Decentralized Voltage Stability Monitoring and Control in the Smart Grid using Distributed Computing Architecture

H. Lee
EECS
Wash. State University
hyojong.lee@wsu.edu

S. Niddodi
EECS
Wash. State University
shwetha.niddodi@wsu.edu

A. Srivastava
EECS
Wash. State University
asrivast@eecs.wsu.edu

D. Bakken
EECS
Wash. State University
bakken@wsu.edu

***Abstract*—To manage the smart electric grid of the future, fundamental changes are required in the system operational paradigm. Availability of high-resolution data at faster speed and advanced computational advancements provide opportunities to bring this fundamental change. Monitoring and control algorithms need to be evolved to match the transition of centralized generation to distributed generation. Intermittency of renewable generation and push towards real time control requires faster control actions, which is possible with decentralized power grid applications. With integration of distributed energy resources (DERs), the stability assessment application need to handle a large number of data points in real time. This requires massive computing resources, and requirements will increase for possible real time control action. Decentralized applications need to be coordinated and manged with existing centralized applications. This paper addresses the development of a fault-tolerant distributed computing architecture (DCBlocks) for implementing a decentralized voltage stability monitoring and control application. Results for IEEE 30 bus system have been provided to validate the developed architecture. Distributed computing algorithms are implemented using open source platform Akka Java and DeterLab test bed.**

I. INTRODUCTION

The electric grid is going through a major upgrade and more changes are expected in future to adopt with high penetration of renewable energy to meet the energy security and sustainability requirements [1] [2]. With replacement of high inertia dispatchable centralized generation by low inertia intermittent renewable generation with limited reactive power, power grid will be more stressed specially for frequency control, dynamic response, and voltage stability.

To manage the electric grid of the future, fundamental changes are required for the system operation. Advancement in sensors technology and computational advancement provide opportunities to bring this fundamental change. Much more sensor data is available today then even just a few years ago from sources such as (Intelligent Electronic Devices (IEDs), Phasor Measurement Units (PMUs), and distribution automation. Monitoring and control algorithms need to be evolved and be decentralized to match the transition of centralized generation to distributed generation as well as to handle large amount of data in a small amount of time.

Traditionally, power algorithms have been single-process programs hosted almost exclusively in control centers. We will call these algorithms *centralized*, one noteable exception has been traditional protection schemes and remedial action scheme (RASs). The more sophisticated distributed edge computations solve equations so the threshold for the control action are dynamically calculated based on present conditions. A generic reason for many of these being decentralized is that, even if they are not quite optimal, local decisions can be made much more quickly, and also avoid a centralized algorithm, which can be a communications bottleneck and a single point of failure or attack.

With increasing number of renewable penetration generally located far from load centers and limited reactive power, voltage stability problem can happen quickly with limit induced bifurcation. Additionally, power electronics control allows operating near the power system margin. Control actions should be taken quickly and probably autonomously. The centralized voltage stability applications are broadly studied by researchers. The Decision Tree (DT) based method is studied in [3], the DT is classified and trained by off-line simulation for all possible cases. The continuation power flow calculates PV curve and a point of collapse by predictor and corrector in [4], [5]. Modal based method have been used to find Point Of Collapse (POC) point using Jacobian matrix. The Jacobian matrix becomes singular matrix at the POC in [6]. Other centralized voltage stability methods are proposed in [7]–[12]. The centralized voltage stability may not work well in time with large number of variables to solve, high intermittency and for limit-induced bifurcation. Voltage stability problem is inherently local and can be solved locally using available reactive power resources in neighborhood dynamically.

This provides a good fit and requirement to develop distributed computing architecture (DCBlocks) for voltage stability. Additionally distributed application allows redundancy and fault-tolerant computing, if designed well.

Contribution of this paper is to provide a distributed computing architecture tailored for implementation of a decentralized voltage stability monitoring and control application. This paper develops decentralized voltage stability algorithm using the concepts from existing centralized and local synchrophasor based voltage stability monitoring and control but modifies it to fit the need of distributed implementation. Paper also provides a test case study using IEEE 30 bus system for validation of developed distributed computing architecture for voltage stability using Akka and Deterlab.

II. DECENTRIALIZED VOLTAGE STABILITY ALGORITHMS (DVS)

The DVS can be used as a secondary control running in several seconds to complement tertiary voltage stability

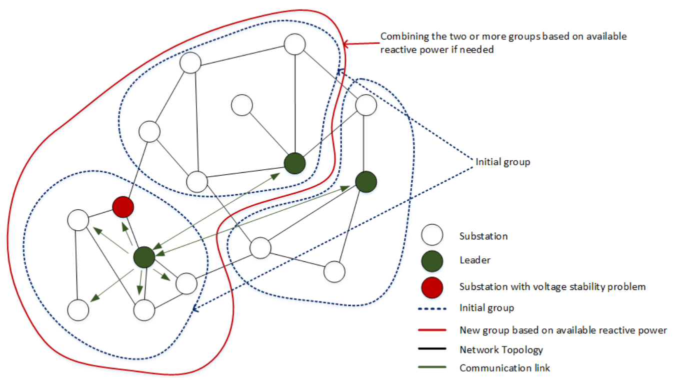

Combining the two or more groups based on available reactive power if needed
Initial group
Substation
Leader
Substation with voltage stability problem
Initial group
New group based on available reactive power
Network Topology
Communication link

Fig. 1: Power system network with group of nodes

optimal control running at control center in several minutes. There are number of different decentralized voltage monitoring and control algorithms proposed by researchers. The decentralized voltage regulator using fuzzy logic with graph partitioning method and bus voltage sensitivity is proposed in [13], [14]. An optimal coordinated voltage controller using a pseudogradient evolutionary programming (PGEP) technique is proposed in [15]. Most of these work address voltage monitoring and not the voltage stability monitoring in decentralized and coordinated manner. The main challenges of DVS are initial group formation and coordinating control action among groups.

Consider a power system network as shown in Figure 1. It consists of a network of substations, which can be grouped together using a technique developed in this paper. Grouping technique is based on the electrical distance, voltage to reactive power sensitivity and reactive power availability. The substations (hereafter referred to as nodes) have the computing devices able to compute voltage stability indices using reduced network model equivalent and synchrophasor measurements from phasor measurement units (PMU) [16], [17]. It is also assumed that these are capable of computing distributed state estimation (DSE) before performing voltage stability assessment. DSE is out of scope of this paper due to limited space and a good set of measurement data has been assumed in this work.

When a voltage stability problem at a particular bus is encountered in a reduced power network, all the substations within that group need to communicate with each other to provide the reactive power needed to improve the voltage stability using control actions discussed in this work.

Once the group is established, these nodes can again coordinate with each other to select one distinguished node having the maximum computational resources to perform voltage stability assessment. Lead computational node can also perform computation for needed control actions to improve the voltage stability.

A. *Group Formation Method for DVS*

Before selecting the leader of each group, the distributed algorithm needs to perform initial grouping from large power

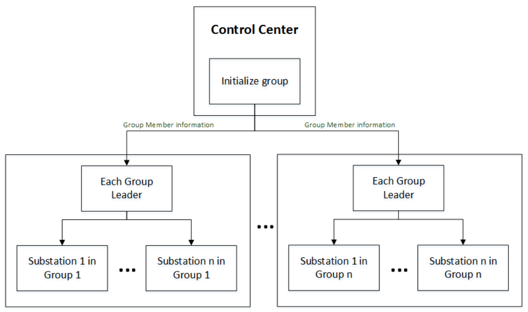


Fig. 2: Flowchart for Initializing Group Formation

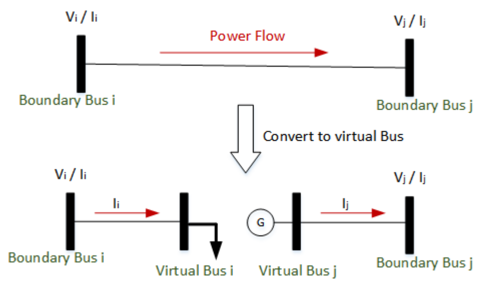


Fig. 3: Tie line inclusion in groups

system to multiple group of small power system nodes and computational resources. Following minimum requirements are considered to form group of nodes:

1) At least one generator
2) At least one transmission line
3) At least one load
4) At least one reactive power source
5) All components are relatively close measured by electrical distance and geographical distance

To determine initial group, this method requires entire power system information for computing Admittance matrix as shown in Figure 2.

The tie line impedance among groups is counted as half of the original impedance of the line in order to replace it with generation or load based on power flow direction. The virtual bus is connected to the boundary bus with half of tie line impedance. The virtual bus are considered as generator bus or load bus from the connected boundary substation as shown in Figure 3. The line flow is solved using PMUs data as shown in Figure 3.

The line flow is calculated as following:

$$S_{ij} = (V_i \times I_i^*) \quad Power\,leaving\,from\,Bus\,i$$
$$S_{ji} = (V_j \times I_j^*) \quad Power\,reaching\,to\,Bus\,j$$

where V_i, V_j, I_i, and I_j are bus voltage at Bus i and j and line current at Bus i and j respectively. The tie lines are replaced by following: a) If real part of S_{ij} has negative value, then the line is replaced as generator, b) If real part of S_{ij} has positive value, then the line is replaced as load .

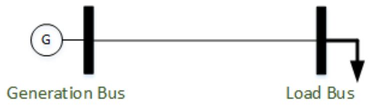


Fig. 4: Example of admittance for Y_{GL}

B. *DVS Monitoring Algorithm*

The DVS monitoring algorithm estimates voltage stability index (VSI) for each load bus using Thevenin's equivalent approach with the active and reactive power limit consideration. To perform Thevenin method, V_{th} and Z_{th} needs to be calculated by limited system information in each group. Once the system is grouped by initial grouping method, the buses are represented as generator bus, boundary bus, and load bus. The network topology can be represented as following:

$$\begin{bmatrix} Y_{GL} & Y_{GT} & Y_{GG} \\ Y_{TL} & Y_{TT} & Y_{TG} \\ Y_{LL} & Y_{LT} & Y_{LG} \end{bmatrix} \tag{1}$$

Y_{GL} are a set of admittance values between Generator and Load connections as figure 4.

Similarly, Y_{GT}, Y_{TL}, Y_{TT},Y_{TG}, and Y_{LL} are Generator to Tie line, Tie line to Load, Tie line to Tie line, Tie line to Generator, and load to load respectively.

-Load, Load-Tie, Tie-Tie, and Tie-Load respectively. Using reformed admittance matrix, Thevenin's equivalent parameters are calculated as following: [18]

$$v_{th_j} = \sum_{m=1}^{M} H_{LG_{jm}} v_{Gm} + \sum_{n=1, i\neq j}^{N} Z_{LL_{ji}} (\frac{-S_{Li}}{v_{Li}})^* \tag{2}$$

$$Z_{th} = Z_{LL} = (Y_{LL} - Y_{LT} Y_{TT}^{-1} Y_{TL})^{-1} \tag{3}$$

where, N is number of generation bus, M is number of load bus, Z_{th} is Thevenin impedance for each load bus, V_{th_j} is Thevenin voltage source for j^{th} load bus.

$$S_{max_j} = V_{th_j} \times (\frac{V_{th_j}}{Z_{th_j}})^* \tag{4}$$

Using calculated maximum apparent power, VSI_P, VSI_Q, and VSI_S is calculated as following:

$$VSI_S = 1 - \frac{S_{max_{eachload}} - S_{L_{eachload}}}{S_{max_{eachload}}} \tag{5}$$

$$VSI_P = 1 - \frac{P_{max_{eachload}} - P_{L_{eachload}}}{P_{max_{eachload}}} \tag{6}$$

$$VSI_Q = 1 - \frac{Q_{max_{eachload}} - Q_{L_{eachload}}}{Q_{max_{eachload}}} \tag{7}$$

$$VSI = max(VSI_S, VSI_P, VSI_Q) \tag{8}$$

where, $S_{max_{eachload}}$ is maximum power transferred at each load, $S_{L_{eachload}}$ is current load at each load.

Similarly, $P_{max_{eachload}}$ and $Q_{max_{eachload}}$ correspond to the real and reactive component of maximum power transferred at each load. Also, $P_{L_{eachload}}$ and $Q_{L_{eachload}}$ are real and reactive

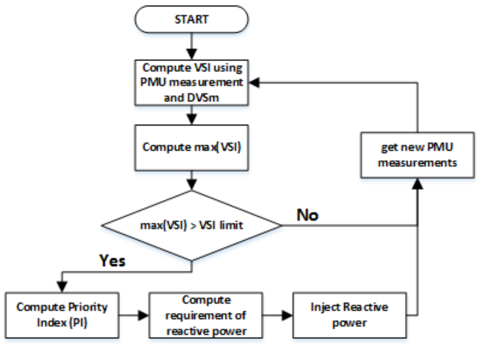
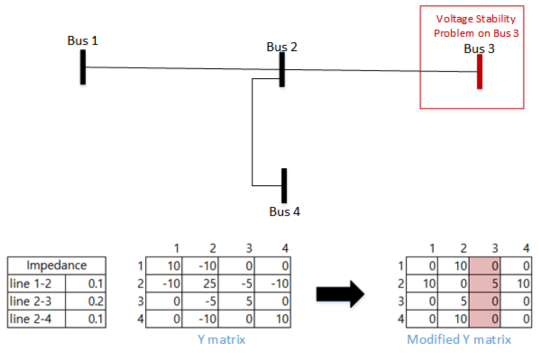

START
Compute VSI using PMU measurement and DVSm
Compute max(VSI)
max(VSI) > VSI limit
No
Yes
get new PMU measurements
Compute Priority Index (PI)
Compute requirement of reactive power
Inject Reactive power

Fig. 5: DVS control action scheme



Impedance	
line 1-2	0.1
line 2-3	0.2
line 2-4	0.1

	1	2	3	4
1	10	-10	0	0
2	-10	25	-5	-10
3	0	-5	5	0
4	0	-10	0	10

	1	2	3	4
1	0	10	0	0
2	10	0	5	10
3	0	5	0	0
4	0	10	0	0

Fig. 6: Example system for priority index

component values of current load at each bus. The maximum VSI is used for Distributed Voltage Stability Control (DVSc) algorithm to control reactive sources based on Priority Index (PI) as discussed in next section.

C. *DVS Control Algorithm*

Our DVS approach is based on the assumption that the DVSc algorithm can be activated under emergency stage within few seconds. The base VSI is predefined by benchmarking with continuous Power flow (CPF) result. The reactive power is redistributed using the developed algorithm and using priority index (PI) as shown in Figure 5. The the DVSc algorithm considers the electrical distance and network sensitivity as shown in Figure 6.

The procedure of finding PI is shown as following.

1) The Y matrix is used to compute priority index.
2) Top priority is given to reactive power source at the bus that has voltage stability problem (e.g. load bus 3)
3) Next set of priorities is given to reactive power sources based on the ascending ranking of electrical distance to the problem load bus for directly connected lines. (As shown in Figure 7, highlighted second row of Y matrix.)
4) Next set of priorities is given based on ascending ranking of cumulative electrical distance of reactive power sources from the target bus (e.g. modified second row using the target bus row)

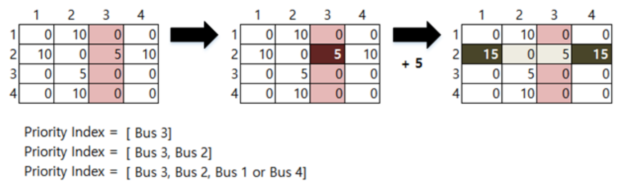

+ 5
Priority Index = [Bus 3]
Priority Index = [Bus 3, Bus 2]
Priority Index = [Bus 3, Bus 2, Bus 1 or Bus 4]

Fig. 7: Priority of reactive power sources using PI

5) Repeat step 3 and 4 until all possible priority index is computed as shown in Figure 7.

Once all the priority indices are computed, reactive power required for compensating the voltage stability problem on target bus is calculated. Part of Jacobian Matrix can be used to calculate reactive power required as following:

$$\frac{\Delta Q}{\Delta V} = \frac{\delta Q}{\delta V} \tag{9}$$

$$(Q_{req} - Q_{PI_i}) = \frac{\delta Q}{\delta V} \times (V_{req} - V_{PI_i})$$

$$Q_{req} = \frac{\delta Q}{\delta V} \times (V_{req} - V_{PI_i}) + Q_{PI_i} \tag{10}$$

where Q_{req} is required reactive power to fix the voltage stability problem, V_{req} is minimum acceptable voltage, Q_{PI_i} is reactive power at target bus, V_{PI_i} is voltage at target bus, and $\frac{\delta Q}{\delta V}$ is the part of Jacobian Matrix.
The control action should be performed multiple times if the voltage problem is not solved in one step control action. The DVSc uses all reactive power reserve until the problem is solved within a group. If the voltage problem is not solved, DCBlocks will communicate with other group lead for routing more reactive powers.

III. Overview of Distributed Coordination Computational Algorithms (DCBlocks)

Robust implementation of Distributed Voltage Stability (DVS) is implemented using distributed computing architecture. The proposed algorithm is called DCBlocks, as it is set of generalized distributed computing blocks, which can be easily plugged in for different decentralized applications [19]. The computing entities periodically coordinate with each other to achieve a common application goal using message passing. Building decentralized applications is non-trivial due to several factors. Some of them are: variable network delay, variable computational delay, messages arriving in different order at each destination, different failure types seen at each coordinating process, perturbations due to cyber security attacks such as Distributed Computing (DC) is a field, which asks the question "how can we best use computational resources and computer networks in order to help distributed applications and services" [20]. The components in the distributed systems can be heterogeneous in terms of computer hardware, operating system, programming language used and still interact with each other using Middleware as needed for implementation of decentralized voltage stability (DVS). The applications running on different systems are not aware of their exact physical location and "discover" each other using some sort of discovery service.

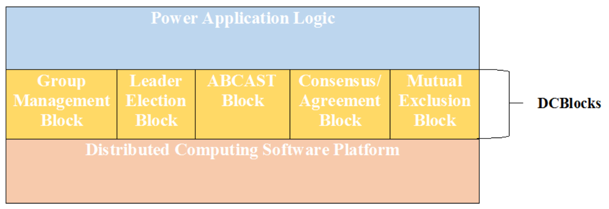


Fig. 8: Distributed Coordination Software Stack using DCBlocks

Some of the important distributed coordination algorithms [21], [22] that can be utilized to enable more robust distributed power applications are:

- Agreement [20], [23], [24]
- Leader Election [20]
- Mutual Exclusion [20], [25]
- ABCAST [26]
- Voting
- Interactive Consistency (IC) [27], [28]
- Group Membership
- Group Discovery/formation
- Supply Agreement

The figure 8 shows the DC software stack for distributed power applications with DCBlocks between the application logic and distributed software (DS) framework. The library of DCBlocks is designed to implement (and simplify access to) all the algorithms discussed above as separate implementation blocks.

A. Distributed Software Platform

The underlying distributed software platform handles the actual communication between the processes in the distributed network with supported features like atomicity (all processes receive messages or none do) and message ordering. Some distributed software platforms also provide other features like group management and fault tolerance support. The DCBlocks library uses one such open source software platform called Akka Java [29]. The Akka Java toolkit uses actor model where in each computing entity is an actor and the actors communicate with each other by exchanging messages asynchronously. Akka Java also provides group (called clusters) membership and member life cycle management services. It has a well-defined supervision hierarchy and failure handling strategy for some of the failures like crash failures. The DCBlocks is built using all of these features.

B. Design of DCBlocks

The following sections provide brief description of each implemented block.

1) Group Management Block: It provides methods to create group of processes (formed for specific purpose), add or remove members from the group, monitor the status of the group members, detect and report member failures. It also provides methods to dynamically reorganize groups with merge (two or more groups are merged into one) and split operations (single group is split into two or more groups).

2) ABCAST Block: This block provides mechanism to multicast messages to all members in its group or to external groups. If a process wants to send a message to all members in the group, it needs to use "publish" method with desired topic to do it. All the processes in the group interested in receiving that message has to subscribe with the same topic. The underlying DS framework handles the actual publish-subscribe communication

3) Leader Election Block: This block provides methods to elect one process among them as leader for the group. Each participating node need to propose a vote to start the election. The leader election is executed through multiple rounds and finally a leader for the group is elected. DCBlocks is designed to be generic to accept any application defined decision criteria to decide on the leader (for example, select node with least computational work load). A Secondary leader can also be elected which can be a fall back leader in case of failure of primary leader.

4) Consensus or Agreement Block: This block implements two consensus or agreement algorithms.

- Simple Consensus [20], [30] - Here processes agree on same decision (scalar) value from list of proposed values. Each process proposes a local value and proceeds through subsequent rounds trying to collect all the proposed values. After f + 1 rounds where 'f' is number of faulty nodes, each process decides on the same value. The algorithm reaches consensus/agreement even in the presence of failures. The decision function can be any function like - computing the minimum, maximum, average etc of proposed values.
- Interactive Consistency Algorithm [20], [30] - Processes agree on a vector of values (one sent by each process). The execution of rounds is similar to simple consensus except that finally all processes agree upon same vector of values.

5) Mutual Exclusion Block: This block provides means to synchronize concurrent access of shared resources among communicating processes. If a process wants to gain access of a shared resource, it sends a REQUEST message and waits for REPLY messages from ALL processes. If a process is holding the shared resource, it does not send REPLY message immediately and puts the REQUEST message in its request queue. After releasing the resource, it sends REPLY message to the process in the front of the queue. Concurrent requests are handled by checking the request having the lowest time stamp.

6) Failure detection and tolerance: DCBlocks can handle failure types like crash, omission, and timing failures. The design employs timeout mechanism to translate crash and omission failures into timing failures using timeout. Application logic can register with group management block to get notified on member failures for example the leader, so that it can switch to secondary leader to perform leader activities.

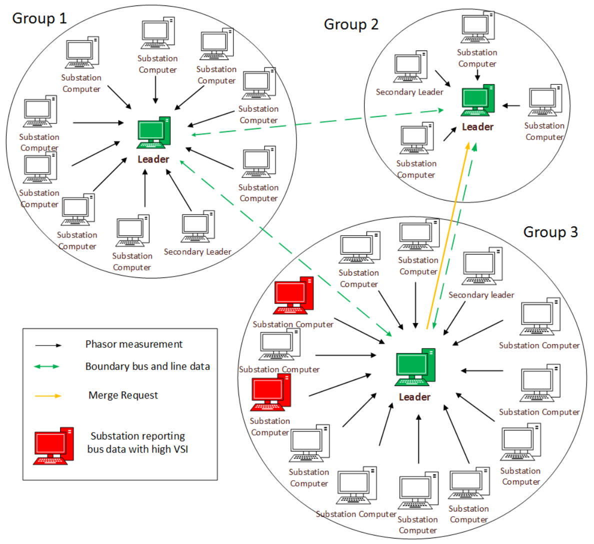


Fig. 9: Initial Groups

C. Application Logic using DCBlocks

The distributed power application logic can use any of the DCBlocks as per their needs. The DVS application can use DCBlocks group management block to form groups, and leader election block to elect a leader for the group.

IV. Implementation of Decentralized Voltage Stablity using DCBlocks

This section describes the implementation of our DVS algorithm using DCBlocks. The power system used to implement DVS application is IEEE 30 bus power system. It is modeled in DETERlab [31] using 30 deter nodes where each deter node represents a substation connected to a power system bus. It is assumed that each substation has computing device running the DVS software. DETERlab is a testbed consisting of hundreds of nodes and can be used for testing experiments in which the nodes may be configured in a variety of ways with any of several existing operating system and with varying network topology. It allows to inject communication network faults, statistical delays, simulate network link failures, introduce security attacks, which makes it a great platform to perform useful and realistic experiments [3], [32].

The implementation steps of DVS using DCBlocks are:

1) On system start up, the DVS application computes the initial grouping as explained in Section II-A.
2) Based on the initial grouping, the nodes join the group using add member method of DCBlocks Group Management Block. The initial grouping is shown in Figure 9.
3) The nodes start a leader election and the nodes having the best possible computational power in the group is elected as the leader with the help of DCBlocks Leader Election

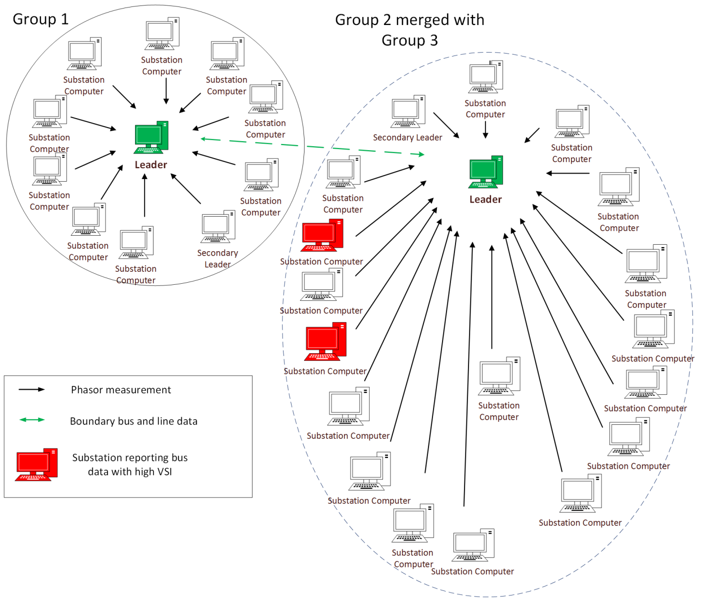
Block. A secondary leader is also selected as back up leader if the primary leader fails.

4) The node selected as the leader starts the leader activities (i.e, wait for incoming power measurements from all group members, run DVS monitoring and control algorithms as required).
5) Each node starts to send its local power measurement (bus and line information) to its group leader at regular intervals.
6) The boundary bus and line information is obtained from the adjacent group leaders.
7) Once all the power measurements are received from the group members, the leader runs the DVS monitoring algorithm as per Section II-B.
8) If the leader detects that maximum VSI is greater than threshold (e.g. 0.7), it performs the control action using the DVS control algorithm as per Section II-C. It might take several control action attempts to resolve the problem.
9) If the problem cannot be resolved within the group, i.e. reactive power is insufficient then merging of one or more adjacent groups becomes necessary.
10) The lead node requests adjacent group leaders to send an estimate of reactive power reserve in their groups (with the help of DCBlocks group management block). The best candidate group is selected and the lead node sends merge request to that group. This is shown as dotted arrow in Figure 9.
11) The group merges with the adjacent group with the help of merge method of DCBlocks Group Management Block. This is shown in Figure 10.
12) New group leader is appointed for the merged group.
13) Steps 4 to 7 is repeated.
14) After the voltage stability problem is resolved, the group is regrouped (split) into original two groups using split method of DCBlocks Group Management Block.
15) Steps 3 to 14 is repeated continuously.

The communication between the processes is accomplished by DCBlocks and underlying DC software layer. It can tolerate upto f = (n - 1)/2 faulty processes in a group where 'n' is the total number of processes in the group. If the leader is detected as faulty, then a secondary back up leader can take over to perform leader activities. The simulation results of this application for different test cases are described in the next section.

V. Test Cases and Simulation Results

To test DVS with DCBlocks, a IEEE 30 Bus power system is used [33]. Although DVS offers a suitable case study for distributed modeling and simulation using physically distributed resources, simulation results here are based on remotely located DETER nodes and locally available computing resources. An extra controllable shunt capacitors and Static VAR compensator (SVC) are installed in load buses (bus 3, 7, 10, 14-22, 24, and 30) for control action. Also, the control action is provided by changing the transformer tap,


Group 1
Group 2 merged with Group 3
Substation Computer
Leader
Secondary Leader
Phasor measurement
Boundary bus and line data
Substation reporting bus data with high VSI
Fig. 10: Merged Groups

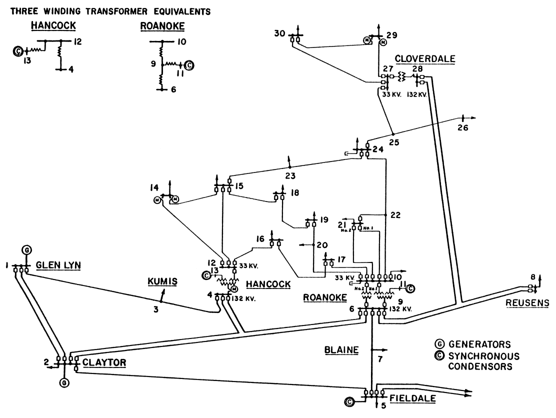
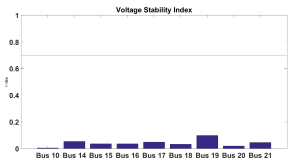


Fig. 11: The IEEE 30 Bus system

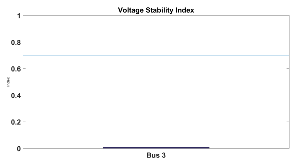
re-scheduling generator, and load shedding. The IEEE 30 bus system is grouped by DVS grouping method in section II-A as follows:

1) Group 1 = [10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]
2) Group 2 = [1, 2, 3, 4, 5]
3) Group 3 = [6, 7, 8, 9, 11, 22, 23, 24, 25, 26, 27, 28, 29, 30]

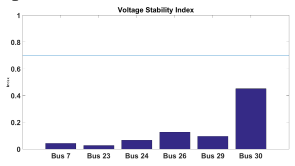
The IEEE 30 bus system has two areas; load area and generation area. By minimum requirements of grouping and maximum limit of group members, load area is formed into group 1 and 3. The generation area is formed as group 2. For test system, the base VSI limit is set as 0.7 based on benchmarking study. DVS application with DCBlocks were simulated in DETERLab testbed. The assumption made here is that each substation has a computational device capable of running this integrated software. The Deter nodes are connected via LAN with a network bandwidth of 500Mbps. With this test setup, a case study of three voltage stability scenarios



(a) Group 1



(b) Group 2



(c) Group 3

Fig. 12: Voltage Stabiltiy Indices for each group for normal case

were conducted and simulation results were obtained.

A. Normal condition

In this scenario, the VSI values at each bus is well below the threshold and the system is working in normal operating condition. Each substation sends the bus and line information to its group leader every 30 seconds. The leader collects the bus and line data from all the group members, boundary bus and line data from adjacent group leaders and runs the DVS monitoring algorithm. The test results for the three groups for this case is shown in Figure 12.

All VSIs stay within the threshold in this case. The control action is not required.

B. Case I: Voltage Stability problem and resolved within the group

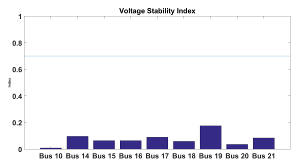
In this scenario, the VSI values for bus number 30 in group 3 becomes 0.757 exceeding the voltage stability limit (0.7) and the leader performs the control action to resolve the problem. In each control action attempt, reactive power resource from the group member is applied to the affected buses as per priority index. The reactive power reserve in group 3 is sufficient to resolve the issue as shown in Figure 13c, the voltage stability problem at bus 30 is resolved after 2 control action attempts. The simulation results for all groups are shown in Figure 13.

C. Case II: Voltage Stability problem and regrouping is needed

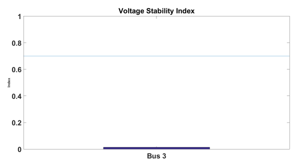
In this scenario, the VSI values for bus number 30 in group 3 becomes high (1.044) as shown in Figure 14. The reactive power reserve in group 3 is not sufficient to resolve the issue as shown in Figure 14 and group 3 is merged into group 1. After couple of control action attempts, the problem is resolved and merged group is split into original group 1 and 3 as shown in Figures 14d and 14e .

VI. Conclusions

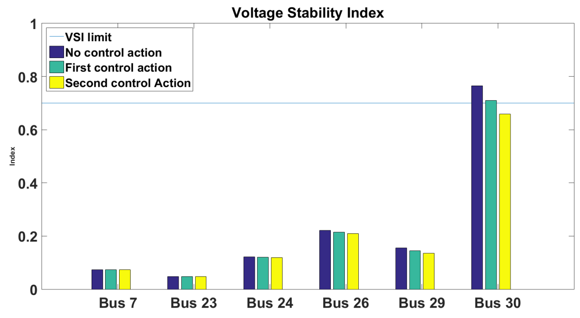
Given the advancement in control algorithms and push towards higher economics the power system is operating closer



(a) Group 1

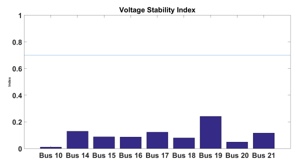


(b) Group 2

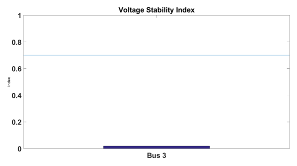


(c) Group 3

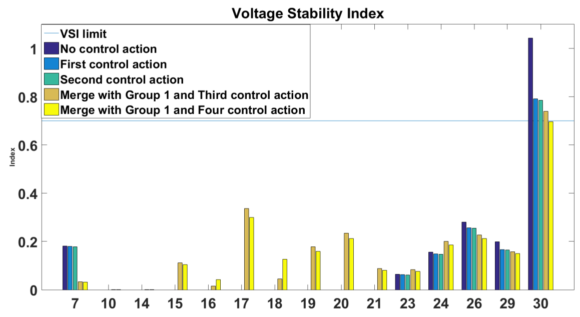
Fig. 13: Voltage Stabiltiy Indices for each group at increased load: case I



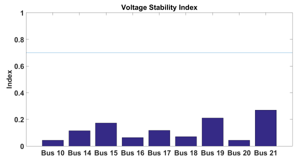
(a) Group 1



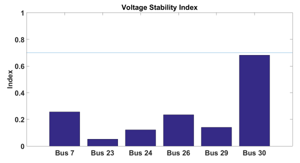
(b) Group 2



(c) Merging Group 1 and 3 with control actions on Group 1



(d) Group 1



(e) Group 3

Fig. 14: Voltage Stabiltiy Indices for each group at increased load: Case 2

to limit leading to stability problems. To manage the stability problem, the voltage stability monitoring and control needs to be performed in real time. Distributed voltage stability monitoring and control implemented using DCBlocks is one possible solution to manage voltage stability problem as secondary control option in conjunction with slow running centralized voltage stability application. Developed algorithm is scalable and computational time is expected to increases with number of devices and data points.

In future, DCBlocks needs to be extended to support dynamic group discovery using mechanism in adhoc mobile networks and implement newly identified supply agreement algorithm. Decentralized voltage stability will be implemented using physically distributed simulators in future. Additionally, DCBlocks need to be adopted for other possible decentralized power system applications.

ACKNOWLEDGMENT

Authors would like to thank US Department of Energy for Award Number DE-OE0000780, RTE-France and Power Systems Engineering Research center (PSERC) for partially supporting this work. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

REFERENCES

[1] S. Amin and B. Wollenberg, "Toward a smart grid: power delivery for the 21st century," *Power and Energy Magazine, IEEE*, vol. 3, no. 5, pp. 34–41, Sept 2005.

[2] E. Santacana, G. Rackliffe, L. Tang, and X. Feng, "Getting smart," *Power and Energy Magazine, IEEE*, vol. 8, no. 2, pp. 41–48, March 2010.

[3] J. Mirkovic, T. Benzel, T. Faber, R. Braden, J. Wroclawski, and S. Schwab, "The deter project: Advancing the science of cyber security experimentation and test," in *Technologies for Homeland Security (HST), 2010 IEEE International Conference on*, Nov 2010, pp. 1–7.

[4] H. Khoshkhoo and S. Shahrtash, "On-line dynamic voltage instability prediction based on decision tree supported by a wide-area measurement system," *Generation, Transmission Distribution, IET*, vol. 6, no. 11, pp. 1143–1152, November 2012.

[5] V. Ajjarapu and C. Christy, "The continuation power flow: a tool for steady state voltage stability analysis," *Power Systems, IEEE Transactions on*, vol. 7, no. 1, pp. 416–423, Feb 1992.

[6] C. Canizares and F. Alvarado, "Point of collapse and continuation methods for large ac/dc systems," *Power Systems, IEEE Transactions on*, vol. 8, no. 1, pp. 1–8, Feb 1993.

[7] B. Milosevic and M. Begovic, "Voltage-stability protection and control using a wide-area network of phasor measurements," *Power Systems, IEEE Transactions on*, vol. 18, no. 1, pp. 121–127, Feb 2003.

[8] S. Ghiocel and J. Chow, "A power flow method using a new bus type for computing steady-state voltage stability margins," *Power Systems, IEEE Transactions on*, vol. 29, no. 2, pp. 958–965, March 2014.

[9] R. Nuqui, A. Phadke, R. P. Schulz, and N. Bhatt, "Fast on-line voltage security monitoring using synchronized phasor measurements and decision trees," in *Power Engineering Society Winter Meeting, 2001. IEEE*, vol. 3, 2001, pp. 1347–1352 vol.3.

[10] P.-A. Lof, T. Smed, G. Andersson, and D. Hill, "Fast calculation of a voltage stability index," *Power Systems, IEEE Transactions on*, vol. 7, no. 1, pp. 54–64, Feb 1992.

[11] J. Wen, Q. Wu, D. Turner, S. Cheng, and J. Fitch, "Optimal coordinated voltage control for power system voltage stability," *Power Systems, IEEE Transactions on*, vol. 19, no. 2, pp. 1115–1122, May 2004.

[12] M. Glavic and T. Van Cutsem, "Wide-area detection of voltage instability from synchronized phasor measurements. part ii: Simulation results," *Power Systems, IEEE Transactions on*, vol. 24, no. 3, pp. 1417–1425, Aug 2009.

[13] H. Mehrjerdi, S. Lefebvre, M. Saad, and D. Asber, "A decentralized control of partitioned power networks for voltage regulation and prevention against disturbance propagation," *Power Systems, IEEE Transactions on*, vol. 28, no. 2, pp. 1461–1469, May 2013.

[14] ——, "Coordinated control strategy considering effect of neighborhood compensation for voltage improvement in transmission systems," *Power Systems, IEEE Transactions on*, vol. 28, no. 4, pp. 4507–4515, Nov 2013.

[15] J. Wen, Q. Wu, D. Turner, S. Cheng, and J. Fitch, "Optimal coordinated voltage control for power system voltage stability," *Power Systems, IEEE Transactions on*, vol. 19, no. 2, pp. 1115–1122, May 2004.

[16] S. Biswas, "Synchrophasor based voltage stablity monitoring and control of power systems," Ph.D. dissertation, Washington State University, 2014.

[17] S. S. Biswas, C. B. Vellaithurai, and A. K. Srivastava, "Development and real time implementation of a synchrophasor based fast voltage stability monitoring algorithm with consideration of load models," in *Industry Applications Society Annual Meeting, 2013 IEEE*, Oct 2013, pp. 1–9.

[18] Y. Gong, N. Schulz, and A. Guzman, "Synchrophasor-based real-time voltage stability index," in *Power Systems Conference and Exposition, 2006. PSCE '06. 2006 IEEE PES*, Oct 2006, pp. 1029–1036.

[19] P. Banerjee, S. Niddodi, H. Lee, A. Srivastava, and D. Bakken, "On the need for robust decentralized coordination to support emerging decentralized monitoring and control applications in electric power grid," in *Proceedings of the Fourth Grid of the Future Symposium, CIGRE*, Chicago, USA, Oct 2015, pp. 1–9.

[20] T. K. G. Coulouris, J. Dollimore and G. Blair., "Distributed systems: Concepts and design, 5ed." *Boston: Addison-Wesley*, 2011.

[21] N. L. M.J. Fischer and M. Paterson, "Impossibility of distributed consensus with one family faulty process," *Journal of the ACM*, no. 32(2), p. 374382, April 1985.

[22] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, no. 1, pp. 11–33, Jan 2004.

[23] L. Lamport, "Generalized consensus and paxos," Microsoft Research, Tech. Rep. MSR-TR-2005-33, March 2005. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=64631

[24] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, ser. USENIX ATC'14. Berkeley, CA, USA: USENIX Association, 2014, pp. 305–320. [Online]. Available: http://dl.acm.org/citation.cfm?id=2643634.2643666

[25] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Commun. ACM*, vol. 21, no. 7, pp. 558–565, Jul. 1978. [Online]. Available: http://doi.acm.org/10.1145/359545.359563

[26] K. Birman and T. Joseph, "Exploiting virtual synchrony in distributed systems," *SIGOPS Oper. Syst. Rev.*, vol. 21, no. 5, pp. 123–138, Nov. 1987. [Online]. Available: http://doi.acm.org/10.1145/37499.37515

[27] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982. [Online]. Available: http://doi.acm.org/10.1145/357172.357176

[28] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *J. ACM*, vol. 27, no. 2, pp. 228–234, Apr. 1980. [Online]. Available: http://doi.acm.org/10.1145/322186.322188

[29] Akka, "Akka documentation," 2015, [Akka 2.3.14 (current stable release) for Scala 2.10 / 2.11 and Java 6+]. [Online]. Available: http://akka.io/docs/

[30] M. Raynal, *Fault-tolerant Agreement in Synchronous Message-passing Systems*, 1st ed. USA: Morgan and Claypool Publishers, 2010.

[31] DETERLab, "Deter project website," 2015. [Online]. Available: http://deter-project.org/

[32] R. Goodfellow, R. Braden, T. Benzel, and D. E. Bakken, "First steps toward scientific cyber-security experimentation in wide-area cyber-physical systems," in *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, ser. CSIIRW '13. New York, NY, USA: ACM, 2013, pp. 39:1–39:4. [Online]. Available: http://doi.acm.org/10.1145/2459976.2460021

[33] U. of Washington. (1993) Power svstems test case archive. [Online]. Available: http://www.ee.washington.edulresearch/pstca