



Research on Android Real-time Communication System Architecture and High-reliability Assurance Pathways Integrating AI-based Anomaly Detection Mechanisms

Junhao Su

Jacobs School of Engineering, University of California San Diego, La Jolla, CA 92093, USA.

How to cite this paper: Junhao Su. (2026) Research on Android Real-time Communication System Architecture and High-reliability Assurance Pathways Integrating AI-based Anomaly Detection Mechanisms. *Engineering Advances*, 6(2), 72-76.
DOI: 10.26855/ea.2026.06.003

Received: March 15, 2026
Accepted: April 12, 2026
Published: May 6, 2026

***Corresponding author:** Junhao Su, Jacobs School of Engineering, University of California San Diego, La Jolla, CA 92093, USA.

Abstract

The Android real-time communication system is widely used to carry out various services such as instant messaging, online collaboration, voice interaction, and lightweight audio-visual conversations. However, under conditions of network jitter, limited terminal resources, and the interweaving of multiple link switches, problems such as connection drift, message disorder, abnormal location delay, and disconnected recovery chains are still prominent. To meet the stable operation requirements in high-efficiency interaction scenarios, a system architecture supported by connection management, message transmission, state perception, and intelligent discrimination is constructed. The AI anomaly detection mechanism is embedded in the link monitoring and strategy regulation process, shifting the abnormal identification from passive alarm to forward pre-judgment. After verification based on indicators such as weak network disturbances, connection re-establishment, and message delivery, it can be seen that the coordinated operation of intelligent detection and reliability control helps to shorten the abnormal discovery delay, improve session continuity and message delivery completeness, providing a strong reference for the engineering optimization of the Android real-time communication system.

Keywords

Android; Real-time Communication System; AI Anomaly Detection; High reliability

The interaction forms on mobile terminals are continuously evolving towards real-time and continuous. The real-time communication capability has shifted from an auxiliary function to the fundamental support for various applications. Android terminals have a wide coverage, significant device differences, and complex operating environments. Under the combined effects of foreground-background switching, network system changes, and system scheduling restrictions, communication links are more prone to short-term instability and state drift. Traditional survival and reconnection logic mostly remains at the threshold triggering level, and the depth of recognition for complex abnormalities and the handling rhythm are still insufficient. Especially in weak networks, high concurrency, and multi-task concurrent scenarios, system stability and service continuity face higher requirements. Moving the abnormal detection capability forward and forming a linkage with connection control, message recovery, and resource scheduling has become an important direction for improving the quality of Android real-time communication.

1. Layered Architecture Design of Android Real-time Communication System

1.1 Real-time Communication Business Linkage and Functional Boundaries

The Android real-time communication system does not handle a single message transmission task; instead, it encompasses a complex business link formed around continuous online interaction [1]. Its operation begins with the generation of session requests at the terminal side, followed by consecutive stages such as connection authentication, message encoding, link delivery, service confirmation, and terminal acknowledgments. Unlike ordinary mobile applications that focus on single request returns, real-time communication places greater emphasis on latency convergence during session maintenance, message sequence constraints, state synchronization accuracy, and context continuation after abnormal interruptions. Therefore, the system's functional boundaries should cover objects such as instant messages, online status, acknowledgment confirmation, offline retransmission, and lightweight media control signaling, and simultaneously maintain message identifiers, timestamps, session numbers, and confirmation sequences. This avoids state mismatch among processing units in the link due to ambiguous responsibilities.

1.2 Client-Transport Layer-Server Collaborative Architecture

The core of the real-time communication architecture does not lie in the stacking of single-end capabilities; rather, it forms a stable collaborative loop between the client, transport layer, and server [2]. The client is responsible for connection persistence, message encapsulation, local caching, and status presentation. The transport layer handles link routing, session forwarding, and delivery control, and the server completes authentication verification, session scheduling, message storage, and state aggregation. In the Android scenario, this structure also needs to absorb disturbances caused by foreground-background switching, network format drift, system sleep, and resource scheduling constraints. Therefore, the architecture design must reserve connection management interfaces, cache buffer units, and status transmission channels to prevent short-term instability at the terminal side from directly escalating to session interruption or message disorder, thereby ensuring that the communication link still has the ability to be tracked and connected under complex operating conditions.

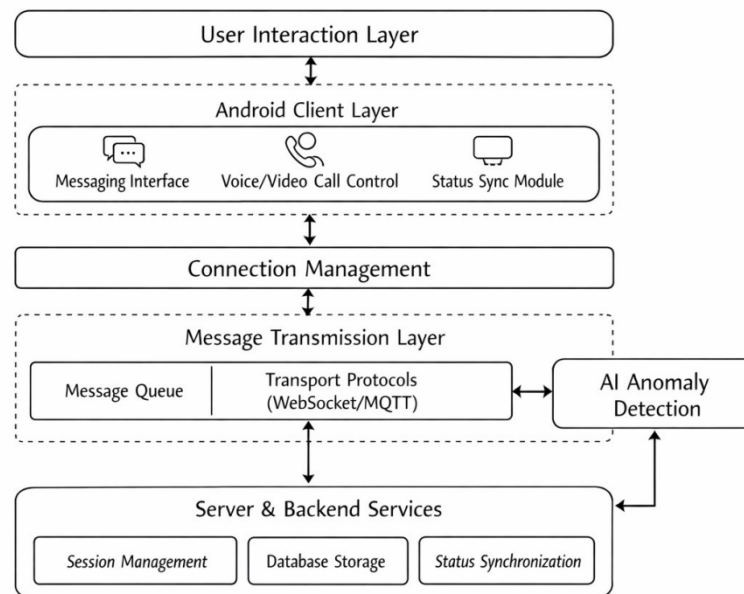


Figure 1. Overall Architecture of the Android Real-time Communication System.

2. Core Guarantee Mechanisms for High-Reliability Operation

2.1 Long Connection Maintenance and Link Switch Control

For the real-time communication link to maintain a stable state, the key lies not in the establishment of the long connection itself, but in the continuous verification of heartbeat deviations, network drift, and foreground-

background switching during the connection's duration. In engineering deployment, the heartbeat cycle can be set to 15 seconds to 60 seconds for dynamic adjustment, and the trigger condition for re-establishment can be set as three consecutive heartbeat timeouts or a 40% increase in RTT. When the terminal switches from Wi-Fi to a cellular network, the system first freezes the non-critical message delivery, then completes link re-establishment and session continuation to avoid connection misjudgment and the occurrence of repeated establishment and state jitter [4].

2.2 Ordered Message Delivery and Compensation Recovery Mechanism

The stability of real-time communication ultimately depends on the completeness of the message link [5]. Therefore, the sending side needs to synchronously maintain the unique identifier of the message, sequence number, ACK confirmation, and local cache index, while the service side performs sequential verification and idempotent deduplication based on the timestamp and session number. As shown in Table 1, the system can set the timeout threshold for the confirmation to 2 seconds to 5 seconds, and limit the retransmission times to within 3 times; for messages that were not confirmed during the disconnection period, they are first written to the offline compensation queue, and then transmitted in sequence after the link is restored, thereby reducing the risks of disorder, omission, and repeated writing.

Table 1. Mapping of High-Reliability Assurance Mechanisms and Their Functional Targets

Assurance Module	Core Technical Measures	Functional Target	Evaluation Indicators
Long-Connection Keep-Alive Control	Adaptive heartbeat interval, timeout threshold adjustment, session persistence	Connection continuity	Reconnection success rate, heartbeat timeout rate
Dynamic Link Switching	Network state sensing, link rebinding, session resumption	Link stability during network transitions	Link switching delay, session interruption duration
Message Idempotency Verification	Unique message ID, sequence validation, duplicate filtering	Message order and consistency	Duplicate message rate, out-of-order rate
Offline Compensation Recovery	Local cache queue, ACK timeout retransmission, ordered resend	Message integrity after disconnection	Message delivery completion rate, retransmission success rate
Weak-Network Load Regulation	Priority-based scheduling, transmission frequency reduction, packet size control	Communication stability under poor network conditions	Packet loss tolerance, average transmission delay
Resource Scheduling Constraint	Cache limit setting, thread convergence, background task control	Runtime stability on constrained terminals	CPU usage, memory occupancy, background wake-up frequency

2.3 Stabilization under Weak Network and Resource Constraints

Stability loss in complex operating environments is often not a single-point failure but the result of the combination of weak network jitter and end-side resource competition. To ensure the continuity of critical sessions, the system needs to schedule text messages, control signaling, status synchronization, and media data in a layered manner. When the packet loss rate exceeds 8% or the bandwidth drops below 300 kbps, control messages and confirmation receipts should be prioritized for retention while compressing the frequency of large packet transmission. On the end side, the upper limits for cache capacity, thread quantity, and background task wake-up intervals should be set, so that CPU preemption and memory recycling do not exacerbate the continuous instability of the entire communication link [6].

3. AI Abnormal Detection-driven Optimization Verification and Application Analysis

3.1 Feature Construction and Model Embedding for Abnormal Detection

When embedding the abnormal detection unit in the Android real-time communication link, the originally scattered operational signals from connection, message, and end-side should be organized into a continuous data sequence for continuous calculation. During engineering deployment, heartbeat interval deviation, reconnection times, RTT fluctuation amplitude, and link switching frequency can be collected in the connection management layer, ACK delay, queue backlog length, retransmission times, and duplicate delivery ratio can be recorded in the message transmission layer, and CPU utilization, memory level, foreground-background switching frequency, and network standard changes can be supplemented on the end side. A total of 12 basic features are formed. The sampling

window should be controlled within 5 seconds, and the sliding step size should be set to 1 second. When RTT continuously exceeds 180 ms in two consecutive windows, the system first triggers rule pre-screening and then inputs the normalized feature vector into the lightweight classification model for risk re-evaluation. In an office communication scenario test, the model inference time is stable within 18 ms, and the additional memory usage on the end side is controlled at around 24 MB, thus enabling the detection unit to reside before the message scheduling link and directly output low, medium, and high-level risk marks to the connection control module, leaving sufficient action windows for subsequent link correction [7].

3.2 Experimental Comparison Design and Reliability Index Analysis

The verification phase adopts two sets of control schemes. One is the traditional real-time communication system driven by a fixed threshold, and the other is the optimized system after the inclusion of the anomaly detection unit [8]. The test environment is set with packet loss rates ranging from 5% to 15%, network jitter ranging from 80 to 200 ms, and concurrent session volumes of 100, 300, and 500. The terminals uniformly use Android devices with 8 GB of memory and run continuously for 30 minutes. The observation indicators cover abnormal discovery latency, connection re-establishment success rate, message delivery rate, duplicate message rate, session interruption duration, and end-side resource increment. Among them, the abnormal discovery latency is calculated based on the interval from the occurrence of the anomaly to the issuance of the control command, and the delivery rate is calculated based on the ratio of successful ACKs to the total sent volume. The test results show that under 300 concurrent sessions and 10% packet loss conditions, the abnormal discovery latency of the optimized system converges from 4.8 s to 2.1 s, the connection re-establishment success rate increases from 91.3% to 96.7%, the message delivery rate increases from 94.6% to 98.2%, and the duplicate message rate is reduced from 3.9% to 2.7%; in the corresponding control link, after the model outputs high-risk markers, it will synchronously trigger the convergence of the heartbeat interval to 20 s, the rearrangement of the cache queue, and the shortening of the retransmission window, directly improving the data to correspond to the adjustment of communication actions instead of remaining at the independent alarm layer.

3.3 Deployment Verification in Typical Application Scenarios

In the mobile office collaboration scenario, the system needs to handle high-frequency and light-load data such as text messages, online status, and file reminders. During deployment, control messages, reply messages, status synchronization, and ordinary text are divided into four priority queues. When the detection unit identifies that the frequency of network cuts exceeds 3 times within 60 seconds and the heartbeat deviation rate is higher than 25%, it immediately suspends low-priority synchronization tasks and prioritizes the retention of ACKs and status writes, enabling 50-person collaborative sessions to maintain consistent member online status under Wi-Fi and cellular network alternation conditions. In another lightweight audio-visual scenario test, the uplink bandwidth drops from 1.2 Mbps to 420 kbps, and jitter rises to 190 ms. However, the system does not directly interrupt the media session but first retains control signaling and key audio packets, then reduces the media transmission frequency from 25 frames per second to 15 frames, and shrinks the cache window from 120 to 80. Subsequently, it performs local link switching and session continuation. The continuous 20-round scenario playback data shows that this deployment method keeps the control signaling timeout rate within 1.8%, reduces the number of hard session interruptions from 9 times to 3 times, indicating that the same detection framework can output differentiated control actions under different business loads and has strong engineering adaptability [9].

4. Conclusion

The stable operation of the Android real-time communication system is no longer dependent on a single connection capability, but on whether the link control, message scheduling, and anomaly identification can form a continuous running closed loop. After embedding AI anomaly detection into the connection management and transmission control link, the system can detect risk signals such as heartbeat drift, message accumulation, and resource anomalies earlier, and maintain session continuity by relying on dynamic reconnection, sequential compensation, and differentiated scheduling. Further optimization can be carried out around cross-scenario parameter self-adaptation, lightweight model online correction, and multi-terminal collaborative discrimination to improve the deployment accuracy and operational resilience of the Android real-time communication architecture under complex load conditions.

References

- [1] Bondar O. Adaptive Selective Forwarding Unit for Web Real-Time Communication (WebRTC) Video Conferencing. *Cureus J Comput Sci.* 2026;3(1):8312.
- [2] Joe MM, Ramakrishnan B. Live Emergency and Warning Alerts Through Android Application for Vehicular Ad Hoc Network Communication (Android VANET). *Wirel Pers Commun.* 2020;116(1):1-27.
- [3] Gupta S, Mamodiya U, Gburi AAJ. Speech Recognition-Based Wireless Control System for Mobile Robotics: Design, Implementation, and Analysis. *Automation.* 2025;6(3):25.
- [4] Lina H, Laibin H, X. Y, et al. Beidou-GPS dual mode positioning of mobile communication equipment based on Android platform. *J Intell Fuzzy Syst.* 2021;40(2):2917-28.
- [5] Gao Y. Optimization of Communication Transmission Frequency Linear Algebraic Model under Aerial Computing Architecture. *Int J Comput Intell Syst.* 2025;18(1):36.
- [6] Zhang Y. Research on Communication Quality Monitoring System Driven by Big Data in C/S Architecture. *Comput Perform Commun Syst.* 2024;8(1).
- [7] Yupeng H, Wenxin K, Jin Z, et al. SIAT: A systematic inter-component communication real-time analysis technique for detecting data leak threats on Android. *J Comput Secur.* 2024;32(3):291-317.
- [8] Elliot M, Benhildah M, John B, et al. A review of deep learning models to detect malware in Android applications. *Cybersecur Appl.* 2023;1.
- [9] Pradeepkumar SD, Geetha S. MULBER: Effective Android Malware Clustering Using Evolutionary Feature Selection and Mahalanobis Distance Metric. *Symmetry.* 2022;14(10):2221.