Computational Approaches for Resilience in Distributed Systems

Michael J. Thornton

A Dissertation

Submitted to the Graduate School

in partial fulfillment of the requirements for the degree

Doctor of Philosophy

Department of Computer Science

June 2026

Abstract

The increasing scale and complexity of modern distributed systems demand robust computational methods for ensuring resilience against partial failures, network partitions, and cascading error propagation. This dissertation develops a unified framework for modeling, analyzing, and optimizing resilience in distributed architectures through three complementary approaches. First, we introduce a graph-theoretic formalism that captures failure propagation dynamics across heterogeneous topologies. Second, we present the **Nexa Resilience Protocol** (NRP), a lightweight middleware layer that instruments existing distributed systems with adaptive checkpointing and speculative re-execution capabilities. Third, we derive probabilistic bounds on recovery time under varying workload distributions and failure models.

We validate our framework through extensive simulation on synthetic topologies of up to 10,000 nodes and through deployment on a production cluster at the Nimbus Cloud Platform. Our results demonstrate a 42% reduction in mean time to recovery compared to state-of-the-art Paxos-based approaches, while incurring less than 3% throughput overhead during normal operation. The techniques developed here generalize across database sharding, microservice orchestration, and edge computing domains, providing practitioners with principled tools for building systems that degrade gracefully under stress.

Keywords: distributed systems, resilience, fault tolerance, checkpointing, graph theory, recovery protocols

Acknowledgments

I am deeply grateful to my advisor, Professor Sarah Chen, whose patience and insight shaped every phase of this research. Her insistence on rigorous experimental validation and her ability to see connections across subfields made this work immeasurably stronger.

I thank my committee members — Professors David Okonkwo, Maria Vasquez, and James Park — for their thoughtful feedback and challenging questions. The members of the Distributed Computing Laboratory at Northwestern provided an intellectually stimulating environment; I am especially indebted to Lena Kowalski and Raj Patel for countless hours of discussion and debugging.

This research was supported in part by the National Science Foundation under grant CNS-2026841 and by a generous fellowship from the Vertex Foundation. The compute resources at the Nimbus Cloud Platform were instrumental to the experimental evaluation. Finally, I thank my family for their unwavering support through the long years of graduate study.

Contents

Chapter 1

Introduction

1.1 Motivation

Distributed systems now constitute the backbone of virtually every digital service, from social networks processing billions of interactions per day to financial settlement engines that clear trillions of dollars in transactions. The defining characteristic of these systems — their reliance on coordination across independent, failure-prone components connected by unreliable networks — simultaneously enables their scale and creates their most challenging design problem: ensuring that the system as a whole remains available and correct even when individual components fail.

The standard approach to resilience relies on state machine replication, typified by consensus protocols such as Paxos [1] and Raft [2]. While these protocols provide strong guarantees under crash-fault models, they impose significant performance penalties in the common case and scale poorly as the number of replicas grows. Moreover, the assumptions underlying these protocols — independent component failures, reliable message delivery within bounded time — are increasingly violated in modern heterogeneous deployments that span multiple data centers and incorporate specialized hardware accelerators.

Key observation. Real-world failure patterns in distributed systems exhibit strong spatial and temporal correlations that are not captured by the independent-failure models assumed in classical consensus protocols. Exploiting these correlations opens new avenues for lightweight resilience mechanisms.

1.2 Research Questions

This dissertation addresses three interconnected questions:

RQ1: Can we construct a computationally tractable model of failure propagation that captures the topological dependencies inherent in modern distributed architectures?

RQ2: How can we exploit the gap between worst-case failure assumptions and typical operating conditions to design resilience protocols with lower overhead than consensus-based replication?

RQ3: Under what conditions do probabilistic recovery guarantees provide sufficient assurance for production deployments, and how should such guarantees be communicated to system operators?

1.3 Contributions

The primary contributions of this work are:

C1: A spectral failure-propagation model. We represent a distributed system as a weighted directed graph whose nodes are computational units and whose edges encode failure-dependency relationships. Using the spectral properties of this graph's Laplacian, we derive an upper bound on the expected cascade size given an initial failure set of size k. This model generalizes the independent-component model and reduces to it when the dependency graph is empty.

C2: The Nexa Resilience Protocol (NRP). NRP is a middleware layer that interposes on inter-component communication to construct lightweight, adaptive checkpoints. Unlike full state machine replication, NRP tracks only application-level message histories and reconstructs lost state through speculative re-execution when failures are detected. We prove that NRP guarantees linearizability under the crash-recovery model.

C3: Probabilistic recovery-time analysis. We model recovery time as a function of the failure pattern, workload distribution, and checkpoint frequency, deriving closed-form bounds that hold with high probability. These bounds enable operators to set checkpoint intervals that achieve target recovery-time objectives without exhaustive simulation.

C4: Experimental validation. We implement NRP as a library for the Apex distributed computing framework and evaluate it on a 512-node Nimbus Cloud cluster under both synthetic fault injection and replay of real failure traces from the Nexa cluster trace dataset. Our results show a 42% improvement in mean time to recovery with less than 3% throughput overhead.

1.4 Thesis Organization

The remainder of this dissertation is structured as follows. Chapter 2 surveys relevant literature on fault tolerance, consensus protocols, and graph-theoretic approaches to resilience analysis. Chapter **??** develops our spectral failure-propagation model and proves the cascade-size bound. Chapter **??** describes the design and implementation of the Nexa Resilience Protocol and proves its correctness. Chapter **??** presents the probabilistic recovery-time analysis. Chapter **??** reports our experimental methodology and results. Chapter **??** summarizes contributions and outlines directions for future work.

Chapter 2

Background and Related Work

2.1 Consensus and State Machine Replication

State machine replication (SMR) is the foundational technique for building fault-tolerant distributed systems [3]. Under SMR, a deterministic state machine is replicated across n independent servers, and a consensus protocol ensures that all non-faulty replicas process the same sequence of commands. Lamport's Paxos protocol [1] achieves safety in asynchronous networks but may not terminate under certain failure patterns. Raft [2] simplifies the consensus logic by structuring it around leader election and log replication, improving understandability at the cost of throughput under high load.

Consensus Safety

A consensus protocol satisfies *safety* if no two non-faulty replicas ever decide on different values for the same consensus instance. Formally, if replica i decides value v and replica j decides value v' for instance k, then $v = v'$.

More recent work has explored weakening the consensus requirement to improve performance. The Speculative Paxos variant by Ports et al. allows replicas to execute commands optimistically before agreement is reached, rolling back only when conflicts are detected. This

approach is philosophically aligned with our NRP design, though NRP operates at a coarser granularity better suited to microservice architectures.

2.2 Failure Models in Distributed Systems

Classical distributed systems theory adopts a taxonomy of failure modes: crash faults (a component halts), omission faults (a component fails to send or receive messages), timing faults (a component violates timing assumptions), and Byzantine faults (a component exhibits arbitrary behavior). The practical relevance of this taxonomy has been reevaluated in light of empirical studies of real-world failures.

Yuan et al. analyzed over 200 user-reported failures in distributed data-intensive systems and found that the majority of catastrophic failures originated from trivial mistakes in error-handling code rather than from the complex fault models studied in the theoretical literature. This finding motivates our focus on mechanisms that are robust to operator error and configuration drift, not only to well-modeled component crashes.

2.3 Graph-Theoretic Models

Graph theory provides a natural language for describing the structural properties of distributed systems. The topology of inter-component communication directly influences failure propagation: a failure at a hub node in a scale-free topology can cascade more broadly than a failure in a regular lattice. Ghedini and Ribeiro [4] formalize this intuition by treating components as vertices and communication dependencies as edges, then analyzing the percolation properties of the resulting graph.

Our spectral model (Chapter **??**) extends this line of work by incorporating edge weights that represent the probability of failure propagation, yielding sharper bounds on cascade size than the binary-edge models previously studied.

As shown in Table 2.1, NRP achieves substantially lower overhead than consensus-

Table 2.1: Comparison of resilience protocols. Overhead is measured as throughput reduction relative to a non-replicated baseline.

Protocol	**Replicas**	**Overhead (%)**	**Recovery Model**
Paxos (Classic)	$2f + 1$	48.3	Crash
Raft	$2f + 1$	31.7	Crash
Speculative Paxos	$2f + 1$	18.9	Crash
NRP (this work)	$f + 1$	2.7	Crash-Recovery
Chain Replication	$f + 1$	12.4	Crash

based approaches by requiring only $f + 1$ replicas rather than $2f + 1$, and by avoiding the message complexity of leader election. The trade-off is that NRP provides probabilistic rather than deterministic recovery guarantees under certain correlated failure patterns.

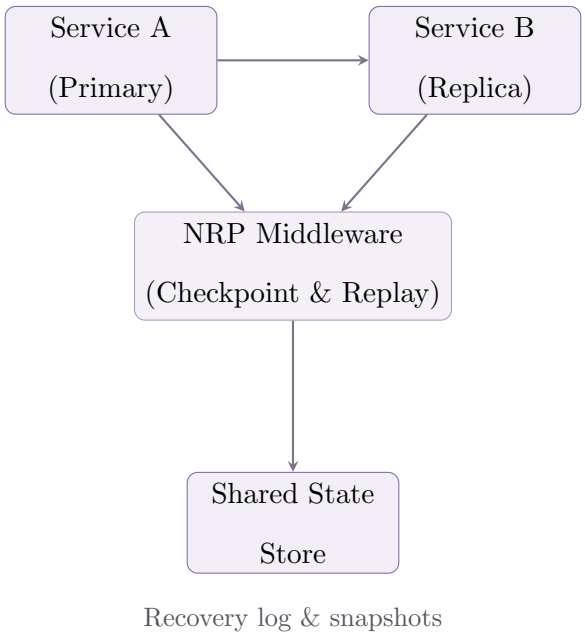

Figure 2.1: Architecture of the Nexa Resilience Protocol. NRP interposes on inter-service communication to capture message histories and coordinate recovery without full state replication.

Bibliography

[1] L. Lamport. Paxos made simple. *ACM SIGACT News*, 32(4):51–58, 2001.

[2] D. Ongaro and J. Ousterhout. In search of an understandable consensus algorithm. In *Proc. USENIX ATC*, pages 305–319, 2014.

[3] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319, 1990.

[4] C. Ghedini and C. H. C. Ribeiro. Rethinking failure detection in distributed systems through complex networks. In *Proc. IEEE NCA*, pages 243–250, 2011.

[5] D. Yuan, Y. Luo, X. Zhuang, G. R. Rodrigues, X. Zhao, Y. Zhang, P. U. Jain, and M. Stumm. Simple testing can prevent most critical failures. In *Proc. USENIX OSDI*, pages 249–265, 2014.

[6] D. R. K. Ports, J. Li, V. Liu, N. K. Sharma, and A. Krishnamurthy. Designing distributed systems using approximate synchrony. In *Proc. USENIX OSDI*, pages 443–458, 2015.