Electrical Power and Energy Systems 114 (2020) 105375

Contents lists available at ScienceDirect

Electrical Power and Energy Systems

journal homepage: www.elsevier.com/locate/ijepes

Cyber-physical framework for emulating distributed control systems in smart grids

Catalin Gavriluta[b,a,*], Cedric Boudinet[a], Friederich Kupzog[b], Antonio Gomez-Exposito[c], Raphael Caire[a]

[a] Univ. Grenoble Alpes, CNRS, Grenoble INP, G2Elab, 38000 Grenoble, France
[b] AIT Austrian Institute of Technology, Giefingasse 2, 1210 Wien, Austria
[c] Seville University, Calle San Fernando 4, 41004 Seville, Spain

ARTICLE INFO

Keywords:
Cyber-physical systems
Smart grid
Distributed control and optimization

ABSTRACT

This paper proposes a cyber-physical framework for investigating distributed control systems operating in the context of smart-grid applications. At the moment, the literature focuses almost exclusively on the theoretical aspects of distributed intelligence in the smart-grid, meanwhile, approaches for testing and validating such systems are either missing or are very limited in their scope. Three aspects need to be taken into account while considering these applications: (1) the physical system, (2) the distributed computation platform, and (3) the communication system. In most of the previous works either the communication system is neglected or over-simplified, either the distributed computation aspect is disregarded, either both elements are missing. In order to cover all these aspects, we propose a framework which is built around a fleet of low-cost single board computers coupled with a real-time simulator. Additionally, using traffic control and network emulation, the flow of data between different controllers is shaped so that it replicates various quality of service (QoS) conditions.

The versatility of the proposed framework is shown on a study case in which 27 controllers self-coordinate in order to solve the distributed optimal power flow (OPF) algorithm in a dc network.

1. Introduction

For the past few years, distributed control algorithms and multi-agent systems have been receiving a lot of attention in the context of modern energy systems. Their highly scalable nature makes them a strong candidate for various smart-grid applications. However, the added complexity given by the close interlacing with the information and communication infrastructure makes the validation of these applications a challenge.

Generally, in order to analyze such a distributed cyber-physical system one needs three main components: (1) the physical system, (2) the distributed computation platform, i.e, the controllers, and (3) the communication network between the controllers. At the moment, most of the existing work on distributed control in smart-grid applications tends to neglect one or more of these components when proposing new algorithms and applications. Often referenced in this field, the work of Kraning et al. [1] proposes a fully distributed optimization algorithm for solving the problem of optimal daily energy dispatch for a microgrid. Their conclusion is that for a fully granular implementation, i.e., one computing processor for each device, without the communication delays, the time required for finding a solution would be less than one second, regardless of the size of the network. While the results are very encouraging, the authors tested the proposed algorithm on a multi-processor computation platform, ignoring the interaction with the physical system, as well as the communication layer. Taking these aspects into consideration, especially the communication layer, will considerably increase the time needed for the algorithm to find a solution.

In the same note, the work presented in [2] revolves around the idea of distributed consensus applied to dc microgrid control. The proposed algorithms are validated on a network with four terminals using hardware in the loop (HIL) methods and dSPACE for fast control prototyping. While this approach is suitable for testing a small-scale system, scaling this approach for testing larger networks will prove very cumbersome and expensive.

Numerous other examples of distributed control systems related to agent based control of microgrids [3–9], distributed OPF seeking controllers in ac power distribution systems [10], or energy market applications [11,12] can be found in the literature. However, in all of them, it is very difficult to assess the applicability of the algorithms to

* Corresponding author at: AIT Austrian Institute of Technology, Giefingasse 2, 1210 Wien, Austria.
E-mail address: catalin.gavriluta@ait.ac.at (C. Gavriluta).

https://doi.org/10.1016/j.ijepes.2019.06.033
Received 16 June 2018; Received in revised form 12 June 2019; Accepted 14 June 2019
Available online 09 July 2019

real-life problems. This is mainly due to oversimplifications that are introduced in the validation stage.

In terms of frameworks and toolsets, a few approaches have emerged in the past few years, mainly from the co-simulation field. For example, in [13] DIgSILENT coupled with a custom JAVA-based multi-agent system is used for testing an adaptive power flow control strategy in transmission systems. Other approaches designed for analyzing the performance of wide area monitoring systems, as well as issues related to cyber-security, are shown in [14,15].

Mosaik [16,17] is a Python based open source software package maintained and actively developed by OFFIS. Mosaik is a flexible co-simulation framework that enables the coupling of existing simulators and models in order to tackle complex smart-grid applications.

In [18] a Simulation Message Bus approach for co-simulation and rapid prototyping of networked systems was proposed. This approach was further developed into Lablink [19], an in-house tool used at the Austrian Institute of Technology. While Mosaik is mainly focused on simulator coupling, Lablink is used for coupling laboratory equipment with simulators and other software modules.

Similar in nature to Lablink, OpSim is a co-simulation environment actively developed by Fraunhofer IEE [20,21]. It allows coupling of both software and hardware through a common message bus. Unlike Mosaik, both Lablink and OpSim are proprietary tools and are not available as open-source.

VILLAS [22,23] is an open source co-simulation platform developed at the Institute for Automation of Complex Power Systems at Aachen University. VILLAS is designed as a holistic framework with a modular and generic architecture and it is mainly targeting coupling of geographically distributed research infrastructure and real-time simulators [24,25].

Bottaccioli et al. propose in [26] a modular co-simulation approach built around paradigms that are specific to internet of things (IoT)-based architectures. We refer the readers to [26,27] for a comprehensive overview of several co-simulation platforms, some designed for single-use cases, other allowing more flexibility and multiple-use cases; some that allow both hardware and software in the loop, and some that remain only at software level.

In this paper we propose and evaluate a cyber-physical framework that allows the investigation of distributed control and optimization strategies. The main focus is on applications that involve a large number of controllers that need to interact and self-coordinate. The focus is not on time critical applications, but rather on applications where the complexity arises from the interaction between the different controllers.

Our main goal was to propose a system that allows experimentation at each of the three layers mentioned above, i.e., physical, distributed computation, as well as communication layer. Moreover, we aimed to make this framework easily accessible and scalable, therefore we tried to use low-cost devices as well as modern open-source tools.

In the following section we present an overview of several ICT concepts that are relevant in the context of the proposed framework. Afterwards, Section 3 presents the framework and Section 4 shows how it can be employed for the evaluation of a distributed optimization algorithm used for solving the OPF problem of a dc network. Finally, Section 5 concludes the work.

2. Overview of relevant ICT concepts

One of the main challenges of developing the proposed framework was understanding and evaluating the vast pool of ICT-related concepts that fall under the scope of our application. A distributed system that implements algorithms for control and optimization in the smart grid is at the border between several research areas. Therefore, as shown in Fig. 1, different protocols, data-models, and architectures originating from smart grid, multi-agent systems, distributed computing, or IoT philosophies can be found in the literature.

In the following paragraphs we will discuss some of the most popular ones. The list is not by any means exhaustive. Its main purpose is to cover several concepts that are typically encountered in the literature; concepts which we also evaluated as possible solutions for the inter-controller communication in the proposed system.

2.1. IP application layer protocols

As shown in [28], the possible communication solutions for smart-grid applications vary widely depending on data rate and latency requirements as well as on geographical span. Out of all the available solutions, internet protocols (IP) are the most proven technology and the most versatile, fitting a large array of scenarios. Therefore, in terms of communication protocols we focused only on the IP protocol. As it is well known, IP has a layered architecture with the top most layer being the application layer. Three application layer protocols are often referenced in the smart-grid and distributed systems literature.

2.1.1. XMPP

The first XMPP (Extensible Messaging and Presence Protocol) version was released in 1999 as a protocol for instant messaging. XMPP has proven to be extremely successful from the start and has been widely used in numerous communication applications. Google's and Facebook's chat applications, as well as WhatsApp, all rely on XMPP and they handle hundreds of millions and in some cases over a billion users [29].

According to [30], XMPP is a very stable and proven technology, it is secure, extensible, and highly scalable. All these characteristics have enabled XMPP to outgrow its original purpose and find its way in other applications, such as voice over IP (VoIP), videoconferencing, file transfer, gaming, etc. Also, lately, it has started to become a solid candidate for IoT and machine-to-machine communication over public networks. XMPP is also slowly making its way into the IEC standardization landscape, and various related applications have started to emerge in the literature. For example, [31] shows an approach based on IEC 61850 mapped to XMPP for the integration of decentralized energy resources and loads into the smart energy grid and into a smart energy market. Meanwhile, in [32] XMPP is evaluated as an enabler of the IEC 61499 standard for distributed event-based automation systems in the context of smart grids and energy applications.

2.1.2. MQTT

Message Queue Telemetry Transport (MQTT) is a relatively new open source protocol that is gathering a lot of interest in the IoT community. The MQTT messaging protocol employs a publish-subscribe pattern which was designed to be extremely lightweight in terms of required processing power as well as network bandwidth. This makes MQTT well suited for applications that involve small remote sensors and actuators with limited processing ability and memory, or systems connected to unreliable communication networks. MQTT is highly scalable, making it possible to create systems that involve hundreds or even thousands of remote sensors or devices [33].

The only aspect that we consider to be a drawback for both MQTT and XMPP in the context of our application is their broker/server based architecture. Clients do not have a direct connection to each other, but instead, the communication is handled by a server with which both clients are registered. If the server becomes unavailable, all communication is compromised. Also, since the server is placed between the two clients, its performance will have a direct impact on the QoS of the communication loop. Of course, these aspects could be mitigated with proper infrastructure, however, they come in contradiction with the idea of decentralized or distributed systems.

2.1.3. HTTP

The Hypertext Transfer Protocol (HTTP) is probably the most known protocol from the IP suite as it sits at the foundation of the world

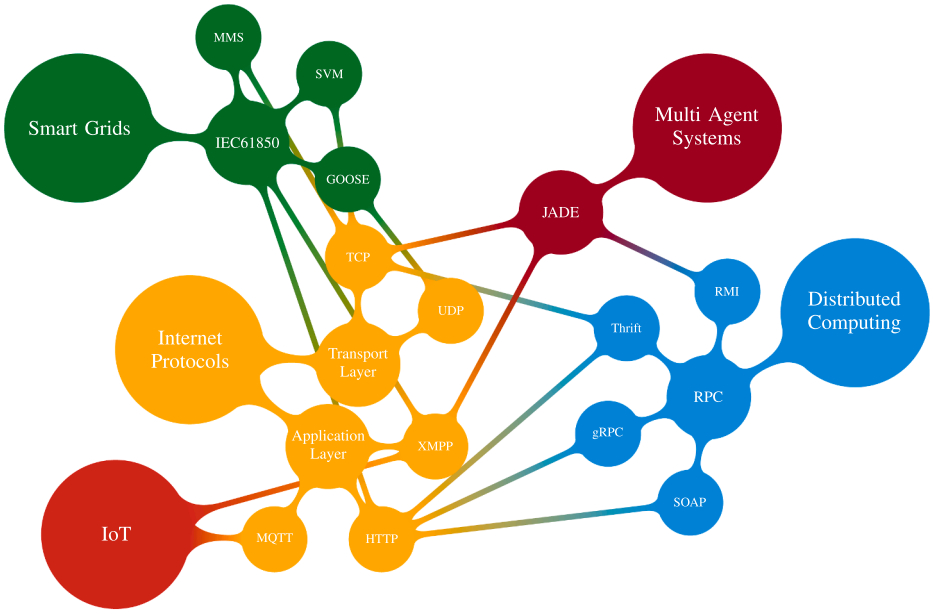

Fig. 1. Overview of relevant ICT concepts.

wide web. HTTP is extremely popular, being omnipresent in modern every-day life. Therefore, it has the great advantage that all the network infrastructure is already configured to efficiently work with it. For example, an application based on HTTP will most probably encounter no problems due to firewalls or other network security measures.

HTTP is client–server and implemented around a stateless request-response paradigm, i.e., the client sends a request to the server and keeps the communication channel opened until the server responds or the request times out. While in XMPP the role of the server is to enable the communication between clients, in HTTP the communication only happens between the client and the server. Unlike XMPP and MQTT, HTTP implies direct point to point communication.

All these protocols, i.e., XMPP, MQTT, and HTTP, have the great advantage of having the support of a very dynamic and open community. Servers, clients, or libraries can be found for almost every operating system and programming language. Therefore, it is relatively easy to develop applications based on them.

2.2. Remote Procedure Calls (RPCs)

Remote procedure calls are a form of inter-process communication mechanism commonly used in distributed computing. RPCs have been created to expand device interoperability. The basic idea behind the concept, as its name implies, is to give a process the possibility of calling routines that belong to a different process that operates on the same connected network.

Numerous protocols have been developed around the concept of RPC, with one of the most popular being SOAP (Simple Object Access Protocol). SOAP implemented with HTTP as a transport mechanism is one of the main building blocks of web-services, a concept widely used in internet-based applications.

SOAP uses XML to encode its calls, which has a clean structure, but has the disadvantage of being verbose. Therefore, more recent RPC frameworks lean towards more compact encoding formats, such as JSON or binary. One such framework is gRPC, a new open-source RPC system created by Google. By taking advantage of the feature set of the latest version of HTTP, i.e., HTTP/2, gRPC facilitates the development of distributed service-based applications that are characterized by low latency and high scalability [34].

Apache Thrift is another popular RPC framework that also has its origins at a large tech company, this time, Facebook. Designed for the development of scalable cross-platform and cross-language services, Apache Thrift supports several transport protocols (HTTP, TCP sockets, etc.) as well as several compact encoding protocols (JSON, binary, etc.) [35].

A great advantage of both gRPC and Apache Thrift is that they offer a language-agnostic interface definition language (IDL). In both these frameworks, a service is declared as an IDL file which is then processed by a code generator in order to produce language-specific bindings.

2.3. JADE and FIPA-ACL

One of the most popular frameworks for the development of multi-agent systems is JADE [36], an open source Java-based software platform that implements the ACL (Agent Communication Language) standard proposed by FIPA (Foundation for Intelligent Physical Agents). JADE covers much more than the inter-agent communication, but at this level it introduces the idea of agent containers. Inside the same container, agents communicate through RMIs (Remote Method Invocations) which is Java's approach to RPCs. Messages between containers are handled either via TCP sockets or via XMPP.

While JADE is a solid platform, it has the major drawback of being extremely limited from an interoperability point of view. In other words, all the agents, across all the devices of the distributed system have to be implemented in Java around JADE. To our knowledge, there is no direct approach for mixing JADE agents with other agents developed using different frameworks and languages.

Besides the limited interoperability, another aspect that raises concerns regarding the future of JADE is the status of FIPA and its activity on ACL. According to their website, the last meeting of FIPA has been in 2005.

2.4. IEC 61850

While JADE has its roots in the multi-agent systems community, IEC 61850 arose from the smart-grid field.

Historically, IEC 61850 is a standard that facilitates the interoperability of substation automation equipment developed by different manufacturers. However, in the last few years the standard has been continually expanding, adding new sections for emerging technologies

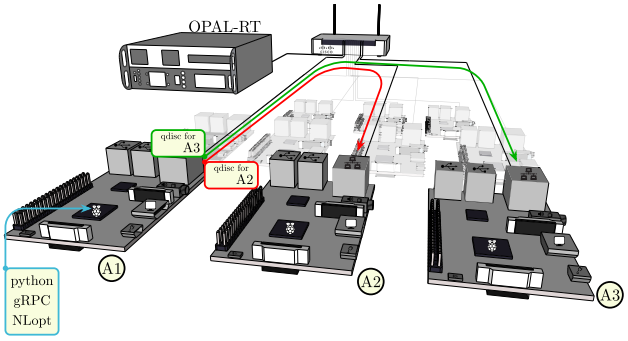

Fig. 2. Framework for real-time testing of distributed power system applications.

such as distributed energy resources and e-mobility. IEC 61850 provides abstract data models that can be mapped to various communication protocols. At the moment, the standard provides documents for mapping to MMS (Manufacturing Message Specification) and work for more world-wide-web oriented approaches is underway. According to [37], the upcoming IEC 61850-8-2 standard will provide a mapping to XMPP. However, other examples such as [38] show the potential of combining the models of the standard with contemporary web protocols.

Given its widespread acceptance, IEC 61850 in combination with modern web protocols could prove to be a suitable platform also for our application. However, the main drawback is the lack of community support and readily available tools. Despite the major excitement around IEC 61850, at the moment there are only two libraries openly available, one written in C [39], and one in Java [40], both of them offering a mapping of the standard to MMS.

3. Proposed framework for evaluating distributed power system applications

As can be seen in Fig. 2, the proposed framework is build around a fleet of single board computers, namely Raspberry Pis, connected to a real-time simulator, namely OPAL-RT.

As mentioned in the introduction, there are three main components that need to be covered. In the next subsections we will explain how they were addressed in the proposed framework and how they interact between them.

3.1. Physical system

Electric networks are highly complex and expensive systems and, therefore, it is rarely possible to experiment on real systems. One can build reduced laboratory-scale replicas, but they typically lack flexibility, as they are built for a specific experiment and also their cost and maintenance is usually rather high. Under these conditions, the typical approach in power-systems research is to run models of the physical system on high performance computation-machines which are able to execute complex calculations in a very small time step. These devices are commonly known in the field as real-time simulators (RTS) and several key players, such as RTDS, OPAL-RT, or Typhoon, are active on the market at the moment. RTSs are commonly used in combination with signal amplifiers for performing *power hardware in the loop* (PHIL) validation of various electrical devices. Similarly, they can also be used for validating and testing individual controllers in *controller in the loop* (CIL) setups.

For the proposed framework we also used a real-time simulator for implementing the behavior of the physical system, as this approach offers maximum flexibility, while at the same time allows the connection of real physical elements for PHIL or CIL experiments.

The main problem is that, as mentioned earlier, real-time simulators

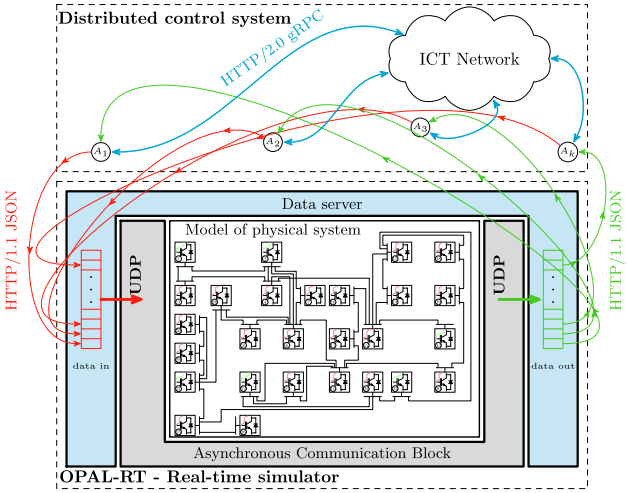

Fig. 3. Interfacing the real-time simulator with the distributed computation platform.

were designed mainly with PHIL or CIL experimentation in mind, therefore, the commercially available models are rather limited in their communication interfaces. In the type of large scale distributed applications that we consider there would be tens or even several hundreds of controllers that need to access data from different parts of the electrical network in a completely asynchronous manner. By default, RTSs do not offer an out-of-the-box solution for dealing with this sort of scenarios.

Fig. 3 shows the method that we propose for mitigating this issue. Our implementation is based on OPAL-RT, as it is the RTS that our laboratory is equipped with. However, to our knowledge, the same approach can be used with other RTSs. As shown in Fig. 3, OPAL-RT is in charge of emulating the physical system. The figure shows our study case, which is a dc electrical network, however, any other dynamical system that fits in the computation limits of the real-time simulator could be used.

In order to allow external controllers to interact with the physical system, we propose to run in parallel with the real-time process an additional process on the OPAL-RT in the form of a data server. The data server keeps two internal buffers, one for inputs and one for outputs, and acts as a gateway into the physical system. Any request from the controllers is being handled using the information stored in these buffers, which are periodically synchronized via UDP with the information from the real-time process.

While this approach allows for a large number of controllers to access information from the real-time process, it introduces an additional layer between the controller and the physical process.

The data-server is implemented as a web-service and external devices can read and write data from the server via HTTP. The data-server synchronizes the information from the buffers with the real-time process every T_{comm}, which in our implementation is 10 ms. Additionally, as it will be seen later in Section 4.1, in our scenario it takes around 3–4 ms for the data-server to resolve a request. Therefore, in the worst case scenario, the signals at the controller level are delayed by 14 ms with respect to the real-time process. For some applications, such as protection, this delay might be unacceptable, but for the type of applications that we considered, i.e., distributed optimization and higher level control, this approach works perfectly. If necessary, the delay could be reduced by decreasing T_{comm}, but it has to be noted that a too small value might have a negative impact on the real-time execution of the physical model. The delay could also be slightly reduced by using a lower level protocol instead of HTTP for the communication with the OPAL-RT.

3.2. Distributed computation system

The previous section showed how the interaction between the controllers and the physical process can be emulated by extending the capabilities of the RTS. The second part of our framework is the distributed computation engine. We considered a fleet of Raspberry PIs (RPIs) to be the most suitable choice for this task, as the cost of scaling up the system is rather low. One can see similar approaches where clusters with thousands of RPIs are being built in order to emulate exascale computing solutions. Los Alamos National Laboratory is building such a cluster with 3000 cores, with plans of scaling to 40000 cores in the near future [41].

The RPI is a single board ARM-based computer that has been gaining a lot of popularity in the past few years. Besides its exceptionally low-cost, there are other numerous advantages of working with the RPI. Firstly, its Linux-based operating system opens up the possibility of using high level programming languages such as Python, R, or Octave and provides easy access to a large collection of open-source scientific computing libraries. This greatly simplifies the process of porting ideas and concepts from other scientific communities, such as machine learning or artificial intelligence, to smart-grid applications.

Performance wise, the model used in our setup is equipped with a 1.2 GHz quadcore processor and 1 GB of RAM, which makes it a suitable candidate for embedded applications of medium complexity.

The controllers and the inter-controller communication, as well as the interface with OPAL-RT, were developed using Python. We selected Python because it is a powerful scientific programming language that allows fast prototyping and concept validation by providing access to a large collection of libraries and modules.

The controllers are hosted on the RPIs and each of them can exchange data with the real-time model of the physical system via the data server, as shown in the previous sub-section.

As seen from Section 2, while there are numerous communication protocols and concepts available, at the moment there is no clear consensus for what would be the most suitable communication protocol and transport mechanism for distributed applications in a future smart-grid. As our area of interest is closer to the field of distributed computing than to the one of telemetry, we selected to implement the communication between controllers using RPCs, more specifically using gRPC. We selected gRPC because it is based on HTTP/2 and uses a compact protocol for encoding the procedure calls. According to [34], Google's internal services register a throughput of 10^{10} RPCs per second. Therefore, we considered gRPC as a perfect candidate for highly scalable systems.

3.3. Communication network

As shown in Fig. 2, all the RPIs are connected to the same ethernet network, together with OPAL-RT. However, one has to distinguish between two different flows of information that occur over this network.

The first one is between the controllers running on the RPIs and OPAL-RT. As described previously, this emulates the local link between the controller and the physical system, typically, a sensor-controller-actuator loop.

The second flow of information is between the controllers. In a distributed system, this would typically travel across an ICT network with certain characteristics in terms of QoS. For example, communication between on-site controllers through a dedicated network will be fast and reliable, meanwhile, GSM communication with remote controllers in the field might be lossy and with large latencies.

The first choice for covering this type of scenarios is to use a communication network simulator such as OMNET++ [42] or gns3 [43]. However, since our framework involves real-time and physical connections between a relatively large number of hardware components, using a simulator is not feasible. Rather than a simulator, one would need a network emulator for this task. Specialized solutions such as CORE [44] exist, and even some simulators can be used in emulation mode. Setting up such a configuration requires additional hardware and software overhead. However, if a detailed representation of the ICT network is required then this is the only feasible solution.

The method that we propose does not include a detailed representation of the ICT network. However, it has the great advantage that it introduces no overhead and it uses only the components that were already introduced as part of the system. Mainly, since the RPIs used for the controllers are powered by a Linux based operating system, their kernel comes prepackaged with NetEm [45], a software package that provides network emulation functionality. Making use of NetEm we developed an approach for shaping the traffic between the controllers so that it emulates different network conditions.

Our solution is based on the traffic control utility (TC) [46] and NetEm, and it was inspired by the methodology Facebook engineers use for testing services under various network conditions [47]. By using TC at the level of each ethernet card in our fleet of controllers we can set different rules and filters (so called queuing disciplines or qdiscs) in order to enforce different latencies, packet losses, and data rates onto the communication channels.

Qdiscs can be set separately for each individual communication link in the network. Therefore, the traffic between the controllers and OPAL-RT can be left unaltered, meanwhile the inter-controller traffic can be shaped to match a certain QoS. For example, in Fig. 2 A1 has one set of qdiscs to dictate the QoS for the communication link with A2, and a completely different set for communicating with A3.

Using this approach, a larger range of communication conditions can be emulated and it makes it possible to assess the minimal requirements of the ICT infrastructure in order to reach a certain performance for a given application.

4. Study case

As a study case, we are going to use the algorithm presented in [48], where a distributed approach for OPF based secondary control for dc networks is proposed.

Fig. 4 displays the graph representation of the network under study. The green nodes correspond to generator buses, meanwhile the red ones correspond to load buses. The operation of each node is managed by a local controller which has complete access to local measurements and, moreover, it is able to communicate with its neighbors. The local controller also knows the conductance of the adjacent cables to which the node is connected.

Unlike traditional power systems in which the secondary control is centralized, in our case, the distributed controller network is in charge of this task. The local controllers implement a distributed optimization algorithm based on the alternating direction of multiplier method (ADMM). The full details of the algorithm can be found in [48].

Due to the number of iterations required for convergence, ADMM is considered a slow algorithm. However, at the moment, optimization routines in power systems are performed completely offline, and even basic secondary control actions involve time-frames in the range of minutes.

We used the proposed framework in order to obtain practical results regarding the time-frames that are to be expected from a distributed OPF approach based on ADMM. As described in the previous section, the physical system together with the primary control layer were implemented in OPAL-RT. The system is initially modeled using Simulink/SimPowerSystems components then compiled and executed on the real-time target with a time step of 100 us. The data exchange between the data-server and the process of the physical model is triggered every 10 ms.

Meanwhile, the secondary controllers were deployed on the RPIs as Python scripts. Since optimization is required at every iteration of the ADMM algorithm, every controller requires access to a local solver. In this regard we used the NLopt non-linear optimization package [49].

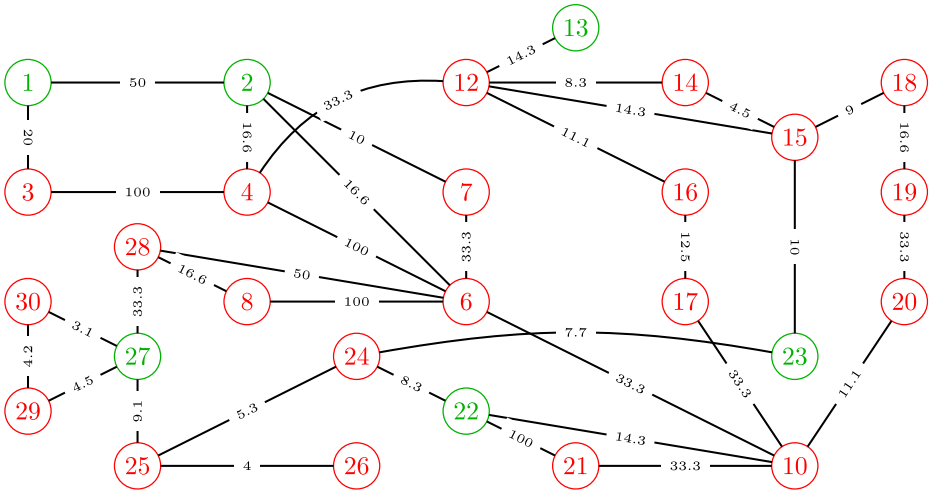

Fig. 4. Graph representation of the 27 terminals MTDC system study case. Green represent generator buses while red nodes represent load buses. The numbers on the edges represent the conductance of the electrical cables connecting the buses together. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

While NLopt provides several solvers, the one that provided the best tradeoff between accuracy and performance for our problem was the COBYLA (constrained optimization by linear approximations) algorithm [50].

4.1. General performance of the framework

We start by providing some general performance metrics for the proposed framework. The data shown in Fig. 5 was compiled from several thousands samples, obtained from different operating scenarios. Firstly, the time it takes for the controllers to access data from the OPAL-RT (A2O), as can be seen from Fig. 5, is smaller than 3.5 ms. This is the delay between the controller and the data-server that runs on OPAL-RT. Secondly, there are two time metrics for the RPCs, i.e., the time it takes for information to flow between controllers. In the first one, $gRPC_{wired}$, the controllers communicate over an 100 Mbit ethernet network, i.e., the RPIs communicate through their wired ethernet card, and the traffic is not shaped in any way through the TC utility. As can be seen from Fig. 5, the average round-trip delay in this case is at around 9 ms.

The RPIs are also equipped with 2.4 GHz 802.11n wireless connectivity. When using this link for the inter controller communication, a considerably larger delay, $gRPC_{WIFI} \approx 57$ ms, is obtained. Also in this case TC is disabled. However, this large delay is mainly due to the fact that in our setup all the controllers are connected to the same wireless access point, which becomes overloaded during peak traffic.

The last metric that we analyzed is the optimization time, i.e., how long does it take for each controller to solve at each iteration the local optimization problem. Over the entire fleet, the average time spent in the optimization routine is ≈18 ms. However, as can be seen from Fig. 5, there is a large variance around this value. Looking at each node individually, we see that the ones that are more connected also spend more time in the optimization routine, e.g., node 6 is the slowest, spending on average ≈64 ms on this task. This is to be expected, as the more connected a node is, the larger the size of its local optimization problem and hence the longer the time required for finding a solution.

4.2. System response to load change

We use this scenario in order to measure the time duration that it takes for the distributed OPF controller to bring the network to a new optimal operating point after a change in the load power. All the results are shown in per unit (p.u.), using 100 MW as base value for the power and 50 kV for the voltage. The network starts in perfect equilibrium at

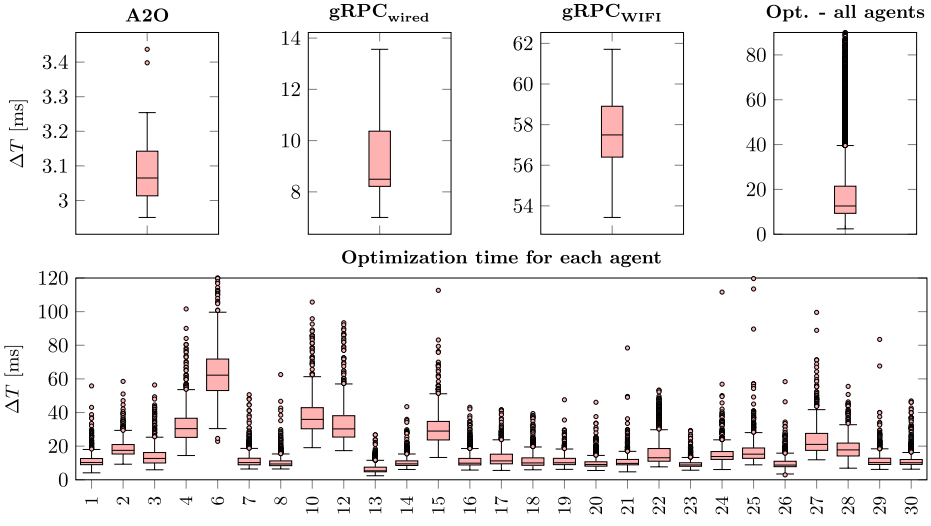

Fig. 5. Performance metrics of the test system.

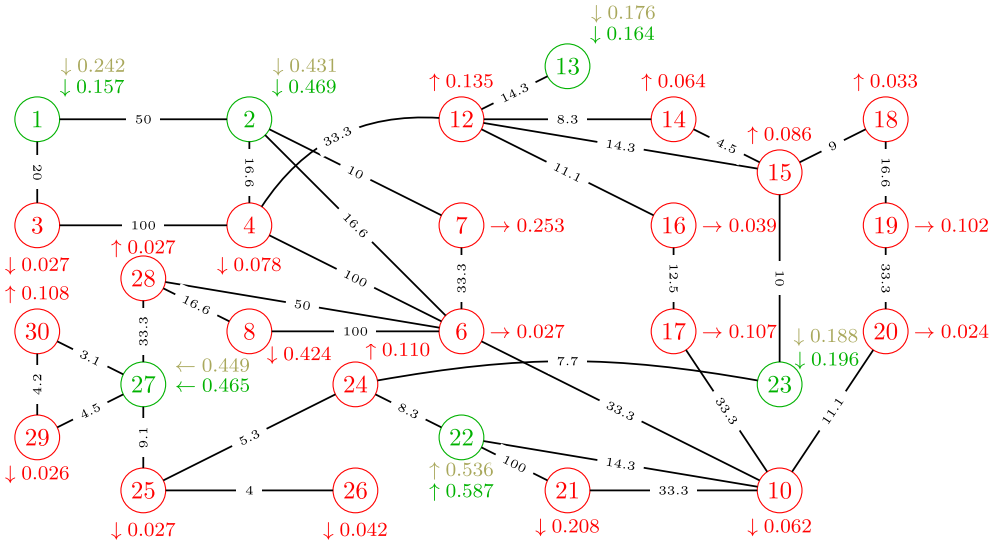

Fig. 6. Power flow diagram. Two values are displayed for every generator. The yellow one represents the power injection after the action of the primary control, meanwhile, the green one is the power after the action of the distributed optimization algorithm. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

nominal conditions, with no power flowing through the network. After approx. 8 s all the loads in the system apply a step change to their power demand. The amplitude of this power step at each node, together with the response of the generators, can be seen in Figs. 6 and 7a,b. Meanwhile, Fig. 7c displays the total system losses.

Both Figs. 6 and 7b show two values for each generator. The first one is the result of the instantaneous reaction of the primary control layer, meanwhile, the second is decided by the coordinated action of the distributed controllers. Since the objective of the implemented algorithm is to reduce the power losses, we can see in Fig. 7c a reduction of 18.57% of this value.

The results of Fig. 7 present just a static snapshot. However, the dynamic response of the network provides a better understanding of the system's operation. These results are displayed in Fig. 8. Here, we show the dynamic response of all the generators, together with two of the loads. We chose not to include the response of all the loads due to space limitations and to keep the exposition of the results clean and concise.

For our study case, according to [48], 250 iterations should be enough for the ADMM algorithm to converge on this particular problem. However, in order to allow for some additional headroom we chose to run it for $n_{\text{it}} = 300$ iterations. We selected the stopping criteria based on the number of iterations because it is simple to implement in practice. Distributed convergence detection for ADMM and for distributed algorithms for that matter is a non-trivial task that is outside the scope of this paper.

Several values are plotted in Fig. 8. The solid black line represents the measured power P^m, which each controller running on the RPI obtains from the model of the physical system through the data-server. The dotted line represents the reference signal P^* which the controllers send to the physical system. Meanwhile, the thick colored line represents the internal signal of the distributed OPF algorithm, P^{admm}.

After a disturbance in the network, the primary control, acting at the physical level, will react instantaneously. This can be seen in the sudden change in the generator power around the 8 s mark. Almost at the same time, the secondary control detects that there is a large difference between the reference power P^* and the measured power P^m and starts a new instance of the optimization algorithm.

Estimating the convergence time T_{conv}, i.e., how long it will take for the algorithm to reach a solution, is not a trivial task, as several sources of uncertainty are present. From Fig. 8 we can see that the algorithm takes around 62 s to run the 300 iterations. However, this value would have been hard to predict without running the experiment.

In order to get an idea about the different parameters that impact T_{conv} one needs to have an overview of how ADMM operates. Broadly speaking, in ADMM we have n controllers that operate in parallel. They all run n_{it} iterations, and at each iteration i every controller k executes the following steps:

1. Solve the local optimization problem. In terms of timing, this step takes τ_i^k. This time is not influenced by the communication delays, but only by the size of the local problem and by the computational resources available for controller k.
2. Send results to neighbors and wait for results from neighbors. This step is highly influenced by the communication delays, as well as by the number of neighbors. We can label this delay as Δ_i^k.
3. Correct the local state with the information received from the neighbors. In terms of timing, this step is negligible, as it only involves some simple linear algebra.

One can express T_{conv} analytically as shown in (1). However, this is not really useful. Getting realistic estimates for τ_i^k and Δ_i^k and how they overlap is a challenge, if not impossible. This is why in order to properly evaluate T_{conv} one needs to run the algorithm in an environment that reproduces as close as possible the real field conditions and this is where the platform that we propose can prove useful.

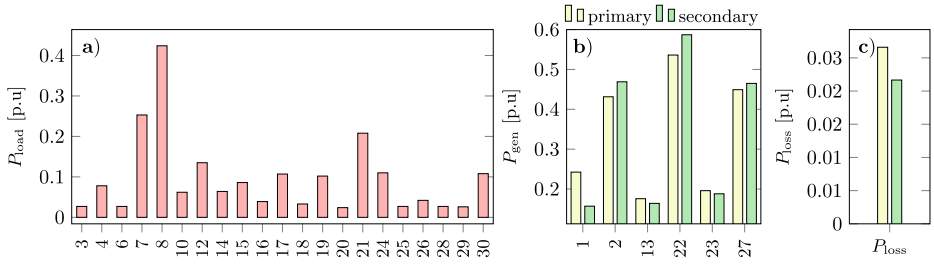

Fig. 7. (a) Power step of loads. (b) Generator output after the action of both control layers. (c) Total network losses.

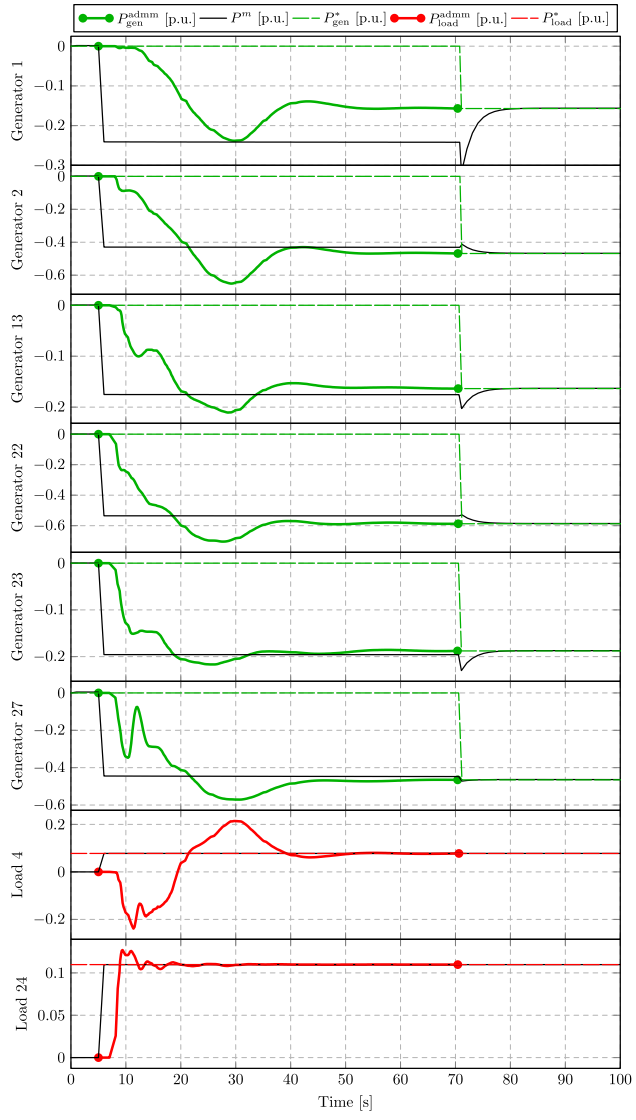

Fig. 8. Large disturbance from nominal condition.

$$T_{conv} = \sum_{i=1}^{n_{it}} \max_{1 \leqslant k \leqslant n} \left(\tau_i^k + \Delta_i^k \right) \tag{1}$$

One can obtain a rough estimate of T_{conv} by doing some preliminary analysis. For example, we know from before that the controller at node 6 is statistically the slowest in solving its local optimization problem, taking on average $\tau^6 \approx 64$ ms. Node 6 is also the most connected, having $n^6_{neigh} = 6$ neighbors. Therefore, we can expect statistically Δ^6, i.e., the time that the controller at node 6 takes to exchange information with its neighbors, to also be the largest.

A rough approximation for Δ^k is shown in (2). Here, n_{msg} represents the number of messages exchanged with each neighbor in the second step of the algorithm described above. Meanwhile, T_{msg} is the average communication delay of a message, and n^k_{neigh} is the number of neighbors of node k.

$$\Delta^k = n_{msg} \cdot n^k_{neigh} \cdot T_{msg} \tag{2}$$

In order to obtain the results of Fig. 8 the 100 Mbit ethernet communication was used. Therefore, taking the average performance for wired communication from the previous subsection, i.e., $T_{msg} = 9$ ms, and using (2) we can obtain a rough estimate for the convergence time of the algorithm $\widetilde{T}_{conv} = 51$ s.

In this particular case, the approximation of 51 s is not too far away from the actual convergence time of 62 s observed in Fig. 8. However, the larger the variance in τ_i^k and Δ_i^k, the worse will this approximation be.

4.3. Traffic shaping. PLC-like communication.

In the second study case we run the same scenario as before, only now, the inter-controller communication traffic is shaped using the methodology described in Section 3.3 in order to emulate the QoS of power line communication (PLC).

PLC is a communication method typically used in classical power systems which uses the electrical cables to also carry data alongside electric power. PLC has a limited data rate and large latencies. Therefore, in order to emulate its performance, additional delay was added for each communication link, as shown in Fig. 9. The delay was set proportional to the length of the electrical cable over which the communication is being carried. The mean round-trip latency was selected as 200 ms, but as can be seen from Fig. 9 and from the overview

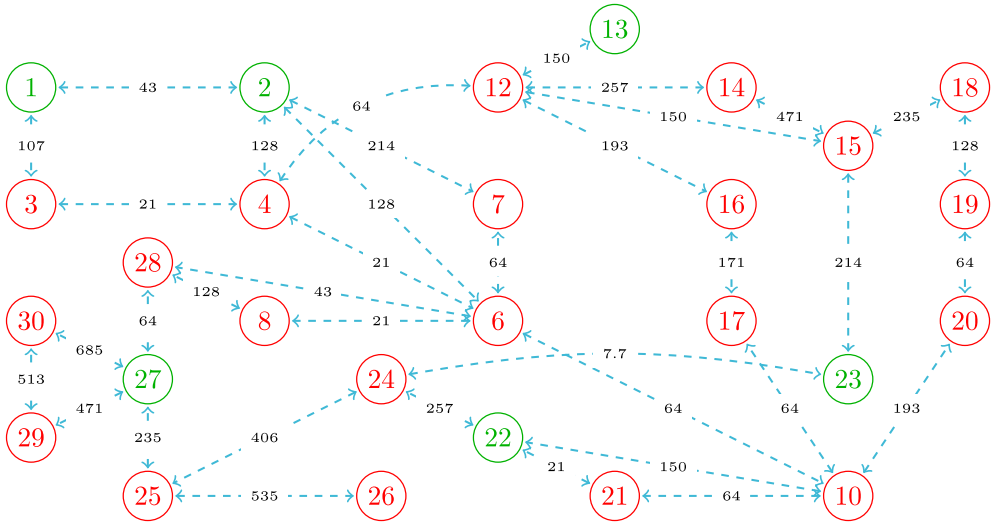

Fig. 9. The numbers on the edges represent the additional latencies in ms added to each communication link in order to emulate a PLC-like communication infrastructure.

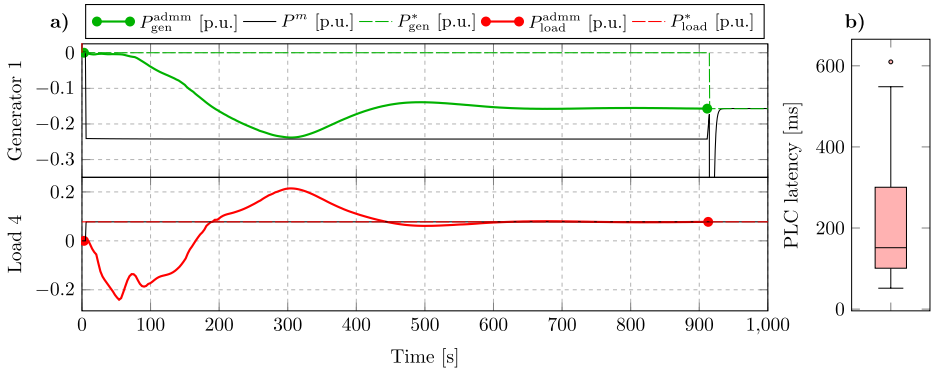

Fig. 10. (a) Large disturbance from nominal condition using PLC for inter-agent communication. (b) PLC latencies.

shown in Fig. 10b, the actual values go up to 600 ms, depending on the length of the cable.

For simplicity, in our experiments the delay that was added to each communication link was fixed to the value shown in Fig. 9. However, NetEm allows for more complex settings. For example, the delay for each communication link can be specified as a probability distribution. Support for normal, pareto, and paretonormal are supported by default, but one can also build custom distributions based on experimental data. Also, since network delays are not purely random, one can specify the amount of correlation between the delays of consecutive data packets. Similarly, one can specify aspects regarding packet loss, duplication, corruption, and re-ordering. As a result, one can replicate quite accurately the conditions of real communication environments without having to simulate the entire infrastructure.

As seen from Fig. 10a, in the case of PLC-like latencies, the 300 ADMM iterations take roughly 15 min to execute, i.e., 15 times slower than in the previous case. The only impact of the larger communication delay on the system is that the distributed algorithm takes longer to find a solution. Therefore, the physical system will operate for a longer time with the sub-optimal references dictated by the primary control. For some applications this is acceptable, but in this case 15 min is too long for the secondary control to take action. Therefore, PLC can not be considered a suitable choice for this type of application.

4.4. Traffic shaping. GSM-like communication

The two case studies presented before approach two communication profiles situated at opposite ends in terms of performance. On one hand, a 100Mbit dedicated local ethernet under which the algorithm converges in 52 s and, on the other-hand, PLC-like communication under which the algorithm takes around 15 min to converge.

In this section we evaluate the algorithm under several GSM-like profiles and see how it performs with respect to the previous two scenarios. Table 1 shows the delays in ms added for each communication link in our network. The table contains five profiles. Three of them correspond to HSPA (the underlying data-protocol used in 3G) under low, medium, and high traffic conditions and two of them to LTE (the data-protocol used in 4G) under low and high traffic conditions. The data for the communication profiles was compiled using the approach shown in [51].

We ran the algorithm for each of the 5 additional cases and we reported the time it takes for it to converge. This time can be seen in the last line of Tabel 1, shown in seconds. As can be seen, high traffic severely impacts the performance of HSPA. In this case it takes the controllers ≈25 min to find a solution, which is even worse than the results obtained with PLC.

On the other end of the spectrum, in a lightly loaded LTE scenario, the solution is reached in less than 3 min. Also for the other cases the convergence time is between 3 and 5 min. We consider these time frames to be more than reasonable for our study case, given that at the moment, secondary controllers in electrical grids are activated 15 min after the primary controllers take action.

Table 1
Communication delays used in order to emulate GSM-like traffic. The last line in the table shows the convergence time of the algorithm for each scenario.

Communication Link		HSPA delays [ms]			LTE delays [ms]	
Node A	Node B	Low	Med.	High	Low	High
1	2	7.8	13.4	71.2	6.6	10.8
1	3	19.5	33.4	177.9	16.6	26.9
2	4	23.3	40.1	213.5	19.9	32.3
3	4	3.9	6.7	35.6	3.3	5.4
2	7	38.9	66.8	355.8	33.2	53.9
2	6	23.3	40.1	213.5	19.9	32.3
4	6	3.9	6.7	35.6	3.3	5.4
6	7	11.7	20.0	106.7	10.0	16.2
6	8	3.9	6.7	35.6	3.3	5.4
6	10	11.7	20.0	106.7	10.0	16.2
6	28	7.8	13.4	71.2	6.6	10.8
4	12	11.7	20.0	106.7	10.0	16.2
8	28	23.3	40.1	213.5	19.9	32.3
12	13	27.2	46.7	249.1	23.3	37.7
12	14	46.7	80.1	427.0	39.9	64.6
12	15	27.2	46.7	249.1	23.3	37.7
12	16	35.0	60.1	320.2	29.9	48.5
14	15	85.6	146.9	782.8	73.1	118.5
16	17	31.1	53.4	284.7	26.6	43.1
15	18	42.8	73.4	391.4	36.6	59.2
18	19	23.3	40.1	213.5	19.9	32.3
19	20	11.7	20.0	106.7	10.0	16.2
10	20	35.0	60.1	320.2	29.9	48.5
10	17	11.7	20.0	106.7	10.0	16.2
10	21	11.7	20.0	106.7	10.0	16.2
10	22	27.2	46.7	249.1	23.3	37.7
21	22	3.9	6.7	35.6	3.3	5.4
15	23	38.9	66.8	355.8	33.2	53.9
22	24	46.7	80.1	427.0	39.9	64.6
23	24	50.6	86.8	462.6	43.2	70.0
24	25	73.9	126.9	676.1	63.1	102.3
25	26	97.3	166.9	889.6	83.1	134.6
25	27	42.8	73.4	391.4	36.6	59.2
28	27	11.7	20.0	106.7	10.0	16.2
27	29	85.6	146.9	782.8	73.1	118.5
27	30	124.5	213.7	1138.7	106.3	172.3
29	30	93.4	160.2	854.0	79.8	129.2
	T_{conv} [s]	191.19	305.13	1485.92	167.81	252.89

5. Discussion and conclusions

As mentioned in the introduction, at the moment there are no readily available tools for evaluating the performance and behavior of distributed control systems in real-life scenarios. While the proposed framework extends the available options in this regard, it does have its

shortcomings. In the following paragraphs we discuss its applicability as well as possible bottlenecks and limitations.

The first obvious limitation is imposed by the interface between the controllers and OPAL-RT. As mentioned earlier, the data server introduces a 14 ms delay between the physical signals and the measured signals at the controller level. Even with additional optimization, it is highly unlikely that this delay can be decreased below the ms range for large scale systems. Therefore, this solution cannot be applied for investigating fast primary controllers, such as current or voltage loops.

Another observation that needs to be made is regarding scalability. From this point of view, the proposed solution is rather flexible; adding new controllers or expanding the physical system requires little effort. The main limitation in this case is the computation power of the real-time simulator. However, we expect scenarios similar to the one shown in Section 4 to scale without problems up to a few hundreds of controllers.

At the communication layer, the proposed solution is able to emulate delays, packet losses and corruption, as well as reduced bandwidth, which are all important aspects when evaluating performance of networked controllers. However, at the moment, it does not include a detailed model of the communication system, the used protocols, or the data models. Therefore, it is not suitable for investigating cyber-security aspects or ICT infrastructure related questions. However, this can also be seen as a feature of our system. As shown by the study cases presented in the paper, our approach allows direct experimentation with various communication profiles and quality of service, without needing a detailed model of the entire ICT network and without other additional communication emulation software or hardware.

From the perspective of potential use-cases, the proposed framework was designed for evaluating applications that involve swarms of controllers involved in secondary control tasks such as energy management, online optimization, state estimation, congestion management, coordination of virtual power plants, etc. Use cases typical to agent based systems, such as plug and play or self-reconfiguration can easily be implemented. One can obtain a real assessment of the behavior of the algorithms, as well as minimum requirements in terms of QoS in order to achieve a certain imposed performance.

The proposed approach extends the way in which real-time simulators are used in the validation of smart-grid applications, i.e., from the already standard approaches of PHIL and CIL to fleet of distributed controllers in the loop. Since both the ICT network and the physical system operate in real-time, our approach avoids all the overhead related to synchronization that a simulation or a co-simulation approach would require.

References

[1] Kraning M, Chu E, Lavaei J, Boyd S. Dynamic network energy management via proximal message passing. Found Trends Optim 2014;1(2):70–122. https://doi.org/10.1561/2400000002.

[2] Meng L, Dragicevic T, Guerrero JM, Vasquez JC. Dynamic consensus algorithm based distributed global efficiency optimization of a droop controlled DC microgrid. 2014 IEEE international energy conference (ENERGYCON) 2014. p. 1276–83. https://doi.org/10.1109/ENERGYCON.2014.6850587.

[3] Nasirian V, Moayedi S, Davoudi A, Lewis FL. Distributed cooperative control of DC microgrids. IEEE Trans Power Electron 2015;30(4):2288–303. https://doi.org/10.1109/TPEL.2014.2324579.

[4] Shafiee Q, Guerrero JM, Vasquez JC. Distributed secondary control for islanded microgrids - a novel approach. IEEE Trans Power Electron 2014;29(2):1018–31. https://doi.org/10.1109/TPEL.2013.2259506.

[5] Sun Q, Han R, Zhang H, Zhou J, Guerrero JM. A multiagent-based consensus algorithm for distributed coordinated control of distributed generators in the energy internet. IEEE Trans Smart Grid 2015;6(6):3006–19. https://doi.org/10.1109/TSG.2015.2412779.

[6] Dorfler F, Simpson-Porco JW, Bullo F. Plug-and-play control and optimization in microgrids. 53rd IEEE conference on decision and control 2014. p. 211–6. https://doi.org/10.1109/CDC.2014.7039383.

[7] Riverso S, Sarzo F, Ferrari-Trecate G. Plug-and-play voltage and frequency control of islanded microgrids with meshed topology. IEEE Trans Smart Grid 2015;6(3):1176–84. https://doi.org/10.1109/TSG.2014.2381093.

[8] Luo F, Chen Y, Xu Z, Liang G, Zheng Y, Qiu J. Multi-agent based cooperative control framework for microgrids' energy imbalance. IEEE Trans Indust Inform 2016;PP (99):1. https://doi.org/10.1109/TII.2016.2591918.

[9] Meng L, Dragicevic T, Roldan-Perez J, Vasquez JC, Guerrero JM. Modeling and sensitivity study of consensus algorithm-based distributed hierarchical control for dc microgrids. IEEE Trans Smart Grid 2016;7(3):1504–15. https://doi.org/10.1109/TSG.2015.2422714.

[10] Zhang Y, Hong M, Dall'Anese E, Dhople S, Xu Z. Distributed controllers seeking AC optimal power flow solutions using ADMM. IEEE Trans Smart Grid 2017;PP(99):1. https://doi.org/10.1109/TSG.2017.2662639.

[11] Vinot B, Cadoux F, Héliot R. Decentralized optimization of energy exchanges in an electricity microgrid. In: 2016 IEEE PES innovative smart grid technologies conference Europe (ISGT-Europe); 2016. p. 1–6. https://doi.org/10.1109/ISGTEurope.2016.7856319.

[12] Nunna HSVSK, Srinivasan D. Multi-agent based transactive energy framework for distribution systems with smart microgrids. IEEE Trans Indust Inform 2017;PP (99):1. https://doi.org/10.1109/TII.2017.2679808.

[13] Müller SC, Häger U, Rehtanz C. A multiagent system for adaptive power flow control in electrical transmission systems. IEEE Trans Indust Inform 2014;10(4):2290–9. https://doi.org/10.1109/TII.2014.2315499.

[14] Georg H, Müller SC, Rehtanz C, Wietfeld C. Analyzing cyber-physical energy systems: the INSPIRE cosimulation of power and ICT systems using HLA. IEEE Trans Indust Inform 2014;10(4):2364–73. https://doi.org/10.1109/TII.2014.2332097.

[15] Adhikari U, Morris T, Pan S. WAMS cyber-physical test bed for power system, cybersecurity study, and data mining. IEEE Trans Smart Grid 2016;PP(99):1. https://doi.org/10.1109/TSG.2016.2537210.

[16] Schutte S, Scherfke S, Troschel M. Mosaik: a framework for modular simulation of active components in smart grids. 2011 IEEE first international workshop on smart grid modeling and simulation (SGMS) 2011. p. 55–60. https://doi.org/10.1109/SGMS.2011.6089027.

[17] Offis mosaik. <https://mosaik.offis.de/> [accessed: 2018-11-09].

[18] Faschang M, Kupzog F, Mosshammer R, Einfalt A. Rapid control prototyping platform for networked smart grid systems. IECON 2013-39th annual conference of the IEEE industrial electronics society 2013. p. 8172–6. https://doi.org/10.1109/IECON.2013.6700500.

[19] Ait lablink. <https://www.ait.ac.at/en/research-fields/smart-grids/network-operators-and-energy-service-providers/ait-lablink/> [accessed: 2018-11-09].

[20] Vogt M, Marten F, Lower L, Horst D, Brauns K, Fetzer D, et al. Evaluation of interactions between multiple grid operators based on sparse grid knowledge in context of a smart grid co-simulation environment. 2015 IEEE Eindhoven PowerTech 2015. p. 1–6. https://doi.org/10.1109/PTC.2015.7232781.

[21] Fraunhofer iee opsim. <https://www.iee.fraunhofer.de/en/schnelleinstieg-wirtschaft/themen/opsim-homepage.html> [accessed: 2018-11-09].

[22] Vogel S, Mirz M, Razik L, Monti A. An open solution for next-generation real-time power system simulation. 2017 IEEE conference on energy internet and energy system integration, (EI2) 2017. p. 1–6. https://doi.org/10.1109/EI2.2017.8245739.

[23] Villas modular co-simulation framework. <https://villas.fein-aachen.org/doc/index.html> [accessed: 2018-11-09].

[24] Stevic M, Estebsari A, Vogel S, Pons E, Bompard E, Masera M, et al. Multi-site european framework for real-time co-simulation of power systems. IET Gener, Transmiss Distrib 2017;11(17):4126–35. https://doi.org/10.1049/iet-gtd.2016.1576.

[25] Monti A, Stevic M, Vogel S, Doncker RWD, Bompard E, Estebsari A, et al. A global real-time superlab: enabling high penetration of power electronics in the electric grid. IEEE Power Electron Magaz 2018;5(3):35–44. https://doi.org/10.1109/MPEL.2018.2850698.

[26] Bottaccioli L, Estebsari A, Pons E, Bompard E, Macii E, Patti E, et al. A flexible distributed infrastructure for real-time co-simulations in smart grids. IEEE Trans Indust Inform 2017;PP(99):1. https://doi.org/10.1109/TII.2017.2702206.

[27] Palensky P, van der Meer A, Lopez C, Joseph A, Pan K. Applied cosimulation of intelligent power systems: implementing hybrid simulators for complex power systems. IEEE Indust Electron Magaz 2017;11(2):6–21. https://doi.org/10.1109/MIE.2017.2671198.

[28] Gungor VC, Sahin D, Kocak T, Ergut S, Buccella C, Cecati C, et al. A survey on smart grid potential applications and communication requirements. IEEE Trans Indust Inform 2013;9(1):28–42. https://doi.org/10.1109/TII.2012.2218253.

[29] XMPP, Uses of XMPP; 2017. <https://xmpp.org/uses/>.

[30] Saint-Andre P, Smith K, Tronçon R, Troncon R. XMPP: the definitive guide. O'Reilly Media Inc; 2009.

[31] Steffen F, Rainer F, Henry D, Thierry D. Decentralized energy in the smart energy grid and smart market - how to master reliable and secure control. Int J Adv Intell Syst, vol. 9, 1–2.

[32] Veichtlbauer A, Parfant M, Langthaler O, Andrén FP, Strasser T. Evaluating XMPP Communication in IEC 61499-based distributed energy applications arXiv:1705.05367, https://doi.org/10.1109/ETFA.2016.7733744. http://arxiv.org/abs/1705.05367.

[33] Lampkin V, Leong WT, Olivera L, Rawat S, Subrahmanyam N, Xiang R, et al. Building smarter planet solutions with mqtt and ibm websphere mq telemetry. IBM Redbooks 2012.

[34] Pai V. gRPC design and implementation; 2016. <http://platformlab.stanford.edu/SeminarTalks/gRPC.pdf>.

[35] Apache/Thrift, Thrift network stack. <https://thrift.apache.org/docs/concepts>.

[36] TILab. JAVA Agent DEvelopment Framework; 2000. <http://jade.tilab.com>.

[37] Kuntschke R, Specht M, van Amelsvoort M, Wagler M, Winter M, Witzmann R. Economic optimization in virtual power plants vs. stable grid operation–bridging the gap. In: 2015 IEEE 20th conference on emerging technologies & factory

automation; 2015. p. 1–5. https://doi.org/10.1109/ETFA.2015.7301567.
[38] Pedersen AB, Hauksson EB, Andersen PB, Poulsen B, Traeholt C, Gantenbein D. Facilitating a generic communication interface to distributed energy resources: mapping IEC 61850 to RESTful Services. 2010 First IEEE international conference on smart grid communications 2010. p. 61–6. https://doi.org/10.1109/SMARTGRID.2010.5622020.
[39] Zillgith M. Open source libraries for IEC 61850 and IEC 60870-5-104; 2017. <http://libiec61850.com/>.
[40] Fraunhofer-ISE, OpenIEC61850; 2017. <https://www.openmuc.org/iec-61850/>.
[41] Scalable clusters make hpc r&d easy as raspberry pi. <https://www.lanl.gov/discover/news-release-archive/2017/November/1113-raspberry-pi.php> [accessed: 2018-11-09].
[42] Omnet++ discrete event simulator. <https://omnetpp.org/> [accessed: 2018-06-13].
[43] Gns3 – the software that empowers network professionals. <https://www.gns3.com/> [accessed: 2018-06-13].
[44] Common open research emulator (core). <https://www.nrl.navy.mil/itd/ncs/products/core> [accessed: 2018-06-13].
[45] Netem. <https://wiki.linuxfoundation.org/networking/netem> [accessed: 2018-06-13].
[46] Introduction to linux traffic control. <http://tldp.org/HOWTO/Traffic-Control-HOWTO/intro.html> [accessed: 2018-06-13].
[47] Facebook, Augmented traffic control: a tool to simulate network conditions. <https://github.com/facebook/augmented-traffic-control>.
[48] Gavriluta C, CAIRE R, Gomez-Exposito A, Hadjsaid N. A distributed approach for OPF-based secondary control of MTDC systems. IEEE Trans Smart Grid 2016;PP (99):1. https://doi.org/10.1109/TSG.2016.2621775.
[49] Johnson SG. The NLopt nonlinear-optimization package. <http://ab-initio.mit.edu/nlopt>.
[50] Powell MJD. A direct search optimization method that models the objective and constraint functions by linear interpolation. Advances in optimization and numerical analysis. Springer; 1994. p. 51–67.
[51] Domenico AD, Gavriluta C, Mendil M, Heiries V, Caire R, Hadjsaid N. Communication network assessment for distributed smart grid applications. 2017 XXXIInd general assembly and scientific symposium of the international Union of Radio Science (URSI GASS) 2017. p. 1–4. https://doi.org/10.23919/URSIGASS.2017.8104962.