SEED: A SIM-Based Solution to 5G Failures

Jinghao Zhao, Zhaowei Tan, Yifei Xu, Zhehui Zhang, Songwu Lu
University of California, Los Angeles
{jzhao,tan,yxu,zhehui,slu}@cs.ucla.edu

ABSTRACT

Failures in 5G mobile networks are becoming the norm with the ongoing global rollout. If left unattended, they affect mobile user experiences and the proper functioning of applications. In this work, we describe SEED, which offers a novel SIM-based solution to 5G failure diagnosis and handling. SEED infers failure causes by exploiting current standardized 5G error codes and decision-tree/online learning algorithms. It further takes corresponding multi-tier reset/redo actions (reset protocol operations, refresh outdated configurations, reload profiles, etc.) once the failure cause is inferred. SEED takes the operator's perspective in its design for fast deployment. SEED design works within the 5G standard framework and does not require changes on the device firmware or infrastructure hardware. Our evaluation has confirmed the viability of SEED.

CCS CONCEPTS

• **Networks** → **Mobile networks**; **Error detection and error correction**; *Network manageability*.

KEYWORDS

5G Network; Cellular Failure; Mobile Failure Diagnosis; SIM-based Failure Diagnosis

ACM Reference Format:
Jinghao Zhao, Zhaowei Tan, Yifei Xu, Zhehui Zhang, Songwu Lu. 2022. SEED: A SIM-Based Solution to 5G Failures. In *ACM SIGCOMM 2022 Conference (SIGCOMM '22), August 22–26, 2022, Amsterdam, Netherlands.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3544216.3544260

1 INTRODUCTION

Failures in the 5G mobile network have become the norm, rather than exceptions. As the system scale grows bigger and small cells are deployed, users and their smartphone applications perceive failures on a regular basis. In this work, we study the diagnosis and treatment of network failures on both control and data planes in the 5G protocol stack. As a showcase study, our analysis on public 5G traces reveals that, 2832 failure cases are uncovered from 24k control/data-plane management procedures. Other prior studies [35] also confirmed the prevalence of failures in 5G. If left unattended, such failures may incur long disruptions for 5G-based Internet access and impede the proper functioning of 5G applications.

Existing solutions to 5G failures take the modem-based scheme [3] or the OS-centric approach [12] at the device. They use the timeout-based detection with multiple timers, and take the sequential retry (i.e., level-by-level from transport, to 5G protocols, to radio module) approach to failure recovery. Consequently, such detection and reaction schemes are ill-suited for the complex 5G failure cases and result in prolonged service disruptions ranging from tens of seconds to tens of minutes. The fundamental problem is that, 5G failures are highly diversified; they arise in the wide spectrum of control- and data-plane managements, and data packet delivery. Without cooperation from the infrastructure and the device to facilitate fine-grained diagnosis, neither approach learns the error causes, but takes the blind, sequential retry scheme to failure management.

In this work, we present SEED, a novel solution to 5G failure diagnosis and treatment. SEED offers the SIM-based, pure software solution. It provides fine-grained failure detection and recovery at runtime by leveraging information from both the device and the 5G network. It exploits the currently available SIM-network communication channel, which is conventionally used for mutual authentication and add-on services [51], for a novel usage: collaborative failure diagnosis. SEED thus leverages the 5G standardized messages to infer failure causes, and devises domain-specific decision-tree and online learning algorithms to enhance failure cause inference. It further uses multi-tiered reset/redo (e.g., reset protocol message operations, refresh the outdated configurations, reload the profiles) for differentiated failure treatments. In contrast to complex failure recovery mechanisms (e.g., rollover, logging and checkpointing, state consistency management), SEED uses simple, yet effective resets to handle all three categories of control-plane, data-plane, and data delivery failures. Moreover, SEED leverages the existing signaling messages defined by the 5G standards to enable real-time failure diagnosis and handling. Consequently, the signaling with diagnosis information could still be transmitted even when the data session is not established or broken upon failures. Our empirical evaluations over the testbed have confirmed that, SEED can reduce the disruption time from 12.4~476.0s by the current modem/OS-based schemes to 0.4~8.0s, a factor 0.6×~792× of reduction. The incurred extra battery usage at the device is about 1.2%.

The design of SEED takes the mobile operator's view, who we believe is in the best position for fast deployment and paced in sync with the ongoing 5G global rollout. The design modules of SEED running at the SIM applet, as well as added operations at the carrier app, can be readily installed when a subscriber activates her/his 5G smartphone; this is already the common practice today. SEED offers two modes with and without root privilege. The SIM and carrier app updates can leverage the current practice via the OTA channel for software upgrade. The infrastructure-side design of SEED is also divided into phases for ease of incremental deployment. Operators could deploy SEED with software updates at their core network to enable diagnosis messages and perform online learning. Furthermore, SEED does not pose new security threats. Everything is under the operator's control. Normal users cannot access/modify its operations. Finally, SEED works within the 5G framework and does

SIGCOMM '22, August 22–26, 2022, Amsterdam, Netherlands

ACM ISBN 978-1-4503-9420-8/22/08.
https://doi.org/10.1145/3544216.3544260

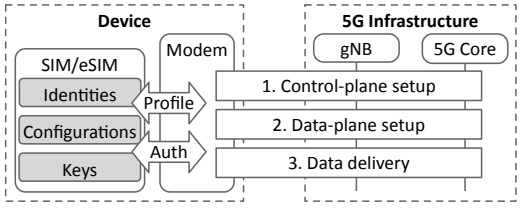


Figure 1: 5G device/infrastructure components

not require changing the current device firmware or infrastructure hardware. It is also applicable to 4G LTE.

2 BACKGROUND

5G cellular network primer In 5G, mobile devices rely on two key modules to access the network, the SIM/eSIM[1] and the modem. The SIM stores critical subscriber information, including user identities, configurations, and security keys, etc. The modem loads the needed information from the SIM to register the device to the network. To this end, the SIM runs its various programs (called applets) to handle profile transmissions and authentication.

For data transfer, the mobile device proceeds with three stages as shown in Figure 1. First, the modem sets up the control plane via 5G signaling messages, including identity exchange, mutual authentication between the SIM and the network, location update, etc. Second, the modem and the infrastructure exchange signaling messages for the data-plane setup[2]. This step completes data-plane configurations including device IP address, and DNS server, etc. Finally, the device starts data packet delivery for its applications.

Cellular network failures Failures have become the norm rather than an exception in real-world 5G usage. Recent studies show that, 30% of 5G devices experience failures during an 8-month measurement [35]. Various failures are also reported by users each day [21, 22]. From the technology perspective, 5G is using small cells in the mmWave bands with smaller single-cell coverage, thus triggering more frequent handovers [62]. Therefore, 5G may incur more failure events with the increased frequency to sync up the control-plane state, security update, and data sessions.

Failures in 5G are consequently becoming diversified. In general, three types of failures arise at different stages of data transfer. Control-plane management failures arise during the control signaling operations, including setup, tracking area update, or teardown procedures. Data-plane management failures affect the data session setup, modification, or release procedures. Data delivery failures incur packet delivery stall with established data-plane sessions due to various causes (e.g., DNS errors, port blocking, etc).

These failures happen at both device and network sides. The device may use outdated configurations, resulting in connectivity failure [56]. The network could suffer from congestion, thus unable to respond to requests in time [67]. Every component (hardware, cellular stack, OS, etc.) could be the source of failure [35], given the diversity and complexity of devices and infrastructure in both hardware and software.

Cellular failure diagnosis & handling In principle, failures need to be diagnosed with their root causes and handled with differentiated actions. Given the limited capability to control the device behavior from the infrastructure, current failure diagnosis and handling are usually performed on the device side. There are two popular approaches at the modem and the OS, respectively.

The modem implements the 5G-specific protocol stack in the firmware. It thus handles control/data-plane management failures [3]. It identifies the failed procedures based on standardized protocol messages and their finite state machines, and decides whether to abort the connection or trigger retransmissions. For example, if the modem fails to receive any response from the network for Registration Request, it will wait for T3511 (10s by default) and then retry. After five such attempts, the modem waits for the longer timer T3502 (12mins by default) and proceeds to retry. The timeout varies for different procedures.

In the OS-based solution, OS implements more detection and reaction mechanisms. The popular Android uses new mechanisms of "timeout based probe and restart" for data delivery failures [12]. It monitors the network statistics, and periodically sends DNS and HTTPS requests for failure detection. Android failure detection can be categorized into three classes: connection issue to a preset URL[3], TCP failure[4], and consecutive DNS timeouts[5]. Android first queries the current connection list from the modem, and handles all such failures with the same progressive approach of "sequential retry": clean up and restart all current TCP connections, re-register to the network, and restart the modem. Android sets a three-minute timer between each action if the previous action did not recover the data connection successfully.

3 LIMITATIONS OF CURRENT SOLUTIONS

Despite the current solutions at the device, 5G users still regularly perceive data service related failures, even with a strong radio signal bar at the smartphone [43]. We thus seek to first understand how failures exhibit in practice given the deployed failure mechanisms through a trace analysis. We then assess the existing modem and Android failure handling. Our results show that both solutions only perform coarse-grained diagnoses and cannot handle diversified failures well. Existing modem schemes incur repeated failures and long disruptions, and Android suffers from prolonged failure detection and false positives.

3.1 Failures in the Real World

In this work, we focus on failures related to the 5G protocol stack. The network failures induced by other components, such as internet outages, erroneous application implementations, or OS firewall settings (e.g., user-determined network restrictions for apps), are out of scope. We first analyze control/data-plane management failures in 5G. The 5G standard provides failure causes to indicate failure reasons for control/data-plane management [3], which inherit from LTE with 5G context. We perform our trace analysis on the publicly available, 6.7TB 5G/4G datasets collected from 2015-Q3 to 2021-Q4. These public datasets include 4.7 million signaling messages collected by 30+ device models using open-source tools of MobileInsight [36] and MI-LAB [42]. The traces contain 8 mobile carriers from the US and China. We found 2832 failure cases from

[1] In this paper, we use SIM to denote both SIM and eSIM for a slight abuse of notation.
[2] This is called bearer setup in 5G standards [1].
[3] Android captive portal domain: connectivitycheck.gstatic.com0.
[4] TCP failure rate exceeds 80%, or over ten outbound packets but no inbound packets during the last minute.
[5] Five consecutive DNS timeouts within 30 minutes.

Class	Failure Causes
Control Plane (56.2%)	UE identity cannot be derived by the network (15.2%)
	No Suitable Cells In tracking area (12.6%)
	PLMN not allowed (10.3%)
	No EPS bearer context activated (7.5%)
	Message type not compatible with the protocol state (2.8%)
Data Plane (43.8%)	Requested service option not subscribed (7.9%)
	Invalid mandatory information (5.9%)
	User authentication failed (4.7%)
	Request rejected, unspecified (2.6%)
	Insufficient resources (1.9%)

Table 1: Top 5 failure causes in control/data plane

24k control/data-plane management procedures; this gives a non-trivial, over 10% failure ratio per control/data-plane management lifespan. Table 1 shows the top 5 failure causes in the control and data plane management. We elaborate on them next.

For control plane management, 3 out of 5 most frequent failures are due to infrastructure-device status synchronization. The infrastructure fails to derive the updated device identity (15.2%), releases previous data bearer context (7.5%), or sends mismatched signaling (2.8%). One common reason for the unsynchronized status is that, when the device moves to a new tracking area after handover, the infrastructure fails to sync up states with the previous tracking area. With state mismatch, the device suffers from long disruptions during reattaching with outdated identities and contexts.

For data plane management, the top 2 failures are due to configurations (requested service option not subscribed, and invalid mandatory information). Although the configurations could be proactively checked from the network side, the operational failures cannot be completely eliminated in practice. The configurations could be outdated on the device and result in data plane failure. When such failures happen, the infrastructure only provides failure causes without the correct, up-to-date configurations. The device thus fails to recover, and repeated failures arise. In addition to those failures from outdated configurations, diverse failures are exhibited, including security check (user authentication failed), insufficient resources, etc. Failures caused by expired subscriptions can only be recovered with user actions such as reactivating the data plan. Reject messages may also include unspecified causes that are seen at the infrastructure or devices.

During data delivery, the three most common failures incur data stall in 5G cellular networks: TCP, UDP, and DNS [35, 48, 62]. The TCP anomaly is observed in operational 5G networks [48]. The UDP port blocking is also widely reported by users under 5G deployment [54]. For DNS failures, public DNS services such as Google DNS typically do not apply to cellular networks. Carriers usually configure users' DNS with their local DNS resolvers (LDNS), which is less stable due to user mobility and congestion [50]. Although operators' DNS servers may work correctly during the device registration, they may experience outages thereafter. Neither Android nor iOS provides default DNS configuration for backup, which makes devices difficult to recover from carrier DNS failures [7]. Users have reported DNS failure instances among operators [15, 53].

3.2 Limitations of Modem Scheme

The current modem-based solution handles both control- and data-plane management failures. However, it does not perform fine-grained diagnosis or take precise actions for different failures. Note that it could have obtained the standardized failure causes from the signaling messages, but did not leverage them for fine-grained diagnosis. The modem either aborts the connection and retracts to idle, or triggers retransmission upon timeout. Our analysis shows that, the modem might keep on resending the signaling message with outdated status, which causes repeated control-plane management failures until the modem reboots. When outdated configurations further trigger data plane management failures, the modem cannot update them. Repeated failures are observed thereafter.

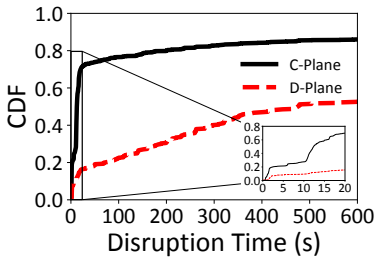


Figure 2: Disruption time with existing modem handling for control and data plane management failures

Empirical Validation We measure the disruption time with the existing modem handling scheme using traces in §3.1. As shown in Figure 2, 50% of control-plane setup failures cause more than 12.4s of disruptions. Only 19% of failures could be recovered within 2s. The modem tries to reattach when various timers are triggered after 10s. However, the timeout prolongs the disruption, and only 27% of failures are recovered within 10s. Repeated failures happen when the modem retries with previous data-plane configurations. For example, when the access point name (APN) is outdated, 5G data plane setup fails. The modem activates reattachment, but still uses the previous APN during the data-plane setup, making the device fail for the same reason repeatedly. The frequent, repeated failures prolong the disruption. Only 9% of data plane management failures could be recovered within 10s. Half of the failures need about 8 minutes to be recovered.

With limited information from the network side, current modems cannot ensure fine-grained failure diagnosis and recovery. Furthermore, modem-based solutions may suffer from three problems when collaborating with the network: First, multiple parties (operators, modem vendors, etc.) need to follow the same protocol for collaborations, requiring a long time to be standardized and deployed; Second, operators may not want to leak their network-side information to third parties (e.g., modem vendors); Third, the security for modem-network collaboration requires extra infrastructure support (e.g., public key infrastructure) and increases deployment cost.

3.3 Android: Insufficient Diag & Handling

Android further monitors failures of the data delivery stage. However, it suffers from limited detection schemes and long detection latency. First, Android does not check for those failures related to UDP, which is widely used in WebRTC, QUIC, etc., for 5G IoT and real-time applications. Second, Android provides a timer-based failure detection without distinguishing application requirements, thus resulting in prolonged disruption for all applications. For example, while video streaming apps could tolerate seconds of disruptions with a large buffer, 5G AR/VR apps fail to function properly with 100ms disruption [38].

Solutions	Failure Detection & Diagnosis?	Failure Recovery (No-user-action Required)		Failure Recovery (User-action Required)
		Config-related	Non-config-related	
Modem-based	Only device-side	Not support	Timer-based retry	Not support
OS-based	Only device-side	Not support	Layer-by-layer retry	Not support
App-based	Only device-side	Not support	Transport reconnection	Not support
Infra-based	Only infra-side	Infra-side config updates	Waiting for device retry	User Notification
SEED	Both infra & device-side	Both-side config updates	Multi-tier reset	User Notification

Table 2: Comparison of different solutions for 5G failure diagnosis and handling

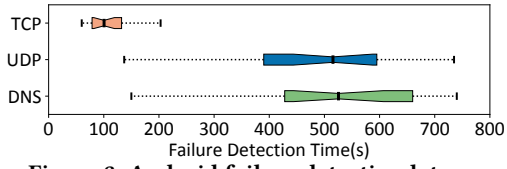


Figure 3: Android failure detection latency

Android takes the sequential retry approach upon detecting failures via timeout. Retry is an effective solution that is also suggested by operators [16, 55]. However, This level-by-level retry actions by Android yield long time intervals between two actions. While the OS-based solution could effectively check the device-side firewall, user-determined app data restrictions, etc., the limited device-side information cannot make it elude those failures from the cellular network. Without fine-grained failure diagnosis, sequential retry incurs prolonged application disruption or even no recovery. For example, if the TCP failure is caused by underlying control/data plane failures, refreshing TCP connections cannot help.

Empirical Validation We assess Android detection for TCP, UDP, and DNS failures on the latest Android 12. We connect the device with Magma cellular testbed [39]. After the device successfully acquires the data service, we block TCP, UDP, and DNS queries, respectively, at the core network. During experiments, we play the same background Youtube video and visit websites from the browser every 5 seconds to simulate daily usage scenarios. We measure the failure detection latency from the time when failure happens to the instant Android reports data stall.

Figure 3 plots the latency distribution for different failure detections. For TCP failures, Android takes 1.8 minutes on average to report data stall. We note DNS and UDP failures are not well dealt with in Android. Results show that 50% of DNS failures cannot be detected within 8.7 minutes. For UDP failures, Android could only detect it if the failure leads to consecutive DNS timeouts; it takes 8 minutes on average. Otherwise, UDP failures could not be detected. Furthermore, we test Android when connections to the preset URL are blocked; this simulates failures due to server issues. Android still reports data stall alerts, which causes false positives. It further triggers recovery actions and disrupts existing connections. The interval between level-by-level reset actions is 3.5 minutes on average, which results in long disruptions.

In summary, both schemes take the device-based approach and suffer from several downsides. First, the restricted device-side information limits the fine-grained failure diagnosis and recovery schemes. Both modem and OS based solutions incur prolonged disruptions upon failures with their timeout-based detection. Second, neither exploits available error causes carried by 5G messages. Specifically, the modem could leverage the embedded failure causes carried by the reject signaling messages for diagnosis. However, it did not. Android maps part of standardized failure causes with DataFailCause [13], but did not use them in failure diagnosis and recovery. Third, failure handling is simplistic. The sequential retry (i.e., level-by-level recovery) leads to long disruptions. The naïve retry by modem does not infer the causes, thus unable to fix failures due to outdated configurations. Simple retry further aggravates congestion upon failures of cell/core overload.

3.4 Solution Space

In addition to the modem-based and OS-based solutions, there exist some application-based proposals. MobileInsight [36] provides an in-device failure detection through continuous monitoring of the diag port messages. Commercial tools such as NetMotion [44] leverage a mobile application to report high-level metrics (e.g., device types, network performance, etc.) to pre-deployed servers for failure analysis. Although the application could detect failures, the recovery is limited. Applications without root can only take the transport-layer reconnection action, which cannot recover cellular stack failures. Even if the app owns root access, similar to modem/OS-based solutions, simple retries can only recover failures from temporary infrastructure-device status unsynchronization, but not failures caused by outdated configurations. Moreover, for failures that require user actions to recover (e.g., expired data plan subscription, identity authentication failures, etc.), the device cannot obtain enough hints about failures to take proper actions. Furthermore, existing SIM add-on services (e.g., [51]) could only monitor the SIM hardware health (say, read/write cycles), cell signal strengths, etc., but cannot diagnose complex protocol failures. In conclusion, the device-side solutions cannot acquire the network-side information for fine-grained failure diagnosis and handling.

While the device-side approach has limitations, the second solution option is an infrastructure-based scheme. This choice also has several limitations in both failure detection and reaction. First, the infrastructure may not have access to higher-layer information (e.g., transport and app layers), thus unable to infer high-layer failures accurately. Second, monitoring data traffic over high-speed 5G may incur significant processing overhead. Third, the infrastructure cannot differentiate which case happens in the absence of data traffic: whether misconfiguration blocks the device's traffic, or the device is idle without data to transfer. While the infra-based solution could send failure notifications through other channels (e.g., email), it lacks device control for failure reactions. Although the infrastructure could acquire standardized causes for control/data-plane management failures, or even pinpoint the root cause (outdated configurations, customized policies, etc.), it can only update infra-side configurations for failure recovery but cannot notify devices at runtime with corresponding action commands (e.g., update SIM configurations). We summarize the existing solutions to 5G failure diagnosis and handling in Table 2.

The third option is to let the device and the infrastructure collaborate in failure diagnosis and reactions. The device can detect high-layer failures reliably and take direct low-level reset actions

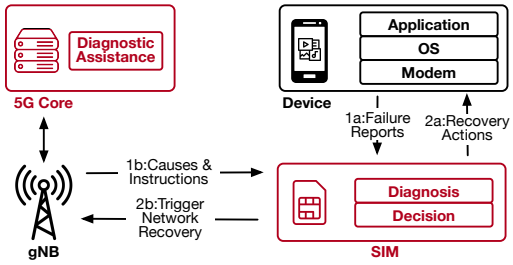


Figure 4: SEED diagram

via modem. The infrastructure can directly correct misconfigurations and use crowdsourcing among devices to infer failure causes. However, a naive approach in this option also suffers from three limitations. First, the device and infrastructure might not communicate failure-related messages upon failures. Second, exposing failures may compromise system security. Third, the solution may not work within the 5G framework.

4 SEED: SIM-BASED FAILURE DIAGNOSIS

4.1 Case for SIM-based Solution

We make a case for a SIM-based solution that addresses all the above limitations. First, the SIM and the network can communicate via signaling messages over signaling channels, rather than data packets. The channel subsists even when the data session is not established or broken. Second, SIM is produced by operators and trusted by the in-network devices. In-SIM keys could further protect the SIM-network communication without extra certificates. Therefore, SIM bridges the network and devices as a trusted agent. Adversaries cannot access the SIM info without the in-SIM keys. Third, SIM is applicable to nearly all cellular-connected devices. New functions could be deployed in the form of a SIM applet. They can be upgraded with the over-the-air (OTA) mechanisms on existing SIM/eSIM. Last, the SIM-based scheme offers a purely software-based solution without changing 5G standards, device firmware, or infrastructure hardware.

4.2 SEED Overview

We thus design SEED, a SIM-based solution to 5G failure diagnosis and handling. SEED offers a software-based scheme deployable by 5G mobile carriers. It offers lightweight, fine-grained failure detection and recovery at runtime with SIM's constrained hardware capability. SEED leverages 5G standardized messages to infer failure causes inside the SIM. It further uses multi-tiered resets/redos for differentiated failure handling. In contrast to complex failure recovery (e.g., rollover, logging and checkpointing, recovery from crashes), SEED uses simple, yet effective resets to handle all three types of control-plane, data-plane, and data delivery failures.

Figure 4 shows the overall system diagram of SEED. First, SIM receives failure reports from applications (1a) and the network (1b). Failure reports include failure clues such as network-side diagnosis, instructions with updated configurations, and device-side failure details (no connection, DNS/UDP failure, etc.). With such clues, SIM performs local diagnosis, makes handling decisions, and triggers recovery actions at the device (2a) or the network (2b). SEED addresses three issues in its SIM-based design:

• **How does SIM pinpoint failures at low overhead?** We ensure the solution is viable on resource-constrained SIM hardware. To this end, SEED combines standardized failure causes with the up-to-date configurations from the infrastructure, as well as OS/App failure reports from the device. SEED further performs fine-grained failure diagnosis with limited SIM processing and storage.

• **How does SIM handle diverse failures that arise at different stages?** We develop simple and fast failure recovery via multi-tier reset. SIM could perform profile reloads, configuration updates and failure notifications on commercial off-the-shelf devices without root access. It further supports faster control/data plane resets with root privilege.

• **How does SIM collaborate with the infrastructure when the data plane is broken?** The SIM obtains information from the infrastructure for fine-grained diagnosis and handling. We leverage existing signaling messages to transmit diagnosis information, thus ensuring runtime SIM-network information exchange upon failures of control/data-plane management or data delivery.

4.3 Lightweight SIM Diagnosis

We now introduce how the SIM performs fine-grained failure diagnosis with both-side information. For control-plane and data-plane management failures, the SIM receives the standardized 5G failure causes from the infrastructure, and leverages them for diagnosis. The SIM further enables apps to report data delivery failures for fast detection and diagnosis.

4.3.1 Failures in control/data-plane management. The 5G standardized failure causes provide a good source for SIM diagnosis. 5G defines 80+ failure codes, which are embedded in signaling messages. Most of the messages containing failure codes are the "reject" messages from the network or the device, such as Authentication Reject, PDU Session Modification Reject, etc. The messages that embed standardized causes have been widely deployed in practice [19].

SEED achieves lightweight and fine-grained SIM diagnosis with such standardized causes. When the infrastructure composes the reject or receives device reject messages, it extracts the embedded standardized cause and sends the cause code to the SIM (more details in §4.5). The SIM applet stores all standardized cause codes and looks up the received cause to quickly detect and pinpoint the failure. Although the SIM's storage is limited (32∼128KB), it is sufficient to hold all cause codes for in-SIM analysis. The causes could be categorized into control-plane management and data-plane management. Control-plane management causes include failures related to UE identification, subscription options, network congestion, authentication, invalid messages, etc. Data-plane management causes include configuration failures and protocol errors. Standardized causes are already supported at the infrastructure and do not need extra modules or algorithms, thus resulting in marginal overhead.

SEED further exploits SIM capability to prevent repeated failures. If the failure cause is related to outdated configurations, simple retries cannot succeed but result in repeated failures. Therefore, when the infrastructure initializes the reject due to outdated device configurations, it sends the up-to-date configuration together with the cause code to the SIM. We list the failure causes related to outdated configurations in Appendix A. Upon receiving these cause codes, the SIM parses the configurations based on the cause code and stores them for next-step handling (§4.4).

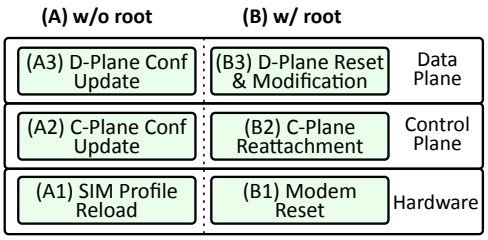


Figure 5: Multi-tier reset w/ and w/o root privilege

4.3.2 Failures in data delivery. For packet delivery failures, current user applications cannot diagnose them with limited low-layer information from the mobile OS. Emerging 5G applications are disruption-sensitive and require quick recovery. SEED thus enables these applications to report failures for fast diagnosis. Applications could call the failure report API if they need fast failure handling. The API carries three parameters (failure type, traffic direction, address). The failure types support the three most common failures discussed in §3.1: DNS, TCP, and UDP. The application indicates the failed traffic direction, including uplink, downlink, or both. The address contains the IP and port for TCP/UDP failures. These fields are used by the 5G Traffic Flow Template (TFT) to regulate the traffic and activate IP/port blocking with incorrect configurations. The domain names are embedded in the address field for DNS failures. The report enables disruption-sensitive apps to bypass long Android detection and speed up the diagnosis. SIM further leverages existing APIs to acquire the Android data stall notification. Whenever receiving the App/OS failure reports, SIM leverages the reported information for fast failure handling in §4.4. Note that SEED does not explicitly diagnose and handle instantaneous, underlying radio link failures. However, such physical-layer issues may affect control/data-plane management and data delivery, which will be observed at higher layers and handled by SEED.

4.4 SIM-Based Failure Handling

With the diagnosis result, we address the next issue of reacting to diverse failures that arise at different stages. Different from the blind retry by the modem and Android, SEED handles diverse failures via the multi-tier reset mechanism directly, facilitated by both-side information to pinpoint the failure, thus leading to fast failure recovery. We first list the multi-tier reset actions taken by the SIM. We then elaborate on how the SIM decides which action to take accordingly.

4.4.1 Handling with Multi-tier Reset. SEED leverages the SIM to initiate multi-tier reset actions. This is nontrivial, since the SIM does not directly control those cellular connections. SEED explores the limited interfaces supported by current 5G devices, and designates two modes with different device privileges. Without root privilege, SEED-U uses the multi-tier reset to reload the failed module directly. When root privilege is available, SEED-R further improves the recovery speed.

Multi-tier reset without root access SEED-U takes multi-tier reset actions for failure handling at three levels as shown in Figure 5(A). At the hardware level, reset enforces the modem to clear its cached contexts, preventing it from being stuck in prolonged attempts with invalid caches. The SIM triggers profile reloading at the modem to sync its SIM profiles (A1). Different from the naive retry scheme in the current modem/OS, the SIM also retrieves the latest configurations from the infrastructure to handle outdated configurations. The SIM updates the control-plane configurations (A2) (e.g., PLMN list) to reduce excessive search time. The mismatched control-plane states/identities (shown in Table 1) are also refreshed in the reset. SEED-U leverages the proactive commands between the SIM and the modem to realize these two actions [23]. The proactive command is usually used to provide carrier services such as OTA updates, which has been supported by current smartphones without root privilege. To our knowledge, SEED-U is the first to leverage it for failure handling. SEED-U could further update the acquired data-plane configurations from the SIM (A3), such as DNNs or APNs, leveraging the Android carrier app [14]. All such actions do not require root privilege at smartphones.

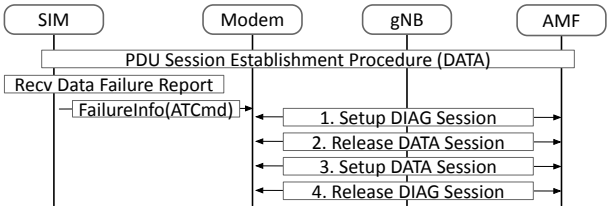


Figure 6: Reset data plane without reattachment

Boost multi-tier reset with root privilege With root privilege, SEED-R further improves granularity and speed for diagnosis. 5G/4G devices provide AT commands [4] as another interface for fine-grained modem control but require root privilege. When the carrier app detects that root access is permitted, it will notify the SIM through APDU to enable the SEED-R mode. Figure 5(B) shows the multi-tier reset with root privilege. Upon hardware failures, SIM restarts the modem (B1). It recovers the modem from being stuck in internal errors. For control-plane failures, SEED-R directly controls the modem for control-plane reattach (B2), which improves the recovery speed without prolonged search procedures.

SIM further collaborates with the infrastructure for data-plane resets (B3). Resetting only the data session speeds up the handling. However, 5G gNB releases the last radio bearer once the last data session is released, thus causing the control-plane reattach. SEED designs the fast data plane reset in Figure 6 without resetting the control plane. Upon receiving failure reports about the initial data session, the SIM triggers the modem to set up another data session with DNN "DIAG". The reattach will not happen when "DATA" session is released as "DIAG" session and corresponding 5G gNB radio bearer still exist. Finally, the device reconnects "DATA" session and releases the "DIAG" session. The network could also modify the existing "DATA" bearer rather than reset it, if only configurations (e.g., TFT) need to be updated. All the session setup/release/modification signalings are standardized in 5G [3]. SEED leverages them to handle data delivery failures without disrupting the existing, established data plane.

4.4.2 Deciding on Reset Actions. Resetting different modules require different latencies. To speed up failure handling, the SIM uses the diagnosis results and the current mode to perform targeted reset action without layer-by-layer retry.

Table 3 shows the SIM handling decisions without root (SEED-U) and with root privilege (SEED-R). For control-plane management failures not caused by outdated configurations, SEED-U reloads the

Diagnosis Class	Failure Handling w/o Root (SEED-U)	Failure Handling w/ Root (SEED-R)
Control-plane Causes	SIM Profile Reload (A1)	Reset Modem (B1)
Control-plane Causes w/ Config	Parameter Update & Profile Reload (A2 & A1)	Control-Plane Reattachment with Update (B2)
Data-plane Causes	SIM Profile Reload (A1)	Data-plane Reset (B3)
Data-plane Causes w/ Config	Configuration Update (A3)	Data-plane Modification (B3)
Data Delivery Failures Reported by App/OS	Configuration Update (A3)	Data-plane Reset / Modification (B3)

Table 3: Failure handling decisions with diagnosis results

SIM profile to reattach (A1). With root privilege, SEED-R performs modem reset using AT commands (B2). When outdated configurations incur failures, SEED-U updates the control-plane parameters and reloads SIM profiles for registration (A2 & A1). With root privilege, SEED-R updates the configurations and triggers reattach on modem for fast handling (B2). As 20% of control-plane management failures could be recovered within 2s (§3.2), SEED sets a 2s timer before triggering hardware and control plane reset. The short timeout enables speedy recovery upon such failures.

When data-plane management failures arise, with SEED-U, the SIM triggers the SIM profile reloading. The data plane will be reset after the control-plane reattach. With root access, SEED-R triggers data-plane reset for faster failure handling (B3). When the SIM acquires data-plane configurations (DNN, PDN type, etc.) from the infrastructure, the SIM further triggers configuration updates (A3) with SEED-U or data-plane modification (B3) with SEED-R.

SIM may receive applications/OS reports for data delivery failures. If it received causes on control/data-plane management failures within the last 5s, there could be an ongoing handling. The SIM does not trigger handling to avoid conflict. Otherwise, the SIM triggers the configuration updates in the carrier app (A3) to reset data connection without root. SEED's rate-limit design does not perform the same reset action consecutively and frequently; the signaling messages are thus not overwhelming. With root access, the SIM sends the failure report collected from App/OS (§4.3.2) to the infrastructure with real-time SIM-Infra collaboration (details in §4.5). The infrastructure checks if the failure type, direction, and address conflict with user policies, or if DNS failure happens. It then modifies the data session with updated user polocies when conflict arises for TCP/UDP, or configures a new DNS server in the followup reset (B3).

4.5 Real-Time SIM-Network Collaboration

We next address the issue of enabling SIM-infra interaction when the data plane is broken, without changing modem/gNB firmware or modifying standardized messages. Note that SIM needs to acquire the information of failure causes and updated configurations from the infrastructure. SIM also notifies the infrastructure for data-plane resets. Although SIM OTA provides a channel for SIM-Infrastructure communication [51, 52], it relies on data service (e.g., TCP/UDP) and cannot work during connection setup. Moreover, upon data delivery failures, packets may not be delivered and SIM OTA is unavailable.

SEED leverages standard-compliant control-plane signaling messages. The infrastructure embeds the failure-related info in Authentication Request signaling messages. SIM embeds failure diagnosis results in PDU Session Establishment Request to trigger data-plane resets. SEED design is compatible with 5G commodity devices without modem or gNB firmware modifications. These messages are available in the presence of failures. Hence, the infrastructure and

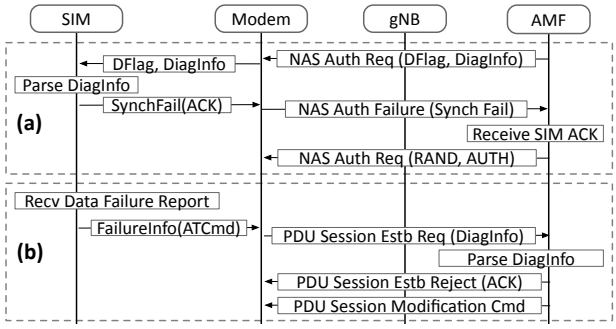


Figure 7: Collaboration with standard-complied signaling messages (a) Network to SIM (b) SIM to Network

the SIM could perform real-time interactions for failure diagnosis and handling, even when the data bearer is not up or broken.

How Does Infrastructure Deliver Info to SIM? In SEED, the network leverages the 5G Authentication Request message used for mutual authentication to send diagnosis info. In this message, the network generates a 16-byte RAND and a 16-byte AUTH, which the modem will forward to the SIM for authentication [30]. SEED reserves a RAND value (FF…FF) as the Diagnosis Flag (DFlag). As shown in Figure 7(a), the 5G network embeds the diagnosis info in the AUTH field and sets the RAND as DFlag. When the SIM sees the reserved DFlag, it does not verify the key but parses the AUTH, which is encrypted and integrity protected with a counter using the pre-shared in-SIM key. SIM returns synchronization failure as the ACK upon successfully receiving the diagnosis. If the sync failure is not responded, the modem may label the network as untrusted. The network then resends a normal Authentication Request. The 16B AUTH suffices to hold the cause code and most updated configurations. The network could embed more information with multiple transmission rounds. Note that, the network can send Auth Request at any time with a NAS signaling connection [3]. Although the control plane is not fully established (with successful completion of both authentication and configurations), the network could still transmit Auth Request to the SIM, thus enabling collaboration in the presence of control/data-plane failures.

How Does SIM Transfer Info to Infrastructure? Data delivery failures may block packet transfer with data plane set up. SEED embeds the failure report collected by SIM in the PDU Session Establishment Request to report data delivery failures, as shown in Figure 7(b). After control-plane setup, the device requests the data bearer with a corresponding Data Network Name (DNN) in 5G [3]. Standards support sending DNNs for multiple data sessions, which enables SEED to report failures anytime after control plane setup with a new request. SEED leverages the undefined field to embed the diagnosis information in the DNN field [2], which is also encrypted and integrity protected using the in-SIM key. The 100B DNN size is sufficient for the current report; our experiments further validate

that, a longer report triggering multiple consecutive requests can be fragmented into several DNNs. SEED triggers the modem to send the request with a special diagnosis DNN starting with "DIAG". After the network gets the DNN and parses the report, it validates the failure with current user settings, responds with a reject as ACK, and modifies the current data session configs or follows Figure 6 for the reset.

The real-time SIM-Infrastructure collaboration is compatible with 5G devices. The gNB/modem firmware remains unchanged. Carriers could update the SIM via OTA for new applet logic. The network-side functions could be extended with a new core module handling diagnosis messages, which current cloud-based core network implementations could quickly deploy [17, 39]. SEED only requires a small amount of extra signaling messages for collaboration with marginal overhead at the device and the network. Note that, the real-time collaboration cannot work if the radio access is broken. The radio link issues and recoveries are well studied [6, 47, 49]. The real-time collaboration in SEED is designed to supply communication channels, in the presence of failures on control/data-plane management and data delivery.

5 ENHANCED FAILURE MANAGEMENT

5.1 Insufficient Standardized Causes

Standardized causes provide a good source for failure diagnosis. However, they are insufficient for devices in three aspects. First, they cannot cover all control/data plane failures. Failures from operators' customized policies, such as the supported device list [60], do not fit into any standardized causes. Second, the data delivery failure could happen due to gNB/core congestion. Without knowing it, the reset may further increase the loads. Third, the failure's root cause could be unspecified without a recovery action. The coarse-grained information is insufficient for failure recovery.

Therefore, the SIM needs more information for failure diagnosis. Thus, we leverage the infrastructure assistance for SIM diagnosis to cover diverse failures (§5.2). We further show how SEED automatically handles failures with an unknown root cause (§5.3).

5.2 Infra-Assisted SIM Diagnosis

We first introduce infrastructure assistance for the diagnosis. This component leverages the deployed metrics in the infrastructure, which avoids redundant processing and is scalable for massive devices. We then elaborate on how SIM diagnoses failures with information from the infrastructure.

Infrastructure Assistance The infrastructure classifies failures with a decision tree as shown in Figure 8. It then sends the corresponding assistance information to the SIM with real-time collaboration (§4.5). The assistance information includes four types: failure causes, suggested configurations, suggested reset actions, and congestion warnings. SEED acquires them from the existing monitoring and management functions in current 5G infrastructure [31, 40] to assist failure diagnosis without complex processing.

The infrastructure classifies the failures into two types: passive and active. The passive type includes failures not initialized by the infrastructure, such as device response timeout, device reject, or SIM-reported data delivery failure. Standardized causes are sent to the SIM as §4. For customized failures, it sends suggested reset actions for SIM handling. It further notifies the SIM with cell/core congestion. The active type includes network-initialized rejects. In addition to standardized failures, the infrastructure provides customized causes with suggested actions to cover failures from customized policies. For causes without suggested actions, we propose an online learning algorithm to handle them (§5.3).

SIM Diagnosis SIM receives the four types of assistance info and performs the following actions accordingly. The SIM applet stores all supported failure causes and assistance info parsing functions. They follow a similar decision tree scheme at the network side and could be deployed with limited SIM processing and storage. SIM handles standardized failures and refreshes configurations as in §4.4. The SIM performs the suggested reset action for customized failures, enabling operators to deploy handling for new failures. When the SIM receives the congestion warning, it does not trigger the reset but waits for a timer embedded in the message. SIM parses the assistance information from the infrastructure, and handle diverse standardized and customized failures accordingly with the multi-tier reset. It further notifies users of failures requiring user actions to recover (e.g., reactivating the data plan).

However, there are still failures causes without corresponding handling. The infrastructure may map unstandardized causes to policies or modules but do not have any clues for the device handling. We further design an online learning algorithm to handle failures without a known handling action. We elaborate on it next.

5.3 SIM Handling with Online Learning

While the root cause is unclear, previous devices' successful handling probably works for new devices facing the same failure. Although failures may appear from different functions in one module, the multi-tier action directly resets the whole module and has similar effectiveness for various devices. SEED proposes an online learning algorithm to crowdsource the handling history from SIMs, and picks out suggested actions when the same failure happens in the future. The infrastructure and SIMs keep evolving and automatically train the model for failures with unknown handling.

Algorithm 1: Collaborative Online Learning

```
def SIM-RecvUnknownFailure(cause):
    for action ← [B3, A3, B2, A2, B1, A1] do
        if DoRecovery(action) == success then
            SIMRecord[cause][action] ← +1
            break
    if SendToInfra(SIMRecord) == success then
        SIMRecord = dict[][]
def Infra-Crowdsource(SIMRecord):
    for cause, action ← SIMRecord do
        NetRecord[cause][action] ←
          +SIMRecord[cause][action]
def Infra-SendUnknownFailure(cause):
    if cause ∈ NetRecord then
        sgstAction = argmax(NetRecord[cause])
        if rand() < 1/(1+e^(−lr∗size(NetRecord[cause]))) then
            SendtoSIM(cause, sgstAction)
            return
    SendtoSIM(cause, null)
```

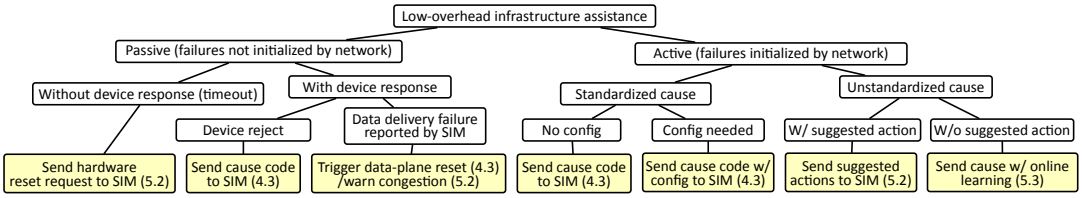


Figure 8: Low-overhead infrastructure assistance classifies failures and sends assistance information to the SIM

The online learning algorithm (Algorithm 1) includes the SIM side (line 1-7) and infrastructure side (line 8-17). When the infrastructure does not know the handling action for a failure, it generates a customized code to identify the failure, such as the conflicting policies or modules. The cause code is sent to the SIM through SIM-Network collaboration (§4.5). SIM tries all the supported retries and resets sequentially from the data plane to the hardware (line 2). It records the successful handling that resolves the issue and notifies the infrastructure with OTA (line 3-7). The infrastructure crowdsources SIM records and updates the network-side record (line 8-10). It then sends out suggested action for part of later devices controlled by a learning rate lr. Otherwise, the infrastructure does not suggest any actions, ensuring that the model is trained and evolving (line 14). If the suggested handling failed, the SIM takes the same action as receiving unknown failure and tries all the supported actions sequentially. In our online learning procedure, the SIM only stores customized error codes and corresponding actions. The data volume is small enough to be held within the limited SIM storage. With the collaboration between SIMs and the infrastructure, online learning provides automatic failure handling for unknown causes. The decision model also evolves gradually without heavy training, which is lightweight and scalable for massive devices.

6 IMPLEMENTATIONS

Figure 9 shows the implementation of SEED. The operator owns controls for all SEED components in practice, including the infrastructure module, SIM applet, and the carrier app.

Solution prototype We develop a SIM applet on Javacard-based eSIM [66], which is compatible with most mobile OS (e.g., Android, iOS, etc.). The applet contains 1244 lines of Java with two modules. The diagnostic module receives the infrastructure assistance information through the modem with APDU interface [23], and app/OS failure report through the carrier app with TelephonyManager API [10]. The decision module uses SEED-U mode by default for the multi-tier reset. For SEED-U mode, the decision module sends proactive commands through APDU to the modem for profile reloading and control-plane configuration updates, and updates data-plane configurations with the carrier app. SIM is notified by the carrier app when root privilege is available. It then enables SEED-R mode and sends AT commands listed in Appendix B to the carrier app for faster failure handling.

We extend the Magma 5G NSA core [39] with a plugin to assist SIM diagnosis with 1035 lines of C++. The diagnosis assistance module hooks the reject generation functions to acquire the standardized failures. It acquires the latest configurations from the orchestrator API [41] and extra information such as RAN/core load from Magma NMS [40]. We extend the orchestrator API to receive SIM recovery records and forward them to the assistance module for online learning (§5.3). The real-time collaboration module reuses

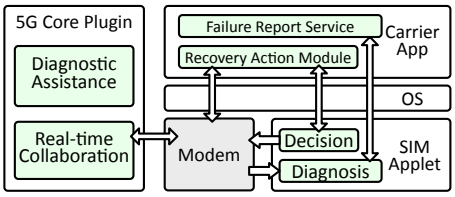


Figure 9: SEED implementation components

the Auth Request functions and hooks the PDU establishment handling function. The information is encrypted with 128-EEA2 and integrity protected with 128-EIA2 using the pre-shared in-SIM key to prevent information leakage and malicious requests.

We develop a carrier app with Android UICC Privilege API [14] to update carrier configurations, and receive data stall notifications, etc. It contains two modules with 842 lines of Java. The failure report service receives app reports with Android Service [9] and OS reports with Connectivity Diagnostics API [11]. For unrooted devices, the recovery action module updates configurations with UICC Privilege API. If it detects root privilege with Runtime API [8], it notifies the SIM to enable SEED-R mode for it to trigger AT commands.

Deploying SEED in practice The operators have access to all components that SEED involves, including the core, SIM, and the carrier app. Therefore, SEED is a viable solution that can be deployed by operators alone, without any help from modem or phone vendors. Besides, SEED extends the current standard without changing any existing protocols. Deploying SEED thus does not affect any operating 5G functions.

Incremental deployment Operators can gradually deploy SEED, as a partial implementation already diagnoses some failures. They can first deploy infra and SIM Applet modules to support diagnosis and handling of control/data-plane failures. These modules can cover 63% of failure cases in the traces. The first stage is easy to deploy: all necessary info for the network module can be extracted from core signalings while the SIM applet could be updated through readily-available OTA channel. The operators can then update the carrier app to include failure report service and action module. The carrier app has been widely deployed by operators [59, 61]. With the enhanced failure handling added, *all* considered failures in this paper can be diagnosed and handled.

7 EVALUATION

We evaluate how SEED diagnoses and handles failures. We first evaluate the overall performance on the testbed with failures in our datasets. We also compare the application performance between SEED and existing failure handling schemes. We then assess SEED overhead, diagnosis time, and recovery speed for multi-tier reset with and without root privilege.

Experimental Setup We implement the diagnosis assistant module with 5G-compliant Magma Core [39] on an Ubuntu 18.04

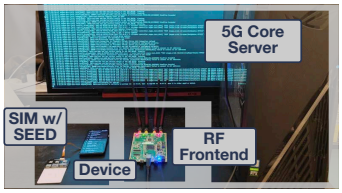


Figure 10: Experimental testbed setup

Failures	Handling	Median	90th
Control Plane	Legacy	12.4	1024.0
	SEED-U	8.0	76.7
	SEED-R	4.4	48.6
Data Plane	Legacy	476.0	2659.4
	SEED-U	0.9	1.0
	SEED-R	0.6	0.7
Data Delivery	Legacy	31.2	45.7
	SEED-U	1.1	1.3
	SEED-R	0.4	0.7

Table 4: Disruption (s) percentile with legacy handling and SEED

Apps	C-plane (s)			D-plane (s)			D-Delivery (s)		
	Leg.	S.U	S.R	Leg.	S.U	S.R	Leg.	S.U	S.R
Video	68.3	1.1	1.0	184.5	0.0	0.0	75.0	0.0	0.0
Live Stream	79.2	4.3	3.5	199.2	1.5	1.1	105.4	0.5	0.0
Web	80.3	6.8	5.4	200.8	1.8	1.6	110.5	0.8	0.3
Navigation	78.3	5.0	4.1	199.9	1.3	1.2	106.7	0.2	0.0
Edge AR	81.9	6.7	5.7	201.9	2.6	2.1	108.2	1.3	0.4

Table 5: Average app disruption (s) with legacy (Leg.) failure handling, SEED-U (S.U) and SEED-R (S.R)

server with i7-9700K 8-core CPU and use USRP B210 as the RF frontend (Figure 10). Our testbed utilizes the 5G-NSA. The long timers are shared between SA and NSA and dominate the disruption time during failures. Moreover, most of SA failure cause codes have inherited NSA error codes. Although SA splits its core functions into different components, the signaling processing and transmission times between SA and NSA incur negligible difference for SEED. The results thus also reflect SEED performance for SA deployment. We deploy the SEED applet on a Javacard-based eSIM [66] with 180KB EEPROM and 8KB RAM. We assess the performance on Google Pixel 5 with Qualcomm Snapdragon 765G running Android 12.

This work does not raise any ethical issues.

7.1 Overall Performance

7.1.1 Comparison of failure diagnosis and handling. We first examine SEED overall performance. To compare it with the current modem-based solution, we utilize the dataset in §3.1 to evaluate how many control/data plane management failures SEED could handle. We extract failure traces from the dataset and reproduce failures on the testbed to assess SEED handling. For data delivery failures, we compare the current Android scheme and SEED handling.

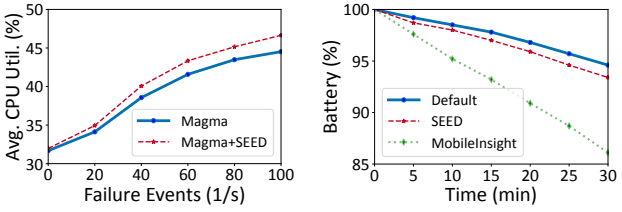
For control-plane management failures, 89.4% of failures in the dataset could be handled by SEED. The remaining cases are due to identity authentication failures from unauthorized subscribers. Table 4 compares the disruption time with existing device failure handling and SEED. Without root privilege, SEED-U could reduce the median disruption time by a factor of 0.6× (12.4⟶8.0s). SEED-R further speeds up the recovery and reduces the 90th percentile disruption by 20× (1024.0⟶48.6s). The waiting timer (2s in testing) in SEED before triggering control-plane failure handling ensures that the transient failure will not be delayed by reset. With the timer, SEED control-plane handling only causes longer disruption for 5% (SEED-U) and 2% (SEED-R) failures.

For data-plane management failures, SEED handles 95.5% of the cases in the dataset with its configuration update and fast data-plane reset. Other cases are from expired subscribers and require reactivation of their data plans. The failure handling without root privilege could recover 90% of cases with <1s disruption. With the root privilege, half of the failure cases could be recovered within 0.6s, which reduces the disruption time by 792×. SEED prevents the long disruptions incurred by repeated, blind retries at devices.

We further evaluate how well SEED handles data delivery failures. Our experiments show that, if data delivery failures are induced by widely reported incorrect network-side configurations (e.g., TCP/UDP blocking, etc.), Android or modems' naive retry schemes cannot recover from them. Current application-level tools (e.g., NetMotion [44]) also rely on the ongoing data connection and cannot report the failure under traffic blocking. In contrast, we validate that SEED successfully transmits failure reports, which can trigger network-side policy checking and updating for failure recovery. For failures that could be recovered from reconnections (outdated gateway status in mobility, etc.), we compare the choices of Android sequential retries and SEED multi-tier reset. The Android timers between recovery actions are set to the recommended configuration values (21s/6s/16s) in [35]. Despite with shorter timers, Android still incurs more than 31.2s disruptions for 50% of cases. In contrast, SEED fast data-plane reset and modification handle all cases in the experiments, and recover the data connection within 0.4s for 50% of cases and 0.7s for 90% of cases.

7.1.2 Reducing Application Disruption. We further examine the failure impact on various applications with current device handling schemes and SEED. In the experiment, we assess both the SEED-U and SEED-R modes. We measure the average app disruption time on five types of latency-sensitive applications, including video (with YouTube [64]), live streaming (with Twitch [57]), Web browsing (with Chrome [18]), navigation app (with Google Maps [28]), and an edge AR application developed by us. The video app has its long buffering time (~30s) while the live streaming possesses a shorter buffer (~3s). The Web browser visits the social network site, and the navigation app periodically uploads its location for the latest traffic information. The AR app keeps sending the camera view to the edge and retrieves real-time recognition results without a video buffer. We collect traffic traces of five applications and develop a background daemon to emulate the corresponding app's traffic pattern and send failure reports for the application.

SEED reduces failure recovery time for all five applications, as shown in Table 5. The fast failure report scheme and multi-tier reset allow the video app to tolerate all data-plane management and data-delivery failures in experiments. SEED reduces disruptions by up to 67× (68.3⟶1.0s) for control-plane failures. For live streaming with a short buffer, SEED reduces the average control-plane failure time to 4.3s (SEED-U) and 3.5s (SEED-R). For data delivery failure, SEED-R could still handle such cases and mask user-perceived disruptions. For Web browsing, SEED reduces disruptions from 80.3~200.8s to 0.3~6.8s for various failures. SEED also reduces the navigation app disruptions from 78.3~199.9s to 0.2~5.0s (SEED-U) or 0~4.1s (SEED-R). The AR app is the most disruption-sensitive app. Although Android is reconfigured with a shorter action timer, its limited detection scheme takes more than 1 minute to detect the data stall failure and recovers after 108.2s disruptions on average. In contrast, SEED takes the fast data-plane reset approach, and recovers the AR service within 1.3s (SEED-U) and 0.4s (SEED-R) on average.



(a) Network-side (b) Device-side

Figure 11: Diagnosis overhead on network/device side

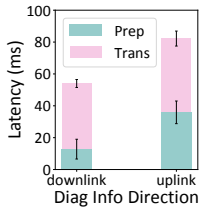


Figure 12: SIM-Infra Collaboration Latency

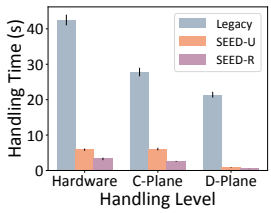


Figure 13: Recovery time for multi-tier reset

7.2 Micro-Benchmarks

7.2.1 Lightweight Failure Diagnosis. We next examine the SEED scalability at the network and the overhead at the device. Our experiments confirm that, SEED is scalable to large device population, and lightweight with low overhead at devices. For the network, we use the magma RAN/UE emulator to emulate loads in the core. We emulate 200 devices performing attach/detach procedures randomly and trigger failure events with different frequencies. Figure 11a shows the average CPU utilization with the default magma core with and without SEED. SEED incurs only 4.7% extra CPU processing, in the stress test of artificially injecting 100 failures per second. SEED scales with decision-tree-based failure diagnosis without heavy processing. The number of extra signaling messages (Auth Request/Failure or PDU Session Estb Request/Reject) from the real-time collaboration is marginal compared with the normal control/data plane procedures.

We further gauge the overhead at the device side. The SEED diagnosis is based on SIM's built-in processor and RAM, which is more energy efficient compared with the phone's CPU. By default, we measure the device battery consumption without background application traffic. We then run a stress test that triggers the SIM diagnosis once per second to quantify its energy overhead. As shown in Figure 11b, SEED consumes an extra 1.2% (5.4%→6.6%) of the total battery in 30 minutes. Given that the failure frequency in our test is much higher than in reality, the SIM diagnosis incurs negligible overhead. We further compare SEED with the device-side cellular diagnosis application MobileInsight [36]. MobileInsight relies on the message decoded from the diag port for analytics and consumes an extra 8.5% (5.4%→13.9%) of the total battery in 30 minutes. SEED thus performs lightweight diagnosis at the device.

7.2.2 Real-time SIM-Network Collaboration. SEED enables the real-time collaboration with standard-compliant signaling messages. Figure 12 shows the total latency of network-to-device (downlink) and device-to-network (uplink) directions. For the downlink, when the network detects the failure, it first prepares the message with extra information and encodes it into the Authentication Request, which takes 12.8ms on average. The transmission takes 41.2ms on average from the message sent out to the ACK received. On the device side, SEED provides APIs for App/OS failure report to speed up the failure detection. The preparation includes information reporting, SIM encoding, and message generation, which takes 35.9ms on average. Then the transmission takes 46.3ms on average to notify the network side for further actions. Compared with the Android failure detection, which needs 1.8 minutes to detect the failure, SEED speeds up the failure detection stage with fast SIM-Infra collaboration.

7.2.3 Multi-tier Reset. With diagnosis information from both sides, SEED performs the multi-tier reset for fast recovery. Compared with the legacy level-by-level sequential retry, SEED directly resets the corresponding module, eliminating long waiting interval between actions. For baseline, we use Android sequential retry with the recommended intervals (21s/6s/16s) between four actions in [35]. Although these intervals are much shorter than the Android default 3-min interval, it still causes a long time to trigger handling actions.

As shown in Figure 13, the legacy scheme takes 42.5s on average to reset the hardware. Without root access, SEED takes 5.9s on average for hardware reset. SEED further speeds up the hardware reset with root privilege leveraging AT commands, which takes 3.3s and reduces 92% waiting time. For control-plane reset, the legacy scheme takes 27.8s. The SIM parameter updates (A2) need to be combined with reloading to trigger the control-plane reset and take 6.1s. With root privilege, the control-plane reattachment takes 2.6s for control-plane reset. The legacy scheme does not reset the data plane but all TCP connections, which still needs 21.4s to trigger the failure handling. SEED triggers the carrier app to update configurations for the data-plane reset (A3) and designs fast data plane reset/modification with root privilege (B3), which takes 0.88s and 0.42s, respectively. SEED multi-tier reset shows a fast failure handling without long level-by-level retries.

7.2.4 Online Learning. The current public datasets do not provide the infrastructure-side failure traces. We validate the effectiveness of the online learning algorithm by triggering failures at our testbed. In our experiments, 6 phones of different models (Google, Xiaomi, etc.) are connected to the testbed network. On the network side, we choose 4 control-plane and 4 data-plane functions and manually trigger failures for each function 50 times to generate unstandardized failures. The network customizes failure codes based on the failed function and performs online learning for future recovery suggestions. Our results show that the crowd-sourced SIM records correctly classify all failures into control or data plane failures and recommend corresponding reset actions, which shows the effectiveness of the online learning algorithm.

7.3 Security Analysis

Our analysis shows that SEED does not degrade the legacy SIM security. The applet could only be installed with the carrier's key; adversaries cannot modify or replace it. Only the operators could update the SIM configurations, or perform reset from the SIM. The SIM-infra communication is encrypted and integrity protected with the pre-shared in-SIM key with the message counter, which uses the same crypto algorithms as the 5G signaling. The new applet interface within the assistance information is protected with cellular-grade security. The in-SIM key is hard to be compromised

by attackers, thus the information is hard to be faked and skip the SIM checking. The applet leverages existing channels provided by the mobile OS to communicate with the modem/carrier app, which does not induce new loopholes. At the device, only the application matching SIM-embedded signatures could acquire the carrier privilege. The carrier app ensures SIM security by isolating direct communication between user apps and SIM. The carrier app further checks and filters the failure report inputs to ensure security.

8 RELATED WORK

Mobile failure diagnosis has been an active topic for years. It is an important area, as a correct diagnosis result helps optimize device performance [20, 33, 34, 37, 65] or fixes RAN/core [5, 46]. Unfortunately, such diagnosis typically relies on the one-side information, either at the device [36, 62, 63] or inside the network [27, 45]. The lack of panoramic view makes diagnosis slow and error-prone. On the other hand, our failure diagnosis combines the information from both sides with a novel SIM-based solution that enables collaboration between the device and the network. To the best of our knowledge, SEED is the first work that proposes SIM-based failure diagnosis. Current commercial diagnosis tools such as NetMotion [44] collect both-side information, but only monitor the high-level metrics (e.g., network performance, connection drop rate, etc.) without failure diagnosis in the cellular stack. The collaboration between the network and the device also halts once the data connection is broken upon failures [52]. In contrast, SEED diagnoses cellular failures and performs runtime handling even if the data connection is broken or not fully established.

Prior efforts focus on other issues and cannot achieve failure handling with decent speed and accuracy. [35] measures cellular reliability and optimizes the existing Android error handling scheme; it lacks fine-grained failure diagnosis and handling. Meanwhile, network-side diagnosis [46] can barely help the devices recover from failures. Our work bridges the network and the device, and achieves runtime, fine-grained failure handling. Unlike conventional data center network failure diagnosis [24, 68] that utilizes active probing [29] or trace monitoring [32], our design leverages the 5G standardized failure causes with readily accessible metrics for diagnosis and handling, thus keeping the solution light-weight and scalable.

9 DISCUSSION

SEED focuses on failure diagnosis and treatment for the 5G protocol stack, but does not explicitly diagnose and handle underlying radio link issues or application-level failures. Such physical-layer or app-level issues may affect control/data-plane management and data delivery. The resulting failures will be implicitly detected and handled by SEED. SEED can be further extended to support radio condition diagnosis with runtime modem measurements and gNB information on dynamic radio signals.

SEED leverages the undefined fields of the 5G signaling messages within the 3GPP standards, thus being compatible with current mobile OSes and radio access networks. SEED still requires changing the SIM applet and the core network. For SIM/eSIM, operators could update the applet through OTA. The eSIM uses a programmable chip to store SIM profiles from different operators. eSIM supports the SIM's applet format, and SEED applet could be directly applied. For the network side, operators with the cloud-based core network implementations could deploy SEED with software updates at the core. A new module can be added to handle diagnosis messages and perform online learning. SEED only introduces a small amount of extra signaling, thus unlikely to trigger anomaly detection deployed by operators. If false alarm is triggered, operators may readjust the related filter rules for diagnosis messages.

The SEED-R mode requires root access to send the AT commands. The current standard has supported the SIM to trigger AT commands directly at the modem with proactive commands [23]. It has been deployed on some IoT modems [58], but not on current 5G smartphone modems yet. If the modem enables the interface, SEED becomes a rootless solution. With the current modems, if the OS provides APIs for the carrier app to initialize AT commands without root, SEED could also become rootless.

SEED can be adapted to diagnose new 5G functionalities. One upcoming feature is network slicing [25, 26], where failure could arise to a given slice. Although this increases the complexity of detection and handling, SEED enables fine-grained diagnosis and handling. Therefore, it could reset or modify the failed network slice without affecting other functioning slices.

10 CONCLUSIONS

The global rollout of 5G mobile systems is underway. Similar to every large-scale networked system, failures become the norm, rather than exceptions, in 5G. This has been confirmed by recent empirical studies [35, 62]. As 5G is going to higher radio spectrum and the cell size is getting smaller, frequent handovers further aggravate the chances for failures. If left unattended, such failures will affect the normal operations of 5G applications, particularly those emerging ones (e.g., AR/VR/MR), due to prolonged network disruptions. The current solutions do not diagnose the error causes and use the blind, sequential retry approach to failure handling.

In this work, we describe the design, implementation, and evaluation of SEED, a novel SIM-based solution to 5G failure diagnosis and handling. SEED leverages the available error codes carried by standardized 5G signaling messages for root cause inference. It further enhances the diagnosis with a simple, domain-specific machine learning algorithm. SEED takes adaptive, multi-tier reset/redo actions (reset protocol operations, refresh outdated configurations, reload profiles, etc.) once the failure cause is inferred. Our evaluation has confirmed the viability of SEED.

For fast adoption and deployment, we take the operator's view in the design of SEED. As the global 5G rollout is ongoing, we believe the operator is in the best position for 5G failure management solutions. The components of SEED can be readily installed to 5G subscribers when they activate their devices with the carrier. The software updates can be easily completed with the current operators' practice. We are working with a prime US operator for assessment and early trials. In the broader context, we believe 5G failure management needs more activities from the research community; this work describes our initial effort along this direction.

Acknowledgements. We greatly appreciate our shepherd Prof. Zhi-Li Zhang, and anonymous reviewers for their intellectual insights and constructive feedback. The work has been substantially improved with their advice. This work is partly supported by NSF CNS-1910150 and NSF CNS-2008026.

REFERENCES

[1] 3GPP. TS38.331: NR; Radio Resource Control (RRC); Protocol specification, Dec. 2020.

[2] 3GPP. TS24.008: Mobile radio interface Layer 3 specification; Core network protocols; Stage 3, Apr. 2021.

[3] 3GPP. TS24.501: Non-Access-Stratum (NAS) protocol for 5G System (5GS); Stage 3, Sep. 2021.

[4] 3GPP. TS27.007: AT command set for User Equipment (UE), Oct. 2021.

[5] Ahmad, M., Jafri, S. U., Ikram, A., Qasmi, W. N. A., Nawazish, M. A., Uzmi, Z. A., and Qazi, Z. A. A low latency and consistent cellular control plane. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (2020), pp. 648–661.

[6] Alhammadi, A., Roslee, M., Alias, M. Y., Shayea, I., and Alraih, S. Dynamic handover control parameters for lte-a/5g mobile communications. In *2018 Advances in Wireless and Optical Communications (RTUWO)* (2018), IEEE, pp. 39–44.

[7] Almeida, M., Finamore, A., Perino, D., Vallina-Rodriguez, N., and Varvello, M. Dissecting dns stakeholders in mobile networks. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies* (2017), pp. 28–34.

[8] Android runtime api. https://developer.android.com/reference/java/lang/Runtime, Jan. 2022.

[9] Android service api. https://developer.android.com/reference/android/app/Service, Jan. 2022.

[10] Android telephonymanager. https://developer.android.com/reference/android/telephony/TelephonyManager, Jan. 2022.

[11] Connectivity diagnostics api. https://source.android.com/devices/tech/connect/connectivity-diagnostics-api, Jan. 2022.

[12] Android. Data connection tracker. https://cs.android.com/android/platform/superproject/+/master:frameworks/opt/telephony/src/java/com/android/internal/telephony/dataconnection/DcTracker.java, Jan. 2022.

[13] Android. Datafailcause. https://developer.android.com/reference/android/telephony/DataFailCause, Jan. 2022.

[14] Android. Uicc carrier privileges. https://source.android.com/devices/tech/config/uicc, Jan. 2022.

[15] Dns server issue. https://forums.att.com/conversations/att-internet-equipment/dns-server-isnt-responding/5e7a4e47fd0835122f9dd15c, Mar. 2020.

[16] Suggestions to recover mobile data. https://www.att.com/support/article/wireless/KM1349609/, Jan. 2022.

[17] Azure for operators. https://azure.microsoft.com/en-us/industries/telecommunications/, Jan. 2022.

[18] Chrome. https://www.google.com/chrome/, Jun. 2022.

[19] Cisco. Failure disconnect reasons reference. https://www.cisco.com/c/en/us/td/docs/wireless/ucc/smf/2021-01-0/SMF_Metrics_Ref/b_ucc-5g-smf-metrics-ref_2021-01/m_failure-disconnect-reasons-reference.html, Jan. 2022.

[20] Deng, H., Li, Q., Huang, J., and Peng, C. icellspeed: increasing cellular data speed with device-assisted cell selection. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (2020), pp. 1–13.

[21] DownDetector. Failures reported on t-mobile. https://downdetector.com/status/t-mobile/, Jan. 2022.

[22] DownDetector. Failures reported on verizon. https://downdetector.com/status/verizon/, Jan. 2022.

[23] ETSI, T. 102 223:"smart cards. *Card Application Toolkit (CAT)* (Jul. 2019).

[24] Fang, C., Liu, H., Miao, M., Ye, J., Wang, L., Zhang, W., Kang, D., Lyv, B., Cheng, P., and Chen, J. Vtrace: Automatic diagnostic system for persistent packet loss in cloud-scale overlay network. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (2020), pp. 31–43.

[25] Network slicing inches closer to reality. https://www.fiercewireless.com/5g/marek-s-take-network-slicing-inches-closer-to-reality, Nov. 2021.

[26] Foukas, X., Patounas, G., Elmokashfi, A., and Marina, M. K. Network slicing in 5g: Survey and challenges. *IEEE Communications Magazine 55*, 5 (2017), 94–100.

[27] Foukas, X., and Radunovic, B. Concordia: teaching the 5g vran to share compute. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (2021), pp. 580–596.

[28] Google map. https://www.google.com/maps, Jun. 2022.

[29] Guo, C., Yuan, L., Xiang, D., Dang, Y., Huang, R., Maltz, D., Liu, Z., Wang, V., Pang, B., Chen, H., et al. Pingmesh: A large-scale system for data center network latency measurement and analysis. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (2015), pp. 139–152.

[30] ISO/IEC. Identification cards—integrated circuit cards—part 4: Organization, security and commands for interchange. *International Organization for Standardization/International Electrotechnical Commission. ISO/IEC 7816-4* (2013).

[31] Kakkavas, G., Stamou, A., Karyotis, V., and Papavassiliou, S. Network tomography for efficient monitoring in sdn-enabled 5g networks and beyond: Challenges and opportunities. *IEEE Communications Magazine 59*, 3 (2021), 70–76.

[32] Kim, C., Sivaraman, A., Katta, N., Bas, A., Dixit, A., and Wobker, L. J. In-band network telemetry via programmable dataplanes. In *ACM SIGCOMM* (2015), vol. 15.

[33] Li, Y., Deng, H., Li, J., Peng, C., and Lu, S. Instability in distributed mobility management: Revisiting configuration management in 3g/4g mobile networks. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science* (2016), pp. 261–272.

[34] Li, Y., Deng, H., Peng, C., Yuan, Z., Tu, G.-H., Li, J., and Lu, S. icellular: Device-customized cellular network access on commodity smartphones. In *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)* (2016), pp. 643–656.

[35] Li, Y., Lin, H., Li, Z., Liu, Y., Qian, F., Gong, L., Xin, X., and Xu, T. A nationwide study on cellular reliability: measurement, analysis, and enhancements. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (2021), pp. 597–609.

[36] Li, Y., Peng, C., Yuan, Z., Li, J., Deng, H., and Wang, T. Mobileinsight: Extracting and analyzing cellular network information on smartphones. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking* (2016), pp. 202–215.

[37] Li, Y., Yuan, Z., and Peng, C. A control-plane perspective on reducing data access latency in lte networks. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking* (2017), pp. 56–69.

[38] Liu, X., Vlachou, C., Yang, M., Qian, F., Zhou, L., Wang, C., Zhu, L., Kim, K.-H., Parmer, G., Chen, Q., et al. Firefly: Untethered multi-user vr for commodity mobile devices. In *2020 USENIX Annual Technical Conference USENIX ATC 20* (2020), pp. 943–957.

[39] Magma core. https://www.magmacore.org/, Jan. 2022.

[40] Magma nms. https://docs.magmacore.org/docs/nms/nms_arch_overview, Jan. 2022.

[41] Magma orchestrator rest api. https://docs.magmacore.org/docs/next/orc8r/dev_rest_api, Jan. 2022.

[42] Milab. http://milab.cs.purdue.edu/, Jan. 2022.

[43] mobile Community, T. Good signal/bars but no network connections. https://community.t-mobile.com/network-coverage-5/good-signal-bars-but-no-network-connection-many-times-daily-36756, Jul. 2021.

[44] Netmotion. https://help.netmotionsoftware.com/support/docs/Diagnostics/450/help/DiagnosticsHelp.htm, Jun. 2022.

[45] Netscout. https://www.netscout.com/, Jan. 2022.

[46] Padmanabha Iyer, A., Erran Li, L., Chowdhury, M., and Stoica, I. Mitigating the latency-accuracy trade-off in mobile data analytics systems. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking* (2018), pp. 513–528.

[47] Park, H.-S., Lee, Y., Kim, T.-J., Kim, B.-C., and Lee, J.-Y. Faster recovery from radio link failure during handover. *IEEE Communications Letters 24*, 8 (2020), 1835–1839.

[48] Poorzare, R., and Augé, A. C. Challenges on the way of implementing tcp over 5g networks. *IEEE access 8* (2020), 176393–176415.

[49] Puttonen, J., Kurjenniemi, J., and Alanen, O. Radio problem detection assisted rescue handover for lte. In *21st Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications* (2010), IEEE, pp. 1752–1757.

[50] Rula, J. P., and Bustamante, F. E. Behind the curtain: Cellular dns and content replica selection. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (2014), pp. 59–72.

[51] What is a sim applet and why is it important for iot & m2m? https://1ot.mobi/resources/blog/iot-hacking-series-6-what-is-a-sim-applet-and-why-is-it-important-for-iot-m2m, Jul. 2019.

[52] Simbae sim based application engine. https://abledevice.com/simbae/, Jun. 2022.

[53] T-Mobile. Dns failures on t-mobile devices. https://community.t-mobile.com/tv-home-internet-7/dns-failures-tonight-south-carolina-on-whitebox-lte-modem-router-34319, Jan. 2021.

[54] T-Mobile. Udp blocking issue in 5g. https://community.t-mobile.com/network-coverage-5/network-blocking-udp-36746, Jun. 2021.

[55] Internet and data issues. https://www.t-mobile.com/support/devices/device-troubleshooting/internet-and-data-issues, Jan. 2022.

[56] Configure outdated apn settings. https://thecellguide.com/configure-apn-settings-galaxy-s8-3506, Jul. 2020.

[57] Twitch. https://www.twitch.tv/, Jan. 2022.

[58] U-blox. Toby-r2 series. https://www.u-blox.com/en/product/toby-r2-series, Jan. 2022.

[59] My verizon app. https://play.google.com/store/apps/details?id=com.vzw.hss.myverizon&hl=en_US&gl=US, Jan. 2022.

[60] Verizon. Supported device brands. https://www.verizon.com/support/brands/, Jan. 2022.

[61] T-mobile app. https://www.t-mobile.com/apps/download-t-mobile-app, Jan. 2022.

[62] Xu, D., Zhou, A., Zhang, X., Wang, G., Liu, X., An, C., Shi, Y., Liu, L., and Ma, H. Understanding operational 5g: A first measurement study on its coverage, performance and energy consumption. In *Proceedings of the Annual conference of the*

ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication (2020), pp. 479–494.

[63] Yang, D., Zhang, X., Huang, X., Shen, L., Huang, J., Chang, X., and Xing, G. Understanding power consumption of nb-iot in the wild: tool and large-scale measurement. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (2020), pp. 1–13.

[64] Youtube. https://www.youtube.com/, Jan. 2022.

[65] Yuan, Z., Li, Q., Li, Y., Lu, S., Peng, C., and Varghese, G. Resolving policy conflicts in multi-carrier cellular access. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking* (2018), pp. 147–162.

[66] Zhao, J., Ding, B., Guo, Y., Tan, Z., and Lu, S. Securesim: rethinking authentication and access control for sim/esim. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking* (2021), pp. 451–464.

[67] Zhou, X., Zhao, Z., Li, R., Zhou, Y., Chen, T., Niu, Z., and Zhang, H. Toward 5g: When explosive bursts meet soft cloud. *IEEE Network 28*, 6 (2014), 12–17.

[68] Zhu, Y., Kang, N., Cao, J., Greenberg, A., Lu, G., Mahajan, R., Maltz, D., Yuan, L., Zhang, M., Zhao, B. Y., et al. Packet-level telemetry in large datacenter networks. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (2015), pp. 479–491.

A STANDARDIZED FAILURE CAUSES RELATED TO CONFIGURATION ISSUES

Control plane management failures with the required configurations from the infrastructure:

- #26-Non-5G authentication unacceptable: Supported RAT
- #27-N1 mode not allowed: Supported RAT
- #31-Redirection to EPC required: Supported RAT
- #62-No network slices available: Sugested S-NSSAI
- #72-Non-3GPP access to 5GCN not allowed: supported RAT
- #91-DNN not supported or not subscribed in the slice: Suggested DNN
- #95-Semantically incorrect message: Invalid/missed config
- #96-Invalid mandatory information: Invalid/missed config
- #100-Conditional IE error: Invalid/missed config

Data plane management failures with the required configurations from the infrastructure:

- #27-Missing or unknown DNN: Suggested DNN
- #28-Unknown PDU session type: Suggested session type
- #33-Requested service option not subscribed: Suggested DNN
- #39-Reactivation requested: Suggested DNN
- #41-Semantic error in the TFT operation: Suggested TFT
- #42-Syntactical error in the TFT operation: Suggested TFT
- #43 –Invalid PDU session identity: Activated PDU session
- #44-Semantic errors in packet filter(s): Suggested packet filter
- #45-Syntactical error in packet filter(s): Suggested packet filter
- #54 –PDU session does not exist: Activated PDU session
- #59-Unsupported 5QI value: Suggested 5QI
- #68-Not supported SSC mode: Suggested packet filter
- #70-Missing or unknown DNN in a slice: Suggested DNN
- #83-Semantic error in the QoS operation: Suggested packet filter
- #84-Syntactical error in the QoS operation: Suggested packet filter
- #95-Semantically incorrect message: Invalid/missed config
- #96-Invalid mandatory information: Invalid/missed config
- #100-Conditional IE error: Invalid/missed config

B AT COMMANDS LIST FOR FAST FAILURE HANDLING

- Modem reset: AT+CFUN
- PLMN selecion: AT+COPS
- Control-plane reattachment: AT+CGATT
- Data session setting: AT+CGDCONT
- Data plane reset: AT+CGACT