Real-Time Fault Tolerance in Edge Computing within Dependable and Distributed Systems

Introduction

Edge computing redefines dependable and distributed systems by decentralizing computation from centralized cloud infrastructures to the network's periphery, closer to data sources such as IoT devices, sensors, and mobile endpoints [2]. This shift enables low-latency processing, reduces bandwidth demands on core networks, and enhances system resilience—attributes critical for applications like autonomous vehicles, smart grids, and real-time healthcare monitoring [6]. However, achieving real-time fault tolerance in edge environments, where continuous operation must persist despite hardware failures, network disruptions, or software errors, presents unique challenges [5]. Fault tolerance in this context must balance safety (preventing catastrophic outcomes), liveness (ensuring ongoing functionality), and stringent timing constraints, all within resource-constrained, heterogeneous edge architectures. Unlike cloud systems, which leverage abundant resources for redundancy, edge computing demands innovative strategies to handle transient faults, Byzantine failures, and dynamic network conditions with limited computational power and energy reserves.

This literature review synthesizes insights from 19 studies (2012–2025) including peer-reviewed studies to examine real-time fault tolerance in edge computing within dependable and distributed systems. Structured thematically around three core areas—fault detection techniques, fault recovery strategies, and the interplay of real-time constraints with dependability—the review traces the evolution of these methods, critiques their limitations, and identifies gaps, particularly in resource-scarce or dynamic environments. It concludes by proposing future research directions to address these shortcomings, emphasizing adaptive, lightweight, and energy-efficient solutions tailored to edge-specific challenges.

Background

Fault tolerance is vital in dependable distributed systems, ensuring uninterrupted service amid failures. In edge computing, it must address resource constraints, network variability, and real-time demands [5], aligning with the Dependability Tree's framework of impairments (faults, errors, failures), means (prevention, tolerance, removal, forecasting), and attributes (reliability, availability, safety). Traditional techniques like replication, checkpointing, and Triple Modular Redundancy (TMR) with Markov-modelled reliability thrive in resource-rich cloud environments but falter in edge architectures due to energy limits, heterogeneous hardware, and unpredictable faults, including transient or Byzantine behaviours from compromised nodes.

Edge computing's proximity to data sources minimizes latency and eases cloud reliance. Yet, it introduces trade-offs: limited processing power (e.g., IoT devices with 32-bit microcontrollers vs. cloud servers with multicore CPUs), dynamic network topologies, and heightened security risks complicate fault tolerance. Real-time systems, categorized as hard (e.g., autonomous vehicles requiring sub-10ms responses) or soft (e.g., video streaming tolerating 100ms delays), demand rapid fault detection and recovery with minimal overhead. Synchronized clocks enable event ordering across distributed nodes, ensuring timing consistency critical for dependability in failure-prone edge environment.

Challenges in Real-Time Fault Tolerance in Edge Computing

Edge computing's promise of low-latency, dependable processing encounters significant obstacles in achieving real-time fault tolerance. A central challenge is the latency-redundancy trade-off. Replication across nodes enhances reliability but increases latency and resource consumption, undermining real-time performance. For example, TMR ensures fault masking in cloud systems but introduces delays (e.g., 20ms voting overhead) and energy costs (e.g., 3x power draw) intolerable for hard real-time edge applications like autonomous vehicles processing 1GB/s of sensor data [3]. Replication also incurs latency penalties during fault recovery, such as system rollbacks or message backups, with studies reporting delays up to 50ms in edge networks due to propagation and contention [12]. These penalties jeopardize safety-critical timing thresholds (e.g., <10ms end-to-end latency) [19].

Resource constraints further complicate fault tolerance. Edge devices such as IoT sensors and mobile endpoint, lack the computational power and energy reserves of cloud servers, so failover options can be more limited [3]. Research highlights that resource scarcity limits fault detection accuracy, as lightweight algorithms struggle to identify transient or Byzantine faults without sufficient processing capacity [18].

Dynamic environments exacerbate these issues. Edge networks face unpredictable node mobility (**heterogeneous resources** and **variations in geographical positions**, linked to localized processing), network partitions, and intermittent connectivity [9]. Traditional fault recovery, reliant on stable topologies, falters as edge nodes join or fail unexpectedly. Methods like node redundancy, task replication, and periodic checkpointing can lead to inefficiencies such as excessive overheads and resource contention in volatile settings. [14]. This dynamism demands adaptive strategies, Satyanarayanan [19] criticizes static and centralized approaches, especially those relying heavily on cloud data centres, for their limitations in responsiveness and adaptability, acknowledges challenges managing dispersed infrastructures and ensuring security, which could create gaps in resilience [19]. Together, these challenges (latency-redundancy trade-offs, resource limitations, and environmental variability) underscore the need for innovative, edge-specific fault tolerance tailored to real-time demands.

Fault Detection in Edge Computing

Fault detection is pivotal for real-time fault tolerance in edge computing, where distributed systems face resource constraints and network heterogeneity. A prerequisite for ensuring dependability in real-time distributed systems where detectors- components checking system predicates like parity or timing thresholds are critical [9] proposes a decentralized LAAECHA (Latency-Aware Algorithm for Edge Computing with High Availability), which emphasizes fault detection, node repairs, and edge node replacements using backup nodes ensuring system reliability without relying on a centralized mechanism [9]. Resource-limited devices complicate timely detection, critical for dependability [3]. Gia [6] exemplifies edge-based fault detection in healthcare IoT, extracting ECG features (e.g., heart rate, P wave, T wave) locally at smart gateways using fog computing, based on a lightweight wavelet transform mechanism lightweight algorithms enhance detection. This achieves low-latency processing, reducing bandwidth usage and ensuring real-time responses. However, its reliance on stable conditions limits applicability in dynamic edge settings [7] and high node density can lead to network congestion, which may increase latency and cause data packet loss[Y8].

Machine learning (ML) highlights edge-deployed models predicting failures from usage patterns [14]. Yet their resource demands challenge real-time constraints with studies like [18] revealing a gap: these methods excel in controlled setups but struggle with unpredictable failures in heterogeneous, dynamic and diverse networks, such as those used in the metaverse are subject to node failures, communication faults, and Byzantine faults, all of which are exacerbated by nature of these networks[18] and hostile military operations, natural disaster recovery IoT deployments [19]. Future research should focus on adaptive, lightweight detectors that adjust to edge variability, balancing speed and accuracy for dependable systems.

Recovery Strategies for Fault Tolerance

Recovery mechanisms are vital for maintaining system continuity in dependable edge applications once faults are detected, often using correctors to restore invariants. Replication and redundancy, akin to N-version programming (NVP), mitigate single-point failures. Shi and Dustdar [1] advocate task replication across edge nodes, while Cao [8] applies this in vehicular networks with a scheduling algorithm redistributing tasks to nearby nodes, ensuring real-time performance. This mirrors Triple Modular Redundancy (TMR), where two-of-three nodes ensure functionality despite a failure, supported by multi-tier control nodes (e.g., main and sub-SDNCs) resembling TMR’s voting logic for reliability in latency-sensitive systems.

However, replication’s scalability is constrained. Abbas [10] notes its overhead in mobile edge computing, doubling bandwidth (e.g., 10MB/s to 20MB/s) and straining storage, pushing latency beyond real-time limits (e.g., 150ms vs. 100ms target). Task offloading, shifting workloads to the cloud or nearby edges, offers an alternative. Zhou [4] explores edge intelligence with cloud redundancy and hybrid models blending local execution with selective offloading, enhancing resilience. Federated learning further supports fault tolerance by enabling local training without centralized data, while decentralized cloud-edge-device coordination reduces single-node dependency. Yet, Porambage [17] warns offloading struggles in disconnected scenarios (e.g., remote traffic systems), where link failures raise latency, suggesting FloodSet-style consensus as a fallback.

Replication thrives in high-bandwidth contexts but falters in dense, low-resource networks, where storage overhead (e.g., 1GB/node) is impractical. Offloading's reliance on stable networks often clashes with edge realities. Future recovery should adaptively toggle between replication and offloading based on real-time conditions, potentially leveraging Practical Byzantine Fault Tolerance (PBFT) phases to manage malicious failures and optimize resources efficiently.

Real-Time Constraints and Dependability

Edge computing prioritizes low-latency processing to meet stringent real-time constraints, setting it apart from traditional distributed systems [3]. Fault-tolerant edge applications must function within tight deadlines despite disruptions. Satyanarayanan [11] exemplifies this with edge processing for autonomous driving, where collision avoidance demands decisions within 100ms—a requirement challenged by dynamic failures like network delays or hardware faults [12]. Research by [10] demonstrates that edge nodes in IoT deployments can mitigate such faults using predictive caching, though this assumes stable connectivity often absent in rural settings.

Security further complicates dependability. Ranasinghe [16] frames attacks like Trojans as faults disrupting timing, requiring detection within milliseconds to preserve real-time performance. Their STRIP defence mechanism detects these threats with a low overhead of 6.125ms, yet this still strains ultra-low latency constraints in critical systems [12]. Meanwhile, [10] explores lightweight encryption for edge devices, reducing latency penalties, though scalability falters under high fault rates with resource constraints amplify these challenges.

Current solutions reveal a trade-off between fault tolerance and timing. [14] and [5] advocate adaptive resource allocation to maintain sub-100ms performance, yet their reliance on high-spec hardware limits broader adoption. Future work must prioritize lightweight, scalable mechanisms to ensure dependability across diverse edge environments [15].

Open Problems and Future Directions

Fault tolerance in edge computing grapples with energy efficiency, scalability, and lightweight security. Redundancy and checkpointing incur high energy costs—e.g., 100mW per cycle in IoT nodes—straining battery life [3, 13]. Scalability falters as networks grow, with replication overhead rising linearly (e.g., 1GB/node in 100-node setups) and detection failing under node proliferation [7], 12, 18]. Security against Byzantine threats remains resource-intensive, with lightweight detectors achieving only 70% accuracy in hostile settings [18].

Future research should prioritize energy-aware recovery (e.g., duty-cycling nodes to cut power by 50%) [15], scalable topology-aware protocols (e.g., clustering nodes to reduce overhead by 30%) [12], and minimal-overhead security primitives (e.g., hash-based checks with <1ms latency) [9]. Bridging these gaps will enable resilient, real-time edge systems capable of meeting dependable computing's stringent demands.

Conclusion

This literature review has explored real-time fault tolerance in edge computing, highlighting its critical role in dependable and distributed systems. Fault detection techniques, such as LAAECHA and lightweight ML models, enable rapid identification of failures, yet struggle with resource constraints and dynamic variability. Recovery strategies like replication and task offloading enhance resilience but introduce trade-offs in latency, scalability, and network dependency, often misaligned with real-time demands. The interplay of timing constraints and dependability reveals persistent gaps, particularly in energy efficiency, scalability, and lightweight security against Byzantine threats. While current approaches advance edge reliability, they fall short in resource-scarce, volatile environments. Future research should prioritize adaptive, energy-aware algorithms, scalable topology-aware protocols, and minimal-overhead security primitives to bridge these gaps, ensuring robust, real-time fault tolerance tailored to edge computing's unique challenges within dependable systems.

STRIP Ranasinghe [16]:

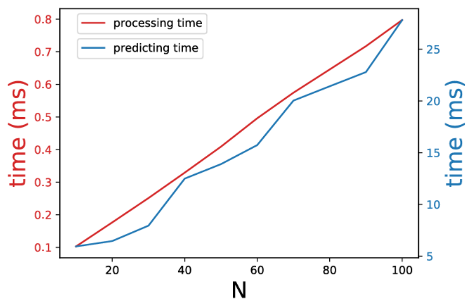
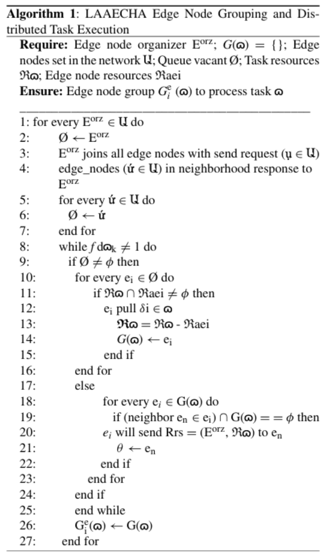


Figure 10: Detection time overhead vs N.

LAAECHA algorithm for edge computing [9]:

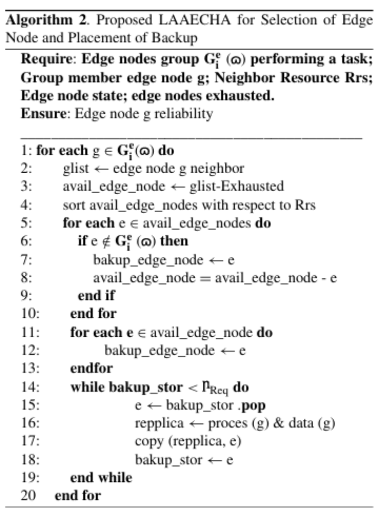


Algorithm 1: LAAECHA Edge Node Grouping and Distributed Task Execution

Require: Edge node organizer E^{orz}; $G(\varpi) = \{\}$; Edge nodes set in the network $\mathfrak{U}$; Queue vacant $\emptyset$; Task resources $\mathfrak{R}\varpi$; Edge node resources $\mathfrak{R}aei$

Ensure: Edge node group G_i^e (ϖ) to process task ϖ

```
for every $E^{orz} \in \mathfrak{U}$ do
        $\emptyset \leftarrow E^{orz}$
        $E^{orz}$ joins all edge nodes with send request ($\mathfrak{u} \in \mathfrak{U}$)
        edge_nodes ($\acute{\mathfrak{r}} \in \mathfrak{U}$) in neighborhood response to $E^{orz}$
        for every $\acute{\mathfrak{r}} \in \mathfrak{U}$ do
          $\emptyset \leftarrow \acute{\mathfrak{r}}$
        end for
        while $f\, d\varpi_k \neq 1$ do
          if $\emptyset \neq \phi$ then
            for every $e_i \in \emptyset$ do
                if $\mathfrak{R}\varpi \cap \mathfrak{R}aei \neq \phi$ then
                  $e_i$ pull $\delta i \in \varpi$
                    $\mathfrak{R}\varpi = \mathfrak{R}\varpi$ - $\mathfrak{R}aei$
                    $G(\varpi) \leftarrow e_i$
                  end if
            end for
            else
                  for every $e_i \in G(\varpi)$ do
                    if (neighbor $e_n \in e_i$) $\cap\, G(\varpi) == \phi$ then
                  $e_i$ will send Rrs $= (E^{orz}, \mathfrak{R}\varpi)$ to $e_n$
                    $\theta \leftarrow e_n$
                  end if
                end for
            end if
            end while
            $G_i^e(\varpi) \leftarrow G(\varpi)$
        end for
```

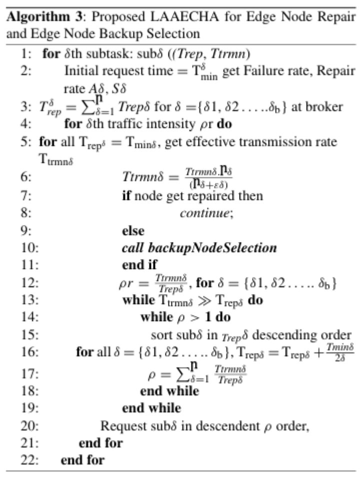


Algorithm 2. Proposed LAAECHA for Selection of Edge Node and Placement of Backup

Require: **Edge nodes group G_i^e (ϖ) performing a task; Group member edge node g; Neighbor Resource Rrs; Edge node state; edge nodes exhausted.**

Ensure: Edge node g reliability

```
for each g ∈ $G_i^e(\varpi)$ do
      glist ← edge node g neighbor
      avail_edge_node ← glist-Exhausted
      sort avail_edge_nodes with respect to Rrs
      for each e ∈ avail_edge_nodes do
        if e ∉ $G_i^e$ $(\varpi)$ then
          bakup_edge_node ← e
          avail_edge_node = avail_edge_node - e
        end if
      end for
      for each e ∈ avail_edge_node do
            bakup_edge_node ← e
      endfor
      while bakup_stor < $\mathfrak{n}_{Req}$ do
                e ← bakup_stor .pop
                repplica ← proces (g) & data (g)
                copy (repplica, e)
                bakup_stor ← e
      end while
end for
```



Algorithm 3: Proposed LAAECHA for Edge Node Repair and Edge Node Backup Selection

```
for $\delta$th subtask: sub$\delta$ (($Trep$, $Ttrmn$)
      Initial request time = $T_{min}^{\delta}$ get Failure rate, Repair rate $A\delta$, $S\delta$
$T_{rep}^{\delta} = \sum_{\delta=1}^{\mathfrak{n}} Trep\delta$ for $\delta$ ={$\delta 1, \delta 2 \ldots\ldots \delta_b$} at broker
      for $\delta$th traffic intensity $\rho r$ do
for all $T_{rep^{\delta}} = T_{min\delta}$, get effective transmission rate $T_{trmn\delta}$
                  $Ttrmn\delta = \frac{Ttrmn\delta.\mathfrak{n}_{\delta}}{(\mathfrak{n}_{\delta+\varepsilon\delta})}$
                  if node get repaired then
                        continue;
                  else
                  call backupNodeSelection
                  end if
                  $\rho r = \frac{Ttrmn\delta}{Trep\delta}$, for $\delta = \{\delta 1, \delta 2 \ldots\ldots \delta_b\}$
                  while $T_{trmn\delta} \gg T_{rep\delta}$ do
                      while $\rho > 1$ do
                        sort sub$\delta$ in $_{Trep}\delta$ descending order
          for all $\delta = \{\delta 1, \delta 2 \ldots\ldots \delta_b\}$, $T_{rep\delta} = T_{rep\delta} + \frac{Tmin\delta}{2\delta}$
                      $\rho = \sum_{\delta=1}^{\mathfrak{n}} \frac{Ttrmn\delta}{Trep\delta}$
                      end while
                  end while
              Request sub$\delta$ in descendent $\rho$ order,
        end for
    end for
```

Figure 11. Replication factor for different reliability levels against node failure. [9]:

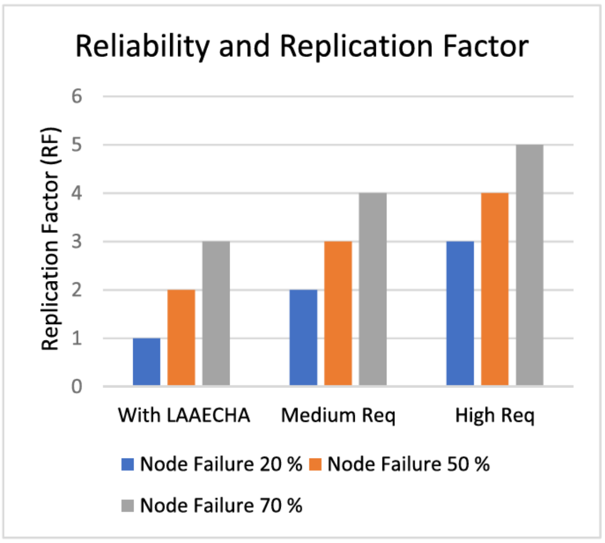
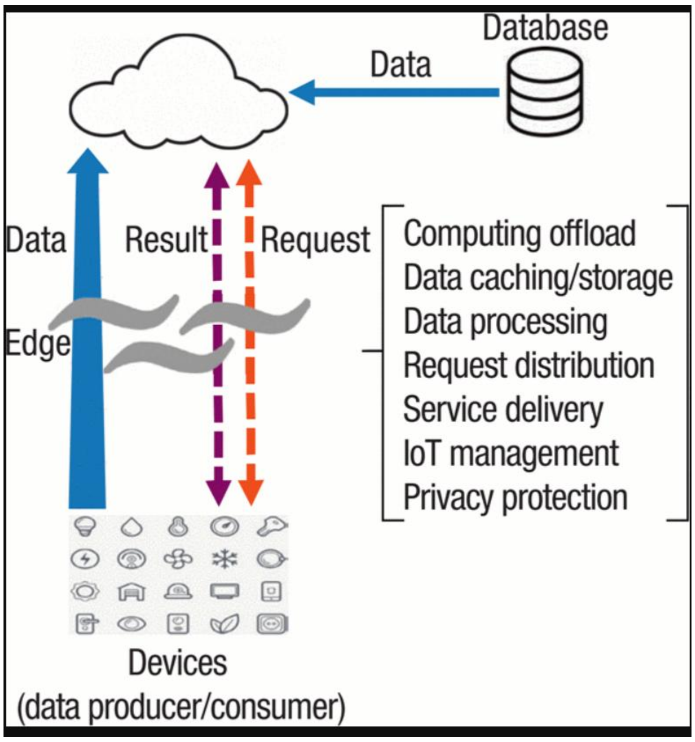


FIGURE 11. Replication factor for different reliability levels against node failure.

Edge computing architecture:



Citations

[1]: W. Shi and S. Dustdar, "The Promise of Edge Computing," in Computer, vol. 49, no. 5, pp. 78-81, May 2016, doi: 10.1109/MC.2016.145. The Promise of Edge Computing | IEEE Journals & Magazine | IEEE Xplore

[2]: Mahadev Satyanarayanan, Pieter Simoens, Yu Xiao, Padmanabhan Pillai, Zhuo Chen, Kiryong Ha, Wenlu Hu, and Brandon Amos. 2015. Edge Analytics in the Internet of Things. IEEE Pervasive Computing 14, 2 (Apr.-June 2015), 24–31. https://doi.org/10.1109/MPRV.2015.32. Edge Analytics in the Internet of Things | IEEE Pervasive Computing

[3]: M. Pourreza and P. Narasimhan, "Fault Tolerant Edge Computing: Challenges and Opportunities," 2023 IEEE 7th International Conference on Fog and Edge Computing (ICFEC), Bangalore, India, 2023, pp. 73-80, doi: 10.1109/ICFEC57925.2023.00018. Fault Tolerant Edge Computing: Challenges and Opportunities | IEEE Conference Publication | IEEE Xplore

[4]: Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo and J. Zhang, "Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing," in Proceedings of the IEEE, vol. 107, no. 8, pp. 1738-1762, Aug. 2019, doi: 10.1109/JPROC.2019.2918951. Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing | IEEE Journals & Magazine | IEEE Xplore

[5]: Flavio Bonomi, Rodolfo Milito, "Fog Computing: A Platform for Internet of Things and Analytics." *IEEE Network*, 29(2), 12–17, Aug. 2012, DOI: 10.1145/2342509.2342513. (PDF) Fog Computing and its Role in the Internet of Things

[6]: T. N. Gia, M. Jiang, A. -M. Rahmani, T. Westerlund, P. Liljeberg and H. Tenhunen, "Fog Computing in Healthcare Internet of Things: A Case Study on ECG Feature Extraction," 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, Liverpool, UK, 2015, pp. 356-363, doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.51. Fog Computing in Healthcare Internet of Things: A Case Study on ECG Feature Extraction | IEEE Conference Publication | IEEE Xplore

[7]: A. U. Rehman, R. L. Aguiar and J. P. Barraca, "Fault-Tolerance in the Scope of Cloud Computing," in IEEE Access, vol. 10, pp. 63422-63441, 2022, doi: 10.1109/ACCESS.2022.3182211. Fault-Tolerance in the Scope of Cloud Computing | IEEE Journals & Magazine | IEEE Xplore

[8]: B. Cao et al., "Edge–Cloud Resource Scheduling in Space–Air–Ground-Integrated Networks for Internet of Vehicles," in IEEE Internet of Things Journal, vol. 9, no. 8, pp. 5765-5772, 15 April15, 2022, doi: 10.1109/JIOT.2021.3065583. Edge–Cloud Resource Scheduling in Space–Air–Ground-Integrated Networks for Internet of Vehicles | IEEE Journals & Magazine | IEEE Xplore

[9]: M. Bukhsh, S. Abdullah and I. S. Bajwa, "A Decentralized Edge Computing Latency-Aware Task Management Method With High Availability for IoT Applications," in IEEE Access, vol. 9, pp. 138994-139008, 2021, doi: 10.1109/ACCESS.2021.3116717. A Decentralized Edge Computing Latency-Aware Task Management Method With High Availability for IoT Applications | IEEE Journals & Magazine | IEEE Xplore

[10]: N. Abbas, Y. Zhang, A. Taherkordi and T. Skeie, "Mobile Edge Computing: A Survey," in *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450-465, Feb. 2018, doi: 10.1109/JIOT.2017.2750180. Mobile Edge Computing: A Survey | IEEE Journals & Magazine | IEEE Xplore

[11]: M. Satyanarayanan, M. (2017). "The Emergence of Edge Computing." *Computer*, 50(1), 30–39, Jan. 2017, DOI: 10.1109/MC.2017.9. The Emergence of Edge Computing

[12]: C. Wang, C. Gill and C. Lu, "FRAME: Fault Tolerant and Real-Time Messaging for Edge Computing," 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 2019, pp. 976-985, doi: 10.1109/ICDCS.2019.00101. FRAME: Fault Tolerant and Real-Time Messaging for Edge Computing | IEEE Conference Publication | IEEE Xplore

[13]: Z. Ning et al., "Partial Computation Offloading and Adaptive Task Scheduling for 5G-Enabled Vehicular Networks," in IEEE Transactions on Mobile Computing, vol. 21, no. 4, pp. 1319-1333, 1 April 2022, doi: 10.1109/TMC.2020.3025116. Partial Computation Offloading and Adaptive Task Scheduling for 5G-Enabled Vehicular Networks | IEEE Journals & Magazine | IEEE Xplore

[14]: S. Tuli, G. Casale and N. R. Jennings, "PreGAN: Preemptive Migration Prediction Network for Proactive Fault-Tolerant Edge Computing," IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, London, United Kingdom, 2022, pp. 670-679, doi: 10.1109/INFOCOM48880.2022.9796778. PreGAN: Preemptive Migration Prediction Network for Proactive Fault-Tolerant Edge Computing | IEEE Conference Publication | IEEE Xplore

[15]: B. Shilpa, P. R. Kumar and R. Kumar Jha, "Spreading Factor Optimization for Interference Mitigation in Dense Indoor LoRa Networks," 2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET), London, United Kingdom, 2023, pp. 1-5, doi: 10.1109/GlobConET56651.2023.10149925. Spreading Factor Optimization for Interference Mitigation in Dense Indoor LoRa Networks | IEEE Conference Publication | IEEE Xplore

[16]: Damith C. Ranasinghe, Yansong Gao, Change Xu, Derui Wang, Shiping Chen, and Surya Nepal. 2019. “STRIP: a defence against trojan attacks on deep neural networks.” In Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC '19). Association for Computing Machinery, New York, NY, USA, 113–125. https://doi.org/10.1145/3359789.3359790. STRIP | Proceedings of the 35th Annual Computer Security Applications Conference

[17]: P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila and T. Taleb, "Survey on Multi-Access Edge Computing for Internet of Things Realization," in IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 2961-2991, Fourthquarter 2018, doi: 10.1109/COMST.2018.2849509. Survey on Multi-Access Edge Computing for Internet of Things Realization | IEEE Journals & Magazine | IEEE Xplore

[18]: Shahzaib Shaikh, Manar Jammal, “Survey of fault management techniques for edge-enabled distributed metaverse applications,” Computer Networks, Volume 254, 2024, 110803, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2024.110803. Survey of fault management techniques for edge-enabled distributed metaverse applications - ScienceDirect

[19]: M. Satyanarayanan, "The Emergence of Edge Computing," in Computer, vol. 50, no. 1, pp. 30-39, Jan. 2017, doi: 10.1109/MC.2017.9. The Emergence of Edge Computing | IEEE Journals & Magazine | IEEE Xplore