Platform for Multiagent Application Development Incorporating Accurate Communications Modeling

Fidelis Perkonigg, Djordje Brujic, *Member, IEEE*, and Mihailo Ristic, *Member, IEEE*

***Abstract*—Multiagent systems are widely recognized as a method of choice for realization of distributed time-critical applications for the smart grid. However, no general solutions have been proposed for the difficult task of system development and validation, ready for deployment, which would fully account for the underlying communication network performance. We propose a novel platform designed for this purpose, which integrates a standard multiagent development framework [Java Agent Development (JADE)] and an industry standard communications network simulator (OPNET modeler). It was realized through generic extensions of the JADE framework to provide discrete event scheduling capabilities, while the OPNET modeler was extended to provide a generic method of associating the network nodes with agents running in JADE. The adopted method adheres to the high-level architecture standard. Importantly, applications developed using this platform may be deployed on the target system without manual modifications. A distributed protection application is presented and the performance is analyzed with respect to candidate agent behaviors and communication scenarios, demonstrating that the feasibility of the application critically depends on the choices made during its design and implementation.**

***Index Terms*—Communication networks, multiagent systems (MASs), smart grid.**

I. Introduction

MULTIAGENT systems (MASs) are now widely recognized as the preferred approach for the development of distributed applications, which involve multiple autonomous units that communicate with each other, such that their coordinated actions address a common goal. In recent years, MASs have become increasingly important in the context of power systems, particularly within the concept of smart grid [1], where intelligent agent technology was shown to be a promising method to automate numerous tasks related to grid management and control. Distributed applications for the smart grid may involve autonomous nodes that are physically separated by tens or even hundreds of kilometers and rely on the available data networks for communication [1]–[3]. This is seen as a significantly more flexible alternative than the traditional centralized supervisory control and data acquisition (SCADA) systems [1], [2]. For time-critical applications, such as grid protection and control, communication delays are a key factor in determining the feasibility and reliability of a particular distributed application. Examples of distributed applications based on MAS include power system protection [1], restoration [4], [5], diagnostics [5], voltage control [6], and control of microgrids [7]. Safety, robustness, and performance characteristics under different network traffic conditions are clearly key issues to address when attempting to develop a scheme of this type. However, the process of development, verification, and deployment of time-critical MAS applications has not been adequately addressed to date.

Fig. 1 shows the development stages for such application, starting with the prototype design of the control method and its analysis using simulation. This results in the specification of performance parameters and constraints (such as permissible latencies) in which the implemented method must meet. The application code will then be implemented using the chosen programming language and the run-time platform. Multiagent development platforms, such as ZEUS [8] or Java Agent Development (JADE) [9], have been adopted by numerous researchers in the field of power systems for this purpose [1]–[7], [10], [11].

At the implementation stage, a number of critical choices and tradeoffs have to be made, including specific agent behaviors, choice of physical network nodes on which certain critical agents are running, communication protocols, and quality-of-service (QoS) strategies. The implemented application code also needs to be analyzed in relation to the specific network components (routers and switches) and data traffic conditions for the target communication network. Geographic dispersion of the target hardware and the risks of damage to vital equipment make it imperative to conduct testing and validation offline.

The deployment on the distributed target system requires fully tested application code and no (or a very minimum) manual code modification. With this in mind, the introduction of MAS to the power industry makes the whole development life cycle of time-critical control and protection applications significantly more challenging.

Previous research related to smart grid has mainly focused on prototype design and simulation of distributed control methods [12]–[14], but the need to test and validate the implemented applications, in a coded form ready for deployment, has not been adequately addressed. This forms the main motivation for the work presented in this paper.

Manuscript received June 16, 2014; revised February 09, 2015; accepted April 15, 2015. Date of publication April 30, 2015; date of current version June 02, 2015. This work was supported by the Energy Programme of the Engineering and Physical Sciences Research Council (EPSRC) Supergen HiDEF Project EP/G031681/1, which is a Research Councils United Kingdom (RCUK) cross-council initiative led by EPSRC. Paper no. TII-14-1349.

F. Perkonigg is with the Department of Computing, Imperial College London, London SW7 2AZ, U.K. (e-mail: f.perkonigg10@imperial.ac.uk).

D. Brujic and M. Ristic are with the Department of Mechanical Engineering, Imperial College London, London SW7 2AZ, U.K. (e-mail: d.brujic@imperial.ac.uk; m.ristic@imperial.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2015.2428633

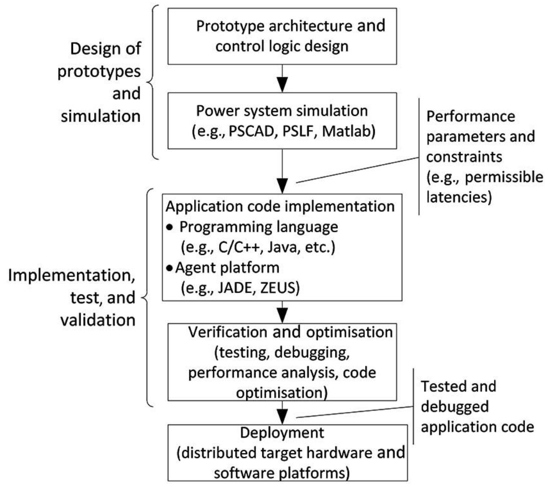

Design of prototypes and simulation
Prototype architecture and control logic design
Power system simulation (e.g., PSCAD, PSLF, Matlab)
Performance parameters and constraints (e.g., permissible latencies)
Implementation, test, and validation
Application code implementation
• Programming language (e.g., C/C++, Java, etc.)
• Agent platform (e.g., JADE, ZEUS)
Verification and optimisation (testing, debugging, performance analysis, code optimisation)
Tested and debugged application code
Deployment (distributed target hardware and software platforms)

Fig. 1. Stages of MAS application development, distinguishing simulation of functional prototypes from application code development, and validation addressed in this work.

Systems such as electric power and communication synchronizing simulator (EPOCHS) [12] and global event-driven co-simulation platform (GECO) [13] have been designed to improve simulation of distributed control prototypes, by combining two distinct simulation tools into a single coupled simulation—a continuous-time power system simulation on one hand, with the NS2[1] discrete-event communications network simulation on the other. However, they do not include a multiagent platform. In EPOCHS, agent functionality is represented using a specially written simulation component, whereas in GECO, it is represented within NS2 as parts of the network node models. Thus, neither system involves the actual application code.

In this paper, we focus on the implementation, test, and validation stages of the process shown in Fig. 1. We propose a platform combining a complete multiagent framework and a full communication network simulation package. In this way, a time-critical MAS application code may be fully developed and validated in preparation for deployment on a target system, without further manual modifications. Although our prime motivation was to address the needs arising in the development of applications for the smart grid, this work is also relevant for a wide range of other distributed control applications.

In line with other power systems research [1], [2], [4]–[7], [14], [15], we have adopted JADE framework [9], [16] as the multiagent platform and we combined it with an established communications network simulator (NS) OPNET modeler[2]. The role of the OPNET modeler is to provide an accurate simulation of the communications network, under various scenarios, as an integral part of the target environment. The JADE framework provides all the necessary the Foundation for Intelligent Physical Agents (FIPA)-compliant[3] middleware for implementation of the agent-based applications and it is supported by a wide range of operating systems running Java. While it is known that Java employs garbage collection mechanisms that may disrupt program execution at unpredictable times, this issue has been addressed by the real-time specification for Java [17], for which several implementations are available.

Several key aspects needed to be resolved to realize this concept and they make the key contributions of this paper. First, JADE offers a platform to run agents in real time, but it has no concept of simulation time; therefore, it needed to be extended to provide it with discrete-event capabilities. Second, we had to provide generic means of mapping application agents (running in JADE) into the OPNET simulation framework, involving extensions to both JADE framework and OPNET. Finally, synchronization and handover of execution between two platforms had to be provided. This was achieved via a runtime infrastructure (RTI) in accordance with the high-level architecture (HLA) standard [18], [19], which also required the provision of generic extensions of both the JADE framework and the OPNET modeler.

The adoption of the HLA standard is in common with the approach used in EPOCHS, GECO, and various other coupled simulation systems. However, this work focuses on the integration between JADE and OPNET and on the extensions required for each of these systems, in order to enable the JADE agent applications, in a coded form deployable on the target system, to run in combination with a discrete event simulation of the communications network. In contrast, the design of EPOCHS and GECO was focused on the issues of combining a continuous-time power system simulation (e.g., using PSCAD and PSLF) with the discrete event simulation of the communications network (e.g., using NS2). Cosimulation of power systems and communications has been extensively surveyed by Mets *et al.* [14]. The methods proposed in [12] may also be used in our system to include power system modeling, although this was not our objective.

The source code developed as part of this work is available to researchers from the corresponding author on request. JADE, OPNET, and RTI are also needed to replicate the platform, all freely available for research.

This paper is organized as follows. Section II describes the design, architecture, and implementation of the proposed system. Section III presents, as an example, the implementation and evaluation of an agent-based zone 3 remote backup relay supervision system [20], in which the performance analysis in relation to the choice of protocols and a specific communication infrastructure are presented in Section IV. Section V presents the conclusion.

II. Integration of Agent and Communication Simulation Platforms

A. Architecture Overview

According to the terminology adopted by the HLA standard [21], a system comprising two or more heterogeneous entities coupled with each other is called a *federation*. Fig. 2 illustrates

[1]*Network simulator 2*. [Online] Available: http://www.isi.edu/nsnam

[2]OPNET, Riverbed Technology. [Online] Availavble: http://www.riverbed.com

[3]FIPA. (2014). *Foundation for Intelligent Physical Agents*. a standards organization of the IEEE Computer Society. [Online] Available: http://www.fipa.org/

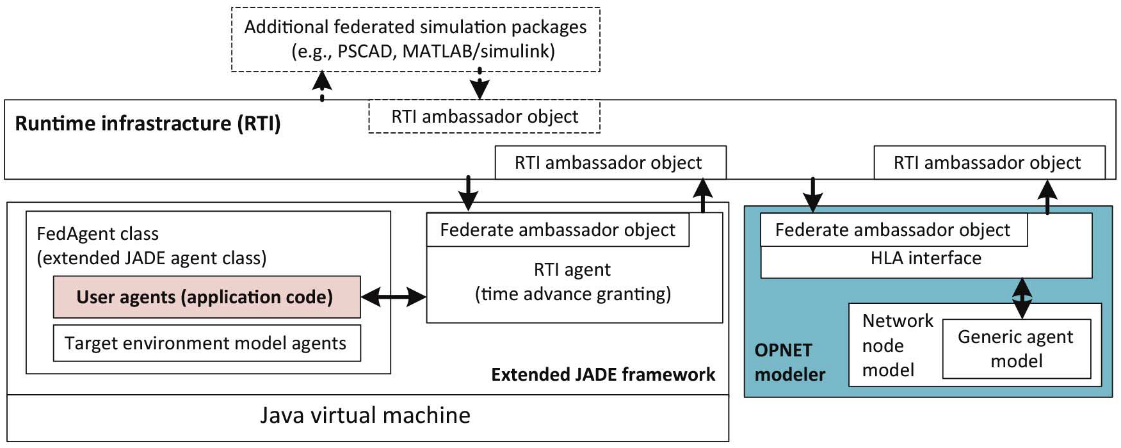

Additional federated simulation packages
(e.g., PSCAD, MATLAB/simulink)
Runtime infrastructure (RTI)
RTI ambassador object
RTI ambassador object
RTI ambassador object
FedAgent class
(extended JADE agent class)
User agents (application code)
Target environment model agents
Federate ambassador object
RTI agent
(time advance granting)
Extended JADE framework
Java virtual machine
Federate ambassador object
HLA interface
OPNET
modeler
Network
node
model
Generic agent
model

Fig. 2. Overall architecture combining the extended JADE framework, extended OPNET simulation, and the HLA-compliant runtime infrastructure. Additional simulations such as power flow or electrical transient simulations can be also added.

the overall architecture of the proposed system, which consists of the following three main components:

1) JADE;
2) OPNET modeler, communications NS;
3) RTI.

The distributed control application is implemented by the *user agents* running in JADE, each associated with a particular node in the simulated communication network (OPNET). The RTI provides common services to all federated systems for message passing, object management, and time management. Possible inclusion of other simulation tools is indicated in Fig. 2 by the dotted line blocks.

Both JADE and OPNET modeler are designed to be standalone tools and therefore, it is a considerable challenge to couple them. OPNET modeler is a discrete-event simulator and uses the concept of simulation time. A discrete-event simulation (DES) maintains a simulation clock and proceeds chronologically from one event to the next. Every event is tagged with a timestamp and kept on an event list until its execution. The event with the smallest timestamp is executed next and then deleted from the list. New events can be added by other events, but their timestamps have to be greater than the current simulation time, i.e., no past events can be scheduled. JADE, on the other hand, offers a platform to run agents as independent threads in real time, but there is no concept of simulation time and simulation events, making it difficult to synchronize its execution with the OPNET discrete-event simulator. In order to make a federation possible, extensions of both JADE and OPNET were developed and realized using strategies of reimplementation, extension of intermediate code, and usage of external APIs. With these extensions, this architecture allows for a synchronized execution of MASs and communication network models.

The original JADE framework was extended as follows (Fig. 2). First, the standard JADE *agent class* was extended to a new *FedAgent class* to allow time synchronization. Second, we have provided a new *RTI agent* as the means of both controlling the time advance and interfacing JADE with RTI. Finally, the mapping between *user agents* and the OPNET network nodes was realized using tables and managed using the *RTI agent*.

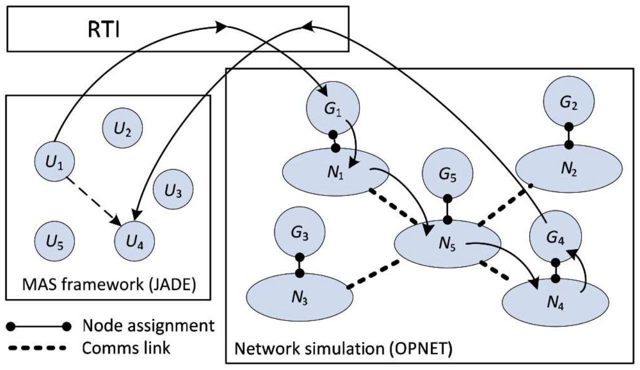


Fig. 3. Illustration of message passing in the federation. A message sent from *user agent* U_1–U_4 (dashed arrow) is relayed via RTI to the network simulation and *generic agents* G_1 and G_4, which are parts of the network node models N_1 and N_4, respecitvely.

The OPNET modeler was extended as follows. First, we have developed the *generic agent*, implemented as an extension of the standard OPNET network node. This provides a generic model of the *user agents* running in JADE. The *user agents* are mapped to their corresponding *generic agents*, the latter being used to transmit all messages between agents in JADE via the correct communication links in the simulated network. Second, integration with RTI was provided based on the HLA standard [18]. All messages passing between OPNET and RTI are performed by the *generic agents* via the HLA interface.

With these extensions in place, Fig. 3 illustrates a synchronized execution of a multiagent application and a communication network model. The *user agents* $U_n \, (n = 1, 2, 3, \ldots)$ implement the distributed application functionality and the nodes $N_n \, (n = 1, 2, 3, \ldots)$ represent the hardware devices on which the agents run. The *generic agent models* $G_n \, (n = 1, 2, 3, \ldots)$ are generic extensions of the node representation in OPNET and correspond to specific *user agents* U_n. The overall system runs as a discrete-event simulator, i.e., it proceeds chronologically from one event to the next.

In the illustration in Fig. 3, when the RTI grants time to JADE to run agents that are scheduled for a specific time, U_1 sends a message to U_4 (indicated by the dotted arrow between

U_1 and U_4). Instead of delivering the message directly, JADE sends an interaction to OPNET that initiates a data transfer in the NS. The message originating from U_1 is routed via RTI to the corresponding g*eneric agent* G_1, incorporated in the model of the network node N_1. It is then routed through the simulated communications network to the node N_4 and the *generic agent* G_4. From G_4, the message is then passed via RTI to JADE and to the corresponding *user agent* U_4.

It is also important to remember that the *user agents* implementing the distributed application, when deployed on the target system, will interact with physical local devices by collecting measurements and performing control actions. Traditional substation automation solutions involve hardwired analogue and binary connections of the substation controller to specific devices such as switchgear and meters, but an increasing trend is to employ merging units (MU) communicating to local devices via the process bus such as IEC61850-9-2 [22] over local Ethernet. Performance analysis of such a scheme was analyzed by Crossley *et al.* [23]. In the absence of the target system at development time, the local environment may be modeled using *target environment model agents,* also implemented within JADE (Fig. 2). These may be designed to communicate with the *user agents* according to relevant standard such as IEC61850 [22] and used to model the performance of local hardware devices and set up test scenarios. It is also possible to extend the federation to include other packages. Power system simulation tools (e.g., PSCAD, MATLAB/Simulink, PSLF, and others) can be added in a similar way as it was shown in EPOCHS [12].

The following sections describe the three components and their implementation in more detail.

B. Runtime Infrastructure

The RTI is a program that provides distributed simulation modeling services. Some of these services ensure that all simulation components start at the same time and advance in time in a synchronized manner. Other services offer ways to exchange information between the individual components through interaction, publish and subscribe mechanisms.

For the RTI, we have adopted the MÄK RTI 4.0.4 implementation (MÄK Technologies, Cambridge, MA, USA) which adheres to the HLA standard [18], [19]. The initial version of the HLA standard HLA 1.3 was published in 1998 and later became an IEEE standard method and is defined under IEEE 1516 [18]. While there are other distributed simulation modeling architectures, such as the Common Object Request Broker (CORBA) and the Java Remote Method Invocation (Java RMI), the HLA is better suited for simulations owing to its simulation-specific services (e.g., time management services) [24].

The HLA standard defines general principles, interface specifications, and an object model template (OMT). The interface specifications define the application programming interface (API) between the simulation components and the RTI. The OMT, on the other hand, defines the allowed structure of object models that are allowed to be exchanged between simulation components. Accordingly, each federated application communicates data to RTI via the standardized *RTIAmbassador object*, and receives data from RTI via the *FederateAmbassador object*.

C. MAS Framework—JADE

Although there are a great number of open-source MAS development toolkits available [25], the need for tools based on standards reduces the choice significantly. Adherence to standards ensures interoperability between MASs that were developed with different toolkits. In recent years, the Foundation for Intelligent Physical Agents' (FIPA) standards has been adopted by the majority of MAS developers.

We have adopted the FIPA-compliant JADE framework, which according to [4] has also become a firm favorite with researchers in power engineering. It is open-source and fully implemented in Java. The framework also provides a set of graphical tools that can be especially helpful in the debugging stages. Importantly, JADE is supported by a wide range of platforms, including embedded controllers and mobile devices, owing to its small memory footprint.

As mentioned above, we have devised a novel approach to control the execution of agents running in JADE to synchronize time advancements and message exchanges with the OPNET modeler and any other time-dependent simulation. This was implemented in the form of a DES extension of the JADE framework, enabling the agents to run as a part of a DES system.

The standard JADE framework provides the a*gent class*, on which it is based. Our DES extension of JADE provides the *FedAgent class*. It is derived from the *agent class* and therefore provides all APIs as the JADE framework, as well as additional APIs needed to control execution externally and redirect message passing between agents. On the other hand, the control of discrete time advance and message routing is performed by the additionally developed *RTI agent*. The *RTI agent* also provides an HLA compliant interface to the RTI for routing all messages and maintains the mapping table that associates specific user agents in JADE with their corresponding network nodes in OPNET (*generic agents*).

With this approach, a distributed application can be implemented as a set of *user agents* and verified using the federated development environment. The subsequent deployment on the target system involves only the *user agent*s running on the JADE platform.

Our DES extension of the JADE framework can also account for processing time at each node, which was an estimate for the purposes of this work, while, in practice, it would be accurately evaluated for the specific platforms employed.

D. Communication Network Simulation Framework—The OPNET Modeler

There are several communication NSs available, of which the NS2 and the OPNET modeler are by far the most widely used. NS2 is open-source software, which has made it popular in the research community. However, OPNET was chosen for this work because it provides an extensive model library and a broad suite of standard protocols and technologies. It also

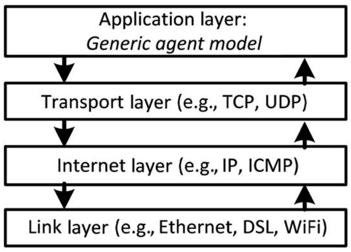

Application layer:
Generic agent model
Transport layer (e.g., TCP, UDP)
Internet layer (e.g., IP, ICMP)
Link layer (e.g., Ethernet, DSL, WiFi)

Fig. 4. *Generic agent model* as part of the extended OPNET node model, representing the application layer of the OSI model.

provides a graphical environment for model development, simulation, and data analysis, and enables rapid creation of network models.

In OPNET modeler, a communication network model is mainly made up of node models, which model devices that are connected via communication channels. Although there is no functionality to explicitly model MASs, it is possible to extend the network node models with user-developed code.

This facility was used to extend the OPNET node model to incorporate the generic agent model, so as to provide a representation of the corresponding user agents which exist in their coded form on the JADE platform. Within OPNET, this implements the application layer, which is connected to the transport layer of the open systems interconnection (OSI) model (ISO/IEC7498-1) as shown in Fig. 4.

The core scheduler of the OPNET modeler provides the option to be externally controlled via an HLA-compliant RTI. This basic module has been extended to handle notifications that are either received from or sent to the MAS platform.

E. Deployment

Our JADE extensions introduce simulation time and are API compatible with standard JADE. When the MAS application is deployed on a target system, the implemented agents only make use of the standard JADE framework.

It is well known that the automatic garbage collection in standard Java can introduce unpredictable delays at run time. For the proposed cosimulation platform, this is of little consequence, because it is a DES and includes a simulation clock under program control. Therefore, the simulation results are not influenced by Java's garbage collector or interrupts/delays caused by other processes running concurrently on the Linux OS.

For deployment, however, the real-time Java virtual machine (RTJVM) and a real-time operating system (such as real-time Linux) should be used, in order to avoid unpredictable time delays that might be introduced by the garbage collection. Java virtual machines adhering to the real-time specification for Java (RTSJ) [17], [26] are available from several providers [27].

III. Agent-Based Relay Supervision Case Study

Agent-based remote backup relay supervision is representative of the type of time-critical, distributed application involving MAS arising in smart grids. Garlapanti *et al.* [20] proposed

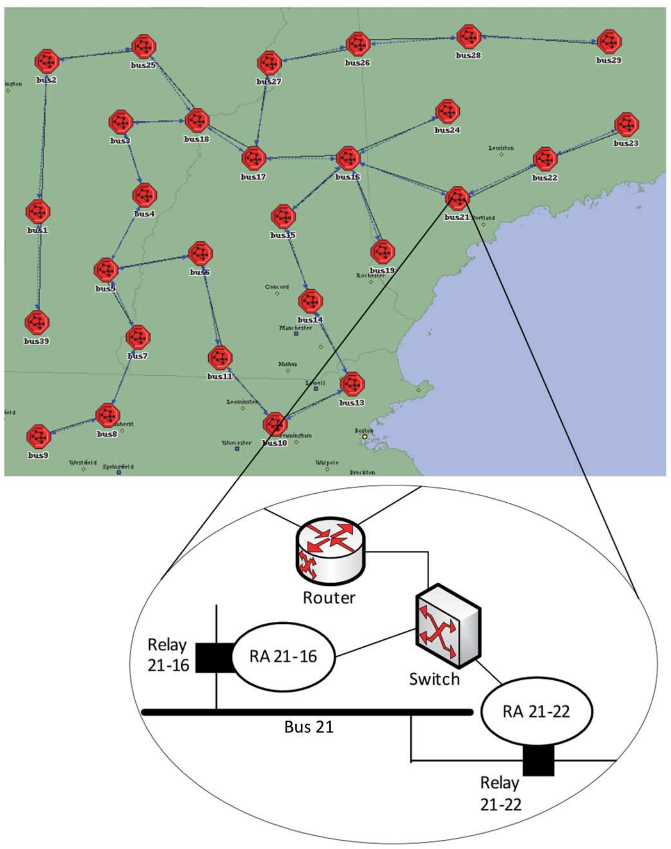


Fig. 5. Communication network model covering the area of New England. Every substation comprises several relay agents (RAs), network switches, and one router. Zoom-in shows a detailed view of substation with bus 21.

an agent-based method for supervision of zone 3 relays and considered its performance in relation to the IEEE 39-bus system. They performed power system simulation of the IEEE 39-bus model using PSLF, in order to study possible paths for cascading failures under the modeled load shedding and generator protection. The recorded event trace from this simulation was then used to define a subsequent NS2 simulation of the communications network to estimate the delays that would result in the communications network.

In this work, we have implemented the proposed agent-based supervision application using JADE. The application was tested in conjunction with the communications NS, where all message exchanges between agents were accurately modeled using OPNET. Different application architectures could be easily configured. The performance was evaluated using the described federated system to assess the performance of different MAS application configurations, protocols, QoS strategies, and network conditions. On this basis, the optimal communication strategy can be selected and validated in relation to the previously determined performance criteria. This is presented in more detail in the following sections.

A. Zone 3 Backup Relay Supervision Method

Fig. 5 shows the New England IEEE 39-bus system, with the communications network infrastructure according to that in

[20]. Protection relays are located at each end of the transmission lines connecting adjacent buses [28] and are all assumed to be directional impedance relays (Mho).

Zones of protection are conventionally defined in relation to the protection of power lines connecting adjacent buses and a relay located at one end of a power line [28]. Distance protection relays detect the impedance between the relay and a fault as the ratio of the magnitudes of the currents and voltages. Since the impedance varies almost linearly with distance, the location of the fault can also be determined. For a given relay, zone 1 protection typically corresponds to 85%–90% of the line length and operates instantaneously. To avoid blind spots in the vicinity of a bus, another zone of protection zone 2 is set at 120%–150% of the line length. A coordination delay of 0.3 s is allowed before zone 2 protection operates. A third protection zone 3 is set up as an additional backup, in case zones 1 and 2 protections fail, and covers 100% of the line on which the relay is situated and 120%–180% of the subsequent line. A coordination delay of 1 s is allowed for zone 3 operation.

Thus, when a zone 3 backup relay detects a fault, it waits, depending on the power network topology, in the order of 1 s for the corresponding zones 1 and 2 relays to react first and isolate the fault. If, after this delay, the fault is still detected, the zone 3 backup relay always trips its line. In the case of a real fault, this behavior is intended and welcome. However, hidden failures due to software or hardware errors could cause a zone 3 relay to trip and remove load unnecessarily.

The purpose of the agent-based supervision of relays is to help prevent unnecessary tripping of zone 3 remote backup relays due to hidden failures. Such failures may occur as a result of software or hardware errors in zone 3 relay. They are relatively rare, but may exist undetected for a long time and lead to an oversensitive zone 3 relay, which may erroneously trip and lead to a cascading failure.

The proposed supervision system [20] assumes that every protection relay can run a software agent, which can access and control the relay. We will call this agent an RA for the rest of this text. In some implementations, the RA may be running on the local substation automation controller as described in [23] and command the relay via a local field bus or substation automation network.

In the previously described situation, such an RA can gather status information from other RAs, while it is waiting to be able to better decide whether the fault is genuine. Because the information gathering of the RA has to be carried out within a very short time, this application is highly time-critical and its validation needs to accurately account for the communication delays.

B. MAS Configuration

Two different MAS were created in JADE and analyzed. Both systems offer the same overall functionality, but their RA communication approach is either based on a client–server (c/s) or peer-to-peer (p2p) behavior, as shown in Fig. 6. Both systems involve a DMA which is needed to maintain up-to-date information about the configuration of RAs and their assignments of zones 1, 2, and 3. Importantly, however, the corresponding

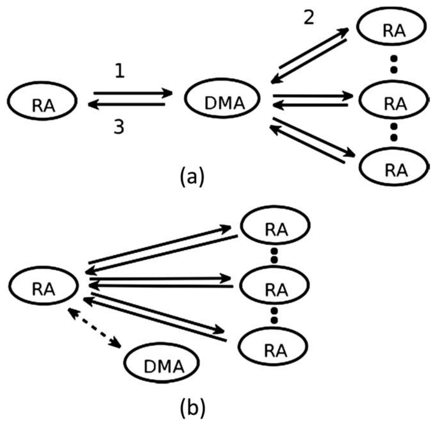


Fig. 6. (a) C/S and (b) p2p strategy of the RAs. Domain master agent (DMA) knows the topology of the bus system and relays for its domain. The numbers indicate the sequence of message exchanges.

decision-making authorities of agents and the messaging paths are very different between the two schemes. This may result in potentially large differences in communication delays.

1) C/S Architecture: Fig. 6(a) illustrates that the execution in this case is centralized around the DMA. When a zone 3 RA detects a fault, it immediately requests help from the DMA [message 1 in Fig. 6(a)]. The DMA has the necessary information to know which of the other RAs might pick up the same fault. It sends a request for information to all potential RAs in parallel and waits for all their replies [message 2 in Fig. 6(a)]. Upon receipt of all replies, DMA decides whether this is a genuine fault and forwards its decision to the initial RA [message 3 in Fig. 6(a)]. With this information, the zone 3 RA either trips the relay after the defined waiting time (i.e., 1 s) or not.

2) P2P Architecture: Fig. 6(b) illustrates the decentralized nature of this architecture. It differs from the c/s architecture in that the zone 3 RA contacts potential RAs directly without going through the DMA. The role of DMA is to send the list of peer agents to the zone 3 relay it should contact in case of a fault detection. This happens when the RAs are turned on and the list only needs to be updated when the topology changes or RAs are added or removed. Since DMA is not involved in time-critical communication, this method is expected to perform better than the c/s approach with respect to the message round-trip time.

The whole distributed MAS was made up of 81 RAs and one DMA. All communication between agents adheres to the FIPA-ACL standard and their message payload consisted of agent sender and receiver identifiers and the status of the relays. The agent message payload size was 250 bytes excluding headers for communication protocols such as TCP, IP, and Ethernet. We also created agents that sent 350 bytes of payload to study the impact of different message sizes. For the purposes of this work, the model assumed that the time taken for behaviors to run on the target hardware is 1 ms.

C. Network Model

Fig. 5 shows the modeled communication network connecting the substations of the IEEE 39-bus system across the area of New England. Round objects denote a local area network (LAN) of a substation, which consists of several RAs, network switches, and one router. The RAs are connected via Ethernet (IEEE 802.3) to the switches, which are connected to the router (see detailed view of substation with bus 21 in Fig. 4). The router is the gateway to other substations and, in this example, is connected through T1 communication links (1.544 Mb/s) to other substations.

We created three different network scenarios based on this communication network.

1) *No background traffic (noBT)*: This assumes that the T1 links are dedicated to the RA communication and not used to transfer any other traffic.

2) *Background traffic (BT)*: Agent communication has to compete with a 1.39-Mb/s BT on the T1 links. This represented a link utilization of about 90%.

3) *QoS*: A QoS strategy was implemented to help the agent communication compete against the 1.39-Mb/s BT. Class-based weighted fair queuing (CBWFQ) with a low-latency queue (LLQ) was enabled on the network interfaces of the routers. The LLQ handled all traffic that was marked as agent traffic, whereas the BT was subject to weighted fair queuing.

The two BT cases represent extremes of communications network loading. A realistic use case that is based on the QoS strategy would likely be somewhere between these two performance levels. All scenarios were simulated for the two most prevalent protocols, TCP and UDP, used for agent communication. Without any extra modeling work, this allowed for a comparison of the slower but reliable TCP protocol to the faster but less reliable UDP protocol.

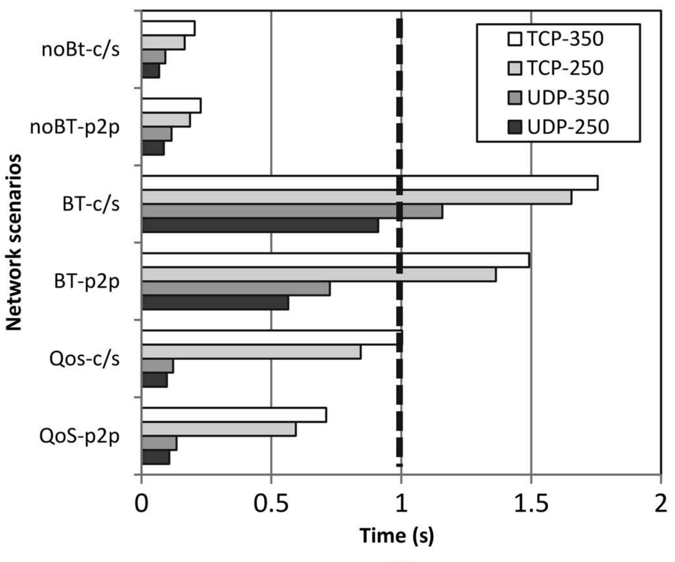
Thus, the overall distributed network model consisted of 28 LANs with 28 routers, 28 network switches, and a total of 82 generic agent models. The DMA was located at the substation of bus 18.

IV. Simulation Results

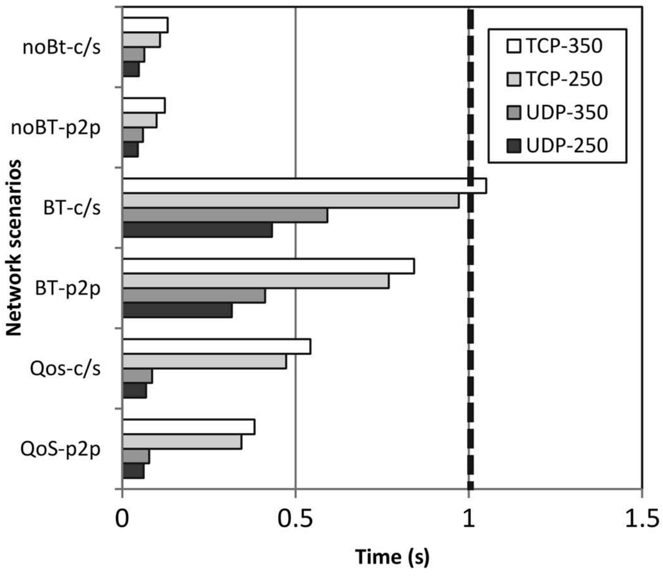
A. MAS Performance Under Different Scenarios

A total of 24 different simulation scenarios were run, involving different network scenarios (noBT, BT, and QoS), agent architectures (c/s and p2p), message sizes (250 and 350 bytes), and communication protocols (TCP and UDP). In every scenario, we consecutively simulated the detection of a fault for every single RA in the given system. Each scenario was simulated in this manner 120 times, in total about 2900 simulation runs. The times for RAs to reach a decision were recorded and subsequently analyzed.

Fig. 7 summarizes the results, showing the maximum and the average recorded decision times for all RAs. The results without BT (noBT-c/s and noBT-p2p) indicate the theoretical lower limits for the given link capacity, achievable only using a network dedicated for this application. All of the methods considered would meet the requirements under these conditions. In the presence of BT, however, the choice of technologies and communication strategies becomes critical. As expected, TCP protocol can be seen to result in significantly longer times than UDP, while the message payload size also has a noticeable



(a)



(b)

Fig. 7. (a) Maximum and (b) average times taken by all RAs to reach a decision. under different BT and MAS scenarios. The specified maximum decision time is 1 s.

effect. If TCP protocol is to be used, then QoS strategy must be employed, but the response time can be guaranteed only for 250 byte message payload.

B. Effect of Link Failure

Further simulations were run to analyze the effect of link failure between substations with buses 11 and 10. Fig. 8 shows the corresponding increase in decision time for different agents. Such information is important for network planning and system failure mode analysis.

C. Optimal Node Position for the DMA

In the above cases, DMA was allocated to run on the substation of bus 18, but it may be configured to run on any other node

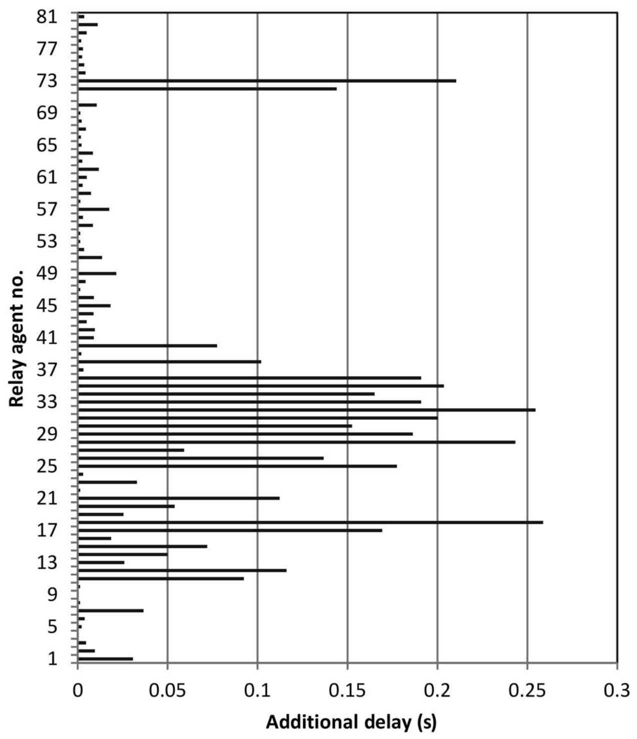


Fig. 8. Impact of a communication link failure between substations with buses 11 and 10. The times show the increase in decision time for each RAs compared to normal operation.

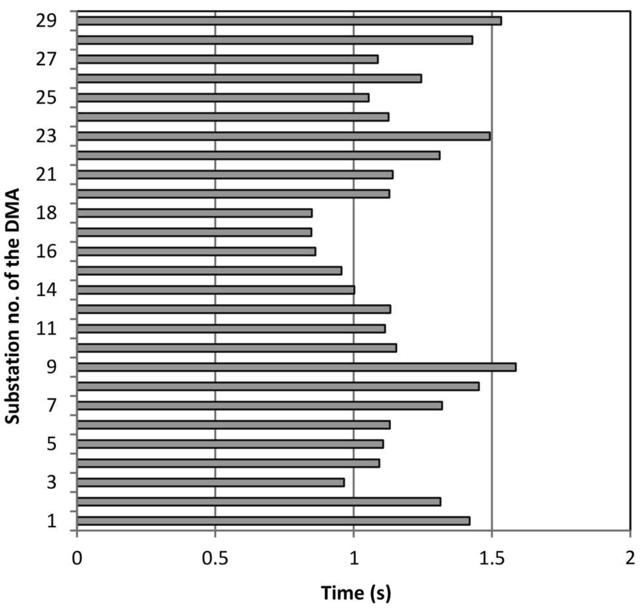


Fig. 9. Average times taken by the worst performing RAs to reach a decision in relation to the substation chosen to run the DMA. The scenario included the client/server approach, BT, QoS, TCP, and message size of 250 bytes.

of the network. Simulations were run for the scenario involving c/s, BT-QoS, TCP, and 250 bytes message payload, with the DMA deployed at different substations. A total of 50 simulations were run for each case. Fig. 9 gives the average times taken by the worst performing RAs to reach a decision, showing significant differences in performance and that the substations of buses 16, 17, and 18 are the best choices.

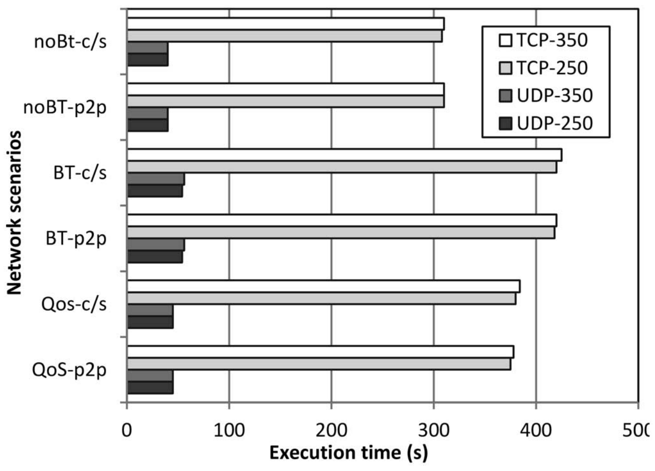


Fig. 10. Execution times for one simulation run involving different communication scenarios (Linux PC, Intel i7 processor, 8-Gb RAM).

D. Simulation Execution Time

It was observed that the simulation execution time was dominated by the time used by the OPNET modeler to simulate communication, while the agent code execution in JADE and message passing via RTI were very small in comparison. Fig. 10 shows the overall execution times for one simulation run involving different communication scenarios on a Linux PC (Intel i7-2700K processor, 3.9 GHz, 16 GB RAM). Simulations involving TCP took about 350 s, compared to only about 50 s for those involving UDP, as a result of the relative complexities of simulation.

V. Conclusion

Development of distributed time-critical applications for automation of the smart grid poses significant challenges in choosing optimal communication strategies and adherence to prescribed performance requirements. The main strength of the proposed system is that it enables the actual MAS application code to be tested on the basis of accurate simulation of the specific target communications network. Particularly important is the ability to optimize the implementation under potentially conflicting performance criteria such as QoS versus speed of response. The system combines JADE as a fully featured platform for the development and implementation of multiagent applications, with a powerful communications network simulation tool OPNET. The extensions necessary to integrate these tools were successfully developed and demonstrated on a realistic application involving protection of a wide area power network. It is envisaged that the smart grid of the future will need to employ a large number of applications of this type, as well as many that are less time-critical, which will share the underlying communication infrastructure. Accurate modeling of data communications and the use of the actual application code provide a high degree of confidence in the performance data obtained before deployment on the target system.

Acknowledgment

The authors would like to thank the Engineering and Physical Sciences Research Council for funding this research and the OPNET University Program for providing a free license for the use of their software.

References

[1] V. C. Gungor *et al.*, "Smart grid technologies: Communication technologies and standards," *IEEE Trans. Ind. Informat.*, vol. 7, no. 4, pp. 529–539, Nov. 2011.

[2] M. Ruofei, C. Hsiao-Hwa, H. Yu-Ren, and M. Weixiao, "Smart grid communication: Its challenges and opportunities," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 36–46, Mar. 2013.

[3] Q. Yang, J. A. Barria, and T. C. Green, "Communication infrastructures for distributed control of power distribution networks," *IEEE Trans. Ind. Informat.*, vol. 7, no. 2, pp. 316–327, May 2011.

[4] S. D. McArthur *et al.*, "Multi-agent systems for power engineering applications—Part I: Concepts, approaches, and technical challenges," *IEEE Trans. Power Syst.*, vol. 22, no. 4, pp. 1743–1752, Nov. 2007.

[5] S. D. McArthur *et al.*, "Multi-agent systems for power engineering applications—Part II: Technologies, standards, and tools for building multi-agent systems," *IEEE Trans. Power Syst.*, vol. 22, no. 4, pp. 1753–1759, Nov. 2007.

[6] D. V. Coury, J. S. Thorp, K. M. Hopkinson, and K. P. Birman, "An agent-based current differential relay for use with a utility intranet," *IEEE Trans. Power Del.*, vol. 17, no. 1, pp. 47–53, Jan. 2002.

[7] F. Ren, M. Zhang, D. Soetanto, and X. Su, "Conceptual design of a multi-agent system for interconnected power systems restoration," *IEEE Trans. Power Syst.*, vol. 27, no. 2, pp. 732–740, May 2012.

[8] H. S. Nwana, D. T. Ndumu, L. C. Lee, and J. C. Collis, "ZEUS: A toolkit and approach for building distributed multi-agent systems," in *Proc. 3rd Annu. Conf. Auton. Agents*, 1999, pp. 360–361.

[9] F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi, "JADE—A java agent development framework," in *Multi-Agent Programming*. New York, NY, USA: Springer, 2005, pp. 125–147.

[10] A. L. Dimeas and N. D. Hatziargyriou, "Operation of a multiagent system for microgrid control," *IEEE Trans. Power Syst.*, vol. 20, no. 3, pp. 1447–1455, Aug. 2005.

[11] H. Wang, "Multi-agent co-ordination for the secondary voltage control in power-system contingencies," *IEE Proc. Gener. Transmiss. Distrib.*, vol. 148, no. 1, pp. 61–66, Jan. 2001.

[12] K. Hopkinson *et al.*, "EPOCHS: A platform for agent-based electric power and communication simulation built from commercial off-the-shelf components," *IEEE Trans. Power Syst.*, vol. 21, no. 2, pp. 548–558, May 2006.

[13] H. Lin, S. S. Veda, S. S. Shukla, L. Mili, and J. Thorp, "GECO: Global event-driven co-simulation framework for interconnected power system and communication network," *IEEE Trans. Smart Grid*, vol. 3, no. 3, pp. 1444–1456, Sep. 2012.

[14] K. Mets, J. Ojea, and C. Develder, "Combining power and communication network simulation for cost-effective smart grid analysis," *IEEE Commun. Surv. Tuts.*, vol. 16, no. 3, pp. 1–26, Aug. 2014.

[15] R. J. Allan, *Survey of Agent Based Modelling and Simulation Tools*. Swindon, U.K.: Science and Technology Facilities Council, 2010.

[16] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems With JADE*, vol. 7. Hoboken, NJ, USA: Wiley, 2007.

[17] G. Bollella and J. Gosling, "The real-time specification for Java," *Computer*, vol. 33, pp. 47–54, 2000.

[18] *S. I. S. Committee of the IEEE Computer Society: IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)*. IEEE Standard 1516-2000, 1516.1-2000, 1516.2-2000, 2000.

[19] S. Straßburger, "On the HLA-based coupling of simulation tools," in *Proc. Eur. Simul. Multiconf.*, 1999, pp. 45–51.

[20] S. Garlapati, H. Lin, S. Sambamoorthy, S. K. Shukla, and J. Thorp, "Agent based supervision of zone 3 relays to prevent hidden failure based tripping," in *Proc. 1st IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, 2010, pp. 256–261.

[21] *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)–Federate Interface Specification*, IEEE Standard 15161-2010 (Revision of IEEE Standard 1516.1–2000), 2010, pp. 1–378.

[22] A. Apostolov and B. Vandiver, "IEC 61850 process bus—Principles, applications and benefits," in *Proc. 63rd Annu. Conf. Prot. Relay Eng.*, 2010, pp. 1–6.

[23] L. Yang, P. A. Crossley, A. Wen, R. Chatfield, and J. Wright, "Design and performance testing of a multivendor IEC61850–9-2 process bus based protection scheme," *IEEE Trans. Smart Grid*, vol. 5, no. 3, pp. 1159–1164, May 2014.

[24] A. Buss and L. Jackson, "Distributed simulation modeling: A comparison of HLA, CORBA, and RMI," in *Proc. 30th Conf. Winter Simul.*, 1998, pp. 819–826.

[25] E. Shakshuki and Y. Jun, "Multi-agent development toolkits: An evaluation," in *Innovations in Applied Artificial Intelligence*. New York, NY, USA: Springer, 2004, pp. 209–218.

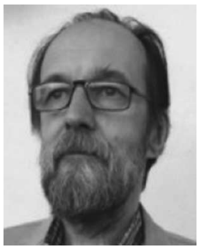
[26] S. G. Robertz and R. Henriksson, "Time-triggered garbage collection: Robust and adaptive real-time GC scheduling for embedded systems," in *Proc. ACM SIGPLAN Not.*, 2003, pp. 93–102.

[27] M. H. Dawson, "Challenges in implementing the real-time specification for Java (RTSJ) in a commercial real-time Java virtual machine," in *Proc. 11th IEEE Int. Symp. Object Oriented Real Time Distrib. Comput. (ISORC)*, 2008, pp. 241–247.

[28] P. M. Anderson, *Power System Protection*. Piscataway, NJ, USA: IEEE Press, 1999.

Fidelis Perkonigg received the B.S. and M.S. degrees in computer science from the Graz University of Technology, Graz, Austria, in 2006, and the Ph.D. degree in mechanical engineering from Imperial College London, London, U.K., in 2015.

He worked with Cisco Systems, Vienna, Austria, before his doctoral studies, and he is a Certified Cisco Routing, Switching, Security, and Voice Professional [Cisco Certified Network Professional (CCNP) level]. He is currently a Teaching Fellow with the Department of Computing, Imperial College London. His research interests include computer networks, operating systems, and software engineering.



Djordje Brujic (M'95) received the Dipl. Ing. degree in electrical engineering from the University of Belgrade, Belgrade, Serbia, in 1975; the M.Sc. degree in control systems from the Cranfield University of Technology, Bedford, U.K., in 1985; and the Ph.D. degree in geometric modelling from Imperial College London, London, U.K., in 2002.

He is a Senior Researcher at Imperial College London, whose research interests span real-time control systems, three-dimensional surface reconstruction, and more recently, distributed systems and multi agent methodologies applied to smart grid problems. He has published over 50 papers in these domains.

Mihailo Ristic (M'94) received the degree in mechanical engineering from University College London, London, U.K., in 1981, and the M.Sc. degree in control systems and the Ph.D. degree in robotics from Imperial College London, London, in 1982 and 1986, respectively.

He is currently a Senior Lecturer in the Mechanical Engineering Department with Imperial College London. His research interests include a wide range of topics in control systems, computer aided design, electrical machines, as well as magnetic resonance imaging. He is a Co-Founder of Turbo Power Systems, Gateshead, U.K., which specializes in high-speed electric machines and power electronics.

Dr. Ristic is a Chartered Engineer and a Fellow of the Institution of Mechanical Engineers.