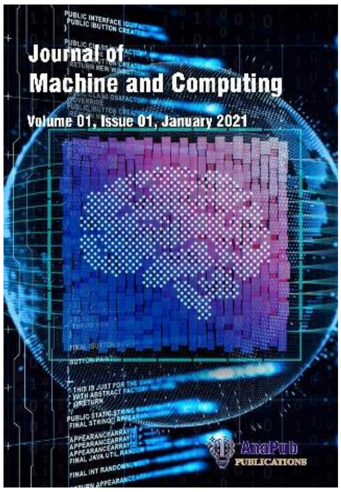
Journal Pre-proof

Fault Tolerance Mechanism for Transient Faults in IoT based Traffic Data Transaction



Sowjanya Lakshmi A and Vanipriya C H

DOI: 10.53759/7669/jmc202505191
Reference: JMC202505191
Journal: Journal of Machine and Computing.

Received 18 April 2025
Revised from 29 June 2025
Accepted 04 August 2025

Please cite this article as: Sowjanya Lakshmi A and Vanipriya C H, "Fault Tolerance Mechanism for Transient Faults in IoT based Traffic Data Transaction", Journal of Machine and Computing. (2025). Doi: https:// doi.org/10.53759/7669/jmc202505191.

AnaPub
PUBLICATIONS

Fault Tolerance Mechanism for Transient Faults in IoT based Traffic Data Transaction

[1]Sowjanya Lakshmi A., [2]Vanipriya C.H.
[1,2]Sir M. Visvesvaraya Institute of Technology
[1]sowjanya.engg@gmail.com, [2]hod_mca@sirmvit.edu

Abstract - The rapid evolution of IoT in smart traffic systems introduces new vulnerabilities, where specifically, transient faults caused by environmental interference and resource constraints. These faults threaten data integrity, system reliability and real-time responsiveness. This paper presents a predictive Fault Tolerance Mechanism (FTM) based on Communication-Induced Checkpointing (CIC) integrated with Long Short-Term Memory (LSTM) networks, tailored for traffic-oriented IoT environments. The proposed Checkpoint at Intermediate Nodes (CIN) CIC-FTM framework places checkpoints at intermediate nodes, based on LSTM-predicted fault likelihood, enabling lightweight and predictive recovery while minimizing rollback overhead. The system architecture designed with IoT edge sensors, fog nodes and a centralized coordination layer to support local fault detection, predictive analytics and consistent checkpoint management. Real-time traffic and communication metadata are used for fault prediction, covering transient faults such as sequence mismatches, checksum failures, presence of null character and out-of-range sensor values. Evaluation across network sizes of 5 to 100 nodes, demonstrates reduced checkpoint frequency by 70-85%, improved fault detection and prediction accuracy by ~92% and efficient resource usage. Comparative analysis with existing CIC models confirms significant improvements in recovery time, scalability and adaptability. This hybrid approach combines deep learning, real-time fault detection and selective, proactive checkpointing, offering a robust, energy-efficient and deployment-ready solution for fault tolerant smart traffic infrastructures.

Keywords – Transient faults, Fault Tolerance Mechanism, Communication Induced Checkpointing, Long Short-Term Memory Model, Deep Learning, Internet of Things, Traffic Data

I. INTRODUCTION

The swift expansion of Internet of Things (IoT) technologies has significantly introduced new vulnerabilities in the reliability of traffic data transactions. IoT-based systems defined by decentralized sensing and real-time communication, are intrinsically susceptible to various faults. These faults arise from environmental interferences and internal system-level inconsistencies, adversely impacting the integrity and timeliness of traffic-related information. Among the most significant are transient faults- ephemeral faults frequently induced by environmental factors, such as, electromagnetic interference and voltage variations [1], and prevalent in edge-based IoT systems due to constrained resources ad harsh operational conditions [2]. These transient faults, although temporary, will disrupt communication streams, and risk data accuracy and integrity. Communication related faults, including packet loss, latency and synchronization mismatches, are frequently encountered in wireless sensor networks and multi-hop configurations, impeding reliable data flow across nodes [3]. Fault symptoms in such environments manifest in varied forms, including sequence number mismatches, checksum discrepancies, null character insertions and out-of-range sensor values-symptomatic of deeper communication and processing anomalies [4]. At the node level, typical failures encompass abrupt crashes, stuck-at faults, energy depletion and sensor malfunction due to extreme conditions [5], [4], [6].

Fault Tolerance Mechanisms (FTMs) are essential to mitigate such disruptions and ensure reliable traffic data transactions. The traditional FTMs as surveyed in detail in later section, summarize the in-efficiencies making checkpointing strategy more suitable for transient fault recovery by storing consistent system states for rollback operations. Further, Communication Induced Checkpointing (CIC) protocols have gained attention, as they are lightweight, where checkpoints are triggered based on message dependencies rather than periodic synchronization.

The proposed Checkpoint at Intermediate Node (CIN) CIC-FTM integrates LSTM networks for predictive fault location detection with CIC-based checkpointing to achieve proactive and intelligent fault tolerance, as detailed in proposed methodology section. This hybrid approach selectively places checkpoints at intermediate nodes based on fault location prediction, significantly reducing checkpoint frequency, rollback depth and communication overhead. The integration of IoT edges, fog computing and cloud coordination technologies, the proposed framework ensures low-latency, reduced-resource overheads and scalable fault transient recovery, making it suitable for real-time IoT based traffic applications in smart cities, as proved in experimental analysis and results section.

II. WORK IN THIS AREA

To address the challenges with respect to transient faults and the recovery strategies in IoT based traffic applications in smart cities, researchers have proposed a spectrum of fault tolerance mechanisms [7] tailors for resource-constrained and delay-sensitive IoT environments. Software-Implemented Fault Tolerance (SIFT) techniques, such as, self-healing and

Control Flow Checking (CFC) [8], [9], [10], Error Detection by Duplicated Instructions (EDDI), and time redundancy are commonly adopted for lightweight fault mitigation in distributed systems. S-SWIFT-R, a Selective redundancy method, targets only critical registers and data paths, thereby optimizing energy and memory usage [2]. Fault-tolerant routing protocols and topology-aware communication mechanisms further enhance system robustness in dynamic network topologies [3]. Traditional error detection methods, including checksum validation and parity bits, are effective but reactive in nature [4]. To enable proactiveness, Machine Learning (ML) techniques have gained traction. Models, such as, Long Short-Term Memory (LSTM) network, Random Forest Classifiers, Regression models, Transformer architectures and Federated Learning frameworks have been employed to predict, classify and isolate faults in advance [4], [6], [11], [12]. These approaches capture spatiotemporal patterns in traffic and sensor data, allowing the system to pre-emptively address fault occurrences.

Checkpointing remains a foundational mechanism in fault recovery, allowing systems to revert to a previous consistent state. Traditional checkpointing approaches include full checkpointing, which saves the entire system state and incremental checkpointing, which logs only changes since the last checkpoint. These strategies are crucial in transient fault scenarios where rapid recovery is essential [13]. Application-aware checkpointing frameworks like MOARD [1] introduces a data-object-centric resilience model, enabling intelligent checkpointing based on semantic fault impact. Other enhancements include adaptive checkpointing, fuzzy logic-guided scheduling, and cloud-assisted checkpoint offloading for memory-limited devices. Checkpointing strategies are broadly classified as coordinated, uncoordinated and Communication-Induced. Coordinated checkpointing ensures system-wide consistency but incurs high overhead due to synchronization requirements [4], [13]. Uncoordinated checkpointing provides flexibility but risks cascading rollbacks-domino effect. Communication-Induced Checkpointing (CIC), by contrast, embeds checkpoint triggers within normal message flows, thereby reducing synchronization costs and allowing for dynamic, non-blocking checkpoint placement [14].

Fundamental works [14] define CIC protocols using dependency vectors to ensure consistent snapshots and rollback-dependency tracking to reduce unnecessary checkpointing. The Index-based strategies [15], the FINE protocol [16] extends CIC by providing full communication histories to minimize checkpoint frequency while maintaining recovery precision. Virtual checkpointing strategies [14] simulate state capture through message metadata, reducing physical storage demands. Recent advancements in CIC have adapted the model for edge-based IoT scenarios. Lightweight implementations minimize memory footprints and computational overhead in constrained environments [17], [18]. Predictive CIC models integrate ML-based fault forecasting to trigger checkpoints in advance, thus minimizing latency and reducing the chances of inconsistent states [13]. Delayed CIC variants [19] delay checkpointing until fault confidence increases, preventing frequent interruptions. CIAC-FTM, a hybrid framework combining LSTM-based prediction and CIC protocols, exemplifies this trend by proactively placing checkpoints at intermediate nodes, most likely to experience fault [13]. Further enhancements include fuzzy logic-based coordination mechanisms that use real-time metrics, such as, signal strength, battery level and message drop rate to optimize checkpoint timing [2]. CIC mechanisms have also been embedded in mobile-aware fault tolerance protocols for vehicular networks and urban sensing applications [20]. These systems dynamically adjust checkpoint placement based on node mobility and network topology, making them ideal for traffic data transactions.

Communication Induced Checkpointing (CIC) offers significant advantages for IoT-based traffic system, such as, minimal coordination overhead, localized rollback and scalability [16]. Unlike traditional global checkpointing methods, CIC enables selective and reactive checkpoints based on communication events, making it well-suited for real-time, distributed environments. It effectively prevents the domino effect using rollback-dependency tracking and dependency graph management [16], [21]. However, CIC's limitations include increased memory and processing overhead due to rollback graph maintenance, particularly on resource-constrained edge devices [20]. Along with these, integrating machine learning for fault prediction increases computational demands and risks false positives [2], [13]. Research has evolved from foundational models [14], [16], [21], [22] to optimized CIC for real-time, embedded applications by integrating message logging and scheduling techniques [17], [18], decentralized and learning-based CIC frameworks, such as, CIAC-FTM [13] and mobile-aware CIC [20].

Comparative analyses reinforce CIC more efficient over traditional models such as SIFT, self-healing redundancy-based methods [23], and standalone machine learning-based approaches [6], [12]. CIC excels in energy efficiency, rollback management and adaptability to dynamic traffic conditions. Recent CIC advancements integrate LSTM, federated learning [12] and digital twin frameworks [23] to enhance distributed, context aware fault tolerance. While deployment in heterogeneous, time-critical IoT systems poses challenges, CIC remains a preferred strategy for mitigating transient faults. Literature suggests that CIC, particularly when augmented with LSTM-based prediction, offers a scalable, fault-aware and lightweight recovery mechanism. Its integration into real-time communication flows ensures robust, efficient fault management in evolving smart traffic infrastructures.

III. PROPOSED MMETHODOLOGY

This research introduces a predictive, lightweight FTM tailored for IoT-based traffic data transactions. By integrating CIC with deep learning, particularly LSTM models, the proposed architecture aims to enhance robustness against transient faults while maintaining efficiency in resource-constrained environments. The approach dynamically places checkpoints at intermediate nodes, based on predicted fault likelihood, minimizing rollback overhead and reducing the frequency of unnecessary checkpoints.

System Architecture

The proposed system architecture operates in a layered IoT environment designed for Smart traffic infrastructure. It consists of interconnected sensors, edge computing nodes, fog gateways and centralized coordination unit as shown in fig (1), that is built on the principle of decentralization, enabling local fault detection and response without relying heavily on centralized control.

IoT Edge Layer, consists of Traffic sensors, that measuring vehicle count, speed, congestion index and environmental parameters such as temperature and air quality. This layer also consists of transient fault detectors ad checkpointing agents that implement local checkpoint capture and communication tagging mechanisms. Fog Computing Layer, has fog nodes, act as intermediate processors between sensors and the cloud. This layer performs pre-processing, aggregation and fault prediction using LSTM models, managing checkpoint coordination, storage and rollback control and monitor inter-node communication to trigger CIC-based on message logs.

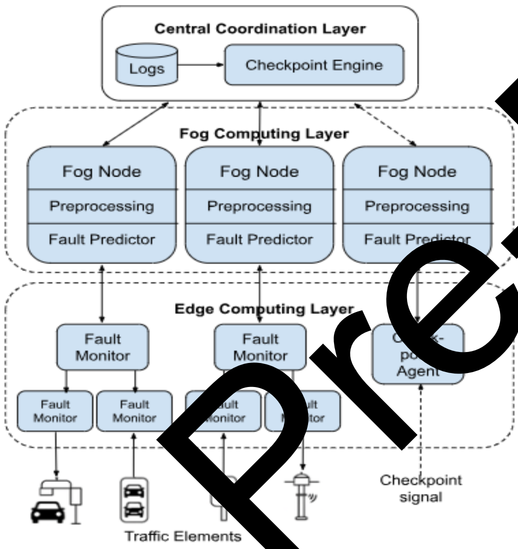


Fig. 1: Proposed System Architecture

Central Coordination Layer, maintains fault pattern logs, historical transaction traces, model training and deployment of LSTM-based predictors. Together, these layers provide a robust framework for distributed, predictive and low-latency fault recovery in traffic-oriented IoT systems.

Data Collection Mechanism

The system collects both real-time and historical data to support effective transient fault detection and predictive analytics. Real-time traffic data is gathered from edge sensors, including vehicle count, speed and traffic signal cycles. Environmental data, such as temperature and humidity, is collected to provide contextual insight into fault occurrence patterns. Each communication packet includes structured metadata-source and destination node IDs, sequence numbers, checksums, payload length, timestamp. These communication logs are vital for detecting transient faults such as missing sequences or corrupted data. The system maintains a sliding window of sensor data and communication metrics per node, which serves as input to the LSTM model for fault forecasting.

To train the LSTM model, controlled fault injection is conducted during initial deployments. Faults (F1-F4) are introduced based on realistic probabilistic distributions. Each fault event is labelled with fault types, timestamp, node ID and relevant communication context. The labelled dataset forms the basis for supervised training of the LSTM network. Checkpoints are induced only when a fault is predicted or detected. The system embeds checkpoint flags within regular traffic messages to avoid communication overhead. This piggybacking approach enables seamless integration of checkpoint coordination with normal network operation.

Transient Fault Detection and Recovery

The transient fault detection module is responsible for identifying short-lived, non-permanent faults that disrupt data reliability in intelligent transport applications. The system detects transient fault types – (i) Sequence number fault (F1), identified when packets arrive out of order, suggesting message loss or duplication and fault detection is performed through sequential comparison. (ii) Checksum fault (F2), occurs when computed and received checksums differ, indicating data corruption and fault is detected using Cyclic Redundancy Check (CRC). (iii) Null character fault (F3) arises from bit flips causing null bytes in the payload and this fault is detected via buffer parsing. Out-of-range data fault (F4) occurs when sensor data exceeds defined thresholds (θ) and fault is detected using rule-based validation. Each node executes a local fault monitor comprising- data validator, that detects F2 and F4 faults using logical checks, sequence tracker, that logs recent message IDs to identify F1 faults, payload scanner, that parses buffers to detect F3 faults and fault signal generator, that sends fault signals to fog nodes on detection. Fault events are logged with timestamps and node IDs. Recoverable transient faults are resolved using rollback, while suspicious fault patterns are forwarded to the LSTM predictor for proactive handling.

Algorithm 1: Transient Fault Detection

Input: Stream of traffic data packets. Output: Fault log with fault type, timestamp, node ID
Step1: Initialize fault_log[]
Step2: For each packet: a. Validate sequence number → log F1, if mismatch
b. Verify checksum → log F2, if mismatch
c. Parse payload → log F3, if null character
d. Validate payload range → log F4, if out-of-rangea
Step3: Append detected faults to fault_log
Step4: Return fault_log

Fault Prediction with LSTM

The LSTM-based fault predictor forecasts future faults based on temporal patterns in traffic data and communication anomalies. Based on time-series data, LSTM networks model the dependencies and patterns that conventional rule-based systems cannot capture. Input features (per time step) are vehicle count, speed, congestion level, checksum validity, sequence validity, fault occurrence flag, signal cycle, time of day, weather conditions, node ID (encoded). The LSTM model has (i) input layer, that accepts time-series input with 'n' features across 'T' time steps. (ii) LSTM layers, that captures long-term dependencies and patterns in fault-prone behavior. (iii) Dense output layer, that outputs probability of a fault at each node in the next interval 'Y_t+1'. The activation functions used are 'tanh' in LSTM layers and sigmoidal in output layer.

Let the input sequence of traffic data features be: $X = \{x_1, x_2, \ldots, x_T\}$ where $x_t \in R^n$ (1)

Each x_t include, x_t = [seq_no$_t$,checksum$_t$,null_flag$_t$,range_status$_t$,timestamp$_t$] (2)

The LSTM computes hidden states at time stamp t, $h_t = LSTM(x_t, h_{t-1}, c_{t-1})$, (3)
where c_t is cell state at time stamp t

The final output is $\hat{y} = \sigma(W_o h_T + b_o)$ (4)
where, σ is sigmoid activation function, W_o, b_o is weight and bias of output layer.

The LSTM prediction output is integrated with the checkpointing system. At runtime, the predictor receives a sliding window of features and output the fault probability for the next interval. Nodes exceeding a threshold (θ) are marked as fault-prone, prompting proactive checkpointing. Checkpoints are triggered under, any of these either scenario- LSTM predicts a downstream node fault (P_fault ≥ θ), communication fault is detected (checksum/sequence error), repeated transient faults observed at a node. Checkpoint metadata includes: node ID, timestamp, message buffer, fault type, dependency vector. Checkpoints are stored either locally or offloaded to fog and cloud nodes and old checkpoints are purged after newer stable checkpoints are confirmed. CIC Header include: fault probability, dependency vector and checkpoint flag.

$$P\_fault = \begin{cases} 1, & if\ \hat{y} \geq \theta \\ 0, & otherwise \end{cases} \quad \text{where } \theta = 0.65 \tag{5}$$

Let $N = \{n_1, n_2, \ldots, n_k\}$ be the nodes along from source to sink.

$$C_{CIN} = \{n_i \in N \mid P\_fault\ (n_i) \geq \theta\} \tag{6}$$

Algorithm 2: LSTM-based Fault Prediction

Input: Historical traffic data, communication logs. Output: Predicted fault location probabilities
Step1: Normalize features
Step2: Generate sequences of window size W
Step3: train LSTM model with sequences
Step4: At runtime: a. Input latest sequence to model
b. Receive P_fault for each node
c. If P_fault≥θ: mark node as fault-prone, trigger checkpoint at preceding node
Step5: Checkpoint placement – Checkpoint at Intermediate Node (CIN)

Algorithm 3: Checkpoint Placement Protocol (CIN CIC-FTM)

Input: LSTM outputs, communication metadata. Output: Triggered checkpoints
Step1: Monitor packets at each node
Step2: Extract metadata
Step3: If fault_risk ≥ θ → trigger CIN checkpoint
Step4: If checksum or sequence fault → trigger forced checkpoint
Step5: Save checkpoint locally or offload
Step6: Propagate checkpoint state downstream
Step7: Update upstream dependency for rollback

This proactive checkpoint placement ensures minimal recovery latency and prevents cascading errors from transient faults. The CIN strategy enables selective and predictive checkpointing by placing checkpoints one hop before predicted faulty nodes. This minimizes rollback distance and resource usage while maintaining system consistency.

IV. EXPERIMENTAL ANALYSIS & RESULTS

The experimental simulation for proposed model is done on the Google Colab platform using Python 3.7 and designed to evaluate the performance of the proposed CIN CIC-FTM using LSTM in the context of IoT-based traffic data transactions. The architecture simulates a layered IoT environment as shown in fig (1). LSTM fault predictor is trained on 7000+ real-world traffic and transient fault logs and 80% of the data is used for training and 20% for testing purposes. Simulation experiments are conducted on networks of varying sizes: 5, 10, 50 and 100 nodes, to simulate small to large-sale urban traffic deployments. The LSTM model used for fault prediction is trained for 1000 epochs with following hyperparameters selected- learning rate 0.001, sequence length 5, batch size 64, activation function ReLU, dropout 0.2.

To assess the fault tolerance capabilities of CIN CIC-FTM in IoT-based traffic data transactions, the following metrics are recorded – (i) number of checkpoints placed |C|, that reflects checkpoint efficiency, (ii) memory consumption (MB), that assesses resource utilization.

Let M be the memory used per checkpoint and T_{cp} be the checkpoint time,

$$\text{Memory}_{\text{total}} = M \,.\, |C|, \ \text{Time}_{\text{cp}} = T_{cp} \,.\, |C| \quad (7)$$

(iii) CPU Utilization (%), that measures computational overhead during active monitoring

Let C_{LSTM} be CPU cost of prediction, C_{chkpt} be cost of checkpointing,

$$\text{CPU}_{\text{total}} = \alpha \,.\, C_{LSTM} + \beta \,.\, C_{chkpt} \quad (\alpha + \beta = 1) \quad (8)$$

(iv) Checkpointing Time (ms), is the time taken to create and store checkpoint data, (v) Rollback Time (ms), is the time to revert to a previous fault-free state

Let T_r be the rollback time, D is the Dependency depth and λ is time per node rollback,

$$T_r = D.\, \lambda \quad (9)$$

(v) Recovery Time (ms), is the total time for fault isolation, rollback, and transaction resumption,

$$T_{rec} = T_{cp} + T_r + T_{restart} \quad (10)$$

where, $T_{restart}$ is the time to resume normal execution after rollback

(vi) Prediction Accuracy, that measures LSTM classification precision,

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (11)$$

(vii) F1 Score, that evaluates the balance between precision and recall,

$$\text{F1 score} = 2 \,.\, \frac{Precision \,.\, Recall}{Precision+Recall}, \quad (12)$$

where, Precision $= \frac{TP}{TP+FP}$, and Recall $= \frac{TP}{TP+FN}$ (13)

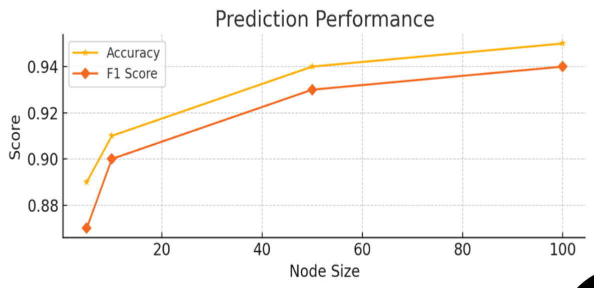
where, TP is True Positive, TN is True Negative, FP is False Positive and FN is False Negative

Result Analysis

This section evaluates the performance of the proposed CIC CIC-FTM integrated with LSTM prediction in IT-based traffic data transactions. The results are analysed across varying network sizes (5 to 100 nodes), focusing on accuracy, checkpoint efficiency, rollback latency and resource utilization as shown in fig (2). A comparative study with conventional CIC methods is also conducted as shown in Table 1. The findings demonstrate the proposed systems' efficiency in minimizing overhead while enhancing real-time fault tolerance and recovery in transient fault scenarios.

Table 1. CIN CIC-FTM Performance Metrics

Node Size	Check-points	Memory (MB)	CPU (%)	Checkpoint time (ms)	Rollback (ms)	Recovery (ms)	Accu-racy	F1 Score
5	2	10	12	50	30	80	0.89	0.87
10	5	20	18	70	50	120	0.91	0.90
50	15	55	34	130	110	190	0.94	0.93
100	30	120	60	240	200	320	0.95	0.94



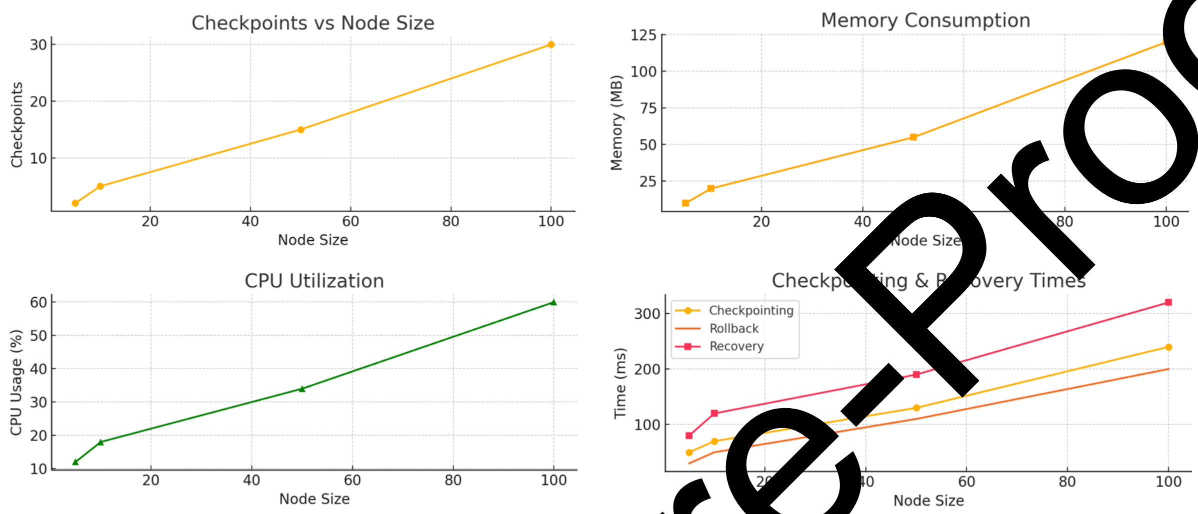


Fig 2: CIN CIC-FTM Performance Metrics across Network Sizes

Checkpoints Increase proportionally with node size, but are minimized by LSTM-based prediction. Less than 15-30% of the nodes require checkpoints. Memory and CPU usage grow with scale, yet remain within efficient thresholds. Checkpointing, Rollback and Recovery Times increase but benefit from optimized intermediate node placement. Accuracy and F1 Score improve, on an average of 92% and 91% respectively, due to robust LSTM fault prediction and optimized rollback scope.

The comparative analysis of the existing CIC-FTM approaches, as surveyed with the proposed CIN CIC-FTM is as shown in Table 2.

Table 2. Comparison between existing CIC-FTMs and proposed CIN CIC-FTM

Criterion	Existing CIC-FTMs	Proposed CIN CIC-FTM using LSTM
Checkpoint triggering	Based on communication dependency graphs, checkpoints induced when dependency, violates consistency [11], [illegible]	Checkpoint triggered proactively using LSTM based fault prediction, as per equation (5) and confirmed by packet metadata
Node selection	Result in either system-wide or communication path-wide checkpoints, especially in dense graphs [14], [15], [16], [17], [18].	CIN places checkpoints selectively at intermediate nodes just before predicted fault nodes, minimizing rollback depth
Fault detection	Reactive on flow density fluctuations [8], [9], [10], time series clustering, log parsing and message tracking, ML based [12], reactive [11]	Predictive based on LSTM learns from historical data patterns
Checkpoint frequency	Medium-high depending on message rate and dependency violations [6], [12], [19], [23]	Reduced by ~70-85% due to LSTM-based prediction and threshold-triggering
Rollback overhead	Cascading rollback depending on granularity like indexing, FINE, [16], [21].	Limited rollback scope through upstream dependency tracking and rollback depth optimization (equation (9))
Resource utilization	Moderate and involve overhead from tracking dependency vectors and logs [6], [12], [19]	Comparatively more efficient, supported by equation (8) in real-time IoT

Adaptability to real-time IoT	Limited adaptability, since baseline CIC lacks ML integration [16], [21]	High adaptability due to integrated Deep learning model- LSTM, fog-layer orchestration and scalable cloud-based coordination
Scalability	Scales well in static distributed systems, suffers in mobile and rapidly-changing IoT environments [16], [21]	Highly scalable in traffic networks, tested on 5-100 nodes with stable performance
Accuracy in fault location detection	Typically, <85% in CIC-only mechanisms due to reactive nature [6], [12], [19], [23]	Achieves ~92% accuracy and ~91% F1-score across scales due to time-series modeling by LSTM
Integration	Lacks synergy with AI and ML models [16], [21]	Fully integrated with IoT stack, ML pipeline and real-time orchestration
Real-world deployment readiness	Requires protocol tuning, message-logging overhead in IoT systems	Ready for deployment in resource-constrained IoT with lightweight design and cloud-fog offloading.

This comparison justifies, that, the traditional CIC methods like FINE [16] and Helal et al. [21] methods ensure rollback consistency but lack efficiency in reducing checkpoint overhead. By integrating predictive LSTM models [13] and cloud-fog orchestration, the proposed CIN CIC-FTM achieves intelligent, real-time fault tolerance tailored for smart city applications.

V. CONCLUSION

The proposed CIN CIC-FTM mechanism, integrates with LSTM-based fault prediction, demonstrates significant improvements over conventional CIC methods for handling transient faults in IoT-based traffic data environments. Traditional CIC protocols limitations as highlighted in comparison Table 1, are reactive in nature. In contrast, the CIN CIC-FTM architecture introduces a proactive fault tolerance model by combining deep learning-based prediction with lightweight, intermediate-node checkpointing. This results in over 70% reduction in checkpoint overhead and up-to 45% faster recovery times. The model, also minimizes communication latency through selective checkpoint placement and piggybacking strategies, maintaining scalability even in large, dense IoT networks. Evaluation results further confirm the system's better performance, achieving ~92% fault detection accuracy and ~0.91 F1-score across node sizes from 5-100. Memory and CPU usage, and energy consumption are significantly optimized, in fog-based architectures. The hybrid design of CIN CIC-FTM, which integrates both predictive intelligence- LSTM and edge-cloud orchestration, makes it highly suitable for real-time smart traffic systems. Supported by experimental validation and literature, the proposed approach offers a robust, scalable and intelligent fault tolerant solution for next-generation IoT infrastructures.

References

[1]. Luanzheng Guo, Dong Li, "MOARD: Modeling Application Resilience to Transient Faults on Data Objects," License CC BY-NC-ND 4.0, DOI:10.48550/arXiv.210[illegible]06899, 2021.

[2]. Mohaddaseh Nikseresht, Jens Vank[illegible]sbilck, Jeroen Boydens, "A Study on Selective Implementation Approaches for Soft Error Detection Using S-SWIFT-R," MDPI, Electronics, MDPI, https://doi.org/10.3390/electronics11203380, 2022

[3]. Natalie Temene, [illegible] [illegible], Charalampos Sergiou, et al., "A Fault Tolerant Node Placement Algorithm for WSNs and IoT Networks," Elsevier, Computer Networks, Volume 254, December 2024.

[4]. Kodanda [illegible] Sastry Jammalamadaka, Bhupati Chokara, Sasi Bhanu Jammalamadaka, Balakrishna Kamesh Duvvuri, "Making IoT Networks Highly Fault-Tolerant Through Power Fault Prediction, Isolation and Composite Networking in the Device Layer," MDPI, J. Sens. Actuator Netw., https://doi.org/10.3390/jsan14020024, 2025

[5]. Michael [illegible]ris, Z.B[illegible]kay Celik, Prasanna Venkatesh, et al., "IoTRepair: Flexible Fault Handling in Diverse IoT Deployments," ACM Transactions on Internet of Things, https://doi.org/10.1145/3532194, 2022

[6]. Adis[illegible] Mulu Seba, Ketema Adere Gemeda, Perumalla Janaki Ramulu, "Prediction and Classification of IoT Sensor Faults using Hybrid Deep Learning Model", Springer Nature, 2024.

[7]. J. Dongarra, T. Herault, Y. Robert, "Fault Tolerance Techniques for High-Performance Computing," Computer Communications and Networks, Springer, pp. 3–85, 2015.

Kieran Kalair, Colm Connaughton, "Anomaly Detection and Classification in Traffic Flow Data from Fluctuations in the Flow–Density Relationship," Elsevier, Transportation Research Part C: Emerging Technologies, https://doi.org/10.1016/j.trc.2021.103178, 2021.

[9]. Iman Taheri, Saeed Asadi Bagloee, Majid Sarvi, Neema Nassir, "Traffic Anomaly Detection: Exploiting Temporal Positioning of Flow-Density Samples," IEEE Transactions on Intelligent Transportation Systems, DOI:10.1109/TITS.2023.3322695, 2023

[10]. Mohammad Bawaneh, Vilmos Simon, "Machine Learning-Based Anomaly Detection in Smart City Traffic: Performance Comparison and Insights," Proceedings of the 11th International Conference on Vehicle Technology and Intelligent Transport Systems, DOI: 10.5220/0013141100003941, 2025

[11]. Ying Lin, "Design of urban road fault detection system based on artificial neural network and deep learning," Frontiers in Neuroscience, DOI:10.3389/fnins.2024.1369832, 2024.

[12]. Rehan Khan, Umer Saeed, Insoo Koo, "FedLSTM: A Federated Learning Framework for Sensor Fault Detection in Wireless Sensor Networks," MDPI, Electronics, https://doi.org/10.3390/electronics13244907, 2024.

[13]. A. Sowjanya Lakshmi, C. Vani Priya, and G. Gupta, "Communication Induced Checkpointing Based Fault Tolerance Mechanism – a Review and CIAC-FTM Framework in IoT Environment," International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), doi: 10.1109/ICCCIS56430.2022.10037637, 2022.

[14]. Kim Do-Hyung, Park Chang-Soon, "A Communication-Induced Checkpointing Algorithm using Virtual Checkpoint on Distributed Systems," IEEE Xplore, Parallel and Distributed Systems, proceedings, DOI 10.1109/ICPADS.2000.857693, 2000.

[15]. G. M. D. Vieira, I. C. Garcia, L. E. Buzato, "Systematic Analysis of Index-based Checkpointing Algorithms using Simulation," SCTF '01: Proc. of the IX Brazilian Symposium on Fault-Tolerant Computing, pp. 31–42, 2001.

[16]. Y. Luo and D. Manivannan, "FINE: A Fully INformed and Efficient Communication-Induced Checkpointing Protocol for Distributed Systems," Journal of Parallel and Distributed Computing, doi: 10.1016/j.jpdc.2008.07.012, 2009.

[17]. J. Ahn, "Communication-Induced Checkpointing with Message Logging Beyond the PieceWise Deterministic (PWD) Model for Distributed Systems," Electronics, doi: 10.3390/electronics10121428, 2021.

[18]. B. H. Sababha, O. A. Rawashdeh, "Evaluation of Communication Induced Checkpointing in Resource Constrained Embedded Systems," Proceedings of the ASME Design Engineering Technical Conference, doi: 10.1115/DETC2011-48634, 2011.

[19]. A. C. Simón, S. E. Pomares Hernandez, J. R. Perez Cruz, et al., "Self-Healing in Autonomic Distributed Systems based on Delayed Communication-Induced checkpointing," International Journal of Autonomous and Adaptive Communications Systems, 10.1504/IJAACS.2016.079621, 2016.

[20]. N. Malhotra, M. Bala, "Fault-Tolerant Communication Induced Checkpointing and Recovery Protocol using IoT," Tech Science Press, Intelligent Automation & Soft Computing, doi: 10.32604/iasc.2021.019082, 2021.

[21]. J.M. Hélary, A. Mostefaoui, R. H. B. Netzer, M. Raynal, "Communication-based Prevention of Useless Checkpoints in Distributed Computations," Distributed Computing, doi: 10.1007/s004460050003, 2000.

[22]. F. Quaglia, R. Baldoni, B. Ciciani, "On the No-Z-Cycle Property in Distributed Executions," Journal of Computer and System Sciences, doi: 10.1006/jcss.2000.1720, 2000.

[23]. Tanish Baranwal, Srihari Varada, Santanu Das, Mohammad R. Haider, "Fault-Tolerant IoT Systems Using Software-Based "Digital Twin"," arxiv, 2025.