Plataforma para el desarrollo de aplicaciones multiagente Incorporación de modelos de comunicaciones precisos

Fidelis Perkonigg, Djordje Brujic, miembro, IEEE y Mihailo Ristic, miembro, IEEE

Resumen—Los sistemas multiagente son ampliamente reconocidos como El método de elección para la implementación de aplicaciones distribuidas de tiempo crítico para la red inteligente. Sin embargo, no existen soluciones generales. Se ha propuesto para la difícil tarea del desarrollo del sistema y validación, lista para su implementación, que daría cuenta completamente de El rendimiento de la red de comunicación subyacente. Proponemos una nueva plataforma diseñada para este propósito, que integra un marco de desarrollo multiagente estándar [Java Agent Desarrollo (JADE)] y un estándar de la industria de las comunicaciones Simulador de red (modelador OPNET). Se realizó mediante extensiones genéricas del marco JADE para proporcionar información discreta capacidades de programación de eventos, mientras que el modelador OPNET fue ampliado para proporcionar un método genérico de asociar la red Nodos con agentes ejecutándose en JADE. El método adoptado se ajusta al estándar de arquitectura de alto nivel. Es importante destacar que las aplicaciones Desarrollado con esta plataforma, puede implementarse en el sistema de destino sin modificaciones manuales. Se presenta una aplicación de protección distribuida y se analiza su rendimiento con respecto a... comportamientos de los agentes candidatos y escenarios de comunicación, lo que demuestra que la viabilidad de la aplicación depende críticamente de las decisiones tomadas durante su diseño e implementación.

Términos del índice : redes de comunicación, sistemas multiagente (MASs), red inteligente.

I. INTRODUCCIÓN

METRO Los sistemas ULTIAGENT (MAS) ahora son ampliamente reconocidos como el enfoque preferido para el desarrollo de aplicaciones distribuidas, que involucran múltiples aplicaciones autónomas Unidades que se comunican entre sí, de modo que sus acciones coordinadas buscan un objetivo común. En los últimos años, los MAS han adquirido cada vez mayor importancia en el contexto del poder sistemas, particularmente dentro del concepto de red inteligente [1], donde Se demostró que la tecnología de agentes inteligentes es prometedora Método para automatizar numerosas tareas relacionadas con la gestión y el control de la red. Aplicaciones distribuidas para la red inteligente. Puede implicar nodos autónomos que estén físicamente separados. por decenas o incluso cientos de kilómetros y dependen de las redes de datos disponibles para la comunicación [1]–[3]. Esto se ve como una alternativa significativamente más flexible que la tradicional control de supervisión centralizado y adquisición de datos (SCADA) sistemas [1], [2]. Para aplicaciones críticas en el tiempo, como la protección y el control de la red, los retrasos en la comunicación son un factor clave en Determinar la viabilidad y la fiabilidad de una aplicación distribuida específica. Ejemplos de aplicaciones distribuidas basadas en... en MAS incluyen protección del sistema eléctrico [1], restauración [4], [5], diagnóstico [5], control de tensión [6] y control de microrredes [7]. Características de seguridad, robustez y rendimiento. En diferentes condiciones de tráfico de red, las cuestiones clave son claramente abordar al intentar desarrollar un esquema de este tipo. Sin embargo, el proceso de desarrollo, verificación e implementación de aplicaciones MAS críticas en el tiempo no se ha llevado a cabo de manera adecuada. abordado hasta la fecha.

La figura 1 muestra las etapas de desarrollo de dicha aplicación, comenzando con el diseño del prototipo del método de control. y su análisis mediante simulación. Esto da como resultado la especificación de parámetros y restricciones de rendimiento (como latencias permisibles) en las que el método implementado debe reunirse. El código de la aplicación se implementará entonces utilizando el lenguaje de programación elegido y la plataforma de ejecución. Plataformas de desarrollo multiagente, como ZEUS [8] o Java El desarrollo de agentes (JADE) [9] ha sido adoptado por numerosos investigadores en el campo de los sistemas de energía para este propósito. [1]–[7], [10], [11].

En la etapa de implementación se deben tomar una serie de decisiones críticas y deben hacerse concesiones, incluidos comportamientos específicos de los agentes, elección de nodos de red física en los que se encuentran ciertas funciones críticas Los agentes se están ejecutando, los protocolos de comunicación y las estrategias de calidad de servicio (QoS). El código de la aplicación implementada... También es necesario analizarlo en relación con la red específica. componentes (enrutadores y conmutadores) y condiciones del tráfico de datos Para la red de comunicación objetivo. Dispersión geográfica. del hardware objetivo y los riesgos de daños a los equipos vitales hacen imperativo realizar pruebas y validaciones desconectado.

La implementación en el sistema de destino distribuido requiere Código de aplicación completamente probado y sin (o con una mínima) modificación manual del código. Con esto en mente, la introducción de MAS a la industria eléctrica hace todo el desarrollo ciclo de vida de aplicaciones de control y protección de tiempo crítico significativamente más desafiante.

Las investigaciones anteriores relacionadas con las redes inteligentes se han centrado principalmente en Diseño de prototipos y simulación de métodos de control distribuido [12]–[14], pero la necesidad de probar y validar los métodos implementados Las aplicaciones, en formato codificado y listas para su implementación, no tienen se han abordado adecuadamente. Esto constituye la principal motivación para el trabajo presentado en este artículo.

Manuscrito recibido el 16 de junio de 2014; revisado el 9 de febrero de 2015; aceptado 15 de abril de 2015. Fecha de publicación 30 de abril de 2015; fecha de la versión actual 02 de junio de 2015. Este trabajo fue apoyado por el Programa de Energía de El Consejo de Investigación en Ingeniería y Ciencias Físicas (EPSRC) Supergen Proyecto HiDEF EP/G031681/1, que es un proyecto de los Consejos de Investigación del Reino Unido Iniciativa interconsular (RCUK) liderada por EPSRC. Documento n.º TII-14-1349.

F. Perkonigg trabaja en el Departamento de Informática del Imperial College. Londres, Londres SW7 2AZ, Reino Unido (correo electrónico: f.perkonigg10@imperial.ac.uk).

D. Brujic y M. Ristic trabajan en el Departamento de Ingeniería Mecánica. Ingeniería, Imperial College London, Londres SW7 2AZ, Reino Unido (correo electrónico: d.brujic@imperial.ac.uk; m.ristic@imperial.ac.uk).

Las versiones en color de una o más de las figuras de este artículo están disponibles en línea. en http://ieeexplore.ieee.org.

Identificador de objeto digital 10.1109/TII.2015.2428633

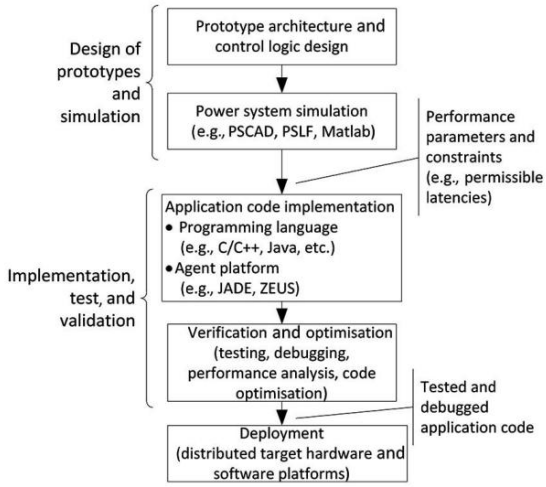

Fig. 1. Etapas del desarrollo de aplicaciones MAS, distinguiendo la simulación de prototipos funcionales del desarrollo del código de aplicación y la validación abordadas en este trabajo.

Sistemas como el simulador de sincronización de energía eléctrica y comunicaciones (EPOCHS) [12] y la plataforma global de cosimulación basada en eventos (GECO) [13] se han diseñado para mejorar la simulación de prototipos de control distribuido mediante la combinación de dos herramientas de simulación distintas en una única simulación acoplada: una simulación de un sistema de energía en tiempo continuo, por un lado, y la simulación de la red de comunicaciones de eventos discretos NS21, por otro. Sin embargo, no incluyen una plataforma multiagente. En EPOCHS, la funcionalidad del agente se representa mediante un componente de simulación específico, mientras que en GECO se representa dentro de NS2 como parte de los modelos de nodos de la red. Por lo tanto, ninguno de los dos sistemas implica el código de la aplicación.

En este artículo, nos centramos en las etapas de implementación, prueba y validación del proceso mostrado en la Fig. 1. Proponemos una plataforma que combina un marco multiagente completo y un paquete completo de simulación de redes de comunicaciones. De esta forma, es posible desarrollar y validar completamente el código de una aplicación MAS crítica en tiempo, preparándolo para su implementación en un sistema de destino, sin necesidad de modificaciones manuales adicionales. Si bien nuestra principal motivación fue abordar las necesidades derivadas del desarrollo de aplicaciones para la red inteligente, este trabajo también es relevante para una amplia gama de otras aplicaciones de control distribuido.

En línea con otras investigaciones sobre sistemas de energía [1], [2], [4]–[7], [14], [15], hemos adoptado el marco JADE [9], [16] como plataforma multiagente y lo combinamos con un simulador de red de comunicaciones (NS) establecido, el modelador OPNET2.

La función del modelador OPNET es proporcionar una simulación precisa de la red de comunicaciones, en diversos escenarios, como parte integral del entorno objetivo. El marco JADE proporciona todo el middleware necesario, compatible con la Fundación para Agentes Físicos Inteligentes (FIPA)3.

1Simulador de red 2. [En línea] Disponible: http://www.isi.edu/nsnam 2OPNET, Riverbed Technology. [En línea] Disponible: http://www.lechodelrío.com

3FIPA. (2014). Fundación para Agentes Físicos Inteligentes. Organización de estándares de la IEEE Computer Society. [En línea] Disponible en: http://www.fipa.org/

Para la implementación de aplicaciones basadas en agentes, es compatible con una amplia gama de sistemas operativos que ejecutan Java. Si bien se sabe que Java emplea mecanismos de recolección de elementos no utilizados que pueden interrumpir la ejecución del programa en momentos impredecibles, este problema ha sido abordado por la especificación de tiempo real para Java [17], para la cual existen varias implementaciones.

Para materializar este concepto, fue necesario resolver varios aspectos clave, los cuales constituyen las contribuciones clave de este artículo. En primer lugar, JADE ofrece una plataforma para ejecutar agentes en tiempo real, pero no comprende el concepto de tiempo de simulación; por lo tanto, fue necesario ampliarla para dotarla de capacidades de eventos discretos. En segundo lugar, se buscó proporcionar métodos genéricos para mapear los agentes de aplicación (que se ejecutan en JADE) en el marco de simulación OPNET, lo que implicó extensiones tanto del marco JADE como de OPNET. Finalmente, se gestionó la sincronización y la transferencia de la ejecución entre dos plataformas. Esto se logró mediante una infraestructura de tiempo de ejecución (RTI) de acuerdo con el estándar de arquitectura de alto nivel (HLA) [18], [19], que también requirió la provisión de extensiones genéricas tanto del marco JADE como del modelador OPNET.

La adopción del estándar HLA es común al enfoque utilizado en EPOCHS, GECO y otros sistemas de simulación acoplados. Sin embargo, este trabajo se centra en la integración entre JADE y OPNET, y en las extensiones requeridas para cada uno de estos sistemas, con el fin de permitir que las aplicaciones del agente JADE, en formato codificado implementable en el sistema de destino, se ejecuten en combinación con una simulación de eventos discretos de la red de comunicaciones. Por el contrario, el diseño de EPOCHS y GECO se centró en los problemas de combinar una simulación de sistemas de potencia en tiempo continuo (p. ej., utilizando PSCAD y PSLF) con la simulación de eventos discretos de la red de comunicaciones (p. ej., utilizando NS2). La cosimulación de sistemas de potencia y comunicaciones ha sido estudiada extensamente por Mets et al. [14]. Los métodos propuestos en [12] también pueden utilizarse en nuestro sistema para incluir el modelado de sistemas de potencia, aunque este no era nuestro objetivo.

El código fuente desarrollado como parte de este trabajo está disponible para los investigadores que lo soliciten a través del autor correspondiente. También se necesitan JADE, OPNET y RTI para replicar la plataforma, todas ellas disponibles gratuitamente para la investigación.

Este artículo se organiza de la siguiente manera. La Sección II describe el diseño, la arquitectura y la implementación del sistema propuesto. La Sección III presenta, a modo de ejemplo, la implementación y evaluación de un sistema de supervisión de relés de respaldo remotos de zona 3 basado en agente [20], y en la Sección IV se presenta el análisis del rendimiento en relación con la elección de protocolos y una infraestructura de comunicación específica. La Sección V presenta las conclusiones.

II. INTEGRACIÓN DEL AGENTE Y LA COMUNICACIÓN PLATAFORMAS DE SIMULACIÓN

A. Descripción general de la arquitectura

Según la terminología adoptada por el estándar HLA [21], un sistema que comprende dos o más entidades heterogéneas acopladas entre sí se denomina federación. La figura 2 ilustra

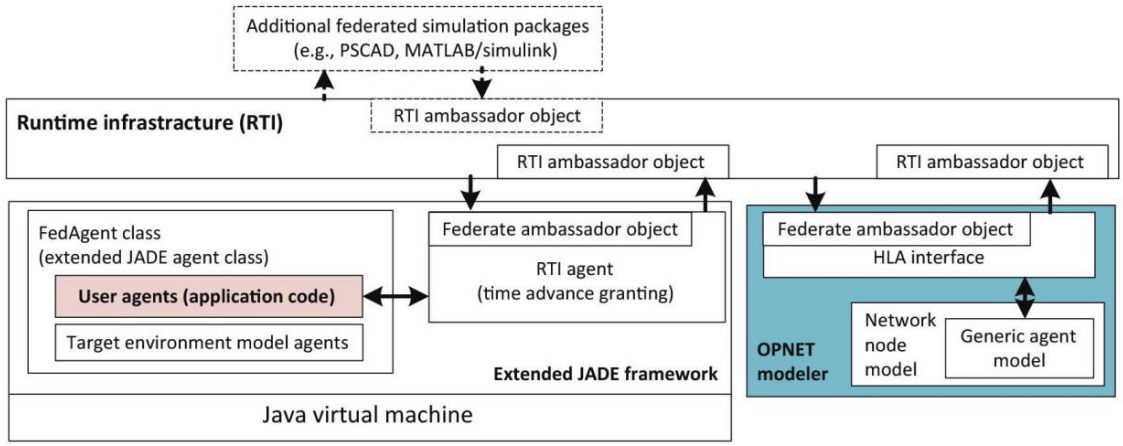

Fig. 2. Arquitectura general que combina el marco JADE extendido, la simulación OPNET extendida y la infraestructura de ejecución compatible con HLA. También se pueden añadir simulaciones adicionales, como simulaciones de flujo de potencia o transitorios eléctricos.

La arquitectura general del sistema propuesto, que consta de los siguientes tres componentes principales: 1) JADE; 2)
modelador
OPNET, comunicaciones NS; 3) RTI.

La aplicación de control distribuido se implementa mediante los agentes de usuario que se ejecutan en JADE, cada uno asociado con un nodo particular en la red de comunicación simulada (OPNET).
La RTI proporciona servicios comunes a todos los sistemas federados para el paso de mensajes, la gestión de objetos y la gestión del tiempo.
La posible inclusión de otras herramientas de simulación se indica en la figura 2 mediante los bloques de líneas punteadas.

Tanto JADE como OPNET Modeler están diseñados para ser herramientas independientes y, por lo tanto, es un desafío considerable para acoplarlos. OPNET Modeler es un simulador de eventos discretos y utiliza el concepto de tiempo de simulación. Una simulación de eventos discretos (DES) mantiene un reloj de simulación y procede cronológicamente de un evento al siguiente. Cada evento se etiqueta con una marca de tiempo y se mantiene en una lista de eventos hasta su ejecución. El evento con la marca de tiempo más pequeña se ejecuta a continuación y luego se elimina de la lista. Nuevos eventos pueden ser añadidos por otros eventos, pero sus marcas de tiempo tienen que ser mayores que el tiempo de simulación actual, es decir, no se pueden programar eventos pasados. JADE, por otro lado, ofrece una plataforma para ejecutar agentes como hilos independientes en tiempo real, pero no hay concepto de tiempo de simulación y eventos de simulación, lo que dificulta su sincronización con el simulador de eventos discretos OPNET. Para posibilitar la federación, se desarrollaron y realizaron extensiones de JADE y OPNET mediante estrategias de reimplementación, extensión de código intermedio y el uso de API externas. Con estas extensiones, esta arquitectura permite la ejecución sincronizada de MAS y modelos de red de comunicación.

El framework JADE original se amplió de la siguiente manera (Fig. 2). En primer lugar, la clase de agente JADE estándar se extendió a una nueva clase FedAgent para permitir la sincronización horaria. En segundo lugar, se proporcionó un nuevo agente RTI para controlar el avance temporal e interconectar JADE con RTI. Finalmente, el mapeo entre los agentes de usuario y los nodos de la red OPNET se realizó mediante tablas y se gestionó mediante el agente RTI.

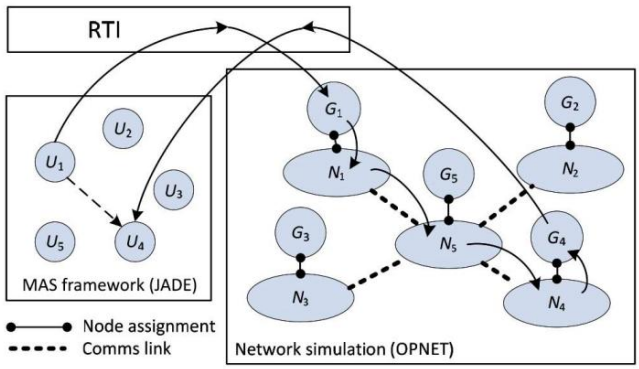

Fig. 3. Ilustración del paso de mensajes en la federación. Un mensaje enviado desde los agentes de usuario U1–U4 (flecha discontinua) se retransmite mediante RTI a la simulación de red y a los agentes genéricos G1 y G4, que forman parte de los modelos de nodos de red N1 y N4, respectivamente.

El modelador OPNET se amplió de la siguiente manera. Primero, desarrollamos el agente genérico, implementado como una extensión del nodo de red OPNET estándar. Esto proporciona un modelo genérico de los agentes de usuario que se ejecutan en JADE. Los agentes de usuario se asignan a sus agentes genéricos correspondientes, que se utilizan para transmitir todos los mensajes entre agentes en JADE a través de los enlaces de comunicación correctos en la red simulada.
En segundo lugar, se proporcionó integración con RTI basada en el estándar HLA [18]. Todos los mensajes que pasan entre OPNET y RTI son procesados por los agentes genéricos a través de la interfaz HLA.

Con estas extensiones implementadas, la Fig. 3 ilustra la ejecución sincronizada de una aplicación multiagente y un modelo de red de comunicaciones. Los agentes de usuario Un (n = 1, 2, 3,...) implementan la funcionalidad de la aplicación distribuida y los nodos Nn (n = 1, 2, 3,...) representan los dispositivos hardware donde se ejecutan los agentes. Los modelos de agente genéricos Gn (n = 1, 2, 3,...) son extensiones genéricas de la representación de nodos en OPNET y corresponden a agentes de usuario Un específicos.
El sistema general funciona como un simulador de eventos discretos, es decir, procede cronológicamente de un evento al siguiente.

En la ilustración de la Fig. 3, cuando la RTI otorga tiempo a JADE para ejecutar agentes que están programados para una hora específica, U1 envía un mensaje a U4 (indicado por la flecha punteada entre

U1 y U4). En lugar de entregar el mensaje directamente, JADE envía una interacción a OPNET que inicia una transferencia de datos en el NS. El mensaje originado en U1 se enruta mediante RTI al agente genérico correspondiente , G1, incorporado en el modelo del nodo de red N1. Posteriormente, se enruta a través de la red de comunicaciones simulada al nodo N4 y al agente genérico G4. Desde G4, el mensaje se transmite mediante RTI a JADE y al agente de usuario correspondiente, U4.

También es importante recordar que los agentes de usuario que implementan la aplicación distribuida, al implementarse en el sistema de destino, interactuarán con los dispositivos físicos locales mediante la recopilación de mediciones y la ejecución de acciones de control. Las soluciones tradicionales de automatización de subestaciones implican conexiones analógicas y binarias cableadas del controlador de la subestación a dispositivos específicos, como equipos de conmutación y medidores, pero una tendencia creciente es emplear unidades de fusión (MU) que se comunican con los dispositivos locales a través del bus de proceso, como IEC61850-9-2 [22], sobre Ethernet local. El análisis de rendimiento de dicho esquema fue realizado por Crossley et al. [23]. En ausencia del sistema de destino durante el desarrollo, el entorno local puede modelarse utilizando agentes de modelado del entorno de destino, también implementados en JADE (Fig. 2). Estos pueden diseñarse para comunicarse con los agentes de usuario según el estándar pertinente, como IEC61850 [22], y utilizarse para modelar el rendimiento de los dispositivos de hardware locales y configurar escenarios de prueba. También es posible ampliar la federación para incluir otros paquetes. Se pueden agregar herramientas de simulación de sistemas de potencia (por ejemplo, PSCAD, MATLAB/Simulink, PSLF y otras) de manera similar a como se mostró en EPOCHS [12].

Las siguientes secciones describen los tres componentes y su implementación con más detalle.

B. Infraestructura de tiempo de ejecución

RTI es un programa que proporciona servicios de modelado de simulación distribuidos. Algunos de estos servicios garantizan que todos los componentes de la simulación se inicien simultáneamente y avancen en el tiempo de forma sincronizada. Otros servicios ofrecen métodos para intercambiar información entre los componentes individuales mediante mecanismos de interacción, publicación y suscripción.

Para la RTI, hemos adoptado la implementación MÄK RTI 4.0.4 (MÄK Technologies, Cambridge, MA, EE. UU.) que se adhiere al estándar HLA [18], [19]. La versión inicial del estándar HLA HLA 1.3 se publicó en 1998 y más tarde se convirtió en un método estándar IEEE y se define bajo IEEE 1516 [18]. Si bien existen otras arquitecturas de modelado de simulación distribuida, como Common Object Request Broker (CORBA) y Java Remote Method Invocation (Java RMI), HLA es más adecuado para simulaciones debido a sus servicios específicos de simulación (por ejemplo, servicios de gestión del tiempo) [24].

El estándar HLA define principios generales, especificaciones de interfaz y una plantilla de modelo de objetos (OMT). Las especificaciones de interfaz definen la interfaz de programación de aplicaciones (API) entre los componentes de simulación y la RTI. La OMT, por otro lado, define la estructura permitida de los modelos de objetos que pueden intercambiarse entre los componentes de simulación. En consecuencia, cada aplicación federada comunica datos a la RTI a través del RTIAmbassador estandarizado. objeto y recibe datos de RTI a través del objeto FederateAmbassador.

C. Marco MAS—JADE

Si bien existe una gran cantidad de kits de herramientas de desarrollo de MAS de código abierto [25], la necesidad de herramientas basadas en estándares reduce significativamente la elección. El cumplimiento de los estándares garantiza la interoperabilidad entre MAS desarrollados con diferentes kits de herramientas. En los últimos años, la mayoría de los desarrolladores de MAS han adoptado los estándares de la Fundación para Agentes Físicos Inteligentes (FIPA).

Hemos adoptado el framework JADE, compatible con FIPA, que, según [4], también se ha convertido en uno de los favoritos de los investigadores en ingeniería eléctrica. Es de código abierto y está completamente implementado en Java. El framework también proporciona un conjunto de herramientas gráficas que pueden ser especialmente útiles en las etapas de depuración. Cabe destacar que JADE es compatible con una amplia gama de plataformas, incluyendo controladores embebidos y dispositivos móviles, gracias a su bajo consumo de memoria.

Como se mencionó anteriormente, hemos ideado un enfoque novedoso para controlar la ejecución de los agentes que se ejecutan en JADE, con el fin de sincronizar los avances temporales y el intercambio de mensajes con el modelador OPNET y cualquier otra simulación dependiente del tiempo. Esto se implementó como una extensión DES del framework JADE, lo que permite que los agentes se ejecuten como parte de un sistema DES.

El marco estándar de JADE proporciona la clase de agente en la que se basa. Nuestra extensión DES de JADE proporciona la clase FedAgent. Esta se deriva de la clase de agente y, por lo tanto, proporciona todas las API del marco de JADE, así como las API adicionales necesarias para controlar la ejecución externamente y redirigir el intercambio de mensajes entre agentes. Por otro lado, el control del avance discreto en el tiempo y el enrutamiento de mensajes lo realiza el agente RTI, desarrollado adicionalmente. El agente RTI también proporciona una interfaz compatible con HLA para el enrutamiento de todos los mensajes y mantiene la tabla de mapeo que asocia agentes de usuario específicos en JADE con sus nodos de red correspondientes en OPNET (agentes genéricos).

Con este enfoque, una aplicación distribuida puede implementarse como un conjunto de agentes de usuario y verificarse mediante el entorno de desarrollo federado. La implementación posterior en el sistema de destino solo involucra a los agentes de usuario que se ejecutan en la plataforma JADE.

Nuestra extensión DES del marco JADE también puede tener en cuenta el tiempo de procesamiento en cada nodo, lo que fue una estimación para los fines de este trabajo, mientras que, en la práctica, se evaluaría con precisión para las plataformas específicas empleadas.

D. Marco de simulación de redes de comunicación: el modelador OPNET

Existen varios sistemas de comunicación NS disponibles, de los cuales NS2 y el modelador OPNET son, con diferencia, los más utilizados. NS2 es un software de código abierto, lo que le ha dado popularidad en la comunidad investigadora. Sin embargo, se eligió OPNET para este trabajo porque ofrece una extensa biblioteca de modelos y un amplio conjunto de protocolos y tecnologías estándar. Además,

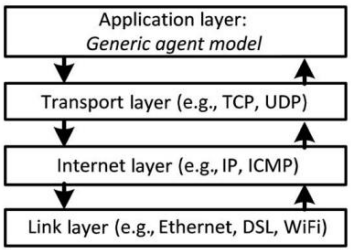

Fig. 4. Modelo de agente genérico como parte del modelo de nodo OPNET extendido, que representa la capa de aplicación del modelo OSI.

Proporciona un entorno gráfico para el desarrollo de modelos, simulación y análisis de datos, y permite la creación rápida de modelos de red.

En el modelador OPNET, un modelo de red de comunicaciones se compone principalmente de modelos de nodos, que modelan dispositivos conectados mediante canales de comunicación. Aunque no existe una funcionalidad para modelar MAS explícitamente, es posible extender los modelos de nodos de red con código desarrollado por el usuario.

Esta función se utilizó para ampliar el modelo de nodo de OPNET e incorporar el modelo de agente genérico, con el fin de proporcionar una representación de los agentes de usuario correspondientes, codificados en la plataforma JADE. Dentro de OPNET, esto implementa la capa de aplicación, conectada a la capa de transporte del modelo de interconexión de sistemas abiertos (OSI) (ISO/IEC7498-1), como se muestra en la figura 4.

El programador central del modelador OPNET ofrece la opción de ser controlado externamente a través de un RTI compatible con HLA. Este módulo básico se ha ampliado para gestionar notificaciones. que se reciben o envían a la plataforma MAS.

E. Despliegue

Nuestras extensiones JADE introducen tiempo de simulación y son compatibles con la API de JADE estándar. Al implementar la aplicación MAS en un sistema de destino, los agentes implementados solo utilizan el marco de trabajo JADE estándar.

Es bien sabido que la recolección automática de basura en Java estándar puede introducir retrasos impredecibles en tiempo de ejecución. Para la plataforma de cosimulación propuesta, esto tiene poca relevancia, ya que es un DES e incluye un reloj de simulación controlado por el programa. Por lo tanto, los resultados de la simulación no se ven afectados por el recolector de basura de Java ni por interrupciones o retrasos causados por otros procesos que se ejecutan simultáneamente en el sistema operativo Linux.

Sin embargo, para la implementación, se recomienda utilizar la máquina virtual Java en tiempo real (RTJVM) y un sistema operativo en tiempo real (como Linux en tiempo real) para evitar retrasos impredecibles que podrían producirse debido a la recolección de elementos no utilizados. Diversos proveedores ofrecen máquinas virtuales Java que cumplen con la especificación de tiempo real para Java (RTSJ) [17], [26] [27].

III. ESTUDIO DE CASO DE SUPERVISIÓN DE RELÉ BASADO EN AGENTES

La supervisión remota de relés de respaldo basada en agentes es representativa del tipo de aplicación distribuida y de tiempo crítico que involucra MAS, que surge en las redes inteligentes. Garlapanti et al. [20] propusieron

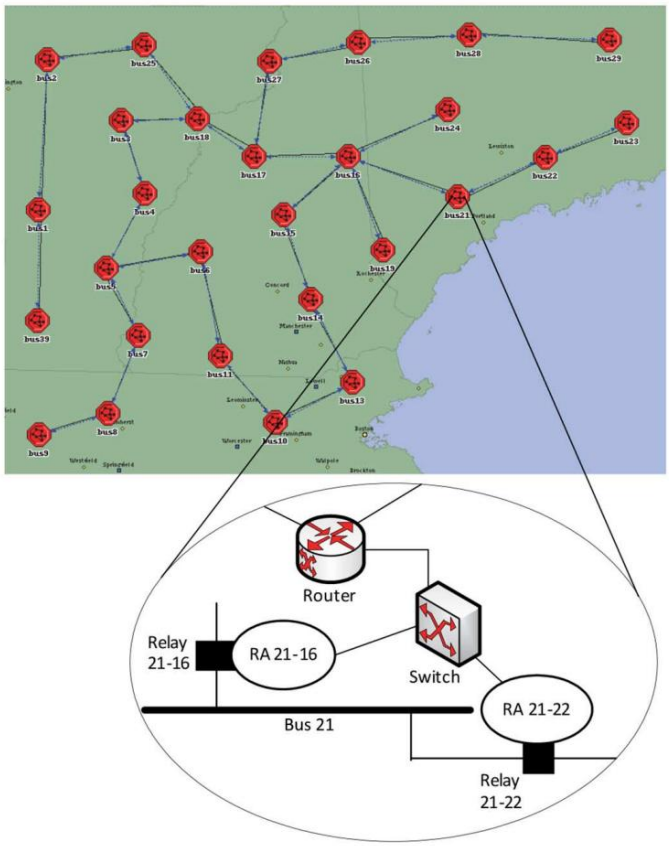

Fig. 5. Modelo de red de comunicaciones que cubre el área de Nueva Inglaterra. Cada subestación consta de varios agentes de retransmisión (RA), conmutadores de red y un enrutador. La vista ampliada muestra una vista detallada de la subestación con la barra 21.

Se diseñó un método basado en agentes para la supervisión de relés de zona 3 y se evaluó su rendimiento en relación con el sistema IEEE 39-bus. Se realizó una simulación del sistema de potencia del modelo IEEE 39-bus utilizando PSLF para estudiar las posibles rutas de fallos en cascada bajo el deslastre de carga y la protección del generador modelados. La traza de eventos registrada en esta simulación se utilizó para definir una simulación NS2 posterior de la red de comunicaciones y estimar los retrasos que se producirían en ella.

En este trabajo, implementamos la aplicación de supervisión basada en agentes propuesta utilizando JADE. La aplicación se probó junto con el sistema de comunicaciones NS, donde todos los intercambios de mensajes entre agentes se modelaron con precisión mediante OPNET. Se configuraron fácilmente diferentes arquitecturas de aplicación. El rendimiento se evaluó utilizando el sistema federado descrito para evaluar el rendimiento de diferentes configuraciones de aplicaciones MAS, protocolos, estrategias de QoS y condiciones de red. Con base en esto, se puede seleccionar y validar la estrategia de comunicación óptima en relación con los criterios de rendimiento previamente determinados. Esto se presenta con más detalle en las siguientes secciones.

A. Método de supervisión del relé de respaldo de la zona 3

La figura 5 muestra el sistema IEEE 39-bus de Nueva Inglaterra, con la infraestructura de red de comunicaciones de acuerdo a la

[20]. Los relés de protección están ubicados en cada extremo de las líneas de transmisión que conectan buses adyacentes [28] y se supone que todos son relés de impedancia direccional (Mho).

Las zonas de protección se definen convencionalmente en relación con la protección de las líneas eléctricas que conectan buses adyacentes y un relé ubicado en un extremo de la línea eléctrica [28]. Los relés de protección de distancia detectan la impedancia entre el relé y una falla como la relación entre las magnitudes de las corrientes y las tensiones.
Dado que la impedancia varía casi linealmente con la distancia, también se puede determinar la ubicación de la falla. Para un relé dado, la protección de la zona 1 generalmente corresponde al 85%-90% de la longitud de la línea y opera instantáneamente. Para evitar puntos ciegos cerca de un bus, se establece otra zona de protección, la zona 2, al 120%-150% de la longitud de la línea. Se permite un retardo de coordinación de 0,3 s antes de que la protección de la zona 2 se active. Una tercera zona de protección, la zona 3, se configura como respaldo adicional, en caso de que fallen las protecciones de las zonas 1 y 2, y cubre el 100% de la línea donde se ubica el relé y el 120%-180% de la línea contigua.
Para el funcionamiento de la zona 3 se permite un retardo de coordinación de 1 s.

Por lo tanto, cuando un relé de respaldo de la zona 3 detecta una falla, espera, dependiendo de la topología de la red eléctrica, aproximadamente 1 s a que los relés correspondientes de las zonas 1 y 2 reaccionen primero y aíslen la falla. Si, después de este retardo, la falla persiste, el relé de respaldo de la zona 3 siempre dispara su línea. En caso de una falla real, este comportamiento es intencionado y bienvenido. Sin embargo, fallas ocultas debido a errores de software o hardware podrían provocar que un relé de la zona 3 se dispare y retire la carga innecesariamente.

El objetivo de la supervisión de relés basada en agente es evitar la activación innecesaria de los relés de respaldo remotos de la zona 3 debido a fallos ocultos. Estos fallos pueden ocurrir como resultado de errores de software o hardware en el relé de la zona 3. Son relativamente poco frecuentes, pero pueden pasar desapercibidos durante mucho tiempo y provocar una sobresensibilidad del relé de la zona 3, que podría activarse erróneamente y provocar un fallo en cascada.

El sistema de supervisión propuesto [20] asume que cada relé de protección puede ejecutar un agente de software que puede acceder y controlar el relé. A este agente lo llamaremos RA en el resto del texto. En algunas implementaciones, el RA puede ejecutarse en el controlador de automatización de la subestación local, como se describe en [23], y controlar el relé a través de un bus de campo local o una red de automatización de la subestación.

En la situación descrita anteriormente, una RA de este tipo puede recopilar información de estado de otras RA mientras espera determinar con mayor precisión si la falla es genuina. Dado que la recopilación de información de la RA debe realizarse en un plazo muy breve, esta aplicación es sumamente crítica en términos de tiempo y su validación debe considerar con precisión los retrasos en la comunicación.

B. Configuración MAS

Se crearon dos MAS diferentes en JADE y se analizaron.
Ambos sistemas ofrecen la misma funcionalidad general, pero su enfoque de comunicación de RA se basa en un comportamiento cliente-servidor (c/s) o punto a punto (p2p), como se muestra en la Fig. 6. Ambos sistemas implican un DMA que es necesario para mantener información actualizada sobre la configuración de los RA y sus asignaciones de las zonas 1, 2 y 3. Sin embargo, es importante destacar que el correspondiente

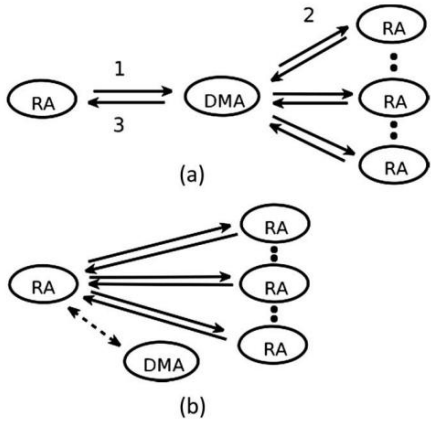

Figura 6. (a) Estrategia C/S y (b) P2P de los RA. El agente maestro de dominio (DMA) conoce la topología del sistema de bus y se retransmite para su dominio. Los números indican la secuencia de intercambio de mensajes.

La autoridad de decisión de los agentes y las rutas de mensajería difieren mucho entre ambos esquemas. Esto puede generar grandes diferencias en los retrasos de comunicación.

1) Arquitectura C/S: La Fig. 6(a) ilustra que, en este caso, la ejecución se centraliza en el DMA. Cuando un RA de zona 3 detecta una falla, solicita inmediatamente ayuda al DMA [mensaje 1 en la Fig. 6(a)]. El DMA dispone de la información necesaria para saber cuál de los otros RA podría detectar la misma falla. Envía una solicitud de información a todos los RA potenciales en paralelo y espera todas sus respuestas [mensaje 2 en la Fig. 6(a)]. Al recibir todas las respuestas, el DMA decide si se trata de una falla real y envía su decisión al RA inicial [mensaje 3 en la Fig. 6(a)]. Con esta información, el RA de zona 3 dispara o no el relé después del tiempo de espera definido (es decir, 1 s).

2) Arquitectura P2P: La Fig. 6(b) ilustra la naturaleza descentralizada de esta arquitectura. Se diferencia de la arquitectura c/s en que el RA de la zona 3 contacta directamente con los RA potenciales sin pasar por el DMA. La función del DMA es enviar la lista de agentes pares al relé de la zona 3 con el que debe contactar en caso de detectarse un fallo. Esto ocurre cuando los RA están activados y la lista solo debe actualizarse cuando cambia la topología o se añaden o eliminan RA. Dado que el DMA no participa en la comunicación crítica en cuanto al tiempo, se espera que este método tenga un mejor rendimiento que el enfoque c/s en cuanto al tiempo de ida y vuelta del mensaje.

El MAS distribuido completo estaba compuesto por 81 RA y un DMA. Toda la comunicación entre agentes cumple con el estándar FIPA-ACL y la carga útil de sus mensajes consistía en los identificadores del emisor y el receptor del agente, así como el estado de los relés. El tamaño de la carga útil del mensaje del agente era de 250 bytes, sin incluir las cabeceras de protocolos de comunicación como TCP, IP y Ethernet. También creamos agentes que enviaban 350 bytes de carga útil para estudiar el impacto de los diferentes tamaños de mensaje. Para este trabajo, el modelo asumió que el tiempo de ejecución de los comportamientos en el hardware de destino es de 1 ms.

C. Modelo de red

La figura 5 muestra la red de comunicación modelada que conecta las subestaciones del sistema IEEE de 39 buses en toda el área.

de Nueva Inglaterra. Los objetos circulares representan la red de área local (LAN) de una subestación, compuesta por varias RA, conmutadores de red y un enrutador. Las RA están conectadas mediante Ethernet (IEEE 802.3) a los conmutadores, que a su vez están conectados al enrutador (véase la vista detallada de la subestación con el bus 21 en la Fig. 4). El enrutador es la puerta de enlace a otras subestaciones y, en este ejemplo, está conectado mediante enlaces de comunicación T1 (1,544 Mb/s) a otras subestaciones.

Creamos tres escenarios de red diferentes basados en esto red de comunicación.

1) Sin tráfico de fondo (noBT): esto supone que los enlaces T1 están dedicados a la comunicación RA y no se utilizan para transferir ningún otro tráfico.
2) Tráfico de fondo (BT): La comunicación del agente debe competir con un BT de 1,39 Mb/s en los enlaces T1. Esto representó una utilización del enlace de aproximadamente el 90 %.
3) QoS: Se implementó una estrategia de QoS para ayudar a la comunicación del agente a competir contra el BT de 1,39 Mb/s. Se habilitó la cola justa ponderada basada en clases (CBWFQ) con una cola de baja latencia (LLQ) en las interfaces de red de los enrutadores. La LLQ gestionó todo el tráfico marcado como tráfico de agente, mientras que el BT se sometió a la cola justa ponderada.

Los dos casos de BT representan extremos de carga en la red de comunicaciones. Un caso de uso realista, basado en la estrategia de QoS, probablemente se situaría en un punto intermedio entre estos dos niveles de rendimiento. Todos los escenarios se simularon para los dos protocolos más comunes, TCP y UDP, utilizados para la comunicación entre agentes. Sin necesidad de modelado adicional, esto permitió comparar el protocolo TCP, más lento pero fiable, con el protocolo UDP, más rápido pero menos fiable.

Por lo tanto, el modelo general de red distribuida constaba de 28 LAN con 28 enrutadores, 28 conmutadores de red y un total de 82 modelos de agentes genéricos. La DMA se ubicaba en la subestación del bus 18.

IV. RESULTADOS DE LA SIMULACIÓN

A. Rendimiento del MAS en diferentes escenarios

Se ejecutaron 24 escenarios de simulación diferentes, que abarcaban distintos escenarios de red (noBT, BT y QoS), arquitecturas de agente (c/s y p2p), tamaños de mensaje (250 y 350 bytes) y protocolos de comunicación (TCP y UDP). En cada escenario, se simuló consecutivamente la detección de un fallo para cada RA del sistema. Cada escenario se simuló de esta manera 120 veces, lo que supone un total de 2900 simulaciones. Se registraron y analizaron los tiempos de decisión de los RA.

La figura 7 resume los resultados, mostrando los tiempos de decisión máximos y promedio registrados para todos los RA. Los resultados sin BT (noBT-c/s y noBT-p2p) indican los límites inferiores teóricos para la capacidad de enlace dada, alcanzables únicamente mediante una red dedicada a esta aplicación. Todos los métodos considerados cumplirían los requisitos en estas condiciones. Sin embargo, en presencia de BT, la elección de tecnologías y estrategias de comunicación se vuelve crucial. Como era de esperar, el protocolo TCP presenta tiempos significativamente mayores que el UDP, mientras que el tamaño de la carga útil del mensaje también presenta una notable...

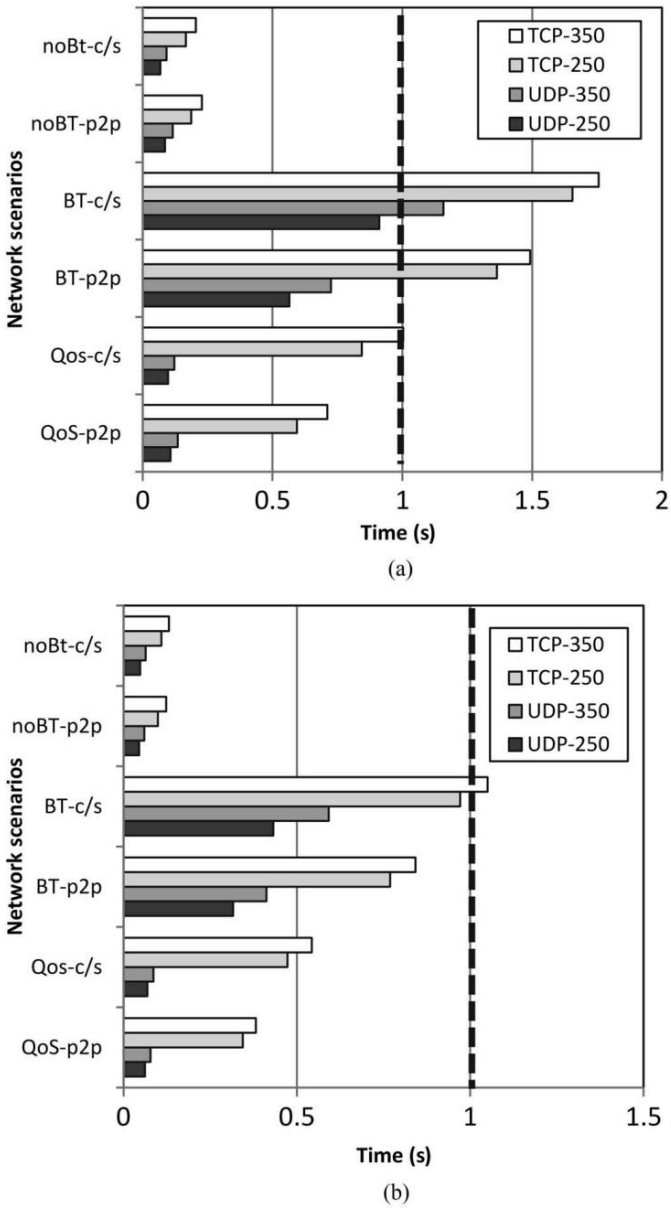

Fig. 7. (a) Tiempo máximo y (b) tiempo promedio que tardan todos los AR en tomar una decisión en diferentes escenarios de BT y MAS. El tiempo máximo de decisión especificado es de 1 s.

Si se utiliza el protocolo TCP, se debe emplear una estrategia de QoS, pero el tiempo de respuesta solo se puede garantizar para una carga útil de mensaje de 250 bytes.

B. Efecto de la falla del enlace

Se realizaron simulaciones adicionales para analizar el efecto de una falla en el enlace entre las subestaciones con los buses 11 y 10. La figura 8 muestra el aumento correspondiente en el tiempo de decisión para diferentes agentes. Esta información es importante para la planificación de la red y el análisis del modo de falla del sistema.

C. Posición óptima del nodo para el DMA

En los casos anteriores, DMA se asignó para ejecutarse en la subestación del bus 18, pero puede configurarse para ejecutarse en cualquier otro nodo.

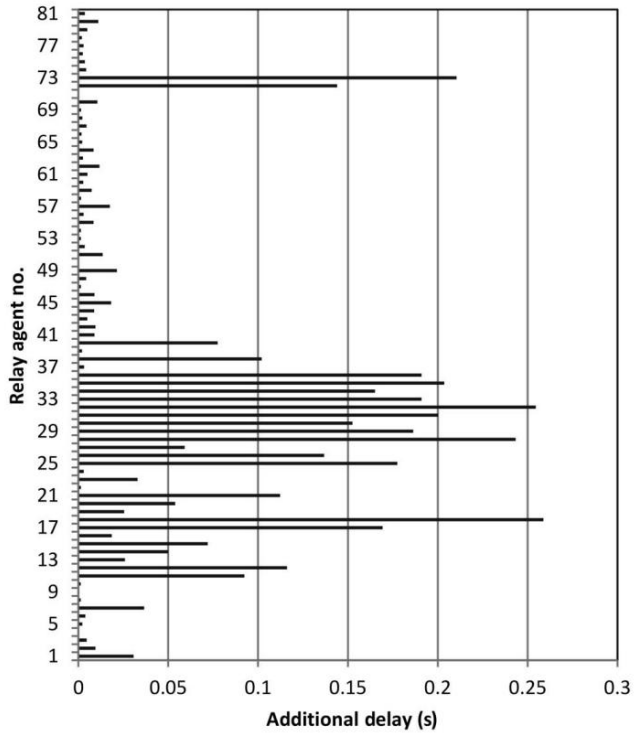

Fig. 8. Impacto de una falla en el enlace de comunicación entre las subestaciones con barras 11 y 10. Los tiempos muestran el incremento en el tiempo de decisión para cada RAs en comparación con la operación normal.

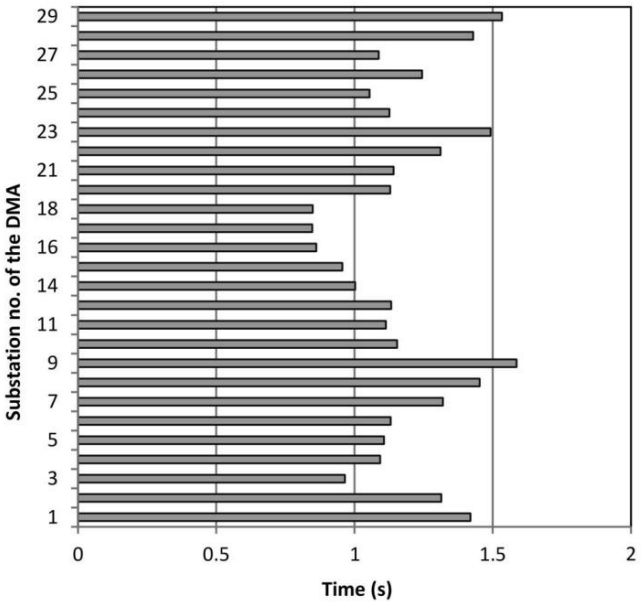

Fig. 9. Tiempos promedio que tardaron los RA con peor desempeño en tomar una decisión sobre la subestación elegida para ejecutar el DMA. El escenario incluyó el enfoque cliente/servidor, BT, QoS, TCP y un tamaño de mensaje de 250 bytes.

de la red. Se ejecutaron simulaciones para el escenario con c/s, BT-QoS, TCP y una carga útil de mensaje de 250 bytes, con el DMA desplegado en diferentes subestaciones. Se ejecutaron un total de 50 simulaciones para cada caso. La figura 9 muestra los tiempos promedio que tardan los RA con peor rendimiento en tomar una decisión, mostrando diferencias significativas en el rendimiento y que las subestaciones de los buses 16, 17 y 18 son las mejores opciones.

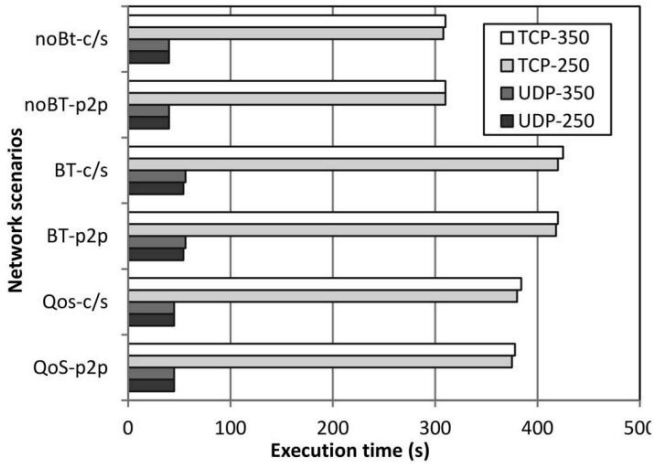

Fig. 10. Tiempos de ejecución para una ejecución de simulación que involucra diferentes escenarios de comunicación (PC Linux, procesador Intel i7, 8 Gb de RAM).

D. Tiempo de ejecución de la simulación

Se observó que el tiempo de ejecución de la simulación fue dominado por el tiempo utilizado por el modelador OPNET para simular la comunicación, mientras que la ejecución del código del agente en JADE y el paso de mensajes vía RTI fueron muy pequeños en comparación.
La figura 10 muestra los tiempos de ejecución totales de una simulación con diferentes escenarios de comunicación en un PC Linux (procesador Intel i7-2700K a 3,9 GHz y 16 GB de RAM). Las simulaciones con TCP tardaron aproximadamente 350 s, en comparación con los tan solo 50 s de las simulaciones con UDP, debido a la relativa complejidad de la simulación.

V. CONCLUSIÓN

El desarrollo de aplicaciones distribuidas de tiempo crítico para la automatización de la red inteligente plantea importantes desafíos a la hora de elegir estrategias de comunicación óptimas y cumplir con los requisitos de rendimiento prescritos. La principal ventaja del sistema propuesto reside en que permite probar el código de la aplicación MAS mediante una simulación precisa de la red de comunicaciones objetivo específica. Resulta especialmente importante la capacidad de optimizar la implementación bajo criterios de rendimiento potencialmente conflictivos, como la calidad de servicio (QoS) frente a la velocidad de respuesta. El sistema combina JADE, una plataforma completa para el desarrollo e implementación de aplicaciones multiagente, con OPNET, una potente herramienta de simulación de redes de comunicaciones. Las extensiones necesarias para integrar estas herramientas se desarrollaron y demostraron con éxito en una aplicación real que implica la protección de una red eléctrica de área extensa. Se prevé que la red inteligente del futuro necesitará emplear un gran número de aplicaciones de este tipo, así como muchas otras con menor tiempo crítico, que compartirán la infraestructura de comunicación subyacente. El modelado preciso de las comunicaciones de datos y el uso del código de la aplicación real proporcionan un alto grado de confianza en los datos de rendimiento obtenidos antes de la implementación en el sistema objetivo.

RECONOCIMIENTO

Los autores desean agradecer al Consejo de Investigación en Ingeniería y Ciencias Físicas por financiar esta investigación y al Programa Universitario OPNET por proporcionar una licencia gratuita para el uso de su software.

REFERENCIAS

[1] VC Gungor et al., “Tecnologías de redes inteligentes: tecnologías y estándares de comunicación”, IEEE Trans. Ind. Informat., vol. 7, n.º 4, págs. 529–539, noviembre de 2011.

[2] M. Ruofei, C. Hsiao-Hwa, H. Yu-Ren y M. Weixiao, “Comunicación en redes inteligentes: sus desafíos y oportunidades”, IEEE Trans. Smart Grid, vol. 4, n.º 1, págs. 36–46, marzo de 2013.

[3] Q. Yang, JA Barria y TC Green, “Infraestructuras de comunicación para el control distribuido de redes de distribución de energía”, IEEE Trans. Ind. Informat., vol. 7, no. 2, pp. 316–327, May 2011.

[4] SD McArthur et al., “Sistemas multiagente para aplicaciones de ingeniería energética: Parte I: Conceptos, enfoques y desafíos técnicos”, IEEE Trans. Power Syst., vol. 22, no. 4, págs. 1743–1752, noviembre de 2007.

[5] SD McArthur et al., “Sistemas multiagente para aplicaciones de ingeniería energética—Parte II: Tecnologías, estándares y herramientas para construir sistemas multiagente”, IEEE Trans. Power Syst., vol. 22, n.º 4, págs. 1753–1759, noviembre de 2007.

[6] DV Coury, JS Thorp, KM Hopkinson y KP Birman, “Un relé diferencial de corriente basado en agente para usar con una intranet de servicios públicos”, IEEE Trans. Power Del., vol. 17, no. 1, pp. 47–53, Jan. 2002.

[7] F. Ren, M. Zhang, D. Soetanto y X. Su, “Diseño conceptual de un sistema multiagente para la restauración de sistemas de energía interconectados”, IEEE Trans. Power Syst., vol. 27, no. 2, págs. 732–740, mayo de 2012.

[8] HS Nwana, DT Ndumu, LC Lee y JC Collis, “ZEUS: Un conjunto de herramientas y un enfoque para la construcción de sistemas distribuidos multiagente”, en Proc. 3.ª Conferencia Anual de Agentes Autonómicos, 1999, págs. 360-361.

[9] F. Bellifemine, F. Bergenti, G. Caire y A. Poggi, “JADE: Un marco de desarrollo de agentes Java”, en Programación Multiagente. Nueva York, NY, EE. UU.: Springer, 2005, págs. 125–147.

[10] AL Dimeas y ND Hatziargyriou, “Operación de un sistema multiagente para el control de microrredes”, IEEE Trans. Power Syst., vol. 20, n.º 3, págs. 1447–1455, agosto de 2005.

[11] H. Wang, “Coordinación multiagente para el control de tensión secundaria en contingencias del sistema eléctrico”, IEE Proc. Gener. Transmiss. Distrib., vol. 148, n.º 1, págs. 61–66, enero de 2001.

[12] K. Hopkinson et al., “EPOCHS: Una plataforma para la simulación de energía eléctrica y comunicación basada en agentes, construida a partir de componentes comerciales listos para usar”, IEEE Trans. Power Syst., vol. 21, n.º 2, págs. 548–558, mayo de 2006.

[13] H. Lin, SS Veda, SS Shukla, L. Mili y J. Thorp, “GECO: Marco de co-simulación global impulsado por eventos para sistemas de energía interconectados y redes de comunicación”, IEEE Trans. Smart Grid, vol. 3, n.º 3, págs. 1444–1456, septiembre de 2012.

[14] K. Mets, J. Ojea y C. Develder, “Combinación de simulación de redes de energía y comunicación para un análisis rentable de redes inteligentes”, IEEE Commun. Surv. Tuts., vol. 16, n.º 3, págs. 1–26, agosto de 2014.

[15] RJ Allan, Encuesta sobre herramientas de simulación y modelado basadas en agentes. Swindon, Reino Unido: Consejo de Instalaciones Científicas y Tecnológicas, 2010.

[16] FL Bellifemine, G. Caire y D. Greenwood, Desarrollo de sistemas multiagente con JADE, vol. 7. Hoboken, NJ, EE. UU.: Wiley, 2007.

[17] G. Bollella y J. Gosling, “La especificación en tiempo real para Java”, Computadora, vol. 33, págs. 47–54, 2000.

[18] Comité SIS de la IEEE Computer Society: Estándar IEEE para Modelado y Simulación (M&S) Arquitectura de Alto Nivel (HLA). Estándar IEEE 1516-2000, 1516.1-2000, 1516.2-2000, 2000.

[19] S. Straßburger, “Sobre el acoplamiento de herramientas de simulación basado en HLA”, en Proc. Eur. Simul. Multiconf., 1999, págs. 45–51.

[20] S. Garlapati, H. Lin, S. Sambamoorthy, SK Shukla y J. Thorp, “Supervisión basada en agente de relés de zona 3 para evitar disparos basados en fallas ocultas”, en Proc. 1st IEEE Int. Conf. Smart Grid Commun. (SmartGridComm), 2010, págs. 256–261.

[21] Estándar IEEE para Modelado y Simulación (M&S) Arquitectura de Alto Nivel (HLA)–Especificación de Interfaz Federada, Estándar IEEE 15161-2010 (Revisión del Estándar IEEE 1516.1–2000), 2010, págs. 1–378.

[22] A. Apostolov y B. Vandiver, “Bus de proceso IEC 61850: Principios, aplicaciones y beneficios”, en Proc. 63.ª Conferencia Anual, Prot. Relay Eng., 2010, págs. 1–6.

[23] L. Yang, PA Crossley, A. Wen, R. Chatfield y J. Wright, “Diseño y pruebas de rendimiento de un esquema de protección basado en bus de proceso IEC61850–9-2 de múltiples proveedores”, IEEE Trans. Smart Grid, vol. 5, n.º 3, págs. 1159–1164, mayo de 2014.

[24] A. Buss y L. Jackson, “Modelado de simulación distribuida: una comparación de HLA, CORBA y RMI”, en Proc. 30th Conf. Winter Simul., 1998, págs. 819–826.

[25] E. Shakshuki e Y. Jun, “Kits de herramientas de desarrollo multiagente: Una evaluación”, en Innovaciones en Inteligencia Artificial Aplicada. Nueva York, NY, EE. UU.: Springer, 2004, págs. 209-218.

[26] SG Robertz y R. Henriksson, “Recolección de basura activada por tiempo: Programación de GC robusta y adaptativa en tiempo real para sistemas integrados”, en Proc. ACM SIGPLAN Not., 2003, págs. 93-102.

[27] MH Dawson, “Desafíos en la implementación de la especificación de tiempo real para Java (RTSJ) en una máquina virtual Java en tiempo real comercial”, en Proc. 11th IEEE Int. Symp. Object Oriented Real Time Distribut. Comput. (ISORC), 2008, págs. 241–247.

[28] PM Anderson, Protección del sistema eléctrico. Piscataway, NJ, EE. UU.: IEEE Prensa, 1999.

Fidelis Perkonigg recibió la licenciatura y la maestría en ciencias de la computación de la Universidad Tecnológica de Graz, Graz, Austria, en 2006, y el doctorado en ingeniería mecánica del Imperial College London, Londres, Reino Unido, en 2015.

Trabajó en Cisco Systems, Viena, Austria, antes de su doctorado y es un Profesional Certificado en Enrutamiento, Conmutación, Seguridad y Voz de Cisco (nivel CCNP). Actualmente es Profesor Titular del Departamento de Informática del Imperial College de Londres. Sus intereses de investigación incluyen redes informáticas, sistemas operativos e ingeniería de software.

Djordje Brujic (M'95) obtuvo su título de Ingeniero Eléctrico en la Universidad de Belgrado, Serbia, en 1975; su maestría en Sistemas de Control en la Universidad Tecnológica de Cranfield, Bedford, Reino Unido, en 1985; y su doctorado en Modelado Geométrico en el Imperial College London, Londres, Reino Unido, en 2002.

Es investigador sénior del Imperial College de Londres. Sus intereses de investigación abarcan sistemas de control en tiempo real, reconstrucción de superficies tridimensionales y, más recientemente, sistemas distribuidos y metodologías multiagente aplicadas a problemas de redes inteligentes. Ha publicado más de 50 artículos en estos campos.

Mihailo Ristic (M'94) recibió el título en ingeniería mecánica del University College London, Londres, Reino Unido, en 1981, y el título de maestría en sistemas de control y el título de doctorado en robótica del Imperial College London, Londres, en 1982 y 1986, respectivamente.

Actualmente es profesor titular del Departamento de Ingeniería Mecánica del Imperial College de Londres. Sus intereses de investigación abarcan una amplia gama de temas relacionados con sistemas de control, diseño asistido por computadora, máquinas eléctricas y resonancia magnética. Es cofundador de Turbo Power Systems, Gateshead, Reino Unido, empresa especializada en máquinas eléctricas de alta velocidad y electrónica de potencia.

El Dr. Ristic es ingeniero colegiado y miembro de la Institución de Ingenieros Mecánicos.