Svedberg Open
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Blockchain Inspired Fault Monitoring And Event Recording In Medium Voltage DC Distribution Systems

Ramanarayana Nandigam[1,*], Ganesan Sivarajan[2] , Veera Kumari Maddipati[3] , Subramanian S.[4]

[1] Department of Electrical and Electronics Engineering, Annamalai University, India; and Department of Electrical and Electronics Engineering, Sir C. R. Reddy College of Engineering, Eluru, Andhra Pradesh, India. , 0000-0003-0038-1352

[2] Department of Electrical and Electronics Engineering, Government College of Engineering, Salem, Salem, India. , 0000-0003-3955-1400

[3] Department of Electrical and Electronics Engineering, Sir C. R. Reddy College of Engineering, Eluru, Andhra Pradesh, India., 0000-0003-0042-0024

[4] Department of Electrical Engineering, Annamalai University, Chidambaram, India. , 0000-0002-9888-5382

* Correspondence: ramanarayana1@sircrrengg.ac.in

Abstract

Fault management in medium-voltage direct-current (MVDC) networks requires coordination between event detection, operational response, and event-history maintenance. This paper presents a blockchain-inspired framework and a MATLAB App Designer prototype for examining these functions. A ten-node network is represented by a random symmetric adjacency matrix, while simulated voltage changes and fault indicators drive predefined response rules. The interface combines network visualization, fault reporting, a ledger-like transaction table, and node-voltage displays. Demonstrated cases show identification of a simulated fault, assignment of an isolation status, event recording, and a subsequent no-fault display. The implementation executes these functions within one MATLAB application, emulating node-associated reporting and contract-like automation. The results establish an application-level proof of concept for organizing fault-management information and operator feedback. Cryptographic consensus, physical interruption, and comparative protection performance remain outside the demonstrated scope. The prototype provides a basis for subsequent electrical modelling, distributed implementation, and reproducible cyber-physical evaluation.

Keywords: MVDC Networks; Decentralized Fault Management; Blockchain-Inspired Coordination; Event Logging; Rule-Based Automation; MATLAB App Designer.

1 Introduction

Medium-voltage direct-current (MVDC) distribution is relevant to power systems that integrate electronic converters, DC loads, energy storage, and distributed generation. Reviews of MVDC applications and fault-detection methods show that network configuration and source behaviour influence both the electrical disturbance and the protection requirements [1, 2]. Fault management also includes information-processing functions: identifying the affected component, recording the event, communicating the operational response, and tracking the conditions for restoration.

Communication, control, and security are interconnected aspects of smart-grid design [3]. Microgrids introduce opportunities for local coordination, but also require appropriate operational interfaces with the wider network [4]. Reliability must therefore be assessed across electrical equipment, communication links, and decision processes [5]. A reduction in the number of centralized decisions can change these dependencies, but does not by itself demonstrate a reduction in fault-clearing time or an improvement in service continuity.

Distributed intelligence and autonomous distribution-network control have been examined in the literature [6, 7]. These approaches motivate assigning reporting and response responsibilities to identifiable network participants. For fault management, the distinction between a local electrical decision and a shared event

record is particularly useful. The former concerns the condition of a protected component; the latter concerns coordination and the traceability of actions taken after an event.

Blockchain introduces mechanisms for maintaining an agreed transaction history across participants [8]. Reviews of smart-grid blockchain applications describe possible uses in energy exchange, data management, and automation [9]. These ideas motivate the framework developed here. However, a graphical transaction table and a distributed cryptographic ledger are different implementations, and each supports a different level of evidence.

This study presents a ten-node MATLAB App Designer prototype that emulates node-associated fault reporting, event logging, and predefined response rules. Its contribution is an integrated application workflow in which the network view, fault record, transaction history, and voltage display are updated from a simulated event. The evidence is a functional demonstration of this workflow. The electrical network is represented logically, and the blockchain and smart-contract functions are emulated within MATLAB. Section 2 reviews related work, Section 3 describes the methodology, Section 4 presents the demonstrated outputs, Section 5 discusses their implications and limitations, and Section 6 concludes the paper.

2 State of the art and research gap

2.1 Distributed monitoring and fault management

Coffey et al. [1] review MVDC applications, technologies, and future prospects, while Blázquez et al. [2] examine fault-detection algorithms for MVDC grids. These studies provide the electrical context for distinguishing fault indication, location, isolation, and interruption. A software notification of a fault is only one part of this process. The behaviour of converters, lines, and protection equipment determines whether the required electrical response can be achieved.

The communication and security issues collected by Muyeen and Rahman [3], the microgrid opportunities discussed by Yoldaş et al. [4], and the reliability perspective of Moslehi and Kumar [5] show why fault-management architecture must be examined beyond its user interface. Strasser et al. [6] review concepts for intelligence in future energy systems, and Lo and Ansari [7] investigate decentralized control and communication. These contributions motivate local decision roles; they do not imply that all centralized systems respond slowly or that every distributed implementation is inherently more reliable.

2.2 Blockchain applications in energy systems

Nakamoto [8] describes the transaction-history mechanism underlying Bitcoin. Smart-grid applications require their own decisions about participant identity, access, consensus, and operational timing. Musleh et al. [9] review blockchain applications and frameworks for smart grids, while Miglani et al. [10] examine blockchain for Internet of Energy management. Rathore et al. [11] discuss blockchain-enabled cyber-physical systems. Together, these works establish a broad architectural context for examining how physical events are represented and exchanged digitally.

Agung and Handayani [12] examine blockchain for the smart grid. Related energy-management studies address different coordination objectives: Gensollen et al. [13] investigate diversification in prosumer coalitions, Nezamabadi and Vahidinasab [14] formulate microgrid bidding in a transactive market, and Abdella et al. [15] evaluate a blockchain-based peer-to-peer energy-trading architecture. Mahesh et al. [16] consider blockchain-supported energy management in DC microgrids. Market coordination and event recording share information-management concerns, but their performance requirements cannot be transferred directly to electrical fault interruption.

2.3 Data integrity and application scope

Bhattacharjee et al. [17] introduce Block-Phasor as a blockchain framework for synchrophasor security. This represents a more specific measurement-data application than a general transaction display. Lim et al. [18] review blockchain applications in supply chains, providing cross-domain context for traceability and information sharing. The industry commentary by Morris [19] discusses potential responses to grid cyber-threats. The latter two sources are used for application context, rather than as experimental evidence for MVDC protection performance.

A common distinction across these applications is the difference between preserving a submitted record and establishing that the submitted measurement is correct. A ledger may support traceability under its trust assumptions, but a faulty or compromised input can still require independent validation. Similarly, a

programmed response can automate a decision without proving that the associated physical action has occurred. These distinctions guide the interpretation of the present prototype.

2.4 Positioning of the present study

The specific problem addressed here is how to organize a node-associated fault-management sequence into a consistent software workflow. The prototype brings together fault indication, response status, event history, and visual feedback in one environment. Table 1 positions this contribution against the related literature. The study does not introduce a new consensus algorithm or a new electrical fault-detection principle. Its purpose is to demonstrate the organization and observability of the coordination process before distributed and physical implementations are evaluated.

Table 1. Related research themes and their relationship to the present prototype.

Research theme	Representative sources	Relationship to this study
MVDC applications and detection	Coffey et al. [1]; Blázquez et al. [2]	Electrical context; dynamic protection methods are not reproduced.
Distributed intelligence and control	Strasser et al. [6]; Lo and Ansari [7]	Motivates node-associated reporting and coordination responsibilities.
Blockchain and cyber-physical systems	Musleh et al. [9]; Miglani et al. [10]; Rathore et al. [11]	Architectural context for event history and automation.
Prosumer and energy-market coordination	Gensollen et al. [13]; Nezamabadi and Vahidinasab [14]; Abdella et al. [15]; Mahesh et al. [16]	Related coordination applications with different evaluation objectives.
Measurement-data security	Bhattacharjee et al. [17]	Context for data integrity; cryptographic measurement validation is not implemented here.
Present prototype	Ten-node MATLAB application	Simulated fault reporting, status assignment, ledger-like logging, and visual feedback.

3 Methodology

3.1 Study design and implementation boundary

The study uses a logic-based prototype developed in MATLAB App Designer. A ten-node network is visualized using a random symmetric adjacency matrix, and node-voltage values are assigned within the application. A simulated event provides the fault indicator, affected-node identifier, and voltage values used to update the interface. This design supports examination of the software sequence without requiring a detailed electrical circuit or a Simulink model.

The implementation emulates blockchain-related functions through an in-application transaction table and conditional response routines. No deployed Solidity contract, Ganache transaction receipt, or independently executing validator is established by the demonstrated workflow. The terms blockchain-inspired and contract-like therefore describe the architectural motivation and rule-based behaviour. A future blockchain deployment would require additional implementation and verification beyond the MATLAB interface.

Figure 1 presents the conceptual relationship between reporting, decision rules, and fault-management components. Figure 2 summarizes the intended roles of a distributed ledger. Security, immutability, and resilience in these conceptual figures are design objectives; the local application alone does not establish those properties.

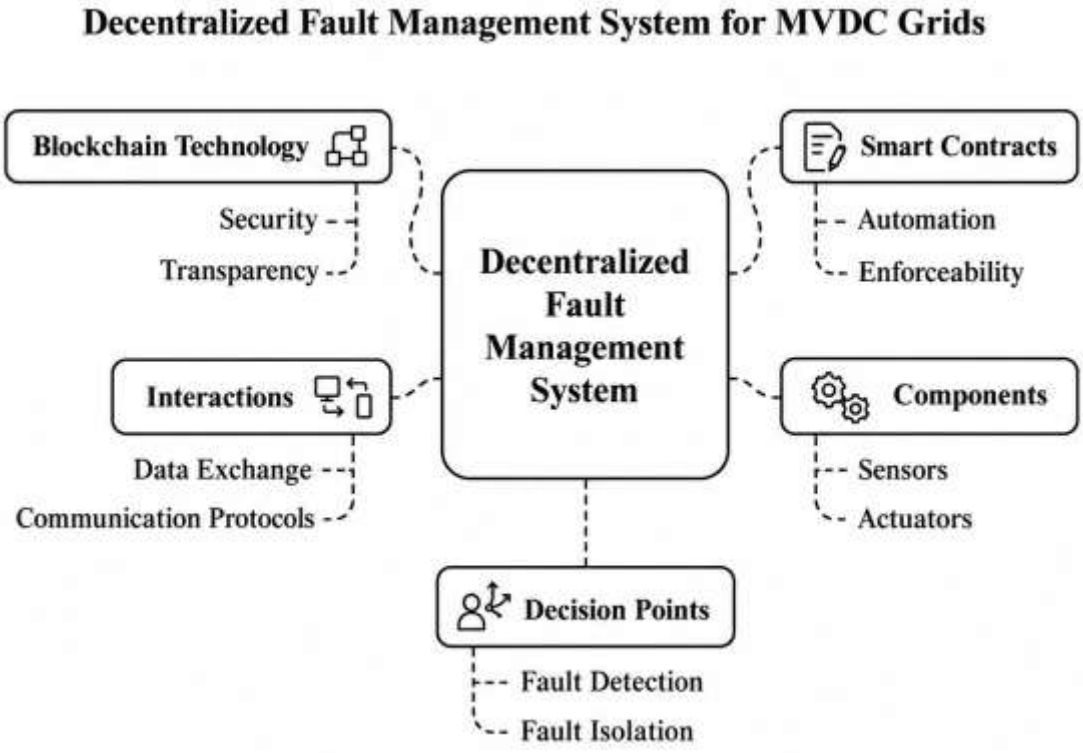

Figure 1. Conceptual components of blockchain-inspired decentralized fault management. The illustrated security and automation functions describe the intended architecture.

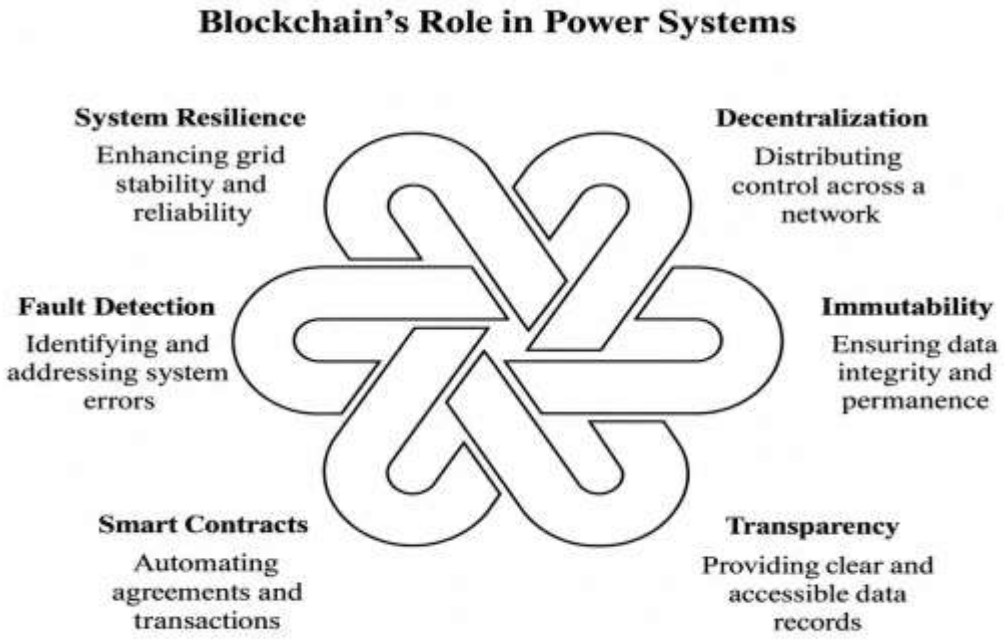

Figure 2. Conceptual ledger roles in power-system coordination. The prototype emulates event recording and automated response; distributed security properties require a separate implementation.

3.2 Network representation and simulated inputs

The adjacency matrix records the logical connections between the ten nodes. Symmetry represents an undirected relationship between connected nodes. The graph layout is used to display node identities and connectivity; its horizontal and vertical coordinates are drawing coordinates, rather than physical distances or electrical quantities. The topology therefore provides a visualization and indexing structure for the application.

The initialized node voltages are nominally 1.0 per unit (pu). A simulated fault changes the selected node voltage and returns an event indication to the response routine. The voltage plot uses a displayed range of 0–1.5 pu. These assigned values illustrate how the interface responds to abnormal data. Line impedances, converter current limits, load-flow equations, and electromagnetic transients are not solved in this prototype.

A node label in the fault table identifies the location supplied by the simulator. The demonstrated algorithm does not independently estimate this location from distributed measurements. Likewise, the generic label Simulated Fault does not distinguish pole-to-pole, pole-to-ground, or open-circuit mechanisms. Such faults motivate the application domain, but require explicit electrical and decision models before type-specific detection performance can be evaluated.

3.3 Functional architecture

The intended architecture separates five roles: a representation of the MVDC network, node-associated reporting, event-history storage, conditional response logic, and operator notification. Figure 3 illustrates these relationships. In the prototype, they are implemented as cooperating functions and interface components inside the same application. Logical separation makes the workflow easier to inspect, while physical decentralization remains an implementation objective.

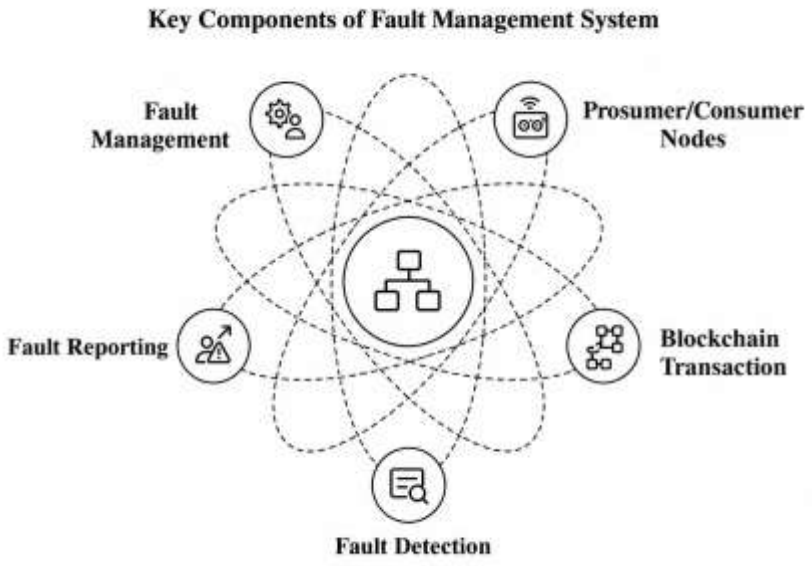

Figure 3. Conceptual interaction of network participants, fault reporting, event recording, and management functions. The demonstrated functions execute within one MATLAB application.

The reporting function passes the simulated event to the response routine. The response routine updates the fault table, associates an isolation status with the affected node, and records the result in the transaction history. The voltage display and status indicators then provide operator feedback. This establishes a visible correspondence between an input event and the software action recorded for it.

3.4 Event history and contract-like response

The transaction display contains timestamp, From, To, Energy (kWh), and Status fields. The original interface includes example energy-transfer rows as well as fault-processing and no-fault rows. For fault events, energy can be shown as N/A because the event represents a change in operational status rather than an energy settlement. These fields are retained in the demonstrated interface, but their presence does not imply that energy trading was evaluated.

The fault table provides a more direct event view through Time, Node, Fault Type, and Status fields. The transaction history supplies the corresponding application action. Reading the two displays together allows the operator to associate the affected node with the recorded response. A timestamp in this table records an application event; without an explicit measurement procedure it does not establish communication delay or ledger finality.

The response routine uses predefined conditions to process a fault, display an alert, assign an isolation status, and update the event history. Figure 4 shows the conceptual automation cycle. Within the prototype, an isolation status is a software state assignment. It is not feedback from a physical circuit breaker, and the automation cycle does not demonstrate cryptographic contract execution.

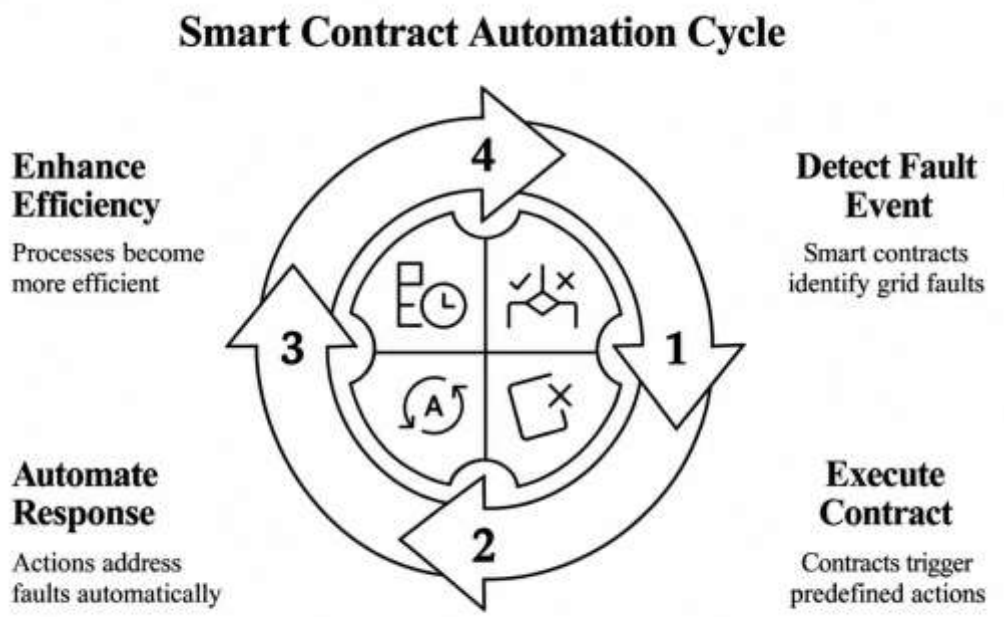

Figure 4. Conceptual cycle for event-driven response. Detection, rule evaluation, and action recording are emulated in MATLAB; the diagram does not establish execution time or efficiency gains.

3.5 Computational algorithm

The computational procedure begins by creating the interface components and initializing the network representation, transaction display, and nominal voltage values. The operator then initiates an application update. The simulator returns whether a fault has been generated, the selected node, and the voltage vector. Conditional logic routes this output to either the fault-processing branch or the no-fault branch.

Table 2. Computational sequence used by the MATLAB prototype.

Step	Operation	Application output
1	Initialize the application and ten-node graph.	Network display, nominal voltages, and initial status.
2	Populate the transaction display with example records.	Visible initial event and transaction history.
3	Receive the Start command and retain the previous displayed fault information where configured.	A new monitoring update is initiated.

Step	Operation	Application output
4	Generate the simulated event indication, selected node, and voltage values.	Inputs for the conditional response routine.
5	If a fault is indicated, display the alert and add the affected-node record with its isolation status.	Fault table and fault-status indicator are updated.
6	If no fault is indicated, show No Fault Detected and clear the current fault table.	No-fault display for the current update.
7	Update the voltage plot and insert the corresponding fault-processing or no-fault history row.	Coordinated visual and event-history output.
8	On Stop, pause the interface and reset the configured indicators and voltage display.	Paused or reset application state.

The algorithm models a sequence of information-processing actions. Its fault indicator is supplied by the event-generation routine, so the sequence should not be interpreted as a validated measurement-based detector. A later electrical implementation would need a defined acquisition interval, channel selection, detection criterion, and mapping from measured data to the affected component. The present study focuses on what the application does once the simulated event is available.

3.6 Demonstration cases and assessment criteria

The displayed evidence covers a fault-present case and a no-fault case, together with individual interface panels. The assessment considers whether the application shows an identifiable fault, assigns a response status, adds an event-history entry, and updates the voltage display. These are functional observations from the demonstrated outputs. They do not constitute repeated-trial detection statistics or a hardware protection test.

Multiple simultaneous faults and controlled restoration are relevant extensions of the architecture. The supplied single-event workflow and screenshots do not establish concurrent event processing, conflict resolution between isolation actions, or successful physical reconnection. These capabilities are therefore discussed as requirements for further development rather than counted as completed performance results.

4 Results

4.1 Network visualization and status display

Figure 5 shows the ten-node graph in the Grid Status panel. Node labels identify the application entities, while graph edges display the generated connectivity. The status lamp and Monitoring Active label provide application feedback. The plotted positions do not encode feeder length, resistance, or voltage level. The figure demonstrates that the topology can be rendered and associated with the monitoring interface.

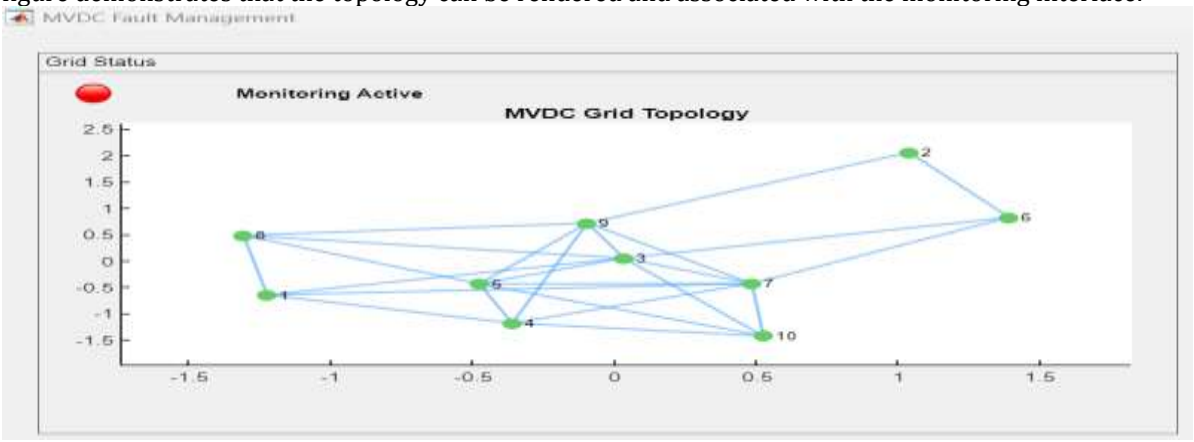

Figure 5. Grid Status panel showing the ten-node logical topology and application status. Graph coordinates are visualization coordinates.

4.2 Fault identification and response record

Figure 6 shows the Fault Management panel for a simulated event at Node 3. The alert names the node, and the table records the timestamp, Simulated Fault label, and Isolated status. The agreement between the alert and table illustrates consistent use of the node identifier in this displayed case. Because the node is supplied by the simulator, this agreement demonstrates software reporting consistency rather than independent electrical fault location.

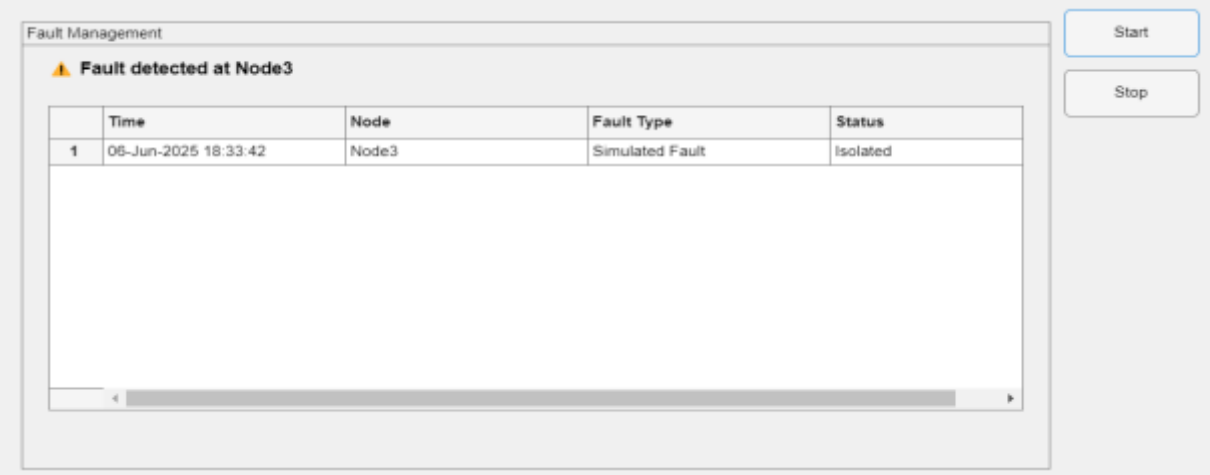

Figure 6. Fault Management panel with a simulated Node 3 event and an application-assigned isolation status.

The isolation entry is the visible output of the response routine. The screenshot does not contain breaker-position feedback, interrupted-current traces, or a verified change in the electrical network. Consequently, the result supports the processing and recording of an isolation decision at the application level. Selective disconnection and continued supply to unaffected loads require additional electrical evidence.

4.3 Ledger-like history and voltage display

Figure 7 shows the transaction panel. A row associates Node 3 with the Isolated destination and Fault Processed status, while earlier rows include example energy transfers and a No Fault status. This display provides a readable history of application entries. Some column names originate from an energy-transaction layout, so fault interpretation depends primarily on the node and status fields. A row containing None and No Fault should not be read as the disconnection of an unspecified electrical component.

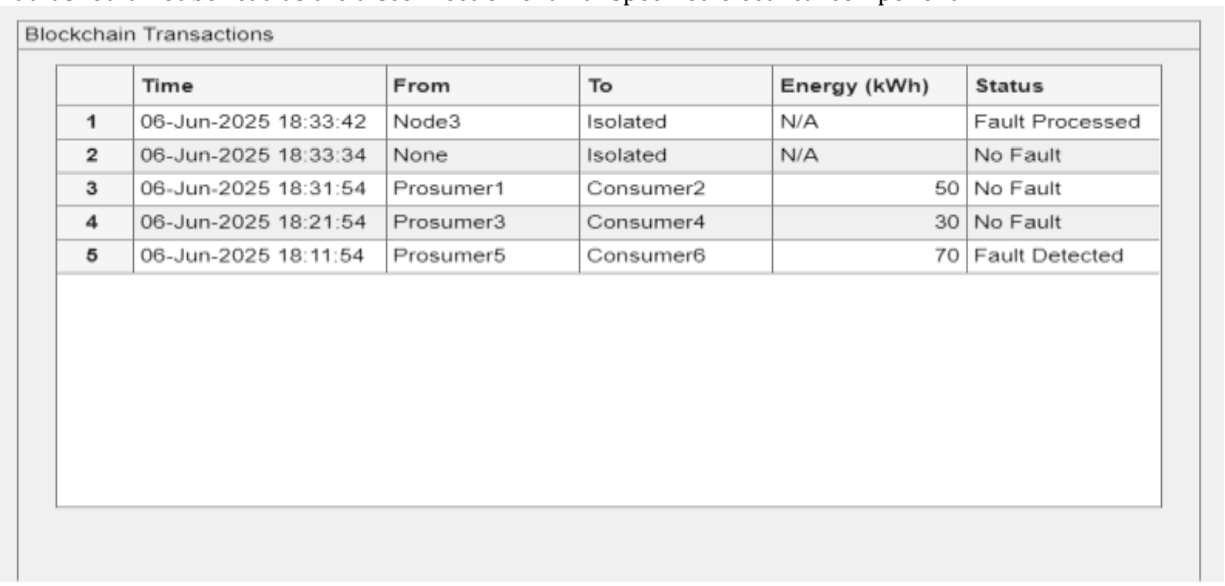

Figure 7. Transaction panel containing example energy-transfer records and application fault-processing entries. This is a local ledger-like history.

Figure 8 shows nominal voltage bars in the logs panel. Although the panel heading refers to fault isolation, the values are assigned application data. Nominal bars demonstrate a displayed state after a software update or reset; they do not independently prove that an isolated physical node has been re-energized. The distinction is relevant because an electrically disconnected bus can have a different voltage from an energized healthy bus.

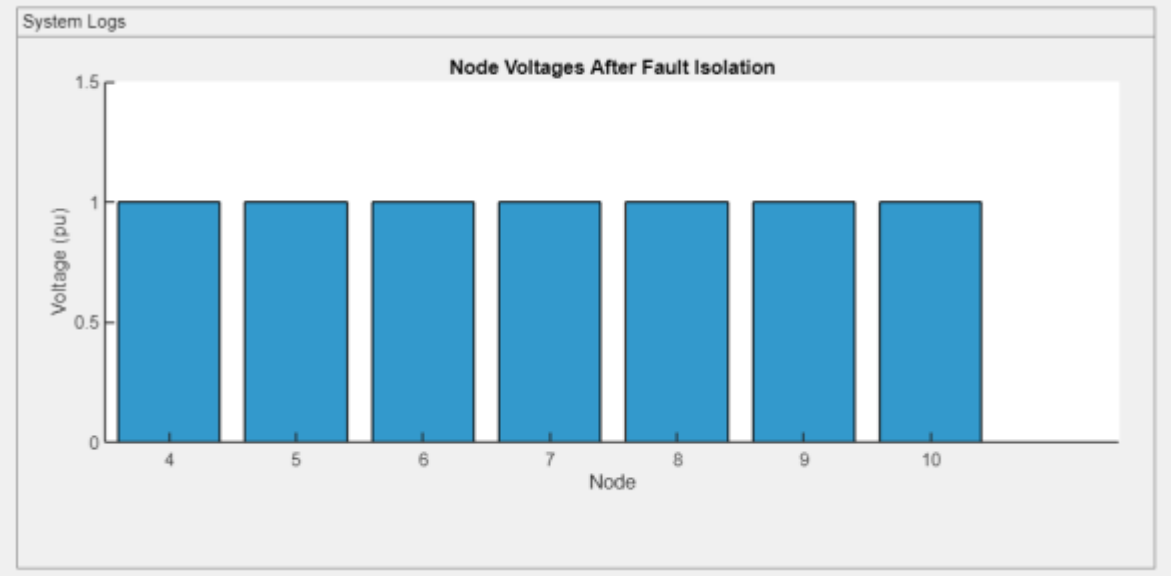

Figure 8. Node-voltage display with nominal assigned values. The panel illustrates the application state rather than a solved post-fault power-flow condition.

4.4 Integrated fault-present and no-fault cases

Figure 9 combines the four panels for a simulated Node 7 event. The fault alert and table identify the event, a transaction row records its processing, and the voltage plot contains a reduced bar at the corresponding node. The displayed voltage is approximately 0.5 pu at that node, while the other bars remain near 1.0 pu. This is an approximate reading of the plotted application output, not a newly calculated electrical result.

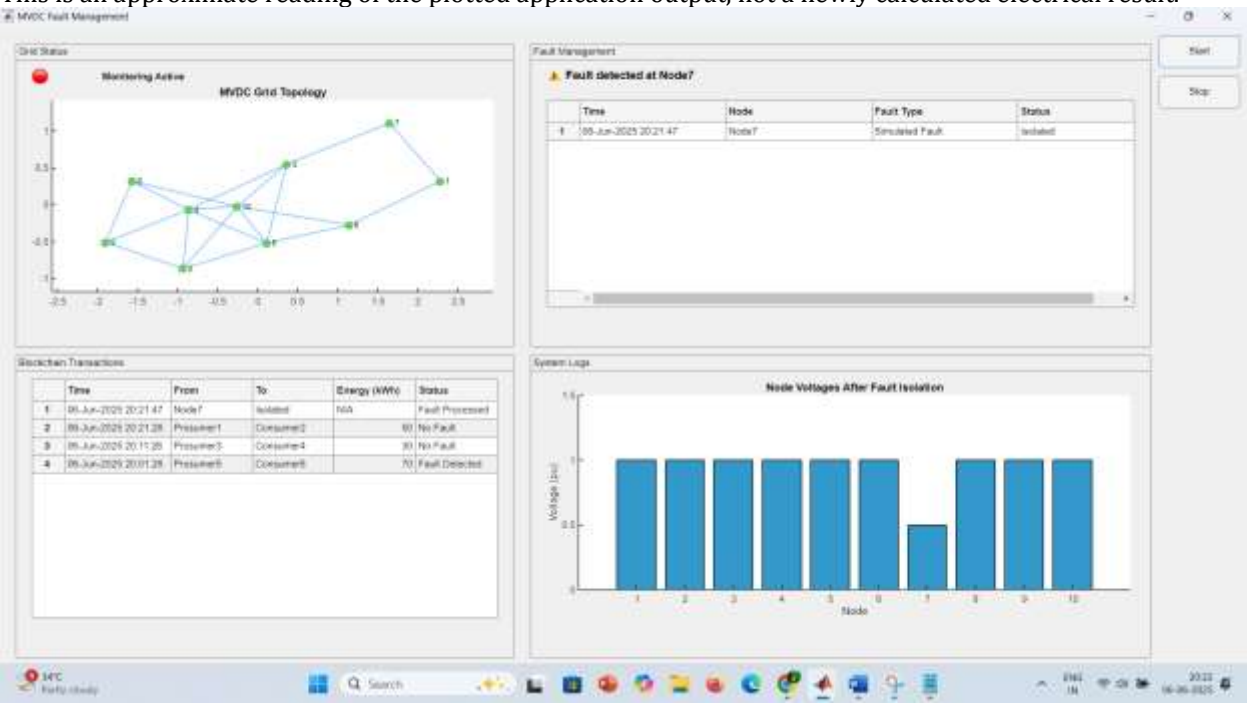

Figure 9. Integrated MATLAB interface for a simulated Node 7 fault. The event alert, response record, transaction history, and reduced assigned voltage are displayed together.

The fault table already shows Isolated while the selected voltage bar remains depressed. This illustrates why the software response label and voltage display must be interpreted as separate outputs. The interface establishes that both are updated within the event workflow; it does not demonstrate the causal electrical effect of opening a switch. A practical implementation would need to distinguish command issuance, acknowledgement, actual switch state, and subsequent measurement recovery.

Figure 10 shows a no-fault display with a green status indicator, an empty current fault table, and nominal voltage bars. The transaction history contains accumulated fault-processing and no-fault rows. These outputs illustrate the ability to show a current no-fault condition while retaining earlier event entries. They do not establish that this image and Figure 9 are synchronized samples from a measured physical recovery sequence.

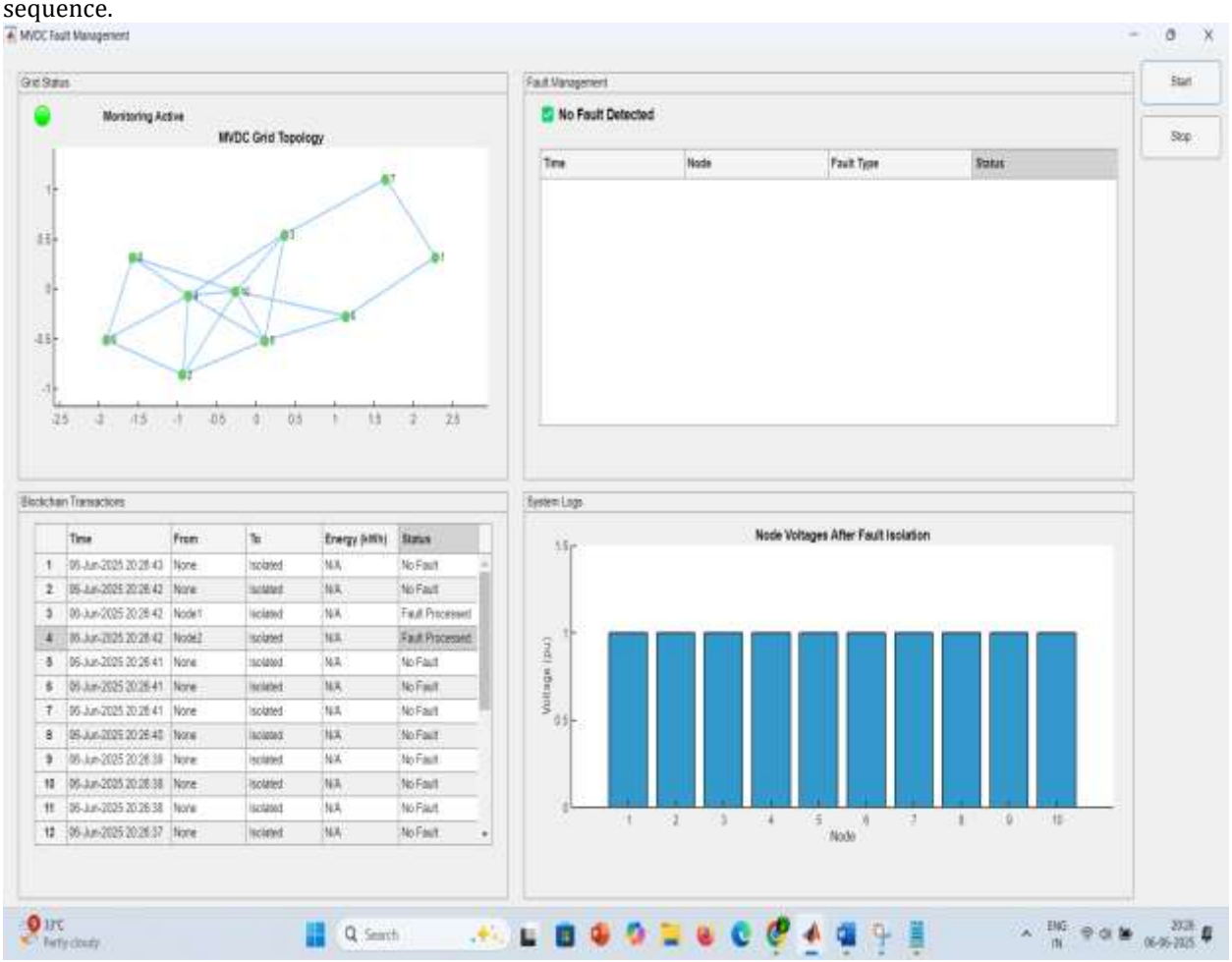

Figure 10. Integrated interface showing a no-fault condition, nominal assigned voltages, and accumulated application history.

4.5 Timing interpretation and evidence summary

The prototype description reports fault-detection times of 10–15 ms, isolation times of 1–2 s, and recovery times below 30 s. These values describe the reported application workflow. Without timestamp definitions, repeated-run records, and physical action feedback, they cannot be interpreted as verified electrical protection times or distributed-ledger finality. They are therefore not used to calculate a performance improvement over a centralized system.

Table 3 summarizes the evidence provided by the prototype. The demonstrated outcome is the coordination of the user-interface response and event history around simulated inputs. No confusion matrix, repeated-run

latency distribution, or matched centralized benchmark is established by these outputs. Consequently, a percentage reduction in fault-resolution time is not inferred from the screenshots.

Table 3. Evidence supported by the displayed prototype outputs.

Function	Observed output	Interpretation
Network display	Ten labeled nodes and generated edges.	Logical topology visualization.
Fault reporting	Node-associated alert and current fault-table entry.	Consistent reporting of a simulated event.
Response handling	Isolated and Fault Processed status fields.	Software response assignment and event recording.
Voltage feedback	Reduced assigned voltage in the fault-present view; nominal bars in the no-fault view.	Visual feedback from simulated data.
Event history	Timestamped example transactions and event rows.	Local application history; consensus and tamper resistance are not demonstrated.
Timing and comparative performance	Timing values stated in the prototype description.	Insufficient evidence for verified latency statistics or superiority over another protection system.

5 Discussion

5.1 Implications for fault-management coordination

The prototype shows how event information can be made visible across several application functions. A single simulated fault leads to a node-specific alert, a response status, a history record, and an updated voltage plot. This organization is useful when developing supervisory coordination software because the relationship between an event and its recorded action can be inspected directly. The result concerns workflow integration, rather than proof of electrical system resilience.

Node-associated reporting provides a logical starting point for distributed implementation. The application can be separated conceptually into reporting, decision, recording, and display responsibilities. However, all of these functions presently depend on one application process. Removing a central point of operational failure would require independent execution, communication, and failure-handling mechanisms. The distributed-control literature [6, 7] provides context for this distinction.

5.2 Relationship between event records and trust

The transaction table supports review of entries produced during the demonstration, but does not establish agreement between independent participants. Blockchain-related properties depend on the implemented protocol and its assumptions [8–11]. A deployment intended to support shared event validation would need explicit participant authorization, record authentication, agreement rules, and behaviour under communication failure. These are additional system properties, not consequences of naming a table Blockchain Transactions.

Data authenticity and electrical validity also require separate treatment. An authenticated report can still contain an erroneous measurement, and an immutable record can preserve an incorrect input. Measurement-security research such as Block-Phasor [17] motivates a more explicit examination of this interface. In the present prototype, the simulator generates the event and no false-data-injection or adversarial-node experiment is demonstrated.

5.3 Protection and restoration requirements

Electrical fault interruption requires an action path that is appropriate for the source, network, and protective equipment [2]. The application-level response should therefore be treated as a coordination function, not a replacement for ultra-fast primary DC protection. A future design must define which decisions

remain local, which require shared validation, and how protection behaves when the coordination or recording service is unavailable.

Restoration requires more than returning a displayed voltage to 1.0 pu. The intended installation would need checks for fault clearance, switch state, permitted topology, and acceptable electrical conditions. The prototype demonstrates resetting and displaying application values, while supervised reconnection remains a separate physical-control task. Similarly, multiple-fault operation would require an explicit event set and rules for resolving overlapping isolation or restoration requests.

5.4 Limitations and future research

The electrical representation is limited to a ten-node graph and assigned voltage values. It excludes feeder impedances, converter dynamics, load characteristics, fault-path models, and breaker behaviour. The random topology also lacks a stated feeder design basis. Consequently, the results cannot quantify voltage-drop propagation, fault current, isolation selectivity, interrupted energy, or continuity of supply. A subsequent study should couple the coordination logic to a defined electrical model before making protection-performance claims.

The software evidence is limited to the documented algorithm and interface outputs. Reproducibility would be improved by preserving the exact adjacency matrix, random seed, event-generation settings, application version, and timestamped run records. Repeated cases should use explicit definitions of detection, command, acknowledgement, and recovery times. A common test setup would then permit a meaningful comparison with a selected centralized or distributed coordination baseline.

Further work should evaluate an actual permissioned ledger and contract deployment, including access control, transaction validation, communication loss, and node failures. Gas or computational cost should be reported only for an implemented execution environment. Network-size sweeps and concurrent-event trials would be needed to support scalability claims. These extensions would turn the present functional demonstration into a basis for quantitative cyber-physical assessment, while preserving the separation between primary protection and coordination.

6 Conclusions

A blockchain-inspired fault-management framework has been demonstrated through a ten-node MATLAB App Designer prototype. The application combines logical topology visualization, simulated fault reporting, contract-like response rules, ledger-like event history, and node-voltage feedback. The displayed cases show that the interface can associate a simulated fault with a node, assign a response status, record the event, and display a no-fault condition in a subsequent application view.

The contribution is an integrated and inspectable application workflow for developing decentralized fault-management concepts. Its scope is a single-process logic prototype: electrical fault clearing, cryptographic consensus, attack resilience, and comparative timing performance are not established. A defined electrical model, independent participant execution, reproducible run data, and physical or real-time validation are the next requirements for assessing the proposed coordination architecture.

Nomenclature

Abbreviations

Abbreviation	Definition
AC	Alternating current
AI	Artificial intelligence
DC	Direct current
DER	Distributed energy resource
DFM	Decentralized fault management
DLT	Distributed ledger technology
GUI	Graphical user interface

Abbreviation	Definition
MATLAB	Matrix Laboratory
MVDC	Medium-voltage direct current
N/A	Not applicable
P2P	Peer-to-peer
pu	Per unit
SCADA	Supervisory control and data acquisition
UI	User interface

Quantities and interface terms

Term	Meaning	Unit or representation
Node voltage	Assigned voltage value displayed for a network node.	pu
Adjacency matrix	Symmetric array representing logical node connections.	Binary connection entries
Node identifier	Label linking a simulated event to its graph node and records.	Integer or Node label
Time	Timestamp shown for an application event or example transaction.	Displayed date and time
Energy	Example transaction-field quantity; not a fault-performance metric.	kWh or N/A
Isolated	Status assigned by the application response routine.	Software label
Fault Processed	History status indicating completion of the application event routine.	Software label

CRediT authorship contribution statement

Ramanarayana Nandigam: Conceptualization, methodology, software, formal analysis, investigation, writing—original draft. Ganesan Sivarajan: Validation, supervision, writing—review and editing. Veera Kumari Maddipati: Writing—review and editing. Subramanian S.: Investigation, supervision, writing—review and editing.

Funding

No external funding was reported for this research.

Declaration of competing interest

The authors report no conflicts of interest.

Data and code availability

The simulation materials, model settings, and computational code supporting this study are available from the corresponding author upon reasonable request.

Declaration of generative AI assistance

Generative AI assisted with manuscript restructuring, language editing, reference formatting, and document preparation. Responsibility for the scientific content and final approval rests with the authors.

References

[1] Coffey, S.; Timmers, V.; Li, R.; Wu, G.; Egea-Àlvarez, A. Review of MVDC Applications, Technologies, and Future Prospects. Energies 2021, 14(24), 8294. https://doi.org/10.3390/en14248294
[2] Blázquez, A.; Pérez-Molina, M.J.; Larruskain, D.M.; Iturregi, A.; Eguia, P. Fault Detection Algorithms in Medium-Voltage Direct-Current (MVDC) Grids. Applied Sciences 2024, 14(23), 11052. https://doi.org/10.3390/app142311052
[3] Muyeen, S.M.; Rahman, S., Eds. Communication, Control and Security Challenges for the Smart Grid. Institution of Engineering and Technology, London, 2017. https://doi.org/10.1049/PBPO095E
[4] Yoldaş, Y.; Önen, A.; Muyeen, S.M.; Vasilakos, A.V.; Alan, İ. Enhancing smart grid with microgrids: Challenges and opportunities. Renewable and Sustainable Energy Reviews 2017, 72, 205–214. https://doi.org/10.1016/j.rser.2017.01.064
[5] Moslehi, K.; Kumar, R. A Reliability Perspective of the Smart Grid. IEEE Transactions on Smart Grid 2010, 1(1), 57–64. https://doi.org/10.1109/TSG.2010.2046346
[6] Strasser, T.; et al. A Review of Architectures and Concepts for Intelligence in Future Electric Energy Systems. IEEE Transactions on Industrial Electronics 2015, 62(4), 2424–2438. https://doi.org/10.1109/TIE.2014.2361486
[7] Lo, C.-H.; Ansari, N. Decentralized Controls and Communications for Autonomous Distribution Networks in Smart Grid. IEEE Transactions on Smart Grid 2013, 4(1), 66–77. https://doi.org/10.1109/TSG.2012.2228282
[8] Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. White paper, 2008. https://bitcoin.org/bitcoin.pdf
[9] Musleh, A.S.; Yao, G.; Muyeen, S.M. Blockchain Applications in Smart Grid–Review and Frameworks. IEEE Access 2019, 7, 86746–86757. https://doi.org/10.1109/ACCESS.2019.2920682
[10] Miglani, A.; Kumar, N.; Chamola, V.; Zeadally, S. Blockchain for Internet of Energy management: Review, solutions, and challenges. Computer Communications 2020, 151, 395–418. https://doi.org/10.1016/j.comcom.2020.01.014
[11] Rathore, H.; Mohamed, A.; Guizani, M. A Survey of Blockchain Enabled Cyber-Physical Systems. Sensors 2020, 20(1), 282. https://doi.org/10.3390/s20010282
[12] Agung, A.A.G.; Handayani, R. Blockchain for smart grid. Journal of King Saud University – Computer and Information Sciences 2022, 34(3), 666–675. https://doi.org/10.1016/j.jksuci.2020.01.002
[13] Gensollen, N.; Gauthier, V.; Becker, M.; Marot, M. Stability and Performance of Coalitions of Prosumers Through Diversification in the Smart Grid. IEEE Transactions on Smart Grid 2018, 9(2), 963–970. https://doi.org/10.1109/TSG.2016.2572302
[14] Nezamabadi, H.; Vahidinasab, V. Microgrids Bidding Strategy in a Transactive Energy Market. Scientia Iranica 2019, 26, 3622–3634. https://doi.org/10.24200/sci.2019.54148.3616
[15] Abdella, J.; Tari, Z.; Anwar, A.; Mahmood, A.; Han, F. An Architecture and Performance Evaluation of Blockchain-Based Peer-to-Peer Energy Trading. IEEE Transactions on Smart Grid 2021, 12(4), 3364–3378. https://doi.org/10.1109/TSG.2021.3056147
[16] Mahesh, G.S.; Babu, G.D.; Rakesh, V.; Mohan, S.; Ranjit, P. Energy management with blockchain technology in DC microgrids. Materials Today: Proceedings 2021, 47, 2232–2236. https://doi.org/10.1016/j.matpr.2021.04.188
[17] Bhattacharjee, A.; Badsha, S.; Shahid, A.R.; Livani, H.; Sengupta, S. Block-Phasor: A Decentralized Blockchain Framework to Enhance Security of Synchrophasor. Proceedings of the 2020 IEEE Kansas Power and Energy Conference (KPEC), Manhattan, KS, USA, 2020, 1–6. https://doi.org/10.1109/KPEC47870.2020.9167676
[18] Lim, M.K.; Li, Y.; Wang, C.; Tseng, M.-L. A literature review of blockchain technology applications in supply chains: A comprehensive analysis of themes, methodologies and industries. Computers & Industrial Engineering 2021, 154, 107133. https://doi.org/10.1016/j.cie.2021.107133
[19] Morris, J. How the blockchain could fight grid cyber-threats. GreenBiz, 31 May 2017. https://trellis.net/article/how-blockchain-could-fight-grid-cyber-threats/ (accessed 14 September 2026).