AN ANALYTICAL STUDY ON THE 2025 AWS SERVER OUTAGE: CAUSES, IMPACTS, AND FUTURE RESILIENCE STRATEGIES

BY

ABHISHEK HANAMANT KINAGI

Email: Kinagiabhishek842@gmail.com

DEPARTMENT OF COMPUTER SCIENCE, PES UNIVERSITY

BENGALURU, KARNATAKA INDIA

November,2025

Abstract:

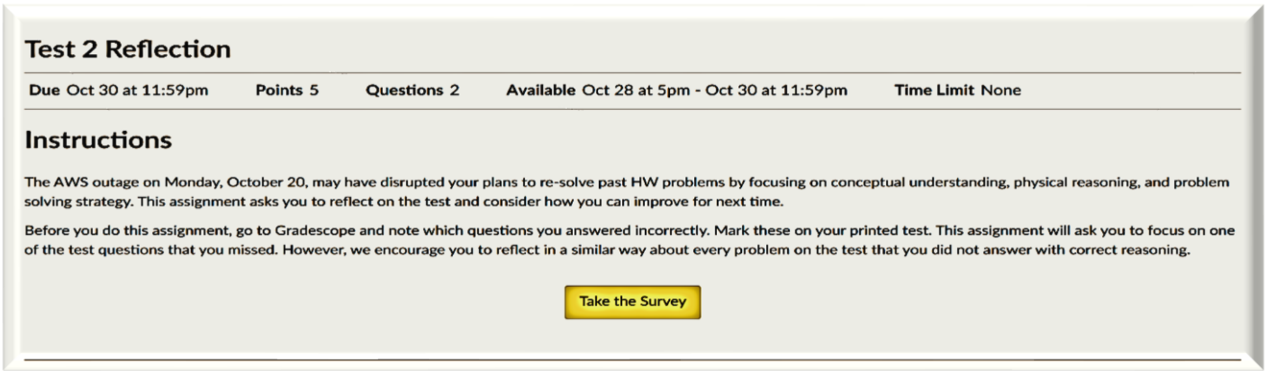
This paper investigates the root technical causes, cascading impacts, and algorithmic parallels underlying the Amazon Web Services (AWS) outage of October 2025. By mapping foundational computer science algorithms—such as Union–Find, Dynamic Programming, and Topological Sort—to fault-tolerance patterns observed in this real-world cloud incident, the study proposes analytical models for improving resilience in distributed systems. Bridging classical algorithmic theory with modern cloud architecture, the proposed framework demonstrates how algorithmic thinking can enhance fault detection, failure containment, and disaster recovery, offering a novel and educational perspective for future cloud resilience engineering.

I. Introduction:

Cloud computing underpins modern digital infrastructure, and outages in leading providers such as Amazon Web Services (AWS) can have world-spanning, cascading effects. In October 2025, AWS suffered a major outage, disrupting countless businesses and users. This work selects that event to analyze causes, impacts, and importantly, to draw parallels between key computer science algorithms (such as Union-Find, Dynamic Programming, and Topological Sort) and the failure and recovery mechanisms in large-scale fault-tolerant systems.

II. AWS October 2025 Outage: Event Overview

On October 20, 2025, AWS suffered a major, multi-hour outage originating in the US-EAST-1 region. The primary trigger was a rare race condition in DynamoDB's automated DNS management system. The fault resulted in the deletion of critical DNS records, making DynamoDB endpoints unreachable. Automated failover monitoring and network health checks, instead of isolating the fault, contributed to cascading failures and broad service disruption in EC2, Lambda, and other AWS products. The AWS outage's reach extended beyond businesses to academic environments as well. For instance, several universities posted assignment notices acknowledging the outage (Fig. 1), showing how deeply cloud dependency affects digital learning platforms.



Test 2 Reflection

Due Oct 30 at 11:59pm **Points** 5 **Questions** 2 **Available** Oct 28 at 5pm - Oct 30 at 11:59pm **Time Limit** None

Instructions

The AWS outage on Monday, October 20, may have disrupted your plans to re-solve past HW problems by focusing on conceptual understanding, physical reasoning, and problem solving strategy. This assignment asks you to reflect on the test and consider how you can improve for next time.

Before you do this assignment, go to Gradescope and note which questions you answered incorrectly. Mark these on your printed test. This assignment will ask you to focus on one of the test questions that you missed. However, we encourage you to reflect in a similar way about every problem on the test that you did not answer with correct reasoning.

Figure 1. Educational acknowledgment of AWS outage disruption (October 2025).

III. Root Causes and Technical Analysis

- **DNS Failure & Race Condition** - AWS's DNS management for DynamoDB involves distributed DNS Planners and Enactors. During the outage, two Enactors (acting in different data centers) competed to process DNS plans. A race condition caused an outdated plan cleanup operation to erase DynamoDB's DNS records, disconnecting services at the network layer.
- **Cascading System Failures** - Dependency on DynamoDB: AWS EC2, Lambda, and other services require DynamoDB for lease management and state. With DNS failures, these services could not perform normal operations and began to fail. The EC2 state-management backlog and health-check failures led to even broader impact. Systems could not recover quickly even after DNS restoration, due to accumulated inconsistency and network health check backlogs.
- **Failure Amplification** - Cloud region centralization: The incident shows the risks of tightly-coupled, high-intensity dependencies in a concentrated cloud region. Automated monitoring and failover, while designed for reliability, can backfire if not designed with distributed failure scenarios in mind.

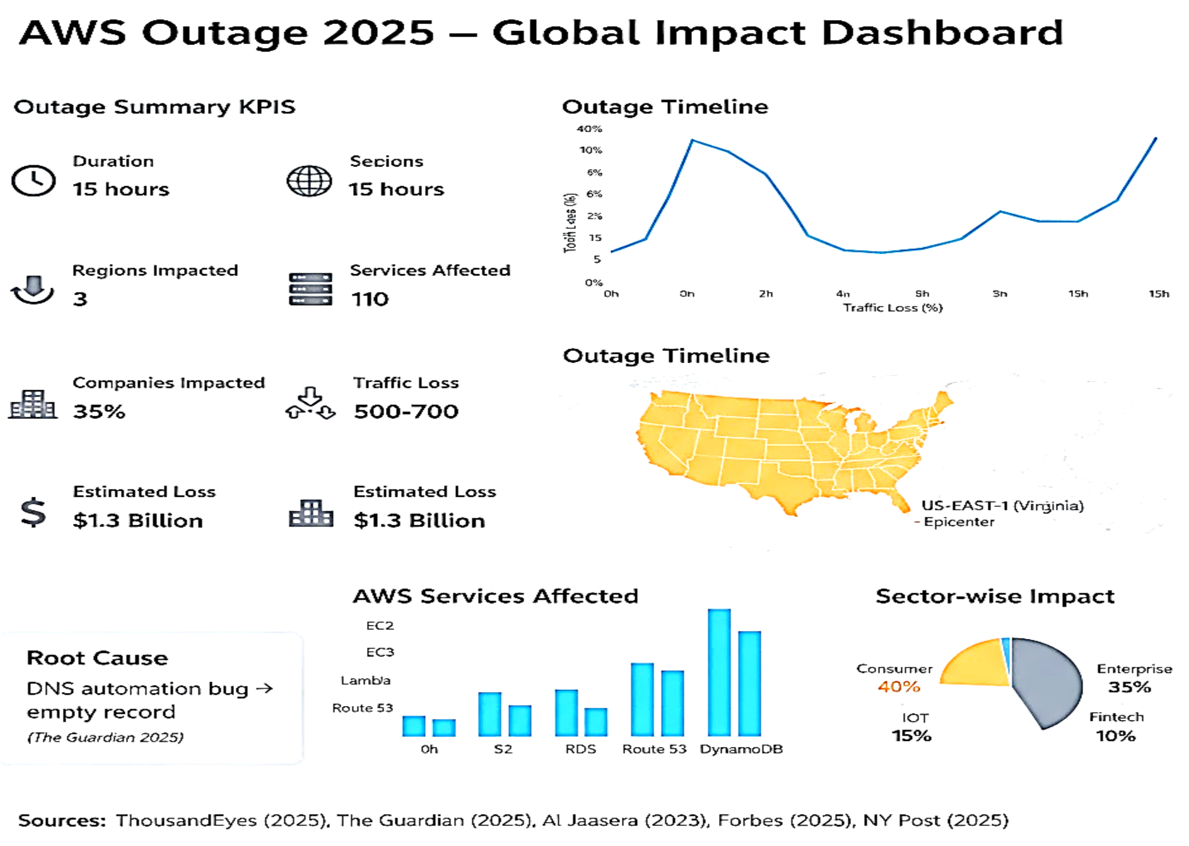


Figure 2A synthesized visualization of the October 2025 AWS outage showing estimated duration (≈ 15 hours), affected regions (3), impacted services (~110), and economic loss (~ US $1.3 billion). Data aggregated from multiple public analyses including ThousandEye

IV. Algorithmic Parallels in Cloud Resilience: From LeetCode to Large-Scale Outages

Modern algorithmic concepts underlie system reliability. This section maps three common algorithmic problems to incident-linked system behaviours and recovery models.

- **Union–Find / Disjoint Set: Detecting Network Partition and Connectivity Recovery**
  Algorithm summary: Union–Find tracks disjoint sets and efficiently detects whether two nodes (services) belong to the same component.

  AWS Outage Analogy: During the outage, AWS Availability Zones and services effectively became isolated due to network partition, mirroring the concept of disjoint sets. A Union–Find inspired detection mechanism could continuously monitor the overall connectivity of a cloud region, raising real-time alerts when a service (e.g., DynamoDB regionally) becomes partitioned from others.

  Improvement Insight: Borrowing from real-time disjoint set detection allows designers to track and remediate accidental partitions and assure continuous system health monitoring, especially in multi-region cloud setups.
  Link:https://www.geeksforgeeks.org/dsa/introduction-to-disjoint-set-data-structure-or-union-find-algorithm/

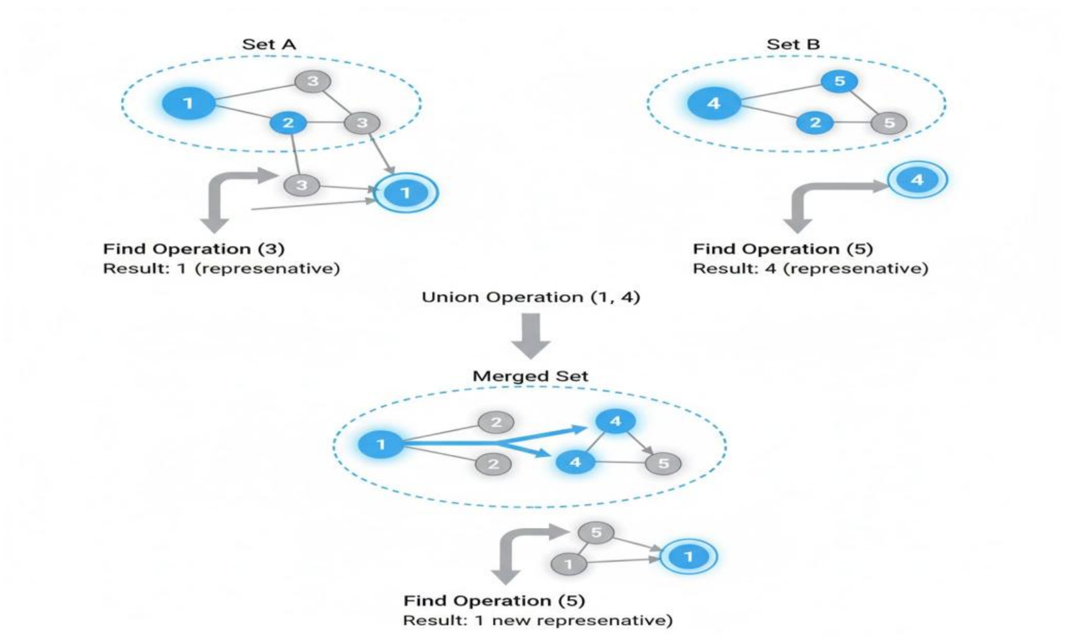


Figure 3.Union–Find structure illustrating detection of network partition and reconnection between AWS regions.

- **Dynamic Programming / Distributed Caching: Avoiding Repeated Work and Failure Amplification**
  Algorithm summary: Dynamic programming (DP) decomposes problems into overlapping subproblems, saving results to prevent redundant computation (memoization or distributed caching).

  AWS Outage Analogy: In the outage, DNS failures meant that repeated dependency resolutions (lookup for endpoints) failed, and retry storms erupted, amplifying the outage load. A DP-like adaptive caching/memoization could help: correctly caching DNS resolutions would limit retries, isolate faults, and promote graceful fallback. Distributed caching strategies further avoid systemic overload in case of partial failures.

  Improvement Insight: Resilience protocols can leverage Dynamic Programming/caching approaches to limit cascading retry storms and failover amplification in distributed cloud systems when a core component is unreachable.
  Link: https://www.geeksforgeeks.org/system-design/what-is-a-distributed-cache/

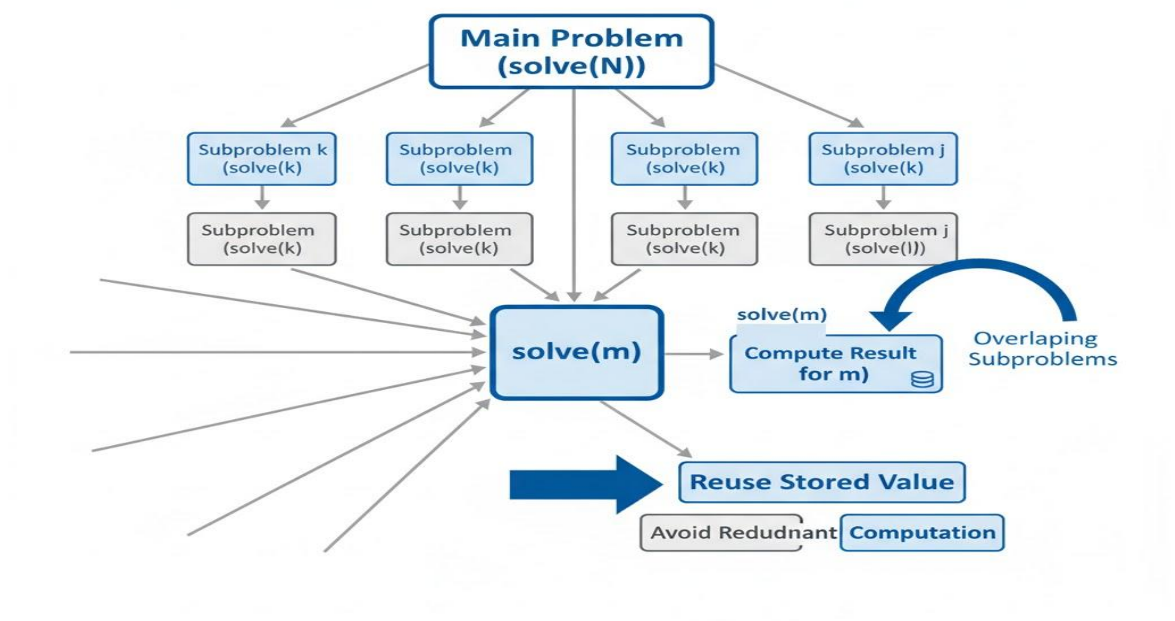


Figure 4.Dynamic Programming analogy showing adaptive caching and retry optimization for distributed resilience.

- **Topological Sort / Dependency Graph: Managing Dependency Chains and Safe Recovery**
  Algorithm summary: Topological sort produces an execution/activation order in a directed acyclic graph so that all dependencies of a node are satisfied before the node runs.
  AWS Outage Analogy: AWS system dependencies (e.g., Lambda → API Gateway → DynamoDB → DNS) mirror a DAG. When a lower-level dependency (like DNS) fails, all upstream services also stop. Outages propagate up the chain (the dependency order). A topological sort–inspired recovery protocol can map the dependency DAG and orchestrate the recovery/restart order for services, prioritizing core dependencies first to achieve a safe and efficient restoration.

  Improvement Insight: Mapping and tracking explicit dependency graphs, and using topological sort algorithms, could help cloud providers automate and optimize failover/recovery in future outages.
  Link: https://www.geeksforgeeks.org/dsa/topological-sorting/

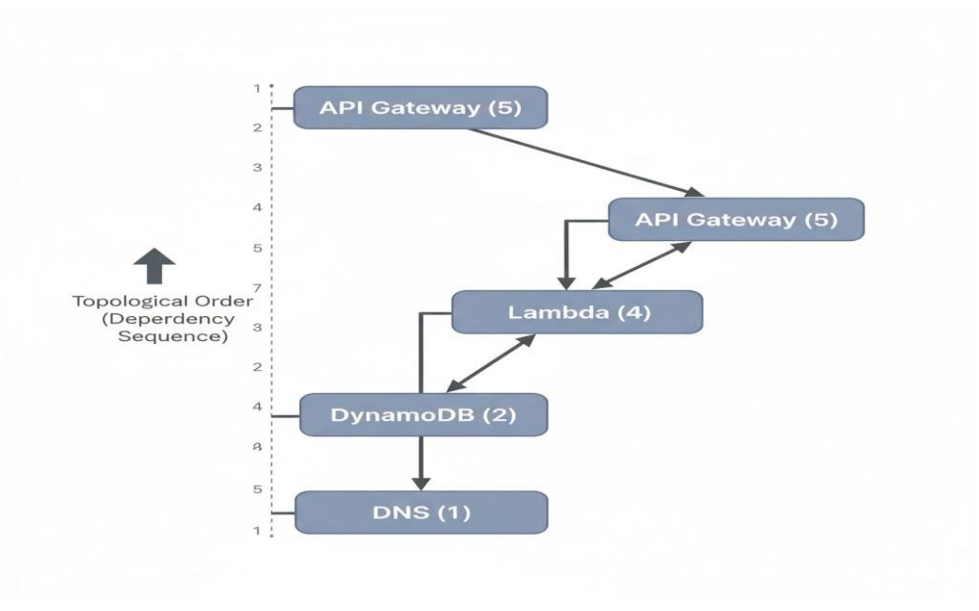


Figure 5.Topological ordering of service dependencies for structured recovery sequence after cloud outage.

V. Discussion and Recommendations

Connecting these core **CS algorithms** to large-scale cloud incidents extends theoretical computer science into applied fault-tolerance practice.

- Early detection of network partitions (Union–Find)
- Smarter load and retry management under extreme partial failure (DP/caching)
- Automated, dependency-respecting disaster recovery sequences (Topological Sort)

Systematic application of these algorithmic models in monitoring, failover, caching, and recovery workflows can make cloud environments more robust and failure-aware.

VI. Conclusion

The 2025 AWS outage highlights both the technological fragility and the sophisticated system design challenges faced in distributed cloud environments. By recognizing that foundational algorithmic principles (as taught in standard computer science courses and reinforced by algorithmic problem-solving platforms) directly inform real-world fault tolerance and resilience mechanisms, both academics and practitioners can drive better research and smarter design.

Future efforts—both by cloud architects and computer science students—should foster tighter collaboration between theoretical computer science and reliability engineering to build next-generation cloud platforms that are not just scalable and performant, but verifiably resilient s in the face of rare and unexpected failures.

VII. Proposed Algorithmic Resilience Framework (ARF)

This framework integrates detection (Union–Find), stabilization (DP/memoization), and recovery sequencing (Topological Sort) into a unified resilience model for distributed clouds.

- *Detection Layer*: Continuously identifies partitioned services using disjoint-set monitoring.
- *Stabilization Layer*: Prevents overload via adaptive caching of transient failures.
- *Recovery Layer*: Automates dependency-ordered restart and verification.

Together, these layers form a verifiable resilience pattern applicable beyond AWS to any distributed platform.

"Similar control-plane dependencies were seen in AWS (2021) and Azure (2023) outages, highlighting a recurring fragility in centralized DNS management."

VIII. Acknowledgments

The author thanks Amazon Web Services Post-Event Reports, ThousandEyes Network Analytics, and open-access research communities for providing valuable technical insights and post-incident data.

IX. Future Work and Limitations

While this study introduces a novel algorithmic framework for interpreting cloud outage behaviour, its findings are primarily conceptual and based on publicly available data. The lack of internal AWS telemetry limits quantitative verification of the proposed resilience models.

Simulation and Validation: Implementing prototype systems to test *Union–Find–based partition detection*, *Dynamic Programming–inspired caching strategies*, and *Topological Sort–driven recovery ordering* under fault-injection conditions.

Cross-Cloud Comparative Analysis: Extending the framework to multi-cloud ecosystems (AWS, Azure, GCP) to evaluate interoperability and resilience across providers

Machine Learning Integration: Leveraging anomaly detection and predictive analytics to automate algorithmic fault detection in real-time environments.

Formal Verification: Applying formal methods and graph-based verification to ensure provable reliability in large-scale distributed systems.

Despite these limitations, the proposed *Algorithmic Resilience Framework (ARF)* provides a structured foundation for bridging algorithmic theory and practical reliability engineering. The results motivate continued collaboration between cloud architects and computer science researchers to develop verifiably fault-tolerant systems.

Reference:

[1] AWS Plain English, "AWS October 2025 Outage Analysis and Technical Overview," aws.plainenglish.io, Oct. 2025.
[2] ThousandEyes, "AWS Outage Analysis – October 2025," thousandeyes.com, Oct. 2025.
[3] INE, "AWS October 2025 Outage: Multi-Region and Cloud Lessons Learned," ine.com, Oct. 2025.
[4] CRN, "Inside AWS's 2025 Cloud Failure," crn.com, Oct. 2025.
[5] Faun, "Dependency Graphs in Distributed Systems," faun.pub, 2025.
[6] AlgoDaily, "Union-Find and Disjoint Set Algorithms Explained," algodaily.com, 2025.
[7] National Science Foundation, "Parallel and Distributed Computing Research Archive," par.nsf.gov, 2025.
[8] NSC Polteksby, "Cloud Computing Architecture and Reliability," nscpolteksby.ac.id, 2025.
[9] Redis Labs, "Adaptive Caching and Fault Resilience," redis.io, 2025.
[10] GeeksforGeeks, "Topological Sort and Graph Dependency Resolution," geeksforgeeks.org, 2025.
[11] Get S D E Ready, "Union-Find and Connectivity Problems in System Design," getsdeready.com, 2025.
[12] Scaler, "Understanding Topological Sort and Dependency Resolution," scaler.com, 2025.
[13] Towards Data Science, "Dynamic Programming Patterns and Optimization," towardsdatascience.com, 2025.