Byzantine Fault Tolerance for Electric Power Grid Monitoring and Control*

Wenbing Zhao and F. Eugenio Villaseca
Department of Electrical and Computer Engineering
Cleveland State University, 2121 Euclid Ave, Cleveland, OH 44115
wenbing@ieee.org and f.villaseca@csuohio.edu

Abstract

The stability of the electric power grid is crucial to every nation's security and well-being. As revealed by a number of large-scale blackout incidents in North America, the data communication infrastructure for power grid is in urgent need of transformation to modern technology. It has been shown by extensive research work that such blackout could have been avoided if there were more prompt information sharing and coordination among the power grid monitoring and control systems. In this paper, we point out the need for Byzantine fault tolerance and investigate the feasibility of applying Byzantine fault tolerance technology to ensure high degree of reliability and security of power grid monitoring and control. Our empirical study demonstrated that Byzantine fault tolerant monitoring and control can easily sustain the 60Hz sampling rate needed for Supervisory Control And Data Acquisition (SCADA) operations with sub-millisecond response time under the local-area network environment. Byzantine fault tolerant monitoring and control is also feasible under the wide-area network environment for power grid applications that demand sub-second reaction time.

Keywords: Byzantine Fault Tolerance, Intrusion Tolerance, Security, Fault Tolerance Middleware, Electric Power Grid Monitoring and Control

1. Introduction

The current data communication infrastructure for electric power grid was developed several decades ago when the industry was heavily regulated. It has been well recognized in recent years that the data communication infrastructure is in urgent need of transformation to modern computer networking and distributed computing technologies. First, with the recent deregulation, many independent parties could enter the utility industry by offering alternative channels for electric power generation, distribution and trade. This demands timely, reliable and secure information exchange among these parties [3]. Second, the current data communication infrastructure lacks the support for large-scale real-time coordination among different electric power grid health monitoring and control systems, which could have prevented the 2003 massive blackout incident in North America [2]. Third, the use of modern networking technology could also revolutionize the everyday electric power grid operations, as shown by the huge benefits of substation automation and the use of Phasor Measurement Units (PMUs) for electric power grid health monitoring [11].

However, the openness and the ease of information sharing and cooperation brought by the infrastructure transformation also increased the likelihood of cyber attacks to the electric power grid, as demonstrated recently by an experiment conducted by the US Department of Energy's Idaho Lab [5]. To address such vulnerability, intrusion detection and intrusion tolerance techniques must be used to enhance the current and future data communication infrastructure for the power grid. Byzantine fault tolerance is a fundamental technique to achieve the objective [4, 9].

In this paper, we focus our discussions on the security and reliability of electric power grid health monitoring and control. We elaborate in detail the need for Byzantine fault tolerance and the challenges of applying Byzantine fault tolerance into this problem domain. In particular, we investigate experimentally the feasibility of using such sophisticated technology to meet potentially very stringent real-time requirement for electric power grid health monitoring and control, while ensuring high degree of reliability and security of the system.

2. The Need for Byzantine Fault Tolerance

A Byzantine faulty process may behave arbitrarily. In particular, it may disseminate different information to other processes, which constitutes a serious threat to the integrity of a system [9]. Because a Byzantine faulty process may also choose not to send a message, or refuse to respond

*This work was supported in part by Department of Energy Contract DE-FC26-06NT42853, and by a Faculty Research Development award from Cleveland State University.

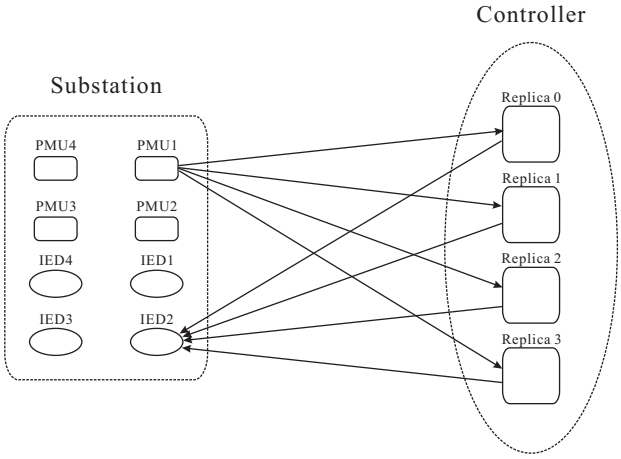

Figure 1. The interaction of substation devices and the replicated controller.

to requests, it can exhibit crash fault behavior as well (i.e., crash faults can be considered as a special case of Byzantine faults). Consider the scenario that multiple PMUs periodically report their measurement results to a controller for power grid health monitoring. Periodically, or when the controller detects an abnormal situation, it may wish to intervene with specific control instructions to the actuating devices (such as Intelligent Electronic Devices, i.e., IEDs [6]) located in the same substation as those PMUs to alleviate the problem. Due to the critical role played by the controller, it must be replicated to ensure high availability. Otherwise, the controller would constitute a single-point of failure. The main components and their interactions are illustrated in Figure 1.

However, under cyber attacks, the controller replicas, the PMUs, and the IEDs, might be compromised. As a result, a Byzantine faulty PMU could potentially send different data to different controller replicas, and a compromised controller replica could send conflicting commands to different IEDs. Without proper coordination among the controller replicas, the state of the replicas might diverge in the former case, which would lead to inconsistent decisions among the replicas. Without a sound mechanism at each IED, a malicious command might be executed in the latter case, which might lead to the destruction of a generator or a transmission line, as shown in the CNN report [5].

Byzantine fault tolerance (BFT) refers to the capability of a system to tolerate Byzantine faults [9]. If BFT is used, the cyber attacks illustrated above could be defeated provided that the number of compromised controller replicas, f, is below a threshold, and the number of correct (i.e., non-compromised) PMUs and IEDs are sufficient for the normal operation of the substation. For the client-server system shown in Figure 1, BFT can be achieved by using $3f + 1$ replicas (to tolerate up to f faulty replicas) and by ensuring all correct replicas to execute the same set of requests in the same order. The latter means that the server replicas must reach an agreement on the set of requests and their relative ordering despite Byzantine faulty replicas and clients. Such an agreement is often referred to as a Byzantine agreement [9]. The Byzantine agreement among the replicas ensures that a faulty client (i.e., a PMU) cannot cause the divergence of the state of correct controller replicas. Furthermore, a Byzantine agreement must be reached among all correct replicas on each command issued by the controller for reasons to be explained in the next section. Before an IED can accept the command, it must wait until it has collected at least $f + 1$ identical command from different replicas.

3. Byzantine Fault Tolerance Mechanisms

In this work, we choose to use a well-known Byzantine agreement algorithm developed by Castro and Liskov [4]. The BFT algorithm is designed to support client-server applications running in an asynchronous distributed environment with the Byzantine fault model. The implementation of the algorithm contains two parts. At the client side, a lightweight library is responsible to send the client's request to the primary replica, to retransmit the request to all server replicas on the expiration of a retransmission timer (to deal with the primary faults), and to collect and vote on the corresponding replies. The main BFT algorithm is executed at the server side by a set of $3f + 1$ replicas to tolerate up to f faulty replicas. One of the replicas is designated as the primary while the rest are backups.

The normal operation of the (server-side) BFT algorithm involves three phases. During the first phase (referred to as the pre-prepare phase), the primary multicasts a pre-prepare message containing the client's request, the current view number and a sequence number assigned to the request to all backups. A backup verifies the request message and the ordering information. If the backup accepts the message, it multicasts to all other replicas a prepare message containing the ordering information and the digest of the request being ordered. This starts the second phase, i.e., the prepare phase. A replica waits until it has collected $2f$ prepare messages from different replicas (including the message it has sent if it is a backup) that match the pre-prepare message before it multicasts a commit message to other replicas, which starts the commit phase. The commit phase ends when a replica has received $2f$ matching commit messages from other replicas. At this point, the request message has been totally ordered and it is ready to be delivered to the server application if all previous requests have already been delivered. If the primary or the client is faulty, the Byzantine agreement on the ordering of a request might not be reached, in which case, a new view is initiated, triggered by

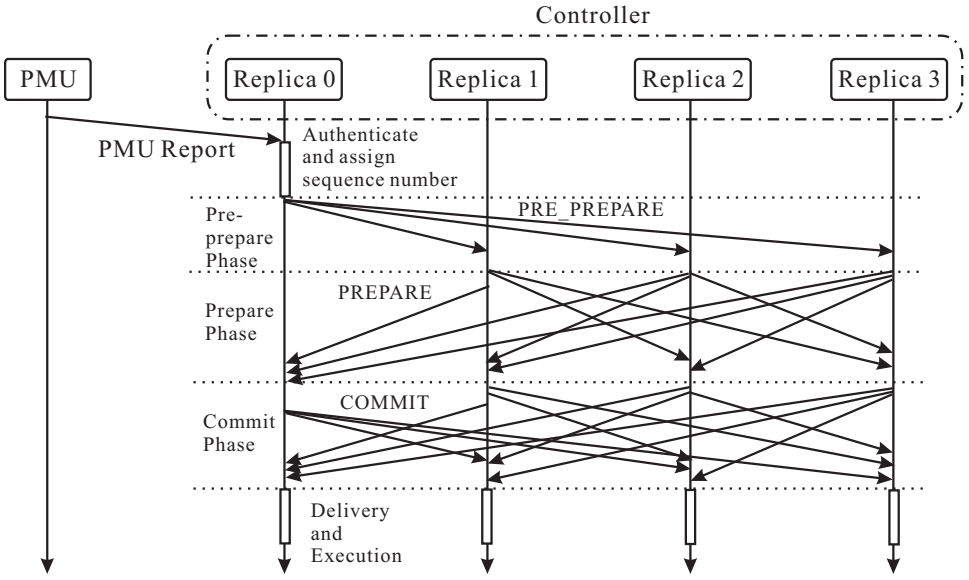

Figure 2. Normal operation of the BFT algorithm for PMU report handling.

a timeout on the current view. A different primary is designated in a round-robin fashion for each new view installed. To prevent a faulty replica or client to impersonate as another correct replica or client, all messages are protected by a digital signature, or an authenticator [4].

For electric power grid health monitoring and control, however, the above BFT algorithm cannot be used directly, because normally the controller replicas collect input from the PMUs and the control commands are issued to IEDs. Furthermore, the updates from PMUs are one-way messages in that the PMUs normally do not wait for an explicit response from the controller. On the other hand, IEDs are acting as the server role when it receives the control commands from the controller replicas.

On collecting PMU data, the controller replicas engage in a Byzantine agreement for each input message as usual, as shown in Figure 2. However, the message delivery procedure must be modified. When a replica reaches a Byzantine agreement on the message, and it has delivered all previously ordered messages, it invokes the callback function provided by the controller to deliver this message. On return of the up-call, no message is sent back to the PMU.

Upon issuing a control command, a controller replica does not directly send the command to the target IED. Instead, a round of Byzantine agreement on the command message is conducted, as shown in Figure 3. The procedure is very similar to that of PMU input message ordering, except that the pre-prepare message is triggered by the issuing of a control command rather than the receiving of a client's request, and the command is sent to the target IED when the Byzantine agreement is reached, instead of delivering a request. As mentioned in the previous Section, the target IED must not accept a control command immediately because the command might have been sent by a faulty controller replica. By waiting until it has received $f + 1$ identical command from different controller replicas, it can guarantee that at least one of them is sent by a correct replicas, because at most f replicas can be faulty according to your assumption.

If the replicas operate completely deterministically and in lock-step, the round of Byzantine agreement for the commands to the IEDs is not necessary. However, it is virtually impossible to guarantee lock-stepped execution of the replicas across a network. If the control command contains information such as the time to execute the command, the commands issued by different replicas would contain different timestamps, which would make it impossible for the IEDs to authenticate and compare the commands for acceptance. Therefore, in general, it is necessary for the replicas to reach an agreement on the command to be issued to the IEDs. Here, we assume that a backup replica is able to verify if the command proposed by the primary is valid. (If the command from the primary is deemed as invalid, the detecting backup replica would initiate a view change.) If a backup replica cannot verify the command issued by the primary, more sophisticated mechanisms must be used, as reported in [15].

4. Feasibility of Byzantine Fault Tolerance for Power Grid Monitoring and Control

The implementation of our Byzantine fault tolerance framework is based on the BFT library developed by Castro and Liskov at MIT [4]. We incorporated the changes

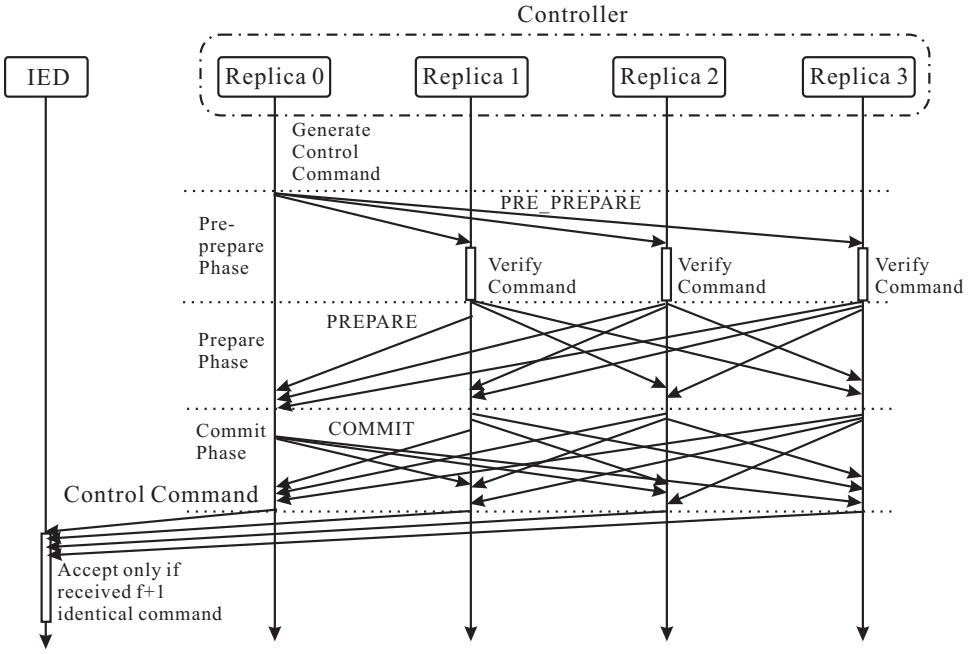

Figure 3. Normal operation of the BFT control command issuing at the controller replicas and the acceptance of the command at an IED.

necessary for electric power grid monitoring and control as mentioned in the last Section.

The test-bed consists of 12 PCs in a local-area network (LAN) connected by a 100Mbps Ethernet, and a single PC in a remote lab at UCSB across a wide-area network (WAN). Four of the PCs in the LAN are equipped with Pentium IV 2.8GHz processors and 256MB memory, and the remaining PCs in the LAN each has a single Pentium III 1GHz processor. All PCs on the LAN run the Red Hat 8.0 Linux. The remote PC has one Pentium IV 3.2GHz processor running CentOS 4.5 Linux.

The main objective of the performance evaluation is to find out if the Byzantine fault tolerance mechanisms are efficient enough to meet the real-time communication requirement for power grid health monitoring and control. Consequently, we characterize the response time and jitter of the Byzantine fault tolerant system in both the LAN and WAN environments.

The test application simulates the electric power grid health monitoring and control scenario as shown in Figure 1. The controller is replicated in the 4 Pentium IV PCs (one replica per PC) and the PMUs and IEDs are run on the remaining 8 Pentium III PCs (a pair of PMU and IED on each PC). During the experiments, up to 8 concurrent PMU-IED pairs are used.

A PMU (as the client) periodically reports its measured data to the replicated controller according to the IEEE 1344 standard [7]. Upon each PMU message received, the controller replicas generate a command and send it to the corresponding IED (collocated on the same node as the reporting PMU). Note that this is done purely for the purpose of performance characterization and might not match the practical usage scenarios. The payload of each PMU report is 14 bytes long. The payload of each control command is set to 128 bytes long.

Furthermore, the PCs in our test-bed are not equipped with high-resolution GPS devices, preventing us from directly measuring one-way latencies for PMU reports and control command notifications. Instead, we measure the round-trip time from the sending of a PMU report to the receiving of a command in response to the report at a collocated IED.

To gain insight on the jitters of networking and Byzantine agreement processing delays, we measure the intervals between two consecutive sending of PMU reports at each PMU and the intervals of consecutive deliveries of the PMU reports at each controller replica, and compare the probability density functions (PDFs) of the sending intervals and the delivery intervals. The PDFs provide a much more detailed and accurate picture on the predictability of the arrival rate of the PMU reports at the controller replicas than using the mean values and standard deviations. For similar reasons, the PDF is used to capture and present the round-trip times. In each run, 10,000 samples are taken.

Figure 4 shows the experimental results under the LAN environment and the normal operation condition. The num-

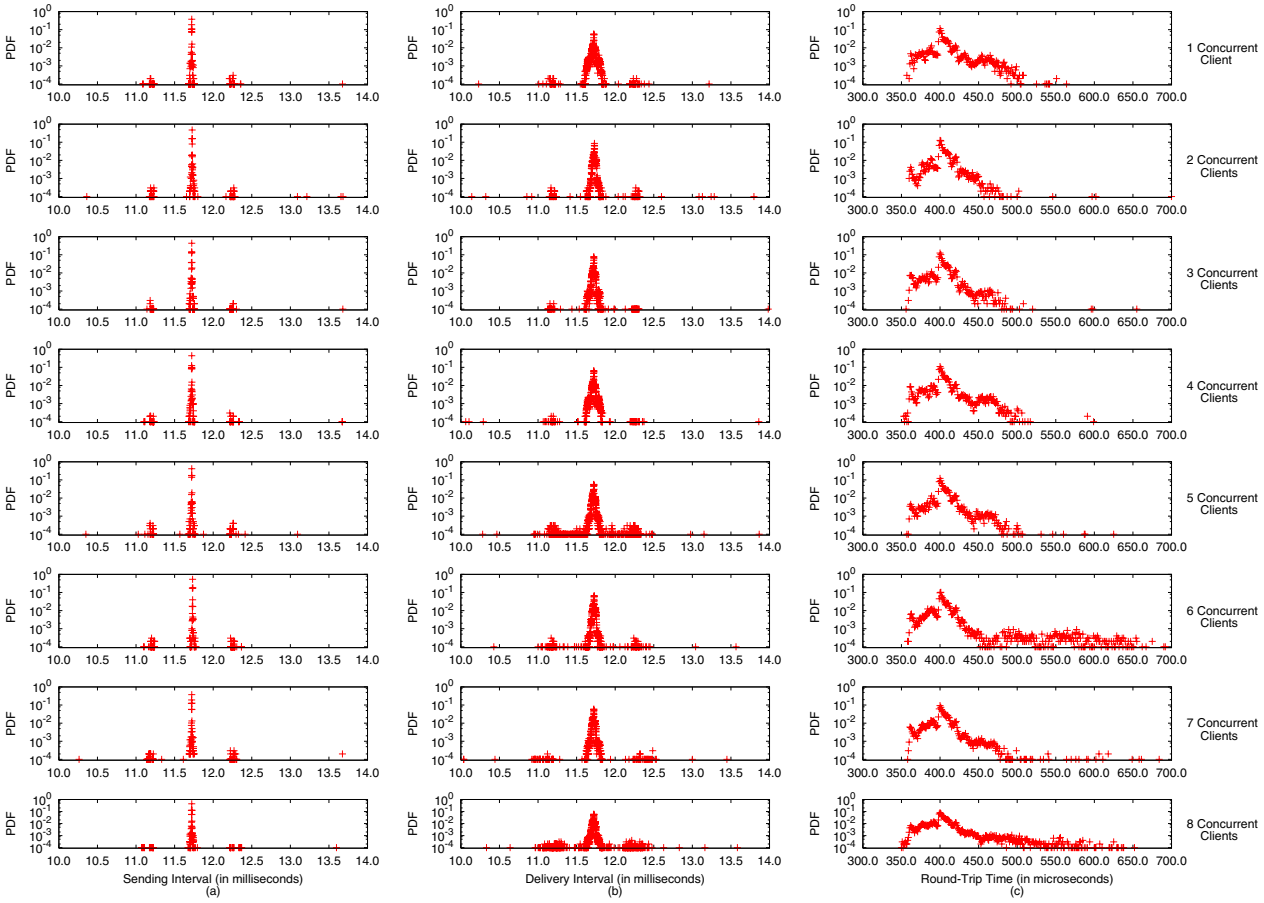

Figure 4. The measured PDFs with various number of PMU-IED pairs in a LAN environment. (a) The PMU report sending interval at the PMU. (b) The delivery interval at the controller replicas. (c) The round-trip time from sending a PMU report and the receiving of a control command.

ber of concurrent PMU-IED pairs varies from 1 to 8. The PDFs for the sending intervals measured at the PMUs are shown in Figure 4(a). The interval between two consecutive sending is controlled by the nanosleep() API provided by Linux. Even though the target interval is 10 milliseconds, the actual intervals vary slightly (with peak value of about 11.6 milliseconds). If there is no jitter in networking and Byzantine agreement processing, the PDFs of the delivery intervals measured at the controller replicas should be identical to those of the sending intervals. The PDFs of the delivery intervals shown in Figure 4(b) indicate that there is noticeable jitter. However, the jitter is small enough to sustain a 60Hz PMU sampling rate for all scenarios tested (up to 8 concurrent PMU-IED pairs), which is often regarded as the most demanding SCADA requirement [8]. Furthermore, Figure 4(c) shows that the round-trip time is in the sub-millisecond range, again for all scenarios measured, which is more than adequate to ensure urgent sensing data delivered and control command acted upon.

When the primary controller replica is faulty, it may take significant amount of time (e.g., 2 seconds) for a view change to complete. During this period of time, the controller is basically out-of-service. To address this issue, the controller should periodically send contingency control commands to the IEDs. If an IED does not receive a control command in time, it should resort to the contingency command. We emphasize that this situation, even though not desirable, is far better than the IED executing a command sent by a malicious controller, which can lead to the destruction of critical components of the power grid.

For the WAN experiment, we use the remote PC to run a pair of PMU and IED. The 4 controller replicas are deployed on the LAN test-bed. The measurement results for the sending/delivery intervals and the round-trip times are summarized in Figure 5. As indicated by the spread of the main peak of the PDF of the delivery intervals in Figure 5(b), the jittering is far more sever than that of the LAN environment. However, only 66 out of the 10,000 samples ob-

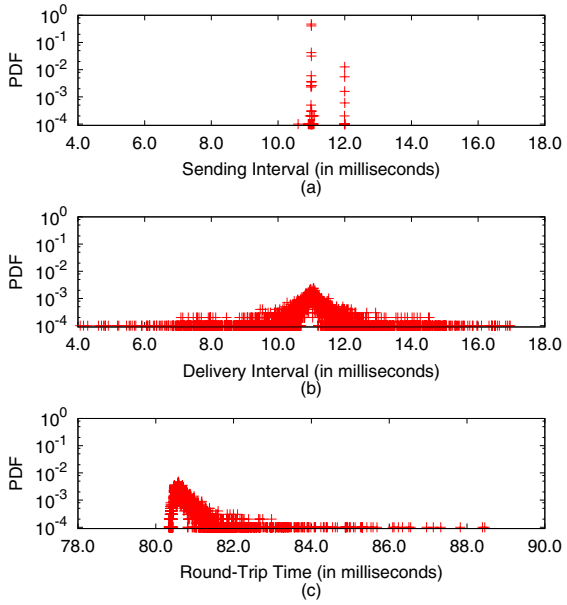

Figure 5. The PDFs measured in a WAN environment for (a) sending interval at a PMU, (b) delivery interval at the controller replica, and (c) round-trip time.

tained exceeded 16.67 milliseconds, which means that less than 1% of the PMU reports could not meet the 60 Hz sampling rate requirement. It may be sufficient for most applications. The round-trip time measurement result shown in Figure 4(c) indicates that most of the sampled response latencies fall between 80 to 90 milliseconds and only about 5% exceeded 100 milliseconds (but no measured round-trip time is larger than 0.5 second). Considering the long geographical distance between LAN test-bed (located in Ohio, USA) and the remote node (located in California, USA), our experiments show that Byzantine fault tolerant control over WANs is feasible for power grid applications that can tolerate 1-second reaction time.

5. Related Work

There is a large body of research work on how to restructure the current data communication infrastructure for electric power grid, such as [2, 3, 6, 11]. The SCADA security issues have also attracted worldwide attention [1]. However, the work that targets both the security and reliability aspects of the infrastructure is rarely seen. To the best of our knowledge, this work is the first to justify the use of Byzantine fault tolerance for electric power grid health monitoring and control, and the first to conduct extensive empirical study to show the feasibility of using Byzantine fault tolerant controls over both the LAN and WAN environments.

Byzantine fault tolerance has been a hot research area in many other areas, such as Web services [12, 14] and data storage systems [13]. Even though the work is in a different context, many insights are useful for BFT controls in electric power grid applications. In particular, the mechanisms designed to cope with the interaction of a replicated object and the unreplicated external entities reported in [12] have been partially incorporated in this work.

The importance of stable sampling rate for networked sensing and control is discussed in [10]. In [10], Liberatore proposed a playback-based method to increase the predictability of the sampling rate. This method can be readily used to enhance our Byzantine fault tolerant sensing and control for power grid.

6. Conclusion and Future Work

In this paper, we presented the justification and a feasibility study of applying the Byzantine fault tolerance technology to electric power grid health monitoring. Our Byzantine fault tolerance framework is adapted from the highly efficient BFT library developed by Castro and Liskov [4]. We proposed and implemented the BFT mechanisms needed to handle the PMU data reporting and control commands issuing to the IEDs. We conducted extensive experiments in both LAN and WAN environments to verify the feasibility of using the BFT technology for reliable and secure electric power grid monitoring and control. We show that under the LAN environment, the overhead and jitter introduced by the BFT mechanisms are negligible, and consequently, Byzantine fault tolerance could readily be used to improve the security and reliability of electric power grid monitoring and control while meeting the stringent real-time communication requirement for SCADA operations. Under the WAN environment, common phasor operation modes such as 20-30 Hz sampling rates could be used, with sub-second reaction time guarantee under normal operations.

While the brief out-of-service time (typically in 1-2 seconds) during a view change can be a concern for electric power grid health monitoring and control, additional mechanism, such as the playback scheme proposed in [10] could be used to alleviate the problem. In any case, the BFT sensing and control ensures that a PMU report from a compromised PMU cannot cause the state divergence of the correct controller replica, and a control command from a compromised control replica is never accepted by a correct actuating device such as an IED.

Future work includes the integration of our BFT sensing and control mechanisms with modern controller designs, and the support for large-scale distributed control and coordination among multiple control centers across the electric power grid.

References

[1] The Center for SCADA Security. Sandia National Laboratories, http://sandia.gov/scada/.

[2] K. P. Birman, J. Chen, E. M. Hopkinson, R. J. Thomas, J. S. Thorp, R. V. Renesse, and W. Vogels. Overcoming communications challenges in software for monitoring and controlling power systems. *Proceedings of the IEEE*, 93(5):1028–1041, May 2005.

[3] A. Bose. Improved communication/computation infrastructure for better monitoring and control. In *Proceedings of the IEEE Power Engineering Society General Meeting*, pages 2715–2716, 2005.

[4] M. Castro and B. Liskov. Practical Byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems*, 20(4):398–461, November 2002.

[5] CNN. Staged cyber attack reveals vulnerability in power grid. http://www.cnn.com/2007/US/09/26/power.at.risk/index.html.

[6] L. Hossenlopp. Engineering perspectives on iec 61850. *IEEE Power and Energy Magazine*, 5(3):45–50, 2007.

[7] IEEE. Std 1344-1995. IEEE Standard for Synchrophasors for Power Systems.

[8] R. Johnston. Obtaining high performance phasor measurements in a geographically distributed status dissemination network. Master's thesis, Washington State University, 2005.

[9] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–340, 1982.

[10] V. Liberatore. Integrated play-back, sensing and network control. In *Proceedings of the 25th IEEE International Conference on Computer Communications*, pages 1–12, Barcelona, Spain, 2006.

[11] A. P. S. Melliopoulos. Substation automation: Are we there yet? *IEEE Power and Energy Magazine*, 5(3):28–30, 2007.

[12] M. Merideth, A. Iyengar, T. Mikalsen, S. Tai, I. Rouvellou, and P. Narasimhan. Thema: Byzantine-fault-tolerant middleware for web services applications. In *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, pages 131–142, 2005.

[13] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz. Pond: the oceanstore prototype. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, March 2003.

[14] W. Zhao. BFT-WS: A Byzantine fault tolerance framework for web services. In *Proceedings of the Middleware for Web Services Workshop*, Annapolis, MD, October 2007.

[15] W. Zhao. Byzantine fault tolerance for nondeterministic applications. In *Proceedings of the 3rd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pages 108–115, Loyola College Graduate Center, Columbia, MD, September 2007.