iCellular: Device-Customized Cellular Network Access on Commodity Smartphones

Yuanjie Li, ***University of California, Los Angeles;*** **Haotian Deng and Chunyi Peng,** ***The Ohio State University;*** **Zengwen Yuan, Guan-Hua Tu, Jiayao Li, and Songwu Lu,** ***University of California, Los Angeles***

https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/li-yuanjie

This paper is included in the Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI '16).

March 16–18, 2016 • Santa Clara, CA, USA

ISBN 978-1-931971-29-4

Open access to the Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI '16) is sponsored by USENIX.

iCellular: Device-Customized Cellular Network Access on Commodity Smartphones

Yuanjie Li[1], Haotian Deng[2], Chunyi Peng[2], Zengwen Yuan[1], Guan-Hua Tu[1], Jiayao Li[1], Songwu Lu[1]

[1] University of California, Los Angeles [2] The Ohio State University

Abstract

Exploiting multi-carrier access offers a promising direction to boost access quality in mobile networks. However, our experiments show that, the current practice does not achieve the full potential of this approach because it has not utilized fine-grained, cellular-specific domain knowledge. In this work, we propose *iCellular*, which exploits low-level cellular information at the device to improve multi-carrier access. Specifically, *iCellular* is proactive and adaptive in its multi-carrier selection by leveraging existing end-device mechanisms and standards-complaint procedures. It performs adaptive monitoring to ensure responsive selection and minimal service disruption, and enhances carrier selection with online learning and runtime decision fault prevention. It is readily deployable on smartphones without infrastructure/hardware modifications. We implement *iCellular* on commodity phones and harness the efforts of *Project Fi* to assess multi-carrier access over two US carriers: T-Mobile and Sprint. Our evaluation shows that, *iCellular* boosts the devices with up to 3.74x throughput improvement, 6.9x suspension reduction, and 1.9x latency decrement over the state-of-the-art selection scheme, with moderate CPU, memory and energy overheads.

1 Introduction

Mobile Internet access has become an essential part of our daily life with our smartphones. From the user's perspective, (s)he demands for high-quality, anytime, and anywhere network access. From the infrastructure's standpoint, carriers are migrating towards faster technologies (*e.g.*, from 3G to 4G LTE), while boosting network capacity through dense deployment and efficient spectrum utilization. Despite such continuous efforts, no single carrier can ensure complete coverage or highest access quality at any place and anytime.

In addition to infrastructure upgrades from carriers, a promising alternative is to leverage multiple carrier networks at the end device. In reality, most regions are covered by several carriers (say, Verizon, T-Mobile, Sprint, and AT&T in the US). With multi-carrier access, the device may select the best carrier over time and improve its overall access quality. The exciting Google *Project Fi* [26] has taken the lead to provide 3G/4G multi-carrier access in practice. Other similar efforts through universal SIM card include Apple SIM [14] and Samsung e-SIM [24]. The upcoming 5G standards also seek to support multiple, heterogenous access technologies [34].

Our empirical study shows that, the full benefits of multi-carrier access can be constrained by today's design. We examine Google *Project Fi* over two carriers (T-Mobile and Sprint), and discover three issues, all of which are independent of its excellent implementations (§3): (P1) The anticipated switch is never triggered even when the serving carrier's coverage is pretty weak; (P2) The switch takes rather long time (tens of seconds or minutes) and prolongs service unavailability; and (P3) the device fails to choose the high-quality network (*e.g.*, selecting 3G with weaker coverage rather than 4G with stronger coverage).

It turns out that, the above issues can be effectively addressed by using low-level cellular information (*e.g.*, available carriers, which carriers to scan, and radio/QoS profile for each carrier) and mechanisms. However, such fine-grained knowledge is not available to commodity phones in their default operations. This is rooted in the fundamental design of 3G/4G networks. With the single-carrier scenario in mind, 3G/4G follows the design paradigm of *"smart core, dumb end"*. It thus does not expose its low-level information to the device in normal operations. For multi-carrier access, however, end intelligence is a necessity, since individual carrier does not have global view on all carriers, which can only be constructed at the device through accessing low-level cellular events. Without using such knowledge, today's carrier selection could encounter issues P1-P3.

While the problem can be solved by the future architecture redesign (say, 5G), it usually takes years to accomplish. Instead, we seek to devise a solution that works with the current 3G/4G network, in line with the ongoing industrial efforts, *e.g.*, Google *Project Fi*, Apple SIM and Samsung e-SIM. Specifically, we address the following problem: *Can we leverage low-level cellular*

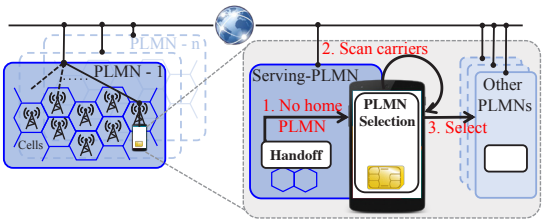


Figure 1: Multi-carrier network access (left) and inter-carrier switch via PLMN selection (right).

information and mechanisms at the device to further improve multi-carrier access? Our study yields a positive answer.

We propose *iCellular*, a client-side service to let mobile devices customize their own cellular network access. Complementing the design of *Project Fi*, *iCellular* further leverages low-level, runtime cellular information at the device during its carrier selection. *iCellular* is built on top of current 3G/4G mechanisms at the device, but applies cross-layer adaptations to ensure responsive multi-carrier access with minimal disruption. To facilitate the device to make proper decisions, *iCellular* exploits online learning to predict the performance of heterogenous carriers, and provides built-in strategies for better usability. It further safeguards access decisions with fault prevention techniques. We implement *iCellular* on commodity phone models (Nexus 6 and Nexus 6P) and assess its performance with *Project Fi*. Our evaluation shows that, *iCellular* can achieve 3.74x throughput improvement and 1.9x latency reduction on average by selecting the best mobile carrier. Meanwhile, *iCellular* has negligible impacts on the device's data service and OS resource utilization (less than 2% CPU usage), approximates the lower bounds of responsiveness and switch disruption, and shields its selection strategies from decision faults.

The rest of the paper is organized as follows. §2 introduces the background. §3 describes our findings and uncovers root causes of multi-carrier access. §4, §5, and §6 present the design, implementation and evaluation of *iCellular*, respectively. §7 discusses remaining issues, and §8 presents the related work. §9 concludes the work.

2 Mobile Network Access Primer

A cellular carrier deploys and operates its mobile network (called public land mobile network or PLMN) to offer services to its subscribers. Each PLMN has many cells across geographical areas. Each location is covered by multiple cells within one PLMN and across several PLMNs (*e.g.*, Verizon, AT&T, T-Mobile, Sprint).

Single-carrier network access. Today's cellular network is designed under the premise of single-carrier access. A mobile device is supposed to gain access directly from its home PLMN. It obtains radio access from the serving cell and further connects to the core carrier network and the external Internet, as shown in the left plot of Figure 1. When the current cell can no longer serve the device (*e.g.*, out of its coverage), the device is migrated to another available cell within the same PLMN. This is called handoff.

Roaming between carriers. When the home PLMN cannot serve its subscribers (*e.g.*, in a foreign country), the device may roam to other carriers (visiting networks). This is realized through the PLMN selection procedure between carriers [12], which is a mandatory function for all commodity phones. It supports both automatic (based on a pre-defined PLMN priority list) and manual modes. As shown in the right plot of Figure 1, once triggered by certain events (*e.g.* no home PLMN service), PLMN selection should first scan the available carriers, and then choose one based on the pre-defined criteria (*e.g.* preference) or the user manual operation. If the device decides to switch, it will deregister from the current carrier network and then register to a new one. In this process, network access may be temporarily unavailable. This is acceptable since inter-carrier switch is assumed to be infrequent, thus having limited impacts.

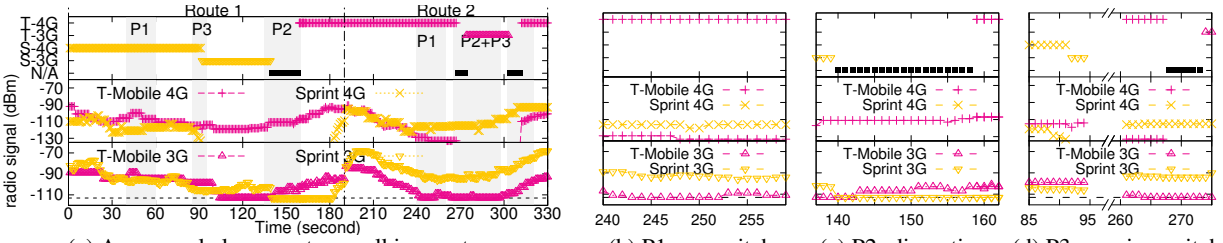
Multi-carrier access with universal SIM card. Recent industrial efforts aim at providing mobile device access to multiple carriers with a single SIM card. They include Google Project Fi [26], Apple SIM [14], and Samsung e-SIM [24]. With the SIM card, the device can access multiple cellular carriers (*e.g.*, T-Mobile and Sprint in *Project Fi*). Given only one cellular interface, the device uses one carrier at a time.

3 Multi-carrier Access: Promises & Issues

We run experiments to quantify the benefits of multi-carrier access, and identify the downsides of the today's efforts. The identified limitations are independent of implementations, but rooted in the 3G/4G design.

Methodology. We conduct both controlled experiments and a one-month user study using two Nexus 6 phones with Google *Project Fi* [26], which was released in May 2015. *Project Fi* provides access to two U.S. carriers (T-Mobile and Sprint) at this time. It develops an automatic carrier selection on commodity phones using a proprietary mechanism. Unfortunately, details of its switching algorithm have not been published. We contacted *Project Fi* team and learned that this algorithm aims at optimizing consumer experience, and considers network performance, battery usage and data activity during selection. We further inferred its decision and execution strategies from our experiments.

In each controlled test, we use a Nexus 6 phone with a *Project Fi* SIM card, and test with *Project Fi*'s automatic carrier selection mode. We walk along two routes



(a) An example log over two walking routes (b) P1: no switch (c) P2: disruption (d) P3: unwise switch

Figure 2: An example log for serving carriers and networks and three problematic instances through *Project Fi*.

within the campus buildings at UCLA and OSU at the idle mode (no data/voice, screen off). We walk slowly ($<$ 1 m/s) and record the serving carrier ("T" for T-Mobile, "S" for Sprint) and its network type (4G or 3G) per second. Meanwhile, we carry other accompanying phones to record the radio signal strength of each access option (T-4G, T-3G, S-4G, S-3G). We run each test 10 times and similar results are consistently observed in all the tests. In the user study (07/31/15 to 09/02/15), we use the *Project Fi*-enabled phone as usual and collect background device and cellular events with *MobileInsight*, an in-phone cellular monitoring tool [4]. We have collected 4.9GB logs with *MobileInsight* in total, with 274,351 messages from radio resource control (RRC), 16,470 messages from mobility management (MM), and 5,365 messages from session management (SM). We next present the results from the controlled experiments as motivating examples. The user study to be described in §4 and §6 confirms that these issues are common in practice.

3.1 Motivating Examples

Merits of multi-carrier access. We first verify that exploiting multiple carriers is indeed beneficial to service availability and access quality. Figure 2a shows the results from the controlled experiments over two routes. On the first route [0s,190s), Sprint gradually becomes weaker and then fades away, but its dead zone is covered by T-Mobile; On the second route [190s, 330s], in contrast, Sprint offers stronger coverage, even at locations with extremely weak coverages from T-Mobile. Multi-carrier access indeed helps to enhance network service availability by boosting radio coverage. For example, in [160s, 180s], the phone switches to T-Mobile and retains its radio access while Sprint is not available. Moreover, we confirm that it further improves data access throughput and user experiences. The *Project Fi* indeed offers a major step forward on mobile Internet access.

Our examples further reveal three issues, which demonstrate that the benefits of multi-carrier access have not been fully achieved.

P1. No anticipated inter-carrier switch. It is desirable for the device to migrate to another available carrier network for better access quality, when the device perceives degraded quality from its current, serving carrier. However, our experiments show that, the device often gets stuck in one carrier network, and misses the better network access (e.g., during [40s, 60s] and [240s, 260s] of Figure 2). As shown in Figure 2b, T-Mobile experiences extremely weak radio coverage ($<$ -130 dBm in 4G and $<$ -110 dBm in 3G), but the phone never makes any attempt to move to Sprint, regardless of how strong Sprint's radio signal is. As a result, the device fails to improve its access quality. Moreover, we find that the expected switch often occurs until its access to the original carrier (here, T-Mobile) is lost. This is rooted in the fact that the inter-carrier switch is triggered when the serving carrier fails. Therefore, the device becomes out of service in this scenario, although better carrier access remains available.

P2. Long switch time and service disruption. Even when inter-carrier switch is eventually triggered, it may disrupt access for tens of seconds or even several minutes (see Figure 6 for the user-study results). In the example of Figure 2c, the phone starts Sprint→T-Mobile roaming at the 140th second, but it takes 17.3s to gain access to T-Mobile 4G. This duration is much longer than the typical handoff latency (possibly several seconds [42]). It is likely to halt or even abort any ongoing data service. We look into the event logs (Figure 3) to examine why the switch is slow. It turns out that, most of the switch time is wasted on an *exhaustive* scanning of all possible cells, including nearby cells from AT&T and Verizon. In this example, it spends 14.7s on radio-band scanning and 2.6s on completing the registration (attachment) to the new carrier (here, T-Mobile). Note that, such heavy scanning overhead is not incurred by any implementation glitch. Instead, it is rooted in the *Project Fi*'s design, which selects a new carrier network only after an exhaustive scanning process. In this work, we want to show that such large latency is unnecessary. It can be reduced without compromising inter-carrier selection.

P3. Unwise decision and unnecessary performance degradation. Our next finding is that, the device fails

Time	Event	
11:19:57.414	**Out-of-service. Start network search**	RF band scanning: 14.7s
11:19:57.628	Scanning AT&T 4G cell 1, unavailable	
11:19:57.748	Scanning AT&T 4G cell 2, unavailable	
…	…	
11:20:11.788	Scanning Verizon 4G cell 1, unavailable	
…	…	
11:20:12.188	**Scanning T-Mobile 4G cell 1, available**	Network registration: 2.6s
11:20:12.771	Attach request (to T-Mobile 4G)	
11:20:14.788	Attach accept	

Figure 3: Event logs during P2 (disruption) of Fig. 2c.

to migrate to the better choice, thus unable to enjoy the full benefits of multi-carrier access. The phone often moves to 3G offered by the same carrier, rather than the 4G network from the other carrier that yields higher speed. Figure 2d illustrates two such instances. After entering an area without Sprint 4G at the 91st second, the device switches to Sprint 3G, despite stronger radio signals from T-Mobile 4G. This indicates that the intra-carrier handoff is preferred over the inter-carrier switch in practice. Unfortunately, such a preference choice prevents the inter-carrier switch from taking effect. Even worse, obstacles still remain even when the network access to the original carrier has been shortly disrupted. For instance, during [267s, 273s], the original carrier (T-Mobile 3G) is still chosen. In this case, T-Mobile 4G and 3G networks almost have no coverage. In short, the device acts as a single-carrier phone in most cases, even with the multi-carrier access capability. Inter-carrier switch is not triggered as expected.

3.2 Insights

The above examples also shed lights on how to solve the three problems. The key is to leverage low-level cellular information and mechanisms at the device when selecting access from multiple carriers.

Specifically, performing the anticipated switch (P1) states that, the device performs inter-carrier switch upon detecting a better carrier, even when the serving carrier is still available. This further requires the device to learn all available carriers and their quality at runtime. Note that such information can be obtained from the low-level cellular events. However, the default operation on commodity phones will not do so. Moreover, the naive approach of forcing the phone to proactively scan other carriers at any time may lead to temporary disconnection from the current carrier network. We elaborate on how we address these issues in §4.1.

To reduce the switch time (P2), the device should refrain from exhaustive search of all carriers at all times. This requires the device to perform fine-grained control on which carriers should be scanned. It can be done by configuring the low-level mechanism for monitoring.

To make a wise selection decision (P3), the device should treat all intra-carrier handoffs and inter-carrier switches equally, and select the best carrier network. This requires the device to directly initiate the inter-carrier switch when needed. This also calls for leveraging the low-level cellular mechanism.

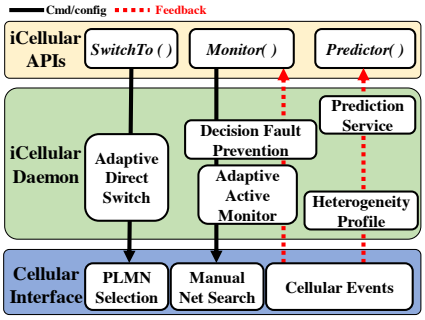


Figure 4: *iCellular* system architecture.

In summary, low-level domain knowledge can be exploited to effectively address all three issues. However, the default operation mode on commodity phones does not expose such fine-grained cellular information and mechanisms to higher layers. The reason is that, the 3G/4G network follows the design paradigm of "smart core, dumb end" with the single-carrier usage scenario in mind. The end device does not need to exploit such information when selecting its carrier access. Since such low-level, cellular-specific domain knowledge is not available for the default operation mode, it might be the reason why *Project Fi* has not explored this direction in its current design.

4 *iCellular* Design

We now present *iCellular*, which explores an alternative dimension to improve multi-carrier access. *iCellular* complements the design of *Project Fi* by leveraging low-level cellular information and mechanisms. It seeks to further empower the end device to have more control on its carrier selection, while addressing the issues in §3.1.

For incremental deployability, *iCellular* is built on top of the PLMN selection [11, 12], a standardized mechanism mandatory on all phones. Note that, however, the basic PLMN selection suffers from similar issues in §3.1: migrating to other carriers is not preferred unless the home carrier fails (P1); the exhaustive scanning (P2) and the preferable intra-carrier handoffs (P3) are still in use. The reason is that, the default PLMN selection scheme is designed under the premise of single-carrier access. While roaming to other carriers is allowed, it is not preferred by the home carrier unless it fails to offer network access to its subscribers. So the basic PLMN selection has the following features: (1) *Passive triggering/monitoring:* When being served by one carrier, the device should not monitor other carriers or trigger the selection until the current one fails (*i.e.*, out of coverage); (2) *Network-controlled selection:* The device should select the new carrier based on the preferences pre-defined by the home carrier and stored in the SIM card; (3) *Hard switch:* The device should deregister from the old car-

Function	Method	Cellular Events	Type
Active monitor (§4.1)	Disruption avoidance	Paging	Meas
		Paging cycle	Config
	Minimal search	Radio meas	Meas
		RRC SIB 1	Config
Prediction service (§4.3)	QoS profile	EPS/PDP setup	Config
	Radio profile	RRC reconfig	Config
Decision fault prevention (§4.4)	Access control	RRC SIB1	Config
	Interplay with net mobility	Cell reselection in RRC SIB 3-8	Config
	Function completeness	GMM/EMM location update	Config

Table 1: Cellular events used in *iCellular*.

rier first, and then register to the new one. We thus need to adapt the PLMN selection scheme to the multi-carrier context by using low-level cellular events.

Figure 4 illustrates an overview of *iCellular*. In brief, *iCellular* systematically enhances the devices' role in every step of inter-carrier switch with runtime cellular information, spanning triggering/monitoring, decision making and switch execution. To be incrementally deployable on commodity phones, we build *iCellular* on top of the existing mechanisms from the phone's cellular interface [7]. We exploit the freedom given by the standards, which allow devices to tune configurations and operations to some extent. To ensure responsiveness and minimal disruption, *iCellular* applies cross-layer adaptations over existing mechanisms (§4.1 and §4.2). To facilitate the devices to make wise decisions, *iCellular* offers cross-layer online learning service to predict network performance (§4.3), and protects devices from decision faults (§4.4). To enable adaptation, prediction and decision fault prevention, *iCellular* incorporates realtime feedbacks extracted from low-level cellular events. Different from approaches using additional diagnosis engine (*e.g.*, QXDM [37]) or software-defined radio (*e.g.*, LTEye [30]), we devise an in-phone mechanism to collect realtime cellular events (§4.5, cellular events are summarized in Table 1). These components are designed to be scalable, without incurring heavy signaling overhead to both the device and the network.

4.1 Adaptive Monitoring

To enable device-initiated selection, the first task is to gather runtime information on available carrier networks. This is done through *active monitoring*. It allows a device to scan other carriers even while being served by one. This would prevent the device from missing a better carrier network (P1 and P3 in §3). For this purpose, the only viable mechanism on commodity phones is the `manual network search` [12]. It was designed to let a device manually scan all available carriers. Once initiated, the device scans neighbor carriers' frequency bands, extracts the network status from the broadcasted system information block, and measures their radio quality. No extra signaling overhead is incurred, since the active monitoring approach does not activate signaling exchanges between the device and the network. To be incrementally deployable, we decide to realize active monitoring on top of the manual network search.

Note that naive manual search does not satisfy properties of minimal-disruption and responsiveness. First, scanning neighbor carriers may disrupt the network service. The device has to re-synchronize to other carriers' frequency bands, during which it cannot exchange traffic with the current carrier. Second, it is *exhaustive* to all carriers by design. Even if the device is not interested in certain carriers (*e.g.*, no roaming contract), this function would still scan them, thus delaying the device's decision and wasting more power. The challenge is that, both issues cannot be directly addressed with application-level information only. *iCellular* thus devises cross-layer adaptions for both issues.

Disruption avoidance. To minimize disruptions on ongoing services, *iCellular* schedules scanning events only when the device has no application traffic delivery. This requires *iCellular* to monitor the uplink and downlink traffic actvities. While the uplink one can be directly known from the device itself, the status for downlink traffic is hard to predict. Traffic may arrive while the device has re-synchronized to other carriers' cells. If so, its reception could be delayed or even lost.

iCellular prevents this by using the low-level cellular event feedback. We observe that in the 3G/4G network, the downlink data reception is regulated by the periodical paging cycle (*e.g.*, discontinuous reception in 4G [9,38]). To save power, the 3G/4G base station assigns inactivity timers for the device. The device periodically wakes up from the sleep mode, monitors the paging channel to check downlink data availability, and moves to the sleep mode again if no traffic is coming. *iCellular* obtains this cycle configuration from the radio resource control (RRC) messages, and schedules its scanning operations only during the sleep mode. Figure 5 shows our one-month logs of 4G per-cell search time at a mobile device with *Project Fi*. It shows that, 79.2% of cells can be scanned in less than one paging cycle. Others need more cycles to complete the scanning. With this design, no paging event is interrupted by monitoring.

One valid concern is that, the monitoring results may become obsolete due to continuous data transmissions, thus leading to wrong decisions. This is unlikely to happen in practice for two reasons. First, most traffic tends to be bursty, which leaves sufficient idle period for background monitoring. Second, network performance tends to vary smoothly, and stale monitoring results do not affect the final selection decision. Furthermore, *iCellular* compares the elapsed time between the decision making and the measurement. Obsolete measurements outside the time window (say, 1 minute) will not be used.

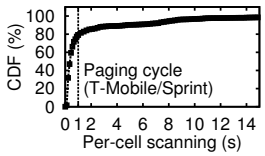


Figure 5: Cell scan time.

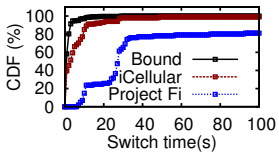


Figure 6: Switch time.

Minimal search. Instead of exhausting all carrier networks, *iCellular* scales the monitoring by restricting the manual search only to those specified by the device. To realize this idea, the practical issue is that no such option is available in the manual network search mechanism. We thus leverage adaptation of the PLMN preference. Given the list of carrier networks of interests , *iCellular* configures the cellular interface to let the manual network search scan these carriers first. This is achieved by assigning them with highest PLMN preferences. During the manual search, *iCellular* listens to the cellular events to see which carrier is being scanned. These events include the per-cell radio quality measurements, and its system information block with PLMN identifiers. Once *iCellular* detects that the device has finished scanning of the device-specified carriers, it terminates the manual network search function.

Monitoring-decision parallelism. Sometimes there is no need to complete all the monitoring to determine the target carrier network. For example, if the user prefers 4G, it can decide to switch whenever a good 4G is reported, without waiting for 3G results. To support this, *iCellular* allows devices to make decisions with partial results, thus further accelerating the process. Instead of waiting for all scanning results, *iCellular* triggers the decision callback whenever new results are available.

4.2 Direct Inter-carrier Switch

iCellular aims at reducing the disruption time incurred by inter-carrier switching as much as it can. We find that, there is enough room for this because *most service disruption time is caused by frequency band scanning* (§3). With the active monitoring function, *iCellular* does not need to scan the carrier networks during switch. Specifically, given a target carrier network, *iCellular* makes a direct switch by configuring the target carrier with highest PLMN preference. It then triggers a manual PLMN selection to the target carrier network. This way, the device would directly switch to the target without unnecessary scanning.

We next show how *iCellular* approximates to the lower bound of the switch time. In cellular networks, switching to another network requires at least deregistration from the old network (detach), and registration to the new network (attach). According to [10], the detach time is negligible, since the device can detach directly without interactions to the old carrier network. So the minimal disruption time in switch is roughly equal to the attach time, *i.e.*, $T_{switch,min} \approx T_{attach}$. For *iCellular*, no extra attempts to other carrier networks are made. Since it is on top of the PLMN selection, the scanning of the target carrier still remains. Therefore, the switch time is

$$T_{switch,iCellular} = n_t T_t + T_{attach} = n_t T_t + T_{switch,min} \quad (1)$$

where n_t and T_t are the cell count and per-cell scanning time for the target carrier network, respectively. Compared with the attach time, this extra overhead is usually negligible in practice. Figure 6 verifies this with our one-month background monitoring results in *Project Fi*. It shows that, *iCellular* indeed approximates the lower bound, despite this minor overhead.

4.3 Prediction for Heterogeneous Carriers

To decide which carrier network to switch to, the device may gather performance information on each carrier network. Ideally, the device needs to measure every available carrier network's current performance (*e.g.*, latency or throughput) and make decisions. Unfortunately, this is deemed impossible. The device can only measure the serving network's performance; other candidates' performances cannot be measured without registration.

Given this fact, *iCellular* decides to assist the device to predict each carrier's performance. Our predication is based on the regression tree algorithm [19]. It models the network/application performance (y) as a function of a feature vector (x_1, x_2), where x_1 is runtime radio measurement and x_2 is carrier network profiles (elaborated below). The model is established using a pre-stored tree at bootstrap and then recursively updated with new online samples. Note that radio measurement alone is insufficient to predict performance, because different carriers may apply heterogeneous radio technologies and resource configurations. Our prediction works as follows.

Prediction metric (y). This metric is used to rank the performances of all available networks. We explore both network-level (link throughput, radio latency) and application-level ones (*e.g.*, web loading latency, video suspension time). They are obtained from both network and application events, for example, Appendix B shows how to obtain app-specific metrics. We want to point out that the app-specific metric often leads to the same selection decision (see the evaluation §6). This is because the performance characteristics of a carrier network tend to have consistent impacts on all applications.

Training sample collection. The training sample (x, y) for a network is collected in the background, without interrupting the device's normal usage. A new training sample is collected when a new observation of the performance metric y is generated (*e.g.*, throughput from physical layer, loading time for Web-page download, latency per second for VoIP). In the meantime, radio mea-

Profile		Sprint		T-Mobile	
		Value	Prob	Value	Prob
QoS	Traffic class	Background	100%	Interactive	97.5%
	Delay class	4 (best effort)	100%	1	100%
	Max dlink rate	200Mbps	100%	256Mbps	100%
	Max ulink rate	200Mbps	100%	44Mbps	100%
Radio	Duplex type	TDD	88.3%	FDD	100%
	Paging cycle	100–200ms	81.5%	100ms	99.4%
	Handoff priority	2/3/6	100%	2/3/6	100%

Table 2: Heterogeneous cellular network profiles.

surement and network profiles for the serving network are recorded as $x = (x_1, x_2)$. For the radio quality x_1, *iCellular* extracts the serving network's RSRP (if 4G) or RSCP (if 3G) from the runtime active monitor (§4.1). For the network profile, *iCellular* currently collects two types (Table 2): (1) QoS profile from the data bearer context in session management, which includes the delay class and peak/maximum throughput; (2) radio parameters from the RRC configuration message, which includes the physical and MAC layer configurations. Note that the device cannot gain these profiles at runtime without registration to the carrier network of interest. To address this issue, we observe that network profiles are quite predictable. This is validated by our 1-month user study. Table 2 lists the predictability of some parameters from this log. For each parameter, we choose the one with the highest probability, and shows its occurrence probability. Note that, most QoS and radio configurations are invariant of time and location. The reason is that, the carriers tend to apply well-tested operation rules (*e.g.*, link adaptation and scheduling), with minor tunings to each base station/controller. As a result, we only store a set of unique values, and reuse it for all the applicable samples until changes are found.

Online predication and training. *iCellular* uses an online regression tree algorithm [19] as its predictor. The predictor is represented as a tree, with each interior node as a test condition over x (radio measurements and profile fields). Each decision is made upon the arrival of the feature vector x. It estimates the per-network metric y and selects the one with the highest rank.

iCellular updates the predictor's decision tree in the online fashion when a new sample arrives. At the bootstrap phase, it pre-stores a regression tree based on an offline training as the basis. Given a new sample (x, y), *iCellular* first determines whether a predictor update is needed. It runs the existing predictor over the heterogeneity information and runtime radio measurements, and obtains an estimated metric $y\prime$. If $|y - y\prime| = \min_{z \in leaf} |y - z|$, which implies the current sample fits well with the existing model, no update is needed. Otherwise, the predictor is updated as follows. Given the new sample and the existing tree, *iCellular* searches a new field (measurement or profile) that best splits the samples by minimizing the impurities in the two children nodes (based on the least-square criterion). Given this new split, we create a new pair of leaves for this new field, and completes the update of the prediction tree. Note that *iCellular* responds to new changes, and does not need to permanently store all training samples. This way, *iCellular* is scalable in storage and computation.

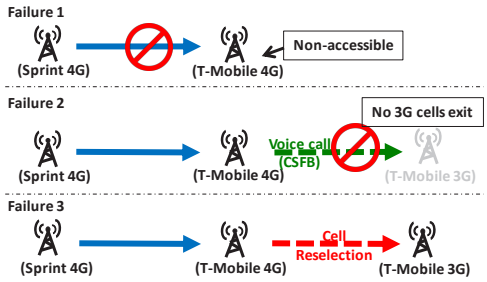


Figure 7: Three types of improper switch decisions.

4.4 Decision Fault Prevention

Letting a device customize its access strategy can be a double-edged sword. With improper strategies, the device may make faulty switch decisions and cause unexpected service disruption. Figure 7 shows three categories of failures caused by decision faults, all of which can only be detected with low-level cellular information:

Failure 1: No network access. Certain networks may be temporarily inaccessible. For example, our user study reports that, a Sprint 4G base station experiences a 10-min maintenance, during which access is denied.

Failure 2: No voice service. In some scenarios, the target carrier network cannot provide complete voice services. Figure 8 shows an instance from our user study. T-Mobile provides its voice service using circuit-switched-fall-back (CSFB), which moves the device to 3G for the voice call. However, there exist areas not covered by T-Mobile 3G (*e.g.*, signal strength lower than -95dBm according to [8]). In this scenario, the user in Sprint 4G should not switch to T-Mobile 4G, which cannot support voice calls without the 3G infrastructure.

Failure 3: Unexpected low-speed data service. The user selection may not be honored by the individual carrier's handoff rules. Figure 9 reports an instance from our user study. The user under Sprint 4G may decide to switch to one T-Mobile 4G. However, under the same condition, T-Mobile's mobility rules (*e.g.*, cell reselection [11]) would switch its 4G users to its 3G. In this case, the user's decision to T-Mobile 4G is improper, because the target network (T-Mobile 3G) is not preferred, and this switch incurs unnecessary disruptions.

To prevent decision faults, *iCellular* chooses to safeguard the device's decisions from those faulty ones. It checks whether each carrier network has any of the above problems, and excludes such carriers from the monitoring results. This prevents the device from switch-

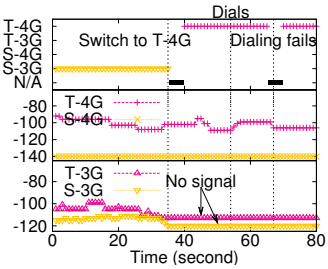


Figure 8: Switch to a network with no voice support.

Time	Event
17:49:07.520	**Deregister from Sprint 4G**
17:49:13.433	Scanning T-Mobile 4G cell 1, available
17:49:13.508	**Cell reselection config in SIB6: switch to 3G when $RSRP_{4G}$<-120dBm**
17:49:15.142	Attach accept
17:49:20.106	**$RSRP_{4G}$= -122dBm**
17:49:21.326	**Cell reselection to T-Mobile 3G**

Figure 9: Interplay between user and network's mobility.

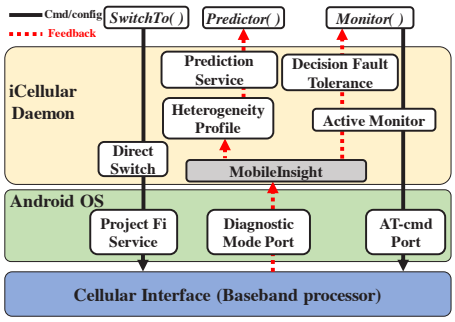


Figure 10: Overview of *iCellular* implementation.

ing to these carrier networks. To this end, *iCellular* first profiles each carrier's low-level access-control list from the RRC system-info-block message [9], data/voice preference configuration from registration/location update messages [10], and the network-side mobility rules from the RRC configuration message [9,11]. At runtime, for each candidate carrier, it checks if it is in the forbidden list (Failure 1), has no voice service with satisfactory 3G radio quality (Failure 2), or has satisfied mobility rules for further switch (Failure 3). If any condition is satisfied, it would be removed from the monitoring list.

4.5 Cellular Events Collection

As shown in §4.1-§4.4, *iCellular* relies on low-level cellular events to perform cross-layer adaptations over the existing mechanisms, predict the network performance, and avoid possible switch faults. The cellular events include the signaling messages exchanged between the device and the network, and radio quality/load measurements. Table 1 summarizes the events required by *iCellular*. Note that some events (*e.g.*, paging) should be extracted at realtime for feedbacks. Unfortunately, obtaining realtime cellular events on commodity phones is not readily available today. These events are not exposed to mobile OS or applications. There exist commercial tools (*e.g.*, QXDM [37]) and research projects (*e.g.*, LTEye [30]) to extract them. However, they require an external platform (*e.g.*, laptop or a special hardware (USRP)) to connect to the mobile device, which limits the device's flexible movement and its applicability. They cannot meet *iCellular*'s realtime requirements. To this end, we develop an in-phone solution MobileInsight [4] by exploiting the existing cellular diagnostic mode. We enable the diagnostic mode on the phone, modify the the virtual device for it (root access needed), and finally expose them to *iCellular*. This solution can be deployed on commodity phones without hardware changes.

5 Implementation

We have implemented *iCellular* on Motorola Nexus 6 and Huawei Nexus 6P. They run Android OS 5.1 and 6.0 using Qualcomm Snapdragon 805 and 810 chipsets, respectively. Both support 4G LTE, 3G HSPA/UMTS/CDMA and 2G GSM. To activate access to multiple cellular networks, we have installed *Project Fi* SIM card on Nexus 6/6P, which supports T-Mobile and Sprint 3G/4G. Figure 10 illustrates the system implementation. *iCellular* runs as a daemon service on a rooted phone. To enable interactions with the cellular interface, we activate the baseband processing tools (in bootloader), and turn on the diagnostic mode [2] and AT-command interfaces.

Basic APIs. *iCellular* allows the device to control its cellular access strategies through three APIs: `Monitor()` for active monitoring (§4.1), `Predictor()` for performance prediction (§4.3) and `SwitchTo()` for direct switching (§4.2). The decision fault tolerance is enabled by default (§4.4). Appendix A presents an illustrative example on how to use them.

Usability-Flexibility tradeoff. The above basic APIs provide most flexible means to customize access strategies. In practice, however, there is no need for most normal users to customize the strategies from the scratch. To support better usability, *iCellular* provides some built-in strategies on top of the basic APIs. Devices can choose these pre-defined ones, rather than build customized versions by themselves. We have developed three strategies: prediction-based, radio quality only and profile only (see §6 for performance comparisons).

Adaptive active monitoring (§4.1). We implement `Monitor()` with manual search and adaptations. Our prototype initiates the search with an AT query command `AT+COPS=?`. The non-disruption and minimal search adaptations are implemented for events of Table 1.

Adaptive direct switch (§4.2). We implement the `SwitchTo()` on top of PLMN selection, with dynamic adaptations for direct switch. Ideally, this can be executed with the AT command `AT+COPS=manual,carrier,network`. However, this command is forbidden by the cellular interface of Nexus 6/6P. We thus take an alternative approach. We modify the preferred network type through Android's API `setPreferredNetworkType`, and change the

carrier with *Project Fi*'s secret code. Admittedly, this approach may incur extra switch overhead, but it is still acceptable (§6.2).

Prediction for heterogenous carriers (§4.3). We implement `Predictor()` in two steps. First, we implement the online sample collection, which collects radio measurements, RRC configurations and QoS profiles as features. We also define a callback to collect the network/application-level performance metrics. We then implement the online regression tree algorithm for training and prediction.

Decision fault prevention (§4.4). The fault prevention function is implemented as a shim layer between active monitoring and basic APIs. It detects the potential switch faults based on monitoring results and heterogeneity profiling, and excludes the unreachable carrier networks from the monitoring results. We further add a runtime checker in `SwitchTo()`, and prevent devices from selecting carriers not in the scanning results.

Cellular events collection (§4.5). We use the built-in realtime cellular loggers from *MobileInsight*. We develop a proxy daemon for the diagnostic port (/dev/diag), and redirect the events to the phone memory.

6 Evaluation

We evaluate *iCellular* along two dimensions. We first present the overall performance improvement by *iCellular* with smart multi-carrier access (§6.1), and then show *iCellular* satisfies various design properties in §4 (§6.2). All experiments are conducted on commodity Nexus 6 phones with *iCellular* in two cities of Los Angeles (west coast) and Columbus (Midwest), mainly around two campuses. The results on Nexsus 6P are similar.

6.1 Overall Performance

We use four representative applications to assess *iCellular*: SpeedTest (bulk file transfer), Web (interactive latency for small volume traffic), Youtube (video streaming) and Skype (realtime VoIP). We evaluate each application with quality-of-experience metrics whenever possible, *i.e.*, downlink speed for SpeedTest, page-loading time for Web [13] (measured with Firefox), video suspension time for Youtube [32] (measured by its APIs), and latency for Skype [27] (measured with its tech info panel). The details to collect application performance metrics are given in Appendix B. We run both pedestrian mobility and static tests. Along the walking routes, we uniformly sample locations. Note that *Project Fi*'s automatic selection protects the device's data connectivity by deferring its switch to the idle mode. For fair comparisons, we move to each sampled location in the idle mode (no voice/data, screen off), wait for sufficiently long time ($\geq$1min) for potential switch during idle, and then start

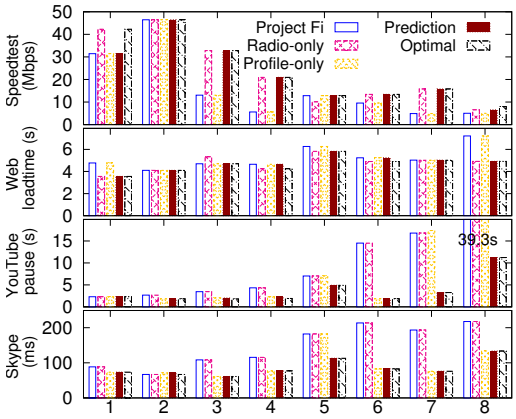


Figure 11: Performance of Speedtest, Web, Youtube, Skype using various multi-carrier access schemes.

to test each application. We have at least five test runs and use the median value for evaluation.

We compare *iCellular* and its variants, with two baselines: *(i)* ***Project Fi*'s automatic selection** and *(ii)* **Optimal strategy:** We obtain the optimal access option by exhausting the application or network performance at each location. It may not be achieved in reality, but it serves as an ideal performance benchmark. We test three built-in *iCellular* decision strategies (§5): (1) **Prediction-based:** the default strategy in *iCellular*, which chooses the carrier with the best ranking metric from the predictor §4.3. The predictor is trained based on our one-month user-study logs, and tested over different routes. (2) **Radio-only**: the *de-facto* handoff strategy in 3G/4G. We implement the standardized cell re-selection scheme [11]. Whenever a network 4G with its signal strength higher than -110dBm (defined in [11]) exists, the strongest 4G carrier is chosen. Otherwise, we choose the strongest 3G network. (3) **Profile-only:** the device is migrated to the carrier network with the highest QoS (see Table 2). For our *iCellular* strategies, we use the carrier list with all network types supported by *Project Fi* (*i.e.*, 3G and 4G in T-Mobile and Sprint).

Figure 11 plots their performances in eight instances (locations), which belong to three categories: both carriers with acceptable coverage (Case 1-2), one carrier with acceptable coverage but the other not (Case 3-5), both carriers with weak coverage and one is even weaker (Case 6-8). We further compare them with the optimal one in two dimensions: accuracy toward the optimality, and the performance gap/improvement.

Accuracy toward optimality. We compare the probability that each scheme reaches the optimal network. Let I and I_{opt} be the access options chosen by the test scheme and the optimal strategy. We define the hit ratio as the matching samples $|(I \doteq I_{opt})|$ over all test samples. Table 3 shows the hit ratios of all schemes by different applications. *iCellular*'s prediction-based strategy

	Project Fi	Radio-only	Profile-only	Prediction
Speedtest	47.3%	63.1%	36.8%	73.6%
Web	57.9%	73.6%	31.6%	57.8%
Youtube	16.9%	22.6%	49.1%	50.9%
Skype	24.5%	7.6%	84.9%	92.5%

Table 3: Statistics of accuracy toward the optimality.

	SpeedTest	Web	Youtube	Skype
Radio meas	36.5%	72.7%	26.4%	8.7%
Heterogeneity profile	63.5%	27.3%	73.6%	91.3%

Table 4: Weights of radio measurement and network profiles in *iCellular*'s prediction strategy.

makes a wiser multi-carrier access decision. The hit ratios are 73.6%, 57.8%, 50.9% and 92.5% in SpeedTest, Web, Youtube and Skype, respectively. They are relatively small in Web and Youtube, but do not incur much performance degradation (explained later). They are usually higher than *Project Fi*'s automatic selection except for Web. The mobility speed has minor impact on the prediction accuracy, since it does not affect sample collection. Both radio measurements and cellular network profiles contribute to the high accuracy, but their impacts on all apps vary. We calculate their normalized variable importance in the regression tree (defined in [31]) and Table 4 shows their weights for four apps. We also find that, the metric specific for one app often locates the better network for other apps at the same location. The reason is that, the characteristics of one carrier network tend to have consistent impact on all apps. When the performance gap between two carriers is significant, it would exhibit on all application-level metrics.

Data service performance. We next examine the data performance by different schemes. We define the gap ratio $\gamma = |x - x^*|/x^*$, where x is the performance using various access strategies, x_{opt} is the optimal performance. We plot CDF of γ in Figure 12 and present the hit ratios and statistics of γ^+ in Table 5. Compared with *Project Fi*, *iCellular* narrows its performance gap (*e.g.*, reducing the maximal speed loss from 73.7% (19.7Mbps) to 25.7%, and the maximal video suspension time gap from 28.1s to 3.2s). The performance gain varies with locations (see Figure 11). With acceptable coverage (Case 1-2), *Project Fi*'s performance also approximates the optimal one. However, at locations with weak coverage, *iCellular* improves the device performance more visibly. The performance gain varies with applications (traffic patterns). Compared with other traffic, *iCellular* provides relatively small improvement for Web browsing. The reason is that, the Web traffic volume is relatively small, and no large performance distinction appears among various access options. However, for heavy traffic (e.g., file transfer), video streaming and voice calls, *iCellular* substantially improves the performance. The average improvement of *iCellular* over *Project Fi* approximates $\gamma_{fi} - \gamma_{icellular}$. On average, *iCellular* increases 23.8% downlink speed and reduces 7.3% loading time in Web, 37% suspension time in Youtube,

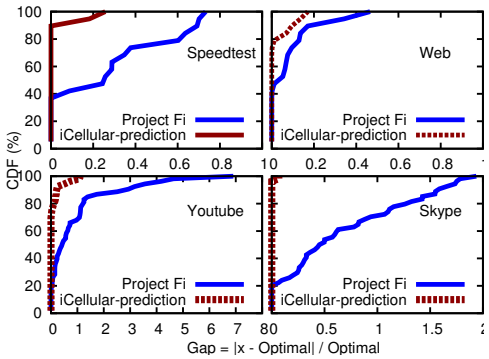


Figure 12: The performance gaps from *Project Fi* and *iCellular*'s prediction strategy to the optimality.

	Project Fi		iCellular-prediction	
	med(γ) ($\lvert x-x^*\rvert$)	max(γ) ($\lvert x-x^*\rvert$)	med(γ) ($\lvert x-x^*\rvert$)	max(γ) ($\lvert x-x^*\rvert$)
SpeedTest (speed)	36.2% 3.8Mbps	73.3% 19.8Mbps	**12.4%** 1.4Mbps	25.7% 9.8Mbps
Web (loadtime)	8.5% 0.5s	46.5% 2.3s	**1.2%** 0.2s	17% 0.7s
Youtube (Pause)	55% 1.4s	690% 28.1s	**18%** 0.3s	111% 3.2s
Skype (Latency)	62.9% 64ms	193.8% 117ms	**2.5%** 4.4ms	6.7% 4.5ms

Table 5: Performance gaps from the optimal one.

60.4% latency in Skype. Since *iCellular* often selects the optimal access, the maximal gain over *Project Fi* can be up to 46.5% in Web, 6.9x in Youtube, 1.9x in Skype, and 3.74x in Speedtest.

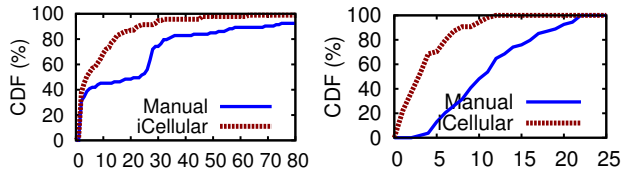
Comparison between *iCellular*'s built-in strategies. *iCellular*'s prediction strategy best approximates the optimal strategy. It outperforms radio-only and profile-only variants (§4.3). We also see that, the importance of profile and radio measurements varies across applications. For example, our log-event analysis shows that, T-Mobile assigns *Project Fi* devices to the interactive traffic class (Table 2), which is optimized for delay-sensitive service [6][1]. Instead, Sprint only allocates the best-effort traffic class to these devices. This explains why the profile-only strategy's performance approximates the optimal strategy for Skype. It also implies that, for a given application (*e.g.*, Skype), simpler strategy (rather than prediction), which incurs smaller system overhead, can be available for close-to-optimal performance.

6.2 Efficiency and Low Overhead

We next present the micro-benchmark evaluations on *iCellular*'s key components, and validate that they are efficient. We examine the active monitoring, direct switch and fault prevention, as well as the overhead of signaling, CPU, memory and battery usage.

Efficiency. We examine *iCellular*'s efficiency through two adaptive module tests. First, we show that, *iCel-*

[1]This QoS is specific to *Project Fi*. For example, we verify that a T-Mobile device with Samsung S5 is assigned lower background class.



(a) Total search time (b) Cell count

Figure 13: *iCellular*'s adaptive monitoring avoids exhaustive search.

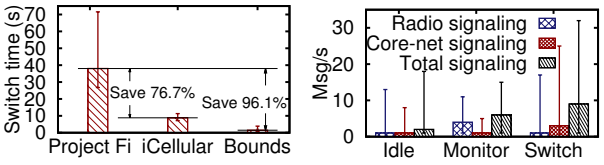
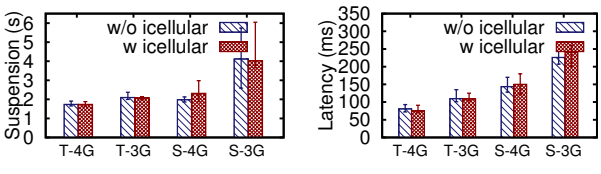


Figure 14: Inter-carrier switch time.

Figure 15: Cellular signaling overhead.

lular's adaptive monitoring is able to accelerate carrier scanning. We compare it with the default manual search, and record the total search time and the number of cells scanned at 100 different locations. Figure 13 shows that, with adaptive search, 70% of the complete search can be completed within 10s, 64% shorter than the exhaustive manual search. Note that devices are allowed to switch before the complete search (§5), so it waits shorter in practice. Figure 13b counts the scanned cells, and validates that such savings come from avoiding those unnecessary cell scans. The search time and the number of cells vary with locations and the cell density.

Second, we examine how well *iCellular*'s adaptive switch reduces service disruption. In this experiment, we place the phone at the border of two carriers' coverages, and test the switch time needed for *iCellular* and *Project Fi* for 50 runs. The inter-carrier switch time is defined as the duration from the de-registration from the old carrier to the registration to the new carrier. For comparison purposes, we also calculate the lower bound based on the *MobileInsight* event logs, described in §4.2. Figure 14 shows that, *iCellular* saves 76.7% switch time on average, compared with *Project Fi*. However, the current *iCellular* prototype has not achieved the minimal switch time: it still requires 8.8s on average. Under high-speed mobility, this may delay the switch to the optimal carrier network. We dig into the event logs, and discover that, the current bottleneck lies in the SIM card reconfiguration. The current *iCellular* implementation relies on *Project Fi*'s system service. It has to wait until the SIM card is reconfigured to switch to another carrier. In the experiments, we find that most of the switch times (7.3s on average) are spent on the SIM card reconfiguration, which is beyond the control of *iCellular*. The phone has no network service in this period. The lower bound implies that, with better SIM card implementation, *iCellular* could save up to 96.1% of switch time compared with the *Project Fi*.



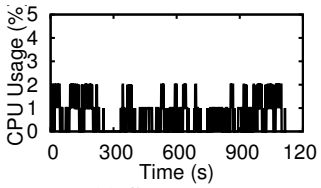
(a) Youtube (b) Skype

Figure 16: *iCellular*'s active monitoring has minor impacts on data performance.

Fault prevention. We next verify that *iCellular* handles fault scenarios and prevents devices from switching to unwise carrier networks. All three failure types in §4.4 have been observed in our one-month user study. Note that the failure scenarios are not very common in reality. We observe one instance of the forbidden access, where a Sprint 4G base station sets the access-barring option for 10 min (possibly under maintenance). We observe another instance of Figure 8, where T-mobile 4G is available but T-Mobile 3G is not available. Since T-Mobile 4G does not provide Voice over LTE (VoLTE) to *Project Fi* and has to rely on its 3G network (using circuit-switching Fallback) for voice calls [41]. Consequently, the correct decision should be to not switch to T-Mobile 4G, since voice calls are not reachable there. *iCellular* detects it from the profiled call preference and location update messages, and excludes this access option from the candidate list. We also observe uncoordinated mobility rules between the network and the device (Figure 9). We validate that *iCellular* can detect and avoid them.

Impact on applications in monitoring. We show that *iCellular*'s active monitor does not disrupt the ongoing data service at the device. We run the active monitor 100 times with/without applications and its active data transfer. We test with four applications and the results with-/without *iCellular*'s monitoring are similar. Figure 16 shows the performance with/without *iCellular*'s monitoring for Youtube and Skype. Enabling/disabling active monitoring has comparable application performance. As explained in §4.1, this is because the carrier scanning procedure is performed only in the absence of traffic.

Signaling overhead. We show that *iCellular* incurs moderate signaling messages to the device and the network. We record the device-side signaling message rate under three conditions (when running our performance tests): (1) *Idle:* No monitoring/switch functions are active. No extra cellular signaling messages are generated; (2) *Monitor: iCellular* initiates its active monitoring. The device should receive more broadcasted signals. However, no extra signaling messages are generated to the network; (3) *Switch: iCellular* initiates the switch to the new carrier network. Because of the registration, extra signaling messages are generated to both the device and the network. For all scenarios, we count the radio-



(a) CPU usage

WeChat	21.03%
iCellular	**4.75%**
Skype:	4.54%
Gmail:	3.33%
GO Power Master	1.21%
Google Play	0.73%
...	

(b) Battery usage

Figure 17: CPU and battery usage of *iCellular*.

level (from RRC layer), core-network level (from mobility and session management layers) and the total signaling rate. Figure 15 shows that, the maximum observed signaling message rate is 32 message/sec.

CPU and memory. In all our tests, the maximum CPU utilization is below 2%, while the maximum memory usage is below 20 MB (including virtual memory). Figure 17a shows a 20-min log during a driving test, where its maximum memory usage is 16.45MB.

Energy consumption. Since we cannot directly measure the consumed power at Nexus 6/6P with an external power meter (its battery is sealed, and hard to remove), we take an application-level approach. We use a fully-charged Nexus 6 phone and run it for 24 hours. We use an app called GO-Power-Master [3] to record energy consumption for each component/app. Figure 17b shows one record, where *iCellular* explicitly consumes about 4.75% of battery. Its energy can be further optimized (*e.g.*, with sleep mode and periodical monitoring).

7 Discussion

Working with network-side solution. Despite a device-side solution, *iCellular* can work in concert with carrier-side mechanisms for better performance. For example, during the inter-carrier switch, *iCellular* could benefit from the network-side buffering and tunneling of downlink traffic, and support more seamless migration. For each individual carrier, its network-side solution can also benefit from *iCellular* with device-side feedbacks on all available carriers. Note that the carrier network still retains its final say on the switch decision by rejecting the device-initiated switch requests.

Hints for future mobile network design. The future multi-carrier access design (*e.g.*, 5G) can benefit from our *iCellular*'s design experience. For example, the idea of adaptive monitoring (§4.1) and direct switch (§4.2) may be instrumental to designing a new inter-carrier switch mechanism beyond the popular PLMN selection. The heterogeneity predictor (§4.3) and decision fault prevention (§4.4) are also directly applicable to 5G.

8 Related Work

In recent years, exploiting multiple cellular carriers attracts research efforts on both network and device sides. The network-side efforts include sharing the radio resource [22, 28, 36, 36] and infrastructure [17, 18, 29, 44] between carriers, which helps to reduce deployment cost. On the device side, both clean-slate design with dual SIM cards [1,20] and single universal SIM card [14,24,26] are used for multi-carrier access. But multi-SIM phones provide multi-carrier access in a constrained fashion. The number of accessible carriers is limited by the number of SIM cards (usually two due to energy and radio interference constraints). Our work complements the single-SIM approach for incremental deployment. It differs from existing efforts by leveraging low-level cellular information, and offering device-defined selections in a responsive and non-disruptive manner.

iCellular leverages the rich cellular connectivity on the device. Similar efforts use multiple physical interfaces from WiFi and cellular, including WiFi offloading [16,21,23] and multipath-TCP [35,43]. *iCellular* differs from all these in that it still uses a single cellular interface. [15] reports similar problems, but we further unveil their root causes. Similar issues may also occur with traditional handoffs within a single carrier [39, 40, 45], which are caused by the carrier's own problematic management. Instead, *iCellular* targets inter-carrier migration, and chooses to let end devices customize the selection strategies among carriers.

9 Conclusion

The current design of cellular networks limits the device's ability to fully explore multi-carrier access. The fundamental problem is that, existing 3G/4G mobile networks place most decisions and operational complexity on the infrastructure side. This network-centric design is partly inherited from the legacy telecom-based architecture paradigm. As a result, the increasing capability of user devices is not properly exploited. In the multi-carrier access context, devices may suffer from low-quality access while incurring unnecessary service disruption. In this work, we describe *iCellular*, which seeks to leverage the fine-grained cellular information and the available mechanism at the device. It thus dynamically selects better mobile carrier through adaptive monitoring and online learning. Our initial evaluation validates the feasibility of this approach.

Acknowledgments: We thank the anonymous reviewers for their constructive comments. We greatly appreciate our shepherd, Dr. Matt Welsh for his continuous and timely guidance to improve the work. We also appreciate Dr. Richard Liu and the Project Fi team for their valuable feedback. This work is supported in part by NSF awards (CNS-1423576, CNS-1421440, CNS-1526456, and CNS-1526985).

References

[1] Dual sim phone. https://en.wikipedia.org/wiki/Dual_SIM.

[2] Enabling diagnostic mode in mobileinsight. http://metro.cs.ucla.edu/mobile_insight/diag_reference.html.

[3] Go power master. https://play.google.com/store/apps/details?id=com.gau.go.launcherex.gowidget.gopowermaster&hl=en.

[4] Mobileinsight project. http://metro.cs.ucla.edu/mobile_insight.

[5] 3GPP. Lte ue category. http://www.3gpp.org/keywords-acronyms/1612-ue-category.

[6] 3GPP. TS23.107: Quality of Service (QoS) concept and architecture.

[7] 3GPP. TS27.007: AT command set for User Equipment (UE), 2011.

[8] 3GPP. TS25.304: User Equipment (UE) Procedures in Idle Mode and Procedures for Cell Reselection in Connected Mode, 2012.

[9] 3GPP. TS36.331: Radio Resource Control (RRC), 2012.

[10] 3GPP. TS24.301: Non-Access-Stratum (NAS) for EPS; , Jun. 2013.

[11] 3GPP. TS36.304: User Equipment Procedures in Idle Mode, 2013.

[12] 3GPP. TS23.122: Non-Access-Stratum (NAS) functions related to Mobile Station (MS) in idle mode, 2015.

[13] V. Agababov, M. Buettner, V. Chudnovsky, M. Cogan, B. Greenstein, S. McDaniel, M. Piatek, C. Scott, M. Welsh, and B. Yin. Flywheel: Google's Data Compression Proxy for the Mobile Web. In *USENIX NSDI*, 2015.

[14] Apple. Apple SIM for iPad. https://www.apple.com/ipad/apple-sim/.

[15] N. Armstrong. Network handover in google fi, 2015. http://nicholasarmstrong.com/2015/08/network-handover-google-fi/.

[16] A. Balasubramanian, R. Mahajan, and A. Venkataramani. Augmenting Mobile 3G Using WiFi. In *ACM MobiSys*, 2010.

[17] R. Copeland and N. Crespi. Resolving ten MVNO issues with EPS architecture, VoLTE and advanced policy server. In *IEEE International Conference on Intelligence in Next Generation Networks (ICIN)*, pages 29–34, 2011.

[18] X. Costa-Pérez, J. Swetina, T. Guo, R. Mahindra, and S. Rangarajan. Radio Access Network Virtualization for Future Mobile Carrier Networks. *IEEE Communications Magazine*, 51(7):27–35, 2013.

[19] S. L. Crawford. Extensions to the cart algorithm. *International Journal of Man-Machine Studies*, 31(2):197–217, 1989.

[20] S. Deb, K. Nagaraj, and V. Srinivasan. MOTA: Engineering an Operator Agnostic Mobile Service. In *ACM MobiCom*, pages 133–144, 2011.

[21] S. Deng, R. Netravali, A. Sivaraman, and H. Balakrishnan. WiFi, LTE, or Both? Measuring Multi-Homed Wireless Internet Performance. In *ACM IMC*, pages 181–194, 2014.

[22] P. Di Francesco, F. Malandrino, and L. A. DaSilva. Mobile Network Sharing Between Operators: A Demand Trace-driven Study. In *ACM CSWS*, 2014.

[23] S. Dimatteo, P. Hui, B. Han, and V. O. K. Li. Cellular Traffic Offloading through WiFi Networks. In *IEEE MASS*, 2011.

[24] Engadget. Apple and Samsung in talks to adopt e-SIM technology. http://www.engadget.com/2015/07/16/apple-samsung-e-sim/.

[25] Google. Youtube android player api. https://developers.google.com/youtube/android/player/reference/com/google/android/youtube/player/YouTubePlayer.

[26] Google. Project fi, 2015. https://fi.google.com/about/.

[27] S. Jelassi, G. Rubino, H. Melvin, H. Youssef, and G. Pujolle. Quality of Experience of VoIP Service: A Survey of Assessment Approaches and Open Issues. *IEEE Communications Surveys & Tutorials*, 14(2):491–513, 2012.

[28] M. Jokinen, M. Mäkeläinen, and T. Hänninen. Demo: Co-primary Spectrum Sharing with Inter-operator D2D Trial. In *ACM MobiCom*, 2014.

[29] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan. NVS: A Substrate for Virtualizing Wireless Resources in Cellular Networks. *IEEE/ACM Transactions on Networking (TON)*, 20(5):1333–1346, 2012.

[30] S. Kumar, E. Hamed, D. Katabi, and L. Erran Li. LTE Radio Analytics Made Easy and Accessible. In *ACM SIGCOMM*, pages 211–222, 2014.

[31] MathWorks. Variable importance in regression tree. http://www.mathworks.com/help/stats/compactregressiontree.predictorimportance.html.

[32] R. K. Mok, E. W. Chan, and R. K. Chang. Measuring the Quality of Experience of HTTP Video Streaming. In *IFIP/IEEE Integrated Network Management (IM)*, pages 485–492, 2011.

[33] Mozilla. Remotely debugging firefox for android. https://developer.mozilla.org/en-US/docs/Tools/Remote_Debugging/Firefox_for_Android.

[34] NGMN. Ngmn 5g white paper. https://www.ngmn.org/work-programme/5g-initiative/.

[35] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure. Exploring Mobile/WiFi Handover with Multipath TCP. In *ACM CellNet*, 2012.

[36] J. S. Panchal, R. Yates, and M. M. Buddhikot. Mobile Network Resource Sharing Options: Performance Comparisons. *IEEE Transactions on Wireless Communications*, 12(9):4470–4482, 2013.

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

[38] S. Rosen, H. Luo, Q. A. Chen, Z. M. Mao, J. Hui, A. Drake, and K. Lau. Discovering Fine-grained RRC State Dynamics and Performance Impacts in Cellular Networks. In *ACM MobiCom*, pages 177–188, 2014.

[39] A. Salkintzis, M. Hammer, I. Tanaka, and C. Wong. Voice Call Handover Mechanisms in Next-Generation 3GPP Systems. *Communications Magazine, IEEE*, 47(2):46–56, 2009.

[40] K. E. Suleiman, A.-E. M. Taha, and H. S. Hassanein. Understanding the interactions of handover-related self-organization schemes. In *Proceedings of the 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '14, pages 285–294, 2014.

[41] G. Tu, C. Peng, H. Wang, C. Li, and S. Lu. How Voice Calls Affect Data in Operational LTE Networks. In *MobiCom*, Oct. 2013.

[42] G.-H. Tu, C. Peng, C.-Y. Li, X. Ma, H. Wang, T. Wang, and S. Lu. Accounting for Roaming Users on Mobile Data Access: Issues and Root Causes. In *ACM MobiSys*, 2013.

[43] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *USENIX NSDI*, 2011.

[44] Y. Zaki, L. Zhao, C. Goerg, and A. Timm-Giel. Lte mobile network virtualization. *Springer Mobile Networks and Applications*, 16(4):424–432, 2011.

[45] H. Zhang, X. Wen, B. Wang, W. Zheng, and Y. Sun. A novel handover mechanism between femtocell and macrocell for lte based networks. In *Proceedings of the 2010 Second International Conference on Communication Software and Networks (ICCSN)*, 2010.

Appendices

A An Example of iCellular APIs

We use a simple example to illustrate how they work. Consider a device who has access to T-Mobile and Sprint 3G/4G networks, and would like to choose in the network with minimal radio link latency. To do so, the device first initiates an *active monitor*, and specifies the list of the carrier networks s/he is interested in:

```
monitor = Monitor(["T-4G","T-3G","S-4G"]);
```

To choose the target carrier network, the user may want to learn each network's performance. The following code shows how user can initiate a latency predictor:

```
predictor = Predictor("Latency");
```

To let devices make responsive decisions, *iCellular* let selection strategy be triggered by the latest and even partial search results. To do so, the device should overload an *event-driven* decision callback function. Devices are given the runtime monitoring results of available carrier networks. Optionally, the device can use the predictor to help determine the target carrier network. The device can call `SwitchTo()` function to perform the switch. The following code shows a strategy that minimizes latency:

```
def decision_callback(monitor):
  min_latency = inf; target = null;
  for network in monitor:
    latency = predictor.predict(network);
    if latency < min_latency:
      min_latency = latency;
      target = network;
  SwitchTo(target);
```

B Collecting App-specific Performance

For SpeedTest, we directly record the downlink speed for each test. Note that Nexus 6 supports LTE category 4, which can yield up to 150Mbps downlink bandwidth in theory [5]. This is why we observe 40+Mbps downlink speed in our tests, which is much higher than most previous measurements. For Web, currently we use Firefox and get the web loading time from its debugging console [33]. For Youtube, we extract its buffering time by tracking `OnBuffer(True)` and `OnBuffer(False)` events from Youtube Android player API [25], and calculating the elapsed time in between, during which the user has to pause the video. For Skype, we collect round-trip latencies (in ms) as the performance metric. To get it, We enabled the `Technical info` panel in the Skype app, which shows the latency in the call. Then we record the round-trip latency in every second.