E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

# Generación de tráfico en sistemas de compartición de ficheros basado en el modelado estocástico de la actividad de los usuarios

Máster Universitario en
Ingeniería de Telecomunicación

Trabajo Fin de Máster

Néstor Eduardo Quezada Castillo
Daniel Morató Osés
Pamplona, 30 de junio de 2020

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

# Índice

# Resumen

Se presenta una herramienta de generación de trazas sintéticas al nivel de protocolo de compartición de ficheros. Esta trata de abstraer el comportamiento del usuario a partir de una traza con datos reales. Esta herramienta se concibe con la finalidad de complementar un programa de detección de Ransomware, disponiendo de un mayor número de muestras de tráfico sin malware con que entrenarlo.

El diseño parte de la caracterización del comportamiento del usuario en base a cuántos ficheros abre (Nº accesos), cuánta información se transfiere (nº bytes w/r) y en qué instante de tiempo los abre (timestamp). Así mismo, se plantearán una serie de parámetros, los cuales serán ajustables, y se cambiarán en la fase de calibrado, buscando el mínimo error entre traza real y sintética. Finalmente, se valida cuál de las diferentes configuraciones se adapta en mayor medida al tráfico real y así poder extraer unas métricas de su funcionamiento.

# Abstract

This work presents a brand-new traffic generator based on a finite state machine (FSM) at the sharing protocol level. The objective of this study is to extract de behavior of the users from a trace that contains real users' data. This tool was conceived in order to complement a Ransomware detection program, creating a larger set of traces that emulates the legit behaviors of real users.

This design starts from the characterization of the users based on the number of files accessed, the number of bytes transferred and the time of occurrence. In addition, a series of parameters will be raised and adjusted in the calibration phase in order to minimize the error between the real and synthetic trace. Finally, a configuration will be chosen in the validation phase and some metrics of the tool will be extracted.


***Keywords***— generador trazas, finite state machine, workload modelling.

# 1 Introducción

El presente proyecto se enmarca dentro de los escenarios de redes de compartición de ficheros de área local. Estos escenarios suelen presentarse en empresas donde es algo común el utilizar estos sistemas NAS (Network Attached Storage), para compartir ficheros entre los trabajadores o los diferentes departamentos. Este tipo de soluciones representan una alterantiva barata, fiable y fácil de configurar. Normalmente estos sistemas corren sobre NFS (Network File System) o SMB (Server Message Block). No son las únicas opciones, pero estos están ligados a los sistemas operativos más utilizados en el ámbito empresarial que son Linux y Windows respectivamente. En la Figura 1 se representa un escenario de esta tipo donde los clientes accederán a sus ficheros a traves de la red.

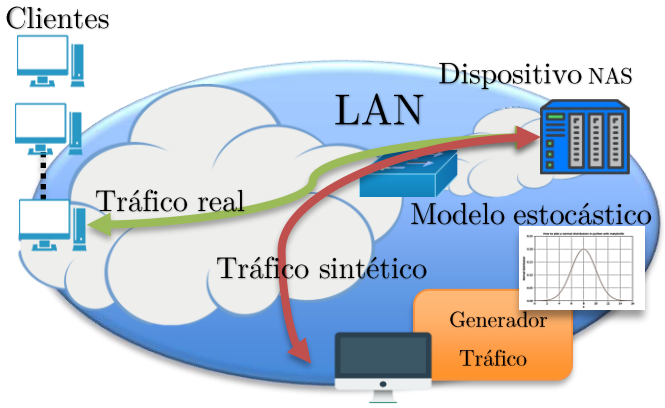


**Figura 1**. Escenario de red de área local con NAS y generador de tráfico.

Los generadores de tráfico sintéticos son herramientas que se usan en el campo de los sistemas de almacenamiento y ficheros sobre todo para evaluar el rendimiento de los diferentes elementos software y hardware [1], por ejemplo, en un escenario NAS. En la Figura 1 vemos un ejemplo de topología de red incluyendo esta herramienta. Existen diferentes clases de generadores según la estrategia de diseño y su complejidad. Una clasificación típica encontrada en la literatura son micro-benchmarks, macro-benchmarks, reproductores de trazas y generadores sintéticos.

Los micro-benchmarks son los más sencillos. Éstos solo simulan una funcionalidad concreta de todo el sistema [2]. Por otro lado tenemos los macro-benchmarks, que incrementan la complejidad con el fin de evaluar el comportamiento de los sistemas de ficheros trabajando a nivel de aplicación [3]. Las cargas de trabajo que simulan estos, aunque tratan de asemejarse a una carga representativa de entornos reales, normalmente no lo consiguen ya que cada escenario tiene sus peculiaridades que pueden diferir en gran medida con estas.

El siguiente es el reproductor de trazas, en este caso es necesario disponer de una traza representativa del entorno que se quiere estudiar. La calidad de los resultados va a depender sobre todo de la veracidad de esta traza y de la herramienta usada para reproducirla, algo que no es una tarea trivial [4]. Esta solución, aunque en principio parece la mejor opción para crear tráfico sintético, al recoger todas las dinámicas del tráfico, presenta unos factores en contra, como el ser una estrategia rígida, al simular directamente las acciones que hay en los logs o traza y no poder crear patrones alternativos.

La última clase es la de los generadores sintéticos, esta solución crea un modelo estocástico tomando como datos de entrada las trazas de usuarios reales. Este modelado es muy flexible, ya que como se ha comprobado en la bibliografía, no existe una metodología general aplicable. Cada diseñador puede centrar el modelo en las características deseadas y hacer las simplificaciones que se considere oportunas [5]. También tiene la ventaja de que se trata de un tipo de generador parametrizable a muy diferentes niveles, con lo cual aparte de generar un tráfico similar al esperado, se pueden crear patrones de comportamiento de los diferentes usuarios, es decir se puede generar poblaciones de usuarios diferentes en base a esa traza inicial [6].

En nuestro caso el objetivo es desarrollar una herramienta software de generación de tráfico a nivel del protocolo de compartición de ficheros en red (ej. SMB o NFS). Teniendo en cuenta la finalidad de nuestro estudio, la tipología que mejor se adapta es la de generadores sintéticos. La intención es usar este tráfico sintético para entrenar un detector de Ransomware. Se trata de poder aumentar las muestras de trazas que reflejan la actividad normal de un trabajador para así poder distinguirlas frente a unas trazas que recogen la actividad de un malware. La finalidad de este desarrollo difiere respecto de los encontrados en la bibliografía, lo que hará que éste se centre más en ciertas propiedades que reflejen el comportamiento de los usuarios.

El trabajo se estructura de la siguiente manera: En la sección 2 se explica el origen y características de la traza que contiene los datos de usuarios reales, base del modelo estocástico. A continuación, en la sección 3 se describe la metodología, simplificaciones, así como los parámetros elegidos en el modelado de la carga de trabajo. En la sección 4 se realiza la validación del modelo extraído, comparando las configuraciones para los mejores casos. Así mismo se elegirá una configuración final de la herramienta. Finalmente, se extraen unas métricas de funcionamiento del programa de generación para estimar la capacidad de almacenamiento necesario para su ejecución. Se termina en la sección 5 con las conclusiones extraídas a partir de la realización del trabajo.

# 2 Estudio traza SMB de usuarios

La traza empleada en este estudio, como fuente de descripción del comportamiento de usuarios, se recogió mediante el método de "port mirroring" aplicado al switch donde estaba conectado el interfaz de una cabina de discos situada en la Universidad Pública de Navarra (ver Figura 2).

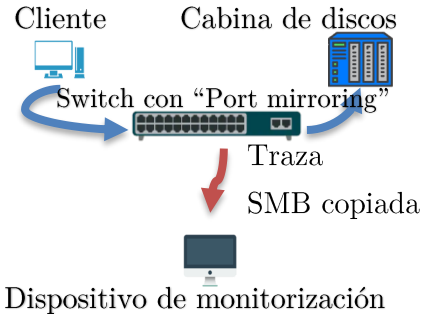


**Figura 2**. Escenario de captura de la traza SMB.

De esta traza se extraen un total de 489 conexiones, de las cuales solo 389 superan la hora de actividad. Este es el tiempo, a partir del cual, se ha considerado que las muestras son representativas, bajo el supuesto de que cada conexión está asociada a un usuario o trabajador diferente.

En la siguiente sección se tratará de caracterizar estas conexiones a nivel de duración y número de ficheros a los que acceden, así mismo se hará un estudio visual a nivel más granular de las series temporales de estas conexiones para así comprobar si existen algún tipo de patrón de comportamiento de los usuarios a lo largo del tiempo.

## 2.1 Análisis estadístico del tráfico SMB de usuarios ofimáticos

### 2.1.1 Duración de las conexiones

En la Figura 3 se representa la distribución acumulada de probabilidad para las duraciones de las 389 conexiones. El rango de duración se mueve entre 1 hora y 15 horas. Vemos que hay tres intervalos donde la gráfica se comporta de manera similar estos son [1,6], (6,8], (8,15].

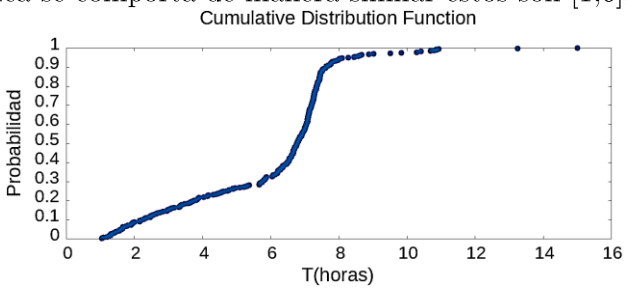


**Figura 3**. CDF para la duración de las conexiones.

Si extraemos los porcentajes (probabilidad empírica) de casos que caen dentro de cada uno de estos intervalos tenemos que en el rango de [1,6] horas se tiene un 32,47 %, en el rango de (6,8] horas se tiene un 60,82% y en el rango de (8,15] horas un 6.7 %. El intervalo con mayor porcentaje refleja la duración de una jornada laboral. Esto quiere decir que la mayor parte de usuarios mantienen una conexión activa a lo largo de su horario laboral. Luego se tiene que el segundo intervalo con mayor porcentaje [1,6] horas. Este porcentaje se puede atribuir a conexiones esporádicas de los usuarios. Es decir, que se conecten durante cierto tiempo durante la jornada, pero el resto del día no realicen tareas relacionadas con acceso a ficheros. Y finalmente tenemos el intervalo de menor porcentaje, pero de mayor duración de las conexiones. Este bajo porcentaje es normal ya que la jornada laboral de un trabajador normalmente es de 8 horas y estas conexiones que se extienden más allá de este horario podrían deberse a algún proceso de automatización de tareas. Estos comportamientos podrían indicar que es plausible aplicar alguna técnica de "clustering" para extraer tipologías de usuarios, sin embargo, en una primera versión de la herramienta se va modelar un usuario global agregado de todos estos.

### 2.1.2 Ficheros por conexión

A continuación, se muestra la CDF (ver Figura 4) correspondiente al número de ficheros que se han operado por conexión. a media global que es de 7262 ficheros por conexión. El mínimo es de 8 ficheros accedidos y el máximo es de 187209.

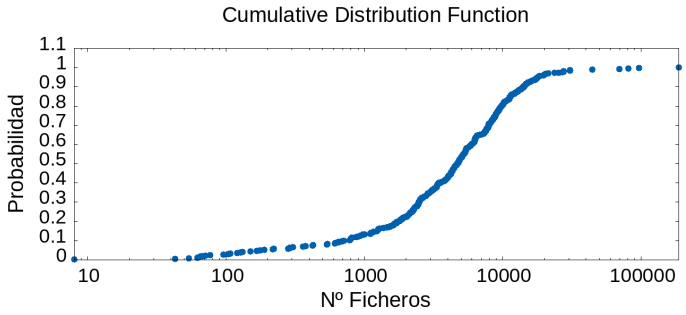


**Figura 4**. CDF para el número de ficheros por conexión

La primera impresión después de analizar esta gráfica es que existen unos usuarios más activos (operan con más ficheros) y otros menos activos. Aunque también se podría dar el caso de que las conexiones más largas sean las que más abran y las más cortas las que menos.

### 2.1.3 Series temporales de las conexiones

En esta subsección se analizará las conexiones a nivel granular, es decir, a nivel de cada acceso a ficheros, con el objetivo de verificar si existen distintos patrones de comportamiento de los usuarios.

En la Figura 5 se representa en el eje x el "timestamp" de los accesos en horas y en el "eje y" el número de bytes por acceso. Se puede observar que la conexión es de corta duración y con comportamiento a ráfagas. También se aprecia que los periodos de inactividad en esta conexión oscilan aproximadamente entre 20 minutos y unos pocos minutos.

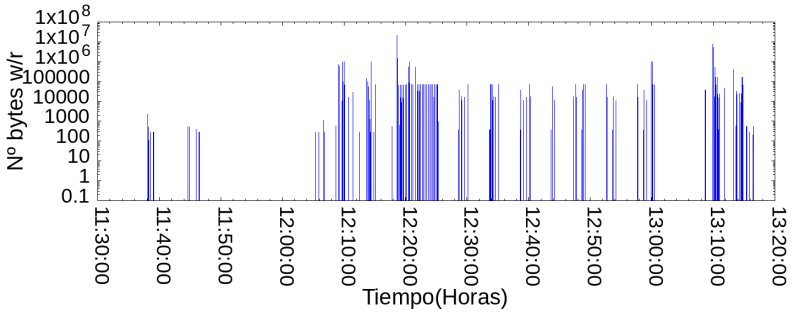


**Figura 5**. Serie temporal para el caso de una conexión que dura poco más de 1 hora y media (1.63 horas).

En la Figura 6 vemos un usuario que a primera impresión parece ser muy activo. Esta conexión corresponde aproximadamente a la duración de una jornada laboral. En este caso podemos apreciar en la gráfica que la duración de los periodos de inactividad oscilan entre algunas decenas de minutos, aproximadamente 45 minutos, y unos pocos minutos. Los patrones de ráfagas se siguen manteniendo para este caso.

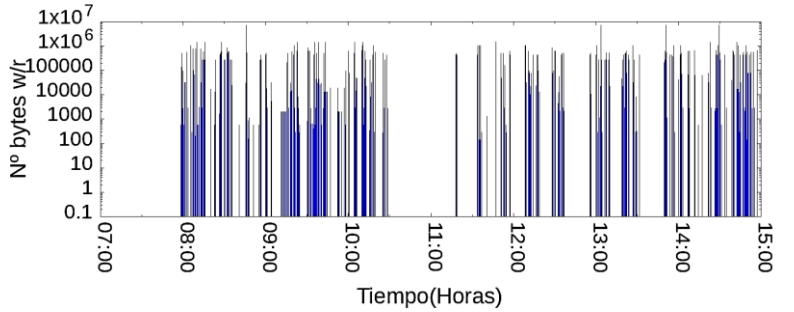


**Figura 6**. Serie temporal para el caso de una conexión que dura aproximadamente 7 horas (6.97 horas).

Como último ejemplo se tiene el caso para una conexión (ver Figura 7) que dura más de una jornada laboral. Parece un usuario de menor actividad que en el caso anterior. Además, estos casos (mayores de 8 horas), como se comentó con anterioridad, son los menos numerosos. También vemos que los periodos de inactividad oscilan entre una hora y unos pocos minutos para esta conexión. Por otra parte, el comportamiento a ráfagas se mantiene para esta conexión de larga duración.

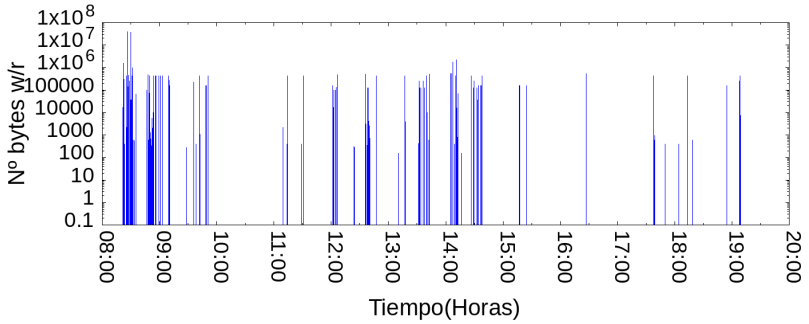


**Figura 7**. Serie temporal para el caso de una conexión que dura aproximadamente 10 horas (10.78 horas).

Como se ha visto en las gráficas los accesos siguen patrones de ráfagas para diferentes duraciones de las conexiones. Esto puede sugerir que de forma habitual hay periodos cortos de gran actividad donde el trabajador accede a varios ficheros y también largos periodos donde el usuario trabaja sin acceder a ficheros remotos, ya sea porque está trabajando de forma local o incluso porque se están viendo reflejados los descansos para comer.

En esta sección se ha podido obtener una primera impresión sobre la traza de datos de los usuarios. Se ha visto que predominan las conexiones de duración próximas a una jornada laboral entorno al 60%, por lo que se podría plantear inicialmente el modelar a un usuario global cuyo periodo de actividad se situé en 8 h. También se ha visto que las conexiones presentan un patrón en ráfagas sin importar la duración de la conexión, además estas ráfagas se presentan más o menos concentradas a lo largo del tiempo. Los periodos "idle" y la variabilidad de su duración también es otra constante que se ha podido apreciar en este análisis visual. Estos dos factores y su variabilidad en el tiempo indican que el comportamiento del usuario también es dependiente del tiempo y que se podría aproximar por una máquina de estados, donde cada estado modelaría el comportamiento de este durante cierta ventana temporal. En la siguiente sección se propone un generador de tráfico cuyo modelo estocástico está basado en máquinas de estado finitas (FSM, Finite State Machine).

# 3 Diseño de un modelo estocástico de usuarios de sistemas de compartición de ficheros

En esta sección se va a detallar la construcción del modelo estocástico basado en máquinas de estado finitas, el cual se usará posteriormente para generar trazas sintéticas. En las siguientes subsecciones se describe paso a paso el porqué de las decisiones en la estrategia de modelado propuesta. En una primera versión de la herramienta se ha decidido modelar un usuario global representativo de la carga, pero se deja la puerta abierta a que en futuras versiones se extraigan diferentes tipologías de usuarios.

## 3.1 Modelado a nivel de actividad macroscópica sobre ficheros

El modelo que se ha planteado consiste en tratar de abstraer las acciones a nivel de usuario, es decir, se busca simular las interacciones de un usuario con el sistema de ficheros. Estas interacciones se pueden resumir en escribir en ficheros y leer de ficheros, por lo que es razonable el modelar al usuario sabiendo cuando accede a cierto fichero y cuánta información escribe o lee de ese fichero. Esto se traduce en caracterizar la transferencia total de bytes ejecutadas sobre un fichero y su timestamp. Por ello, de la totalidad de información disponible por cada conexión se ha optado por utilizar solamente las peticiones (request) al servidor de ficheros donde había transferencia de bytes e imprimir estas en un fichero de texto, mucho más ligero de operar que un "pcap" con el total de los datos. Cabe aclarar que el modelado no se realiza a nivel de operaciones w/r, sino que se contabiliza el total de bytes transferidos por cada acceso a un fichero y una única referencia temporal asociada a este.

Con todo ello los datos, punto de partida del estudio, son tantos ficheros de texto como conexiones existen en la traza. Cada fichero de conexión dispone del siguiente formato: timestamp, Nº bytes leídos y Nº bytes escritos. En la Tabla 1 se muestra un ejemplo del formato de los ficheros de conexiones. El campo Nº bytes leídos contabiliza el total de bytes que se han escrito sobre un fichero concreto, similar para las escrituras.

| ***Timestamp*** | ***Nº bytes leídos*** | ***Nº bytes escritos*** |
|---|---|---|
| 1484550180.656317000 | 111 | 0 |
| 1484550181.370658000 | 1978 | 0 |
| 1484550181.383099000 | 2006 | 0 |
| 1484550181.496043000 | 0 | 510 |

**Tabla 1**. Formato de los datos del fichero procesado de una conexión.

En esta sección se ha expuesto la primera decisión de diseño del modelo, que consiste en caracterizar al usuario mediante tres variables que son: el momento del día cuando accede a un fichero, la cantidad de información que lee de ese fichero y la cantidad de información que escribe sobre ese fichero. En la siguiente sección se explica el ranurado de las conexiones como punto de partida para el modelado de la máquina de estados.

## 3.2 Análisis temporal de la actividad

Como se ha visto anteriormente, las conexiones objeto de estudio, presentan una variabilidad dependiente del tiempo, tanto para las ráfagas de accesos como para los periodos de inactividad, por tanto, es razonable el ranurar las series temporales de las conexiones en intervalos de tiempo, para así poder crear un modelo de máquinas de estados finita que simule durante cada estado (intervalo de temporal) un comportamiento diferente de la actividad del usuario. Con este objetivo en esta sección se detallará la metodología para la ranuración de las conexiones.

En primer lugar, teniendo en cuenta el comportamiento a ráfagas descrito en secciones anteriores se ha empleado un ranurado de 5 minutos. Más adelante en el documento se probará con otros valores de ranuración para comprobar sus efectos sobre los resultados obtenidos.

Por otro lado, con la finalidad de poder modelar el comportamiento dependiente del tiempo y poder agrupar muestras de diferentes conexiones, se ha establecido una referencia común para el enventanado, esta es las 00:00 del día. A continuación, se muestra gráficamente (ver Figura 8) la metodología para el ranurado, en base a esta referencia.

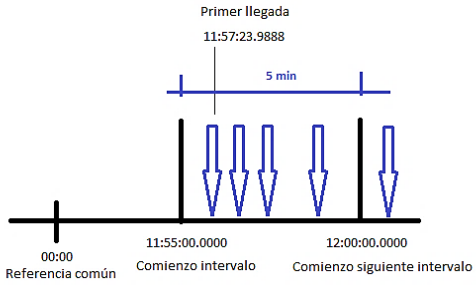


**Figura 8**. Construcción de las ventanas temporales para el caso de intervalos contiguos, las flechas azules representan las peticiones de accesos a ficheros.

En la Figura 8 se muestra la forma de dividir los accesos en intervalos de tiempo para el caso inicial de 5 minutos, pero es extensible a otro valor. En primer lugar, operando con el primer timestamp de la conexión y con referencia común (00:00 h) se obtiene el tiempo donde inicia la ventana (11:55 h en la figura). Al usar una referencia común se consigue ranurar diferentes conexiones con ventanas iguales, es decir, se logra sincronizar las diferentes conexiones.

Cabe señalar que cada acceso (flecha azul) representa el momento de inicio de la transferencia de bytes. Cada transferencia puede estar compuesta por múltiples reads() o writes() que se ejecutarán

más allá de esta referencia de inicio. En este modelo no se precisa el caracterizar a tan bajo nivel, ya que, como se comentó en la sección anterior, se busca abstraer el comportamiento a nivel de usuario, por ello solo se modela cuando se inició esa transferencia (usuario accede al fichero) y cuantos bytes se transfirieron.

En esta sección se ha visto como se crean los intervalos temporales que conforman cada conexión, estos quedan definidos por la dupla (N.º bytes w/r, N.º accesos). Los intervalos o ventanas temporales son la base sobre la que se edifican las decisiones de diseño de las secciones siguientes.

## 3.3 Análisis de las CDF de N.º de accesos y N.º bytes w/r de los intervalos temporales

El objetivo de esta sección es lograr extraer patrones de comportamiento a partir de esas duplas de N.º de bytes y N.º de accesos, con el objetivo final de poder agrupar estas duplas dentro de un número limitado de estados, es decir, recogiendo dentro de cada estado aquellas duplas que presenten un comportamiento similar. De este modo se podría modelar a un usuario como transiciones entre estados de actividad baja, media y alta. Todo ello teniendo en mente el modelado de la máquina de estados finita.

Con este propósito se ha procedido a calcular sus distribuciones acumuladas de probabilidad tanto para el N.º de bytes w/r por intervalo como para el N.º de accesos por intervalo. En un primer paso se optó por la ranura de 5 minutos, más adelante se probará que esta duración del intervalo no es la más indicada.

En la Figura 9 se puede observar, que el porcentaje de veces que no ha habido accesos es considerable alrededor de un 20 %. Estos se corresponden con los intervalos 'idle' o de inactividad. Por otro lado, vemos que para ranurado de 5 minutos los intervalos con pocos accesos son elevados. Alrededor del 50 % de los casos totales se cubren en el rango de [0,2] accesos. Esto puede ser un indicador de que quizás es necesario el incrementar la duración de la ventana temporal.

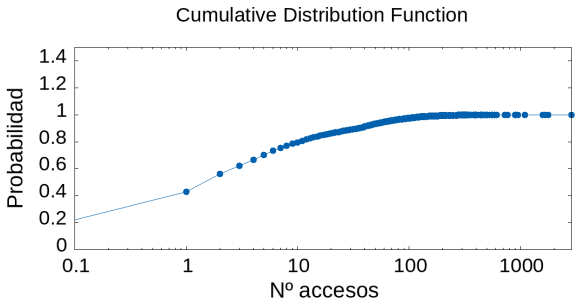


**Figura 9**. CDF para el número de accesos durante un intervalo temporal (caso 5 minutos)

Se observa en la Figura 10 que los casos donde no hay ni escrituras ni lecturas es igual al caso anterior (20%) como es lógico. Por otra parte, vemos que el mínimo de bytes escritos es 2 y el máximo 1.5 GB aproximadamente.

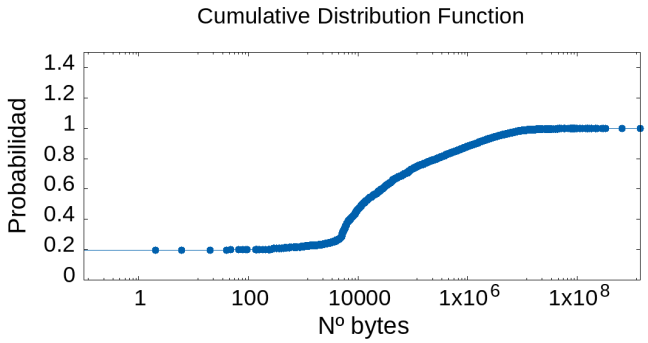


**Figura 10**. CDF para el número de bytes w/r durante un intervalo temporal (caso 5 minutos)

En la Figura 11 se representa la totalidad de los intervalos temporales extraídos de la traza. En el eje x tenemos el Nª de accesos y en el eje y el Nº de bytes w/r. Cada punto azul representa un intervalo de duración igual a la de la ventana temporal. Los intervalos de tiempo sin ningún acceso no aparecen ya que el eje es logarítmico, pero representan un porcentaje considerable (alrededor de 20 %). También vemos que hay cierta tendencia cuando hay intervalos con pocos accesos, lo cual se aprecia en una mayor dispersión en el eje y. Finalmente, cuando el número de accesos crece la suma de bytes w/r también tiende a crecer. A partir de esta gráfica se puede llevar a cabo cierta clasificación. Por ejemplo hay intervalos con mayor actividad (esquina superior derecha), otros con menor actividad (esquina inferior izquierda) y el resto de puntos con una actividad intermedia.

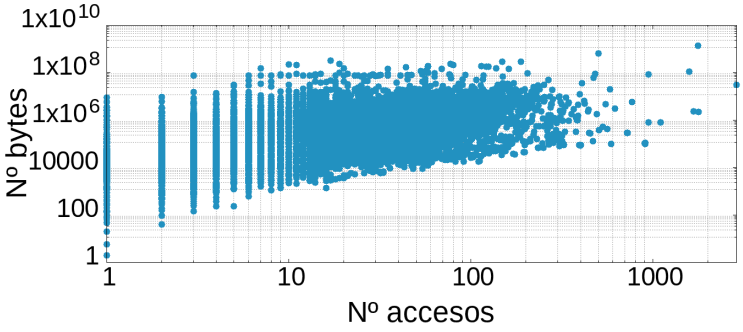


**Figura 11**. N.º bytes w/r frente a N.º accesos.

En esta sección se ha podido comprobar que es posible modelar a los usuarios como una máquina de estados finita, donde cada estado representa el comportamiento de mayor o menor actividad durante cierto espacio de tiempo.

## 3.4 Modelado máquina de estados finita (FSM)

En esta sección se abordará la construcción de la máquina de estados finita, más concretamente una cadena de Markov, ya que la decisión de transitar de un estado se hace en base a probabilidades.

### 3.4.1 Aproximación inicial

En la Figura 12 se muestra un ejemplo de lo que se pretende construir con esta estrategia de modelado. Como se comentó en la sección anterior el objetivo es representar estados de baja, media y alta actividad, por ello cada círculo representa un estado con un nivel de actividad diferente del usuario. Las flechas indican las probabilidades de transitar entre esos dos estados. El modelado de estas transiciones se explicará más adelante.

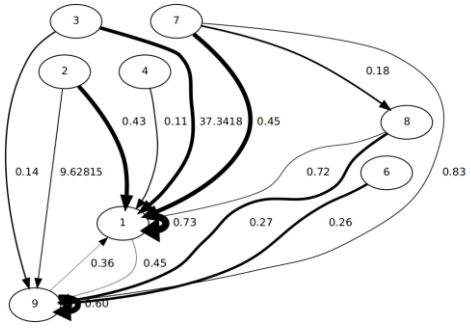


**Figura 12**. Ejemplo de máquina de estados finita (FSM)

Para crear los diferentes estados de esta máquina los intervalos se clasificarán según su número de bytes w/r y según su número de accesos. Los límites que marcan estas clases o estados se han determinado tratando de recoger en cada segmento del correspondiente eje 1/3 de las muestras. Esto corresponde a dividir las CDF's anteriormente descritas con límites que recojan en cada intervalo el 33,33% de la probabilidad y tomar ese caso límite como valor del correspondiente eje X o Y de la Figura 13. En posteriores secciones se experimentará con otras configuraciones, es decir, otras metodologías de división de estados para verificar cuál de ellas responde mejor.

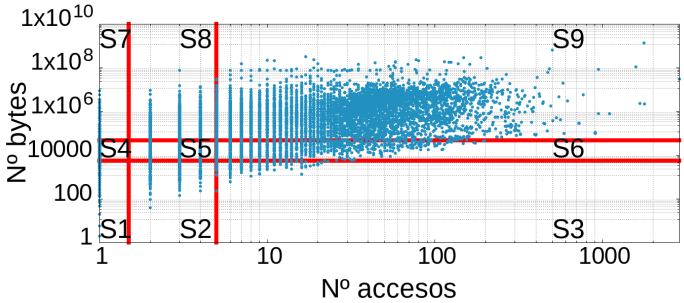


**Figura 13**. División en nueve estados de la totalidad de los intervalos (límites verticales y horizontales fijos).

La Figura 13 muestra la división para un enventanado de 5 minutos y con límites en eje x e y fijos. Cada recuadro resultado de esta división representa un estado de actividad del usuario. El estado 1 se sitúa en la esquina inferior izquierda, corresponde con el de menor actividad (menos accesos y menos bytes w/r), seguido en al misma fila del estado 2 (actividad intermedia, pocos bytes w/r y unos cuantos accesos) y 3 (actividad media-alta, muchos accesos pero pocos bytes w/r), en la fila superior a esta se tiene el estado 4(actividad media, muy pocos accesos pero cantidad considerable de bytes w/r), 5 ( unos cuántos accesos y cantidad considerable de bytes w/r) y 6(actividad media), y finalmente en la última fila tenemos los estados 7, 8 y 9, siendo el nueve el de mayor actividad, ya que se ubica en la zona donde hay mayor número de accesos y mayor número de bytes w/r. Se puede observar que el reparto de intervalos resultantes es bastante heterogéneo, es decir, en el estado 1 se recogen un porcentaje considerable de los intervalos totales, alrededor del 30%, mientras que en estado tres no llega a 1%. Si este reparto desigual es muy marcado hace que carezca de sentido la división en estados ya que la máquina pasará la mayor parte del tiempo en esos estados donde la probabilidad es dominante.

### 3.4.2 Límites variables y estado "idle"

Se ha visto que un reparto desequilibrado de casos entre los estados resulta en estados degenerados o muy poco representados. Para una cierta cantidad de estados objetivo intentaremos que todos tenga una representatividad similar. Para conseguir este reparto más homogéneo se combinan límites fijos con límites variables. En la Figura 14 se ve cómo los límites (rectas) fijos corresponden al eje Y, y los ejes variables al eje X.

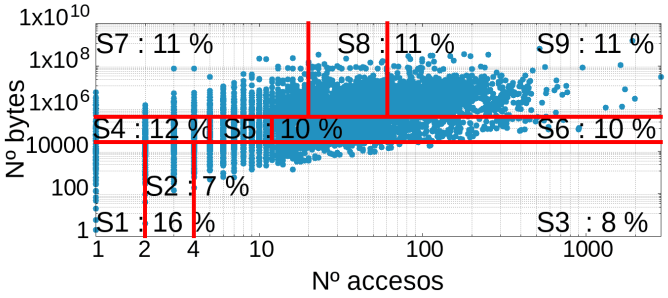


**Figura 14**. División de los estados mediante combinación de ejes fijos (rectas horizontales en eje Y) y ejes variables (rectas verticales en eje X).

Para crear estas divisiones variables se ha procedido en primer lugar a clasificar la totalidad de intervalos dentro de tres subregiones, creadas por límites fijos en el eje Y. A continuación, utilizando la CDF, construida solo con los intervalos que caen dentro de cada subregión, se ha subdividido estos a su vez en tres subregiones. Esta estrategia mejora el reparto de intervalos por estados, llegando a un reparto casi homogéneo.

Una segunda estrategia se basaría en mantener las divisiones en el eje Y fijas y variar las del eje X. La metodología para extraer estos últimos es similar a la comentada. Se observa (ver Figura 15) que el reparto mejora al igual que el caso anterior, aunque sigue sin ser del todo homogéneo.

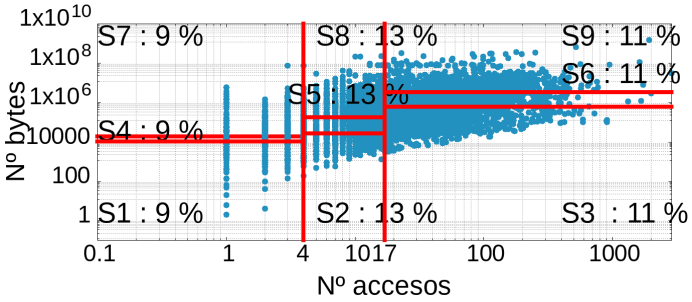


**Figura 15**. División de los estados mediante combinación de ejes fijos (rectas verticales en eje X) y ejes variables (rectas horizontales en eje Y).

Analizando las distribuciones de probabilidad acumuladas se ha comprobado que esto se debe a que hay un porcentaje demasiado alto de intervalos con pocos accesos (1,2 accesos) y a que los casos con cero accesos también afectan, sobre todo al estado 1. Por ello, se ha decidido crear un estado de inactividad aparte del resto y mantener el diseño con límites variables. Para el caso representado en la Figura 15 pasaríamos a tener una FSM de 9 + 1 estados (9 con actividad y 1 estado "idle"). En la sección de calibrado se detallará cuáles son los ajustes concretos más adecuados para modelar la carga.

### 3.4.3 Caracterización estadística de los estados

En la subsección anterior se han extraído múltiples estados con finalidad de que cada uno de ellos represente cierto tipo de actividad del usuario durante cierta ventana temporal. Por ello, las propiedades que se van a modelar en esta sección, de forma estadística, son el número de accesos por estado y la cantidad de información transferida en cada uno de esos accesos.

En primer lugar, para poder simular el comportamiento de los usuarios a nivel granular, es decir a nivel del usuario U accede a un fichero y transfiere X bytes, en un instante de tiempo T es necesario modelar el Nº de accesos por cada estado mediante los tiempos entre llegadas o tiempos entre accesos recogidos en cada intervalo temporal que cae dentro de este estado y posteriormente agregarlos para obtener unas estadísticas por estado. El resultado de esta operación es similar al mostrado en la Figura 16.

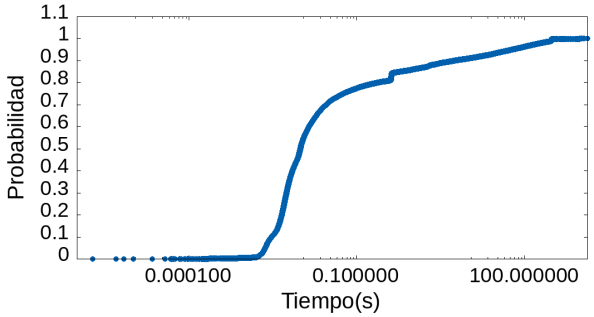


**Figura 16**. Ejemplo de CDF relativa a tiempos entre llegadas para un estado.

Una vez se dispone de esta distribución de probabilidad por cada estado, ya es posible generar los accesos dentro del mismo. Para ello se aplica el método de la transformada inversa para obtener muestras aleatorias (ej. 0.1, 0.001, etc.) que sigan esta distribución empírica.

El segundo parámetro que caracteriza cada estado es la distribución acumulada de probabilidad del número de bytes que se escribirá o leerá por acceso. Para construir esta distribución se ha seguido una serie de pasos similares al caso anterior. La diferencia está en que ahora se ha contabilizado el número de bytes de cada acceso recogido dentro de los intervalos. En la Figura 17, se muestra un ejemplo de esta distribución.

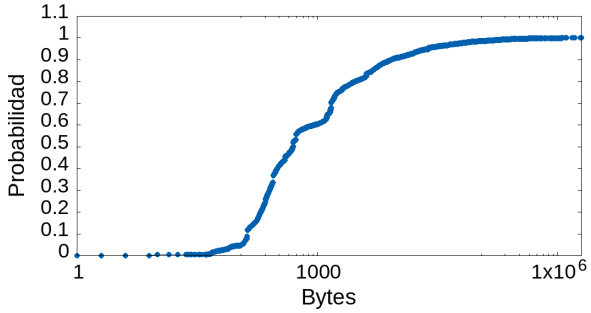


**Figura 17**. Ejemplo de CDF para el Nº de bytes.

Igualmente, para generar muestras aleatorias (ej. 111 bytes, 19238 bytes, etc.) a partir de esta distribución se aplica el método de la transformada inversa. Finalmente, estos valores se escribirán sobre un fichero de texto que se pasará al usuario o cliente para que ejecute estas órdenes en tiempo real. El desarrollo a nivel de software de esta funcionalidad está disponible en el Anexo I.

### 3.4.4 Matriz de transiciones

La matriz de transiciones recoge la relación entre los estados, comentados en secciones anteriores, en forma de probabilidades. En este caso solo se construirá una matriz ya que como se comentó al inicio de la sección en una primera versión de la herramienta se construirá solo una tipología de usuario. Pero en el caso de querer generar más tipologías esta matriz sería un factor clave, ya que una relación diferente entre los estados determinaría un comportamiento diferente.

Para crear esta matriz en primer lugar es necesario ranurar, mediante la metodología propuesta, todas las conexiones disponibles. Obteniendo un resultado como el que se muestra en la Figura 18, para una conexión. Los puntos representan las duplas (Nº bytes w/r, Nº accesos).

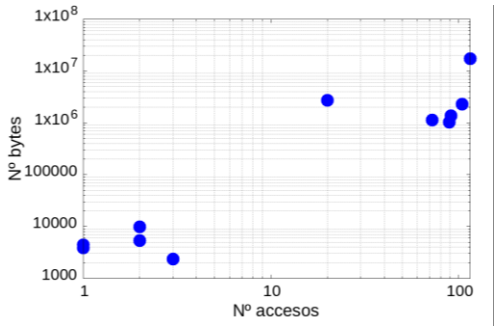


**Figura 18**. Ejemplo de ranurado de una conexión.

A continuación, se clasifica cada intervalo dentro de uno de los estados definidos. En la Figura 19 se puede ver la serie temporal de los intervalos ya clasificados.

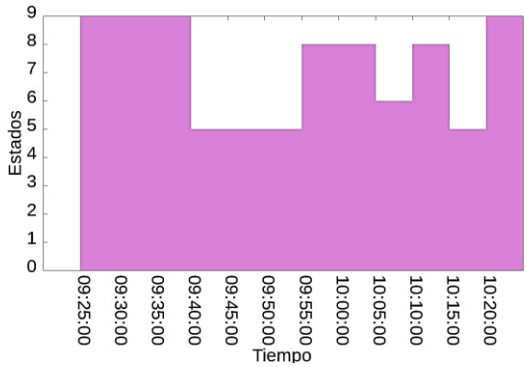


**Figura 19**. Serie temporal de los estados para un ranurado de 5 min. en una conexión.

El siguiente paso es contabilizar esas transiciones para cada conexión y agregarlas. Para ello si definimos el estado ***i*** como el actual y el estado siguiente como ***j***, tenemos que cada transición queda representada por:

$$a_{i,j} = n(\, j \mid i \,), \text{ donde n representa el número de ocurrencias.} \quad (1)$$

Entonces la matriz empírica de transiciones **A** para una cadena de Markov de S estados, donde pueden ocurrir transiciones entre cualquier estado, se define como:

$$A := (a_{i,j})_{mxn}, \ \forall\, i,j \in \{1...m\} \text{ and } m=n=S \quad (2)$$

Un ejemplo de esta matriz empírica para el caso de S = 10:

$$A = \begin{pmatrix} 385 & 88 & 25 & 21 & 7 & 18 & 13 & 32 & 17 & 10 \\ 101 & 265 & 50 & 37 & 13 & 45 & 27 & 41 & 24 & 27 \\ 31 & 71 & 184 & 62 & 71 & 60 & 47 & 27 & 56 & 39 \\ 27 & 38 & 72 & 187 & 41 & 59 & 80 & 39 & 44 & 35 \\ 9 & 22 & 78 & 46 & 314 & 40 & 23 & 38 & 38 & 12 \\ 16 & 37 & 64 & 64 & 48 & 181 & 57 & 61 & 92 & 33 \\ 14 & 40 & 42 & 103 & 34 & 74 & 126 & 47 & 82 & 85 \\ 41 & 43 & 27 & 36 & 54 & 62 & 28 & 263 & 51 & 21 \\ 25 & 32 & 56 & 51 & 49 & 80 & 61 & 65 & 170 & 58 \\ 10 & 15 & 34 & 52 & 22 & 61 & 117 & 42 & 108 & 180 \end{pmatrix} \tag{3}$$

En la Figura 20 vemos la representación de la matriz (ecuación 3) mediante colores, con la que se puede analizar de forma más clara cómo se distribuyen las transiciones. Por ejemplo, vemos que hay muchos casos donde se ha pasado del estado 0 al 0.

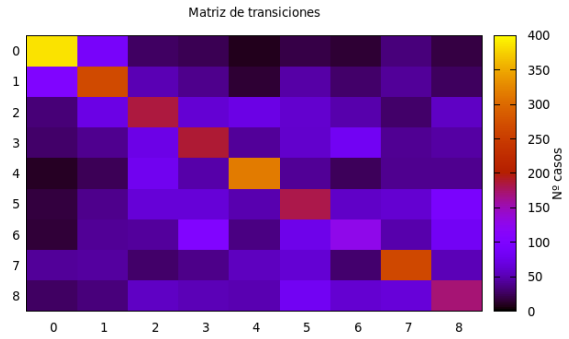


**Figura 20**. Heatmap para el caso de la matriz de transiciones de la ecuación 3

Posteriormente, ha sido necesario procesar esta matriz ya que se necesitan probabilidades y no casos para poder modelar la carga. Más concretamente, se necesita disponer de la probabilidad condicional de transitar al estado ***j*** si en el instante anterior se está en el estado ***i***. Esta probabilidad se define como:

$$p_{i,j} = \frac{a_{i,j}}{\sum_{k=1}^{n} a_{i,k}} \text{ donde } a_{i,j} \in A \tag{4}$$

El resultado de aplicar la fórmula (ecuación 4) a la matriz del ejemplo (ecuación 3), es el siguiente:

$$P = \begin{pmatrix} 0.625 & 0.142 & 0.040 & 0.034 & 0.011 & 0.029 & 0.021 & 0.051 & 0.027 & 0.016 \\ 0.160 & 0.420 & 0.079 & 0.058 & 0.020 & 0.071 & 0.042 & 0.065 & 0.038 & 0.042 \\ 0.047 & 0.109 & 0.283 & 0.095 & 0.109 & 0.092 & 0.072 & 0.041 & 0.086 & 0.060 \\ 0.043 & 0.061 & 0.115 & 0.300 & 0.065 & 0.094 & 0.128 & 0.062 & 0.070 & 0.056 \\ 0.014 & 0.035 & 0.125 & 0.074 & 0.506 & 0.064 & 0.037 & 0.061 & 0.61 & 0.019 \\ 0.024 & 0.056 & 0.098 & 0.098 & 0.073 & 0.277 & 0.087 & 0.093 & 0.140 & 0.050 \\ 0.021 & 0.061 & 0.064 & 0.159 & 0.052 & 0.114 & 0.194 & 0.072 & 0.126 & 0.131 \\ 0.065 & 0.068 & 0.043 & 0.057 & 0.086 & 0.099 & 0.044 & 0.420 & 0.081 & 0.033 \\ 0.038 & 0.049 & 0.086 & 0.078 & 0.075 & 0.123 & 0.094 & 0.100 & 0.262 & 0.089 \\ 0.015 & 0.023 & 0.053 & 0.081 & 0.034 & 0.095 & 0.182 & 0.065 & 0.168 & 0.280 \end{pmatrix} \tag{5}$$

En la Figura 21 vemos de forma más clara las probabilidades de transición. Por ejemplo, se ve que partiendo del estado 0 es muy probable que se vuelva a transitar a sí mismo. Esta representación se usará en las siguientes secciones para tratar de validar gráficamente las elecciones de calibración de parámetros.

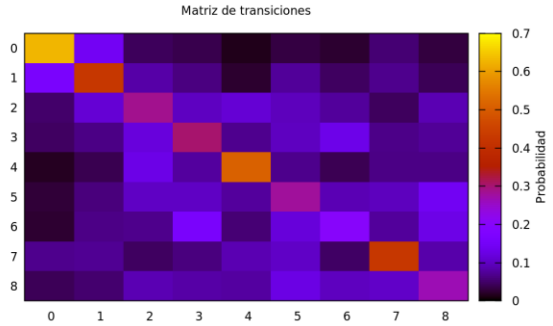


**Figura 21**. Heatmap para el caso de la matriz de transiciones de probabilidad

A lo largo de esta sección se ha descrito la metodología para desarrollar una máquina de estados finita cuya finalidad es modelar el comportamiento de los usuarios, donde cada estado representará un nivel de actividad del usuario. A su vez cada estado quedará caracterizado por una dupla (CDF Nº de accesos y CDF Nº de bytes) cuya distribución de probabilidad se extraerá a partir del ranurado de los datos de usuario iniciales. Finalmente, también se ha detallado como se interconectan estos estados en forma de una matriz de transición que recoge las probabilidades entre todas las combinaciones posibles de transición.

# 4 Calibración de los parámetros y evaluación del modelo

En esta sección se expondrá el ajuste de los parámetros del modelo estocástico con el objetivo de que la carga de trabajo sintética se asemeje lo máximo posible al tráfico real de los usuarios.
Los parámetros que se han calibrado son los comentados en secciones anteriores, tales como duración de la ventana temporal, número de estados de la FSM y combinación de límites variables y fijos. Al final de la sección se justificará la configuración óptima mediante una validación que comparará las diferentes alternativas propuestas. Cabe señalar, que se va a trabajar con resultados extraídos de la herramienta software que ya se ha implementado. El desarrollo software de este generador se detalla en el apéndice.

## 4.1 Experimentos con intervalos 10, 20 y 30 minutos y combinaciones de límites fijos y variables

Anteriormente se ha trabajado con ventanas temporales de 5 minutos para la explicación de la metodología, pero se ha comprobado que esta opción es inviable. El motivo está en que para la caracterización de los estados se necesita disponer de los tiempos entre llegadas, es decir poder calcular la diferencia entre al menos dos accesos, y para el caso de 5 minutos existía un estado (estado 1) donde, aunque se apliquen divisiones de los estados variables, solo se recogían intervalos temporales con un solo acceso, lo cual imposibilita el uso de esta ranuración.

### 4.1.1 Comparativa inicial para diferentes ventanas

Con el objetivo de solventar el problema de los intervalos con un único acceso y así poder disponer de al menos dos accesos entre los que calcular el tiempo entre llegadas se ha representado la CDF para el número de accesos y el número de bytes w/r para diferentes tamaños de ventana temporal. De este modo se puede comprobar visualmente con qué ranurado se obtiene un menor número de intervalos con pocos accesos. Más concretamente se busca disponer de intervalos con más de un acceso en la primera división de la CDF de Nº de accesos, dentro de la primera subregión de 33% que marca el primer límite vertical de los estados.

Como se puede observar en la Figura 22 el caso de 5 minutos es el que ofrece una mayor probabilidad para los casos con un solo acceso, superando el 33% en los casos con un solo acceso. El siguiente valor de 10 minutos consigue que las probabilidades de los casos con un acceso caigan de forma considerable, cayendo hasta un 3.3%, aunque para los casos de dos y tres accesos la probabilidad sigue siendo relevante. Para 20 y 30 minutos también se estaría por debajo del umbral de 33%.

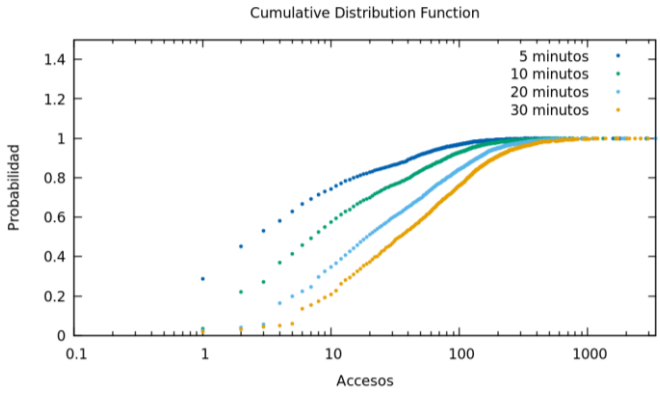


**Figura 22**. CDF del Nº de accesos para diferentes duraciones de la ventana.

En la Figura 23 se muestra cómo varía el número de bytes por intervalo según el tamaño de la ventana. En general, se ve que la tendencia es que las cantidades pequeñas de bytes w/r son menos probables conforme aumenta la ventana. Esto parece lógico, ya que al aumentar la duración del intervalo temporal lo normal es que aumente el número de accesos y con ello el número de bytes w/r transferido.

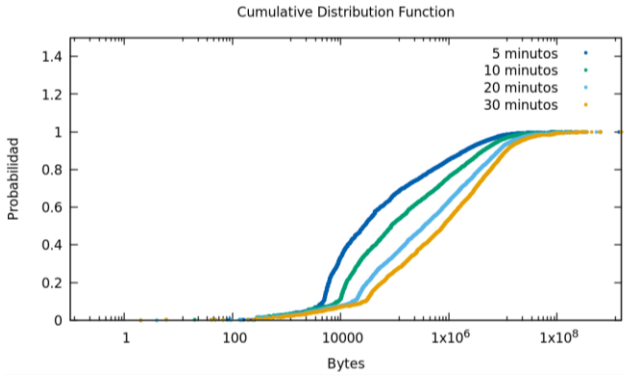


**Figura 23**. CDF del Nº de bytes w/r para diferentes duraciones de la ventana.

En las siguientes subsecciones se estudiarán los casos de 10, 20 y 30 minutos, que son los tamaños que cumplen que la probabilidad de un solo acceso es baja, por debajo de 0.033. Con esto se espera conseguir que al menos haya dos accesos por estado para poder extraer los tiempos entre llegadas.

### 4.1.2 Metodología empleada en la evaluación del modelo

La metodología seguida en estos experimentos consiste en delimitar los estados con cierta combinación de límites variables y fijos y posteriormente programar esta configuración en la herramienta software. Tras ejecutar el simulador y obtener una traza de tráfico se extraen los resultados en forma de CDF para las variables Nº de accesos y Nº de bytes w/r. Finalmente se lleva a cabo la comparación de las mismas con los datos extraídos del tráfico de los usuarios reales. Cabe destacar que en las gráficas de nubes de puntos que se mostrarán en esta sección no se ha representado el estado 0 (inactividad), ya que no aporta información visualmente relevante.

En la Tabla 2 se detallan la totalidad de experimentos que se ha llevado a cabo, la configuración programada en el generador de trazas, así como el total de muestras tras ranurar la traza sintética obtenida. Se ha simulado para cada experimento un único usuario global durante 2000 horas. Se ha considerado esta cifra ya que de los datos iniciales se disponen también 2000 horas aproximadamente y además vemos en la tabla que se consigue un elevado número de muestras en todos los experimentos.

| ***Ventana*** | ***Configuración*** | ***Descripción*** | ***Muestras*** |
|---|---|---|---|
| 10 | x var. 10est. | Lím. eje X variables, 2 lím. eje Y fijos y 10 estados | 12000 |
| | y var. 10est. | Lím. eje Y variables, 2 lím. eje X fijos y 10 estados | 12000 |
| 20 | x var. 10est. | Lím. eje X variables, 2 lím. eje Y fijos y 10 estados | 6000 |
| | y var. 10est. | Lím. eje Y variables, 2 lím. eje X fijos y 10 estados | 6000 |
| | x var. 13est. | Lím. eje X variables, 3 lím. eje Y fijos y 13 estados | 6000 |
| | x var. 13est. | Lím. eje Y variables, 3 lím. eje X fijos y 13 estados | 6000 |
| 30 | x var. 10est. | Lím. eje X variables, 2 lím. eje Y fijos y 10 estados | 4000 |
| | y var. 10est. | Lím. eje Y variables, 2 lím. eje X fijos y 10 estados | 4000 |
| | x var. 13est. | Lím. eje X variables, 3 lím. eje Y fijos y 13 estados | 4000 |
| | x var. 13est. | Lím. eje Y variables, 3 lím. eje X fijos y 13 estados | 4000 |

**Tabla 2**. Configuraciones de los experimentos.

Cabe aclarar que estos experimentos se han llevado a cabo a nivel del módulo generador de observaciones (ver Anexo I), es decir se han realizado a nivel de comprobar que los ficheros de texto, que contienen las acciones (timestamp, op. w/r) que los usuarios ejecutarán posteriormente en tiempo real, presentan las distribuciones de probabilidad ajustadas a las de la traza de datos inicial. Ha sido posible realizar los experimentos de esta forma, ya que en un experimento anterior se pudo comprobar que la implementación software no afectaba a los resultados obtenidos, con lo cual, saltarse el paso de ejecución en tiempo real ha permitido agilizar el proceso de depurado del modelo.

### 4.1.3 Ventana temporal de 10 minutos

Se estudiará el primer caso que cumplía la premisa de que el número de intervalos con 1 acceso estaba por debajo del 33% para límites fijos. A diferencia de los casos 20 y 30 minutos, que se explicarán más adelante, para este ranurado no se ha podido crear configuraciones con 13 estados por la limitación de tener 1 solo acceso en los estados situados a la izquierda de la gráfica con la nube de puntos.

En este escenario se emplean 2 límites fijos en el eje Y, y dos límites variables en el eje X. En la siguiente Figura 23 podemos ver el resultado de esta división. Se observa que se sigue produciendo

el problema de un único acceso en el estado 1, con lo cual no se puede calcular tiempos entre llegadas. Por lo que esta combinación se descarta.

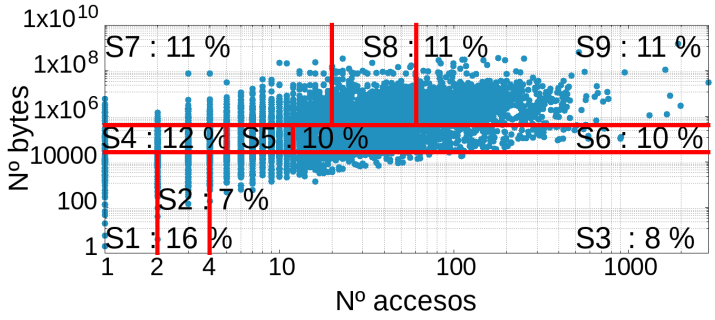


**Figura 23**. Nº bytes w/r frente a Nº de accesos para ventana de 10 minutos, lím. x var, 10 estados.

La siguiente configuración consiste en 2 límites verticales fijos y límites variables en el eje y. Como se ve en la Figura 24, con esta combinación ya se dispone de una división de estados viable (más de 1 acceso en los estados 1, 4, y 7). No obstante, el reparto de intervalos no acaba de ser del todo uniforme.

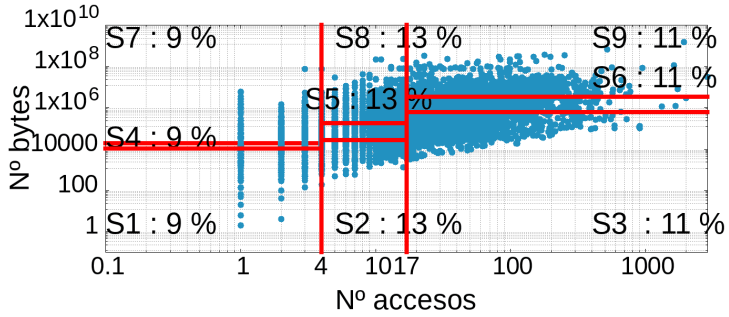


**Figura 24**. Nº bytes w/r frente a Nº de accesos para ventana de 10 minutos, lím. y var, 10 estados.

Si se observa la Figura 25 se muestra la CDF para el número de accesos extraída a partir de la traza real y la obtenida a partir de la traza sintética (generada con nuestra herramienta) utilizando las configuraciones expuestas. Vemos que la CDF sintética no se adapta bien a la CDF real. Se observan una serie de curvas (similares a oscilaciones, este fenómeno se explicará más adelante). Estas se traducen en que hay casos (duplas Nªbytes, Nº accesos) que nuestro modelo no llega a reproducir con exactitud, es decir son menos probables de lo que se ve para el caso de la traza real.

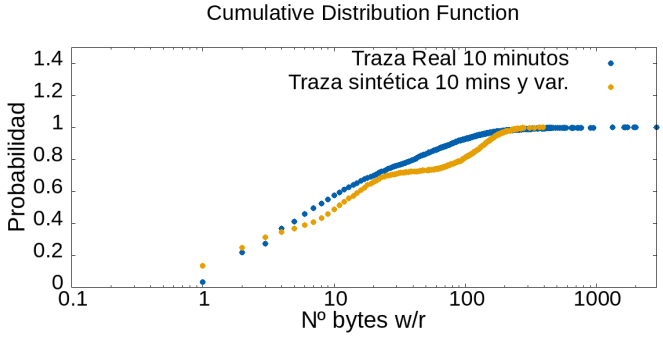


**Figura 25**. Comparación CDF Nº accesos, Real (Línea azul) vs Sintética (Línea amarilla)

En la Figura 26 se compara igualmente la CDF de la traza real y la sintética, solo que esta vez es para el número de bytes w/r. Se observa que en este caso el ajuste de la CDF sintética a la real es mayor en comparación a la de número de accesos.

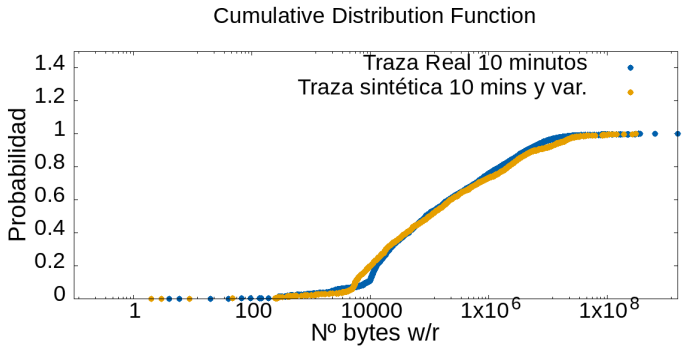


**Figura 26**. Comparación CDF Nº bytes w/r, Real (Línea azul) vs Sintética (Línea amarilla)

### 4.1.4 Caso ventana temporal 20 minutos

En este caso se incrementa la duración de la ventana con el objetivo de recoger intervalos con más accesos en los estados de la izquierda de la gráfica de puntos, habiendo superado con creces la restricción de menos del 33% de casos para un único acceso.

La combinación que se muestra en la Figura 27 es de límites variables en el eje x y 2 límites en el eje y fijos. La principal diferencia respecto del caso anterior está en que en el estado 1, que pasa a recoger intervalos con mayor número de accesos (de 1 a 3 accesos).

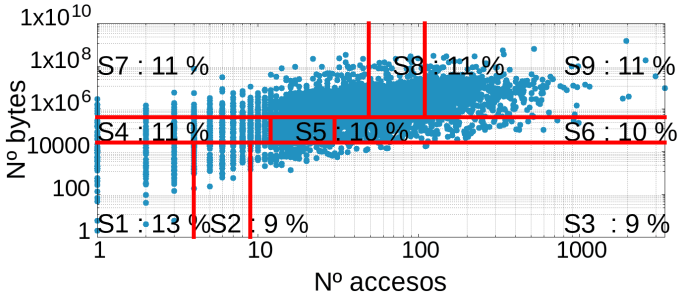


**Figura 27**. Nº bytes w/r frente a Nº de accesos para ventana de 20 minutos, lim. x var, 10 estados.

También, el porcentaje de intervalos por estado mejora respecto al caso de 10 minutos e igual configuración. Aun así, el estado 1 sigue siendo problemático al recoger un porcentaje de intervalos mayor (13%) que sus estados vecinos. Se busca esta uniformidad en los estados para evitar estados poco representativos.

En este otro caso (ver Figura 28) se presenta la configuración con 2 límites fijos en el eje vertical y límites variables en el eje Y. También se aprecia que el porcentaje de intervalos por estado ha

mejorado considerablemente. Del mismo modo los estados problemáticos de la izquierda recogen intervalos con hasta con 9 accesos.

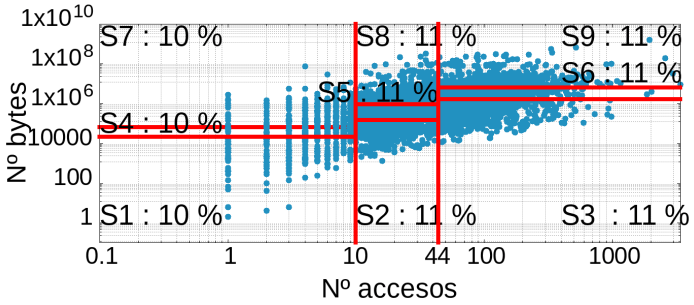


**Figura 28**. Nº bytes w/r frente a Nº de accesos para ventana de 20 minutos, lím. y var, 10 estados

En la Figura 29 vemos que esta nueva configuración consiste en 3 límites fijos en el eje y, y límites variables en el eje x, como resultado se tienen un total de 13 estados (12 activos + 1 inactivo). Con esta división se tiene que el porcentaje de intervalos por estado es bastante heterogéneo sobre todo para los estados de la primera fila.

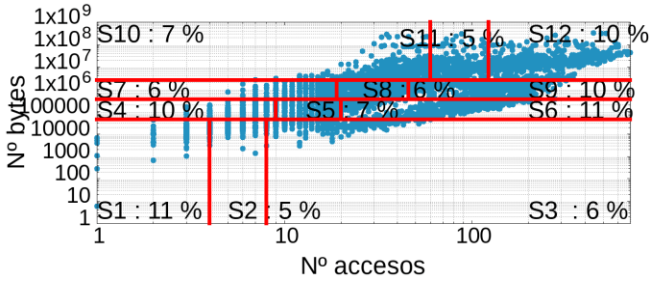


**Figura 29**. Nº bytes w/r frente a Nº de accesos para ventana de 20 minutos, lím.x var, 13 estados

También vemos que en el estado 1 se recogen sólo los casos hasta 3 accesos. Con lo cual el modelo parece haber empeorado con esta configuración.

En esta otra configuración (ver Figura 30) también para 13 estados, tenemos 3 límites fijos verticales y límites variables en el eje y. El reparto de porcentajes por intervalo mejora, en relación del otro caso de 13 estados, exceptuando para los estados en la zona de la esquina superior derecha.

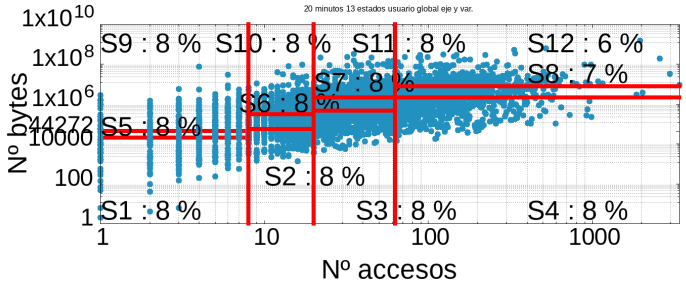


**Figura 30**. Nº bytes w/r frente a Nº de accesos para ventana de 20 minutos, lím.y var, 13 estados

En la Figura 31, se representan igual que en el caso anterior, una comparación entre las distribuciones de probabilidad acumuladas extraídas de los datos reales frente a la generada con

nuestro modelo. Vemos que en comparación con el caso anterior de 10 minutos, en general las CDF sintéticas se asemejan más. Destacan los casos donde los límites variables están en el eje x (x var, línea negra y amarilla), que visualmente parecen más similares a la real. También se observa que para los casos de límites variables en el eje y (y var, líneas verde y naranja), se mantiene la tendencia que se veía para 10 minutos con oscilaciones, más adelante se comentará el motivo de este fenómeno.

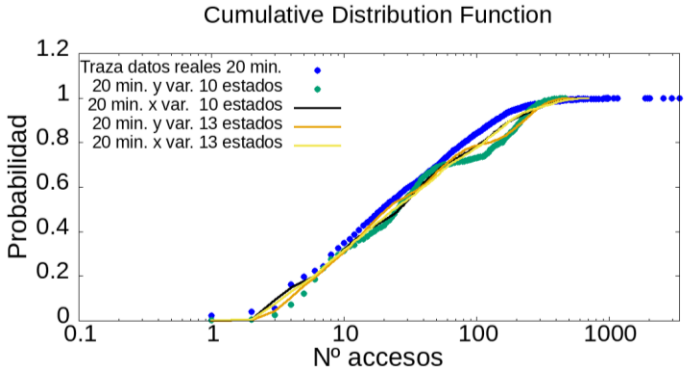


**Figura 31**. Comparación CDF Nº accesos (Línea Morada corresponde a la CDF de datos reales)

Para el caso de la CDF de bytes w/r vemos en la Figura 32 que por lo general se ajustan en gran medida sin importar la configuración. Esto puede indicar que problema de ajuste de la CDF de Nº de accesos está en la las divisiones que se aplican a este eje.

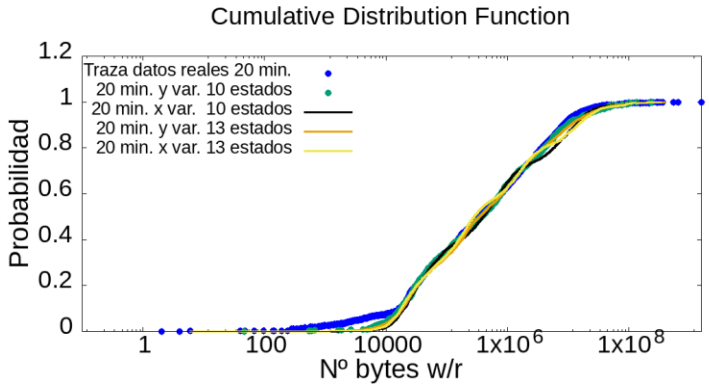


**Figura 32**. Comparación CDF Nº bytes (Línea Morada corresponde a la CDF de datos reales)

### 4.1.5 Caso ventana temporal 30 minutos

Se aumenta el tamaño de la ventana temporal por el mismo motivo del caso anterior a costa de perder granularidad en las transiciones de estados, es decir, ahora un usuario se mantendrá durante media hora en el mismo estado, cuando con las configuraciones anteriores podría estar 10 ó 20 minutos.

En este primer caso en la Figura 33, se tiene una configuración de límites en el eje Y fijos y límites en el eje X variables. Se puede apreciar que el porcentaje de intervalos temporales por estado está muy próximo a ser uniforme (próximos a 11%) y en el estado 1 se recogen los intervalos de hasta 5 accesos. Visualmente parece una buena opción de configuración.

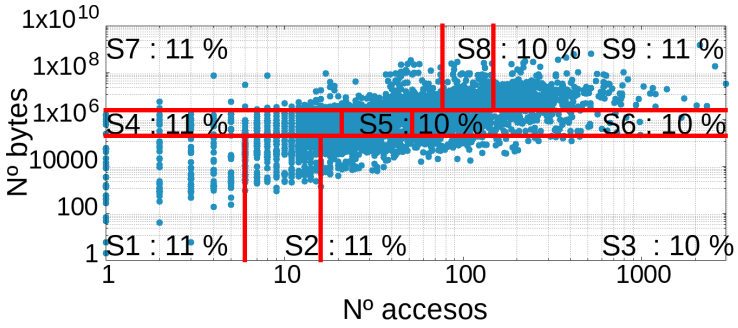


**Figura 33**. Nº bytes w/r frente a Nº de accesos para ventana de 30 minutos, lím.x var, 10 estados

La siguiente combinación está formada por 2 límites verticales fijos y límites variables horizontales (y var.). En la Figura 34 que se ha logrado un reparto homogéneo de los intervalos temporales por estado. Y además en los estados de la columna izquierda se recogen los casos de hasta 17 accesos.

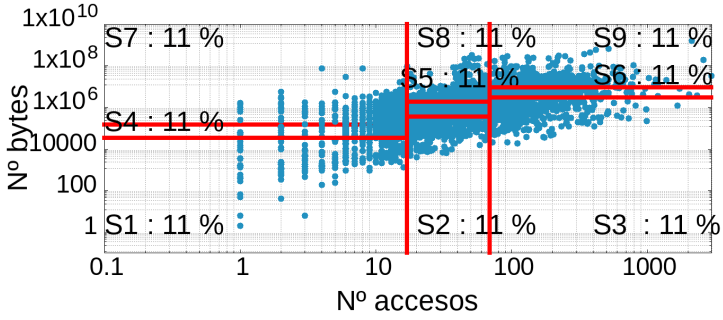


**Figura 34**. Nº bytes w/r frente a Nº de accesos para ventana de 30 minutos, lim.y var, 10 estados

Esta combinación está formada por 3 límites horizontales fijos y límites variables verticales (x var.). Se puede observar en la Figura 35 que el reparto de intervalos temporales empeora, en relación para el caso de 10 estados, sobre todo en la primera fila.

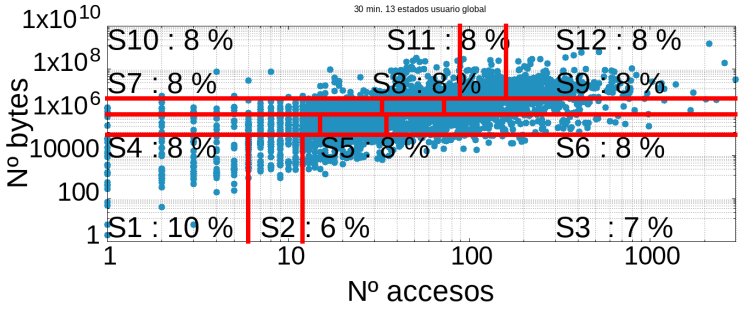


**Figura 35**. Nº bytes w/r frente a Nº de accesos para ventana de 30 minutos, lim.x var, 13 estados

Por último, tenemos el caso de 3 límites verticales y límites variables horizontales. Se aprecia en la Figura 36 que el reparto de intervalos(puntos) por estado es bastante desigual. Aun así, mejora en lo referente al estado 1, recogiendo los casos hasta 12 accesos.

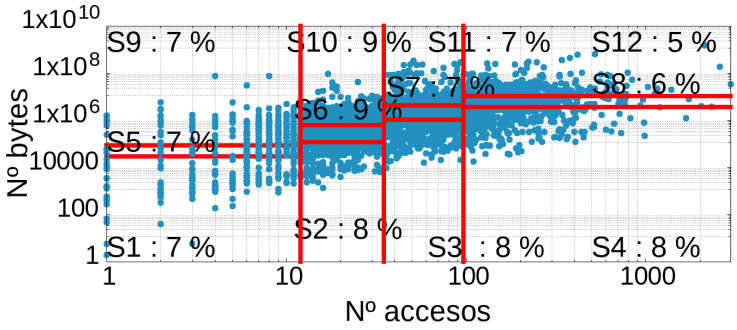


**Figura 36**. Nº bytes w/r frente a Nº de accesos para ventana de 30 minutos, lim.y var, 13 estados

En este caso se observa (ver Figura 37) que, al menos de forma gráfica, las distribuciones sintéticas parecen ajustarse más que para el caso de 20 minutos. Destaca el caso de límites variables verticales y 10 estados (línea negra). Sin embargo, el resto de casos también se acercan bastante al resultado esperado.

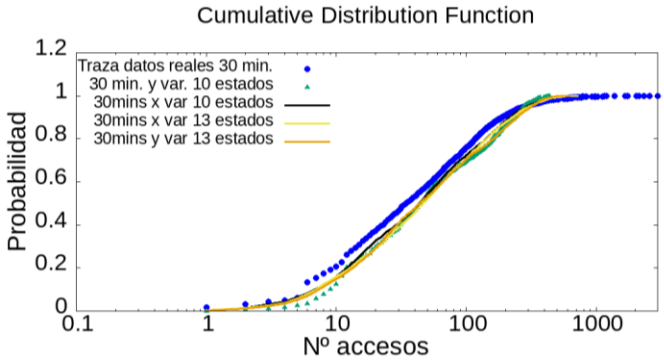


**Figura 37**. Comparación CDF Nº accesos (Línea Morada corresponde a la CDF de datos reales)

En cuanto a la gráfica de bytes w/r (ver Figura 38), vemos que se asemeja en gran medida, igual que ocurría en el caso de 20 minutos.

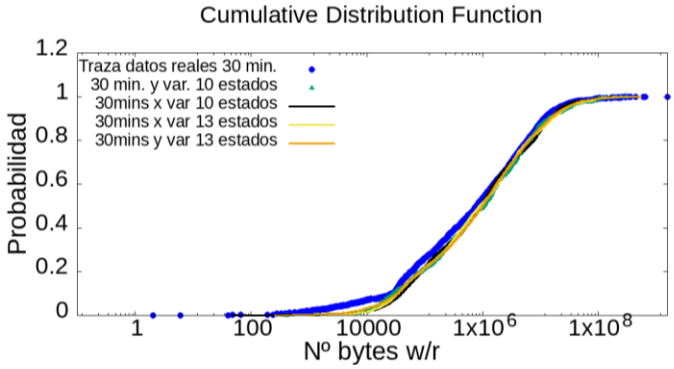


**Figura 38**. Comparación CDF bytes w/r (Línea Morada corresponde a la CDF de datos reales)

Por último, como se ha visto gráficamente para ciertas configuraciones ocurre un fenómeno de oscilaciones de la CDF del Nº de accesos, estas configuraciones están compuestas por los límites del eje Y variables (yvar) y límites en las abscisas fijos. También se ha visto que se mantienen,

aunque se aumente la ventana temporal. Por ello, con objeto de encontrar el motivo de esta anomalía se ha graficado una comparación para 10, 20 y 30 minutos.

En la Figura 39 se puede observar como las oscilaciones tienden a aplanarse conforme se incrementa la duración de la ventana temporal. También se puede apreciar que los picos de estas se van desplazando hacia la derecha conforme se aumenta la duración del ranurado.

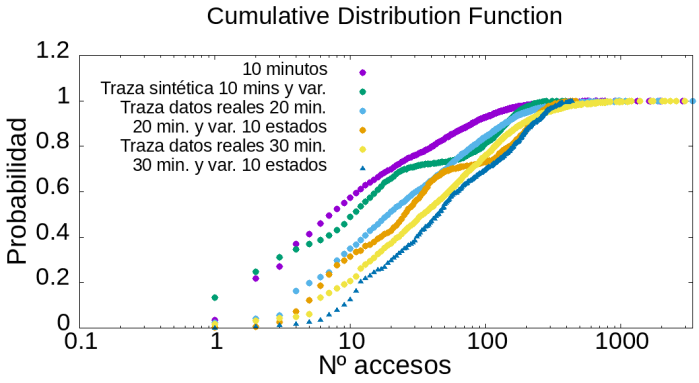


**Figura 39**. Comparativa de las oscilaciones en las CDF de Nº de accesos con oscilaciones

Si contrastamos la Figura 40 (ejes verticales) con la Figura 39 (línea naranja) vemos que los picos de estas oscilaciones parecen coincidir con los límites de los estados en el eje de número accesos. Más concretamente parecen coincidir en 10 y 44 accesos, valor de los límites en el eje X.

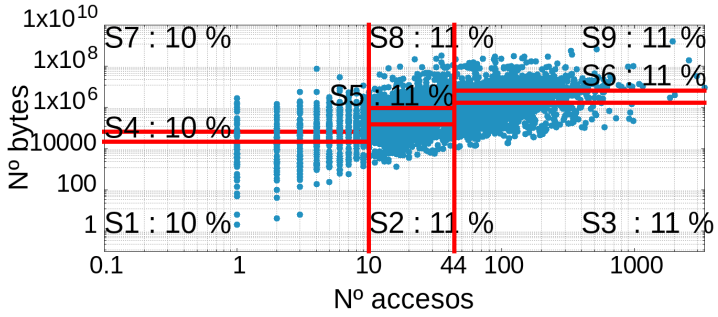


**Figura 40**. Comparativa de las oscilaciones en las CDF de Nº de accesos con oscilaciones

Para entender la posible causa de este fenómeno, es preciso recordar que el modelo estocástico está planteado en base a una máquina de estados finita. Dentro de cada estado, de duración igual a la ventana, se necesita información estadística a nivel de bytes w/r y tiempos entre llegadas por cada acceso a fichero. Estos tiempos entre llegadas se modelan de forma empírica a partir de la extracción de la CDF de los datos de entrada. Teniendo esta premisa en cuenta y también que el comportamiento de accesos a ficheros de los usuarios presenta patrones de ráfagas, se podría decir que los tiempos entre llegadas generados por este modelo no se adaptan bien a los patrones de ráfagas que presenta el usuario real. Lo que parece hacer nuestro modelo es generar aquellos tiempos que son más probables, los cuáles coincidirán con los valores de tiempos entre llegadas más numerosos extraídos de la traza real, más concretamente los tiempos inter ráfagas (tiempos

más pequeños). Por lo tanto, puede ser que el modelo esté generando más accesos de los que debería para ciertos estados aumentando el porcentaje total de estos y reduciendo el de los otros. Esto explicaría esas oscilaciones, que realmente representan una redistribución no deseada de la probabilidad de obtener ciertos casos (Nº bytes w/r, Nº accesos) de intervalos temporales.

En la **Tabla 2** se listan las configuraciones extraídas en esta sección con mayor interés de cara a modelar la carga con la mayor fidelidad posible.

| ***Ranurado*** | ***Configuración*** | ***Descripción*** |
|---|---|---|
| 20 min | x var. 10 est. | Límites verticales variables, 2 límites horizontales fijos y 10 estados |
| | y var. 10 est. | Límites horizontales variables, 2 límites verticales fijos y 10 estados |
| | x var. 13 est. | Límites verticales variables, 3 límites horizontales fijos y 13 estados |
| | x var. 13 est. | Límites horizontales variables, 3 límites verticales fijos y 13 estados |
| 30 min | x var. 10 est. | Límites verticales variables, 2 límites horizontales fijos y 10 estados |
| | y var. 10 est. | Límites horizontales variables, 2 límites verticales fijos y 10 estados |
| | x var. 13 est. | Límites verticales variables, 3 límites horizontales fijos y 13 estados |
| | x var. 13 est. | Límites horizontales variables, 3 límites verticales fijos y 13 estados |

**Tabla 2**. Descripción de las configuraciones más relevantes.

# 5 Validación modelo

Hemos visto que existen varios parámetros que conforman el modelo estocástico y que además estos se pueden ajustar a conveniencia. A continuación, se muestra cual es el grado de fidelidad con que el modelo, con las configuraciones antes expuestas, se ajusta al tráfico real.

## 5.1 Comparación de los QQ-plots para ranurado de 20 y 30 minutos

En este apartado se van a comparar las mejores configuraciones que se han obtenido en la sección anterior. Estas se corresponden con los casos con ventanas temporales de 20 y 30 minutos. El caso para 10 se descarta porque el fenómeno de las oscilaciones es bastante fuerte, y también por el problema del "estado 1" con solo un acceso que no permite variar las combinaciones de límites variables y fijos.

### 5.1.1 Validación gráfica de los QQ-plots

Como podemos ver, en la Figura 41 se representan los qq-plots para las diferentes configuraciones con una ventana de 20 minutos. Estos se han calculado a partir de las CDF de Nº de accesos ya que eran las distribuciones que más diferencias mostraban gráficamente.

En la Figura 41 se ve la misma tendencia que en las comparaciones de las distribuciones de probabilidad para el Nº de accesos. Vemos que, se mantienen las oscilaciones para los casos de límites horizontales variables (yvar, Línea verde y naranja). Así mismo, los casos para límites verticales variables(xvar) presentan un ajuste más lineal, sin embargo, no es posible extraer un candidato óptimo de manera visual.

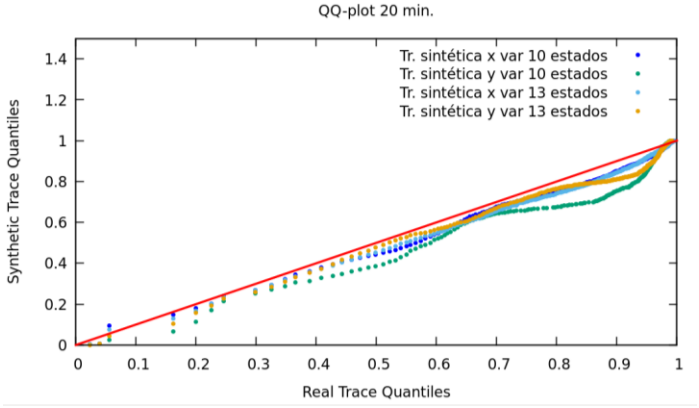


**Figura 41**. Comparación QQ-plots 20 minutos.

Por otra parte, si nos fijamos en la Figura 42, con una ranuración de 30 minutos, vemos que ocurre algo similar al caso anterior, ya que se mantienen las formas que se veían en las distribuciones de probabilidad. Tampoco es posible determinar el candidato óptimo a partir de esta representación,

aunque el ajuste parece mayor que con 20 minutos, ya que los puntos se acercan más a la recta ideal.

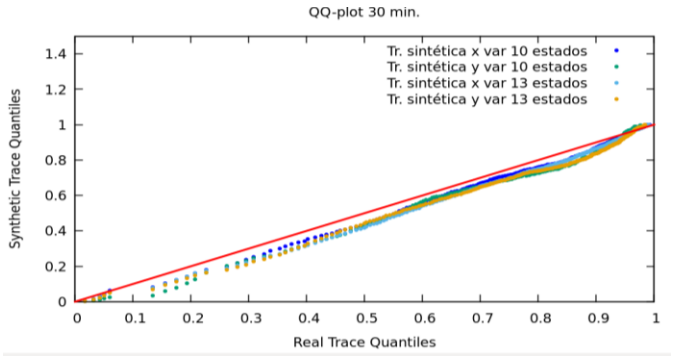


**Figura 42**. Comparación QQ-plots 30 minutos

### 5.1.2 Validación mediante el MSE extraído a partir de los QQ-plots

Para realizar una comparación con base matemática se ha extraído los errores cuadráticos medios (MSE, Mean Squared Error) a partir de las gráficas anteriores. Vemos en la Figura 43 que, para el ranurado de 20 minutos, los casos con límites verticales variables(xvar) son los que presentan menor error, seguido del caso límites horizontales variables(yvar) y 13 estados. La configuración que peor se comporta es la de límites horizontales variables y 10 estados. Esto puede deberse a esas oscilaciones tan pronunciadas que se contemplaban en la distribución de probabilidad.

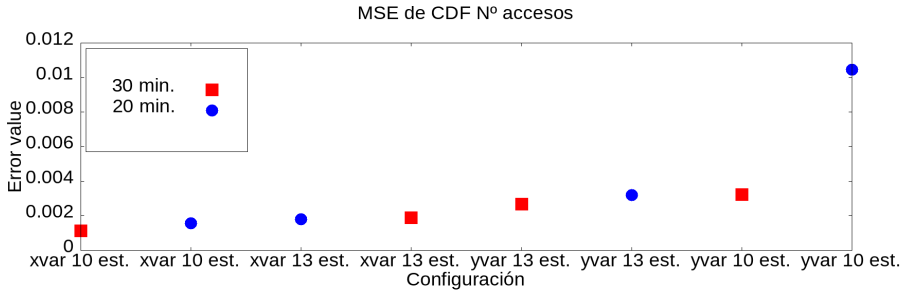


**Figura 43**. MSE para la CDF Nº accesos.

Para el ranurado de 30 minutos se observan similitudes con el caso anterior, ya que las combinaciones con límites verticales variables (xvar) son las que presentan menor error. Sin embargo, las configuraciones con límites horizontales variables (yvar) no tienen un error tan elevado como en los casos de 20 minutos. Esto seguramente se deba a que con 30 minutos el fenómeno de las oscilaciones tiende a aplanarse.

## 5.2 Análisis de la similitud entre matrices de transición reales y sintéticas

Otra forma de validar si la configuración del modelo se ajusta a lo ideal, es comprobar si las transiciones entre estados de la FSM son similares en el caso real y sintético.

### 5.2.1 Metodología para la validación mediante la matriz de transición

En la Figura 44 tenemos un ejemplo de matriz real y otro de sintética. Idealmente estas dos matrices deberían presentar los mismos valores y con ello los mismos colores.

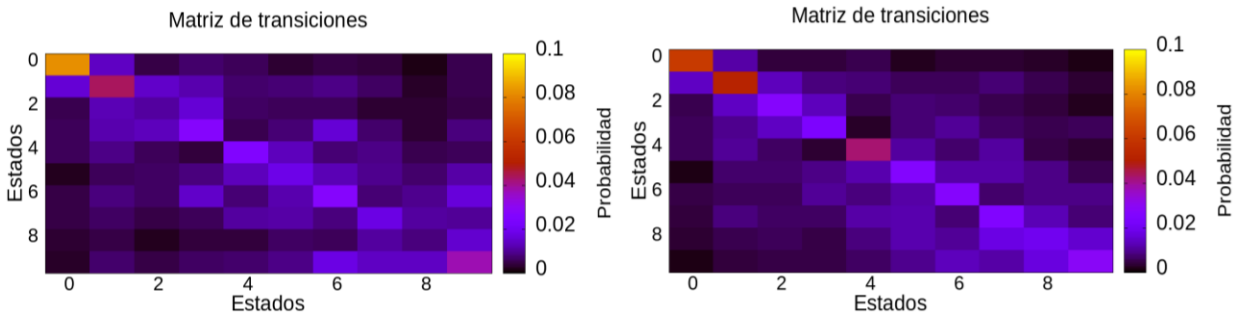


**Figura 44**. Heatmaps de matrices de transiciones de datos reales (izquierda) y sintéticos (derecha).

En la práctica esto no es así y va existir cierto error (ver Figura 45). Esta matriz de error se ha calculado a partir de la diferencia entre las matrices anteriores y posteriormente elevándolo al cuadrado. En este ejemplo vemos que para el estado 0 el error es bastante alto (esquina superior izq.). Esto significa que el generador sintético crea más transiciones entre el estado 0 y el mismo en exceso, o en otras palabras genera más intervalos temporales de los que debería que se clasificarán como estado 0.

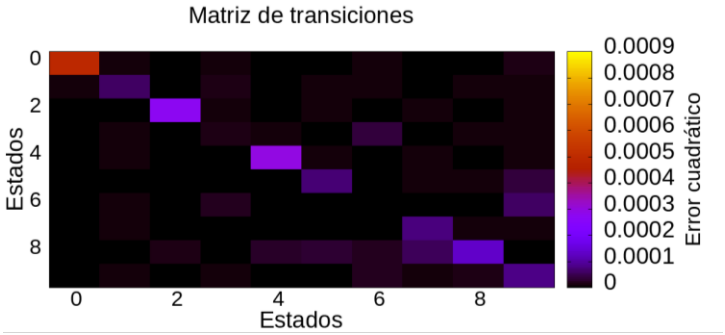


**Figura 45**. Ejemplo de Heatmap para el error cuadrático de la matriz sintética.

### 5.2.2 Matriz de error para las configuraciones con ranurado de 20 minutos

En la Figura 46.a vemos que el error se concentra sobre todo en la diagonal de la matriz. Destaca el error en la transición $0 \rightarrow 0$. Esto implica que el modelo con esta configuración no es capaz de simularlo correctamente, o bien que esté generando mal los estados aledaños y en el momento de

estudiar los resultados se vean clasificados como un estado que realmente no son. En la Figura 46.b este caso el error se dispersa un poco más, ya no está solo en la diagonal. La transición que más falla es la de 7 → 7. La razón puede ser similar al anterior caso.

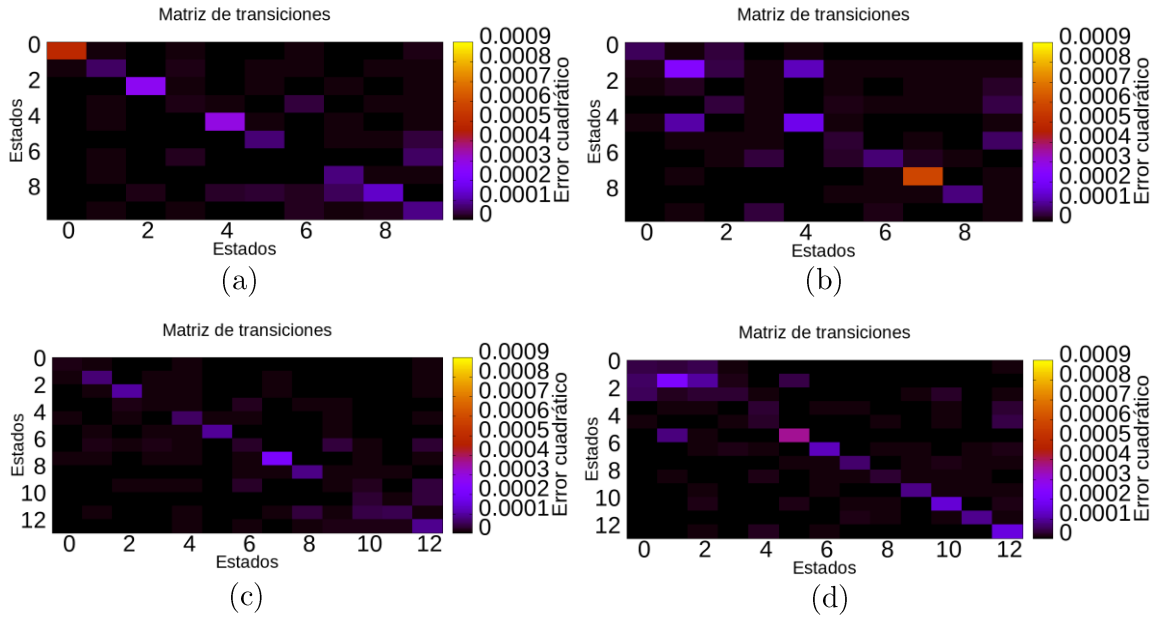


**Figura 46**. Matriz de error cuadrático para las diferentes configuraciones de 20 min. La matriz (a) corresponde a "xvar 10 est.", la (b) a "yvar 10 est.", la (c) a "xvar 13 est." y la (d) a "yvar 13 est."

Si se aumenta a 13 estados (ver Figura 46.c) se observa que el error parece repartirse más entre ellos. Aun así, sigue siendo más pronunciado en la diagonal. El motivo de esto puede estar en que en la matriz real estos valores son los más elevados, es decir, una gran parte de las transiciones se concentran en esta diagonal. Con el mismo número de estados, pero con límites variables en el eje y, vemos en la Figura 46.d que el resultado presenta bastante error en la esquina superior izquierda, entorno a la transición 1 → 1. Puede ser que esos "estados 1" generados no sean correctos y estén desplazándose hacia el estado 2 y 0, o viceversa ya que el error es cuadrático.

### 5.2.3 Matriz de error para las configuraciones con ranurado de 30 minutos

Para el primer caso (ver Figura 47.a) se puede observar que el error se mantiene a lo largo de la diagonal, con la diferencia de que el más elevado se corresponde con el estado de 9 → 9, que es un estado de gran actividad del usuario. Las razones pueden ser similares a las comentadas para el caso de 20 minutos. Si se mantiene el número de estados y se cambia los límites variables al eje y se observa (ver Figura 47.b) que el error se dispersa e incluso aumenta sobre todo en la diagonal.

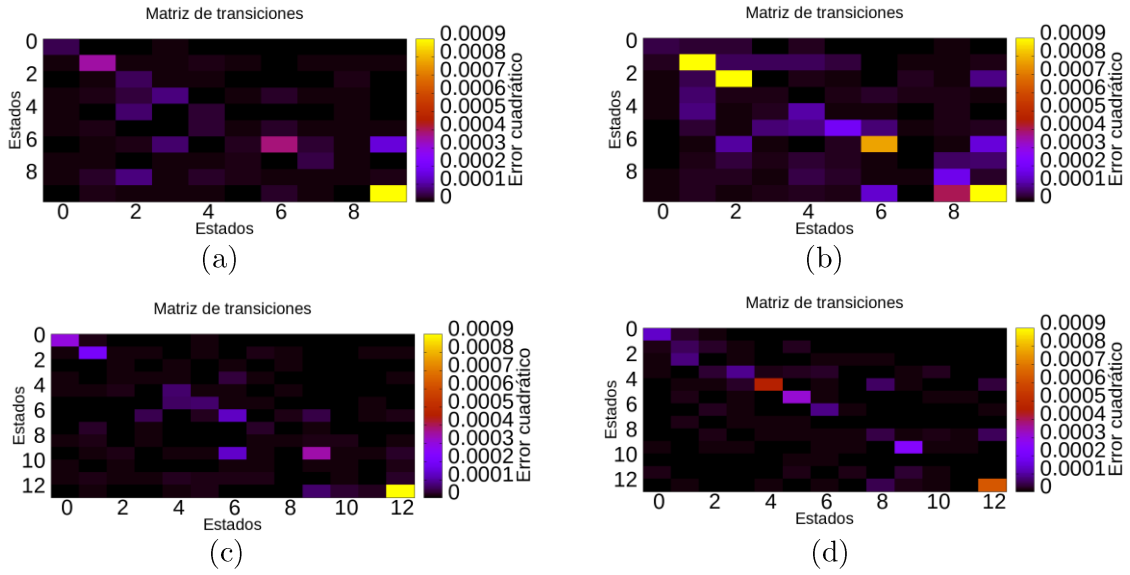


**Figura 47**. Matriz de error cuadrático para las diferentes configuraciones de 30 min. La matriz (a) corresponde a "xvar 10 est.", la (b) a "yvar 10 est.", la (c) a "xvar 13 est." y la (d) a "yvar 13 est."

Por otra parte, al aumentar el número de estados se aprecia (ver Figura 47.c) que el error disminuye en partes de la diagonal, sin embargo, ocurre los mismo que en los anteriores casos y se ve un error elevado en la transición de uno de los estados de mayor actividad ($12 \rightarrow 12$). Por último, en la Figura 47.d vemos que el error se concentra en la diagonal e incluso que es muy elevado en dos casos particulares ($12 \rightarrow 12$ y $4 \rightarrow 4$).

### 5.2.4 Validación mediante el MSE de las matrices de transiciones

En las secciones anteriores hemos visto que, dependiendo de las configuraciones de los límites, número de estados y duración del ranurado se puede atenuar el error. Aunque visualmente se podía tener algún candidato para la configuración final, se va realizar una comparación de forma matemática para así elegir la configuración que menor error presente.

Vemos en la Figura 48 que para el ranurado de 20 minutos el mejor caso es 13 estados y límites verticales variables. No obstante, el resto de casos no difieren en gran medida de este. Para el ranurado de 30 minutos el mejor caso es para 13 estados, y límites verticales variables. Las configuraciones de 10 estados muestran un elevado error para este ranurado, sobre todo para yvar que si recordamos es el caso donde ocurre el fenómeno de las oscilaciones.

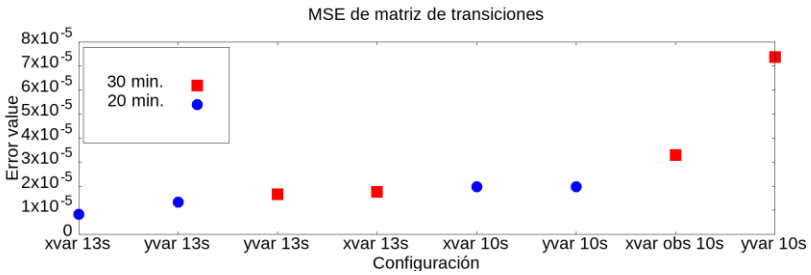


**Figura 48**. MSE de las matrices de transiciones.

Si se contrasta las dos validaciones expuestas del MSE vemos que no hay un candidato óptimo que sea el mejor caso en ambos casos, por lo que se ha de elegir el caso mediante un compromiso en el error permitido. A continuación, se repite la Figura 49 correspondiente a el MSE de los QQ-plots para mayor claridad.

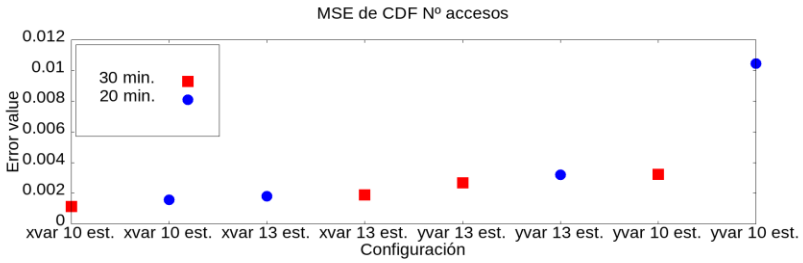


**Figura 49**. MSE de los QQ-plots (CDF Nº de accesos)

Revisando posibles combinaciones para minimizar este compromiso de error se ha decido optar la configuración siguiente:

- ✓ Ranurado: 20 minutos
- ✓ Nº estados: 13
- ✓ División estados: Límites verticales variables y 3 ejes horizontales fijos

El motivo es que con esta configuración obtenemos un MSE mínimo para la matriz de transiciones y el tercero menor para el error de las distribuciones de probabilidad (MSE QQ-plots).

En la Figura 50 se puede visualizar la configuración de límites elegida. También se observa que, aunque el porcentaje de cada estado no es uniforme (primera fila), es una de las configuraciones que mejor se comportan a la hora de generar tráfico.

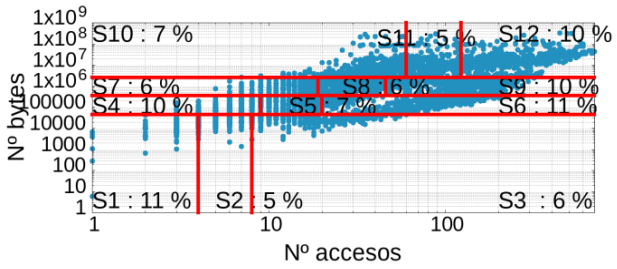


**Figura 50**. Gráfica de la configuración final.

## 5.3 Métricas de funcionamiento del generador

En esta sección se va aplicar a la herramienta de generación la configuración final elegida y se analizará su comportamiento en lo relativo a tráfico de bytes generado para poder prever las capacidades de almacenamiento necesarias para ejecutar el programa durante una jornada laboral de 8h.

### 5.3.1 Medias bytes w/r y tiempos entre llegadas

En la Figura 51 vemos que el mayor valor es de 272 s y corresponde al estado 1. El mínimo corresponde al estado 12 con un valor de 3.57 s. Estos datos concuerdan con la división de estados de la configuración elegida (ver Figura 50), ya que el estado 1 se sitúa en la esquina inferior izquierda, zona de menor actividad, y el estado 12 en la esquina superior derecha, zona de mayor actividad.

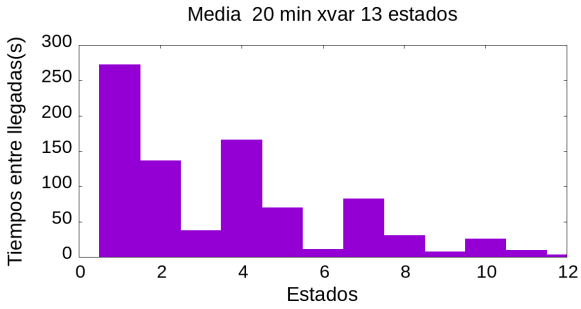


**Figura 51**. Gráfica con las medias de los tiempos entre llegadas por cada estado.

También podemos ver que la gráfica presenta un patrón de escaleras que se repite cada cierto número de estados. Si tenemos en cuenta como se han planteado los límites de los estados, se puede extraer una relación entre estos, ya que para el caso de la primera columna (estados 1,7,4 y 10) tenemos que estos coinciden con los "picos" de la escalera. Esto es porque en la primera columna se sitúan los estados con menor número de accesos, o en otras palabras con mayores tiempos entre llegadas. Similares conclusiones se pueden extraer para el resto de gráficas.
En la Figura 52 se observa una tendencia contraria a la anterior. Esto se debe a que los estados de la derecha son los de la fila superior en la gráfica de división de estados (ver Figura 50), y estos son los que más bytes transfieren.

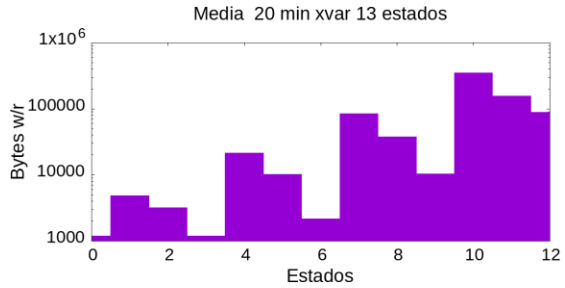


**Figura 52.** Medias para el número de bytes w/r por cada estado.

### 5.3.2 Desviación típica bytes w/r y tiempos entre llegadas

En la Figura 53 se puede ver como la dispersión de los datos incrementa conforme aumentamos de estado y también al movernos hacia estados donde su media de bytes transferidos es mayor.

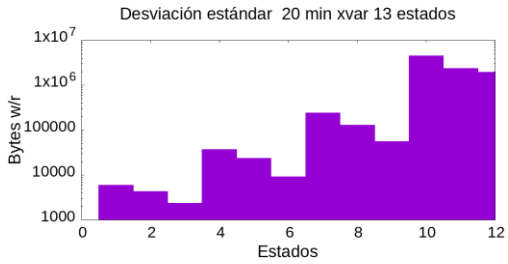


**Figura 53**. Desviaciones típicas de los bytes w/r por cada estado.

En la Figura 54 vemos que la dispersión es muy alta para los estados con pocos accesos y disminuye conforme aumenta el número de accesos por estado. Esto podría deberse a que en los estados con un gran número de accesos el tiempo entre llegadas tiende a uniformizarse. En cambio, en los estados con pocos accesos, al ser un tráfico a ráfagas, los tiempos tienden a diferir más.

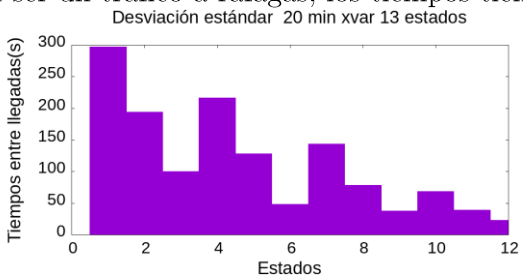


**Figura 54**. Desviaciones típicas de los tiempos entre llegadas por cada estado

### 5.3.3 Tráfico medio generado por cliente/8h

En la Figura 55 podemos ver los estados más y menos activos. Tenemos que el estado con menor actividad es el 1 y el más activo el 12. Si promediamos estos valores junto con el porcentaje de tiempo que el usuario pasa en cada estado, extraído de la matriz de transiciones, obtenemos un valor de **110 MB** de tráfico por jornada de **8 h** y por cliente. Este es el valor que nos será útil para prever qué capacidad de almacenamiento se necesita dependiendo de los clientes que se quiera ejecutar.

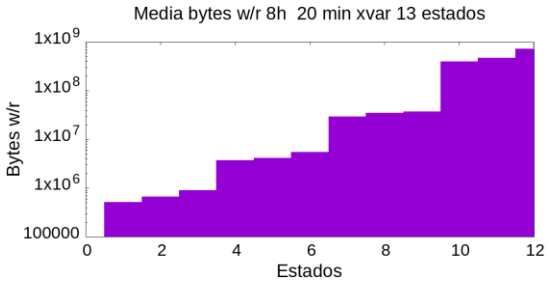


**Figura 55**. Media del número de bytes transferido por 8 horas y estado.

### 5.3.4 Porcentajes de escrituras y lecturas

Para poder generar el tráfico sintético la herramienta software necesita saber qué tipo de operaciones debe realizar, por ejemplo, solo escribir, solo leer o bien ambas intercaladas. En el caso de realizar ambas intercaladas habría que decidir en qué orden, y este orden habría que modelarlo a partir de la traza inicial. Esta problemática significaría aumentar la complejidad del modelo estocástico. Con el objetivo de simplificar el generador y evitar esta problemática se han analizado los porcentajes de w/r de la traza de partida. En la Figura 56 podemos ver los resultados de este análisis. Como se puede apreciar dominan los casos de una sola operación (barras verdes y violetas), y los casos donde se dan ambas intercaladas son algo marginal.

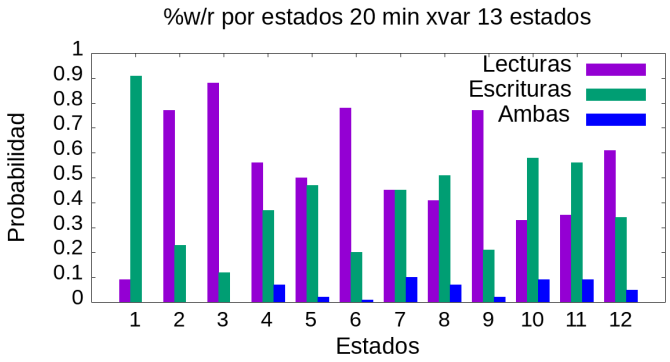


**Figura 56**. Porcentajes para solo escrituras, solo lecturas y las dos operaciones a la vez por estados (Notar "eje y" en escala logarítmica).

Si se extraen los porcentajes globales, tenemos 54 % lecturas, 42% escrituras y 4% de ambas. Como el porcentaje de que se realicen ambas operaciones intercaladas es muy bajo se decide no implementar este comportamiento. Por lo tanto, en este modelo se leerá en media un 56% y se escribirá un 44% de las veces que se acceda a un fichero. Además, teniendo en cuenta la media de tráfico por jornada,110 MB /8h por cliente, tenemos que 61.6 MB serán lecturas y 48.4 MB serán escrituras.

# 6 Conclusiones

En este documento se ha presentado un generador de tráfico sintético transparente al nivel de protocolo de compartición de ficheros en red. Este tipo de generadores se construye a partir de un modelo estocástico extraído de datos de usuarios reales, en este caso una traza SMB de los trabajadores de la Universidad Pública de Navarra. La metodología empleada en el diseño de esta herramienta se asemeja a la de varios casos encontrados en la bibliografía pertenecientes a la misma clase de generadores, pero con finalidades distintas, normalmente evaluar el rendimiento de los sistemas de almacenamiento y de ficheros. En cambio, la finalidad de esta herramienta es complementar a un detector de Ransomware para mejorar sus resultados, por lo que el foco está en que el tráfico se asemeje de forma fidedigna al comportamiento del usuario.

Por otra parte, se ha podido comprobar la flexibilidad que ofrece esta estrategia de generación en el diseño, ya que, desde su concepción, se disponen de una serie de parámetros, ajustables durante la etapa de calibrado, para lograr que el tráfico sintético se comporte de la forma más similar posible al tráfico real.

También, a pesar de las simplificaciones realizadas en la caracterización del tráfico real, (ej. Tiempos entre llegadas uniformes o solo simular una op. w/r a la vez) el modelo es capaz de generar trazas sintéticas bastante similares a las originales, aunque esta tiene mucho potencial para mejorar.

Como parte novedosa de esta herramienta, aparte de la finalidad comentada, tenemos que la forma de modelar a cada cliente es mediante contenedores, a diferencia de los artículos de la literatura que simulan cada cliente o usuario como un proceso. Gracias a esta característica se podría aumentar la similitud del tráfico sintético, ejecutando también las aplicaciones que los usuarios corren en su día a día y que pueden estar afectando al tráfico con el servidor de ficheros.

Se ha descrito sólo una primera versión de la herramienta, el objetivo es continuar con su desarrollo para aminorar el error del tráfico sintético al máximo e intentar crear una metodología de caracterización del tráfico válida para diferentes tipos de tráfico. Del mismo modo se plantea buscar otras formas de validación de la herramienta que no sean del tipo comparar con lo esperado si no someter a un sistema con las cargas sintéticas y reales y conseguir que este reaccione de la misma manera.

# A. Anexo I. Desarrollo software

En esta sección se explica la implementación software y el funcionamiento de la herramienta de generación de tráfico sintético. Esta ha sido desarrollada en base al modelo estocástico descrito en a lo largo de documento. El programa debe ser capaz de generar trazas sintéticas a nivel de protocolo de compartición de ficheros, en este caso se ha particularizado la implementación para el caso de SMB, pero está concebida para que sea muy sencillo el poder trabajar con otro protocolo como NFS.

## A.1 Diseño de la herramienta

El punto de partida de este desarrollo han sido las siguientes premisas:

- Arquitectura cliente-servidor
- El modelo estocástico es una Máquina finita de estados (FSM).
- Cada estado tiene la duración de la ventana temporal.
- Tiene que ser parametrizable.
- Tiene que generar tráfico a nivel de protocolo de compartición de ficheros, en este caso SMB.
- Debe poder simular múltiples clientes.
- Debe ser escalable.
- Fácil de configurar y de ejecutar.

Con estos requisitos en mente se ha planteado la siguiente estrategia:

- Simular los clientes y el servidor de ficheros mediante contenedores, los contenedores ofrecen más flexibilidad que si se modela un cliente como un solo proceso.
- Se lanzará un hilo por cada operación que ejecute cada cliente(contenedor), con el fin de respetar los tiempos entre llegadas.
- Usar los lenguajes Bash y Python, Bash para crear scripts de comandos en Linux, y Python porque ofrece acceso a librerías para realizar operaciones matemáticas complejas.
- Utilizar un software de terceros para generar los árboles de directorios, sobre todo para agilizar el desarrollo (Impressions).
- Utilizar la herramienta LXD/LXC de Canonical para crear y gestionar los contenedores
- La interfaz hacia el usuario será del tipo script ejecutable junto con sus parámetros (ej. ./run_generador -p1 -p2 …)

El segundo paso del desarrollo está en modelar la lógica del programa. En la Figura 57 se muestra la funcionalidad del programa. La matriz de transiciones y las CDF de bytes w/r y tiempos entre

llegadas de cada estado (círculos con línea discontinua) definen el tráfico que se va generar, por lo tanto, hay que realizar el programa de forma que estas características sean fácilmente modificables. La matriz controla la probabilidad de moverse entre estados y las distribuciones de probabilidad controlan cómo se comporta el usuario durante cada ventana temporal.

Por otra parte, la FSM genera una serie operaciones de accesos a ficheros para cada cliente. Posteriormente, este ejecutará las operaciones contra el servidor de ficheros, el cual se deberá crear al inicio del experimento con todos los ficheros necesarios para que los clientes puedan ejecutar su actividad. Además, cada cliente dispone de su propio árbol de directorios. Tanto cliente como servidor se supone pertenecen al mismo sistema NAS.

Como vemos en la Figura 57, mediante la FSM se facilita el poder implementar diferentes tipologías de usuarios, ya que se dispone de un punto de control (matriz de transiciones) que el usuario de la herramienta puede cambiar a conveniencia.

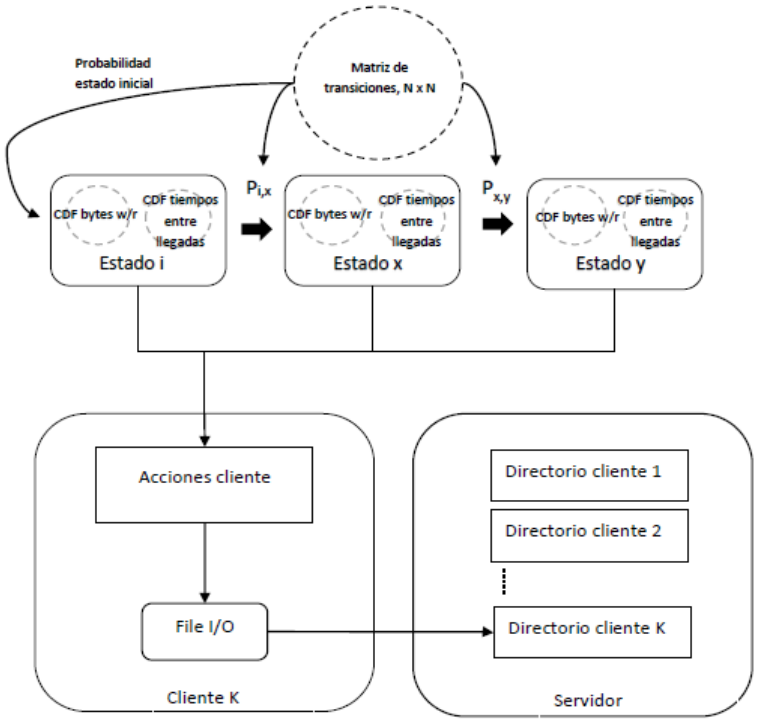


**Figura 57**. Estructura lógica del generador

Por otra parte, es necesario plantear cómo se va a distribuir esta lógica de comportamiento del generador para su implementación software. Por ello se ha planteado separar la lógica de generación de operaciones w/r de cada cliente en el PC host, la lógica de ejecución de operaciones del cliente se delega al contenedor-cliente y la estructura de ficheros es responsabilidad del contenedor-servidor. En la siguiente sección se detalla esta arquitectura.

## A.2 Arquitectura del generador

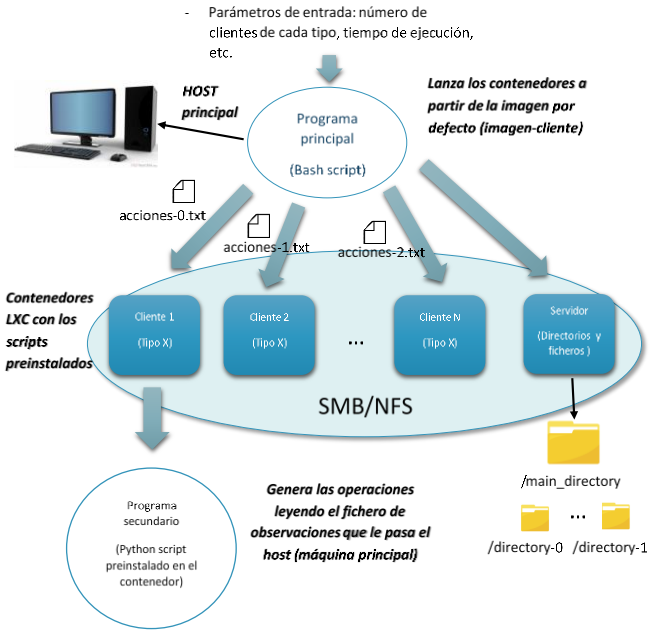


**Figura 58**. Arquitectura del generador

Como vemos en la Figura 58 hay un script principal escrito en Bash [9] que se encarga de recoger los parámetros introducidos por el usuario, crear y configurar tanto servidor como clientes y realizar el mount de directorios [11].

Además, como se comentó, el host dispone de un script en Python (módulo generador de operaciones) encargado de generar los ficheros con las acciones que deben realizar los usuarios, estos ficheros se crean al inicio y son transferidos a cada cliente.

Por otro lado, el árbol de directorios y ficheros se genera mediante un programa llamado Impressions, que es parametrizable mediante un fichero de texto. Este programa se comunica con el generador mediante una interfaz en bash que permite modificar ese fichero de texto por script y así poder ejecutar diferentes configuraciones (Nº ficheros y Path destino) de forma automatizada. Este genera los directorios y subdirectorios de forma aleatoria, siguiendo una distribución uniforme.

Una vez existe conectividad entre los diferentes nodos se lanza el script encargado de leer el fichero de acciones en los clientes para que vaya leyéndolo línea a línea y ejecutando el tipo de operación marcada (w/r) junto con la cantidad transferida de bytes correspondiente.

## A.3 Flujo de ejecución del módulo generador de observaciones

En la Figura 59 se observa el funcionamiento del script en Python encargado de generar las acciones [10], que se volcarán sobre un fichero, y que posteriormente ejecutará, en tiempo real, cada cliente.

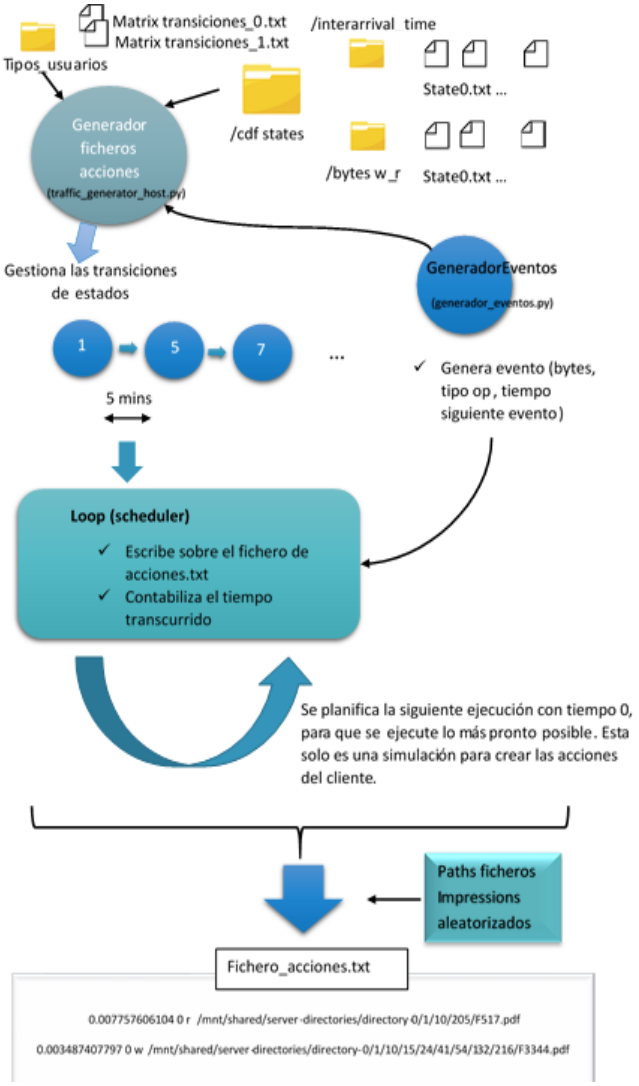


**Figura 59**. Flujo de ejecución del módulo generador de operaciones

En la Figura 59 se ve que el intervalo dura 5 min. pero es un parámetro configurable por el usuario.

El flujo es el siguiente:

1. En primer lugar, se cargan los ficheros, tales como CDF's o matrices de transiciones, estos caracterizan el modelo estadístico detallado en puntos anteriores.

2. En segundo lugar, el programa lanza el loop principal. Dentro de este se lleva un contador del tiempo transcurrido, ya que, al ser una simulación, el tiempo no es real y no se puede usar funciones propias de Python.
3. Una vez dentro del loop se pasa al estado inicial y se van generando operaciones w/r junto con el número de bytes y tiempos entre llegadas. Estas operaciones se escriben en un fichero.
4. Gracias al contador del tiempo se puede detectar cuando se cumple la duración de la ventana temporal. Una vez se detecta que ha transcurrido esta ventana se ejecuta la función get_next_state(current_state) que devuelve el siguiente estado dependiendo del estado actual. Por lo tanto, se pasará a otro estado o no dependiendo de las distribuciones de las matrices de transiciones.
5. Se repiten los pasos 3 y 4 hasta que se cumpla el tiempo de ejecución.
6. Cabe destacar que dentro de cada estado el generador usará el fichero de CDF's de bytes w/r y tiempos entre llegadas diferentes, es decir, en el estado 1 usará el fichero cdf_bytes_1.txt y en el estado 2 usará el fichero cdf_bytes_2.txt, y así sucesivamente.
7. En pocas palabras el objetivo de este módulo es obtener en pocos ms todas las acciones que un usuario generaría en el tiempo programado de ejecución.

## A.4 Imagen de los contenedores

Después de revisar la lista de imágenes disponibles para los contenedores LXC/LXD se ha decidido utilizar [8]:

- ALPINE LINUX 3.11 (2.40MB)

Esta imagen es muy ligera y se configura mediante línea de comandos. La razón de haber optado por ella está en que se pretende crear múltiples contenedores, por lo que es necesario consumir menos recursos. Con esto también se consigue que la creación de los contenedores sea más rápida. De todos modos, hay que tener en cuenta que el tamaño del contenedor será mayor ya que es necesario instalar dependencias por ejemplo las de NFS o SMB para poder montar los directorios además de otras.

Por otra parte, la idea es utilizar esta imagen como base para crear las imágenes tanto para cliente como para servidor. Esto es posible ya que mediante LXD se puede crear una imagen a partir de un contenedor ya existente que tenga ya todas las dependencias instaladas. Ha sido necesario crear las siguientes imágenes:

- 1 imagen por cada tipo de cliente
- 1 imagen para el servidor

En base a la imagen Alpine por defecto ha sido necesario instalar una serie de dependencias y scripts relativos a la herramienta de generación en cada contenedor. Según la funcionalidad del del contenedor, estas son:

a) Imagen cliente

Este contenedor que hará la función de cliente se dedica básicamente a leer un fichero y ejecutar operaciones de I/O, contra el servidor. Por ello, se debe instalar dependencias para:

- Python 3, para leer el fichero, crear hilos, y ejecutar las operaciones de I/O
- SMB/CIFS para que haya conectividad con el servidor NAS
- At, módulo de Linux para planificar ejecución de tareas

Esta es la imagen más ligera, al haber minimizado las funciones que esta debe realizar.

b) Imagen servidor

- Python 3, para leer el fichero, crear hilos, y ejecutar las operaciones de I/O
- SMB/CIFS para operar en la red NAS
- Impressions para generación de directorios
- Tshark, para capturar la traza SMB
- At, módulo de Linux para planificar ejecución de tareas

## A.5 Parámetros de ejecución de la herramienta

La herramienta se planteó inicialmente con el objetivo de ofrecer ciertos parámetros configurables con el objeto de poder generar patrones alternativos, diferentes respecto de los datos de entrada, por ello se presentan los siguientes parámetros:

- **Tamaño bloque I/O (opción -b)**, número de bytes que se escribirán o leerán por cada llamada a write() o read().(ej. 4096 bytes)

- **Tiempo de inicio de la generación (opción -s)**, marca la hora deseada para que los clientes comiencen a ejecutar operaciones I/O. (ej. 12:30 h)

- **Tiempo total de ejecución del programa (opción -t)**, por defecto son 8h (jornada laboral)

- **Duración de la ventana temporal (opción -i)**, determina el tiempo en que la FSM se mantiene en cada estado (ej. 10 minutos, 20 minutos, etc.).

- **Número y tipos de usuario (opción -u)** (ej. 10,2,5). Esto quiere que se creen 10 clientes del tipo 1, 2 clientes del tipo 2 y 5 clientes del tipo 3. En la primera versión del programa solo se implementa una tipología de usuario, pero el programa está creado para poder extenderse a más tipologías.

- **Matrices de transiciones**, se deja disponible una carpeta donde el usuario de la herramienta puede pegar sus propios ficheros de matrices, un fichero por cada tipología de usuario.

- **CDF bytes w/r y tiempos entre llegadas de cada estado,** similar al caso de las matrices de transiciones.

Hay que tener en cuenta que las matrices de transiciones y las CDF tienen una relación muy estrecha en cuanto a que el número de estados de la FSM, define el tamaño de esta matriz y además debería existir un fichero de CDF de bytes w/r y tiempos por cada estado.

Un ejemplo de ejecución de la herramienta sería:
$./run_generador -b 4096 -i 20 -t 8 -s 8:00 -u 50

Esta orden programará el lanzamiento de 50 contenedores-clientes a las 8:00 a.m con una ventana temporal de 20 minutos, tamaño de bloques para escritura/lectura de 4096 bytes y con una duración de la ejecución de 8h.

## A.6 Test retardo creación threads

Como se ha visto, cada cliente está representado por un contenedor. Dentro de cada contenedor hay un script que ejecuta las operaciones de I/O. Estas operaciones se implementan de forma que por cada una se lanza un hilo [7]. Este tiempo de creación de un hilo es dependiente del hardware. Por ende, ha sido necesario la creación de un test para contabilizar el tiempo medio de creación de hilos y compararlo con los casos de tiempos entre llegadas que se desean simular.

### A.6.1 Descripción test hilos Python

Para averiguar el tiempo de creación de hilos se ha construido un experimento (ver Figura 60) que consiste, en primer lugar, en crear varios contenedores(parámetro), de modo que convivan consumiendo recursos del hardware y así poder simular un escenario de ejecución normal. Una vez iniciados los contenedores, el contenedor objetivo del test empezará a lanzar hilos de forma continuada, tantos como se le haya pasado por parámetros, a su vez se va registrando el tiempo que tarda desde ejecutar el comando de creación de hilo hasta que se hace efectiva la primera orden de código ejecutada dentro del hilo. Esta diferencia se contabiliza para todas las creaciones de hilos y al final se hace una media de todas ellas.

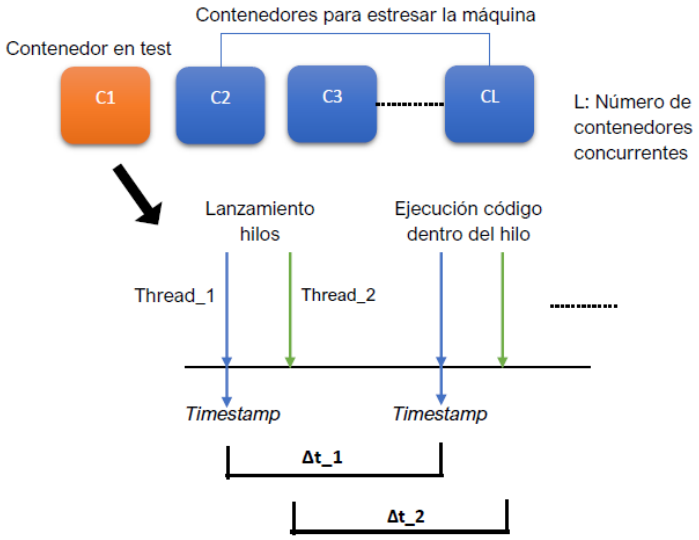


**Figura 60**. Esquema del test de creación de hilos en Python

El valor medio de creación de un hilo se calcula de la siguiente manera:

$$\mu_{thread} = \frac{\sum_{k=1}^{N} \Delta t_k}{N}$$ , donde N es el número de hilos a crear.

## A.6.2 Casos relevantes de tiempos entre llegadas

En la Figura 61 se representan los casos más críticos de tiempos entre llegadas para diferentes estados. Se puede observar que los tiempos 0.01ms, 0.1 ms y 1 ms son muy poco probables en todos los estados.

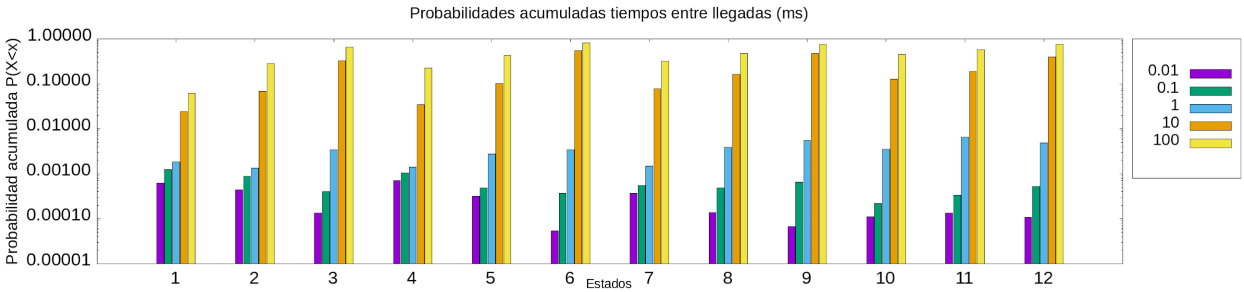


**Figura 61**. Probabilidad acumulada para casos críticos

Además, gran parte de los tiempos se concentran entre 1ms y 100 ms. En el apartado siguiente se extraerán las medias globales para estos tiempos y se compararán con el tiempo de creación de un hilo en Python.

### A.6.3 Validación implementación hilos Python

Para comparar los tiempos de creación de hilos de Python y los casos más críticos que se van a simular se ha calculado en primer lugar su promedio global, es decir, se tiene en cuenta el porcentaje que el usuario permanece en media en un estado. En la Figura 62 tenemos una representación gráfica de estos casos. Como vemos los casos 1 y 2 ms son muy poco probables, en cambio para 5 y 10 ms esta probabilidad empieza a tomar valores relevantes.

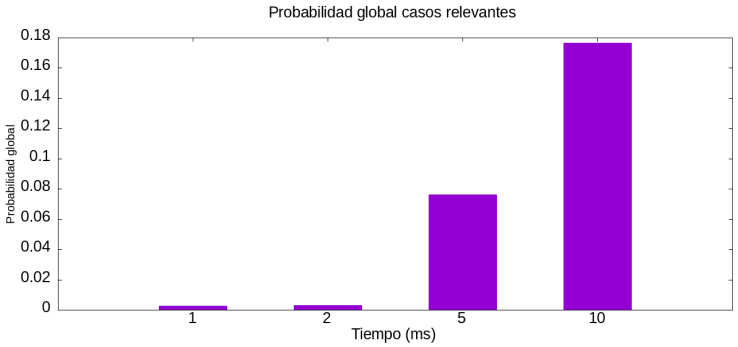


**Figura 62**. Media global para los casos 1, 2, 5 y 10 ms

En la Figura 63 se muestran los resultados del experimento descrito en la subsección anterior. Vemos que para un bajo número de contenedores concurrentes el tiempo de creación de un hilo es bastante bajo, alrededor de 0.5ms, pero si se aumenta a 50 contenedores pasa a valer 1 ms aproximadamente. El caso peor se da con 100 contenedores obteniendo un valor de 2 ms.

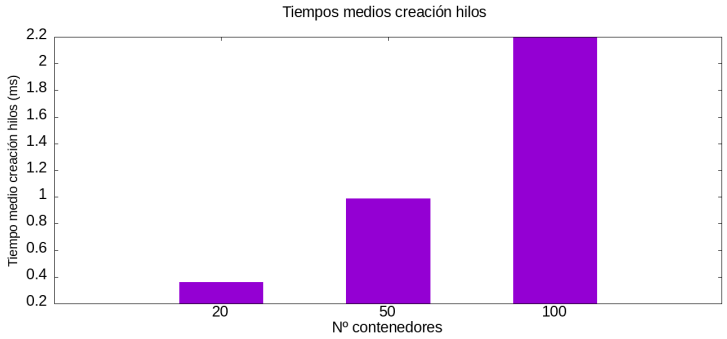


**Figura 63**. Media tiempo creación de un hilo para 20, 50 y 100 contenedores corriendo de forma concurrente

Contrastando los tiempos entre llegadas más críticos y los tiempos medios de creación de hilos se podría establecer un caso límite a partir de 100 contenedores. El motivo está en que la probabilidad de generar un valor de tiempo entre llegadas mayor de 2 ms empieza a cobrar relevancia. Con lo cual, si se lanzase más de 100 contenedores, cada cliente/contenedor no podría adaptarse al ritmo de generación de operaciones programado.

# Referencias

[1] TANG, D. 1995. Benchmarking filesystems. Tech. Rep. TR-19-95, Harvard University

[2] Darrell C. Anderson and Jeffrey S. Chase. Fstress: A flexible network file service benchmark. Technical Report CS-2002-01, Duke University, Department of Computer Science, January 2002. URL:https://www2.cs.duke.edu/ari/publications/fstress-tr.pdf

[3] Avishay Traeger, Erez Zadok, Nikolai Joukov, and Charles P. Wright. 2008. A nine year study of file system and storage benchmarking. ACM Trans. Storage 4, 2, Article 5 (May 2008), 56 pages. DOI:https://doi.org/10.1145/1367829.1367831

[4] Zhu, Ningning, Jiawu Chen, Tzi-cker Chiueh, and Daniel Ellard. "An NFS trace player for file system evaluation." (2003).

[5] Kao, W. I., & Iyer, R. K. (1992, June). A user-oriented synthetic workload generator. In [1992] Proceedings of the 12th International Conference on Distributed Computing Systems (pp. 270-277). IEEE.

[6] Robert Bodnarchuk and Richard Bunt. 1991. A synthetic workload model for a distributed system file server. In Proceedings of the 1991 ACM SIGMETRICS conference on Measurement and modeling of computer systems (SIGMETRICS '91). Association for Computing Machinery, New York, NY, USA, 50–59. DOI:https://doi.org/10.1145/107971.107978

[7] The Python Standard Library.
https://docs.python.org/3/library/, Último acceso mayo de 2020.

[8] Documentation LXD
https://linuxcontainers.org/lxd/docs/master/, Último acceso marzo de 2020.

[9] Documentation of the GNU Project
https://www.gnu.org/manual/manual.html, Último acceso junio de 2020.

[10] Numpy
https://numpy.org/, Último acceso marzo de 2020.

[11] SMB Protocol

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-cifs/5cd5747f-fe0b-40a6-89d0-d67f751f8232 , Último acceso mayo de 2020.