Real-Time Byzantine Resilient Power Grid Infrastructure: Evaluation and Trade-offs

Sahiti Bommareddy, Maher Khan, David J Sebastian Cardenas, Carl Miller,
Christopher Bonebrake, Yair Amir, and Amy Babay
Department of Computer Science, Johns Hopkins University — {sahiti, yairamir}@cs.jhu.edu
School of Computing and Information, University of Pittsburgh — {maherkhan, babay}@pitt.edu
Pacific Northwest National Labs (PNNL) — {d.sebastiancardenas, carl.miller, christopher.bonebrake}@pnnl.gov

***Abstract*—Increasing threats to power grid infrastructure are driving the need to build Byzantine-resilient systems that can continue to operate correctly despite failures and attacks. However, the real-time requirements of power grid infrastructure call for a more rigorous evaluation of Byzantine resilient systems than the traditional evaluations performed in the context of standard IT applications. We discuss these requirements, and the potential of commercial-off-the-shelf and open source solutions to support real-time resilient systems.**

I. Introduction

The rising number of cyberattacks against critical infrastructure reinforces the need to build Byzantine resilient power grid infrastructure [1], [2]. While Byzantine resilient techniques have been developed in the context of IT applications [3], applying them in the power grid domain brings the need to consider strict real-time requirements. Supervisory Control and Data Acquisition (SCADA) operations in power grid control centers typically require latency of 100-200ms [4], [5], and the requirements become much more stringent as we move from control centers to substations that carry out critical protection functions, due to the physical properties of the system.

During a fault condition, currents flowing into the faulted zone experience a rapid surge (many orders of magnitude greater than during normal conditions). This sudden increase in current is capable of damaging grid equipment in just a few electrical cycles (in the US, a cycle is 16.667ms). Therefore, according to industry and standardization bodies (IEEE, IEC), fault identification must occur within a quarter power cycle, i.e. 4.167ms [4].

Figure 1 shows a fault condition, where the current swelling behavior can be observed. In a well-designed system, a quick relay response enables the circuit breaker to have more time to safely dissipate the energy present within the system. Due to the amount of energy being released, circuit breakers can take 1-3 cycles to fully interrupt the fault/arcing current, drastically reducing the time to fault identification. If this fault is not identified quickly, damage to physical equipment, and cascading escalations to otherwise healthy systems may be unavoidable. Damaged grid equipment like the 345kV transformer can cost millions of dollars and have lengthy replacement process in the orders of years. Therefore, meeting the 4.167ms requirement is critical.

Most existing Byzantine resilient techniques are not designed for or evaluated under these strict real-time requirements. Hence, the first goal of our work is to investigate the type of evaluation needed to provide confidence that Byzantine-resilient systems can meet the real-time requirements of power grid infrastructure in practice.

The second goal of our work is to use the rigorous evaluation strategy we propose to explore new deployment trade-offs. In particular, we investigate the potential of commercial-off-the-shelf (COTS) and open-source solutions to support real-time Byzantine-resilient infrastructure. Compared to specialized real-time systems, COTS and open-source systems are less expensive, easier to deploy and manage, and easier to audit [6], making them an attractive option.

II. Landscape of BFT SMR Evaluation

Byzantine Fault Tolerant State Machine Replication (BFT SMR) is a classical approach to tolerate intrusions in networked systems. Initial works on BFT SMR focused on proving that the techniques are correct and practical for IT applications like file systems and storage [3], [7]. As the research matured, performance became a key focus [8]–[11], but most works focused on normal-case throughput and scaling evaluations. Some of these works expanded evaluation to include malicious behavior [8], [11]–[14], and even designed protocols to maintain performance under attack [12], [15], [16], but their empirical evaluations are still based on IT domain requirements. Even in more recent BFT SMR protocols designed for permissioned blockchains, the focus of evaluations remains on throughput scalability and blockchain requirements [17]–[19]. Frameworks have been proposed to evaluate and compare the performance of BFT algorithms, but they generally focus on quantifying performance analytically, based on the number of cryptography operations or messages exchanged, rather than on empirical evaluation [20], [21].

III. Evaluating Byzantine-Resilient Power Grid Infrastructure

To gain confidence that a system can meet real-time requirements in practice, it is necessary to (1) consider the worst-case number of constraint violations, not only average latency, (2) evaluate the most demanding operating conditions that the system is expected to work under, including all failure/attack

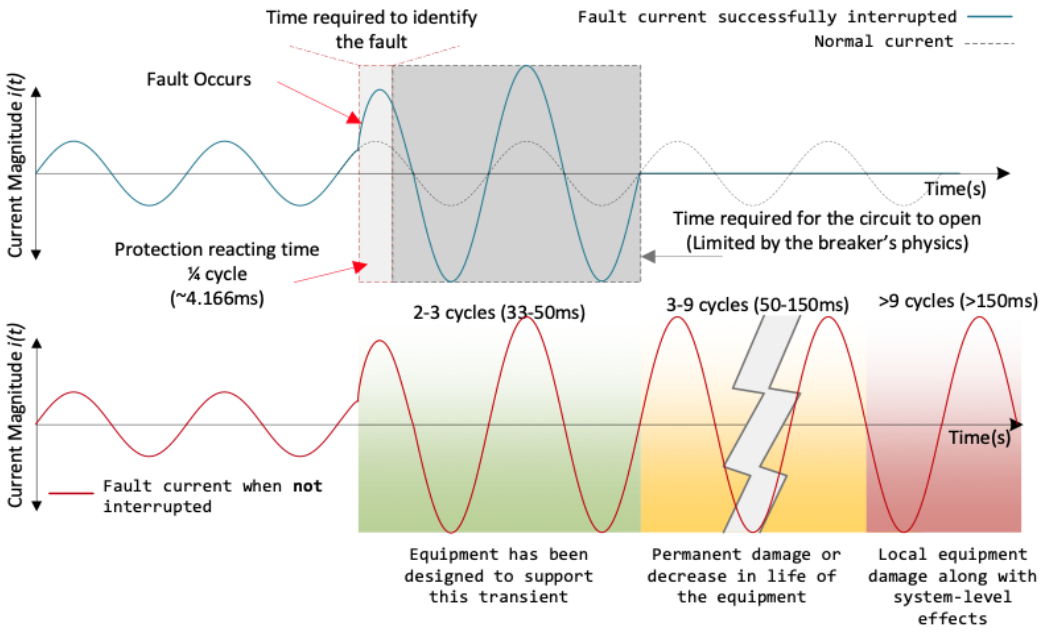

Fig. 1. Fault detection timing requirements in substation

TABLE I
PERFORMANCE IN DIFFERENT OPERATING CONDITIONS WITH FOUR RELAY NODES ($f = 1, k = 1$)

Operating Condition	Normal Kernel (microseconds)			Real-Time Kernel (microseconds)		
	Minimum	Average	Maximum	Minimum	Average	Maximum
Fault-Free (Normal)	**1723**	**2187**	**3323**	**1637**	**1950**	**3596**
Fail-Stop Fault or Proactive Recovery	**1871**	**2260**	**3326**	**1608**	**1976**	**3726**
Fail-Stop Fault and Proactive Recovery	**1912**	**2328**	**7617** (8*)	**1750**	**2015**	**3996**
Byzantine Fault	**1737**	**2227**	**3785**	**1665**	**1984**	**4002**
Byzantine Fault and Proactive Recovery	**1867**	**2313**	**7699** (6*)	**1767**	**2019**	**4101**

* The count of actions that crossed 4.167 milliseconds (out of 1 million total actions)

cases, (3) evaluate each operating condition over a sufficiently long period of time, and (4) use a realistic testbed.

Our Spire intrusion-tolerant SCADA system took a step towards evaluating real-time requirements at the control-center level [22]. It used an extensive 30-hour evaluation on a real wide-area network for the normal-case, and shorter under-attack evaluations in a cluster environment with emulated wide-area latencies, analyzing how many updates failed to meet the target latency of 100-200ms [22].

However, at the substation level, even an evaluation like the one in [22] is not sufficient, as shown by our work on Spire for the Substation, a real-time Byzantine-resilient system for substation protection [23]. In Spire for the Substation, we developed two Byzantine-resilient protocols: Arbiter Protocol and Peer Protocol. Both use four relay nodes to simultaneously tolerate one Byzantine relay and one proactive recovery but have different deployment tradeoffs [23]. In that work, we rigorously evaluated both protocols, running each failure/attack case for 24 hours, processing 1 million actions. That evaluation showed that while the Peer Protocol has a smaller attack surface and can seamlessly integrate into the substation, its tradeoff is a slight risk of not meeting the latency requirement under the most demanding attack scenarios.

Figure 6 illustrates the importance of this extensive evaluation. Running the Peer Protocol for 1 million actions with a simultaneous Byzantine fault and proactive recovery, we see that 6 actions cross the 4.167ms threshold. While an evaluation with 100,000 under-attack actions (taking hours) would be significantly more extensive than those of many BFT systems (see Section II), it would not necessarily show these constraint violations. For example, in Figure 6, in the slices from $[320000, 520000]$, $[620000, 720000]$, $[720000, 820000]$, $[820000, 920000]$, we see intervals of 100,000 actions or more with none crossing 4.167ms. The 1-million-action evaluation is needed for an accurate view of the protocol's performance.

Given the Peer Protocol's tradeoff of better deployment properties vs the risk of not meeting real-time requirements, one possibility is to explore specialized real-time hardware and/or software. However, maintaining a COTS hardware and open-source software environment simplifies system management and maintenance [24], [25]. Therefore, we explore a different trade-off, evaluating whether the Linux real-time kernel option can achieve the needed performance.

Real-time Kernel Evaluation. We deployed Spire for the

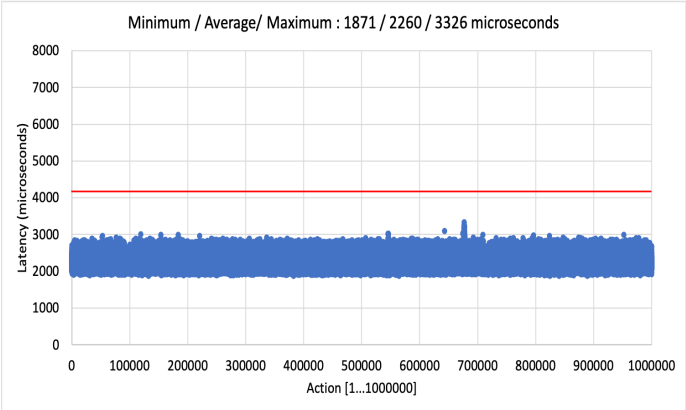

Fig. 2. Normal Kernel: Fail-Stop Fault

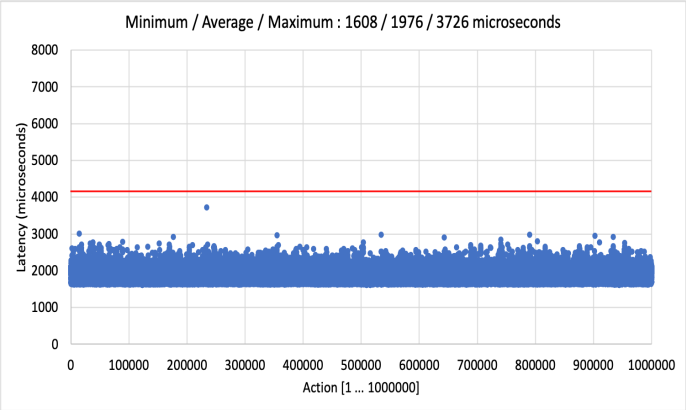

Fig. 3. Real-Time Kernel: Fail-Stop Fault

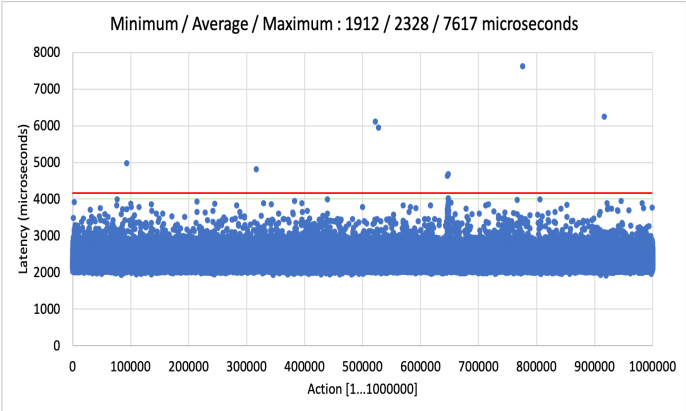

Fig. 4. Normal Kernel: Fail-Stop Fault with Proactive Recovery

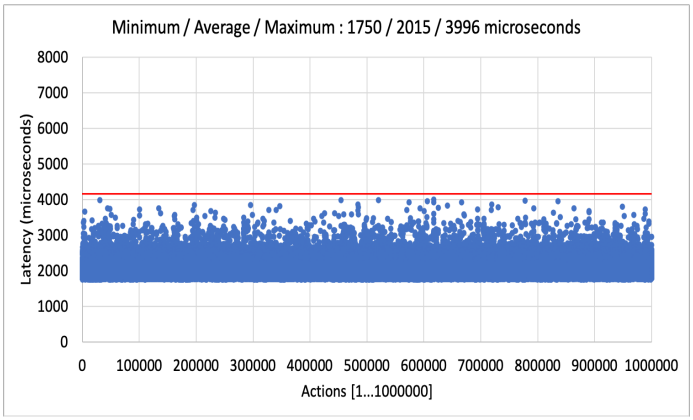

Fig. 5. Real-Time Kernel: Fail-Stop Fault with Proactive Recovery

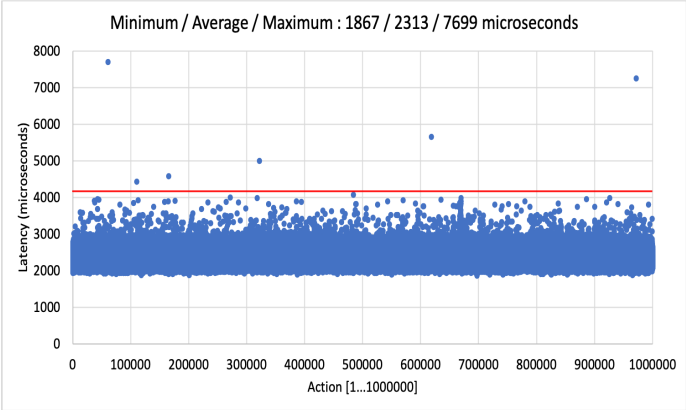

Fig. 6. Normal Kernel: Byzantine Fault with Proactive Recovery

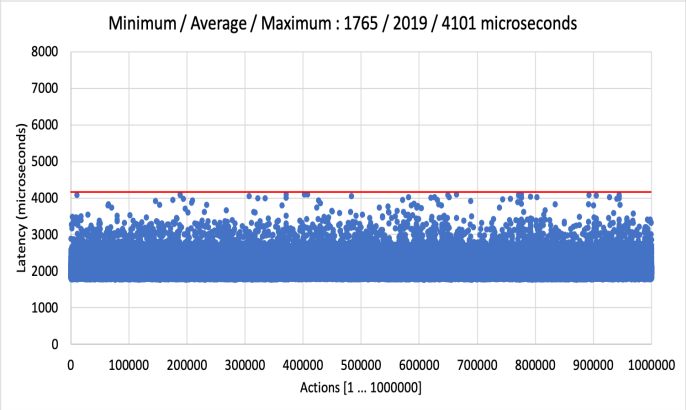

Fig. 7. Real-Time Kernel: Byzantine Fault with Proactive Recovery

Substation (Peer Protocol) in our testbed with four Intel Xenon E3 servers with 16GB RAM running CentOS 8 and connected by a 1Gbps network[1]. The deployed system consists of four relay nodes and therefore can simultaneously tolerate up to one faulty relay node (fail-stop or Byzantine) and up to one additional relay node undergoing proactive recovery. Using the same 24-hour, 1-million-action evaluation strategy described above, we compared the system using the normal Linux kernel, and with the real-time kernel patch and tuning on the same machines. Table I reports the minimum, average and maximum latency in microseconds for all five operating

[1]This is a different testbed than the one in [23]. Although the results show minor quantitative differences between the two testbeds, their qualitative profiles are the same.

conditions supported by the threat model for both the normal and real-time kernels.

While Table I summarizes the results, Figures 4, 5, 6 and 7 focus on the most demanding scenarios: Fail-Stop fault with simultaneous proactive recovery and Byzantine fault with simultaneous proactive recovery. In the former, one of the relay nodes is unavailable due to a fail-stop fault while, simultaneously, an additional relay node is undergoing proactive recovery. In the latter, the Byzantine relay node performs two simultaneous attacks for each action. First, the Byzantine relay node sends a corrupt message that will not be useful. Second, the Byzantine relay node performs a short intermittent denial of service attack on the other relay nodes to consume their network and computational resources further. In both of these operating conditions, the proactive recovery node is not rejuvenated for the entire test duration, so there are only two correct relay nodes available throughout the test. We refer to this condition with only two correct relay nodes available as non-optionality condition.

Under non-optionality, meeting the latency requirement is particularly tough. With only two nodes available, a random delay on either node (e.g., from network delays, kernel scheduling, or even effects of a Byzantine node's actions) would be reflected in the end-to-end latency. Compare this situation to Fail-Stop fault condition in which three nodes are available. In such a case, delays would have to occur on two of the three nodes independently and at the same time in order to be reflected in the final latency. For both the normal and real-time kernel, comparison of their Fail-Stop fault condition performance (Figures 2 & 3) to that of Fail-Stop fault with simultaneous proactive recovery condition (Figures 4 & 5) exemplify this effect i.e., Figure 2 vs Figure 4 and Figure 3 vs Figure 5.

The normal linux kernel is optimized for throughput and fair scheduling of tasks, while the real-time kernel is optimized to maintain low latency, consistent response time and determinism. These characteristics are particularly important in the conditions with non-optionality where any random delay impacts the system performance. Due to its features, in real-time kernel benchmarks, when we use high priority and FIFO scheduling policy, all actions meet the 4.167ms requirement, and average latency is reduced by about 300 microseconds across all operating conditions (Table I). These observations can also be immediately noted by comparing benchmark plots of normal kernel to those of real-time kernel in the three operating conditions shown: Fail-Stop fault (Figure 2 vs Figure 3), Fail-Stop fault with simultaneous proactive recovery (Figure 4 vs Figure 5), and Byzantine fault with simultaneous proactive recovery (Figure 6 vs Figure 7) .

The determinism and latency stability of real-time kernel offers a new deployment option for real-time Byzantine resilient critical infrastructure, enabling us to get the benefits of the Peer Protocol while meeting real-time requirements.

COTS and Open Source Discussion. The real-time kernel option enables us to remain in the more flexible open-source realm while still meeting real-time requirements, which is of increasing interest in the power industry. The protective relay is a relatively expensive device (tens of thousands of dollars). Each substation has multiple relays employed for protection schemes. There are hundreds to thousands of substations across a country (e.g., U.S has over 55,000 substations). Hence, to reduce cost, there is a high incentive to limit the additional relays needed in a Byzantine resilient scheme.

In a deployment, each relay node discussed above consists of a protective relay device directly connected to the Spire for the substation harness. Protective relays are typically specialized hardware relays with proprietary software implementing the protection functions. As we need four relays in the four relay nodes to tolerate an intrusion, the system will be costly. To reduce the cost, we experimented by substituting a hardware protective relay in one of the relay nodes with a software relay. The software relay is implemented on an Intel NUC (CentOS) with real-time kernel to perform the same protection function as hardware protective relays. In fact, the system successfully underwent red team experiments with three hardware relays (from GE, Siemens and Hitachi Energy) and a single software relay (on a different test bed at Pacific Northwest National Laboratory). However, the implications of using a real-time kernel on other processes running on the nodes, overall throughput and the deployment environment needs further investigation.

IV. Conclusion

We have presented the need for rigorous evaluation of Byzantine-resilient systems with respect to the real-time requirements of critical infrastructure. We also investigated the potential of COTS and open source systems to support real-time Byzantine resilient power grid infrastructure.

Acknowledgement

This work was supported in part by the Department of Energy (DOE) Offices of Cybersecurity, Energy Security, and Emergency Response (CESER); Electricity (OE); and Nuclear Energy (NE) under the Grid Modernization Laboratory Consortium (GMLC) Topic 5.1.4 – Cyber-Physical Security. Its contents are solely the responsibility of the authors and do not represent the official view of DOE.

This work was also supported in part by the DoD Strategic Environmental Research and Development Program (SERDP) grant RC20-1138.

Yair Amir is a co-founder and member of and holds equity in Spread Concepts LLC. The results discussed in this paper could affect the value of Spread Concepts LLC. This arrangement has been reviewed and approved by the Johns Hopkins University in accordance with its conflict of interest policies.

References

[1] "Cost of a data breach 2022." [Online]. Available: https://www.ibm.com/reports/data-breach

[2] T. Group, "2022 thales data threat report." [Online]. Available: https://cpl.thalesgroup.com/critical-infrastructure-data-threat-report

[3] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OsDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[4] IEEE, "Ieee standard communication delivery time performance requirements for electric power substation automation," *IEEE Std 1646-2004*, pp. 1–24, 2005.

[5] J. Deshpande, A. Locke, and M. Madden, "Smart choices for the smart grid," *Alcatel-Lucent Technolgy White Paper*, 2011.

[6] R. Hat, "The state of enterprise open source," 2021.

[7] L. Lamport, "Paxos made simple," *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pp. 51–58, 2001.

[8] M. Abd-El-Malek, G. R. Ganger, G. R. Goodson, M. K. Reiter, and J. J. Wylie, "Fault-scalable byzantine fault-tolerant services," *ACM SIGOPS Operating Systems Review*, vol. 39, no. 5, pp. 59–74, 2005.

[9] J. Cowling, D. Myers, B. Liskov, R. Rodrigues, and L. Shrira, "Hq replication: A hybrid quorum protocol for byzantine fault tolerance," in *Proceedings of the 7th symposium on Operating systems design and implementation*, 2006, pp. 177–190.

[10] P.-L. Aublin, R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić, "The next 700 bft protocols," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 4, pp. 1–45, 2015.

[11] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: speculative byzantine fault tolerance," in *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, 2007, pp. 45–58.

[12] Y. Amir, B. Coan, J. Kirsch, and J. Lane, "Prime: Byzantine replication under attack," *IEEE transactions on dependable and secure computing*, vol. 8, no. 4, pp. 564–577, 2010.

[13] P.-L. Aublin, S. B. Mokhtar, and V. Quéma, "Rbft: Redundant byzantine fault tolerance," in *2013 IEEE 33rd International Conference on Distributed Computing Systems*. IEEE, 2013, pp. 297–306.

[14] A. Bessani, J. Sousa, and E. E. Alchieri, "State machine replication for the masses with bft-smart," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014, pp. 355–362.

[15] A. Clement, E. L. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, "Making byzantine fault tolerant systems tolerate byzantine faults." in *NSDI*, vol. 9, 2009, pp. 153–168.

[16] G. S. Veronese, M. Correia, A. N. Bessani, and L. C. Lung, "Spin one's wheels? byzantine fault tolerance with a spinning primary," in *2009 28th IEEE International Symposium on Reliable Distributed Systems*. IEEE, 2009, pp. 135–144.

[17] A. Bessani, J. Sousa, and M. Vukolić, "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform," in *Proceedings of the 1st workshop on scalable and resilient infrastructures for distributed ledgers*, 2017, pp. 1–2.

[18] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 31–42.

[19] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019, pp. 347–356.

[20] J. R. Clavin, Y. Huang, X. Wang, P. M. Prakash, S. Duan, J. Wang, and S. Peisert, "A framework for evaluating bft," in *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2021, pp. 193–200.

[21] D. Gupta, L. Perronne, and S. Bouchenak, "Bft-bench: Towards a practical evaluation of robustness and effectiveness of bft protocols," in *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, 2016, pp. 115–128.

[22] A. Babay, T. Tantillo, T. Aron, M. Platania, and Y. Amir, "Network-attack-resilient intrusion-tolerant scada for the power grid," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 255–266.

[23] S. Bommareddy, D. Qian, C. Bonebrake, P. Skare, and Y. Amir, "Real-time byzantine resilience for power grid substations," in *41st International Symposium on Reliable Distributed Systems*, 2022, pp. 213–224.

[24] B. Fitzgerald, "The transformation of open source software," *MIS quarterly*, pp. 587–598, 2006.

[25] "S.4913 - 117th congress (2021-2022): Securing open source software act of 2022," accessed: 2022-9-30. [Online]. Available: https://www.govinfo.gov/content/pkg/BILLS-117s4913is/pdf/BILLS-117s4913is.pdf