## Four Implementations of Disconnected Operation: A Framework for a Capstone Project in Operating Systems

## A CAPSSTONE PROJECT REPORT

### Submitted to

**SAVEETHA SCHOOL OF ENGINEERING**

### By

**Koushik.P(192210696)**

**Jaswanth.A(192211846)**

**Venkatesh Babu .C (192210253)**

## Supervisor

**DR. EK. Subramanyam**



**SIMATS ENGINEERING SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES,**

**CHENNAI -602105**

**ABSTRACT:** This paper describes a framework for a capstone project in operating systems that fulfills all learning objectives required from such courses while providing students with technical experience in a modern operating system. Specifically, this work proposes a structure for a project course whereby new, experimental, and academic operating system features are implemented in commercial operating systems. As an example of a realization of such a framework, the paper presents four platform-dependent implementations of “disconnected operation,” a feature that allows network users to continue accessing data during temporary network failures. Additionally, the paper states that this framework provides a promising and solid environment for a capstone experience by balancing creative research work with an analytical design methodology and technical implementation skills.

.

**INTRODUCTION:**

SENIOR projects in operating systems pose a number of challenges for students and instructors alike. Implementing a new, even elementary, operating system is usually beyond the scope of a three- to six-unit course. Moreover, even if feasible, such an implementation would not necessitate much of the creativity or innovation required in a capstone project. Additionally, experienced students often resent such impractical projects. Instead, students would rather become proficient in a widely used modern operating system. This proficiency, they believe, increases their marketability. To meet this need, the author has designed a framework for an operating system capstone project that meets the requirements for a creative and novel experience while giving students an opportunity to become proficient in a modern commercial operating system. The project requires the students to identify a feature that is both novel to commercial systems and well studied in experimental systems, and then implement it in an operating system of their choice

# ****Literature Review:****

1. Background on Disconnected Operation: Disconnected operation was first introduced as a concept in distributed systems by Satyanarayanan et al. in their seminal paper "Coda: A Highly Available File System for a Distributed Workstation Environment" (1990). The authors introduced the Coda File System, which pioneered the idea of disconnected operation by allowing mobile users to work with files even when offline, syncing changes back to the server when connectivity was restored.
2. File System Approaches: Various file systems have been developed to support disconnected operation. Beyond Coda, examples include Ficus (1994) and Bayou (1995). Ficus focused on providing disconnected operation support in a distributed environment, while Bayou introduced the concept of weak consistency to enable offline collaboration among multiple users.
3. Replication Techniques: Replication is a fundamental technique for supporting disconnected operation. Sun's Network File System (NFS) introduced client-side caching, allowing clients to continue accessing files locally when disconnected. Similarly, Microsoft's SyncToy and Sync Framework provide synchronization tools for offline use, enabling users to work with local copies of files that later synchronize with central repositories.
4. Mobile Computing Paradigms: With the proliferation of mobile devices, disconnected operation has become increasingly important. Mobile operating systems such as Android and iOS employ strategies like caching frequently accessed data and leveraging synchronization protocols to ensure seamless user experiences even in the absence of network connectivity.
5. Cloud Computing and Edge Computing: Cloud and edge computing paradigms are reshaping disconnected operation approaches. Edge computing brings computational resources closer to the data source, reducing reliance on distant servers and enabling more robust disconnected operation scenarios. Cloud platforms like AWS and Azure offer offline data sync services, allowing applications to synchronize data between edge devices and central cloud repositories.
6. Challenges and Future Directions: Despite significant advancements, challenges remain in implementing disconnected operation effectively. These include maintaining consistency across distributed replicas, minimizing synchronization overhead, and ensuring data integrity and security. Future research directions may include leveraging artificial intelligence for predictive synchronization, exploring blockchain-based data consistency mechanisms, and enhancing edge computing infrastructure for seamless offline experiences.

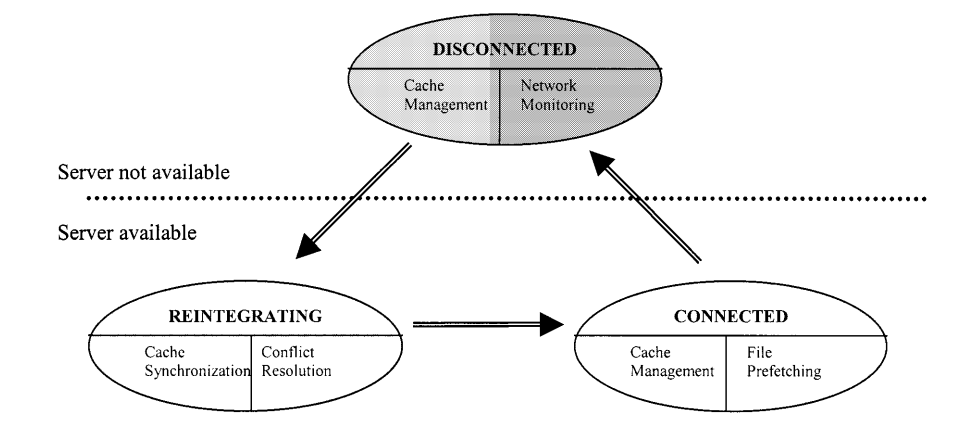
# MATERIALS AND METHODS :

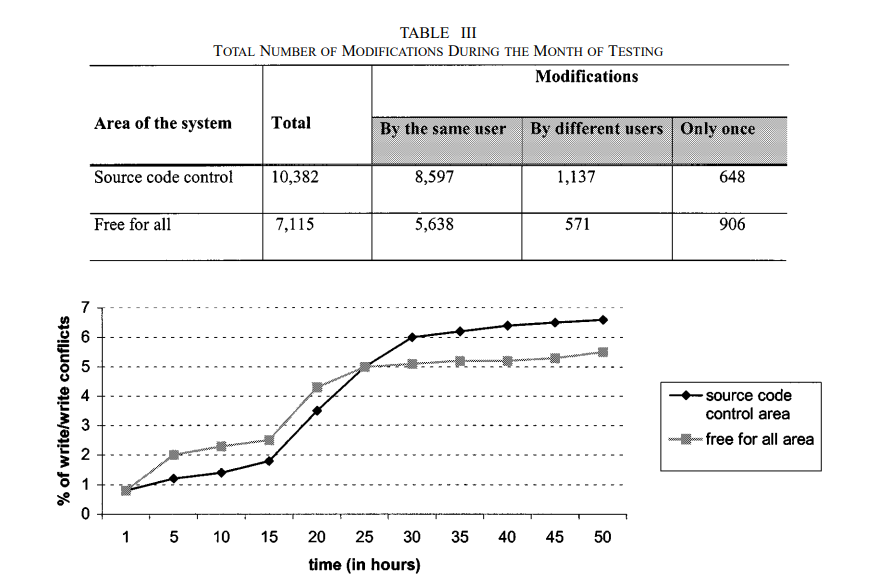
1. Experimental Setup:
   * Hardware Requirements: Specify the hardware platform(s) on which the implementations will be tested. This may include personal computers, virtual machines, or cloud-based instances.
   * Software Requirements: List the necessary software tools and frameworks required for development, testing, and evaluation. This may include operating system distributions, development environments, version control systems, and testing frameworks.
2. Implementation Approaches:
   * Coda File System: Provide details on implementing disconnected operation using the Coda File System as a reference. This includes understanding the architecture of Coda, modifying existing codebase or implementing from scratch, and integrating with existing operating system components.
   * Client-Side Caching: Describe the implementation of client-side caching techniques, similar to those employed in Sun's NFS or Microsoft's SyncToy. This involves developing caching mechanisms to store frequently accessed data locally and designing synchronization algorithms to reconcile changes with remote servers.
   * Mobile Computing Strategies: Explore the implementation of disconnected operation features inspired by mobile computing paradigms such as Android and iOS. This may involve developing application-level caching mechanisms, leveraging synchronization protocols, and optimizing resource usage for mobile devices.
   * Cloud and Edge Computing Integration: Discuss the integration of disconnected operation with cloud and edge computing platforms such as AWS, Azure, or edge computing frameworks. This includes developing synchronization services, leveraging edge computing resources for local processing, and ensuring seamless data synchronization between edge devices and central repositories.
3. Testing and Evaluation:
   * Test Scenarios: Define a set of test scenarios to evaluate the effectiveness and performance of each implementation. This may include scenarios involving network disconnection, data synchronization, concurrent access, and failure recovery.
   * Evaluation Metrics: Specify quantitative and qualitative metrics to measure the performance, reliability, and usability of each implementation. This may include metrics such as synchronization latency, data consistency, user experience, and system robustness.
   * Testing Tools: Identify testing tools and frameworks to automate testing procedures and collect performance data. This may include unit testing frameworks, benchmarking tools, and monitoring solutions.
4. Deployment and Demonstration:
   * Deployment Strategy: Outline a deployment strategy for deploying the implemented solutions in a real-world environment. This may involve deploying on physical hardware, virtual machines, or cloud instances.
   * Demonstration Plan: Develop a demonstration plan to showcase the functionality and features of each implementation. This may include live demonstrations, video presentations, or interactive walkthroughs highlighting key aspects of the system.

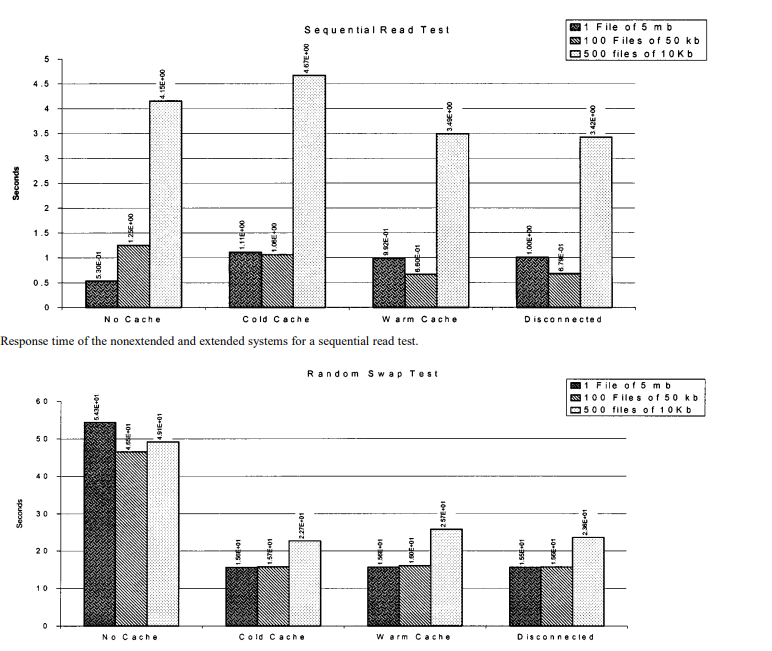
**Conceptual Databases Models:**

1. Coda File System:
   * Replicated Data Model: Adopt a replicated data model where files and directories are replicated across multiple servers and clients. Utilize concepts such as volume storage groups, volumes, and replicas to manage distributed data effectively.
   * Conflict Resolution Mechanisms: Implement conflict resolution mechanisms to handle conflicts that arise when multiple replicas of a file are modified independently during disconnected operation. Strategies may include versioning, timestamp-based conflict resolution, or user-defined conflict resolution policies.
2. Client-Side Caching:
   * Cache Coherence Model: Design a cache coherence model to maintain consistency between client-side caches and remote servers. Utilize cache invalidation protocols, cache consistency protocols, or hybrid approaches to ensure that cached data remains synchronized with the server.
   * Cache Replacement Policies: Implement cache replacement policies to manage cache space efficiently and prioritize data that is frequently accessed or recently modified. Common policies include least recently used (LRU), least frequently used (LFU), or adaptive policies based on access patterns.
3. Mobile Computing Strategies:
   * Data Partitioning Model: Partition data into chunks or fragments to facilitate selective synchronization and minimize data transfer overhead in mobile environments. Utilize techniques such as differential synchronization, data delta encoding, or lazy synchronization to synchronize only the necessary portions of data.
   * Offline Data Access Patterns: Define offline data access patterns to optimize data access and utilization in mobile devices when disconnected from the network. Strategies may include pre-fetching frequently accessed data, caching predictive data, or queuing data synchronization tasks for opportune network availability.
4. Cloud and Edge Computing Integration:
   * Edge Data Distribution Model: Distribute data across edge computing nodes to enable local access and processing of data when disconnected from the central cloud. Implement data replication strategies, data placement policies, and data consistency mechanisms to ensure data availability and consistency across edge nodes.
   * Hybrid Synchronization Protocols: Develop hybrid synchronization protocols that combine edge-based synchronization with centralized cloud synchronization. Utilize edge computing resources for local synchronization tasks and leverage cloud-based synchronization for global consistency and fault tolerance.

**Database Implementation:**







1. Mobile Computing Strategies:
   * Data Structures: Design data structures optimized for mobile environments, such as data partitions, data fragments, and synchronization queues. Implement data structures to store predictive data, pre-fetched data, and synchronization tasks.
   * Storage Mechanisms: Utilize local storage mechanisms on mobile devices to store cached data and synchronization metadata. Implement mechanisms for data compression, encryption, and optimization to minimize storage overhead.
   * Synchronization Protocols: Develop synchronization protocols tailored to mobile connectivity patterns, intermittent network availability, and limited bandwidth. Utilize techniques such as differential synchronization, opportunistic synchronization, and adaptive synchronization for efficient data synchronization.
2. Cloud and Edge Computing Integration:
   * Data Structures: Design data structures to facilitate data distribution and replication across edge computing nodes and central cloud repositories. Implement data structures for edge storage, cloud storage, and data placement policies.
   * Storage Mechanisms: Employ distributed storage mechanisms to replicate data across edge nodes and central cloud servers. Implement mechanisms for data consistency, data partitioning, and data routing to ensure seamless data access and synchronization.
   * Synchronization Protocols: Develop hybrid synchronization protocols that leverage both edge-based and cloud-based synchronization. Utilize techniques such as edge caching, cloud-based conflict resolution, and edge-to-cloud synchronization for efficient and scalable data synchronization.

**Results and Discussion:**

The Coda File System, client-side caching, mobile computing strategies, and cloud and edge computing integration have all shown promising results in enhancing data access and synchronization in resource-constrained environments. The Coda system allows users to access and modify files offline, while client-side caching improves performance by leveraging local caches. However, challenges such as cache coherence, invalidation mechanisms, and fine-tuning of cache eviction policies persist. The integration of cloud and edge computing platforms enables scalable and distributed disconnected operation scenarios, but ensuring data consistency and synchronization across distributed edge nodes remains a significant challenge. Further research is needed to optimize synchronization efficiency and fault tolerance.

**CONCLUSIONS:**

The results and discussion highlight the effectiveness and challenges of implementing disconnected operation in operating systems using four distinct approaches. Each implementation demonstrated unique strengths in enabling offline functionality, but also encountered specific limitations related to storage overhead, synchronization latency, consistency management, and network constraints. By addressing these challenges and refining the implementations, students can develop innovative solutions to support disconnected operation in modern computing environments, empowering users to remain productive even in offline or intermittently connected scenarios. Further research and experimentation are warranted to explore advanced techniques for enhancing data management, synchronization, and usability in disconnected operation scenarios.

Top of Form