A Nationwide Study on Cellular Reliability: Measurement, Analysis, and Enhancements

Yang Li[†*], Hao Lin[†*], Zhenhua Li[†], Yunhao Liu[†]
Feng Qian[§], Liangyi Gong[†], Xianlong Xin[‡], Tianyin Xu[¶]

[†]Tsinghua University [‡]Xiaomi Technology Co. LTD
[§]University of Minnesota, Twin Cities [¶]University of Illinois at Urbana-Champaign

ABSTRACT

With recent advances on cellular technologies (such as 5G) that push the boundary of cellular performance, cellular reliability has become a key concern of cellular technology adoption and deployment. However, this fundamental concern has never been addressed due to the challenges of measuring cellular reliability on mobile devices and the cost of conducting large-scale measurements. This paper closes the knowledge gap by presenting the first large-scale, in-depth study on cellular reliability with more than 70 million Android phones across 34 different hardware models. Our study identifies the critical factors that affect cellular reliability and clears up misleading intuitions indicated by common wisdom. In particular, our study pinpoints that software reliability defects are among the main root causes of cellular data connection failures. 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 on Android's cellular implementations have reduced 40% cellular connection failures for 5G phones and 36% failure duration across all phones.

CCS CONCEPTS

• **Networks** → **Mobile networks**; **Network reliability**; *Network measurement*; *Network performance analysis*;

KEYWORDS

Cellular Network; 5G Network; Reliability; Measurement; Mobile Operating System; Cellular Connection Management

ACM Reference Format:
Yang Li, Hao Lin, Zhenhua Li, Yunhao Liu, Feng Qian, Liangyi Gong, Xianlong Xin, Tianyin Xu. 2021. A Nationwide Study on Cellular Reliability: Measurement, Analysis, and Enhancements. In *ACM SIGCOMM 2021 Conference (SIGCOMM '21), August 23–27, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3452296.3472908

[*]Co-primary authors. Zhenhua Li and Yunhao Liu are the corresponding authors.

SIGCOMM '21, August 23–27, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8383-7/21/08…$15.00
https://doi.org/10.1145/3452296.3472908

1 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 [24]. 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. 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 devices per square kilometer (100× more than 4G) [14].

While we have been mainly focusing on the performance of cellular network technologies and the availability of cellular network services, the cellular reliability of mobile devices has been largely overlooked—without reliable cellular connections, performance would be a mirage. From a mobile device's perspective, cellular data connections can fail mostly in the following three ways[1]:

- Data_Setup_Error [4][2]: The mobile device can receive signals from a nearby base station (BS) but cannot establish a data connection with the BS.
- Out_of_Service [8]: The data connection has been established, but the mobile device cannot receive cellular data.
- Data_Stall [5]: The mobile device can receive cellular data, but the data connection abnormally stalls (for longer than one minute as suggested in Android).

However, cellular reliability is rarely studied or measured, but constantly acts as an X-factor in discussion or decision making [14, 36]. Certainly, understanding cellular reliability at large is challenging. First, as we will discuss in §2, existing mobile systems do not provide sufficient tracing and logging support for low-level cellular connection components, creating significant barriers to precisely capturing failure events and effectively diagnose their root causes. Second, it is difficult and expensive to conduct large-scale reliability measurements on real-world mobile devices; controlled lab studies could help but hardly yield representative characteristics [59].

To close the knowledge gap, 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 on cellular reliability from the device perspective. Specifically, our

[1]We use Android terminologies throughout the paper.
[2]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.

goal is to measure the prevalence and severity of cellular reliability problems perceived by user devices and reveal the root causes of cellular (data connection) failures, including Data_Setup_Error, Out_of_Service, Data_Stall, and so forth. Interestingly, understanding cellular reliability turns out to be well aligned with the business need of Xiaomi, as cellular connection issues were a major contributor to their customer reports, but had been elusive problems for Xiaomi engineers. Therefore, we have the common interests in understanding cellular failures and improving cellular reliability.

Measurement. We build a continuous monitoring infrastructure on top of a customized Android system called Android-MOD. Android-MOD records system-level traces (without requiring root privileges) upon the occurrence of suspicious cellular failure events. To extract true failure events and collect diagnostic information, we instrument relevant system services to record detailed device/network state information and carefully filter out false positives.

We invited all users of Xiaomi to participate in the measurement study by installing Android-MOD on their phones, and finally 70M users opted in and shared data with us for eight months (Jan.–Aug. 2020). The dataset involves 34 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 5.3M BSes. All the data were collected with informed consent of opt-in users, and no personally identifiable information (PII) was collected during the measurement.

Analysis. Our measurement reveals that cellular failures are prevalent on all the 34 models of devices. For each model, 0.15%–45% (averaging at 23%) of the devices have experienced at least one cellular failure. On average, as many as 33 failures occur to a device during the measurement, and a failure lasts for as long as 3.1 minutes. Newer OS versions (*e.g.,* Android 10) and communication modules (*e.g.,* 5G modules) substantially aggravate the situation, while better hardware does not seem to relieve the situation. In particular, our results indicate that cellular failures are mainly caused by software reliability defects rather than inexpensive hardware, *e.g.,* the implementation that blindly prioritizes 5G connection in Android 10 greatly impairs the stability of cellular connections.

Moreover, we find that most (94%) failure duration is owing to Data_Stall failures. To recover a cellular connection from Data_Stall failures, Android implements a three-stage progressive mechanism which sequentially tries light (cleaning up and restarting the current connection), moderate (re-registering into the network), and heavy (restarting the radio component) recovery techniques based on one-minute probations. Our data show that for the majority of Data_Stall failures, either the user device can automatically fix them in less amount of time, or the user would manually reset the connection after ~30 seconds. Thus, the three-stage design is not efficient.

From the viewpoint of ISPs, cellular failures occur more prevalently (27.1%) on ISP-B users than on ISP-A users (20.1%) and ISP-C users (14.7%) due to the inferior signal coverage of ISP-B. Counter-intuitively, while both the number and overall signal coverage of 3G BSes are smaller than those of 2G or 4G BSes, the prevalence of failures on 3G BSes is lower than that on 2G or 4G BSes. This can be ascribed to the fact that 3G access is usually not preferred when 4G access is available and its signal coverage is worse than that of 2G when 4G access is unavailable, and thus is confronted with less resource contention.

With respect to 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 in fact more likely to happen in this case than when there is weaker (level 1 to 4) RSS. We clear up the mystery—most of the 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 LTE mobility management [12, 29], causing frequent failures tagged with EMM_ACCESS_BARRED, INVALID_EMM_STATE, *etc.* [3].

Enhancements. Our study provides insights to improve cellular reliability at scale 1) for mobile phone vendors to roll out 5G modules and new OS versions, 2) for mobile ISPs to make use of radio resources, *e.g.,* utilizing "idle" 3G BSes and planning BS deployment density in public areas, and 3) for promoting cross-ISP infrastructure.

More importantly, some of our enhancements have been practically deployed with real-world impact. First, instead of aggressively pursuing the potential high data rate of 5G, we optimize the radio access technology (RAT) selection policy in Android 10 by judiciously considering the likelihood of cellular failures and meanwhile utilizing the novel 4G/5G dual-connectivity mechanism [47].

Second, we optimize the three-stage cellular-connection recovery mechanism in Android by replacing its fixed-time trigger with a flexible and dynamic trigger based on a time-inhomogeneous Markov process [49] (TIMP). TIMP advances the traditional Markov process that can only model a stationary process, to model complex state transitions in a time-sensitive manner. The TIMP-based recovery helps most user devices recover more quickly (the three probations are adaptively tuned as 21, 6 and 16 seconds, each being much shorter than one minute) and effectively with negligible overhead.

Since the release of the patched Android-MOD system with the above two-fold optimizations (adopted by 40% of the 70M opt-in users in late Oct. 2020), we have successfully reduced 40% cellular failures for 5G phones (without sacrificing the data rate) and 38% Data_Stall duration (equivalent to 36% total failure duration) for all phones during Nov.–Dec. 2020.

Code Release. The failure diagnosis and fixing code involved in the study is publicly available at https://CellularReliability.github.io.

2 STUDY METHODOLOGY

We conducted a large-scale measurement study on cellular failures based on continuously monitoring 70M opt-in user devices over eight months. The study is enabled by Android-MOD, a customized Android system that provides lightweight, privacy-preserving tracing and analysis beyond the capability of the vanilla Android system.

2.1 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 [2] as shown in Figure 1: a total of five states are used to represent different stages of a cellular connection, including Inactive, Activating, Retrying, Active, and Disconnect. As one state changes 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.

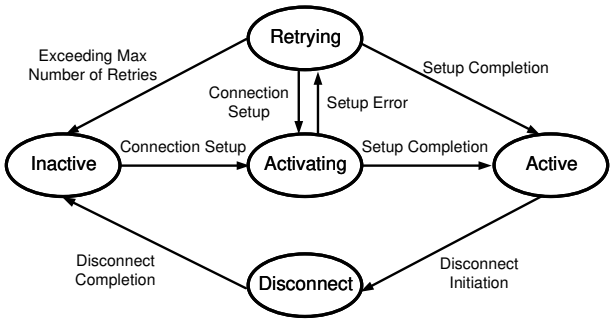


Figure 1: The state machine that models the life cycle of a cellular data connection in Android.

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 [8]. 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 [5] 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 by simply developing an Android app. 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.

2.2 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 Android-MOD, 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) [16].

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 registering 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 facilitate our uncovering the root causes, and 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 [3], and recognized tens of error codes that are highly correlated with false positives as critical auxiliary information.

Third, we note that the existing Data_Stall detection mechanism in Android cannot provide an accurate measurement of a Data_Stall failure's duration, given its fixed detection time (as long as one minute). According to our observations (detailed later in §3.1), 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 duration. 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 Android-MOD. Once a suspicious Data_Stall failure is

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

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 [46]), 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 [22]), 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 the probing results indicate that Data_Stall has not been fixed, we will initiate a new probing process; otherwise, we add up the duration values recorded in all the previous probing processes (since the beginning of this Data_Stall failure) to approximate the actual duration of this Data_Stall failure. Thus, our measurement error is at most five seconds (<< 1 minute). Furthermore, to avoid excessive network overhead, if a Data_Stall failure lasts for longer than 1200 seconds (in few than 10% cases, as illustrated later in §3.1), we will multiplicatively increase the timeout values by a factor of two in the next probing process to balance the incurred error and overhead. Finally, if either timeout value grows to longer than one minute, we will revert to Android's original detection mechanism to estimate the Data_Stall duration.

All in all, our modifications to Android involve system-level information logging (primarily through existing interfaces) and lightweight network probing activities. For even a low-end Android phone at the moment, Android-MOD only incurs <2% CPU utilization, <40 KB of memory usage, and <100 KB of storage space; the network usage per month is <100 KB. 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 to our backend server 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 70M users that participated in our study, the aggregate network overhead per second on the entire cellular networks of the three involved ISPs was below 500 KB, and thus had negligible influence on the performance, availability, and reliability of the studied cellular networks.

2.3 Large-Scale Deployment

With the continuous monitoring infrastructure, in Dec. 2019 we invited all the users of Xiaomi to participate in our measurement study of cellular reliability by installing Android-MOD on their phones. Note that the installation is a lightweight update that will not affect their installed apps, existing data, and OS version. Eventually, 70,965,549 users opted in and collected data for us for eight months (Jan.–Aug. 2020). 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.

3 MEASUREMENT RESULTS

In this section, we first present the general characteristics of our collected measurement data (§3.1). 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 Android phones (§3.2) and ISPs/BSes (§3.3), respectively.

3.1 General Statistics

With the crowdsourcing help from 70,965,549 Android-MOD user devices with 34 different phone models (as listed in Table 1), we record the system-level traces with regard to 2,315,314,213 cellular failures, involving 16,183,145 user devices, 3 mobile ISPs and 5,273,972 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: the former denotes the fraction of devices that experience at least one cellular failure, and the latter is the average number of cellular failures experienced per phone. Our measurement results reveal that cellular failures occur prevalently on all the 34 studied phone models. As indicated in Table 1 and Figure 2, 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 3), and the average number of cellular failures happening to a specific model varies from 2.3 to 90.2 (see Table 1). In a nutshell, while the majority (77%) of phones do not report cellular failures during the measurement, the maximum number of cellular failures happening to a single phone can reach 198,228 (see Figure 3).

Apart from prevalence and frequency, we are also concerned with the duration of cellular failures. As shown in Figure 4, the average duration of cellular failures is as long as 188 seconds (= 3.1 minutes). This is an astonishing value in case of emergency, considering that a victim user is expected to be out of contact for 3.1 minutes. In detail, the duration distribution is highly skewed—while 70.8% cellular failures last for less than 30 seconds, the maximum duration can reach 91,770 seconds (= 25.5 hours). Closer examination reveals that such long-duration failures typically occur in remote regions

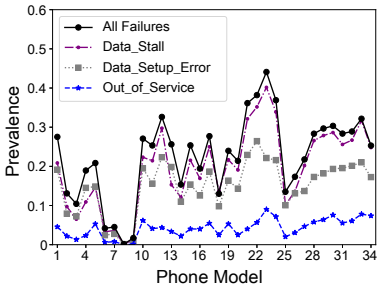


Figure 2: Prevalence of cellular failures on each model of phones.

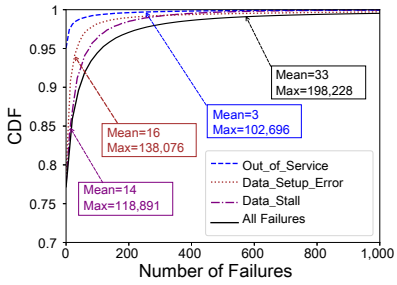


Figure 3: Number of cellular failures happening to a single phone.

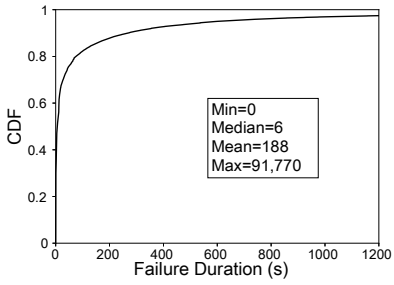


Figure 4: Duration of our recorded cellular failures.

Table 1: Hardware configurations of our studied 34 phone models, generally ordered from low-end to high-end. The rightmost five columns correspond to the phone's 5G capability (5G), Android version (Version), user percentage (Users), fraction of devices that experience at least one cellular failure (Prevalence), and average number of cellular failures experienced per phone (Frequency) during our measurement, respectively.

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

such as mountain and offshore areas, where the BSes have been long neglected and in disrepair.

Among the 2.32 billion collected 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 [7]), whose functions and enabling techniques have been stable for nearly 20 years. As depicted in Figure 3, an average of 16 Data_Setup_Error, 14 Data_Stall, and 3 Out_of_Service events occur to a single phone in our study. While most (95%) phones do not experience Out_of_Service events, the maximum number of Out_of_Service events on a single phone is as large as 102,696. In particular, Data_Stall failures lead to the vast majority of (94%) cellular failure duration among all the collected cellular failure events, which will be explained in §3.2 and practically mitigated in §4.2.

3.2 Android Phone 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 generally indicate the opposite: as shown in Figure 2 and Figure 5, 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 1) 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.

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 6 and Figure 7, 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 production state at the moment.

Android Version. The studied 34 phone models use either Android 9 or Android 10, which are released in Aug. 2018 and Sep. 2019, respectively. Android 11 was recently released in Sep. 2020 and thus was not covered in our study (from Jan. to Aug. 2020). We note that as Android evolves from version 9 to 10, a number of new functions and performance improvements have been implemented [1]. Although these updates are supposed to benefit users, we unexpectedly find that both the prevalence and frequency of cellular

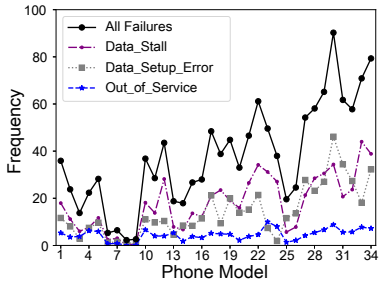


Figure 5: Frequency of cellular failures on each model of phones.

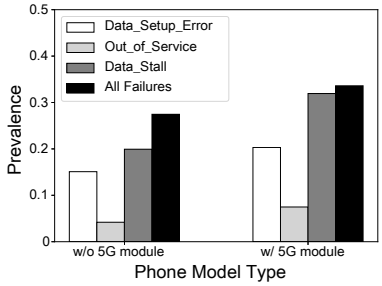


Figure 6: Prevalence of cellular failures on models w/ and w/o the 5G module.

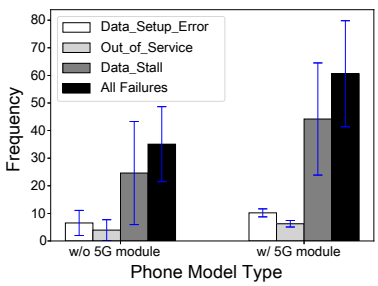


Figure 7: Frequency of cellular failures on models w/ and w/o the 5G module.

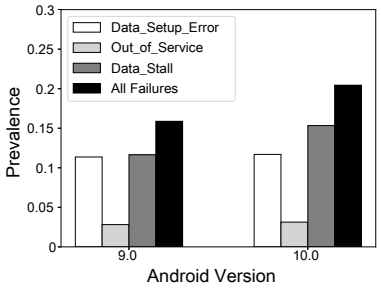


Figure 8: Prevalence of cellular failures for different Android versions.

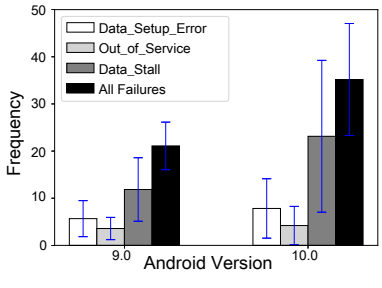


Figure 9: Frequency of cellular failures for different Android versions.

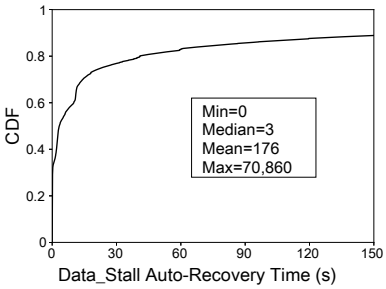


Figure 10: Most Data_Stall failures are automatically fixed in a few seconds.

failures on Android 10 phones are higher than those on Android 9 phones, as demonstrated in Figure 8 and Figure 9. 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 [6]. While these novelties can enable various high-demanding applications such as UHD video streaming and AR/VR [24], 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 of 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.

Data_Setup_Error Decomposition. When a data connection setup fails, an error code will be generated by the radio interface to describe the reasons of the Data_Setup_Error failure, based on the responses to the issued setup requests (if any) or the return values of executed modem commands. In Android, a total of 344 error codes are defined. In Table 2 we list the top 10 most common error codes (after removing false positives) and their corresponding meanings and percentages. As shown, the top 10 codes account for nearly half (46.7%) of the Data_Setup_Error failures. In addition, we find that these error codes and their related causes are quite diverse, covering cellular failures occurring at the physical layer (*e.g.,* SIGNAL_LOST and IRAT_HANDOVER_FAILED), the data link or MAC layer (*e.g.,* PPP_TIMEOUT), and the network layer (*e.g.,* INVALID_EMM_STATE).

Data_Stall Recovery. 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 [5]. According to our measurement, ~40% of the cellular failures are Data_Stall events, which however account for 94% duration of all cellular failures, thus posing broad and disruptive impact on user

[4]Given that 5G phone models can only run Android 10 (since Android 9 does not support 5G), 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 non-5G models running Android 10. Similarly, for a fair comparison between Android 9 and Android 10 regarding their impacts on cellular failures, we should only compare the phone models running Android 9 with the non-5G models running Android 10. Since the corresponding fair-comparison results are similar to those shown in Figure 6, Figure 7, Figure 8, and Figure 9, we choose not to plot additional figures to demonstrate them.

Table 2: Brief description and percentage of top 10 most common Data_Setup_Error error codes in Android.

Error Code	Brief Description	Percentage
GPRS_REGISTRATION_FAIL	Failures due to unsuccessful GPRS registration	12.8%
SIGNAL_LOST	Failures due to network/modem disconnection	7.2%
NO_SERVICE	No service during connection setup	6.5%
INVALID_EMM_STATE	Invalid state of EPS Mobility Management in LTE	4.9%
UNPREFERRED_RAT	Current RAT is no longer the preferred RAT	4.3%
PPP_TIMEOUT	Failures at the Peer-to-Peer Protocol setup stage due to a timeout	3.5%
NO_HYBRID_HDR_SERVICE	No hybrid High-Data-Rate service	2.2%
PDP_LOWERLAYER_ERROR	Packet Data Protocol error due to radio resource control failures or a forbidden PLMN	1.9%
MAX_ACCESS_PROBE	Exceeding maximum number of access probes	1.8%
IRAT_HANDOVER_FAILED	Unsuccessful transfer of data call during an Inter-RAT handover	1.6%

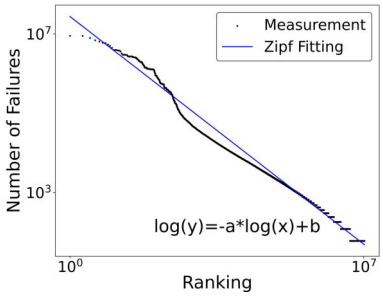


Figure 11: BS Ranking by the experienced number of cellular failures.

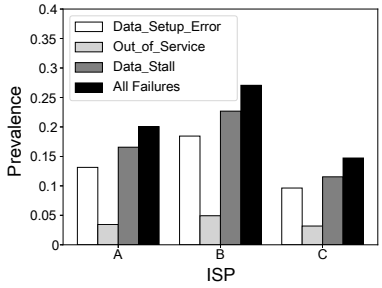


Figure 12: Prevalence of cellular failures for different ISPs' users.

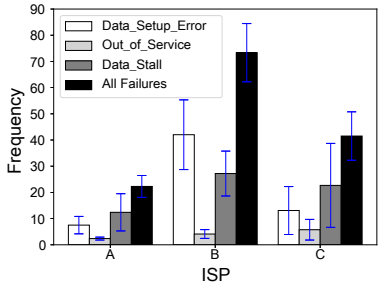


Figure 13: Frequency of cellular failures for different ISPs' users.

experiences. 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 (by constantly examining whether Data_Stall still exists).

In practice, we observe that this mechanism is quite effective—once executed, even the first-stage, lightweight operation (cleaning up the current connection) 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 10. 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 in practice.

3.3 ISP and Base Station Landscape

As mentioned in §3.1, our measurement captures a total of 2.32 billion cellular failure events with regard to 5.3M 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 [60] skewed distribution as depicted in Figure 11 (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, 44.8%, 29.4%, and 25.8% BSes belong to ISP-A, ISP-B, and ISP-C, respectively. From Figure 12, we can see that cellular failures occur more prevalently (27.1%) on ISP-B's users than on ISP-A's (20.1%) and ISP-C's (14.7%), 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 13.

Radio Access Technology (RAT). Among the involved BSes, 23.4%, 10.2%, 65.2%, and 7.3% 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

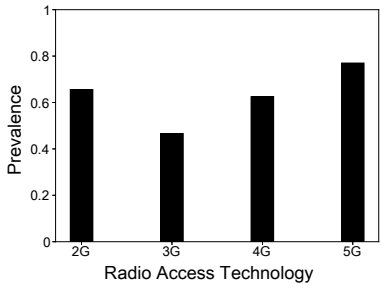


Figure 14: Prevalence of cellular failures on 2G, 3G, 4G and 5G BSes.

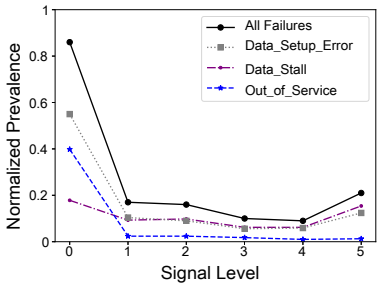


Figure 15: Normalized prevalence (or simply likelihood) of cellular failures for different signal levels.

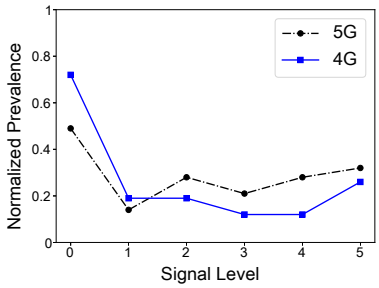


Figure 16: Normalized prevalence of cellular failures for different 4G/5G signal levels.

both the number and overall signal coverage of 3G BSes are smaller than those of 2G or 4G BSes, we observe that the prevalence of cellular failures on 3G BSes is lower than that on 2G or 4G BSes, as indicated in Figure 14. This is probably because 3G access is usually not favored by user devices when 4G access is available, and the signal coverage of 3G is much worse than that of 2G when 4G 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.

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 15 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 1) 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 [45]. 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 [12, 38], causing frequent cellular failures tagged with EMM_ACCESS_BARRED, INVALID_EMM_STATE, *etc.* [3]. This is especially the case when multiple ISPs adopt similar deployment strategies without coordinations.

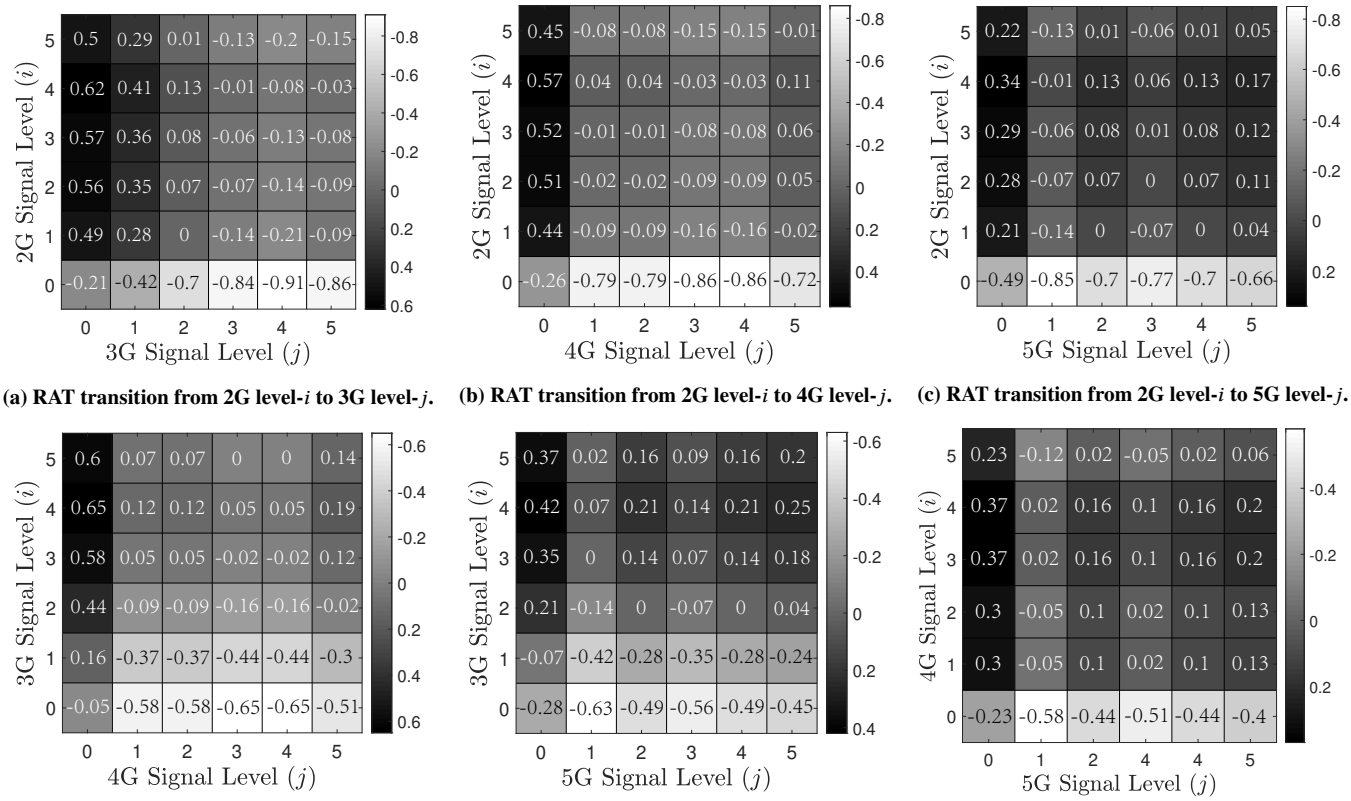
4 ENHANCEMENTS

Our multifold findings on cellular data connections failures in §3 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 (§4.1), as well as practical enhancements that have registered large-scale deployment and yielded real-world impact (§4.2).

4.1 Guidelines in Principle

In §3 we have revealed a variety of technical and business issues that could lead to or aggravate cellular failures. As elucidated in §3.2, cellular failures on 5G phones are more prevalent and frequent than the other phones (without 5G capability), most probably owing to the high network workload and immature production state of today's 5G communication modules. Thus, we suggest that mobile phone vendors be cautious when incorporating 5G modules to their products; more specifically, we encourage the vendors to comprehensively validate the new 5G modules' coordination and compatibility with existing hardware/software, so as to produce more reliable phone models in terms of cellular communication.

Also in §3.2, we note that Android 10 phones are more subject to cellular failures than Android 9 phones, 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. We have reported the discovered problems of Android 10 to its official development team, but have not got useful feedback. Hence, we propose that for the vendors, sufficient testing for new characteristics (*e.g.,* the 4G/5G switching policy) should be carried out before pushing a new OS to certain phone models.



(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.**

Figure 17: 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 17f ($i = 4$, $j = 0$) 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.

For mobile ISPs, we unravel in §3.3 that due to less workload and radio resource contention from user devices, 3G BSes are less subject to cellular failures than 2G and 4G BSes. Thereby, ISPs may consider making better use of these relatively "idle" infrastructure components to alleviate the burdens on busy 2G/4G 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 [11], 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.

4.2 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. §3.2) during RAT transition and the conservative Data_Stall recovery mechanism (cf. §3.2) 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. Both efforts have been put into practice and produced promising results.

Stability-Compatible RAT Transition. As introduced in §3.2, 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 16, 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 17f, four cases of RAT transitions (including 4G level-1 → 5G level-0, 4G level-2 → 5G level-0, 4G level-3 → 5G level-0, and 4G level-4 → 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→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 1, 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.

In addition, to achieve more smooth RAT transition, we integrate the novel 4G/5G dual connectivity mechanism advocated by 3GPP [47] on compatible devices (including all the four 5G models in Table 1). It allows a device to establish and maintain control-plane cellular connections with a 4G BS and a 5G BS simultaneously, where the master connection is also responsible for data-plane packet transfer while the slave connection is not. Then, when a RAT transition is decided, the transition process can be effectively shortened and thus would incur less disturbance to user experience.

Apart from the major case of 4G→5G transition, Figure 17 also depicts the increase of normalized prevalence of cellular failures for the other RAT transition cases. Similar as in the 4G→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 15. 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 §3.2 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 §3.2, 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 18, 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 [49].

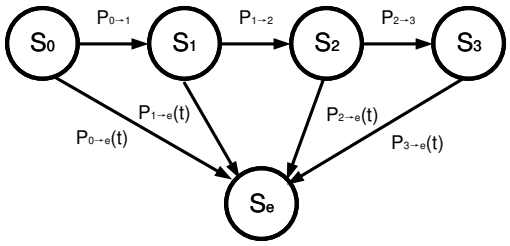


Figure 18: The time-inhomogeneous Markov process (TIMP) that models the Data_Stall recovery process in Android, where the transition probabilities among the five states are also impacted by the elapsed time (t).

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 10 (the user device can automatically fix the problem as time goes by).

Thus, using our dataset we build a *time-inhomogeneous Markov process* [49] (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 $\mathbb{P}_{i\to e}(t)$. Thereby, the probability of its not recovering (thus entering S_{i+1}) is $\mathbb{P}_{i\to i+1} = 1-\mathbb{P}_{i\to e}(\sigma Pro_i)$, 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:

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

The first part is the integral of $\mathbb{P}_{i\to e}(t)$ over the time window $[S_i, S_{i+1}]$, *i.e.,* $\int_{\sigma Pro_{i-1}}^{\sigma Pro_i} \mathbb{P}_{i\to e}(t)dt$, representing that Case-1 occurs. The second part is the probability of the device's entering the next state ($\mathbb{P}_{i\to i+1}$) multiplying the expected recovery time (T_{i+1}) after entering the next state, *i.e.,* $\mathbb{P}_{i\to i+1} \cdot T_{i+1}$, 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 $\mathbb{P}_{i\to e}$ 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 $i = 3$, $T_3 = \int_{\sigma Pro_2}^{t_m} \mathbb{P}_{3\to e}(t)dt + O_3$, 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.

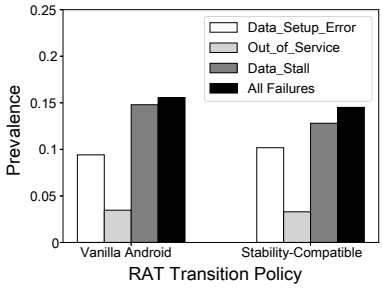


Figure 19: Prevalence of cellular failures with the RAT transition policy of vanilla Android and our Stability-Compatible RAT Transition.

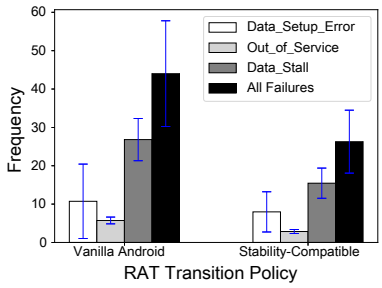


Figure 20: Frequency of cellular failures with the RAT transition policy of vanilla Android and our Stability-Compatible RAT Transition.

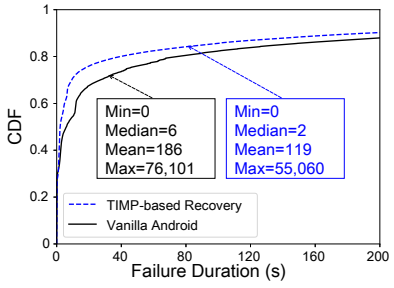


Figure 21: Duration of cellular failures with the Data_Stall recovery mechanism in vanilla Android and our TIMP-based Flexible Recovery.

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 [42] 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.* §3.2). 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.

4.3 Deployment and Evaluation

In order to validate the real-world effect of our design, we patched the above two-fold mechanisms to Android-MOD and invited the original 70M opt-in users in late Oct. 2020 to participate in our evaluation of the optimization mechanisms (§4.2). This time, 40% of the 70M users opted-in and upgraded to our patched system. The evaluation has been conducted for two months (Nov.–Dec. 2020).

As shown in Figure 19 and Figure 20, 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 §4.2). 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 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.

Further, as shown in Figure 21, after our designed TIMP-based Flexible Data_Stall Recovery mechanism is put into practice, 38% reduction on the duration of Data_Stall failures is achieved on average, corresponding to 36% reduction on the total duration of all types of failures. More notably, the median duration of all failures is remarkably reduced by 67% (from 6 seconds to 2 seconds). Most importantly, our TIMP-based recover mechanism works in a principled and flexible manner, so it will automatically adapt to the possible pattern changes of Android system behaviors and cellular reliability in the future.

At the same time, to evaluate the overhead of our patched Android-MOD, we perform small-scale benchmark experiments using 34 different phones as described in Table 1. The results demonstrate that our optimizations incur little overhead to a low-end Android phone: <3% CPU utilization, ~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 (as shown in Figure 3), the incurred CPU, memory, and storage overheads are still acceptable: <9% CPU utilization, ~3 MB of memory usage, and <20 MB of storage space; the network usage is ~20 MB per month.

5 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 [17, 19, 56, 57], user devices [12, 21, 26, 27, 34, 37, 39, 48], user-to-device interactions [15, 32, 51], and device-to-device communications [33, 44]. Researchers have also developed measurement tools and platforms for conducting cellular network measurements [23, 30, 41].

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, which is not only relatively unfamiliar to the community but also more complicated and difficult to measure and analyze. Hui *et al.* [20] 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 [28, 31] and unavailability [12, 43] 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 [50, 55, 58], energy efficiency [9, 10, 18], and security enhancements [13, 25], as well as under various scenarios like video streaming [52, 53], web browsing [54], and cellular-WiFi interaction [35, 40]. In particular, researchers have proposed to leverage lower-layer cellular information to boost the upper-layer user-perceived performance [52–55]. 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.

6 DISCUSSION ON LIMITATIONS

In §2 we present Android-MOD, which incorporates dedicated low-level system tracing and active network probing mechanisms, as well as our domain knowledge to continuously and accurately capture cellular failures at scale. With this monitoring infrastructure and its deployment in the wild, we were able to obtain a large dataset that offers us the unique opportunity of revealing multifold problems with regard to both the cellular infrastructure and mobile operating systems. Nevertheless, we have to note that our dataset and analyses still bear several limitations, owing to realistic constraints stemming from end-user devices' limited cellular context information.

In Android-MOD, upon Data_Stall occurrences, we adopt an active probing-based mechanism to accurately measure their durations and rule out possible false positives. This, however, inevitably interferes with the cellular and network environments, and thus could lead to potential errors in the measurement results. Although our evaluation (at the end of §2.2) has shown that this mechanism is unlikely to incur measurement errors given that its process is well time-bounded and lightweight, alternative passive monitoring mechanisms are still worth exploring. For example, Hui *et al.* [20] propose cross-device/network measurement that monitors network packets for uncovering the root causes of cellular problems. Similarly, Wang *et al.* [48] employ passive packet capturing in high-speed railways' on-board LTE gateways to analyze the flow characteristics of LTE networks. In this work, we do not adopt similar methodologies because capturing network flows is rather intrusive for user devices, but would like to leave other possible passive methodologies that are non-intrusive as important future work.

Limited by the cellular context information Android-MOD can collect from user devices, some of our analyses in §3 may lack direct evidence and thus may deviate from the actual cause(s) which relate to other impact factors beyond our current reach. For such cases, we have provided best-effort validations by carefully considering different aspects of the dataset. To further demystify the relevant problems and complement this work, in the future we wish to collaborate with mobile ISPs to carry out more comprehensive researches.

In addition, our study currently does not involve Android 11 as it was released after our measurement period. Nevertheless, by closely examining the concerned source code in Android 11, we find that the majority of cellular reliability problems we have revealed in this work remain in Android 11, especially the aggressive RAT transition policy and the lagging Data_Stall recovery mechanism. Therefore, our findings and proposed enhancements should also be beneficial to even the latest devices running Android 11.

7 CONCLUSION

This paper presents our efforts towards understanding and combating the reliability issues in cellular networks. Despite prior focuses on cellular performance and availability, the fundamental reliability situations are still not clearly understood at scale. We close this critical knowledge gap by conducting a large-scale, crowdsourcing-based measurement study with the help from 70 million opt-in users. 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 for the first time. 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 28 million users, generating prominent realistic impacts.

ACKNOWLEDGMENTS

We appreciate Wenli Shi, Junjie Hou, Rongyan Sun and Daliang Sun for their help in data collection and analysis. We sincerely thank the anonymous reviewers for their insightful comments, and our shepherd Lili Qiu for guiding us through the revision process. This work is supported in part by the National Key R&D Program of China under grant 2018YFB1004700, as well as the National Natural Science Foundation of China (NSFC) under grants 61822205, 61632020, 61632013 and 61902211.

REFERENCES

[1] Android.org. 2021. Android 10 Highlights. https://developer.android.com/about/versions/10/highlights.
[2] Android.org. 2021. Data Connection Management in Android. https://android.googlesource.com/platform/frameworks/opt/telephony/+/master/src/java/com/android/internal/telephony/dataconnection/DataConnection.java.
[3] Android.org. 2021. Data Fail Cause in Android. https://android.googlesource.com/platform/frameworks/base/+/master/telephony/java/android/telephony/DataFailCause.java.
[4] Android.org. 2021. Data Setup Error in Android. https://android.googlesource.com/platform/frameworks/opt/telephony/+/refs/heads/master/src/java/com/android/internal/telephony/dataconnection/DcTracker.java.
[5] Android.org. 2021. Data Stall Report in Android. https://developer.android.com/reference/android/net/ConnectivityDiagnosticsManager.DataStallReport.
[6] Android.org. 2021. Enhance Your Apps with 5G. https://developer.android.com/training/connectivity/5g/enhance-with-5g.
[7] Android.org. 2021. SMS Manager in Android. https://developer.android.com/reference/android/telephony/SmsManager.
[8] Android.org. 2021. State Out of Service in Android. https://developer.android.com/reference/android/telephony/ServiceState.

[9] Pavan K Athivarapu, Ranjita Bhagwan, Saikat Guha, Vishnu Navda, Ramachandran Ramjee, Dushyant Arora, Venkat N Padmanabhan, and George Varghese. 2012. Radiojockey: Mining Program Execution to Optimize Cellular Radio Usage. In *Proc. of ACM MobiCom*. 101–112.

[10] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. 2009. Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. In *Proc. of ACM SIGCOMM*. 280–293.

[11] Lorela Cano, Antonio Capone, Giuliana Carello, Matteo Cesana, and Mauro Passacantando. 2017. On Optimal Infrastructure Sharing Strategies in Mobile Radio Networks. *IEEE Transactions on Wireless Communications* 16, 5 (2017), 3003–3016.

[12] Haotian Deng, Chunyi Peng, Ans Fida, Jiayi Meng, and Y Charlie Hu. 2018. Mobility Support in Cellular Networks: A Measurement Study on Its Configurations and Implications. In *Proc. of ACM IMC*. 147–160.

[13] Haotian Deng, Weicheng Wang, and Chunyi Peng. 2018. Ceive: Combating Caller ID Spoofing on 4G Mobile Phones via Callee-only Inference and Verification. In *Proc. of ACM MobiCom*. 369–384.

[14] Clare Duffy. 2020. The big differences between 4G and 5G. https://www.cnn.com/2020/01/17/tech/5g-technical-explainer/index.html.

[15] Aaron Gember, Aditya Akella, Jeffrey Pang, Alexander Varshavsky, and Ramon Caceres. 2012. Obtaining In-context Measurements of Cellular Network Performance. In *Proc. of ACM IMC*. 287–300.

[16] Jann Horn. 2021. Mitigations Are Attack Surface, Too. https://googleprojectzero.blogspot.com/2020/02/mitigations-are-attack-surface-too.html.

[17] Zhenxian Hu, Yi-Chao Chen, Lili Qiu, Guangtao Xue, Hongzi Zhu, Nicholas Zhang, Cheng He, Lujia Pan, and Caifeng He. 2015. An In-depth Analysis of 3G Traffic and Performance. In *Proc. of ACM SIGCOMM Workshop (AllThingsCellular)*. 1–6.

[18] Junxian Huang, Feng Qian, Alexandre Gerber, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. 2012. A Close Examination of Performance and Power Characteristics of 4G LTE Networks. In *Proc. of ACM MobiSys*. 225–238.

[19] Junxian Huang, Feng Qian, Yihua Guo, Yuanyuan Zhou, Qiang Xu, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. 2013. An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance. *ACM SIGCOMM Computer Communication Review* 43, 4 (2013), 363–374.

[20] Jie Hui and Kevin Lau. 2013. T-Mobile QoE Lab: Making Mobile Browsing Faster and Open Research Problems. In *Proc. of ACM MobiCom*. 239–242.

[21] Keon Jang, Mongnam Han, Soohyun Cho, Hyung-Keun Ryu, Jaehwa Lee, Yeongseok Lee, and Sue B Moon. 2009. 3G and 3.5 G Wireless Network Performance Measured from Moving Cars and High-Speed Trains. In *Proc. of ACM MICNET*. 19–24.

[22] Anant Kumar, Jon Postel, Cliff Neuman, Peter Danzig, and Steve Miller. 1993. *Common DNS Implementation Errors and Suggested Fixes*. Technical Report. Oct. 1993, RFC 1536.

[23] Swarun Kumar, Ezzeldin Hamed, Dina Katabi, and Li Erran Li. 2014. LTE Radio Analytics Made Easy and Accessible. *ACM SIGCOMM Computer Communication Review* 44, 4 (2014), 211–222.

[24] Zeqi Lai, Y Charlie Hu, Yong Cui, Linhui Sun, Ningwei Dai, and Hung-Sheng Lee. 2019. Furion: Engineering High-quality Immersive Virtual Reality on Today's Mobile Devices. *IEEE Transactions on Mobile Computing* (2019).

[25] Chi-Yu Li, Guan-Hua Tu, Chunyi Peng, Zengwen Yuan, Yuanjie Li, Songwu Lu, and Xinbing Wang. 2015. Insecurity of Voice Solution VoLTE in LTE Mobile Networks. In *Proc. of ACM SIGSAC*. 316–327.

[26] Li Li, Ke Xu, Tong Li, Kai Zheng, Chunyi Peng, Dan Wang, Xiangxiang Wang, Meng Shen, and Rashid Mijumbi. 2018. A Measurement Study on Multi-Path TCP with Multiple Cellular Carriers on High Speed Rails. In *Proc. of ACM SIGCOMM*. 161–175.

[27] Li Li, Ke Xu, Dan Wang, Chunyi Peng, Qingyang Xiao, and Rashid Mijumbi. 2015. A Measurement Study on TCP Behaviors in HSPA+ Networks on High-Speed Rails. In *Proc. of IEEE INFOCOM*. 2731–2739.

[28] Yuanjie Li, Haotian Deng, Jiayao Li, Chunyi Peng, and Songwu Lu. 2016. Instability in Distributed Mobility Management: Revisiting Configuration Management in 3G/4G Mobile Networks. In *Proc. of ACM SIGMETRICS*. 261–272.

[29] Yuanjie Li, Qianru Li, Zhehui Zhang, Ghufran Baig, Lili Qiu, and Songwu Lu. 2020. Beyond 5G: Reliable Extreme Mobility Management. In *Proc. of ACM SIGCOMM*. 344–358.

[30] Yuanjie Li, Chunyi Peng, Zengwen Yuan, Jiayao Li, Haotian Deng, and Tao Wang. 2016. Mobileinsight: Extracting and Analyzing Cellular Network Information on Smartphones. In *Proc. of ACM MobiCom*. 202–215.

[31] Yuanjie Li, Jiaqi Xu, Chunyi Peng, and Songwu Lu. 2016. A First Look at Unstable Mobility Management in Cellular Networks. In *Proc. of ACM HotMobile*. 15–20.

[32] Yang Li, Jianwei Zheng, Zhenhua Li, Yunhao Liu, Feng Qian, Sen Bai, Yao Liu, and Xianlong Xin. 2020. Understanding the Ecosystem and Addressing the Fundamental Concerns of Commercial MVNO. *IEEE/ACM Transactions on Networking* 28, 3 (2020), 1364–1377.

[33] Zhenhua Li, Weiwei Wang, Christo Wilson, Jian Chen, Chen Qian, Taeho Jung, Lan Zhang, Kebin Liu, Xiangyang Li, and Yunhao Liu. 2017. FBS-Radar: Uncovering Fake Base Stations at Scale in the Wild. In *Proc. of ISOC NDSS*. 1–15.

[34] Zhenhua Li, Weiwei Wang, Tianyin Xu, Xin Zhong, Xiang-Yang Li, Yunhao Liu, Christo Wilson, and Ben Y Zhao. 2016. Exploring Cross-Application Cellular Traffic Optimization with Baidu TrafficGuard. In *Proc. of USENIX NSDI*. 61–76.

[35] Yeon-sup Lim, Erich M Nahum, Don Towsley, and Richard J Gibbens. 2017. ECF: An MPTCP Path Scheduler to Manage Heterogeneous Paths. In *Proc. of ACM CoNEXT*. 147–159.

[36] Craig J. Mathias. 2008. Opinion: How not to Build more Reliable Cellular Networks. https://www.computerworld.com/article/2537774/opinion--how-not-to-build-more-reliable-cellular-networks.html.

[37] Ruben Merz, Daniel Wenger, Damiano Scanferla, and Stefan Mauron. 2014. Performance of LTE in A High-Velocity Environment: A Measurement Study. In *Proc. of ACM SIGCOMM Workshop (AllThingsCellular)*. 47–52.

[38] Martin Klaus Müller, Martin Taranetz, and Markus Rupp. 2015. Providing Current and Future Cellular Services to High Speed Trains. *IEEE Communications Magazine* 53, 10 (2015), 96–101.

[39] Arvind Narayanan, Eman Ramadan, Jason Carpenter, Qingxu Liu, Yu Liu, Feng Qian, and Zhi-Li Zhang. 2020. A First Look at Commercial 5G Performance on Smartphones. In *Proc. of WWW*. 894–905.

[40] Ashkan Nikravesh, Yihua Guo, Feng Qian, Z Morley Mao, and Subhabrata Sen. 2016. An In-depth Understanding of Multipath TCP on Mobile Devices: Measurement and System Design. In *Proc. of ACM MobiCom*. 189–201.

[41] Ashkan Nikravesh, Hongyi Yao, Shichang Xu, David Choffnes, and Z Morley Mao. 2015. Mobilyzer: An Open Platform for Controllable Mobile Network Measurements. In *Proc. of ACM MobiSys*. 389–404.

[42] Ralph HJM Otten and Lukas PPP van Ginneken. 2012. *The Annealing Algorithm*. Vol. 72. Springer Science & Business Media.

[43] Chunyi Peng and Yuanjie Li. 2016. Demystify Undesired Handoff in Cellular Networks. In *Proc. of IEEE ICCCN*. 1–9.

[44] M Zubair Shafiq, Lusheng Ji, Alex X Liu, Jeffrey Pang, and Jia Wang. 2013. Large-scale Measurement and Characterization of Cellular Machine-to-Machine Traffic. *IEEE/ACM Transactions on Networking* 21, 6 (2013), 1960–1973.

[45] Jie Sheng, Ziwen Tang, Cheng Wu, Bo Ai, and Yiming Wang. 2020. Game Theory-Based Multi-Objective Optimization Interference Alignment Algorithm for HSR 5G Heterogeneous Ultra-Dense Network. *IEEE Transactions on Vehicular Technology* 69, 11 (2020), 13371–13382.

[46] Pyda Srisuresh, Bryan Ford, Senthil Sivakumar, Saikat Guha, et al. 2009. *NAT Behavioral Requirements for ICMP*. Technical Report. Apr. 2009, RFC 5508.

[47] 3GPP TS 37.340 v15.0.0. 2018. *3rd Generation Partnership Project; NR; Technical Specification Group Radio Access Network; Multi-connectivity; Overall description; Stage-2 (Release 15)*. Vol. 15. 3GPP.

[48] Jing Wang, Yufan Zheng, Yunzhe Ni, Chenren Xu, Feng Qian, Wangyang Li, Wantong Jiang, Yihua Cheng, Zhuo Cheng, Yuanjie Li, et al. 2019. An Active-Passive Measurement Study of TCP Performance over LTE on High-Speed Rails. In *Proc. of ACM MobiCom*. 1–16.

[49] Gerhard Winkler. 2012. *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods: A Mathematical Introduction*. Vol. 27. Springer Science & Business Media.

[50] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. 2013. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks. In *Proc. of USENIX NSDI*. 459–471.

[51] Ao Xiao, Yunhao Liu, Yang Li, Feng Qian, Zhenhua Li, Sen Bai, Yao Liu, Tianyin Xu, and Xianlong Xin. 2019. An In-depth Study of Commercial MVNO: Measurement and Optimization. In *Proc. of ACM MobiSys*. 457–468.

[52] Xiufeng Xie and Xinyu Zhang. 2017. Poi360: Panoramic Mobile Video Telephony over LTE Cellular Networks. In *Proc. of ACM CoNEXT*. 336–349.

[53] Xiufeng Xie, Xinyu Zhang, Swarun Kumar, and Li Erran Li. 2015. piStream: Physical Layer Informed Adaptive Video Streaming over LTE. In *Proc. of ACM MobiCom*. 413–425.

[54] Xiufeng Xie, Xinyu Zhang, and Shilin Zhu. 2017. Accelerating Mobile Web Loading Using Cellular Link Information. In *Proc. of ACM MobiSys*. 427–439.

[55] Yaxiong Xie, Fan Yi, and Kyle Jamieson. 2020. PBE-CC: Congestion Control via Endpoint-Centric, Physical-Layer Bandwidth Measurements. In *Proc. of ACM SIGCOMM*. 451–464.

[56] Dongzhu Xu, Anfu Zhou, Xinyu Zhang, Guixian Wang, Xi Liu, Congkai An, Yiming Shi, Liang Liu, and Huadong Ma. 2020. Understanding Operational 5G: A First Measurement Study on Its Coverage, Performance and Energy Consumption. In *Proc. of ACM SIGCOMM*. 479–494.

[57] Yin Xu, Zixiao Wang, Wai Kay Leong, and Ben Leong. 2014. An End-to-End Measurement Study of Modern Cellular Data Networks. In *Proc. of PAM*. 34–45.

[58] Yasir Zaki, Thomas Pötsch, Jay Chen, Lakshminarayanan Subramanian, and Carmelita Görg. 2015. Adaptive Congestion Control for Unpredictable Cellular Networks. In *Proc. of ACM SIGCOMM*. 509–522.

[59] Gongzheng Zhang, Tony QS Quek, Aiping Huang, and Hangguan Shan. 2015. Delay and Reliability Tradeoffs in Heterogeneous Cellular Networks. *IEEE Transactions on Wireless Communications* 15, 2 (2015), 1101–1113.

[60] George Kingsley Zipf. 1949. *Human Behavior and the Principle of Least Effort: An Introd. to Human Ecology*. Addison-Wesley Press.