9

Capítulo 9

Simulación de comportamientos de sistemas distribuidos para obtener robustez y auto-recuperación de fallas

Capítulo 9

Simulación de Comportamientos de Sistemas Distribuidos para obtener robustez y Auto-recuperación de fallas

Arles Rodríguez y Jonatan Gómez

Resumen

Este capítulo muestra algunos aspectos fundamentales para definir una simulación de comportamientos de sistemas distribuidos para lograr que un sistema sea robusto y se recupere a sí mismo de fallas. Se determina como tarea de interés la colección y la sincronización de datos en nodos conectados en una red y se mencionan algunos subproblemas de interés relacionados con la definición de las fallas y la obtención de las propiedades de robustez y recuperación de fallas. Se utilizaron sistemas multi-agente para diseñar la simulación porque permiten modelar componentes autónomos mediante el uso de un programa de agente. Se definieron dos funciones principales: agentes móviles que realizan tareas de sincronización de datos y nodos que constituyen agentes estáticos donde se almacena o se replica la información. Partiendo de dicha definición, se muestran diferentes problemas abordados y sus soluciones utilizando la simulación propuesta. Algunos resultados publicados en la literatura se presentan con su respectiva referencia y enlaces a su código fuente. Los resultados permiten determinar aspectos clave en la determinación de un sistema más robusto y permiten recuperar partes clave del sistema para completar una tarea distribuida de colección y sincronización de datos.

Palabras clave: sistemas distribuidos, sistemas multi-agente, modelo, fallas, robustez, auto-recuperación, control descentralizado.

Introducción

La simulación de comportamientos de sistemas distribuidos es una forma de probar y validar diferentes algoritmos distribuidos, aplicaciones o configuraciones independientemente de los recursos computacionales o de red disponibles. En algunos casos, los sistemas reales no están disponibles para realizar experimentación y los experimentos pueden no ser reproducibles debido a la gran variedad de implementaciones de *software* y *hardware* disponibles (Velho y Legrand, 2009). Mediante la realización de simulaciones se espera ofrecer un balance entre modelos teóricos complejos que pueden ser irrealistas o difíciles de implementar y el tiempo y recursos empleados para configurar una plataforma real (Casanova, Giersch, Legrand, Quinson y Suter, 2014).

Los principales desafíos al simular un sistema distribuido son lograr escalabilidad y precisión (Casanova, Legrand y Quinson, 2008). Una simulación es escalable si permite realizar modelos de simulación validados previamente y si permite modelar diferentes topologías de red. Además, es escalable si permite la adición y eliminación de elementos del sistema distribuido con precisión y de una forma rápida. La precisión hace referencia a la obtención de resultados reales y puede medirse dependiendo del propósito de la simulación; por ejemplo, tratando de predecir el tiempo para completar las comunicaciones entre los enlaces o estimar el ancho de banda de muchos flujos que compiten en una topología de red aleatoria (Fujiwara y Casanova, 2007). Se requiere un balance entre la rapidez de la simulación y la precisión para tener una simulación de calidad (Casanova *et al.*, 2008).

Existen diferentes modelos de simulación que se enfocan en estudiar distintos aspectos de un sistema distribuido: procesador, red y almacenamiento. En el procesador se tiene la simulación de tiempos en la unidad de procesamiento central (CPU), donde se determina el costo computacional de una forma analítica (Simgrid, 2012). En la red se estima la latencia en segundos para enviar datos y el ancho de banda en bytes. Se tienen simuladores de los protocolos de red que simulan todas las capas de red y son muy precisos, pero que pueden ser lentos al simular sistemas a gran escala, pues se utilizan simulaciones en tiempo discreto (Bhardwaj y Dixit, 2010; Rana, Sardar, Mandal y Saha, 2017). También se tienen simuladores que estiman de una forma matemática el flujo de datos en una red siendo más rápidos que aquellos basados en simulación de eventos discretos (Velho y Legrand, 2009). En modelos de almacenamiento se tiene la simulación de

dispositivos utilizando diferentes configuraciones de sistemas de archivos y componentes de *hardware* como buses, tipos de partición o almacenamiento (por ej., RAID, SSD, etc.) (Agrawal *et al.*, 2008).

En este capítulo se mostrará una simulación de un sistema distribuido diferente a los anteriores utilizando una aproximación basada en del sistemas multi-agente. Los simuladores anteriormente descritos no modelan comportamientos de autoorganización. El concepto de autoorganización toma inspiración de la naturaleza (por ej., los insectos sociales: hormigas, abejas o termitas). A partir de la definición de ciertas reglas se puede reproducir, por ejemplo, el comportamiento de las termitas y obtener un algoritmo que explore un espacio de forma eficiente. Mediante la autoorganización se espera generar sistemas que logren ciertas metas globales a partir de interacciones locales (Bicocchi y Zambonelli, 2007). El principal desafío para obtener autoorganización consiste en identificar las reglas o procesos que producen el comportamiento global deseado (Lalanda, McCann y Diaconescu, 2013). En ese sentido, los sistemas multi-agente definen componentes con la capacidad de realizar acciones independientes para producir un comportamiento global deseado y con la habilidad de interactuar con otros agentes. Un sistema multi-agente es en sí un sistema distribuido, porque para realizar su tarea, los agentes deben cooperar, coordinar y negociar con otros para cumplir un propósito y cada agente hace su tarea de forma autónoma basado en una especificación dada. Los sistemas multi-agente son una base para modelar tareas distribuidas y acciones que permitan obtener un comportamiento autoorganizado (Balaji y Srinivasan, 2010).

Un sistema multi-agente es en sí un sistema distribuido, porque para realizar su tarea, los agentes deben cooperar, coordinar y negociar con otros para cumplir un propósito y cada agente hace su tarea de forma autónoma basado en una especificación dada. Los sistemas multi-agente son una base para modelar tareas distribuidas y acciones que permitan obtener un comportamiento autoorganizado (Balaji y Srinivasan, 2010).

Además, los sistemas multi-agente se utilizan para modelar sistemas complejos porque están basados en una descripción algorítmica de los individuos que simulan un determinado comportamiento esperado (Rousset, Herrmann, Lang y Philippe, 2016). Los sistemas multi-agente permiten, asimismo, emplear una aproximación

bottom-up opuesta a la aproximación *top-down* de los modelos tradicionales. Con sistemas multi-agente es posible modelar comportamientos locales utilizando simulación de tiempo discreto mediante una computación en rondas (Raynal, 2013).

En este capítulo, a partir del modelo de agentes se modelan los comportamientos de un sistema distribuido. Dichos comportamientos permiten a cada componente comunicarse y coordinar sus tareas con otros componentes, desarrollando una tarea de interés bajo la definición de que cada componente es autónomo (Van Steen y Tanenbaum, 2017). Dicha autonomía determina aspectos como un tiempo discreto propio para cada componente, la capacidad de comunicarse con otros componentes así como definir modos de envío de mensajes y reglas de procesamiento de dichos mensajes que están definidas para cada componente de forma independiente. Además, se asume que cada componente desconoce el comportamiento de los demás y tiene un conocimiento limitado del sistema distribuido como un todo, por lo que se requiere la interconexión entre dichos componentes.

Cuando se tienen aplicaciones en la nube como las ofrecidas por Amazon, se tiene una gran cantidad de componentes interconectados que pueden fallar. La presencia de fallas en los componentes de un sistema distribuido puede llevar a pérdidas del orden de miles de dólares por minuto y se requiere personal experto que realice el mantenimiento de forma manual (Lalanda *et al.*, 2013). Cuando se realizan operaciones de mantenimiento pueden suceder errores humanos, incluso los administradores pueden eliminar partes vitales de un sistema por error, como sucedió con el servicio de Amazon en la zona de Virginia del Norte donde se causó la caída en una serie de servidores cruciales debido a un comando ingresado incorrectamente ("Summary of the Amazon S3 Service Disruption in the Northern Virginia [US-EAST-1] Region", n.d.).

En este capítulo se mostrarán algunos aspectos tenidos en cuenta en la definición de una simulación basada en sistemas multi-agente que pretende resolver a la pregunta: ¿cómo obtener una simulación de comportamientos de sistemas distribuidos para la obtención de las propiedades de robustez y auto-recuperación de fallas? Para responder a esta pregunta, en la sección 1 se determinan algunos problemas de interés y los supuestos tenidos en cuenta en la simulación. Luego, en la sección 2 se explican los principios y las funciones básicas definidas para los agentes y algunos problemas tratables con el simulador con sus resultados y referencias a trabajos de investigación ya publicados con la ayuda del simulador con su respectiva contribución. Finalmente, se presentan algunas conclusiones.

1. Problemas a modelar con la simulación y supuestos

Para poder crear una simulación de comportamientos distribuidos se identificaron los siguientes subproblemas de interés: 1) determinar qué comportamientos de un sistema distribuido son interesantes para simular; 2) definir diferentes tipos de falla en un sistema distribuido y 3) desarrollar diferentes procesos que permitan ofrecer las propiedades de robustez y recuperación de las fallas definidas.

El primer subproblema que surge es cómo modelar comportamientos de un sistema distribuido usando un entorno multi-agente simulado. Se requiere una simulación que sea precisa y escalable para que modele los aspectos más relevantes del comportamiento de un sistema distribuido. En esta parte se estudiaron técnicas basadas en el movimiento animal que son la base para definir una tarea de colección y sincronización de datos en sistemas distribuidos. La selección de una tarea de colección y sincronización de datos es importante porque provee una base para mantener la información o recoger información en caso de fallas en aplicaciones distribuidas reales (Aguilera, Van Renesse y Guerraoui, 2010).

Algunos problemas conocidos relacionados con la exploración de terrenos son cubrimiento en comunicaciones con robots móviles (Beal, Correll, Urbina y Bachrach, 2009) y exploración de terrenos. Una tarea común en entornos con múltiples robots es obtener tanta información como sea posible en la menor cantidad de tiempo con el supuesto de que los agentes no tienen ningún conocimiento sobre el terreno (Perea-Ström, Bogoslavskyi y Stachniss, 2017). El cubrimiento en comunicaciones pretende formar una red sobre un espacio determinado utilizando robots con capacidades de locomoción y percepción (Winfield y Nembrini, 2012). En términos de locomoción, los algoritmos basados en movimientos aleatorios en un terreno son simulados y estudiados (Beal *et al.*, 2009).

Los principales principios en el problema de cubrimiento en comunicaciones incluyen mover agentes a áreas no exploradas y evitar otros agentes para maximizar el espacio explorado (Lopes, Frisch, Boeing, Vinsen y Bräunl, 2011). Además, como supuesto principal, los agentes únicamente tienen información local obtenida de sus sensores locales mientras exploran. El movimiento animal es estudiado como una posible forma de lograr cubrimiento en las comunicaciones. Por un lado, las moscas de fruta o los monos araña presentan ciertos patrones cuando buscan comida o exploran un ambiente determinado para cubrir un ambiente determinado. Beal (2013) muestra que estos patrones pueden ser modelados utilizando caminatas

de Lévy. Estas caminatas siguen una aproximación superdifusiva que explora y cubre un espacio determinado en una forma más rápida y efectiva comparada con las caminatas aleatorias o una aproximación de fuerzas repulsivas. En Rhee *et al*. (2011) y Saha, Misra y Pal (2015), las caminatas de Lévy son utilizadas para modelar el movimiento humano en un espacio determinado para formar redes móviles.

El segundo subproblema es definir fallas y evaluar que tan robusto puede ser un sistema distribuido. En esta simulación se consideran tres tipos de fallas. Los agentes pueden fallar mientras realizan su tarea (por ej., fallas en *software*, agentes sin batería). En un entorno de red, los agentes pueden perderse en las comunicaciones (en el caso de redes *ad hoc*) si se tiene una red no confiable. Por último, los nodos de una red pueden fallar, interrumpiendo potencialmente la comunicación entre nodos vecinos, dividiendo la red en particiones y causando que los agentes o procesos de *software* que se ejecutan en los nodos fallen o evitando que un sistema determinado ejecute la labor que se le designó.

Los sistemas distribuidos pueden fallar o ser atacados. Construir sistemas distribuidos con alta disponibilidad es un desafío, porque los nodos de los sistemas distribuidos son autónomos y, por tanto, desconocen el estado de los demás nodos de la red en un tiempo dado. En este trabajo, se presentará un sistema distribuido simulado de tipo asíncrono, porque no se tienen supuestos sobre tiempos de espera en el envío de mensajes, o tiempo tomado para ejecutar una determinada tarea. Los agentes descritos en este capítulo no tienen ni esperan acciones de otros agentes para actuar. Los modelos asíncronos son más fáciles de replicar porque no tienen supuestos de tiempo (Chandra y Toueg, 1996). Aunque no se tengan supuestos de tiempo en una simulación en tiempo discreto, cada agente puede tener un contador interno de tiempo y el tiempo se define en rondas de programa de agente.

Un tercer problema consiste en qué hacer para que el sistema repare estas fallas por sí mismo. Esta parte implica —como lo muestran algunos autores— un proceso de monitoreo, detección y recuperación de fallas (Rodosek, Geihs, Schmeck y Stiller, 2009). En esta parte se deben determinar los supuestos para que el sistema pueda recuperarse a sí mismo de fallas, el tipo de acciones que debe seleccionar el sistema para recuperarse y los eventos en el monitoreo que producen la activación de las acciones que permitan recuperar el sistema. El supuesto más importante es el tiempo entre fallas en los componentes del sistema distribuido que debe ser menor al tiempo tomado por las acciones desencadenadas para restaurar el sistema.

En este caso se puede pensar en un organismo viviente. El organismo puede recuperarse de fallas y sobrevivir siempre y cuando sus fallas no sean críticas y la velocidad en la que se recupera el organismo es superior a la que los componentes del organismo fallan.

1.1. Supuestos de la simulación

Como primer supuesto, la simulación presentada en este capítulo utiliza sistemas multi-agente. Por medio de los sistemas multi-agente se definen componentes autónomos con la capacidad de interactuar con otros (Balaji y Srinivasan, 2010). Además, los agentes utilizan un programa de agente para definir su comportamiento basado en unas percepciones obtenidas del ambiente y la producción de acciones puede tener un efecto tanto en el agente como en su entorno (Russell y Norvig, 2010).

En el desarrollo de la simulación propuesta se identificaron dos tipos de agente con su respectivo programa (Figura 1): se definieron agentes móviles identificados con un dibujo de termita, con el objetivo de explorar un ambiente dado y nodos que constituyen localizaciones en un mundo bidimensional o vértices en una red modelada como un grafo.

Los agentes móviles, como indica la Figura 1, exploran un ambiente dado (que puede ser bidimensional o que corresponde a un grafo). Además, realizan tareas de administración de datos relacionadas con la colección y replicación de datos en el ambiente seleccionado. Asimismo y para moverse incluyen diferentes implementaciones de algoritmo de movimiento que le permiten moverse entre nodos de una red dada o en un espacio definido. Los algoritmos de movimiento incluyen inicialmente: selección de una dirección al azar a un vecino de grado uno de la localización actual de un agente móvil, la definición de caminatas de Lévy y el diseño de algoritmos basados en colonias de insectos y el uso de feromonas para permitir explorar el espacio de diferentes formas, evaluando si existen diferencias cuando se aplican dichos algoritmos de movimiento para realizar una tarea de administración de datos.

Por medio de los sistemas multi-agente se definen componentes autónomos con la capacidad de interactuar con otros (Balaji y Srinivasan, 2010). Además, los agentes utilizan un programa de agente para definir su comportamiento basado en unas percepciones obtenidas del ambiente y la producción de acciones puede tener un efecto tanto en el agente como en su entorno (Russell y Norvig, 2010).

Como primitiva de comunicación, cuando dos agentes se encuentran en un mismo nodo o son vecinos en un espacio bidimensional se definieron reglas de intercambio de datos basadas en un proceso conocido como trofalaxis. El proceso de trofalaxis es importante en la comunicación de insectos sociales como termitas, abejas y hormigas, pues a través de este transfieren una porción de comida encontrada previamente y almacenada de forma intestinal a uno o varios miembros del enjambre quienes retransmiten dicha información a otros (Rodríguez y Gómez, 2011; Suárez y Thorne, 2000).a.

Como segundo supuesto, los nodos en el ambiente serán los encargados de almacenar y proporcionar la información a ser coleccionada por los agentes móviles. Tienen la capacidad de comunicarse con los agentes móviles y con sus nodos vecinos. En los escenarios de recuperación de datos, los nodos también serán los responsables de recrear agentes móviles u otros nodos con el fin de recuperar la estructura de una red dada. Los nodos también cuentan con la capacidad de conectarse con otros nodos. La conexión con otros nodos permite la definición de diferentes tipos de red presentes en la vida real como la red mundial WWW (*World Wide Web*), redes de correo electrónico o servidores virtuales configurados para realizar una tarea determinada, o granjas o configuraciones de nube.

En tercer lugar, los dos tipos de agente modelados utilizan un programa de agente. El Algoritmo 1 muestra un ejemplo de programa base de agente. En este caso se realizó una simulación en tiempo discreto a partir del concepto de rondas (en

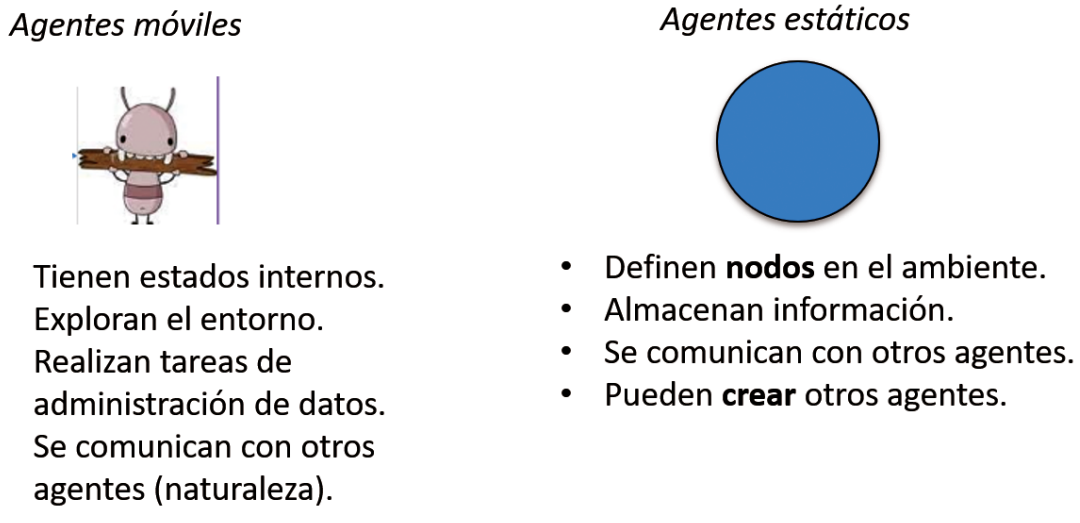


Figura 1. Tipos de agente modelados en la simulación

inglés *round*). En cada ronda, cada uno de los agentes percibe del entorno datos dependiendo de su rol (línea 10). Por ejemplo, si se trata de un agente móvil puede percibir y obtener un dato en un nodo o la cantidad de feromona que está depositada en un nodo dado para coleccionar información de dicho nodo y decidir el siguiente nodo a explorar respectivamente. Si es un nodo, puede percibir a los vecinos que tiene interconectados a él o diferentes tipos de mensaje a los que debe dar respuesta. Después de percibir la información del ambiente, el agente seleccionará una acción a partir de las percepciones dadas. En este caso, las acciones tienen efecto en el agente y también en el ambiente. En el programa de agente también incluye el concepto de falla en las líneas 5 a 7. La falla está definida en este momento como la culminación del hilo de ejecución del agente y está definida como que el componente realiza su tarea hasta que falla y no se vuelve a saber nada de este (Van Steen y Tanenbaum, 2017). Como parámetro para que el sistema falle, se define una variable fija en el intervalo [0,1] llamada probabilidad de falla p_f. En cada ronda de programa de agente, se genera un número pseudoaleatorio en el mismo intervalo [0, 1] —línea 5 del algoritmo 1—. Si el número generado es menor a p_f (línea 6), se considera que el agente falló cambiando el estado de agente e interrumpiendo el ciclo en el programa de agente (líneas 7-8), haciendo que se detenga el proceso creado para dicho agente.

```
1: Percept p
2: Action action
3: round ← 0
4: while Agent.status ≠ Fail do
5:     λ ← U[0, 1]                          ▷ uniform random number
6:     if λ < p_f then
7:         Agent.status ← Fail
8:         break
9:     end if
10:    p ← environment.sense()
11:    Action ← computeAction(p)
12:    environment.act(Agent, Action)
13:    round ← round+1
14: end while
```

Algoritmo 1. Programa de agente

2. Tipos de problema tratables con la simulación

A continuación, se mostrará cómo la simulación propuesta en este capítulo y basada en sistemas multi-agente resuelve los subproblemas de interés planteados en

la sección 1. Durante el desarrollo de la simulación, se modelaron los siguientes problemas: el primero consistió en evaluar la robustez en sistemas distribuidos móviles utilizando inteligencia de enjambres y trofalaxis (Rodríguez, Gómez y Diaconescu, 2015a, 2015b). El segundo en estudiar la robustez y el rendimiento de diferentes algoritmos para coleccionar datos en redes complejas (Rodríguez, Gómez y Diaconescu, 2017a). En tercer lugar, aborda el problema de recuperar los agentes móviles que fallan (Rodríguez, Gómez y Diaconescu, 2017b). Finalmente, se establece el problema que se intenta resolver actualmente correspondiente a la introducción de fallas en los agentes estáticos.

El simulador permite obtener métricas para realizar análisis estadísticos que son críticos para perfilar nuevos diseños de agentes. El análisis estadístico está basado en la generación de diagramas de cajas y la obtención de datos relacionados con la cantidad de mensajes empleada, el tiempo de colección de datos por el mejor agente, generación de datos y estadísticas correspondientes a métricas de grafos y otra información adicional. A partir de *scripts* definidos, se puede repetir un experimento varias veces para determinar la validez estadística de los resultados obtenidos en la simulación. A partir de los datos obtenidos, se pueden realizar diferentes pruebas estadísticas para evaluar si existen diferencias significativas en una métrica del modelo; por ejemplo, evaluar si el tiempo de colección de los datos por un agente móvil aplicando una técnica de movimiento definida es significativamente diferente al de un agente móvil aplicando otra técnica de movimiento dado.

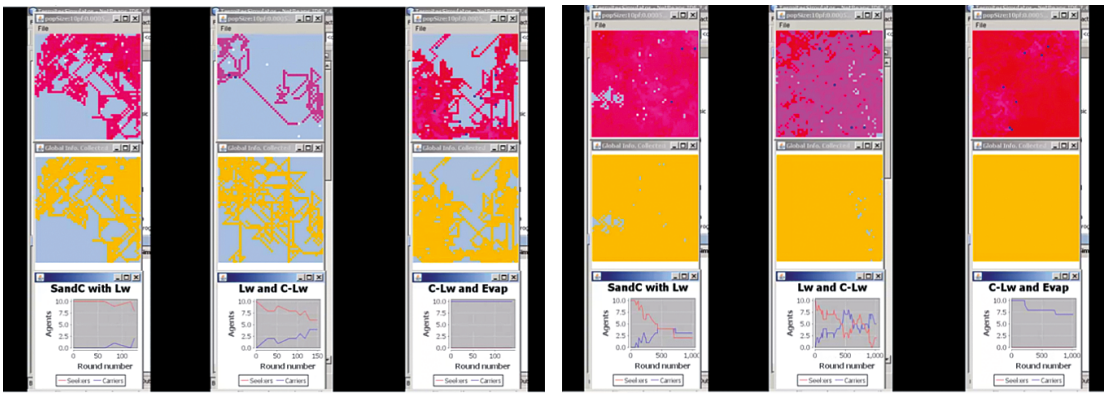
2.1. Robustez en exploración de sistemas distribuidos usando inteligencia de enjambres y trofalaxis

Inicialmente, en la simulación se diseñó y se implementó un ambiente de tipo bidimensional y agentes móviles que exploran una matriz de nodos generando un ambiente de tipo toroide. A cada nodo en el ambiente se le asignó un dato. Como objetivo de la simulación definida, se logró que al menos un agente obtuviese información de ambientes bidimensionales dados incrementando el número de agentes y el tamaño del espacio de manera que se mantuviera el mismo valor de densidad.

Mediante la exploración de diferentes algoritmos de movimiento, se mostró que se obtiene una mayor robustez en la tarea cuando se incrementa la probabilidad de falla p_f si se tiene un algoritmo de movimiento que hace que los agentes exploren el espacio tan rápido como sea posible.

Además, se modelaron distintos algoritmos de movimiento como las caminatas de Lévy (Beal, 2013) y se propuso un algoritmo de movimiento basado en el algoritmo de colonias de hormigas (Dorigo, Di Caro y Gambardella, 1999). En la Figura 2 se observa la simulación con diferentes algoritmos en distintas instantes de tiempo. Se observan tres algoritmos de movimiento basados en colonias de hormigas en la parte a) en la ronda 150, y en la parte b) en la ronda 1.000. Se tiene una $p_f = 0{,}0005$. La situación actual del ambiente se aprecia en tonos de rojo con la cantidad de feromona almacenada por los agentes móviles (un rojo más intenso significa que una mayor cantidad de agentes han pasado por un punto dado); en amarillo se aprecia que los agentes al avanzar el tiempo de forma global obtienen información de todo el terreno con excepción de algunas pequeñas partes del ambiente apreciadas como huecos en color azul claro. En la parte inferior se encuentra la cantidad de agentes que desempeñan un determinado rol *versus* el tiempo.

Como conclusión de este trabajo, se mostró que la exploración basada en colonias de hormigas es la que permite explorar el ambiente más rápido presentando una mayor robustez. En el caso de las caminatas de Lévy, se tiene una dispersión de los ambientes en el entorno que resulta más rápida que la exploración aleatoria sin dejar marcas en el ambiente. También se mostró que el mecanismo de trofalaxis implementado, permite a agentes vecinos intercambiar información para adquirir información global del ambiente mediante cooperación haciendo que la exploración sea más robusta (resista una mayor probabilidad de falla).



a) Exploración de terrenos bidimensionales r=150 b) Exploración de terrenos bidimensionales r=1000

Figura 2. Exploración de ambientes bidimensionales

El desarrollo de esta simulación permitió concluir que la velocidad en la que se completa la exploración del mundo bidimensional es una característica esencial

para lograr mejores tasas de éxito. Los mecanismos que favorecen la exploración son más resistentes a fallas que aquellos que se enfocan en incrementar la comunicación entre los agentes. Esto quiere decir que entre más rápido sea el algoritmo de exploración en recolectar información de un espacio dado, va a ser más resistente a fallas. La documentación, datos y recursos sobre esta parte del simulador están disponibles para su uso en http://www.alife.unal.edu.co/ ~ aerodriguezp/termites/.

2.2. Colección de datos autoorganizada en redes complejas

El segundo ambiente del simulador estudia la robustez y el rendimiento de diferentes algoritmos para colección y sincronización de datos en redes complejas (Rodríguez, Gómez y Diaconescu, 2017). En este trabajo mediante la definición de los nodos se modelan diferentes tipos de redes, tanto tradicionales (línea, círculo, retículo) como complejas (*small-world*, *community* y *scale-free*) con diferentes parámetros. Mediante esta simulación se adaptaron diferentes algoritmos de movimiento del ambiente bidimensional para hacer posible que los agentes móviles exploren y recolecten datos de una red compleja determinada. Del mismo modo, se pueden realizar experimentos con diferentes valores de p_f.

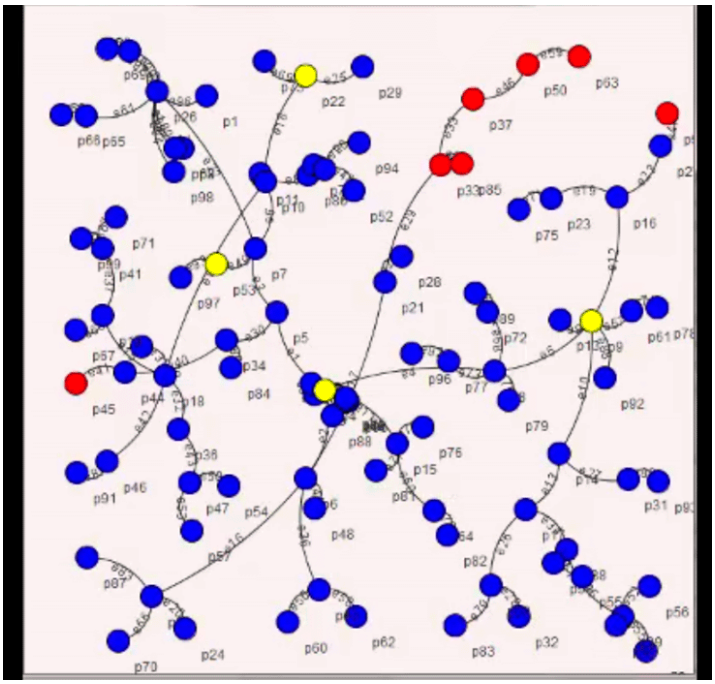


Figura 3. Cuatro agentes móviles explorando una red compleja de tipo scale-free

La Figura 3 presenta un tipo de red modelada en el simulador y agentes mostrados como puntos amarillos en el simulador. Los nodos en rojo son posiciones no exploradas. Los agentes utilizan diferentes algoritmos de movimiento para explorar la red compleja. Y se observa cómo los agentes de tipo nodo se conectan y son capaces de modelar interacciones con agentes móviles y otros nodos.

Una red compleja consiste en un alto número de componentes interconectados caracterizados por propiedades topológicas que no son triviales. Pues son redes que ni tienen una estructura totalmente regular (por ej., todos los nodos tienen el mismo grado) ni sus conexiones son generadas totalmente de forma aleatoria (Van Der Hofstad, 2017). Características típicas de una red compleja incluyen mantener distancias relativamente pequeñas entre los nodos (Mori, Uehara y Matsumoto, 2015), presentar distribuciones de potencias en el grado de los nodos (Barabási y Bonabeau, 2003) o formar comunidades o clústeres.

Las redes se definen en este simulador como un grafo G(V, E), con una colección de vértices V y un conjunto de arcos E. Para generar las redes complejas se utilizaron reglas de interconexión que se encuentran en la literatura y un *framework* que permite definir y calcular métricas sobre grafos llamado JUNG (White, 2005).

Como algoritmos de movimiento, en esta parte se modelaron movimientos aleatorios y una adaptación del algoritmo basado en enjambres dada la imposibilidad de modelar caminatas de Lévy ante la incapacidad de definir un sistema de direcciones. Los resultados experimentales de esta parte de la simulación muestran que la velocidad de colección de los datos depende del tipo de red. Para la mayoría de las redes, se tiene que la exploración basada en colonias de insectos es más rápida que una exploración aleatoria. Sin embargo, observando los parámetros de las redes, se encontró una relación entre la desviación estándar de la métrica de intermediación de un grafo y la velocidad de colección de la información. A mayor desviación de esta métrica, se requiere mayor tiempo para recolectar datos de una red dada y un movimiento aleatorio puede resultar más rápido que en el modelo basado en hormigas, definido en Rodríguez *et al.* (2017a). Algunos resultados están disponibles en http://www.alife.unal.edu.co/ ~ aerodriguezp/datacol/.

2.3. Replicación de agentes que exploran redes complejas

En esta simulación se planteó el problema de definir un tipo de falla que depende de las comunicaciones. Los nodos están conectados por enlaces que pueden fallar. Se asume que estos enlaces, por ejemplo, en un entorno de computación

en la nube corresponden a conexiones entre servidores que pueden ser físicos o virtuales. Cuando se envía información entre nodos se pueden perder paquetes y del mismo modo los agentes pueden fallar cuando se están moviendo de un nodo a otro (Rodríguez *et al.*, 2017b). Se diseñó un mecanismo que permite a los nodos, aunque los enlaces fallen transmitiendo la información de los agentes, crear nuevas réplicas de agente móvil, teniendo una primera aproximación basada en reglas locales que recrea agentes fallidos, logrando que los agentes móviles recolecten información de toda una red, incluso con altos valores en su probabilidad de fallar.

Para corregir estas fallas en los enlaces, se modelan tiempos de espera en los que al no recibir respuesta de un agente por un tiempo dado el nodo sea capaz de replicar dicho agente que falla. En ese sentido, se pueden presentar varios escenarios posibles. En la Figura 4 se aprecian tres posibles escenarios: el primero (llamado *No replication*) donde no hay replicación y los agentes fallan con una tasa alta de fallas $p_f = 0,5$. Una tasa de probabilidad de 0,5 es pensar en que cada vez que el agente se va a mover de un nodo a otro, se lanza una moneda y si cae cara, por ejemplo, las comunicaciones fallan haciendo que el agente detenga su ejecución. En este primer escenario, se observan muchos nodos en rojo, pues algunos de los agentes fallaron y no se logró la tarea de colección de datos en la red dada. En el segundo escenario (llamado *With replication and no delay*), se tiene que, si el tiempo de espera es adecuado, se logra generar réplicas que con el tiempo permiten que un agente al menos recolecte información de todos los nodos que conforman la red. En el tercer escenario (*With replication and delay 3s*) se aprecia que, si se tiene un retraso en las comunicaciones mayor al tiempo de espera, los nodos localmente tenderán a crear réplicas de los agentes causando sobrepoblación de estos y consumiendo una mayor cantidad de recursos que puede implicar sobreuso de los recursos de red que tiene el sistema disponible para realizar su tarea dada.

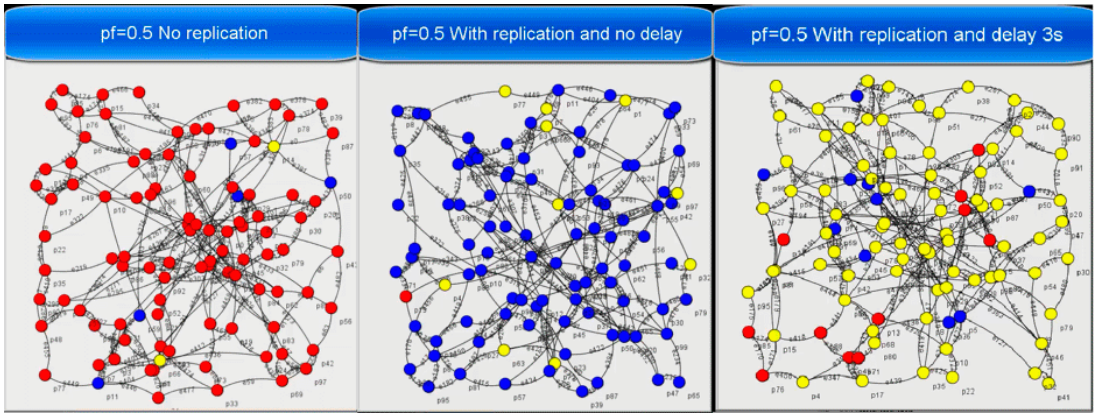


Figura 4. El problema de la replicación de nodos

Para definir un escenario realista, lo ideal fue definir diferentes retrasos en cada uno de los enlaces de la red. Para comprobar qué tan robusto es el sistema, se definieron diferentes probabilidades de fallas que se fueron incrementando conforme al diseño de experimentos. El modelamiento de diferentes retrasos para cada uno de los enlaces implicó definir reglas en los nodos que permitieran aproximar los tiempos de espera de toda la red con el tiempo, partiendo de una estimación y manejando ventanas de tiempo correspondientes a los tiempos obtenidos de los agentes que se logran mover entre cada par de nodos.

Este modelo requiere que los tiempos de espera se adapten automáticamente para evitar sobrepoblación en el número de agentes móviles. Por medio de esta simulación en Rodríguez, Gómez y Diaconescu (2017c), se propuso un modelo que asegura que una tarea se complete, incluso con valores de p_f hasta de 0,5. Si no se recibe notificación de un nodo en un tiempo esperado, se crea una réplica del agente móvil. Además, utilizando los tiempos de los agentes móviles que pasan de un determinado nodo a otro, se estiman los tiempos límite para cada uno de los enlaces.

Como resultado de este modelo, partiendo de la aproximación dada por el movimiento de los agentes y guardando un historial de los tiempos de espera, mostró en Rodríguez *et al*. (2017b) que se evita la sobrepoblación, pues el número de agentes tiende a ser la cantidad inicial de agentes definida en cada experimento y también que el número mínimo de agentes se aproxime al número inicial mediante la reducción del tiempo de espera si inicialmente es más alto que los tiempos de las comunicaciones. El análisis experimental generado por la simulación incluye tasas de éxito, consumo de memoria, rapidez en la colección de datos con y sin replicación y análisis del número de agentes. La documentación, videos y resultados adicionales están disponibles en http://www.alife.unal.edu.co/~aerodriguezp/networksim/.

2.4. Auto-recuperación de la estructura de una red compleja

En esta última parte del simulador se planteó el problema en el que se está trabajando en la actualidad. Inicialmente se definen fallas en los nodos de una red compleja dada y se implementa una métrica de similitud dada para comparar a una misma red en diferentes instantes de tiempo dado. Cuando se presenta una falla en un nodo específico, tanto el nodo como las conexiones que tiene ese nodo desaparecen y la estructura de la red cambia.

Se pretende, utilizando las capacidades locales que tiene cada nodo, que los nodos sean capaces de crear nuevas réplicas de otros nodos fallidos, utilizando información sobre la topología de la red y a través de la percepción de los vecinos de grado uno de cada nodo en la red. En esta parte del simulador, la exploración robusta de las redes complejas, utilizando los algoritmos de movimiento definidos en las partes anteriores de la simulación, permite diseñar un mecanismo en el que la información de la red a sincronizar sea dada por los agentes móviles quienes sincronizarán la información sobre la topología de la red.

Los agentes hasta la sección 2.3. Donde se plantea el modelo de replicación de agentes, contaban con capacidad de almacenamiento ilimitada para guardar la información que tenían almacenada los nodos en la red. En esta simulación, los agentes móviles cuentan con almacenamiento limitado. De este modo, solo podrán almacenar partes de la información sobre la topología de la red. Se espera que al distribuir dicha información se pueda permitir a un nodo saber si alguno de sus vecinos falló.

Utilizar agentes móviles para explorar una red compleja puede ser importante en casos donde la estructura de la red no se conoce, pero se desea mantener. Como las fallas en una red pueden darse en cualquier momento, es posible que en un punto del tiempo al agregar o eliminar nodos de una red llegue a desconocerse la estructura de esta. Por otro lado, si se sabe que una estructura de la red es más eficiente que otra para realizar una tarea determinada de replicación de datos, puede ser importante mantener dicha estructura.

Conclusiones

En este capítulo se presentaron diferentes problemas relacionados con: 1) la selección de una tarea distribuida para ser modelada utilizando una simulación; 2) las partes principales de un sistema distribuido que pueden ser modeladas y que corresponden a la estructura de la red y los programas de nodo como vértices de dicha red y agentes móviles que realizan una tarea de administración de datos; 3) el modelamiento y la justificación de la aplicación de un modelo basado en agentes para realizar la simulación mencionada; 4) la definición de fallas en los diferentes componentes del sistema; 5) la medición del efecto de estas fallas en el sistema y la obtención de robustez en una tarea de colección de datos; 6) el impacto de la aplicación de reglas locales en la obtención de las propiedades de robustez y recuperación de fallas.

Se definieron como base dos tipos de agente: agentes móviles y nodos. Los agentes móviles tienen estados internos, exploran un ambiente y realizan tareas de administración de datos y se comunican con otros agentes que encuentran mientras desempeñan la exploración de un ambiente dado (sea un entorno bidimensional o una red compleja). Los nodos son agentes estáticos que definen el ambiente distribuido a ser explorado. Los nodos almacenan feromonas, proveen mecanismos como evaporación pasiva y almacenan información local de interés. Además, los nodos pueden comunicarse con otros nodos o agentes móviles y crear réplicas de otros agentes.

El simulador en sus diferentes aspectos ha sido una herramienta de ayuda en la definición de reglas locales que permiten obtener comportamientos para evaluar la robustez en distintas tareas seleccionadas propias de un sistema distribuido. La metodología empleada para obtener los resultados obtenidos está basada en la definición de reglas locales y en la experimentación de distintas opciones de reglas locales que producen los comportamientos de autoorganización deseados. La parte difícil de la obtención de modelos autoorganizados corresponde al descubrimiento de las reglas locales que permiten generar el comportamiento deseado. Por medio de la simulación planteada y mediante error y ensayo con diferentes configuraciones de parámetros, se buscó en cada parte de la simulación inicialmente reglas que garantizaran un comportamiento deseado y posteriormente una mejora de estas.

En las simulaciones mostradas y citadas en este capítulo, se tiene un número de nodos fijo igual a cien, pues se pretendía realizar experimentos lo más rápido posible en un único equipo de cómputo donde los recursos de memoria y del procesador son limitados. Como trabajo futuro, se pretende extender la cantidad de nodos en las diferentes redes generadas a mil o a diez mil nodos para probar la calidad de las reglas obtenidas hasta el momento. A parte de extender la cantidad de los nodos, también se desea aproximar y modelar otro tipo de tareas distribuidas que representan aplicaciones de la industria vigentes, como son las redes de servidores virtuales y el uso de contenedores de aplicaciones. Se pretende comparar mecanismos de colección de información tradicionalmente utilizados por estos sistemas con el modelo propuesto de agentes móviles.

Otros temas de interés, cuando se tiene simulación de sistemas distribuidos corresponden a mantener coherencia en la información complementaria utilizando modelos de consenso. En este momento, el simulador trabaja con información complementaria que no es contradictoria. Como trabajo futuro, se piensa integrar

también mecanismos de elección de líder como el propuesto por Ongaro (Ongaro y Ousterhout, 2014), donde se elige un servidor para que decida qué cambios deben aplicarse en la red para mantener coherencia en la información. El tema de consenso es complementario al trabajo realizado, pues asume que los nodos fallan y son recuperados por un proceso externo. Cuando se recupera el proceso externo, es actualizado a partir de la información que posee un líder dado. Se espera también a futuro, estudiar mecanismos de consenso que provienen de la naturaleza, con el fin de ver si se pueden modelar aún comportamientos más simples que le permitan a una red recuperarse de fallas y, al mismo tiempo, mantener coherencia en la información. Otro tema complementario es la integración de mecanismos de seguridad al modelo. Este tema no es cubierto en la simulación planteada, pues está concentrada en la obtención de autopropiedades.

Agradecimientos

Se desea agradecer a la doctora Ada Diaconescu de Telecom ParisTech, quien es colaboradora en el proyecto del simulador. La doctora ha asesorado la construcción del simulador mostrado en este capítulo a partir de su experiencia en el área de computación autonómica y autoorganización. Ada Diaconescu es Ph.D en Ingeniería Electrónica y Computación de la Universidad de Dublín, en Irlanda. Actualmente se desempeña como profesora titular asistente (*Maître de conférences*, en francés) en Telecom ParisTech.

Referencias

Agrawal, N., Prabhakaran, V., Wobber, T., Davis, J. D., Manasse, M., & Panigrahy, R. (2008). Design tradeoffs for SSD performance. In *USENIX 2008 Annual Technical Conference* (pp. 57-70). Berkeley, CA, USA: USENIX Association. Retrieved from http://dl.acm.org/citation.cfm?id = 1404014.1404019.

Aguilera, M. K., Van Renesse, R., & Guerraoui, R. (2010). *Replication*. In B. Charron-Bost, F. Pedone & A. Schiper (Eds.), *Lecture notes in computer science* (including subseries *Lecture notes in artificial intelligence and lecture notes in bioinformatics*) (vol. 5959). Berlin, Heidelberg, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-11294-2.

Balaji, P. G., & Srinivasan, D. (2010). An introduction to multi-agent systems. *Studies in Computational Intelligence*, *310*, 1-27. http://doi.org/10.1007/978-3-642-14435-6_1.

Barabási, A.-L., & Bonabeau, E. (2003). Scale-free networks. *Scientific American*, *288*(5), 60-69. http://doi.org/10.1038/scientificamerican0503-60.

Beal, J. (2013). Superdiffusive dispersion and mixing of swarms with reactive levy walks. *International Conference on Self-Adaptive and Self-Organizing Systems, SASO*, 141-148. http://doi.org/10.1109/SASO.2013.9.

Beal, J., Correll, N., Urbina, L., & Bachrach, J. (2009). Behavior modes for randomized robotic coverage. *2009 Second International Conference on Robot Communication and Coordination*. http://doi.org/10.4108/ICST.ROBOCOMM2009.5863.

Bhardwaj, R., & Dixit, V. S. (2010). An overview on tools for peer to peer network simulation, *International Journal of Computer Applications*, *1*(19), 70-76.

Bicocchi, N., & Zambonelli, F. (2007). Autonomic communication learns from nature. *IEEE Potentials*, *26*(6), 42-46. http://doi.org/10.1109/MPOT.2007.906119.

Casanova, H., Giersch, A., Legrand, A., Quinson, M., & Suter, F. (2014). Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, *74*(10), 2899-2917. Retrieved from http://hal.inria.fr/hal-01017319.

Casanova, H., Legrand, A., & Quinson, M. (2008). SimGrid: A generic framework for large-scale distributed experiments. *Tenth International Conference on Computer Modeling and Simulation (Uksim 2008)*, 126-131. http://doi.org/10.1109/UKSIM.2008.28.

Chandra, T., & Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of the ACM (JACM)*, *43*(2), 225-267. http://doi.org/10.1145/226643.226647.

Dorigo, M., Di Caro, G., & Gambardella, L. M. (1999). Ant algorithms for discrete optimization. *Artificial Life*, *5*(2), 137-172. http://doi.org/10.1162/106454699568728.

Fujiwara, K., & Casanova, H. (2007). Speed and accuracy of network simulation in the simgrid framework. In *Proceedings of the 2nd International Conference on Performance Evaluation Methodologies and Tools*. Retrieved from http://dl.acm.org/citation.cfm?id=1345279.

Lalanda, P., McCann, J. A., & Diaconescu, A. (2013). *Autonomic computing: Principles, design and implementation*. Springer.

Lopes, S., Frisch, B., Boeing, A., Vinsen, K., & Bräunl, T. (2011). Autonomous exploration of unknown terrain for groups of mobile robots. *IEEE Intelligent Vehicles Symposium, Proceedings*, (IV), 157-162. http://doi.org/10.1109/IVS.2011.5940455.

Mori, H., Uehara, M., & Matsumoto, K. (2015). Parallel architectures with small world network model. *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, 467-472. http://doi.org/10.1109/WAINA.2015.84.

Ongaro, D., & Ousterhout, J. (2014). In search of an understandable consensus algorithm. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference* (pp. 305-320). USENIX Association. Retrieved from https://ramcloud.stanford.edu/wiki/download/attachments/11370504/raft.pdf.

Perea-Ström, D., Bogoslavskyi, I., & Stachniss, C. (2017). Robust exploration and homing for autonomous robots. *Robotics and Autonomous Systems*, *90*, 125-135. http://doi.org/10.1016/j.robot.2016.08.015.

Rana, M. K., Sardar, B., Mandal, S., & Saha, D. (2017). Implementation and performance evaluation of a mobile IPv6 (MIPv6) simulation model for ns-3. *Simulation Modelling Practice and Theory*, *72*, 1-22. http://doi.org/http://dx.doi.org/10.1016/j.simpat.2016.12.005.

Raynal, M. (2013). *Distributed algorithms for message-passing systems*. Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-38123-2.

Rhee, I., Shin, M., Hong, S., Lee, K., Hong, S., Kim, S. J., & Chong, S. (2011). On the Levy-Walk nature of human mobility. *IEEE/ACM Transactions on Networking*, *19*(3), 630-643. http://doi.org/10.1109/TNET.2011.2120618.

Rodosek, G. D., Geihs, K., Schmeck, H., & Stiller, B. (2009). Self-healing systems: Foundations and challenges. In A. Andrzejak, K. Geihs, O. Shehory & J. Wilkes (Eds.), *Self-healing and self-adaptive systems* (vol. 9201, pp. 1-6). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Germany. Retrieved from http://drops.dagstuhl.de/opus/volltexte/2009/2110/pdf/09201.SWM.ExtAbstract.2110.pdf.

Rodríguez, A., & Gómez, J. (2011). *Termites system with self-healing based on autonomic computing*. Universidad Nacional de Colombia. Retrieved from http://www.bdigital.unal.edu.co/5414/.

Rodríguez, A., Gómez, J., & Diaconescu, A. (2015a). Foraging-inspired self-organisation for terrain exploration with failure-prone agents. In IEEE (Ed.), *2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems* (vol. 2015-Octob, pp. 121-130). Boston MA, USA: IEEE. http://doi.org/10.1109/SASO.2015.20.

Rodríguez, A., Gómez, J., & Diaconescu, A. (2015b). Towards failure-resistant mobile distributed systems inspired by swarm intelligence and trophallaxis. In *Proceedings of the European Conference on Artificial Life 2015, At The University of York UK* (vol. 2, pp. 448-455). http://doi.org/http://dx.doi.org/10.7551/978-0-262-33027-5-ch080.

Rodríguez, A., Gómez, J., & Diaconescu, A. (2017). Exploring complex networks with failure-prone agents. In S. Verlag (Ed.), *15th Mexican International Conference on Artificial Intelligence, MICAI 2016* (pp. 81-98). Lecture notes in computer science. http://doi.org/0302-9743.

Rodríguez, A., Gómez, J., & Diaconescu, A. (2017a). Exploring complex networks with failure-prone agents. In O. Pichardo-Lagunas & S. Miranda-Jiménez (Eds.), *Advances in soft computing: 15th Mexican International Conference on Artificial Intelligence, MICAI 2016, Canc{ú}n, Mexico, October 23-28, 2016, Proceedings, Part II* (pp. 81-98). Cham: Springer International Publishing. http://doi.org/10.1007/978-3-319-62428-0_7.

Rodríguez, A., Gómez, J., & Diaconescu, A. (2017b). Replication-based self-healing of mobile agents exploring complex networks. In Y. Demazeau, P. Davidsson, J. Bajo & Z. Vale (Eds.), *Advances in practical applications of cyber-physical multi-agent systems: The PAAMS Collection: 15th International Conference, PAAMS 2017, Porto, Portugal, June 21-23, 2017, Proceedings* (pp. 222-233). Cham: Springer International Publishing. http://doi.org/10.1007/978-3-319-59930-4_18.

Rodríguez, A., Gómez, J., & Diaconescu, A. (2017c). Replication-based self-healing of mobile agents exploring complex networks. In Y. Demazeau, P. Davidsson, J. Bajo & Z. Vale (Eds.), *Advances in practical applications of cyber-physical multi-agent systems: The PAAMS Collection: 15th International Conference, PAAMS 2017, Porto, Portugal, June 21-23, 2017, Proceedings* (pp. 222-233). Lecture notes in computer science. http://doi.org/10.1007/978-3-319-59930-4_18.

Rousset, A., Herrmann, B., Lang, C., & Philippe, L. (2016). A survey on parallel and distributed multi-agent systems for high performance computing simulations. *Computer Science Review*, *22*, 27-46. http://doi.org/10.1016/j.cosrev.2016.08.001.

Russell, S., & Norvig, P. (2010). *Artificial intelligence a modern approach* (Third). New Jersey: Prentice-Hall.

Saha, B., Misra, S., & Pal, S. (2015). Utility-based exploration for performance enhancement in opportunistic mobile networks. *IEEE Transactions on Computers*, *9340*(c), 1. http://doi.org/10.1109/TC.2015.2441700.

Simgrid. (2012). Getting started with SIM GRID models. Retrieved from http://simgrid.gforge.inria.fr/tutorials/surf-101.pdf.

Suárez, M. E., & Thorne, B. L. (2000). Rate, amount, and distribution pattern of alimentary fluid transfer via trophallaxis in three species of termites (Isoptera: Rhinotermitidae, Termopsidae). *Annals of the Entomological Society of America*, *93*(Shapiro 1990), 145-155. http://doi.org/10.1603/0013-8746(2000)093[0145:RAADPO]2.0.CO;2.

Summary of the Amazon S3 Service Disruption in the Northern Virginia (US-EAST-1) Region. (n.d.). Retrieved from https://aws.amazon.com/es/message/41926/.

Van Der Hofstad, R. (2017). *Random graphs and complex networks*, vol. I. *Available on http://www. win. tue. nl/rhofstad/NotesRGCN. pdf* (vol. I). Cambridge Series in Statistical and Probabilistic Mathematics. Retrieved from http://www.win.tue.nl/ ~ jkomjath/NotesRGCN2013may.pdf.

Van Steen, M., & Tanenbaum, A. (2017). *Distributed systems*. (M. Van Steen, Ed.) (vol. 3.01). http://doi.org/10.1145/90417.90747.

Velho, P., & Legrand, A. (2009). Accuracy study and improvement of network simulation in the SimGrid framework. In *Proceedings of the Second International ICST Conference on Simulation Tools and Techniques*. ICST. http://doi.org/10.4108/ICST.SIMUTOOLS2009.5592.

White, S. (2005). Analysis and visualization of network data using JUNG. *Journal of Statistical Software*, *VV*(Ii), 1-35. Retrieved from http://www.citeulike.org/group/206/article/312257.

Winfield, A. F. T., & Nembrini, J. (2012). Emergent swarm morphology control of wireless networked mobile robots. In *Morphogenetic engineering, toward programmable complex systems* (pp. 239-271). Springer. http://doi.org/10.1007/978-3-642-33902-8_10.