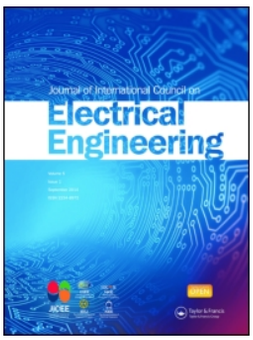

Taylor & Francis
Taylor & Francis Group

Journal of International Council on Electrical Engineering

ISSN: (Print) 2234-8972 (Online) Journal homepage: http://www.tandfonline.com/loi/tjee20

Multi-agent systems and their applications

Jing Xie & Chen-Ching Liu

To cite this article: Jing Xie & Chen-Ching Liu (2017) Multi-agent systems and their applications, Journal of International Council on Electrical Engineering, 7:1, 188-197, DOI: 10.1080/22348972.2017.1348890

To link to this article: https://doi.org/10.1080/22348972.2017.1348890

Published online: 14 Jul 2017.

JOURNAL OF INTERNATIONAL COUNCIL ON ELECTRICAL ENGINEERING, 2017
VOL. 7, NO. 1, 188–197
https://doi.org/10.1080/22348972.2017.1348890

Taylor & Francis
Taylor & Francis Group

Multi-agent systems and their applications

Jing Xie[a] and Chen-Ching Liu[a,b§]

[a]School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA, USA; [b]School of Mechanical and Materials Engineering, University College Dublin, Dublin, Ireland

ABSTRACT

The number of distributed energy components and devices continues to increase globally. As a result, distributed control schemes are desirable for managing and utilizing these devices, together with the large amount of data. In recent years, agent-based technology becomes a powerful tool for engineering applications. As a computational paradigm, multi-agent systems (MASs) provide a good solution for distributed control. In this paper, MASs and applications are discussed. A state-of-the-art literature survey is conducted on the system architecture, consensus algorithm, and multi-agent platform, framework, and simulator. In addition, a distributed under-frequency load shedding scheme is proposed using the MAS. Simulation results for a case study are presented. The future of MASs is discussed in the conclusion.

ARTICLE HISTORY
Received 14 January 2017
Accepted 26 June 2017

KEYWORDS
Multi-agent system (MAS); intelligent agent; multi-agent platform; distributed control; system architecture; consensus algorithm; smart grid; under-frequency load shedding

1. Introduction

1.1. Motivation of distributed control

The energy system is evolving rapidly. Indeed, from smart buildings to smart grids, digital technologies are producing numerous data streams that offer in-depth information about the system. For example, the number of distributed intelligent electronic devices (IEDs) and distributed energy resources (DERs) continue to increase globally. The total number of installed phasor measurement units (PMUs) has increased to more than 2000 in North America [1]. In addition, many more remote control switches have been installed in distribution systems to enhance the resilience of distribution feeders. Furthermore, development of renewable energy contributes to the increase in DERs. By 2017, the total capacity of utility-scale solar panels operating in the U.S. is expected to be over 20 GW [2]. Decentralized and distributed control schemes are needed for managing and utilizing these widely distributed devices.

Two reasons behind the increasing number of devices are the growing investment driven by policies and lower hardware prices. Moore's law has been valid for a long period of time. Hardware is becoming cheaper and better in performance. This provides the foundation for distributed control. Considering the increasing scale of the number of devices in a system, distributed methods are needed to shift the computational burden to local controllers.

At the same time, integrating renewable energy generation, energy storage, and electric vehicles brings new challenges [3,4]. A critical problem comes from the decentralized ownership of energy system components. For example, in Pullman's distribution system (Washington, U.S.A.), a 1 MW flow battery is installed by Avista and a 72 kW solar panel is established on the campus of Washington State University (WSU). While the electric energy trading and regulatory environment is evolving, decentralized ownership of energy system components becomes a problem for centralized control methods with participation from a large number of consumers. Another difficulty arises from the random nature of these components. For instance, the energy demand, arrival time, and departure time of electric vehicles (EVs) are random [5,6]. Centralized methods are complex and impractical for modeling and managing these stochastically behaved EVs.

1.2. Why multi-agent systems?

Modeling and computation tasks are becoming much more complex as the size continues to increase. As a result, it is laborious and difficult to handle using centralized methods. Although motivations to apply multi-agent

CONTACT Jing Xie jing.xie6@wsu.edu
[§]Dr. Chen-Ching Liu is a Visiting Professor at the School of Mechanical and Materials Engineering, University College Dublin, Dublin, Ireland

systems (MASs) for researchers from various disciplines are different, as indicated in [7], the major advantages of using multi-agent technologies include: (1) individuals take into account the application-specific nature and environment; (2) local interactions between individuals can be modeled and investigated; and (3) difficulties in modeling and computation are organized as sublayers and/or components. Therefore, MASs provide a good solution to distributed control as a computational paradigm. In addition, artificial intelligence (AI) techniques can be utilized. In [8], an agent is defined as a computer system that is situated in an environment that is capable of autonomous actions in this environment to meet its design objectives. In [9], a MAS is defined as a system that comprises two or more agents, which cooperate with each other while achieving local goals.

1.3. Applications of MASs

MASs have been applied to various problems, including market simulation, monitoring, system diagnosis, and remedial actions [10–12]. In [13], a substation physical security monitoring (SPSM) system is proposed within the framework of strategic power infrastructure defense (SPID) [14,15]. It monitors from remote the physical security of power substations. In [16], an approach is developed to prevent interconnected power systems from catastrophic failures. It uses a MAS-based defense system that allows agents to have adaptive decision criteria. In [17], an integrative and flexible method is proposed that uses agent-based modeling for assessment of market designs. Agents are facilitated by Q-learning. Compared with the competitive benchmark, they can exploit market flaws to make higher profits. For remedial actions, a distributed under-frequency load shedding (UFLS) scheme is developed in [18] using the MAS. It is improved and being implemented as a demonstration for the RIAPS platform [19]. Details of the UFLS scheme are presented in Section 4.

Progress in AI, hardware, and sensor technologies have been achieved by the MAS community, resulting in agent technologies that are applied successfully to real-world industrial problems. A project [20] was supported by U.S. Department of Energy (DOE) to transfer the VOLTTRON™ software platform to Tranformative Wave. Details of VOLTTRON™ developed by Pacific Northwest National Lab (PNNL) are discussed in Section 3.2. In addition, it provides Tranformative Wave with technical support to develop products and services that improve the operating efficiency of buildings and resilience of power grids. This project indicates that industry is adopting agent technologies.

The remaining of this paper is organized as follows. A literature survey is presented in Sections 2 and 3, including research on the system architecture, consensus algorithm, and multi-agent platform, framework, and simulator. A MAS-based UFLS scheme is discussed in Section 4 with a study case. The future of MASs is discussed in the conclusion.

2. System design – system architecture, agent type, and consensus algorithm

System design is critical as it covers many aspects in the development of MASs, e.g. agent models, coordination, data collection, and interaction among agents. In this section, a literature survey of system architecture, agent type, and consensus algorithm is presented.

2.1. System architecture and agent type

There are abstract and concrete architectures [8]. Components and the basic engine structure are defined in the abstract architecture, which needs to be as generic as possible. Starting from an abstract architecture, the concrete architecture is developed by assigning a type to each component and implementing each macro instruction of the engine.

Intelligent agents are classified into several types with respect to their functionalities and decision-making mechanisms. (1) Purely reactive agents make decisions using only the present information without referring to historical data. Thus, they utilize direct mapping from situation to action and respond to the environment directly. For example, in [21], reactive agent techniques are utilized to build up car-following models based on artificial neural networks. (2) Logic-based agents make decisions through logical deduction. In [22], a method is developed for handling multiple hypotheses and performing logic-based fault diagnosis. Scenarios from the Italian power system are used for evaluation. (3) Belief-desire-intention (BDI) agents are built using symbolic representations of the intentions, beliefs, and desires of agents. In [23], the stages of autonomy determination for software agents are discussed. The recognition of potential autonomy is provided utilizing the BDI paradigm. (4) Layered architectures incorporate several software layers. Each layer combines agents that deal with different abstract levels of the environment. For example, in [14,15], a SPID system is designed in the layered architecture for integrating various types of protection systems and defense schemes in different levels.

2.2. Consensus algorithm

The cooperation of agents is achieved through information interaction to reach a consensus. The performance and

functionalities highly rely on the communication layer, especially the connection topology and associated protocols. The consensus and interactive consistency problems are two important aspects of the Byzantine agreement problem [24]. The terms of the above three problems are used interchangeably in the literature, although they have different formal definitions. In this study, the consensus problem is addressed. The well-known and widely used Paxos and average-consensus algorithms are described in this subsection. In addition, the fault tolerance is discussed.

2.2.1. Paxos algorithm

The Paxos algorithm was originally proposed by L. Lamport in 1988 [25]. In 2001, it was published in [26] with a user-friendly presentation. As one of the simplest distributed algorithms, the Paxos algorithm is designed to implement a fault-tolerant distributed system. The 'synod' consensus algorithm [25] serves as the heart of the Paxos algorithm. Each Paxos agent can be a proposer, acceptor, or learner. If most nodes are available, this algorithm guarantees that agents will converge to one value. It has been used by many software products, e.g. (1) the BigTable which is adopted by many products of Google; and (2) the search engine, Bing, of Microsoft.

2.2.2. Average-consensus algorithm

The average-consensus algorithm has been applied to many fields for the consensus and cooperation of networked MASs. Its convergence and convergent speed are reported in [27] and [28]. As an adaptive distributed algorithm, it requires only communication among neighboring agents. Therefore, the communication burden is low. All available agents will reach an agreement, which is equal to the average of their initial values. The simple mathematic model is summarized. A graph, $N = (V, B)$, represents the network of n agents. The set of agents is denoted by $V = \{1, 2, \ldots, n\}$. A nonnegative $n \times n$ adjacency matrix $B = [b_{ij}]$specifies the interconnection topology of network N. If an active communication link exists from agent i to agent j, b_{ij} is a positive value. Otherwise, $b_{ij} = 0$.

$$\dot{y}_i(t) = \sum_{j=1}^{n} b_{ij}\left(y_j(t) - y_i(t)\right) \tag{1}$$

$$Y(t) = \mathrm{col}\left(y_1(t), y_2(t), \ldots, y_n(t)\right) \tag{2}$$

$$\Delta = \mathrm{diag}\left(\sum_{j=1}^{n} b_{1j}, \ldots, \sum_{j=1}^{n} b_{ij}, \ldots, \sum_{j=1}^{n} b_{nj}\right) \tag{3}$$

$$L = -B + \Delta \tag{4}$$

$$\dot{Y}(t) = (-\Delta + B)Y(t) = -LY(t) \tag{5}$$

where Δ and L are the degree and Laplacian matrices, respectively.

2.2.3. Fault tolerance

Fault tolerance is focused on both continuous availability and the elimination of recovery time. As an important performance metric of fault tolerance, downtime refers to periods of time in which a system is not operational [29]. In order to minimize the downtime, specialized software routines are needed to detect failures of hardware (e.g. sensors, actuators, storage devices, and communication channels) and switch to backup devices automatically. This type of self-checking logic should be provided by the operating system (OS) and governed by the software platform designed for distribution applications. In addition, the capability to remove, disconnect, and repair the problematic devices without disruption to the computer system is critical. Furthermore, important data sources should be checked and calibrated periodically, e.g. PMUs.

With respect to the requirements on cyber security, firewalls and access control should be provided by the OS to prevent unauthorized users and access. Encryption is needed to secure the communication among nodes. The basic mechanisms of encryption should be provided by the OS. The security level can be enhanced further with digital certificates if it is supported by the platform. Once the cyber intrusions are detected and traced promptly and accurately by the OS and platform, mitigation actions are applied to block unauthorized access. In most applications, it is assumed that non-Byzantine fault is considered. Therefore, no cheating agent(s) is (are) considered in the multi-agent based applications. Both fault tolerance policies and mechanisms become simpler.

In general, the OS and platform should support the recovery of nodes due to software or other defects (e.g. power failure). The recovery includes two parts: (1) automatically restarting the node; and (2) recovering the node to a well-known state. It is expected that multiple states will be recorded and available for recovery. As a result, applications will be able to achieve fault tolerance and decide which state should be selected for recovery. In addition, self-checking should be performed immediately once a node restarts, including hardware, software, and interfaces. If functionalities of the restored node cannot be guaranteed, the node may be required by applications to remain silent instead of participating in the coordination. Therefore, self-checking is critical and its reliability matters.

Fault tolerance becomes more complicated if communication channels are unreliable. The outage of communication lines and noisy measurements should be taken into account. Therefore, it is harder to reach a consensus if the underlying communication channels are unreliable. For example, the balanced digraph may become unbalanced

due to the loss of edge(s), i.e. communication channel(s). Note that problematic communication links usually cannot be recovered soon.

3. Multi-agent platforms, frameworks, and simulators

Much effort has been made in designing the system architecture, agent model, and interaction between agents. Nevertheless, the real implementation of MASs remains in an early stage. Since late 90s, a large number of multi-agent platforms, frameworks, and simulators have been developed targeting the gap between concept design and implementation. As a result, it is a critical issue for developers to select the tool properly. In this section, a comparison study is performed and tabulated. In particular, VOLTTRON™ that is targeted for energy system applications is discussed.

3.1. Comparison

In order to perform a meaningful comparison, the evaluation criteria must be selected systematically. In [30], four platforms are selected and compared using four criteria: completeness, applicability, complexity, and reusability. The process of developing MASs is divided into four stages. Besides analysis, design, and development stages of traditional software construction, deployment is added as an important stage due to the complexity of MASs. In [31], a comparative review of 24 multi-agent platforms, frameworks, and simulators is reported using 28 universal criteria. It is focused on all available platforms despite the fact that the development of some platforms has been terminated.

Typical multi-agent platforms, frameworks, and simulators developed by different academic and industrial groups (e.g. universities, research institutes, and companies) are compared. The up-to-date results are shown in Table 1. A notable feature of most platforms is the open source and non-proprietary nature that allows free distribution and development. In addition, the compatibility with OSs matters. For example, JADE benefits greatly from its pure design using Java™, which is cross-platform with respect to the associated Java virtual machine (JVM). Applications developed using JADE can run on multiple operation systems without modification. Therefore, it becomes one of the most widely used platforms for research purpose. Furthermore, the supported programming languages are likely to influence the choices of developers. Programmers usually intend to use tools that support programming languages with which they are familiar and are easy to use, regardless of characteristics of the problem and model. For instance, agent-oriented programming languages are not a preferred option for new developers due to its lower level of abstraction.

3.2. VOLTTRON™

Almost all platforms are developed to target a wide spectrum of applications in different fields. Nevertheless, there is a strong desire for specialized platforms that focus on particular fields. To this end, a flexible and modular software platform, VOLTTRON™ [36], is developed and specialized for applications of energy systems, e.g. integration of DERs, decentralized control of microgrids, and distributed automation systems. The predecessor and prototype of VOLTTRON™ [40] was initially presented in 2011 as a framework that brings intelligence to the actuators and sensors in a smart grid. A year later, it was developed from a framework to a platform called VOLTTRON™ [35]. A demonstration is shown in [41]. It is further applied to the integration of electric vehicles and smart grids [42]. Applications can be developed on top of VOLTTRON™ to improve energy efficiency, support grid services, monitor the performance of building systems, etc.

The VOLTTRON™ platform is developed in Python utilizing many open source libraries. In addition, a set of utilities has been created to speed up development in Python-based agents. However, applications are not tied

Table 1. Comparison of selected multi-agent platforms, frameworks, and simulators.

Software	Type	Developer	Programming language	Operation system(s)	Open source
JADE™ [32]	Framework	Telecom Italia	Java™	All that supports JVM	✓
Agent Factory [33]	Framework	University College Dublin	Agent Factory Agent Programming Language (AFAPL)	All that supports JVM	✓
OpenFMB™ [34]	Framework	Duke Energy	Java™	Linux	✓
RIAPS [19]	Platform	Vanderbilt University	Python (and others in future)	Linux	✓
VOLTTRON™ [35], [36]	Platform	Pacific Northwest National Laboratory (PNNL)	All	Linux	✓
Janus [37]	Platform	Laboratoire Systèmes et Transports (SET) and Grupo de Investigación en Tecnologías Informáticas Avanzadas (GITIA)	SARL, Java™	Linux, Unix, Windows, Mac OS, Android	✓
MASON [38]	Simulator	George Mason University	Java™	Windows	✓
GAMA [39]	Simulator	IRD/UPMC International Research Unit UMMISCO	GAML	Windows, Linux, Mac OS	✓

to a specific programming language, such as Python or Java™. This lives up to developers' expectation of flexibility. Agents written in other languages are supported once they can communicate over the message bus. The common VOLTTRON™ base can simplify and accelerate the development of applications. It includes 'base' classes and 'helper' utilities that support rapid development and deployment of applications in a scalable and cost-effective way. The goal is to allow researchers to focus on their algorithms rather than interaction with the platform.

As an open source and open architecture platform, VOLTTRON™ supports the peer-to-peer network of distributed agents that bring computation closer to data sources cooperatively. It supports non-Intel central processing units (CPUs). In addition, the requirements on CPU, memory, and storage footprint are low. Therefore, applications can be built to manage energy use more efficiently among appliances and devices.

VOLTTRON™ is widely used for implementation of applications. Besides, it is employed as the base of other platforms. The BEMOSS™ system [43,44] developed by Virginia Tech serves as a good example. It adds auto-discovery capability for several devices together with a user interface for controlling those devices. All features are built on the core of VOLTTRON™. Another example is the Transactional Network Platform (TNP) [45], which consists of VOLTTRON™ as agent execution software. It includes agents that perform critical services and specific functions (e.g. demand response, fault detection, weather service, etc.). TNP is proposed to achieve operational, financial, and energy transactions between connected individuals.

4. Distributed UFLS scheme using MAS

In [46], it is indicated that the wide-area special protection system (WASPS) is critical for dynamic stability and frequency problems. As an important component of the WASPS, distributed UFLS is a natural problem for the application of agent technologies. In this section, an adaptive UFLS algorithm is described. The purpose of this UFLS scheme is to mitigate power system blackouts using the MAS. It was originally proposed in [18]. In this paper, the distributing step has been extended and a case study is performed. It is being implemented as an application of the RIAPS ('Resilient Information Architecture Platform for the Smart Grid') platform. In the future, the proposed UFLS method will be improved to deal with wind turbines, solar panels, and batteries, targeting the increasing penetration and impact of DERs integration on UFLS schemes [27].

4.1. Methodology

The system framework is shown in Figure 1. Buses and transmission lines are depicted in the power system layer using thick bars and thin lines, respectively. The multi-agent based UFLS scheme has two types of agents: generation agent (GA) and substation agent (SA). If an agent has a generation unit, it belongs to the GA type. Otherwise, it is a SA. The agent framework is shown in Figure 2. Each RIAPS node includes a number of agents for each application. Each agent has components for typical functionalities. The flowchart showing monitoring, estimation, and distributing steps is given in Figure 3. In the monitoring step, frequency variation is monitored by agents through two triggers. If abnormal conditions are detected, agents start to communicate with adjacent neighbors. Based on the shared information, the amount of load shedding is estimated and distributed.

4.2. Extended distributing step

It is known that load buses that are closer to generators are more effective in load shedding. Therefore, they have higher priorities to shed the load. In the distributing step, each agent decides how much load should be shed by itself.

$$\Delta P_{A_k} = \sum_{i=1}^{N_k} \Delta P_{LS_i^k} \tag{6}$$

$$\Delta P_{\mathrm{LS}_i^k} = \begin{cases} \Delta P_i^k, & \text{if } \Delta P_i^k > 0 \text{ and } \Delta P_i^k < P_{L_i^k} \\ P_{L_i^k}, & \text{if } \Delta P_i^k > 0 \text{ and } \Delta P_i^k \geq P_{L_i^k} \\ 0, & \text{if } \Delta P_i^k \leq 0 \end{cases} \tag{7}$$

$$\Delta P_i^k = \Delta P - \sum_m P_{L_m^k} \tag{8}$$

$$m \in I_i = \left\{ u | u \in B \text{ and } a_u < a_i \right\} \tag{9}$$

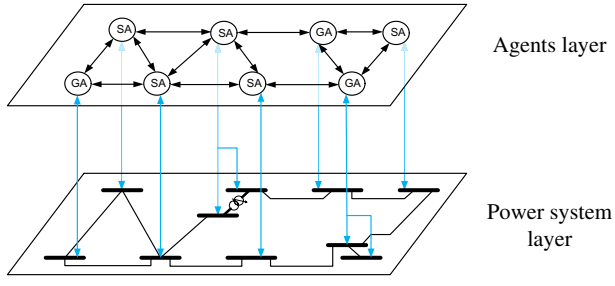

Figure 1. System framework.

$$a_i = \min\{\mathrm{LG}_i\} \tag{10}$$

$$\mathrm{LG}_i = \left\{ \frac{\left|\theta_i - \theta_g\right|}{S_g} \mid g \in G \right\} \tag{11}$$

where N_k denotes the number of load buses of the kth agent. $P_{L_i^k}$ is the amount of active power load available to be shed at load bus i. $\Delta P_{\mathrm{LS}_i^k}$ is the active power load to be shed by the kth agent at load bus i. ΔP is the total active power deficiency, i.e., the total amount of load to be shed by all agents. It is calculated by agents in the estimation

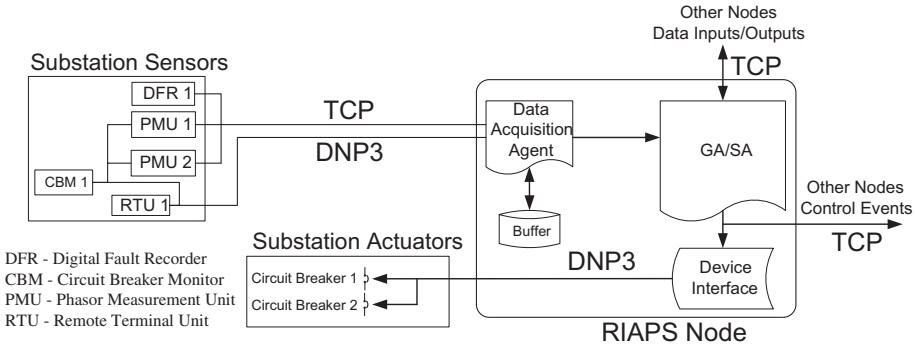

Figure 2. Agent framework.

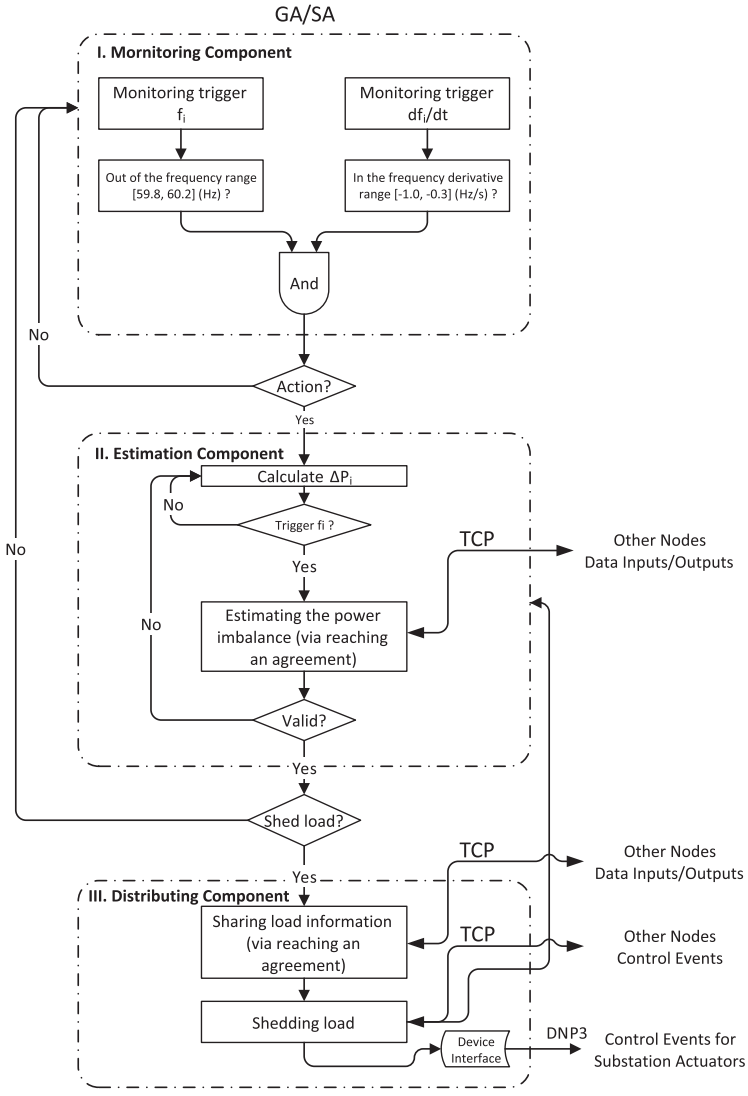

Figure 3. Flowchart of GA/SA.

step. All load bus indices of agents are comprised in a finite index set B. The symbols θ_i and θ_g are voltage angle of the load bus i and generator g, respectively. S_g is the nominal apparent power of the generator g. G is the finite index set that comprises all generator indices.

It is assumed that the amount of reactive power shed, $\Delta Q_{\mathrm{LS}_i^k}$ follows the same percentage of the active power load to be shed.

$$\Delta Q_{\mathrm{LS}_i^k} = \Delta P_{\mathrm{LS}_i^k} \frac{Q_{\mathrm{L}_i^k}}{P_{\mathrm{L}_i^k}} \tag{12}$$

Where $Q_{\mathrm{LS}_i^k}$ is the amount of reactive power load available to be shed at load bus i of the kth agent.

Each agent determines the timing and sequence of load buses by the priorities. The timing is determined by

$$t^i_{A_k} = t_{\mathrm{fm}} + t_{\mathrm{op}} + \frac{T_{\mathrm{all}}}{P_{\mathrm{all}}} \cdot \sum_m P_{L^k_m} \tag{13}$$

$$t^i_{A_k} - t^j_{A_k} = \frac{T_{\mathrm{all}}}{P_{\mathrm{all}}} \cdot \Delta P^k_{\mathrm{LS}_j} \tag{14}$$

where $t^i_{A_k}$ denotes the timing to shed load at bus i by the kth agent. t_{fm} and t_{op} are the delays of monitoring frequency and opening circuit breakers, respectively. T_{all} indicates the time required to shed all load in the system. In [48], it specifies 0.9 s to shed 451 MW load in the IEEE 39-bus system. Therefore, T_{all} is set to be 12.17 s. The total load of the test system is denoted as P_{all}. The priority of load bus i of agent k is one level lower than the load bus j of agent l.

4.3. Case study

The IEEE 39-bus system is used and the contingency scenario is shown in Figure 4. Transmission lines are tripped at 0.5, 1.5, and 2.2 s. An advanced load shedding strategy developed in [18] is adopted for the performance comparison. An amount of 607.73 MW active power imbalance is estimated at 2.54 s. Load shedding actions are presented in Table 2. If there is no load shedding scheme, a system collapse takes place, leading to the loss of all loads in the North Island.

The results of comparison are shown in Table 3. The proposed method sheds 35.62 MW and 21.93 MVAR less load but stops the frequency decay earlier. See Figures 5 and 6. The frequency does not return to 60 Hz because the secondary control is not included in the simulation.

In the simulation, all loads are assumed to be of the same category. However, in a real power grid, there are several categories of loads and a tradeoff is needed. One simple option for tradeoff is to shed loads closer to generators within the same priority category. In the third step of the UFLS scheme, through communications, all agents share the information of loads that can be shed.

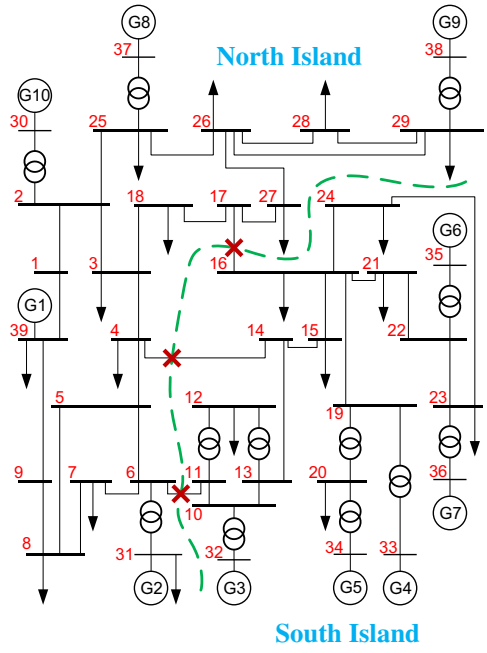

Figure 4. IEEE 39-bus system and contingency scenario.

Table 2. Load shedding actions.

Agent	Load bus	Reactive power load (MVAR)	Active power load (MW)	Percentage	Time (s)
PA5	39	135.54	598.53	54.215%	2.74
PA2	31	4.6	9.2	100%	2.72

Table 3. Comparison results.

Methods	Multi-agent based distributed	Advanced
Reactive power load (MVAR)	140.14	162.07
Active power load (MW)	607.73	643.35
Frequency decay ends (s)	3.97	8
Frequency (Hz)	59.83	59.8

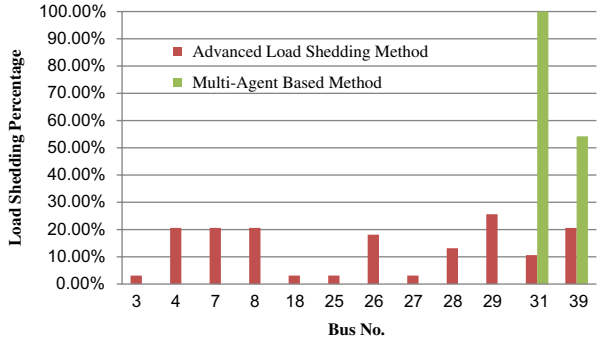

Figure 5. Distributed load shedding.

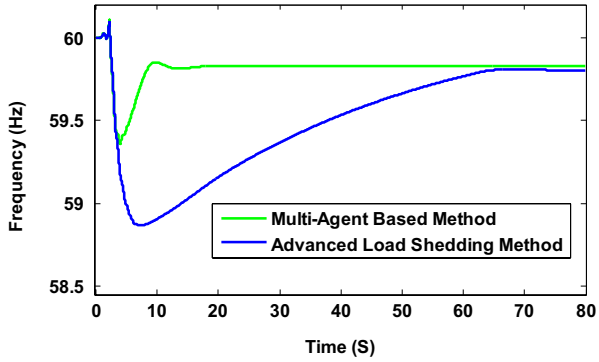

Figure 6. System responses.

In addition, the information of loads includes the type and category. Thus, agents will be able to determine the category of loads which should be involved to reach the total amount of load shedding. The load at a lower priority category will be shed first. Among loads in the same category, the load closer to the generator will be shed first.

4.4. From design to evaluation and implementation

Once the MAS is selected and applied to a problem, the first step starts from system design. Based on the properties and characteristics of this problem, a suitable agent architecture can be determined together with the consensus protocol. In this step, flexibility is a major concern with respect to future system extension and its integration with other systems. In addition, major restrictions on the consensus protocol arise from communication channels, network topology, and requirements on performance. For instance, the UFLS scheme adopts the layered architecture and average-consensus algorithm. The rationale is: (1) The cyber-physical model of modern power systems is layered; (2) As a remedial action scheme (RAS), the MAS-based UFLS scheme should be easily integrated with other RASs. This flexibility can be achieved by utilizing the layered architecture; and (3) UFLS requires fast decision-making and quick response. Therefore, the lightweight average-consensus is selected rather than other distributed optimization algorithms.

Next, from the software engineering point of view, the MAS should be modeled and evaluated before implementation. Therefore, in this case, agents of the UFLS scheme and the power grid are modeled in MATLAB and DIgSILENT, respectively. Data from the transient simulation is fed to agents for evaluation and debugging purpose. Of course, researchers have many options (different programming languages and software tools) in this step. For example, agents can be modeled in Python and DIgSILENT can be replaced by comparable tools. If the physical environment and intelligent agents are modeled in different tools/platforms, the interface and interactions between them becomes a critical issue. Typically, each tool provides a limited number of interfaces for the interactions. Therefore, after a tool is determined, options for other tools will be reduced.

Once the design has been evaluated, implementation is the final stage except for maintenance. In Section 3, a survey on multi-agent platforms, frameworks, and simulators is reported. Again, a critical issue is how to select the tool properly for developers. Some rules apply: (1) Requirements on agent performance and response time are highly dependent on the supported programming language. The C/C++ implementation of an algorithm should be faster than a JAVA implementation. Thus, VOLTTRON and RIAPS are more suitable for the UFLS scheme compared with JADE™, Agent Factory, etc.; (2) From a cost perspective, an open-source platform is preferred especially if the budget is constrained. In addition, flexibility is provided by the open-source code. The platform can be revised by developers for convenience of agent implementation. Both VOLTTRON and RIAPS are open-source; and (3) If agents need to be implemented in hardware nodes/boards, the supported operation system becomes an important issue. Most single-board computers, e.g. Raspberry Pi and BeagleBone Black, support Linux well. Although some of them also support Windows, there is an extra cost to purchase the Windows OS and associated software for development. In addition, the burden on running the OS should be minimized for agents to reboot quickly from a fault. Both VOLTTRON and RIAPS support Linux well.

5. Conclusion and discussion

Research on MASs is mainly focused on system architecture, consensus algorithm, distributed optimization, and software tools for simulation and implementation. It is anticipated that agent technologies will continue to evolve in the future. In addition, more industries are looking for ways to use MASs, indicating that agent technologies are making good progress in commercial use.

The software development process of MASs requires technical support from the selected platform. It matters whether documentation and interface are user-friendly. The development efficiency can be significantly enhanced if good software development kit (SDK) is provided. Once a MAS is deployed, maintenance becomes a long-term issue concerning the complex environment and geographically widely distributed nodes. Specialized and professional services on maintaining distributed hardware devices and software components are critical in future.

Acknowledgment

The authors would like to thank Dr. Marino Sforna from TERNA, Rome, Italy, for his helpful suggestions.

The views and opinions of the authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work is supported by the Advanced Research Projects Agency - Energy (ARPA-E), U.S. Department of Energy, at Washington State University under [grant number DE-AR0000666] through the RIAPS project, 'Resilient Information Architecture Platform for the Smart Grid'. This research was supported by EU Framework Programme FP7 at University College Dublin (UCD) through the AFTER project, 'A Framework for electrical power sysTems vulnerability identification, dEfense and Restoration'.

Notes on contributors

Jing Xie received his B.E. from Tongji University, Shanghai, China, in 2010. He was a M.S. student at Tongji University from 2010 to 2011. Dr. Xie received his Ph.D. degree at University College Dublin, Ireland, in 2015. He is currently an Assistant Research Professor with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA, USA. His research interests include physical security of substations, distributed control of power systems, and testbed technologies.

Chen-Ching Liu **(F'94)** is Boeing Distinguished Professor at Washington State University, Pullman, WA. He is also Visiting Professor at the School of Mechanical and Materials Engineering, University College Dublin, Ireland. At WSU, Professor Liu serves as Director of Energy Systems Innovation Center (ESIC). Professor Liu received an IEEE Third Millennium Medal in 2000 and the Power and Energy Society Outstanding Power Engineering Educator Award in 2004. In 2013, Dr. Liu received a Doctor Honoris Causa from Polytechnic University of Bucharest, Romania. Professor Liu is a Fellow of the IEEE and Member of the Washington State Academy of Sciences.

References

[1] Nuthalapati S, Phadke AG. Managing the grid: using synchrophasor technology. IEEE Power Ener Mag. 2015 Sep;13(5):10–12.
[2] "Solar industry data – solar industry growing at a record pace". Solar Energy Industries Association (SEIA®). [Accessed 2016 Nov 21].
[3] Jian L, Zheng Y, Xiao X, et al. Optimal scheduling for vehicle-to-grid operation with stochastic connection of plug-in electric vehicles to smart grid. Appl Ener. 2015;146:150–161.
[4] Chan CC, Jian L, Tu D. Smart charging of electric vehicles – integration of energy and information. IET Electr Syst Transp. 2014;4(4):89–96.
[5] Wang Y, Nazaripouya H, Chu C-C, et al. Vehicle-to-grid automatic load sharing with driver preference in microgrids. Proceedings of Innovative Smart Grid Technologies (ISGT Europe '14); 2014 Oct; Istanbul, Turkey.
[6] Wang Y, Shi W, Wang B, Chu C-C, Gadh R. Optimal operation of stationary and mobile batteries in distribution grids. Appl Energy. Mar. 2017;190:1289–2017.
[7] Yu N-P, Liu C-C. Multiagent systems. In Advanced solutions in power systems: HVDC, FACTS, and artificial intelligence. Hoboken, NJ: John Wiley & Sons; 2016. p. 903–930.
[8] Wooldridge M. An introduction to multiagent systems. New Jersey: Wiley; 2008.
[9] Weiss G. Multiagent systems: a modern approach to distributed artificial intelligence. Cambridge: MIT Press; 1999.
[10] McArthur SDJ, Davidson EM, Catterson VM, et al. Multi-agent systems for power engineering applications – part I: concepts, approaches, and technical challenges. IEEE Trans Power Syst. 2007 Nov;22(4):1743–1752.
[11] McArthur SDJ, Davidson EM, Catterson VM, et al. Multi-agent systems for power engineering applications – part II: technologies, standards, and tools for building multi-agent systems. IEEE Trans Power Syst. 2007 Nov;22(4):1753–1759.
[12] Catterson VM, Davidson EM, McArthur SDJ. Practical applications of multi-agent systems in electric power systems. Eur Trans Electr Power. 2012;22(2):235–252.
[13] Xie J, Liu C-C, Sforna M, et al. On-line physical security monitoring of power substations. Int Trans Electr Energy Syst. 2016 Jun;26(6):1148–1170.
[14] Liu C-C, Jung J, Heydt GT, et al. The strategic power infrastructure defense (SPID) system. A conceptual design. IEEE Control Syst Mag. 2000 Aug;20(4):40–52.
[15] Li H, Rosenwald GW, Jung J, et al. Strategic power infrastructure defense. Proc IEEE. 2005 May;93(5):918–933.
[16] Jung J, Liu C-C, Tanimoto SL, et al. Adaptation in load shedding under vulnerable operating conditions. IEEE Trans Power Syst. 2002 Nov;17(4):1199–1205.
[17] Yu N-P, Liu C-C, Price J. Evaluation of market rules using a multi-agent system method. IEEE Trans Power Syst. 2010 Feb;25(1):470–479.
[18] Xie J, Liu C-C, Sforna M. Distributed underfrequency load shedding using a multi-agent system. Proceedings of IEEE PowerTech (POWERTECH '15); 2015 Jul; Eindhoven, Netherlands.
[19] Emfinger W, Dubey A, Volgyesi P, et al. Demo abstract: RIAP – a resilient information architecture platform for edge computing. Proceedings of 2016 IEEEACM Symposium on Edge Computing (SEC); 2016 Oct; Washington, DC, USA. p. 119–120.
[20] "VOLTTRON™ – Tranformative Wave," DOE [Online]. Available from: https://energy.gov/eere/buildings/volttron [Accessed 2016 Nov 21).
[21] Panwai S, Dia H. Neural agent car-following models. IEEE Trans Intell Transp Syst. 2007 Mar;8(1):60–70.
[22] Jung J, Liu C-C, Hong M, et al. Multiple hypotheses and their credibility in on-line fault diagnosis. IEEE Trans Power Deliv. 2001 Apr;16(2):225–230.
[23] Hexmoor H. Stages of autonomy determination. IEEE Trans Syst Man Cybern C Appl Rev. 2001 Nov;31(4):509–517.

[24] Pease M, Shostak R, Lamport L. Reaching agreement in the presence of faults. J ACM. 1980 Apr;27(2):228–234.
[25] Lamport L. The part-time parliament. ACM Trans Comput Syst. 1998 May;16(2):133–169.
[26] Lamport L and others. Paxos made simple. ACM SIGACT News. 2001 Dec;32(4):18–25.
[27] Olfati-Saber R, Murray RM. Consensus problems in networks of agents with switching topology and time-delays. IEEE Trans Autom Control. 2004;49(9):1520–1533.
[28] Xiao L, Boyd S. Fast linear iterations for distributed averaging. Syst and Control Lett. 2004;53:65–78.
[29] Laudon K, Laudon JP. Management information systems: managing the digital firm. Vol. 7. New Jersey: Pearson Education, Inc., Upper Saddle River; 2015.
[30] Ricordel P-M, Demazeau Y. From analysis to deployment: a multi-agent platform survey. First International Workshop on Engineering Societies in the Agents World (ESAW); 2000 Aug; Berlin, Germany. p. 93–105.
[31] Kravari K, Bassiliades N. A survey of agent platforms. J Artif Soc Soc Simulat. 2015;18(1):11.
[32] Bellifemine FL, Caire G, Greenwood D. Developing multi-agent systems with JADE. Vol. 7. Hoboken, NJ: John Wiley & Sons; 2007.
[33] Russell S, Jordan H, O'Hare GMP, et al. Agent factory: a framework for prototyping logic-based AOP languages. Proceedings of 9th German Conference on Multiagent System Technologies (MATES '11); 2011 Oct; Berlin, Germany. p. 125–136.
[34] Mallett E. NAESB developing framework for current and future grid interoperability. Nat Gas Electr. 2015 Nov;32(5):6–10.
[35] Akyol B, Haack J, Carpenter B et al. Volttron: an agent execution platform for the electric power system. Proceedings of the Third International Workshop on Agent Technologies for Energy Systems; 2012 Jun; Valencia, Spain. p. 117–131.
[36] VOLTTRON 3.0: User Guide. Pacific Northwest National Laboratory (PNNL). 2015 Nov.
[37] Galland S, Gaud N, Rodriguez S, et al. Janus: another yet general-purpose multiagent platform. Proceedings of 7th Agent-Oriented Software Engineering Technical Forum (TFGASOSE-10); 2010 Dec; Paris, France.
[38] Luke S, Cioffi-Revilla C, Panait L, et al. Mason: a multiagent simulation environment. Simulation. 2005;81(7):517–527.
[39] Grignard A, Taillandier P, Gaudou B, et al. GAMA 1.6: advancing the art of complex agent-based modeling and simulation. Proceedings of 16th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA, 2013); 2013 Dec; Dunedin, New Zealand. p. 117–131.
[40] Akyol B, Haack J, Tews C, et al. An intelligent sensor framework for the power grid. 5th International Conference on Energy Sustainability; 2011 Aug; Washington, DC, USA. p. 1485–1494.
[41] Haack J, Akyol B, Carpenter B, et al. Volttron: an agent platform for the smart grid. Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems (AAMS); 2013; Richland, SC. p. 1367–1368.
[42] Haack J, Akyol B, Tenney N, et al. VOLTTRON: an agent platform for integrating electric vehicles and smart grid. International Conference on Connected Vehicles and Expo (ICCVE); 2013 Dec; Las Vegas, Nevada, USA. p. 81–86.
[43] Khamphanchai W, Saha A, Rathinavel K, et al. Conceptual architecture of building energy management open source software (BEMOSS). Proceedings of Innovative Smart Grid Technologies (ISGT Europe '14); 2014 Oct; Istanbul, Turkey. p. 1–6.
[44] Khamphanchai W, Pipattanasomporn M, Kuzlu M, et al. An agent-based open source platform for building energy management. Proceedings of Innovative Smart Grid Technologies-Asia (ISGT ASIA '15); 2015 Nov; Bangkok, Thailand. p. 1–6.
[45] Katipamula S, Lutes RG, Ngo H, et al. Transactional network platform: applications. Pacific Northwest National Laboratory (PNNL); 2013 Oct. (Tech. Rep. PNNL-22941).
[46] Nakajima A, Morita M, Hayashi T, et al. Development of an advanced wide-area special protection system. J Int Counc Electr Eng. 2013;3(4):334–339.
[47] Xue F, Li Y, Song X-F, et al. Impact of large-scale wind power integration on the UFLS scheme. J Int Counc Electr Eng. 2014;4(3):251–257.
[48] Terzija VV. Adaptive underfrequency load shedding based on the magnitude of the disturbance estimation. IEEE Trans Power Syst. 2006;21(3):1260–1266.