# Resilient Autonomous Wind Farms

## Preprint

Aaron Barker, Benjamin Anderson, and Jennifer King

*National Renewable Energy Laboratory*

*Presented at the 2020 American Control Conference (ACC)*
*July 1–3, 2020*

**NREL is a national laboratory of the U.S. Department of Energy**
**Office of Energy Efficiency & Renewable Energy**
**Operated by the Alliance for Sustainable Energy, LLC**

This report is available at no cost from the National Renewable Energy Laboratory (NREL) at www.nrel.gov/publications.

Contract No. DE-AC36-08GO28308

**Conference Paper**
NREL/CP-5000-75998
July 2020

# Resilient Autonomous Wind Farms

## Preprint

Aaron Barker, Benjamin Anderson, and Jennifer King

*National Renewable Energy Laboratory*

**Suggested Citation**
Barker, Aaron, Benjamin Anderson, Jennifer King. 2020. *Resilient Autonomous Wind Farms: Preprint*. Golden, CO: National Renewable Energy Laboratory. NREL/CP-5000-75998. https://www.nrel.gov/docs/fy20osti/75998.pdf.

**NREL is a national laboratory of the U.S. Department of Energy**
**Office of Energy Efficiency & Renewable Energy**
**Operated by the Alliance for Sustainable Energy, LLC**

This report is available at no cost from the National Renewable Energy Laboratory (NREL) at www.nrel.gov/publications.

Contract No. DE-AC36-08GO28308

**Conference Paper**
NREL/CP-5000-75998
July 2020

National Renewable Energy Laboratory
15013 Denver West Parkway
Golden, CO 80401
303-275-3000 • www.nrel.gov

**NOTICE**

This work was authored [in part] by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. This work was supported by the Laboratory Directed Research and Development (LDRD) Program at NREL. The views expressed herein do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.



*Cover Photos by Dennis Schroeder: (clockwise, left to right) NREL 51934, NREL 45897, NREL 42160, NREL 45891, NREL 48097, NREL 46526.*

NREL prints on paper that contains recycled content.

# Resilient Autonomous Wind Farms

Aaron Barker, Benjamin Anderson, Jennifer King

***Abstract*— With the advent of an increasing number of control strategies that seek to optimize wind turbine performance on a farm level, taking into account individual wind turbine information to achieve wind-farm-level objectives has become an increasingly important goal. Methods for controlling wind turbines on an individual and farm level have experienced significant development, and an abundance of new implementations for gathering and using data from turbines have created potential for novel control mechanisms that can further optimize the performance and delivery characteristics of a wind farm. A key element of making these wind farms more efficient is to develop reliable algorithms that use local sensor information that is already being collected, such as from local meteorological stations, nearby radars, sodars, and lidars, and supervisory control and data acquisition (SCADA) data. Making use of information from all wind turbines in a wind farm can enable such approaches as determining the atmospheric conditions across the farm, improving fault-finding, and ensuring more efficient overall control of farmwide optimizations through mechanisms such as wake steering. However, these approaches typically involve a centralized communications and control center. In order to ensure the resilient operation of the farm, it is necessary to develop an approach that distributes the calculation and communication amongst multiple nodes throughout the farm. In this fashion, a redundant, robust, and secure network can be created, which can tolerate faults in calculation, communication, and even external attacks that seek to disrupt the operation of the wind farm. This paper introduces the use of the Raft-Byzantine-Fault-Tolerant algorithm in the implementation of autonomous control of a wind farm. This implementation will allow for fault tolerance for malfunctioning nodes, sensors, transmitters, and connectors. This approach is equally extensible to account for malicious actors. It will be shown to achieve overall consensus, provided the number of faults/malicious nodes is less than $3n$+1, where $n$ is the number of turbine cluster faults that may occur, and to be robust in the face of multiple arbitrary faults.**

## I. Introduction

As the percentage of energy contributed to the grid by renewables continues to grow rapidly, the grid of the future will need to ensure that it remains reliable, robust, and resilient to attacks that may seek to take it offline or interrupt its operation. Resilience is becoming increasingly important to renewable energy deployment and it involves having an understanding of the areas that encompass failure modes from common devices, sensor and transmission failures, and deliberate and malicious threats that may seek to disrupt the operation of a wind farm [1], [2]. Changes to the grid structure and energy markets mean that soon it is likely that wind, solar, and other renewable energy technologies will be paid to provide ancillary grid services, further increasing the importance of resilience. Energy security is increasingly looked at as an important aspect of the nation's generation/electricity policy [3].

In this context, methods for controlling wind turbines on an individual and farm level have experienced increasing development. An abundance of new implementations for gathering data from turbines has created potential for novel control mechanisms that can further optimize performance and delivery characteristics of a wind farm [4]. In addition, as the cost of wind turbines continues to drop, and as they become more power dense, the way in which sites are built for optimal financial return is also changing. Turbines can be spaced closer together to reduce associated Balance-Of-Station costs [5] [6]. Drawbacks from this come in the form of increased wake effects downstream of the turbine. This closer spacing can also reduce the overall lifetime of the turbines. Advanced control methods, such as wake steering [7]–[10], can be used to mitigate this impact, but they rely on having a very accurate and consistent understanding of the wind direction to operate efficiently. However, these advanced control methods need access to wind direction and speed measurements taken by the turbines, potentially exposing them to faults and malicious threats.

Furthermore, wind farms are moving toward operating autonomously through these advanced optimization and control algorithms. Consensus-based algorithms allow turbines to share information with each other about the conditions they are experiencing [11]. This has been investigated using turbine supervisory control and data acquisition (SCADA) data to reach a consensus of the wind direction experienced at each turbine using the Alternating Direction Method of Multipliers (ADMM) [12], [13], as described in Section II-A. This method has shown promise and allows turbines to reach consensus on wind direction while being tolerant of some faults. However, to further develop these methods, and for more robust, reliable, and resilient results, this approach needs to tolerate a greater range of faults. These faults include sensor errors, connectivity issues, damage, and interruption to operation. Irrespective of the source of the fault, consensus should still be able to be reached quickly to

A. Barker is with the National Renewable Energy Laboratory; email: aaron.barker@nrel.gov.

B. Anderson is with the National Renewable Energy Laboratory; email: benjamin.anderson@nrel.gov.

J. King is with the National Renewable Energy Laboratory; email: jennifer.king@nrel.gov.

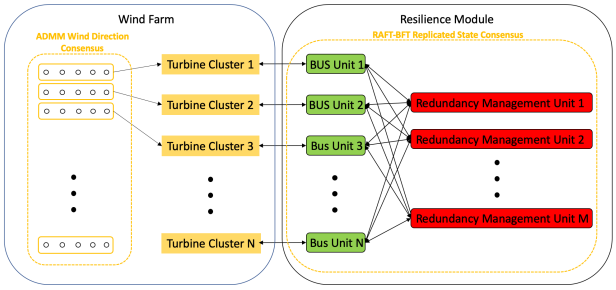


Fig. 1: Interaction between wind direction consensus and Raft-Byzantine Fault Tolerant checking and replication of reported values. Consensus is performed at the wind farm level and the results are compared within generals (i.e., the nodes appointed as leaders of the clusters) to determine the accuracy of the data received.

maximize the utility of using the wind farm network in this distributed consensus fashion.

The work that follows will outline the framework by which we can implement a Byzantine Fault Tolerant (Raft-BFT) algorithm within a wind farm network, using the ADMM approach to determine wind direction consensus across all turbines in the farm (shown in Section II). This approach will be demonstrated to achieve distributed consensus on wind direction amongst different nodes within the wind farm cluster (Section III), and to provide fault tolerance against any arbitrary sensor, communication, calculation, or malicious fault within the system. The implementation of this algorithm will significantly increase wind farm resilience and bring us closer to a wind farm that is capable of operating autonomously. At the same time, the algorithm will dramatically increase fault tolerance to fail-stop faults and improve information on likely wind direction by leveraging information from neighboring clusters of turbines. Results are shown in Section IV. Finally, conclusions and future directions for this work are presented in Section V.

## II. Problem Formulation

This paper integrates wind direction consensus using ADMM with a variant of Raft-BFT [14], [15]. Figure 1 shows the interaction between the two algorithms. The left block (described in more detail in Section II-A) determines the wind direction across the wind farm and the right block determines the accuracy of the communicated information (described in detail in Section II-B).

### A. Wind Direction Consensus

To determine the wind direction across the wind farm, as shown in the left block of Figure 1, we apply a consensus-based approach that uses a network to reliably determine the wind direction at every turbine, considering both the turbine's measurement and those of the wind farm. Currently, turbines typically rely on wind vanes and anemometers mounted on the back of the nacelle to provide measurements to their controllers. Individual measurements, on their own, can be unreliable because of the complex flow created as the wind passes through the rotor, preventing accurate measurements and thus inputs into the individual turbine yaw controller.

In this paper, the SCADA measurements recorded at each turbine are used to determine an estimate of the wind direction at every turbine, as is done in [11]. The turbine receives wind direction measurements from every other turbine. In future work, only measurements from the nearby neighbors will be included.

*1) Node and Edge Objective Functions:* Each turbine uses its own wind direction measurement, as well as the wind direction measurement from the connected turbines (all other turbines in this case) to determine its local wind direction. The objective of the individual turbine, $i$, is to minimize the error between the wind direction measurement at turbine $i$ and the estimated wind direction. The edge objective incorporates information from each turbine to ensure a reliable estimate of the wind direction at an individual turbine. The optimization problem can be written as:

$$\underset{x}{\text{minimize}} \quad \underbrace{\sum_{i\in\mathcal{V}}^{N} (x_{i,measure} - x_i)^2 + \alpha|b_i|}_{\text{node objective}} + \underbrace{\sum_{(j,k)\in\mathcal{E}} w_{jk}|x_j - x_k|}_{\text{edge objective}} \tag{1}$$

where $N$ is the number of turbines, $x_{i,measure}$ is the wind direction measurement recorded at the turbine $i$, $b_i$ is the measurement bias in the wind direction with respect to true north, $w_{jk}$ is a weight placed on the connection between turbines $j$ and $k$, $x_j$ is the estimated wind direction at turbine $j$, and $x_k$ is the estimated wind direction at turbine $k$. In this case, the node objective function is convex and can be updated with a closed-form solution. The edge objective minimizes the differences in estimated wind direction between neighboring turbines. The weights, $w_{jk}$, are set based on the distance to the turbine.

*2) Alternating Direction Method of Multipliers:* To solve the objective function posed in the previous section, we use the ADMM approach. This algorithm is particularly useful in this case, as each individual turbine can solve its own optimization in parallel, communicate the solution to neighboring turbines, and iterate this process until each node within the wind farm network has converged. In this paper, each turbine determines the local wind direction at each individual turbine by using information from all turbines in the wind farm. ADMM is used to solve a network

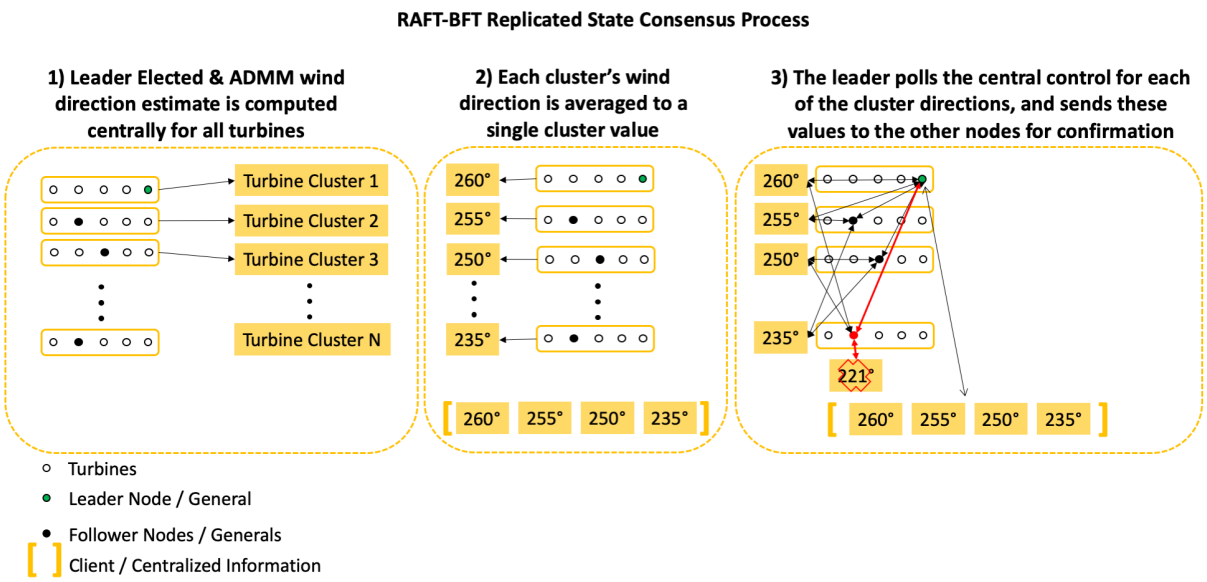


Fig. 2: Raft-BFT: 1) Process of leader and follower election, 2) ADMM computation of consensus wind direction, and 3) Raft-BFT checking and replication of "true" wind values

optimization with connecting nodes, such that:

$$\underset{x_i,z_i}{\text{minimize}} \quad \sum_{i}^{N}(x_{i,measure}-x_i)^2 + \lambda \sum_{(j,k)\in\mathcal{E}} w_{j,k}\|z_{jk}-z_{kj}\|_2 \tag{2}$$

$$\text{subject to:} \quad x_i = z_{ij}, \quad j \in N(i) \tag{3}$$

where $z_{jk}$ is a copy of $x_j$ at turbine $k$, such that the wind farm reaches an "almost" consensus of the wind direction across the wind farm.

The optimization problem is solved by minimizing the augmented Lagrangian:

$$\mathcal{L}_\rho(x,b,z,u) = \sum_{i\in\mathcal{V}} f_i(x_i,b_i) + \sum_{(j,k)\in\mathcal{E}} \lambda w_{jk}\|z_{jk}-z_{kj}\|_2 + \frac{\rho}{2}\left(\|u_{jk}\|_2^2 + \|u_{kj}\|_2^2\right) + \frac{\rho}{2}\left(\|x_j+z_{jk}+u_{jk}\|_2^2 + \|x_k-z_{kj}+u_{kj}\|_2^2\right) \tag{4}$$

where $u$ is the scaled dual variable and $\rho > 0$ is the penalty parameter. The following steps are used in an iterative way to solve for $x$, $z$, $u$:

$$x^{k+1} = \underset{x}{\text{argmin}} \quad \mathcal{L}_\rho(x, z^k, u^k) \tag{5}$$

$$z^{k+1} = \underset{z}{\text{argmin}} \quad \mathcal{L}_\rho(x^{k+1}, z, u^k) \tag{6}$$

$$u^{k+1} = u^k + (x^{k+1} - z^{k+1}) \tag{7}$$

The variables $x$, $b$, $z$, and $u$ are updated in serial. In this setup, the bias, $b_i$, does not have a $z$ or $u$ update step because the biases are only known to the individual turbines (i.e., they are not communicated to nearby turbines).

For the wind farm example, turbines near each other should have similar wind direction estimates. There are two penalty parameters, $\lambda$ and $\rho$, that can be used to weight an individual turbine's measurement against the measurements of the connected turbines. A small $\lambda$ corresponds to almost no consensus among the turbines. A large $\lambda$ corresponds to total consensus among the turbines. The value of $\lambda$ can change based on how accurate/reliable another turbine's information is perceived to be. In this paper, $\lambda$ is held constant. However, in future work, $\lambda$ will be updated based on the results of the Byzantine Fault-Tolerant step (featured in the right block in Figure 1).

### *B. Byzantine Fault Tolerance*

Next, this process uses Byzantine Fault Tolerance to provide a layer of resiliency to the wind farm. In particular, Byzantine Fault Tolerance is the characteristic defining a system that tolerates the class of failures that belong to the Byzantine Generals problem [16]. A Byzantine fault presents different symptoms to different observers. These are faults that have no restrictions and can have any kind of arbitrary value, which they deliver to other nodes while posing as an honest actor. These faults are particularly severe and difficult to deal with, and many systems that require extreme robustness (particularly networks with large numbers of sensors) will require Byzantine Fault Tolerance to operate correctly in all cases. The algorithm discussed above is Byzantine Fault Tolerant, provided the number of bad actors does not exceed 1/3 of the nodes.

A Byzantine failure, on the other hand, occurs when there is a loss of a system service because of a Byzantine fault in systems that require consensus. For a system to exhibit a Byzantine failure, its operation must require consensus. Many distributed systems have an implied system-

level consensus requirement, such as a synchronized timing aspect, which requires consensus for coordinated operation. A message transmitted slightly too early, for example, would be accepted only by the nodes of the system that have slightly faster clocks, or whose values matched that of the node sending the early message.

*1) Raft-BFT:* The Byzantine Fault Tolerant method used in this paper is known as Raft-BFT, a modification of Raft in which cryptographic hashes are used to generate a fingerprint for the actual data/messages. Raft is a consensus algorithm for managing a replicated log and a subset of Byzantine Fault Tolerance. Raft bears a lot of similarities to other consensus algorithms, but a number of novel features have motivated the decision to use Raft, and subsequently Raft-BFT for the work in this paper.

Raft uses a stronger form of leadership than other consensus algorithms. In Raft's implementation, log entries only flow from the leader to other servers. This simplifies the management of the replicated log. Traditional Raft uses randomized timers to elect leaders, which helps resolve new leadership election conflicts simply and rapidly. Raft additionally features a robust mechanism to handle changing or increasing the number of servers a cluster uses. This mechanism uses a joint consensus approach, where the majorities of two different configurations overlap during transitions. This overlap allows the cluster to continue operating normally during configuration changes.

These measures ensure safety and further enhance resiliency in wind turbine operation. In addition to this, Raft is simpler and more understandable than competing algorithms, with a description available that enables full implementation. There are many open-source implementations available with efficiency comparable to other algorithms, and more importantly, its safety properties have been formally specified and proven.

The implementation of the Raft-BFT algorithm also ensures safety under all non-Byzantine conditions, including network delays, partitions, packet loss, duplication, and reordering. They are fully functional (available), as long as any majority of the servers are operational and can communicate with each other and with clients. Thus, a typical cluster of five servers can tolerate the failure of any two servers. Servers are assumed to fail by stopping; they may later recover from a stored state on stable storage and rejoin the cluster.

Though Raft is adept at handling fail-stop bugs, it does not guarantee full resilience, as systems in a distributed system can exhibit arbitrary behavior that makes them vulnerable to bugs, hardware faults, and malicious faults. In order to give higher guarantees on correctness, its desirable to have a Byzantine Fault Tolerant system. In the Byzantine Fault Tolerant version of Raft, The use of hashing allows nodes to verify their agreement on a certain value.

In Raft-BFT, nodes transmit a hash value instead of the actual data chunk until the transaction is to be committed, which reduces message overhead for transactions that do not proceed. Hashing is additionally used to verify the consistency of the local states at different nodes, thus ensuring that transactions are processed in the same order across them. Additional measures are taken to make the Raft leader election process fault-tolerant. Whereas in Raft, a faulty node could ignore the time-out and trigger the leader election, and two collaborating nodes could exploit this by then switching leaders back and forth; in Raft-BFT, the leader selection algorithm from Practical Byzantine Fault Tolerance (PBFT) is adopted. In this rigged leader election, which happens after a fixed time-out per node, each node votes in a way that increases the possibility of consensus forming.

Figure 2 provides a detailed view of what the Raft algorithm does to provide resiliency to the wind farm. Raft implements consensus by first electing a leader, then giving the leader complete responsibility for managing the replicated log. The leader accepts log entries from clients, replicates them on other servers, and tells servers when it is safe to apply log entries to their state machines. A Raft cluster contains several servers, typically five, which allows the system to tolerate two failures. Each server can be in one of three states: leader, follower, or candidate. In normal operation, there is one leader and all the other servers are followers. The followers are passive, not issuing requests on their own, but simply responding to requests from the leaders and candidates. The leader handles all client requests. When servers start up, they begin as followers. A server will remain in the follower state as long as it receives a valid Remote Procedure Call (RPC) from a leader or candidate. Leaders send periodic heartbeats to all followers to retain their authority. In the candidate state, a new leader is elected. If a follower receives no communication over a period of time, called the election time-out, then it assumes there is no viable leader and begins a new election.

Having a leader simplifies the management of the replicated log. For example, the leader can decide where to place new entries in the log without consulting other servers, and data can flow in a simple fashion from the leader to other servers. A leader can fail or become disconnected from the other servers, in which case a new leader is elected. In summary, Raft decomposes the consensus problem into three relatively independent subproblems:

- Leader election
- Log replication
- State machine safety.

## III. Raft-ADMM Applied to Wind Farms

The Raft-ADMM approach was demonstrated using a 25-turbine setup, shown in Figure 3. In this case, the wind direction was coming from the west, i.e., 270° and the turbines were spaced 500 m apart in the streamwise and spanwise directions.

The wind turbines are first split into five clusters, selected based on the wind direction (i.e., the turbines are clustered along the rows, as shown in Figure 3). The clustering has two primary motivations. First, the turbines are clustered such that their information and/or behavior is influenced primarily by their neighboring turbines, which are most likely to share

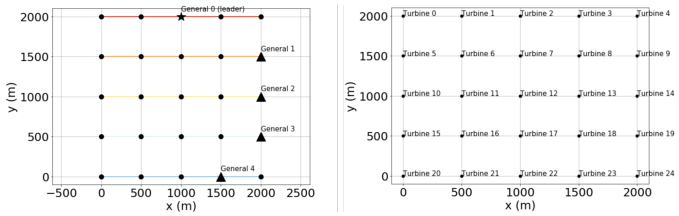


Fig. 3: Simulation setup with 25 turbines split into 5 clusters. The wind direction is assumed to be coming from the west.

a characteristic wind direction. Second, the clustering allows for a reduction in communication and overhead for the Raft-BFT algorithm in operation. Again, Figure 1 shows a graphical representation of the communication interactions between the layers generated by Raft-ADMM, known as classical full exchange. This method requires each bus unit to take values from general purpose nodes (i.e., turbines) and send their values to all the redundancy units (nodes/generals). The values at each turbine are used by a central client to compute the consensus wind direction at all turbines. These wind direction values are then averaged across the clusters of turbines. The leader node takes these cluster average values and propagates them out to the nodes, which also run the ADMM wind direction consensus calculation for each turbine, and report their own averaged cluster directions. Each node checks the transaction hash of the message sent to it by the leader, verifying that the information the leader wishes to add to the log is the same as what they have calculated. If a quorum of responses matching the leader's value is received, consensus on the cluster wind direction values is achieved, and the values are stored to a log.

## IV. Results

### A. Practical Test Cases for Raft-BFT/ADMM Algorithm

These Byzantine faults are not merely theoretical, but have been shown to occur in real-world usage, and occur with a frequency far higher than what might typically be expected [17]. The following test cases show how errors can pass checks that would typically be expected to contain faults within supposedly redundant systems.

The base assumption is that all nodes will receive the same data, as they are connected to the same source; and that they will propagate these values to the leader. However, Byzantine fault propagation may invalidate these system failure assumptions. There are essentially four possible overarching manifestations of the results (with some nuance in how these are handled by the algorithm):

- Message arrived on time with a good checksum
- Message arrived on time with a bad checksum
- Message arrived late with a good checksum
- Message arrived late with a bad checksum.

*1) General (Node) Fault/Malicious General/Outside Actor:* In the case of a single node fault (as shown in Figure 4), a single general reports either intentionally or unintentionally incorrect information for any of the wind turbine cluster's wind directions. In the former scenario, the outside party will not have access to the private key required to sign the transaction being returned to the leader, and the information will be rejected. In the latter scenario, the hash value of the data will not match that agreed upon by the node and the leader in the pre-append phase. As a result, the transaction hashes sent by the leader containing the "true" wind direction information are not mirrored by the general with false information. This results in an invalid transaction hash, and the result from this node is not added to the replicated state log.

*2) Delayed Response from General:* In the case of delayed transmission by a node (as shown in Figure 5), the algorithm will time out. If a quorum of results has been reached from all nodes, this will not impact the ability to reach consensus. However, if a sufficient number of node transmissions are delayed or lost to prevent consensus being reached, the consensus process will fail.

*3) Delayed Transmission + Fault:* In the case of delayed transmission by a node coupled with a fault in the values returned by another general (as shown in Figure 6), the algorithm will not be able to reach consensus.

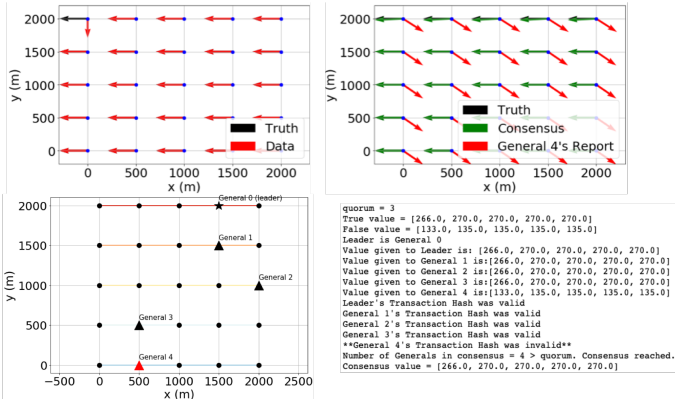


Fig. 4: Consensus is reached despite a fault at a single General, General 4, which contained the incorrect values for cluster wind direction—resulting in an invalid transaction hash sent to the leader.

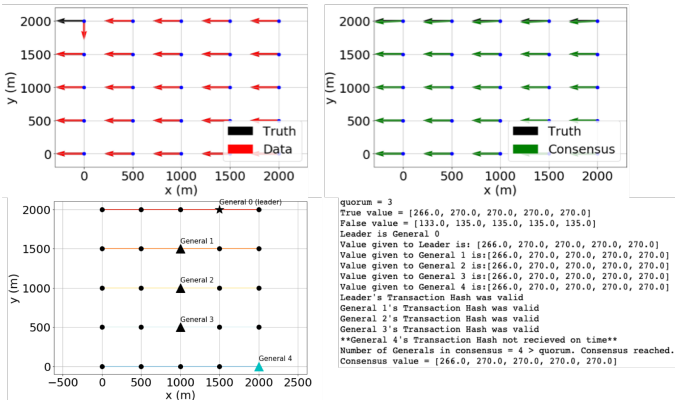


Fig. 5: Consensus is reached despite a delayed response from a single General (General 4), as 4 correct responses are sufficient to reach a quorum.

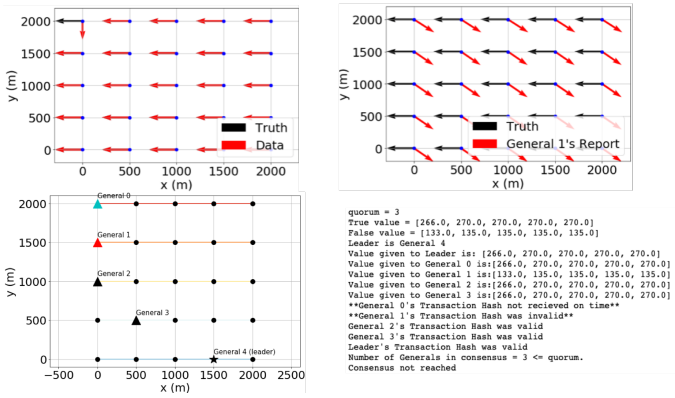


Fig. 6: Consensus is not reached, with both a fault and a delayed response preventing the algorithm from reaching the number of correct responses required to reach consensus.

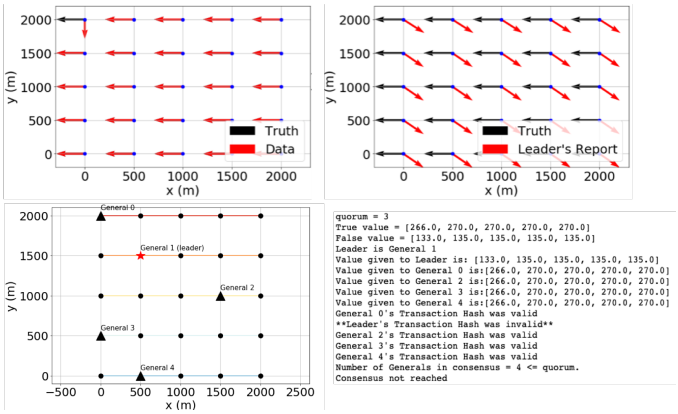


Fig. 7: Consensus is not reached, because of a fault with the leader itself.

*4) Leader Fault:* The Raft-BFT algorithm relies heavily on the leader to manage the state of the replicated log. While this provides a number of advantages, as previously discussed, it does make consensus particularly reliant on the values reported by the leader. In the case that the leader reports incorrect values (shown in Figure 7), no consensus can be reached—and a new leader is elected to take their place.

## V. Conclusions

This paper has demonstrated the implementation of the Raft-BFT algorithm to enable distributed fault-tolerant operation of a wind farm; a crucial step in enabling full autonomous control and operation. The robustness of this approach to tolerating malfunctioning nodes, sensors, transmitters, connectors, and arbitrary classes of faults has been explored, explained, and demonstrated; with faults classified by their observable impact and subsequently tested. The algorithm implemented has proven its ability to achieve overall consensus, provided the number of reliable nodes is greater than $3n$+1 (where $n$ is the number of turbine cluster faults that may occur). As part of the development of this approach, a large number of consensus algorithms have been investigated and trialed, with the reason for the ultimate selection of the Raft-BFT algorithm outlined.

The BFT method has additionally been layered with ADMM, enabling the determination of wind farm direction consensus, which, in itself, is robust and adaptive to faults. The result is a crucial step toward enabling autonomous and resilient operation of wind farms that is resistant to cyberattacks and malicious actors, thereby acting to secure the nation's energy future.

Future work will explore several different topologies, with an eye toward practical and implementable ways of actually applying this methodology to a real farm. In addition, future work will examine different adaptations of the ADMM/Raft-BFT methodology, which will help to further improve the robustness, utility, and applicability of the consensus process for wind farms, including:

- Implementing ADMM at the cluster level, rather than across the entire farm.
- Implementing ADMM at the cluster level, determining consensus amongst clusters, then implementing ADMM again between clusters for the farm. In this fashion, and combined with the Scalable Processor-Independent Design for Electromagnetic Resilience (SPIDER)/Reliable Optical Bus (ROBUS) topology, clusters can be isolated from each other, which is a further step in advancing robust operation.
- Performing consensus calculation within a cluster if it repeatedly fails in the overall cluster consensus calculation. In the current approach, the general can be changed when consensus is not reached in any case, but this method would offer a method of diagnosing the root cause of persistent problems in cluster consensus.
- Modifying the existing algorithm to feed a penalization value back to the ADMM calculation for clusters that routinely do not reach consensus. This approach will remove the need to manually select penalization values for the ADMM calculation.

Other BFT algorithms will be explored, some of which can handle nondeterminism, such as SIEVE [18], which uses a modular approach and filters out nondeterministic operations in an application; ensuring that all correct processes produce the same outputs and that their internal states do not diverge.

In summary, this paper outlines the framework for distributed consensus amongst wind turbines, and moves toward cooperative, robust, and redundant autonomous wind farms.

## References

[1] M. Amin, "Challenges in reliability, security, efficiency, and resilience of energy infrastructure: Toward smart self-healing electric power grid," in *2008 IEEE Power and energy society general meeting-conversion and delivery of electrical energy in the 21st century*. IEEE, 2008, pp. 1–5.

[2] P. E. Roege, Z. A. Collier, J. Mancillas, J. A. McDonagh, and I. Linkov, "Metrics for energy resilience," *Energy Policy*, vol. 72, pp. 249–256, 2014.

[3] B. Unel and A. Zevin, "Toward resilience: Defining, measuring, and monetizing resilience in the electricity system." Institute for Policy Integrity, New York University School of Law, Wilf Hall, 139 MacDougal Street, New York, New York 10012, 2018.

[4] S. Boersma, B. Doekemeijer, P. M. Gebraad, P. A. Fleming, J. Annoni, A. K. Scholbrock, J. Frederik, and J.-W. van Wingerden, "A tutorial on control-oriented modeling and control of wind farms," in *American Control Conference (ACC), 2017*. IEEE, 2017, pp. 1–18.

[5] P. A. Fleming, A. Ning, P. M. Gebraad, and K. Dykes, "Wind plant system engineering through optimization of layout and yaw control," *Wind Energy*, vol. 19, no. 2, pp. 329–344, 2016.

[6] A. P. Stanley, J. Thomas, A. Ning, J. Annoni, K. Dykes, and P. Fleming, "Gradient-based optimization of wind farms with different turbine heights," in *35th Wind Energy Symposium*, 2017, p. 1619.

[7] P. A. Fleming, P. M. Gebraad, S. Lee, J.-W. van Wingerden, K. Johnson, M. Churchfield, J. Michalakes, P. Spalart, and P. Moriarty, "Evaluating techniques for redirecting turbine wakes using sowfa," *Renewable Energy*, vol. 70, pp. 211–218, 2014.

[8] P. Gebraad, F. Teeuwisse, J. Wingerden, P. A. Fleming, S. Ruben, J. Marden, and L. Pao, "Wind plant power optimization through yaw control using a parametric model for wake effects a cfd simulation study," *Wind Energy*, vol. 19, no. 1, pp. 95–114, 2016.

[9] M. Bastankhah and F. Porté-Agel, "Experimental and theoretical study of wind turbine wakes in yawed conditions," *Journal of Fluid Mechanics*, vol. 806, pp. 506–541, 2016.

[10] M. F. Howland, J. Bossuyt, L. A. Martínez-Tossas, J. Meyers, and C. Meneveau, "Wake structure in actuator disk models of wind turbines in yaw under uniform inflow conditions," *Journal of Renewable and Sustainable Energy*, vol. 8, no. 4, p. 043301, 2016.

[11] J. R. Annoni, C. Bay, K. E. Johnson, E. Dall'Anese, E. W. Quon, T. W. Kemper, and P. A. Fleming, "Wind direction estimation using scada data with consensus-based optimization," *Wind Energy Science (Online)*, vol. 4, no. NREL/JA-5000-74366, 2019.

[12] D. Hallac, J. Leskovec, and S. Boyd, "Network lasso: Clustering and optimization in large graphs," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2015, pp. 387–396.

[13] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[14] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, 2014, pp. 305–319.

[15] J. Clow and Z. Jiang, "A byzantine fault tolerant raft."

[16] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.

[17] K. Driscoll, B. Hall, M. Paulitsch, P. Zumsteg, and H. Sivencrona, "The real byzantine generals," in *The 23rd Digital Avionics Systems Conference (IEEE Cat. No.04CH37576)*, vol. 2, Oct 2004, pp. 6.D.4–61.

[18] C. Cachin, S. Schubert, and M. Vukolic, "Non-determinism in byzantine fault-tolerant replication," *CoRR*, vol. abs/1603.07351, 2016. [Online]. Available: http://arxiv.org/abs/1603.07351