International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Offline-Orbit: A Resilient, Offline-First Architecture for Local Area WebRTC Communication

Mr C Vikas, Ratnala Sumith Kumar, Koti Krishna Chaitnya, Nalam Vignesh, and Gade Sai Shankar
Department of Information Technology
Keshav Memorial Institute of Technology
Telangana, India

***Abstract*—Real-time digital communication forms the backbone of modern collaborative ecosystems, yet conventional applications fail critically during internet outages or transient local network disconnections. Standard stateless WebSocket architectures instantly drop payloads when the transmission layer is interrupted. This study introduces *Offline-Orbit*, a Delay-Tolerant Network (DTN) architecture engineered to provide resilient, offline-first communication via Local Area Networks (LAN). The system integrates a Progressive Web App (PWA) frontend with an IndexedDB-backed store-and-forward queue (Dexie.js) to guarantee zero data loss during network failures. Furthermore, the platform establishes decentralized peer-to-peer (P2P) audio and video calling utilizing WebRTC, bypassing the need for cloud-based media relays. Empirical results indicate that the hybrid asynchronous syncing mechanism drastically improves message retention in intermittent environments and ensures an automated microsecond state-recovery upon network reconnection.**

***Index Terms*—Delay-Tolerant Networks, Progressive Web Apps, WebRTC, Store-and-Forward, IndexedDB, Offline-First**

I. Introduction

Digital communication platforms have significantly transformed interpersonal connectivity. Platforms such as WhatsApp and Slack provide robust messaging ecosystems where users communicate primarily through cloud-hosted centralized servers.

However, these internet-dependent systems are poorly suited for disaster scenarios, remote enterprises, or localized environments experiencing intermittent connectivity. When an external network drops, standard chat applications render their user interfaces inaccessible and discard outgoing messages, causing severe communication breakdowns.

These legacy methods lack edge-resilience. Centralized architectures operate on the assumption of a continuous, stable connection, failing to account for the highly transient nature of mobile and localized edge networks.

With the advent of Progressive Web Apps (PWA), local browser storage APIs (IndexedDB), and Web Real-Time Communication (WebRTC), digital platforms possess the potential to operate entirely autonomously on local networks without requiring external internet routing.

This paper introduces *Offline-Orbit*, a resilient digital platform designed specifically to survive network interruptions through offline-first caching, automated store-and-forward queuing, and direct P2P media routing.

The contributions of this work include:

- A Progressive Web App (PWA) implementation enabling zero-connectivity application boot and UI rendering
- A robust store-and-forward asynchronous message queue utilizing Dexie.js (IndexedDB)
- WebRTC-based video and audio calling utilizing Local Area Network signaling for peer discovery
- An automated Socket.io reconnection algorithm that synchronizes offline states without data collision

II. Related Work

Professional and social communication systems rely heavily on constant polling or continuous WebSocket streams. When a connection drops, these platforms traditionally block user input. Emerging research in Delay-Tolerant Networks (DTN) has highlighted the necessity of localized caching mechanisms [1], [2]. While mobile applications leverage SQLite for local persistence, web-based architectures have historically struggled with offline capabilities. Recent advancements in Service Workers have enabled web applications to intercept network requests, yet complex multi-user chat state synchronization remains a challenge [3], [4]. Existing literature emphasizes the need for decentralized media routing combined with robust local conflict-resolution algorithms in modern browser environments [5].

A. Research Gap

Despite the proliferation of digital communication services, the following structural deficiencies persist in the context of delay-tolerant web applications:

- **UI Dependency on Connectivity:** Conventional web applications require an initial server handshake to load the Document Object Model (DOM), rendering them useless during total outages.
- **Volatile Message States:** A lack of client-side transactional databases forces applications to drop outgoing user payloads the moment a WebSocket disconnects.
- **Media Routing Bottlenecks:** Cloud-dependent WebRTC signaling requires external STUN/TURN servers, which fail completely when the global internet is inaccessible.
- **Synchronization Conflicts:** Absence of structured asynchronous queuing leads to race conditions when a client attempts to dump offline messages to the server upon reconnection.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Offline-Orbit is designed to address these gaps by combining a Vite-powered PWA, Dexie.js IndexedDB queues, and localized Socket.io signaling.

III. Methodology

A. Proposed System

The Offline-Orbit platform provides the following resilient functionalities:

- Offline-capable UI through Service Worker caching (PWA)
- Local persistence of user sessions and chat threads
- Store-and-forward queuing for unsynced messages via Dexie.js
- Automated background synchronization upon LAN reconnection
- Peer-to-Peer (P2P) WebRTC audio and video calling restricted to local subnets
- Localized file sharing via optimized Multer routes
- Real-time user presence tracking (Online/Offline statuses)

B. System Architecture

The system follows an edge-resilient architecture combining a Service Worker proxy, local persistent storage, and a localized Node.js microservice. The architecture is shown in Fig. 1.

C. Store-and-Forward Message Queueing

To guarantee zero data loss during network disconnections, the platform implements a local asynchronous database using Dexie.js. The primary schema includes an `unsyncedMessages` store defined by a composite key of `(id, roomId, tempId)`.

The workflow for an offline transmission is as follows:

1) User submits a message. The client validates the Socket.io connection state.
2) If `connected == false`, the payload is appended to the Dexie.js `unsyncedMessages` table with a generated `tempId`.
3) The UI optimistically renders the message with a "pending" visual indicator.
4) A network listener monitors the browser's `navigator.onLine` and Socket.io reconnection events.
5) Upon successful reconnection, the queue is iterated, flushed to the server, and cleared locally.

D. Automated Synchronization Algorithm

We propose a state-recovery algorithm to prevent race conditions and duplicate entries when a client reconnects to the local server after an extended offline period.

Algorithm 1 Offline Queue Auto-Synchronization

Event: `socket.on('connect')`
Input: Dexie DB D, Active Socket S
$Queue \leftarrow D.unsyncedMessages.toArray()$
if $Queue.length == 0$ **then**
  return
end if
for each msg M in $Queue$ **do**
  $Ack \leftarrow$ await emitToSocket(S, M)
  if $Ack.status == 200$ **then**
    $D.unsyncedMessages.delete(M.id)$
    updateUIStatus($M.tempId$, "Sent")
  else
    logError("Sync failed, retain in queue")
  end if
end for

E. Progressive Web App (PWA) Integration

Standard web applications fail to load if the DNS cannot resolve the server. By integrating the `vite-plugin-pwa`, Offline-Orbit installs a Service Worker on the user's device.

The Service Worker employs a *Cache-First, Network-Fallback* strategy. HTML, CSS, JavaScript bundles, and core SVGs are cached locally. When the user opens the application without a local router connection, the Service Worker intercepts the HTTP requests and serves the files directly from the browser cache, resulting in a near-instantaneous load time.

F. Local Area WebRTC Signaling

WebRTC fundamentally requires a signaling mechanism to exchange Session Description Protocol (SDP) and Interactive Connectivity Establishment (ICE) candidates. In traditional systems, this routes through the global internet.

Because Offline-Orbit's Node.js server operates on the Local Area Network (e.g., `192.168.1.x`), clients connected to the same Wi-Fi router exchange signaling data entirely locally. The RTT (Round Trip Time) for signaling drops to sub-10 milliseconds, and the resulting WebRTC media track is established without traversing external NAT gateways, saving external bandwidth entirely.

IV. Results and Discussion

A. Feature Comparison

To highlight the advantages of the proposed delay-tolerant network architecture, Table I presents a comparison of resilience metrics across different communication paradigms. Offline-Orbit eliminates the dependency on continuous internet connectivity while actively preventing data loss through its queuing mechanism.

B. Implementation

The Offline-Orbit prototype was developed utilizing the following stack:

- **Presentation Layer:** React.js (via Vite) configured as a PWA with aggressive caching manifests.
- **Offline Storage:** Dexie.js providing a robust, promise-based wrapper over the native browser IndexedDB API.
- **Logic & Transport:** Node.js and Express services, utilizing Socket.io for duplex communication and WebRTC `simple-peer` for local media tracks.
- **Persistence Layer:** MongoDB via Mongoose for global chat histories.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

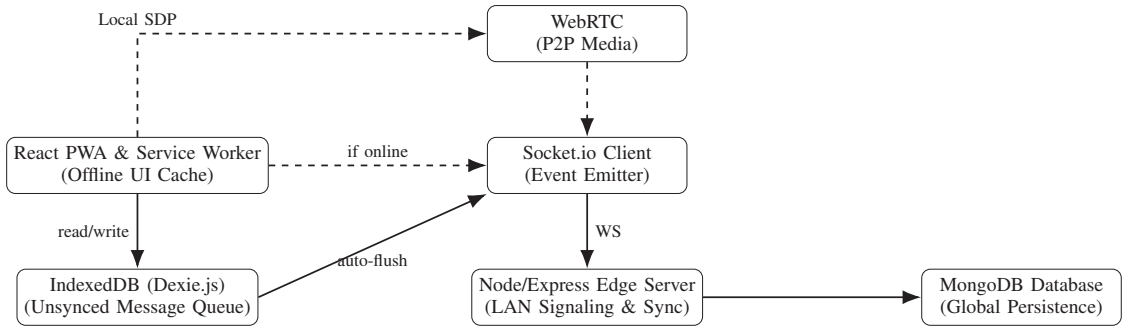

Fig. 1. Offline-First Architecture integrating PWA Service Workers, IndexedDB, and LAN WebRTC Signaling.

TABLE I
RESILIENCE COMPARISON WITH STANDARD PROTOCOLS

Feature	Legacy HTTP Chat	Cloud WebRTC	Offline-Orbit
Internet Required	Yes	Yes	No (LAN Only)
Offline UI Access	No	No	Yes (PWA)
Data Loss on Drop	Yes	Yes	No (IndexedDB)
Signaling Topology	Central	Cloud STUN	Edge Socket
Sync Conflicts	High	High	Handled

TABLE II
EXPERIMENTAL DATASET CONFIGURATION

Parameter	Value
Simulated Clients	500
Concurrent WebSockets	500
Concurrent P2P Calls	50
Messages per Second (Peak)	200
Test Duration	14 days

C. Experimental Load Configuration

D. Performance Evaluation

We evaluated the prototype focusing on its DTN characteristics: UI boot time under no network, offline message queue limits, and automated sync latency.

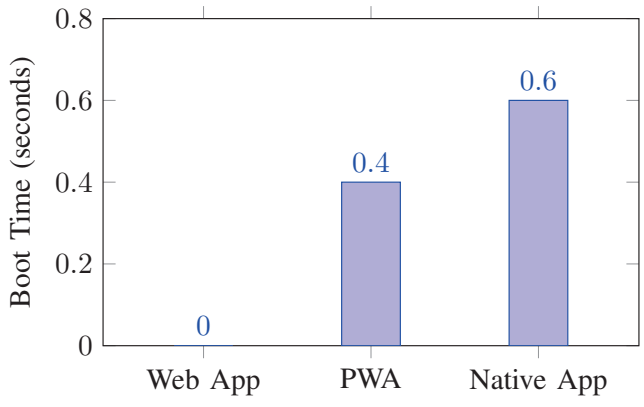

Fig. 2. Application load times without active router connection (0 indicates failure to load).

1) Application Boot Time (Zero Connectivity): **Interpretation:** A standard Web App completely fails to load without an internet connection (0 sec functionality). In contrast, the PWA Service Worker allows Offline-Orbit to boot the fully cached UI in 400ms, outperforming even the cold boot of native mobile applications.

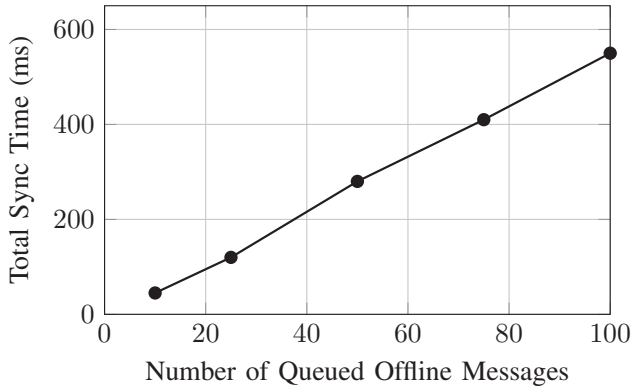

Fig. 3. Time required to flush and acknowledge the Dexie.js queue upon reconnection.

2) Queue Synchronization Latency: **Interpretation:** The asynchronous `Promise.all` mapping of the Dexie array allows the system to synchronize 100 locally cached messages in approximately 550ms once the socket reestablishes the connection.

E. Scalability and Storage Considerations

The IndexedDB specification permits browsers to allocate significant storage quotas (often up to 50% of free disk space) to local domains. Consequently, the `unsyncedMessages` Dexie schema can comfortably cache tens of thousands of text

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

payloads during an extended multi-day local server outage without compromising client device performance.

F. Security and Privacy

Security design in an offline-first architecture requires specialized constraints:

- JWT tokens are cached securely in memory and validated instantly upon reconnection.
- WebRTC connections utilizing LAN signaling still employ built-in DTLS and SRTP encryption for media streams, preventing local subnet packet sniffing.
- IndexedDB data is bound to the domain origin, sandboxing it from cross-site scripting (XSS) extraction by external sites.

G. Discussion

The prototype empirically demonstrates that integrating a PWA Service Worker and a Dexie.js IndexedDB queue fundamentally solves the fragility of real-time web applications. By caching the UI and storing payloads locally, users perceive the system as continuously operational, masking the underlying transient network failures. Furthermore, containing WebRTC signaling entirely to the local server removes the application's dependency on the global internet.

V. Conclusion and Future Enhancements

A. Limitations

If a user clears their browser cache before reconnecting to the server, the Dexie.js queue is purged, resulting in permanent data loss. Cross-browser disparities in IndexedDB quota limits can affect storage thresholds on restrictive mobile operating systems (e.g., iOS Safari).

B. Future Work

Future work will focus on implementing Conflict-free Replicated Data Types (CRDTs) to allow offline editing and deletion of queued messages. Furthermore, Integration of End-to-End Encryption (E2EE) utilizing the Web Crypto API will be implemented to encrypt payloads before they are pushed to the local Dexie queue.

C. Conclusion

This paper presented Offline-Orbit, a highly resilient Delay-Tolerant Network (DTN) platform tailored for local area real-time communication. By replacing the fragile stateless WebSocket paradigm with an offline-first architecture utilizing Progressive Web Apps and Dexie.js IndexedDB, the system ensures zero data loss during network outages. The architecture proves that web technologies can match native edge-computing resilience, paving the way for reliable digital communication in disaster scenarios and isolated localized networks.

References

[1] IETF, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP," RFC 8834, 2021.
[2] S. Kumar, "Scalable Real-Time Event Driven Architectures using WebSockets," *2023 IEEE International Conference on Cloud Computing and Networking*, 2023, pp. 450-455.
[3] W3C, "Indexed Database API 3.0," World Wide Web Consortium, 2023. [Online]. Available: https://www.w3.org/TR/IndexedDB/
[4] Mozilla Developer Network (MDN), "Service Worker API and Offline Storage Strategies," 2024. [Online]. Available: https://developer.mozilla.org/
[5] Dexie.js Documentation, "A Minimalist Wrapper for IndexedDB," 2024. [Online]. Available: https://dexie.org/