JITEL 2017
XIII Jornadas de Ingeniería Telemática
27-29 Septiembre 2017, Valencia (España)

Actas de las XIII Jornadas de Ingeniería Telemática (JITEL 2017), Valencia (España), 27-29 de Septiembre de 2017.

ISBN: 978-84-9048-595-8
DOI: http://dx.doi.org/10.4995/JITEL2017.2017.6581

Entorno de simulación distribuida de redes basado en la nube computacional

Sergio Serrano Iglesias, Eduardo Gómez Sánchez, Miguel L. Bote Lorenzo,
Juan I. Asensio Pérez, Manuel Rodríguez Cayetano
Departamento de Teoría de la Señal, Comunicaciones e Ingeniería Telemática
Universidad de Valladolid
ETSI de Telecomunicación, Paseo de Belén 15, 47011 Valladolid
sergio@gsic.uva.es, edugom@tel.uva.es, migbot@tel.uva.es, juaase@tel.uva.es, manrod@tel.uva.es

***Resumen*—Las simulaciones de barridos de parámetros tienen un gran potencial en el estudio de redes telemáticas, especialmente en contextos docentes. Sin embargo, el elevado tiempo necesario habitualmente para completar este tipo de simulaciones es una limitación importante para su uso. En este artículo se propone DNSE3, un entorno que permite la ejecución distribuida de tareas de simulación en el simulador ns-3 dentro de un entorno de nube computacional, a través de una arquitectura de servicios RESTful. El sistema se ha diseñado para ser autoescalable, aprovisionando y liberando dinámicamente recursos de la nube computacional en función de la carga de simulaciones demandada, y garantizando un reparto equitativo de los recursos entre los distintos usuarios. Además, DNSE3 se ha implementado reutilizando servicios presentes en *middlewares* de nube populares, y ha sido evaluado mediante pruebas sintéticas. La implementación de DNSE3 ha demostrando un correcto comportamiento funcional y un rendimiento considerablemente superior a otras alternativas cuando el número de simulaciones es muy elevado.**

***Palabras Clave*—Nube computacional, simulación, barrido de parámetros, escalado, entorno distribuido, REST**

I. INTRODUCCIÓN

La simulación es una herramienta de gran ayuda para el estudio de redes telemáticas. Ofrece un mecanismo de evaluación de protocolos y despliegues que no son fácilmente caracterizables mediante modelos analíticos. Del mismo modo, permite estudiar el comportamiento de las redes sin tener que recurrir a instalaciones reales, generalmente costosas [1], [2]. Entre los usos que se dan a estas simulaciones se encuentran la investigación y desarrollo de nuevos estándares y protocolos, así como la caracterización del comportamiento del despliegue de una infraestructura ante la llegada de diferentes patrones de tráfico [2]. Algunos de los simuladores de redes más empleados y conocidos son el ns-2 [3], muy utilizado en investigación; el ns-3 [4], siguiente versión del ns-2; OMNet++; [5] y Riverbed Modeller [6].

La simulación de redes es también una aproximación muy utilizada en entornos académicos. Gracias a las simulaciones, los alumnos pueden reforzar los conocimientos adquiridos en las sesiones teóricas mediante la replicación de los escenarios planteados. Las simulaciones, a su vez, permiten estudiar redes que no se pueden poner fácilmente en marcha con equipamiento real (por falta de recursos o potenciales problemas de seguridad); o, incluso, estudiar redes que sí están disponibles, pero en situaciones que son difíciles de reproducir en la realidad.

Para el análisis de redes es muy habitual recurrir a las denominadas simulaciones de barrido de parámetros, en las cuales algunos de sus parámetros toman valores dentro de un rango definido por el usuario para ver cómo se ve afectado el modelo de simulación. Entre los usos que se dan a este tipo de simulaciones se incluye la revisión del desempeño de la red en función de los valores de los distintos factores que entran en juego en la comunicación (velocidades binarias de enlaces, tamaños de ventana, etc.) o la optimización de dichos factores para alcanzar determinados requisitos deseables de calidad de servicio.

Normalmente la ejecución de un barrido de parámetros implica la realización de múltiples simulaciones individuales, tantas como número de combinaciones diferentes de valores de parámetros sean posibles dentro de los rangos definidos por el usuario. En un ordenador convencional, estas simulaciones se ejecutarán secuencialmente. Aunque el tiempo de ejecución de una única simulación puede ser relativamente corto, en una de barrido de parámetros éste se ve incrementado por el gran volumen de procesos necesarios para completarla, lo que alarga la espera hasta disponer de los resultados finales. Dentro del contexto educativo, este retardo puede afectar negativamente al desarrollo de las prácticas de laboratorio, que disponen de un tiempo limitado en sus sesiones de trabajo.

Dado que cada una de las simulaciones en las que se descompone un barrido de parámetros constituye una tarea individual e independiente del resto, se puede re-

ducir el tiempo requerido para su finalización mediante la ejecución de varias simulaciones en paralelo, ya sea explotando las capacidades de computación paralela de un ordenador (múltiples núcleos) o de un sistema distribuido (múltiples ordenadores) [7]. En la computación paralela en un ordenador se reparten los diferentes trabajos en hilos y procesos de ejecución diferentes, con el fin de sacar el máximo rendimiento de los recursos de los que dispone la máquina. Algunos de los simuladores de redes, como el ns-3 u OMNet++, incorporan esta técnica para la ejecución de las simulaciones. Aunque se pueden conseguir mejoras en el rendimiento, la escalabilidad de esta solución se ve limitada por el número de núcleos o procesadores instalados en el único ordenador donde se realiza la simulación [7].

Por su parte, en la computación distribuida se utilizan múltiples ordenadores para repartir los diferentes trabajos a realizar. Así, la escalabilidad no está limitada por los recursos de computación de un ordenador, ya que pueden incorporarse ordenadores adicionales. Por el contrario, los límites a la mejora del rendimiento los define la capacidad de la red y la necesidad de sincronizar los diferentes procesos, en el caso de que haya dependencias [7]. Como se ha mencionado, las simulaciones resultantes de un barrido son totalmente independientes, no existiendo este coste de sincronización. Por ello, si una simulación se completa en un tiempo T, N simulaciones podrían completarse en N ordenadores en un tiempo marginalmente superior a T. Sin embargo, y hasta donde sabemos, los simuladores de redes telemáticas existentes no hacen uso de esta técnica.

Dada esta problemática, los autores de este artículo desarrollaron el DNSE (*Distributed Network Simulation Environment*) [8], un entorno que permitía la ejecución de simulaciones de barrido del simulador ns-2 dentro de una infraestructura de *grid* computacional. Este entorno tuvo una buena acogida entre los profesores y alumnos de la E.T.S. de Ingenieros de Telecomunicación de la Universidad de Valladolid. A pesar del buen funcionamiento que mostraba, el sistema presentaba una serie de problemas ligados a la propia infraestructura de *grid* en la que funcionaba. El escalado del sistema se tenía que administrar manualmente, lo que propiciaba situaciones en las que o bien el entorno disponía de recursos aprovisionados sin utilizarse, o bien no se alcanzaban tiempos de respuesta suficientemente bajos que mejorasen la calidad de servicio. Otro problema importante era la considerable dificultad técnica y administrativa que suponía la puesta en marcha de un *grid* en el que participaran múltiples organizaciones administrativas con el objetivo de asegurar la cantidad de recursos necesarios para soportar el uso del DNSE a gran escala. Un problema adicional que existía en la época en la que se estuvo utilizando fue la falta de madurez que ofrecían los *middlewares* disponibles para el desarrollo de aplicaciones orientadas al grid.

En los últimos años ha surgido un paradigma que consigue solucionar algunos de estos problemas: la nube computacional. La nube computacional ha cobrado gran popularidad con la oferta de nubes públicas, entre las que se encuentran Google App Engine [9], Microsoft Azure [10] o Amazon Web Services (AWS) [11]. Las denominadas nubes de infraestructura como servicio (*Infrastructure as a Service*)[12] presentan una serie de características que cubren las carencias del *grid*, como son: el aprovisionamiento de recursos bajo demanda y su gestión remota, lo que permite un despliegue rápido de máquinas virtuales y simplifica su administración; y la monitorización de los servicios y recursos empleados y el escalado rápido de éstos [12]. Estas dos últimas características, junto con la capacidad de ofrecer recursos aparentemente infinitos [12], permiten realizar un seguimiento del sistema y ajustar el número de máquinas desplegadas a las necesidades computacionales. Además, gracias al interés comercial generado por este paradigma [12], el *middleware* disponible es mucho más estable y robusto, existiendo varias alternativas, tanto en la nube pública como para desplegar nubes privadas, con interfaces de servicios muchas veces compatibles.

En este artículo se presenta el DNSE3 (*Distributed Network Simulation Environment 3*), un rediseño importante inspirado en el sistema anterior, que introduce cambios significativos: por un lado, cambia la infraestructura del *grid* por la nube computacional gracias a las ventajas que aporta; por otro, aprovecha la evolución del simulador ns-2 a ns-3, con sus correspondientes mejoras de rendimiento [2]; finalmente, introduce funcionalidades detectadas como necesarias en el estudio anterior.

A lo largo de este artículo se presentarán los siguientes apartados. En primer lugar se revisarán los trabajos que es posible encontrar en la literatura con propuestas relacionadas con la realización de simulaciones de forma distribuida. A continuación, se tratará el diseño del DNSE3, discutiendo los principales requisitos y la propuesta de arquitectura resultante. La siguiente sección estará enfocada en los principales problemas que debe resolver el DNSE3 y la forma en la que se han tratado. Finalmente se discutirá sobre las conclusiones y el trabajo futuro.

II. ESTADO DEL ARTE

En la literatura se pueden encontrar diferentes proyectos que han permitido el uso distribuido de los simuladores de diferentes campos de estudio. Uno de ellos es el DNSE [8], que, según se mencionó en la sección anterior, distribuía las simulaciones de barrido de parámetros de redes telemáticas para el simulador ns-2 dentro de un *grid* computacional. Aparte de mejorar el tiempo de respuesta de las simulaciones, ofrecía una GUI (*Graphic User Interface*) para facilitar tanto la gestión de los trabajos de simulación como el visionado de las animaciones generadas a partir de la herramienta NetAnim, ofrecida de forma conjunta con el ns-2.

Otro ejemplo de despliegue en el *grid* fue DSoG (*Distributed Simulation on Grid*) [13], que facilita el estudio y modelado de motores de aviación. Esta propuesta, a diferencia de las mencionadas en esta sección, no pretende optimizar la ejecución de las simulaciones. En su lugar,

ofrece un entorno colaborativo en el que sus usuarios puedan compartir sus datos y modelos, de tal forma que se dispone de una amplia biblioteca con multitud de recursos con los que generar los modelos de simulación finales.

Si nos centramos ahora en aquellos proyectos que hagan uso de la nube, se pueden encontrar dos entornos orientados a la simulación de tráfico automovilístico: SEMSim CS (*Scalable Electro-Mobility Simulation Cloud Service*) [14], usado principalmente para el estudio del escenario en el que se reemplazasen todos los vehículos actualmente existentes por una alternativa eléctrica, y C^2SuMo (*Cloud-based, Collaborative ans Scale-up Modelling and Simulation Framework for STEM Education*) [15], un entorno educativo para estudiantes de educación superior con el que estudiar simulaciones ambientadas en el mundo real. En ambas propuestas se aprovecha la flexibilidad de la nube para mejorar el tiempo de respuesta de las simulaciones.

Aunque existen otros simuladores que hacen uso de la nube computacional, no existen propuestas que empleen la nube computacional enfocadas al estudio de las redes telemáticas y, los que sí están enfocados, utilizan otro tipo de infraestructuras. Con la propuesta de este artículo se espera cubrir esta carencia.

III. DISEÑO DEL DNSE3

En esta sección se cubre el diseño del DNSE3. En primer lugar se presentan los requisitos más importantes que debe cumplir para utilizarse en un ambiente multiusuario con acceso concurrente, para, finalmente, presentar la arquitectura empleada.

A. *Requisitos del DNSE3*

El DNSE3 es un entorno de simulación distribuida de redes para ser usado en contextos académicos. Debe satisfacer un conjunto de funcionalidades básicas para que los alumnos gestionen simulaciones de manera sencilla, y debe funcionar de manera robusta y eficiente, para ser un ayuda y no un impedimento al aprendizaje. A continuación se detallan brevemente los requisitos impuestos en el diseño del DNSE3.

En cuanto a los **requisitos funcionales**, el sistema debe:

- **Gestionar los proyectos de simulación**. Los usuarios deben ser capaces de crear proyectos de simulación, en los que se definen el modelo a emplear en las diferentes simulaciones, los parámetros soportados por el modelo y los resultados que se generan tras su ejecución. Toda esta información es necesaria para que el sistema pueda ejecutar adecuadamente las simulaciones y se puedan recuperar los resultados deseados.
- **Ejecutar simulaciones de barrido de parámetros**. Además de la ejecución de simulaciones individuales, se debe permitir a los usuarios la libre configuración de los parámetros y la selección de los resultados para el barrido.
- **Controlar la ejecución de las simulaciones**. El usuario podrá elegir, en todo momento, si quiere iniciar la ejecución de una simulación previamente creada o detenerla, en caso de que quiera modificar alguno de sus parámetros o ficheros recolectados. Los usuarios pueden confundirse en los valores proporcionados y deben ser capaces de alterar dichos valores sin tener que esperar a que se termine la ejecución completa y descartar los resultados generados. Por otro lado, en caso de detectar un fallo en la ejecución de las simulaciones, se deberá notificar al usuario, detener la ejecución de todas las tareas asociadas y esperar a que el usuario lo resuelva antes de continuar con su ejecución.
- **Recuperar los resultados**. Tras la ejecución de las simulaciones, los ficheros de resultados se deben almacenar de forma persistente, incluso después de que el usuario haya recibido una copia. En el caso de los resultados de los barridos, el sistema debe aclarar al usuario qué resultado se corresponde con qué combinación de parámetros.

Por otra parte, existen una serie de **requisitos no funcionales** que condicionan el diseño y las decisiones tecnológicas acerca del DNSE3 y que se enumerarán a continuación.

- **Escalabilidad del sistema**. El tiempo requerido para la ejecución de las simulaciones de barrido no deberá crecer de forma proporcional al número de simulaciones derivadas de estas. Por este motivo, será necesario utilizar múltiples máquinas virtuales que ejecutarán las simulaciones en paralelo. Para conseguir optimizar tanto los tiempos de respuesta como los recursos utilizados, el sistema debe ser autoescalable. Para ello, deberá contar con una cantidad de recursos suficiente y ser capaz de determinar el número de recursos necesarios para las simulaciones solicitadas, ya sea aprovisionado o liberándolos, con el menor impacto posible en el resto de funciones.
- **Reparto de recursos**. El sistema será empleado por diferentes alumnos, y se debe garantizar un reparto equitativo de los recursos disponibles entre todos ellos. Será necesario incorporar mecanismos que impidan que un alumno (de manera intencionada o no) acapare todos los recursos del sistema e impida el progreso de las tareas del resto de usuarios.
- **Tolerancia a fallos**. El sistema debe reaccionar a los fallos en la infraestructura (se pierde alguna máquina virtual) o de alguno de sus servicios (se bloquea un proceso de simulación) ocultando estos eventos al usuario final y replanificando las tareas, de manera que en todo caso la degradación sea paulatina (*graceful degradation*).
- **Clientes de acceso**. Se debe disponer de clientes, tanto en línea de comandos (orientados a pruebas o usuarios avanzados, y de los que no se hablará en este artículo) como a través de una interfaz web, que permitan el control remoto de las operaciones del sistema.

B. Arquitectura del sistema

El DNSE3 presenta una arquitectura orientada a servicios (SOA, *Service Oriented Architecture*) que propone la separación de las diferentes funciones de un sistema en diferentes servicios con una interfaz de acceso expuesta a cualquier clase de cliente. Con este tipo de arquitectura obtendremos un sistema más sencillo de construir, mantener y escalar [16]. Para las aplicaciones desplegadas en una nube de computación, este tipo de arquitectura ofrece ventajas en su administración. Una vez configurados los servicios que funcionarán en cada una de las instancias, se puede generar una instantánea que mantenga su configuración en ese momento y, cuando se produzca un fallo en alguna de las máquinas, se puede crear una nueva máquina a partir de esa instantánea. De igual manera, cuando sea necesario replicar servicios para conseguir escalar hacia arriba, sólo será necesario levantar nuevas máquinas virtuales a partir de estas imágenes.

Cada uno de los servicios que forman parte de la aplicación DNSE3 ha sido diseñado siguiendo el estilo arquitectural REST (*REpresentational State Transfer*), que propugna una arquitectura orientada a recursos (ROA, *Resource Oriented Architecture*) y un conjunto de interfaces homogéneo. De las diferentes características de REST, como son el uso de una interfaz uniforme o la navegación a través de sus recursos, la que más nos interesa ahora es la carencia de estado de peticiones previas o *statelessness*. Los servicios RESTful, según este principio, no mantendrán información relacionada con ninguna petición previa de cualquiera de sus clientes, sino que estos deberán mantener esa información. Esto facilita el equilibrado de carga, al poder dirigir las peticiones a diferentes réplicas de un servicio y, por ello, la tolerancia a fallos y la escalabilidad.

La propuesta final de la arquitectura del DNSE3 se compone de un total de siete servicios, mostrados en la figura 1, entre los que nos podemos encontrar el *servicio de orquestación*, encargado de recibir las peticiones de los usuarios y coordinar al resto del sistema mediante el envío de tareas de simulación al servicio de; el *servicio de colas*, que gestiona el reparto de tareas de simulación; el *servicio de simulación*, capaz de ejecutar las simulaciones de los usuarios; el *servicio de informe*, que se encarga de preparar los resultados de las simulaciones para que puedan ser utilizados posteriormente por los usuarios; el *servicio de almacenamiento*, que ofrece un almacenamiento persistente compartido por el resto de servicios; el *servicio de monitorización*, que recolecta métricas que muestran el uso que se hace del sistema, y el *servicio de escalado*, capaz de ajustar el número de instancias de simulación a la demanda de trabajos de los usuarios.

La propuesta final de la arquitectura del DNSE3 se compone de un total de siete servicios, mostrados en la figura 1, entre los que nos podemos encontrar el *servicio de orquestación*, encargado de recibir las peticiones de los usuarios y coordinar al resto del sistema mediante el envío de tareas de simulación al servicio de; el *servicio de colas*, que gestiona el reparto de tareas de simulación; el *servicio*

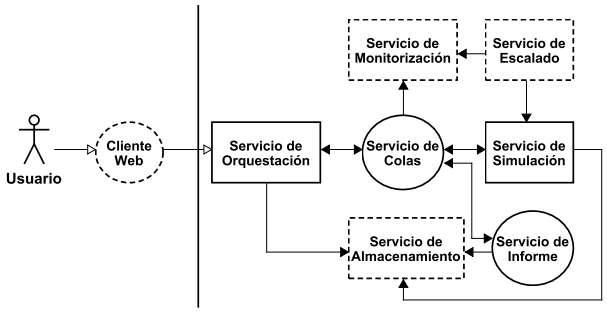

Figura 1. Servicios que componen el DNSE3 y las interacciones que establecen.

de simulación, capaz de ejecutar las simulaciones de los usuarios; el *servicio de informe*, que se encarga de preparar los resultados de las simulaciones para que puedan ser utilizados posteriormente por los usuarios; el *servicio de almacenamiento*, que ofrece un almacenamiento persistente compartido por el resto de servicios; el *servicio de monitorización*, que recolecta métricas que muestran el uso que se hace del sistema, y el *servicio de escalado*. Este último servicio es de vital importancia para el correcto desempeño de la aplicación, al ser el encargado de ajustar el número de instancias de simulación a la demanda de trabajos de los usuarios mediante la definición de una serie de reglas basadas en las métricas del servicio de monitorización.

Cada uno de los servicios previamente mencionados presentan diferentes grados de acoplamiento con la aplicación, lo que facilita su agrupación basada en la facilidad que presentan para ser integrados en otros proyectos diferentes. De esta forma, los servicios de orquestación y simulación presentan las funciones más ligadas a la aplicación (el primero gestiona el flujo de esta aplicación, el segundo envuelve e invoca al ns-3) y ofrecen mayor dificultad para ser utilizado en otros ámbitos. Por otro lado, los servicios de colas e informe presentan funciones más genéricas y fácilmente extensibles (podrían servir para, con facilidad, desplegar otro sistema paralelo y escalable y en otras infraestructuras diferentes de la nube). Finalmente, es interesante observar que los servicios de almacenamiento, monitorización y escalado son de uso habitual en diversas aplicaciones desplegadas en esta clase de entornos y son ofrecidos de forma nativa por muchos *middlewares* de nube, por lo que, a diferencia de los anteriormente mencionados, no necesitan ser desarrollados *ad-hoc* sino que se accederá a su API pública. Esta distinción de los servicios se ve reflejada en la figura 1, perteneciendo los servicios representados con un rectángulo con trazo sólido, un círculo con trazo sólido y un rectángulo con trazo discontinuo, respectivamente, a cada una de las tres agrupaciones mencionadas.

IV. IMPLEMENTACIÓN DE LOS SERVICIOS

A. *Aspectos tecnológicos*

Los entornos de nube computacional presentan una gran heterogeneidad. Por una parte, se presentan diferentes modelos de nube en función de la abstracción que nos ofrezcan de su infraestructura, como son IaaS, PaaS (*Platform as a Service*) o CaaS (*Container as a Service*), entre otros. Por otra parte, tanto las instancias desplegadas como las máquinas anfitrionas no están sujetas a un tipo de arquitectura o sistema operativo concreto. De entre los diferentes modelos de nube, se ha optado por un diseño orientado a IaaS para el DNSE3, al ofrecer un mayor nivel de gestión y configuración requeridos por algunos servicios, como sucede con el de simulación.

El despliegue se ha realizado en una nube privada que utilizaba OpenStack [18] como sistema gestor de la infraestructura. Este *middleware*, muy utilizado en nubes privadas, ofrece un entorno IaaS compatible con AWS, con el que se ofrece la posibilidad de extender a una nube híbrida, orientada a cubrir picos de demanda cuando los recursos de la nube privada no sean suficientes. Entre los servicios de OpenStack destacan Swift y Cinder, para el almacenamiento distribuido basado en objetos y volúmenes, respectivamente; Ceilometer, para la publicación de métricas; y Heat que, junto a Ceilometer, permiten gestionar las políticas de escalado del sistema. Todos estos servicios han sido integrados dentro del DNSE3 por medio de la invocación a su API REST.

Dada la libre configuración de entornos en la nube, se ha decidido desarrollar los diferentes servicios del DNSE3 en Java, al tratarse de un lenguaje multiplataforma muy popular que facilita la libre elección del entorno empleado. Además, existen diferentes *APIs* (*Application Programming Interface*) para el despliegue de servicios RESTful en Java. En el DNSE3 se ha empleado la API Restlet [17], que permite el desarrollo tanto de servidores RESTful como clientes REST, está disponible para diferentes plataformas (Java SE, Java EE, OSGI...) y ofrece un gran abanico de funcionalidades adicionales como son las implementaciones de seguridad para la integración de otros servicios.

Otro aspecto importante en una arquitectura SOA es la representación de la información intercambiada en las interfaces de servicio. Para reducir el volumen del tráfico cursado, se utilizarán documentos JSON (*JavaScript Object Notation*) para el intercambio de mensajes. Este formato de documento es mucho más liviano que otro tipo de lenguajes de marcado, como sucede en XML (*eXtensible Markup Language*). Esta simplicidad y ligereza conlleva la pérdida de la validación de la estructura del documento (que sí está disponible en XML). Este problema no supone un impacto mayor gracias a la disposición de librerías como GSON o Jackson, que permiten la conversión de los datos contenidos en el documento a objetos de clases conocidas o desarrolladas.

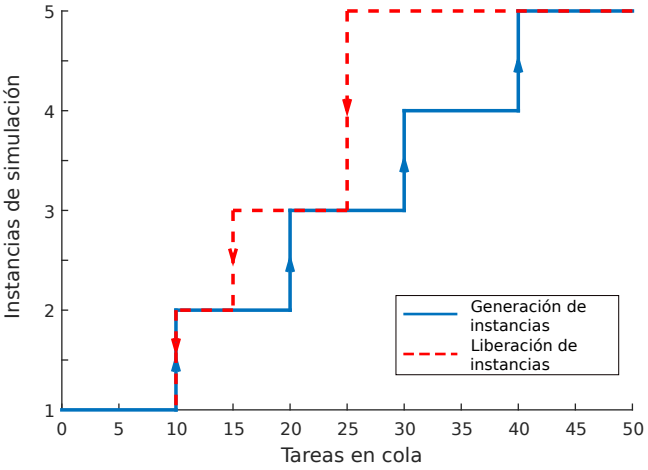

Figura 2. Representación de cómo las reglas de escalado definidas van haciendo aumentar el número de instancias disponibles según aumentan los trabajos pendientes en la cola de simulación (línea azul, progreso de izquierda a derecha), y de cómo esas mismas reglas van eliminando instancias al reducirse el número de trabajos pendientes (línea roja, progreso de derecha a izquierda).

B. *Escalado del sistema*

Esta función es la más importante del DNSE3 y la que aporta mayor valor añadido respecto a la versión anterior. El sistema debe ser capaz de acomodar el número de recursos desplegados a la demanda de trabajos de simulación solicitada por los usuarios. Para que este ajuste sea lo más eficiente posible, se debe lograr, por un lado, evitar continuas fluctuaciones en el número de instancias utilizadas y, por otro, mantener la mayor ocupación posible.

El tiempo de arranque de las máquinas no es despreciable y podría suceder que, una vez se haya lanzado una nueva máquina, se decida eliminar otra de las desplegadas al haberse reducido la demanda en ese tiempo de carga. Esta situación supondría un desperdicio de recursos de la máquina anfitriona, con un impacto negativo en el rendimiento.

Así, en lugar de eliminar las instancias tan pronto como se reduzca la carga, interesa que las instancias sólo sean eliminadas cuando la carga se reduzca considerablemente. Este sobreaprovisionamiento temporal permitirá mejorar el tiempo de respuesta y reaccionar rápidamente a nuevas subidas de carga y, en cualquier caso, desaparecerá cuando el número de tareas descienda de manera importante.

La política de escalado empleada en el DNSE3 para determinar cuándo y cómo se efectúa este escalado y que sea capaz de cumplir los requisitos anteriores se basa en el número de trabajos que debe efectuar cada una de las instancias de simulación, tal y como se ilustra en la figura 2. Hasta que esta proporción no supera un determinado umbral (en nuestro caso 10 simulaciones por instancia), no se aprovisionarán nuevos recursos de simulación. En el caso contrario, mientras no se reduzca la carga de cada instancia al 50 % del umbral anterior (5 simulaciones por instancia), se mantendrá el número de recursos desplegados. Una vez se reduzca, se liberan la

mitad de las instancias desplegadas.

Una ventaja de esta política es su sencillez de implementación, Los servicios de escalado de los entornos de nube permiten incluir políticas basadas en reglas *IF-THEN*, en las que se comparan los valores almacenados en el servicio de monitorización con un umbral previamente fijado, o bien la agrupación de éstas para generar reglas más complejas. En nuestro caso sólo se necesitan 2 reglas de comparación: si $\frac{tareas}{instancias} > 10$, entonces $instancias++$; y, si $\frac{tareas}{instancias} < 5$, entonces $instancias = 0{,}5 \times instancias$, redondeando al entero mayor. En las reglas anteriores el término $tarea$ agrupa tanto a las simulaciones pendientes de ejecutar como las que están siendo procesadas por las diferentes instancias de simulación. Para que estas métricas puedan aplicarse, el servicio de colas deberá enviar a Ceilometer una métrica que contenga el reparto equitativo de las tareas solicitadas (dato conocido al contener todo el listado de trabajos) entre las instancias de simulación (que deberá de solicitar de forma periódica a Heat).

Cuando la carga de trabajo decrece a los niveles indicados por la política de escalado, se procede a la eliminación de las instancias de simulación sobrantes. En el caso de que estas máquinas estén ejecutando alguna simulación, se intentará esperar a que ésta se complete sin solicitar un nuevo trabajo antes de eliminarla, aunque puede darse la situación de que se interrumpa dicho trabajo. En este último escenario, los trabajos asociados volverían a estar disponibles una vez haya expirado su tiempo de reserva al no recibir actualizaciones del servicio de simulación.

C. *Asignación de trabajos de simulación*

Un aspecto fuertemente ligado al escalado es el lugar en el que se realiza la planificación de recursos. Si el servicio de colas actuase de *scheduler* centralizado, debería conocer qué servicios de simulación están disponibles, cuáles se están apagando, si se están levantando nuevos, y sus URI de acceso. Como se puede prever, esto supone un aumento de complejidad en el servicio.

En su lugar, se ha seguido un modelo de asignación *work-stealling*, en el que, en lugar de asignar los trabajos a los diferentes trabajadores, son estos los que solicitan el trabajo que deben realizar. Cuando se inicia una nueva instancia, ésta conoce de antemano la dirección de acceso al servicio de colas para poder preguntar por el siguiente trabajo a realizar. El servicio de simulación está siempre intentando trabajar, así que o está ocupado, o demanda nuevos trabajos al servicio de colas. Además, periódicamente informa al servicio de colas de su actividad, de manera que cuando deja de comunicarse con él, el servicio de colas puede asumir que ha caído. Como en los servicios RESTful no se mantiene información de las conexiones previas, este diseño permite que el servicio de colas sólo se tenga que preocupar de mantener la cola de trabajos y determinar el trabajo siguiente a procesar, tareas que no dependen del número de instancias de simulación que haya, lo que simplifica enormemente la escalabilidad.

En la determinación de la siguiente tarea a ejecutar se realiza una rotación (*round-robin*) de dos niveles entre los trabajos pendientes, ilustrada en la figura 3, con el fin de evitar tanto el acaparamiento del sistema por parte de un usuario (de forma intencionada o no) como la recolección de un mismo trabajo por parte de dos instancias diferentes. El primer nivel de rotación va alternando de usuario propietario de los trabajos, de forma que dos trabajos solicitados consecutivos no pueden pertenecer a un mismo usuario (salvo que no haya trabajos pendientes de otro usuario), mientras que el segundo nivel recorre los trabajos pendientes de ejecutar del usuario escogido en el nivel anterior. Dentro de los trabajos pendientes se incluyen aquellos no asignados y los asignados anteriormente a un servicio que ha caído (que ha dejado de reportar su actividad).

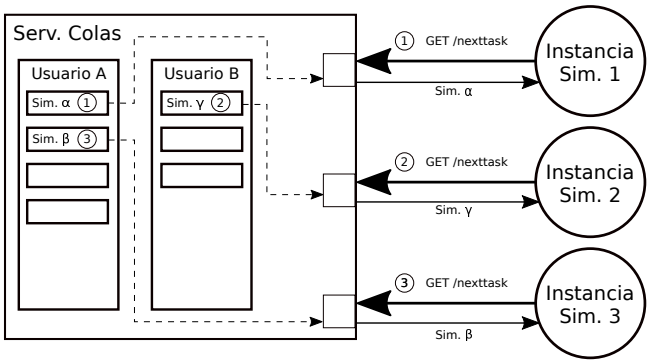

Figura 3. Ilustración de la asignación del trabajo siguiente mediante la rotación *round-robin* de los usuarios propietarios de los trabajos.

V. INTERFAZ DE USUARIO

Aunque los servicios RESTful, gracias a su interfaz uniforme, que en el caso de los servicios web se corresponde con los métodos de del protocolo HTTP (*Hyper-Text Transfer Protocol*), permiten usar cualquier cliente web (por ejemplo, un navegador) para ser invocados, es complejo que el usuario final prepare y consuma los cuerpos de peticiones y respuestas. Es por ello que se ha optado por desarrollar un cliente del DNSE3 que es una aplicación web con una interfaz de usuario ilustrada en la figura 4. Este cliente se encarga de recibir y procesar las peticiones de los usuarios finales, adecuándolas a la interfaz REST del servicio de orquestación y de reproducir las respuestas recibidas en representaciones HTML visualmente agradables para devolver al navegador. Para que esta aplicación sea usable desde cualquier navegador web y tipo de dispositivo, se ha desarrollado usando el *framework* Bootstrap [19], que ofrece soporte con los dispositivos móviles como *smartphones* o tabletas, muy populares entre los alumnos en los últimos años.

Adicionalmente, la seguridad se puede gestionar de manera delegada: el servicio de orquestación acepta todas las peticiones provenientes del cliente web, mientras que éste es el encargado de validar las credenciales de los usuarios. Como no se aceptan en el servicio de orquestación peticiones de otros orígenes, resulta más sencillo proteger al DNSE3 de ataques externos.

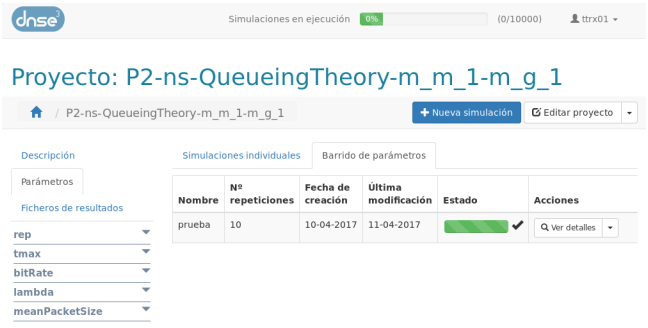

Figura 4. Intrefaz web del DNSE3

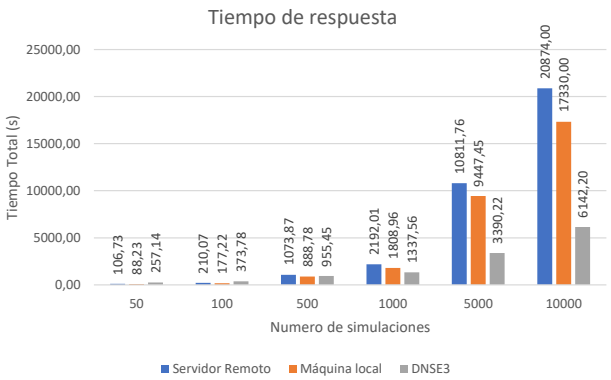

Figura 5. Tiempos de respuesta ofrecidos por cada uno de los sistemas evaluados ante diferentes tamaños de simulaciones a ejecutar

VI. EVALUACIÓN DEL RENDIMIENTO

Una vez se ha completado el desarrollo de una versión de pruebas del sistema, se ha evaluado si podría ser desplegada en un entorno de producción. Las pruebas realizadas incluyen el acceso concurrente de varios usuarios, que no debe comprometer la calidad de servicio (QoS, *Quality of Service*) del resto de usuarios; la publicación de simulaciones; la recolección de los resultados; y el reparto de los recursos según se comentó en la sección C.

A falta de una evaluación con usuarios reales, en lo que respecta a temas de rendimiento, se han efectuado una serie de pruebas sintéticas para evaluar el escalado automático de las instancias de simulación. Estas pruebas han consistido en la ejecución de un barrido de parámetros sintético consistente en varias simulaciones de duración más o menos semejante. Estas pruebas se han repetido en tres sistemas: el servidor del laboratorio de la asignatura (ordenador con procesador Intel(R) Xeon(R) E5-2620 de 4 núcleos a 2 GHz y 8GB de memoria RAM), que los alumnos podrían usar fuera del horario de prácticas; los ordenadores del laboratorio (ordenadores con procesador Intel(R) Core(TM) i5-2400 de 4 núcleos a 3,10 GHz y 4 GB de memoria RAM), sólo accesibles en horario lectivo; y el DNSE3. En los dos primeros casos se han buscado momentos en los que no había otros usuarios corriendo procesos en el sistema. En el caso del DNSE3 se partía de una situación de reposo (un único servicio de simulación creado) en cada experimento, y el servicio de escalado podría llegar a crear hasta un máximo de 15 instancias simultáneas de servicios de simulación.

La figura 5 muestra los tiempos requeridos para la ejecución completa de todas las simulaciones, para barridos de parámetros sintéticos consistentes en un distinto número de trabajos. El tiempo típico para completar una simulación individual ha rondado los 2-3 segundos tanto en las máquinas locales como en el servidor del laboratorio, mientras que en el DNSE3 este tiempo se incrementaba hasta los 7 segundos. Este incremento se debe a las comunicaciones realizadas entre los diferentes servicios: la publicación de la tarea en el servicio de colas, la recogida de la tarea por parte del servicio de simulación, la obtención del modelo de simulación almacenado en el servicio de almacenamiento y la posterior carga de los resultados y la notificación del estado de ejecución al servicio de orquestación.

En vista a los resultados obtenidos, se pueden distinguir dos situaciones diferentes. En primer lugar, si el número de simulaciones a ejecutar es bajo, en barridos que den lugar hasta unas 500 simulaciones, el DNSE3 no presenta una ventaja comparativa: por una parte, el número de simulaciones es bastante bajo y puede completarse secuencialmente en un tiempo pequeño. Aunque el servicio de escalado aprovisiona nuevas instancias, estas tardan en arrancar unos 75 segundos, por lo que llegan a asumir pocas tareas, obteniéndose un beneficio muy limitado de la paralelización. Por otra parte, a medida que aumenta el número de simulaciones totales a ejecutar, las nuevas instancias levantadas por el servicio de escalado llegan con todavía muchos servicios pendientes de procesar en la cola, así que en un momento dado se están realizando un gran número de ejecuciones en paralelo, compensando sobradamente la sobrecarga de la distribución, llegándose a alcanzar una reducción del 65 % del tiempo de respuesta con barridos de 5000 simulaciones.

Aún así, como en el entorno de pruebas se ha restringido el número de máximo de servicios de simulación, a partir de un punto el tiempo de respuesta se incrementa de forma lineal. Para evitar esta situación, se puede hacer uso de una nube híbrida y así conseguir recursos virtualmente infinitos.

VII. CONCLUSIONES Y TRABAJO FUTURO

El uso de los simuladores de redes telemáticas es beneficioso para la investigación y análisis de despliegues de infraestructura y protocolos de comunicaciones. Aun así, las simulaciones de barrido de parámetros tienen un coste computacional excesivo, lo que limita su utilización especialmente en entornos educativos, con recursos computaciones y tiempo escasos.

Para tratar este problema, este artículo ha propuesto el DNSE3, una aplicación distribuida orientada a servicios que, aprovechando las ventajas de la nube computacional, introduce mecanismos de autoescalado para aprovisionar recursos dinámicamente en función del número de simulaciones demandadas por los alumnos. Su diseño también ha tenido en cuenta otros aspectos relevantes, como el reparto

equitativo de los recursos entre usuarios y la tolerancia a fallos.

Para ilustrar su funcionamiento, se ha programado un prototipo de la aplicación con una interfaz de usuario web. Posteriormente, se ha evaluado el rendimiento del escalado sometiendo al DNSE3 a barridos de simulación sintéticos de distintos tamaños, observándose que cuando el número de simulaciones individuales es bastante elevado el tiempo de respuesta obtenido por el DNSE3 es hasta un 65 % menor que el alcanzado con las otras alternativas disponibles en la ETSIT de Valladolid. Además, se ha observado que el comportamiento era funcionalmente correcto, y que se lanzaban y detenían servicios de simulación en función del número de tareas en la cola, siguiendo las reglas de escalado definidas.

Aun así, las pruebas realizadas hasta ahora se han hecho en situaciones controladas por los autores de este trabajo. Es, por lo tanto, necesario repetir este estudio con usuarios reales (alumnos) en un contexto real (muchos alumnos compitiendo simultáneamente por los recursos). Este nuevo estudio permitirá validar los resultados obtenidos anteriormente y, además, recoger el patrón de peticiones de simulación llevado a cabo por los usuarios reales, lo que a su vez facilitará la propuesta de nuevos experimentos sintéticos más realistas en el futuro. También se aprovecharán estas pruebas para conocer la opinión de los usuarios acerca de aspectos de usabilidad e interfaz de usuario, así como para indagar en el impacto que el uso de esta aplicación tiene en sus procesos de aprendizaje.

Otra vía de trabajo futuro se refiere a la exploración de alternativas a las políticas de escalado descritas en la sección B, para estudiar el coste/beneficio de utilizar políticas más agresivas o el mantenimiento de un número mínimo de instancias de simulación permanentemente disponibles.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por los proyecto TIN2014-53199-C3-2-R del Ministerio de Economía y Competitividad, y VA082U16 de la Junta de Castilla y León, con cofinanciación FEDER. Los autores agradecen al resto del grupo de investigación GSIC/EMIC por su apoyo e interés en el proyecto.

REFERENCIAS

[1] A. M. Law and W. D. Kelton, *Simulation modeling and analysis*. McGraw-Hill, 1991.

[2] E. Weingartner, H. vom Lehn, and K. Wehrle, "A Performance Comparison of Recent Network Simulators," in *2009 IEEE International Conference on Communications*, June 2009, pp. 1–5.

[3] "The network simulator - ns-2," 2017. [Online]. Available: http://www.isi.edu/nsnam/ns/

[4] "The network simulator - ns-3," 2015. [Online]. Available: https://www.nsnam.org/

[5] "Omnet++ discrete event simulator," 2017. [Online]. Available: https://omnetpp.org/

[6] *Riverbed Modeler*, Riverbed Technology, 2015. [Online]. Available: https://www.riverbed.com/es/products/steelcentral/steelcentral-riverbed-modeler.html

[7] R. M. Fujimoto, "Research Challenges in Parallel and Distributed Simulation," *ACM Transactions on Modeling and Computer Simulation*, 2016.

[8] M. L. Bote-Lorenzo, J. I. Asensio-Pérez, E. Gómez-Sánchez, G. Vega-Gorgojo, and C. Alario-Hoyos, "A grid service-based Distributed Network Simulation Environment for computer networks education," *Computer Applications in Engineering Education*, 2010.

[9] *Google App Engine*, Alphabet Inc., 2017. [Online]. Available: https://cloud.google.com/appengine/?hl=es

[10] *Microsoft Azure*, Microsoft Corporation, 2017. [Online]. Available: https://azure.microsoft.com/es-es/

[11] *Amazon Web Servicies*, Amazon Inc., 2017. [Online]. Available: https://aws.amazon.com/es/

[12] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," *Communications of the ACM*, 2010.

[13] Y. Cao, X. Jin, and Z. Li, "A distributed simulation system and its application," *Simulation Modelling Practice and Theory*, 2007.

[14] D. Zehe, A. Knoll, W. Cai, and H. Aydt, "SEMSim Cloud Service: Large-scale urban systems simulation in the cloud," *Simulation Modelling Practice and Theory*, 2015.

[15] F. Caglar, S. Shekhar, A. Gokhale, S. Basu, T. Rafi, J. Kinnebrew, and G. Biswas, "Cloud-hosted simulation-as-a-service for high school STEM education," *Simulation Modelling Practice and Theory*, 2015.

[16] S. Vinoski, "REST Eye for the SOA Guy," *IEEE Internet Computing*, 2007.

[17] *Restlet Framework*, Restlet, Inc., 2017. [Online]. Available: https://restlet.com/open-source/

[18] *OpenStack*, OpenStack Foundation, 2017. [Online]. Available: https://www.openstack.org/

[19] "Bootstrap," 2017. [Online]. Available: http://getbootstrap.com/