Device-Customized Multi-Carrier Network Access on Commodity Smartphones

Yuanjie Li, Chunyi Peng, Haotian Deng, Zengwen Yuan, Guan-Hua Tu, Jiayao Li, Songwu Lu, and Xi Li

***Abstract*— Accessing multiple carrier networks (T-Mobile, Sprint, AT&T, and so on) offers a promising paradigm for smartphones to boost its mobile network quality. However, the current practice does not achieve the full potential of this approach because it has not utilized fine-grained, cellular-specific domain knowledge. Our experiments and code analysis discover three implementation-independent issues: 1) it may not trigger the anticipated switch when the serving carrier network is poor; 2) the switch takes a much longer time than needed; and 3) the device fails to choose the high-quality network (e.g., selecting 3G rather than 4G). To address them, we propose iCellular, which exploits low-level cellular information at the device to improve multi-carrier access. 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 U.S. carriers: T-Mobile and Sprint. Our evaluation shows that, iCellular boosts the devices' throughput with up to 3.74× throughput improvement, 6.9× suspension reduction, and 1.9× latency decrement over the state of the art, with moderate CPU, and memory and energy overheads.**

***Index Terms*— 4G mobile communication, 5G mobile communication, multi-carrier network access, project Fi.**

I. Introduction

MOBILE Internet access has become an essential part of our daily life. 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 and future 5G), 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.

Besides the infrastructure-side upgrades, a promising alternative approach 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 industrial efforts have recently emerged to provide 3G/4G multi-carrier access via universal SIM card, including Google Project Fi [26], Apple SIM [14], Samsung e-SIM [24], Huawei Skytone [27], to name a few. The ongoing 5G desgins also seek to support multiple, heterogenous access technologies [40].

Unfortunately, our study shows that, the full benefits of multi-carrier access can be constrained by today's design. We examine Google Project Fi via experiments and code analysis. We discover three issues (§III): (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). All issues are independent of the implementations. Project Fi users have experienced these problems [15], but without knowing the causes or solutions.

It turns out that, these issues are not specific to the implementation of Project Fi. Instead, they are rooted in the conflicts between legacy mobile network design and device's multi-carrier access requests. With the single-carrier scenario in mind, the 3G/4G design chooses *"smart core dumb end"*. For this reason, it does not expose the low-level cellular information (*e.g.*, available candidate carriers, which carriers to scan in switch, and radio/QoS profile for each carrier) to OS on end device. In multi-carrier access, however, the end intelligence is a necessity, since individual carrier no longer has global view for high-quality decision. Without leveraging low-level cellular information on device, today's carrier selection does not fulfill the end intelligence of exploring multiple carriers.

While the problem may be solved by the future architecture redesign (say, 5G), it usually takes years to accomplish. Moreover, we observe that some ongoing 5G standardizations [11] inherit the inter-carrier switch mechanisms today, and may suffer from the same issues. Instead, we seek to devise a solution that works with the current (and future 5G) network and the ongoing industrial efforts. Specifically, we explore the following problem: *Can we leverage low-level cellular information and mechanisms at the device to further improve multi-carrier access?* Our study yields a positive answer.

Manuscript received October 2, 2017; revised May 18, 2018; accepted August 26, 2018; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Y. Yi. The work of Dr. Yuanjie Li, Mr. Zengwen Yuan and Dr. Songwu Lu was partially supported by NSF grants 1423576 and 1526985. The work of Dr. Chunyi Peng is partially supported by NSF grants 1749049, 1421440, and 1750953. Dr. Xi Li is the corresponding author, and acknowledges the funding support of NSFC grants 61772482 and 61272131. Dr. Yuanjie Li is the student author. *(Corresponding author: Yuanjie Li.)*

Y. Li, Z. Yuan, J. Li, and S. Lu are with the University of California at Los Angeles, Los Angeles, CA 90095 USA (e-mail: yuanjie.li@cs.ucla.edu; zyuan@cs.ucla.edu; likayo@ucla.edu; slu@cs.ucla.edu).

C. Peng and H. Deng are with Purdue University, West Lafayette, IN 47907 USA (e-mail: chunyi@purdue.edu; deng164@purdue.edu).

G.-H. Tu is with Michigan State University, East Lansing, MI 48824 USA (e-mail: ghtu@msu.edu).

X. Li is with the University of Science and Technology of China, Hefei 230000, China (e-mail: llxx@ustc.edu.cn).

Digital Object Identifier 10.1109/TNET.2018.2869492

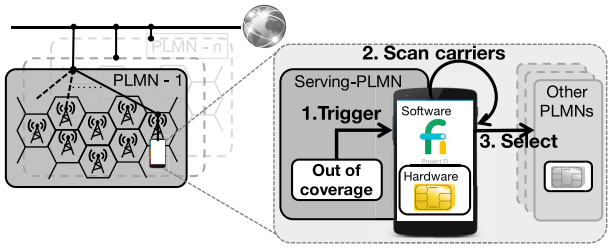


Fig. 1. Multi-carrier network access (left) and inter-carrier switch via PLMN selection (right).

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 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 (with offline bootstrap) 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 phones (Nexus 6, Nexus 6P and Pixel). 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.

II. Mobile Network Access Primer

A carrier 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 each PLMN and across several PLMNs (*e.g.*, 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 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 Figure 1. If 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 [10], 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, the PLMN selection takes three steps: *(1) Trigger.* The procedure is invoked when the home network service is no longer available (*e.g.* out of coverage); *(2) Scanning.* The mobile device scan the available carriers' spectrums and radio quality. The availability is usually determined by their radio quality (*e.g.*, the signal strength larger than one threshold). *(3) Select.* The selection of carriers is based on pre-defined criteria or user manual operation. In the automatic mode, the available PLMN with the highest priority will be selected. The PLMN selection will then stop the scanning and change the carrier. In the manual mode, the subscriber is provided the list of all available carriers for the decision. If the device decides to switch, it will deregister from the current carrier network and then register to a new one. In this process, the data/voice contexts (*e.g.* IP address, QoS configuration) has to change accordingly. The network access may thus 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 such SIM card, the device can access multiple cellular carriers (*e.g.*, T-Mobile, Sprint and US Cellular in Project Fi). This is achieved by maintaining multiple cellular carriers' profiles (including the identity, security keys, and radio parameters) inside the SIM card. Given only one cellular interface, the device uses one carrier at a time, *i.e.* only one cellular carrier profile can be activated. Similar to roaming, the switch between carriers is based on the PLMN selection inside the hardware interface. The inter-carrier switch decisions can be controlled by the software logic, as we will show below.

III. Multi-Carrier Access: Promises & Issues

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

Methodology: We take a two-step approach to studying the current multi-carrier access. We first conduct an empirical study to unveil the benefits and limitations of the multi-carrier access in reality. Then to understand the root causes of these limitations, we perform the code analysis of Google Project Fi [26], a popular multi-carrier access solution since 2015. We decipher its phone-side app, uncover its switch mechanisms and algorithms, and validate that these limitations are independent of specific implementations.

○ *Empirical study:* We conduct both controlled experiments and a two-month user study using two Nexus 6 phones with Google Project Fi [26], which was released in May 2015. Project Fi provides access to three carriers (T-Mobile, Sprint and U.S. Cellular) in U.S. and covers 135 countries around the world. It develops a proprietary mechanism on commodity phones for the automatic carrier selection. We contacted Project Fi team and learned that such mechanism aims to optimize user experience, and considers network performance, battery usage and data activity during selection. We validate this by demystifying Project Fi's implementation (§III-B).

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 within the campus buildings in Los Angeles (west coast) and Columbus (Midwest) 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).[1] We run each test 10 times

[1]U.S. Cellular is not available in our areas.

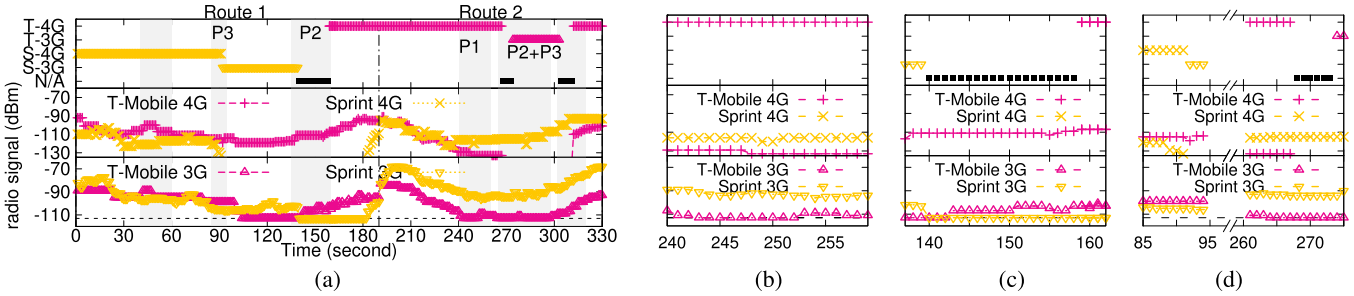


Fig. 2. An example log for serving carriers and networks and three problematic instances through Project Fi. (a) An example log over two walking routes. (b) P1: no switch. (c) P2: disruption. (d) P3: unwise switch.

and similar results are consistently observed in all the tests. In the user study (07/31/15–09/02/15 and 09/01/17–09/30/17), we use the Project Fi-enabled phones (Nexus 6/6p and Pixel) as usual and collect background device and cellular events with MobileInsight, an in-phone cellular monitoring tool [36]. We have collected 10.0GB logs with MobileInsight in total, with 1,250,596 messages from radio resource control (RRC), 217,971 messages from mobility management (MM), and 39,358 messages from session management (SM). In §III-A, we present the results from the controlled experiments as motivating examples. The user study to be described in §IV and §VI confirms that these issues are common in practice.

◦ *Code Analysis:* The Project Fi's phone-side app is not open sourced. To analyze it, we decipher its installation package that is readily available inside the phones (in /system/app/Tycho/Tycho.apk).[2] We decompile it into Java code using smali/baksmali [30]. Note that this code has been obfuscated as anonymous variables, classes, and member methods. To this end, we annotate the code as follows. We collect Project Fi's logs (from Android logcat) by running it under the same settings as the experiments, compare the log information with the debugging messages inside the decompiled source code, and map the corresponding code to Project Fi's solution components. Our analysis is based on Project Fi version J.2.5.18, which includes 91,438 lines of code after the decompilation.

A. Motivating Examples

Merits of Multi-Carrier Access: We first verify that exploiting multiple carriers is indeed beneficial to service availability and 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 at locations with 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 improves data access throughput and user experiences. Project Fi is indeed promising to improve the mobile Internet quality.

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

[2]Tycho is the internal codename of Google Project Fi.

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 7 for the user-study results). In the example of Figure 2c, the phone starts Sprint→T-Mobile roaming at the 140th second since it leaves Sprint's 3G coverage (≤ -116dBm according to [6]), 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 [47]). 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 to migrate to the better choice, thus is 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

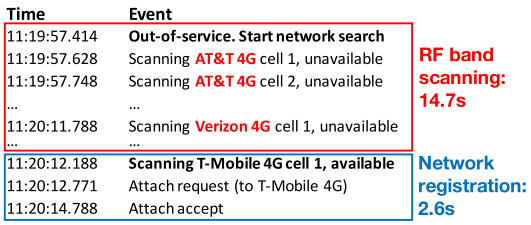


Fig. 3. Event logs during P2 (disruption) of Fig. 2c.

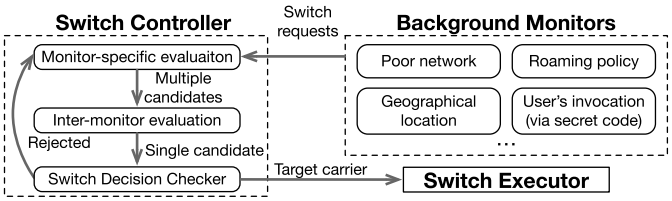


Fig. 4. Google Project Fi's multi-carrier access app.

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.

B. Demystifying Google Project Fi

We next perform the code analysis of Google Project Fi. We show its key mechanisms and switch logics, and demonstrate why it cannot address the limitations P1–P3 in §III-A.

Project Fi provides inter-carrier switching using a phone-side system app (overviewed in Figure 4). It has three major components: A pool of background monitors, a central switch decision controller, and a switch executor. Each monitor tracks specific Android-level events, and triggers a switch request when certain criteria is satisfied (details below). The controller accepts asynchronous switch requests from all monitors, and runs its decision logic to determine the target carrier. Then it invokes the switch executor to perform the inter-carrier switch.

Concurrent Background Monitors: Project Fi evaluates the serving carrier networks based on multiple factors, such as the network quality, geographical locations, and international roaming policies. To this end, it runs multiple background monitors to track these information. These monitors work asynchronously, and collect the carrier information mainly through the public, OS-level APIs. For example, to evaluate the serving carrier network's qualities, Project Fi's poor network monitor queries the network status from Android's `ConnectivityManager` [13]. It issues a switch request when the serving carrier network is no longer accessible.

Three-Phase Decision Logic: Upon receiving the switch requests, the controller runs a three-phase logic to determine the target carrier network. First, it applies monitor-specific criteria to accept (or reject) switch requests from each monitors. Consider the example of the poor network monitor.

TABLE I

PREDEFINED RATING IN THE POOR NETWORK MONITOR

Network Type	Operator	Rating	Lockdown Timer (s)
4G LTE	Any	1000	Inf
3.5G	T-Mobile	900	1800
	Sprint/US Cellular	800	1500
3G	T-Mobile	700	1200
	Sprint/US Cellular	600	900
2.5G	T-Mobile	400	300
	Sprint/US Cellular	300	300
2G	T-Mobile	200	120
	Sprint/US Cellular	100	120

TABLE II

PRIORITIES FOR SOME OF PROJECT FI'S MONITORS

Monitor Name	Switch Request Priority
User's manual invocation	1002
Geographical location	950
Poor network	900
Country policy	50

The controller pre-defines the rating of each carrier network, as shown in Table I. It seeks to switch to another network if the current one does not have the highest rating. Upon receiving the request to switch to another carrier, the switch controller cannot determine if the candidate carrier network has a better rating, since the Android APIs cannot gather information on available carrier networks. To this end, the controller uses a trial-and-error approach: It first invokes the switch (details below), and tests if this candidate carrier network has higher rating according to Table I. If not, the controller will block this carrier network, and try the next one in the next round of selection. To prevent excessive switches, a lockdown timer is adopted for the serving carrier network. The controller will not accept the next switch request until the timer expires.

Second, given the switch requests from multiple monitors, the controller selects the single target carrier. Note that these monitors run independently, and may raise requests with conflicting carrier networks. Project Fi resolves the conflicts by predefining the priorities for the monitors (Table I), and accepting the switch request with the highest priority.

Last, to prevent dirsupting the ongoing network services, Project Fi performs the runtime checking. It will pend the switch request if the phone is making a voice call, or sending/receiving the cellular data. It will reject the switch request if (1) there is already an inter-carrier switch in progress; or (2) automatic switching is disabled. Otherwise, the switch request will be approved and forwarded to the switch executor.

Switch Execution: Project Fi performs the inter-carrier switch using the standard APIs in Android. It first de-registers from the serving carrier network. Next, Project Fi reconfigures the SIM card so that the device can be recognized by the target carrier network. By calling a kernel-space daemon, it deactivates the old carrier's SIM-card profile, and activates of the new-carrier's profile. Afterwards, it re-enables the cellular service. The underlying cellular modem will be responsible for scanning the cells and registering to the target carrier network.

Limitations: While Project Fi has made extensive efforts to fulfill the potentials of multi-carrier access using Android APIs, we next show that it cannot fully solve P1–P3.

The fundamental problem is that, it does not leverage the low-level cellular information or mechanisms. Note that such limitation is not specific to Project Fi's implementations. Instead, *any* implementations that leverage the OS-level information and mechanisms only will have the same limitation.

○ *P1: No anticipated inter-carrier switch.* To perform the anticipated switch, the device should keep monitoring other carriers even when the serving carrier is still available. Unfortunately, Project Fi cannot achieve this: Its poor network monitor relies on Android's `ConnectivityManager` only, which does not offer this capability. In fact, Android does not provide APIs to perform proactive carrier monitoring; such action can be realized by the low-level hardware (modem).

○ *P2: Long switch time and service disruption.* To reduce the switch time, the device should refrain from exhaustive search of all carriers at all times. This can only be controlled by the underlying cellular mechanisms. Unfortunately, Project Fi's switch executor relies on the standard PLMN selection (§II). Since PLMN selection scans all available carriers, it cannot prevent the exhaustive search.

○ *P3: Unwise decision and unnecessary performance degradation.* To make a wise selection, the device should treat all intra/inter-carrier switches equally, collect information from available carriers, and select the best carrier network. However, Project Fi's three-phase decision logic can only control the *inter-carrier* switch; intra-carrier switch is still controlled by the low-level cellular mechanisms (handoffs). Moreover, it lacks sufficient information from available carriers to determine the best carrier network. While its trail-and-error approach can help collect some information, it is at the cost of more switches and thus service disruptions.

C. Insights

The above examples and code analysis also shed lights on how to solve the three problems. The key is to leverage low-level cellular information and mechanisms at the device.

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 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 may lead to temporary disconnection from the current carrier network. We elaborate on how we address them in §IV-A.

To reduce the switch time (P2), the device should refrain from exhaustive search of all carriers. This requires the 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.

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 "smart core, dumb end" paradigm 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.

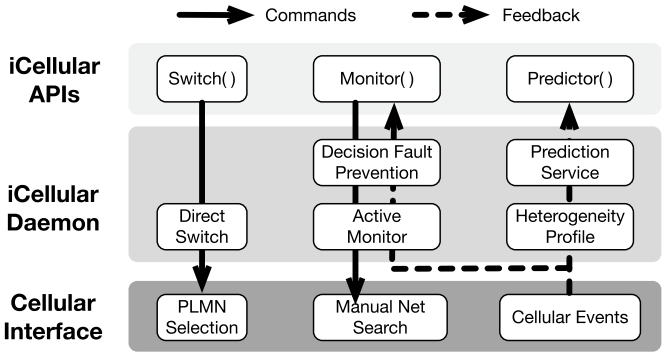


Fig. 5. iCellular system architecture.

IV. 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 §III-A.

For incremental deployability, iCellular is built on top of the PLMN selection [9], [10], a standardized mechanism mandatory on all phones. Note that, however, the basic PLMN selection suffers from similar issues in §III-A: 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 carrier 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 5 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 [5]. 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 (§IV-A and §IV-B). To facilitate the devices to make wise decisions, iCellular

TABLE III
CELLULAR EVENTS USED IN ICELLULAR

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

offers cross-layer online learning service to predict network performance (§IV-C), and protects devices from decision faults (§IV-D). To enable adaptation, prediction and decision fault prevention, iCellular incorporates realtime feedbacks extracted from low-level cellular events. Different from approaches using mobile OS APIs, we leverage the in-phone diagnostic port to collect the runtime cellular events for iCellular (Table III). These components are designed to be scalable, without incurring heavy signaling overhead to the device and network.

A. 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 §III). For this purpose, the only viable mechanism on commodity phones is the manual network search [10]. 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. For incremental deployability, we realize the active monitoring atop 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 app-level information only. iCellular thus devises cross-layer adaptions.

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 when the device has re-synchronized to other carriers. If so, its reception could be delayed or even lost.

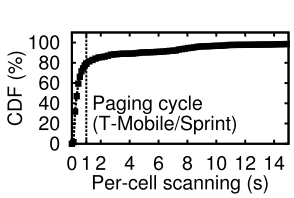


Fig. 6. Cell scan time.

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 [7], [43]). 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 6 shows our 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.

A valid issue is that, the monitoring results may become obsolete due to continuous data transmissions, thus leading to wrong decisions. It 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 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.

Minimal Search: Instead of exhausting all carriers, iCellular scales the monitoring by restricting the manual search only to device-specified carriers. The issue is that no such option is available in the manual network search mechanism. We thus adapt the PLMN preference. Given the list of carrier networks of interests, iCellular configures the cellular interface to let the manual search scan these carriers first. This is achieved by assigning them with the highest PLMN preferences. In 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 search. In this way, at most one more cell would be scanned than user-specified ones (lower-bound):

$$n_{search,min} \leq n_{search,\text{iCellular}} \leq n_{search,min} + 1 \qquad (1)$$

where $n_{search,min}$ is the minimal number of cells to scan, and $n_{search,\text{iCellular}}$ counts the cell scanned by iCellular.

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. iCellular thus allows devices to make decisions with partial results, thus further accelerating the process. Instead of waiting for all scanning results, iCellular triggers the decision whenever new results are available.

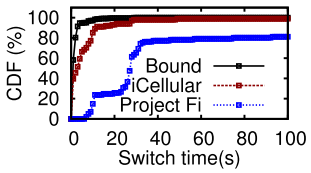


Fig. 7. Switch time.

B. 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* (§III). 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. This way, the device would directly switch 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 de-registration from the old network (detach), and registration to the new network (attach). According to [8], 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. The switch time is thus

$$T_{switch,\text{iCellular}} = n_t T_t + T_{attach} \approx n_t T_t + T_{switch,min} \quad (2)$$

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 7 verifies this with our background monitoring results in Project Fi. It shows that, iCellular indeed approximates the lower bound, despite this minor overhead.

C. Prediction for Heterogeneous Carriers

To decide which carrier network to switch to, the device may gather the information on each carrier network. Ideally, the device needs to measure every available carrier network's current performance (*e.g.*, latency or throughput). 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 a combination of offline [19] and online [28] regression tree with domain-specific optimizations. 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 (detailed below). Note that radio measurement alone is insufficient to predict performance, because different carriers may apply heterogeneous radio technologies and resource configurations. Our solution works as follows.

TABLE IV
HETEROGENEOUS CELLULAR NETWORK PROFILES

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%

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 in [35] 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 §VI). This is because the performance characteristics of a carrier network tend to have consistent impacts on all applications.

Feature 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 measurement 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 (§IV-A). For the network profile, iCellular currently collects two types (Table IV): (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 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 IV lists the predictability of some parameters. 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 parameters, with minor tunings to each base station/controller. We thus only store a set of unique values, and reuse it for all the applicable samples until changes are found.

Training and Prediction Over Streaming Samples: To train the predictor, iCellular combines the offline training (for bootstrap) with online update using streaming data. The predictor is represented as a tree, with each interior node as a test condition over x (radio measurements or 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. The training and prediction work as follows.

◦ *Bootstrap with offline training.* For the cold start, iCellular pre-trains a regression tree as a basis. We extract a complete set of feature vector samples $S = \{(x, y)\}$ from our user-study dataset. Then we train the regression tree as follows:

1) We search one field $x_i \in x$ (radio measurement or profile) that maximizes the split of the data into two sub-sets S_1 and S_2 ($S_1 \cup S_2 = S$, $S_1 \cap S_2 = \emptyset$). The following function I quantifies the gain of split (based on samples' variance), and is used as the optimization function:

$$I = sd(S) - (sd(S_1) + sd(S_2)) \quad (3)$$

$$sd(S) = \frac{1}{|S|^2} \sum_{i \in S} \sum_{j \in S} \frac{1}{2}(y_i - y_j)^2 \quad (4)$$

Then we create the root using x_i as the test condition.

2) After the split, we recursively apply step 1) to S_1 and S_2. In this way, we can construct two sub-trees and connect them to the root. The search stops when no further gain of splitting function I can be obtained.

◦ *Online prediction tree update.* To adapt to the temporal/spatial dynamics of the carrier network behaviors, iCellular takes the new samples of radio measurements and carrier profiles to improve its prediction accuracy over time. This is performed in parallel with the normal usages, thus not delaying iCellular's functionalities. While there exists large body of work training the prediction tree in a batched, offline fashion, iCellular incrementally updates the regression tree based on the runtime observations. It does not require to store the historical samples, thus saving the memories inside the phone.

When iCellular detects a new sample (x, y), it feeds the sample into the predictor and runs two-phase processing:

1) *Drift detection and model adaptation:* The new sample may deviate from the existing predictions. To this end, iCellular detects the drift and incrementally update the sub-trees with high errors. It runs the existing predictor over new sample x, and obtains an estimated metric $y\prime$. We perform Page–Hinckley (PH) change detection test on the absolute error $|y - y\prime|$. If some parts of the prediction tree does not fit the new observed data well, the error on that part will surpass a given threshold. Such event will trigger our model adaptation. We build an alternate sub-tree for the region where drift is detected. New samples will be supplied to re-grow the alternate sub-tree for that region. Once the alternate tree provides better accuracy, it replaces the old part and hence the model is adapted.

2) *Predictor update:* Given the prediction tree model, iCellular further updates the predictors with the incoming samples. Similar to the bootstrap phase, iCellular runs every N new samples, and searches a new field (measurement or profile) that best splits the samples using the same criteria in (3). Given the new split, we create a new pair of leaves for this new field, and completes the update of the prediction tree. Note that, the choice of N affects how often the predictor may be updated. In fact, the following Hoeffding bounds applies: After N independent observations of a performance metric y with value range R, with confidence $1 - \delta$, the prediction error $\epsilon = |y - y\prime|$ is bounded as follows

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2N}} \quad (5)$$

Our current implementation chooses $N = 30$.

◦ *Online prediction.* Given the runtime observation x, iCellular predicts the performance metric y as follows. Starting from the root, it traverses the prediction tree based on the value of x, until it reaches the leaf. The leaf value will be reported as the predicted performance metrics. As one optimization, we observe that heterogeneity profile x_2 is highly predictable (Table IV). So iCellular further optimizes the algorithm by caching the branches whose parent node tests heterogeneity profiles. This reduces the computation overheads.

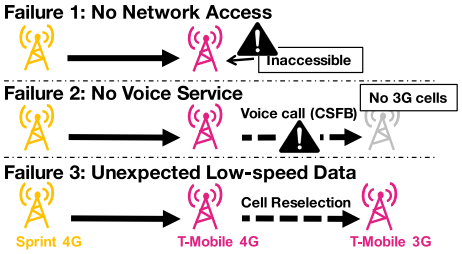


Fig. 8. Three types of improper switch decisions.

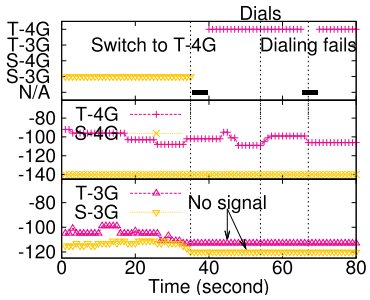


Fig. 9. Switch to a network with no voice support.

D. Decision Fault Prevention

Device-customized access strategy can be a double-edged sword. With improper strategies, the device may make faulty switch decisions and unexpectedly disrupt the service. Figure 8 shows three categories of failures caused by decision faults. All 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 9 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 [6]). In this scenario, the user in Sprint 4G should not switch to T-Mobile 4G, which cannot support voice calls without 3G.

Failure 3 (Unexpected Low-Speed Data Service): The user selection may not be honored by the individual carrier's handoff rules. Figure 10 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 re-selection [9]) 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.

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**

Fig. 10. Interplay of user/network's mobility.

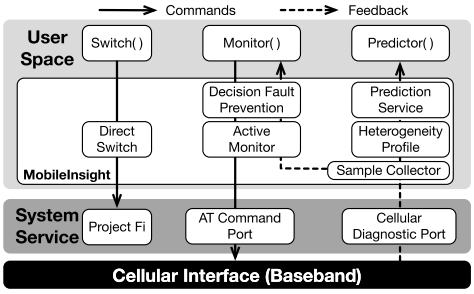


Fig. 11. Overview of iCellular implementation.

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 switching to these carrier networks. iCellular first profiles each carrier's low-level access-control list from the RRC SIB message [7], data/voice preference configuration from registration/location update messages [8], and the network-side mobility rules from the RRC configuration message [7], [9]. 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 list.

V. Implementation

We have implemented iCellular on Motorola Nexus 6, Huawei Nexus 6P, Google Pixel and Pixel 2. They run Android OS 5.1/6.0/7.1 using Qualcomm Snapdragon 805/810/821 chipsets. All these phone models 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 these phones. Figure 11 illustrates the system implementation. At the user space, 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 activate the diagnostic mode (`/dev/diag`) and AT-command interfaces (`/dev/smd11`).

Basic APIs: iCellular allows the device to control its cellular access strategies through three APIs: `Monitor()` for active monitoring (§IV-A), `Predictor()` for performance prediction (§IV-C) and `SwitchTo()` for direct switching (§IV-B). The decision fault tolerance is enabled by default (§IV-D). To customize its decision logic, the device can write a decision callback using these APIs. Appendix A in [35] presents an illustrative example on how to use them. Besides, 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 §VI for performance comparisons).

Runtime Access to Low-Level Cellular Information: As shown in §IV-A–§IV-D, 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. Table III summarizes the events required by iCellular, including the signaling messages exchanged between the device and the network, and radio quality/load measurements. Unfortunately, the mobile OS has very limited access to these information. To this end, we develop an in-phone solution MobileInsight [36] by exploiting the existing cellular diagnostic mode. We enable the diagnostic mode on the phone, redirect the cellular events from the diagnostic port to the memory, and finally expose them to iCellular.

Adaptive Active Monitoring (§IV-A): We implement `Monitor()` with manual search and adaptations. We leverage the standard AT command interface [5] that controls the basic phone functions and is readily available inside smartphones. Our prototype initiates the search with an AT query command `AT+COPS=?`. The non-disruption and minimal search adaptations are implemented using the events in Table III.

Adaptive Direct Switch (§IV-B): We implement the `SwitchTo()` on top of PLMN selection, with dynamic adaptations for direct switch. Ideally, this can be executed with the AT commands `AT+COPS=manual,carrier,network`. An exception is to switch to Sprint 3G, which uses the 3GPP2's EvDo/CDMA2000 technologies and disables this command. In this case, we 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 (`34777` for Sprint, `34866` for T-Mobile). Admittedly, this approach may incur extra switch overhead, but it is still acceptable (§VI-B).

Prediction for Heterogenous Carriers (§IV-C): We implement `Predictor()` in two steps. First, we implement the online sample collection, which extracts radio measurements, RRC configurations and QoS profiles as features from runtime cellular messages. We also define a callback to collect the network/application-level performance metrics. We then realize the regression tree algorithm for training and prediction. To bootstrap the online training, we have pre-trained a regression tree using the offline algorithm and our user-study traces.

Decision Fault Prevention (§IV-D): The fault prevention is implemented as a shim layer between the 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 also add a runtime checker in `SwitchTo()`, and prevent devices from selecting carriers not in the scanning results.

VI. Evaluation

We first present the overall performance improvement by iCellular (§VI-A), and then show iCellular satisfies various design properties in §IV (§VI-B). All experiments are conducted on commodity Nexus 6 phones in two cities of Los Angeles (west coast) and Columbus (Midwest), mainly around two campuses. The results on Nexsus 6P and Pixel are similar.

A. Overall Performance

We use four representative applications to assess iCellular: SpeedTest (bulk file transfer), Web (interactive latency for

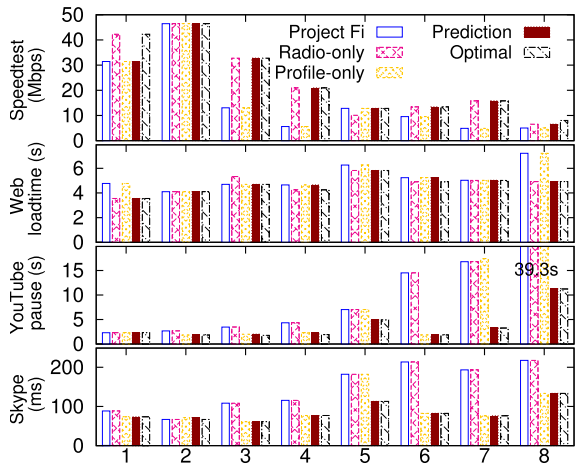


Fig. 12. Performance of Speedtest, Web, Youtube, Skype using various multi-carrier access schemes.

bursty traffic), Youtube (video streaming) and Skype (realtime VoIP). We evaluate each app with quality-of-experience metrics whenever possible, *i.e.*, downlink speed for SpeedTest, page-loading time for Web [12] (measured with Firefox), video suspension time for Youtube [38] (all videos with 720p quality, measured by its APIs), and latency for Skype [29] (measured with its tech info panel). The details to collect performance metrics are given in Appendix B in [35]. 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 (§III-B). 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 in idle, and then start to test each app. 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 (§V): (1) **Prediction-based:** the default strategy in iCellular, which chooses the carrier with the best ranking metric from the predictor §IV-C. The predictor is trained based on our 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 [9]. Whenever a network 4G with its signal strength higher than -110dBm (defined in [9]) 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 IV). For our iCellular strategies, we use the carrier list with all network types supported by Project Fi.

Figure 12 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.

TABLE V
STATISTICS OF ACCURACY TOWARD THE OPTIMALITY

	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 VI
WEIGHTS OF RADIO MEASUREMENT AND NETWORK PROFILES IN ICELLULAR'S PREDICTION STRATEGY

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

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 V shows the hit ratios of all schemes by different applications. iCellular 's prediction-based 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 [37]) and Table VI 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 13 and present the hit ratios and statistics of γ^+ in Table VII. 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 12). 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

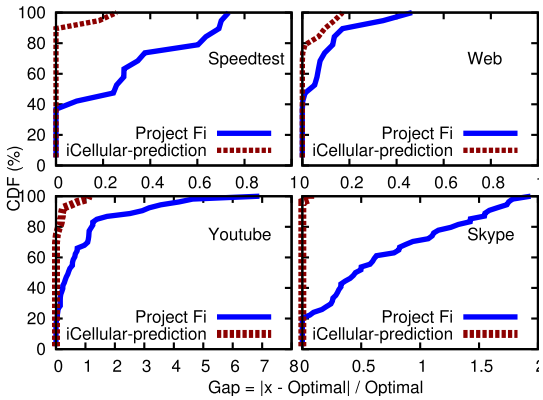


Fig. 13. The performance gaps from Project Fi and iCellular's prediction strategy to the optimality.

TABLE VII
PERFORMANCE GAPS FROM THE OPTIMAL ONE

	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

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, 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 the Built-In Strategies: iCellular's prediction strategy best approximates the optimal strategy. It outperforms radio-only and profile-only variants (§IV-C). We also see that, the importance of profile and radio measurements varies across applications. For example, our log analysis shows T-Mobile assigns Project Fi devices to the interactive traffic class (Table IV), which is optimized for delay-sensitive service [4].[3] 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 overhead, can be available for close-to-optimal performance.

B. Efficiency and Low Overhead

We show the micro-benchmark evaluations on iCellular's key components, and validate their efficiency. We examine the active monitoring, direct switch and fault prevention, and the overhead of signaling, CPU, memory and battery usage.

Efficiency: We examine iCellular's efficiency through two adaptive module tests. First, we show that, iCellular's adaptive monitoring is able to accelerate carrier scanning. We compare it with the default manual search, and record the total search time and number of cells scanned at 100 different locations. Figure 14 shows that adaptive search finishes 70% complete searches within 10s, 64% shorter than the exhaustive manual search. Note that devices are allowed to switch before the complete search (§V), so it waits shorter in practice. Figure 14b counts the scanned cells, and validates that such savings come from avoiding the unnecessary cell scans. The search time and 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 derived in (2) in §IV-B. Figure 15 shows that, iCellular saves 76.7% switch time on average, compared with Project Fi. Note that the current prototype has not achieved the minimal switch time: it 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 reconfiguration (mainly due to the I/O overhead to the external hard SIM). iCellular has to wait until the SIM card is reconfigured to switch to another carrier. In the experiments, we find that 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 faster SIM card (*e.g.* in-memory soft SIM), iCellular could save up to 96.1% of switch time compared with Project Fi.

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

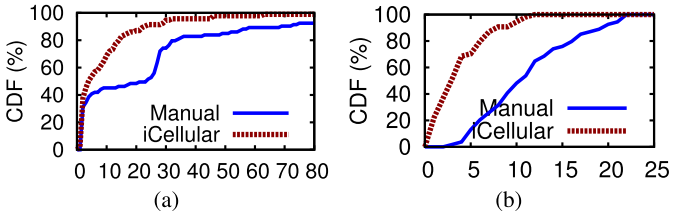


Fig. 14. iCellular's monitor avoids exhaustive search. (a) Total search time. (b) Cell count.

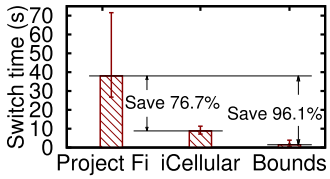


Fig. 15. Switch time.

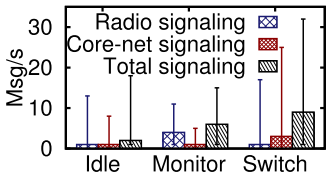


Fig. 16. Signaling cost.

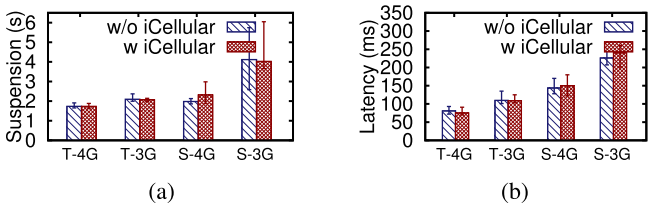


Fig. 17. iCellular's active monitoring has negligible impacts on data performance. (a) Youtube. (b) Skype.

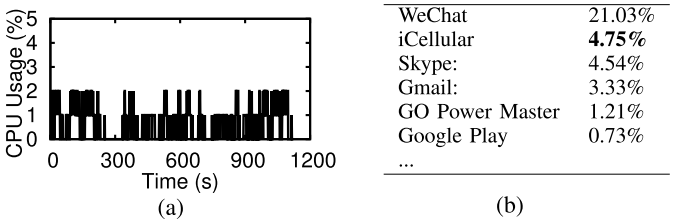


Fig. 18. CPU and battery usage of iCellular. (a) CPU usage. (b) Battery usage.

Fault Prevention: We next verify that iCellular handles fault scenarios and prevents devices from switching to unwise carrier networks. All three failure types in §IV-D have been observed in our user study. Note that the failure scenarios are less 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 9, 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 [46]. 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 10). We validate that iCellular can detect and avoid them.

Impact on Apps 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 17 shows the performance with/without iCellular's monitoring for Youtube and Skype. Enabling/disabling active monitoring has comparable application performance. As explained in §IV-A, this is because the carrier scanning procedure is performed only in the absence of traffic.

Signaling Overhead: iCellular incurs moderate signaling messages to the device and network. We record the device-side signaling message rate under three conditions (in 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-level (from RRC layer), core-network level (from mobility and session management layers) and the total signaling rate. Figure 16 shows that, the maximum observed signaling message rate is 32 message/sec (or 0.79 Kbps message volume equivalently). Such low overhead is because that, iCellular only relies on the control-plane cellular messages, whose volumes are relatively small by design.

CPU & Memory: In all our tests, the maximum CPU utilization is below 2%. The maximum memory usage is below 20 MB (including virtual memory). Figure 18a shows a 20-min log in a driving test: Its uses 16.45MB memory at maximum.

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 [2] to record energy consumption for each component/app. Figure 18b shows one record, where iCellular explicitly consumes about 4.75% of battery.

VII. Hints for Multi-Carrier Access in 5G

While iCellular is designed for incrementally deployable in 3G/4G, its solutions are instrumental to future 5G. We discuss some extensions in 5G that could benefit from iCellular, and how iCellular can be extended in the new design.

Working With Network-Side Assistance: iCellular can work in concert with network-side mechanisms for better performance. For example, during the inter-carrier switch, iCellular could benefit from the network-side downlink traffic buffering and tunneling for more seamless migration. For each 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.

Concurrent Access to Multiple Carriers: Due to the hardware limitations, iCellular and existing solutions support access to one carrier network each time. In long term, it is still preferred to access multiple carriers simultaneously to aggregate available carriers' connectivity. We find that this is achievable with the phone-side hardware modifications. The solution is to adapt the sleep mode (called discontinuous reception in 4G [7]). To save the energy, the carrier network can instruct the device to go to the sleep mode (by configuring the timers for the DRX) when there is no active data delivery. In this case, the device can switch to other carriers to send/receive data, and switch back to the old carrier when the timer expires.

The concurrent access mechanism can benefit from our iCellular's design experience. For example, the adaptive monitoring (§IV-A) can be incorporated with it to detect the better carrier networks. The heterogeneity predictor (§IV-C) and decision fault prevention (§IV-D) are also directly applicable to determine whether it is worth using other carrier networks.

On Google Project Fi: Project Fi makes pioneering efforts toward multi-carrier access in commodity phones. It explores intelligent device-side inter-carrier switch by leveraging various system-level information. In this work, we show that to fully address the problems in §III, the low-level cellular information is critical for device-side decision. iCellular provides

an alternative dimension to improve Project Fi's efficiency with the cross-layer cellular knowledge.

VIII. Related Work

In recent years, exploiting multiple cellular carriers attracts research efforts. These efforts span on both network and device sides. The network-side efforts include sharing the radio resource [22], [31], [42] and infrastructure [17], [18], [33], [49] between carriers, which helps to reduce deployment cost. On the device side, both 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 accesses on the device. Similar efforts use multiple physical interfaces from WiFi and cellular, including WiFi offloading [16], [21], [23], P2P interfaces between devices using different carriers [34], and multipath-TCP [41], [48]. These efforts have been partially standardized by IETF [32]. iCellular differs from all these in that it still uses a single cellular interface. Similar issues may also occur with traditional handoffs within a single carrier [44], [45], [50], which are caused by the network-side problematic management. Instead, iCellular targets inter-carrier migration, and chooses to let end devices customize the selection strategies among carriers.

IX. Conclusion

The current design of cellular networks limits the device's ability to fully explore multi-carrier access. The fundamental problem is that, existing 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 end intelligence is limited: Without runtime, fine-grained cellular information, the device cannot properly exploit the multi-carrier access. 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. We hope our study could stimulate more research efforts in the multi-carrier access, and provide valuable input into the standardization process of the upcoming 5G.

Acknowledgments

The authors would like to thank anonymous reviewers for their constructive comments.

References

[1] *Dual SIM Phone*. Accessed: Sep. 2018. [Online]. Available: https://en.wikipedia.org/wiki/Dual_SIM

[2] *GO Power Master*. Accessed: Sep. 2015. [Online]. Available: https://play.google.com/store/apps/details?id=com.gau.go.launcherex.gowidget.gopowermaster&hl=en

[3] 3GPP. *LTE UE Category*. Accessed: Sep. 2018. [Online]. Available: http://www.3gpp.org/keywords-acronyms/1612-ue-category

[4] *Quality of Service (QoS) Concept and Architecture*, document TS23.107, 3GPP, 2002.

[5] *AT Command Set for User Equipment (UE)*, document TS27.007, 3GPP, 2011.

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

[7] *Radio Resource Control (RRC)*, document TS36.331, 3GPP, 2012.

[8] *Non-Access-Stratum (NAS) for EPS*, document TS24.301, 3GPP, Jun. 2013.

[9] *User Equipment Procedures in Idle Mode*, document TS36.304, 3GPP, 2013.

[10] *Non-Access-Stratum (NAS) Functions Related to Mobile Station (MS) in Idle Mode*, document TS23.122, 3GPP, 2015.

[11] *5G NR; User Equipment (UE) Procedures in Idle Mode*, document Ts38.304, 3GPP, Sep. 2017.

[12] V. Agababov *et al.*, "Flywheel: Google's data compression proxy for the mobile Web," in *Proc. USENIX NSDI*, 2015, pp. 367–380.

[13] Android. *Connectivitymanager*. Accessed: Sep. 2018. [Online]. Available: https://developer.android.com/reference/android/net/ConnectivityManager.html

[14] Apple. *Apple SIM for iPad*. Accessed: Sep. 2018. [Online]. Available: https://www.apple.com/ipad/apple-sim/

[15] N. Armstrong. (2015). *Network Handover in Google Fi*. [Online]. Available: http://nicholasarmstrong.com/2015/08/network-handover-google-fi/

[16] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3G using WiFi," in *Proc. ACM MobiSys*, 2010, pp. 209–222.

[17] R. Copeland and N. Crespi, "Resolving ten MVNO issues with EPS architecture, VoLTE and advanced policy server," in *Proc. IEEE Int. Conf. Intell. Next Gener. Netw. (ICIN)*, Oct. 2011, pp. 29–34.

[18] X. Costa-Perez, J. Swetina, T. Guo, R. Mahindra, and S. Rangarajan, "Radio access network virtualization for future mobile carrier networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 27–35, Jul. 2013.

[19] S. L. Crawford, "Extensions to the CART algorithm," *Int. J. Man-Mach. Stud.*, vol. 31, no. 2, pp. 197–217, 1989.

[20] S. Deb, K. Nagaraj, and V. Srinivasan, "MOTA: Engineering an operator agnostic mobile service," in *Proc. ACM MobiCom*, 2011, pp. 133–144.

[21] S. Deng, R. Netravali, A. Sivaraman, and H. Balakrishnan, "WiFi, LTE, or both?: Measuring multi-homed wireless Internet performance," in *Proc. ACM IMC*, 2014, pp. 181–194.

[22] P. Di Francesco, F. Malandrino, and L. A. DaSilva, "Mobile network sharing between operators: A demand trace-driven study," in *Proc. ACM CSWS*, 2014, pp. 39–44.

[23] S. Dimatteo, P. Hui, B. Han, and V. O. K. Li, "Cellular traffic offloading through WiFi networks," in *Proc. IEEE MASS*, Oct. 2011, pp. 192–201.

[24] Engadget. *Apple and Samsung in Talks to Adopt E-SIM Technology*. Accessed: Jul. 2015. [Online]. Available: http://www.engadget.com/2015/07/16/apple-samsung-e-sim/

[25] Google. *YouTube Android Player API*. Accessed: Sep. 2015. [Online]. Available: https://developers.google.com/youtube/android/player/reference/com/google/android/youtube/player/YouTubePlayer

[26] Google. (2015). *Project Fi*. [Online]. Available: https://fi.google.com/about/

[27] Huawei. *Skytone*. Accessed: Sep. 2018. [Online]. Available: http://skytone.vmall.com/

[28] E. Ikonomovska, J. Gama, and S. Džeroski, "Learning model trees from evolving data streams," *Data Mining Knowl. Discovery*, vol. 23, no. 1, pp. 128–168, 2011.

[29] 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 Commun. Surveys Tuts.*, vol. 14, no. 2, pp. 491–513, 2nd Quart., 2012.

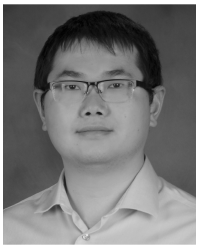
[30] JESUSFREKE. (2017). *Smali and Baksmali Assembler/Disassembler*. [Online]. Available: https://github.com/JesusFreke/smali

[31] M. Jokinen, M. Mäkeläinen, and T. Hänninen, "Co-primary spectrum sharing with inter-operator D2D trial," in *Proc. ACM MobiCom*, 2014, pp. 291–294.

[32] S. Kanugovi, S. Vasudevan, F. Baboescu, J. Zhu, S. Peng, and J. Mueller, "Multiple access management services," IETF, Tech. Rep. draft-kanugovi-intarea-mams-protocol-03, Mar. 2017.

[33] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan, "NVS: A substrate for virtualizing wireless resources in cellular networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1333–1346, Oct. 2012.

[34] J. Lee, K. Lee, Y. Kim, and S. Chong, "CarrierMix: How much can user-side carrier mixing help?" *IEEE Trans. Mobile Comput.*, vol. 16, no. 1, pp. 16–29, Jan. 2017.
[35] Y. Li *et al.*, "iCellular: Device-customized cellular network access on commodity smartphones," in *Proc. 13th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*. Santa Clara, CA, USA: USENIX Association, Mar. 2016, pp. 643–656.
[36] Y. Li *et al.*, "Mobileinsight: Extracting and analyzing cellular network information on smartphones," in *Proc. ACM 22nd Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)* New York, NY, USA, 2016, pp. 202–215.
[37] MathWorks. *Variable Importance in Regression Tree*. Accessed: Sep. 2018. [Online]. Available: http://www.mathworks.com/help/stats/compactregressiontree.predictorimportance.html
[38] R. K. P. Mok, E. W. W. Chan, and R. K. C. Chang, "Measuring the quality of experience of HTTP video streaming," in *Proc. IFIP/IEEE Integr. Netw. Manage. (IM)*, May 2011, pp. 485–492
[39] Mozilla. *Remotely Debugging Firefox for Android*. Accessed: Sep. 2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Tools/Remote_Debugging/Firefox_for_Android
[40] NGMN. *NGMN 5G White Paper*. Accessed: Feb. 2015. [Online]. Available: https://www.ngmn.org/work-programme/5g-initiative/
[41] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure, "Exploring mobile/WiFi handover with multipath TCP," in *Proc. ACM CellNet*, 2012, pp. 31–36.
[42] J. S. Panchal, R. D. Yates, and M. M. Buddhikot, "Mobile network resource sharing options: Performance comparisons," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4470–4482, Sep. 2013.
[43] S. Rosen *et al.*, "Discovering fine-grained RRC state dynamics and performance impacts in cellular networks," in *Proc. ACM MobiCom*, 2014, pp. 177–188.
[44] A. K. Salkintzis, M. Hammer, I. Tanaka, and C. Wong, "Voice call handover mechanisms in next-generation 3GPP systems," *IEEE Commun. Mag.*, vol. 47, no. 2, pp. 46–56, Feb. 2009.
[45] K. E. Suleiman, A.-E. M. Taha, and H. S. Hassanein, "Understanding the interactions of handover-related self-organization schemes," in *Proc. 17th ACM Int. Conf. Modeling, Anal. Simulation Wireless Mobile Syst. (MSWiM)*, 2014, pp. 285–294.
[46] G.-H. Tu, C. Peng, H. Wang, C. Y. Li, and S. Lu, "How voice calls affect data in operational LTE networks," in *Proc. MobiCom*, Oct. 2013, pp. 87–98.
[47] G.-H. Tu *et al.*, "Accounting for roaming users on mobile data access: Issues and root causes," in *Proc. ACM MobiSys*, 2013, pp. 305–318.
[48] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proc. USENIX NSDI*, 2011, p. 8.
[49] Y. Zaki, L. Zhao, C. Goerg, and A. Timm-Giel, "LTE mobile network virtualization," *Mobile Netw. Appl.*, vol. 16, no. 4, pp. 424–432, 2011.
[50] H. Zhang, X. Wen, B. Wang, W. Zheng, and Y. Sun, "A novel handover mechanism between femtocell and macrocell for LTE based networks," in *Proc. 2nd Int. Conf. Commun. Softw. Netw. (ICCSN)*, 2010, pp. 228–231.



Yuanjie Li received the bachelor's degree from Tsinghua University in 2012, and the the Ph.D. degree in computer science from the University of California at Los Angeles (UCLA), Los Angeles, in 2017. His research interests include networked systems, mobile computing, and network security. He is a member of the Wireless Networking Group.

Chunyi Peng received the Ph.D. degree in computer science from University of California at Los Angeles, in 2013. He is currently an Assistant professor of computer science with Purdue University. Previously, she was an Assistant Professor with the Department of Computer Science Engineering, The Ohio State University.

Haotian Deng received the B.E. degree from Tongji University in 2013 and the M.S. degree from The State University of New York at Buffalo in 2015. He is currently pursuing the Ph.D. degree in computer science with Purdue University. His research interests are mainly on mobile networks.

Zengwen Yuan received the B.S. degree from Shanghai Jiao Tong University in 2015. He is currently pursuing the Ph.D. degree in computer science with the University of California at Los Angeles. His research interests include mobile networks, mobile computing, and wireless security.

Guan-Hua Tu received the Ph.D. degree in computer science from the University of California at Los Angeles, Los Angeles, CA, USA, in 2015. He is currently an Assistant Professor with the Computer Science and Engineering Department, Michigan State University, MI, USA. His research interests cover focusing on mobile networks, mobile IoT, wireless networking, and network security.

Jiayao Li is currently pursuing the master's degree in computer science with the University of California at Los Angeles.

Songwu Lu is currently a Professor of computer science with the University of California at Los Angeles. His research interests include mobile networking and systems, cloud computing, and network security.

Xi Li was the Director of research programs with the Embedded System Lab, examining various aspects of embedded systems with a focus on performance, availability, flexibility, and energy efficiency. He has led several national key projects, several national 863 projects, and NSFC projects. He is currently an Associate Professor and the Vice Dean of the School of Software Engineering, University of Science and Technology of China. He is a member of the ACM and a Senior Member of the CCF.