodos de destino distinto de él mismo, o la llegada a cierto nodo es una composición de salidas de varios otros nodos, casos representados en la figura 2.2.

Por otra parte, en el teorema de Burke solo habla de redes cuyos nodos se *suceden* unos a otros. La limitación del teorema de Burke para el análisis de redes de colas se encuentra en la aparición de bucles de retroalimentación entre nodos. Esta retroalimentación crea llegadas a los nodos retroalimentados que no son de Poisson [1, Cap. 15]. Afortunadamente, y como veremos a continuación, seguiremos pudiendo analizar de manera sencilla este tipo de redes.

3.2. Teorema de Jackson

En una red de colas con retroalimentación entre nodos, los procesos de llegadas no son de Poisson, lo que a priori limita el estudio de estas redes. J. R. Jackson demostró que a pesar de esto, en redes de Jackson todos los nodos actúan *como si* estuvieran alimentados por procesos de Poisson [1, Cap. 15]. Esto favorece la simplicidad del estudio de las redes, puesto que se podrá considerar la entrada retroalimentada de un nodo como de Poisson, pudiendo entonces utilizar los resultados analíticos de las colas. Además, aplicando el teorema de Burke, las salidas de estos nodos también tendrán consideración de Poisson, por lo que, tendremos un escenario donde se podrá evaluar cada nodo por separado y encontrar una solución conjunta en forma de producto.

3.2.1. Red de Jackson

Considerando una red de M nodos, será una red de Jackson si tiene las siguientes propiedades:

- Cada nodo tiene una cola **FCFS** y un número c_i de servidores exponenciales de tasa μ_i.
- Las llegadas externas son de Poisson con tasa γ_i.
- Después de completar la estancia en el nodo i, pasará a un siguiente nodo j con probabilidad r_{ij} o saldrá del sistema con probabilidad $1 - \sum_{j=1}^{M} r_{ij} > 0$.

3.2.2. Ecuaciones de flujo

Teniendo en cuenta esta definición de una red de Jackson, podemos calcular la tasa total de llegadas a un nodo i, que notaremos λ_i, como la agregación de tasas de llegadas externas, γ_i y las llegadas que provienen de otros nodos $\lambda_j, \forall j \in M$ (que hemos visto que no tienen por qué ser de Poisson). La fórmula para calcular la tasa de llegadas a cada nodo i es:

$$\lambda_i = \gamma_i + \sum_{j=1}^{M} \lambda_j r_{ji} \tag{2.8}$$

El conjunto de estas ecuaciones para cada nodo de una red de Jackson forman un sistema de ecuaciones lineal no homogéneo con matriz de coeficientes no singular (una única solución al sistema), cuya incógnita es el vector de tasas λ.

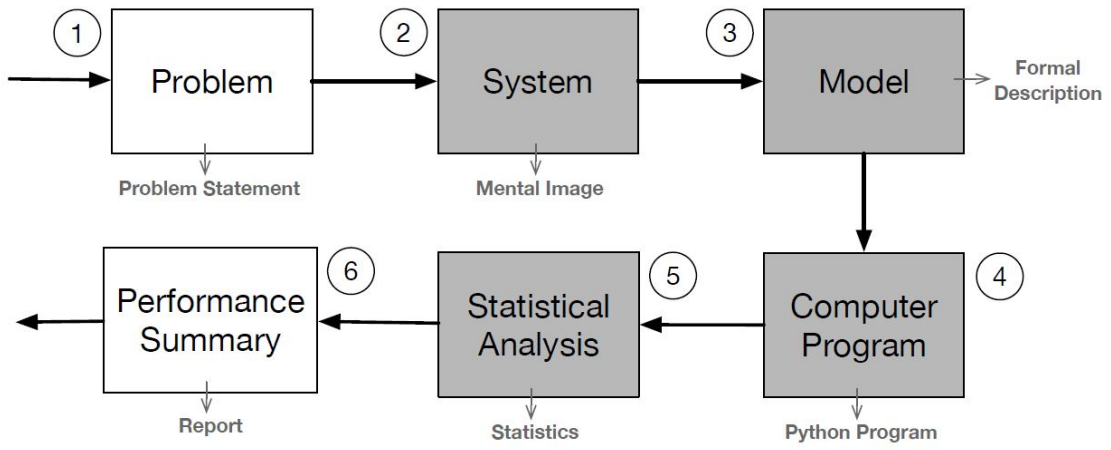


Figura 2.3: Fases de un estudio de simulación [2, Cap. 1]

4. Simulación

La simulación es una herramienta que permite el estudio de sistemas reales o teóricos, en los que de otra manera, a través de la experimentación sobre el sistema o del análisis matemático, sería muy complejo y costoso. Formalmente, "la simulación es el proceso de modelar un sistema y ejecutarlo para generar datos que posteriormente serán analizados para la extracción de información de estos"[2].

Para realizar un estudio de simulación, en primera instancia habrá que realizar un modelo del sistema que se quiera simular. El modelo se basará en las características claves que se quieran destacar del sistema a simular y en el comportamiento de este. Al realizar un modelo, se realizarán ciertos compromisos que obviaran alguna parte del sistema a simular. El desarrollo de un modelo de simulación será más o menos complicado según la complejidad del sistema a simular, y los resultados que se deseen obtener. Posteriormente, este modelo deberá ser traducido a un lenguaje de programación para que sea ejecutado por un ordenador. La carencia de lenguajes dedicados a la simulación y de librerías adaptadas a la simulación, hace de esta tarea una tarea delicada y minuciosa. La ejecución del modelo deberá aportar datos, que de alguna manera serán tratados estadísticamente para analizarlos dentro del marco del estudio que se desee realizar. Las distintas fases a llevar en un estudio de simulación se pueden ver gráficamente en la figura 2.3. En [2, Cap. 1] se explica en más detalle cada fase.

Un estudio de simulación puede verse como una vía rápida para el estudio de un sistema sin mucho coste de análisis teórico. Mientras que esto es cierto, la simulación no deja de ser una de las muchas herramientas disponibles para estudios, y no esta exenta de inconvenientes, que hay que considerar individualmente. En la tabla 2.1 se retoman las ventajas e inconvenientes que surgen del uso de la simulación para un estudio. Estos puntos han sido retomados de [2, Cap. 1].

Ventajas	Inconvenientes
Escenarios críticos (límites) del sistema pueden ser analizados a coste bajo y sin riesgo.	La generación de número aleatorios esta sujeta a errores estadísticos: un mismo modelo puede dar resultados ligeramente distintos.
Se pueden cambiar parámetros del modelo sin alterar el sistema físico modelado	El estudio de simulación puede requerir personal altamente cualificado y de recursos computacionales elevados.
La simulación es más flexible y accesible que el análisis matemático	El desarrollo de un modelo de simulación puede ser difícil.
A diferencia de un análisis teórico, no es necesario hacer suposiciones del sistema con objetivo de simplificar su análisis.	Los lenguajes de programación no están diseñados para la simulación, por lo que la programación del sistema se vuelve ardua.
El estudio temporal del modelo es más flexible: se pueden simular varios años de un sistema en unos minutos de tiempo computacional.	

Tabla 2.1: Comparación ventajas e inconvenientes de un estudio de simulación

4.1. Simulación por eventos discretos

Una simulación se puede abordar de distintas maneras, que pueden ser más o menos adecuadas según el sistema a simular. Presentaremos brevemente los distintos tipos de simulación, y explicaremos en detalle el tipo que se adoptará para desarrollo del trabajo.

Podemos encontrar simulaciones de procesos continuos, que se basan en sistemas de ecuaciones diferenciales, por ejemplo en el ámbito de la meteorología [1, Cap. 19]. Asimismo, encontramos simulaciones en las que el tiempo de simulación avanza en incrementos constantes ΔT [2]. En el caso que nos concierne, la simulación será por *eventos discretos*, donde el tiempo de simulación avanzará en intervalos variables, según ocurran nuevos eventos que influyen sobre el sistema. Estas dos ultimas maneras de abordar la simulación según el avance del tiempo de simulación son referidas respectivamente como de *simulación síncrona* y *simulación asíncrona*.

En la simulación por eventos discretos, aunque el sistema exista en tiempo continuo, los cambios que el sistema experimenta ocurren al producirse eventos definidos [1, Cap. 19]. Por ende, como comentado anteriormente, los incrementos del tiempo de simulación serán variables, a cada aparición de nuevos eventos del sistema.

En un primer lugar definiremos los componentes individuales de una simulación por eventos discretos, para finalmente describir una arquitectura del sistema completo de simulación.

Para la simulación por eventos discretos, definiremos los elementos principales que intervienen en el modelo de simulación. Estos elementos son:

- Descriptor del sistema: variable (o variables) que define(n) el sistema a simular.
- Eventos: los acontecimientos que hacen avanzar el sistema.
- Generación de eventos: define en base a que parámetros se generan los eventos.

- Gestor de eventos: definen el impacto que ha de tener el evento tratado sobre el descriptor del sistema.
- Planificación de los eventos: define como serán programados los eventos que van surgiendo a lo largo de la simulación. Cada evento ocurre en un cierto instante temporal, por lo que su ejecución deberá ser acorde a este orden.

A lo largo de la simulación, se van generando eventos, los cuales serán puestos en una lista de eventos acorde a la planificación de los eventos. Cuando se ejecuta un evento, el gestor de eventos materializa el efecto de este sobre el descriptor del sistema. De esta manera se relacionan los distintos elementos de la simulación por eventos discretos.

Para iniciar la simulación, en primer lugar, el sistema se pondrá en un estado inicial, que por simplicidad se definirá como todos los elementos del descriptor del sistema a 0. Otras aproximaciones a la inicialización se podrían usar, como el la asignación del descriptor del sistema por los valores esperados teóricos. Posteriormente se generarán los eventos iniciales del sistema. Esto nos asegura el "arranque" del sistema. Estas dos etapas marcan la inicialización de la simulación. Durante la simulación, se irán muestreando los estadísticos que se consideren oportunos, usualmente en base al estado del descriptor del sistema. Finalmente, la simulación finalizará cuando se haya alcanzado el tiempo de simulación preestablecido (alternativamente, se puede marcar un número máximo de evento en el sistema, como umbral para la finalización de la simulación). Cuando la simulación haya acabado, se calcularán los valores estadísticos para su posterior análisis, en el marco del estudio. El resumen de estas etapas se puede ver en la figura 2.4.

4.2. Precisión de los resultados

Como se ha podido mencionar en apartados anteriores de esta sección, distintas simulaciones del mismo modelo pueden resultar en resultados ligeramente diferentes. Esto es debido principalmente a la naturaleza pseudoaleatoria de la generación de números aleatorios. En cada simulación realizada, la semilla de este generador de números aleatorias cambiará, para que la generación de números no tenga dependencia con las simulaciones previas. Para extraer resultados robustos para su estudio, se agregaran los resultados de distintas simulaciones. A cada una de estas simulaciones se las conoce como "réplicas"de la simulación.

4.2.1. Intervalos de confianza

Al agregar resultados de varias simulaciones, se calcularán *intervalos de confianza*, que estiman un intervalo en el cual el valor buscado se encontrará con cierta probabilidad p. Pongamos un caso en el que se tiene un estimador α de la media de una determinada muestra. Este estimador muestral estimará el valor de μ. Sin embargo, el estimador muestral no manifiesta la exactitud o precisión con la que se esta estimando el valor real. Si α difiere de μ, el estimador muestral no cuantifica esta diferencia.

Por el contrario, un intervalo de confianza cuantificará este variación del valor, pero no dará un valor "exacto"para la estimación de este. Un intervalo de confianza nos dirá con que probabilidad $1 - p$ el valor buscado μ se encuentra en un intervalo $(\alpha - \epsilon, \alpha + \epsilon)$, con α el estimador muestral de la media mencionado en el párrafo anterior. Los intervalos de confianza estarán centrados en el

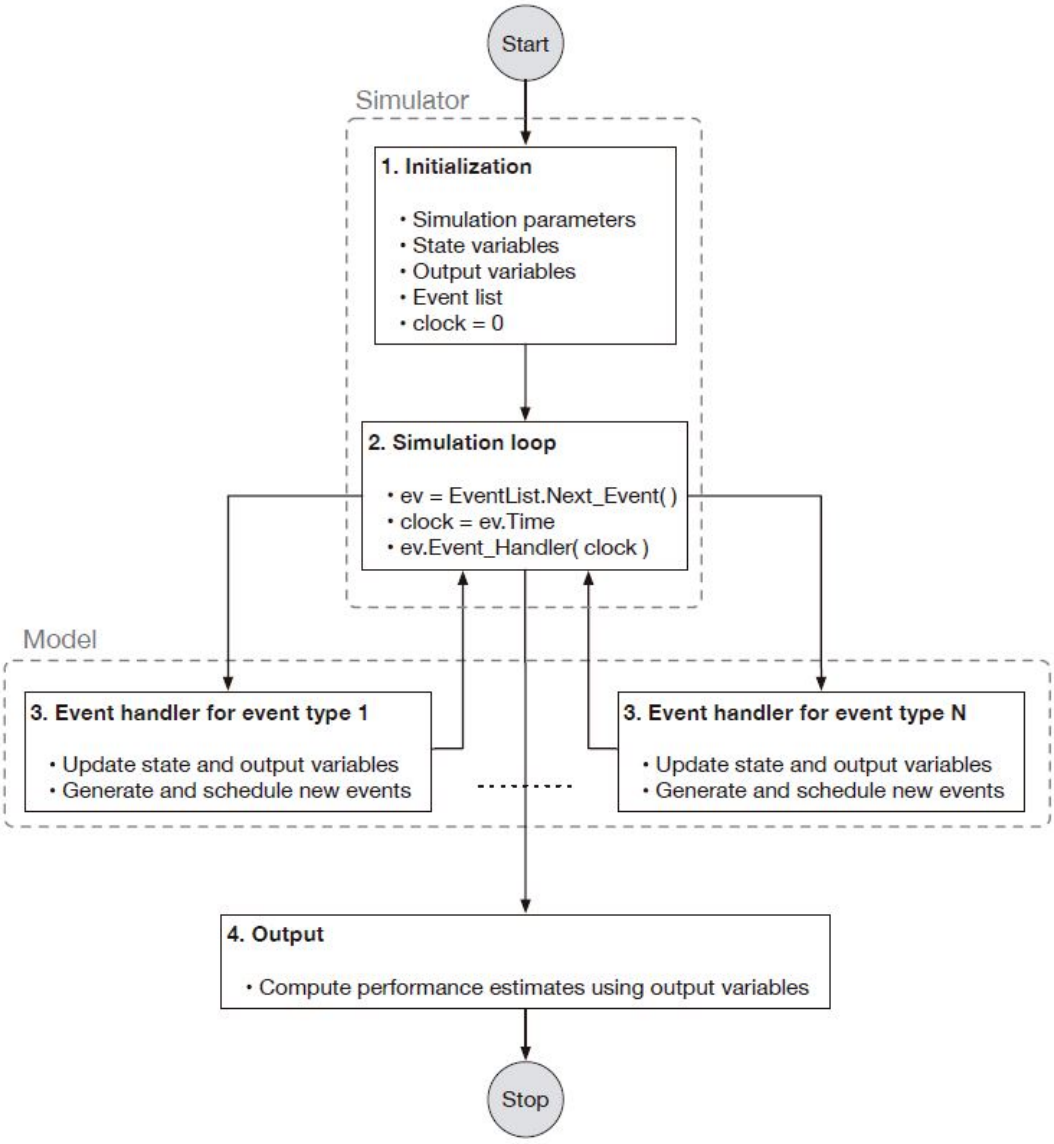


Figura 2.4: Arquitectura de un sistema de simulación por eventos discretos [2, Cap. 8]

estimador muestral de la media α. Este intervalo dependerá del tamaño de la muestra. De [1, Cap. 20] extraemos la fórmula del valor ϵ del intervalo de confianza buscado:

$$\epsilon = \frac{z_{\zeta/2}S}{\sqrt{n}} \tag{2.9}$$

donde S es el estimador de la desviación típica

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \alpha)^2 \tag{2.10}$$

y n el tamaño de la muestra. El valor $z_{\zeta/2}$ dependerá de la "precisión"buscada p. Sea $Z = \frac{\alpha-\alpha}{S/\sqrt{n}}$, esta variable aleatoria Z tiene una distribución *t* de Student. Entonces, $z_{\zeta/2}$ es el valor para el cual $P(Z > z_{\zeta/2}) = \frac{\zeta}{2}$, con $\zeta = 1 - p$.

4.2.2. Transitorio de la simulación

Tras la inicialización del sistema en cada réplica, el sistema se encuentra en un estado inicial. Por lo tanto el sistema no estará funcionando en régimen permanente, y hasta que este se alcance los valores estadísticos muestreados podrían no ser del todo exactos. Pongamos que tenemos una cola con un número determinado de servidores y 0 posiciones de espera. En su estado inicial tiene 0 usuarios. Hasta que a esta no haya llegado una cierta cantidad de usuarios, la probabilidad de que un usuario temprano no pueda ser atendido es nula. Pero en régimen permanente, la probabilidad de no ser atendido (con cierta tasa de llegadas y de servicio) podría no ser nula. Por lo tanto, hay un claro sesgo en la estimación "temprana" de los estadísticos. Este período se conoce como el *transitorio*.

Existen varias maneras de detectarlo y mitigar su efecto. En [2] se explica una manera sencilla, que consiste en mirar la evolución temporal del estadísticos buscado sobre varias réplicas, es decir, promediar (para eliminar cualquier efecto provocado por la aleatoriedad) esta evolución temporal, y buscar el instante temporal en el que este empieza a converger hacia un valor.

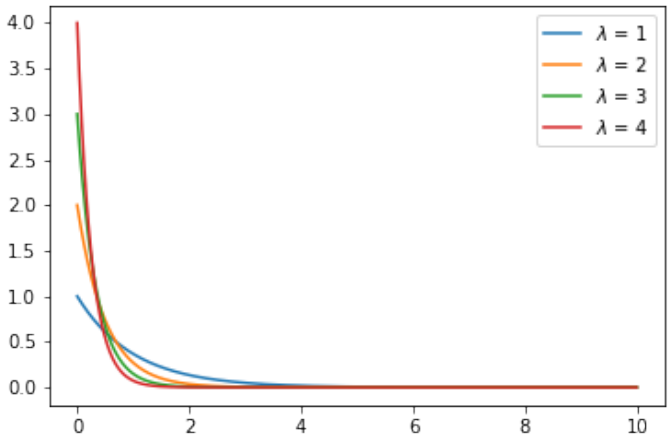


Figura 2.5: PDF distribución exponencial para distintas λ.

5. Distribuciones

En esta sección se presentaran las distribuciones de probabilidad que serán usadas para modelizar los tiempos entre llegadas, exclusivamente la exponencial, y los tiempos de residencia y de sesión.

Será de interés conocer tanto la esperanza como el coeficiente de variación (por consecuente su varianza) de la distribución. El coeficiente de variación expresa la desviación típica como fracción de la esperanza, mostrando el grado de variabilidad de la distribución independientemente de los valores adoptados por la variable aleatoria que siga esa distribución. Esto permite comparar con más precisión la variabilidad de distintas distribuciones, a diferencia de la desviación típica. Además, nos permite caracterizar a partir de este valor las distribuciones. Se define como:

$$C_V = \frac{\sigma_X}{E[X]} \tag{2.11}$$

5.1. Exponencial

La distribución exponencial tiene la siguiente función de densidad de probabilidad (PDF en inglés):

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & si \quad x \geq 0 \\ \\ 0 & si \quad x < 0 \end{cases} \tag{2.12}$$

La esperanza y la varianza de la distribución exponencial son:

$$E[X] = \frac{1}{\lambda}, \quad \sigma^2 = \frac{1}{\lambda^2} \tag{2.13}$$

Por lo que deducimos el valor del coeficiente de variación:

$$C_V = \frac{\sqrt{\sigma_X^2}}{E[X]} = \frac{\sqrt{1/\lambda^2}}{1/\lambda} = 1 \tag{2.14}$$

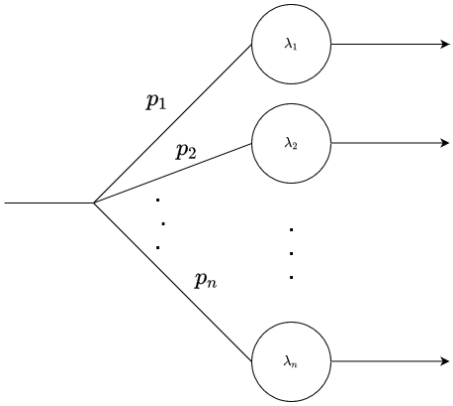


Figura 2.6: Diagrama de una función hiper-exponencial

Su principal característica, y lo que la hace interesante, es su propiedad de **memoria nula**, que se expresa de la siguiente forma, para una variable aleatoria X que sigue una distribución exponencial:

$$\begin{gathered}\text{si } X \sim \exp(\lambda), \text{entonces} \\ P(X \leq t_0 + t | X > t_0) = P(X \leq t)\end{gathered} \tag{2.15}$$

En el caso del tiempo entre llegadas, esta propiedad nos dice que si pasa un tiempo t_0 después de una llegada, no podemos inferir nada sobre el tiempo restante hasta la siguiente llegada; o lo que es lo mismo, no tiene en cuenta cuánto tiempo ha pasado desde la última llegada.

5.2. Hiperexponencial

La función de densidad de probabilidad para una variable aleatoria X que sigue una distribución hiperexponencial es

$$f_X(x) = \sum_{i=1}^{n} f_{Y_i}(x) p_i \tag{2.16}$$

donde $Y_i \sim \exp(\lambda_i)$ y $\sum p_i = 1$. Gráficamente se expresaría como en la figura 2.6.
Al juntar varias exponenciales conseguimos aumentar el *coeficiente de variación* [1, Cap. 7].
La esperanza de la distribución hiper-exponencial y su varianza es:

$$E[X] = \sum_{i=1}^{n} \frac{p_i}{\mu_i}, \quad \sigma_X^2 = \left[\sum_{i=1}^{n} \frac{p_i}{\mu_i}\right]^2 + \sum_{i=1}^{n}\sum_{j=1}^{n} p_i p_j \left(\frac{1}{\mu_i} - \frac{1}{\mu_j}\right)^2 \tag{2.17}$$

Lo que dará un coeficiente de variación superior a la unidad [3]:

$$C_V = 1 + \frac{1}{E[X]} \sum_{i=1}^{n}\sum_{j=1}^{n} p_i p_j \left(\frac{1}{\mu_i} - \frac{1}{\mu_j}\right)^2 \geq 1 \tag{2.18}$$

5.3. Lognormal

Si la variable Y que sigue una distribución lognormal entonces $X = \ln(Y)$ sigue una distribución normal. La función de densidad de probabilidad para la variable, representada en la figura 2.7,

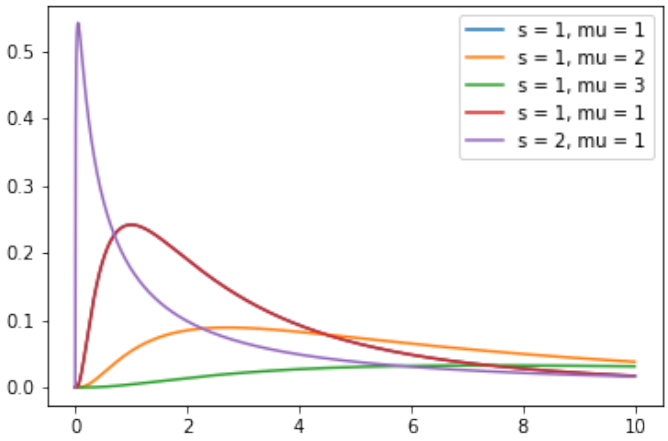


Figura 2.7: PDF de la distribución log-normal con variación de parámetros.

$Y \sim Lognormal(\mu_x, \sigma_x^2)$, suponiendo $\ln(Y) \sim \mathcal{N}(\mu,\, \sigma^2)$, es la siguiente:

$$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}} \tag{2.19}$$

Con esperanza:

$$E[Y] = e^{\mu + \frac{\sigma^2}{2}} \tag{2.20}$$

Para conseguir una distribución de media μ_x y varianza σ_x^2, se usarán los siguientes valores para la distribución normal subyacente:

$$\mu = \ln \frac{\mu_x^2}{\sqrt{\mu_x^2 + \sigma_x^2}}, \quad \sigma^2 = \ln(1 + \frac{\sigma_x^2}{\mu_x^2}) \tag{2.21}$$

5.4. Weibull

La distribución Weibull tiene la siguiente función de densidad de probabilidad (dos parámetros), representada en 2.8:

$$f(x; \eta, \beta) = \frac{\beta}{\eta} \left(\frac{x}{\eta}\right)^{\beta - 1} e^{-\left(\frac{x}{\eta}\right)^{\beta}}, \quad x \geq 0 \tag{2.22}$$

Si $\eta = 1$, encontramos la distribución exponencial vista anteriormente (2.12).
La esperanza de la distribución Weibull es:

$$E[X] = \eta \Gamma \left(1 + \frac{1}{\beta}\right) \tag{2.23}$$

y su varianza

$$\sigma_X^2 = \eta^2 [\Gamma(1 + \frac{2}{\beta}) - \Gamma(1 + \frac{1}{\beta})^2]. \tag{2.24}$$

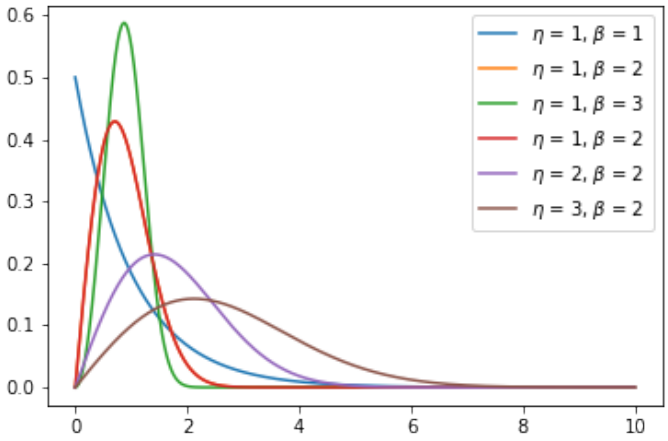


Figura 2.8: PDF de la distribución Weibull con variación de parámetros.

De estos resultados derivamos el calor del coeficiente de variación:

$$C_V = \sqrt{\frac{\Gamma(1+\frac{1}{\beta})}{\Gamma(1+\frac{2}{\beta})^2} - 1} \tag{2.25}$$

5.5. Pareto

Las distribuciones de Pareto son una familia de distribuciones. Nos centraremos en la "Tipo I". La distribución de Pareto de Tipo I tiene la siguiente función de densidad de probabilidad, representada en 2.9:

$$f(x;\alpha,\sigma) = \frac{\alpha\sigma^{\alpha}}{x^{\alpha+1}}, \quad x \geq \sigma \tag{2.26}$$

Donde α es el parámetro de forma y σ de escala. σ corresponde con el mínimo valor que la distribución alcanza. Su esperanza y varianza son:

$$E[X] = \begin{cases} \frac{\alpha\cdot\sigma}{\alpha-1}, & \alpha > 1 \\ \\ \infty, & \alpha \leq 1 \end{cases} \tag{2.27}$$

$$\sigma_X^2 = \begin{cases} \frac{\sigma^2\alpha}{(\alpha-1)^2(\alpha-2)}, & \alpha > 2 \\ \\ \infty, & \alpha \leq 2 \end{cases} \tag{2.28}$$

De estos resultados derivamos el valor del coeficiente de variación:

$$C_V^2 = \frac{\sigma^2\alpha}{(\alpha-1)^2(\alpha-2)} \cdot (\frac{\alpha-1}{\alpha\cdot\sigma})^2 = \frac{1}{\alpha(\alpha-2)}, \quad \alpha > 2. \tag{2.29}$$

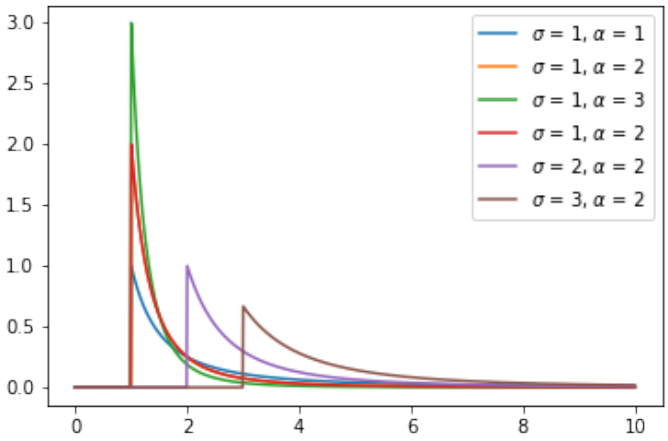


Figura 2.9: PDF distribución Pareto para distintos parámetros α, σ.

5.6. Gamma

La distribución gamma tiene la siguiente función de densidad de probabilidad:

$$f(x;\mu,n) = \frac{\mu^n x^{n-1} e^{-\mu x}}{\Gamma(n)}, \quad x > 0 \tag{2.30}$$

Siendo $\Gamma(n)$ la función gamma definida por (2.31), n el factor de forma y μ el factor de escala. El factor de forma altera la forma de la curva de la función de densidad de probabilidad como se puede ver en la figura figura 2.10a, mientras que el factor de escala solo aumenta o disminuye vertical y horizontalmente la curva, como se puede ver en la figura figura 2.10b.

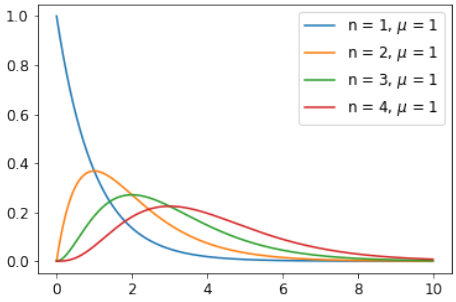
$$\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x} dx, \quad \alpha > 0 \tag{2.31}$$

La esperanza y la varianza de la distribución gamma es:

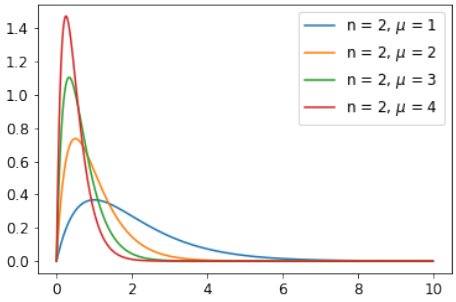
$$E[X] = \frac{n}{\mu}, \quad \sigma^2 = \frac{n}{\mu^2} \tag{2.32}$$

Por lo tanto, el coeficiente de variación para la distribución gamma será:

$$C_V = \frac{\sqrt{n/\mu^2}}{n/\mu} = \frac{1}{\sqrt{n}} \tag{2.33}$$



(a) PDF Gamma en función del factor de forma n.



(b) PDF Gamma en función del factor de escala μ.

Figura 2.10: Comparación de la PDF en función de como varían los factores de forma y de escala

Capítulo 3

Simulador

En este capítulo se explicará en detalle el desarrollo del simulador de redes móviles que implementan network slicing.

El objetivo es desarrollar un simulador de redes celulares con implementación de "slices", parametrizable a través de un archivo de configuración, que proporcione estadísticas útiles para el estudio de redes móviles que implementan network slicing.

Para el desarrollo de nuestro simulador, nos apoyaremos en el libro *Osais, Computer Simulation, A foundational approach using Python* [2], que desarrolla un simulador por eventos discretos para una cola separada, que extenderemos para simular una red de colas con clase de usuarios.

En primer lugar se enumerarán las distintas herramientas técnicas utilizadas para la programación del simulador, para luego detallar su desarrollo. Finalmente, se validará el funcionamiento del simulador.

1. Herramientas técnicas

Para el desarrollo de un tal sistema se ha procedido a usar las siguientes herramientas técnicas.

1.1. Python

Para el desarrollo del simulador se ha optado por utilizar el lenguaje Python, debido a su sencillez y gran cantidad de librerías que simplifican el desarrollo. La versión utilizada ha sido la 3.8.5 [4].

Numpy
Numpy es una librería de Python de cálculo numérico. Da soporte al uso de vectores y matrices. Como parte del cálculo numérico, proporciona funciones de álgebra lineal que nos serán útiles en el desarrollo del trabajo. Es el equivalente de Matlab dentro del lenguaje multipropósito Python.

Matplotlib
Matplotlib es una librería de Python de generación de gráficas, a partir de estructuras de datos

nativas de Python y también de vectores y matrices de Numpy. Ofrece una gran variedad de gráficos y de personalización de estos, como de opciones de exportación hacia archivos de tipo mapa de bits (.png, ...) como de formato vectorial (.eps, ...).

1.2. Visual Studio Code

El entorno de desarrollo usado ha sido Visual Studio Code [5] junto con la extensiones adaptadas para trabajar con los archivos ".ipynb".

1.2.1. Jupyter

El entorno Jupyter [6], instalado a través de Anaconda [7], permite la ejecución de bloques de código Python secuencialmente. Tiene su propio formato de archivo ".ipynb", conocidos popularmente como "notebooks". Su principal ventaja viene a la hora de analizar y representar datos, ya que la secuenciación en bloques permite aislar de manera lógica la parte de lectura de datos, su procesado y su posterior visualización, de manera más dinámica que en un script tradicional.

1.3. Git

Git es una herramienta de control de versiones, que permite mantener un historial de las modificaciones de los archivos, para comparar código con versiones antiguas, o poder trabajar en nuevas funcionalidades a la vez que se guarda una referencia a una versión funcional anterior. Esta funcionalidad, sumado a la función de ramas de Git, permite que se pueda guardar diferentes historiales de cada fichero (por nueva característica por ejemplo), como si de ramas de un árbol se trataran.

2. Modelo de simulación

Como hemos definido, una simulación requiere de la modelización del sistema antes de pasarlo a un lenguaje de programación.

Usaremos las redes de cola para modelizar la red celular con slices que se pretende simular. Cada célula de la red será modelada como una cola cuya capacidad representará la cantidad de recursos de los cuáles dispone la célula. Las slices serán representadas por clases de usuarios de una red de colas [1, Cap.15]. Las clases de usuarios serán caracterizadas por el tiempo que residen dentro de una célula de la red y por la duración de las sesiones.

3. Componentes del simulador

Apoyándonos en las especificaciones de un simulador por eventos discretos visto en el la sección 4, definiremos como se ha desarrollado nuestro sistema adecuándolo a las redes de colas con clase de usuarios. Al final de la sección se mostrará el diagrama final de la estructura del software del simulador.

3.1. Descriptor del sistema

Para una red de colas, el descriptor del sistema será un vector Q cuyo elemento de índice n guardará el número de usuarios en el nodo n de nuestra red (empezando a numerar los nodos a partir de 0).

3.2. Los eventos

Los eventos en nuestra simulación serán representados por una tupla (estructura de datos nativa de Python) con la siguiente estructura:

$$(t, ID, nodo, handler, clase\_de\_servicio, tiempo\_de\_sesión) \tag{3.1}$$

- t: el instante temporal en el que el evento ocurrirá.
- ID: identificador único del evento.
- $handler$: función encargada de tratar el evento.
- $clase\_de\_servicio$: corresponde a la slice a la que el usuario pertenece.
- $tiempo\_de\_sesión$: instante temporal en el que la sesión finalizará.

Los eventos serán guardados en una *cola con prioridad*. Una cola es una estructura de datos cuyo último elemento añadido es el primero en salir. La prioridad altera esta propiedad básica puesto que nos permite modificar el orden de salida de los elementos. En nuestro caso usaremos *PriorityQueue* de la librería *queue* de Python. El campo t de los eventos nos permite gestionar la prioridad de los eventos dentro de la cola. Una marca temporal t inferior será siempre colocada por delante en la cola de salida que otro evento cuya marca temporal sea superior. Esto nos permite mantener el orden cronológico de los eventos. El campo ID, que se auto-incrementa con cada evento creado, será el siguiente verificado en caso de dos eventos con el mismo valor en el campo t. Esto permite que no haya problemas al añadir dos eventos que deben ocurrir al mismo tiempo. El ser ID un campo que incrementa con la creación de cada evento, los eventos posteriores tendrán un ID mayor, por lo que en caso del conflicto mencionado anteriormente, este evento será puesto más tarde en la cola de salida: si dos eventos ocurren al mismo tiempo, el primer evento creado será el primero en ocurrir.

Los eventos en nuestra simulación se dividirán en 4 tipos:

- llegadas externas
- llegadas de traspasos (llegadas desde otro nodo, a las que nos referiremos a veces como internas)
- salidas de traspasos (salida hacia otro nodo de la red)
- salidas externas (finalización de una sesión)

3.3. Gestión de los eventos

Una vez se extraen los eventos de la cola, serán tratados por la función handler, cuya referencia va incorporada al propio evento, como indicado en 3.1. Esta función materializa el impacto que tiene el evento sobre el sistema.

Una sesión nueva será admitida en la red si se dispone de suficientes recursos para atenderla. Al tratarse de una sesión nueva, habrá que mirar que queden recursos libres "no prioritarios", es decir, descartando los recursos reservados para los traspasos. Un traspaso será admitido en un nuevo nodo si esta cuenta con recursos disponibles, al tratarse de usuarios prioritarios, estos podrán hacer uso de todos los recursos del nodo.
Una llegada externa ocupará un recurso dentro del nodo de llegada, además de generar un nuevo evento del mismo tipo, en función de la tasa de llegadas externas y la distribución que siga. Una llegada interna, solo ocupará un recurso en el nodo de llegada. Cuando un usuario llega a un nodo (independientemente de su proveniencia), se genera el evento de salida de este, a partir de la distribución y los parámetros del tiempo de residencia del usuario. Esta salida será hacia otro nodo de la red, o fuera de la red, si la sesión hubiera terminado. Cuando se trata un evento de salida, se liberará un recurso en el nodo de origen, y si se trata de un traspaso, además se pasaría a ocupar un recurso en el nodo de destino (si este es aceptado).

3.3.1. Cálculo nodo de destino

Para determinar el nodo de destino de un traspaso se usará la matriz de encaminamiento R^c, característica para cada "slice". Corresponde a uno de los parámetros de entrada del simulador. La matriz de encaminamiento contiene en la posición (i, j) la probabilidad, condicionada a que se produzca un traspaso, de transición del usuario de clase c del nodo i al nodo j, que notaremos de aquí en adelante r_{ij}^c. Como veremos más adelante, la probabilidad total de que un usuario de la slice c transite de un nodo i a otro nodo j, que notaremos p_{ij}^c, se calculará en función de las tasas de sesión y de residencia, respectivamente μ_S^c y μ_r^c.

3.4. La generación de eventos

Otro elemento del simulador es la generación de los eventos. La generación de estos se basa en la distribución de probabilidad que sigue cada variable aleatoria asociada a cada evento. Por una parte, el tiempo entre llegadas, al tratarse de un proceso de Poisson, sigue una distribución exponencial. Serán caracterizadas por una tasa de llegadas por nodo y por slice, definidas en un diccionario de Python, cuyas claves serán el identificador de la slice y el valor corresponderá con un vector que definirá el valor por nodo de esta propiedad. Por otra parte, el tiempo de sesión y el tiempo de residencia, pueden adoptar cualquiera de las distribuciones mencionadas en la sección 5. También se definirán con la ayuda de un diccionario, salvo que esta vez, el valor de esta propiedad es característica de la clase de usuario en cuestión, por lo que el valor asociado no será un vector pero un valor (o dos, como veremos a continuación) que la caracterizará.

3.4.1. Generación de números aleatorios

Para la generación de números aleatorios usaremos la librería *random* de Numpy. Esta proporciona un *generador de números aleatorios* que es capaz de generar números aleatorios de las distribuciones que necesitaremos. Este generador será reinicializado con una nueva semilla en cada réplica.

Como cada distribución requiere de un número distinto de argumentos dados (la distribución exponencial sólo 1, el resto de distribuciones 2), se creará unas funciones "fachada", patrón de diseño "facade" en inglés, de cada distribución que permitan una ejecución transparente de estas.

Para caracterizar las distribuciones de probabilidad, nos fijaremos como referencia la distribución exponencial. Esta se caracteriza por la tasa a la que ocurren los eventos λ. Esta es igual a la inversa de la esperanza de una variable que siga esta distribución $E[X] = \frac{1}{\lambda}$. Su coeficiente de variación es de 1. Para el resto de distribuciones nos pondremos como objetivo mantener la misma "tasa", equivalentemente, mantener la misma media. Partiendo de la tasa, requerimos de otro parámetro para poder caracterizar las distribuciones . Este segundo valor característico será el *coeficiente de variación*. Esto nos permite obtener de las distribuciones un comportamiento similar (en cuanto a la esperanza se refiere) a la exponencial, pero con mayor o menor grado de variación. Usando las funciones fachada, el argumento de entrada será de 1 valor numérico para la exponencial, y de una lista de 2 valores numéricos, correspondientes a la "tasa" y el coeficiente de variación, para el resto de distribuciones.

3.4.1.1. Parametrización de la distribución gamma

Los parámetros de entrada de la distribución son μ y n, como descrito en el apartado 5. En entrada, tendremos $\lambda = \frac{1}{E[X]}$ y C_V. Basándonos en los resultados de la sección 5.6, despejamos los parámetros de entrada de la distribución:

$$n = \frac{1}{C_V^2} \tag{3.2}$$

$$\mu = \lambda n = \frac{\lambda}{C_V^2} \tag{3.3}$$

3.4.1.2. Parametrización de la distribución lognormal

Los parámetros de entrada de la distribución lognormal son μ y σ^2, media y desviación típica de la distribución normal subyacente. Como se ha visto en 5.3, podemos deducir estos valores a partir de la media y la varianza deseada para la distribución lognormal, μ_x y σ_x^2. Como antes, tenemos de entrada $\lambda = \frac{1}{E[X]}$ y C_V. Entonces los parámetros de entrada serán:

$$\mu_x = \frac{1}{\lambda}, \;\; \sigma_x^2 = (C_V \mu_x)^2 = \frac{C_V^2}{\lambda^2}. \tag{3.4}$$

Podemos sustituir estos valores en la fórmula para μ y σ^2 (5.3) y así conseguir la distribución deseada.

3.4.1.3. Parametrización de la distribución Weibull

Los parámetros de entrada de la distribución Weibull son β y η. A partir de los resultados de la sección 5.4 deduciremos los valores de entrada en función de los parámetros $E[X]$ y C_V. Para

encontrar el valor de β a partir de C_V se puede hacer uso de tablas numéricas existentes de C_V en función de β. Una vez tenemos β, despejaremos η en (2.23).

$$\eta = \frac{E[X]}{\Gamma(1 + \frac{1}{\beta})} \tag{3.5}$$

3.4.1.4. Parametrización de la distribución Pareto

Los parámetros de entrada de la distribución Pareto son α y σ. Para deducir su valor a partir de $E[X]$ y C_V en primer lugar calcularemos el valor de α. Su cálculo se hará despejando α de (2.29), resultando:

$$\alpha^2 - 2\alpha - 1/C_V^2 = 0. \tag{3.6}$$

Nos aseguraremos de que $\alpha > 2$, para que la distribución tenga valores finitos tanto de media como de varianza. Luego, simplemente calcularemos el valor de σ como sigue:

$$\sigma = \frac{E[X](\alpha - 1)}{\alpha}. \tag{3.7}$$

3.4.2. Tiempo de ocupación de recursos

La generación de los tiempos entre llegadas y de sesión son de aplicación directa, en cambio, la generación del tiempo de residencia dentro de una célula no es trivial, como veremos a continuación. Durante una sesión, un usuario potencialmente transita entre nodos de la red. En cada nodo (zona de cobertura de una célula) el usuario estará físicamente durante cierto tiempo que denominaremos *tiempo de residencia*. Durante este tiempo el usuario podrá hacer uso de los recursos de la red, por una llamada entrante o saliente, etc. Este tiempo lo denominaremos *tiempo de ocupación de recursos*. Estos dos tiempos son potencialmente distintos.

Cuando una nueva sesión se inicia en un nodo, podemos considerar que el usuario ya se encontraba en el nodo en cuestión, y por lo tanto, no podemos asumir que el tiempo de ocupación de recursos sea igual al tiempo de residencia. Es decir, no podemos decir que el usuario ha ocupado los recursos de la célula durante todo el tiempo que este ha estado en ella, puesto que no se tiene en cuenta el tiempo que ya se ha residido antes del comienzo de la sesión. Este tiempo que le queda al usuario en el nodo, igual al tiempo de ocupación de recursos, se conoce como el *tiempo de residencia residual*.

Notaremos cada "tiempo" implicado de la siguiente manera:

- tiempo de residencia residual: $\hat{t}_r$
- tiempo de residencia: t_r
- tiempo de sesión: t_s
- tiempo de ocupación de recursos: t_h

El algoritmo aplicado para determinar el tiempo de ocupación de recursos en un nodo de un usuario es el siguiente:

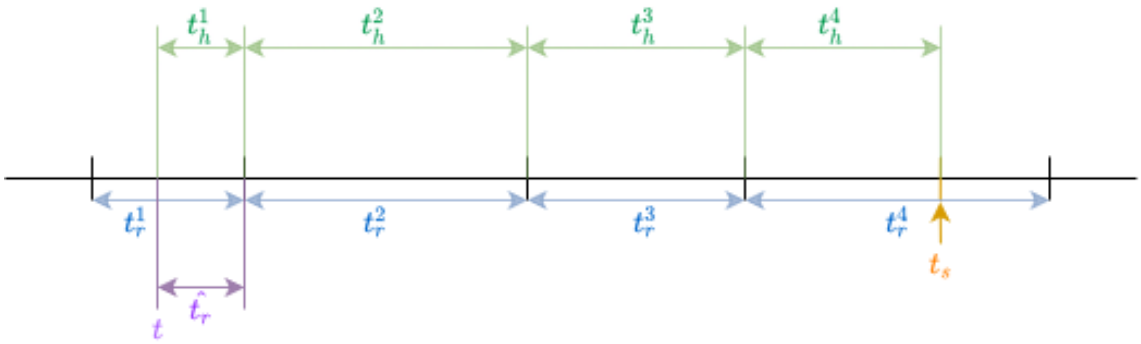


Figura 3.1: Relación entre los tiempos de residencia, tiempo de residencia residual, tiempo de sesión y tiempo de ocupación de recursos.

Distribución	Parámetros para t_r	Parámetros para $\hat{t_r}$
Gamma	$Gamma(\mu, n)$	$\mathcal{U}_{[0,1]} \cdot Gamma(\mu, n+1)$
Lognormal	$Lognormal(\mu, \sigma)$	$\mathcal{U}_{[0,1]} \cdot Lognormal(\mu + \sigma^2, \sigma)$

Tabla 3.1: Resumen de la distribuciones del tiempo de residencia residual según distribución

1. El evento de llegada es creado con una marca temporal t y un instante de finalización de sesión t_S (t_S es un "tiempo absoluto": relativo a $t_0 = 0$).

2. Dependiendo de si la sesión es nueva, o en cambio, está en curso, se procederá de manera diferente.

   - **el evento es tratado por primera vez:** en primer lugar se calcula el tiempo de residencia residual $\hat{t_r}$ (cuánto tiempo le queda el usuario dentro de la célula). Seguidamente, se procederá a determinar si el siguiente evento relacionado con la llamada en cuestión es un traspaso (la sesión seguirá en curso) o la finalización de esta. Asignaremos el tiempo del siguiente evento tal que así $min(t + \hat{t_r}, t_s)$. Es decir, el tiempo real que el usuario pasa ocupando los recursos dentro de una célula corresponde ya sea al tiempo que le queda transitando esa célula o al tiempo que le quede a su sesión.
   - **el evento es un traspaso:** suponemos que el traspaso ocurre en un instante t'. Se calcula el tiempo de residencia t_r (no el residual como en el otro caso) que el usuario estará en el nodo. El siguiente evento será el que antes ocurra, ya sea una salida hacia otro nodo o la finalización de la sesión. La marca temporal de ese evento será: $min(t' + t_r, t_S)$.

La Figura 3.1 nos muestra en un diagrama en función del tiempo, la relación entre los instantes temporales mencionados previamente.

Para el cálculo del *tiempo de residencia residual*, se ha recurrido al siguiente artículo [8] donde se explica, para cada distribución posible del tiempo de residencia, como se deduce la expresión buscada. Se resumirá en la siguiente tabla 3.1, siendo $\mathcal{U}_{[0,1]}$ una variable aleatoria que sigue la distribución uniforme en el intervalo $[0, 1)$.

3.4.2.1. Verificación cálculo de tiempo residual

A continuación, comprobaremos que las funciones realizadas con la parametrización mencionada arrojan los resultados que se buscan. Se presentarán los resultados para la distribución Gamma, haciendo un proceso análogo para el resto de distribuciones.

En primer lugar generaremos números aleatorios usando nuestra función con tasa $0{,}1$ y coeficiente de variación 5, 3 y $0{,}25$. En el caso de la distribución residual Gamma, obviaremos la multiplicación por la distribución uniforme y miraremos que la generación de la Gamma se produce con los parámetros indicados en 3.1: $Gamma(\mu, n+1)$. Para esta distribución tendremos los siguientes valores de media y coeficiente de variación, a partir de los parámetros de entrada:

$$E[X] = \frac{\frac{1}{C_V^2} + 1}{\frac{1}{C_V^2}\mu^{-1}}$$

$$C_V = \frac{1}{\sqrt{\frac{1}{C_V^2} + 1}}$$

con μ la media deseada. El código de generación de estos valores y los resultados obtenidos son los siguientes:

```
import numpy as np
from Distributions import gamma, residual_gamma

rng = np.random.default_rng()
plt.figure()
l = [5, 3, 0.25]#, 1, 0.75, 0.5, 0.25, 0.0625]
k = 0.1
for cv in l:
    d = [gamma(rng, [k, cv]) for i in range(10000000)]
    print("Expected results ",1/k, cv)
    print("Function results ", "{:.2f}".format(np.mean(d)),
        "{:.2f}".format(np.std(d)/np.mean(d)))
    # Residual disitrbution
    a = 1/cv**2
    b = (a*k)
    d = [residual_gamma(rng, [k, cv]) for i in range(10000000)]
    print("Expected results ", "{:.2f}".format((a+1)/b),
        "{:.2f}".format(1/np.sqrt(a+1)))
    print("Function results ", "{:.2f}".format(np.mean(d)),
        "{:.2f}".format(np.std(d)/np.mean(d)))

############## Output ##############
## Expected results  10.0 5
## Function results  10.01 5.00
## Expected results  260.00 0.98
## Function results  259.91 0.98
## Expected results  10.0 3
## Function results  10.00 3.00
```

```
## Expected results  100.00 0.95
## Function results  100.01 0.95
## Expected results  10.0 0.25
## Function results  10.00 0.25
## Expected results  10.62 0.24
## Function results  10.63 0.24
```

Como vemos, todos los valores obtenidos con las funciones parametrizadas en relación a la distribución Gamma coinciden con los valores esperados. Podemos confirmar confiadamente que la generación del tiempo residual de las distintas distribuciones se realizará correctamente.

4. Parámetros y resultados del simulador

El simulador tendrá una serie de parámetros de entrada y una serie de variables de salida. Las entradas permiten la parametrización de los componentes del simulador, mientras que los resultados provienen del muestreo de las variables de salida (mayoritariamente el descriptor del sistema) del simulador.

4.1. El archivo de parámetros

Los parámetros necesarios para los distintos elementos del simulador se introducen a este a través de un archivo de configuración. Este archivo es el punto de interacción entre el simulador y el usuario. Se usará el formato JSON para guardar los nombres de las variables y sus valores. Contará con los siguientes parámetros:

- **(session | service)_distribution**: distribución que sigue el tiempo de sesión o de residencia (referido como de servicio).
- **R**: es un diccionario con claves las slices y como valor la matriz de encaminamiento, como definida anteriormente, correspondiente.
- **GAMMA**: tasa de llegadas externas por clase y por nodo. Es un diccionario con clave cada clase de servicio y valor una lista con las tasas de llegada a cada nodo.
- **MU**: tasa de residencia por clase. Es un diccionario con clave cada clase de servicio y valor la tasa correspondiente del tiempo de residencia. Para distribuciones distintas de la exponencial, este valor consistirá en una lista constituida por el valor de la tasa, que se hará corresponder con la esperanza de la distribución, y por otro lado el coeficiente de variación que caracterizará el comportamiento de la distribución.
- **S**: tasa de servicio por clase. Como para ”MU”, se trata de un diccionario con las mismas características, esta vez, caracterizando la distribución de servicio.
- **flag_(average_users | prob_distribution | prob_more | prob_blocking | prob_losses)**: permite activar o no el cálculo de los estadísticos respectivos.
- **N_i**: vector adicional en caso de querer calcular la probabilidad de que la ocupación en un nodo sea mayor o igual que un número predefinido (prob_more).

- **priority**: valor booleano que expresa si los traspasos tiene prioridad por delante de nuevas sesiones en caso de alta demanda.
- **SIMULATION_TIME**: tiempo total de simulación.
- **THRESHOLD_TIME**: tiempo umbral a partir del cual se empieza a muestrear.
- **REPL**: número de réplicas de la simulación a realizar.

Un ejemplo de fichero de configuración se puede encontrar en 3.5.

4.2. Resultados

Los resultados no se obtienen directamente del simulador, pero se derivan del estado de este. Se utilizará una clase aparte que llamaremos Sim.py, que por una parte se encargará de ir sacando los eventos de la lista de evento, y por otra parte muestreará los valores requeridos. El muestreo se realizará a partir de cierto tiempo umbral, parametrizado en la configuración, para evitar el transitorio del sistema.

Los resultados producidos tendrán los formatos siguientes, en función de los resutlados requeridos:

- **average_users**: diccionario que contiene para cada clase un arreglo indicando el número de usuarios medio para cada nodo de la red. Contendrá además el número de usuarios medio en global.
- **prob_distribution**: diccionario que contiene para cada clase un arreglo de arreglo indicando en los índices del primer arreglo el nodo de la red y en el segundo arreglo la distribución de probabilidad de los usuarios indicando en cada índice la probabilidad de que el número de usuarios para ese nodo sea igual al índice.
- **prob_more_n_i**: arreglo que indica para cada nodo la probabilidad de que el número de usuarios en ese nodo supere un umbral definido el el archivo de configuración.
- **probabilidad de bloqueo o de pérdida**: diccionario que contiene para cada clase y en total un arreglo con la probabilidad correspondiente por nodo.

4.2.1. Archivo de resultados

El simulador realiza una simulación. Para obtener resultados más precisos se realizan diversas réplicas, cuya ejecución será controlada por un script (simulate.py). Los resultados de cada réplica serán agregados respetando las estructuras de datos de cada uno. Además, usando el mismo formato que el fichero de entrada (configuración), los resultados se guardarán en el formato JSON. Junto con los resultados, se copiarán todos los parámetros de la simulación para añadir contexto a estos, para cuando eventualmente se requiera realizar cálculos teóricos.

4.2.2. Visualización de resultados

El formato de salida JSON es práctico para intercambiar datos entre programas informáticos, pero es difícil de interpretar a simple vista por un humano. Para facilitar la lectura de los resultados de

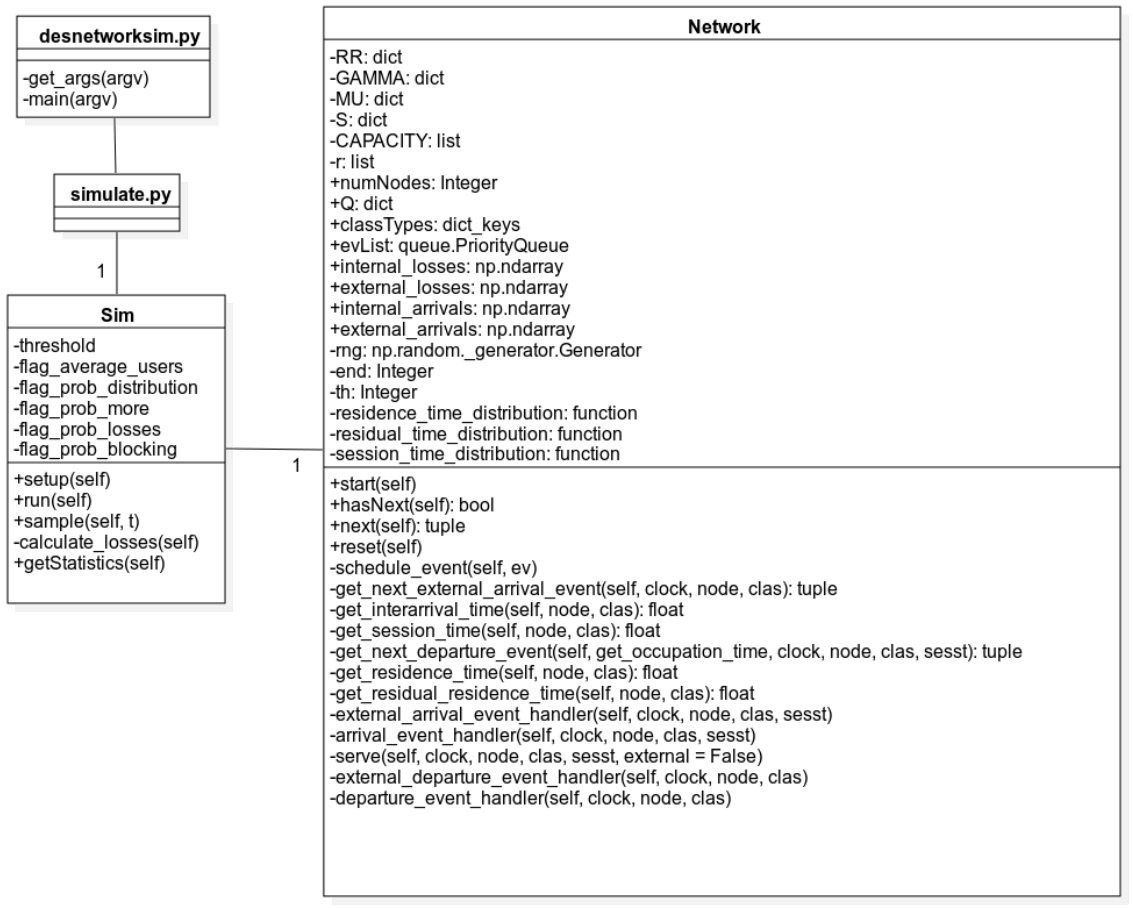


Figura 3.2: Diagrama de clases de la arquitectura completa del simulador

las simulaciones al usuario, se han desarrollado scripts de visualización de los resultados. Ejemplos de las gráficas producidas se podrán ver más adelante.

5. La arquitectura del software

Las los componentes del simulador se reagrupan en una misma clase descrita en el archivo *Network.py*. Este corresponde al modelo del simulador de la red de colas. Por encima, habrá otra clase, *Sim.py*, que se usará como controlador de la simulación: orquestará la ejecución de los eventos, a la vez que se encarga del muestreo de valores. A su vez, ejecutaremos varias réplicas de la simulación para tener una mejor estimación de los resultados, a través de un script Python *simulate.py*, que llamará tantas veces como réplicas se quieran a la clase Sim.py, encargada de realizar la simulación. La entrada de uno o varios ficheros de configuración se realiza en otro script python, *desnetworksim.py*, habilitado para su uso en línea de comandos.

El diagrama de clases de la figura 3.2 expresa la relación entre las distintas clases y scripts (denotado con la extensión ".py") de la solución desarrollada. El diagrama de secuencia 3.3 representa en detalle el flujo normal de ejecución de la simulación completa.

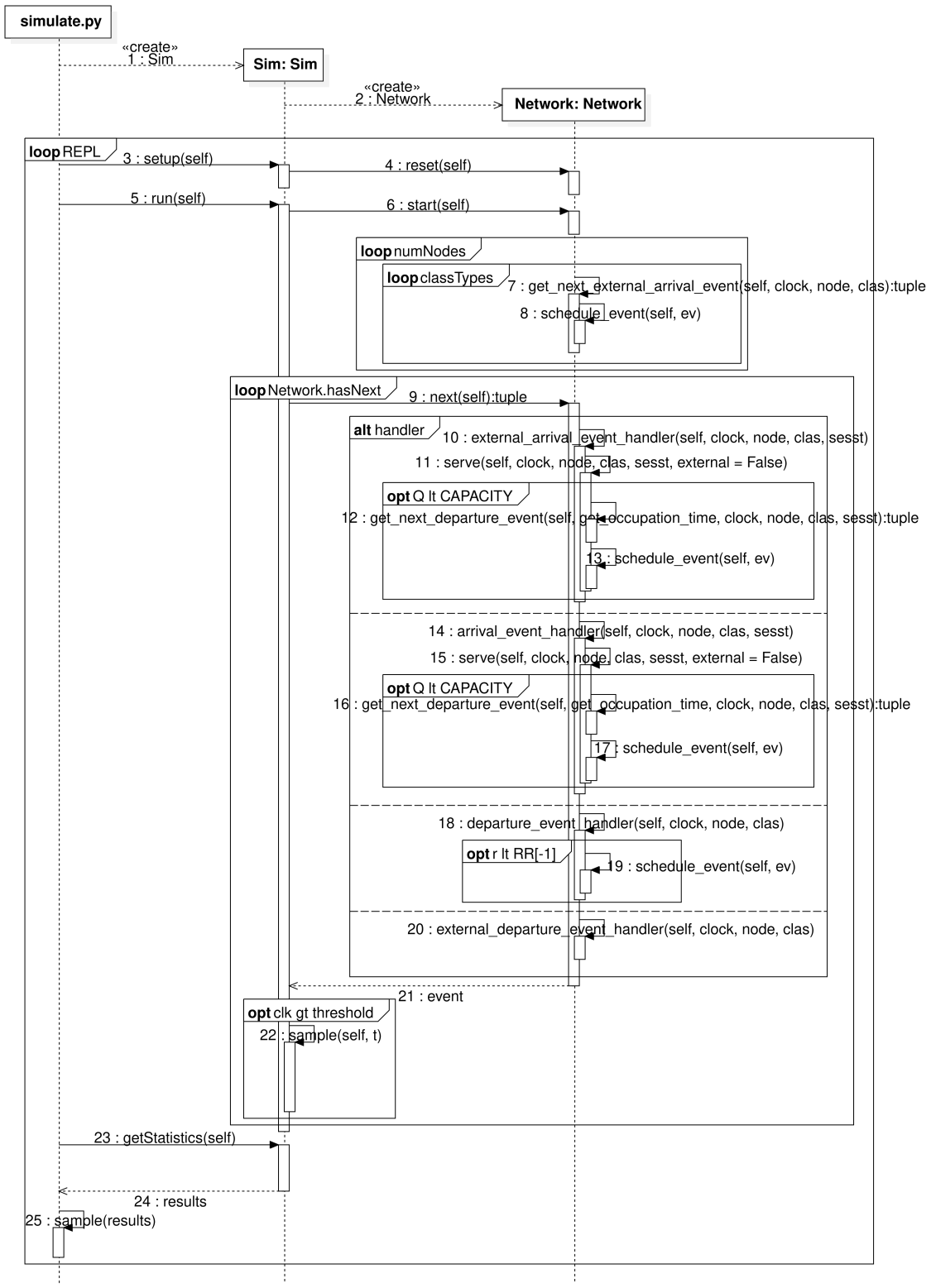


Figura 3.3: Diagrama de secuencia de una simulación con réplicas

6. Verificación del simulador

El simulador ha sido desarrollado a partir de un programa de simulación de una cola disociada completamente válido. Sin embargo, es pertinente realizar diversas pruebas en las que se conozca el resultado a priori para verificar el correcto funcionamiento del simulador. Algunas partes se han verificado individualmente (por ejemplo, la parametrización de las distribuciones). El objetivo ahora es verificar que una red de colas sea simulada correctamente. Para ello definiremos un escenario de verificación simple y calcularemos teóricamente los resultados esperados de los estadísticos, que finalmente compararemos con los obtenidos con la simulación, para evaluar la exactitud de nuestro simulador.

6.1. Escenario de verificación

El escenario deberá ser sencillo para poder calcular correctamente los resultados esperables de la simulación. En primer lugar, el escenario deberá contar tanto con tiempos de residencia como de entre llegadas y de sesión que sigan una distribución exponencial, ya que los resultados teóricos de las colas y redes de colas asumen esta condición. En segundo lugar, se ha escogido el número medio de usuarios en cada nodo como el estadístico a comparar, por su simplicidad. Si tanto el estadístico muestreado como los resultados teóricos coinciden, podremos estar seguros de que la simulación de la red es correcta.

El escenario a simular se puede visualizar en la figura 3.4. El fichero de configuración se puede ver en 3.5. La matriz de encaminamiento se ha escogido de manera arbitraria, con algún bucle de retroalimentación. Se disponen de dos slices (diferenciadas por color) para comprobar el buen funcionamiento del simulador con múltiples clases de usuarios. La capacidad del sistema será tanto infinita como finita. En este último caso, la cantidad de recursos por nodo será suficientemente elevada como para que no se produzcan pérdidas en los nodos, lo que no está contemplado en la teoría. Más adelante en el desarrollo del trabajo veremos como tratar con las pérdidas.

6.1.1. Resultados teóricos esperados

Al tratarse de una red de Jackson, definida en la sección 3.2, los resultados para la red global se podrán calcular en cada nodo independientemente. El haber varias slices dentro de la red no supone un problema adicional en los resultados, ya que al tratarse de flujos de Poisson, estos pueden agregarse fácilmente. En otras palabras, para encontrar el número de usuarios medios en un nodo, basta con calcular el número de usuarios medios por clase de usuario en cada nodo y sumarlos. En el caso de que la capacidad sea finita, usaremos la fórmula a usar serán 2.3.2. En cambio, para capacidad infinita, serán las fórmulas vistas en 2.3.1. Para la resolución de (2.8), hará falta un paso adicional. En el modelo de la ecuación, se asume $\sum_i r_{ji}^c \neq 1$, si no estaríamos ante una red cerrada, que para el sistema de ecuaciones definido, resultaría en un sistema indeterminado. Como se ha mencionado en apartado anteriores, hemos definido las matrices R^c (por slice) de tal manera que $\sum_i r_{ji} = 1$. En el simulador la salida del sistema está "controlado"por el tiempo de finalización de la sesión 3.1 que depende de la distribución que siga el tiempo de sesión. En un nodo, se acabará la sesión o no dependiendo de si se realiza un traspaso hacia un siguiente nodo o no, por lo que podemos calcular la probabilidad de que se produzca un traspaso de un usuario de clase c desde i

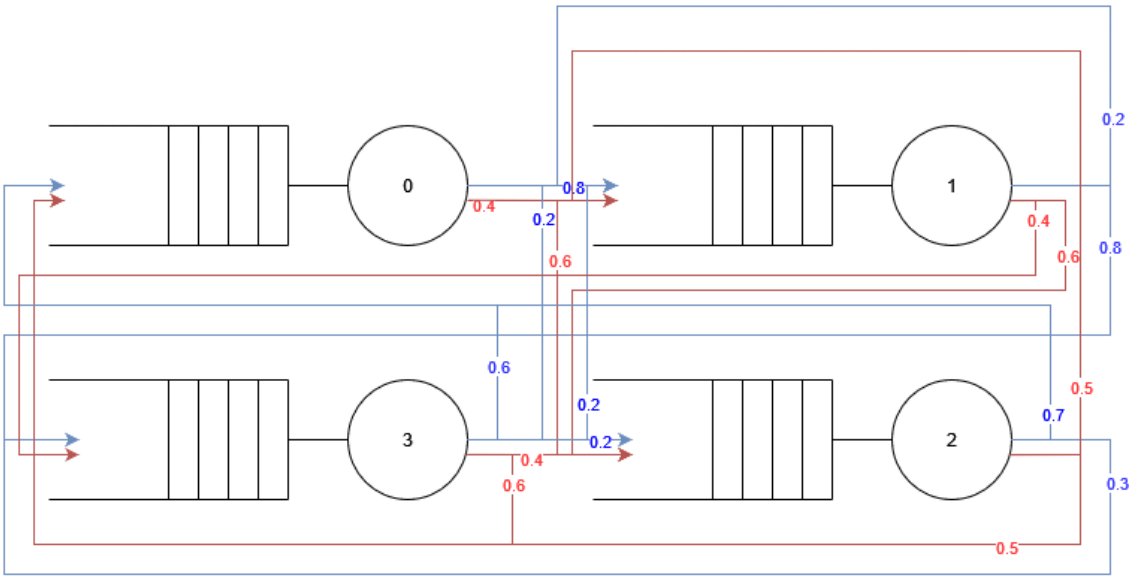


Figura 3.4: Diagrama de la red de verificación del simulador

a j como:

$$p_{ij}^c = \frac{\mu_r^c}{\mu_r^c + \mu_S^c} r_{ij}^c, \tag{3.8}$$

siendo r_{ij}^c el valor en la posición (i, j) de la matriz R^c. Esta nueva probabilidad corresponde a la que se asume en el sistema (2.8). Podemos reescribir el sistema como:

$$\lambda_i^c = \gamma_i^c + \sum_{j=1}^{M} \lambda_j^c p_{ji}^c \tag{3.9}$$

Una vez se tienen las tasas de entrada a cada nodo por clase, es necesario conocer la tasa de ocupación en los nodos y por clase. En un modelo de cola tradicional, se habla de tasa de servicio como la tasa a la que los usuarios son servidos dentro de un nodo, es decir, la inversa de el tiempo que un usuario pasa dentro del servidor. En nuestro caso, no se trata directamente de un tiempo de servicio, pero de un tiempo de ocupación de los recursos de nuestra red. Este tiempo corresponderá al mínimo entre una finalización de sesión o de la finalización de la residencia de un usuario en ese nodo. Por lo que la tasa de ocupación de un servidor será el mínimo entre estas dos tasas (asumiendo distribución exponencial), que notaremos:

$$\mu_{oc}^c = \mu_S^c + \mu_r^c \tag{3.10}$$

Tras estos dos ajustes, tendremos tanto las tasas de llegadas como las de “servicio”para el cálculo de los resultados teóricos.

6.1.2. Estimación del transitorio

Como se ha mencionado en el apartado teórico, los sistemas tienden a pasar por un estado transitorio antes de llegar al régimen permanente. Esto es fatal para el muestreo de estadísticos, puesto que

```
{
    "service_distribution" : "exponential",
    "session_distribution": "exponential",
    "flag_average_users" : true,
    "flag_prob_distribution" : true,
    "flag_prob_more" : true,
    "flag_prob_losses" : false,
    "flag_prob_blocking": false,
    "N_i": [20, 21, 10, 25],
    "R" : { "1": [  [0,0.8,0.2,0],
                    [0,0.2,0,0.8],
                    [0.7,0,0,0.3],
                    [0.6,0.2,0.2,0]],
            "2": [  [0,0.4,0.6,0],
                    [0,0,0.6,0.4],
                    [0.5,0.5,0,0],
                    [0.6,0.4,0,0]]},
    "GAMMA" : { "1": [10,10,10,10],
                "2": [10,10,10,10]},
    "CAPACITY": [240, 240, 240, 240],
    "MU" : {"1": 4,
            "2": 1},
    "S" : {"1": 1,
           "2": 1},
    "priority" : false,
    "SIMULATION_TIME" : 1000,
    "THRESHOLD_TIME" : 550,
    "REPL" : 6
}
```

Figura 3.5: Fichero de configuración del escenario de verificación

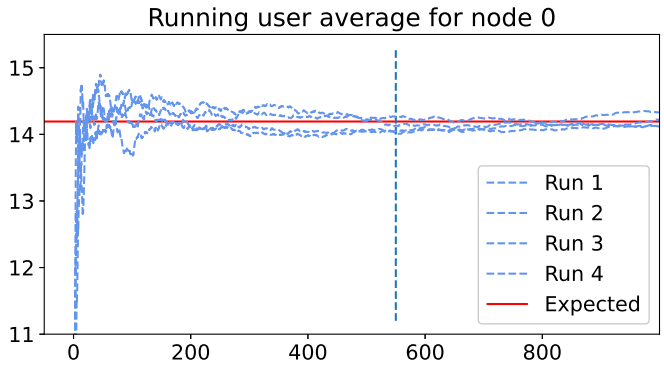


Figura 3.6: Evolución temporal del número de usuarios medio en el nodo 0

Intervalos de confianza de la simulación	[0.04853, 0.05476]	[0.1072, 0.1211]	[0.6137, 0.6270]	[0.0004991, 0.0009421]
Resultados modelo de colas	0.06034	0.1234	0.6353	0.001012

Tabla 3.2: Tabla de comparación probabilidad de que el número de usuarios supere cierto umbral

la no consideración de esta etapa podría disminuir la precisión de la estimación. Para el escenario de verificación, se ha aplicado un método para estimar la duración del transitorio, y eliminarlo del muestreo.

Este método consiste en primer lugar de realizar una simulación del escenario y muestrear la evolución temporal de un estadístico, en nuestro caso el número medio de usuarios en un nodo en particular. Este seguimiento se hará en repetidas ocasiones, y se promediará, para que la aleatoriedad no influya demasiado. Cuando se tengan los datos, solo hará falta visualizar el instante temporal en el que el estadístico empiece a converger.

Como vemos en la figura 3.6, el valor comienza a converger alrededor del instante temporal 550, por lo que para las siguientes simulaciones, pondremos este valor como umbral de muestreo para mejorar nuestros resultados.

6.1.3. Resultados de la simulación del escenario de verificación

En la gráfica 3.7 se representa sobre la misma figura el número de usuarios medio, tanto los resultados obtenidos en la simulación como los resultados calculados teóricamente.

Como se ve claramente, ambos valores se superponen, por lo que podemos confirmar la validez de nuestro simulador.

En la figura 3.8 y la tabla 3.2 se muestran el resto de resultados que se pueden obtener gracias al simulador y su similitud con los resultados teóricos. No se muestran los resultados de pérdidas ya que el escenario se ha diseñado teniendo en mente que no haya pérdidas. La figura 3.8 se representan la distribución del número de usuarios por clase en cada nodo. En la tabla 3.2 se muestra la probabilidad de que el número de usuarios en un nodo, agregando todas las slices, supere cierto umbral N_i definido en el fichero de configuración 3.5.

De estos resultados, vemos que se aproximan bastante bien, aunque en algunos casos, como en

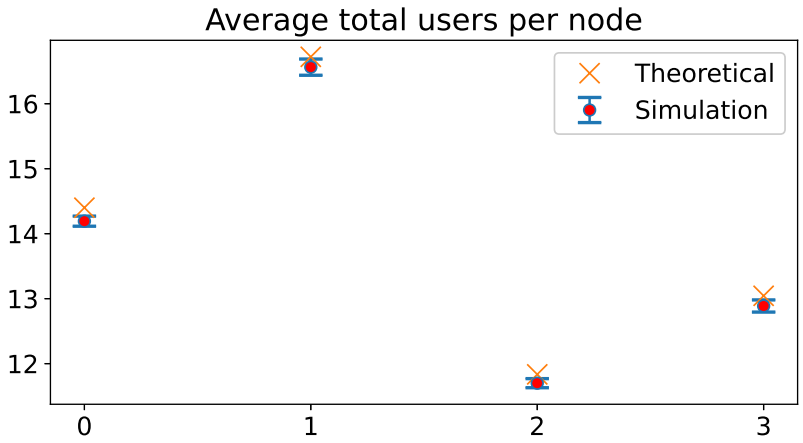
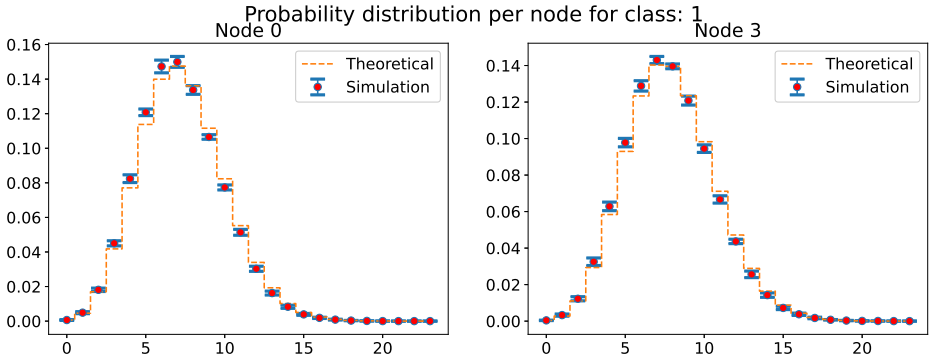
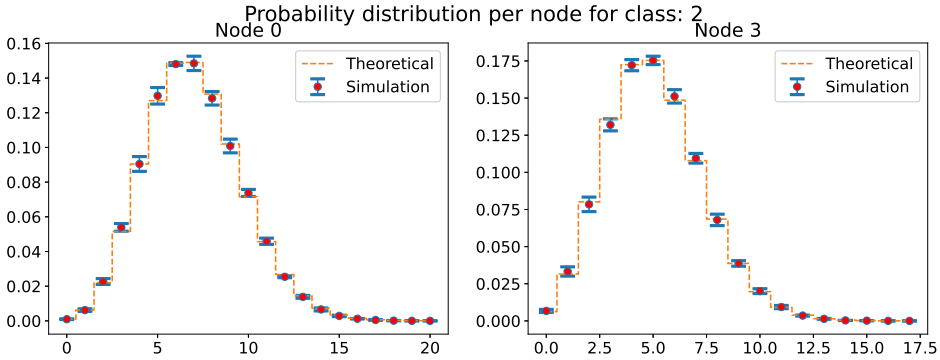


Figura 3.7: Número de usuarios medios por nodo escenario de verificación



(a) Clase "1"



(b) Clase "2"

Figura 3.8: Distribución del número de usuarios por clase en los nodos 0 y 3

la tabla 3.2, los resultados varían ligeramente de lo esperado. Aún así, validaremos el uso del simulador, puesto que los resultados en general se acercan a lo esperado.

En la siguiente sección nos aventuraremos en la simulación de un escenario más complejo, una vez que ya nos hemos asegurados de que los resultados que este produce son válidos.

Capítulo 4

Simulación

Una vez desarrollado el simulador, se usará para evaluar un método aproximado para el dimensionado de la red, mostrando así las capacidades de este.

El escenario que simularemos es una zona donde varios operadores virtuales concurren en distintas áreas de cobertura. Cada operador virtual corresponde a una slice. Se usará el modelo hexagonal de células de una red móvil. La red se compone de 42 nodos hexagonales, que cubren la región en forma de rectángulo, como se puede ver representado en 4.1. Esta red alberga 7 operadores virtuales. Un primer operador cubre la totalidad de los nodos (azul oscuro), 4 otros operadores que cubren cada uno una esquina de la región (en verde, naranja, rojo y morado), y otros 2 que corresponden al modelo de dos carreteras (cian). Para modelar el flujo de las carreteras, se pondrá en la práctica una slice por sentido de circulación en cada carretera, para un total de 4 slices para las 2 carreteras. Para conseguir una distribución uniforme de los usuarios sobre las zonas de cobertura, las transiciones en los bordes de las zonas serán cíclicas, es decir, una salida por un borde resultará en una llegada en el nodo que correspondería si repitiésemos la zona por los bordes. Por ejemplo, refiriéndonos al esquema de la red 4.1, si nos situásemos en la zona de cobertura roja y quisiéramos ir al nodo arriba a la izquierda del nodo 14, llegaríamos al nodo 38.

Las transiciones entre nodos serán equiprobables en todos los casos: para n vecinos adyacentes, la probabilidad de ir a cada uno de esas células será $\frac{1}{n}$. En caso de células modelizadas hexagonalmente, esta probabilidad será de $\frac{1}{6}$. Esta probabilidad corresponde al valor r_{ij}^c de las matrices de encaminamiento R^c. Para los cálculos teóricas usados en las aproximaciones, usaremos la probabilidad de transición total, que hemos definido anteriormente en (3.8). Los usuarios de cada slice se caracterizarán por su coeficiente de movilidad. Este coeficiente se definirá como el ratio entre la tasa de residencia y la tasa de sesión $c_{mob} = \mu_r^c/\mu_S^c$. De esta manera, habrá un número de traspasos diferente dependiendo de la slice, lo que en las zonas donde estas concurren, se producirán escenarios heterogéneos, donde habrá tanto usuarios de una slice que ocuparán durante casi toda la sesión los recursos de una misma célula, como usuarios que ocuparán durante un menor tiempo estos recursos pero los ocupará en un mayor número de células. Este escenario permite evaluar la red en un caso más dinámico. Se han escogido 3 valores para el coeficiente de movilidad en nuestro escenario: 0.1, 4 y 10. Las carreteras gozarán inherentemente del coeficiente de movilidad más alto. Asignaremos una movilidad de 4 a las zonas verde y morada. El resto tendrá un coeficiente de movilidad de 0.1. Además, fijaremos la tasa de sesión a 1 para todas las slices, a modo de referencia.

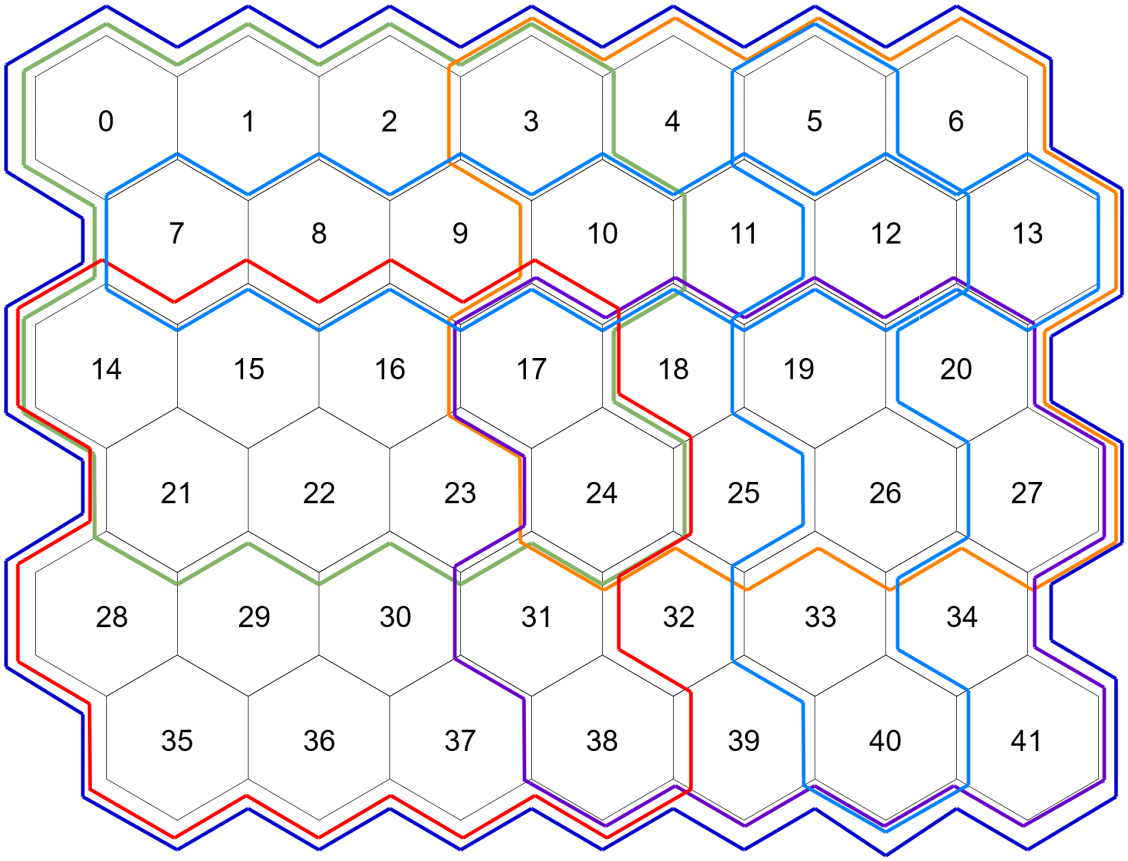


Figura 4.1: Diagrama de la red

1. Provisión de recursos

Una vez definida las características de la red, se usará un método aproximado para determinar la cantidad de recursos con los que cada nodo cuenta, teniendo en cuenta tanto las tasas de llegadas externas como los traspasos entre nodos y las pérdidas que ambos puedan experimentar. Nos apoyaremos en el marco teórico expuesto en la primera parte del trabajo para modelar un escenario aproximado de pérdidas, que nos permita dimensionar acorde con los objetivos de pérdidas la red, el cual será validado por la simulación. El modelado analítico para el dimensionado del sistema, y su posterior validación por el simulador, nos permitirá disponer de una aproximación matemática robusta para su aplicación en el control de admisiones de nuevas slices.

En el escenario planteado, se han propuesto dos casos: con o sin prioridad de los traspasos entre células. Un traspaso entre células corresponde a la continuación de una llamada en curso cuando la usuario móvil transita de la zona de cobertura de una célula a otra. En este caso se deja libre un recurso en el origen y se pasa a ocupar uno en destino. Si se priorizaran, una cierta cantidad de recursos serán compartidos con los usuarios que inician nuevas llamadas, y un número reducido serían reservados para los traspasos, de manera que, una llegada externa, a pesar de encontrar recursos libres, podría quedarse sin ser atendido. Esto reduce la probabilidad de que una llamada en curso sea bloqueada en detrimento de las nuevas llamadas, cuando la utilización de los recursos es elevada minimizando la probabilidad de acabamiento forzoso.

El dimensionado de la red se hará en función de un objetivo de pérdidas. Estas pérdidas se asumen al tener una cantidad de recursos finita.

En primer lugar, se calcularán las tasas de llegadas totales a cada nodo, teniendo en cuenta las pérdidas que puedan experimentar los flujos tanto de usuarios nuevos como de traspasos. En segundo lugar, aproximaremos el dimensionado de la red en función de las tasas calculadas con el procedimiento anterior. Y finalmente, ajustaremos el dimensionado de la red, que como veremos, necesita iterarse hasta encontrar la solución convergente.

1.1. Cálculo de las tasas

El flujo de llegadas a un nodo corresponden con la agregación de los usuarios que inician nuevas sesiones y de los usuarios que realizan un traspaso desde otro nodo de la red. La tasa a las que estas entran en cada nodo dependerá tanto de las tasas de ambos usuarios mencionados previamente, como de las pérdidas que estos experimenta, puesto que estas pérdidas disminuirán la tasa efectiva de entrada al nodo. Esto se resume en la figura 4.2. Para el cálculo de las tasas λ_i^c adaptaremos el sistema de ecuaciones utilizado en secciones anteriores (3.9), para que tenga en cuenta las pérdidas. Tendremos el siguiente sistema:

$$\lambda_i^c = (1 - p_n)\gamma_i^c + (1 - p_h) \sum_{j=1}^{M} \lambda_j^c p_{ij}^c \tag{4.1}$$

con p_h y p_n los vectores de pérdidas por nodo.

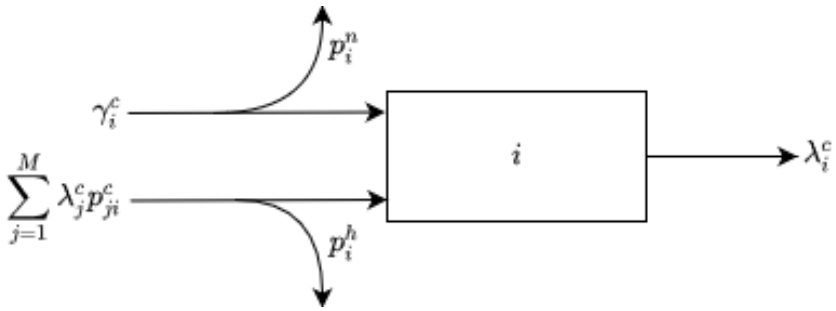


Figura 4.2: Diagrama ecuaciones de flujo por clase

1.2. Dimensionado de la red

En esta sección veremos como a partir de unas tasas dadas, calcularemos la cantidad de recursos necesarios en cada nodo de la red para no sobrepasar la probabilidad de pérdidas que se haya puesto como objetivo. El proceso de dimensionado se hará tanto para un escenario de prioridad de traspasos como sin prioridad.

1.2.1. Sin prioridad

Nos fijaremos un objetivo del 1 % de probabilidad de pérdidas. Al no haber prioridad entre flujos, tanto el flujo de llegadas externas como el flujo de llegadas internas, verán la misma probabilidad de pérdidas a su llegada: $p_n = p_h$. La probabilidad de pérdidas en un nodo se estimará usando la fórmula de pérdidas Erlang, o fórmula Erlang-B, $B(c, \lambda/\mu)$, definida recursivamente como:

$$B(0, \lambda/\mu) = 1, \quad B(c, \lambda/\mu) = \frac{(\lambda/\mu)B(c-1, \lambda/\mu)}{c + (\lambda/\mu)B(c-1, \lambda/\mu)} \tag{4.2}$$

Definimos a_i, la intensidad de tráfico total al nodo i, como la suma de las intensidades de tráfico por slice, que depende directamente de las tasas λ_i^c y μ_{oc}, definido en (3.10):

$$a_i = \sum_c \frac{\lambda_i^c}{\mu_{oc}^c} \tag{4.3}$$

Teniendo en cuenta estas definiciones, la cantidad de recursos c_i asignados al nodo i en un escenario sin prioridad será:

$$c_i = \text{mín}\{c, B(c, a_i) \leq 0{,}01\} \tag{4.4}$$

Es decir, el mínimo número de recursos necesarios para no sobrepasar una probabilidad de pérdidas del 1 %, fijada como objetivo.

1.2.2. Con prioridad

Nos fijaremos un objetivo del 1 % de probabilidad de pérdidas de nuevas sesiones. Al tener prioridad, el flujo de llegadas internas no verá la misma probabilidad de pérdidas que las llegadas externas, al tener estas últimas menos recursos disponibles. Las pérdidas internas se considerarán muy bajas, debido a la prioridad, con un valor estimado de 10^{-3}.

Este caso es un poco más complejo que el anterior, puesto que hay que tener en cuenta individualmente la tasa de llegadas de usuarios prioritarios. En un primer lugar, estimaremos el número de recursos necesarios para ambos flujos agregados de usuarios para llegar a una probabilidad de pérdidas inferior al 1 %. Este número lo notaremos c_{inf}. Esta vez, la cantidad de recursos necesarios para cumplir las pérdidas para ambos flujos c_{inf} (escenario análogo al anterior), se estimará usando una aproximación basada en la cola $M/M/\infty$ y no usando (4.2), ya que consideramos, como ocurre en una cola $M/M/\infty$, que pueden haber llegadas tras haber llegado a un estado de c servidores ocupados. Usaremos la fórmulas derivadas en 2.3.1. La cantidad de recursos c_{inf_i} por nodo se calculará tal que:

$$c_{\text{inf}_i} = \text{mín}\{c, P(N \geq c)_{M/M/\infty} \leq 0{,}01\} \tag{4.5}$$

Esta cantidad de recursos asegurará el objetivo de pérdidas para ambos flujos, pero hará falta ajustar los recursos reservados para los usuarios prioritarios, para cumplir su condición de prioridad: probabilidad de pérdidas despreciable. Por ende, en segundo lugar, determinaremos la cantidad de recursos reservados para estos usuarios. Definimos a_{h_i} la intensidad de tráfico al nodo i de usuarios prioritarios, independiente de la slice, como:

$$a_{h_i} = (1 - p_h) \sum_{j=1}^{M} \lambda_j^c p_{ij}^c / \mu_{oc} \tag{4.6}$$

Ponderaremos este valor para considerar el caso a partir del cuál el sistema se encuentra en c_{inf_i} usuarios. Notaremos este valor como $a_{h_i}^o = a_{h_i} * B(c_{\text{inf}_i}, a_{h_i})$, siendo $B(c, a)$ la fórmula Erlang-B (4.2). Aplicaremos este valor de las tasas prioritarias a la fórmula Erlang-B, siguiendo el mismo principio que en el caso sin prioridad, para obtener r_i, cantidad de recursos reservados por nodo, de tal manera que este flujo de usuarios experimente unas pérdidas muy bajas

$$r_i = \text{mín}\{c, B(c, a_{h_i}^o) \leq 0{,}001\}. \tag{4.7}$$

1.3. Ajuste de la capacidad

En las secciones anteriores hemos visto como calcular las tasas de llegada y como a partir de estas determinar el número de recursos necesarios para cumplir con una probabilidad de pérdidas objetivo. La probabilidad de pérdidas de la red dependerá de la cantidad de recursos que haya en cada nodo, por lo que esta no se podrá conocer hasta después del dimensionado. Pero hasta que no sepamos el valor de las pérdidas, no podremos determinar con exactitud los valores de las tasas (4.1). Sin las tasas no podremos dimensionar el sistema. Y así sucesivamente. Por lo tanto habrá que establecer un método para resolver esta paradoja.

Dimensionaremos la red a partir de una hipótesis de pérdidas nulas e iremos ajustando iterativamente el valor de la capacidad hasta que este sea el definitivo. El problema reside principalmente en la determinación de la probabilidad de pérdidas. Esta se usa para determinar las tasas, pero a su vez, la probabilidad de pérdidas depende de las tasas, puesto que a menor tasas de llegadas, menos pérdidas habrá en ese nodo en concreto. Por lo tanto, para una capacidad dada, usaremos el método del punto fijo para determinar el valor de las pérdidas, que nos permitirá determinar el valor de las tasas de llegadas. Este método determina iterativamente el valor buscado, comparando con la iteración anterior, para determinar la convergencia de este.

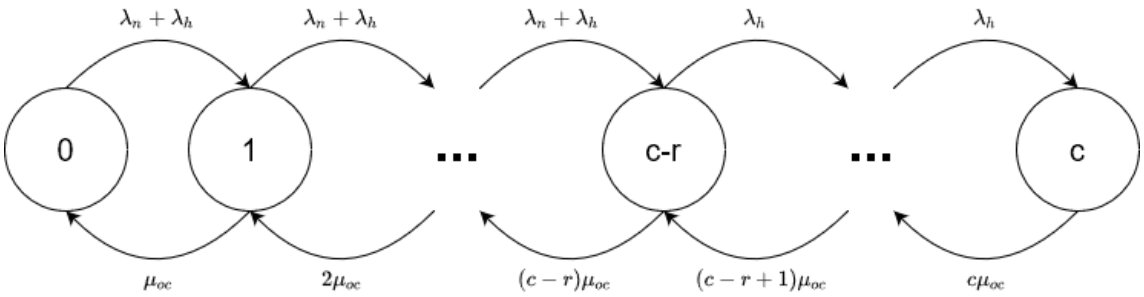


Figura 4.3: Cadena de Markov en un nodo con prioridad para los traspasos

Para determinar el valor de p_h y p_n, para una capacidad dada, primero determinaremos las tasas, usando (4.1) asumiendo la hipótesis de pérdidas nulas. Seguidamente, a partir de estas tasas y la capacidad, estimaremos el valor de las pérdidas. Una vez tengamos esta estimación de las pérdidas, compararemos con los valores anteriores, si no han variado mucho (en nuestro caso, se ha evaluado esto usando la siguiente comparación máx$_i$ $||p_{n_{old_i}} - p_{n_i}|| < 10^{-3}$), concluiremos el procedimiento. En cambio, si no se cumple este criterio de convergencia, se repetirá el proceso: estimaremos las tasas con (4.1), estimaremos las pérdidas en función de las nuevas tasas y de la capacidad dada, compararemos con la iteración usando el criterio de convergencia. Y así sucesivamente hasta encontrar los valores de p_h y p_n.

1.3.1. Estimación de la probabilidad de pérdidas

Para ajustar el valor, se necesita realizar una estimación del valor de las probabilidades de pérdidas tanto externas como internas para una capacidad y unas tasas dadas, que luego serán aplicadas al sistema (4.1) para obtener una solución más precisa sobre las tasas. De nuevo se usarían estas tasas para mejorar el resultado de las pérdidas, y así sucesivamente hasta llegar hasta los valores finales.

1.3.1.1. Sin prioridad

En el caso sin prioridad el cálculo de la probabilidad de pérdidas es de aplicación directa. Tanto las llegadas externas como los traspasos ven las mismas pérdidas, por lo que se recurrirá directamente a la fórmula Erlang-B para estimar la probabilidad de pérdidas, tal que:

$$p_{h_i} = p_{n_i} = B(c_i, \frac{\lambda_i^c}{\mu_{oc_i}^c}) \tag{4.8}$$

1.3.1.2. Con prioridad

Para este caso, recurriremos al modelo markoviano de colas para dar una estimación de ambas probabilidades de pérdidas. La cadena de Markov correspondiente a este proceso se puede visualizar en la figura 4.3, donde λ_n^c y λ_h^c corresponden respectivamente a las tasas de llegadas de usuarios nuevos y de traspasos. A partir de la ecuación (2.1), podemos calcular recursivamente las probabilidades de cada estado, y deducir las probabilidades de pérdidas tanto de las llegadas externas p_n

como de los traspasos p_h, tal que:

$$\begin{aligned} \pi_n =& \frac{(\lambda_n^c + \lambda_h^c)}{\mu_{oc}^c n}\pi_{n-1}, \;\; 1 < n \leq c - r \\ \pi_n =& \frac{\lambda_h^c}{\mu_{oc}^c n}\pi_{n-1}, \;\; c - r < n \leq c \\ p_h =& \pi_c \\ p_n =& \sum_{n=c-r}^{c} \pi_n \end{aligned} \tag{4.9}$$

En este punto, ya tenemos todas las herramientas necesarias para el correcto dimensionado de la red.

1.3.2. Algoritmo de ajuste

Inicialmente, consideraremos las probabilidades de pérdidas nulas: $p_n = p_h = 0$. Posteriormente, iniciaremos el proceso de iteración de punto fijo para obtener los valores p_n y p_h. En este punto tendremos los valores de las tasas de llegadas correspondientes al dimensionado inicial. Finalmente, se volverá a realizar el dimensionado de la red, teniendo en cuenta las nuevas tasas. Si no hay variación en los valores de la capacidad de la red, se guardarán esos valores. En cambio, si el dimensionado varía, se volverá a repetir el proceso, teniendo en cuenta que el número de recursos en la red ha cambiado, y que por lo tanto, la estimación de las probabilidades de pérdidas será distinta. El proceso de iteración se resume en el esquema 4.4.

2. Rendimiento de la red

Una vez se ha completado la provisión de recursos de la red, se pasará el modelo al simulador para evaluar las prestaciones de la red.

En un primer lugar, se ha usado la distribución exponencial para los tiempos de residencia y de sesión, para mantenerse lo más ceñidos posible a la teoría y que las aproximaciones sean lo más validas posibles. Más tarde, se han variado las distintas distribuciones.

Las simulaciones aquí descritas se han realizado usando el simulador en lenguaje Python en un ordenador con las siguientes características: procesador Intel Core i7-7700HQ @2.8GHz (4 núcleos/8 hilos) y 16GB de memoria RAM. Para completar las 7 réplicas que se han configurado, el programa ha tardado de media alrededor de 100 minutos.

2.1. Validación del escenario

En las figuras 4.5 y 4.6, vemos como por una parte, en términos absolutos, el dimensionado consigue bajar del objetivo de pérdidas, y por otra parte, vemos que la aproximación matemática utilizada se ajusta con muy poco error a los resultados de la simulación. De estos resultados se observan varias cosas. En un primer lugar, la aproximación matemática utilizada deja más margen para las

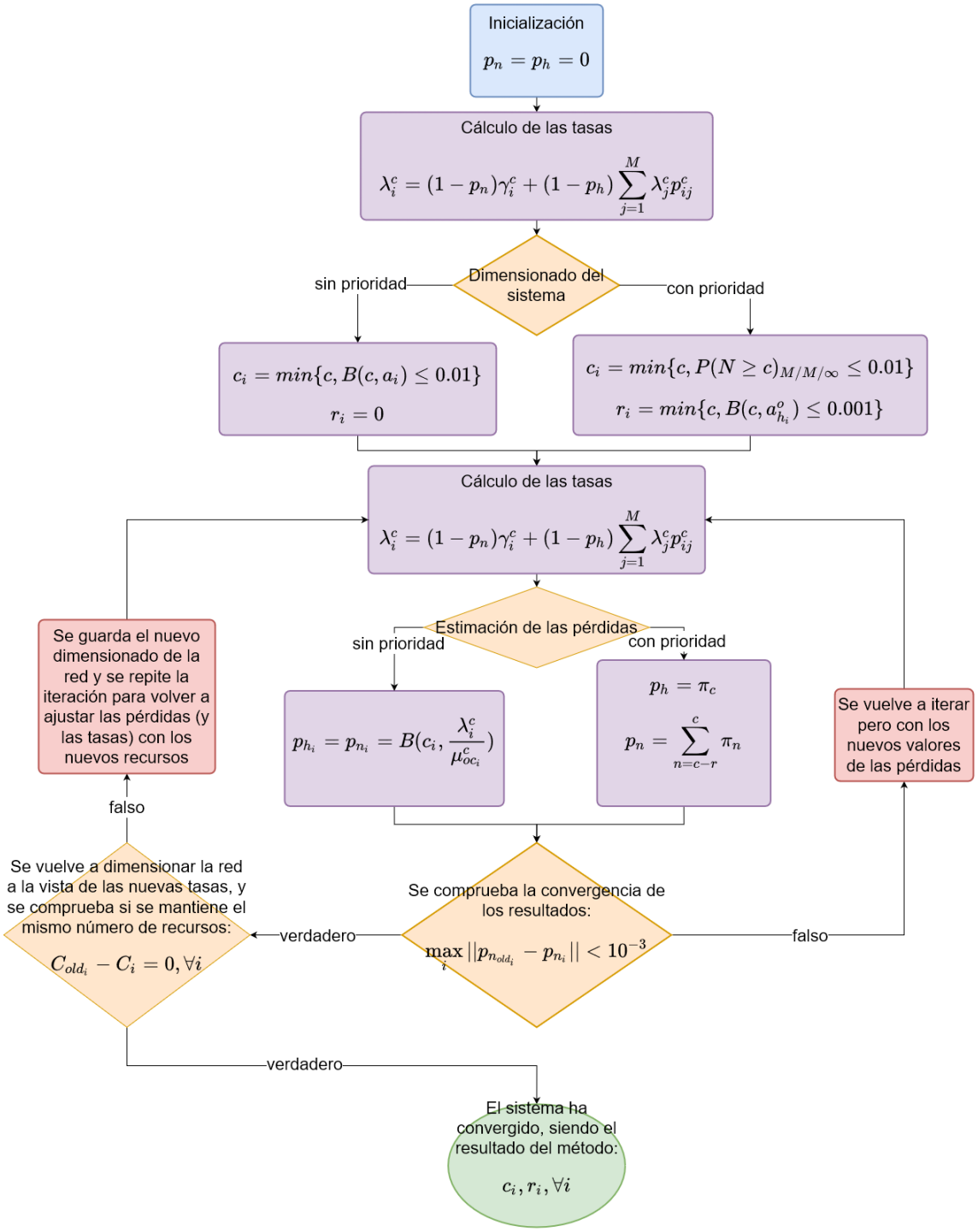
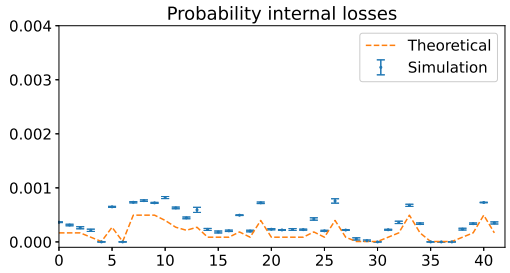
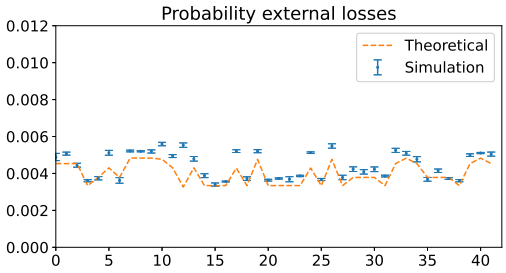


Figura 4.4: Diagrama del flujo de la iteración de punto fijo para obtener el dimensionado de la red

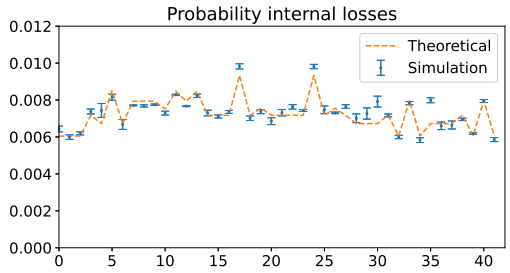


(a) Probabilidad de pérdidas internas por nodo

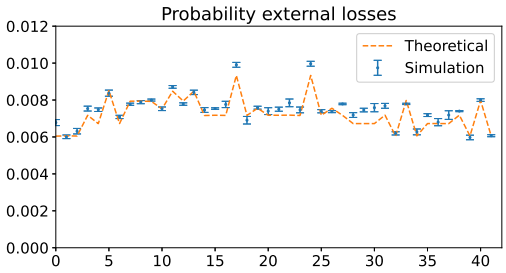


(b) Probabilidad de pérdidas externas por nodo

Figura 4.5: Resultado de la simulación con prioridad



(a) Probabilidad de pérdidas internas por nodo



(b) Probabilidad de pérdidas externas por nodo

Figura 4.6: Resultado de la simulación sin prioridad

pérdidas en el caso que haya prioridad, frente al caso opuesto. Como se puede ver, para un objetivo del 1 % de pérdidas externas, la aproximación realizada deja unas pérdidas efectivas entorno al 0.5 %, mientras que para el caso sin prioridad, los resultados están más próximos al 1 %, no bajando nunca del 0.6 %. Por otro lado, en la gráfica 4.5b vemos que el valor de pérdidas esperadas arrojados por nuestra fórmula analítica no se superpone tan bien sobre los resultados de la simulación, como si pasa en el caso contrario 4.6b. Esto probablemente se deba a que para realizar la aproximación del caso con prioridad, se han realizado más asunciones que en la aproximación para el caso sin prioridad, lo que inherentemente provoca que la precisión de esta baje. No obstante y pese a esto, la aproximación analítica realizada se puede considerar considerablemente precisa en su estimación.

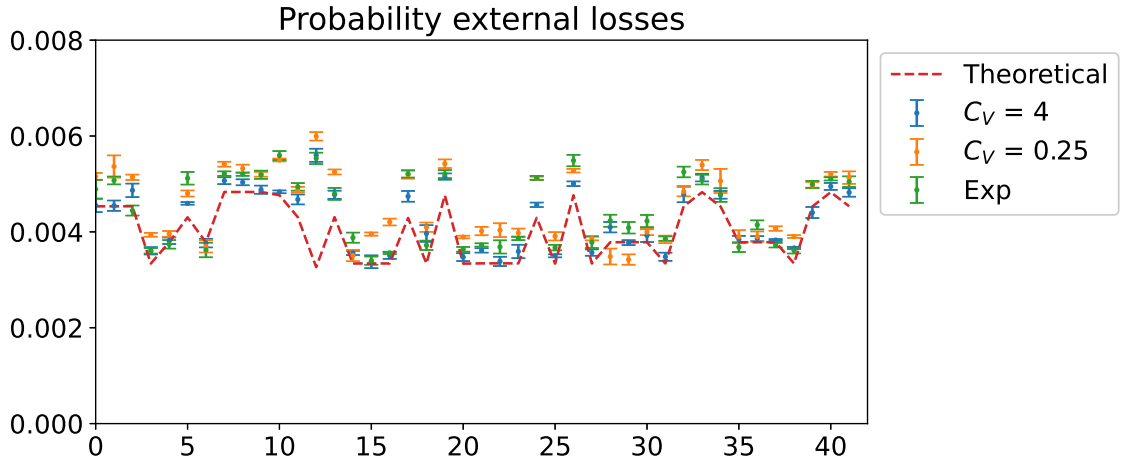
2.2. Relajación de las hipótesis

Variar las distribuciones de los distintos tiempo, nos permite ver el impacto que tiene considerar usuarios con comportamiento distinto, comparado con la distribución exponencial. Esto nos permite a su vez, evaluar la robustez de nuestras aproximaciones matemáticas frente a la relajación de las hipótesis de base del marco teórico.

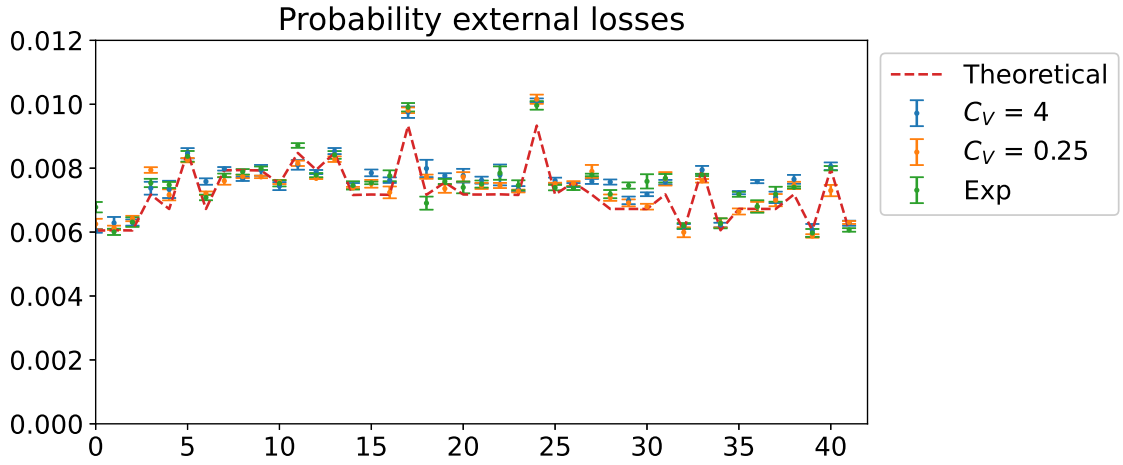
En esta sección, presentaremos los resultados de la variación del tipo de distribuciones y del coeficientes de variación, dentro de un mismo tipo de distribución manteniendo el mismo valor medio. En un primer lugar, variaremos la distribución del tiempo de residencia a la distribución lognormal, además de cambiar en distintas simulaciones el coeficiente de variación, adoptando los valores $\frac{1}{4}$ o 4, tanto en los caso con prioridad como sin. Posteriormente, haremos lo mismo pero variando la distribución del tiempo de sesión. Y finalmente, realizaremos este segundo conjunto de simulaciones, pero esta vez utilizando la distribución Gamma. Los resultados presentados serán únicamente los resultados de probabilidad de pérdidas externas. En caso de que no haya prioridad en los traspasos, tanto la probabilidad de pérdidas externas como internas es la misma, por lo que no hace falta presentar ambos resultados. En cambio, si hubiera prioridad, veríamos que la probabilidad de pérdidas en los traspasos es despreciable. Por lo tanto, en ambos casos visualizar presentar las pérdidas internas no presenta ninguna ventaja para su análisis. En las gráficas presentadas, se usarán los resultados vistos en 4.5 y 4.6 como referencia para evaluar las variaciones.

2.2.1. Tiempo de residencia o tiempo de sesión con distribución lognormal

A la vista de los resultados en las figuras 4.7 y 4.8, podemos afirmar dos cosas sobre la variación tanto del coeficiente de variación como de la distribución. Por una parte, el variar la distribución del tiempo de residencia no tiene mucho efecto sobre las pérdidas en ninguno de los casos, con o sin prioridad, como se aprecia en 4.7. En el caso 4.7a se podría especular alguna tendencia a la baja al aumentar el coeficiente de variación, pero no se cumpliría en todos los nodos, por lo que no sería muy prudente conjeturar nada para este caso. Por otra parte, y aquí se ve mucho más claro que en las anteriores gráficas, cambiar el coeficiente de variación y la distribución del tiempo de sesión incide directamente sobre las pérdidas de la red, como vemos en 4.8. Fijándonos en el coeficiente de variación en el caso sin prioridad 4.8b, vemos claramente como disminuye la probabilidad de pérdidas al aumentar el coeficiente de variación. Esta tendencia también se puede observar, aunque en menor medida, en el caso con prioridad.

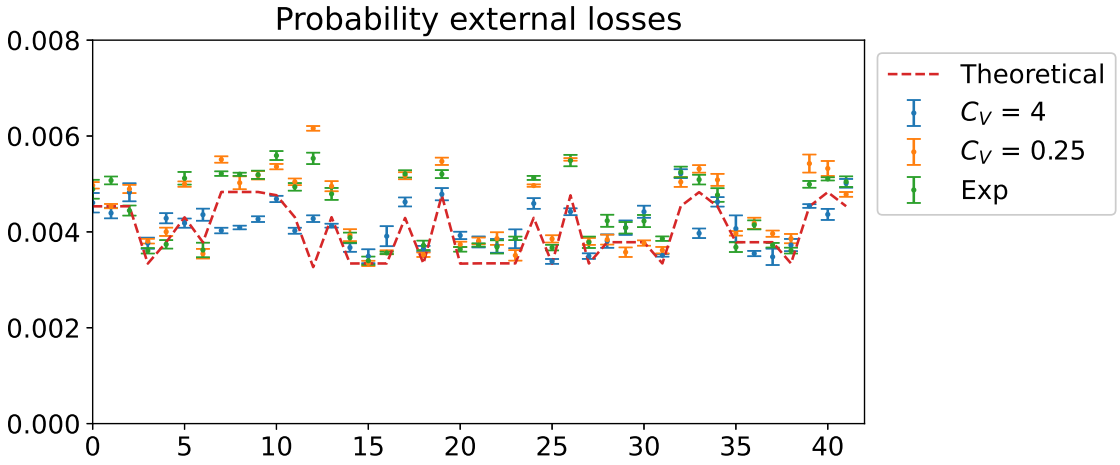


(a) Probabilidad de pérdidas externas por nodo con prioridad

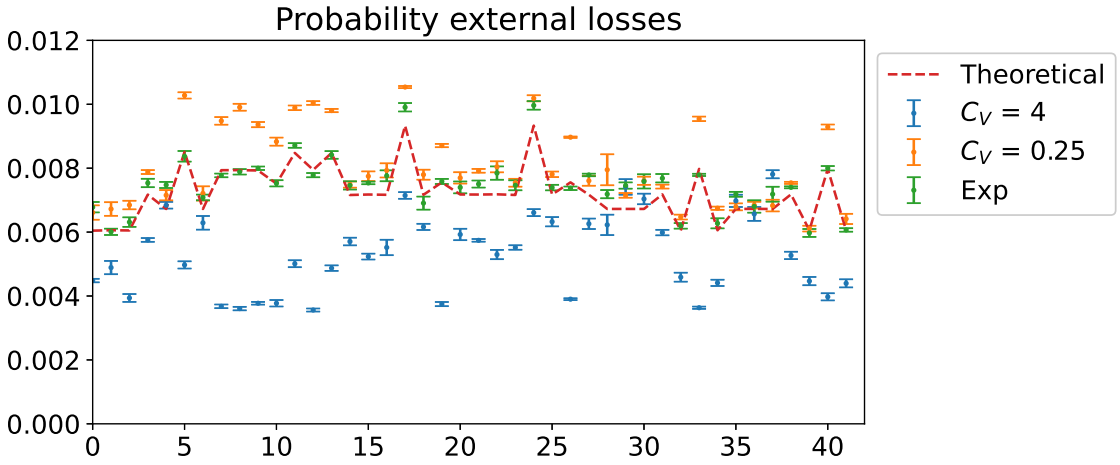


(b) Probabilidad de pérdidas externas por nodo sin prioridad

Figura 4.7: Resultado de la simulación con distribución del tiempo de residencia lognormal



(a) Probabilidad de pérdidas externas por nodo con prioridad



(b) Probabilidad de pérdidas externas por nodo sin prioridad

Figura 4.8: Resultado de la simulación con distribución del tiempo de sesión lognormal

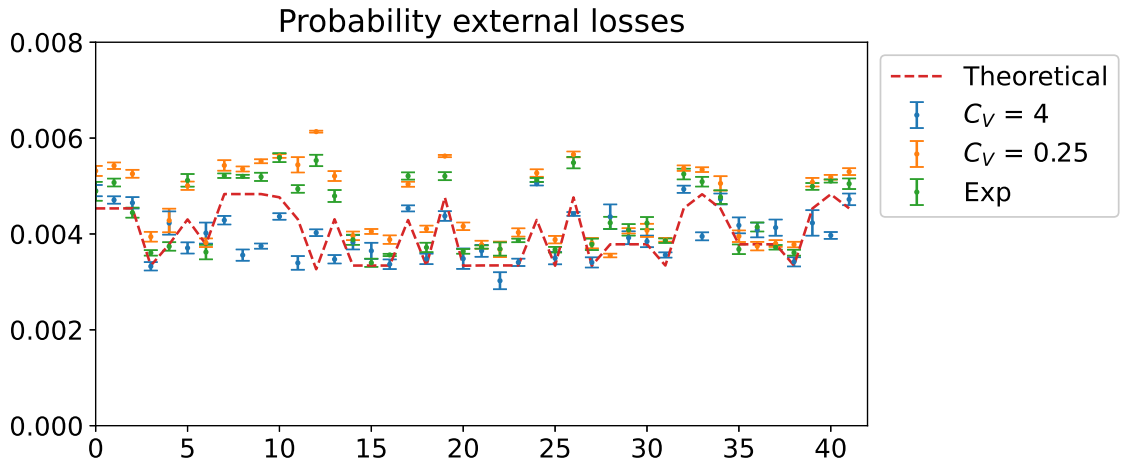


Figura 4.9: Probabilidad de pérdidas externas por nodo con prioridad variando ambos tiempos

2.2.2. Tiempo de residencia y tiempo de sesión con distribución lognormal

Para explorar más en detalle este escenario, miraremos que pasa si ambas distribuciones varían acordemente, es decir, basándonos en los casos anteriores, tanto la distribución del tiempo de residencia como del tiempo de sesión, serán lognormal, a la que haremos variar de la misma manera el coeficiente de variación.

A la vista de las figuras 4.9 y 4.10, vemos que las tendencias comentadas en el párrafo anterior se mantienen, pero cuantitativamente no se aprecian diferencias notables que lleven a pensar que variar ambas distribuciones a la vez tenga un efecto distinto al ya observado. Lo que realmente hace variar las pérdidas en la red es la variación de la distribución del tiempo de sesión, y a su vez del coeficiente de variación.

2.2.3. Tiempo de residencia distribución gamma

Finalmente, mostraremos los resultados obtenidos al poner la distribución Gamma como distribución del tiempo de sesión. Variaremos de nuevo el coeficiente de variación.

Los resultados 4.11 y 4.12 nos ayudan a corroborar lo concluido anteriormente. Como pasa con la distribución lognormal, en este caso podemos ver como a mayor coeficiente de variación, menores pérdidas se experimentan en la red.

A la vista de estos resultados, apreciamos que a pesar de la variación del valor de las pérdidas al cambiar el coeficiente de variación en la distribución de los tiempos, principalmente del tiempo de sesión; en términos absolutos la probabilidad de pérdidas se mantiene por debajo del 1 % (aunque en algún caso se pueda sobrepasar este valor ligera y puntualmente). La aproximación matemática utilizada para el dimensionado de la red es robusta frente a cambios en las características de la red y del servicio.

El análisis del escenario propuesto, nos ha permitido por una parte validar el funcionamiento del simulador en un escenario complejo, pero también de evaluar favorablemente una primera aproxi-

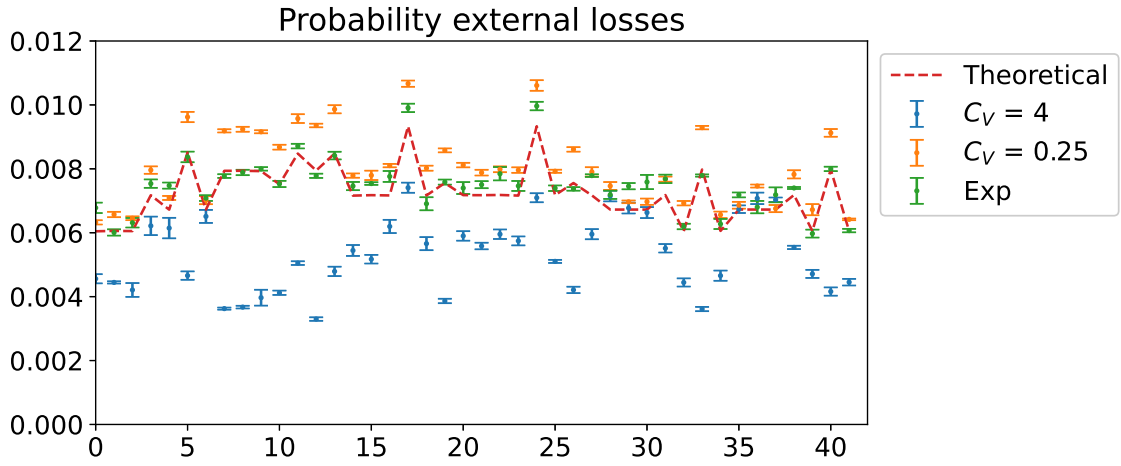


Figura 4.10: Probabilidad de pérdidas externas por nodo sin prioridad variando ambos tiempos

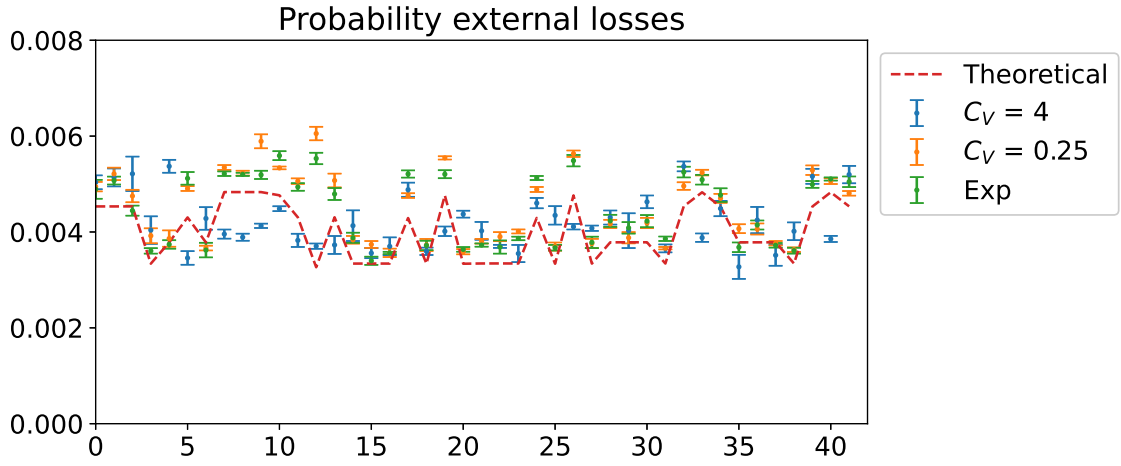


Figura 4.11: Probabilidad de pérdidas externas por nodo con prioridad distribución gamma

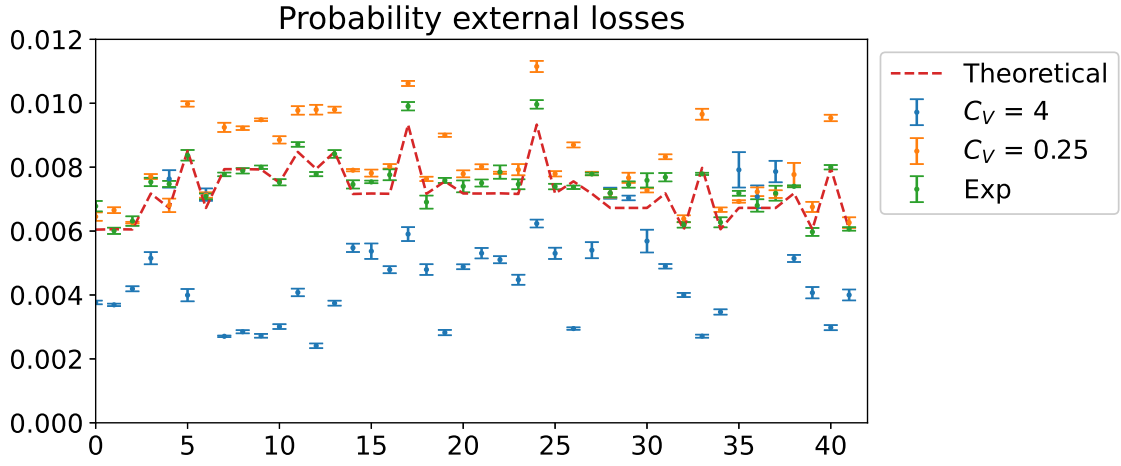


Figura 4.12: Probabilidad de pérdidas externas por nodo sin prioridad distribución gamma

mación al dimensionado de la red, cuyo análisis se ha llevado más allá, variando las características de las distribuciones, para evaluar la robustez de la aproximación, lo que permite su uso para una estimación en el control de admisiones de nuevas slices.

Capítulo 5

Conclusión

A lo largo de esta memoria, hemos visto cómo se ha desarrollado el simulador de redes móviles que implementan network slicing y cómo ha sido validado. Posteriormente, se ha planteado un escenario que ha sido probado en el simulador. El dimensionado del escenario ha sido realizado haciendo un estudio analítico previo y ciertas aproximaciones, para a continuación ser validado por los resultados del simulador. Además, se ha probado el mismo escenario variando las distribuciones tanto del tiempo de residencia como del tiempo de sesión, viendo cómo esto afecta al rendimiento de la red. La variación de la distribución del tiempo de sesión y del coeficiente de variación tiene un efecto sobre la probabilidad de pérdidas: a mayor coeficiente de variación, menores son las pérdidas. Aun con estas diferencias en la probabilidad de pérdidas con la variación de las hipótesis, se ha visto cómo la aproximación analítica al dimensionado se mantiene válida, llegando en todos los casos a los valores objetivos de pérdidas. Podemos concluir que los objetivos planteados en el trabajo han sido completado con éxito.

El desarrollo del trabajo me ha permitido una aplicación directa de las asignaturas vistas en el grado, como la asignatura de Redes Telemáticas o asignaturas relacionadas con la probabilidad y procesos estocásticos; de profundizar en los temas ya vistos en las asignaturas, como lo son las redes de colas, que derivan, de cierta manera, de la teoría de colas; y finalmente de ampliar los conocimientos en temas relacionados con la Ingeniería de Telecomunicaciones, como el network slicing o la gestión de recursos de una red. Además, he podido ver en detalle la simulación por eventos discretos y sus entrañas, como la generación de números aleatorios o la generación de réplicas de la simulación, que añaden una nueva herramienta a mi bagaje técnico. Por último, la programación del simulador ha sido clave para afianzar mis habilidades de programación en Python, lenguaje de programación utilizado cada vez más en numerosas aplicaciones.

Aunque el trabajo se ha desarrollado con éxito, este no queda exento de limitaciones, por la complejidad de alguna solución o por el número de horas asignadas al TFG. Presentaremos brevemente las limitaciones en el trabajo y cómo estas pueden servir de base para un ampliación del trabajo.

En primer lugar, los servicios dentro de la red, en todas las slices, han sido considerados siempre del mismo tipo, tratándose de llamadas (o vídeollamadas), por simplicidad. Tener un mismo tipo de servicio implica una necesidad de recursos idéntica para todos los servicios, igual a la unidad en nuestro caso. Esto presenta un escenario relativamente homogéneo en la red. En un escenario más realista, puede haber distintos tipos de usuarios con distintas necesidades de recursos para cada uno. Esto se conoce en inglés como servicios “multi-rate”. Por consiguiente, la ampliación sugerida

en este escenario sería dar compatibilidad a usuarios con distintas necesidades de ocupación de recursos, lo que implicaría una ocupación de varias unidades de recursos en su servicio (a diferencia de solo una).

En segundo lugar, también relacionado con el tipo de servicio, en los escenarios simulados se ha tenido en cuenta solamente tráfico noelástico, cuya duración de sesión no depende del número de recursos disponibles. La propuesta de mejora en este apartado es la consideración de tráfico elástico dentro de la red en los escenarios de simulación propuestos. El tráfico elástico tiene un tiempo de sesión variable en función del ancho de banda disponible. Considerando las llamadas y vídeo-llamadas, en general tienen una duración de igual características, independientemente del ancho de banda disponible, a pesar de que la calidad de la llamada pueda bajar durante su transcurso. Las llamadas sería un tráfico de tipo noelástico. En el otro lado encontraríamos servicio como la transferencia de un archivo a través de la red, cuyo tiempo de descarga es variable en función del ancho de banda disponible: cuantos más recursos en la red están disponibles, antes finalizará la descarga. Por consiguiente, el tiempo de la sesión va en función de los recursos disponibles en cada momento. El tráfico noelástico se podría definir a partir del tiempo medio que dura una sesión, mientras que el tráfico elástico se podría definir a partir del número de bits que quieran ser transmitidos.

Finalmente, como se ha mencionado en la introducción, el no respeto del uso acordado de los recursos presenta un problema. En nuestros escenarios, se ha asumido el respeto de este, por medio de la caracterización de las tasas de llegadas y de uso de los recursos por parte de los usuarios de la red, que en ningún momento aumentan. Para proteger la red frente a las violaciones de los acuerdos de uso, se podría poner en marcha un mecanismo de monitorización del uso declarado de las diferentes slices de la red y de control, a la vez que se podría caracterizar cierta tendencia de alguna slice a aumentar su tráfico. Esto podría ser un complemento al simulador.

Teniendo en cuenta estas limitaciones, considero que el desarrollo del trabajo ha sido completo y responde a la problemática planteada en el trabajo.

Referencias

[1] William J. Stewart. *Probability, Markov chains, queues and simulation*. Princeton University Press, 2009. ISBN: 0691140626.

[2] Yahya E. Osais. *Computer Simulation. A foundational approach using Python*. CRC Press, 2018. ISBN: 9781498726825.

[3] H.T. Papadopolous; C. Heavey; J. Browne. "Queueing Theory in Manufacturing Systems Analysis and Design". En: Springer, 1993. ISBN: 9780412387203.

[4] Python Software Foundation. *Python 3.8.5 documentation*. URL: `https://docs.python.org/release/3.8.5/`. (accessed: 06.06.2022).

[5] *Visual Studio Code*. URL: `https://code.visualstudio.com/`. (accessed: 06.06.2022).

[6] *Jupyter Project*. URL: `https://jupyter.org/`. (accessed: 06.06.2022).

[7] *Anaconda*. URL: `https://www.anaconda.com/`. (accessed: 06.06.2022).

[8] Hui-Nien Hung, Pei-Chun Lee y Yi-Bing Lin. "Random number generation for excess life of mobile user residence time". En: *IEEE Transactions on Vehicular Technology* 55.3 (2006), págs. 1045-1050. DOI: `10.1109/TVT.2006.874578`.