iCellSpeed: Increasing Cellular Data Speed with Device-Assisted Cell Selection

Haotian Deng
Purdue University

Qianru Li
UCLA

Jingqi Huang
Purdue University

Chunyi Peng
Purdue University

ABSTRACT

In this paper, we propose iCellSpeed, an on-device solution to increase data access speed by substantiating unrealized performance potentials. We find that performance potentials are missed in today's mobile networks, as the data speed a user device gets is much lower than what the device could get. The issue is rooted in the current cell selection practice, which misses good candidate cells that offer faster access speed, thus under-utilizing the available capabilities in mobile networks. We design iCellSpeed to facilitate network-controlled cell selection with proactive device-side assistance towards more desirable cells. Our evaluation over AT&T and Verizon confirms its effectiveness. iCellSpeed increases data access speed by more than 10 Mbps at 79% of test locations (> 25Mbps at 29% of locations, up to 80.6 Mbps). It doubles access speed at 62.5% of locations with the gain up to 28.4x. Datasets are available at [7].

CCS CONCEPTS

• **Networks** → **Mobile networks**; **Network performance evaluation**; **Network control algorithms**.

KEYWORDS

4.5G, Downlink Speed, Cell Selection, Device-Assisted, iCellSpeed

ACM Reference Format:
Haotian Deng, Qianru Li, Jingqi Huang, and Chunyi Peng. 2020. iCellSpeed: Increasing Cellular Data Speed with Device-Assisted Cell Selection. In *The 26th Annual International Conference on Mobile Computing and Networking (MobiCom '20), September 21–25, 2020, London, United Kingdom.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3372224.3419201

1 INTRODUCTION

Increasing user access speed has been a main driver for mobile network evolution. Operators expect to deliver faster broadband experience through continuous infrastructure upgrade. They acquire wider radio spectrum, deploy denser cells, and migrate to advanced technologies (say, from 4G LTE to 4.5G LTE-Advanced to 5G New Radio). They enhance raw system capabilities to offer users higher speed (say, from tens of Mbps to a few Gbps).

In this work, we argue that it is equally important to make full use of already available capacities, rather than deploy new ones, to boost data access performance to user devices. Without exploiting full performance potentials, we end up with excessive infrastructure waste, which further drives the operators for infrastructure upgrade even more aggressively, thus being trapped in a vicious cycle. Specifically, the data performance would stay poor if those cells with low access speed are selected to serve a device, despite many candidate cells which are able to offer faster speed. Eventually, those cells that offer faster access are not chosen, thus being wasted. The overall system utilization remains low.

Unfortunately, we discover the above is not rare in reality. We observe that significant performance gaps emerge in various settings between what a device could get and what it actually gets. For example, we see that a mobile phone receives 4.6 Mbps on average by default while 49.3 Mbps is available and 907% speed potential is missed (Figure 3a); We observe significant performance potentials missed in our measurement study of all four top-tier US operators (AT&T, Verizon, T-Mobile and Sprint)[1] (Figure 11, §5.2). We further dive into why such low-utilization cases arise. It turns out that, the state-of-the-practice cell selection[2] is held accountable. Without proper selection of those cells that yield higher access speed, the current scheme under-utilizes the available infrastructure resources and misses fast wireless access to user devices.

Fundamentally, the problem is rooted in the network-centric cell selection scheme, which is designed for seamless connectivity but not for superior data performance. Note that, dense cell deployment is the norm, rather than exception, in most US regions. A user device can be served by multiple candidate cells (up to tens of cells) in principle. Despite abundant choices, few are considered in practice, and the poor ones are further chosen not without technical rationale. The practice favors those cells with good radio signal, but ignores other non-radio factors (e.g., channel width) that may have bigger impact on user-perceived data performance. The handoff procedure states that, as long as the radio quality is not too bad, current cell would remain effective despite bad data performance. It thus discourages leaving the current cell. When searching for new cells, the procedure stops once a candidate cell with acceptable radio quality is found. It thus often stops at a local sub-optimum out of a subset of candidates constrained by the current cell. Consequently, it misses the cells with the best performance out of *all choices*.

In this work, we propose iCellSpeed to address the identified underutilization issue. iCellSpeed has three design requirements. First, it seeks to reach the performance-oriented, *"global"* optimum from all candidates, rather than radio-based, *"local"* optimum. Second, it is compatible with the 3GPP standards without any infrastructure change. Third, iCellSpeed is designed as an on-device software solution to boosting mobile data access speed for the device. In brief, the design of iCellSpeed transforms a mobile device from a telecom-based *dumb, passive* terminal to an *intelligent, proactive*

[1] T-Mobile and Sprint run separately during this study despite their pending merger.
[2] It is called handoff in 3GPP. We focus on how to select serving cells in this paper. We use both terms interchangeably.

MobiCom '20, September 21–25, 2020, London, United Kingdom

ACM ISBN 978-1-4503-7085-1/20/09…$15.00
https://doi.org/10.1145/3372224.3419201

machine. Instead of passively following the decision made by the network, the device learns what is best for itself and supplies its own favored choices for the final selection made by the network.

There are two main technical challenges. First, device capabilities at the software space are quite limited. Even though the device learns the desired choice, it cannot directly select it. Second, it must work with the current selection mechanism that makes the final decision at the infrastructure side.

To tackle both challenges, iCellSpeed takes an approach of the *"device-assisted, infrastructure-decided"* selection (§4). The device takes runtime measurements and maintains historical performance profile to challenge whether it should accept the default decision made by the network; It performs online learning to infer missed potentials and determines its corresponding action (device-side customization). It further monitors the outcome to ensure better performance than the default choice.

We implement iCellSpeed on commodity smartphones, and confirm its effectiveness with large speed gains in both AT&T and Verizon. It increases access speed by at least 10 Mbps at 79.2% of locations out of 50 runs per location. It more than doubles the speed at 62.5% of locations, and achieves the gain up to 8.11x at one location (up to 28.4x in a single run). We want to highlight that iCellSpeed has one limitation that it would disrupt ongoing traffic for about 2 seconds on the application layer. As a result, it may not benefit mice traffic despite its improvement for bandwidth-intensive heavy traffic flows like video streaming, conferencing and file downloading. This limitation is rooted in its implementation constraint, as the device is incapable of taking action swiftly in the *software* space without corresponding support from the chipset. We note that iCellSpeed is designed to exploit under-utilized potentials, not to increase raw system potentials. There is no speed benefit on the network side in case full potentials have been used up by other devices. In a word, iCellSpeed is not perfect but offers a promising on-device solution to increase data speed.

In summary, iCellSpeed offers arguably the first on-device, pure software solution that guides infrastructure-centric cell selection for higher access speed. We make three main contributions:

(1) We conduct extensive measurements to identify and quantify the missed performance potentials in today's mobile networks (Findings F1-F5 in §3.1, §5.2).
(2) We unveil the limitations of current cell selection for missed potentials (F6 - F12 in §3.2).
(3) We design, implement, and evaluate iCellSpeed on commodity smartphones with instant speed gains (§4 and §5).

2 RADIO ACCESS PRIMER

We introduce necessary background on radio access in mobile networks, as illustrated in Figure 1. We focus on 4.5G and beyond (e.g., 4.5G LTE-Advanced and 4.75G LTE-Advanced Pro), because all four major US operators have advanced to 4.5G and beyond.

Cells and frequency channels/carriers. A mobile device's radio access is provided by one or more serving cells. A cell is a *logical* unit that runs over a contiguous spectrum frequency block (referred to a frequency channel or a component carrier) to serve devices within its vicinity ranging from tens of meters to several kilometers. Each channel has fixed operational frequency and bandwidth (say, 1.4, 3, 5, 10, 15 and 20 MHz, Table 1), as regulated by 3GPP [11].

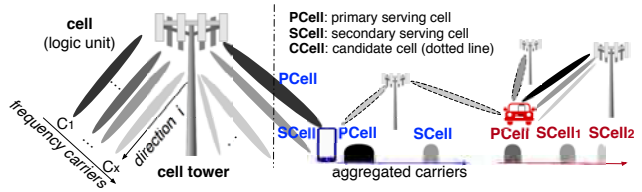


Figure 1: Radio access in 4.5G and beyond.

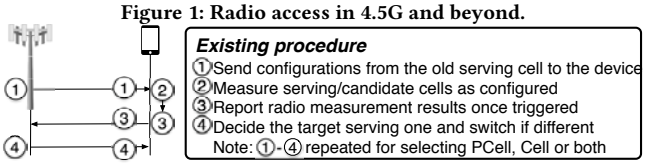


Figure 2: State-of-the-practice cell selection procedure.

Band	Downlink	Width	Channel No.	Bandwidths (MHz)
2	1930 – 1990	60 MHz	600 – 1199	1.4, 3, 5, 10, 15, 20
4	2110 – 2155	45 MHz	1950 – 2399	1.4, 3, 5, 10, 15, 20
5	869 –894	25 MHz	2400 – 2649	1.4, 3, 5, 10
12	728 – 746	18 MHz	5010 – 5179	1.4, 3, 5, 10
30	2350 – 2360	10 MHz	9770 – 9869	5, 10
66*	2110 – 2200	90 MHz	66436 – 67335	1.4, 3, 5, 10, 15, 20

Table 1: Examples of bands and downlink channels [11, 23].
* **band 66 is a superset of band 4.**

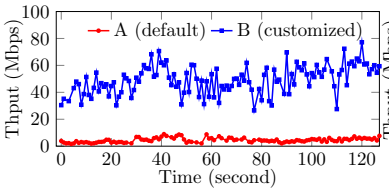
It is uniquely identified by the channel number, say, EARFCN for 4G/4.5G and beyond. All channels reside within multiple licensed bands (e.g., bands 2,4,5,12,29,30,66 for AT&T, bands 2,4,5,13,66 for Verizon [23]). A cell is operated at a cell tower, which is a *physical* entity deployed for a small geographic area, accommodating a number of logical cells via directional antennas and multiple channels. Channels are assigned during cell deployment phase.

Carrier aggregation in 4.5G and beyond. Intuitively, larger bandwidth (wider spectrum) promises higher data speed. A key advance from 4G to 4.5G and beyond is carrier aggregation (CA), which allows more than one serving cells to offer simultaneous radio access, thus increasing bandwidth on an aggregated carrier (over multiple individual carriers with each being used by a cell) . The set of serving cells consists of a primary cell (PCell) and several secondary cells (SCells). PCell is mandatory and needed for data transmission and connection management. SCells are optional and used for data transmission only. In principle, CA supports up to 100 MHz by aggregating maximum five 20MHz carriers [9]. In our study, we see at most two SCells in all four major US operators.

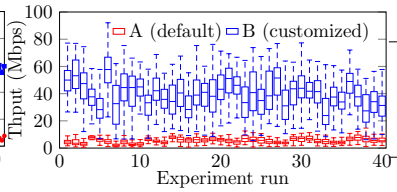
Selection of serving cell(s). Serving cells are selected from a number of candidate cells (dash line in Figure 1). When the device is active with ongoing traffic, cell selection is realized by the standard procedure handoff [12]. Figure 2 depicts its operation flow in four steps: ① configuration, ② measurement, ③ reporting, and ④ decision and execution. The first step of configuration defines criteria and parameters to trigger, decide and execute a handoff, including whether to invoke measurement, what/when to measure, whether/what/when to report and how to decide the next cell, and so on. The followup steps (② - ④) are invoked when the conditions pre-configured by the current serving cell (PCell) are met at runtime. With carrier aggregation, cell selection chooses PCell first. PCell then determines SCells out of available candidates.

3 MISSED PERFORMANCE POTENTIALS

We use real-world instances to reveal missed potentials for user access to today's mobile networks, and analyze their root causes.



(a) One instance (4.6 Mbps vs. 49.3 Mbps)



(b) 40 run pairs

	$\Delta^{(50)}$	$\Gamma^{(50)}$
min	29.3	351%
25th	35.9	459%
med	42.2	600%
75th	47.3	837%
max	60.6	1506%

(c) Speed gains

	Main sets of serving cells	(%)
A	S19: {2425 (363) - none - none}	100%
B	S1: {5145 (16) - 850 (16) - none}	47.3%
	S3: {850 (16) - 5145 (16) - none}	39.2%
	S4: {2000(103) - 5145(16) - 2175(103)}	10.4%
	others	3.1%

(d) Serving cell sets in use. Each set is represented as {PCell - $SCell_1$ - $SCell_2$}.

Figure 3: An motivating example at one location L1 (AT&T).

3.1 An Motivating Example

We first use an example to illustrate two new findings:

F1. In today's mobile networks, the device is not always served by the cells that can offer the highest access speed (Figure 3a);

F2. Higher speed can be achieved when the device takes extra actions without changes to the current network infrastructure (Figure 3a).

Figure 3a plots the downlink throughput in two paired runs, where the phone keeps on downloading the same large file (500MB) at a fixed location using AT&T. The first run (A) is performed under the default network operations; It starts after the user arrives at the test location (L1 in Figure 10) along a fixed walking route. The second run (B) is performed at the same spot right after run A, while taking the device-side action (disabling band 5). In this case, the phone obtains 4.6 Mbps on average by default (A), while 49.3 Mbps is available (B). It is clear that the default network operations fail to select those serving cells that are used in run B and available in run A, thus miss higher speed (907% miss) (A).

We repeat the above experiment with more runs, and observe the same finding. Specifically, we repeat the paired runs at different hours of the day (from 9AM to 22 PM) and at different days of two weeks (in Dec 2019 and Feb 2020). Each run (B) with device customization (blocking band 5) immediately follows a default run (A). Figure 3b compares downlink throughput (0, 25, 50, 75 and 100 percentile) in 40 test pairs. We treat the gap in each pair to be the what-if speed gain if B's choice were selected to serve the device in reality. This is a reasonable estimate because the achievable potential is no smaller than the achieved one and we run what-if experiments almost simultaneously (under the same condition). We make two more observations.

F3. The significant gaps persistently occur; Poor performance in reality can be largely avoided through device-side action (Figure 3b, 3c);

F4. Persistent gaps imply that they are unlikely caused by transient factors, say, dynamic loads and time-varying radio signal quality. Instead, they are due to the poor cell selection in the current network operations (Figure 3d, §3.2).

In all our tests, device customization (blocking band 5) brings significant speed gains. By default, the phone constantly gets poor performance (medium speed < 8.5 Mbps, maximum <12.4Mbps). To quantify speed gain, we define two metrics,

$$\Delta_k^{(\rho)} = \psi_{k,B}^{(\rho)} - \psi_{k,A}^{(\rho)}, \quad \Gamma_k^{(\rho)} = (\psi_{k,B}^{(\rho)} - \psi_{k,A}^{(\rho)})/\psi_{k,A}^{(\rho)}. \qquad (1)$$

ρ is the percentile from 0 (min) to 100 (max). $\psi_{k,A}^{(\rho)}$ and $\psi_{k,B}^{(\rho)}$ are the ρ-percentile performance (here, throughput) in the default (A) and customized (B) run at the k-th pair. At this location, we observe that the median speed increases by at least 29.3 Mbps and up to 60.6 Mbps. In more than 50% tests, the median rate grows by at least 42.2 Mbps. In terms of $\Gamma^{(50)}$, the simple device customization scheme brings up to 15-fold gain with the minimal growth of 351%.

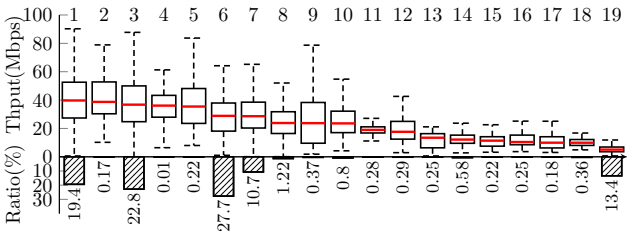


Figure 4: Performance and occurrence frequency of main serving cell sets at L1 (AT&T) in a 3-month reality check.

Data speed fluctuates over a small window of time (Figure 3a and 3b). Such fluctuations are partly induced by dynamic resource allocation under changing traffic load and partly caused by varying radio channels. These transient factors contribute to speed variance in each run, but not the significant gaps observed in all tests. Instead, they are primarily attributed to the quality of the serving cells.

We examine the set of serving cells in these 40 tests (Figure 3d). In the default runs (A), only one serving cell is used (the potential of CA is wasted). Each cell is represented by its channel number(cell ID) here, e.g. 2425(363). Channel 2425 is centered at 871.5 MHz on band 5, with 5MHz channel bandwidth. In the device-customized runs (B), this poorly-performed cell is filtered out by disabling band 5 at the device. Data speed rises with distinct serving cells. We see that CA takes effect and more serving cells are being used. It leads to three main choices and the first two use identical cells. The occurrence frequency is computed in terms of the observation duration (but not the number of runs), because more than one serving cell sets are used in some runs.

The total channel bandwidth does matter. All these new sets have larger aggregated bandwidth (25/25/20 MHz) than the default one (5MHz). We will confirm the impact of channel bandwidth later in §3.2. We would like to highlight that the speed variances in the default runs are relatively small. This implies that such poor performance is primarily determined by the selected serving cell(s), despite small variations due to transient factors.

F5. The above case is not rare. Significant potential miss is frequently observed among all four operators (§5.2).

We conduct a measurement study across a small city and several regions of three other cities in the US. We find that significant potential miss happens frequently in all four US carriers (§5.2). Missed potentials are caused by the poorly-performed cell selection.

3.2 Why Are Significant Potentials Missed?

We next examine why. It turns out that

F6. The legacy design for seamless radio access should be held accountable. Today's cell selection is largely designed for seamless *connectivity, but not for best connectivity and data performance, thus missing good candidates for serving cells (Figure 4, 5, 6).*

In fact, current selection practice favors the cells with stronger radio signal quality over those offering higher speed. Such practice is not without rationale. However, in today's mobile networks where dense cell deployment is the norm, it is likely to underutilize high speed potentials empowered by rich cell choices at most places.

We use the above example at L1 (AT&T) to illustrate its limitations. Same/similar conclusions hold at many other spots for all four US operators in our large-scale reality check (detailed in §5.2). At L1, we run an extensive study; Different from §3.1, we perform only the default runs without taking any extra device-side action. To observe as many candidate cells as possible, we take diverse walking or driving routes to reach L1. We run 192 tests sporadically from Dec 2019 to Feb 2020. Each run lasts for 2 to 10 minutes and the total time observed at L1 is about 10.5 hours. We see 57 cell sets out of 31 cells over 6 bands (2,4,5,12,30,66). Among them, 19 sets out of 21 cells are used for more than 60 seconds. Figure 4 plots their data performance and occurrence frequency, in descending order of median speed. We make two observations at L1.

First, abundant choices bring rich diversity in cell selection (F7, Figure 4). There is no single winner at a location. We see that five sets (1, 3, 6, 7 and 19) are frequently selected at L1. S1 (short for set 1), S3 and S19 refer to Figure 3d. S6 is {2425(363)-1150(413)-none} that enables CA compared to S19. S7 is {66486(103)-none-none} which uses a PCell on band 66 (a superset of band 4). Other set information is omitted due to space limit. Figure 5 plots radio quality of four common PCells at L1 in all the runs. RSRP/RSRQ (reference signal received power/quality) are two key measures of radio evaluation. We see that all four cells have comparable radio quality, stronger than most other cells at L1. This is why they are often selected.

Second, abundant choices cannot guarantee the selection quality (F8, Figure 4). The choice with poor data performance is regularly or even repeatedly selected in reality. At L1, S19 is frequently selected but the offered speed is 8× slower than the best (via S1). We observe that S19 is always selected when the device moves to L1 along one direction (§3.1). Figure 6 plots RSRPs of these three cells in such a walk instance towards L1.

We further use this instance to illustrate how current selection practice misses good choices in four aspects.

∘ *Cell selection is primarily radio-based and ignores channel bandwidth (F9, Figure 5, 6).* All steps of measurement, reporting and decision to perform a cell selection (Figure 2) count on radio quality evaluation. For instance, whether to measure candidate cells depends on how weak the current one is; Which cell to report depends on whether its radio signal quality is strong enough; Which cell to select depends on who has the strongest radio quality out of those reported ones; This is consistent with prior studies [16, 18, 25].

Such radio-based cell selection ignores non-radio factors which impact data performance, such as channel bandwidth. Cell 2425(363) has only 5 MHz and offers the lowest data speed at L1. 850(16) uses 20 MHz and offers higher speed. But 850(16) is not favored at locations where its radio signal is slightly weaker than 2425(363) (say, < 50m). Note that, along the route towards L1 (Figure 6), 2425(363) is stronger than both good choices of 850(16) and 5145(16)

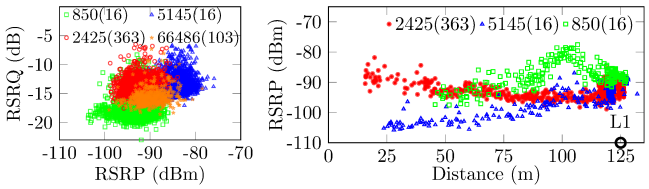


Figure 5: Four cells' RSRP/RSRQ at L1.

Figure 6: RSRP of three selected cells in one walking instance towards L1.

at < 50 m locations (more than 75 m away from L1). This is why 2425(363) often wins at these places. Selecting the cells regardless of their bandwidths results in high likelihood of selecting poorly-performed cells.

∘ *Radio-based cell selection discourages leaving the current cell despite poor performance (F10, Figure 6).* We find that no cell selection will be triggered as long as the current cell's radio signal quality is tolerable. In this walk instance, we see that cell 850(16) is even stronger than 2425(263) when we move closer to L1, say at [50m, 125m]. However, 850(16) is not considered in this instance and rarely considered in all the instances with cell 2425(263) as PCell. This is because 2425(363) has RSRP around −95 dBm and its RSRQ stays above −15 dB, which does not meet the criterion to measure other cells. Hence, cell 850(16) and other cells (e.g., cell 5145(16)) are missed despite their far better performance.

∘ *Current selection practice fails to make full power of CA (F11, Figure 4, 12).* CA expects to leverage SCells to increase channel bandwidth and data performance. However, its power is hampered. CA is feasible at L1 when 2425(363) is a PCell, as S6 {2425(363)-1150(413)-none} is commonly observed (Figure 12). However, in the walk instance, no CA is enabled. It loses the second chance (via SCells) to make up the missed potentials when radio-driven selection picks a poorly-performed PCell. We further find that CA does not explore all possible SCells. The 3GPP standard [12] requires that PCell and SCells must co-locate on the same cell tower, to implement CA within the same radio protocol stack. However, not all the cells on the same tower are open to CA. We check all the cell sets in the neighborhood of L1 and group the cells as long as they appear in one cell set. We see 6 cells on the tower of cell 850 (16) and all of them have acceptable signal strength. Theoretically, there are $C^2_{6-1} + C^1_{6-1} + C^0_{6-1} = 18$ CA options. However, we only observe 2 cell sets in reality: S3 {850(16) - 5145(16) - None} and S12 {850(16) - None - None}. It seems that the PCell - SCell combination is constrained out of a small number of pre-set choices. This is partly validated in our large-scale study (Figure 12). We gauge that it is constrained by cell deployment and CA managed by the operator.

∘ *Last but not least, current selection practice seeks for local optimum, not for global optimum (F12, Figure 6).* We find that its initial cell choice profoundly restricts the subsequent operations and consequences. It does not only discourage a cell selection, but also limits the scope of candidate cells considered for a selection. Take radio measurement as an example to illustrate this local constraint. There are intra-freq and inter-freq measurements [12]. The former measures the cells over the same channel and the latter measures more cells over distinct channels. What cells to measure depends on the current one and its radio quality. In the walk instance (Figure 6), cell 850(16) or cell 5145(16) is not measured because no

inter-measurement is invoked. We want to point out that the current serving cell is unaware of this biased choice at runtime when it decides the next target cell out of a small subset, not out of a whole set of all candidate cells. This way, there is no escape to better cells once the current cell gets stuck at a local stable choice. Multiple stable choices correspond to multiple winners at L1.

We would like to emphasize that we do not intend to blame the selection practice. It is indeed effective and efficient to ensure seamless radio coverage while retaining reasonable overhead (e.g., avoiding ping-pong effects). It switches to new cells only when the current ones are about to fail. At the early phase when full coverage was still a big concern, good radio was highly correlated to good performance. However, good radio ≠ good performance in today's mobile networks with abundant choices at place. We argue that it is right time to revisit cell selection and solve its underutilization problem that ends with unnecessarily poor performance.

4 ICELLSPEED DESIGN

We propose iCellSpeed to increase data speed by tackling the underutilization problem that stems from the existing cell selection scheme. iCellSpeed seeks to achieve higher speed gains by selecting better cells that better utilize available capabilities rather than through enhancing raw capabilities. The ideal and straightforward solution is to directly implement a clean-state selection scheme for performance optimality out of all choices (discussed in §8). However, it is not practical (at least in the near future) because of big changes to network infrastructure. We thus propose iCellSpeed, a device-side solution which exploits software power available at commodity smartphones and works in concert with the current cell selection practice. It is compatible with standard mechanisms and operational network infrastructure, with no need to change the network side. If successful, iCellSpeed promises to bring immediate benefits for the device and by the device.

4.1 Overview of iCellSpeed

Figure 7 depicts iCellSpeed's main components and operation flows. The center is to use performance profiling as the main instrument and enable proactive device-customization atop of the existing network-centric cell selection procedure (Figure 2). Profiling leverages historical measurements to accumulate global knowledge and thus offers the ability to challenge and correct the improper decision out of partial information at runtime. The device switches its role from a dumb, reactive terminal to an intelligent, proactive one. Instead of simply following the commands from the network and executing rigid reactions pre-implemented in the chipset, the device proactively overrides the default reactions to influence the consequence of cell selection. But the serving cell is still decided by the network eventually.

Two core enabling modules are iCustomize (2a) and iProfile (0 and 2c). There are two main operation flows.

1. Device-side customization at runtime. iCellSpeed makes one change at the device side, incorporating with the existing handoff procedure (① – ④ in Figure 2). It adds the iCustomize step to break a direct chain between measurement (②) and reporting (③) which are configured by the network and executed at the device. Instead of passively following the commands from the network,

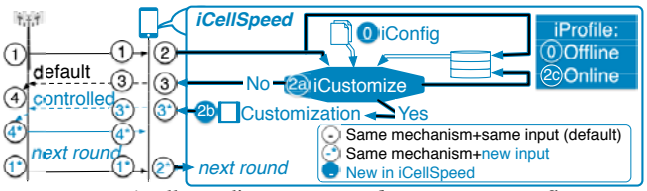


Figure 7: iCellSpeed's overview and main operation flows.

the device is empowered to question and challenge the default selection operations, because the device stays alert of the identified limitations in §3.2 and the resulted potential miss. Intuitively, iCustomize is to compare real-time performance and current choice with the profiles learned in advance; It then determines whether better cells are missed at runtime, and which action from device side is feasible to prompt better cells to be selected. If iCustomize confirms no need of further device actions, no change will be made and the default procedure proceeds (see the branch ②-2a-No-③-④). Otherwise, it executes device-side customization to override the default reaction and thus influence the selected cells (see the branch ②-2a-Yes-2b-③'-④'). Note that iCellSpeed does not alter the existing mechanism or network-side functions (say, ①, ③ and ④), which are also beyond control. Instead, it leverages just device power to change the *input* of these functions so as to indirectly affect the *consequence* of selected cells, towards a more desirable choice. iCellSpeed uses more intelligent device assistance to comply and complement the cell selection practice, while improving the chance of selecting cells with better performance.

2. Device-side profiling at runtime and offline. iProfile is to leverage measurements in the past and accumulate *global* knowledge regarding performance at every place so as to combat the limitations of local views used for cell selection at runtime. It gathers what cells are available, how they are selected, and how they perform in reality. Then it aggregates observed samples into the profiles regarding availability of choices (cell deployment), frequency of choices (cell selection), and performance of choices. It supports both offline and online modes. The former creates initial profiles through offline training or crowdsourced measurements. The latter updates the profiles with measurements over time.

4.2 iCustomize

It is not easy to decide a device-side action because its impact is not deterministic. As illustrated in Figure 7, the final decision is still made by the network. Take the example at L1 (Figure 3d) to illustrate control uncertainty. Blocking band 5 results in three possible consequences, not a single deterministic one. Fortunately, all three new choices are positive, offering much higher speed than the default one (which is the worst set observed at that location). However, it is not always true. At other places, multiple new possibilities can be positive and negative. The key of iCustomize is to predict both gains and risks (negative gains) and tame control uncertainty to determine the proper device-side action.

Specifically, iCustomize addresses three technical questions:

Q1. Should the device accept the default cell set?

Q2. If no, what action should be taken?

Q3. Should the device accept the previous action?

A naive idea. The straightforward solution is to predict the gain

associated with each possible action and then maximize the predicated gain to determine whether and how to take actions (Q1 and Q2). It can be formulated into a classic Expectation-Maximization (EM) problem [14]. Let us assume that the iProfile module has sufficient samples in the past and thus gathers reliable knowledge for a given location. Let $\mathcal{S}$ be all possible choices at the given location, i.e. $\mathcal{S} = \{S_u | u = 1, 2, \cdots, U\}$. X_{S_u} is a random variable for data performance of cell set S_u. The resulted performance upon action π is a weighted sum of multiple random variables as $X_\pi = \sum_{u=1}^{U} P(S_u|\pi) \cdot X_{S_u}$, where $P(S_u|\pi)$ is the likelihood of selecting cell set S_u upon action π. Consequently, the decision is made by maximizing performance expectation,

$$\pi_* = \arg\max_{\pi \in \Pi} E[X_\pi]. \quad (2)$$

No action is taken when $E[X_{\pi_*}]$ is no better than the default one. Otherwise, π_* is taken. Q3 is used to address control uncertainty in case the new action brings a negative gain. To answer it, it monitors performance under action π_*. If the resulted new cell set is worse than the default one at the new round, the previous action is withdrawn and excluded from the action set Π. At next round, a new action is derived out of the updated action set the Π to maximize Eq. (2). Otherwise, it stops.

Our practical solution. However, the storage overhead of the above computation is too huge. We thus work on a practical solution that *approximately* predicts the gain and looks for a *reasonably good* action (customization) so that we probably boost data speed in an efficient way. This solution tradeoff is feasible according to the famous PAC (probably approximately correct) learning theory [**?**]. In our context, we enable a fuzzy logic that incorporates non-deterministic profile models and deterministic domain knowledge. Note the consequence of our customization is approximately, not precisely accurate. Once the previous action is incorrect, we take just-in-time reinforcement learning and correct our previous "mistake" with up-to-date feedback.

Figure 8 shows the core logic of iCustomize. At the first level, it goes to NO-branch when no device customization is performed previously; Otherwise, it goes to the YES-branch at the next round to check whether the speed is worse; If no, no more action is needed. Otherwise, it excludes the previously-used actions and runs iCustomize* (the NO-branch).

iCustomize* uses a two-level decision subtree. To decide whether to challenge the default choice, it takes the following factors into account: real-time performance of the current cell set, historical performance of the current cell set, and potential gains of device-side actions. On the first level (speed-OK trigger), we examine if the current speed is satisfactory. We devise intuitive rules such as current performance no worse than a portion of the best (e.g., above 70% of the 50-percentile performance of the best set), or current performance above a certain percentile (e.g., the midpoint of the performance range. Note that the used information (e.g., the best/worst performance) can be easily learned by iProfile. We test with several rules in our evaluation (§5.3) and shows that these intuitive rules work robustly to differentiate good runs from runs with missed potentials. The reason is simple. At places with significant performance potential miss (see the example of S19 at L1 in Figure 4), there exists huge room to design speed-OK rules

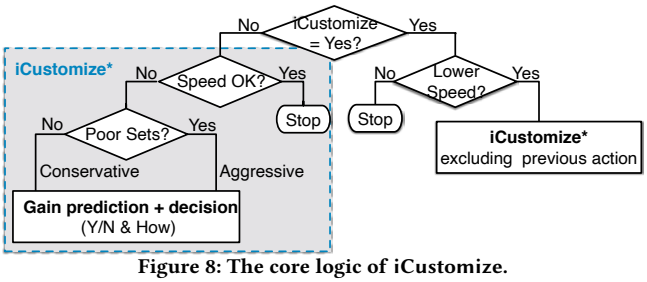


Figure 8: The core logic of iCustomize.

to differentiate good and poor performance; At places with small or overlapping performance gaps, low accuracy is tolerable since the likelihood to realize missed potential is low as well. As a result, intuitive rules suffice in iCellSpeed. On the next level, we check if the current cell set is poor based on iProfile (§ 4.3). It decides whether iCellSpeed takes aggressive or conservative actions.

We next predict the gain and make a decision. We exploit domain knowledge to approximate Eq. (2) and simplify gain prediction.

We find that device-side actions are very limited. The ideal customization is to allow the device to directly lock the target serving cell(s) which offers desirable performance. However, such explicit cell locking is not available at almost all commodity phones (the exceptions [8**?**] cannot work); Software interfaces such as API [5], AT commands [1] and secret codes [6] can not lock cells. Constrained by available software power, three actions are considered:

A1: lock one frequency band;

A2: block one or multiple bands;

A3: reset, particularly via turning off and on mobile data (or the flight mode);

A1 is one special case of A2, when blocking all the other possible bands, except the one to be locked. A1 and A2 will rule out all the cells over certain bands. A3 is to give equal chances to all candidate cells by clearing the impact of the current choice. We note that all these actions have one downside in practice. Because they change frequency bands/channels or reset radio access, these device-side actions require to disrupt radio resource control (RRC) and thus suspend data connection for a while. We evaluate the impact of the disruption time in §5.3. It is about two seconds at the application (APP) layer and several hundreds of milliseconds at RRC. Note that the disruption at RRC is mandatory but the extra disruption at the APP layer is avoidable with better mobile OS support. The disruption time is tolerable for elephant traffic flows which last long and require huge throughput. We thus consider bulk file downloading in this work and leave its extension to other applications as future work.

We exploit the above knowledge to estimate $P(S_u|\pi)$. For A1/A2, $P(S_u|\pi)$ is zero as long as one cell in set S_u is explicitly ruled out. We further estimate $P(S_u|\pi)$ in proportion to $P(S_u)$ only for those eligible choices allowed by action π. A winner in the default runs is likely a winner under action π , because of the same network selection function; In reality, they often have relatively stronger radio quality. For A3, it is to reset radio access and the likelihood is the same as $P(S_u)$.

Given $P(S_u|\pi)$, we look for the action that maximize $E[X_\pi]$. We start with blocking one band. We iteratively expand the actions to be considered. We stop when there is no more chance to find an action with gain larger than the current maximum. One action's gain is

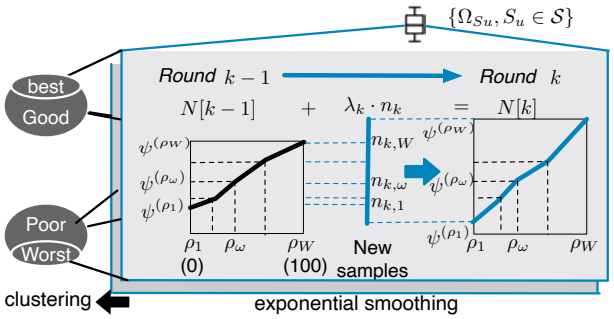


Figure 9: Profiles and online performance profiling.

estimated as the weighted sum of all eligible choices. We also notice that a big gain change may occur when one action is linked with a mixture of good and poor sets; So we continue only when the gain of the good choice is larger than the current maximum. Regarding performance, we use runtime measurement for the current choice and profiles for other choices. There is a slight difference when a good set is in use. Its runtime performance does not match its profile. We need to remove all high-rank actions which allow the current set and re-calculate the action with the maximal gain based on runtime performance measurement. If blocking any band(s) does not bring performance gain and current performance is way below the expectation, we can consider A3. Otherwise, no action is feasible. Its actual calculation is simple because we see that there are only 4-5 popular sets in most cases; Blocking one band is enough at most cases; We block at most 2 bands in this study.

4.3 iProfile

The iProfile module maintains and updates three types of profiles to support device-assisted cell selection.

(I) Availability of choices. This records working cell sets at each location. This provides global knowledge regarding cell deployment which remains stable for a long period (months or years). Its update is straightforward. Whenever a new serving cell set or cell is observed, it is inserted into its set of choices at the given location. This helps us to efficiently profile abundant choices in reality. Even given measurements holes at some places, we can infer these choices based on records in its close vicinity.

(II) Frequency of choices. This records the consequence of cell selection in reality. $P(S_u)$ is the frequency of choice S_u in the default cell selection. We record the count of instances over time. It is updated as follows. The sum of $P(S_u)$ remains invariant (here, $\sum = 1$) but its individual value changes accordingly as the count of the observed choice (say, S_v) increases. When one specific action π is taken, we take a similar way to update its occurrence frequency.

(III) Performance of choices. This accumulates global knowledge on performance of choices at this location. Rather than storing a huge amount of raw samples over time for X_{S_u}, we record its performance statistics over time, which is represented by multiple percentiles, $\Omega = [\psi^{(\rho_1)}, \psi^{(\rho_2)}, \cdots, \psi^{(\rho_W)}]$. W and ρ_ω ($\omega = 1, 2, \cdots, W$) are constants configured at the start. They are ascending percentiles such that (min) $0 = \rho_1 < \cdots < \rho_\omega < \rho_{\omega+1} < \cdots < \rho_W = 100$ (max). Offline profiling is simple. Given $N[0]$ samples for one serving cell set S_u at location L_i, we calculate its performance statistical vector.

Online profiling is illustrated in Figure 9. We develop a fast algorithm to update $\Omega[k]$, given the previous profile $\Omega[k-1]$ and n_k performance samples measured at round k. We update the total amount of effective samples as $N[k] = \lceil N[k-1] + \lambda_k \cdot n_k \rceil$. Here, λ_k is a tuning parameter to pace the rate of forgetting historical records. It is used to balance sample staleness and approximation accuracy. Without loss of generality, we assume that new samples are sorted in ascending order; Namely, $\phi_{k,1} \le \phi_{k,2} \le \ldots \le \phi_{k,n_k}$. We iteratively update $\psi^{(\rho_\omega)}[k]$ when ω increases from 1 to W. Clearly, when $\omega = 1, W$, we update the minimum and maximum as follows:

$$\psi^{(\rho_1)}[k] = \min(\psi^{(\rho_1)}[k-1], \phi_{k,1}), \quad (3)$$

$$\psi^{(\rho_W)}[k] = \max(\psi^{(\rho_W)}[k-1], \phi_{k,n_k}). \quad (4)$$

We then iteratively update $\psi^{(\rho_\omega)}[k]$, $2 \le \omega \le W-1$. The mathematical derivation looks sophisticated, and we present its simple heuristics first. There is no need to change Ω if new samples *perfectly* match with the existing performance profile. That is,

$$\phi_{k,n_{k,\omega}} \le \psi^{(\rho_\omega)}[k-1] \le \phi_{k,n_{k,\omega}+1}, 2 \le \omega \le W-1. \quad (5)$$

$n_{k,\omega} = \lceil n_k \cdot \rho_\omega / 100 \rceil$ is the position of ρ_ω-percentile sample in the new sample set. Otherwise, the profile should be updated. The displacement between $n_{k,\omega}$ and the sample position corresponding to $\psi^{(\rho_\omega)}[k-1]$ marks the potential update scope. The core idea is to approximate the old sample sets by assuming a uniform distribution between adjacent percentiles. We omit its mathematical derivation. The above update is iteratively performed with modest computation overhead. In turn, we will update $\psi^{(\rho_1)}[k] \rightarrow \psi^{(\rho_2)}[k]$ (based on $\psi^{(\rho_1)}[k]$ and $\psi^{(\rho_2)}[k-1]$) $\rightarrow, \cdots, \rightarrow \psi^{(\rho_\omega)}[k]$ (based on $\psi^{(\rho_{\omega-1})}[k]$ and $\psi^{(\rho_{x \le \omega})}[k-1]$), and so on. The above generic-form works for any W and $\{\rho_\omega\}$. In this work, we consider a common setting like $W = 5$, $\{\rho_\omega\} = \{0\%, 25\%, 50\%, 75\%, 100\%\}$. We validate that our performance profiles are accurate enough for iCellSpeed.

To facilitate the iCustomize module (e.g., configure the decision criteria), we aggregate per-set performance into per-location profile (illustrated by the outer box of Figure 9). We run a clustering algorithm to learn the top and bottom groups, referred to as *good* and *poor*. Note that the *best* and *worst* cell set belong to the *good* and *poor* groups, respectively. In case the performance profiles of the *best* and *worst* sets are similar, iCellSpeed is not needed because there is no significant performance missed. Otherwise, we are able to learn *good* and *poor* groups and use their performance profiles to determine thresholds used in iCustomize.

5 IMPLEMENTATION AND EVALUATION

We implement iCellSpeed on rooted Android smartphones. It is conceptually implementable on non-rooted phones but we currently prototype it on rooted phones for two reasons. First, blocking/locking a band is unavailable through the existing APIs of commodity Android OS releases (e.g., class `telephony` [5]). We use an encapsulated library over confidential secret codes. Root is not required if ROM is customized [4]. Second, we collect raw traces for debugging and evaluation. These traces include packet traces captured by tcpdump and mobile network signaling messages collected by MobileInsight [2, 20]. Both work with rooted phones only. The second feature is not necessary for normal operations of iCellSpeed. We implement several intuitive rules in the proof-of-concept prototype.

5.1 Methodology and Datasets

Our evaluation is primarily conducted in a small city, West Lafayette, IN (4 Km × 4.5 Km), marked as C1 in Figure 10. We also consider several locations and routes in three other cities – C2 (Los Angeles, CA), C3 (Austin, TX) and C4 (Lafayette, IN) – to validate its effectiveness and wide applicability in diverse real-world circumstances. We run two types of experiments without (A) and with (B) enabling iCellSpeed. We perform both static and driving tests. In static tests, we randomly choose 24 locations, including 17 locations in C1 in two representative zones: campus (aka, urban) and residence (suburban) and 2-3 locations each in other three cities. In driving tests, we use 10 fixed routes (Table 2) in four cities. These routes are popular in the test cities (e.g., routine routes between work and home) and cover representative types. U/S represents urban/suburban and L/P represents local (< 35mph) and parkway (45-55 mph).

We assess data performance in terms of downlink speed. We primarily use heavy traffic load which keeps downloading a large file (500MB) from our lab server. We monitor the server's outbound link rate and ensure that it is not a speed bottleneck in our mobile network experiments. We later evaluate iCellSpeed while running some applications like video steaming and conferencing. We measure all four US operators while AT&T is considered in all the cases, because the unlimited plan of AT&T only temporarily throttles rate if the network is busy after using more than 100GB. Sprint has the smallest dataset (10.8 hours, mainly on 2 routes in C1) due to its worst rate throttling. We purchase multiple lines to ensure that the rate is not throttled when we run downloading experiments. For each operator, one test phone is used at one time to avoid contention for radio access unless specified. We use eight phones out of three models: Google Pixel 3/2/2XL. They use Qualcomm Snapdragon 835/845/855 chipsets which all support 4.5G. The results are not phone-specific. Similar findings are observed in all four operators unless specified. To evaluate iCellSpeed, we first conduct extensive real-world measurements (A only) from Sep 2019 to Feb 2020, and then run A+B experiments together, primarily in Feb, March and June 2020. Our dataset D1 (Table 3) collects data speed samples for about 372 hours (static + driving) over 5,953 Km (driving).

To learn real-world cell deployment and characterize abundant choices available, we perform a wider-area driving experiment in C1. In addition to six routes, we do a city-scale scan to cover every road multiple times (main roads: ≥ 30, almost all local roads: ≥ 5) in C1. We use mice traffic (ping Google every second) to keep radio connectivity active at all time. We run experiments primarily from July to Dec 2019 and get dataset D2 over 549 hours and 5,113 Km.

Both datasets are public available at [7].

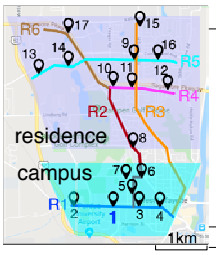


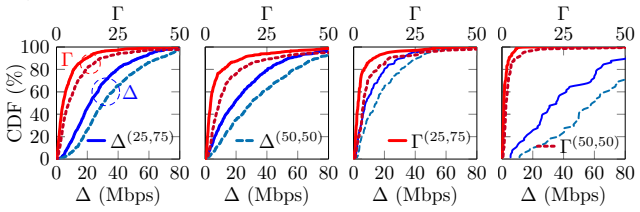
Figure 10: Map@C1.

	@	Distance	Type
R1	C1	2.9 Km	U, L
R2	C1	3.6 Km	U+S, L
R3	C1	5.8 Km	U+S, L
R4	C1	1.8 Km	S, P
R5	C1	4.2 Km	S, L
R6	C1	2.4 Km	S, P
R7	C2	2.0 Km	U, L
R8	C3	2.6 Km	U, L
R9	C4	3.2 Km	U, L
R10	C4	1.8 Km	U, P
Total		30.3 Km	

Table 2: Routes.

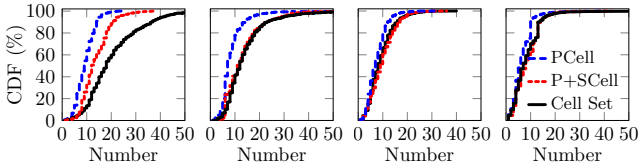
	Time	Distance
D1 (A+B, heavy load)		
A	247.9 hr	2,561 Km
V	68.6 hr	1,977 Km
T	45.1 hr	1,182 Km
S	10.8 hr	233 Km
D2 (A, light load)		
A	111.2 hr	1,221 Km
V	231.6 hr	1,493 Km
T	137.7 hr	1,259 Km
S	68.5 hr	1,140 Km
Total:	921.4 hr	11,066 Km

Table 3: Datasets.



(a) AT&T (b) Verizon (c) T-Mobile (d) Sprint

Figure 11: CDF of the observed speed gaps across C1.



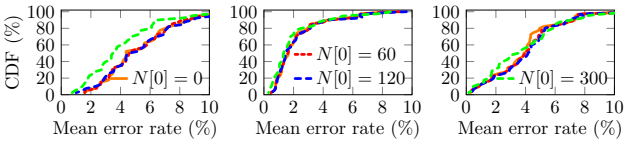
(a) AT&T (b) Verizon (c) T-Mobile (d) Sprint

Figure 12: CDF of the number of serving cells and sets in C1.

5.2 Reality Check Without iCellSpeed

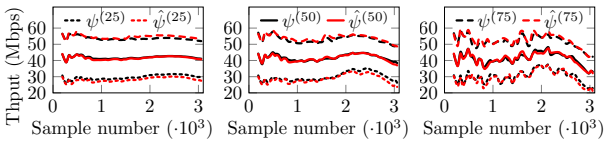
Missed potentials in reality. We first present our real-world measurements without enabling iCellSpeed in D1. This helps to better understand the test locations for iCellSpeed's evaluation. Our reality check shows that significant performance potential miss is frequently observed at many places for all four operators (Findings F1, F4 and F5). At all 24 selected locations (AT&T), we observe significant performance gaps as we see at L1 (Figure 4). The median speed gap is at least 10Mbps and up to 74 Mbps. Due to space limit, we present only the results of the driving tests in C1, which covers a wider area than the static tests. We divide the roads into small grids (each approximately 55m x 42m) and retrieve missed performance per grid. We use a pair of metrics $\Delta^{(\rho1,\rho2)}$ and $\Gamma^{(\rho1,\rho2)}$ to characterize the absolute and relative gaps between the $\rho1$-percentile performance of the *best* set and the $\rho2$-percentile performance of the *worst* set at each grid. We use two pairs: (1) $\rho1 = \rho2 = 50$, (2) $\rho1 = 25, \rho2 = 75$. Clearly, the latter is a more conservative approximation. Figure 11 plots their CDFs across all the grids with enough runs and samples. Note that insufficient runs can not capture real-world diversity on serving cells and insufficient samples can not capture data speed dynamics. Hence, we only consider grids with > 10 runs and samples > 100 seconds. There are 690, 438, 139 and 66 grids considered for A, V, T, S, respectively. We want to highlight that the actual gap at each grid can be still underestimated given our limited measurement scale. Even in a more conservative way, we see $\Delta^{(25,75)}$ > 20 Mbps at more than 55%, 48%, 20% and 74% for A,V, T, S, respectively. $\Gamma^{(25,75)}$ is at least 1 (speed doubled) at more than half of locations for all four operators. A and V have similar results on significant performance gaps. It is mainly due to their larger cell diversity than T and S. We notice that in Sprint, the absolute gap (Δ) is larger but the relative gap (Γ) is smaller. This is because the observed speed is much higher than other carriers, up to 160 Mbps; The lowest data speed is larger than 20 Mbps at almost all places, much faster than several Mbps or even hundreds of Kbps observed for other operators.

Cell deployment in reality. We use a city-scale measurement in C1 to show that dense deployment is the norm for all four operators (Finding F7). We see similar or even denser deployment in other cities, particularly in C2 which is one of top-3 US cities. We combine our driving tests in D1 and D2 and plot the CDF of the number of



(a) 25th-percentile (b) 50th-percentile (c) 75th-percentile

Figure 13: CDF of average performance profiling error rate over all popular cell sets at all the static locations.



(a) $\lambda = 1$ (b) $\lambda = 1.1$ (c) $\lambda = 1.5$

Figure 14: Profiling accuracy in one example (Set 1 at L1)

serving cells and sets observed across the measured grids in Figure 12. We exclude grids with insufficient runs and samples in the same way. We see more than 9 PCells (12 P+SCells) and 18 cell sets at more than 50% grids in AT&T. T and S have more than 5 PCells (7 P+SCells) at more than 50% places. We also performed limited test at several locations in rural areas and find that it is much less dense for all four operators. We see that the performance gap is not significant without abundant choices. As a result, we believe that abundant cell choices are available in the cities.

We also check all the cells used in C1 by four operators. We see that AT&T deploys 501 unique cells over 19 channels at 6 bands (2, 4, 5, 12, 30, 66). But the use of these channels is quite uneven. The most popular channels (> 10%,) are 850, 2000, 5145, 2175 and 2425 in descending order. Only three channel bandwidths of 5, 10, 20 MHz are used. We admit that such device-side measurement may be still incomplete despite our extensive study. We observe similar results for other three operators and omit them.

5.3 Micro-Benchmark Evaluation

We first use the test results at the static locations to evaluate how iCellSpeed's main components work in reality.

iProfile. We compare performance profiling accuracy with different parameters. In particular, we test with different numbers of initial samples, $N[0] = 0, 60, 120, 300$, and three smoothing weights $\lambda = 1, 1.1, 1.5$. iProfile takes every experiment run by a chronological order. Meanwhile, the ground truth is calculated out of all the samples at this moment. Figure 13 shows the average error rate for all popular cell sets (sample number > 600) at all static locations when $N[0]$ varies from 0 to 300. We show 25, 50, 75-th performance percentiles of cell set, and omit 0 (min) and 100 (max)-th percentiles because they are updated accurately. Figure 14 uses one instance to illustrate the impact of smoothing weight over time (samples). We show performance profile of cell set 1 {5145(16) - 850(16)} at L1 with $\lambda = 1, 1.1, 1.5$ and $N[0] = 60$.

We have three observations. First, iProfile achieves accurate estimation, by storing several performance percentiles, not a huge number of raw samples. Second, the accuracy results are very similar given various sizes of initial samples (even $N[0] = 0$). The low-percentile (25) is more sensitive to the initial sample size because fewer performance samples are considered for a low percentile. Third, the estimated profiles are quite accurate regardless of λ. Note that the smoothing weight of 1.5 induces more fluctuation in Figure 14, which is consistent with our observations at other locations. This is because higher weight accelerates the pace of forgetting historical data. Ideally, the weight should be tuned according to the elapsed time since last update. Considering most of our evaluation experiments are heavily conducted within one month (every few days), we set $\lambda = 1.5^{\lfloor week \rfloor}$ in our implementation. In the following evaluation, the default parameters are $\lambda = 1$ (1.5^0), $N[0] = 60$.

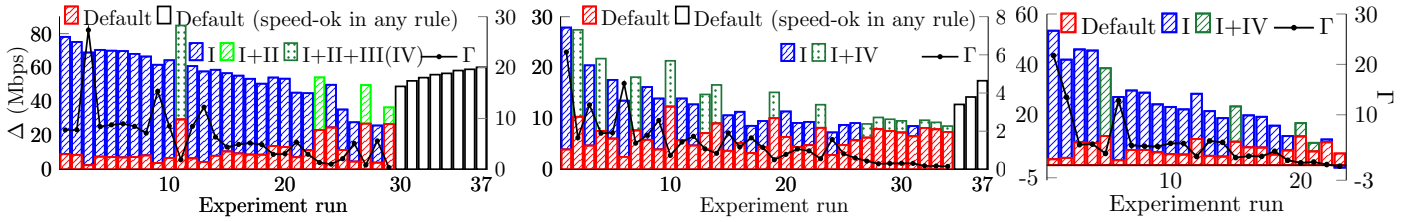
iCustomize. We evaluate the impact of the rules of "speed-not-OK", which determines whether iCellSpeed needs further actions. We define 4 rules accordingly:

Rule I: $\psi^{(75)}_{current} < 0.8 \cdot \psi^{(25)}_{best}$ or $\psi^{(25)}_{best} - 10$;
Rule II: $\psi^{(75)}_{current} < \psi^{(25)}_{best}$;
Rule III: $\psi^{(50)}_{current} < 0.7 \cdot \psi^{(50)}_{best}$ or $\psi^{(50)}_{best} - 15$;
Rule IV: $\psi^{(50)}_{current} < 0.9 \cdot \psi^{(50)}_{best}$ or $\psi^{(50)}_{best} - 5$.

$\psi^{(p)}_{current}$ and $\psi^{(p)}_{best}$ are ρ-th percentile of performance in the current run and in the existing profile. We run the experiment as follows. Start file downloading on the device and manually disable iCellSpeed for the first $2-3$ minutes; That is, iCustomize is running (monitor performance for decision making), but *none* of its decisions is made. Afterwards, as long as *any* rule above is satisfied, the device takes action correspondingly. We use the collected traces to evaluate the rule impact.

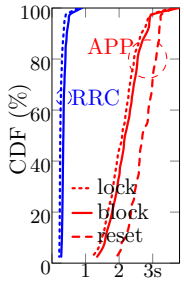
Figure 15 presents three showcases at L17 (C1), L15 (C1) and L18 (C2). We use average speed gain to evaluate the impact of the above four rules. In those plots, red bars represents default performance before the device action is taken, the stacked bars above red ones represent performance gains achieved by iCellSpeed with different rules. There are also default runs with pretty high data speed (none of rules are satisfied), referring to black bars. We have two main findings at L17. First, all the rules work well. They detect all runs with big improvement room and do not bother with good runs (the last 8 runs). iCellSpeed exploits great potentials at L17 with speed gains of $20-70$ Mbps, up to 27-fold. Second, these rules work slightly different because Rules I to IV become more aggressive. There is small difference in not-OK runs detected by four rules at L17. Because the default performance is too poor in most runs so even the most conservative rule (Rule I) is met. Rules I - IV miss potential gains in 4 runs (11, 23, 27, 29), 1 run (11) and 0 runs. Note the missed gains are relatively small and the only exception is at run 11, where the default speed is not too bad (~ 30Mbps) but the gain empowered by iCellSpeed is extraordinarily large. Generally, the speed-not-OK rule is a tuning knob to balance speed gains and the missed rate. The more aggressive rule brings smaller miss rate, but more likely with smaller gain.

We further apply Rules I and IV to other two locations. At L15, iCustomize improves poor situations by 8 Mbps and 7 Mbps on average for I and IV, given a limited achievable gain (median of the best cell set is 17 Mbps); At L18, iCustomize decides to take action in all 23 runs. It makes good catch in 19 runs with more than 1× enhancement. Dynamics in performance provided by the target cell set accounts for small or negative gain in other runs, especially like run 23. This implies that more action rounds should be considered.



(a) At L17 (C1) (b) At L15 (C1) (c) At L18 (C2)

Figure 15: Showcases of speed gains under different speed-not-ok rules.



(a) CDF

	RRC (ms)
Lock	248
Block	340
Reset	-
	App
Lock	2092
Block	2176
Reset	2694

(b) median

Figure 16: Disruption time.

Case 1: Idle	Usage
Phone idle	12%
Google Play	6%
Android System	6%
Mobile Net.	5%
Screen	2%
System UI	1%
iCellSpeed	**< 1%**
Case 2: Active	
YouTube	45%
iCellSpeed	**5%**

Table 4: Battery usage.

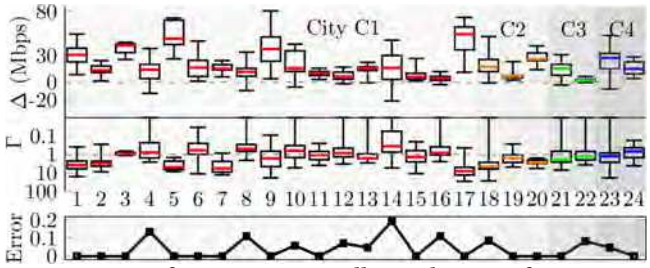


Figure 17: Performance gains at all static locations for AT&T.

Disruption. Due to implementation constraints, iCellSpeed has to disrupt ongoing traffic while taking actions to influence the default cell selection. We run experiments to measure the disruption time at both application (APP) and radio resource control (RRC) layers. We evaluate several actions of iCellSpeed: blocking one or two frequency bands, locking one specific band, and resetting mobile networks. In our test phones, blocking any band automatically blocks band 66. Figure 16 shows blocking or locking band have similar disruption time at APP layer, which is slightly less than resetting mobile network (2,694 ms). Locking disrupts RRC and APP for 248 ms and 2092 ms (median); Blocking has a little larger disruption time: 340 ms (RRC) and 2176 ms (APP). This matches with expectation because locking limits the spectrum bands to scan and has lower overhead. Resetting mobile network results in the longest disruption because it has to restart Radio Interface Layer (RIL) daemon [**?**]; We cannot measure its RRC disruption because it powers off the radio directly without releasing RRC connection. We would like to emphasize that the disruption at RRC is unavoidable as a solution compatible to the existing mechanism and infrastructure; The disruption from RRC to APP can be reduced with advances on mobile phone OSes and chipsets.

5.4 Data Speed Gains by iCellSpeed

Static tests (AT&T). We use two metrics to evaluate iCellSpeed.

Figure 17 presents its speed gains at all test locations in AT&T. When an device action is taken in one run k, iCellSpeed's gain is calculated as Δ_k or Γ_k, for the absolute or relative gap between the average throughput in the default and customized runs. Note that the gain can be negative if the customized run is worse. We show the speed after the first round, without showing the final result after multiple rounds because iCellSpeed is eventually no worse than the default one. We clearly see that iCellSpeed significantly boosts data speed at many locations. It boosts data speed at *all* locations in terms of the 25-th percentile absolute gain; The 50-th percentile (median) gain is larger than 10 Mbps at 19 out of 24 locations (79.2%). Note that our test locations are *randomly* selected and the achieved gain is bounded by the missed performance potentials which vary across locations (Figure 11a). This is why the absolute gain is not significant at all the locations. We see that the 75-th percentile gain is smaller than 10 Mbps at 4 out of 24 locations (here, 15, 16 in C1, 19 in C2 and 22 in C3). Hence, we use the relative gain Γ to deal with variance in the bound of the best achievable performance. Actually, at 15 out of 24 locations (62.5%), the relative gain is larger than 100% in more than half of runs, up to 28.4× (at location L17).

The second metric is the error rate at the first round, i.e. the number of runs with negative gains over all device-customized runs. 14 out of 24 locations have zero error rate. The error rate is below 0.1 at 6 out of the rest 10 locations. This indicates that iCellSpeed reliably tames uncertainty and balances gains and risks well. iCellSpeed corrects the mistake finally while its intermediate performance may get hurt.

Other operators. We evaluate iCellSpeed for Verizon and T-Mobile. Rate throttling makes it hard for us to evaluate Sprint. We see similar results with significant gains in Verizon (plots omitted). iCellSpeed increases data speed by more than 30 Mbps at 57% of test locations (up to 91Mbps). It at least doubles speed at 87% of test locations (up to 27.3x). We do not often observe expected gains for T-Mobile. We observe that resetting mobile networks sometimes boosts data speeds but its impact is random, depending on which cell set being selected. We find that most missing performance in T-Mobile is from the use or combined use of band 66. However, our current implementation automatically disables band 66 when blocking any bad band. It is the implementation constraint that prevents us from increasing data speed through iCellSpeed. Gains are possible once this constraint is released.

Driving tests. Figure 18 demonstrate iCellSpeed's speed gains using three examples on driving routes R1, R3 and R6. We use the left plot (R1) to illustrate how iCellSpeed works. We drive the same route without (default) and with iCellSpeed. In the default run, the

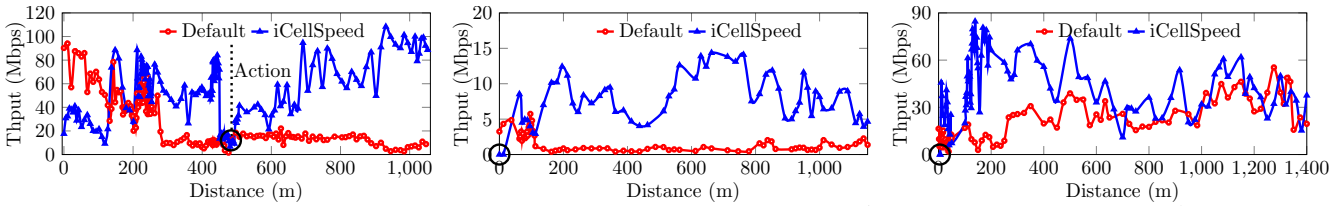
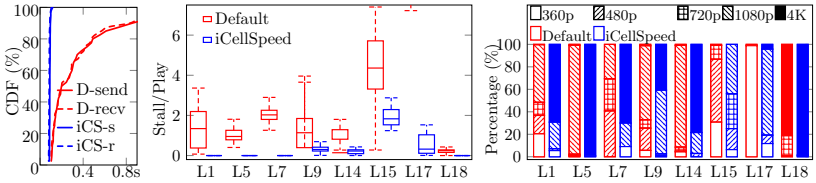


Figure 18: Three examples of iCellSpeed's speed gains over distinct routes (R1, R3 and R6, from left to right).

	VBR/ Main Res.	CBR Stall/Play 1K	CBR Stall/Play 4K	Zoom Delay (ms) Send	Zoom Delay (ms) Recv	Download Time (s) 25MB	Download Time (s) 100MB
D	360p	4.5	41.7	210	201	55.1	329.6
iCS	1080p	0	0.84	91	85	17.7	82.4
D/iCS	-	∞	**49.6**	**2.3**	**2.4**	**3.1**	**4.0**



(a) Multiple applications at L17 (D: default, iCS: iCellSpeed) (b) Zoom at L17 (c) 4K CBR video streaming (d) VBR video streaming

Figure 19: iCellSpeed's performance gains for test applications at multiple locations (AT&T)

Route	R1	R2	R3	R4	R5	R6
Gain (γ)	275%	115%	181%	97%	135%	119%

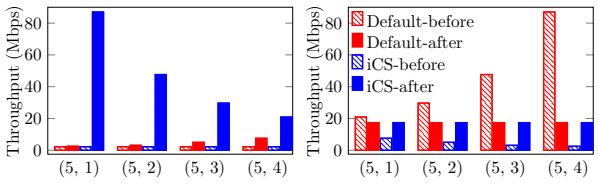
Table 5: Speed gains by iCellSpeed on 6 driving routes in C1.

device suffers poor performance no more than 15 Mbps from the point of 270 m. It is connected to cell set {2425(363)-None-None} (270-464 m) and {66911(421)-None-None} (464-1047 m). For a run with iCellSpeed, it detects poor performance with {2425(363)-None-None} in the area of 450-480 m. Then iCellSpeed takes the action of blocking band 5 and 66 and thus moves to a good cell set. The average performance grows from 13 Mbps to 64 Mbps.

We define $\gamma = (\bar{\psi}_{\text{iCellSpeed}} - \bar{\psi}_{\text{default}})/\bar{\psi}_{\text{default}}$ to quantify the speed gain in a driving test, where $\bar{\psi}_{\star}$ is the average speed by $\star$ (default or iCellSpeed) over the same route segment after iCellSpeed is triggered (including 0Mbps during 2-second disruption). Note that we use the average speed over the same segment (distance), not over the same duration because the driving time changes at each run. We see 352% and 120% gains in other two examples (R3 and R6). Table 5 shows the average gain observed on six routes in C1 where iCellSpeed is in use.

Other applications. We test iCellSpeed with three popular applications with elephant flows: DASH video streaming, video conferencing and file downloading. Note that iCellSpeed is not applicable to mice flows only because it is not triggered in a short flow lifespan. Figure 19 plots the results at 8 representative locations (with various data speeds and iCellSpeed gains) including 7 locations in C1 and 1 location (L18) in C2 using AT&T. For video streaming, we test with a demo video at bitmovin [?] with constant bitrate encoding (CBR) and variable bitrate encoding (VBR). For video conferencing, we use Zoom to set up a call between a mobile phone and another device via WiFi with abundant bandwidth. For file downloading, we consider two sizes: 25MB and 100MB. In this test, we see the worst data performance at L17 and thus choose L17 as an example to demonstrate that iCellSpeed has greatly boosted performance for all test applications in Figure 19a. Specifically, iCellSpeed helps VBR streaming to increase it dominant bitrate from 360p to 1080p (1K). It drops the stall/play ratio for CBR video streaming, from 4.5 to 0 for a 1K video (from 41.7 to 0.84 for a 4K video), making at least 1K video steaming affordable at L17; In Zoom video conferencing, it reduces the send/uplink latency by 2.3x (downlink: 2.4x), from 210ms (201ms) to 91ms (85ms). It accelerates file downloading by 3.1x (25MB) and 4x (100MB). Due to space limit, we only present CBR and VBR streaming results at test locations (Figure 19c and 19d). We see that iCellSpeed enables 4K video streaming at 5 locations (L1, L5, L7, L14, L18); All these gains are attributed to increased speeds by iCellSpeed. We notice that 4K CBR is not an acceptable viewing option in these default runs, which all suffer from extremely high stall/play rate except at L18. We choose it as a stress test to see how badly it can be without iCellSpeed. Similar enhancements are observed in the VBR tests, with more acceptable viewing experience by default, thanks to lower bitrates in use.

Multiple devices. We next evaluate how iCellSpeed performs in a multi-device scenario. We use (n, m) test phones where n is the total number of co-located phones and m is the number of phones running iCellSpeed ($m = 1, \cdots, n-1$). We consider in two scenarios: (a) all the phones are initially served by poor cells; (b) only those running iCellSpeed are initially served by poor cells and others are served by good cells (which actually run iCellSpeed to move to good cells first and then disable iCellSpeed). The first setting is to evaluate how iCellSpeed's effectiveness scales up, and the second is to evaluate its impact on those devices without iCellSpeed. Figure 20 plots the results before and after iCellSpeed takes effects at L5 (AT&T) in both settings. The results at other locations are similar and omitted. We test with n up to 5 and present the results when $n = 5$, as similar results are observed at $n = 2, 3, 4$. We have three observations. First, we clearly see that iCellSpeed boosts performance of those iCellSpeed-enabled phones previously served by the poor cells in both scenarios. Second, the gain declines as m grows when all the phones are previously served by the poor cells. The data speed grows from 2.2Mbps to 87Mbps (39.5x), 47.7Mbps (21.7x), 29.7 Mbps (13.5x) and 20.9Mbps (9.5x) when m grows from 1 to 4. Third, this results in the dropped speed for those phones which are previously served by the good cells. In Figure 20b, the phone without iCellSpeed gets its speed from 87 Mbps to 17.4Mbps when four phones switch to the good cells. This is expected because iCellSpeed is to exploit under-utilized capabilities, but not to increase raw capabilities. Once more devices move to the good cell(s), they compete resources and thus the obtained share drops. We observe significant enhancements in our real-world tests. This implies that the network capabilities are often under-utilized in



(a) n phones are poor (b) m phones are poor

Figure 20: iCellSpeed in (n, m) multi-device tests at L5 (AT&T).

presence of a number of real user phones beyond our control.

Energy overhead. We further assess iCellSpeed's energy consumption in two scenarios where the device is idle (with iCellSpeed being in the background but not triggered) or active with heavy traffic (here, YouTube, with iCellSpeed being triggered). We test it with a Pixel 2 device (2700 mAh) and Table 4 shows the results. In the idle setting, we see that extra energy consumed by iCellSpeed is negligible, compared to built-in services and components running in the background. In the active setting, we keep playing an YouTube video and find that iCellSpeed consumes 5% extra energy when YouTube uses 45% energy. The energy overhead is mainly used to monitor GPS, cellular signaling messages and system level throughput, which can be cut with energy-efficient monitoring.

6 RELATED WORK

There is no prior work to boost data performance through device-assisted cell selection. We briefly introduce most relevant work on cell selection/handoff in mobile networks and device-only performance improvement. Our work is inspired by recent handoff studies [15, 16, 18, 25]. [18] examines how handoff is performed and whether a handoff decision converges; [16, 25] investigate how handoff decisions impact performance and show that a handoff may result in worse performance; [15] is our preliminary study and unveils the problem of missed performance, by measuring the gap between the worst and the best choices. Our work differs from all the above studies, and targets to solve the problem. Device-centric performance solutions are divided into three categories: higher-layer APP/TCP optimization (e.g., [17, 21]), cross-layer optimization that leverages cellular-specific lower-layer information to enhance higher-layer functions (e.g., [13, 22, 24]), and lower-layer optimization (e.g., [19]). Our work also does lower-layer optimization. [19] enhances multi-operator access in Google Fi and we improve dominant radio access within a single operator. Given distinct causes for poor data performance, the proposed solution is different.

7 DISCUSSION

We discuss other possibilities and remaining issues.

Are larger gains possible? Absolutely yes. The gain would be larger when more device power is available, for example, when the device can directly lock the desired cells instead of blocking some bands to indirectly influence cell selection. The gain would be larger if the device supports swift band switching, without the 2-second disruption which is constrained by current practice of device chipsets and OSes.

Should we do it on-device only? Absolutely no. Instead, we argue that network is a better place to solve this problem. The gain is much larger or even reaches its full potential when the changes are allowed at the network side. For instance, it performs a global performance-driven optimization, which is not myopia constrained by partial observations at runtime. In this work, our aim is to offer a working solution even with modest gains and demonstrate feasibility to increase speed missed by current cell selection practice.

Does it hurt other devices and/or network? Maybe. The operators may intend to sacrifice user experience with rational, e.g, load balancing for network-side optimization, or throttling data for those without premium plans. We do not argue that they have to select cells that offer the best performance to mobile users. Our goal is to pursue better data performance which is sometimes unnecessarily missed in reality. We believe that boosting performance with no need of changing physical infrastructure is aligned with the interests of both operators and users in some real-wold circumstances. If they do not match, the network always holds the right and final power to decide what to serve the device. We notice that iCellSpeed may benefit the device but at the cost of performance degradation of other devices (Figure 20b). This is because the default selection is unfair and should take the blame at the first place. In theory, iCellSpeed may oscillate when a large number of iCellSpeed-enabled devices are synced to intervene cell selection and impact each other. It is not observed in practice as iCellSpeed removes this cell choice once it underperforms. In the worst case, it conservatively goes back to the default selection.

Will the problem disappear in 5G? It will not go away as 5G proceeds. The identified issue of missed performance potentials conceptually exists in 5G that still takes radio signal quality, not the resulted performance into account [10]. This issue is likely even worse in 5G with much denser deployment, more spectrum choices, and bigger performance gaps contributed by advanced technologies (e.g., 10Gbps vs tens of Mbps). It was reported that the device failed to get 5G where 5G was available in an early 5G measurement [3].

8 CONCLUSION

In this work, we present the design, implementation and evaluation of iCellSpeed. Our effort is motivated by the premise that today's network-centric cell selection may result in a nonnegligible, sub-optimal choice of cells for a given device. Consequently, the user device suffers from large access speed dip from the highest available one in practice, while the network suffers from significantly underutilizing the available resources. Our extensive measurements have confirmed both. The root cause lies in today's network-centric cell selection scheme where the device has minimal influence on decision making. While this might work on the dumb terminals in the past telecom age, it does not work well with the increasingly capable and smart devices in the Internet and AI age.

iCellSpeed thus explores a new paradigm of "device-assisted, infrastructure-decided" design for 4.5G and beyond. As a result, it is a win-win game for both the device and the infrastructure. The infrastructure better utilizes its current resources, while the device gains its desirable, higher access speed. iCellSpeed thus improves device performance without upgrading the infrastructure, but via smart decision inputs from users and better utilization for networks.

Acknowledgments. We appreciate our shepherd and reviewers for their constructive comments. This work was partially supported by NSF grants: CNS-1749049, CNS-1750953 and CNS-2027650.

REFERENCES

[1] 2016. AT Command User Guide. http://gamma.spb.ru/images/pdf/L506_AT_Command_User_Guide_V2.1.pdf. Access: 01/24/2020.

[2] 2019. MobileInsight. http://www.mobileinsight.net,2019.

[3] 2020. 5G Speed Report: Early 5G Experience Provides Mixed Results. https://www.telecompetitor.com/5g-speed-report-early-5g-experience-provides-mixed-results/. Access: 03/01/2020.

[4] 2020. Android Open Source Project. https://source.android.com/. Access: 01/19/2020.

[5] 2020. Android.Telephony. http://developer.android.com/reference/android/telephony/package-summary.html. Access: 01/20/2020.

[6] 2020. Codes Google Pixel. https://www.hardreset.info/devices/google/google-pixel/codes/. Access: 01/19/2020.

[7] 2020. iCellSpeed Datasets. https://github.com/mssn/iCellSpeed-Dataset.

[8] 2020. Network Signal Guru. https://play.google.com/store/apps/details?id=com.qtrun.QuickTest. Access: 01/21/2020.

[9] 3GPP. 2013. Carrier Aggregation explained. https://www.3gpp.org/technologies/keywords-acronyms/101-carrier-aggregation-explained.

[10] 3GPP. 2019. TS23.502: Procedures for the 5G System (5GS). V15.8.0 (Release 15).

[11] 3GPP. 2019. TS36.101: E-UTRA; User Equipment (UE) radio transmission and reception. V15.6.0 (Release 15).

[12] 3GPP. 2020. TS36.331: E-UTRA; Radio Resource Control (RRC). V15.9.0 (Release 15).

[13] Arjun Balasingam, Manu Bansal, Rakesh Misra, Kanthi Nagaraj, Rahul Tandra, Sachin Katti, and Aaron Schulman. 2019. Detecting if LTE is the Bottleneck with BurstTracker. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom'19)*.

[14] Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)* 39, 1 (1977), 1–22.

[15] Haotian Deng, Kai Ling, Junpeng Guo, and Chunyi Peng. 2020. Unveiling the Missed 4.5G Performance In the Wild. In *Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications* (Austin, TX, USA) *(HotMobile'20)*.

[16] Haotian Deng, Chunyi Peng, Ans Fida, Jiayi Meng, and Charlie Hu. 2018. Mobility Support in Cellular Networks: A Measurement Study on Its Configurations and Implications. In *ACM Internet Measurement Conference* (Boston, MA, USA) *(IMC'18)*.

[17] Prateesh Goyal, Anup Agarwal, Ravi Netravali, Mohammad Alizadeh, and Hari Balakrishnan. 2020. ABC: A Simple Explicit Congestion Control Protocol for Wireless Networks. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI'20)*.

[18] Yuanjie Li, Haotian Deng, Jiayao Li, Chunyi Peng, and Songwu Lu. 2016. Instability in Distributed Mobility Management: Revisiting Configuration Management in 3G/4G Mobile Networks. In *ACM International Conference on Measurement and Modeling of Computer Science* (Antibes Juan-Les-Pins, France) *(SIGMETRICS'16)*.

[19] Yuanjie Li, Haotian Deng, Chunyi Peng, Guan-Hua Tu, Jiayao Li, Zengwen Yuan, and Songwu Lu. 2016. iCellular: Define Your Own Cellular Network Access on Commodity Smartphones. In *USENIX Symposium on Networked Systems Design and Implementation* (Santa Clara, CA, USA) *(NSDI'16)*. 643–656.

[20] Yuanjie Li, Chunyi Peng, Zengwen Yuan, Jiayao Li, Haotian Deng, and Tao Wang. 2016. MobileInsight: Extracting and Analyzing Cellular Network Information on Smartphones. In *ACM International Conference on Mobile Computing and Networking (MobiCom'16)*.

[21] Ashkan Nikravesh, Yihua Guo, Xiao Zhu, Feng Qian, and Z Morley Mao. 2019. MP-H2: A Client-only Multipath Solution for HTTP/2. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom'19)*.

[22] Zhaowei Tan, Yuanjie Li, Qianru Li, Zhehui Zhang, Zhehan Li, and Songwu Lu. 2018. Supporting mobile VR in LTE networks: How close are we? *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 2, 1 (2018), 1–31.

[23] Wikipedia. 2017. LTE Frequency bands. https://en.wikipedia.org/wiki/LTE_frequency_bands.

[24] Xiufeng Xie, Xinyu Zhang, and Shilin Zhu. 2017. Accelerating Mobile Web Loading Using Cellular Link Information. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys'17)*.

[25] Shichang Xu, Ashkan Nikravesh, and Z Morley Mao. 2019. Leveraging Context-Triggered Measurements to Characterize LTE Handover Performance. In *International Conference on Passive and Active Network Measurement (PAM)*. 3–17.