Fault tolerant aggregation for power system services

Kosek, Anna Magdalena; Gehrke, Oliver; Kullmann, Daniel

Published in:
Proceedings of IWIES 2013

Link to article, DOI:
10.1109/IWIES.2013.6698570

Publication date:
2013

Fault tolerant aggregation for power system services

Anna Magdalena Kosek*, Oliver Gehrke* and Daniel Kullmann*
*Technical University of Denmark, Department of Electrical Engineering
Energy Systems Operations and Management
Frederiksborgvej 399, Building 776, 4000 Roskilde, Denmark
Email: {amko,olge,daku}@elektro.dtu.dk

***Abstract*—Exploiting the flexibility in distributed energy resources (DER) is seen as an important contribution to allow high penetrations of renewable generation in electrical power systems. However, the present control infrastructure in power systems is not well suited for the integration of a very large number of small units. A common approach is to aggregate a portfolio of such units together and expose them to the power system as a single large virtual unit.**

In order to realize the vision of a Smart Grid, concepts for flexible, resilient and reliable aggregation infrastructures are required. This paper presents such a concept while focusing on the aspect of resilience and fault tolerance. The proposed concept makes use of a multi-level election algorithm to transparently manage the addition, removal, failure and reorganization of units. It has been implemented and tested as a proof-of-concept on the distributed smart grid test bed SYSLAB at the Technical University of Denmark.

I. Introduction

Power grids in many countries are facing the challenge of integrating an increasing amount of fluctuating power production from renewable resources. This creates a need for additional flexibility in the operation of the grid; in particular, the provision of power system services from distributed energy resources (DER) has been a strong focus of smart grid research in recent years. A significant part of the potential for this additional flexibility lies in small DER units, including demand-side units at the household level. In order to provide a useful contribution to system services, the response of many of these small units will have to be coordinated. This calls for communication and control solutions which are able to scale to a large number of units. A common approach to achieve this is the use of a hierarchy of aggregators which combine a portfolio of controllable units into a single representation at one or multiple levels. A well-known example of a control concept using aggregation is that of virtual power plants (VPP) [1].

Many DER units allow their operating pattern to be modified without compromising their primary function; this flexibility may be used to support the operation of the grid. The range of services to be provided depends on the capabilities of the unit and include frequency control services such as primary frequency response or contribution to regulating power, voltage support, peak shaving, power quality control or MVAr control at the feeder level. Aggregation may offer services to different entities in the power system, including transmission system operators (TSO), distribution system operators (DSO) or balancing responsible parties (BRP), either through bilateral contracts or as part of a market-based setup. Aggregation schemes offering direct service provision can be found in several projects such as PowerHub [2] or FENIX [1]. Aggregation in connection with service markets is e.g. proposed by the ADDRESS project [3]. One of the main weaknesses of these existing schemes is their static hierarchy; if one of the key nodes in the system fails, the aggregator may fail as a whole.

Reliability and resilience are key requirements for aggregation systems; power system services have both monetary value and an impact on system stability. If the aggregators and their communication relations are static and inflexible, single points of failure will exist. A related problem is how to keep track of the aggregated units and maintain stable operation, in an environment where units are added and removed on a regular basis and where units may fail permanently or temporarily. This paper proposes a dynamic aggregation concept which addresses both issues. Unit addition, removal and failure are handled automatically and transparently. The dynamic assignment of concentrators on multiple levels increases the resilience and reliability of the system.

Exploiting the potential for redundancy offered by distributed architectures for systems with high reliability requirements are a subject of ongoing research. One area of application for this type of research is in wireless multi-hop sensor networks. Sensors, particularly environmental sensors outside of buildings, are often battery-constrained, and any data collection system will have to anticipate the failure of sensors and temporary or permanent failure of communication subpaths. A data routing protocol for wireless sensor network is presented in [4], where data collectors, aggregators and sinks are chosen dynamically through leader election. Other related research efforts target reliable multicast message delivery [5], in particular for multimedia streaming applications [6]. In [5], an election algorithm is used to dynamically select a node responsible for maintaining communication between several sub-networks. In [6], dynamic election of a local directory server is demonstrated. However, the authors are not aware of any previous work attempting to directly transfer these concepts in the domain of aggregation in power systems, and succeeding with an implementation.

In section II of this paper we present the considerations behind the design of the proposed system, including the aggregation concept and issues related to fault tolerance. Section III continues with a discussion of a concrete implementation of this concept. Section IV describes setup and results from proof-of-concept testing on a physical smart grid test bed.

II. Design considerations

A. Aggregation concept

In this paper we consider a portfolio of DERs able to provide a specific service by decreasing or increasing their active power production or consumption. These DERs may be grouped into one or more aggregates in order to jointly provide a service. The grouping may be based on location, DER type or other criteria.

Aggregation can be performed using one or multiple stages in order to scale to a larger number of aggregated units. Here, we limit the scope to the special case of a two-level aggregation hierarchy. Each DER unit is being dynamically assigned to one of several **local controllers** (LC) which acts as the first level aggregator for the unit. The LCs are then further aggregated by a single **supervisory controller** (SC). The supervisory controller is responsible for managing the overall service provided by the aggregation system (figure 1).

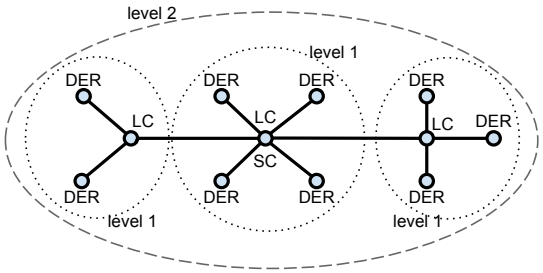

Fig. 1. Proposed layout of a two-level aggregation. Level 2 represents the total aggregation supervised by SC, level 1 aggregations group DERs supervised by LC.

In order to increase the resilience of the system, all DER units possess the capability to act as a LC, as a SC or in both roles simultaneously. In case either the active SC or one of the active LCs fails, one of the remaining units is assigned in its place and the aggregation relations are updated to reflect the new situation. This reassignment must be implemented in a decentralized way; were it to be decided by a central authority, it would remove the extra redundancy gained by the dynamic role assignment and only replace one single point of failure with another one.

B. Fault tolerance

The objective of the dynamic aggregation and fault tolerance mechanisms is to increase the resilience of the aggregation system. This can be achieved by making the system support different types of transparency. Transparency, as defined by Tanenbaum [7], is a characteristic of *"a distributed system that enables to present itself to users and applications as if it were only a single computer system [...]"*. Various types of transparency can be distinguished, among them failure, replication, concurrency, location and mobility transparency. Location transparency allows resources to be accessed without knowledge of their location. Relocation transparency enables moving resources while in use. Replication transparency hides that a resource is replicated. Concurrency transparency ensures that a resource can be shared by several competing users. Failure transparency enables concealment of failures [7].

For the system discussed in this paper, failure transparency is an obvious goal to be achieved. Replication transparency hides the dynamically changing topology of the aggregation hierarchy from the service user. Concurrency may be a desired feature if e.g. services delivered to a BRP aggravate congestion issues in a distribution grid. Location transparency may or may not be desirable, depending on whether the service delivered depends on location, such as e.g. in a voltage control scheme. Mobility transparency is not addressed in this paper but could be relevant in connection to services provided by electric vehicles.

C. System architecture

All DER units are paired with a gateway node which maps the specific properties and capabilities of the unit onto a service-based communication interface and data model. In this way, the gateway separates the details of local fieldbus communication and device-specific control from the global task of aggregating multiple DERs. All gateway nodes collaborate to build and maintain an aggregation hierarchy. These two functions are located in two software modules: A service provision module and a hierarchy builder module (figure 2). The service provision module is responsible for offering different services to the power system. Different DERs may offer different sets of services at different times.

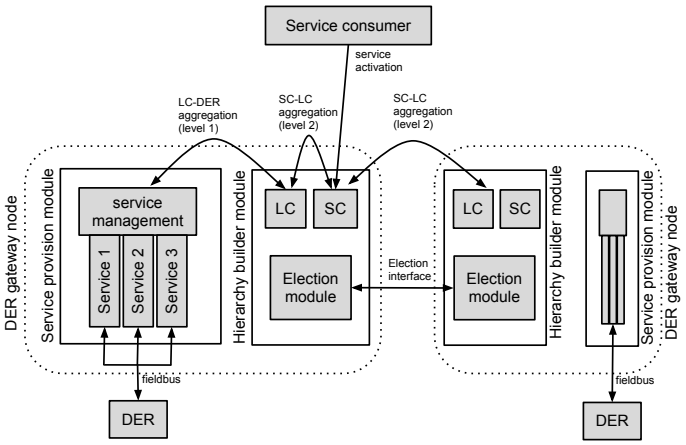

Fig. 2. Components of the presented system architecture.

The hierarchy builder modules across all nodes are responsible for negotiating their position and role in a single-level or multiple-level aggregation tree consisting of one supervisory controller (SC) at the root of the tree, local controllers (LC) managing individual branches and regular nodes at the leaf positions. Because each node must possess the capability to perform either of these three functions, each hierarchy builder module has fully-featured SC and LC components that are activated or deactivated depending on the role of the node in the aggregation tree. The SC acts as a single point of access for the activation of a service by service consumers. Node roles at each level are negotiated by leader election (see section III-B). This functionality is located in an election submodule.

III. Implementation

The following section provides additional detail on how the above design can be implemented in practice.

A. Unit roles

Depending on the aggregated service and DER commitment, a DER can fulfill one or several of the following roles in the proposed architecture: Passive unit, service provider, service committer, service executer, local controller or supervisory controller.

- A DER is a **passive unit**, when it is able to join an aggregation, but does not currently provide one of the aggregated services. Aggregators could either ignore passive units or include them in the aggregate as an uncontrollable unit for the purpose of monitoring its state as part of the system state. The present implementation uses the first option and does not include passive units in the aggregate.
- A **service provider** is a DER which is able to join an aggregation and to provide a specific service to the aggregator. Aggregators may chose to exclude service providers from the aggregate, for example based on their location.
- If a DER is accepted to provide a service in an aggregation, it becomes a **service committer**. A committed DER is still operating autonomously on its internal controller, but it is ready to change its behaviour based on external commands or setpoints from the aggregator.
- A DER becomes a **service executer** if one or more of the services it offers becomes activated by the aggregator.

B. Bully Algorithm

In this paper a distributed systems election algorithm is used to choose a coordinator to play a distinct role among nodes and avoid a single point of failure in the aggregation. A node can participate in an election if it is capable of performing a coordinator role. In the Bully algorithm proposed by Garcia-Molina [8], considered the most popular distributed system election algorithm, the choice of a coordinator is based on a process number, i.e. a unique ID such as the MAC address of the device. After a coordinator has been chosen, all other nodes continuously monitor if the coordinator is still available by sending heartbeat messages. If the coordinator fails to respond before a specified timeout, the coordinator is assumed to have failed and a new election round is triggered.

The Bully algorithm can be implemented as a finite state machine (FSM), detailed information about the FSM can be found in [8]. The implementation used for the paper makes use of eight states: The algorithm is started in *INIT* state. If no coordinator is known, or if the previous coordinator is assumed to have failed, the node enters *NO_COORDINATOR* state and broadcasts a proposal to start a new election to all other nodes with a higher process number than itself and enters *ELECTION* state. If there is no response to this request within a time time *T'*, the node switches to *NO_RESPONSE* state and proclaims itself coordinator by broadcasting to all nodes with a lower process number than itself before switching to *COORDINATOR* state. If on the other hand a response to the election message is received, the node halts in *WAIT* state for a defined time *T"* to see if another node with a higher process number would claim to be coordinator. If this happens, it saves the process number of the new coordinator, switches to *REGULAR* state and continues its operation. If there is no coordinator message after time *T"*, the node broadcasts for a new election and switches again to *ELECTION* state.

In a system with *n* nodes, in the worst case scenario, when a node with the lowest process number starts an election, $O(n^2)$ messages are sent [9].

C. Two-level Bully algorithm

It is possible to nest multiple instances of the Bully algorithm in order to achieve a multi-layer system. Two level Bully Algorithm consist of several steps executing bully algorithm for different groups. In the initial state all DERs are assigned a location tag, dividing the aggregation into level 2 groups, for reference see figure 1. Every group from level 2 elects a coordinator using the Bully algorithm. Once a coordinator is chosen it is assigned with a *local_control* tag and included in a local controller group. Members of this group initiate a bully election to choose a single supervisory controller. After a bully algorithm is executed a unit can be in one of two states: *regular* and *coordinator*, where only one unit can be in the coordinator state. After the two level bully algorithm is executed, a unit can be in any of three states: *regular*, *local controller* or *supervisory controller*, where only one unit can be in the supervisory controller state.

The aggregation process presented in this paper is executed in two election steps and adopts bottom-up approach, electing leaders from lower group first, then choosing higher leaders from leaders in groups below. The aggregation forms a hierarchy tree that can be reconfigured in case of any unit failure with help of two level bully algorithm.

D. Failure of a DER node

A failure of any DER can affect the aggregation and disturb service provision. Hierarchy builder module is responsible for recognizing the failure and reorganize the aggregation to continue its operation with as little disturbance as possible. Bully algorithm for leader election is used to form the aggregation structure and reconfigure in case of a failure with use of the election module. The election module reports to hierarchy builder module every time a new configuration is assembled. The hierarchy builder module informs LC and SC about changed configuration or activates and deactivates SC and LC components depending on the election outcome for the local DER.

As described in this section presented architecture allows controllable DERs to play six roles in the aggregation process. Four of these roles: service committer, service executer, LC and SC are roles involved in the operation of the aggregation. In case of failure of different roles, the aggregation need to be reconfigured and lost roles need to be reassigned. A disappearance of a DER is first recognized by the election module due to a missing heartbeat signal. When a failure is recognized and confirmed the election module starts a new election. Independently of the DER role a new leader election must be executed, because the faulty DER might have been a leader in one of the aggregation levels. The reaction of the hierarchy builder module is presented in Table 1.

TABLE I. FAILURE OF A DER ROLE AND REACTION OF THE HIERARCHY BUILDER MODULE.

DER role	election module	hierarchy builder module	LC and SC
SC	Reconfigure level 2, report the change of configuration to the hierarchy builder module	If DER elected for SC activate the SC component	Restart SC
LC	Reconfigure disturbed group from level 1, reconfigure level 2, report failure to hierarchy builder module	If elected for LC activate the control	Restart LC and send a control signal from SC to LC
Service comitter	Reconfigure disturbed group from level 1, report to hierarchy builder module	Report to LC and SC components	SC updates topology description and correct calculations, SC sends control signals to LC, LC sends control signals to DERs
Service provider	Reconfigure disturbed group from level 1, report to hierarchy builder module	Report to SC and LC components	SC updates topology description and corrects calculations, SC dispatch new service activation commands to LC, LC sends control signals to DERs

When the node providing the supervisory controller role disappears from the aggregation, a new SC needs to take over. The election algorithm is re-run in the level 2 group. Likewise, a local controller failure will trigger an election in the associated level 1 group. When a new LC has been elected, it joins the level 2 group and thereby triggers a level 2 election. If a regular DER node fails, i.e. a node with neither LC or SC duties, an election is held only in the affected level 1 group. Subsequently, this may also cause a re-election in the level 2 group. If the failing node has been acting as a service provider, LC and SC must recalculate their aggregation in order to reflect the changed DER portfolio.

IV. PROOF OF CONCEPT

In order to demonstrate the viability of the concept, a physical experiment was conducted using the SYSLAB facility at the Technical University of Denmark's Ris campus.

A. Experimental set-up

SYSLAB is a research facility for intelligent, active and distributed power systems. The current setup is spread across four substation sites on the Ris campus. A 400V, 3-phase grid with a total of 16 busbars serves as the electrical backbone of the facility. A large variety of DER units are connected to the grid at different sites: wind, solar and diesel generation, energy storage systems and a different types of loads. Among the latter are controllable buildings, electrical vehicles and various load simulators. The power system is able to run grid-connected or isolated. All components of the grid - DER units and substation equipment - are equipped with an intelligent node. A distributed software platform is deployed on the nodes and provides SCADA functionality, control interfaces and a container for embedding controllers [10]. The SYSLAB infrastructure can be used for experiments with either centralized and decentralized approaches to control. For the purpose of demonstrating the aggregation concept, ten DER nodes are used: Three PV systems and seven controllable loads, connected to the grid at three different substations as shown in figure 3. One of the loads (PFH) is an office building with 10kW of controllable electrical heating load. Six other buildings (SH1 through SH6) are real-time hardware-in-the-loop simulations based on a thermal model of the first building. The power demand of these simulated buildings is applied to the physical grid through controllable load simulators connected at the respective points in the grid. The three PV systems with grid-tie inverters have a rated power of 10kW, 10kW and 7kW respectively.

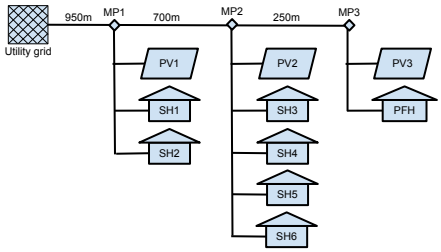

Fig. 3. Power system setup for the experiment, including PV systems (PV), physical building (PFH) and simulated houses (SH1-6)

In order to separate the dynamic aggregation problem from that of power system control, the experiment described in this paper focuses on demonstrating the dynamic aggregation part. The supervisory and local controllers are represented as empty containers without any actual capability to influence the operation of the DER units or the state of the grid. However, the geographically distributed execution environment and the details of the deployment would be the same if a grid controller was running inside the containers.

B. Results

The ten DER nodes used in the experiment are distributed between three node groups according to the grid substation they are connected to: *A*, *B* and **C** (see figure 3). The corresponding host names of the nodes - which are subsequently used to identify them - are listed in the following table:

Group *A*	Group *B*	Group *C*
syslab-02	syslab-10	syslab-v02
syslab-03	syslab-v13	syslab-ui5
syslab-04	syslab-21	
	syslab-22	
	syslab-24	

For example, syslab-02 is the host name of the node controlling the PV system PV1, connected to the first substation and consequently a member of group A.

Figure 4 shows the state of the system over time. Before the experiment, all nodes have been assigned to a single *default* group and are configured to participate in a single-level election algorithm. At timestamp zero, a reconfiguration command is sent to each node: They are now assigned to one of the three groups and are ordered to build a two-level aggregation hierarchy by electing local and supervisory controllers. The states of the two levels of the election algorithm are shown as separate tracks in the figure. Additionally, the dotted red line together with the "LC" and "SC" labels indicates which nodes have been assigned local or supervisory controller roles as the result of the election.

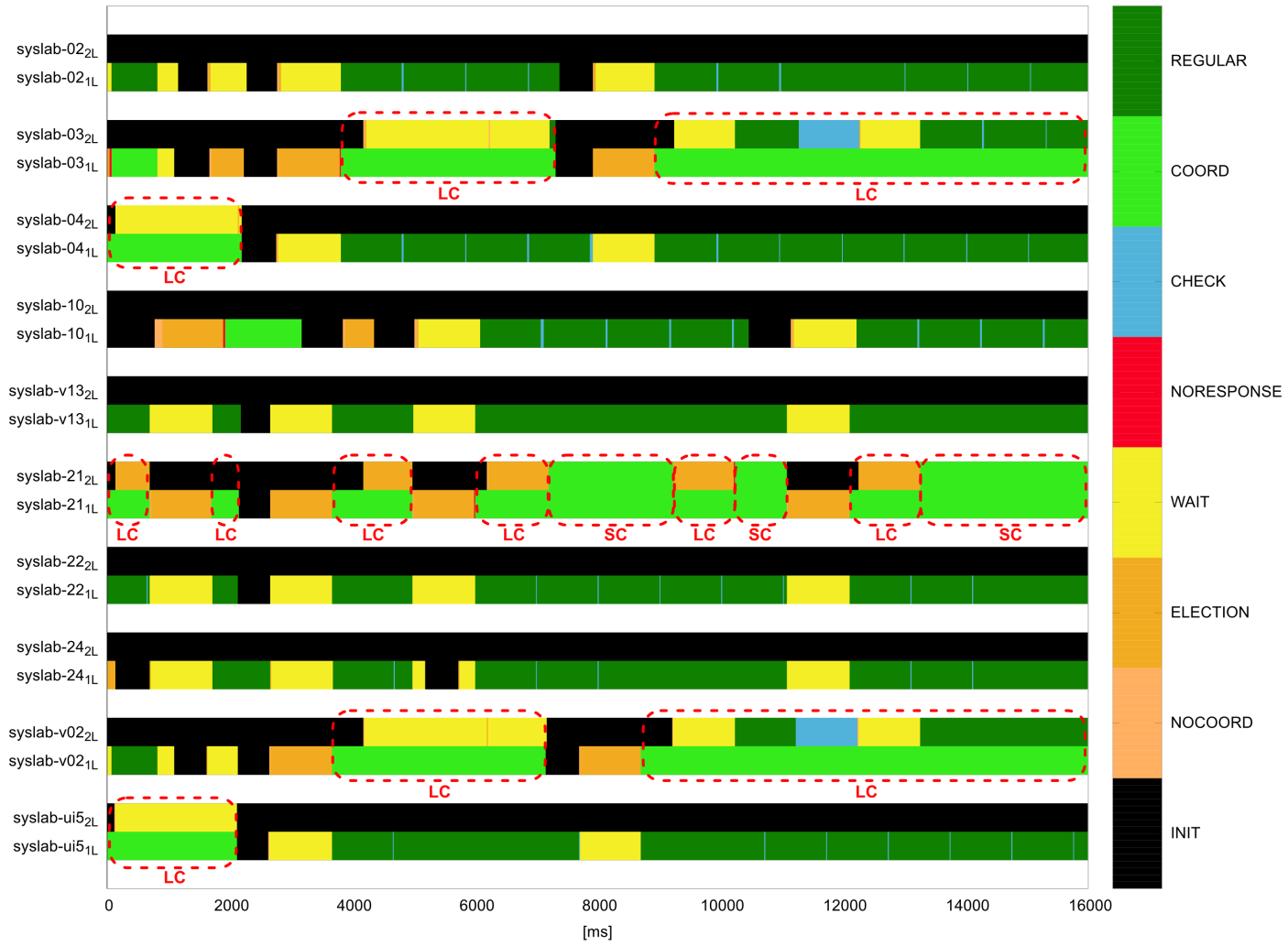

Fig. 4. Representation of the state of each node in the aggregation scheme during the experiment. The colors correspond to the eight possible states of the Bully algorithm on each of the ten nodes. The lower track (index 1L) for each node represents the state of the first level election. The upper track (index 2L) represents the state of the second level election. The dotted red lines mark the nodes which have been elected to LC or SC during a particular period of time.

In the configuration period every node is expected to reconfigure several times: start operation, change level and group in level 1. For example 1L state of the node *syslab-02* turns into INIT three times, first when the node is initialized at 1 second, next when a group is assigned at second 2, ten after second 7, when the level is assigned. It can be seen that, due to differences in processor speed, not all nodes react equally fast.

In case of a disappearance of a coordinator, units participating in the same group react by choosing next coordinator among themselves. In the second two of the experiment *syslab-04* 1L state changes from COORDINATOR to INIT state, removing a controller from the group *A*. Node *syslab-03* takes over the responsibility in less than 2 seconds, as shown in figure 4.

About 13 seconds into the experiment the aggregation reaches a steady state. The delay in reaching a steady state in the aggregation appears due to slow response of some nodes. For example, node *syslab-10* is the last one to finish its reconfiguration, with a delay of about 7 seconds compared to node *syslab-21*.

The results presented in figure 4 can only reflect a partial behavior of the aggregation, showing state of the aggregation topology while nodes were reconfiguring. Change of a state in one node effects behavior of other nodes, so when a leader was lost, another responsible unit was immediately chosen. The overall state of the state comes into steady state when individual node behavior is stabilized. The results from the performed experiments are showing practical behavior of a small aggregation, performing dynamic reconfiguration and fault recognition and tolerance, while sustaining control hierarchy.

1) Performance measurement: The experiment was run multiple times with different numbers of participating nodes, in order to extract two measures of performance: number of messages sent between nodes before reestablishing a steady state after a disturbance, and the time required to reach steady state. The number of participating nodes was varied between

2 and 21.

In order to compare the classical Bully algorithm and the two-level extension proposed in this paper, identical experiments were conducted for a system with only one aggregation level, where all nodes share the same group.

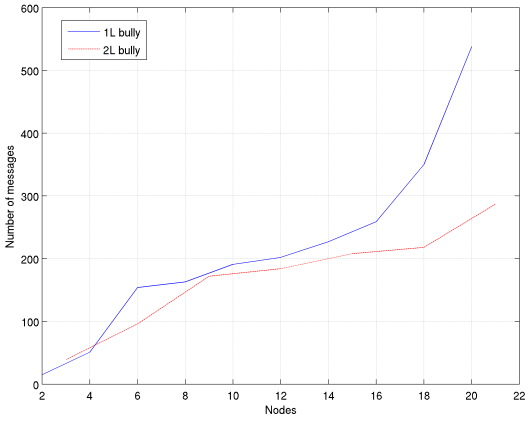

Fig. 5. Number of election messages exchanged between nodes for one-level and two-level algorithms, as a function of the number of nodes.

Figure 5 compares the performance of the one-level and two-level algorithms based on the number of election messages, shown for different number of nodes. In both cases some messages have been excluded from these calculations, such as the *check* and *ack-check* messages used to start a re-election in case a node fails to respond. These messages are only present during the steady state of the system and should therefore not be counted as a measure of effort to reach the steady state. As shown in figure 5 the two-level algorithm performs fewer elections. This is one of the expected benefits of the two-level system and can easily be explained since only nodes from the same group respond to the failure of a node.

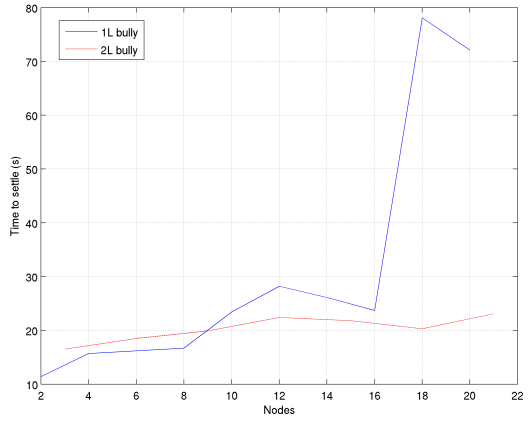

Fig. 6. Time to reach steady state for one-level and two-level algorithms, as a function of the number of nodes.

Figure 6 shows that the settlement time for the one-level algorithm increases rapidly with the number of nodes. This result is not unexpected considering the exponential growth in the number of messages (figure 5). The settlement time for the two-level algorithm increases also with the number of nodes, but much slower.

V. Conclusion and Future Work

This paper presents a distributed and fault tolerant dynamic aggregation concept for the provision of grid services from DER units. The proposed architecture consists of two main modules: a service provision module and a hierarchy builder module, each of which are executing locally at each DER unit. The dynamic aggregation hierarchy is formed using a multi-level election algorithm. The performance of a system has been tested as part of a proof-of-concept implementation on a Smart Grid test bed.

The two-level election algorithm performs better than its single-level counterpart with respect to the number of messages exchanged and the time to settle. This promises improved scalability and faster adaptation to changing conditions. Even for the two-level case however, the number of messages begins to grow superlinearly above a certain number of nodes. It may be possible to at least partially mitigate this effect by appropriate partitioning of the communication network: Meaningful failover is still possible even if the election process is partially localized, for example based on the point of grid connection of a DER unit. Among other issues, this will be the subject of future work.

As a next step, it is planned to add actual control functions to the local and supervisory controller containers. Of particular interest in this context is the performance and uninterrupted function of controllers while a re-election takes place in part of the aggregation hierarchy.

References

[1] M. Sebastian, J. Marti, and P. Lang, "Evolution of dso control centre tool in order to maximize the value of aggregated distributed generation in smart grid," in *SmartGrids for Distribution, 2008. IET-CIRED. CIRED Seminar*. IET, 2008, pp. 1–4.

[2] P. Vinter, "Introduction to power hub," DONG Energy, Tech. Rep., November 2011.

[3] T. Kostic, C. Effantin, and E. Lambert, "How to increase interoperability in european smartgrid projects? the address experience regarding model driven integration based on international standards," 2012, pp. 652–657.

[4] L. A. Villas, D. L. Guidoni, R. B. Araujo, A. Boukerche, and A. A. Loureiro, "A scalable and dynamic data aggregation aware routing protocol for wireless sensor networks," in *Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems*. ACM, 2010, pp. 110–117.

[5] A. A. Sharifloo, M. Mirakhorli, M. Esmaeili, and A. Haghighat, "A leader election algorithm for clustered groups," in *Industrial and Information Systems, 2007. ICIIS 2007. International Conference on*. IEEE, 2007, pp. 1–4.

[6] K. H. Wan and C. Loeser, "An overlay network for replica placement within a p2p vod network," *International journal of high performance computing and networking*, vol. 3, no. 5, pp. 320–335, 2005.

[7] A. S. Tanenbaum and M. Van Steen, *Distributed systems : principles and paradigms*. Prentice Hall, 2007, vol. 2.

[8] H. Garcia-Molina, "Elections in a distributed computing system," *Computers, IEEE Transactions on*, vol. C-31, no. 1, pp. 48 –59, jan. 1982.

[9] A. N. Alslaity and S. A. Alwidian, "A k-neighbor-based, energy aware leader election algorithm (kelea) for mobile ad hoc networks," *International Journal*, vol. 59.

[10] O. Gehrke and H. Bindner, "Building a test platform for agents in power system control: Experience from syslab," in *Intelligent Systems Applications to Power Systems, 2007. ISAP 2007. International Conference on*. IEEE, 2007, pp. 1–5.