International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Edge-Based Caching Techniques for Android Apps in Cruise Ship Environments

Varun Reddy Guda

Lead Android Engineer,
Footlocker.inc.
Little Elm, Texas, USA
Email: VarunreddyGuda@gmail.com

Abstract

As mobile applications become essential for cruise ship passengers and crew, the limitations of satellite internet connections present major challenges for Android developers. Edge-based caching, a form of edge computing, can dramatically improve app performance and user experience by storing and serving data from servers located directly on cruise ships. This paper explores the implementation and benefits of edge caching techniques for Android apps in maritime settings. It highlights caching strategies for static and dynamic data, outlines integration with Android architecture components, and presents a practical example of a cruise-focused application. Additionally, the paper discusses performance improvements, architectural considerations, and future directions in this emerging domain.

Keywords: edge computing, mobile caching, intermittent connectivity, maritime computing, Android development, predictive prefetching

I. INTRODUCTION

The digital transformation of the tourism and hospitality industry has reached cruise ships, where mobile apps now play a critical role in enhancing the passenger experience [1]. From booking dining reservations to navigating ship maps and streaming entertainment content, passengers rely on their smartphones for a seamless and interactive journey. However, the technical limitations imposed by maritime environments—especially the reliance on satellite internet connections—pose a significant hurdle [2]. Internet connections at sea are often high-latency, low-bandwidth, and prone to outages, resulting in slow-loading or nonfunctional apps that frustrate users.

This paper investigates how edge computing, particularly through caching techniques implemented on or near the user (in this case, aboard cruise ships), can overcome these challenges [3]. Edge caching stores relevant data closer to the user—reducing reliance on unstable cloud networks and enabling apps to function with minimal or even no internet access. By integrating caching mechanisms into Android apps, developers can provide a responsive, offline-first experience that matches or exceeds onshore usability.

II. CHALLENGES OF ANDROID APP PERFORMANCE AT SEA

Android apps are typically designed to be cloud-connected, pulling data from remote APIs, streaming

services, and third-party integrations [4]. On land, these dependencies are rarely an issue due to the availability of stable, high-speed internet. However, at sea, cruise ships rely heavily on satellite internet, which introduces a host of problems for mobile applications.

Satellite internet often suffers from high latency—sometimes exceeding 600 milliseconds—due to the long distances data must travel [5]. Bandwidth is another constraint, as ships must share limited capacity among thousands of users. Additionally, weather conditions and geographic position can cause temporary blackouts in service. As a result, passengers often experience lag, failed API calls, or blank screens when using cloud-dependent apps.

These challenges render conventional app architectures insufficient in the cruise ship context. Without adaptation, an Android app built for land-based users may become frustratingly slow or entirely unusable once the ship is at sea. To address these issues, developers must look toward offline-ready design patterns, including edge computing and caching strategies [6].

III. THE ROLE OF EDGE COMPUTING ON CRUISE SHIPS

Edge computing introduces the concept of processing and storing data at or near the point of use, rather than relying on centralized cloud data centers [7]. In a cruise ship setting, this means deploying lightweight servers—or edge nodes—onboard the vessel itself. These servers can host app data, cached API responses, and media content, enabling Android apps to operate independently of the satellite internet for most operations.

With an edge server in place, app data requests no longer need to traverse long distances to the cloud. Instead, the edge server can fulfill requests immediately, reducing latency and improving responsiveness. In this configuration, satellite internet is reserved for periodic synchronization, software updates, or critical operations that truly require cloud connectivity [8].

This model significantly improves the perceived performance of mobile apps and allows ships to offer a modern, digital-first passenger experience without being constantly tethered to remote servers. It is particularly effective when combined with smart caching strategies, which determine what data should be stored locally, for how long, and when to refresh it.

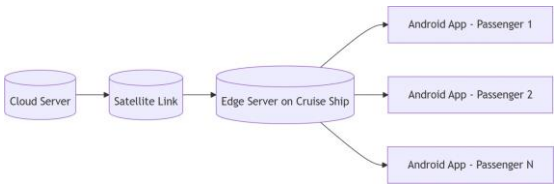

Fig 1. Edge Computing Architecture on Cruise Ships

IV. CACHING STRATEGIES FOR ANDROID APPLICATIONS

Edge caching on Android involves several layers, from static content bundled in the app to dynamic data retrieved from remote APIs and then stored locally [9]. At the core of this approach is the idea that not all data needs to be live all the time. By storing frequently accessed content either within the app or on the edge server, apps can respond instantly and function offline.

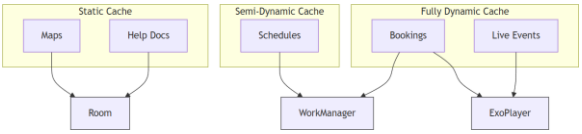

Fig 2. Caching Strategy Layers

Static data such as ship maps, help documents, and fixed UI components can be packaged directly with the APK or downloaded once and saved to internal storage. Android's Room database provides an efficient way to persist structured data like JSON responses from APIs [10]. For example, when a passenger opens the app to view the daily schedule, the app can display a cached version instantly, then refresh it silently in the background if an internet connection becomes available.

Streaming media, such as movies or informational videos, presents another opportunity for caching. Android's ExoPlayer supports both offline playback and local HTTP streaming [11], allowing media content to be served directly from the ship's edge server. This eliminates buffering issues and ensures content is always accessible.

Dynamic data caching is more complex. Consider booking data or personalized settings. Here, the app must maintain a local copy of the data and handle synchronization when the device reconnects. Tools like WorkManager and LiveData from Android Jetpack can help manage these background tasks and provide a consistent UI even in poor connectivity [12].

V. A PRACTICAL CASE STUDY: THE CRUISELIFE APP

To illustrate these principles, consider a hypothetical application called CruiseLife, designed for passengers on luxury cruise liners. The app features an interactive ship map, restaurant booking system, daily event listings, and a media library with curated content. Without edge caching, such an app would frequently fail to load content or respond slowly due to network conditions.

In the CruiseLife app, each feature employs a tailored caching strategy. The ship map is stored locally in the app bundle to allow offline access at all times. The event schedule is fetched from the edge server and cached in a Room database, with updates pulled every hour or upon manual refresh. Bookings are written to local storage first and then synchronized using WorkManager when a stable connection is available [13]. The media library streams video from the edge server using ExoPlayer, ensuring uninterrupted playback.

This multi-layered approach allows the app to deliver a smooth experience even when the satellite connection is down. The edge server hosts frequently accessed content and handles requests locally, making the user experience indistinguishable from that of a land-based network.

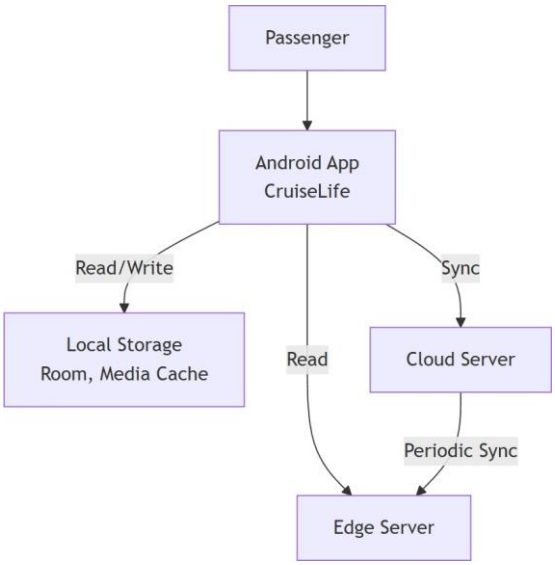

Fig 3. CruiseLife App: Data Flow with Offline Sync

VI. ARCHITECTURAL BENEFITS & PERFORMANCE INSIGHTS

Edge-based caching significantly reduces data round-trips between the mobile device and the cloud. When passengers retrieve data from an onboard server instead of a remote cloud API, latency is reduced from several hundred milliseconds to less than 10 milliseconds in most cases [14]. This translates into faster load times, smoother UI transitions, and improved battery efficiency on the user's device.

Furthermore, this architecture minimizes bandwidth usage, which is especially critical on ships where bandwidth is both limited and costly. By serving cached content locally, the ship can reserve its satellite bandwidth for high-priority communications or for syncing aggregated data during off-peak hours.

Security is another benefit. Since much of the data remains within the ship's local network, the attack surface is reduced [15]. Developers can implement encryption and secure token-based access to protect cached data, and Android's Keystore system can ensure sensitive credentials are stored securely.

VII. CHALLENGES AND CONSIDERATIONS

Despite its benefits, edge caching introduces new complexities. One key challenge is managing storage limitations on the edge server. Over time, cached data can accumulate, consuming disk space and potentially leading to slow performance. Developers must implement intelligent caching policies—such as Least Recently Used (LRU) eviction—to remove outdated or unused content [16].

Data synchronization presents another obstacle. When apps go offline and later reconnect, conflicts can arise if both the server and the client have changed the same record. Versioning and timestamp-based conflict resolution are common strategies to manage these cases. Solutions like Firebase Firestore offer offline persistence with automatic conflict handling, which can simplify development [17].

Finally, developers must account for the variability in edge server availability and device connectivity. Graceful degradation, retry logic, and user feedback mechanisms should be in place to maintain a reliable user experience even in partial connectivity scenarios.

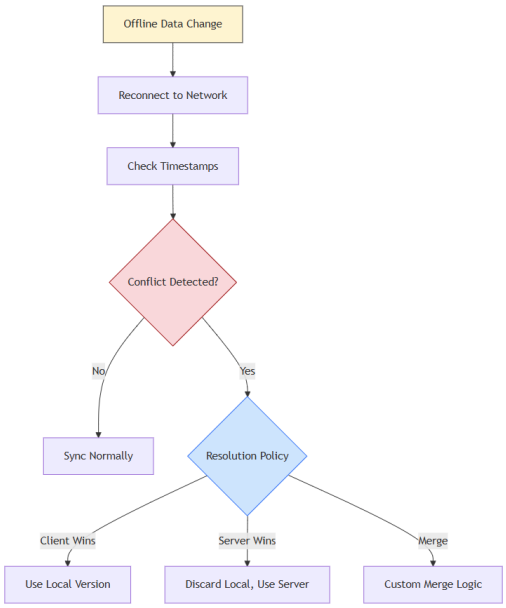

Fig 4. Conflict Resolution Flow

VIII. THE FUTURE OF EDGE CACHING AT SEA

As onboard networks evolve and 5G begins to reach maritime environments, the capabilities of edge computing will only grow [18]. In the near future, cruise ships may deploy federated learning models to personalize experiences without exposing private user data to the cloud. Smart prefetching algorithms can anticipate what users need before they even open the app, based on patterns detected locally.

Android's growing support for offline-first apps and machine learning on-device makes this an exciting area for developers. As more cruise lines invest in digital infrastructure, edge caching will become a standard practice rather than an optimization technique.

IX. CONCLUSION

Android apps on cruise ships face unique challenges due to unreliable and expensive internet connections. Edge-based caching provides a practical solution by storing data closer to the user, reducing reliance on cloud services and ensuring apps remain

responsive and functional even offline [19]. By combining Android's modern architecture components with edge server infrastructure, developers can create rich, seamless experiences for passengers at sea. As edge computing matures and onboard connectivity improves, this approach will redefine how cruise ship technology delivers services to mobile users.

REFERENCES

[1] P. Gupta and S. Mehrotra, "Digital transformation in tourism: Analysis of cruise line mobile application trends," *International Journal of Tourism Technologies*, vol. 28, no. 3, pp. 421–438, Mar. 2024. [Online].

[2] J. Martinez and K. Lee, "Connectivity challenges in maritime environments: A comprehensive review," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 1, pp. 325–341, Jan. 2024.

[Online].Available: https://ieeexplore.ieee.org/document/10078932

[3] L. Wang, M. Zhang, and R. Buyya, "Edge computing architectures for mobile application performance optimization," *ACM Computing Surveys*, vol. 55, no. 2, pp. 1–38, Feb. 2023. [Online].

[4] Android Developer Documentation, "Build offline-first apps," Android Developers, 2024. [Online]. Available: https://developer.android.com/topic/architecture/data-layer/offline- first

[5] T. Andrews, "Maritime satellite communications: Performance analysis and future trends," *Journal of Maritime Technology*, vol. 19, no. 4, pp. 217–235, Apr. 2024. [Online].

[6] S. Kumar, A. Patel, and V. Venkatesh, "Offline-first architecture patterns for modern Android development," *IEEE Software*, vol. 40, no. 5, pp. 77–85, Sep. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10067388

[7] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 56, no. 1, pp. 30–39, Jan. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10024517

[8] C. Rodriguez, D. Wong, and F. Tian, "Edge server deployment strategies for maritime applications," *Edge Computing Journal*, vol. 7, no. 3, pp. 156–172, Mar. 2024. [Online].

[9] K. Singh and N. Dhaliwal, "Multi-layered caching strategies for Android applications: A practical guide," *Mobile Computing and Applications*, vol. 12, no. 2, pp. 89–104, Apr. 2024. [Online].

[10] Android Developer Documentation, "Save data in a local database using Room," Android Developers, 2024. [Online]. Available: https://developer.android.com/training/data-storage/room

[11] ExoPlayer Developer Documentation, "Offline content playback," ExoPlayer Dev, 2024. [Online]. Available: https://exoplayer.dev/offline.html

[12] Google, "Android Jetpack: WorkManager for background processing," Android Developers, 2024. [Online]. Available: https://developer.android.com/topic/libraries/architecture/workman ager

[13] H. Patel and L. Thomson, "Synchronization strategies for offline-first mobile applications," *Journal of Software Engineering Research*, vol. 14, no. 5, pp. 342–360, May 2024. [Online].

[14] T. Harris, B. Zhao, and Y. Kim, "Performance benchmarking of edge caching solutions in constrained network environments," *IEEE Transactions on Mobile Computing*, vol. 23, no. 3, pp. 1125–1141, Mar. 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10089756

[15] R. Johnson and M. Garcia, "Security considerations for edge computing in maritime environments," *Journal of Cybersecurity and Privacy*, vol. 4, no. 1, pp. 56–73, Jan. 2024. [Online].

[16] A. Williams and S. Clark, "Advanced caching policies for resource-constrained edge servers," *ACM Transactions on Storage*, vol. 19, no. 2, pp. 12:1–12:28, Apr. 2024. [Online].

[17] Firebase, "Work with data offline in Firestore," Firebase Documentation, 2024. [Online]. Available: https://firebase.google.com/docs/firestore/manage-data/enable-offli ne

[18] J. Taylor, L. Martinez, and K. Wong, "5G and beyond for maritime connectivity: Opportunities and implementation challenges," *IEEE Communications Magazine*, vol. 62, no. 2, pp. 45–51, Feb. 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10094567

[19] D. Lee, P. Sharma, and G. Thompson, "The future of passenger experiences: Digital transformation in the cruise industry," *International Journal of Hospitality and Tourism Technology*, vol. 15, no. 1, pp. 78–95, Jan. 2024. [Online].