

Fault Tolerance and Self-Healing in Iot-Enhanced P2p Systems

Ibrokhimkhuja Rikhsikhujaev

Student at Amity University in Tashkent

Article information:

Manuscript received: 4 Sep 2025; **Accepted:** 10 Oct 2025; **Published:** 11 Nov 2025

Abstract: In this article different self-healing and fault-tolerance concepts and approaches that can be integrated in IoT enhanced P2P systems were studied. Overall, the main purpose of this article is to give an overview on importance of self-healing and fault tolerance in order to ensure seamless operation and reliable data transfer. In addition, there are some recommendations provided for improving resilience and reliability.

Keywords: Fault tolerance, Self-healing, Internet of Things (IoT), Peer-to-peer networks, Distributed systems, Failure detection, Replication, Checkpointing and recovery, Adaptive fault management, Redundancy.

Introduction.

Various definitions of distributed P2P (Peer-to-Peer) networks have been provided in different literatures, according to [1] a distributed P2P network architecture is an architecture in which participants' portions of hardware resources are shared (storage, computing power, bandwidth of a network connection and etc.).

In addition, participants of distributed systems have the same goal to be achieved. Each node has its own part of the task and after completing it, nodes resubmit the outcomes to submission node. According to there are two types of distributed systems homogeneous (Cluster) and heterogeneous (Grid, Cloud and P2P) [2]. They are used to solve certain types of tasks such as Quality of Service (QoS), Load Balancing and Fault Tolerance.

Fault tolerance is the ability of a system to continue functioning properly in case if a failure or malfunction of one or more of its components. Fault tolerance consists of two parts; faults detection and recovery. Furthermore, as it was mentioned in distributed systems it is challenging to keep systems running after failures occur. There are many techniques used for fault tolerance such as Replication, Checkpointing and Logging etc.

As the research scope fault tolerance and self-healing in IoT enhanced P2P systems were chosen, which means that the scope of the studies are distributed P2P networks that connect IoT devices.

In the context of IoT-enhanced P2P systems, where a network of independent computers presents itself as a unified entity to users, fault tolerance takes on added significance. The distributed nature of these systems elevates concerns about hardware or software failures, necessitating specialized techniques for detection and recovery.

Resilience in such environments is achieved through the application of fault tolerance techniques, including retry mechanisms, replication, check pointing, message logging, etc. These mechanisms collectively contribute to maintaining seamless operation, even in the face of unexpected component disconnects or malfunctions.

The paper delves into foundational concepts of fault tolerance specific to IoT-enhanced P2P systems and explores self-healing mechanisms within this dynamic context. A comprehensive analysis of related work is presented, offering insights into the current state of research and development. The paper concludes, contributing to the ongoing discourse on bolstering the reliability of distributed systems amid the IoT and P2P connectivity era.

Preview of literature.

In the main attributes of dependability were listed [3]. Therefore, there are four main characteristics included in the Figure 1: Availability, Reliability, Safety and Maintainability.

Figure 1. Characteristics of Dependability



Fault tolerance techniques are of two main types: Proactive and reactive.

Reactive fault tolerance.

Reactive fault tolerance techniques aim to minimize the impact of system failures once they have occurred. Examples of this method include Retry, Replication, Check-pointing, and Message Logging.

1. Retry involves multiple attempts to recover from failures,
2. Replication creates task replicas on different hosts to ensure continued execution,
3. Check-pointing regularly saves the system state for recovery after a crash, and
4. Message Logging reduces the need for costly check-pointing by replaying messages in a defined order for consistent system states.

Proactive fault tolerance.

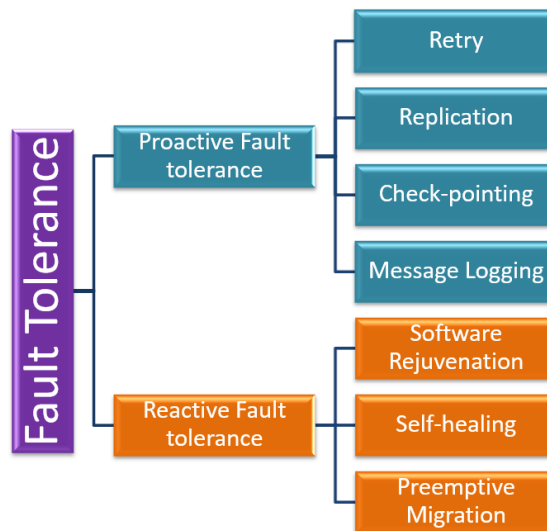
Proactive fault tolerance anticipates faults and replaces faulty components preemptively.

Techniques include:

1. Software Rejuvenation, where planned changes occur with each system reboot
2. Self-healing automates failure recovery for applications running on multiple virtual machines
3. Preemptive Migration constantly observes and analyzes applications for potential issues.

There is a diagram explaining fault tolerance and its types and techniques in Figure 2.

Figure 2. Types of Fault tolerance and Techniques



Methodology.

Make a Survey of Existing Literature: Conduct a survey about existing literature. Find and explore key studies, research, and recent developments. **Identify Fundamental Concepts and Theories:** Study main concepts, hypotheses, and methodologies. Get familiar with the evolution of ideas over time and their contributions to better understanding of the subject. **Find Research Gaps:** Find gaps in the current field of the term paper. Examine for further research and opportunities for implementation in projects (minor, major). **Analytically Evaluate Methodologies:** Consider methodologies from previous studies. Differentiate between the strengths and limitations of diverse approaches for various usage scenarios. **Investigate Trends and Innovations:** Get information about emerging trends, innovations, and cutting-edge developments. **Compare Findings:** Make a comparative table of findings from various studies.

Analysis and findings.

I have considered as a base paper and I extend this research and focus only on P2P distributed systems, which can be used in the field of IoT. P2PBNM [4] (Peer-to-Peer-based Network Management) system incorporates various techniques to implement fault tolerance and self-healing capabilities. Here are the techniques used in the system:

1. Peer Grouping:

Peers are grouped according to the administrative services they provide. This grouping helps in the replication of management components and improves the availability of management services. Each peer group is responsible for specific management tasks.

2. Replication of Management Components:

The system uses replication of management components within peer groups. This replication enhances fault tolerance by ensuring that even if some peers fail, there are redundant components available in other peers to continue providing the required management services.

3. Proactive Peer Organization:

The organization of peers into groups based on the services they expose is a proactive technique. It is designed to prepare the system for potential failures by ensuring that there

are redundant peers capable of providing the same service.

4. Monitoring Service:

The system employs a monitoring service that periodically verifies the state of managed devices/systems. This monitoring service detects anomalies and faults by comparing the observed state with predefined criteria for normal and anomalous states.

5. Healing Service:

A healing service is implemented to address faults detected by the monitoring service. The healing service initiates a predefined healing workplan to take corrective actions and restore the system to a healthy state.

6. Self-Healing Components:

The self-healing facility is implemented as a management component, which is responsible for both monitoring and healing services. These components are dynamically loaded and communicate with the grouping service to synchronize with other peers providing the same services.

7. Service Replication for Healing:

To enhance the dependability of the recovery service, the recovery work plan is duplicated across a set of peers. This replication ensures that multiple peers are capable of executing the healing workplan, enhancing responsiveness and scalability.

8. Dynamic Binding of Self-Healing Services:

The system administrator can dynamically bind the service with automatic recovery capabilities to a managed element at runtime. This involves issuing requests to the healing and monitoring groups, obtaining service identifiers, and binding them together.

9. Token-Signalized Round-Robin Execution:

Monitoring peers overseeing the monitoring plan form a structured ring. The monitoring plan is executed at regular intervals in a round-robin manner signaled by a token, ensuring systematic and efficient monitoring.

10. Peer Communication and Coordination:

Peers communicate and coordinate with each other, particularly within peer groups. Communication is essential for the replication of management components, synchronization of self-healing components, and coordination of monitoring and healing activities.

These techniques collectively contribute to the fault tolerance and self-healing capabilities of the P2PBNM system, ensuring the reliable and continuous operation of network management services. [5] mentions the use of “Checkpointing-and-Recovery” as a recovery technique in the context of addressing challenges related to the reliability of task execution in a P2P-Grid environment. Checkpointing-and-Recovery involves saving the state of a computation at a particular point in time (checkpointing) and recovering tasks from the last checkpoint in case of node failures.

It discusses techniques and mechanisms used in a P2P-Grid environment to address challenges related to the reliability of task execution. Here are some key techniques and components mentioned in the text:

1. Failure Detection Mechanisms:

The text emphasizes the importance of effective failure detection mechanisms in P2P-Grids. It discusses the use of a probe approach, where nodes probe their neighbors with “can you

hear me?” messages and wait for acknowledging messages. This approach helps detect node failures or departures, reducing the unavailability interval and facilitating faster recovery.

2. Checkpointing-and-Recovery:

The text introduces the concept of Checkpointing-and-Recovery as a fault-tolerance mechanism. Checkpointing involves saving the state of a computation at a particular point in time, and recovery allows tasks to resume execution from the last checkpoint in case of node failures. Checkpointing nodes are introduced to handle unreliability caused by nodes constantly joining or leaving the P2P-Grid system.

3. Redundancy in Checkpointing:

In order to enhance reliability against failures in nodes, redundancy is introduced into the system. The simplest form of redundancy discussed is storing checkpointing details in various storage nodes. Checkpointing data is stored in idle nodes within the P2P-Grid system, and multiple checkpointing peers can share checkpoint data from different working peers.

4. P2P-Grid System Model:

The text describes a P2P-Grid system model that includes four major components: Dispatcher, Working Peer (WP), Working Candidate Peer (WCP), and Checkpointing Peer (CPP). The dispatcher handles administrative duties, WPs execute tasks, WCPs handle checkpointing, and CPPs store checkpoint data. The system is designed to distribute tasks efficiently among working peers.

5. Peer Roles and Selection Criteria:

The roles of peers in the system include dispatcher, working peer, working candidate peer, and checkpointing peer. The selection criteria for these roles are based on factors such as processing speed, peer lifetime, capability to carry out tasks, and checkpointing capability of transmitting data. Longer peer lifetimes and high performance are preferred for certain roles.

6. Adaptive Failure Detection:

The passage explores the idea of adaptive failure detection, which strives to adjust to evolving network conditions. It introduces the average failure detection time (AFDT) as a measure for assessing the effectiveness of failure detection mechanisms. The objective is to rapidly identify instances of node failures or departures and find alternative replacements with the help of checkpoints.

7. Related Work Comparison:

The P2P-Grid compares the proposed approach with related works in fault tolerance for P2P-Grids, such as Dome, Condor, Gridbus, and XCAT3 framework. It emphasizes variations in emphasis, mechanisms, and objectives, underscoring that the suggested work is directed towards enhancing fault tolerance concerning repeated node participation and other failures within a dynamic and diverse P2P-Grid system.

Overall, the text outlines a comprehensive approach to addressing reliability challenges in P2P-Grids through a combination of failure detection mechanisms and checkpointing-and-recovery techniques, considering the dynamic nature of the environment.

The paper delves into the intricacies of structured Peer-to-Peer (P2P) networks, emphasizing the challenges associated with maintaining ring topology amidst churn and failures [6]. It critiques existing solutions such as Chord, citing their struggles with temporary inconsistencies and the complexity of synchronized join/leave algorithms involving multiple peers. In response, the paper proposes a novel approach, advocating for the management of

ring topology through a self-organizing system driven by feedback loops, aiming to streamline operations and enhance resilience. The Relaxed-Ring Design, a central focus of the paper, introduces several innovative concepts:

- **Join Algorithm Optimization:** The join algorithm is restructured into two steps, each requiring the agreement of only two peers. This strategic division mitigates the need for expensive synchronization, presenting a more efficient alternative to traditional approaches.
- **Branching for Resilience:** Unlike traditional structures like Chord, the Relaxed-Ring Design permits branches off the core ring. Peers are only required to connect to their successors, eliminating the need for connections to predecessors. This adaptation enhances resilience, particularly in scenarios involving connectivity issues.
- **Feedback Loop Implementation:** Drawing inspiration from control systems, the design incorporates feedback loops from monitors to actuators, establishing a mechanism to autonomously maintain ring structure. Join and failure events are modeled as perturbations, contributing to the self-organizing nature of the system.
- **Failure Recovery Strategies:** The paper Relaxed-Ring outlines a robust failure recovery mechanism triggered by the predecessor via the successor list. This mechanism unifies leave events, addresses false suspicions, and ensures the propagation of successor list changes. Additional safeguards are provided through the utilization of predecessor lists and fingers as backup resources.
- **The Analysis section of the paper further solidifies the proposed approach:**
- **Consistency Guarantees:** The authors rigorously prove lookup consistency guarantees, establishing the reliability of the Relaxed-Ring Design. Additionally, the paper argues that the introduction of branches does not permit responsibility range overlaps, adding a layer of formal assurance.
- **Handling Failure Scenarios:** The analysis evaluates the system's performance in various failure scenarios, including concurrent crashes and joins. It sheds light on the intricate interplay between join and failure loops, offering insights into the system's robustness.
- **Addressing Difficult Cases:** The paper extends its scrutiny to challenging scenarios such as branch failures and network partitions. Specific limitations are identified and formalized, contributing to a comprehensive understanding of the system's capabilities.

In the Comparison to Related Work section, the authors position the Relaxed-Ring Design against existing solutions like Chord and DKS. The argument centers on the design's superior resilience and compatibility with realistic failure detectors.

The Conclusion succinctly synthesizes the paper's contributions:

- **Self-Management through Feedback Loops:** The paper effectively demonstrates that ring topology can be autonomously managed using feedback loops, underscoring the self-organizing capabilities of the proposed design.
- **Robustness during Churn:** The simplified join process and unified leave mechanisms make the Relaxed-Ring Design resilient to failures during churn, addressing a critical aspect of P2P network management.
- **Towards Reliable Applications:** The provision of formal guarantees and the identification of limitations contribute valuable insights for building reliable applications in the context of structured P2P networks.

To enhance the precision of the analysis across various research endeavors addressing fault

tolerance on diverse platforms, the creation of a comprehensive comparative table (refer to Table 1) is suggested summarizing key findings and distinctions.

Table 1. Different fault detection and tolerance techniques used in p2p distributed systems

Article	Platform	Techniques implemented	Faults found	Proactive / Reactive or Combined	Support Large Scale	Contribute to Load Balancing	Availability	Reliability	Recovery techniques implemented	Overall remark
P2PBNM [4]	P2P based network management systems (P2PBNM)	Peer Grouping, Monitoring and Healing Services, Token-Signalized Round-Robin Execution, Peer Communication and Coordination	Connectivity Issues, Hardware Failures, Software/Application Failures, Performance Degradation, Security Threats, Configuration Errors, Resource Exhaustion, Service Availability, Environmental Conditions	Combined	>	>	>	>	Replication of management components, Replication of healing workplan,	The experiments have shown that there is a trade-off among total traffic and the mean duration for the self-recovery procedure
P2P-Grid [5]	P2P-Grid systems	Peer Roles, Adaptive Failure Detection, Blocking check pointing	Node Failures and Departures	Proactive	,	,	,	,	Checkpointing, Checkpointing Nodes, Working Candidate Peers (WCP)	4 roles for peers were introduced. In addition, there was used blocking check pointing.
Relaxed-Ring [6]	Implemented with Mozart-Oz programming system	Feedback loops for self-management	Node failures, network failures, broken links	Combined	>	,	>	>	Maintaining backup routing information, redirecting requests, triggering stabilized join process	Introduced "relaxed ring" topology, which is self-organizing and fault-tolerant. In comparison with a strict ring topology, relaxed ring handles failures and temporary network issues better.

Conclusion.

In conclusion, the investigation into "Fault Tolerance and Self-healing in IoT enhanced P2P systems" through the analysis of three pertinent articles has provided valuable insights into the evolving landscape of resilient peer-to-peer networks. The comparative table I constructed has served as a systematic tool to distill the nuances, strengths, and limitations inherent in each approach.

Across the examined articles, common themes emerged, highlighting the critical role of fault tolerance and self-healing mechanisms in mitigating the challenges posed by the dynamic nature of IoT-enhanced P2P systems. Each research work contributed distinct perspectives, shedding light on diverse strategies for ensuring system robustness and continuity.

Through this exploration, the significance of adaptive and intelligent fault detection mechanisms is recognized, coupled with efficient recovery strategies, in sustaining the integrity and performance of IoT-enhanced P2P networks.

In the broader context of P2P systems, this exploration into fault tolerance and self-healing mechanisms stands as a testament to the ongoing efforts to fortify the foundations of our interconnected world. As technology advances and IoT integration becomes more pervasive, the lessons drawn from this study will undoubtedly inform the development of robust, fault-tolerant, and self-healing P2P systems, ensuring the seamless functioning of the interconnected devices that define our modern digital landscape.

References.

1. R. Schollmeier, "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications," Proceedings First International Conference on Peer-to-Peer Computing, Linköping, Sweden, 2001, pp. 101-102, doi: 10.1109/P2P.2001.990434. Available: <https://ieeexplore.ieee.org/document/990434>
2. A. Ledmi, H. Bendjenna and S. M. Hemam, "Fault Tolerance in Distributed Systems: A Survey," 2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS), Tebessa, Algeria, 2018, pp. 1-5, doi: 10.1109/PAIS.2018.8598484. Available: <https://ieeexplore.ieee.org/document/8598484>
3. Tanenbaum, A.S. and M. Van Steen, Distributed systems: principles and paradigms. 2007: Prentice-Hall. Available: https://vowi.fsinf.at/images/b/bc/TU_Wien-Verteilte_Systeme_VO_%28G%C3%B6schka%29_-_Tannenbaum-distributed_systems_principles_and_paradigms_2nd_edition.pdf
4. P. A. P. R. Duarte, J. C. Nobre, L. Z. Granville and L. M. R. Tarouco, "A P2P-Based self-healing service for network maintenance," 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops, Dublin, Ireland, 2011, pp. 313-320, doi: 10.1109/INM.2011.5990706. Available: <https://ieeexplore.ieee.org/document/5990706>
5. W. Huan and N. Hidenori, "Fault Tolerance in P2P-Grid Environments," 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, Shanghai, China, 2012, pp. 2482-2485, doi: 10.1109/IPDPSW.2012.308. Available: <https://ieeexplore.ieee.org/abstract/document/6270874>
6. B. Mejias and P. Van Roy, "A Relaxed-Ring for Self-Organising and Fault-Tolerant Peer-to-Peer Networks," XXVI International Conference of the Chilean Society of Computer Science (SCCC'07), Iquique, Chile, 2007, pp. 13-22, doi: 10.1109/SCCC.2007.15. Available: <https://ieeexplore.ieee.org/document/4396973>