Choreography-based Development of Reliable Distributed Control Systems in the Energy Sector

Hugo A. López
Department of Applied Mathematics and Computer Science
Technical University of Denmark
Kongens Lyngby, Denmark

Kai Heussen
Department of Electrical Engineering
Technical University of Denmark
Kongens Lyngby, Denmark

ABSTRACT

Energy Systems are facing a significant change in the way their management and control is conceived. With the introduction of distributed and renewable energy based resources a shift to a more distributed operation paradigm is emerging, overturning the conventional top-down design and operation principles. This shift creates a demand for distributed and transactive control systems to facilitate a more adaptive and efficient power networks. One key challenge here is to ensure the required reliability of distributed control systems. By focusing on the communication aspect required for the coordination of distributed resources we build on the notion of *Quality Choreographies*, a formal model for the development of failure-aware distributed systems. In this paper we discuss how quality choreographies respond to the needs presented by distributed and transactive control systems. We demonstrate their applicability by modelling the Bully Algorithm, one of the de-facto election algorithms used in the coordination of Distributed Control Systems.

Categories and Subject Descriptors

D.2.4 [**Software/Program Verification**]; D.3.2 [**Language Classifications**]: Concurrent, distributed and parallel languages; B.8.1 [**Reliability, Testing, and Fault-Tolerance**]

General Terms

Languages, Verification

Keywords

Choreographic Programming, Failure-awareness, Session Types, Programming Languages, Concurrency Theory

1. INTRODUCTION

The operation and control of energy infrastructures such as electric power systems is becoming increasingly dependent on information technology. With the "Smart Grid" development a cost effective system operation and the integration of renewable energy sources is facilitated, often by communication-centred and transactive applications of distributed control systems. Due to their better performance and adaptability, these are aimed at complementing and replacing existing centralised and decentralised systems. However, electric power networks are critical infrastructures with a long tradition in ensuring reliable operation in part by avoiding reliance on communication in the design of control systems. The deployment of communication-centred approaches is therefore facing adoption barriers: distributed communication-centred solutions can only be cost-effective if evidence of reliability, availability, and security of the underlying communications can be given. Similar conditions apply in other energy infrastructure systems such as heat-supply networks or gas pipelines. In more general terms, Cyber-Physical Systems (CPS) are heterogeneous systems where *availability* and *reliability* requirements are crucial. Ensuring available and reliable CPS is hard, since in many cases the systems operate in adverse conditions (e.g.: with energy sources located in difficult access conditions), or are critical for operation, such that correction at runtime is not possible. Furthermore, the development of coordination algorithms for distributed control systems (herewith DCS) is error-prone: each component develops its own part of the code, small changes in the order of communications implemented in a component might lead the system to a deadlock situation, or to executions that deviate from the intended ones; finally, each component must be tested against the whole infrastructure. Apart from the costs incurred in this approach, testing cannot not guarantee the absence of errors.

One key challenge in the communications aspect of such Cyber-Physical Distributed Control Systems (CP-DCS), is to ensure that CP-DCS comply with the appropriate application-specific quality of service requirements and reliability constraints prior to deployment. Such constraints include that communication among components do not block (deadlock-freedom), or that implementations adhere to a specific protocol (protocol-fidelity). In this paper we report how this challenge from a new generation of electrical distributed control systems can be mitigated by using techniques coming from formal modelling of distributed systems. In particular, this article presents how *choreographic programming* may serve as a methodology for the development of correct-by-design distributed control systems.

Our Contribution. This paper reports a multidisciplinary effort between electrical engineering and formal models of software development. First, the application and structure of distributed control systems in Smart Grids is discussed and a recently published taxonomy is reviewed. The development of programming-language support techniques is motivated from the needs of distributed control systems. We explore how a variant of choreographic programming and the theory of endpoint projections [8] can map to the requirements present in CP-DCS. As a case study, we present how the selected variant of choreographic programming (Quality Choreographies [29]), can provide support for the specification of the Bully leader election algorithm [13], which is used as coordination algorithm in CP-DCS. This paper is organised as follows: In Section 2 the requirements and different varieties of CP-DCS are presented. Section 3 introduces quality choreographies, as well as the development methodology that permits the transformation from choreographies to failure-aware distributed components. Section 4 discusses how the choreographies capture the requirements of CP-DCS and illustrates the approach presenting a choreographic specification of the Bully algorithm. Finally, Section 5 concludes.

2. DISTRIBUTED CONTROL SYSTEMS IN SMART GRIDS

Conventional power systems utilise a combination of hierarchically layered centralised, communication-centred, and decentralised, non-communication-centred, control strategies [26]. These centralised and decentralised control strategies have complementary properties [14], which together meet the requirements of conventional power system operation. However, distributed control systems are increasingly proven to offer utility beyond these traditional control concepts [5]. With the increasing availability of distributed energy resources (DER), which have the ability to change roles as providers of power system services and energy, CP-DCS architectures which are scalable, composable, and transactive are becoming unavoidable [14]. In this view, *transactive energy* [12] is an increasingly adopted paradigm enabling distributed energy trading and system management that involves DER as active participants in grid management. *Transactive control* [20, 23] refers to the extension of transactive energy to the management of real-time control systems.

A taxonomy of control architecture patterns presented in [14], defines the following major categories:

- *Centralised.* One central control element collecting information from remote sites and deciding set-points for remote actuation.
- *(Dec)entralised.* A central (common) control objective is decentralised to independent local control elements; the local control elements only use local measurements and actuators.
- *(D)istributed.* Multiple control elements organised in a common architecture jointly responsible decomposing objectives and deciding actuation.

The main distinctive element between these categories is in their allocation of control responsibility (central, local, shared), which implies that successful control will depend in different degrees on the availability of communication links among control elements. It is possible to combine architectural patterns as separate *Control Layers*, which are functional levels in a layered control architecture, representing an aspect of the overall control problem at a particular time scale or abstraction level of the physical system. The driver for layering different control patterns is typically the need to meet several design criteria that cannot be achieved in a desirable performance by a single integrated algorithm. The layering is thereby not necessarily driven by a hierarchical formulation of the control problem, but may be simply addressing another aspect of the control problem, thus meeting other design constraints.

Reliable, communication-independent, operation is achieved in conventional centralised/decentralised control architecture, by means of a decentralised control strategy, i.e. autonomous local control elements with a clearly defined control behaviour. Centralised control system reliability is then typically improved by redundancy of control centres, communication and computational infrastructure. Instead of active coordination, conflict of control actions between centralised and decentralised control layers is avoided by means of a model-based decomposition of control objectives [17]. Distributed architecture offers the advantage of flexibility in meeting design objectives, considering trade-offs, such as privacy vs. fairness, or optimality vs. scalability. One of the trade-offs to be considered is the increased dependency on communication that comes with distribution. The communication patterns of CP-DCS are diverse which is also expressed in the further refined categories of [14]:
D-(V)ertical. Decisions of one control element are imposed on other control element, so that a hierarchy among the control elements exists.

- *D-V-(D)eterminate*: control decisions are based on hierarchical information collection and central decision-making.
- *D-V-(I)terative*: control decisions are in an iterative negotiation process facilitated by a central clearing instance; this central instance is responsible for concluding the negotiation and committing the control decisions.

D-(H)orizontal. Different from *D-V*, the responsibilities of control elements in this category are symmetrical, and the functions being executed in the control elements are similar.

- *D-H-(C)entralized shared processing*, a single entity is required to relay information among control elements as an essential part of the control algorithm; it does however not commit participants to a control decision; examples are publish-aggregate-subscribe patterns or blackboard architectures.
- *D-H-(P)2P* represents fully distributed Peer-to-peer (P2P) control architecture; here the roles of control elements are not fixed and may be re-allocated based on capabilities.

This range of coordination and implied communication requirements leads to a rather indirect relation between communication link reliability and control functionality: whereas a D-V-D pattern can be expected to have a similar response to communication failure as a centralised pattern, in the D-H-P case communication failure will affect system reliability at a comparable rate as component failure in a decentralised pattern. The commonly assumed direct linkage between communication failure and function failure thus does not apply here. Yet, reliability dependency due to indirect linkage is not typically qualified for smart grid applications [39, 38].

While formal methods have been applied to safety critical systems, such as railway control systems [16], they are not commonly emplyed in distributed control. Modeling and analysis wrt. networked and distributed control systems entails assessment of network properties [1, 38]. An applicable automation standard, IEC61499 [36] includes provisions for code generation and testing based on function block specifications. However, formal methods are hardly used as part of distributed control systems specifications in energy networks. Although terminology from different disciplines is likely to overlap and conflict, we recognise that the above categorisation is strongly linked to the communication patterns required within the respective control layer. In the remainder of this contribution we aim to exploit this linkage by focussing on providing reliable communications for distributed control systems.

3. PROGRAMMING-LANGUAGE SUPPORT FOR RELIABLE DCS

We advocate for a methodology for the rigorous development of distributed control systems. In particular, we propose choreographies [9, 10, 29] as a programming-language methodology applicable for DCS. Choreographies are a high-level programming model that describes the communication behaviour of a distributed system. They can simplify the programmer's life by providing a way to describe a unique description of the sequence of interactions required in the systems from a birds-eye perspective. Moreover, choreographies are amenable to automated (static) verification, with the advantage that verified choreographies will compile to reliable code implementing the behaviour of each component in the distributed system.

3.1 Programming Language Requirements

Since their inception, a vast number of choreographical languages have been proposed. The selection of which model to use will depend on specific characteristics of the application domain. In the case of Distributed Control Systems, we can rely on a language that the communication behaviour of Cyber-Physical Systems [29]:

- **Asynchrony.** Control elements may be deployed in harsh environmental conditions, such as arctic or marine locations. Such conditions affect elements' lifespan, increasing the probability of failure. An *asynchronous control* model permits to cope with elements' changing availability conditions.

- **Self-Configuration.** Changing availability conditions mean that control units cannot fully rely on a centralised infrastructure. Instead, each control element must be able to adapt according to availability changes, including the change of roles and capabilities at runtime.

- **Application-Centric Protocols.** Control elements aim at accomplishing a common, universal goal, typically related to maintaining an application-level quality of service (QoS). As a result, the importance of a single control element is superseded by the fairness of the collection of elements involved, where decisions are made from the aggregate data of control elements, rather than the specific data coming from any of them [35].

- **Multicast Communication.** One-to-many and many-to-one communications rather than P2P message passing.

All of these characteristics also pertain to the CP-DCS applications discussed in Section 2. In particular, the desired reliability characteristics correspond to application-centric QoS requirements.

3.2 Choreographies

Works on verification of interaction protocols date back to the works on sortings for the polyadic π-calculus [32], and later in [37, 18] (see [11] for a thorough overview). Here, interaction is seen at a local level, featuring constructs for message passing, labelled selection and delegation of responsibilities. A type discipline (*session types* [18]) guarantees that components are engaged in interactions in a *structured* and complementary way. That is, if the specification of a protocol involves components sending and receiving information, then the endpoints generated have a balance between their input/output primitives, and their concurrent execution behaves according to the specific order of interactions in the specification. In [7, 19], correspondences between *global types* (describing interactions at a global *–choreographical–* level) and *local types* (describing interactions of endpoints) have been studied. The *Endpoint Projection* ensures that all and only the interaction behaviour described by the global types is present in their endpoints. Choreographies for CPSs require the combination of models of asynchronous, multiparty and multiparty communications [10], as well as collective (many-to-one, one-to-many) communications [4, 25, 28].

3.3 Generating Reliable Code

A methodology based on choreographies follows a top-down approach. It can be summarised as follows:

1. Describe the communicating behaviour of a Distributed Control System in terms of a Choreography.

2. Verify that the produced choreography is well-typed with respect to a session type.

3. Generate endpoint code for all involved participants.

In essence, the first step provides a skeleton of the communicating behaviour of components in the system, similarly to other specification models, such as UML's interaction diagrams, Message Sequence Charts or WS-CDL specifications [15, 22]. Considering the requirements placed in Section 3.1, we have opted for Quality Choreographies [29] as the selected choreographic language. Among others, quality choreographies model multiparty, asynchronous communications, where component-based reliability is relaxed, modelling the fact that some components can be unavailable at runtime. Even at this early stage, choreographies can benefit from static verification techniques. For instance, failure-aware considerations used in the description of a choreography might not be able to be satisfied in collective communications, leading to choreographies that, if implemented, will deadlock. A type discipline will guarantee that the communication primitives used in a choreographic specification can indeed be implemented with available components.

Once generated a choreographic model, a transformation from top-level specifications to abstract endpoints must be performed. The theory of Multiparty Session Types (MPST) [19] provides a formal ground to this transformation. In particular, the work in [31] presents how specifications of quality choreographies paired with (global) multiparty session types, can project to abstract endpoint processes. Pairing Quality Choreographies with MPST is of vital importance in order to generate endpoints that preserve the behaviour described at top level. Reliability guarantees are expressed in terms on the behaviours that implementations generated from a choreography can exhibit. Also known as safety, reliability says that all the execution paths of a CP-DCS preserve a safe ("good") behaviour in the system. We list a non-exhaustive list of variants of safety that apply to CP-DCS:

- *Progress* (or deadlock-freedom): The components in a CP-DCS do not get stuck due to errors in the communication protocol [10].

- *Liveness*: Each interaction described in the protocol eventually occurs [34].

- *Protocol-fidelity:* Once the choreography is projected onto endpoints, the distributed execution of CP-DCS components will describe all and only the behaviours described by the original choreography [18].

- *Availability-by-design:* It describes failure-awareness conditions in a protocol. Under a minimum set of available components, the protocol is guaranteed to advance in its execution [29]. This property subsumes deadlock-freedom, in the sense that, while deadlock-freedom is a *component-based* property that can be invalidated if one of the components stops working, availability-by-design is a *system-based* property, invalidated if not enough components can provide the global availability guarantees required in the protocol.

- *Fault-Tolerance:* In the event that not enough replicated components are available, the execution of a distributed system can preserve its execution by reconfiguring components according to new conditions.

While quality choreographies provide ways of describing failure-aware protocols, guaranteeing availability-by-design, they do not bring any kind of support when the minimum threshold of available components is unreachable. As it is the case for many coordination protocols for CP-DCS (including the Bully algorithm presented in Section 4.2), the correct operation of a distributed system relies on the assumption that such a minimum set is always available. Fault tolerance for choreographies will imply that once we cannot contact the minimum amount of required available components, an exceptional protocol will take charge of reconfiguring the system in such a way operational constraints are satisfied. To achieve this, the language Quality Choreographies may require additional extensions, among them Exceptional/Compensating behaviour [6], 2. Timeouts [3, 30], and 3. Runtime Monitoring [2]. We will leave considerations regarding extensions from failure awareness to fault tolerance for future work.

4. CHOREOGRAPHING CP-DCS

We derive the application specific requirements that facilitate specification of reliable CP-DCS and illustrate some necessary choreography language extensions on a case study.

4.1 Reliability of Distributed Control Systems

In physical infrastructures, the notion of reliability pertains the physical functional requirements of electrical components, including (local, decentralised) control systems. System reliability further includes the operational principle of *secure operation*, which refers to the ability to steer the system operating state within a distance from undesirable operating states [27, 21]. Maintaining secure operation typically is associated with centralised control, for which also availability of communication functions is relevant, but characterised simply by availability of individual communication links.

In a distributed control system, reliability requires a combination of both the physical availability of components, as well as the availability for interactions via a communication network. In this case, guaranteeing communication among components becomes a functional requirement, as the overall objective will be fulfilled partially by each of the components of the distributed control system. The interpretation of reliability of distributed control systems is therefore contingent on the interpretation of "reliability" of its communication patterns, rather than individual communication links. From a CP-DCS point of view, the ability that a choreographic approach has at providing guarantees for certain communication-dependent systems properties is of utmost interest.

It is possible to draw a parallel between CP-DCS and Choreographies. The concept of *control elements* directly corrresponds to *participants* in a choreography. Moreover, the separation of concerns achieved by *control layers* can be mapped directly to the session scoping provided in choreographies. Communication aspects of CP-DCS as categorised in [14], can be modeled using by multiparty choreographies, considering multiple control layers being executed at once. Other aspects of the categorisation appear to be orthogonal to choreography concepts, such as the domain-specific type of *control objectives* and execution context, as well as properties of the respective distributed control algorithm. While properties such as *progress*, *liveness*, and *protocol-fidelity* can be considered basic functional requirements of distributed communication, the properties *availability-by-design*, and *fault-tolerance* directly have

```
start $t[\widetilde{Client}] : a(k)$;
rec $Y = (\mathsf{t}_i$ -> $\&^{\forall}(\vec{\mathsf{t}}) : k[cmd]; Y)$ in
        $Y+$
        $+_{\mathsf{t}_i \in \widetilde{\mathsf{t}}}\ \mathsf{t}_i$ -> $\&^{m/n}(\vec{\mathsf{t}} \backslash \mathsf{t}_i) : k[election]$;

        $\&^{n/m}(\mathsf{t}_1.\mathsf{int}(\mathsf{t}_1), \ldots, \mathsf{t}_{i-1}.\mathsf{int}(\mathsf{t}_{i-1}), \mathsf{t}_{i+1}.\mathsf{int}(\mathsf{t}_{i+1}),$
            $\ldots, \mathsf{t}_n.\mathsf{int}(\mathsf{t}_n))$ -> $\mathsf{t}_i : x : \langle k, max\rangle$;
        if$(x@\mathsf{t}_i < \mathsf{int}(\mathsf{t}_i))\{$
          $\mathsf{t}_i$ -> $\&^{m/n}(\vec{\mathsf{t}}) : k[halt]$;
          foreach $\mathsf{t}_j$ in $\vec{\mathsf{t}}$ do $\{$

            $\mathsf{t}_j$ -> $\&^{m/n}(\bigcup_{\mathsf{t}_k \in \vec{\mathsf{t}} \backslash \mathsf{t}_j \cup \mathsf{t}_i} \mathsf{t}_k) : k[election]$;

            $\&^{m/n}(\bigcup_{\mathsf{t}_k \in \vec{\mathsf{t}} \backslash \mathsf{t}_j \cup \mathsf{t}_i} \mathsf{t}_k.e_k)$ -> $\mathsf{t}_j : y : \langle k, max\rangle$
            $\}$
          $\&^{m/n}(\bigcup_{\mathsf{t}_j \in \vec{\mathsf{t}}} \mathsf{t}_j.y)$ -> $\mathsf{t}_i : z : \langle k, id\rangle$;
          if$(z@\mathsf{t}_i = \mathsf{int}(\mathsf{t}_i))\{$
            $\mathsf{t}_i$ -> $\&^{m/n}(\vec{\mathsf{t}}) : k[Reorganize]$;
            $\mathsf{t}_i.c$ -> $\&^{m/n}(\vec{\mathsf{t}} : c) : k$
            $\}$
          $Y\}$
```

Figure 1: Choreographic description of the Bully algorithm. $\vec{\mathsf{t}}$ is the shorthand notation for $\widetilde{\mathsf{t}} \backslash \mathsf{t}_i$

an effect on reliability. These safety properties will affect each of the CP-DCS taxonomy categories introduced in Section 2 in a different way. For example, for distributed control and of the *D-V* group (determinate / iterative), communication failure of endpoints will lead to proportional degradation - a *failure-aware* choreography would ensure that the experienced degradation is equally proportional. For a *D-H-S*, and more importantly, a *D-H-P* pattern, availability by design can be expressed. By means of these properties, a quantification of the respective reliability properties can be mapped from communication link reliability rather directly to control function reliability. Utilizing the choreographic methods will therefore facilitate CP-DCS both the correct-by-construction development methods and the potential for deterministic qualification of CP-DCS reliability properties. Choreography-based programming currently offers the majority of these properties, excepting fault-tolerance.

4.2 Case Study on CP-DCS

In order to exemplify the use of choreographies for CP-DCS, we give a specification of the Bully Algorithm [13]. Bully is a leader election algorithm in distributed systems that has already been used to coordinate the provision of grid services for control elements [24]. Its specification is presented in Figure 4.1, following the syntax and semantics of the Global Quality Calculus, extended with standard general and primitive recursion operators [10, 28].

Example 4.1 (Bully Algorithm). The Bully algorithm coordinates the election of a component as a coordinator in a distributed system. The set of threads is denoted by $\widetilde{\mathtt{t}}$, whose elements are ranked using a unique identification number. The idea is that, after the failure of one of more of the components in the distributed system, components will elect the one with the highest identification number. They do so by performing many-to-one communication actions, such as broadcast, reduce, and collective selection.

Lines 1–3 describe a normal master-subordinate relation between the coordinator component and its subordinates. Line 1 creates a new session k, that will identify the communication among threads. Sessions are an essential component in the description of dynamic behaviour of distributed systems. By using them, one can guarantee that interactions among different components in a distributed system do not mix, since they belong to different sessions whose identifiers are generated at runtime. Line 4 presents a *collective choice*: a coordinator $\mathtt{t}_i$ determines the execution of command *cmd* to all its peers. Collective operations (selection, broadcast and aggregation) are parameterised by *availability conditions* of the present nodes [33]. In this particular case, the selection is deemed to proceed as long as all components are available, which is denoted by the $\forall$ predicate in the communication. Lines 5–19 describe the election phase: Any node can start a new election by performing a collective selection to all of its peers. The interaction will deem successful as long as a threshold is maintained (in this case, that m out the total n number of components is available). In this way one can ensure that a minimum amount of components (c.f. the minimum number of threads according to Byzantine failures) respond. Lines 6 models *message aggregation*: all requested components answer with their identification number. The result will be collected by the sender, who will compute an aggregation of data received (in this particular case, it simply computes the the maximum of the values received). If none of the components has a higher identification number than the component eliciting the election, this will assume the position of a new leader. First, he stops the execution of components available (Line 8), forcing them to perform an election among their peers (Lines 9–12). These elections serve as step for synchronising all available components, making them ready to assume a new leader. Finally, in Lines 14–17 the new leader distributes its identity among all available components, assuming him as the new coordinator.

The specification in Figure 4.1 presents interesting aspects for electronic elections. First, it provides a concise specification of the sequence of messages exchanged among all participants involved. Second, novel collective operations (broadcast, reduce, collective selection) allow for the modelling of complex one-to-many and many-to-one communications. Third, success criteria at each interaction is parameterised with availability conditions, allowing flexible specifications where some interactions require all components, and others do not. Fourth, it explicitly isolates the decision process pertaining the coordinating and communicating behaviour of each component, abstracting details regarding the implementation of each component.

5. CONCLUSIONS AND FUTURE WORK

In this work we have presented how the theory of choreographic programming can provide guarantees for the development of reliable distributed control systems from the energy sector. By providing a formal foundation capable of guaranteeing reliable executions in the system, we expect to facilitate the transition from the centralised and decentralised architecture patterns currently used towards distributed ones, therefore providing a ground where transactional energy frameworks can build upon. To the best of our knowledge, this is the first attempt at using a formal account of choreographies and session type techniques in the electrical sector.

This work also motivates further studies on choreographic programming. Moving from failure-awareness towards fault-tolerance seems like a natural step. For this to happen, further explorations on frameworks combining Application-based QoS guarantees (needed for CP-DCS), exceptional behaviour, time and runtime monitoring are required.

6. REFERENCES

[1] J. Baillieul and P. J. Antsaklis. Control and communication challenges in networked real-time systems. *Proceedings of the IEEE*, 95(1):9–28, Jan 2007.

[2] L. Bocchi, T. Chen, R. Demangeon, K. Honda, and N. Yoshida. Monitoring networks through multiparty session types. In *FMOODS/FORTE*, volume 7892 of *LNCS*, pages 50–65. Springer, 2013.

[3] L. Bocchi, W. Yang, and N. Yoshida. Timed multiparty session types. In P. Baldan and D. Gorla, editors, *Procs. of (CONCUR)*, volume 8704 of *LNCS*, pages 419–434. Springer Berlin Heidelberg, 2014.

[4] E. Bonelli and A. Compagnoni. Multipoint session types for a distributed calculus. In *Trustworthy Global Computing*, pages 240–256. Springer, 2008.

[5] A. Borghetti, R. Bottura, M. Barbiroli, and C. A. Nucci. Synchrophasors-based distributed secondary voltage/var control via cellular network. *IEEE Transactions on Smart Grid*, PP(99):1–1, 2016.

[6] M. Carbone. Session-based choreography with exceptions. In N. Yoshida and V. Vasconcelos, editors, *PLACES'08*, volume 241 of *ENTCS*, pages 35–55, 2008.

[7] M. Carbone, K. Honda, and N. Yoshida. Structured communication-centred programming for web services. In *ESOP*, pages 2–17, 2007.

[8] M. Carbone, K. Honda, and N. Yoshida. Structured interactional exceptions in session types. In *Procs. of (CONCUR)*, volume 5201 of *LNCS*, pages 402–417. Springer, 2008.

[9] M. Carbone, K. Honda, N. Yoshida, R. Milner, G. Brown, and S. Ross-Talbot. A theoretical basis of communication-centred concurrent programming. *Web Services Choreography Working Group mailing list, to appear as a WS-CDL working report*, 2006.

[10] M. Carbone and F. Montesi. Deadlock-freedom-by-design: Multiparty asynchronous global programming. In *POPL*, pages 263–274. ACM, 2013.

[11] M. Dezani-Ciancaglini and U. De'Liguoro. Sessions and session types: an overview. In *Proceedings of the 6th international conference on Web services and formal methods*, pages 1–28. Springer-Verlag, 2010.

[12] D. Forfia, M. Knight, and R. Melton. The view from the top of the mountain: Building a community of practice with the gridwise transactive energy framework. *IEEE Power and Energy Magazine*, 14(3):25–33, 2016.

[13] H. Garcia-Molina. Elections in a distributed computing system. *IEEE transactions on Computers*, 100(1):48–59, 1982.

[14] X. Han, K. Heussen, O. Gehrke, H. W. Bindner, and B. Kroposki. Methodology to evaluate distributed control strategies used in distributed energy resource applications. *CoRR*, abs/1603.03232, 2016.

[15] D. Harel and P. Thiagarajan. Message sequence charts. *UML for Real*, pages 77–105, 2004.

[16] A. E. Haxthausen and J. Peleska. Formal development and verification of a distributed railway control system. *IEEE Transactions on Software Engineering*, 26(8):687–701, Aug 2000.

[17] K. Heussen, A. Saleem, and M. Lind. Control architecture of power systems: Modeling of purpose and function. In *2009 IEEE Power Energy Society General Meeting*, pages 1–8, July 2009.

[18] K. Honda, V. T. Vasconcelos, and M. Kubo. Language Primitives and Type Discipline for Structured Communication-Based Programming. In *ESOP*, pages 122–138. Springer, 1998.

[19] K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. *POPL*, 43(1):273–284, 2008.

[20] J. Hu, G. Yang, K. Kok, Y. Xue, and H. W. Bindner. Transactive control: a framework for operating power systems characterized by high penetration of distributed energy resources. *Journal of Modern Power Systems and Clean Energy*, pages 1–14, 2016.

[21] E. Karangelos and L. Wehenkel. Probabilistic reliability management approach and criteria for power system real-time operation. In *Power Systems Computation Conference*, 2016.

[22] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, Y. Lafon, and C. Barreto. Web services choreography description language version 1.0. *W3C Working Draft*, 17:10–20041217, 2004.

[23] J. K. Kok, C. J. Warmer, and I. G. Kamphuis. Powermatcher: multiagent control in the electricity infrastructure. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, AAMAS '05, pages 75–82, New York, NY, USA, 2005. ACM.

[24] A. M. Kosek, O. Gehrke, and D. Kullmann. Fault tolerant aggregation for power system services. In *Intelligent Energy Systems (IWIES), 2013 IEEE International Workshop on*, pages 107–112. IEEE, 2013.

[25] D. Kouzapas, R. Gutkovas, and S. J. Gay. Session types for broadcasting. In *PLACES*, volume 155 of *EPTCS*, pages 25–31, 2014.

[26] P. Kundur. *Power System Stability and Control*. McGraw-Hill, Inc., epri edition, 1994.

[27] P. Kundur, J. Paserba, V. Ajjarapu, G. Andersson, A. Bose, C. Canizares, N. Hatziargyriou, D. Hill, A. Stankovic, C. Taylor, T. Van Cutsem, and V. Vittal. Definition and classification of power system stability ieee/cigre joint task force on stability terms and definitions. *Power Systems, IEEE Transactions on*, 19(3):1387 – 1401, Aug. 2004.

[28] H. A. López, E. R. B. Marques, F. Martins, N. Ng, C. Santos, V. T. Vasconcelos, and N. Yoshida. Protocol-based verification of message-passing parallel programs. In *OOPSLA*, pages 280–298. ACM, 2015.

[29] H. A. López, F. Nielson, and H. R. Nielson. Enforcing availability in failure-aware communicating systems. In E. Albert and I. Lanese, editors, *FORTE*, volume 9688 of *LNCS*, pages 195–211. Springer, 2016.

[30] H. A. López and J. A. Pérez. Time and Exceptional Behavior in Multiparty Structured Communications. In *WS-FM*, volume 7176 of *LNCS*, pages 48–63. Springer, 2012.

[31] H. A. López, F. Nielson, and H. R. Nielson. A Theory of Available-by-Design Communicating Systems. *CoRR*, abs/1611.05651, Nov. 2016.

[32] R. Milner. *Communicating and Mobile systems. The Pi Calculus*. Cambridge University Press, 1999.

[33] H. R. Nielson, F. Nielson, and R. Vigo. A calculus for quality. In *FACS*, pages 188–204. Springer, 2013.

[34] L. Padovani, V. T. Vasconcelos, and H. T. Vieira. *Typing Liveness in Multiparty Communicating Systems*, pages 147–162. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[35] S. Pattem, B. Krishnamachari, and R. Govindan. The impact of spatial correlation on routing with compression in wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 4(4):24, 2008.

[36] T. Strasser, F. Andrén, J. Kathan, C. Cecati, C. Buccella, P. Siano, P. Leitão, G. Zhabelova, V. Vyatkin, P. Vrba, and V. Mařík. A review of architectures and concepts for intelligence in future electric energy systems. *IEEE Transactions on Industrial Electronics*, 62(4):2424–2438, April 2015.

[37] K. Takeuchi, K. Honda, and M. Kubo. An interaction-based language and its typing system. In C. Halatsis, D. G. Maritsas, G. Philokyprou, and S. Theodoridis, editors, *PARLE*, volume 817 of *LNCS*, pages 398–413. Springer, 1994.

[38] Z. Wang, A. Scaglione, and R. J. Thomas. Generating statistically correct random topologies for testing smart grid communication and control networks. *IEEE Transactions on Smart Grid*, 1(1):28–39, June 2010.

[39] Y. Yan, Y. Qian, H. Sharif, and D. Tipper. A survey on smart grid communication infrastructures: Motivations, requirements and challenges. *IEEE Communications Surveys Tutorials*, 15(1):5–20, First 2013.