International Journal of Computer Engineering in Research Trends
ISSN: 2349-7084/ https://doi.org/10.22362/ijcert/2025/v12/i12/v12i1201
Volume 12, Issue 12, December 2025

Research Paper

Replication based Fault Tolerant Algorithm for Component Based Distributed System

[1*] Lalu Banothu, [2] M. Chandra Mohan, [3] C. Sunil Kumar

[1] Research Scholar, Dept. of Computer Science Engineering, JNTUH College of Engineering, Hyderabad, India*
Email ID: lalunb.csegnitc@gniindia.org
[2] Professor, Dept. of Computer Science Engineering, JNTUH College of Engineering, Hyderabad, India
[3] Professor & HOD in CSE, The Apollo University, Chittoor, Andhra Pradesh, India

**Corresponding Author(s): lalunb.csegnitc@gniindia.org*

Received: 05/10/2025 Revised: 21/10/2025 Accepted: 21/12/2025 Published: 31/12/2025

Abstract: Component based distributed systems are growing rapidly due to their interoperability and ability to reuse components in location transparent manner. When components owned by third parties are reused, it is indispensable to have reliable mechanisms for smooth functioning of the system. Different kinds of faults might occur when a distributed system is deployed and running. There is need for tolerating faults and ensure that the system continuously serves its clients with reliability. Replication of server components is one of the strategies for fault tolerance. However, it needs to be carried out with optimal care as it causes overhead. In this paper, we proposed a replication based fault tolerant algorithm for component based distributed system. The algorithm is named as Replication based Reliable and Fault Tolerant Algorithm (RRFTA). Our algorithm exploits a replication manager and fault detectors at local and global level. The distributed system case study considered for empirical study has number of server components running in different machines at server side. The proposed algorithm ensures utilization of replicas appropriately in order to achieve smooth functioning of the system. The experimental results revealed the efficiency of the proposed algorithm in terms of its replication strategy.

Keywords: Software engineering, component based distributed system, fault tolerance, reliability, and replication

---

1. Introduction

Distributed applications are made up of number of reusable components. These components may be in the local server or remote server. In other words, distributed application picks several components in geographically located servers. For this reason, it is essential to have strategies to tolerate faults. Faults can occur due to a variety of reasons. They include hardware failures, network failures, component failures and even operator mistakes. When there is a fault in the system, it is manifested in the form of an error and then the error leads to system failure. When system is failed, it stops rendering desired services to clients or users. There are many researchers focused on distributed component based systems and their reliability and availability.

Jasim et al. focused on the FTF-mHealth-IoT framework improves healthcare services for patients, using RLLT for risk assessment and AHP for hospital selection. Kaiwartya et al. proposed a fault-tolerant framework for optimizing virtualization in WSNs for IoT, demonstrating efficient results surpassing existing methods. Shi et al. managed faults in electric shipboard power systems is crucial for system survivability. Various fault management techniques are explored. Aguiar et al. emphasized fault-tolerance in cloud computing, addressing features, methodologies, and future research directions for improved reliability. Ye et al. introduced the HCEFT model for secure and efficient fault-tolerant edge storage, including the ECWSS optimization method. From the literature it was understood that fault tolerance can be implemented with component replication. However, there is need for a strategy for optimal replication to have balance between replication and reliability.

Key contributions in this study are as follows.

1. We proposed a replication based fault tolerant algorithm for component based distributed

system. The algorithm is named as Replication based Reliable and Fault Tolerant Algorithm (RRFTA). Our algorithm exploits a replication manager and fault detectors at local and global level.

2. The distributed system case study considered for empirical study has number of server components running in different machines at server side. The proposed algorithm ensures utilization of replicas appropriately in order to achieve smooth functioning of the system.

3. We built an application to realize the case study and the proposed algorithm and evaluated it.

The remainder of the paper is structured as follows. Section 2 reviews literature on existing methods of fault tolerance in component based distributed systems. Section 3 provides preliminaries required for understanding the proposed method. Section 4 presents our methodology for fault tolerance. Section 5 presents our experimental results. Section 6 concludes our work besides reflecting on future scope of the research.

2. Related Work

This section reviews literature on existing works on fault tolerance methods in distributed systems. Zheng et al. [1] Contemporary IoT applications rely on centralized messaging, prone to tampering. Trinity, a decentralized publish-subscribe broker with blockchain, ensures security and immutability. Girau et al. [2] physical substrates face more defects, fault-tolerant computing with neural models emerges, and mitigating failure impacts actively and passively. Madni et al. [3] Proposed DCLCA improves fault tolerance in cloud computing through dynamic clustering and intelligent scheduling, outperforming other techniques significantly. Jasim et al. [4] The FTF-mHealth-IoT framework improves healthcare services for patients, using RLLT for risk assessment and AHP for hospital selection. Madari et al. [5] expands real-time applications of fog computer like Smart Grids. RIAPS provides a resilient Discovery Manager for collaborative Smart Systems. Kaiwartya et al. [6] proposes a fault-tolerant framework for optimizing virtualization in WSNs for IoT, demonstrating efficient results surpassing existing methods. Kaur et al. [7] The PSO-UFC protocol solves energy imbalance and fault tolerance issues in wireless sensor networks, outperforming existing protocols. Yang et al. [8] Discussed fault diagnosis and fault-tolerant operation for MMCs, ensuring modularity, reliable switch open-circuit fault management seamlessly.

Zaidan et al. [9] reviews challenges in healthcare provision through fault-tolerant mHealth systems, emphasizing the need for comprehensive solutions. Liu et al. [10] CPS integrates computation, communication, and control technology, facing challenges despite potential for societal advancement and technological leadership. Dotta et al. [11] smart grid integrates information technology, enhancing power grid awareness. Genetic algorithms aid robust controller design despite communication losses. Guo et al. [12] enables flexible network management of SDN, but with network expansion, single controllers face processing demands. Multi-controller solutions are explored. Ni et al. [13] Smart grid facilitates efficient two-way communication, but privacy concerns arise. Proposed scheme ensures data aggregation, privacy, and fault tolerance. Riad et al. [14] face challenges including energy, connection issues, and faults. Paper proposes self-healing methodology to enhance network performance. Cao et al. [15] paradigm shift in IoT to Edge Mesh distributes decision-making tasks among devices, enabling distributed intelligence. Shi et al. [16] Managing faults in electric shipboard power systems is crucial for system survivability. Various fault management techniques are explored. García et al. [17] analytics of Big Data relies on effective processing algorithms. Map Reduce framework supports scalable, fault-tolerant data processing.

Netto et al. [18] Computer virtualization in data centers enables efficient resource management. DORADO protocol integrates coordination services into Kubernetes for efficient container management. Pachghare et al. [19] Block chain is a linked record system secured with cryptographic hash. Consensus algorithms ensure trust, but private block chain research is limited. Min et al. [20] Fog Computing aims to reduce data delivery latency. This platform's architecture and resource allocation for latency reduction are discussed. Zhao et al. [21] Protecting CNN inference against soft errors is critical. Proposed ABFT schemes efficiently detect and correct errors with minimal overhead. Singh et al. [22] Scientific workflows demand robust scheduling. Proposed fault-tolerant algorithm learns replication heuristics, enhancing efficiency and reliability in unstable environments. Aguiar et al. [23] emphasizes fault-tolerance in cloud computing, addressing features, methodologies, and future research directions for improved reliability. Liu et al. [24] proposes MobileRE, a hybrid fault tolerance approach combining erasure codes and replicas for dynamic mobile systems. Ye et al. [25] introduce the HCEFT model for secure and efficient fault-tolerant edge storage, including the ECWSS optimization method. Islam et al. [26] examines fault tolerance methods in cloud computing, categorizing them into Reactive, Proactive, and Resilient approaches. Machine learning aids

in resilience. Common methods include replication, checkpointing, and migration.

Xin et al. [27] presents PEFS, an advanced scheduling approach for cloud data centers, employing AI prediction models for task classification and energy reduction. Empirical evidence supports its effectiveness. Future efforts aim to expand simulations and enhance dependability. Mitra et al. [28] introduces a load balancing algorithm for heterogeneous processors in a dynamic network. It prioritizes local balancing, employing expander routing, demonstrating superior scalability and performance over other routing methods in various network scenarios, including failure-prone environments. Wang et al. [29] proposes an IoT data storage optimization algorithm, boosting efficiency and fault tolerance in cloud computing. Simulation validates its efficacy, urging further research. Youness et al. [30] addresses fault-tolerant scheduling in MPSoCs with task replication, presenting the weighted average make span metric. Two algorithms improve system reliability without performance compromises. Further research on optimizing parameters and hybrid backup is required. From the literature it was understood that fault tolerance can be implemented with component replication. However, there is need for a strategy for optimal replication to have balance between replication and reliability.

3. Preliminaries

A distributed system is made up of number of connected computers and components running in those computers. In such distributed applications faults can arise leading to failures due to many reasons such as component failure, hardware failure, congestion, overload and even bugs in software. Figure 1 shows how faults can cause errors leading to failure.

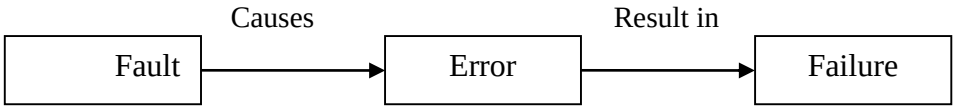


Fig.1. Failure and its cause

In distributed systems reliability plays crucial role. In such systems, therefore, it is important to incorporate fault avoidance and fault tolerance. The former reduces likelihood of failure with good design practices while the latter is to see the system works fine even in presence of a fault. Fault is nothing but a source that leads to an error. An error is nothing but the manifestation of fault in the system. Failure is the state of the system where it deviates in giving expected service.

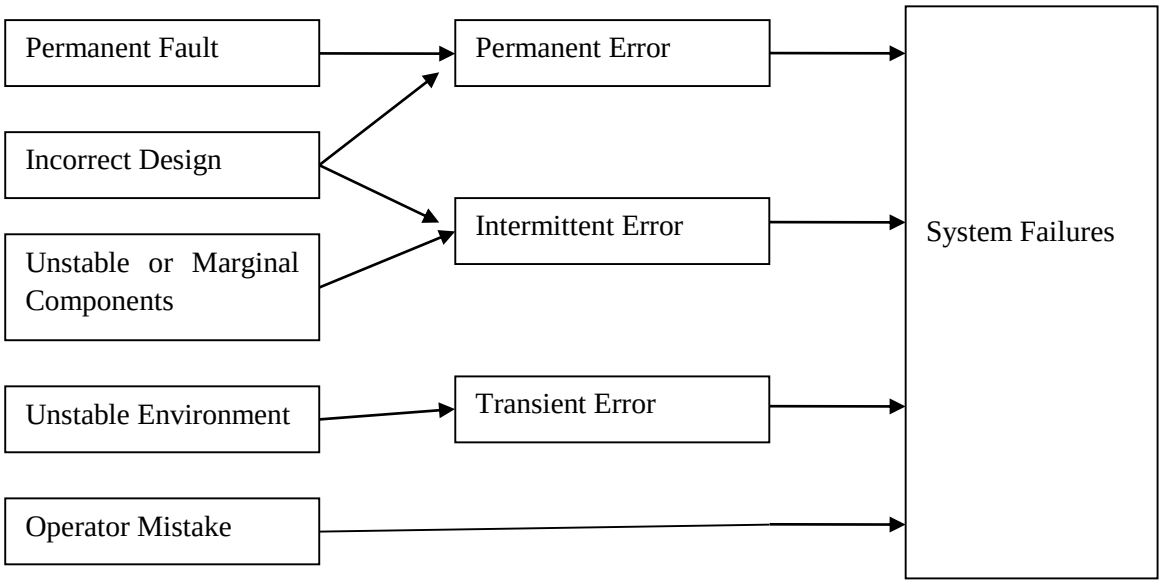


Fig.2. Different kinds of errors and corresponding sources leading to system failure

As presented in Figure 2, permanent error is the error that causes irreversible system failure which is originated from a permanent fault or incorrect system design. Intermittent error is the error that is reversible. It may occur due to incorrect design of the system or unstable hardware components being used. Transient error is the error which is temporary due to unstable environment. Sometimes, operator mistake may also lead to potential system failure. Failure rate is the measure of number of failures of the system in a given period. Fault coverage is

the capability of the system to tolerate faults as expressed in Eq. 1.

$$C = P(fault\ recovery|fault\ existence) \quad (1)$$

where probability is denoted as P and C denotes fault coverage. Reliability of the system on the other hand is conditional probability of the system this is surviving for a time interval [0, t] provided that the system was functioning fine at time t=0. Reliability of the system is thus measured as expressed in Eq. 2.

$$R(t) = \Pr\{0\ failures\ in\ [0.t]|no\ failure\ at\ t = 0\} \quad (2)$$

Considering the number of components $N_0(t)$ that are functioning a given time t and the $N_f(t)$ denoting number of failed components at t while N denotes total number of components in the system at t, the reliability can be expressed as in Eq. 3.

$$R(t) = \frac{N_0(t)}{N} = \frac{N_0(t)}{N_0(t)+N_f(t)} \quad (3)$$

In the same fashion, the unreliability of the system can be expressed as in Eq. 4.

$$Q(t) = \frac{N_f(t)}{N} = \frac{N_f(t)}{N_o(t)+N_f(t)} \quad (4)$$

From this, it can be understood that at runtime t, the expression R(t) = 1-Q(t). Assuming that the system is functioning and its failure rate is a constant denoted as λ, then the reliability can be expressed as its exponential function as in Eq. 5.

$$R(t) = e^{-\lambda t} \quad (5)$$

When the system is delivering expected services, it is said to be dependable. Fault tolerance of the system can improve its dependability. System availability also intuitively describes its reliability. It does mean that the system is operational when it is needed by clients or users. There are many metrics that help in understanding system availability. Mean time to failure, as expressed is in Eq. 6, is one measure that indicates the operational time of the system before first occurrence of failure.

$$MTTF = \frac{\sum_{i=1}^{N} t_i}{N} \quad (6)$$

where N denotes number of times the system is measured, t_i denotes the time period system is functioning well and the start time is denoted as t=0. Another measure, expressed in Eq. 7, named mean time to repair finds what the average time system takes for repair is.

$$MTTR = \frac{\sum_{i=1} t_i}{N} \quad (7)$$

This measure can be simplified repair rate per given time. Than the measure is expressed as in Eq. 8.

$$MTTR = \frac{1}{\mu} \quad (8)$$

Mean time between failures is the measure used to know the average time that is between two failures. The average number of failures is expressed in Eq. 9 while the MTBF is as expressed in Eq. 10.

$$n_{avg} = \sum_{i-1}^{N} \frac{n_i}{N} \quad (9)$$

$$MTBF = \frac{T}{n_{avg}} \quad (10)$$

Table 1. Notations used in the paper

Notation	**Meaning**
C	Denotes fault coverage
P	Represents fault probability
$N_0(t)$	At time t, it denotes number of components functioning well
$N_f(t)$	At time t, it denotes number of failed components
N	At time t, it denotes the total number of components
λ	Denotes failure rate
t_i	Time before occurrence of first fault
R	Denotes reliability of the system

As discussed above, there are number of metrics used for finding system reliability. When there is efficient fault tolerance method in place, system reliability increases. The notations used in this paper are presented in Table 1.

3.1 Need for Replication for Reliability

In the contemporary era, enterprises rely on applications that run in multiple server machines. They depend on distributed computing and may use number of software components. The distributed nature is meant for reliability in rendering services. However, the components have dependency among them. Therefore, the component failure leads to system failure. Towards solving this problem, many fault tolerant strategies are proposed as found in the literature. One of the strategies is to replicate components in different machines so as to ensure smooth functioning of the system. Component failure leads to failure of system or generation of incorrect results to the users. In distributed applications, failures can have different effects. Due to breakdown of communication links in networks also it is possible that components in the system fail in rendering services. Detection of kind of failure is also important. Many fault tolerant techniques use replicated components to achieve system reliability. It is a real life concept such as a flight having multiple engines for fault tolerance. This strategy is also used in computer hardware with duplicated parts in order to tolerate partial failures. When the distributed application relies on multiple server components, it is important to follow replication strategy. Such strategy ensures improved system performance with fault tolerance besides increasing system availability.

4. Proposed System

4.1 Problem Statement

Component-Based Distributed Systems (CBDS) increasingly rely on reusable, third-party server components deployed across geographically distributed machines. While this improves scalability and interoperability, it also significantly increases system vulnerability to faults, such as component failures, network issues, hardware breakdowns, and transient errors. A failure in even a single critical component can propagate and lead to partial or complete system failure, thereby degrading reliability, availability, and user trust.

Although component replication is a well-known fault tolerance technique, uncontrolled or naive replication introduces excessive overhead, increased resource consumption, and performance degradation. Existing approaches either focus on replication without considering optimal utilization or lack coordinated mechanisms to detect failures and manage replicas dynamically at both local and global levels.

4.2 System Model

This section presents the proposed methodology and the underlying algorithm for fault tolerance in distributed component based system. The proposed system is based on a case study application known as Distributed Reservation System (DRS). The system model associated with DRS is presented in Figure 1. The system is implemented with Java distributed component technology known as Remote Method Invocation (RMI). The system has many server components that are used by an application. The server components are remotely located and used by the client application. Since the components work in a distributed environment, the system is suitable candidate for the research of fault tolerance. It has provision for reservation of all travelling related services. The server side components include Car Server, Flight Server, Room Server and Middleware Server. These server components that run in different geographical locations are part of a client application. Therefore, it is indispensable that the application needs fault-tolerant approach in the usage of components.

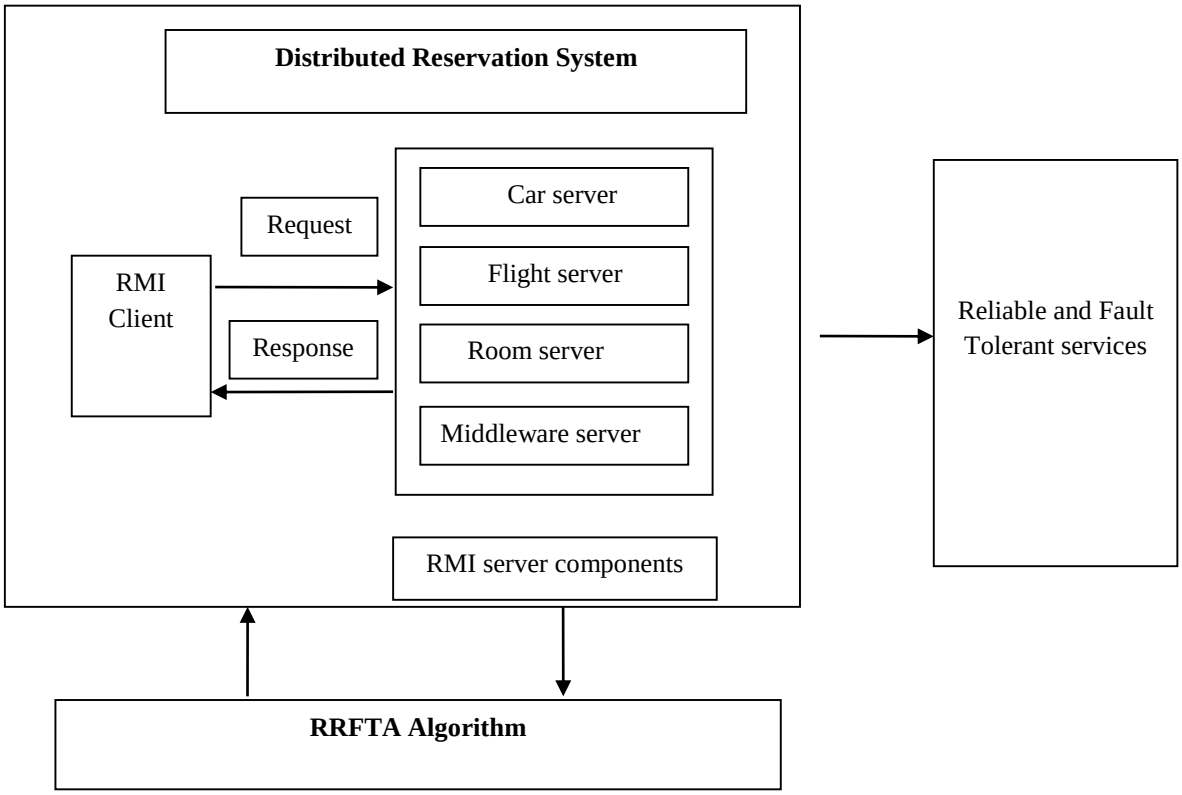
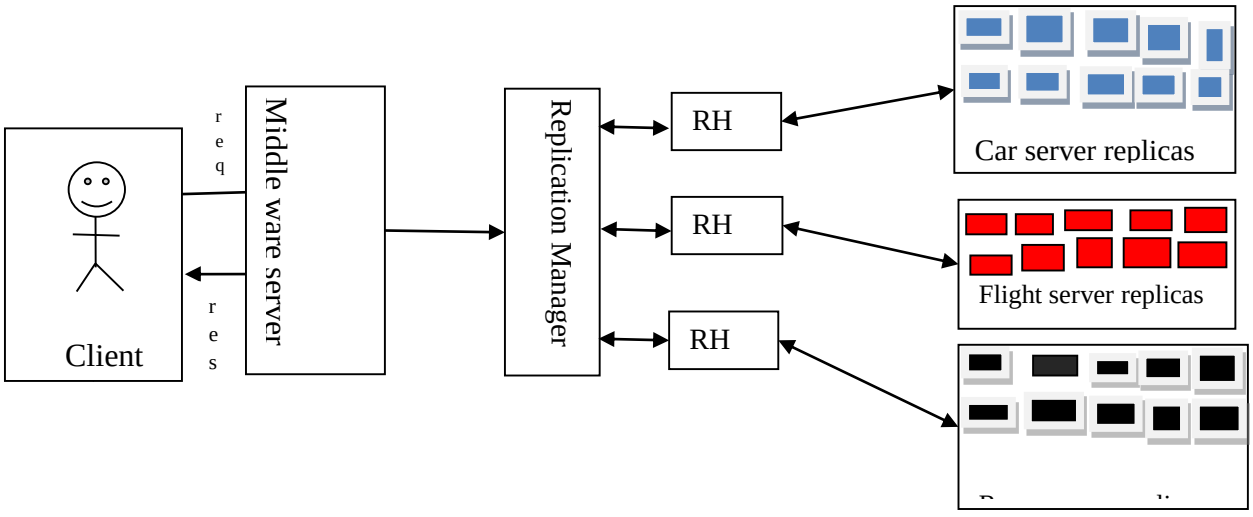


Fig.3.The system model

The system model is used in order to identify different functions and ensure that the proposed fault-tolerance algorithm ensures smooth functioning of the application. The functions from the server components are mapped to functions with measurement of frequency of invocations. It is made based on the function ranking approach.

4.2 Proposed Replication Strategy



4.3 Proposed Algorithm

This paper proposes a Replication-Based Reliable and Fault Tolerant Algorithm (RRFTA) to improve reliability and availability in component-based distributed systems. The algorithm aims to ensure uninterrupted service delivery in the presence of component or node failures while limiting the overhead caused by excessive replication.

RRFTA employs a Replication Manager supported by local and global fault detectors to monitor the operational

status of distributed server components. Local fault detectors identify potential failures through heartbeat monitoring, while the global fault detector validates failures to prevent false detection. The Replication Manager maintains the status of primary components and their replicas.

During request processing, client requests are dynamically bound to active components. The primary component is preferred when available; otherwise, requests are transparently redirected to a healthy replica. In case of invocation failure, RRFTA enables rapid failover by selecting an alternative replica without client involvement.

Furthermore, RRFTA adopts a controlled replication strategy based on component usage frequency and failure behavior, ensuring an effective trade-off between fault tolerance and resource utilization. The detailed steps of the proposed algorithm are presented in Algorithm 1.

Algorithm 1: RRFTA — Replication-based Reliable and Fault Tolerant Algorithm

Input:

Client request req, set of services S, replica pools RP, heartbeat interval Δ, timeout τ

Output:

Reliable response res

Steps:

1: Initialize primary components and their replicas for each $s \in s$.

2: Start local fault detectors on all nodes and a global fault detector at the Replication Manager (RM).

3: RM maintains a replica table RT with component status.

4: Monitoring Phase

5: Periodically collect heartbeats from all nodes.

6: If heartbeat timeout exceeds τ, mark node as failed and update RT.

7: Request Processing Phase

8: Client sends request req to Middleware Server.

9: Middleware forwards req to RM for component selection.

10: If primary component of requested service is alive, select it.

11: Else select an alive replica from RP.

12: If no replica is available, return failure.

13: Failover Phase

14: Invoke request on selected component.

15: If invocation fails, select another alive replica and retry.

16: Replication Control

17: Adjust number of replicas based on invocation frequency and observed failures.

18: Return response res to client.

5. Experimental Results

Table 2. System Failure Probability under Different Fault Tolerance Strategies

Component Failure Probability (%)	System Failure Probability		
	No FT Method	Random Method	RRFTA (Proposed)
1	0.1393	0.1262	0.0005
2	0.2592	0.2354	0.0020
3	0.3624	0.3320	0.0045
4	0.4512	0.4150	0.0081

5	0.5276	0.4913	0.0126
6	0.5934	0.5545	0.0183
7	0.6501	0.6149	0.0250
8	0.6988	0.6624	0.0326
9	0.7408	0.7025	0.0415
10	0.7769	0.7402	0.0513

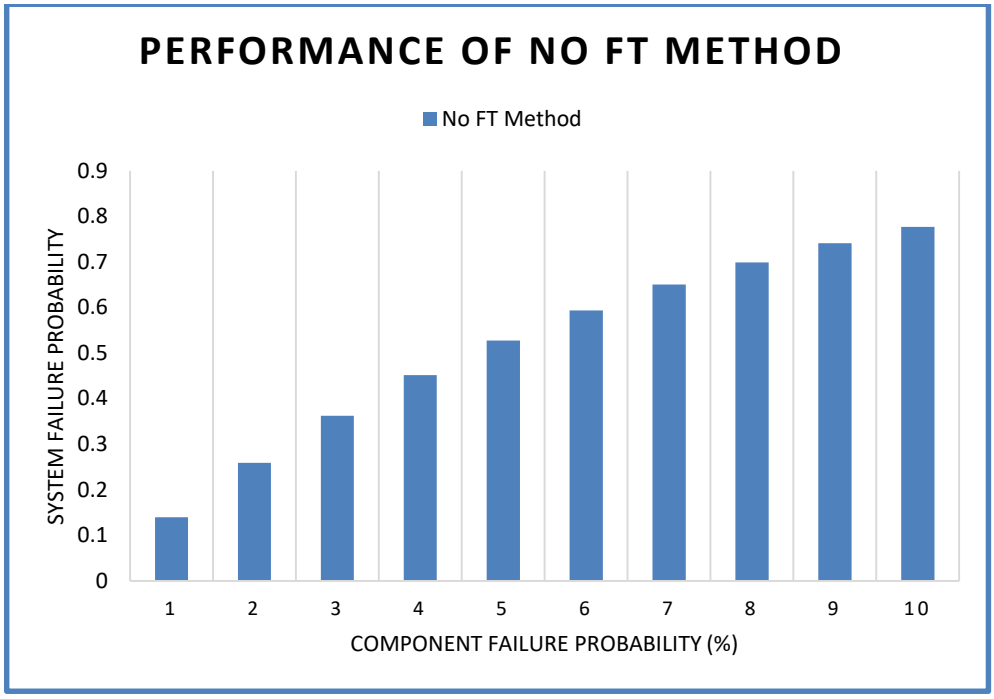


Fig.4. Performance of the No Fault Tolerance (No-FT) Method in terms of System Failure Probability.

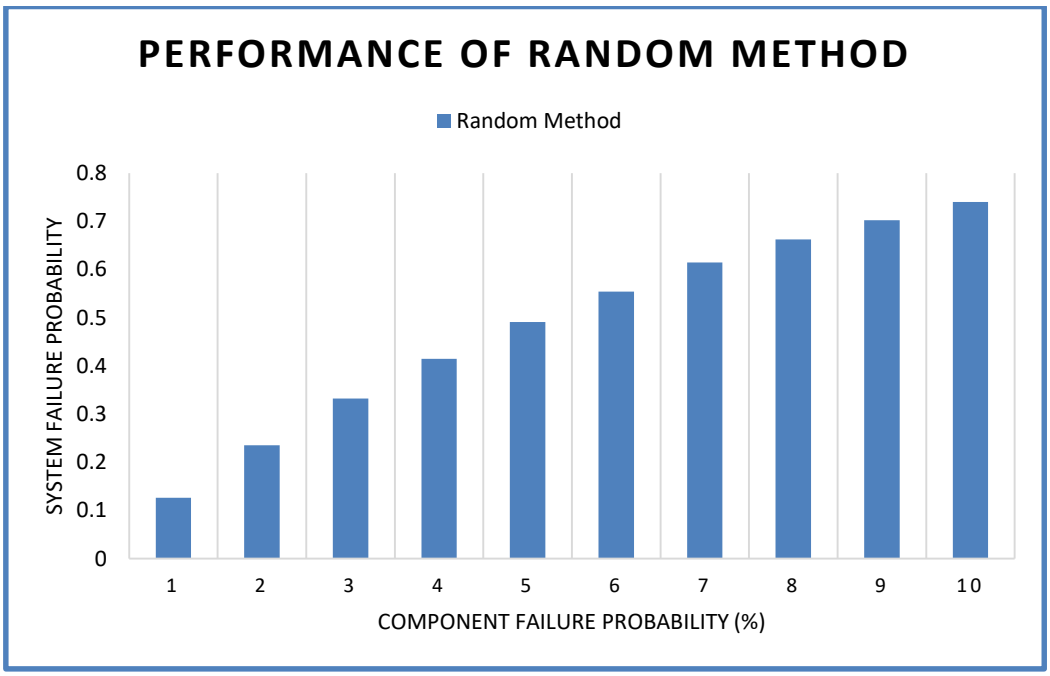


Fig. 5. Performance of the Random Replication Method in terms of System Failure Probability.

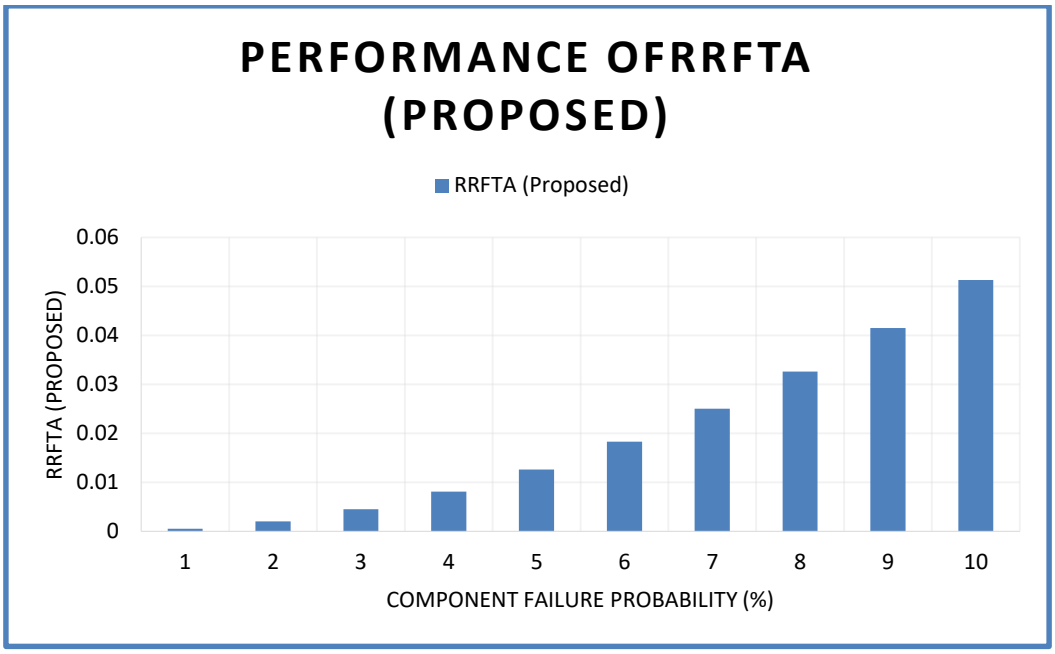


Fig. 6. Performance of the proposed RRFTA in terms of system failure probability.

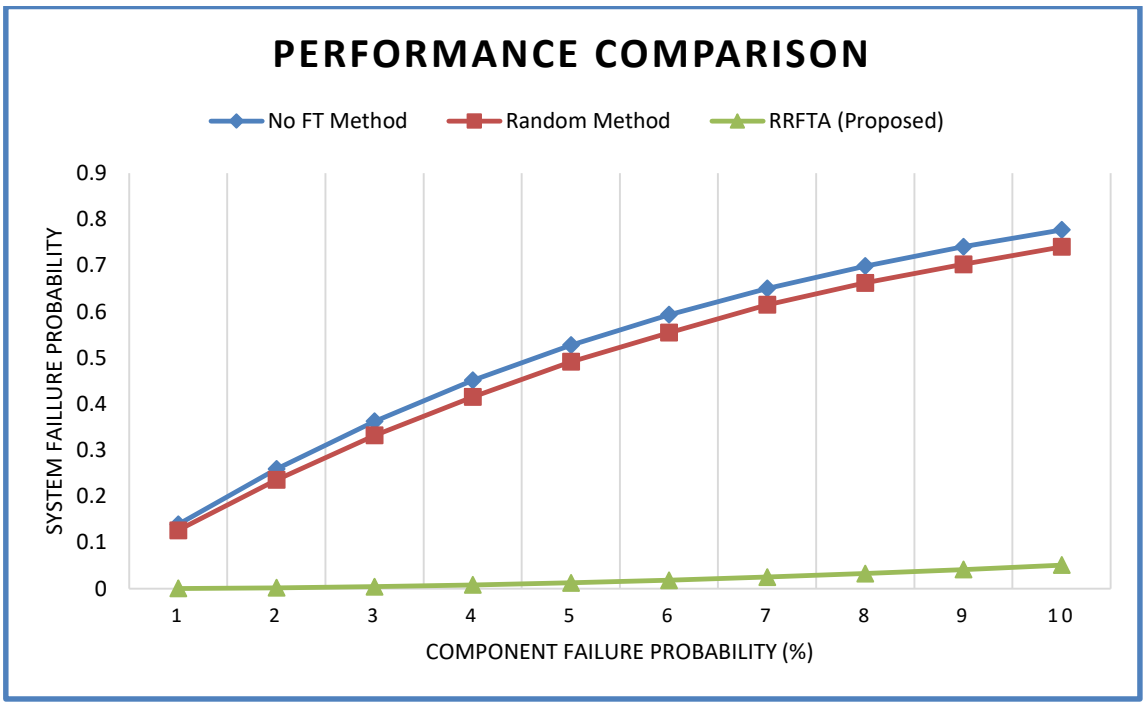


Fig. 7. Comparative analysis of system failure probability under different fault tolerance methods.

6. Conclusion And Future Work

In this study, we proposed a replication based fault tolerant algorithm for component based distributed system. The algorithm is named as Replication based Reliable and Fault Tolerant Algorithm (RRFTA). Our algorithm exploits a replication manager and fault detectors at local and global level. The distributed system case study considered for empirical study has number of server components running in different machines at server side. The proposed algorithm ensures utilization of replicas appropriately in order to achieve smooth functioning of the system. The experimental results revealed the efficiency of the proposed algorithm in terms of its replication strategy. In future we improve our methodology considering machine learning for dynamically changing the need for component replicas for improving reliability without causing unnecessary overhead.

Author Contributions

Lalu Banothu conceptualized the study, proposed the Replication-Based Reliable and Fault Tolerant Algorithm (RRFTA), designed the system model, implemented the replication strategy, and conducted the experimental evaluation and result analysis. M. Chandra Mohan contributed to the design and refinement of local and global fault detection mechanisms and supported reliability and overhead analysis. C. Sunil Kumar supervised the research, provided methodological guidance, validated the results, and critically reviewed and edited the manuscript. All authors read and approved the final manuscript.

Data availability: Data available upon request.

Conflict of Interest: There is no conflict of Interest.

Funding: The research received no external funding.

Similarity checked: Yes.

References

[1] Ramachandran, G. S., Wright, K.-L., Zheng, L., Navaney, P., Naveed, M., Krishnamachari, B., & Dhaliwal, J. (2019). Trinity: A Byzantine Fault-Tolerant Distributed Publish-Subscribe System with Immutable Blockchain-based Persistence. 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), p1-9.

[2] Torres-Huitzil, C., & Girau, B. (2017). Fault and Error Tolerance in Neural Networks: A Review. IEEE Access, 5, p17322–17341.

[3] Abdulhamid, S. M., Abd Latiff, M. S., Madni, S. H. H., & Abdullahi, M. (2016). Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm. Neural Computing and Applications, 29(1), p279–293.

[4] Albahri, O. S., Albahri, A. S., Zaidan, A. A., Zaidan, B. B., Alsalem, M. A., Mohsin, A. H., … Shareef, A. H. (2019). Fault-Tolerant mHealth Framework in the Context of IoT-Based Real-Time Wearable Health Data Sensors. IEEE Access, 7, p50052–50080.

[5] Eisele, S., Mardari, I., Dubey, A., & Karsai, G. (2017). RIAPS: Resilient Information Architecture Platform for Decentralized Smart Systems. 2017 IEEE 20th International Symposium on Real-Time Distributed Computing (ISORC), p125-132.

[6] Kaiwartya, O., Abdullah, A. H., Cao, Y., Lloret, J., Kumar, S., Shah, R. R., … Prakash, S. (2018). Virtualization in Wireless Sensor Networks: Fault Tolerant Embedding for Internet of Things. IEEE Internet of Things Journal, 5(2), p571–580.

[7] Kaur, T., & Kumar, D. (2018). Particle Swarm Optimization-Based Unequal and Fault Tolerant Clustering Protocol for Wireless Sensor Networks. IEEE Sensors Journal, 18(11), p4614–4622.

[8] Yang, S., Tang, Y., & Wang, P. (2018). Seamless Fault-Tolerant Operation of a Modular Multilevel Converter With Switch Open-Circuit Fault Diagnosis in a Distributed Control Architecture. IEEE Transactions on Power Electronics, 33(8), p7058–7070.

[9] Albahri, A. S., Zaidan, A. A., Albahri, O. S., Zaidan, B. B., & Alsalem, M. A. (2018). Real-Time Fault-Tolerant mHealth System: Comprehensive Review of Healthcare Services, Opens Issues, Challenges and Methodological Aspects. Journal of Medical Systems, p1-56.

[10] Yang Liu, Yu Peng, Bailing Wang, Sirui Yao, and Zihe Liu. (2017). Review on Cyber-physical Systems. IEEE. 4 (1), p27-40.

[11] Bento, M. E. C., Dotta, D., Kuiava, R., & Ramos, R. A. (2018). A Procedure to Design Fault-Tolerant Wide-Area Damping Controllers. IEEE Access, 6, p23383–23405.

[12] Hu, T., Guo, Z., Yi, P., Baker, T., & Lan, J. (2018). Multi-controller Based Software-Defined Networking: A Survey. IEEE Access, 6, p15980–15996.

[13] Ni, J., Zhang, K., Alharbi, K., Lin, X., Zhang, N., & Shen, X. S. (2017). Differentially Private Smart Metering With Fault Tolerance and Range-Based Filtering. IEEE Transactions on Smart Grid, 8(5), p2483–2493.

[14] Walaa Elsayed, Mohamed Elhoseny, A.M. Riad, and Aboul Ella Hassanien. (2017). Autonomic Self-healing Approach to Eliminate Hardware Faults in Wireless Sensor Networks. Springer, p1-10.

[15] Sahni, Y., Cao, J., Zhang, S., & Yang, L. (2017). Edge Mesh: A New Paradigm to Enable Distributed Intelligence in Internet of Things. IEEE Access, 5, p16441–16458.

[16] Babaei, M., Shi, J., & Abdelwahed, S. (2018). A Survey on Fault Detection, Isolation, and Reconfiguration Methods in Electric Ship Power Systems. IEEE Access, 6, p9430–9441.

[17] Ramírez-Gallego, S., Fernández, A., García, S., Chen, M., & Herrera, F. (2018). Big Data: Tutorial and guidelines on information and process fusion for analytics algorithms with MapReduce. Information Fusion, 42, p51–61.

[18] Netto, H. V., Lung, L. C., Correia, M., Luiz, A. F., & Sá de Souza, L. M. (2017). State machine replication in containers managed by Kubernetes. Journal of Systems Architecture, 73, p53–59.

[19] Pahlajani, S., Kshirsagar, A., & Pachghare, V. (2019). Survey on Private Blockchain Consensus Algorithms. 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT), p1-6.

[20] Liu, Y., Fieldsend, J. E., & Min, G. (2017). A Framework of Fog Computing: Architecture, Challenges, and Optimization. IEEE Access, 5, p25445–25454.

[21] Kai Zhao; Sheng Di; Sihuan Li; Xin Liang; Yujia Zhai; Jieyang Chen; Kaiming Ouyang; Franck Cappello and Zizhong Chen; (2021). Algorithm-Based Fault Tolerance for Convolutional Neural Networks . IEEE Transactions on Parallel and Distributed Systems. http://doi:10.1109/tpds.2020.3043449

[22] Nirmala, S. Jaya; Setlur, Amrith Rajagopal; Singh, Har Simrat and Khoriya, Sudhanshu (2020). An efficient fault tolerant workflow scheduling approach using replication heuristics and checkpointing in the cloud. Journal of Parallel and Distributed Computing, 136, 14–28. http://doi:10.1016/j.jpdc.2019.09.004

[23] A. U. REHMAN, RUI L. AGUIAR and JOÃO PAULO BARRACA. (2022). Fault-Tolerance in the Scope of Cloud Computing. IEEE. 10, pp.63422-63441. http://doi:10.1109/ACCESS.2022.3182211

[24] Yu Wu; Duo Liu; Xianzhang Chen; Jinting Ren; Renping Liu; Yujuan Tan and Ziling Zhang; (2021). MobileRE: A replicas prioritized hybrid fault tolerance strategy for mobile distributed system . Journal of Systems Architecture. http://doi:10.1016/j.sysarc.2021.102217

[25] JUNQI CHEN, YONG WANG, MIAO YE AND QIUXIANG JIANG. (2023). A Secure Cloud-Edge Collaborative Fault-Tolerant Storage Scheme and Its Data Writing Optimization. IEEE. 11, pp.66506-66521. http://doi:10.1109/ACCESS.2023.3291452

[26] Muhammad Asim Shahid; Noman Islam; Muhammad Mansoor Alam; M.S. Mazliham and Shahrulniza Musa; (2021). Towards Resilient Method: An exhaustive survey of fault tolerance methods in the cloud computing environment . Computer Science Review. http://doi:10.1016/j.cosrev.2021.100398

[27] Marahatta, Avinab; Xin, Qin; Chi, Ce; Zhang, Fa and Liu, Zhiyong (2020). PEFS: AI-driven Prediction based Energy-aware Fault-tolerant Scheduling Scheme for Cloud Data Center. IEEE Transactions on Sustainable Computing, 1–1. http://doi:10.1109/TSUSC.2020.3015559

[28] Chatterjee, Moumita; Mitra, Anirban; Setua, Sanjit Kumar and Roy, Sudipta (2020). Gossip-based fault-tolerant load balancing algorithm with low communication overhead. Computers & Electrical Engineering, 81, 106517–. http://doi:10.1016/j.compeleceng.2019.106517

[29] Wang, Mingzhe and Zhang, Qiuliang (2020). Optimized data storage algorithm of IoT based on cloud computing in distributed system. Computer Communications, 157, 124–131. http://doi:10.1016/j.comcom.2020.04.023

[30] Hassan Youness; Aly Omar and Mohamed Moness; (2021). An Optimized Weighted Average Makespan in Fault-Tolerant Heterogeneous MPSoCs . IEEE Transactions on Parallel and Distributed Systems. http://doi:10.1109/tpds.2021.3053150.