Experience: A Five-Year Retrospective of MobileInsight*

Yuanjie Li*, Chunyi Peng†, Zhehui Zhang‡, Zhaowei Tan‡, Haotian Deng†, Jinghao Zhao‡, Qianru Li‡, Yunqi Guo‡, Kai Ling†, Boyan Ding‡, Hewu Li*◇, Songwu Lu‡

* Tsinghua University † Purdue University ‡ University of California, Los Angeles ◇ BNRist

ABSTRACT

This paper reports our five-year lessons of developing and using MobileInsight, an open-source community tool to enable software-defined full-stack, runtime mobile network analytics inside our phones. We present how MobileInsight evolves from a simple monitor to a community toolset with cross-layer analytics, energy-efficient real-time user-plane analytics, and extensible user-friendly analytics at the control and user planes. These features are enabled by various novel techniques, including cross-layer state machine tracking, missing data inference, and domain-specific cross-layer sampling. Their powerfulness is exemplified with a 5-year longitudinal study of operational mobile network latency using a 6.4TB dataset with 6.1 billion over-the-air messages. We further share lessons and insights of using MobileInsight by the community, as well as our visions of MobileInsight's past, present, and future.

CCS CONCEPTS

• **Networks** → **Mobile networks**; **Network monitoring**; *Network performance analysis*; *Network measurement*.

KEYWORDS

Mobile network, cellular network, mobile data science and analysis, MobileInsight

ACM Reference Format:

Yuanjie Li et.al.. 2021. Experience: A Five-Year Retrospective of MobileInsight. In *The 27th Annual International Conference On Mobile Computing And Networking (ACM MobiCom '21), October 25–29, 2021, New Orleans, LA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3447993.3448138

1 INTRODUCTION

The mobile network has been a critical global infrastructure for decades. Together with the wired Internet, it offers ubiquitous wireless network access and wide-area seamless mobility. The 4G LTE and upcoming 5G have successfully served billions of users today, and will enable trillions of Internet-of-Things in the foreseen future.

Despite its great success, the mobile network remains a complex "black box" for its users. To enable "anywhere, anytime" network services, it incorporates various wireless communication, mobility management, data transfer, and security functions on the control plane and user plane. These functions work together to enable network services, and involve complex interactions in a distributed environment. Moreover, the mobile network inherits the "*smart core, dumb terminal*" design philosophy from the telephony network: Most functions are placed inside the infrastructure, leaving limited visibility to the end devices. Such opaqueness prevents devices from understanding *what* goes on, *why* it happens, and *how* to deal with it. Furthermore, operators are reticent to share their insights from the infrastructure side. So it is hard for researchers and developers to understand and exploit the operational mobile network.

To open up the "black box" mobile network operations to devices, we started the MobileInsight project in 2015. Our goal was to build an open-source community software tool that enables *software-defined full-stack, runtime* mobile network monitoring and analytics inside our commodity phones. This tool should enable open access to operational mobile network data and offer extensible in-device analytics for runtime network behaviors. It should facilitate researchers and developers to readily and accurately understand and exploit the mobile network, *without* relying on the netwrok infrastructure or operators.

We released the first version of MobileInsight to the community in 2016 [13, 53]. Since then, MobileInsight has evolved from a simple in-device network monitor to an enabler of device-based network analytics, diagnosis, and customization. We are thrilled to see numerous real uses with MobileInsight, such as performance boosting, energy analysis, configuration diagnosis, security threat detection, to name a few (see representative studies in Table 3). We continuously refine its design based on user feedback, extend it with advanced features, use it to build large-scale operational datasets, and repeatedly validate its value in diverse scenarios.

This paper is a retrospective of our experiences in building and using MobileInsight over the past five years. We revisit the (un)successful lessons from MobileInsight as a community tool. Rather than focusing on MobileInsight's specific issues, we use MobileInsight as an example to address three general questions:

(1) How can an in-device software tool help analyze the "black-box" mobile network, and enable various new applications that existing solutions cannot?
(2) Looking back, what design choices did the original MobileInsight design make right, and what did not?
(3) What are the potentials and limitations of device-based, data-driven mobile intelligence and customization?

The rest of the paper will answer these questions from both the developers' and users' perspectives. We explain why it is hard to enable in-device mobile network analytics in §2. Then in §3, we

*Chunyi Peng and Hewu Li are the corresponding authors. Yuanjie Li and Hewu Li are with the Institute for Network Sciences and Cyberspace, Tsinghua University. Hewu Li is also with the Beijing National Research Center for Information Science and Technology (BNRist). More information about MobileInsight is available online at http://www.mobileinsight.net.

ACM MobiCom '21, October 25–29, 2021, New Orleans, LA, USA

ACM ISBN 978-1-4503-8342-4/21/10…$15.00
https://doi.org/10.1145/3447993.3448138

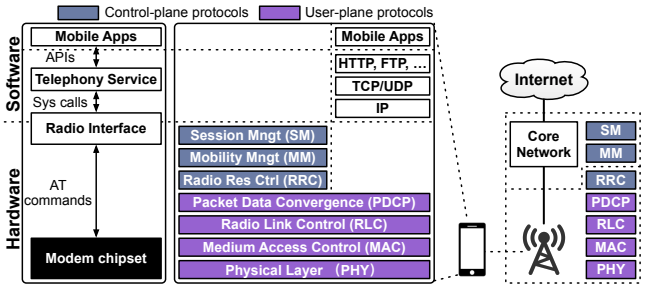


Figure 1: 4G/5G mobile network architecture, protocol stack, and traditional information flow in the device.

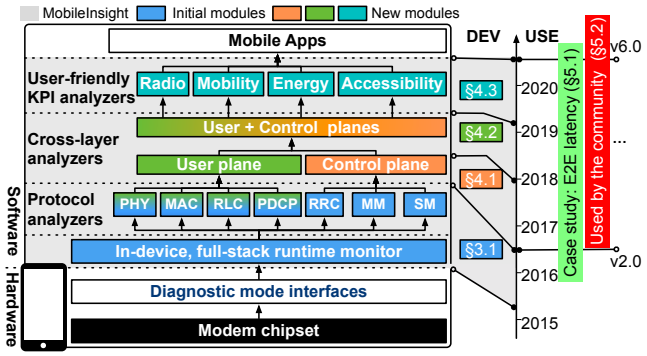


Figure 2: The evolution and milestones of MobileInsight.

overview MobileInsight's goals, initial design with limitations, and its evolutions and milestones thereafter. In §4, we elaborate on how MobileInsight evolves from a simple monitor to an advanced software tool with cross-layer analytics, energy-efficient real-time user-plane analytics, and user-friendly and extensible KPI analytics. In §5, we exemplify the powerfulness of these features with a 5-year longitudinal study for operational network latency, and review the experiences of using MobileInsight by the community. We compare MobileInsight with related work in §6, and summarize our visions of MobileInsight's past, present, and future in §7.

In summary, this work has three main contributions:

(1) We report our experiences of developing MobileInsight in the past five years. We show how MobileInsight evolves to a community tool with advanced network analytics. We review our positive and negative lessons of development;
(2) We summarize the quantitative benefits and lessons of using MobileInsight in diverse scenarios by the community, and demonstrate the potentials and limitations of device-based mobile network intelligence with MobileInsight;
(3) We exemplify MobileInsight's powerfulness with a 5-year longitudinal study of mobile network latency using a 6.4TB dataset with 6.1 billion over-the-air messages, and show how MobileInsight unveils various insights of the operational networks for the community, and by the community.

The source code of MobileInsight is public available and updated at [13, 17] and the dataset is released at [16].

2 WHY IS IN-DEVICE ANALYTICS HARD?

The mobile network is a complex "black box" for devices. This limits the devices' ability to understand and utilize the network. It is not easy to enable in-device network analytics due to two challenges.

Complex network architecture: The 4G/5G mobile network consists of a radio access network with base stations, and a core network (Figure 1). Similar to Internet, the mobile network also defines its control/user planes and a layered protocol stack on the device and network sides. However, it has more functions to accommodate than Internet and is thus more complex. At its control plane, the mobile network defines signaling protocols to facilitate radio resource control (RRC), wide-area mobility management (MM), and traffic session management (SM) for QoS/billing. At its user plane, the mobile network defines wireless channel structures at PHY layer, traffic scheduling/multiplexing/error control at MAC (Medium Access Control) layer, data segmentation and reliable in-order delivery at RLC (Radio Link Control) layer, and data ciphering and integrity protection at PDCP (Packet Data Convergence Protocol) layer. To enable ubiquitous network services, these protocols work together with complex interplays in a distributed environment.

Limited in-device access to network information: Figure 1 shows the device-side mobile network protocols are realized inside the chipset. They are mostly invisible to the OS kernel and user-space apps. Only basic network information, such as data/voice status and radio signal strength, is accessible to the OS via radio interface layer (RIL). For user-space apps, the OS exposes a subset of RIL to APIs, such as `TelephonyManager` in Android [3, 12].

3 OVERVIEW OF MOBILEINSIGHT

The goal of MobileInsight is to build a community software tool for full-stack, runtime mobile network monitoring and analytics in commodity mobile devices. Specifically, it should enable

(1) **In-device, full-stack runtime monitor** of over-the-air messages between device and network;
(2) **Real-time deep analytics** of mobile network status, configurations, policies, and data transfers in devices;
(3) **Open and extensible platform** for community to customize analytics and accommodate emergent features (e.g., 5G/IoT).

This section reviews MobileInsight's initial design and limitations, and summarizes its evolution milestones from 2016 to 2020.

3.1 Initial Design

Figure 2 illustrates the evolution of MobileInsight. At its initial phase, our objective was to quickly build basic blocks while retaining its long-term extensibility [53]. We intentionally kept MobileInsight simple and open with two modules:

• **In-device, full-stack runtime monitor:** For the first time, MobileInsight exposed runtime over-the-air messages from the chipset to user-space. Table 2 summarizes these messages. They carry rich information of mobile network protocols' states, configurations, and operation logic that legacy APIs did not expose. To collect them, MobileInsight exploits the *diagnostic mode*, a second channel between the hardware chipset and software. The diagnostic mode uses a virtual interface (e.g., `/dev/diag` for Qualcomm chipsets, summarized in Table 1) to expose in-chipset messages to the USB port. It is available in major OSes and chipsets from Qualcomm, MediaTek, Samsung, and Huawei. To this end, MobileInsight emulates an external logger inside the device, pulls the binary raw logs from virtual interfaces, parses their metadata headers and message contents, and feeds them to protocol analyzers.

Table 1: Virtual interfaces for the diagnostic mode.

Mobile OS	OS Version	Chipset	Virtual interface	Open driver
Android	4.4–11.0	Qualcomm	/dev/diag	[2]
Android	4.4–11.0	MediaTek	/dev/ccci_md_log_ctrl	[1]
Android	4.4–11.0	Intel XMM	/dev/mdmTrace	[11]
iOS	9–14	Apple A6	/private/var/logs/Baseband	[4]

• **Protocol analyzers:** The initial MobileInsight focused on the analysis of individual control-plane signaling protocol. Given raw messages, MobileInsight infers each protocol's states, triggering conditions for state transitions, and taken actions. Moreover, it infers certain protocol operation logic that uses operator-defined policies and configurations. MobileInsight abstracts each protocol analytics as an `analyzer` class with open APIs. Users can customize their own analytics on top of MobileInsight's built-in analyzers.

3.2 Why is Initial MobileInsight Not Enough?

We publicly released the initial MobileInsight in 2016. From user feedback and our own experiences, we realized the initial MobileInsight has not yet achieved all its aforementioned goals:

(1) In-device, full-stack runtime monitor: This feature has been successful since MobileInsight's first release, with positive user feedback and high satisfaction. Our first release only supported Qualcomm chipsets. Later we supported MediaTek chipsets, and more messages from new standards (from 3GPP Release 7 to Release 15). To support new chipsets and OSes, we first verify whether they have provided similar diagnostic ports. If yes, we can follow the similar way in the initial design (§3.1) to enable in-device monitor. As summarized in Table 1, we have validated that similar diagnostic ports are indeed available for Qualcomm, MediaTek, Huawei, and Intel chipsets. We have also validated that similar in-device monitor can be realized in iOS (although jailbreak is needed). In the past five years, we have explored new virtual interfaces to enable MobileInsight at more device models (with new chipsets and OS versions) and continuously upgraded MobileInsight accordingly to support new 3GPP standards (up to Release 15). Our experience is that MobileInsight is extensible to future 3GPP standards and device models, given its successful extension to new standards, chipsets, and mobile OS versions in the last years. We only need to acquire latest commodity phones for testing and exploit diagnostic ports' driver codes as the ground truth. Moreover, the monitor-analyzer design is flexible for long-term extension across devices: the monitor can be extended independently due to device-specific characteristics, while most analyzers can be reused across the devices.

(2) Real-time deep mobile network analytics: The initial MobileInsight mainly analyzed individual signaling protocols in the control plane. This is not enough for two reasons. First, many usage scenarios involve multiple protocols across layers in a distributed environment. It has been widely reported [54, 71, 73] that, even if each individual protocol behaves well, the interactions between protocols can still be problematic in reality. Understanding these interplays calls for *cross-layer, vertical analytics* (§4.1). Second, the user-plane analytics was largely missing, especially for the link/physical-layer analytics below the TCP/IP stack. Without it, many issues in runtime data transfer remained mysterious for devices. However, different from the control-plane analytics, the user plane one faces an explosive growth of over-the-air messages. This poses challenges on energy-efficient, real-time analytics (§4.2).

Table 2: Available over-the-air messages in MobileInsight.

		Message types
Control plane	SM	Default or dedicated session setup, modification and release; PDN connectivity setup, modification and release.
	MM	Attach/detach; Authentication request, response, and failure; Security mode control; Service request; Paging; Identification request and response.
	RRC	Radio connection setup, release, re-establishment and reconfiguration; System info blocks; Handover command; Measurement control and report; Radio capability query; Paging; Security model command.
User plane	PDCP	Uplink and downlink control/data packets; Ciphering, integrity check, and compression configuration.
	RLC	Uplink and downlink control/data segments, sequence number, and acknowledgment; Scheduling, retransmission and timer configuration; Traffic delivery statistics.
	MAC	Uplink and downlink transport blocks, and positive/negative acknowledgment; Uplink scheduling request; Uplink buffer status report; Uplink random access trigger and attempt; Retransmission configuration.
	PHY	Radio band indicator; DL/UL radio resource allocation; Channel estimation (signal strengths, CQI, PMI, RI, path loss); Modulation and coding scheme; Block error rate; Physical data rate; Cell search, measurement and selection; Uplink transmission power control; Downlink reception power control (DRX); Random access status.

(3) Open and extensible platform: The initial MobileInsight indeed offered open APIs for fine-grained protocol analytics, customizable data collection, and extensible plugins. However, according to numerous technical inquiries and feedback from the community, they were not friendly to new users due to their nature of low-level semantics. Using these APIs requires deep understanding of mobile network and its complex operations, which is difficult for most users. This limits MobileInsight's wide adoption (§4.3).

3.3 Five-year Milestones

To address these limitations, we have progressively enhanced MobileInsight since its first public release in 2016. These refinements follow MobileInsight's simple framework in §3.1. Now the latest MobileInsight (v6.0, released in December 2020) has 60,409 lines of code (33,001 lines of C++ and 27,408 lines of Python). Figure 2 shows MobileInsight's evolution roadmap and milestones in the past five years, from both the developers' and primary users' perspectives.

• **Developers' milestones:** MobileInsight evolves as follows:

◦ *2015–2016: Basic in-device monitor (§3.1).* We built the alpha MobileInsight for the in-device full-stack monitoring.

◦ *2016–2017: Control-plane protocol analytics (§3.1).* With runtime over-the-air messages, we enabled basic analytics of each individual signaling protocol, including the protocol state tracking and operation logic inference.

◦ *2017–2018: Cross-layer vertical analytics (§4.1).* We added cross-layer analytics on top of individual protocol's analytics. We started from control-plane analytics, and later extended it to user plane and interplay across control/user planes.

◦ *2018–2019: Energy-efficient runtime user-plane analytics (§4.2).* We devised cross-layer sampling with missing data inference (§4.1) for energy-efficient, real-time user-plane analytics.

◦ *2019–2020: User-friendly, extensible analytics (§4.3).* To broaden the adoption, we followed the user feedback to extend MobileInsight with user-friendly, extensible KPI analyzers to streamline its usage.

• **Users' milestones:** Since 2016, MobileInsight has attracted global uses from 350+ academia and industry institutes (Figure 3).

Table 3: Representative use cases of MobileInsight in diverse scenarios. C1-C4 are MobileInsight's components: C1: control-plane analytics, C2: user-plane analytics, C3: cross-layer analytics and C4: KPI analytics.

Cat.	MobileInsight's use cases	Protocols	C1 (§3)	C2 (§4.2)	C3 (§4.1)	C4 (§4.3)	Lessons of using MobileInsight
Performance	Link capacity prediction for TCP [32, 33], Web [76], WebRTC [48], & 360°video [75]	PHY, TCP, APP	×	✓	×	✓	MobileInsight helps the device infer base stations' "black-box" uplink/downlink radio scheduling policies for performance optimization.
	Handover policy inference [71, 79] for performance analysis and boost	RRC	×	✓	✓	✓	MobileInsight facilitates the device to infer the "black-box" infrastructure-side handover policies for performance analysis and boost.
	Control-plane acceleration for low-latency access [54]	RRC, MM, SM	✓	×	✓	✓	MobileInsight unveils that device-side state replica helps accelerate network-side signaling.
	Latency reduction for VR [71], edge [83], remote sensing [72] and driving [57]	PHY, MAC, RLC, RRC, APP	✓	✓	✓	✓	MobileInsight facilitates the application-driven customization of uplink data transfer that the infrastructure cannot.
	Multi-carrier network access (e.g., Google Fi [50, 52])	RRC, MM, SM	✓	×	✓	✓	MobileInsight empowers the device to proactively select the carrier networks, which cannot be achieved by each individual operator.
	A 5-year longitudinal study for the evolution of mobile network latency (§5.1)	PHY, MAC, RLC, PDCP, RRC, NAS	✓	✓	✓	✓	By crowdsourcing MobileInsight traces from devices, the community can analyze the long-term evolution of the operational network performance in diverse network and device contexts.
Reliability	Misconfiguration analysis for mobility management [36]	RRC	×	✓	×	×	MobileInsight lets the device gain visibility of fine-grained mobile network configurations.
	Policy conflicts in among mobile network carriers [49, 81]	RRC, MM	✓	×	✓	×	A device with MobileInsight can unveil policy conflicts that each individual operator cannot.
	Reliable handover for high-speed trains [56, 74]	PHY, RRC, MM	✓	✓	✓	✓	MobileInsight demystifies various handover failures, and enables device-based masking of failure-induced disruptions at higher layers.
Energy	Modem energy drain analysis [35]	RRC	✓	×	×	✓	MobileInsight helps the device quantify the fine-grained energy costs due to network operations.
Security	Control-plane security threat analysis via user-side semi-automated testing [45]	RRC, MM, SM	✓	×	×	×	MobileInsight facilitates the security analysis for operational mobile networks by exposing the real cross-layer signaling messages.
	Voice call spoofing defense [37]	SM, APP	✓	×	×	×	MobileInsight makes it possible for the victim callee to proactively detect the call spoofing.
	Location privacy leakage [42, 47]	MAC, RRC	✓	✓	✓	×	MobileInsight not only benefits benign users, but also empowers attackers of curiosity.

(a) By geographical areas

Academia	54.9%
Industry	21.7%
Unknown	23.4%
Total	100%

(b) By institutes

Figure 3: Global usage of MobileInsight.

We are thrilled to see numerous new device-based designs of mobile network intelligence with MobileInsight, including but not limited to data speed boosting [32, 54, 74], energy saving [35], failure and misconfiguration diagnosis [36, 81], security threat detection [37, 41], and emergent scenarios such as VR/AR [71], 360° video [75], high-speed trains [56, 74], and many more. Table 3 shows representative use cases of MobileInsight by the community. We will elaborate on the lessons of using MobileInsight in §5.

4 EVOLUTION OF MOBILEINSIGHT

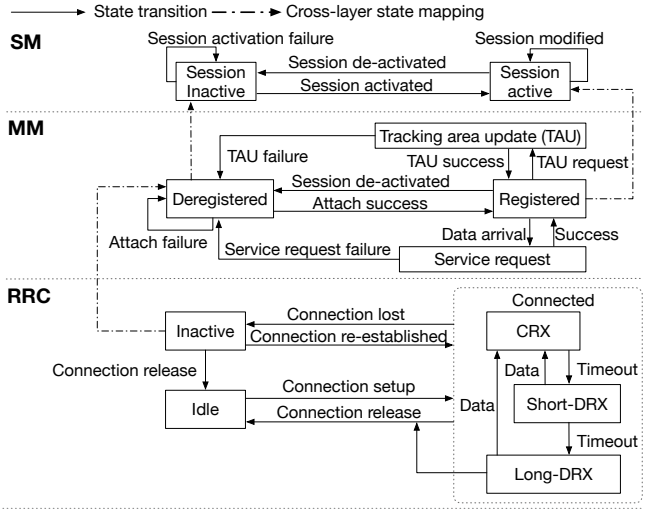
We next elaborate on the lessons from MobileInsight's evolution.

4.1 Cross-layer Vertical Analytics

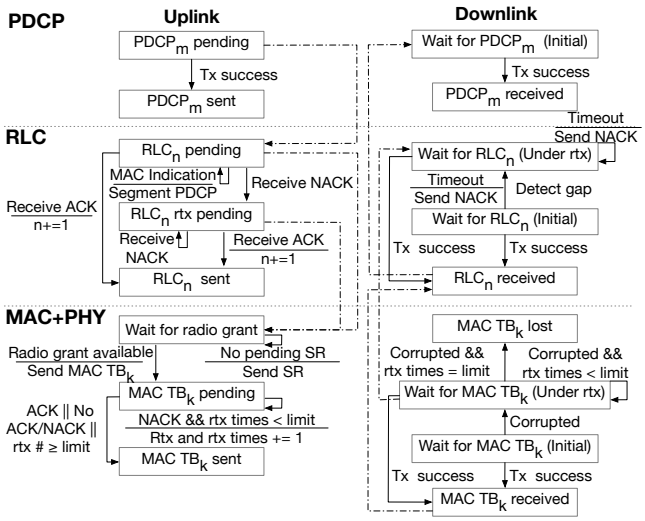
The initial MobileInsight only tracked the states and operation logic of each *individual* protocol (§3.1). Starting in 2017, we enhanced MobileInsight with *cross-layer, vertical analytics.*

Taxonomy of cross-layer interactions: In the mobile network, cross-layer interactions are a norm rather than an exception. To enable ubiquitous services, multiple protocols across layers must be involved in a distributed environment. Unfortunately, various studies [44, 45, 54, 71, 73] and uses of MobileInsight in Table 3 indicate that, even if each individual protocol behaves well, the interactions between protocols can still be unreliable, slow or insecure in reality. In general, there are three classes of cross-layer interactions:

◦ *Control-plane interactions:* To facilitate data transfer, the signaling protocols (in RRC, MM, and SM) collaboratively establish and maintain the session between a device and network. Figure 4a exemplifies how signaling protocols work together. To access the data service, the control plane should first create a data session between a device and network. This involves the RRC protocol



(a) Control-plane signaling protocol states



(b) User-plane per-packet delivery states

Figure 4: Simplified cross-layer state machines.

to establish the radio connectivity, the MM protocol to mutually authenticate the device and network, and the SM protocol to create a data session (§2). As the device moves, the control plane should migrate the session to new network nodes for seamless data access. To this end, the RRC protocol migrates the radio connectivity to the new base station via handover, and the SM/MM protocol migrate the session across location domains via tracking area update. Before these procedures are successfully completed, the data transfer will be blocked, thus incurring extra latencies.

∘ *User-plane interactions:* In the mobile network, each packet traverses across link/physical layers. These protocols collaboratively deliver the user traffic through complex interactions. As shown in Figure 4b, The PDCP layer labels each IP packet with a sequence number, and pushes it to the RLC layer. The RLC layer divides each packet into multiple segments based on the available runtime physical-layer radio resource. For in-order reliable delivery, RLC maintains per-segment sequence number and acknowledgment. The MAC maps RLC segments to transport blocks, buffers them to wait for physical-layer transfer, multiplexes them for delivery, and corrects block errors via HARQ procedure.

∘ *Interplays across control and user planes:* The control and user planes mutually impact each other. On one hand, before the control-plane signaling procedures finish, the user-plane data is blocked. The signaling protocols also configure the user-plane protocols with channel bandwidth, scheduling parameters, retransmission timers, to name a few. We will exemplify such configurations in §5.1. On the other hand, the user plane also delivers the control-plane signaling messages. The message loss/corruption will affect the signaling protocols' functionality, reliability and performance.

Challenges of cross-layer analytics: MobileInsight faces two challenges in analyzing cross-layer network behaviors. First, it should tackle the complex interactions at the control plane, user plane, and across planes. Second, at the user plane, MobileInsight should tolerate the *missing data* from the packet loss or corruption. Missing data state from one layer can propagate to other layers and complicate the analytics. For example, if a corrupted MAC block cannot be recovered, its error will spread to RLC for retransmission with new missing data states. Inferring such missing states is more difficult for downlink, since the device-side MobileInsight has limited ground truth of downlink packets' status.

MobileInsight's cross-layer analytics: To address both issues, MobileInsight derives the control/user-plane state machines, tracks runtime states across layers, and infers the missing data state by leveraging the temporal inter-packet dependency.

∘ *Cross-layer state machine tracking:* At the control plane, MobileInsight tracks signaling protocols' runtime states and their interactions. It first extracts each protocol's state machines from 3GPP standards [25, 26, 29, 30]. As shown in Figure 4a, the RRC states decide the radio connectivity and power-saving mode, the MM states define device's registration status to core network, and the SM states decide the session (de)activation. These state machines are stacked and interconnected by standardized cross-layer state mapping (illustrated as dotted lines in Figure 4a): The lower-protocol's state change (e.g., "Inactive" state in RRC due to radio link failure) will propagate to upper layers (de-registration in MM and session deactivation in RRC in this example). MobileInsight tracks each protocol by feeding runtime messages to the state machines. Upon a protocol state change, MobileInsight updates other protocols' states accordingly based on cross-layer state mapping.

At the user plane, MobileInsight tracks *per-packet* delivery status across below-IP layers. Figure 4b shows the simplified per-packet state machines from standards [18–20, 22–24] and their cross-layer state mapping. The PDCP states define the delivery status of a ciphered IP packet. The RLC states track the in-order reliable delivery status of segments. The MAC states decide the buffering, transfer, and error status of transport blocks. These protocols' states are correlated on a per-packet basis: A packet at PDCP layer is divided into multiple RLC segments, each of which are further mapped to MAC transport blocks based on available physical-layer radio resource. The delivery state (sent, received, pending, lost/corruption, and retransmission) at one layer will decide the delivery states of other layers (dotted lines in Figure 4b). MobileInsight keeps a replica of these state machines per packet, maps it to RLC segments and MAC

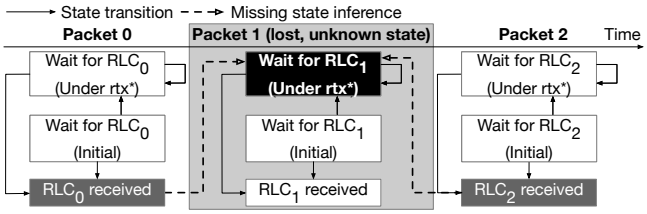
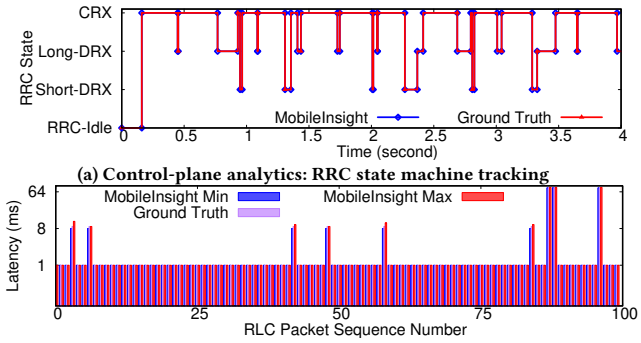


Figure 5: Missing state inference via data dependency.



(a) Control-plane analytics: RRC state machine tracking

(b) User-plane analytics: Downlink data latency tracking

Figure 6: Accuracy of MobileInsight compared to the infrastructure-side ground truth (OAI [6] over USRP B210).

transport blocks, tracks the delivery of them based on the runtime messages at each layer, and updates the fine-grained state across layers based on the dependency in Figure 4b.

Last, MobileInsight tracks the interplay across control and user planes. By checking the signaling protocol states, it decides if the data transfer is allowed and updates each packet's state accordingly. Moreover, it tracks the delivery of signaling messages at user plane, and updates the control-plane states if the message is lost/corrupted.

∘ *Missing data state inference via inter-packet dependency:* If a user-plane packet is lost, its status is unknown to MobileInsight. To infer its state, MobileInsight utilizes the temporal dependency between packets. Figure 5 exemplifies this with two received packets and one missing packet. The link-layer in-order reliable delivery couples packets' state machines. A missing packet's possible state can be *bounded* by its nearby packets. By checking the received packets' states before/after the missing data, MobileInsight narrows down the missing data's possible states to provide bounded information such as retransmission latency.

Comparison with the ground truth: To quantify the accuracy of MobileInsight's cross-layer analytics, we compare it with the ground truth. We first compare MobileInsight with two device-side analytics tools: QXDM [67] and Network Signal Guru [5]. We confirm MobileInsight unveils identical standardized control/user-plane operations to these tools, because all these tools have access to the same mobile network information from the hardware modem.

We next compare MobileInsight's accuracy with the infrastructure-side ground truth. We build a controlled LTE infrastructure using a commodity server (as core network), USRP B210 (as a radio base station) and OpenAirInterface [6] software cellular protocol stack. We use a commodity phone with MobileInsight to connect to this infrastructure, and compare the runtime MobileInsight analytical results with those on the infrastructure side. Figure 6 exemplifies the comparison result at the control and user planes. At the control plane, Figure 6a verifies MobileInsight unveils identical signaling protocol states to those on the infrastructure side. At the user plane, Figure 6b compares the upper/lower-bound of the downlink packet latency inferred by MobileInsight with the infrastructure-side ground truth. When there is no retransmission due to data loss/corruption, MobileInsight unveils identical per-packet downlink data latency to the ground truth. In presence of the retransmission, MobileInsight's missing data inference via inter-packet dependency ensures marginal errors compared to the ground truth, with an error of 0.19ms on average and 4ms at maximum (i.e., ≤1.4% estimation errors).

4.2 Efficient Real-time User-plane Analytics

In 2018, we extended MobileInsight for the below-IP user-plane analytics. Different from control plane, user plane faces an explosive growth of over-the-air messages. This poses challenges to the energy-efficient, real-time analytics in commodity devices.

Characteristics of user-plane analytics: Compared to control-plane messages, user-plane messages are *simpler* with fewer fields, but *more intensive* with massive amount of packets to deliver. Figure 7 shows the user plane's messages are 2∼3 orders of magnitude more frequent than the control plane's.

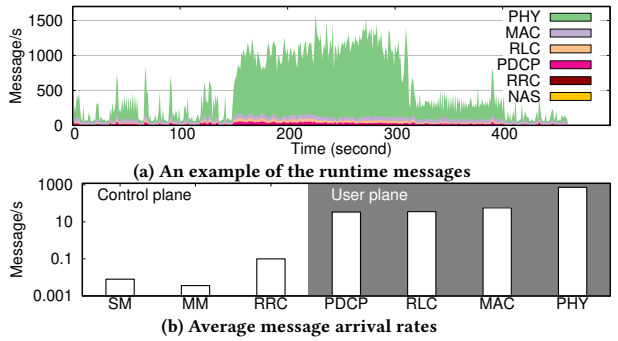
We next quantify how well the initial MobileInsight tackles user-plane analytics inside devices. We enable all messages in Table 2 to evaluate the initial MobileInsight's runtime responsiveness, energy, and CPU usage. We repeat this test with data collection only, collection + message parsing, and collection + parsing + analysis. For energy usage, we also compare MobileInsight with the worst-case background scenario when the screen is always on. Figure 8 shows the results. We make three observations:

• **Real-time responsiveness:** Surprisingly, even with intensive user-plane messages, MobileInsight can still timely process them *before* the next message arrives. For each message, we define its accumulative lag as the elapsed time that its processing is after the next message's arrival. Figure 8a shows MobileInsight can analyze ≥95% user-plane messages with ≤1ms lag. The maximal lag is ≤8ms. This is because most user-plane messages are simple to process.

• **Energy deficiency:** The initial MobileInsight's responsive real-time user-plane analytics is at the cost of huge energy and CPU usage. Figure 8d shows that, with all messages enabled, MobileInsight consumes 21% battery in 1 hour, which is 1.5× compared to the scenario with the always-on screen. The battery is mostly used by software, since the data collection from chipset consumes comparable energy to the scenario without MobileInsight. Its energy consumption is proportional to message volumes.

• **Heavy CPU usage:** With all messages, Figure 8c shows the initial MobileInsight occupies one core and uses 12–23% CPU in total. The CPU usage is proportional to message volumes, and dominated by message parsing (≥99%) in software space.

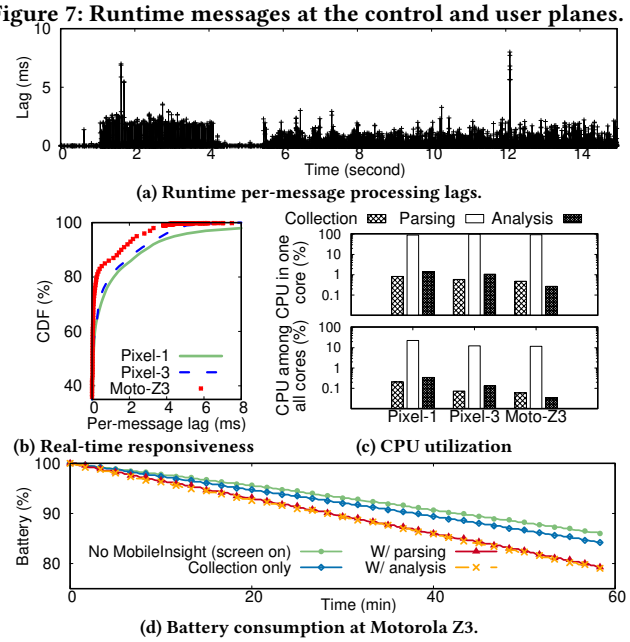
Vanilla solution: Domain-specific independent sampling To save the battery and CPU, MobileInsight should reduce the cost of processing intensive link/physical-layer messages *and* retain high analytics accuracy. For non-real-time tasks, MobileInsight can collect raw messages in device and analyze them offline. For real-time analytics, MobileInsight can sample the messages to analyze. The initial version of MobileInsight's user-plane analytics uniformly sampled each physical/link-layer's messages independently. Figure 8c implies sampling can be approximated by parsing only a



(a) An example of the runtime messages

(b) Average message arrival rates

Figure 7: Runtime messages at the control and user planes.



(a) Runtime per-message processing lags.

(b) Real-time responsiveness

(c) CPU utilization

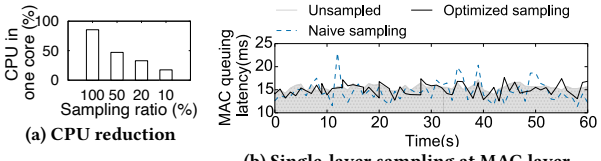
(d) Battery consumption at Motorola Z3.

Figure 8: The initial MobileInsight's user-plane analytics.

subset of messages in software. This approach still retains all the raw data in the collection phase to facilitate full-fledged offline analysis. Figure 9a confirms it effectively reduces the CPU usage.

Moreover, the sampling can be optimized with domain-specific knowledge. We find 4 types of messages (MAC buffer status, block error rate, serving cell measurements, and uplink transmission power) contribute ≥71.8% of total messages. We customize their analytics to be sampling-aware, treat un-sampled messages as missing data, and optimize their analytics accuracy with the missing data inference in §4.1. Figure 9b exemplifies the customization for the uplink MAC-layer queuing delay for each segment. To track a segment's queuing delay, MobileInsight needs two timestamps for it when entering and leaving the buffer. The naive sampling is unaware of this timestamp dependency and simply samples the MAC logs uniformly. This is prone to miss one of the timestamps and thus failure of tracking the latency. Instead, by tracking logs continuously to cover both timestamps for each segment, the optimized sampling is more accurate under the same sampling ratio.

Our solution: Domain-specific cross-layer sampling. However, independent sampling turns inaccurate in *cross-layer* analytics. As shown in §4.1, an IP packet will traverse across link/physical layers for delivery, during which it can be divided into multiple

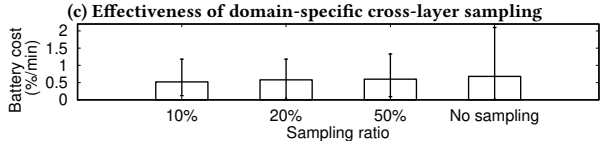


(a) CPU reduction

(b) Single-layer sampling at MAC layer

	Actual sampling ratio			Per-packet latency analysis error		
Target sampling ratio	10%	20%	50%	10%	20%	50%
Independent sampling	0.04 %	0.52%	11.38%	6.76ms (34.2%)	6.64ms (33.6%)	1.69ms (8.6%)
Cross-layer sampling	9.37%	18.1%	49.45%	1.63ms (8.3%)	1.48ms (7.5%)	0.50ms (2.5%)

(c) Effectiveness of domain-specific cross-layer sampling



(d) Energy efficiency of cross-layer sampling (Motorola Z3)

Figure 9: MobileInsight's energy-efficient real-time user-plane analytics (uplink data latency analysis as an example).

RLC segments and MAC blocks based on available physical radio resource. If messages from different layers are sampled independently, the cross-layer dependency between the IP/PDCP packets, RLC segments, MAC blocks, and physical resource can be lost. This causes not only inaccurate analytics, but also inefficient sampling since sampled messages across layers are mismatched and wasted. Figure 9c exemplifies this deficiency when tracking the uplink packet latency. With 10%, 20% and 50% sampling ratio, independent sampling only ensures 0.04%, 0.52% and 11.38% IP packets can be fully tracked across layers. This leads to 34.2%, 33.6% and 8.6% estimation errors for uplink packet latency, respectively.

To this end, we devise *cross-layer sampling*. Rather than independent sampling among layers, MobileInsight first uniformly samples the IP packets at PDCP based on the target sampling ratio. Then for each sampled IP packet, MobileInsight runs the cross-layer dependency tracking in §4.1 to locate the corresponding messages related to its RLC segments, MAC blocks, and PHY radio resource allocation. MobileInsight only parses (samples) these messages and drop the remaining for efficiency. In this way, the cross-layer dependencies for these sample IP packets are all retained for high analytics accuracy. This approach is applicable to both uplink and downlink data transmission, because the dependency across the PHY, MAC, RLC, and PDCP layers exists for both uplink and downlink.

Figure 9 evaluates MobileInsight's cross-layer sampling. In terms of its accuracy, Figure 9c shows MobileInsight reduces the data latency estimation error by 4.1×, 4.5×, and 3.4× with 10%, 20%, and 50% sampling ratio, respectively. It retains comparable actual sampling ratios (i.e., the percentage of IP packets that can be correctly tracked across layers) to the target. Figure 9d confirms viable energy saving (by up to 47.6%) with MobileInsight's cross-layer sampling. Due to the variance of traffic and battery drain, the energy saving is not strictly proportional to the actual sampling ratio. This suggests more energy savings are possible with further refined solutions.

4.3 User-Friendly, Extensible Analytics

The new features in §4.1–4.2 empower MobileInsight with deep mobile network analytics. But they were not widely used as expected, since they are unfriendly to new users because of their low-level

Table 4: Available runtime KPIs from MobileInsight.

Category	Key performance indicator (KPI)	Protocol
Radio	Signal strength (RSRP, RSRQ, RSSI, EvDo, RSCP)	PHY
	Channel estimation (CQI, PMI, RI)	PHY
	Modulation and coding scheme	PHY
	Radio resource block allocation	PHY
	Block error rate (BLER)	PHY
	Downlink path loss	PHY
Accessibility	Radio connection setup success rate	RRC
	Attach success rate	MM
	Session setup success rate	SM
	Service request success rate	RRC+MM
	Session QoS/billing class	MM+SM
Mobility	Tracking area update success rate	MM
	Tracking area update latency	RRC+MM
	Handover success rate	RRC
	Handover disruption latency	RRC
	Handover HOL blocking latency	RLC+RRC
Retainability	Abnormal RRC connection drop rate	RRC
Integrity	Data throughput	PDCP
	Packet loss ratio	PDCP
	Link-layer data loss ratio	RLC+MAC
Energy	Uplink transmission power	PHY
	Downlink reception power (DRX)	PHY+RRC

```
# Initialize an online monitor
src = OnlineMonitor()
# Initialize the KPI manager
kpi = KPIManager()
# Enable KPI analyzer on handover latency
kpi.enable_kpi("Mobility.HANDOVER_LATENCY")
# Bind the analyzers to the monitor
kpi.set_source(src)
# Start analysis
src.run()
```

KPI analyzers

```
# check whether handover is triggered
# by tracking the RRC state machine
if ho_istriggered(event):
    HO_triggered = True
    ts = event.timestamp
# check whether handover is completed
# by tracking the RRC state machine
if HO_triggered and ho_iscomplete(event):
    latency = event.timestamp - ts
    HO_triggered = False
```

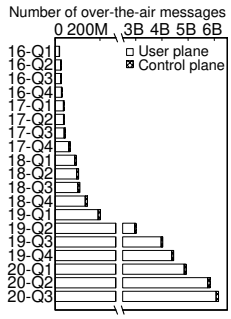
Simplified underlying implementation

Figure 10: APIs of KPI analyzers in MobileInsight (handover disruption latency as an example).

natures (§3.2). So in 2019, we started to streamline MobileInsight's analytics to balance its comprehensiveness and user-friendliness.

We extend MobileInsight with user-friendly, extensible performance indicator (KPI) analyzers. Table 4 summarizes the latest MobileInsight's available KPIs from 3GPP standards [27, 28] and user requests. They have covered the control and user planes, and various aspects such as reliability, performance, and energy efficiency. They are not available from the legacy mobile OS APIs. Compared to the protocol/packet state machines in basic MobileInsight analyzers, KPIs are more intuitive for users to understand runtime mobile network operations. As summarized in Table 3, the KPI analyzers can simplify many usage scenarios by the community.

Figure 10 exemplifies MobileInsight's APIs for the runtime KPIs (control-plane handover latency as an example). MobileInsight defines user-friendly hierarchical names (category.kpi_name) for the KPIs in Table 4. For each KPI, MobileInsight defines a KPI analyzer, and calls the corresponding state machine-based analyzers in §3.1 and §4.1–4.2 to track it. Instead of calling the complex low-level analyzers, a new user can easily track KPIs by simply declaring them via names, without worrying about the low-level details of mobile network protocols. Meanwhile, an experienced user can dive into the underlying behaviors related to KPIs. MobileInsight supports both *periodic* and *event-driven* runtime KPI reporting. By locally storing the historical data, it allows users to query the aggregated KPI statistics by time, location, network node (e.g., cells) and client.



(a) Dataset size over time

Duration	2016–2020
Dataset size	6.4 TB
Total message	6,163,692,448
Layers	SM: 0.01%
	MM: 0.04%
	RRC: 1.50%
	PDCP: 3.08%
	RLC: 3.45%
	MAC: 13.04%
	PHY: 78.88%

(b) Dataset statistics

Figure 11: Summary of the dataset from MobileInsight.

5 LESSONS OF USING MOBILEINSIGHT

We are excited to see a broad use of MobileInsight in diverse scenarios since 2016. This section reports the lessons of using MobileInsight by ourselves and the community. With a five-year longitudinal study in §5.1, we exemplify how to apply MobileInsight in §3–4 to unveil unforeseen insights from the operational network. Then in §5.2, we report the adoption of MobileInsight in broader scenarios, quantify its benefits, and summarize the usage experiences.

5.1 Longitudinal Case Study: Network Latency

This section uses network latency to exemplify how MobileInsight empowers in-device mobile network analytics. We run a 5-year longitudinal study to unveil and understand end-to-end (E2E) latency over operational mobile networks. Due to space limit, the results here are not intended to be complete; more results have been available from MobileInsight with the collected dataset being released. In this work, we showcase how to use MobileInsight's new modules in §4, and how these modules unveil some new insights that were not visible.

Dataset: Since 2016, we have sporadically collected 4G/4.5G LTE over-the-air messages with the evolving MobileInsight and accumulated a five-year dataset. We ran MobileInsight over the test phones when using ping (primary), iperf, web, video streaming, or virtual reality applications in static, walking and driving scenarios. Figure 11a shows the accumulated dataset size every quarter and Figure 11b summarizes our dataset as of August 15, 2020. It has been collected from 50+ phone models and 58 global operators over 20+ countries and regions including the USA, China, India, South Korea, Singapore, France, Spain, Germany, Norway, Hungary, Egypt, Australia, New Zealand etc. Most data (82%) is collected in the US (AT&T, Verizon, T-Mobile, Sprint, Google Fi), covering 39 states and 260,000+ miles. The early tests in 2016–2018 only enabled the control-plane messages and partial user-plane messages. With MobileInsight's user-plane features in §4.2, we further enabled full link/physical-layers messages for cross-layer analytics in late 2018 and afterwards. We ran a large-scale across-the-US driving test in May - July 2019, resulting in a surge of the dataset size in 2019 Q2.

Analytics methodology: We apply MobileInsight's new features in §4 to analyze the mobile network latency. We group our dataset by quarter/month and device context (phone models, mobility patterns, and locations) and replay them to extract the mobile network latencies using KPI analyzers in §4.3. As summarized in Table 4, these KPI analyzers provide simple interfaces to unveil the user-perceived network latencies from the control plane, user plane, and

(a) AT&T (1%) (b) AT&T (50%) (c) AT&T (95%)

Figure 12: Ping RTT latency over the US (AT&T).

across planes. They depend on the cross-layer analytics (§4.1) and sampling (§4.2) for efficient and accurate KPI extractions. To analyze the root causes of the latency variations (detailed in §5.1.3), we further perform the cross-layer analytics in §4.1 over the dataset to track the signaling procedures at the control plane, IP packet delivery across link/physical layers at the user plane, and the interplays between the control and user plane.

5.1.1 *What does LTE network latency look like today?*

Figure 12 plots ping latency (round-trip time) measured in AT&T in the past year (Q2 2019 - Q2 2020). Results for other carriers in the U.S. (Verizon, T-Mobile, Sprint and Google Fi) are similar and omitted. Unsurprisingly, E2E latency varies over the location (much worse in freeways, mountain and rural areas than in big cities). In particular, the 1th (a robust estimate of minimal latency), 50th (median) and 95th percentiles of RTT are below 40ms, 70ms and 120ms in almost all the cites; But the 1th percentile is > 80ms in mountain areas and > 40 ms over the freeways across the US. CDFs are omitted due to space limit. These results are consistent with other measurement studies by FCC [38, 39], Ookla speedtest [15, 61] and OpenSignal [62–64], but offer finer-grained spatial analysis.

5.1.2 *How does LTE network latency evolves over time?*

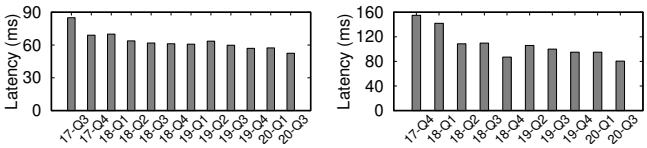
MobileInsight's KPI analyzers report that, in the past 5 years, LTE network latency decreases at both control *and* user planes overall. At control plane, the signaling protocols take time to establish the radio connectivity and migrate it in device mobility, before which all data delivery will be blocked (§4.1). For fair comparison, we examine the latency evolution in one eastern U.S. city using AT&T. Similar trends hold for other cities and operators in the same dataset. Figure 13 shows a 38.2% average latency reduction in radio connectivity setup, and 48.1% reduction in handover. At user plane, Figure 14 shows a 50.0% average reduction of the first uplink packet's delivery latency after radio connectivity setup, and increment of downlink bandwidth (thus faster transmission).

5.1.3 *Why is LTE latency shortened?*

To understand the root causes of latency reduction, we perform the cross-layer analytics in §4.1 over the dataset. MobileInsight reports diverse causes of the latency reduction on the infrastructure and device across control and data planes.

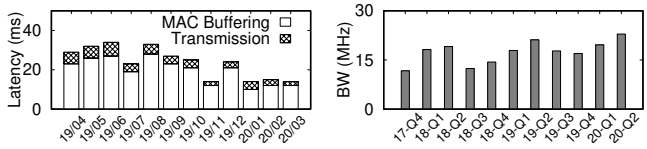
Infrastructure-side latency reduction: MobileInsight's cross-layer analytics reports continuous refinements of the operational infrastructure for lower latency, including (but not limited to)

∘ *Faster signaling processing:* For the control-plane signaling latency in Figure 13, MobileInsight reports no significant reduction of the radio transmission delay. Instead, the processing delay of the signaling procedure at the base station and the device are both reduced. Figure 15a exemplifies this acceleration based on MobileInsight's control-plane analytics. By tracking the time of delivery/receipt of the signaling messages, MobileInsight can infer



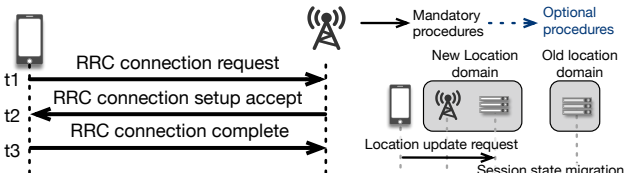
(a) Radio connectivity setup (b) Handover across cells

Figure 13: Control-plane signaling latency.



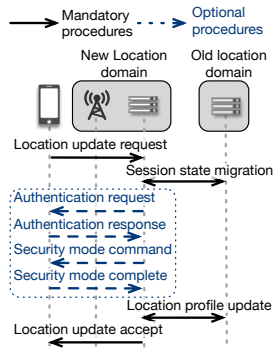
(a) Uplink per-packet latency (b) Downlink total bandwidth

Figure 14: User-plane latency and wireless bandwidth.



	2017-09-30	2019-09-29
t1	16:51:36.418	18:18:02.094
t2	16:51:36.502	18:18:02.149
t3	16:51:36.520	18:18:02.154
t2-t1	**84ms**	**55ms**
t3-t2	**18ms**	**5ms**

(a) Faster signaling processing



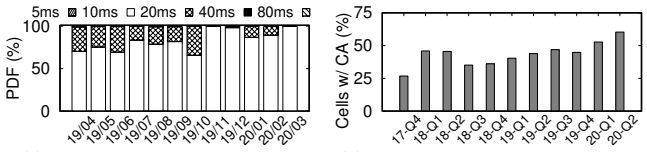
(b) Simplified signaling

Figure 15: Shortened control-plane latency in LTE network.

the fine-grained processing latency at the device and base station. In this case, the device-side signaling processing latency ($t_3 - t_2$) reduces from 18ms in 2017 to 5ms in 2019. The round trip between the device and base station ($t_2 - t_1$) reduces from 84ms in 2017 to 55ms in 2019. Such reduction of round trip could be due to either the shorter radio transmission delay or faster signaling processing at the infrastructure. But the 4G LTE's standard physical channel allocation ensures ≤10ms radio latency for signaling messages [19]. So such significant latency reduction implies faster signaling process by the base station and device, probably due to the infrastructure upgrade and advances of hardware in the newer phone models.

∘ *Simplified signaling procedures:* In device mobility, *optional* control-plane procedures (e.g., re-authentication and temporary ID re-allocation [25]) can be triggered by infrastructure during the tracking area update. Recently, MobileInsight reports more network nodes disable these optional procedures for shorter signaling latency. Figure 15b illustrates this simplification at one location in Los Angeles. For this location, we observe less optional re-authentication procedures if the device is in the connected state. The infrastructure simplifies this procedure for low latency, while stilling retaining reasonable security level since the new location domain can still derive the new security context from the old ones with the standard process [21].

∘ *Shorter scheduling interval configurations:* At user plane, the uplink latency reduction in Figure 14a is due to control-plane reconfiguration. As shown in Figure 4b, LTE adopts on-demand uplink traffic scheduling. To request radio grants for uplink transfer, the device should send a scheduling request to base station. This request *cannot* be initiated anytime. Instead, the base station uses



(a) Uplink: Scheduling interval (b) Downlink: Carrier aggregation

Figure 16: Control-plane settings for user-plane delay.

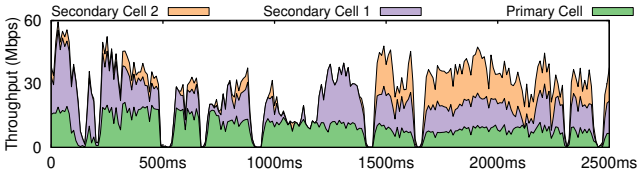


Figure 17: MobileInsight's carrier aggregation breakdown.

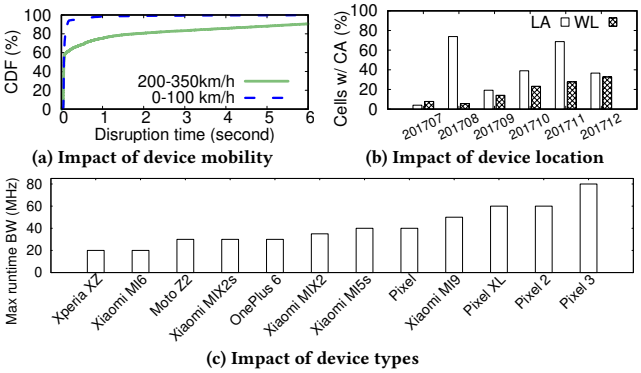
RRC signaling protocol to configure periodic physical-layer time slots for the device to send it[1]. Before this slot is ready, all uplink packets are queued in the MAC buffer and thus delayed. Figure 14a shows such buffering delay dominates the latency of the first uplink packet after radio connectivity setup. As shown in Figure 16a, since 2019, MobileInsight reports that operators have shortened the configuration of this uplink schedule interval. This results in the significant latency reduction in Figure 14a.

∘ *More bandwidth with carrier aggregation:* The user-plane downlink data transfer in Figure 14b is accelerated by more runtime bandwidth. This is enabled by LTE's carrier aggregation, by combining multiple radio frequency carriers. Figure 17 exemplifies the downlink throughput improvement with carrier aggregation. MobileInsight reports that, mobile operators continuously enable the carrier aggregation with more cells. Figure 16b shows the percentage of cells with carrier aggregation increases from 7.6% in 2017 to 60.1% in 2020. Note even if a base station is capable of carrier aggregation, it may not always activate it for all users in reality. To save operators' bandwidth, whether to enable the carrier aggregation depends on the users' runtime traffic demands. The traffic variance results in non-monotonic bandwidth increment in Figure 14b.

Device-side contexts' impact on LTE latency: MobileInsight's device-based analytics also reveal new insights that infrastructure-side analytics cannot easily observe. By grouping and comparing the latencies among the devices, we find that the device contexts have a viable impact on the LTE latency. Figure 18 demonstrates some impacts, including (but not limited to)

∘*The impact of device mobility:* MobileInsight reports longer network latencies when a device moves faster. Figure 18a compares the handover disruption latency under different device movement speed. It shows a device on the high-speed train (200–300 km/h) suffers from longer tail latency than a device at lower speed (0–100 km/h). By tracking the control-plane signaling procedures across the control-plane RRC and user-plane MAC protocols (§4.1), we find this is mainly caused by the frequent handover failures in fast device mobility. As shown in Figure 19, a fast-moving device may leave its serving cell's radio coverage *before* receiving the handover command to the next cell. In this case, the device has to re-scan all available cells to find the next cell to connect, thus prolonging the network service disruption. The faster the device moves, the

[1]This configuration is called SR-ConfigIndex in 4G/5G [29, 30].



(a) Impact of device mobility (b) Impact of device location

(c) Impact of device types

Figure 18: The device context's impact on LTE latency.

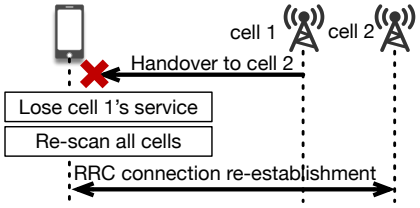


Figure 19: Longer latency under faster device mobility.

more likely this failure would happen. Our traces indicate that, the handover failure ratio increases from 4.3% at 0–100 km/h to 12.3% at 200–350 km/h, thus resulting in the long-tail latency in Figure 18a.

∘*The impact of device location:* The user-perceived network latencies also vary with locations due to heterogeneous radio coverages, network capacities and configurations. As an example, Figure 18b shows the percentage of cells with carrier aggregation in Los Angeles (LA) and West Lafayette (WL) in 2017. Despite the variations over time, Los Angeles typically has more cells with carrier aggregation than West Lafayette, thus offering shorter downlink transmission delay for devices. A possible explanation is that, carrier aggregation was a relatively new technology in 2017, so it was first widely deployed in larger cities like Los Angeles in 2017. After three years, we note West Lafayette now has comparable percentage of cells with carrier aggregation to Los Angeles.

∘*The impact of device types:* Different mobile devices have heterogenous hardware modems and capabilities that affect their latencies and bandwidth. Based on MobileInsight's runtime physical-layer logs, Figure 18c shows the maximum radio bandwidth allocated for each device. In general, a device with earlier or lower-end generations of hardware modem has less bandwidth and thus slower data speed. With advanced radio technologies in newer chipsets, and device can enjoy more bandwidth and less latency at runtime.

5.2 Using MobileInsight in Broader Scenarios

We summarize the experiences of using MobileInsight in broader scenarios. We review the community users' diverse incentives, the diverse use cases, and the lessons of adopting MobileInsight.

5.2.1 *Who have incentives to adopt MobileInsight?*

As shown in Figure 3, MobileInsight has been used by 350+ academic and industrial institutes since 2016. Most MobileInsight users are app developers, researchers, regulators, and mobile operators.

For app developers and researchers, MobileInsight enables an alternative approach to conduct mobile network R&D *without* collaboration with the mobile operators. For regulators, MobileInsight allows them to *independently* verify the operational mobile networks, without unconditionally trusting the operators. For mobile operators, although they could conduct analytics with their 4G/5G infrastructure, they still lack *end-to-end* analytical results from the end device. In fact, many user-experienced performances and reliability issues are invisible from infrastructure-side analytics. MobileInsight fills out this gap to complement operators' infrastructure-side solutions.

5.2.2 *How is MobileInsight used in diverse scenarios?*

Table 3 shows some representative use cases of MobileInsight by the community. MobileInsight has been applied to diverse usage scenarios (Web, video, VR, mobile sensing, remote driving, high-speed trains, to name a few) to enhance the network reliability, performance, energy efficiency, and security. These use cases accumulate various lessons of adopting MobileInsight for device-based, data-driven mobile intelligence and customization.

The necessity of MobileInsight's advanced features (§4): As evidenced in Table 3, the initial MobileInsight design cannot fully meet the demands from all these usage scenarios. Various tasks require to study the reliability deficiencies, performance bottleneck, energy wastes, and security vulnerabilities across the protocols at the control and user planes. This justifies the necessity of MobileInsight's cross-layer analytics in §4.1. Besides, many tasks need efficient runtime user-plane analytics (e.g., link capacity estimation [32, 33, 48] and latency reduction [71, 72] for performance boosting) inside the devices, which motivates MobileInsight to build cross-layer sampling in §4.2 to meet this demand. Furthermore, we note that many analytics in Table 3 can be tracked by MobileInsight's recent KPI analyzers in §4.3, which helps users simplify their tasks.

The benefits with MobileInsight: As shown in Table 3, MobileInsight has demonstrated its potentials to benefit a broad range of usage scenarios. It enables accurate physical-layer link capacity estimation inside the device, which achieves up to 93% accuracy of detecting if LTE downlink radio is the TCP bottleneck [32], helps reduce the web loading time by 30% [76], and improve PSNR by up to 6dB in WebRTC-based video streaming [48, 75]. It allows for accurate inference of infrastructure-side handover decision policies with up to 95% accuracy, which helps prevent unnecessary TCP degradation on high-speed trains [74] and masks latency at the application layer. MobileInsight's cross-layer analytics helps unveil various deficiencies of network latency, and helps achieve 2.1–11.5× control-plane latency reduction [54] and ≤25ms user-plane latency with ≥95% probability for VR [71]. MobileInsight helps detect diverse network misconfigurations [36] and policy conflicts [49] some of which cannot be detected by the infrastructure (e.g., conflicts among carriers in Google Fi [81]). It also helps unveil network-induced energy deficiencies [35], and security threats from call spoofing [37] (with ≈100% accuracy) and signaling protocols [45].

The limitations of the current MobileInsight: While encouraging, users have also reported some scenarios that the current MobileInsight cannot fully satisfy. First, some useful mobile network information has not been revealed, such as the radio resource allocation among devices (for congestion control [77]) and the angle of antenna (for user localization [58]). This could be resolved by extending MobileInsight for crowdsourcing analysis among users, and developing more inference techniques. Second, MobileInsight requires system privilege (root for Android, jailbreak for iOS) to access the fine-grained network information, which limits its applicability to more users [68, 82]. We plan to mitigate this issue by integrating MobileInsight into the mobile OS as a system application, or developing rootless inference techniques inspired by MobileInsight [72]. Last but not least, MobileInsight's existing cross-layer user-plane sampling may still not be enough for battery-constrained IoT devices [80]. We plan to enhance MobileInsight's future releases with more IoT-friendly analytics.

6 RELATED WORK

Mobile network has been one of the most active research areas for decades. Numerous efforts have been made to refine performance [32, 54, 59, 76], resiliency [49, 56, 73, 81], energy efficiency [35, 78], operational cost [36, 51, 55], and security [37, 41, 69] in diverse usage scenarios [40, 60, 66, 75]. This creates great demands for experimentation and validation with real mobile networks, and therefore various open platforms such as ORAN [7], Powder [8], CoLTE [70], srsLTE [10], OpenAirInterface [6], etc. These platforms largely run in a controlled environment. MobileInsight complements them with *operational* mobile network data and analytics.

In the area of mobile network analytics, MobileInsight is also orthogonal to most solutions today. Traditional analytics reside in the infrastructure [43, 65]. MobileInsight complements them with in-device analytics. While QXDM [67], OWL [34], and LTEye [46] support device-side analytics, they require external hardware (i.e., desktop or USRP). X-CAL Mobile [31] and Network Signal Guru [5] have similar data collection to MobileInsight. But they are mainly a logger for offline analysis, rather than in-device runtime analytics. To our best knowledge, only MobileInsight offers open-source full-stack, runtime mobile network analytics in commodity devices.

7 DISCUSSION AND CONCLUSION

MobileInsight is the first attempt to build an *open* community tool that enables in-device, software-defined mobile network analytics. It follows the end-to-end principle to facilitate the shift from infrastructure-based to device-based mobile network intelligence and customization. While encouraging, more efforts are needed to refine MobileInsight*for* the community, and *by* the community. We conclude this work by summarizing our thoughts of the past, present, and future of MobileInsight.

Looking back: Rethinking choices in MobileInsight's design. MobileInsight's success is largely attributed to three key choices we made. First, we aimed at an *in-device, software-defined* solution. This ensures that MobileInsight can be readily deployed in commodity phones today, and meet demands from a broader community of researchers and developers. Second, we decided to keep MobileInsight's framework simple, modular and extensible. This facilitates MobileInsight's continuous evolution and refinement as shown in §4. Last but not least, we chose to open-source MobileInsight. This not only results in wide adoption, but also encourages more users to contribute to MobileInsight's development.

Of course, we also made some sub-optimal choices for MobileInsight. First, MobileInsight took a bottom-up evolution from low-level analytics to user-friendly KPIs (§3.3). In the early stages of developing MobileInsight, we focused on implementing a *complete* set of low-level analytics features but ignored most users' real demands for simple APIs for intuitive mobile network analytics. This unnecessarily complicated the MobileInsight codebase for most users and delayed its wide adoption. Instead, a top-down evolution could broaden MobileInsight's usage, i.e., interview and understand the users' specific demands for mobile network analytics, enable *the corresponding* high-level KPIs first to attract more users (rather than a complete yet complicated low-level primitives), educate them with hands-on KPI experiences, and then extend MobileInsight to low-level analytics. Second, to enable extensible analytics plugins, we built MobileInsight with Python because of its script-like semantics. This turns out to be painful. Without Android's native support, we built MobileInsight with a 3rd-party Python-for-Android package [9]. This caused many unexpected bugs and slow Python2→Python3 migration in MobileInsight's development. We are now migrating MobileInsight to Java and C/C++ using Android's native support, and retain its extensible analytics plugin with Javascript. Third, MobileInsight was packaged as a standalone app, which is hard for 3rd-party apps to call and integrate. We plan to repackage MobileInsight as an Android SDK library to facilitate 3rd-party programming and integration.

Looking now: Potentials and limitations of using MobileInsight for device-based network intelligence. MobileInsight was originally designed for network analytics only. But it turns out that, MobileInsight has also facilitated diverse device-based mobile network customization and intelligence as shown in §5 and Table 3. These efforts unveil three general insights on empowering the device-based network intelligence with MobileInsight:

∘ *Full-stack device-side customization:* As a device-side solution, MobileInsight has the unique advantage of *full-stack* intelligence, from app to physical layers. This was not envisioned when initiating the MobileInsight project in 2016. MobileInsight facilitates app-driven network customization and network-aware app adaptation that infrastructure cannot. Specifically, emergent scenarios (AR/VR, video surveillance, smart home, IoT, etc.) involve *heavy uplink data transfer*. By customizing the uplink network data transfer with local app information, MobileInsight can help achieve ≤25ms user-plane latency with ≥95% probability for mobile VR [71], reduce the web loading time by 30% [76], and improve PSNR by up to 6dB in WebRTC-based video streaming [48, 75]. These improvements cannot be easily achieved by the infrastructure, which has no full access to the device-side app demands and uplink traffic patterns.

∘ *Inferring network-side operations:* Even with MobileInsight, some network-side operations are still not fully visible to devices, such as the radio resource scheduling and handover decision policies. In this case, MobileInsight can infer *device-perceived* operation logic at best. Such inferred logics are indeed not fully identical to the ground truth, but they can be still helpful for devices. For example, with the runtime physical radio resource allocation information from MobileInsight, [32] achieves 93% accuracy of detecting if LTE downlink radio is the TCP bottleneck and [48] predicts the uplink radio bandwidth with the mean error rate as low as 7.67%. These inferences can guide TCP to avoid unnecessary data rate drops.

∘ *Harnessing from mobility:* Device mobility across the network nodes complicates analytics and optimization. To tackle it, existing infrastructure-side solutions [43, 65] must coordinate the network nodes. This is complex or sometimes impossible if network nodes belong to different operators (e.g., international roaming [55] and virtual operators like Google Fi [14]). Instead, our experience of using MobileInsight unveils a unique opportunity to address this issue: As a single vantage point across network nodes, a device with MobileInsight can reveal more network-side insights as it moves, simplify infrastructure-side solutions, or complement them with insights that network alone cannot gain. For example, with MobileInsight, [36, 49] can use commodity phones to conduct mobility misconfiguration analysis for 18,000+ cells and 32,000 handover instances as the phone moves across base stations, without relying on these base stations to coordinate or share their local configurations. Moreover, in the multi-carrier access such as Google Fi [14] (which combines T-Mobile and Sprint for better coverage), each individual operator does not have global view on all others to fully optimize their network services at the infrastructure side. Instead, by allowing the device to analyze each carrier with MobileInsight during its mobility, [50] realizes intelligent carrier selection with 3.74× throughput increment and 1.9× latency reduction, and [81] resolves Google Fi's persistent handover oscillations due to the mobility management policy conflicts between T-Mobile and Sprint.

Looking forward: The future of MobileInsight. MobileInsight will continue its evolution toward a comprehensive, efficient, and user-friendly open community analytics paradigm. In the foreseen future, we will extend MobileInsight to support upcoming 5G and cellular IoT technologies[2]. We will extend MobileInsight to the mobile devices beyond commodity phones, and customize its features and energy efficiency for battery-constrained IoT devices (e.g., via application-driven intelligent sampling). Beyond a single device, MobileInsight will unleash more network intelligence by crowdsourcing massive devices and cooperating with software-defined radios. To scale to more devices, we will relax MobileInsight's dependency on system privilege by exploring learning-based paradigms with coarse-grained data. In the long term, we envision MobileInsight could facilitate the users and operators to share their network knowledge and achieve collaborative mobile network intelligence. We wish more community efforts would join us to move toward a transparent and intelligent next-generation mobile network.

Acknowledgements. We greatly thank our anonymous shepherd and reviewers for their constructive comments. We thank all MobileInsight users for their valuable feedback and suggestions. We are also grateful to all the participants in the five-year longitudinal study. We thank Zengwen Yuan, Zizheng Liu, Wei Liu, Jiayi Liu, Jiayao Li and Zhehan Li for their contributions to MobileInsight by courtesy. Yuanjie Li and Hewu Li are sponsored by the National Key Research and Development Plan of China (2018YFB1800301) and National Natural Science Foundation of China (61832013). Others are in part supported by National Science Foundation (CNS-1750953,CNS-1910150, CNS-2008026 and CNS-2027650).

[2]The latest v6.0 beta release (in Nov 2020) has provided the preliminary support for 5G control plane analytics [13]. More support for 5G, NB-IoT and LTE-M is ongoing.

REFERENCES

[1] Android source code for Meadiatek cellular diagnostic mode. https://android.googlesource.com/kernel/mediatek/+/android-6.0.1_r0.110/drivers/misc/mediatek/.

[2] Android source code for Qualcomm cellular diagnostic mode. https://android.googlesource.com/kernel/msm.git/+/android-11.0.0_r0.34/drivers/char/diag/diagchar_core.c.

[3] Android.telephony. http://developer.android.com/reference/android/telephony/package-summary.html.

[4] iOS baseband commands. https://www.reddit.com/r/jailbreak/comments/37ap5g/is_there_a_way_to_view_my_baseband_logs_in_ifile/.

[5] Network Signal Guru. https://play.google.com/store/apps/details?id=com.qtrun.QuickTest&hl=en_US.

[6] OpenAirInterface. https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/home.

[7] ORAN Alliance. https://www.o-ran.org/.

[8] Powder: Platform for Open Wireless Data-driven Experimental Research. https://powderwireless.net/.

[9] Python-for-android project. https://python-for-android.readthedocs.org/en/latest/.

[10] srsLTE. https://www.srslte.com/.

[11] xgoldmon. https://github.com/2b-as/xgoldmon.

[12] Android telephonymanager class. https://tinyurl.com/yjznbh9, 2019.

[13] www.mobileinsight.net, Dec 2020.

[14] Google Fi. https://fi.google.com/, 2020.

[15] Speedtest Global Index: United States. https://www.speedtest.net/global-index/united-states#market-analysis, 2020. Accessed on 07/18/2020.

[16] Dataset of a five-year delay study. https://github.com/mobile-insight/mobicom21-data-release, 2021.

[17] Mobileinsight code repo. https://github.com/mobile-insight, Jan 2021.

[18] 3GPP. TS36.322: Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Link Control (RLC) protocol specification, Sep. 2012.

[19] 3GPP. TS36.321: Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification, Mar. 2014.

[20] 3GPP. TS36.321: Evolved Universal Terrestrial Radio Access (E-UTRA); Packet Data Convergence Protocol (PDCP) specification, Jun. 2014.

[21] 3GPP. TS33.401: 3GPP System Architecture Evolution (SAE); Security architecture, Jun. 2016.

[22] 3GPP. TS38.322: Technical Specification Group Radio Access Network; NR; Packet Data Convergence Protocol (PDCP) specification, Jun. 2017.

[23] 3GPP. TS38.321: 5G NR; Medium Access Control (MAC) protocol specification, Jun. 2019.

[24] 3GPP. TS38.323: 5G NR; Packet Data Convergence Protocol (PDCP) specification, Jun. 2019.

[25] 3GPP. TS24.301: Non-Access-Stratum (NAS) for EPS, Jul. 2020.

[26] 3GPP. TS24.501: Non-Access-Stratum (NAS) protocol for 5G System (5GS), Jul. 2020.

[27] 3GPP. TS28.554: 5G end to end Key Performance Indicators (KPI), Jul. 2020.

[28] 3GPP. TS32.450: Key Performance Indicators (KPI) for Evolved Universal Terrestrial Radio Access Network (E-UTRAN), Jul. 2020.

[29] 3GPP. TS36.331: Radio Resource Control (RRC), Jul. 2020.

[30] 3GPP. TS38.331: 5G NR: Radio Resource Control (RRC), Jul. 2020.

[31] Accuver. XCAL-Mobile. http://www.accuver.com.

[32] Balasingam, A., Bansal, M., Misra, R., Nagaraj, K., Tandra, R., Katti, S., and Schulman, A. Detecting if lte is the bottleneck with bursttracker. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom)* (2019), pp. 1–15.

[33] Bui, N., Michelinakis, F., and Widmer, J. Fine-grained lte radio link estimation for mobile phones. *Pervasive and Mobile Computing 49* (2018), 76–91.

[34] Bui, N., and Widmer, J. Owl: A reliable online watcher for lte control channel measurements. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges* (2016), pp. 25–30.

[35] Chen, X., Meng, J., Hu, Y. C., Gupta, M., Hasholzner, R., Ekambaram, V. N., Singh, A., and Srikanteswara, S. A fine-grained event-based modem power model for enabling in-depth modem energy drain analysis. *Proceedings of the ACM on Measurement and Analysis of Computing Systems (POMACS) 1*, 2 (2017), 1–28.

[36] Deng, H., Peng, C., Fida, A., Meng, J., and Hu, Y. C. Mobility support in cellular networks: A measurement study on its configurations and implications. In *Proceedings of the Internet Measurement Conference 2018* (2018), ACM, pp. 147–160.

[37] Deng, H., Wang, W., and Peng, C. Ceive: Combating caller id spoofing on 4g mobile phones via callee-only inference and verification. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking* (2018), pp. 369–384.

[38] Federal Communications Commission (FCC). Measuring Broadband America Mobile 2013-2018 Coarsened Data. https://www.fcc.gov/reports-research/reports/measuring-broadband-america/measuring-broadband-america-mobile-2013-2018, 2019.

[39] Federal Communications Commission (FCC). Measuring Mobile Broadband. https://www.fcc.gov/general/measuring-mobile-broadband-performance, 2020.

[40] Foukas, X., Nikaein, N., Kassem, M. M., Marina, M. K., and Kontovasilis, K. Flexran: A flexible and programmable platform for software-defined radio access networks. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies* (2016), pp. 427–441.

[41] Hussain, S., Chowdhury, O., Mehnaz, S., and Bertino, E. Lteinspector: A systematic approach for adversarial testing of 4g lte. In *Network and Distributed Systems Security (NDSS) Symposium 2018* (2018).

[42] Hussain, S. R., Echeverria, M., Chowdhury, O., Li, N., and Bertino, E. Privacy Attacks to the 4G and 5G Cellular Paging Protocols Using Side Channel Information. In *NDSS* (2019), vol. 19, pp. 24–27.

[43] Iyer, A., Li, L. E., and Stoica, I. Celliq: Real-time cellular network analytics at scale. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI'15)* (2015), pp. 309–322.

[44] Jia, Y. J., Chen, Q. A., Mao, Z. M., Hui, J., Sontinei, K., Yoon, A., Kwong, S., and Lau, K. Performance characterization and call reliability diagnosis support for voice over lte. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* (2015), pp. 452–463.

[45] Kim, H., Lee, J., Lee, E., and Kim, Y. Touching the untouchables: Dynamic security analysis of the lte control plane. In *2019 IEEE Symposium on Security and Privacy (S&P)* (2019), IEEE, pp. 1153–1168.

[46] Kumar, S., Hamed, E., Katabi, D., and Erran Li, L. Lte radio analytics made easy and accessible. *ACM SIGCOMM Computer Communication Review 44*, 4 (2014), 211–222.

[47] Lakshmanan, N., Budhdev, N., Kang, M. S., Chan, M. C., and Han, J. A stealthy location identification attack exploiting carrier aggregation in cellular networks.

[48] Lee, J., Lee, S., Lee, J., Sathyanarayana, S. D., Lim, H., Lee, J., Zhu, X., Ramakrishnan, S., Grunwald, D., Lee, K., et al. Perceive: deep learning-based cellular uplink prediction using real-time scheduling patterns. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services (MobiSys)* (2020), pp. 377–390.

[49] 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 *The 42nd ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'16)* (Antibes Juan-les-Pins, France, June 2016).

[50] Li, Y., Deng, H., Peng, C., Tu, G.-H., Li, J., Yuan, Z., and Lu, S. iCellular: Define Your Own Cellular Network Access on Commodity Smartphones. In *Accepted by USENIX NSDI* (March 2016). Draft available: http://arxiv.org/abs/1510.08027.

[51] Li, Y., Kim, K.-H., Vlachou, C., and Xie, J. Bridging the data charging gap in the cellular edge. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)* (2019), ACM, pp. 15–28.

[52] Li, Y., Peng, C., Deng, H., Yuan, Z., Tu, G.-H., Li, J., Lu, S., and Li, X. Device-customized multi-carrier network access on commodity smartphones. *IEEE/ACM Transactions on Networking 26*, 6 (2018), 2542–2555.

[53] Li, Y., Peng, C., Yuan, Z., Li, J., Deng, H., and Wang, T. Mobileinsight: Extracting and analyzing cellular network information on smartphones. In *The 22nd ACM Annual International Conference on Mobile Computing and Networking (Mobicom'16)* (New York, USA, Oct. 2016).

[54] Li, Y., Yuan, Z., and Peng, C. A Control-Plane Perspective on Reducing Data Access Latency in LTE Networks. In *ACM MobiCom* (Snowbird, Utah, USA, Oct. 2017).

[55] Li, Y., Zheng, J., Li, Z., Liu, Y., Qian, F., Bai, S., Liu, Y., and Xin, X. Understanding the ecosystem and addressing the fundamental concerns of commercial mvno. IEEE.

[56] Li, Yuanjie and Li, Qianru and Zhang, Zhehui and Baig, Ghufran and Qiu, Lili and Lu, Songwu. Beyond 5g: Reliable extreme mobility management. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)* (2020), ACM.

[57] Liu, R., Kwak, D., Devarakonda, S., Bekris, K., and Iftode, L. Investigating remote driving over the lte network. In *Proceedings of the 9th International Conference on Automotive User Interfaces and Interactive Vehicular Applications* (2017), pp. 264–269.

[58] Meng, J., Sharma, A., Tran, T. X., Balasubramanian, B., Jung, G., Hiltunen, M., and Hu, Y. C. A Study of Network-Side 5G User Localization Using Angle-Based Fingerprints. In *2020 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)* (2020), IEEE, pp. 1–6.

[59] Misra, R., Gudipati, A., and Katti, S. Quickc: practical sub-millisecond transport for small cells. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking* (2016), pp. 109–121.

[60] Moradi, M., Sundaresan, K., Chai, E., Rangarajan, S., and Mao, Z. M. Skycore: Moving core to the edge for untethered and reliable uav-based lte networks. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking* (2018), pp. 35–49.

[61] Ookla. United States: Q1-Q2 2019. https://www.speedtest.net/reports/united-states/, 2019. Accessed on 07/18/2020.

[62] OpenSignal. USA: Mobile Network Experience Report July 2019. https://www.opensignal.com/reports/2019/07/usa/mobile-network-experience, 2019. Accessed on 08/24/2019.

[63] OpenSignal. USA: Mobile Network Experience Report January 2020. https://www.opensignal.com/reports/2020/01/usa/mobile-network-experience, 2020. Accessed on 03/28/2020.

[64] OpenSignal. USA: Mobile Network Experience Report July 2020. https://www.opensignal.com/reports/2020/07/usa/mobile-network-experience, 2020. Accessed on 07/16/2020.

[65] 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), ACM, pp. 513–528.

[66] Qazi, Z. A., Walls, M., Panda, A., Sekar, V., Ratnasamy, S., and Shenker, S. A high performance packet core for next generation cellular networks. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (2017), pp. 348–361.

[67] Qualcomm. QxDM Professional - QUALCOMM eXtensible Diagnostic Monitor. http://www.qualcomm.com/media/documents/tags/qxdm.

[68] Raca, D., Quinlan, J. J., Zahran, A. H., and Sreenan, C. J. Beyond throughput: a 4g lte dataset with channel and context metrics. In *Proceedings of the 9th ACM Multimedia Systems Conference* (2018), pp. 460–465.

[69] Rupprecht, D., Kohls, K., Holz, T., and Pöpper, C. Breaking lte on layer two. In *2019 IEEE Symposium on Security and Privacy (S&P)* (2019), IEEE, pp. 1121–1136.

[70] Sevilla, S., Johnson, M., Kosakanchit, P., Liang, J., and Heimerl, K. Experiences: Design, implementation, and deployment of colte, a community lte solution. In *The 25th Annual International Conference on Mobile Computing and Networking* (2019), pp. 1–16.

[71] Tan, Z., Li, Y., Li, Q., Zhang, Z., Li, Z., and Lu, S. Enabling Mobile VR in LTE Networks: How Close Are We? In *ACM SIGMETRICS* (2018).

[72] Tan, Zhaowei and Zhao, Jinghao and Li, Yuanjie and Xu, Yifei and Lu, Songwu. Device-Based LTE Latency Reduction at the Application Layer. In *Symposium on Networked Systems Design and Implementation (NSDI)* (2021), USENIX.

[73] Tu, G.-H., Li, Y., Peng, C., Li, C.-Y., Wang, H., and Lu, S. Control-Plane Protocol Interactions in Cellular Networks. In *SIGCOMM* (2014).

[74] Wang, J., Zheng, Y., Ni, Y., Xu, C., Qian, F., Li, W., Jiang, W., Cheng, Y., Cheng, Z., Li, Y., et al. An active-passive measurement study of tcp performance over lte on high-speed rails. In *The 25th Annual International Conference on Mobile Computing and Networking* (2019), ACM, pp. 1–16.

[75] Xie, X., and Zhang, X. Poi360: Panoramic mobile video telephony over lte cellular networks. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies* (2017), pp. 336–349.

[76] Xie, X., Zhang, X., and Zhu, S. Accelerating mobile web loading using cellular link information. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services* (2017), pp. 427–439.

[77] Xie, Y., Yi, F., and Jamieson, K. PBE-CC: Congestion Control via Endpoint-Centric, Physical-Layer Bandwidth Measurements.

[78] 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.

[79] Xu, S., Nikravesh, A., and Mao, Z. M. Leveraging context-triggered measurements to characterize lte handover performance. In *International Conference on Passive and Active Network Measurement* (2019), Springer, pp. 3–17.

[80] 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 (MobiCom)* (2020), pp. 1–13.

[81] Yuan, Z., Li, Q., Li, Y., Lu, S., Peng, C., and Varghese, G. Resolving Policy Conflicts in Multi-Carrier Cellular Access. In *The 24th ACM Annual International Conference on Mobile Computing and Networking (MobiCom'18)* (New Delhi, India, Oct. 2018).

[82] Yue, C., Jin, R., Suh, K., Qin, Y., Wang, B., and Wei, W. Linkforecast: cellular link bandwidth prediction in lte networks. *IEEE Transactions on Mobile Computing 17*, 7 (2017), 1582–1594.

[83] Zhang, Z., Shi, S., Gupta, V., and Jana, R. Analysis of cellular network latency for edge-based remote rendering streaming applications. In *Proceedings of the ACM SIGCOMM 2019 Workshop on Networking for Emerging Applications and Technologies* (2019), pp. 8–14.