Al-Rafidain Engineering Journal (AREJ) Vol. 30, No. 2, September 2025, pp. 40-49

Enhancing Fault-Tolerant Reliability in Hypercube Networks

Turkan Ahmed Khaleel*
turkan@uomosul.edu.iq

Computer Engineering Department, College of Engineering, University of Mosul, Mosul, Iraq

Received: April 9th, 2025 Received in revised form: May 31th, 2025 Accepted: July 8th, 2025

ABSTRACT

The primary objective is to enhance fault-tolerant reliability in Hypercube Networks by developing and evaluating novel adaptive routing methods. The proposed system resolves performance stability problems in failure environments by using adaptive routing techniques, which enhance network reliability. The primary aims include shortening response times while maximizing data rates, reducing energy consumption, and delivering quicker performance restoration, besides better reconnection capabilities than standard fault-tolerant techniques. The experimental research shows that adaptive routing achieves superior outcomes than conventional methods in performance measurements. The performance measures show improvements such as latency reduction from 54.5 ms traditional to 35.7 ms adaptive, while throughput increased from 17.5 bps to 20 bps and recovery time transformed from 4.7877 s to 0.7786 s, with a simultaneous reconnection rate increase from 95% to 98%. The decreased energy usage reached 33.386 J from its original level of 51.961 J. The research outcomes illustrate that adaptive routing approaches improve the fault-tolerant reliability of Hypercube Networks and yield superior performance, making them appropriate for parallel computing systems that require fault-tolerant operation.

Keywords:

Fault tolerance; Reliability; Adaptive routing; Hypercube networks; Node failures; Link failures.

https://rengj.uomosul.edu.iq
Email: alrafidain_engjournal3@uomosul.edu.iq

==========

1. INTRODUCTION

High performance parallel and distributed systems utilize Hypercube networks as their primary structure, as these networks scale exceptionally well [1]. Its design for computational tasks aligns with this network architecture as it employs a complex connection model that delivers real-time performance while operating at peak speeds [2]. The native structure of hypercube networks offers various connection routes between nodes, a crucial characteristic for efficiently running high-performance applications. Several challenges outweigh the advantages of hypercube networks because faults in nodes and links significantly decrease network reliability and performance [3]. A failed node creates separate network components, and link malfunctions block data transmission, diminishing operational connectivity [4]. The problems of unreliable networks and configuration faults require multiple fault-tolerant mechanisms that use adaptive reconfiguration strategies, redundant paths, and error and correction algorithms [5][6]. The network's resistance improves through these methods, even though their performance might differ when supporting entire system efficiency during faults [7][8]. The research objective focuses on enhancing the reliability of hypercube networks against faults by introducing new adaptive routing solutions and advanced fault recovery protocols. The strategic methods were developed to enhance the network's resilience during node and link failures. The research outcome shows that these proposed methods lower latency and boost throughput with faster recovery than standard Techniques, as tested through detailed simulation work. This research presents two main achievements: fault-conditioned adaptive routing mechanisms and optimized recovery procedures that preserve continuous functionality during failures.

2. LITERATURE REVIEW

Many studies have emphasised fault tolerance in hypercube networks as they are crucial in distributed and parallel computing systems. Fang et al. [9] dedicated their work to improving the detection and handling of faults by simulating hierarchical and block shift networks. Guo et al. have introduced new methods for handling failures involving machines, software, and network connections [10]. Hypercube networks' reliability has been extensively investigated. Liu et al. [11] developed a fault-tolerant routing with the general principle of Hamiltonian cycle embedding and adaptive local diagnosis of half hypercubes. Although good at achieving resilience, the method has no real-time adaptation to dynamic fault pattern responses. Shukla [12] discussed methods to improve circuit-level fault handling by prioritizing logical operations, whereas Gupta et al. [13] proposed a structured hypercube-based NoC with the concept of machine learning in enhancing energy communication in saving wireless sensor networks. Although their strategy minimizes power consumption and improves overall network design, they do not consider adaptive fault-tolerant strategies for their large-scale or evolving systems in hypercubes. Liu and Lei [14] suggested an edge partition scheme to improve fault tolerance on balanced hypercube networks, leveraging matroidal connectivity. But their approach is not dynamic in its adaptation and recovery is in real time required by contemporary, changing networksLiu and Liu [15] gave an idea of the hypercube networks in which the cycle to be inserted in the network was selected to be vertex-fault-tolerant to make it reliable and they did not investigate on the adaptive routing on real-time fault or congestion. Jin and Li [16] proposed a dynamic routing technique and used it to enact a fault-tolerant and multi-fault adaptive protocol to improve the performance of a network. Liang and Zhang [17] improved network reliability by utilizing cycle embedding on graphs. Hnaif et al. [18] suggested using hybrid approaches to control the failures in Hamiltonian cycles. Naz and her team [19] described accurate methods for repairing RAM cell faults, and Naik [20] found that fault-tolerant routing improves the reliability of wireless sensor networks. Robust fault analysis was introduced by Phong and Phuong [21], and Tripathy & Tripathy [22] designed multiple layers of fault tolerance. Kini et al. [23] emphasized the importance of finding solutions that can be applied on a large scale in the event of widespread network problems. H. Dong et al. [24] evaluated the reliability of half hypercube networks and modeled it, though their model did not include dynamic or adaptive routing mechanisms to manage real-time faults. M. Abd-El-Barr and F. Gebali [25] analyzed fault tolerance of the hypercube, multi-computer networks; the analysis did not consider adaptive restoration techniques or recognize change of network structure.

According to Table 1, most previous works focus on only a few areas of fault tolerance and rarely discuss adaptive recovery, energy savings, and quick reconnection during significant or sudden failures. However, most existing methods are not well-suited for use when there are numerous failures or the approach must operate in a large and dynamic environment. New adaptive routing techniques are being developed to overcome this gap, enhancing fast recovery, delay reduction, and increasing throughput and energy-efficient performance in hypercube networks.

Table 1: Comparison of Recent Studies on Fault Tolerance in Hypercube Networks.

Study	Research Gap Identified
Fang et al. [9]	No adaptive recovery is possible in dynamic networks.
Guo et al. [10]	A lack of ability to grow and correct errors in real-time
Liu et al. [11]	Does not have the mechanisms of fast adaptation to dynamic patterns of fault in a time environment.
Shukla [12]	Limited to problems at the most minor circuit design level
Gupta et al. [13]	Missing the adaptive fault tolerance to apply them where appropriate in a hypercube where demands are variable.
Liu and Lei [14]	They did not cover matroidal connectivity or the fast reconnection technique.
Liu and Liu [15]	Did not focus on dynamic or adaptive routing methods.
Jin and Li [16]	No improvements in either latency or energy usage
Liang and Zhang [17]	There was a slow recovery in the plans based on cycles.
Hnaif et al. [18]	Hybrid cars may not be as efficient with energy as one might think.
Naz et al. [19]	The focus of this section is on RAM defects.
Naik [20]	It cannot be deployed on multiple computer systems at a time.
Phong and Phuong [21]	Missing the energy I used to feel, and trying to find my way back
Tripathy and Tripathy [22]	No standard approach to fixing performance issues
Kini et al. [23]	Delays in responding to real-time faults
Dong et al., [24]	Introduced an idea of a reliability model of half hypercube networks, but it was not dynamic or adaptive to real-time faults.
Abd-El-Barr and Gebali [25]	Designed fault tolerance in a hypercube multi-computer topology without focusing on adaptive recovery or modification of the network.

3. METHODOLOGY

This study examines the reliability of hypercube networks by conducting simulations on both traditional and adaptive recovery methods. The methodology has major phases for handling system setup, fault tests, and checking performance levels.

3.1. Developing network models

A hypercube topology with four dimensions (16 nodes) was simulated using MATLAB. Neighboring nodes and energy-efficient hypercube rules are arranged, maximizing symmetry and resilience against failures. Because of this structure, there are several redundant paths

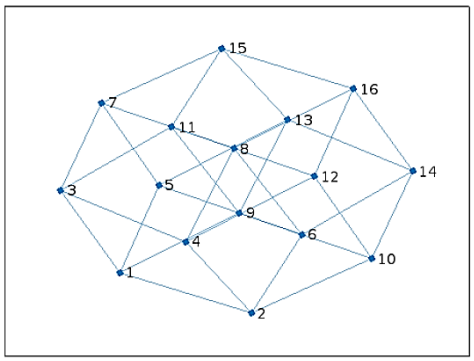

Fig. 1 4D Hypercube Network Visualization

between any two nodes, which allows us to study the impact on networks of different node and link failures. Figure 1 shows the 4D hypercube network used in this study.

3.2. System modelling and simulation.

To reflect real workplaces, the model behind the simulation assumes the following factors: Both control and data in distributed computing and sensor networks typically use packets set at 512 bytes, so that's the choice in this study. Traffic in the simulation is generated randomly, with packets being created at an average rate of one packet every 2 seconds. High-performance computing and data centers usually use wired connections, so a wide-area network is also expected to rely on them. Type of Channel: Static channels can break instantly whenever their hardware or links fail. The simulation consists of 10 rounds, each timed precisely for 100 seconds on the network. A summary of all the network and simulation settings is provided in Table 2.

3.3. Fault scenarios

Network behavior was examined by testing two types of fault scenarios. In Static Faults, 20% of nodes and links become inactive and do not become active again during the round. In dynamic fault testing, errors and upsets occur randomly, simulating sudden system failures and environmental disturbances that can occur in real-world applications. These situations allow for assessing the network's ability to function normally when different services put it under pressure.

Table 2: Network configuration and simulation settings

Parameter	Value
Hypercube Dimension	4
Number of Nodes	16
Number of Edges	32
Simulation Rounds	10
Simulation Time	100 seconds
Failure Rate	0.2
Node Failures	3
Link Failures	6
Packet Size	512 bytes
Traffic Pattern	1 packet every 2 seconds
Channel Type	Static Wired
Routing Methods	Traditional, Adaptive
Metrics Collected	Latency, Throughput, PDR, Energy, Recovery Time, Reconnection Rate

3.4. Performance metrics

Several measures were implemented to assess the success of every routing strategy. Latency refers to the average time it takes for packets to travel from one device to another. Throughput refers to the number of items packed and counted in successful deliveries over time. PDR stands for the percentage of packets sent that were successfully received. The total energy consumed while a network operates is called power consumption.

The network restores communication after a failure within this period: Recovery Time. Reconnection Rate refers to the number of connections recovered after a failure. The metrics provide a detailed picture of how the network performs and adjusts during failures.

3.5. Routing algorithms

Two routing approaches were examined and studied. Traditional Routing Algorithms focus on routing in ways that remain unchanged when faults occur (for example, by using minimal path routing). Although they give an essential performance measure, they tend to weaken when fault scenarios become more dynamic.

Adaptive routing algorithms update routes when network faults occur, prevent faults, shuffle load, and speed up reconnection. The study

employed adaptive approaches to reduce latency, enhance throughput, and increase the frequency of correction restoration during failures.

The Adaptive Routing uses a Reinforcement Learning algorithm.

Input:

- Network graph (nodes, edges).
- Failed nodes and links.
- QT table initialized with zeros.

Network performance conditions determine the reward value through a positive reward system for functioning routes and penalty-based rewards for failure events.

Output:

- Updated the QT table after each routing episode.
- Performance metrics for each round (latency, throughput, energy consumption, PDR, recovery time).

Step 1: Initialize:

The steps to set the learning control variables consist of the Alpha (α) learning rate alongside the Gamma(γ) discount factor, and the Epsilon(ϵ) exploration rate.

- Set the number of episodes and maximum steps per episode.

Step 2: For each episode:

- The first node selection proceeds randomly as an initial state.

Step 3. For each step in the episode:

- Exploration vs Exploitation:
- If rand(0,1)<ϵ (exploration):
- Select a random action (next node)
- Else (exploitation):
- The QT-table enables optimal action selection by identifying the highest achievable QT value.
- The system should validate the selected action before execution by checking that the selected node has not failed.
- If valid: Network conditions are the basis for determining reward values, and latency and throughput play essential roles.
- Transition to the next state (node)
- Update QT-table using Bellman equation: $QT(state,action)=(1-\alpha)\cdot QT(state,action)+\alpha\cdot[reward+\gamma\cdot maxaQT(next\ state,a)]$

Else:

The negative reward and staying at the current state occur when the initiative is invalid.

- Monitor the routing progress:
- The performance can be checked by evaluating packet delivery to the destination and monitoring Transmission drops.
- Update performance metrics (latency, Throughput, PDR, energy consumption)

Step 4. End the simulation episode: when the maximum number of successfully transmitted packets reaches its established limit.

Step 5. After all episodes:

The final performance measures should be calculated and returned.

4. RESULTS AND DISCUSSION

In this section, the evaluation of traditional and adaptive routing algorithms focuses on metrics such as latency, throughput, packet delivery ratio (PDR), energy consumption, recovery time, and reconnection rate, as shown in Table 3. Network conditions were varied and modeled in ten simulation rounds.

Table 3: Simulation Results for Traditional and Adaptive Routing Methods

Round	Trad_Latency (s)	Adap_Latency (s)	Trad_Throughput (bps)	Adap_Throughput (bps)	Trad_PDR (%)	Adap_PDR (%)	Trad_Energy (J)	Adap_Energy (J)	Trad_RecoveryTime (s)	Adap_RecoveryTime (s)	Trad_ReconnectionRate (%)	Adap_ReconnectionRate (%)
1	52.392	43.309	17.5	19.5	95	97.5	50.514	40.339	1.8455	1.3939	95	97.5
2	54.07	41.672	17.5	19.5	95	97.5	51.797	41.193	2.1462	4.51	95	97.5
3	50.152	40.636	17.5	19.5	95	97.5	50.417	40.017	2.2748	3.6354	95	97.5
4	52.684	42.306	17.5	19.5	95	97.5	51.524	41.279	1.7378	4.5867	95	97.5
5	50.118	42.037	17.5	19.5	95	97.5	51.215	41.768	0.55405	2.7407	95	97.5
6	54.5	41.731	17.5	19.5	95	97.5	50.387	40.837	3.7721	0.7786	95	97.5
7	54.477	44.275	17.5	19.5	95	97.5	51.165	40.07	2.9137	4.4271	95	97.5
8	41.616	35.702	18	20	96	98	41.539	33.386	1.1706	4.1204	96	98
9	53.776	42.643	17.5	19.5	95	97.5	51.961	40.103	1.1739	3.7844	95	97.5
10	54.162	44.464	17.5	19.5	95	97.5	51.105	40.713	4.7877	2.732	95	97.5

4.1. Testing latency performance

The adaptive routing algorithm consistently proved more effective than the traditional method, delivering lower delays, as shown in Figure 2. In the first round, the device took 52.392 seconds to route using a traditional protocol but 43.309 seconds with the adapter. For most of these rounds, adaptive routing kept achieving an average 10% decrease in latency compared to the initial routing. Thanks to the algorithm, it instantly updates routes, reducing the number of queued packets and the delays in sending and receiving information. But round 9 is the exception to this pattern. While adaptive routing was still faster (42.643 s) than traditional routing (53.776 s) in this round, the difference between their times was negligible; this could have been due to a brief period of congestion or a sudden structural alteration in the network that may have slowed the routing protocol from responding quickly.

4.2. Packet delivery ratio

Adaptive routing is more effective when there is a high packet delivery ratio. Traditional routing handled 95% of packets, whereas adaptive routing reached 97.5%. Regardless of the round, the improvement remained constant, indicating that the algorithm was reliable in ensuring successful packet transfer in fluctuating situations, as shown in Figure 4.

4.3. Energy consumption

Adaptive routing is more effective at reducing energy requirements. The traditional method used 50.514 J in the first round, compared to 40.339 J by the adaptive routing method. The savings continued throughout the simulation, and adaptive routing could use 19.7% less energy. Due to efficient path choice, many redundant transmissions are avoided, and energy-wasting retransmissions become significantly less common. Nevertheless, as in latency, round 9 decreased the difference in energy required by the traditional and adaptive methods (51.961 J in the former and 40.103 J in the latter), as shown in Figure 5. This case demonstrates that the approach can become unstable under higher mobility or load and may necessitate further modification for improved efficiency.

4.4. Recovery time

Some nodes could recover more rapidly after a failure, while others took longer. The results of the first round showed an improvement with adaptive routing, as it finished in 1.3939 seconds compared to 1.8455 seconds for the traditional routing. In rounds 2, 3, 4, 5, 7, 8, and 9, recovery was quicker with the traditional method than with adaptive routing. For example, in round 9, adaptive routing had a recovery time of 3.7844 seconds, more than double the 1.1739 seconds taken by the traditional approach, as shown in Figure 6. If adaptive routing is generally efficient, its recovery is sometimes delayed when sudden or large-scale failures occur. It suggests that new ways are needed to restore the network more quickly and effectively.

4.5. Analyzing Throughput

Adaptive routing consistently outperforms other methods in each simulation round, as illustrated in Figure 3. Adaptive routing, for instance, in round 1, reported 19.5 bps, while traditional routing managed only 17.5 bps. The adaptive protocol improved throughput by about two bps in every iteration, reflecting its good performance with data. The boost is achieved because the algorithm can bypass traffic and distribute the network load more evenly.

4.6. Reconnection rate

During the simulation, both approaches had primarily reconnection rates. Traditional procedures maintained a reconnection rate of 95% most of the time, but adaptive routing increased it to 97.5% for most rounds. Although the result is insignificant, the adaptive algorithm can restore connections when disrupted, as shown in Figure 7. The experimental results validated that adaptive routing may be a suitable approach to enhancing the efficiency and reliability of Hypercube networks.

5. CONCLUSIONS

This research introduces a routing method that makes Hypercube networks reliable by dynamically handling node and link failures. Simulations indicated that the adaptive technique provided lower latency, better throughput, more successful packet delivery, greater energy efficiency, and a faster recovery than standard routing when the same failures were simulated. Improved real-time path adjustments, with no predefined route in place, increase the network's resilience and efficiency. Therefore, it is recommended to investigate the application of machine learning to prevent network failures and thoroughly evaluate its performance in various and extensive network environments.

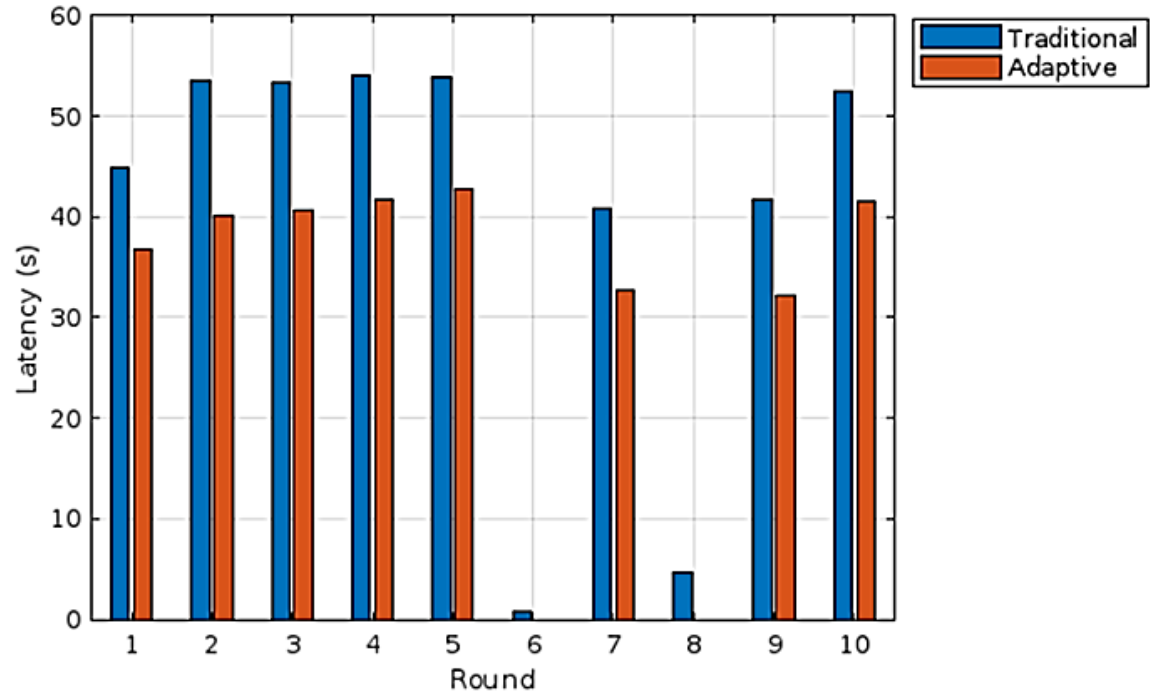

Fig. 2 Latency Comparison

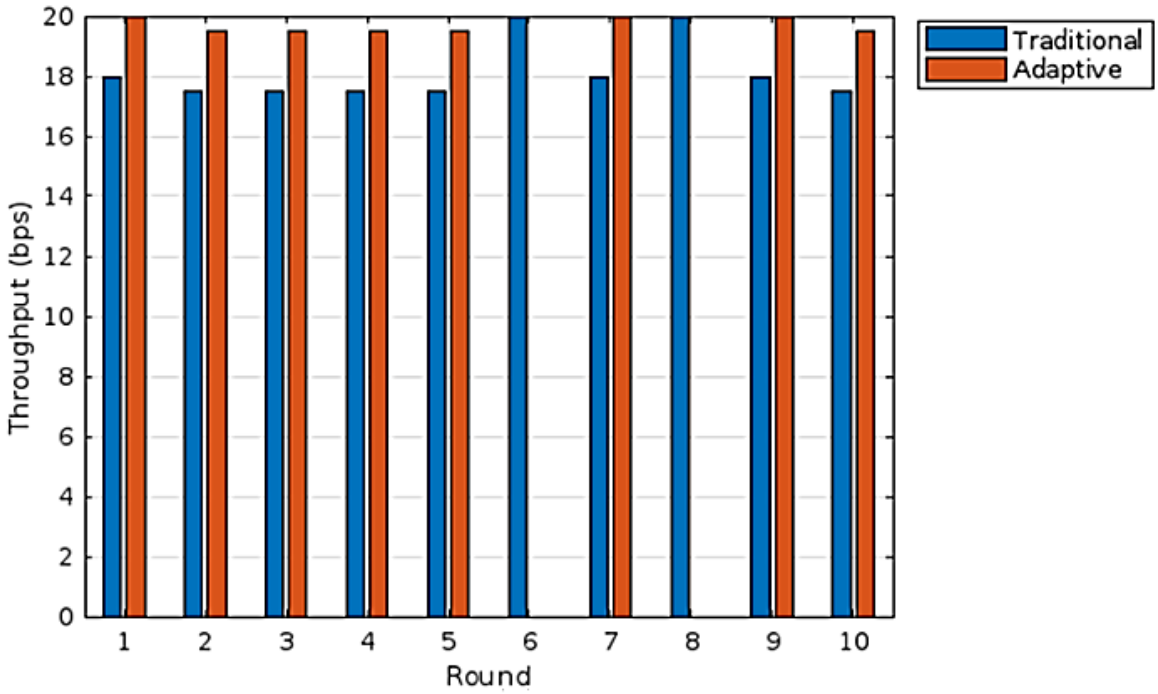

Fig. 3 Throughput Comparison

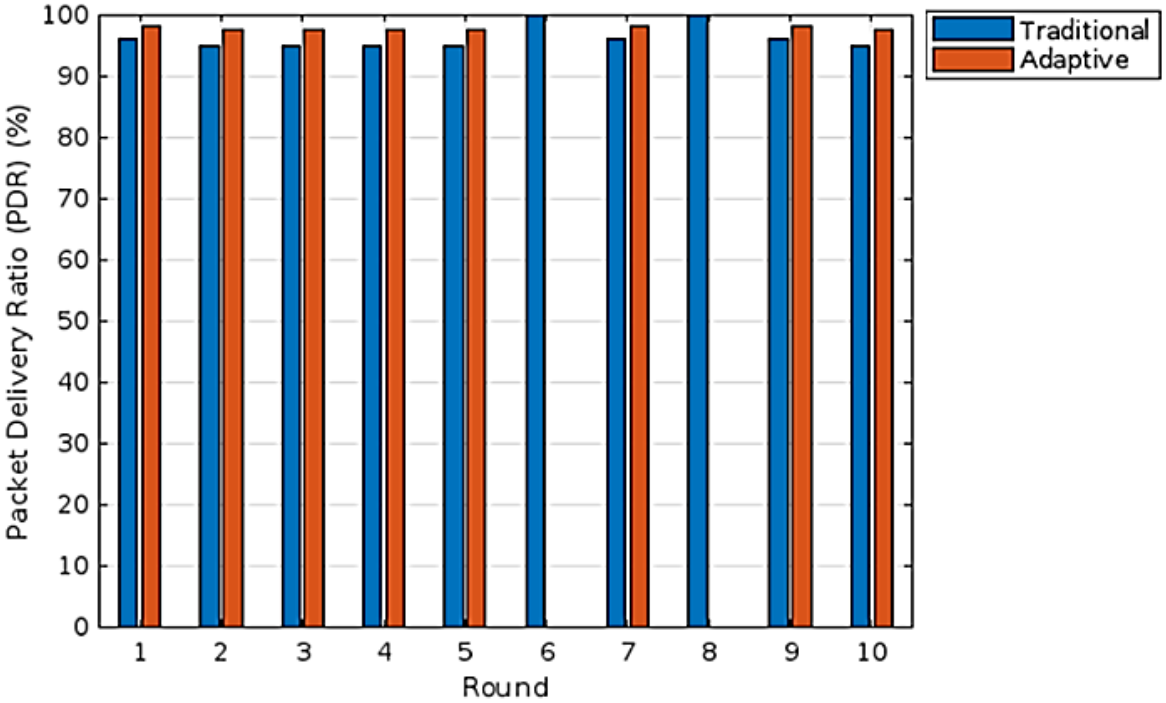

Fig. 4 Packet Delivery Ratio Comparison

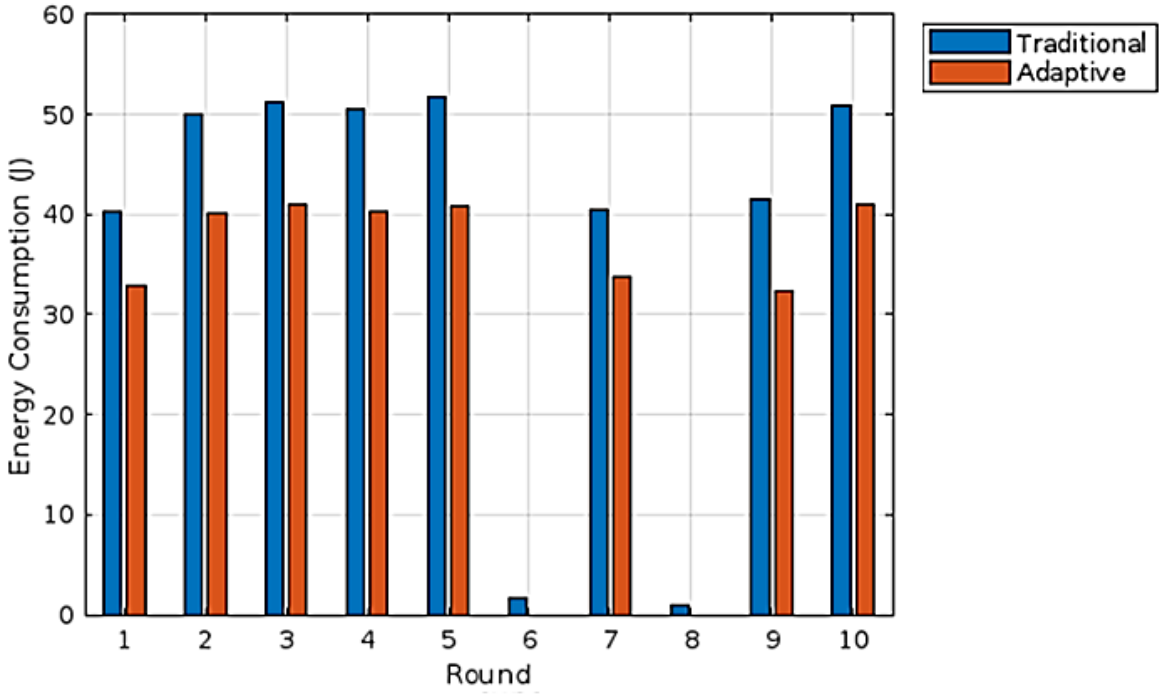

Fig. 5 Energy Consumption Comparison

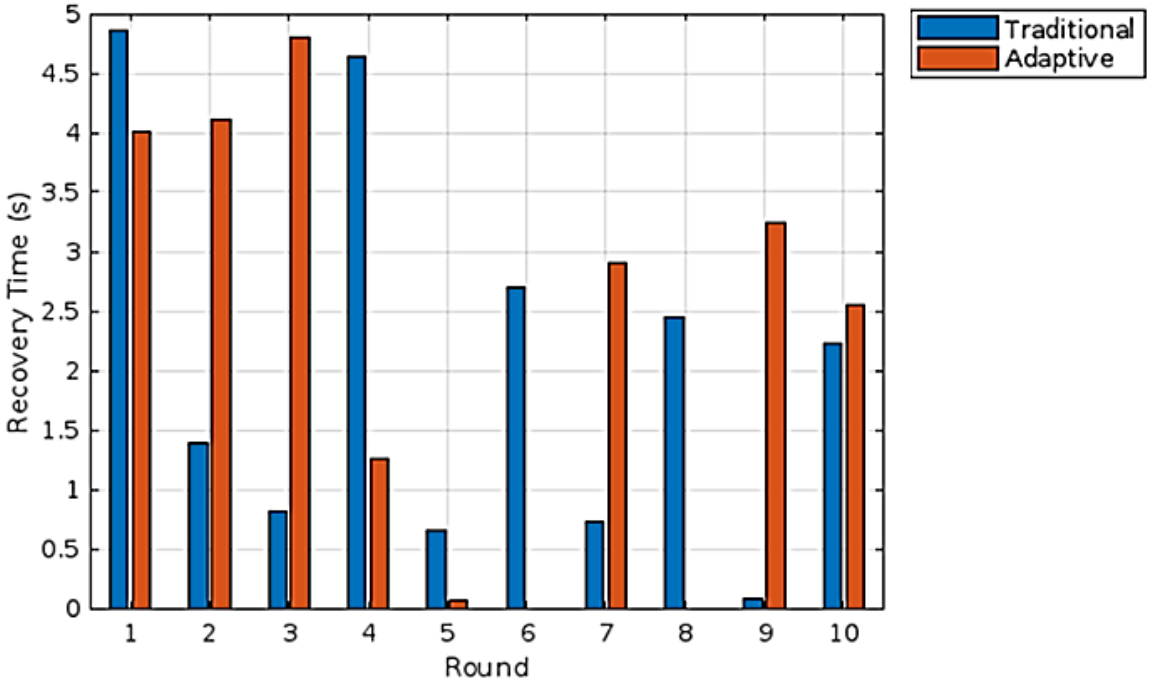

Fig. 6 Recovery Time Comparison

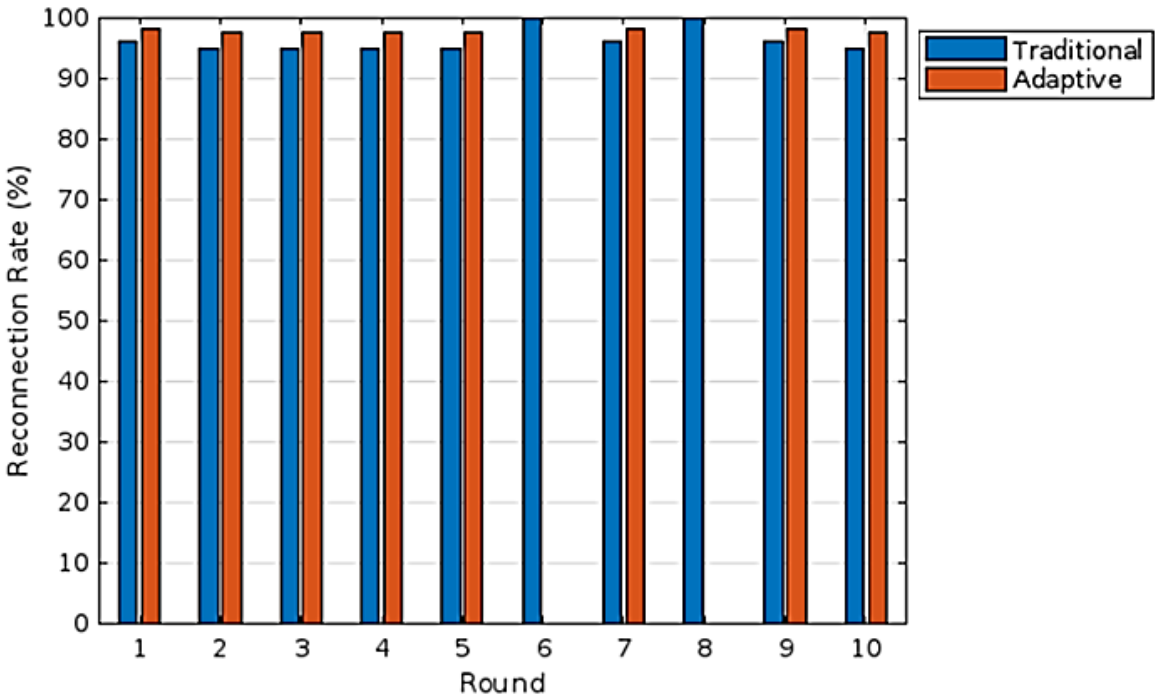

Fig. 7 Reconnection Rate Comparison

REFERENCES

[1] N. F. Tzeng, "Analysis of a variant hypercube topology," Computer Architecture News, vol. 18, no. 3b, pp. 60–70, 1990. https://doi.org/10.1145/255129.255140.

[2] A. Al-Sadi, J. Day, and M. Ould-Khaoua, "Probability-based fault-tolerant routing in hypercubes," Computer Journal, vol. 44, no. 5, pp. 368–377, 2001. https://doi.org/10.1093/comjnl/44.5.368.

[3] M. Liu, "Vertex-fault-tolerant cycles embedding in 4-conditionally faulty enhanced hypercube networks," IEEE Access, 2024. https://doi.org/10.1109/access.2024.3410175.

[4] W. Wu and E. Sabir, "Embedding spanning disjoint cycles in hypercube networks with prescribed edges in each cycle," Axioms, vol. 12, no. 9, art. 861, 2023. https://doi.org/10.3390/axioms12090861.

[5] P. Sarkar, N. De, and A. Pal, "Correction to: On some multiplicative version topological indices of block shift and hierarchical hypercube networks," Opsearch, vol. 59, no. 2, p. 573, 2021. https://doi.org/10.1007/s12597-021-00548-y.

[6] T. Gao and I. Ahmed, "Distance-based polynomials and topological indices for hierarchical hypercube networks," Journal of Mathematics, vol. 2021, pp. 1–11, 2021. https://doi.org/10.1155/2021/5877593.

[7] A. Micheal, L. JiaBao, and S. Arul Jeya, "Vertex decomposition method for wire length problem and its applications to enhanced hypercube networks," IET Computers & Digital Techniques, vol. 13, no. 2, pp. 87–92, 2019. https://doi.org/10.1049/iet-cdt.2018.5100.

[8] S. Zhao and R. Hao, "Reliability assessment of hierarchical hypercube networks," IEEE Access, vol. 7, pp. 54015–54023, 2019. https://doi.org/10.1109/ACCESS.2019.2912014

[9] J. Fang, I. Ahmed, A. Mehboob, K. Nazar, and H. Ahmad, "Irregularity of block shift networks and hierarchical hypercube networks," Journal of Chemistry, vol. 2019, pp. 1–12, 2019. https://doi.org/10.1155/2019/1042308.

[10] Y. Guo, J. Liang, F. Liu, and M. Xie, "Novel fault diagnosis parallel algorithm for hypercube networks," Computer Science, vol. 46, no. 5, pp. 73–76, 2019. https://doi.org/10.11896/j.issn.1002-137X.2019.05.011.

[11] X. Liu, Y. Liu, L. Wang, and J. Zhang, "Hamiltonian cycle embedding with fault-tolerant edges and adaptive diagnosis in half hypercube," The Journal of Supercomputing, vol. 79, no. 10, pp. 14203–14228, Oct. 2023. https://doi.org/10.1007/s11227-023-05674-6.

[12] S. K. Shukla, "Study of logical operations in vectors of a circuit of hypercube," International Journal of Advanced Research in Computer Science, vol. 15, no. 1, pp. 30–32, 2024. https://doi.org/10.26483/ijarcs.v15i1.7051.

[13] N. Gupta, K. S. Vaisla, and R. Kumar, "Design of a structured hypercube network chip topology model for energy efficiency in wireless sensor network using machine learning," SN Comput. Sci., vol. 2, art. 376, 2021. https://doi.org/10.1007/s42979-021-00766-7.

[14] X. Liu and B. Lei, "Enhancing fault tolerance of balanced hypercube networks by the edge partition method," Theoretical Computer Science, 2023. https://doi.org/10.1016/j.tcs.2023.114340.

[15] M. Liu and H. Liu, "Vertex-fault-tolerant cycles embedding on enhanced hypercube networks," Acta Mathematicae Applicatae Sinica, English Series, vol. 32, pp. 187–198, Apr. 2016. https://doi.org/10.1007/s10255-016-0547-z.

[16] D. Jin and H. Li, "The structure fault-tolerance of enhanced hypercube networks," Destech Trans. on Eng. and Tech. Research, 2018. https://doi.org/10.12783/dtetr/ecar2018/26350.

[17] J. Liang and Q. Zhang, "The t/s-diagnosability of hypercube networks under the PMC and comparison models," IEEE Access, vol. 5, pp. 5340–5346, 2017. https://doi.org/10.1109/access.2017.2672602.

[18] A. A. Hnaif, A. A. Tamimi, A. M. Abdalla, and I. H. Jebril, "A fault-handling method for the Hamiltonian cycle in the hypercube topology," Computers, Materials & Continua, vol. 68, no. 1, pp. 505–519, 2021. https://doi.org/10.32604/cmc.2021.016123.

[19] S. F. Naz et al., "Optimizing fault tolerance of RAM cell through MUX based modeling and design using symmetries of QCA cells," Scientific Reports, vol. 14, no. 1, 2024. https://doi.org/10.1038/s41598-024-59185-2.

[20] D. Naik, "QoS in wireless sensor network-fault tolerance and efficient bandwidth allocation," International Journal of Innovative Research in Computer and Communication Engineering, vol. 12, no. 3, pp. 23–29, 2024. https://doi.org/10.15680/ijircce.2024.1203504.

[21] T. L. Phong and T. T. Phuong, "Distributed SignSGD with improved accuracy and network-fault tolerance," IEEE Access, vol. 8, pp. 191839–191849, 2020. https://doi.org/10.1109/access.2020.3032637.

[22] L. Tripathy and C. R. Tripathy, "Hierarchical hexagon: A new fault-tolerant interconnection network for parallel systems," Cybernetics and Information Technologies, vol. 21, no. 1, pp. 32–49, 2021. https://doi.org/10.2478/cait-2021-0003.

[23] Y. Zhang et al., "Fault-tolerant routing algorithm based on disjoint paths in 3-ary n-cube networks with structure faults," The Journal of Supercomputing, vol. 77, pp. 13090–13114, 2021. https://doi.org/10.1007/s11227-021-03799-0.

[24] H. Dong, M. Lv, W. Fan, and G. Wang, "Reliability evaluation of half hypercube networks," Theoretical Computer Science, vol. 975, art. no. 114142, 2023. https://doi.org/10.1016/j.tcs.2023.114142.

[25] M. Abd-El-Barr and F. Gebali, "Reliability analysis and fault tolerance for hypercube

multi-computer networks," Information Sciences, vol. 276, pp. 295–318, 2014. https://doi.org/10.1016/j.ins.2013.10.031.

تعزيز موثوقية تحمل الأخطاء في شبكات المكعبات التشعبية

توركان احمد خليل

قسم هندسة الحاسوب، كلية الهندسة، جامعة الموصل، موصل، العراق

turkan@uomosul.edu.iq

تاريخ الاستلام: 9 ابريل 2025 استلم بصيغته المنقحة: 31 مايو 2025 تاريخ القبول: 8 يوليو 2025

الملخص

يهدف البحث الحالي الى *تحسين موثوقية تحمل الأعطال في الشبكات التشعبية من خلال تطبيق طرق توجيه تكيفية جديدة. استندت الطريقة المقترحة بحل مشكلات استقرار الأداء في بيئات الفشل باستخدام تقنيات التوجيه التكيفية، مما يعزز من موثوقية الشبكة. تشمل الأهداف الأساسية تقليل أوقات الاستجابة مع زيادة معدلات نقل البيانات، وتقليل استهلاك الطاقة، وتوفير استعادة أسرع للأداء، بالإضافة إلى قدرات إعادة الاتصال الأفضل مقارنة بالتقنيات التقليدية لتحمل الأعطال. لقد حققت طريقة التوجيه التكيفي نتائج أفضل من الطرق التقليدية في كافة مقاييس الأداء. اظهرت مقايس الاداء تحسّن واضح مثل تقليل زمن التأخير من 54.5 مللي ثانية (في الطرق التقليدية) إلى 35.7 مللي ثانية (في التوجيه التكيفي)، وارتفاع معدل النقل من 17.5 بت في الثانية إلى 20 بت في الثانية، وتحول وقت الاسترداد من 4.7877 ثانية إلى 0.7786 ثانية، مع زيادة في معدل إعادة الاتصال من 95% إلى 98%. كما انخفض استهلاك الطاقة إلى 33.386 جول بعد أن كان 51.961 جول. تُبرز نتائج البحث الحالي أن طريقة التوجيه التكيفية تُحسن موثوقية تحمل الأعطال في شبكات التشعبية وتُحقق أداءً متفوقًا، مما يجعلها مناسبة لأنظمة الحوسبة المتوازية التي تتطلب عمليات قادرة على تحمل الأعطال.*

الكلمات الدالة:

تحمل الأعطال، الموثوقية، التوجيه التكيفي، شبكات التشعبية، أعطال العقد، أعطال الوصلات.