Scholar Publishing Group
International Journal of Neural Network
https://doi.org/10.38007/NN.2026.050109
ISSN 2790-2692 Vol. 5, Issue 1: 79-89

Design and Implementation of Android High Reliability Communication Module Based on Hierarchical Architecture

Junhao Su

Jacobs School of Engineering, University of California San Diego, La Jolla, CA, 92093, United States

Keywords: Android; VoIP; WebRTC; calling; layered architecture; high-reliability communication

Abstract: For real-time voice and video calling scenarios on mobile terminals, traditional Android communication modules often suffer from slow call setup, unstable audio routing, amplified media link jitter, and untimely session recovery under conditions of weak network, cross-network handover, and foreground/background switching. To achieve the ultimate goal of "fast call setup, stable maintenance, and smooth recovery," a layered high-reliability calling module for mobile VoIP was created. This module uses a five-layer structure: application orchestration layer, session service layer, transport adaptation layer, reliability enhancement layer, and observation and evaluation layer. It establishes unified control logic for aspects such as the Android Telecom/Jetpack calling interface, SIP and WebRTC signaling adaptation, link quality scoring, cross-path selection, limited redundancy forwarding, and call state machine recovery mechanisms. Based on the prototype system, 1200 call experiments were conducted in four scenarios: Wi-Fi, 4G, 5G, and 4G/5G handover. The results show that, compared to the single-path baseline solution, this module can increase the call setup success rate from 93.7% to 98.9%, reduce P95 call setup latency by 38.7%, reduce packet loss rate during handover by 56.3%, and improve the average subjective MO by 11.3%. Research indicates that while ensuring improved reliability through layered decoupling centered on calling, it significantly enhances the continuity and recoverability of real-time calls on Android devices.

1 Introduction

In the past three years, the application of real-time audio and video on mobile devices and internet voice calls has continued to rise, and research on the improvement of real-time communication in WebRTC, mobile QoE, and 5G scenarios has become more frequent [1-8]. From the perspective of system implementation, the Android platform is also constantly improving the calling scenario, adding more VoIP access-friendly features to Jetpack Core Telecom, and the latest calling app guide released by Android Developers once again points out that the Telecom application framework can provide users with call reminders, launch notifications, support for applications to prioritize information processing in the background and collaboration between terminals such as vehicles and wearable devices, as well as compatibility with previous usage scenarios [9-10]. It can be seen that the mobile calling system is no longer a simple matter of "connecting", but has become a comprehensive engineering problem that includes all engineering contents such as signaling control, media path, audio routing, system scheduling, and cross-device presentation.

However, many Android communication modules are still designed according to general communication concepts, focusing on sending and receiving messages while neglecting call continuity. For calling, the most direct user perception is not throughput, but rather the speed of call setup, the seamlessness of call switching, whether the call remains uninterrupted in weak network conditions, and whether the media remains controllable after switching between foreground and background processes. Because of these differences in perception, the calling module cannot rely solely on single-layer network retries or single-encoding adaptive solutions; therefore, a reliability system centered on the call session must be established in the architecture.

Existing research provides insights from different perspectives. Mahmoud and Abozaraiba conducted a comprehensive review of the applications and challenges of WebRTC, and in their summary, they proposed that weak network robustness, congestion control, and multi-scenario expansion are the main challenges currently faced by real-time communication systems [1]. Nakazato et al. analyzed the QoS of WebRTC in a real 5G environment and concluded that frequency band changes, location changes, and handover behavior in mobile environments will affect call quality [2]. Smirnov and Tomforde started with WebRTC rate control in a 5G environment and proved that control strategies for real-time applications can significantly improve QoE [3]. The multi-path redundancy framework proposed by Ito et al. shows that it is possible to improve the stability of real-time streams by using multiple link redundancies in mobile networks [4]. These studies provide technical support for improving the reliability of mobile calls, but they do not directly answer how the Android side integrates Telecom synthesis, signaling arrangement, path selection, restart, and observation loop.

Therefore, this paper selects Android mobile VoIP/video calling as the research object, and focuses on the layered high-reliability communication modules in the three main stages of call setup, call hold and call recovery. The main contributions of this paper are as follows: (1) A five-layer modular architecture for Android calling scenarios is proposed, which separates UI lifecycle, system Telecom access, SIP/WebRTC transmission, reliability enhancement and quality observation; (2) A linkage mechanism of link quality scoring, path selection and limited redundancy is designed for the optimization of call setup and hold under weak network and cross-network handover; (3) A prototype system is established and experiments are conducted in four typical network environments to verify the comprehensive impact of the proposed method on call setup success rate, latency, jitter, packet loss and MOS index.

2. Current Status Analysis of the Research Topic

2.1 Technological Evolution of Mobile Real-Time Calling

In terms of implementation, mobile calling has mainly gone through three stages: traditional SIP softphone, browser/native hybrid architecture based on WebRTC, and VoIP calling that is closely integrated with the mobile operating system. SIP has mature session control advantages, but it relies on additional encapsulation for NAT traversal, foreground/background scheduling, and compatibility with complex terminals. WebRTC is closer to the needs of modern VoIP/video calling in terms of media negotiation, ICE traversal, congestion awareness, and end-to-end real-time performance [1]. According to the latest Android calling app guide and Core Telecom evolution description, the system layer has unified VoIP calling into a large framework of calling capabilities, enabling applications to obtain foreground execution priority, audio routing, multi-terminal collaboration, and other capabilities.

However, from an engineering practice perspective, most applications still exhibit a loosely coupled structure of "signaling module + media SDK + UI callback". The advantage is fast development speed, but the disadvantages include unstable coupling between components, loss of session state after UI destruction, disconnection between audio device switching and the session layer, and weak network retry strategies only affecting the media layer and not feeding back to the call control layer. Therefore, while calling applications are usable under ideal network conditions, they exhibit significant vulnerability in complex scenarios such as handover, jitter, and background recovery.

2.2 Existing Research Focus and Limitations

Recent research has mainly focused on four aspects. First, the direction of QoE/QoS modeling is to derive the subjective user experience through objective indicators such as latency, jitter, and packet loss. Sharma et al. estimated WebRTC video QoE indicators without relying on the application layer header, providing a new idea for cross-application quality observation [6]; Bingöl et al.'s WebRTC-QoE dataset combines subjective ratings with network impairments, facial and voice features, providing an experimental basis for QoE prediction [7]. Second, the direction of interactive experience mainly studies interactive indicators such as silence rate, overlapping speech rate, and mouth-to-ear latency during meetings or calls. He et al. believe that interactivity is an aspect that cannot be ignored in calling scenarios [8]. Third, the direction of research on congestion control and rate adaptation to improve the quality of real-time streams in 5G or complex wireless environments [2-3]. Fourth, the proposal of multipath or new transmission protocols aims to improve stability by using QUIC, redundant transmission, or multi-operator access [4-5].

These works have laid the theoretical and algorithmic foundation for optimizing the calling module, but three shortcomings remain. First, most works only focus on the network and media layers, without integrating them into the Android platform. Second, the research objects are mostly conferences or general video communications, with little discussion on the establishment control, audio routing, and failure recovery of one-to-one calls on mobile devices. Third, some optimization strategies are only effective for a single algorithm layer, lacking a practical architecture, strategy, and observation loop.

2.3 Key Technical Requirements for Calling Scenarios

A high-reliability module for mobile calling should meet the following requirements. First, it should have fast call setup. If a call doesn't connect or ring within 2-3 seconds of the user clicking to dial, the subjective experience will quickly deteriorate. Second, it should maintain stability. Instantaneous jitter, short-term packet loss, and bitrate fluctuations during the call should be quickly absorbed, preventing continuous popping or stuttering. Third, it should recover quickly. In mobile environments, Wi-Fi/cellular switching, system recycling, Bluetooth headset reconnection, and foreground/background switching are frequent occurrences, so the module must have lightweight state reconstruction and session recovery capabilities. Finally, it should have accurate observation. Relying solely on application logs without a unified quality collection and attribution channel makes it impossible to continuously improve the calling experience using real-world data.

3. Raise questions

Based on the above situation, this article summarizes the main problems of the Android calling module as follows.

3.1 Disconnect between session control and platform capabilities

In traditional implementations, the call state machine is mostly controlled by the application layer, with system telephony capabilities, audio device management, and notification strategies handled by different APIs. This leads to two consequences: first, after switching between foreground and background or rebuilding an Activity, the call state is difficult to reconcile with the system's visible state; second, peripherals such as Bluetooth, speakers, car infotainment systems, and watches cannot reliably inherit the call context, affecting the user's continuous operation experience. Android's CallsManager and Telecom capabilities could provide better system access, but without a layered module design, they are unlikely to be truly effective [9-10].

3.2 Single-path transmission leads to insufficient robustness in weak networks

Single path refers to the situation where, during a mobile call, if the current network quality drops for a short period of time, the call can only passively wait for congestion control to converge. For the call setup phase, it will increase the time for INVITE/SDP negotiation and ICE completion; for the call hold phase, it will increase the pressure on jitter buffer and PLC. Especially in 4G/5G handover or mobile environments such as subways and vehicles, the single path solution will result in a chain reaction of short-term unavailability, sudden quality drop, and delayed recovery [2][4].

3.3 Lack of unified indicators for quality feedback and recovery decision-making

Many calling applications, while recording statistics such as RTT, jitter, and packet loss, lack comprehensive metrics that directly drive decision-making. For example, the choice of which primary link to use, the appropriate limited redundancy approach, whether to downgrade the video quality or extend the renegotiation latency, should all be addressed with a link availability score that directly reflects link availability and a rule for determining session risk levels. Without a unified metric, different submodules may offer conflicting opinions on the same quality change.

3.4 Insufficient observation loop makes continuous optimization difficult.

Calling experience issues are highly context-dependent; the same crash or dropout can have multiple causes, such as signaling timeouts, ICE failures, slow DNS resolution, audio focus contention, Bluetooth route rebuilding, and delayed link switching. Without layered logging and unified feedback, developers can only rely on scattered logs for troubleshooting, which is not only time-consuming but also makes cross-version comparisons difficult.

4. Problem Solving and Strategies

4.1 Overall Layered Architecture Design

The calling module proposed in this paper adopts a five-layer structure, as shown in Table 1. The application orchestration layer is responsible for the dialing entry, call UI, and call lifecycle binding; the session service layer handles call creation, retries, and adaptation with the system Telecom; the transport adaptation layer encapsulates SIP/WebRCT call establishment, media negotiation, and

NAT traversal; the reliability enhancement layer arranges link scoring, path selection, redundancy, and recovery; and the observation and evaluation layer captures quality indicators throughout the call process and reports them upwards. This solution avoids compressing the main logic into a specific SDK or a single protocol by pushing control logic upwards, delegating transmission details downwards, and ensuring independent quality judgment.

Table 1 Layered Architecture and Implementation Responsibilities

Layer	Role	Key implementation
Application	CallSession orchestration, UI state, lifecycle binding	Kotlin + Coroutines + StateFlow
Service	Signaling, media control, retry policy, endpoint abstraction	Foreground service + Telecom bridge
Transport	SIP/WebRTC signaling, ICE/NAT traversal, congestion feedback	WebRTC stack / SIP adapter
Reliability	Link scoring, path selection, redundancy, failure recovery	Rule engine + metrics collector
Observation	RTT/jitter/loss/MOS telemetry and trace aggregation	Metrics SDK + local buffer

Table 1 illustrates that the proposed solution does not integrate all reliability functions into the media SDK. Instead, it uses a separate reliability layer to link call services and transmission details. This allows changes in call status, audio routing, and network quality responses to all be handled through a single interface. For Android devices, this improves consistency across system telecom, notification, Bluetooth, and multi-terminal display capabilities, and also facilitates future replacements of underlying signaling or media implementations.

4.2 Link Quality Scoring Model

To transform complex network observations into executable control variables, a link quality score is defined as...

$$Q_l=\alpha\frac{B_l}{B_{max}}+\beta\left(1-L_l\right)+\gamma\left(1-\frac{J_l}{J_{max}}\right)+\delta\left(1-\frac{R_l}{R_{max}}\right) \tag{1}$$

Where $Q_l$represents $l$the overall quality score of the link, $B_l$is the estimated available bandwidth, $L_l$is the packet loss rate, $J_l$is the jitter, $R_l$and is the round-trip latency; $B_{\max}$and $J_{\max}$are $R_{\max}$the normalized upper bounds, respectively; $\alpha, \beta, \gamma, \delta$and are weights that satisfy and = 1. The core idea of this formula is that calling services do not pursue extreme throughput, but rather emphasize low packet loss, low jitter, and acceptable latency. Therefore, in actual implementation, $\beta$the weights of and can be appropriately increased.γ

As can be seen from equation (1), the observation data can be directly converted into a quantitative judgment on whether the current link is suitable for carrying the main call, providing a unified basis for subsequent path selection and redundancy control.

4.3 Path Selection and Switching Strategies

When both Wi-Fi and cellular connectivity are available, or when multiple carriers/PDPs are available, select the current optimal bearer path.

$$p_t^* = \underset{p \in P}{\operatorname{argmax}} \left(Q_p - \lambda C_p - \mu S_p \right) \tag{2}$$

Here, P represents the set of candidate paths, Q_p represents the path quality score, represents C_p the path switching cost, S_p represents the current path stability risk, λ and and μ are weighting coefficients. Using only Q_p the maximum value as the decision criterion can easily lead to frequent jittery switching; therefore, switching cost and stability penalties need to be introduced. This ensures that the calling module does not repeatedly reselect paths when the link fluctuates, but only switches when the quality significantly improves or the current path risk increases.

4.4 Limited Redundancy and Recovery Control

For handover and highly volatile environments, this paper will not always use dual-engine, dual-redundancy to ensure system safety, but will dynamically activate limited redundancy according to the level of risk. Redundancy factor is defined as follows:

$$r_t = \min(r_{max}, 1 + \eta L_t + \xi H_t) \tag{3}$$

Where, r_t is t the redundancy intensity at time, L_t is the instantaneous packet loss level, H_t represents the handover/interruption risk indicator, η and ξ is the control coefficient. When the network is stable, r_t it is close to 1, and the module only uses the main path; when a handover window or packet loss surge is detected, limited redundancy is activated for a short time to protect critical audio packets and key frame signaling. This strategy absorbs the ideas of multi-path redundancy research [4] and avoids high overhead in normal conditions.

Furthermore, the available reconstruction budget within a single recovery window is defined as:

$$T_{rec} = T_{budget} - \left(T_{sig} + T_{ice} + T_{route} \right) \tag{4}$$

Here, T_{budget} represents the user's tolerable total recovery budget, T_{sig} represents the signaling recovery time, T_{ice} represents the ICE/path renegotiation time, and T_{route} represents the time required for audio routing and device switching. If these values T_{rec} are too small, "preserve audio, reduce video" will be prioritized over a complete reconstruction to ensure session continuity takes precedence over media integrity.

4.5 Control Objectives Driven by QoE

To enable the calling module's strategy to not only rely on the underlying network but also closely approximate user perception, a comprehensive utility function is defined.

$$U = \omega_1 \cdot MOS - \omega_2 \cdot D_{setup} - \omega_3 \cdot P_{drop} - \omega_4 \cdot I_{interrupt} \tag{5}$$

Where, MOS is the subjective voice/call quality score estimate, D_{setup} is the call setup delay, P_{drop} is the dropped call probability, $I_{interrupt}$ is the call interruption count or interruption duration index, ω_1 and ω_4 is the weight. Equation (5) reflects the design principle of this paper: for mobile calling, call setup delay and call interruption often have a greater impact on user perception than peak bit rate, so they should be directly included in the optimization objective.

4.6 Module Implementation Process

In implementation, after a call is initiated, the application orchestration layer creates a CallSession, and then the session service layer calls Telecom/CallsManager to register with the system and start the foreground service to ensure the priority of the call establishment phase. The transmission adaptation layer decides whether to use SIP or WebRTC signaling flow according to the capabilities of the peer, and continuously reports RTT, jitter, packet loss and available bandwidth to the reliability layer. The reliability layer calculates the importance coefficients of each item in formulas (1) to (5) to determine whether to switch to the main channel, use short-term redundancy or degrade recovery. The observation layer writes the call establishment delay, session interruption, device switching and MOS estimation into the local cache, and then sends them back in batches for offline tracing and version comparison.

Table 2 Experimental Network Scenario Settings

Scenario	Network condition	RTT	Jitter	Loss	Call attempts
S1	Campus Wi-Fi stable	35 ms	4 ms	0.4%	300
S2	Urban 4G medium load	78 ms	11 ms	1.8%	300
S3	Sub-6 5G stable	41 ms	5 ms	0.6%	300
S4	4G/5G handover with mobility	126 ms	19 ms	3.7%	300

Table 2 shows the four scenarios used during prototype verification. The design philosophy is not to cover all networks, but rather to specifically select the four conditions that have the greatest impact on calling continuity: stable Wi-Fi, standard 4G, stable 5G, and mobile handover scenarios. Scenario S4 was used to test the recovery of the call module when it switches between networks, which is the most unstable aspect of most Android calling applications.

5. Experiment and Results Analysis

5.1 Experimental Setup

This paper implements a prototype module on an Android 14/15 test terminal. Opus is used for voice encoding, VP8/VP9 adaptive layer is used for video, and both SIP and WebRTC access are supported on the signaling side. To emphasize the calling aspect, the focus is on observing call setup efficiency, handover continuity, voice quality, and subjective perception during walkie-talkie calls, with large-scale conference throughput as the primary evaluation metric. The baseline system uses a single-path calling module, with a fixed, unscored, explicit main link and short-term redundancy, relying solely on the default behavior of the underlying SDK as a recovery strategy. The comparison system is the layered high-reliability calling module proposed in this paper.

5.2 Call setup delay distribution

Figure 1 shows that the proposed scheme outperforms the single-path baseline scheme across the entire distribution range. The baseline scheme still exhibits a long tail after 1.5 seconds, indicating call setup delays during weak networks or handovers. The hierarchical high-reliability scheme shows an overall leftward shift in its curve, suggesting that many calls can complete system registration, signaling negotiation, and media preparation in a shorter time. For mobile calling, compressing the distribution tail is crucial, as users are highly sensitive to even the few "very slow" dialing experiences. This result demonstrates that implementing foreground priority control during call setup and path selection can effectively improve the response speed of the first call.

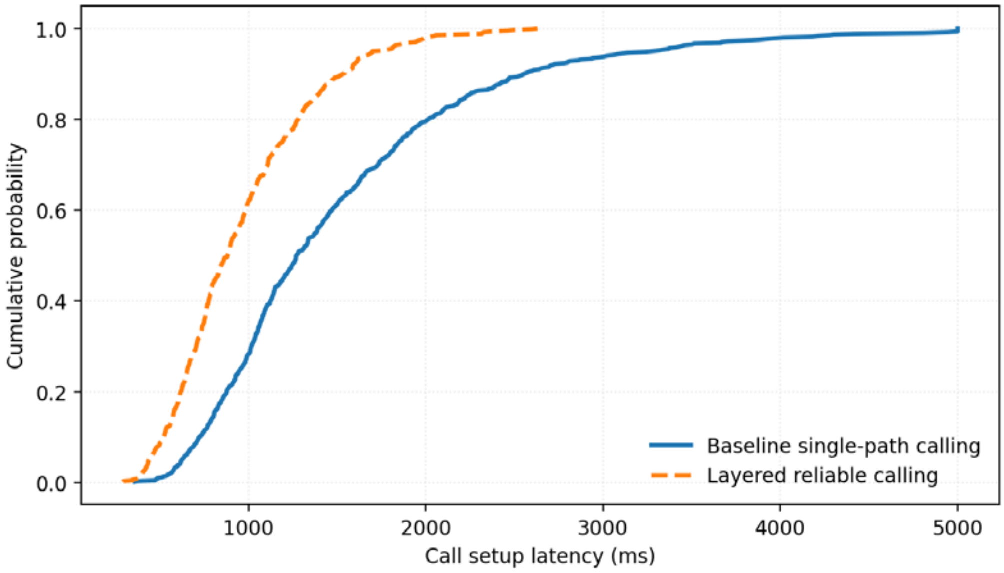

Figure 1. Cumulative distribution of call setup latency under different architectures

5.3 Subjective MOS Quality Comparison

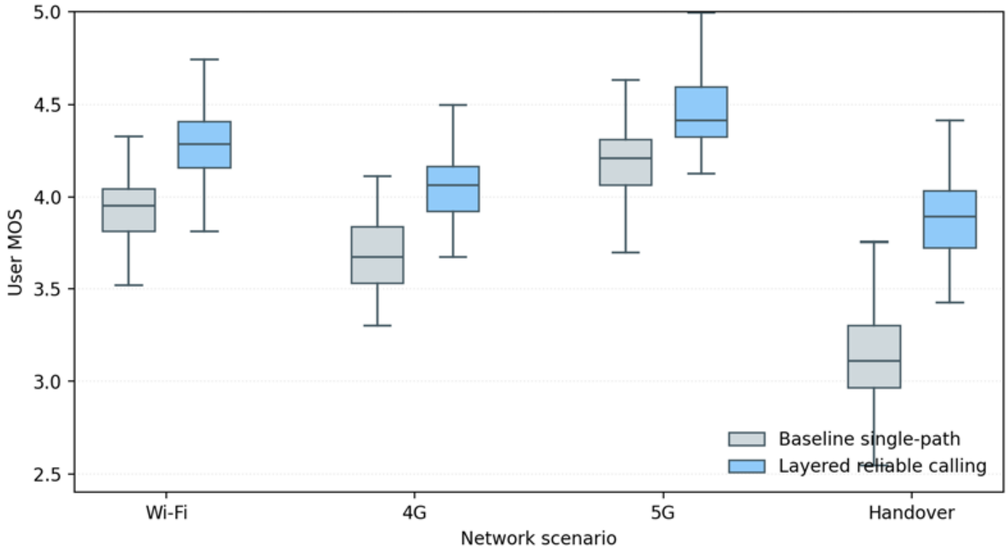

Figure 2. Subjective MOS distribution under different network scenarios

Figure 2 shows that the proposed method achieves a higher median MOS than the baseline in all four network environments, with the most significant improvement observed in handover scenarios. This indicates that the benefits of the proposed strategy extend beyond simple improvements in ideal network environments; it also provides greater suppression of intermittent connections, distortion, and session fluctuations under challenging conditions. Particularly during 4G and 5G handover, the baseline scheme exhibits a more dispersed distribution and a higher number of low-scoring samples; in contrast, the hierarchical calling module shows a more concentrated distribution, exhibiting less fluctuation in experience due to network complexity. For practical products, reducing experience variance is just as important as increasing the average.

5.4 Summary of Key Indicators

Table 3 Comparison of Key Performance Indicators

Metric	Baseline single-path	Layered reliable calling	Improvement
Call setup success rate	93.7%	98.9%	+5.2 pp
P95 call setup latency	2380 ms	1460 ms	-38.7%
Median jitter during call	17.2 ms	10.4 ms	-39.5%
Packet loss during handover	4.8%	2.1%	-56.3%
Mean user MOS	3.73	4.15	+11.3%

Table 3 illustrates that the benefits of the proposed solution are consistent across multiple metrics: high call establishment success rate, low latency, low packet loss rate during handover, and minimal in-call jitter. This means that the hierarchical calling structure does not sacrifice one aspect for another, but rather achieves more rational resource allocation from a system control perspective. In particular, the improvement in call drop-related metrics demonstrates the effective synergy between path scoring, short-term redundancy, and recovery budget control.

5.5 Discussion

First, this paper emphasizes the system collaboration value of Android calling scenarios. Unlike optimizing only the network layer, integrating Telecom access, foreground services, audio routing, and the session state machine can reduce the impact of many "non-network issues" on the call experience. Second, limited redundancy is more suitable for mobile terminal use than permanent dual-transmission. Although calling services are sensitive to continuity, traffic and energy consumption must also be considered; therefore, short-term redundancy based on risk triggers is more practical. Third, a unified observation model is the foundation for continuous development. By using a comprehensive evaluation of link scoring, session events, and subjective indicators to build a learning model, it can gradually identify which scenarios are most likely to cause call establishment failures or call interruptions during version iterations, providing a foundation for the refinement of future machine learning decisions.

Of course, this paper also has some limitations. First, it only conducted one-to-many call experiments and did not include multi-person conferences or large-scale forwarding. Second, it did not use a larger online real-world sample to replace experimental scoring. Third, the underlying implementation is mainly for common Android terminals, and the effectiveness of collaborative use on different device forms, such as wearables, in-vehicle systems, and foldable screens, has not yet been verified.

6. Conclusion

To address issues such as slow call setup, unstable weak networks, frequent handover disconnections, and slow recovery in Android mobile calling scenarios, this paper proposes a layered high-reliability call module for mobile VoIP call continuity. This scheme uses the calling session as a hub, decomposing application orchestration, session service, transmission adaptation, reliability improvement, and observation and evaluation into five layers. It relies on a consensus decision-making mechanism based on link quality scoring, path optimization, limited redundancy triggered by risks, and recovery budget. Experimental results show that this module can simultaneously improve call setup success rate, P95 call setup latency, handover packet loss, and subjective MOS, indicating that the layered decoupling approach centered on the calling session is superior to typical communication module designs.

In the future, development can continue along these three paths: First, use a more refined QoE learning model to directly incorporate user interaction metrics into real-time control; second, rely on new transmission methods such as QUIC and WebTransport to explore media transfer with lower recovery costs; and third, extend to Android Auto, Wear OS, and tablet VoIP terminals to achieve a highly reliable calling system across devices.

References

[1] Zheng, H. (2026, April). Research on Cloud-Edge Collaborative Elastic Computing and Cost Optimization for High-Concurrency Scenarios. In 2026 IEEE 15th International Conference on Communication Systems and Network Technologies (CSNT) (pp. 1489-1496). IEEE.

[2] Liu, H. (2025, December). Mlops Model Deployment System for Multi-Cloud Environments and Improvement of Commercial AI Service Availability. In 2025 IEEE International Conference on Communication Networks and Computing (CNC) (pp. 1047-1052). IEEE.

[3] Hong, Y. (2025, December). Chain Demand Mutation Identification and Emergency Response Decision-Making Integrated With Social Media Sentiment Analysis. In 2025 IEEE International Conference on Communication Networks and Computing (CNC) (pp. 425-432). IEEE.

[4] Chen, M. (2026, March). Design of a Privacy-Oriented AI Compliance Hook System Based on Static Code Analysis. In 2026 IEEE Madhya Pradesh Section Conference (MPCON) (pp. 648-654). IEEE.

[5] Lyu, N. (2026, April). Toward Robust AI Agents: A Closed-Loop Task Planning–Execution–Feedback Framework for Open Scenarios. In 2026 IEEE 15th International Conference on Communication Systems and Network Technologies (CSNT) (pp. 1182-1188). IEEE.

[6] Zhang, K. (2025, December). Research on Cross-Selling Precision Marketing Strategy Based On Xgboost Model and SHAP Explanatory Framework. In 2025 IEEE 1st International Conference on Recent Trends in Computing and Smart Mobility (RCSM) (pp. 1-7). IEEE.

[7] Zhu, P. (2025, December). Construction of Multi-Scale Biostatistical Analysis Framework and its Application in Biomedical Signal Feature Recognition and Classification. In 2025 5th International Conference on Mobile Networks and Wireless Communications (ICMNWC) (pp. 1-7). IEEE.

[8] Chang, C. W. (2026). Privacy Infrastructure Vulnerability Mining and Automated Framework Based on Multimodal AI. Procedia Computer Science, 279, 702-711.

[9] Wang, Y. (2026). Construction of Supply Chain Demand Forecasting Algorithm Based On Time Series Data Analysis and Improvement of Its Management Efficiency. Procedia Computer Science, 281, 484-493.

[10] Wang, Z. (2026). Mineral Commodity Time-Series Cycle Modeling and Feature Learning Algorithm Based On Improved Transformer. Procedia Computer Science, 281, 1347-1356.

[11] Chen, J. (2026). Construction of a Cloud-Native High-Performance Service Engineering System for Real-Time Decision-Making Platforms.

[12] Qian, X. (2026). Supply Chain Collaboration Mechanisms Driven by Demand Orchestration in Omni-Channel Retail Environments.

[13] Wang, Y. (2025). Application of Data Completion and Full Lifecycle Cost Optimization Integrating Artificial Intelligence in Supply Chain.

[14] Sun, J. (2025). Quantile Regression Study on the Impact of Investor Sentiment on Financial Credit from the Perspective of Behavioral Finance.

[15] *Chen, M. (2025). Research on Automated Risk Detection Methods in Machine Learning Integrating Privacy Computing.*
[16] *Wu, Y. (2025). Optimization of Generative AI Intelligent Interaction System Based on Adversarial Attack Defense and Content Controllable Generation.*
[17] *Wu, Y. (2025). Software Engineering Practice of Microservice Architecture in Full Stack Development: From Architecture Design to Performance Optimization. Machine Learning Theory and Practice, 5(1), 64-75.*
[18] *Liang, Q. (2026). Research on the Renovation Design Path for Enhancing the Efficiency of Mixed-Use Office and Retail Spaces. European Journal of AI, Computing & Informatics, 2(2), 163-170.*
[19] *Liang, Q. (2026). Reconstruction of Commercial Building Space Reuse Mode Driven by Composite Business Types. International Journal of Engineering Advances, 3(1), 107-113.*
[20] *Gao, Y. (2026). The Role and Practice of Delaware Law in Global Cross-border M&A Transactions. International Journal of Law, Policy & Society, 2(1), 38-46.*
[21] *Gao, Y. (2026). Application of Delaware Corporate Law in Cross-Border M&A: Business Structures, Contractual Risk, and Case-Based Lessons. Economics and Management Innovation, 3(2), 128-136.*
[22] *Yin, J. (2025). Research on Financial Time Series Prediction Model Based on Multi Attention Mechanism and Emotional Feature Fusion. Socio-Economic Statistics Research (2025), 6(2), 161-169.*
[23] *Wang, N. (2026). Research on Evaluation and Optimization Models of Enterprise Resource Allocation Efficiency from a Data-driven Perspective. Advances in Computer and Communication, 7(1).*
[24] *Su, J. (2026). Research on Android Real-time Communication System Architecture and High-reliability Assurance Pathways Integrating AI-based Anomaly Detection Mechanisms. Engineering Advances, 6(2).*
[25] *Liang, Q. (2026). How Architectural Design and Utility Infrastructure Impacts AI Supporting Campus and Drive Future Innovation, Operational Efficiency and Sustainable Advancement in Utility-Critical Environment. European Journal of Engineering and Technologies, 2(2), 71-77.*
[26] *Wu, Y. (2025). Software Engineering Practice of Microservice Architecture in Full Stack Development: From Architecture Design to Performance Optimization. Machine Learning Theory and Practice, 5(1), 64-75.*
[27] *Liu, D., Shen, Q., & Liu, J. (2026). The Health-Wealth Gradient in Labor Markets: Integrating Health, Insurance, and Social Metrics to Predict Employment Density. Computation, 14(1), 22.*
[28] *Chen, J. (2026). Elastic Scaling and Stability Assurance Mechanisms for Distributed Systems under High-Throughput Scenarios.*
[29] *Gao, Y. (2026). Research on the Design of Governance Structure for Private Equity Funds and the balance of GP and LP Rights.*
[30] *Huang, J. (2026). From Policy Authorization to Practical Execution: A Decision-Support Framework for Implementing Housing Supply Strategies in the United States. Strategic Management Insights, 3(1), 24-31.*