A Five-Year Retrospective of Cellular Reliability Evolution: The Encouraging, Disappointing, and Further Enhancements

Yunhao Liu, *Fellow, IEEE*, Hongyi Wang, Yang Li, Zhenhua Li, *Senior Member, IEEE*, Guoquan Zhang, and Lei Yang

***Abstract*—With recent advances on cellular technologies pushing the boundary of cellular performance, cellular reliability has become a key concern of their adoption and deployment. To fully understand cellular reliability, we work with a major Android phone vendor, Xiaomi, to conduct a long-term (2020-2024) and large-scale (involving 123M users) measurement study in China, with coarse-grained general statistics and fine-grained sampling diagnostics. Our measurement reveals contrasting evolution trends of cellular failures in different stages of the data connection: in the past five years, failures after connection establishment decrease remarkably (by 29%), while failures during connection setup exhibit a sharp increase (by 38%). Our analysis illustrates that the contrast stems from the joint impact of multiple stakeholders, including ISPs' increasing deployment of 5G base stations, 5G infrastructure upgrade from NSA (Non-Standalone) to SA (Standalone) mode, software defects coming from Android's adaptation to new cellular technologies, and so forth. Our work provides actionable insights for improving cellular reliability at scale. More importantly, we have built on our insights to develop enhancements that effectively address cellular reliability issues with remarkable real-world impact—our optimizations have reduced 38% cellular connection failures for 5G phones and 31% failure recovery time across all phones.**

***Index Terms*—Cellular network, 5G, reliability, measurement, android, cellular connection management.**

I. Introduction

CELLULAR technologies have been the keystone of mobile systems and applications that empower our daily lives, all the way from wireless telephony and mobile Internet, to emerging applications such as ultra high-definition (UHD) video streaming and AR/VR [1]. The rise of 5G technologies has started to realize even higher-bandwidth and lower-latency cellular networks, driving the grand vision of AI, IoT, and self-driving vehicles [2]. Specifically, 5G cellular networks support up to 10 Gbps bandwidth (100× faster than 4G), 1 ms latency (*cf.* 30-50 ms for 4G), and connection density of 1000^2 devices per square kilometer (100× more than 4G) [3].

However, the performance of cellular network technologies and the availability of cellular network services depend crucially on reliable connections, making cellular reliability a critical concern. From a mobile device's perspective, cellular data connections can fail mostly in the following three ways (we use Android terminologies throughout the paper):

- Data_Setup_Error [4] [1]: The mobile device can receive signals from a nearby base station (BS) but cannot establish a data connection with the BS.
- Out_of_Service [5]: The data connection has been established, but the mobile device cannot receive cellular data.
- Data_Stall [6]: The mobile device can receive cellular data, but the data connection abnormally stalls (for longer than one minute as suggested in Android).

Measurement. To fully understand cellular reliability in the wild, we collaborate with a major Android phone vendor, Xiaomi Co. LTD, which serves hundreds of millions of mobile users in China, to conduct a large-scale, in-depth study from the device perspective. We build a continuous monitoring infrastructure on top of a customized Android system called CrowdEye, which records system-level traces (without requiring root privileges) once suspicious cellular data connection failure events (abbreviated as *cellular failures*) occur. To extract true failure events and collect diagnostic information, we instrument relevant system services to record detailed device/network state information and filter out false positives.

We invited all users of Xiaomi to participate in the study by installing CrowdEye on their phones, and finally 123M user devices opted in and shared data with us for a total of 16 months comprising two separate periods: 01/01/2020–08/30/2020 and 10/01/2023–06/30/2024. The dataset involves 70 different models of Android phones, 3 mobile ISPs (referred to as ISP-A: China Mobile, ISP-B: China Telecom, and ISP-C: China Unicom), and 8.1M BSes. All the data were collected under informed user consent and a proper IRB.

Analysis. Combining the above fine-grained dataset with Xiaomi's coarse-grained dataset collected from its users during 2020-2024, we have several important observations. Particularly, over the past five years, despite substantial advances on cellular technologies, the severity of cellular reliability problems has still worsened on the whole: the *prevalence* (the

Received 21 November 2024; revised 16 April 2025; accepted 17 May 2025; approved by IEEE TRANSACTIONS ON NETWORKING Editor Y. Chen. Date of publication 18 June 2025; date of current version 17 October 2025. This work was supported in part by the National Key Research Plan under Grant 2021YFB2900100 and in part by NSFC under Grant 62332012 and Grant 62441233 and in part by the Some preliminary results have been presented in the SIGCOMM'21 conference. *(Corresponding author: Yunhao Liu.)*

Yunhao Liu is with Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: yunhaoliu@gmail.com).

Hongyi Wang, Yang Li, and Zhenhua Li are with the School of Software, Tsinghua University, Beijing 100084, China (e-mail: hywangthu@gmail.com; liyang14thu@gmail.com; lizhenhua1983@gmail.com).

Guoquan Zhang and Lei Yang are with Xiaomi Technology Company Ltd., Beijing 100085, China (e-mail: zhangguoquan@xiaomi.com; yanglei14@xiaomi.com).

Digital Object Identifier 10.1109/TON.2025.3573839

[1] Strictly speaking, some Data_Setup_Error events defined in Android are not true failures since they occur rationally due to BS overloading. Such false positives will be carefully removed in our study.

fraction of devices that experience at least one cellular failure) of cellular failures grows by 39% (23% → 32%) and the *frequency* (the average number of cellular failures experienced per phone) grows by 3% (33 → 34). To demystify the counter-intuitive observation, we delve deep into the variations of cellular failures in different stages of the data connection.

Encouragingly, cellular failures (*i.e.,* Out_of_Service and Data_Stall) after connection establishment decrease by 29.4%, with a remarkable drop on 5G phones by 66.0%. This is largely due to China's ISPs' enormous capital and technical investments in 5G infrastructure in recent years, such as increasing the number of 5G BSes (by 317%) to improve signal coverage, and deploying the Massive MIMO technique for indoor scenarios to reduce signal interference [7].

Disappointingly, cellular failures (*i.e.,* Data_Setup_Error) during connection setup increase by 37.5%, with an astonishing rise (by 200%) on 5G phones. This is primarily because the 4G/5G dual connectivity mechanism [8] is now rarely needed, as most 5G BSes have transitioned from NSA (Non-Standalone) to SA (Standalone) mode. User devices now need to establish new connections when switching between 4G and 5G; consequently, more frequent connection setups lead to a higher likelihood of Data_Setup_Error failures.

In a nutshell, our results indicate that cellular failures are mainly caused by software defects rather than inexpensive hardware. From the viewpoint of user devices, the software implementation that blindly prioritizes 5G connection since Android 10 greatly impairs the stability of cellular connections. Besides, the three-stage progressive mechanism for Data_Stall recovery since Android 7 turns out to be rather inefficient [8]. In addition, the premature failure judgment in Android 14 leads to considerable *overdue cellular reconnections*.[2]

From the viewpoint of ISPs, cellular failures occur more prevalently (31%) on ISP-B users than on ISP-A (24%) users and ISP-C users (25%) due to the inferior signal coverage of ISP-B. While both the number and overall signal coverage of 3G BSes are smaller than those of 2G, 4G or 5G BSes, the prevalence of failures on 3G BSes is lower than that on 2G, 4G or 5G BSes. This can be ascribed to the fact that 3G access is usually not preferred when 4G/5G access is available and its signal coverage is worse than that of 2G when 4G/5G access is unavailable, resulting in less resource contention.

From the viewpoint of BSes, common wisdom suggests a positive correlation between cellular reliability and received signal strength (RSS). However, our measurement shows the opposite when there is excellent (level 5) RSS—failures are more likely to happen in this case (24%) than when there is weaker (level 1 to 4) RSS (11%—21%). We clear up the mystery—most excellent-RSS failures come from densely-deployed BSes around public transport hubs; while such BSes offer excellent RSS, they increase the control-channel overhead of mobility management [9], [10], causing frequent failures tagged with EMM_ACCESS_BARRED, INVALID_EMM_STATE, *etc.* [11].

[2]If an app requests to start a network session (when the cellular connection is not set up) and then quickly cancels the request, the connection will be terminated during its establishment. Unfortunately, Android 14 misregards all the cellular connections terminated during establishments (dubbed *aborted connections*) as Data_Setup_Error failures. It will no longer perform reconnections even if new network sessions require them, as it assumes that retrying aborted connections would still be unsuccessful, which is often the case but not always. In the misjudged case, the undesirable situation will persist until a manual reset of the connection, leading to a very long recovery time.

Finally, we are concerned with the failure recovery time, which mostly comes from Data_Stall in 2020 but Data_Setup_Error in 2023/2024. In 2020, a Data_Stall failure lasts for as long as 2.9 minutes on average, due to the aforementioned inefficient three-stage Data_Stall recovery mechanism. Around four years later, Data_Setup_Error failures account for the most of the recovery time and take an average of 2.7 minutes to fix; particularly, the recovery time in 18.4% cases exceeds 3 minutes, and the majority (>60%) of these cases are attributed to the aforementioned premature failure judgment in Android 14.

Enhancements. In our preliminary work [8], we propose two-fold optimizations by 1) devising a stability-compatible RAT transition mechanism to enhance cellular reliability and 2) leveraging the TIMP (time-inhomogeneous Markov process [12]) model to accelerate the Data_Stall recovery. Real-world evaluation demonstrates that the former reduces 40% cellular failures for 5G phones and the latter decreases 38% Data_Stall recovery time for all phones.

In this work, we make further enhancements by identifying app-induced aborted connections and speeding up their recovery with cross-layer context tracing and meticulous failure judgment. During the establishment of a cellular connection, if no network session is found to require the connection, we trace backwards to the network session that triggered the connection establishment. If the setup request for this network session is actively canceled by its initiator, we must not consider the connection as a failure; instead, we let it terminate normally. In this way, the cellular connection can be timely re-established when a new network session requires it, largely saving the recovery time of Data_Setup_Error failures.

Since the release of the patched CrowdEye system with the meticulous failure judgment (adopted by 16M opt-in users in late Jul. 2024), we have successfully reduced Data_Setup_Error failures by 14% and shortened their recovery time by 56% during Aug.–Sep. 2024. We have also reported the design defect of Android 14 to its official development team in Google, who has confirmed the issue and promised to fix it in the upcoming release [13]. Overall, the three enhancements jointly reduce 38% cellular failures for 5G phones and save 31% failure recovery time across all phones.

Code Release. All the diagnosis and fixing code for cellular failures involved in the study is publicly available at https://CellReliabilityEvo.github.io.

II. Study Methodology

We conducted a large-scale measurement study on cellular failures based on continuously monitoring 122M opt-in user devices over a total of 16 months comprising two separate periods (01/01/2020–08/30/2020 and 10/01/2023–06/30/2024). The study is enabled by CrowdEye, a customized Android system that provides lightweight, privacy-preserving tracing and analysis beyond the capability of the vanilla Android system.

A. Limitations of Vanilla Android

Cellular connection management exists as a system service in Android, where the life cycle of a cellular data connection is modeled by a state machine [14] as shown in Figure 1: a total of five states are used to represent different stages of a cellular connection, including Disconnected, Connecting, Retrying, Connected, and Disconnecting. As one state changes

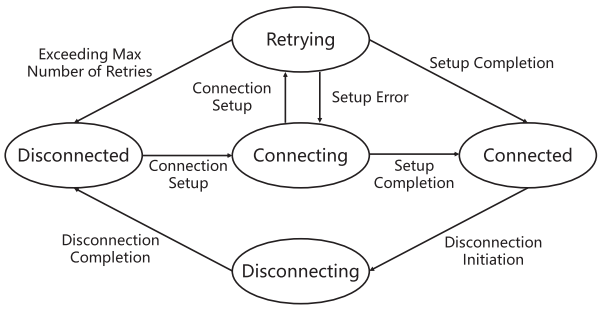


Fig. 1. The state machine that models the life cycle of a cellular data connection in Android.

to another, Android provides quite a few facilities to monitor various problems during the process, most of which are related to our targeted cellular failures.

First of all, if a user device fails to establish a data connection to a nearby base station (BS), a Data_Setup_Error [4] failure event will be reported to relevant system services (but not to user-space apps); then, a retry attempt will be initiated, trying to establish the data connection once again. Here the failure may occur at the physical layer (*e.g.,* radio signal loss), the data link or MAC layer (*e.g.,* device authentication failure), and/or the network layer (*e.g.,* IP address allocation failure). Upon any failure, an error code will be generated by the underlying radio interface, based on either the received responses to the issued connection-setup negotiation requests (if any) or the return values of the modem commands executed by the underlying radio interface.

Further, if the data connection is successfully established but the user device still cannot access the cellular network, *i.e.,* the user device cannot send/receive data to/from outside, Android will mark its current service state as Out_of_Service [5]. Worse still, even if data can be sent to or received from outside, sometimes the data connection can abnormally stall for a long time, incurring annoying user experience. This phenomenon is termed as Data_Stall [6] in Android. In detail, when there have been over 10 outbound TCP segments but not a single inbound TCP segment during the last minute (the statistics are made by the Linux kernel in its network protocol stack), a Data_Stall failure is reported to both relevant system services and user-space apps. In addition, there are other types of failures we do not elaborate here but will mention when necessary in the remaining parts.

For all the abovementioned failure events, Android currently provides basic notification interfaces with which the relevant system services can register themselves as the event listeners. Nevertheless, only a part of the interfaces (including the Data_Stall notifier and Out_of_Service checker) are exposed to user-space apps, and some interfaces are inaccessible even with root privileges. Therefore, we are unable to capture all the concerned failure events without modifications at the system level. To make the matter worse, some of the abovementioned failure events are in fact not true failures. For example, a data connection setup request may be rejected by a nearby BS which is currently overloaded; in this case, a Data_Setup_Error event will be reported but does not imply a true failure. Additionally, the event-related information reported by Android is often insufficient for in-depth analysis. In fact, Android typically only reports the occurrence of a failure event without capturing other important in-situ information, such as the desired BS information, received signal strength (RSS), protocol error code, and network state.

B. Continuous Monitoring Infrastructure

To practically address the above-described multifold challenges, we customize the vanilla Android system for continuously acquiring fine-grained system-level traces upon the occurrence of suspicious cellular failure events, which are otherwise impossible to obtain but are crucial to our study requirements. The resulting system is called CrowdEye, in which we focus on modifying the Framework-layer programs. We do not make modifications to the hardware abstract layer (HAL) or the kernel layer—while HAL/kernel modifications can help us collect more underlying and detailed data, they can easily impair the system stability and robustness in practice (even with careful testing) [15].

At a high level, our modifications are made to realize three goals: 1) system service instrumentation, 2) concerned information logging, and 3) failure recovery monitoring. Specifically, we first instrument the Android system service of cellular connection management by integrating our developed monitoring service as its event listener, so that all the occurrences of Data_Setup_Error, Out_of_Service, Data_Stall, and other concerned failure events can be captured in real time. It is worth noting that when instrumenting the service, we carefully rule out a variety of false failure events (*a.k.a.,* false positives), such as connection disruption by incoming voice calls, service suspension due to insufficient account balance, and manual disconnection of the network.

Second, we need to record important radio- and BS-related information upon the occurrence of a cellular failure for in-depth analysis. Such information includes the current radio access technology (RAT, *e.g.,* 4G LTE or 5G NR), received signal strength (RSS), access point names (APNs), and BS ID that consists of Mobile Country Code (MCC), Mobile Network Code (MNC), Location Area Code (LAC), and Cell Identity (CID).[3] All these information can be accessed via the APIs provided by the Android TelephonyManager and ServiceState services. Besides, we record the protocol error codes for Data_Setup_Error events to further rule out possible false positives such as rational setup rejection due to BS overloading. In particular, we have carefully analyzed all the 344 cellular connection-related error codes defined in Android [11], and recognized tens of error codes that are highly correlated with false positives as critical auxiliary information.

Further, to deeply understand the root causes of Data_Setup_Error failures, we meticulously trace the entire process of data connection establishment by instrumenting key functions (*e.g.,* `DataProfileManager.getApnSettingForNetworkRequest()`, `PhoneInterfaceManager.setDataEnabled()`, and `DataNetwork.setupDataCall()`) along the way in CrowdEye in 2023. In detail, our monitoring service acts as a centralized listener for these concerned functions, recording their execution time, parameters, and return values in real time. When the monitoring service captures a Data_Setup_Error failure, it integrates these traces with the aforementioned radio- and BS-related information, forming a comprehensive failure snapshot. In this way, we can capture detailed information of the establishment (*e.g.,* function

[3]For some CDMA BSes, System Identity (SID), Network Identity (NID), and Base Station Identity (BID) are recorded instead of MNC, LAC, and CID.

call stacks, state machine transitions, and message handling results), facilitating the in-depth postmortem analysis.

Third, we note that the existing Data_Stall detection mechanism in Android cannot provide an accurate measurement of a Data_Stall failure's recovery time, given its fixed detection time (as long as one minute). According to our observations (detailed later in § III-A), in most (>80%) cases a Data_Stall failure lasts for <300 seconds, so the incurred measurement error is non-trivial relative to the Data_Stall failure's recovery time. Also, detection results of this mechanism may contain false positives for lack of crucial knowledge regarding the current states of network stack and Internet connectivity.

To address these issues, we build a network-state probing component in CrowdEye. Once a suspicious Data_Stall failure is detected, this component checks the states of network stack and Internet connectivity by simultaneously sending an ICMP message to the local IP address (127.0.0.1), as well as sending an ICMP message and a DNS query (for our dedicated test server's domain name) to each of the user device's assigned DNS server(s). If the ICMP message intended for the local IP address reaches a timeout (empirically configured as one second as suggested by the ICMP protocol [16]), we know that the problem lies at the system side rather than the network side (hence a false positive case). In practice, such false positives typically involve erroneous firewall configurations, problematic proxy settings, and modem driver failures. Otherwise, if all the DNS queries reach a timeout (empirically configured as five seconds as suggested by the DNS protocol [17]), we know the problem lies at the network side. However, if timeouts only occur to the DNS queries but not to the ICMP messages sent to the DNS servers, we figure out that the problem is induced by the unavailability of DNS resolution service (also a false positive case).

The above probing process needs at most five seconds, given the one-second timeout for the ICMP message deliveries and five-second timeout for the DNS queries. If all the probing results are successful, indicating that Data_Stall has been fixed, we will calculate the total recovery time by adding up the durations of all the previous probing processes (since the beginning of this Data_Stall failure); otherwise, we will initiate a new probing process. Thus, our measurement error is at most five seconds ($\ll$ 1 minute). Furthermore, to avoid excessive network overhead, if a Data_Stall failure lasts for longer than 1200 seconds (in fewer than 10% cases, as illustrated later in § III-A), we will multiplicatively increase the time interval between probing processes by a factor of two to balance the incurred error and overhead. Finally, if the time interval grows to longer than one minute, we will revert to Android's original detection mechanism to estimate the recovery time of Data_Stall failures.

Moreover, we observe that vanilla Android does not record or provide the recovery times of Data_Setup_Error and Out_of_Service failures. To address this limitation, we obtain them through lightweight instrumentation of corresponding state machines. Specifically, for Data_Setup_Error failures, we track the cellular connection state machine, recording the timestamp when the failure occurs and when the state machine first transitions to the active state (*i.e.,* when the failure recovers). In this way, the recovery time can be deemed as the time interval between the two timestamps, with a negligible error ($\ll$ 1 second) introduced by the recording process. Similarly, for Out_of_Service failures, we monitor the cellular service state machine and determine the recovery time as the interval between the failure occurrence and the state machine's reentry into the In_Service state.

All in all, our modifications to Android involve system-level information logging (primarily through existing interfaces), key function tracing, and lightweight network probing activities. For even a low-end Android phone at the moment, CrowdEye only incurs negligible CPU utilization (<2%), memory usage (<40 KB), and storage space (<0.1 MB); the network usage per month is <0.1 MB. Note that here the CPU utilization is measured by the portion of additional CPU overhead induced by our monitoring infrastructure within the duration of detected failures, rather than during the entire measurement process. As a matter of fact, in daily usages without cellular failures, our monitoring infrastructure is dormant at the client side and thus does not incur additional CPU overhead.

On the other hand, we do notice that for a small fraction (<1%) of user devices, they experience as many as 40,000+ failures (as to a single user) in a month. Even so, the incurred CPU, memory, and storage overheads are still acceptable: <8% CPU utilization, <2 MB of memory usage, and <20 MB of storage space; the network usage per month can reach 20 MB, so the recorded data are uploaded only when there is WiFi connectivity.

Finally, the network overhead incurred by our measurement is fairly low even in a cumulative sense. For all the users that participated in our study, the aggregate network overhead per second on the entire cellular networks of the three involved ISPs was below 0.5 MB, and thus had negligible influence on the performance, availability, and reliability of the studied cellular networks.

C. Large-Scale Deployment

With the continuous monitoring infrastructure, in Dec. 2019 and Sep. 2023 we invited all the users of Xiaomi to participate in our measurement study of cellular reliability by installing CrowdEye on their phones. Note that the installation is a lightweight update that will not affect their installed apps, existing data, and OS version. Eventually, 123,292,648 users opted in and collected data for us for a total of 16 months (01/01/2020—08/30/2020 and 10/01/2023—06/30/2024). All data are compressed and uploaded to our backend server for centralized analysis.

Ethical Concerns. All analysis tasks in this study comply with the agreement established between Xiaomi and its users. The users who participated in the study opted-in as volunteers with informed consent, the analysis was conducted under a well-established IRB, and no personally identifiable information (*e.g.,* phone number, IMEI, and IMSI) was collected. We never (and have no way to) link collected information to users' true identities.

D. Analysis Pipeline

We analyze the collected data through a two-stage process. First, we conduct a macro-level analysis and calculate key metrics including prevalence, frequency, and recovery time for cellular failures. Through longitudinal comparisons, we uncover changes in failure severity over the past five years and identify different evolution trends across the three types of failures. Second, we examine the correlations between multiple influencing factors and cellular failures from two perspectives. From the user device perspective, we assess how hardware configurations and Android versions affect the prevalence and frequency of distinct types of failures and how these

TABLE I

HARDWARE CONFIGURATIONS OF 34 PHONE MODELS STUDIED IN 2020, GENERALLY ORDERED FROM LOW-END TO HIGH-END. THE RIGHTMOST 5 COLUMNS CORRESPOND TO THE PHONE'S 5G CAPABILITY (5G), ANDROID VERSION (VERSION), USER PERCENTAGE (USERS),FRACTION OF DEVICES THAT EXPERIENCE AT LEAST 1 CELLULAR FAILURE (PREVALENCE), AND AVG. NUMBER OF CELLULAR FAILURES EXPERIENCED PER PHONE (FREQUENCY) DURING OUR MEASUREMENT, RESPECTIVELY

Model	CPU	Memory	Storage	5G	Version	Users	Prevalence	Frequency
1-1	1.8 GHz	2 GB	16 GB	–	10.0	2.71%	28%	35.9
1-2	1.95 GHz	2 GB	16 GB	–	9.0	3.02%	13%	23.8
1-3	2 GHz	2 GB	16 GB	–	9.0	7.31%	10%	13.8
1-4	2 GHz	3 GB	32 GB	–	9.0	3.90%	19%	22.4
1-5	2 GHz	3 GB	32 GB	–	9.0	2.85%	21%	28.2
1-6	2 GHz	3 GB	32 GB	–	10.0	4.33%	4%	5.3
1-7	2 GHz	3 GB	32 GB	–	10.0	1.44%	5%	6.4
1-8	2 GHz	3 GB	32 GB	–	9.0	4.07%	0.15%	2.3
1-9	2 GHz	3 GB	32 GB	–	10.0	5.47%	2%	2.6
1-10	2.2 GHz	4 GB	32 GB	–	9.0	5.78%	27%	36.8
1-11	1.8 GHz	4 GB	64 GB	–	10.0	1.18%	25%	28.5
1-12	2 GHz	4 GB	64 GB	–	10.0	1.44%	33%	43.5
1-13	2.05 GHz	6 GB	64 GB	–	10.0	5.39%	26%	18.7
1-14	2.2 GHz	6 GB	64 GB	–	9.0	2.98%	15%	17.9
1-15	2.2 GHz	4 GB	128 GB	–	10.0	3.98%	25%	26.7
1-16	2.2 GHz	4 GB	128 GB	–	10.0	3.02%	19%	28.0
1-17	2.2 GHz	6 GB	64 GB	–	10.0	1.09%	28%	48.4
1-18	2.2 GHz	6 GB	64 GB	–	10.0	0.26%	13%	38.8
1-19	2.2 GHz	6 GB	64 GB	–	10.0	1.31%	24%	44.8
1-20	2.2 GHz	6 GB	64 GB	–	10.0	0.57%	21%	33.0
1-21	2.2 GHz	6 GB	64 GB	–	10.0	2.80%	36%	46.6
1-22	2.2 GHz	6 GB	128 GB	–	9.0	0.44%	38%	61.1
1-23	2.4 GHz	6 GB	64 GB	YES	10.0	0.84%	44%	49.6
1-24	2.4 GHz	6 GB	128 GB	YES	10.0	3.25%	37%	38.0
1-25	2.45 GHz	6 GB	64 GB	–	9.0	4.99%	14%	19.6
1-26	2.45 GHz	6 GB	64 GB	–	9.0	2.15%	17%	24.6
1-27	2.8 GHz	6 GB	64 GB	–	10.0	1.84%	22%	54.2
1-28	2.8 GHz	6 GB	64 GB	–	10.0	7.14%	28%	58.1
1-29	2.8 GHz	6 GB	64 GB	–	10.0	1.31%	30%	65.1
1-30	2.8 GHz	6 GB	128 GB	–	10.0	1.01%	30%	90.2
1-31	2.84 GHz	6 GB	64 GB	–	10.0	1.88%	28%	61.7
1-32	2.84 GHz	6 GB	64 GB	–	10.0	3.63%	29%	57.8
1-33	2.84 GHz	8 GB	128 GB	YES	10.0	4.78%	32%	70.9
1-34	2.84 GHz	8 GB	256 GB	YES	10.0	1.84%	25%	79.3

effects evolve over time. We also explore the Android's failure recovery mechanisms to understand recovery time variations. From the ISP and BS perspective, we delve deep into how geographic locations, ISPs, radio access technologies, and signal strengths impact cellular failures. We further reveal the underlying causes of these impacts by investigating ISP infrastructure and BS deployment.

III. MEASUREMENT RESULTS

In this section, we first present the general characteristics of our collected measurement data (§ III-A). Then, to systematically describe cellular failures and their underlying causes in a more readable manner, we present our data analysis results from the viewpoints of user decives (§ III-B) and ISPs/BSes (§ III-C), respectively.

A. General Statistics

With the crowdsourcing help from 123,292,648 CrowdEye user devices with 70 different phone models (34 models involved in 01/01/2020–08/30/2020 as listed in Table I and 36 models involved in 10/01/2023–06/30/2024 as listed in Table II), we record the system-level traces with regard to 4,104,056,765 cellular failures, involving 32,847,235 user devices, 3 mobile ISPs and 8,130,797 base stations. To the best of our knowledge, this is so far the largest dataset regarding cellular failures in the wild.

First of all, we are concerned with the prevalence and frequency of cellular failures. Our measurement results reveal that despite substantial advances in cellular technologies over the past five years, the severity of cellular reliability problems has worsened overall. According to the coarse-grained dataset collected during 2020–2024, both the prevalence and

TABLE II

HARDWARE CONFIGURATIONS AND GENERAL STATISTICS OF OUR STUDIED 36 PHONE MODELS IN 2023/2024, GENERALLY ORDERED FROM LOW-END TO HIGH-END. THE COLUMNS ARE THE SAME WITH TABLE I

Model	CPU	Memory	Storage	5G	Version	Users	Prevalence	Frequency
2-1	2 GHz	4 GB	64 GB	–	11.0	3.78%	12%	4.3
2-2	2 GHz	4 GB	64 GB	–	11.0	2.19%	11%	4.1
2-3	2 GHz	4 GB	64 GB	–	12.0	1.01%	10%	4.2
2-4	2 GHz	4 GB	64 GB	–	14.0	1.65%	16%	13.5
2-5	2 GHz	4 GB	128 GB	–	12.0	2.05%	7%	2.5
2-6	2.2 GHz	4 GB	128 GB	YES	13.0	1.40%	21%	9.9
2-7	2 GHz	6 GB	128 GB	YES	14.0	2.82%	33%	34.0
2-8	2.2 GHz	6 GB	128 GB	YES	12.0	2.22%	29%	17.3
2-9	2.2 GHz	6 GB	128 GB	YES	14.0	2.13%	44%	49.3
2-10	2.2 GHz	6 GB	128 GB	YES	14.0	1.26%	30%	45.5
2-11	2.3 GHz	6 GB	128 GB	YES	13.0	4.25%	23%	9.9
2-12	2.4 GHz	6 GB	128 GB	YES	13.0	6.52%	22%	26.2
2-13	2.4 GHz	8 GB	128 GB	YES	12.0	1.01%	25%	15.7
2-14	2.4 GHz	8 GB	128 GB	YES	13.0	1.71%	29%	13.0
2-15	2.5 GHz	8 GB	128 GB	YES	13.0	3.67%	32%	21.6
2-16	2.4 GHz	8 GB	256 GB	YES	13.0	2.96%	15%	15.8
2-17	2.4 GHz	12 GB	256 GB	YES	14.0	2.14%	40%	55.6
2-18	2.6 GHz	12 GB	256 GB	YES	14.0	2.03%	31%	44.0
2-19	2.84 GHz	8 GB	256 GB	YES	13.0	1.56%	30%	41.4
2-20	2.85 GHz	8 GB	256 GB	YES	14.0	3.34%	26%	25.8
2-21	2.91 GHz	12 GB	256 GB	YES	14.0	4.26%	44%	36.3
2-22	3.2 GHz	8 GB	256 GB	YES	14.0	5.57%	30%	26.4
2-23	3.2 GHz	8 GB	256 GB	YES	13.0	2.64%	31%	27.5
2-24	3 GHz	12 GB	128 GB	YES	13.0	1.28%	26%	19.3
2-25	2.85 GHz	12 GB	256 GB	YES	14.0	3.02%	27%	33.2
2-26	2.8 GHz	12 GB	512 GB	YES	14.0	1.14%	39%	74.1
2-27	3.0 GHz	12 GB	256 GB	YES	14.0	4.56%	41%	40.2
2-28	3.05 GHz	12 GB	256 GB	YES	14.0	0.98%	31%	37.8
2-29	3.1 GHz	12 GB	256 GB	YES	14.0	1.62%	33%	65.0
2-30	3.2 GHz	12 GB	256 GB	YES	14.0	4.23%	48%	51.0
2-31	3.1 GHz	12 GB	512 GB	YES	14.0	0.98%	38%	70.4
2-32	3.35 GHz	12 GB	512 GB	YES	14.0	1.52%	39%	76.0
2-33	3.35 GHz	12 GB	512 GB	YES	14.0	1.51%	41%	63.6
2-34	3.2 GHz	16 GB	512 GB	YES	14.0	4.23%	50%	51.4
2-35	3.3 GHz	16 GB	512 GB	YES	14.0	4.04%	55%	73.2
2-36	3.3 GHz	16 GB	1 TB	YES	14.0	1.95%	55%	85.0

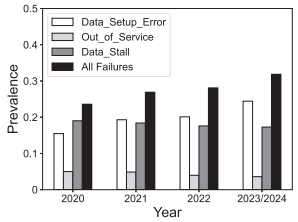


Fig. 2. Prevalence of cellular failures over time.

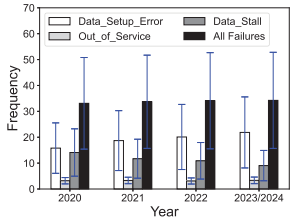


Fig. 3. Frequency of cellular failures over time.

frequency exhibit a consistent upward trend over the five-year period, as illustrated in Figure 2 and Figure 3. In detail, the prevalence of cellular failures grows by 39% (23% $\rightarrow$ 32%) and the frequency grows by 3% (33 $\rightarrow$ 34).

To gain deeper insights, we further conduct a more detailed analysis based on the fine-grained dataset collected in 2020 and 2023/2024. Regarding results in 2020, as shown in Table I and Figure 4, on different models of phones the prevalence varies from 0.15% to 45% and averages at 23%. More notably, as many as 33 cellular failures occur to a phone on average during our 8-month measurement (see Figure 5), and the average number of cellular failures happening to a specific model varies from 2.3 to 90.2 (see Table I). As for the results in 2023/2024, on different models the prevalence of cellular

Fig. 4. Prevalence of cellular failures on each model of phones in 2020.

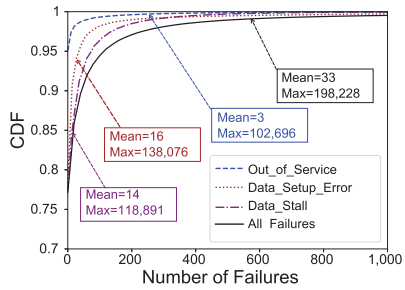


Fig. 5. Number of cellular failures happening to a single phone in 2020.

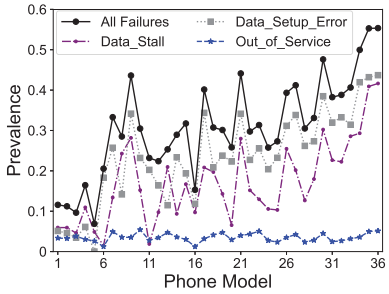


Fig. 6. Prevalence of cellular failures on each model of phones in 2023/2024.

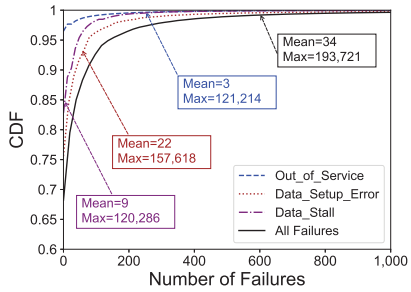


Fig. 7. Number of cellular failures happening to a single phone in 2023/2024.

failures ranges from 7% to 55%, with an average of 32% (see Table II and Figure 6), and the frequency ranges from 2.5 to 85.0, with an average of 34 (see Table II and Figure 7).

Specifically, we observe distinct evolution trends of different types of cellular failures. Among the 4.10 billion cellular failures, the vast majority (>99%) include Data_Setup_Error, Out_of_Service, and Data_Stall events. The remainder (<1%) are mainly related to the traditional short message and voice call services that are less frequently used today (*e.g.,* short message sending failure tagged by Android as RIL_SMS_SEND_FAIL_RETRY [18]), whose functions and enabling techniques have been stable for nearly 20 years.

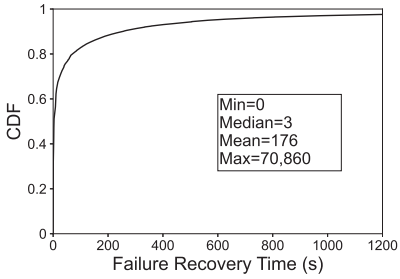


Fig. 8. Recovery time of Data_Stall failures in 2020.

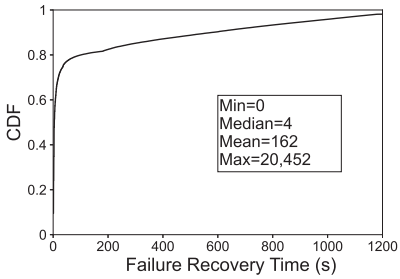


Fig. 9. Recovery time of Data_Setup_Error failures in 2023/2024.

As shown in Figure 3 and Figure 5, an average of 16 Data_Setup_Error, 14 Data_Stall, and 3 Out_of_Service events occur to a single phone in 2020. After four years, as shown in Figure 3 and Figure 7, the frequency of Data_Stall events declines to 9, and the frequency of Out_of_Service events remains at 3, while the frequency of Data_Setup_Error events increases to 22. Meanwhile, as shown in Figure 2, the prevalence of Data_Stall and Out_of_Service events drops by 11% (19% → 17%) and 40% (5% → 3%), respectively, whereas the prevalence of Data_Setup_Error events increases by 60% (15% → 24%). We will demystify the reasons for the distinct results in § III-B.

Apart from prevalence and frequency, we are also concerned with the failure recovery time, which mostly comes from Data_Stall in 2020 but Data_Setup_Error in 2023/2024. In 2020, as shown in Figure 8, a Data_Stall failure lasts for as long as 176 seconds (=2.9 minutes) on average. In 2023/2024, as shown in Figure 9, Data_Setup_Error failures take an average of 162 seconds (=2.7 minutes) to fix. We will explain the reasons for the unsatisfactory recovery time in both cases in § III-B. Moreover, we notice that the maximum recovery time for Data_Stall failures can reach 70,860 seconds (=19.7 hours), and for Data_Setup_Error failures it can reach 20,452 seconds (=5.68 hours). Closer examination reveals that such failures, which take a long time to recover, typically occur in remote regions such as mountain and offshore areas, where the BSes have been long neglected and in disrepair.

B. User Device Landscape

In this part, we go deeper into the internals of user devices with respect to their hardware, software and OS components that may pose impact on cellular failures, especially those with regard to the emerging technologies.

Hardware Configuration. Intuitively, using higher-end cell phones should help to mitigate cellular failures as they usually imply the adoption of more reliable and/or powerful hardware components. However, our measurement results

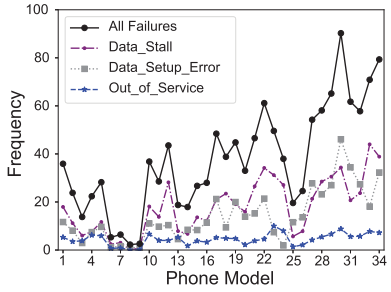


Fig. 10. Frequency of cellular failures on each model of phones in 2020.

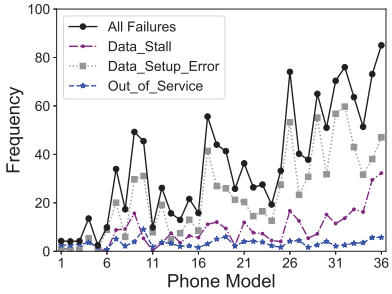


Fig. 11. Frequency of cellular failures on each model of phones in 2023/2024.

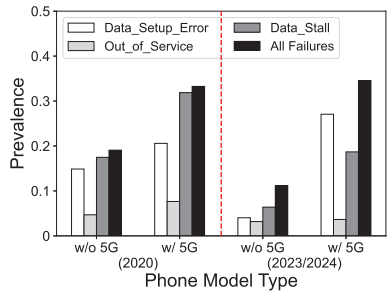


Fig. 12. Prevalence of cellular failures on models w/ and w/o the 5G module in 2020 and 2023/2024.

generally indicate the opposite: as shown in Figure 4, 6, 10, and 11, both the prevalence and frequency of cellular failures tend to increase with better hardware configurations. To demystify this, we examine the correlation between each feature (in Table I and Table II) and the prevalence/frequency of cellular failures, finding that two features, *i.e.,* 5G capability and Android version, have significant influence on the occurrence of cellular failures, while further analysis reveals that the similar correlations across two periods (in 2020 and in 2023/2024) stem from very different underlying reasons.

Since 5G just sprang up in 2020, only four out of the 34 phone models are equipped with 5G modems, in which Models 23 and 24 have typical hardware configurations at the moment while Models 33 and 34 have the best. As illustrated in Figure 12 and Figure 13, both the prevalence and frequency of cellular failures on 5G phones are higher than those on non-5G phones. This suggests that the emerging 5G communication modules have negative impact on the reliability of cellular connections, probably due to the high workload they inflict on the network stack and system kernel of Android for processing large volumes of inbound/outbound data in short time, as well as their relatively immature development stage at the moment.

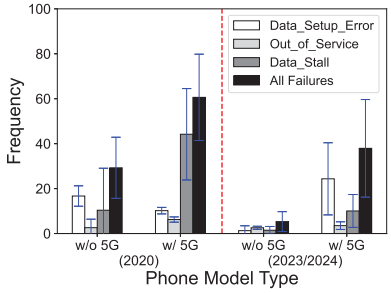


Fig. 13. Frequency of cellular failures on models w/ and w/o the 5G module in 2020 and 2023/2024.

Four years later, 5G phones have dominated the market (31 of 36 phone models). Despite the evolution of 5G technologies, cellular failures still happen more prevalently and frequently on 5G phones than on non-5G phones (see Figure 12 and Figure 13). We uncover the root cause as a combination of the advancements in 4G and the potential side effects of 5G development. Specifically, we observe that improvements in 4G infrastructure along with the decrease in 4G users significantly alleviate resource contention and improve spectral efficiency, leading to a remarkable decrease in the prevalence and frequency of failures on non-5G phones.

In particular, the number of 4G base stations (BSes) has increased by 24% (from 5.7M to 7.1M) between 2020 and 2024. Additionally, ISPs have adopted multiple LTE-Advanced technologies during the past few years. For instance, carrier aggregation [19] enables devices to utilize multiple frequency bands simultaneously, thereby mitigating network congestion; enhanced inter-cell interference coordination [20] reduces interference between macro and small cells, particularly improving cellular reliability in dense urban areas; and coordinated multi-point [21] allows multiple BSes to jointly transmit data, minimizing cellular failures during BS handovers.

More notably, over the past five years, the three types of failures on 5G phones exhibit different statistical trends: the severity of Out_of_Service and Data_Stall events on 5G phones has largely eased, closely resembling the situation of non-5G phones in 2020, while the situation of Data_Setup_Error events has dramatically deteriorated. We attribute the former encouraging fact to the substantial investments made by China's ISPs in 5G infrastructure, including an enormous increase in the number of 5G BSes (by 317%) to improve signal coverage, the optimization of the Massive MIMO technology for indoor environments to mitigate signal interference [7], *etc.*.

Disappointingly, advances in 5G technologies have an unexpected negative impact on Data_Setup_Error events, far outweighing the positive impact observed for Data_Stall and Out_of_Service events. As most 5G BSes transition from NSA (Non-Standalone) mode to SA (Standalone) mode, the 4G/5G dual connectivity mechanism is rarely used now, requiring user devices to establish new connections switching between 4G and 5G. Additionally, phones experience more frequent handovers when connected to 5G than 4G, due to the smaller coverage area of 5G BSes. With the large-scale adoption of 5G in 2023/2024, devices are now staying on 5G networks for a much longer time than in 2020. Consequently, the more frequent connection setups and BS handovers lead to a higher probability of encountering Data_Setup_Error failures.

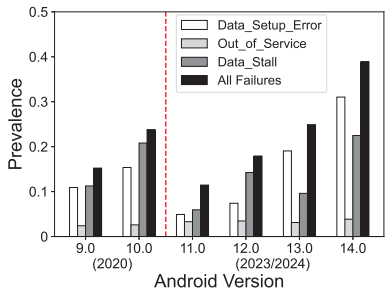


Fig. 14. Prevalence of cellular failures for different Android versions.

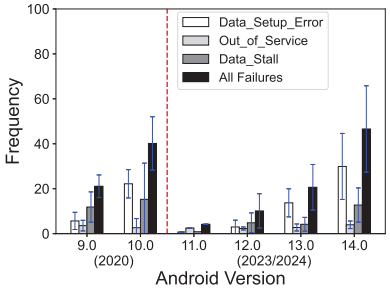


Fig. 15. Frequency of cellular failures for different Android versions.

Android Version. The Android versions of the studied 70 phone models include Android 9-14, released between Aug. 2018 and Oct. 2023. In detail, the 34 models involved in the former eight months use Android 9/10, and the 36 models involved in the latter eight months use Android 11-14.

In our preliminary work [8], we note that as Android evolves from version 9 to 10, a number of new functions and performance improvements have been implemented [22]. Although these updates are supposed to benefit users, we unexpectedly find that both the prevalence and frequency of cellular failures on Android 10 phones are higher than those on Android 9 phones, as demonstrated in Figure 14 and Figure 15. We attribute this primarily to the better stability and robustness of Android 9, as Android 10 was still undergoing constant fixes and patches during our measurement.[4]

Adapting to the emerging 5G techniques, Android 10 added considerable dedicated services, network functions, and programming APIs [23]. While these novelties can enable various high-demanding applications such as UHD video streaming and AR/VR [1], they inevitably bring defects, risks, and vulnerabilities to the cellular connection management modules of Android from the perspective of software engineering.

In particular, we note that in the RAT (radio access technology) selection policy upgraded in Android 10, 5G is blindly preferred to the other RATs, probably aiming to maximize the potential benefit of 5G, especially its remarkably higher peak bandwidth. Nevertheless, this policy could incur severe cellular failures. For example, when a user device can establish either a 5G connection with low received signal strength (RSS) or a 4G connection with high RSS, the preferred usage of 5G might bring rather unstable cellular performance and even cellular failures, although the co-existing 4G connection might have better cellular performance. Worse still, this example is not a rare case but happens frequently in our everyday life, thus leading to a great number of cellular failures on 5G phones.

Similar to the situation in 2020, failures occur much less prevalently and frequently on phones running lower Android versions (*i.e.,* Android 11/12) in 2023/2024, as more than three years have passed since their official release, allowing them to become highly stable. Still, higher Android versions (*i.e.,* Android 13/14) exacerbate cellular failure issues, and Data_Setup_Error failures worsen the most. In addition to the lack of robustness of these versions, we uncover that Android 14 inadvertently introduces a new design defect when updating the cellular connection management, leading to considerable misjudged Data_Setup_Error failures. We will explain this later in this section.

Data_Stall Recovery. As mentioned in § III-A, Data_Stall failures account for the most of the recovery time in 2020 and last for as long as 2.9 minutes on average, thus posing broad and disruptive impact on user experiences. Recall that in Android, when there have been over 10 outbound TCP segments but not a single inbound TCP segment during the last minute, a Data_Stall event happens [6]. To tackle the problem, when a Data_Stall event is detected, Android launches the *three-stage progressive mechanism* that sequentially tries light (cleaning up and restarting the current connection), moderate (re-registering into the network), and heavy (restarting the radio component) recovery techniques. Before carrying out each of the above operations, Android would wait for one minute to watch whether the problem has already been fixed.

In practice, we observe that this mechanism is quite effective— once executed, even the first-stage, lightweight operation can fix the problem in 75% cases. Nevertheless, our measurement shows that this mechanism is overly conservative and thus rather time-consuming. In fact, for the majority of Data_Stall events, the user device can automatically recover them in less time, as illustrated in Figure 8. For example, 60% Data_Stall failures are automatically fixed in just 10 seconds. Also, we notice that the victim user would manually reset the data connection within ∼30 seconds (according to our sampling user survey). Therefore, the one-minute "probation" adopted by Android is unnecessarily long, rendering the recovery mechanism to be neither efficient nor user-friendly.

Data_Setup_Error Recovery. In 2023/2024, the failure recovery time mostly comes from Data_Setup_Error, due to the aforementioned design defect introduced in Android 14. As illustrated in Figure 9, the average recovery time of Data_Setup_Error failures reaches 162 seconds (=2.7 minutes), with 18.4% of cases taking over 180 seconds to recover, significantly affecting user experiences. A closer analysis reveals that the majority (>60%) of these prolonged reconnections (>180s) happen when an app requests to start a network session (with no cellular connection established) and then quickly cancels the request, causing the connection to terminate during its setup. This is mainly due to rapid changes in app behavior, such as being opened and then swiftly closed, or enabling cellular

[4]Given that 5G phone models can only run Android 10 (since Android 9 does not support 5G) in 2020 and only Android 12 runs on both 5G and non-5G phones among the involved phones (due to the wide adoption of 5G) in 2023/2024, to make an independent analysis and a fair comparison, when comparing the prevalence and frequency of cellular failures on 5G and non-5G models earlier in this section, we should only select models running Android 10 and 12. Similarly, for a fair comparison among Android versions regarding their impacts on cellular failures, we should only compare the phone models running Android 9 with the non-5G models running Android 10 and 12, or compare among 5G models running Android 10, 12, 13, and 14. Since the corresponding fair-comparison results are similar to those shown in Figure 12, Figure 13, Figure 14, and Figure 15, we choose not to plot additional figures to demonstrate them.

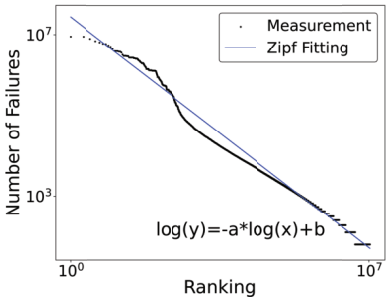


Fig. 16. BS Ranking by the experienced number of cellular failures.

data acceleration while the phone is connected to WiFi and soon disabling it, which are typically caused by shifts in user actions.

Unfortunately, in Android 14, the system misregards all cellular connections terminated during their setups (named *aborted connections*) as Data_Setup_Error failures, and will no longer perform reconnections (NO_RETRY_FAILURE) even if new network sessions require them. This design is primarily aimed at conserving resources and reducing power consumption, as the system assumes that retrying aborted connections would still fail, which is often the case (*e.g.,* no available gateway for the APN) but not always (*e.g.,* the aforementioned unusual app behavior). The premature failure judgment introduces a significant number of misjudged Data_Setup_Error failures, worsening cellular reliability issues on phones running Android 14 as mentioned before. It also leads to considerable overdue cellular reconnections, as in the false positive NO_RETRY_FAILURE case, the undesirable situation will persist until the user manually resets the cellular connection, leading to the very long recovery time.

C. ISP and Base Station Landscape

As mentioned in § III-A, our measurement captures a total of 4.10 billion cellular failure events with regard to 8.1M BSes. In this part, we look at cellular failures from the viewpoint of ISPs and BSes, by considering the geographic locations, ISP discrepancies, radio access technologies, and signal strengths.

Geographic Location. By ranking the involved BSes with their experienced number of cellular failures (in descending order), we observe a Zipf-like [24] skewed distribution as depicted in Figure 16 (where $a = 0.82$ and $b = 17.12$). The median and average numbers are 1 and 444 respectively, while the maximum number reaches 8,941,860. We then delve into the 10,000 top ranking BSes, and find that they are mostly located in crowded urban areas. Hence, they are confronted with essentially more ambient interferences and heavier cellular access workloads, both of which aggravate the problems.

ISP Discrepancy. The BSes involved in our study belong to three mobile ISPs, referred to as ISP-A, ISP-B, and ISP-C. Specifically, 49.6%, 23.0%, and 27.4% BSes belong to ISP-A, ISP-B, and ISP-C, respectively. From Figure 17, we can see that cellular failures occur more prevalently (30.6%) on ISP-B's users than on ISP-A's (25.1%) and ISP-C's (24.3%), mainly due to the inferior signal coverage of ISP-B's BSes. In detail, while ISP-B's BSes are a bit more than ISP-C's, to our knowledge most of ISP-B's BSes have a smaller signal coverage because they usually use a higher radio frequency. The situation is similar in terms of frequency, as shown in Figure 18.

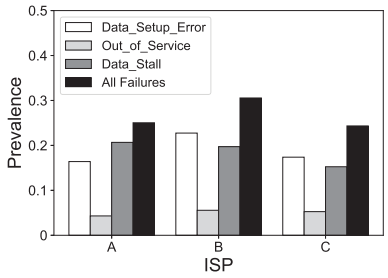


Fig. 17. Prevalence of cellular failures for different ISPs' users.

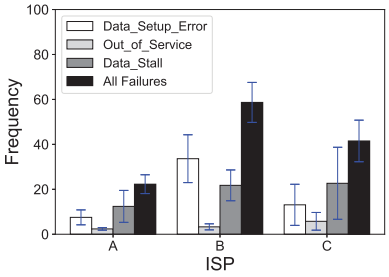


Fig. 18. Frequency of cellular failures for different ISPs' users.

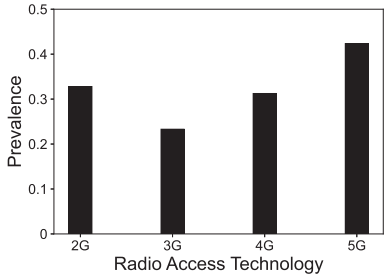


Fig. 19. Prevalence of cellular failures on 2G, 3G, 4G and 5G BSes.

Another important factor contributing to the reliability discrepancy across ISPs is the timing of refarming low-band spectrum for 5G deployment to support broad-area coverage. Specifically, ISP-A refarmed its 700 MHz band for 5G in 2021, and ISP-C started utilizing its 900 MHz band for 5G in late 2022. In contrast, ISP-B only received approval to refarm its 800 MHz band in mid-2023. As a result, during our measurement in 2023–2024, ISP-A and ISP-C were able to provide better 5G signal coverage in remote areas using low-band 5G BSes, while ISP-B's low-band 5G coverage was not yet widely available. The poorer 5G signal coverage in such regions results in more reliability issues for users of ISP-B.

Radio Access Technology (RAT). Among the involved BSes, 16.3%, 7.7%, 48.6%, and 34.8% support 2G, 3G, 4G, and 5G access, respectively. Here the four percentages add up to more than 100% because some BSes simultaneously support multiple RATs. While both the number and overall signal coverage of 3G BSes are smaller than those of 2G, 4G or 5G BSes, we observe that the prevalence of cellular failures on 3G BSes is lower than that on 2G, 4G or 5G BSes, as indicated in Figure 19. This is probably because 3G access is usually not favored by user devices when 4G/5G access is available, and the signal coverage of 3G is much worse than that of 2G when 4G/5G access is unavailable. In other words, 3G networks currently face less resource contention from the users (*i.e.,* relatively "idle") and thus manifest fewer cellular failures.

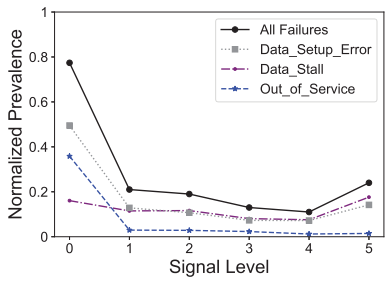


Fig. 20. Normalized prevalence (or simply likelihood) of cellular failures for different signal levels.

Received Signal Strength (RSS). In common sense, RSS is a key factor that impacts the reliability of cellular service, and a higher RSS level (or simply signal level) is usually expected to come with better service reliability. However, our measurement results in Figure 20 refute this common understanding by revealing that excellent RSS seems to increase the likelihood of cellular failures. As the signal level increases from 0 (the worst) to 4 (good), the *normalized prevalence* of cellular failures monotonously decreases. Here the "normalized" prevalence denotes the regular prevalence (as explained and computed in Table I) divided by the total time during which the device is connected to a BS. We have to use the normalized prevalence because the durations of different signal levels can differ greatly from each other; in order to account for this discrepancy, we divide each prevalence by its average duration to achieve a fair comparison (the duration data are also provided by Xiaomi based on a nationwide measurement). On the other hand, when the signal level goes to 5 (excellent), the normalized prevalence of cellular failures suddenly grows to larger than each case of level 1 to 4.

To demystify this counter-intuitive phenomenon, we carefully examine a series of in-situ information corresponding to such excellent-RSS cellular failures, including the BS location, serving ISP, RAT, error code, *etc.*. As a result, we find that this phenomenon usually happens around public transport hubs, where the nearby BSes tend to be problematic simultaneously, regardless of the serving ISPs and RATs. Actually, ISPs often choose to densely deploy their BSes around a public transport hub so as to better cope with the large volume of human traffic. Owing to this special BS deployment strategy, the nearby user devices can typically have excellent (level 5) RSS. On the other hand, such densely deployed BSes could bring non-trivial signal interferences to each other [25]. In fact, the three ISPs' radio frequency bands are fairly close to each other (more specifically, with the median frequency being ISP-B's $>$ ISP-C's $>$ ISP-A's) and even occasionally overlap one another, thus leading to potentially significant adjacent-channel interference. More importantly, dense BS deployment could make LTE mobility management highly complicated and challenging [9], [26], causing frequent cellular failures tagged with EMM_ACCESS_BARRED, INVALID_EMM_STATE, *etc.* [11]. This is especially the case when multiple ISPs adopt similar deployment strategies without coordinations.

IV. Enhancements

Our multifold findings on cellular data connections failures in § III drive us to rethink the current techniques widely employed by cell phones, mobile OSes, and ISPs with respect to their influence on the reliability of cellular connections. Accordingly, in this section we provide insightful guidance for addressing various cellular failures at scale (§ IV-A), as well as practical enhancements that have registered large-scale deployment and yielded real-world impact (§ IV-B).

A. Guidelines in Principle

In § III we have revealed a variety of technical and business issues that could lead to or aggravate cellular failures. As elucidated in § III-B, cellular failures on 5G phones are more prevalent and frequent than the other phones (without 5G capability), even after the full development of 5G technologies in 2023/2024, most probably owing to the unexpected negative impact on Data_Setup_Error events brought by upgrading 5G BSes to SA mode. Thus, we suggest that mobile phone vendors be cautious with the rapidly emerging 5G enhancements. More specifically, we encourage the vendors to comprehensively evaluate the impact of new 5G technologies on cellular reliability in advance, so as to provide more reliable phone models by proactively addressing potential negative effects.

Also in § III-B, we note that phones with newer Android versions are more subject to cellular failures than phones with older ones, due to the typically worse stability and robustness of newly released OSes, in particular, the blindly prioritized usage of 5G connection over 4G/3G/2G connections since Android 10 and the premature failure judgment in Android 14. Hence, we propose that for the vendors, sufficient testing for new characteristics (*e.g.,* the 4G/5G switching policy and the cellular connection management) should be carried out before pushing a new OS to certain phone models.

For mobile ISPs, we unravel in § III-C that due to less workload and radio resource contention from user devices, 3G BSes are less subject to cellular failures than 2G, 4G and 5G BSes. Thereby, ISPs may consider making better use of these relatively "idle" infrastructure components to alleviate the burdens on busy 2G/4G/5G BSes. Further, our in-depth investigation into the correlation between signal level (or RSS) and cellular failures uncovers that due to ISPs' dense BS deployment around public transport hubs, cellular failures can be rather severe despite very high signal levels, for reasons of intensive signal interferences and highly complex mobility management requirements. Therefore, we advise ISPs to carefully control their BS deployment density in such areas. Finally, we advocate the recent campaign of cross-ISP infrastructure sharing [27], which aims to coordinate the BS deployment among different ISPs for more efficient utilization of radio infrastructure resources and thus can help mitigate cellular failures.

B. Real-World Practices

Apart from the above heuristic guidelines for the broad community, by collaborating with Xiaomi, we have practically explored optimization opportunities with respect to the aggressive 5G usage policy (cf. § III-B) during RAT transition, the conservative Data_Stall recovery mechanism (cf. § III-B) and the premature failure judgment (cf. § III-B) in vanilla Android. Based on critical insights obtained from our measurement study, below we first devise a stability-compatible RAT transition mechanism to make cellular connections more reliable, and then leverage the time-inhomogeneous Markov process (TIMP) model to accelerate the Data_Stall recovery, and finally apply meticulous failure judgment to speed up the Data_Setup_Error recovery. All efforts have been put into practice and produced promising results.

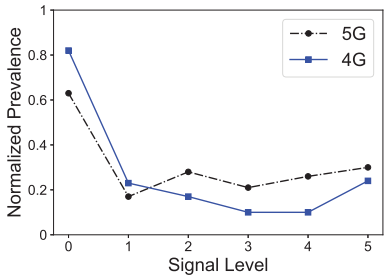


Fig. 21. Normalized prevalence of cellular failures for different 4G/5G signal levels.

Stability-Compatible RAT Transition. As introduced in § III-B, we observe that Android 10 adopts a quite aggressive strategy to prioritize the usage of 5G connections during RAT transition, which pays little attention to the cellular network status (*e.g.,* signal level) and thus leads to a large number of cellular failures. In fact, as shown in Figure 21, the normalized prevalence (or simply likelihood) of cellular failures varies significantly across different signal levels under 4G/5G networks. More specifically, as depicted in Figure 22f, four cases of RAT transitions (including 4G level-1 $\rightarrow$ 5G level-0, 4G level-2 $\rightarrow$ 5G level-0, 4G level-3 $\rightarrow$ 5G level-0, and 4G level-4 $\rightarrow$ 5G level-0) drastically increase the likelihood of cellular failures, and thus should be avoided if no side effect is incurred.

Here the side effect mainly lies in the potential data rate increase if we allow such 4G$\rightarrow$ 5G RAT transitions. Nonetheless, given that in all the four cases the 5G access is coupled with level-0 signal strength (and thus can hardly provide a high data rate), the "potential" increase in data rate brought by these RAT transitions can scarcely happen in principle. To check this practically, we conduct small-scale benchmark experiments using four different 5G phones as listed in Table I, finding that these RAT transitions almost always (>95%) decrease the data rate. Consequently, we conclude that in general the four undesirable cases of RAT transitions can be safely avoided to preserve the stability of cellular connections.

Apart from the major case of 4G$\rightarrow$ 5G transition, Figure 22 also depicts the increase of normalized prevalence of cellular failures for the other RAT transition cases. Similar as in the 4G$\rightarrow$ 5G transition, for all the RATs we can observe "undesirable" transition cases where the prevalence of cellular failures is largely increased. A common pattern of such cases is that failures tend to occur when there is level-0 RSS after transition. This can be intuitively explained by the highest prevalence of cellular failures with regard to level-0 RSS, as shown in Figure 20. Therefore, we suggest OS developers to carefully avoid these cases so as to improve cellular reliability. Meanwhile, avoiding these problematic cases should not negatively impact the devices' data rates, as the RSS is extremely weak after transition and thus can hardly provide better cellular performance.

TIMP-based Flexible Data_Stall Recovery. Recall in § III-B that to address Data_Stall failures, Android has implemented a *three-stage progressive recovery mechanism* that attempts to repair the user device's cellular connection with three operations: (1) cleaning up current connections, (2) re-registering into the network, and (3) restarting the device's radio component. Before entering each stage (including the first stage), Android would passively monitor the existence of Data_Stall for one minute (which we call the "probation") in case that the previous (more lightweight) operation has already fixed the problem. Although the three recovery operations can be quite effective when executed, as discussed in § III-B, in practice we notice that the fixed-time (*i.e.,* one-minute) recovery trigger is usually lagging and not user-friendly.

To figure out an appropriate trigger, our key insight is that the conceptual three-stage progressive recovery in Android is essentially a state transition process. As depicted in Figure 23, the process includes five states: S_0, S_1, S_2, S_3, and $S_e = S_4$. Here S_0 denotes the start point (when Data_Stall is detected by Android), S_1, S_2, S_3 respectively represent starting the execution of the aforementioned three recovery operations, and S_e marks the end of the process. According to our measurement, the state transition from S_i to the next state is basically only dependent on S_i and other stochastic events, and thus can be modeled by a Markov process [12].

With the above understanding, we can then formalize the expected overall recovery time (denoted as $T_{recovery}$) so as to calculate more suitable triggers that are able to minimize $T_{recovery}$. However, the traditional Markov process can only model a stationary process where the state transition probability is not affected by the elapsed time *t*, and thus is not applicable to our scenario where the state transition probability also depends on *t*, as indicated in Figure 8 (the user device can automatically fix the problem as time goes by).

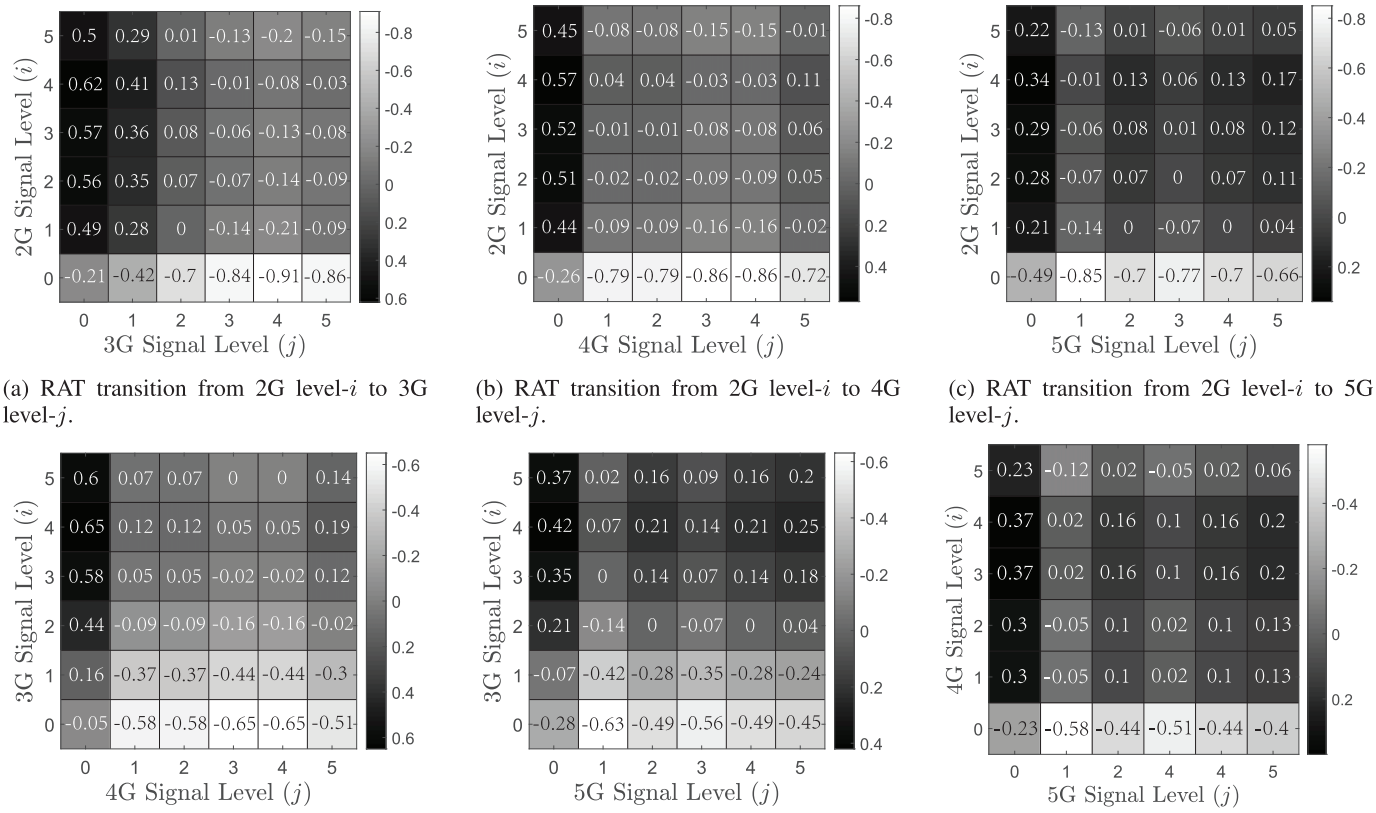
Thus, using our dataset we build a *time-inhomogeneous Markov process* [12] (TIMP) to model the complex state transitions during the Data_Stall recovery process in a time-sensitive manner, by incorporating recovery probabilities within different time windows. Specifically, after entering S_i, the user device either automatically recovers from Data_Stall within the time window $[S_i, S_{i+1}]$ (referred to as Case-1), or enters the next state after Pro_i seconds (referred to as Case-2), where Pro_i denotes the probation time for leaving S_i. For any elapsed time t within the time window, we denote the probability of the device's recovering from Data_Stall as $\mathbb{P}_{i\rightarrow e}(t)$. Thereby, the probability of its not recovering (thus entering S_{i+1}) is $\mathbb{P}_{i\rightarrow i+1} = 1 - \mathbb{P}_{i\rightarrow e}(\sigma Pro_i)$, where $\sigma Pro_i = \sum_{k=0}^{i} Pro_k$ is the elapsed time from S_0 to S_{i+1}.

At this point, we can formalize the expected recovery time after entering state S_i (denoted as T_i) as the sum of three parts:

$$T_i = \int_{\sigma Pro_{i-1}}^{\sigma Pro_i} \mathbb{P}_{i\rightarrow e}(t)dt + \mathbb{P}_{i\rightarrow i+1} \cdot T_{i+1} + O_i. \quad (1)$$

The first part is the integral of $\mathbb{P}_{i\rightarrow e}(t)$ over the time window $[S_i, S_{i+1}]$, *i.e.,* $\int_{\sigma Pro_{i-1}}^{\sigma Pro_i} \mathbb{P}_{i\rightarrow e}(t)dt$, representing that Case-1 occurs. The second part is the probability of the device's entering the next state ($\mathbb{P}_{i\rightarrow i+1}$) multiplying the expected recovery time (T_{i+1}) after entering the next state, *i.e.,* $\mathbb{P}_{i\rightarrow i+1} \cdot T_{i+1}$, representing that Case-2 occurs. Finally, the third part is the time overhead for executing each recovery operation, denoted as O_1, O_2, and O_3, where $O_1 < O_2 < O_3$ given the progressive nature of the three recovery operations.

In detail, we can obtain the approximate values of $\mathbb{P}_{i\rightarrow e}$ and O_i using our duration measurement data of Data_Stall failures. Specially, when $i = 0$, $O_i = 0$ since no recovery operation is executed at this stage; when $i = 3$, $T_3 = \int_{\sigma Pro_2}^{t_m} \mathbb{P}_{3\rightarrow e}(t)dt + O_3$, where t_m is the maximum duration of Data_Stall failures. Thus, we know that the expected overall recovery time $T_{recovery} = T_0$ is essentially determined by the three probations Pro_0, Pro_1, and Pro_2.



(a) RAT transition from 2G level-i to 3G level-j.

(b) RAT transition from 2G level-i to 4G level-j.

(c) RAT transition from 2G level-i to 5G level-j.

(d) RAT transition from 3G level-i to 4G level-j.

(e) RAT transition from 3G level-i to 5G level-j.

(f) RAT transition from 4G level-i to 5G level-j.

Fig. 22. Increase of normalized prevalence of cellular failures for different RAT transitions (*e.g.,* from 4G level-*i* to 5G level-*j*). Deeper color represents larger increase. For example, the dark cell in Figure 22f ($i = 4$, $j = 0$) means that when a cell phone switches from 4G level-4 signal access to 5G level-0 signal access, the normalized prevalence of cellular failures will sharply increase by 0.37, implying that this RAT transition will significantly increase the likelihood of cellular failures.

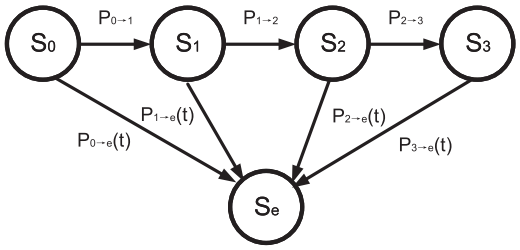


Fig. 23. The time-inhomogeneous Markov process (TIMP) that models the Data_Stall recovery process in Android, where the transition probabilities among the five states are also impacted by the elapsed time (t).

Our optimization objective is then to minimize $T_{recovery}$ for different possible values of Pro_0, Pro_1, and Pro_2. To this end, we use the annealing algorithm [28] to search for the global minimum, thus knowing that $T_{recovery}$ is minimized when $Pro_0 = 21$ seconds, $Pro_1 = 6$ seconds, and $Pro_2 = 16$ seconds. Consequently, the desired $T_{recovery} = 27.8$ seconds, which is smaller than a normal user's tolerance of Data_Stall duration (∼30 seconds, *cf.* § III-B). In contrast, using the default probations ($Pro'_0 = Pro'_1 = Pro'_2 = 60$ seconds) in the original recovery mechanism of Android, the expected recovery time is 38 seconds, indicating that our designed trigger clearly outperforms the original one in Android.

Meticulous Failure Judgment. As mentioned in § III-B, the Android system aggressively regards all aborted connections as Data_Setup_Error failures, which results in considerable false positives (*e.g.,* app-induced aborted connections). Moreover, they will not be reconnected (NO_RETRY_FAILURE) even if new network sessions require them, as the system assumes that retrying aborted connections would still be unsuccessful. Therefore, in the misjudged case, the undesirable situation will persist until a manual reset, leading to a long recovery time.

To mitigate the impact of the premature failure judgment, we rebuild it as a more meticulous method with the help of cross-layer context tracing. The detailed pipeline is shown in Figure 24. Specifically, when a network session triggers the establishment of a data connection, we will log the detailed information of the session and its initiator. If the setup request for this session is canceled before the data connection is successfully established, we will further record whether the cancellation is actively executed by its initiator. In this way, when encountering an aborted connection due to being no longer needed by any network session during its establishment, we can use the aforementioned information to assess if it stems from app behaviors.

An active cancellation of the request indicates that the termination of the establishment is not caused by an error. In this case, we must not consider it as a failure; instead, we allow it to terminate normally. Consequently, the cellular connection can be timely re-established when a new network session requires it, rather than waiting for the manual resetting, significantly reducing the recovery time. We have also reported

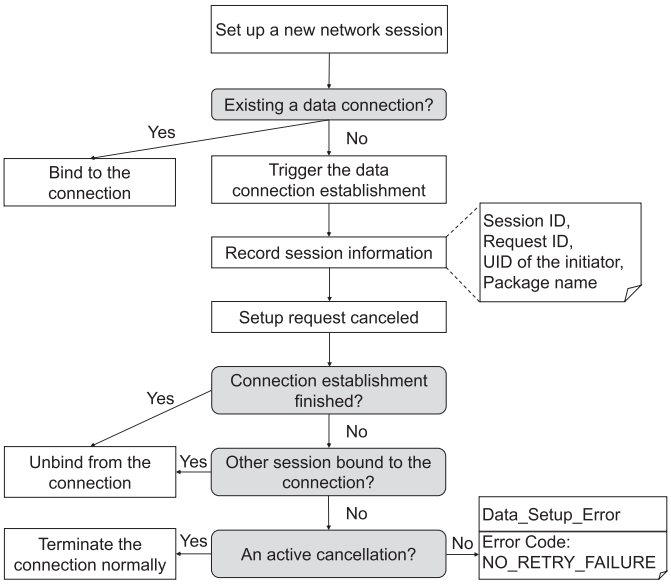


Fig. 24. The pipeline of our meticulous failure judgement.

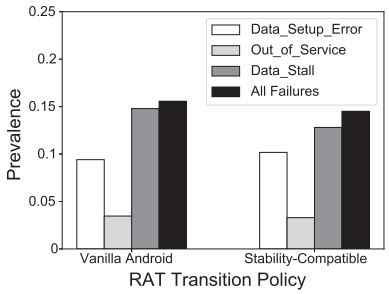


Fig. 25. Prevalence of cellular failures with the RAT transition policy of vanilla Android and our Stability-Compatible RAT Transition.

the design defect of Android to the official development team in Google, who have confirmed the issue and promised to fix the issue in the upcoming Android release [13].

C. Deployment and Evaluation

In order to validate the real-world effect of our design (§ IV-B), we conduct a two-period evaluation. We initially patched the first (Stability-Compatible RAT Transition) and the second (TIMP-based Flexible Data_Stall Recovery) mechanisms to CrowdEye and invited all the opt-in users in late Oct. 2020 to participate in our evaluation and 28M users opted-in and upgraded to our patched system. This period of evaluation has been conducted for two months (Nov.–Dec. 2020). We further patched the third (Meticulous Failure Judgment) mechanism and carried out the second period of evaluation from Aug. 2024 to Sep. 2024 with the help of 16M opt-in users.

As shown in Figure 25 and Figure 26, thanks to our Stability-Compatible RAT Transition mechanism, cellular failures occur 10% less prevalently and 40.3% less frequently on the participant 5G phones, without sacrificing the data rate (as explained in § IV-B). In detail, for Data_Setup_Error, Data_Stall, and Out_of_Service failures, the decrease of prevalence (frequency) is −7% (25.72%), 13.45% (42.4%), and 5% (50.26%), respectively. Here the only exception lies in the prevalence of Data_Setup_Error failures, which slightly increases after our optimization is applied; however, given

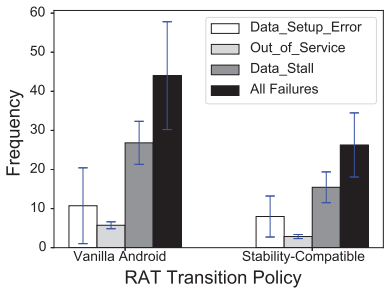


Fig. 26. Frequency of cellular failures with the RAT transition policy of vanilla Android and our Stability-Compatible RAT Transition.

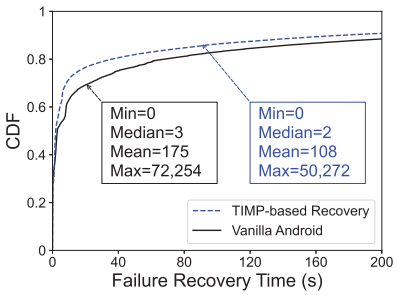


Fig. 27. Recovery time of Data_Stall failures with the Data_Stall recovery mechanism in vanilla Android and our TIMP-based Flexible Recovery.

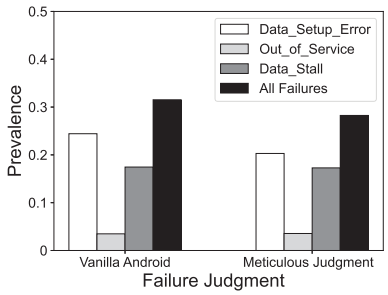


Fig. 28. Prevalence of cellular failures with the failure judgment in vanilla Android and our Meticulous Failure Judgment.

the occurrence frequency is significantly reduced by 25.72% by our optimization, we feel that the exception is most probably due to normal statistical fluctuation during the evaluation—after all, the measurement study and the evaluation study are conducted in two disjoint time periods.

Moreover, as shown in Figure 27, after our designed TIMP-based Flexible Data_Stall Recovery mechanism is put into practice, 38% reduction on the recovery time of Data_Stall failures is achieved on average and the median recovery time is reduced by 33% (from 3 seconds to 2 seconds). Furthermore, our TIMP-based recovery mechanism works in a principled and flexible manner, so it will automatically adapt to the possible pattern changes of Android system behaviors and cellular technologies in the future.

Further, as shown in Figure 28 and Figure 29, the Meticulous Failure Judgment mechanism achieves a 16.9% (13.6%) reduction in the prevalence (frequency) of Data_Setup_Error failures by avoiding the misjudgment of app-induced aborted connections, corresponding to 10.4% (9.3%) reduction on the total prevalence (frequency) of all types of failures. More notably, as illustrated in Figure 30, the average recovery time for Data_Setup_Error failures reduces by 56%

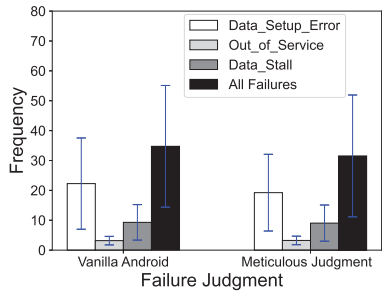


Fig. 29. Frequency of cellular failures with the failure judgment in vanilla Android and our Meticulous Failure Judgment.

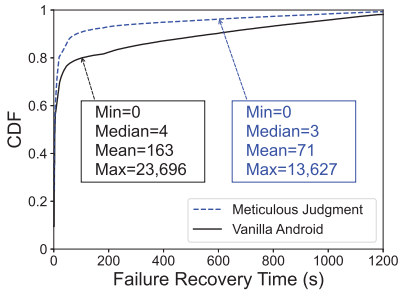


Fig. 30. Recovery time of Data_Setup_Error failures with the failure judgment in vanilla Android and our Meticulous Failure Judgment.

(from 163 seconds to 71 seconds), with the median time drop by 25% from 4 seconds to 3 seconds.

Overall, the three enhancements jointly reduce 38% cellular failures for 5G phones and save 31% failure recovery time across all phones. Our enhancements only involve lightweight modifications to the vanilla Android, primarily including recording minimal system-level information and adjusting the timing of recovery operations, without introducing any new system services or persistent background processes. To evaluate the overhead of our patched CrowdEye, we perform small-scale benchmark experiments using the 70 involved phone models. The results demonstrate that our optimizations incur little overhead to a low-end Android phone: $<$3% CPU utilization, $\sim$60 KB of memory usage, and $<$100 KB of storage space; the network usage is $<$100 KB per month. Even in the worst case where the monthly number of failures reaches 24,000+ on a single phone (see Figure 7), the incurred CPU, memory, and storage overheads are still acceptable: $<$9% CPU utilization, $\sim$3 MB of memory usage, and $<$20 MB of storage space; the network usage is $\sim$20 MB per month.

V. Related Work

With the fast and wide penetration of wireless cellular networks across the globe, the quality of cellular service is becoming more and more important to a person's everyday life, an organization's collaborative work, and even a nation's industrial information ecosystem. In the past ten years or so, there has been a plethora of work studying the characteristics of cellular networks, from the perspectives of mobile ISPs/base stations [29], [30], user devices [31], [32], [33], user-to-device interactions [34], and device-to-device communications [35]. Researchers have also developed measurement tools and platforms for conducting cellular network measurements [36], [37].

Most of the above studies investigate the common aspects (performance and availability) of cellular networks, such as bandwidth, delay, BS density, and signal coverage. Till now, only a few studies focus on cellular reliability. Hui and Lau [38] at T-Mobile leverage a cross-layer measurement strategy to understand the Data_Stall failures and their impact on mobile QoE, by analyzing the data collected at both sides of BSes and user devices. They uncover specific root causes for the problem, including link corruptions and packet drops during radio state transitions, as well as incorrect implementation of the radio network controller's scheduling algorithm.

Prior work also suggests that BS and RAT handoffs can be rather difficult to handle under complex environments, leading to cellular service unreliability [39], [40] and unavailability [9], [41] issues. Our study distinguishes itself from the above in its extremely large user scale and ISP/BS coverage, and particularly the comprehensive considerations of various cellular failures in the wild. Moreover, when seeking the reasons of cellular failures, our work is featured by the joint analysis of phone hardware configurations, OS internals, BS characteristics, and BS-to-phone interactions.

Measurement studies shed light on possible design and implementation optimizations. Cellular network optimizations have been extensively discussed from multiple aspects like congestion control [42], [43], energy efficiency [44], and security enhancements [45], as well as under various scenarios like video streaming [46], web browsing [47], and cellular-WiFi interaction [48]. In particular, researchers have proposed to leverage lower-layer cellular information to boost the upper-layer user-perceived performance [46], [47]. Contrasting the above efforts, we identify and explore the unique optimization opportunities (in coordination with a major Android phone vendor) to enable more reliable and faster recovery from several types of severe cellular failures. We have also performed large-scale deployment of our proposed solutions which yielded real-world impact.

VI. Discussion

This section discusses the generalizability of our findings and enhancements to other regions and ecosystems.

A. Generalizability to Other Regions

To explore the generalizability of our findings across regions, we conduct a one-month benchmark study in India and Spain in Mar. 2025, by inviting Xiaomi users in these regions to install CrowdEye on their phones. Finally, 8M users in India and 2M users in Spain opted in, involving 11 and 9 device models respectively. The hardware configurations of these devices are similar to those listed in Table II.

The measurement results highlight the worldwide severity of cellular reliability issues and demonstrate the broader applicability of our conclusions. In general, cellular failures occur frequently in both regions: devices in India and Spain experienced an average of 5 and 9 failures in a month, respectively. Moreover, we notice that the situation in these regions is more severe than in China, where the monthly average is 4 failures per device. This increased severity can be explained by multiple factors. Specifically, the higher failure frequency in India can be attributed to insufficient cellular coverage in remote and rural areas. In Spain, it may stem from the broader adoption of mmWave technology, which—while offering higher bandwidth—is less stable than mid-band 5G [49]. Additionally, the larger proportion of devices running Android 14 further exacerbates reliability issues in Spain, due to the system's premature failure judgment

discussed in § III-B. Taken together, these findings suggest that the key factors we identified in China, including ISP's infrastructure investments, instability introduced by advanced cellular technologies, and software defects on mobile devices, also critically influence cellular reliability at a global scale.

B. Generalizability to Other Ecosystems

We believe that our findings and optimizations extend beyond Android devices and are applicable to enhancing cellular reliability in other ecosystems. Taking iOS for instance, considerable user reports suggest that iOS devices face similar reliability issues where cellular connections fail to work [50], [51]. These issues often occur following system updates [50], [52], suggesting that they may stem from underlying system design defects in iOS. Furthermore, these failures often persist until a manual reset of the connection is performed, which resembles the phenomenon caused by Android's premature failure judgment, as discussed in § III-B. Thus, although originally designed for Android, our three enhancements provide valuable insights for iOS developers as well. Moreover, our findings in § III-C regarding the relationship between cellular reliability and ISPs/BSes are also observed on mobile devices running other OSes. For instance, Apple acknowledges that iOS devices may suffer from cellular availability or reliability issues in crowded areas, despite showing a strong signal strength [53]. Therefore, our guidelines in principle, especially suggestions for ISPs, can benefit devices across different OSes.

VII. Conclusion

This paper presents our efforts towards understanding and combating the reliability issues in cellular networks. We conduct a long-term and large-scale measurement study with the crowdsourcing help from 123 million opt-in users from 2020 to 2024. Collaborating with a major Android phone vendor, we develop and deploy a continuous monitoring platform to collect fine-grained, in-situ system traces, leveraging which we reveal the nationwide prevalence and frequency of cellular failures, along with their evolution trends over the past five years. More in depth, we uncover severe reliability problems in the cellular connection management of Android, as well as the BS utilization and deployment strategies of mobile ISPs. Driven by the study insights, we provide useful guidelines to help tackle a variety of cellular failures. Most importantly, some of our solutions have been adopted by 44 million users, generating prominent realistic impacts. Furthermore, our observations and enhancements could inspire mobile ISPs, phone vendors, and OS developers to promote a more reliable cellular ecosystem, with implications extending to regions beyond China, complex contexts such as mobility management, and future planet-wide 6G deployment.

Acknowledgment

The authors appreciate the anonymous reviewers for their insightful comments. They sincerely thank to Yanjun Kang, Dan Xiao, and Daliang Sun for their valuable feedback. They also appreciate Quan Tao and Yongqing Li for their help in data collection and analysis.

References

[1] Z. Lai, Y. C. Hu, Y. Cui, L. Sun, N. Dai, and H.-S. Lee, "Furion: Engineering high-quality immersive virtual reality on Today's mobile devices," *IEEE Trans. Mobile Comput.*, vol. 19, no. 7, pp. 1586–1602, Jul. 2020.

[2] Y. Liu et al., "Embodied navigation," *Sci. China Inf. Sci.*, vol. 68, no. 4, pp. 1–39, 2025.

[3] C. Duffy. (2020). *The Big Differences Between 4G and 5G*. [Online]. Available: https://www.cnn.com/2020/01/17/tech/5g-technical-explainer/index.html

[4] Android.org. (2021). *Data Setup Error in Android*. [Online]. Available: https://android.googlesource.com/platform/frameworks/opt/telephony/+/refs/heads/master/src/java/com/android/internal/telephony/dataconnection/DcTracker.java

[5] Android.org. (2021). *State Out of Service in Android*. [Online]. Available: https://developer.android.com/reference/android/telephony/ServiceState

[6] Android.org. (2021). *Data Stall Report in Android*. [Online]. Available: https://developer.android.com/reference/android/net/ConnectivityDiagnosticsManager.DataStallReport

[7] Huawei.com. (2020). *Deployment of the Massive MIMO Technique for Indoor Scenarios (in Chinese)*. [Online]. Available: https://www.huawei.com/cn/news/2020/9/chinaunicom-distributed-massive-mimo-verified

[8] Y. Li et al., "A nationwide study on cellular reliability: Measurement, analysis, and enhancements," in *Proc. ACM SIGCOMM Conf.*, Aug. 2021, pp. 597–609.

[9] H. Deng, C. Peng, A. Fida, J. Meng, and Y. C. Hu, "Mobility support in cellular networks: A measurement study on its configurations and implications," in *Proc. Internet Meas. Conf.*, Oct. 2018, pp. 147–160.

[10] Y. Li, Q. Li, Z. Zhang, G. Baig, L. Qiu, and S. Lu, "Beyond 5G: Reliable extreme mobility management," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.*, Jul. 2020, pp. 344–358.

[11] Android.org. (2021). *Data Fail Cause in Android*. [Online]. Available: https://android.googlesource.com/platform/frameworks/base/+/master/telephony/java/android/telephony/DataFailCause.java

[12] G. Winkler, *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods: A Mathematical Introduction*, vol. 27. Cham, Switzerland: Springer, 2012.

[13] Google.com. (2024). *Premature Judgment of Data Setup Error*. [Online]. Available: https://github.com/CellReliabilityEvo/CellReliabilityEvo.github.io/blob/main/dsejudgment/report.pdf

[14] Android.org. (2021). *Data Connection Management in Android*. [Online]. Available: https://android.googlesource.com/platform/frameworks/opt/telephony/+/master/src/java/com/android/internal/telephony/dataconnection/DataConnection.java

[15] J. Horn. (2021). *Mitigations Are Attack Surface, Too*. [Online]. Available: https://googleprojectzero.blogspot.com/2020/02/mitigations-are-attack-surface-too.html

[16] P. Srisuresh et al., *NAT Behavioral Requirements for ICMP*, document RFC 5508, Apr. 2009.

[17] A. Kumar, J. Postel, C. Neuman, P. Danzig, and S. Miller, *Common DNS Implementation Errors and Suggested Fixes*, document RFC 1536, Oct. 1993.

[18] Android.org. (2021). *SMS Manager in Android*. [Online]. Available: https://developer.android.com/reference/android/telephony/SmsManager

[19] 3GPP. (2022). *Carrier Aggregation*. [Online]. Available: https://www.3gpp.org/technologies/101-carrier-aggregation-explained

[20] M. Rahman and H. Yanikomeroglu, "Enhancing cell-edge performance: A downlink dynamic interference avoidance scheme with inter-cell coordination," *IEEE Trans. Wireless Commun.*, vol. 9, no. 4, pp. 1414–1425, Apr. 2010.

[21] F. Qamar, K. B. Dimyati, M. N. Hindia, K. A. B. Noordin, and A. M. Al-Samman, "A comprehensive review on coordinated multi-point operation for LTE-A," *Comput. Netw.*, vol. 123, pp. 19–37, Aug. 2017.

[22] Android.org. (2021). *Android 10 Highlights*. [Online]. Available: https://developer.android.com/about/versions/10/highlights

[23] Android.org. (2021). *Enhance Your Apps With 5G*. [Online]. Available: https://developer.android.com/training/connectivity/5g/enhance-with-5g

[24] G. K. Zipf, *Human Behavior and the Principle of Least Effort: An Introd. to Human Ecology*. Reading, MA, USA: Addison-Wesley, 1949.

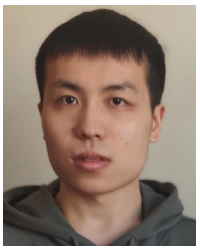
[25] J. Sheng, Z. Tang, C. Wu, B. Ai, and Y. Wang, "Game theory-based multi-objective optimization interference alignment algorithm for HSR 5G heterogeneous ultra-dense network," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13371–13382, Nov. 2020.

[26] M. K. Müller, M. Taranetz, and M. Rupp, "Providing current and future cellular services to high speed trains," *IEEE Commun. Mag.*, vol. 53, no. 10, pp. 96–101, Oct. 2015.

[27] L. Cano, A. Capone, G. Carello, M. Cesana, and M. Passacantando, "On optimal infrastructure sharing strategies in mobile radio networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 5, pp. 3003–3016, May 2017.
[28] R. H. Otten and L. P. van Ginneken, *The Annealing Algorithm*, vol. 72. Cham, Switzerland: Springer, 2012.
[29] Z. Hu et al., "An in-depth analysis of 3G traffic and performance," in *Proc. 5th Workshop All Things Cellular: Operations, Appl. Challenges*, Aug. 2015, pp. 1–6.
[30] D. Xu et al., "Understanding operational 5G: A first measurement study on its coverage, performance and energy consumption," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.*, Jul. 2020, pp. 479–494.
[31] L. Li et al., "A measurement study on multi-path TCP with multiple cellular carriers on high speed rails," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2018, pp. 161–175.
[32] J. Wang et al., "An active-passive measurement study of TCP performance over LTE on high-speed rails," in *Proc. ACM MobiCom*, Aug. 2019, pp. 1–16.
[33] A. Narayanan, E. Ramadan, J. Carpenter, Q. Liu, F. Qian, and Z. L. Zhang, "A first look at commercial 5G performance on smartphones," in *Proc. ACM Web Conf.*, 2020, pp. 894–905.
[34] Y. Li et al., "Understanding the ecosystem and addressing the fundamental concerns of commercial MVNO," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1364–1377, Jun. 2020.
[35] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, and J. Wang, "Large-scale measurement and characterization of cellular machine-to-machine traffic," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 1960–1973, Dec. 2013.
[36] S. Kumar, E. Hamed, D. Katabi, and L. E. Li, "LTE radio analytics made easy and accessible," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 211–222, Feb. 2015.
[37] A. Nikravesh, H. Yao, S. Xu, D. Choffnes, and Z. M. Mao, "Mobilyzer: An open platform for controllable mobile network measurements," in *Proc. 13th Annu. Int. Conf. Mobile Syst., Appl., Services*, May 2015, pp. 389–404.
[38] J. Hui and K. Lau, "T-mobile QoE lab: Making mobile browsing faster and open research problems," in *Proc. ACM MobiCom*, 2013, pp. 239–242.
[39] Y. Li, H. Deng, J. Li, C. Peng, and S. Lu, "Instability in distributed mobility management: Revisiting configuration management in 3G/4G mobile networks," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Sci.*, 2016, pp. 261–272.
[40] Y. Li, J. Xu, C. Peng, and S. Lu, "A first look at unstable mobility management in cellular networks," in *Proc. 17th Int. Workshop Mobile Comput. Syst. Appl.*, Feb. 2016, pp. 15–20.
[41] C. Peng and Y. Li, "Demystify undesired handoff in cellular networks," in *Proc. 25th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Aug. 2016, pp. 1–9.
[42] K. Winstein, A. Sivaraman, and H. Balakrishnan, "Stochastic forecasts achieve high throughput and low delay over cellular networks," in *Proc. 10th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2013, pp. 459–471.
[43] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, "Adaptive congestion control for unpredictable cellular networks," in *Proc. ACM Conf. Special Interest Group Data Commun.*, 2015, pp. 509–522.
[44] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *Proc. 9th ACM SIGCOMM Conf. Internet Meas.*, Nov. 2009, pp. 280–293.
[45] H. Deng, W. Wang, and C. Peng, "CEIVE: Combating caller ID spoofing on 4G mobile phones via callee-only inference and verification," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw.* Washington, DC, USA: Combat, Oct. 2018, pp. 369–384.
[46] X. Xie and X. Zhang, "POI360: Panoramic mobile video telephony over LTE cellular networks," in *Proc. 13th Int. Conf. Emerg. Netw. EXperiments Technol.*, Nov. 2017, pp. 336–349.
[47] X. Xie, X. Zhang, and S. Zhu, "Accelerating mobile web loading using cellular link information," in *Proc. 15th Annu. Int. Conf. Mobile Syst., Appl., Services*, 2017, pp. 427–439.
[48] Y.-S. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "ECF: An MPTCP path scheduler to manage heterogeneous paths," in *Proc. 13th Int. Conf. Emerg. Netw. Exp. Technol.*, Nov. 2017, pp. 147–159.
[49] R. A. K. Fezeu et al., "Unveiling the 5G mid-band landscape: From network deployment to performance and application QoE," in *Proc. ACM SIGCOMM Conf.*, Aug. 2024, pp. 358–372.
[50] A. S. Community. (2023). *Iphone Shows No Service*. [Online]. Available: https://discussions.apple.com/thread/256019105
[51] A. S. Community. (2023). *Cellular Data Not Working on Iphone*. [Online]. Available: https://discussions.apple.com/thread/254833516
[52] A. S. Community. (2024). *Cellular Data Problem on Iphone After IOS 18 Update*. [Online]. Available: https://discussions.apple.com/thread/255763551
[53] A. Inc. (2023). *Iphone Running Slow*. [Online]. Available: https://support.apple.com/en-us/102598

Yunhao Liu (Fellow, IEEE) received the B.Sc. degree from the Automation Department, Tsinghua University, and the M.Sc. and Ph.D. degrees in computer science and engineering from Michigan State University. He is currently a Full Professor and the Dean of the Global Innovation Exchange (GIX), Tsinghua University. His research interests include sensor networks, the IoT, RFID, distributed systems, and cloud computing. He is a Fellow of ACM.

Hongyi Wang received the B.Sc. degree from the Department of Electronic Engineering, Tsinghua University, in 2021. She is currently pursuing the Ph.D. degree with the School of Software, Tsinghua University. Her research interests include network measurement and machine learning.



Yang Li received the B.Sc. and M.Eng. degrees from the School of Software, Tsinghua University, in 2018 and 2021, respectively, where he is currently pursuing the Ph.D. degree. His research interests include big data analysis, network measurement, and machine learning.

Zhenhua Li (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees from Nanjing University in 2005 and 2008, respectively, and the Ph.D. degree from Peking University in 2013, all in computer science and technology. He is currently a Tenured Associate Professor with the School of Software, Tsinghua University. His research interests include cover network measurement, mobile networking/virtualization, and cloud computing/gaming. He is a Senior Member of ACM.

Guoquan Zhang received the M.Eng. degree from the University of Chinese Academy of Sciences in 2009. He is currently a Senior Software Engineer with Xiaomi Technology Company Ltd. He works in mobile system development and mobile networking.

Lei Yang received the B.Sc. degree from Taiyuan University of Science and Technology in 2018 and the M.Eng. degree from Taiyuan University of Technology in 2021. She is currently a Senior Software Engineer with Xiaomi Technology Company Ltd. She works in network measurement and mobile networking.