



EDGEHYDRA: AN EFFICIENT FAULT-TOLERANT EDGE DATA DISTRIBUTION FRAMEWORK USING ERASURE CODING

Noohi Sadiya¹, Dr. Mohd Umar Farooq²

¹PG Scholar, Department of CSE, Shadan Women's College of Engineering and Technology, Hyderabad,
noohisadiya24@gmail.com

²Professor, Department of CSE, Shadan Women's College of Engineering and Technology
umarfarooq.mohd@gmail.com

ABSTRACT:

Low-latency access in edge computing systems depends on the effective distribution of popular data from the cloud to edge servers. Current distribution systems split a file into many data blocks and send them to target edge servers in simultaneously. Once all of the blocks are received, the file is reconstructed. This method speeds up data delivery, but as the successful reconstruction depends on receiving each data block, it is susceptible to transmission delays and errors brought on by network fluctuations or server outages. This study presents EdgeHydra, an erasure coding-based fault-tolerant data distribution strategy, to overcome these constraints. By encoding a file into both data and parity blocks, EdgeHydra enables edge servers to reconstruct the original file from any adequate subset of these blocks without having to wait for all communications to finish. Additionally, it uses a leaderless block supplement mechanism that allows edge servers to coordinate distributedly to recover lost blocks without the need for a centralized controller. According to experimental data, EdgeHydra achieves up to 50.44% quicker distribution times than state-of-the-art techniques, greatly enhancing resilience against delays and failures.

KEYWORDS: Data distribution, distributed transmission, erasure coding, and edge data caching

1. INTRODUCTION

Over the past ten years, the number of smart devices and services has significantly expanded due to the quick development of contemporary technologies like 5G connectivity and artificial intelligence (AI) [1], [2]. Additionally, there is a significant growth in the need for real-time data processing [3], [4], [5]. Data is kept in distant data centers for user access under the conventional cloud computing paradigm. When users access this data, delay is unavoidably introduced by the distance between users and data centers [6], [7]. Applications' performance is greatly impacted by this [8], particularly those that need real-time data access [9]. By moving processing and storage resources from the cloud to the network edge, edge computing has emerged as a viable solution to this basic problem [10], [11], and [12]. App providers can cache popular data on edge servers based on user data requests, allowing users to access it with extremely low latency [13], [14], and [15]. A collection of edge servers spanning various geographical areas within the region make up an edge caching system servicing a certain geographic area [16]. The design of edge caching solutions to satisfy the various demands of app suppliers has been the subject of several recent research [13], [17], [18], [19], and [20]. App providers must transfer data from cloud servers to specific servers inside the edge caching system in order to put these caching solutions into practice. But moving data from the cloud to the periphery is expensive and time-consuming [21], [22]. Edge data distribution (EDD) attempts to send a file f from the cloud server to the system for caching on each of the edge servers in S , given a set of target edge servers in an edge caching system, represented by $S = \{s_1, \dots, s_n\}$. DataSync [23]: DataSync is an end-to-end data transfer service offered by Amazon. There are several ways to send f

to S . As seen in Fig. 1(a), f is sent peer-to-peer directly from the cloud server to the target servers. Low latency in data dissemination is guaranteed via data sync. However, DataSync-based edge data distribution is frequently not cost-effective since the cost of moving data from the cloud to the edge is significantly higher than that of moving data between edge servers (\$0.05–\$0.09 per GB against \$0.01 per GB) [24]. Gossip [25]: In distributed systems, gossip is frequently employed to move data across storage nodes [25]. Under the Gossip method, just one copy of the file is sent from the cloud to the edge caching system in order to save distribution costs. f is initially sent to a chosen entry server based on Gossip, as seen in Fig. 1(b). When all of the target servers get f , the transfer procedure is finished by these edge servers. The Gossip strategy lowers transmission costs when compared to DataSync, but it necessitates talking among edge servers, which frequently results in significant delays in the distribution process [21]. EDD-A [24]: EDD-A is a scheme specifically designed for edge data distribution, considering both transmission delay and cost. EDD-A builds a minimal Steiner tree with the cloud server acting as the root node and the target servers acting as either leaf nodes or intermediate nodes, as seen in Fig. 1(c). After then, the cloud server moves f along the Steiner tree's edges. EDD-A lowers the transmission cost by transferring less data from the cloud server to the edge caching system. However, runtime errors like as server outages and network oscillations might affect the distribution of f under the EDD-A scheme [26]. The entire data distribution process will be greatly impacted by a network delay or server failure that occurs across any path in the Steiner tree. The most advanced solution developed for the EDD problem is called EdgeDis [21]. The app vendor first divides f into a few data



blocks under the Edgedis scheme, as illustrated in Fig. 1(d), and then concurrently sends each block to a designated entry server. The target servers can then combine incoming blocks to reconstitute f after receiving the block from these entry servers. In order to increase the EDD process's resilience, EdgeDis chooses one of the system's edge servers to act as a coordinator and provide any blocks that are missing to the target servers, guaranteeing that all of the target servers will ultimately get f [21]. EdgeHydra can withstand the distribution of m -coded blocks into the edge caching system by distributing coded blocks rather than data blocks. On the other hand, in Stage 2, an entry server could be able to send its coded blocks to some target servers but not to others because of transmission errors. To solve his problem, EdgeDis uses a Raft-like [34] leader election method to choose a target server as a coordinator. The coordinator sends missing data blocks to target servers after gathering data blocks from other target servers. However, given the limited computation and network capabilities of edge servers, having a single server gather and re-transmit all the missing data blocks exerts enormous demands on that server and may exhaust it. We conduct an experiment on EdgeDis to measure the traffic the coordinator experiences when providing incomplete data in different EDD circumstances in order to statistically confirm this. Three parameters are varied in these scenarios: file size, number of target servers, and transmission failure rate. The results of the experiment are shown in Fig. 5. With a 3% transmission failure rate and an average extra traffic of up to 97.27% of the file size, it is clear that the coordinator faced a significant transmission effort. The coordinator's supplement traffic rose along with the number of target servers and file size. Target servers that are lacking coded blocks make sporadic requests to a target server that has them. This lessens the strain on their resources by distributing the supplement workloads over several servers. To reduce any potential delays in the EDD process, missing coded blocks are concurrently delivered to edge servers that require them. Additionally, it eliminates the potential for single-point failure brought on by the coordinator being overburdened by workloads or EdgeDis assaults. Traffic overheads are introduced when heartbeat messages are broadcast. Thankfully, it makes only a very small percentage of the total traffic produced during the EDD process.

2. STIMULATING

In order to provide effective and dependable file delivery in edge computing environments, the EdgeHydra project focuses on developing a strong, fault-tolerant edge data distribution mechanism. Its range consists of: Erasure coding (EC) is used to build data and parity blocks in files, allowing edge servers to reconstruct the original file even if certain blocks are delayed or absent. putting in place a leaderless block supplement system that eliminates the need for a

centralized controller by enabling distributed cooperation among edge servers to recover lost blocks. enhancing edge caching systems' resistance to malfunctioning servers, such as sluggish or unsuccessful entry servers, in order to minimize data distribution disturbance. Comparing EdgeHydra's performance to cutting-edge EDD methods, it is shown to have quicker distribution times, more fault tolerance, and lower transmission overhead. The project mainly focuses on edge networks, such as content delivery networks (CDNs), Internet of Things platforms, and real-time analytics systems, where low latency access, dependability, and fault tolerance are essential for data-intensive applications.

3. OBJECTIVE

The EdgeHydra project's primary goals are: Erasure coding can be used to improve fault tolerance in edge data dissemination by enabling file reconstruction using a subset of coded blocks. to reduce the EDD process's delays and failures brought on by sluggish or unsuccessful entry servers. to create distributed systems that provide scalability and resilience by adding missing blocks without depending on a central controller. to evaluate EdgeHydra's performance in terms of resource efficiency, robustness, and distribution speed against current EDD methods. to show that the suggested plan is workable in real-world edge computing settings with fluctuating workloads and possible server outages.

3.1 What The Project Is About

To guarantee low-latency access for end users in contemporary edge computing systems, it is essential to distribute popular data from the cloud to edge servers effectively. Current data distribution methods split files into many data blocks and send them in parallel to edge servers, which only reassemble the original file once they have received every block. Under ideal circumstances, this method speeds up data transmission, but it is extremely susceptible to server outages, network fluctuations, and runtime exceptions like software bugs, hardware malfunctions, or cyberattacks. These problems may result in transmission failures or delays, which can disrupt edge data distribution (EDD) procedures and raise latency and transmission costs. For example, other servers have to wait for the transfer to finish if one entry server is delayed or dies, which delays the reconstruction of the entire file. Conventional approaches, such as EdgeDis schemes, make the unreasonable assumption that entry servers always function well in real-world edge networks.

4. EXISTING SYSTEM:

- Current edge data distribution schemes split a file into multiple data blocks for parallel transmission from the cloud to edge servers, requiring all blocks to be received for file reconstruction. This method is extremely vulnerable to transmission



failures and delays brought on by server outages and network fluctuations, which might result in data availability disruptions and ineffective distribution procedures.

4.1 Existing System Disadvantages:

- Ineffective and unreliable data dissemination is caused by a lack of fault tolerance.
- N is susceptible to missing or delayed blocks since it requires all data blocks to be received before the file can be reassembled.
- Extremely vulnerable to transmission delays brought on by network variations vulnerable to server malfunctions and outages while data is being sent.

4.2 THE PROPOSED SYSTEM

- This work introduces EdgeHydra, a novel EDD technique that distributes erasure-coded blocks rather than data blocks to lessen the effects of these malfunctioning servers. The cloud server initially uses erasure coding to encode file f into a collection of coded blocks under EdgeHydra. These coded blocks are then sent by the cloud server to entry servers so they may be distributed across the edge caching system.
- In this manner, an edge server can rebuild f without requiring all of the coded blocks. Rather, it simply needs a part of these coded blocks to recreate f . This enables EdgeHydra to withstand sluggish or unsuccessful data transfer from malfunctioning servers. It is EdgeHydra's central concept.

4.2.1 Proposed System Advantage

- lessens reliance on individual blocks, allowing for the tolerance of transmission link failures and delays.
- By using a leaderless block supplement method, distributed coordination is improved and single points of failure are avoided.
- The resilience and reliability of data delivery in edge contexts are greatly enhanced.

5. METHODOLOGIES

1. Edge Cluster Module

The network architecture is ready for effective and safe edge data delivery thanks to the edge server initialization procedure. In order to process data closer to end users rather than just depending on a central cloud server, several edge servers are built and arranged into an edge cluster at this phase. To set them apart from other servers in the cluster, each edge server is given a distinct Edge ID. The system can determine available resources by using the Status attribute, which shows whether the server is online, inactive, or undergoing maintenance. In order to help the system choose the fastest server for data transfer, latency is a representation of the communication delay between the server and users. Bandwidth describes the

server's data transfer capacity, which affects communication speed and effectiveness.

2. Distribute File Module

First, the data is separated into several clusters, each of which is a logical collection of connected data. For safe processing, these clusters are then further divided into blocks. The SHA-256 hashing technique is used to create a distinct 256-bit hash value for each block, which serves as a digital fingerprint of the data. Data integrity and tamper detection are ensured by the fact that even a small change to the block produces an entirely different hash. The block is encoded into a secure format that may be transmitted or stored after hashing. Encoding, SHA-256 hashing, and clustering work together to improve data management efficiency, security, and integrity.

3. Distributed Files Module

In a distributed edge computing setting, the data reconstruction capability is intended to guarantee safe and dependable data recovery. The system checks the availability of the edge servers that hold the distributed data blocks before allowing the user to rebuild and download the original data. To effectively rebuild the entire file, there must be at least five active edge servers. The system gathers the necessary data blocks from various servers, checks their integrity, and combines them to replicate the original data if at least five edges are active. Because the data is still recoverable even in the event that certain edge servers malfunction or stop working, this method increases fault tolerance. Nevertheless, the reconstruction process is rejected if there are fewer than five operational edge servers, preventing the generation of incomplete or damaged data.

4. Benchmark: Edge Hydra vs Replication Module

Data availability, reaction time, storage efficiency, bandwidth usage, and fault tolerance are all assessed in the benchmark comparison between the replication-based and edge-based storage approaches. The edge-based strategy enables users to reach adjacent servers with reduced latency and better download rates by dynamically distributing data over several edge servers. Through threshold-based reconstruction, this technique minimizes network congestion and enhances resource utilization while preserving data reliability. The replication strategy, on the other hand, uses additional bandwidth during synchronization and increases storage overhead by storing many full copies of the same data on other servers. According to experimental findings, the edge-based strategy improves scalability, lowers storage costs, and offers faster, lower-latency data access. Replication delivers great availability, but it needs a lot more storage and upkeep. As a result, the benchmark shows that edge-based data distribution offers a more effective, economical, and scalable way to store and retrieve secure data.



6. ALGORITHM USED

6.1 Proposed Algorithm

➤ EdgeHydra and Fault tolerance

A fault-tolerant edge data distribution technique called EdgeHydra was created to provide dependable and low-latency data delivery in edge computing settings, which is necessary for many security applications. A file is divided into many data blocks using traditional edge data distribution systems, and in order to reassemble the original file, each block must be correctly delivered. They are therefore extremely susceptible to server outages, transmission delays, and network volatility. By incorporating erasure coding into the data distribution process, EdgeHydra gets around these restrictions and enables edge servers to reassemble files even in cases when certain blocks are absent or delayed. The fundamental concept of EdgeHydra is to use erasure coding techniques to encode an original file into a series of data and parity blocks. An edge server just requires a sufficient portion of these coded blocks to reconstruct the file, rather than relying on every block. Because of this, the system can withstand unreliable or sluggish entry servers without affecting the distribution process as a whole. This greatly increases resilience and cuts down on wait times brought on by slow or malfunctioning servers. EdgeHydra adds a leaderless block supplement technique to further improve fault tolerance. This method removes the need for a centralized controller by having edge servers work directly with one another to exchange and add missing coded blocks. Even in the event of fluctuating workloads or partial network disruptions, this decentralized coordination guarantees continuous data availability and enhances scalability. All things considered, EdgeHydra offers robust fault tolerance by fusing decentralized recovery techniques with erasure coding. Experimental evaluations show that it significantly reduces distribution latency and improves reliability compared to traditional edge data distribution methods, making it well-suited for real-world edge environments where failures and delays are common.

6.2 EXISTING ALGORITHM

➤ Data Distribution Scheme

Rows with defined values that match one of the specified values in a list of discrete values are placed in the same fragment under this kind of fragmentation. A list of one or more constant expressions that match one or more table columns is specified for each fragment. the column or group of columns that the fragment key is derived from. A leftover fragment, which contains all rows that don't fit the requirements for any other fragment, can be optionally specified. Additionally, because its fragment expression is NULL or IS NULL, you can optionally supply a NULL fragment that saves rows with missing data in the fragment key fields. The requirement that each value in the list for each fragment be distinct from all the lists for fragments of

the same table or index is the primary distinction between fragmentation by list and fragmentation by expression.

7. SYSTEM ARCHITECTURE

In contemporary edge computing settings enabled by 5G technologies, the EdgeHydra system architecture supports effective, economical, and fault-tolerant edge data distribution (EDD). By providing intelligent data distribution and recovery techniques among cloud servers, entry servers, and edge servers, the architecture expands on conventional cloud-centric approaches.

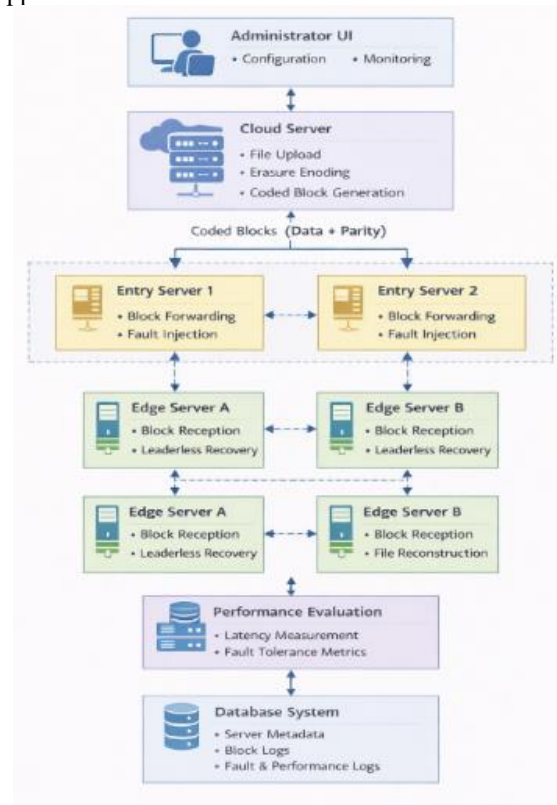


FIG:7 System Architecture

Its main objective is to get beyond the drawbacks of current EDD techniques, such large backhaul traffic, expensive transmission, lengthy distribution delays, and runtime failure susceptibility. A Cloud Server serves as the primary data source at the top level of the architecture. Application suppliers upload and keep popular application data first on the cloud. The cloud server uses erasure coding to encode each file into a series of data and parity blocks rather than sending whole files straight to every edge server. Because this encoding approach only permits file reconstruction from a fraction of coded blocks, the system can withstand transmission delays and server outages.



8. RESULTS

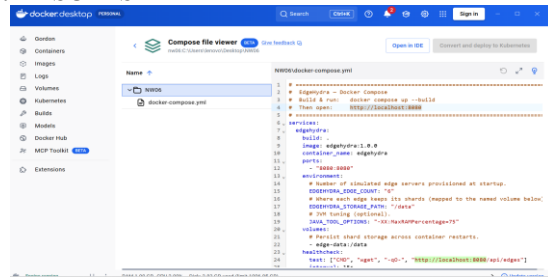


Fig 8.1: Docker Containers Running EdgeHydra Services

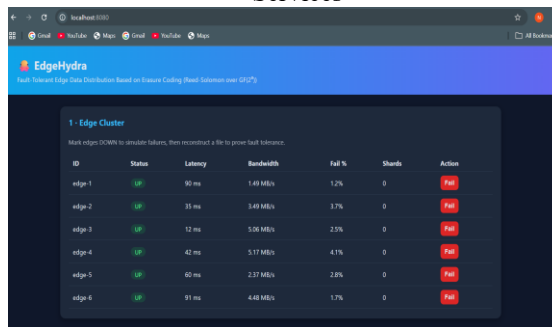


Fig 8.2: Edge Clustering

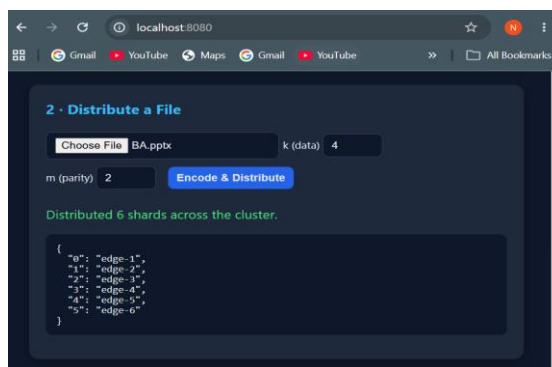


Fig 8.3: File Upload to the EdgeHydra System

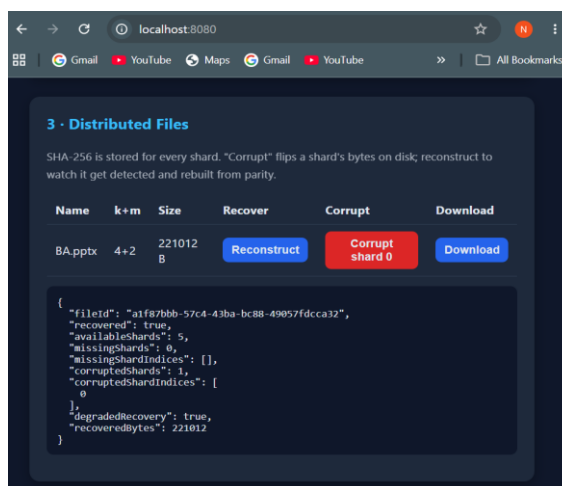


Fig 8.4: Recovery of Original File Using Erasure Coding

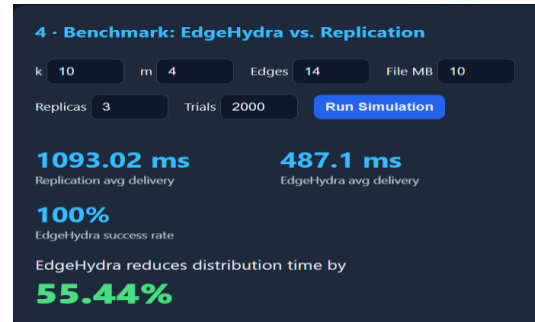


Fig 8.5 Performance Evaluation Results

9. CONCLUSION

In order to overcome the drawbacks of conventional parallel block distribution techniques in edge computing settings, this work introduced EdgeHydra, an erasure coding-based edge data distribution (EDD) approach. EdgeHydra eliminates reliance on sluggish or unreliable servers by encoding files into coded blocks that allow edge servers to reconstruct data using just a sufficient portion of blocks. A decentralized block supplement technique eliminates the need for centralized coordination and also guarantees balanced transmission burdens. Experiments show that EdgeHydra reduces distribution time by up to 50.54% as compared to the most advanced EDD techniques, thereby enhancing resilience and efficiency under dynamic network circumstances. In practical edge computing systems, EdgeHydra is an efficient, scalable, and fault-tolerant method for dependable data delivery.

9.1 FUTURE ENHANCEMENT

Even though EdgeHydra offers significant fault tolerance and performance gains, there are a number of improvements that might be investigated in further research. To further maximize performance, adaptive erasure coding settings may be dynamically adjusted based on failure rates, workload intensity, and real-time network circumstances. Second, using machine learning-based intelligent fault prediction techniques might proactively detect possible sluggish or malfunctioning servers and modify block distribution tactics appropriately. Furthermore, EdgeHydra may be expanded to accommodate diverse edge settings with different capacities for computing and storage. Lastly, EdgeHydra would be appropriate for critical and extensive edge applications if it were integrated with security features like encryption, access control, and secure block verification. Additionally, future work can focus on implementing energy-aware storage and recovery mechanisms to reduce power consumption in resource-constrained edge environments. The integration of latency-aware data placement strategies can further improve response times by storing data closer to end users and frequently accessed applications. Another promising direction is the development of self-healing capabilities that automatically detect node failures, reconstruct missing data blocks, and rebalance workloads without human



intervention. EdgeHydra can also be extended to support multi-cloud and hybrid-cloud infrastructures, enabling seamless data sharing, backup, and disaster recovery across distributed environments. Moreover, blockchain-based auditing and integrity verification techniques may be incorporated to provide transparent and tamper-resistant tracking of storage operations. Finally, AI-driven resource optimization approaches can continuously analyze workload patterns and network conditions to improve data placement, redundancy management, and recovery scheduling, making EdgeHydra more scalable, reliable, and efficient for next-generation edge computing applications.

REFERENCES

- [1] A. Osseiran et al., "The foundation of the mobile and wireless communications system for 2020 and beyond: Challenges, enablers and technology solutions," in *Proc. IEEE Veh. Technol. Conf.*, 2013, pp. 1–5.
- [2] Z. Xu et al., "Age-aware data selection and aggregator placement for timely federated continual learning in mobile edge computing," *IEEE Trans. Comput.*, vol. 73, no. 2, pp. 466–480, Feb. 2024.
- [3] R. Bhardwaj et al., "Ekya: Continuous learning of video analytics models on edge compute servers," in *Proc. 19th USENIX Symp. Netw. Syst. Des. Implementation*, 2022, pp. 119–135.
- [4] X. Hou and S. Dey, "Motion prediction and pre-rendering at the edge to enable ultra-low latency mobile 6DoF experiences," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 1674–1690, 2020.
- [5] F. Nawab, D. Agrawal, and A. El Abbadi, "DPaxos: Managing data closer to users for low-latency and mobile applications," in *Proc. Int. Conf. Manage. Data*, 2018, pp. 1221–1236.
- [6] Amazon, "AWS wavelength for media & entertainment," 2021. [On line].
- [7] Z. Xu, Y. Fu, Q. Xia, and H. Li, "Enabling age-aware Big Data analytics in serverless edge clouds," in *Proc. IEEE Conf. Comput. Commun.*, 2023, pp. 1–10.
- [8] E. Schurman and J. Brutlag, "The user and business impact of server delays, additional bytes, and HTTP chunking in web search," *Velocity Web Perform. Operations Conf.*, O'Reilly, 2009.
- [9] S.-C. Lin et al., "The architectural implications of autonomous driving: Constraints and acceleration," in *Proc. 23rd Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2018, pp. 751–766.
- [10] S. Ma, S. Guo, K. Wang, W. Jia, and M. Guo, "A cyclic game for service-oriented resource allocation in edge computing," *IEEE Trans. Serv. Comput.*, vol. 13, no. 4, pp. 723–734, Jul./Aug. 2020.
- [11] Z. Xu et al., "Collaborate or separate? distributed service caching in mobile edge clouds," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 2066–2075.
- [12] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Fourth Quarter 2017.
- [13] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, "Online collaborative data caching in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 281–294, Feb. 2021.
- [14] J. Zhou, F. Chen, Q. He, X. Xia, R. Wang, and Y. Xiang, "Data caching optimization with fairness in mobile edge computing," *IEEE Trans. Serv. Comput.*, vol. 16, no. 3, pp. 1750–1762, May/Jun. 2023.
- [15] R. Luo, H. Jin, Q. He, S. Wu, and X. Xia, "Enabling balanced data deduplication in mobile edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 5, pp. 1420–1431, May 2023.
- [16] X. Xia et al., "Formulating cost-effective data distribution strategies online for edge caches systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4270–4281, Dec. 2022.
- [17] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 447–457, Jan. 2020.
- [18] B. Li et al., "Cooperative assurance of cache data integrity for mobile edge computing," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 4648–4662, 2021.
- [19] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 207–215.
- [20] J. Peng et al., "MagNet: Cooperative edge caching by automatic content congregating," in *Proc. ACM Web Conf.*, 2022, pp. 3280–3288.
- [21] B. Li, Q. He, F. Chen, L. Lyu, A. Bouguettaya, and Y. Yang, "EdgeDis: Enabling fast, economical, and reliable data dissemination for mobile edge computing," *IEEE Trans. Serv. Comput.*, vol. 17, no. 4, pp. 1504–1518, Jul./Aug. 2024.
- [22] Q. Xia, Z. Xu, W. Liang, S. Yu, S. Guo, and A. Y. Zomaya, "Efficient data placement and replication for QoS-aware approximate query evaluation of Big Data analytics," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 12, pp. 2677–2691, Dec. 2019.
- [23] Amazon, "AWS DataSync," 2018, Accessed: Mar. 20, 2022. [Online].
- [24] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, "Cost effective app data distribution in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 31–44, Jan. 2021.
- [25] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2508–2530, Jun. 2006.
- [26] L. Yuan et al., "CoopEdge: A decentralized blockchain-based platform for cooperative edge computing," in *Proc. Web Conf.*, 2021, pp. 2245–2257.
- [27] B. Li, Q. He, F. Chen, H. Jin, Y. Xiang, and Y. Yang, "Auditing cache data integrity in the edge



- computing environment,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1210–1223, May 2021.
- [28] B.Li, Q. He, L.Yuan, F. Chen, L. Lyu, and Y. Yang, “EdgeWatch: Collaborative investigation of data integrity at the edge based on blockchain,” in *Proc. 28th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2022, pp. 3208–3218.
- [29] Q. He et al., “A game-theoretical approach for mitigating edge DDoS attack,” *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 4, pp. 2333–2348, Jul./Aug. 2022.
- [30] C. Huang et al., “Erasure coding in windows azure storage,” in *Proc. USENIX Annu. Tech. Conf.*, 2012, pp. 15–26.
- [31] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, “A hitchhiker’s guide to fast and efficient data reconstruction in erasure-coded data centers,” in *Proc. ACM SIGCOMM Conf.*, 2014, pp. 331–342.
- [32] J. Yang, A. Sabnis, D. S. Berger, K. Rashmi, and R. K. Sitaraman, “C2DN: How to harness erasure codes at the edge for efficient content delivery,” in *Proc. 19th USENIX Symp. Netw. Syst. Des. Implementation*, 2022, pp. 1159–1177.
- [33] H. Jin, R. Luo, Q. He, S. Wu, Z. Zeng, and X. Xia, “Cost-effective data placement in edge storage systems with erasure code,” *IEEE Trans. Serv. Comput.*, vol. 16, no. 2, pp. 1039–1050, Mar./Apr. 2023.
- [34] D. Ongaro and J. Ousterhout, “In search of an understandable consensus algorithm,” in *Proc. USENIX Annu. Tech. Conf.*, 2014, pp. 305–319.
- [35] Y. Sahni, J. Cao, L. Yang, and Y. Ji, “Multi-hop multi-task partial computation offloading in collaborative edge computing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1133–1145, May 2021.
- [36] L. He, F. Li, H. Xu, W. Xia, X. Zhang, and X. Tao, “Blockchain-based vehicular edge computing networks: The communication perspective,” *Sci. China Inf. Sci.*, vol. 66, no. 7, 2023, Art. no. 172301.
- [37] S. Li and T. Lan, “Hotdedup: Managing hot data storage at network edge through optimal distributed deduplication,” in *Proc. Conf. Comput. Commun.*, 2020, pp. 247–256.
- [38] X. Xia et al., “Data, user and power allocations for caching in multi access edge computing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 5, pp. 1144–1155, May 2022.
- [39] X. Xia et al., “Budgeted data caching based on k-median in mobile edge computing,” in *Proc. 27th IEEE Int. Conf. Web Serv.*, 2020, pp. 197–206.
- [40] T. X. Tran and D. Pompili, “Adaptive bitrate video caching and processing in mobile-edge computing networks,” *IEEE Trans. Mobile Comput.*, vol. 18, no. 9, pp. 1965–1978, Sep. 2019.
- [41] T. Zhao, I.-H. Hou, S. Wang, and K. Chan, “Red/LeD: An asymptotically optimal and scalable online algorithm for service caching at the edge,” *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1857–1870, Aug. 2018.
- [42] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. S. Shen, “Content popularity prediction towards location-aware mobile edge caching,” *IEEE Trans. Multimedia*, vol. 21, no. 4, pp. 915–929, Apr. 2019.
- [43] Y. Jiang, M. Ma, M. Bennis, F.-C. Zheng, and X. You, “User preference learning-based edge caching for fog radio access network,” *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 1268–1283, Feb. 2019.
- [44] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. S. Shen, “Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing,” *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 968–980, Third Quarter 2021.
- [45] S. M. Azimi, O. Simeone, A. Sengupta, and R. Tandon, “Online edge caching and wireless delivery in fog-aided networks with dynamic content popularity,” *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1189–1202, Jun. 2018.
- [46] J. Gao, S. Zhang, L. Zhao, and X. S. Shen, “The design of dynamic probabilistic caching with time-varying content popularity,” *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1672–1684, Apr. 2021.