Survey of End-to-End Mobile Network Measurement Testbeds

Utkarsh Goel, Mike P. Wittie, Kimberly C. Claffy, and Andrew Le

Abstract—Mobile (cellular) networks enable innovation, but can also stifle it and lead to user frustration when network performance falls below expectations. As mobile networks become the predominant method of Internet access, research, development, and regulatory communities have taken an increased interest in measuring mobile network *performance* and its *impact on user experience*. In this survey we examine current approaches to end-to-end mobile network performance measurement, diagnosis, and application prototyping. We compare available tools and their shortcomings with respect to the needs of researchers, developers, regulators, and the public. We intend for this survey to provide a comprehensive view of *currently active efforts* and some auspicious directions for future work in mobile network measurement and mobile application performance evaluation.

Index Terms—Mobile network, measurement, testbeds.

arXiv:1411.5003v2 [cs.NI] 20 Nov 2014

1 Introduction

Mobile (cellular) network applications deliver interactive services, generally supported by back-end logic deployed on cloud infrastructure. These applications support a wide breadth of functionality, such as live video, social gaming, communication services, and augmented reality [1]–[4]. Future services will increasingly leverage cloud-based datasets and processing power for innovative applications of live speech translation, real-time video analysis, or other computationally intensive tasks [5], [6]. As the frequency of interactions between mobile devices and back-end servers increases, application responsiveness and user experience will be increasingly tightly coupled with end-to-end network performance.

To innovate in the interactive mobile application space, developers deploy communication protocols with sophisticated data delivery techniques that support responsive communications under a range of network conditions [7]–[11]. These techniques are not always sufficient and developers are sometimes forced to redesign application functionality to mask poor network performance. However, these latter optimizations require detailed network performance data that is often not readily available, which results in challenges across the cellular ecosystem. For example, **developers** face the undesirable choice of evaluating performance of their mobile applications in limited private deployments that lack geographic diversity, or distributing their code to users without adequate testing [12], [13]. **Researchers** lack network performance data, or tools to acquire such data, in order to rapidly test hypotheses and focus on realistic network performance problems. Finally, **regulators** have a limited view of network performance, especially with respect to traffic shaping by network providers, impeding their ability to tackle performance challenges and roadblocks for sustained innovation in the mobile space [14], [15].

This paper provides a comparative analysis of currently available tools for end-to-end mobile network measurement, monitoring, and experimentation. Based on our review of current measurement efforts, we observe that although existing approaches comprise only a patchwork of needed functionality, they already generate powerful insights to guide development, research, and regulatory actions. However, in spite of the relative maturity of several tools, daunting problems remain including support for wide-scale application prototyping and deployment, detection of traffic shaping, and long-term network performance monitoring. Most existing mobile measurement tools have been developed in isolation, and one motivation for this survey is to foster more concerted and cooperative efforts at standardization of measurement libraries, privacy policies, and technology exchange [16]–[19].

The rest of this paper is organized as follows. Section 2 reviews goals of end-to-end mobile network measurement, and Section 3 provides details on and a classification of current tools for end-to-end mobile network measurement. Section 4 presents directions for future work and concluding thoughts.

- *U. Goel and M.P. Wittie are with the Computer Science Department, Montana State University, Bozeman, MT 59717.*
  E-mail: utkarsh.goel, mwittie@cs.montana.edu
- *KC Claffy is with CAIDA, La Jolla, CA 92093*
  E-mail: kc@caida.com
- *A.Le is with Mintybit, Santa Barbara, CA 93111*
  E-mail: andrew@mintybit.com

2 Goals of End-to-End Mobile Network Measurement

The 2014 CAIDA workshop on Active Internet Measurements (AIMS 2014) brought together researchers,

developers, and regulators interested in mobile (and wireless) network performance [18]. Although these communities share the goal of understanding and improving performance of current mobile networks, they focus on different metrics, and thus the tools they produce (Section 3) take different approaches.

2.1 Developers' View of Network Performance

Developers, ranging from independent game developers to industry giants such as Google or Weibo, want to provide a responsive application experience to their users. Although much of the delay experienced by user requests is due to back-end processing and front-end rendering [10], as hardware and software processing speed improves, network latency becomes a dominant concern. At the same time, network latency does not necessarily benefit from advances in communication technology. As Internet Service Providers (ISPs) seek to minimize forwarding costs, they configure their networks to maximize throughput, to the detriment of latency. Specifically, ISPs direct traffic onto inexpensive but circuitous routes, which inflates hop counts and path latency [20]. ISPs also configure cellular schedulers to delay transmissions until carrier channel conditions are favorable [21]. Mobile networks remain bandwidth-constrained, which motivate ISPs to employ traffic shaping mechanisms on video streaming and P2P traffic to increase the usable bandwidth for all mobile users [22]. However, this approach induces high latency that impede performance of dynamic content applications such as interactive Web, live video, and group communication and collaboration tools [23].

Thus, developers need tools to study the performance of *their* application traffic in mobile networks to *their* application servers. Few tools support such realistic experimentation prior to application deployment. Most mobile network testbeds allow users to measure only upload and download speeds, ping latency, and traceroutes, but do not support prototyping of mobile application traffic, or detect traffic shaping in cellular networks. An alternative to public testbeds are paid services that evaluate application performance across multiple types of mobile devices. However, these services currently provide access only to stationary cellular devices, which limits measurement realism in terms of geographic and network diversity [12], [13].

Additionally, while there are best practices (for both native and Web-based applications) that mobile developers consider in their implementation [7], there are not many tools to track an application's communication performance throughout its lifetime. Although server monitoring and reporting tools, for example from Librato, can monitor application performance indicators such as request queue length at a server, they do not support end-to-end network measurement [24]. Other analytics tools, for example the Google Analytics platform, provides performance measurement from a client perspective, but captures only the timing of the request-response cycle, and not response characteristics, e.g., size, compression, protocol [25]. Developers need to measure and understand application performance in a realistic network environment before and after deployment, particularly as data needs and application requirements evolve.

2.2 Researchers' View of Network Performance

Researchers need versatile tools and datasets to test their hypotheses. The research community has produced several tools that offer significant flexibility to execute a variety of network experiments [20], [26]–[38]. Yet, the availability of these tools and knowledge of how to use them often remains limited by practical barriers to collaboration across research groups. Researchers may need to set up their own infrastructure for data collection [39], obtain Institutional Review Board (IRB) approvals [27], or revive code that is no longer maintained [40], [41]. Even when maintainers of a given tool can help set up experiments, communication rounds take time, especially when software modifications are needed. As a result, researchers often decide it is more expedient to develop new tools, even when it duplicates other efforts and achieves only a small scale evaluation [27], [30].

Several organizations are working to lower the barrier to entry and promote concerted development of network measurement tools. For example, M-Lab maintains a repository of measurement tools, including MobiPerf, NDT (Mobile client), and WindRider, discussed later in this paper [42]. One of M-Lab's goals is for new tools to leverage existing code base, for example the Mobilizer library [34]. M-Lab also supports the development of common ethical guidelines for network measurement data collection [16]. However, the continued flow of proposals for new, independently deployed cellular tools (five in 2014 [30], [34], [37], [38], [43], seven in 2013 [20], [26]–[29], [35], [44], two in 2012 [12], [33], one in 2011 [31], one in 2010 [36], and one in 2009 [45]) suggests that more needs to be done to improve collaboration among different research groups.

2.3 Regulators' View of Network Performance

Finally, regulators need monitoring tools to inform their understanding of availability, reliability, and performance of mobile networks over time. Constrained network performance and delayed upgrades to next generation technologies, e.g., 4G, have long been seen as stifling innovation in the US [46], [47]. Further, traffic shaping mechanisms and anti-competitive behavior by some network providers impede deployment of new services [15], [48]–[55]. Even developers of popular measurement tools struggle to create incentives for longitudinal and widespread measurement [44]. A few tools that have gained traction with users rely on user-initiated network tests, which limits measurement frequency and representativeness [31], [36].

2.4 Shared Challenges

Developers, researchers, and regulators face the same challenges in deploying end-to-end mobile measurement tools: incentivizing a statistically significant sample of users to install and execute the tool; protecting those users' resources from abuse; and preserving user privacy.

To motivate user participation, testbed designers have used schemes such as bundling testbed code with other functionality [39], offering free devices [27], press coverage [36], [44], or simply appealing to user altruism and curiosity [44]. These approaches result in either a narrowly focused user base or short-lived deployments, both of which limit testbed utility.

The second challenge is how to protect contributed testbed resources from abuse. Some peer-to-peer systems have used tit-for-tat mechanisms to ensure fair resource sharing [56], but mobile network measurement testbeds thus far rely on user altruism on the one hand and conscientiousness on the other [31]–[33], [44]. Scaling and sustaining measurement testbeds over the long term will require more rigorous resource protection methods in existing tools.

Finally, a testbed should isolate personally identifiable information from experimental data collected on a mobile device. Measurement tools discussed in this paper offer a range of solutions to maintain this separation. Google has supported the development of a proposed set of ethical guidelines for the design of mobile-based network measurement tools [16]. These guidelines have informed the design of some tools, specifically MITATE and Mobiperf, but the disparate legal frameworks for user privacy around the world make it difficult to create conformant tools for the global mobile Internet [18].

3 Measurement Testbeds

The developer, research, and regulatory communities have created a variety of tools to monitor end-to-end mobile network performance. In this section we discuss their various approaches and how they meet community measurement needs and challenges. We also describe the most salient features of each tool, and how some features differ across tools. Specifically, we classify these tools based on their capabilities to support application traffic prototyping (Section 3.1), collect end-to-end network measurements (Section 3.2), diagnose network problems (Section 3.3), and evaluate new protocols in controlled wireless environments (Section 3.4). Table 1 further compares the tools in terms of their experimentation flexibility, device selection criteria, resource protection, and other features.

3.1 Mobile Application Prototyping

Mobile application developers need to know how well a network can deliver their application content. Custom network experiments that emulate communication protocols of their applications can create performance profiles in different network settings to inform application design. End-to-end systems that support such functionality need to balance the flexibility of their feature set against potential abuse of contributed user resources and threats to user privacy. We divide systems into curated (PhoneLab [27], and PortoLan [28]) and uncurated (MITATE [20] and Seattle [26]) approaches, depending on how they resolve this conflict for new experiments.

3.1.1 MITATE

Mobile Internet Testbed for Application Traffic Experimentation (MITATE), designed by Utkarsh Goel *et al.* at Montana State University, enables experimentation with mobile application traffic in live mobile networks [20]. Traffic experiments execute on user-volunteered devices that meet specified criteria, such as signal strength, geographic location, or network provider. Developers can use MITATE to evaluate the performance of mobile application traffic under a wide range of conditions before their applications are deployed, or even fully developed. MITATE also supports configurable active network measurements to detect network traffic shaping configured by mobile ISPs and integration with other tools, such as optimizer packages, for example to determine parameters tradeoffs in communication protocol configurations.

Functionality: MITATE supports flexible active network measurements on mobile devices. MITATE experiments are configured through XML files that describe the content of active experiment data transfers, transport layer protocols, network endpoints, and timing. An XML configuration also describes criteria that volunteered devices should meet to execute an experiment, such as network type (cellular, or Wi-Fi), signal strength, geographic location, or network carrier, minimum battery power, and device model name. To ensure that experiments are defined correctly, MITATE servers validate new XML configuration files against an XML schema definition (XSD). Users interact with MITATE through an API that allows upload of XML configuration files and download of collected data.

Each mobile device polls MITATE servers for new experiments with criteria matching the device capability. Devices download static traffic definitions that specify what traffic to exchange between mobile and back-end servers.

MITATE mobile devices can interact with third party systems, for example DNS and CDN servers, through explicitly configured, well-formed request packets, and by recording reply content and delay. Although each MITATE experiment is a series of static transmissions, complex logic can be implemented across processing rounds, e.g., DNS lookups and ping transactions require two rounds. Such an experiment specifies a device ID as a criteria, which allows for the same device to issue DNS

	MITATE [20]	Seattle [57]	PhoneLab [27]	LiveLabs [30]	PortoLan [28]	FCC Speed Test [44]	MobiPerf [31]	WindRider [45]	MySpeedTest [33]	Akamai Mobitest [12]	ALICE [43]	NDT (Mobile Client) [35]	Netalyzr [36]	SciWiNet [37]	PhantomNet [38]
Measurement Capabilities															
Traffic shaping/DPI	✓	✓	✓		✓			✓							
Passive measurements			✓	✓			✓	✓	✓		✓				
Active measurements	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Measurement data publicly available	2		✓			2	✓				✓	✓			
Custom packet content	✓	✓	✓							1					
Peer-to-peer traffic	2	✓													✓
ICMP traceroutes	2		✓		✓		✓				✓		3		
Programmable execution environment	4	✓	✓	✓							✓		✓		✓
Access to mobile sensor data	✓	2	✓	✓	✓		5,6	5	6		5,7		5,7	✓	
Experiments can be scheduled on specific devices	✓	✓	✓	✓						✓	✓		✓		✓
IPv6 support	2	2			✓		✓						✓		
Traffic on ports < 1024	3	3	✓												
Diagnoses network problems							✓					✓	✓		
Mobile platform	8	8,9,12	8	8,9	8	8,9	8	10	8	8,9,11	8	8	8	8	8
Device Selection Criteria															
Geographic location	✓	✓	✓	✓										✓	
Device model	✓						✓			✓					
Battery charge level	✓	✓	✓												
Carrier signal strength	✓		✓												
Network carrier	✓										✓				
Network type (Wi-Fi/Cellular)	✓		✓								✓				
Time of day	✓	✓	✓	✓						✓	✓				✓
Resource Usage Limits															
Transmission rate		✓													
Bandwdith cap	✓	✓			✓	✓	✓		✓		✓			✓	
Minimum battery charge	✓	2			✓		✓								
Port restrictions		✓												✓	
Miscellaneous															
Measurement scheduling API	✓	✓		✓			✓				✓				✓
Supports devices behind NAT/Wi-Fi	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	
Requires rooted phones			✓							✓					✓
Open to public	✓	✓	✓		✓	2	✓	✓	✓	✓	✓	✓	✓	✓	✓
User incentive model	R	R,A	U	S,U	A	C	C,S	C	C	A,C	S	S	S		
Experiments require approval			✓	✓											
Open-source	✓	✓				✓	✓	✓	✓	✓		✓			
Currently active	D	✓	✓	D	✓	✓	✓	✓	✓	✓	D	✓	✓	D	D
Records hardware specs	O					✓	✓	✓		✓				✓	
Records hardware performance	O	O	O						✓	6					

Legend:
1 – measurements can be directed to arbitrary web servers.
2 – planned functionality.
3 – on rooted phones only.
4 – through multiple experiment rounds on the same device.
5 – GPS readings.
6 – battery readings.
7 – radio state.
8 – Android.
9 – iOS.
10 – Windows.
11 – Blackberry
12 – Nokia
R – Reciprocal (tit-for-tat).
U – sUbsidized.
A – user Altruism to support measurement capacity.
C – user Curiosity to understand their own network performance.
S – provides Service to clients other than measurement data.
D – under Deployment.
O – Optionally.

TABLE 1
Experimentation flexibility matrix of end-to-end measurement tools.

lookups in round one and subsequent pings in round two.

Data Collection: MITATE records the delay of each data transfer as well as metadata such as signal strength, accelerometer readings, and device location. The delay of each transfer allows calculations of 42 metrics, including uplink and downlink latency, throughput, jitter, and loss, as well as mobile sensor readings [58]. For example, an experiment estimates available bandwidth by dividing the size of a large transfer by its duration. MITATE experiments may also use a series of small transfers to estimate packet round trip time (RTT), loss, and jitter. At the start of an experiment, MITATE estimates the clock offsets between a device and each server endpoint, which allows separate measurement of uplink and downlink latency.

Collected data is available for download in the form of SQL insert statements to populate a local instance of a MySQL database for each user. MITATE allows users to download data only for their own experiments and those whose data is made public. Aggregate metrics, for example mean latency, are computed through queries to the local database instance. This design reduces the load on the MITATE database servers and allows users to run arbitrary queries over their experiment data.

Resource Incentives and Protection: MITATE is a collaborative framework built around incentives for user participation, inspired by BitTorrent's tit-for-tat mechanism [56]. MITATE users earn data *credit* by contributing their mobile resources. Users can then spend credit to run experiments on others' devices. Earned credit expires after 24 hours to prevent its accumulation and use for large experiments that might overwhelm available system-wide resources at any point in time.

MITATE's credit exchange system both encourages ongoing participation and protects contributed resources from abuse. Users can leverage MITATE resources in direct proportion to how much data they contribute to the system. MITATE does not rate-limit device transmissions (although users can set monthly data caps and battery limits on their devices), which permits realistic load-testing experiments. Although distributed denial of service (DDoS) attacks launched from multiple devices are technically possible in MITATE, they are destined to be short lived, because rapid transmissions from multiple devices will quickly deplete the malicious user's earned credit.

Privacy Protection: A significant challenge to expanding measurement systems on volunteered personal devices is the threat to user privacy. To limit the exposure of personally identifiable information, MITATE captures data only from active traffic experiments and does not monitor non-MITATE device traffic. Collected data is also indexed by virtual device IDs, rather than personally identifiable phone and International Mobile Equipment Identity (IMEI) numbers.

Remaining Challenges and Limitations: Although MITATE provides a flexible platform for traffic experimentation in live mobile networks and represents a significant step forward, the system has some limitations (as of October 2014). First, MITATE deployment on M-Lab is ongoing. Second, MITATE does not yet (but we plan to) support peer-to-peer transmission between mobiles, which will be important to experimentation with social gaming, augmented reality and Internet of Things (IoT) applications. Finally, in 2014 MITATE supports only Android devices; we hope to extend support to iOS devices in the future.

3.1.2 Seattle

Seattle, designed by Justin Cappos *et al.* at University of Washington, also supports mobile application prototyping [57]. The design goal was to increase the diversity of testbed hardware to provide a more realistic prototyping environment than testbeds relying on dedicated hardware, such as PlanetLab, Emulab, or GENI [59]–[61]. Seattle runs on volunteered devices in last mile networks, and on institutional servers. Seattle users receive immediate access to Seattle upon registration, as opposed to waiting for an institutional approval process. While developed for wired hosts, as of 2013 Seattle includes over 500 mobile devices and over 20,000 nodes in total.

Functionality: Seattle experiments run on sandboxed virtual machines in a pared down implementation of Python called Repy. Seattle libraries support Repy in functions such as data serialization, cryptography, and processing URLs, HTTP messages, and other protocols. Repy code is pushed to devices registered with Seattle through an API. Users can select devices based on the device location and the network environment to which device is connected, but Seattle lacks some device selection criteria appropriate in the mobile setting, such as device speed, network provider, or device model. A powerful feature of Seattle is its support for P2P communication among devices.

Data Collection: Seattle does not collect network performance data by default. Instead users define their own metrics through experiments implemented in Repy. On mobile devices, Seattle does not provide access to device sensors [62]. Instead, Seattle provides an API, through which sensor applications can make sensor data available to Repy programs. Finally, the Sensibility testbed is an extension of Seattle, which allows Repy experiments to interact with mobile sensor data, but not to transmit, or capture network traffic [63].

Resource Incentives and Protection: The Seattle incentive model is based on a tit-for-tat approach, where a user has access to ten volunteered devices for every device she

registers with the system. While this policy makes sense in the wired setting, where devices are not generally restricted by monthly data caps, users who register wired hosts but experiment with others' mobile devices can deplete the mobile data cap. As a mitigating step, by default Seattle limits data transmissions to 10 Kbps, so even if the testbed fully uses the contributed bandwidth, the owner can likely continue using their device. However, this limit prevents Seattle experiments from measuring available bandwidth and generating load-testing traffic – limitations not present in MITATE's credit-based model. Future versions of Seattle will include limits on battery drain.

Privacy Protection: Seattle protects user privacy by allowing experiment code execution only in sandboxed virtual machines, which isolates experiment processes from each other and from non-Seattle processes.

Remaining Challenges and Limitations: The authors of Seattle list several limitation of the current system [26]. The limitation most pertinent to performance measurement is the inability of Seattle devices to send ICMP traffic, due to Repy restrictions.

3.1.3 PhoneLab

PhoneLab is a programmable smartphone testbed, designed by Anandatirtha Nandugudi *et al.* at University at Buffalo, for experimentation flexibility on par with application deployment [27], [64]. PhoneLab experiments are implemented as mobile applications pushed to rooted Android smartphones given to student volunteers at the University at Buffalo. PhoneLab's model supports long-term, passive experiments that can record network transitions, battery drain, and use of other applications on the device. Because passive measurement is privacy-sensitive, experiments are reviewed by an Institutional Review Board (IRB) as well as by PhoneLab developers, resulting in curated access to the testbed.

Functionality: PhoneLab experiments are pushed to participants either via the Google Play Store, or as platform over-the-air updates. PhoneLab can benchmark third-party mobile applications without modifications to their code, which may be required in other testbeds. PhoneLab mobile applications can run experiments in the background or interactively. PhoneLab also supports experiments at the platform level, with modifications to the Android runtime system. Platform experiments are vetted by the PhoneLab development team and go through pre-deployment testing. Researchers submit experiments as XML configuration files that specify background experiments to start or stop, log tags to collect, and where to upload collected data. The PhoneLab Conductor fetches configuration files from PhoneLab servers and pushes them to testbed devices.

Data Collection: PhoneLab data collection relies on the Android logging interface, which gives experiments access to device operational data (such as phone status, battery level, etc.), as well as custom application log data. All log data is uploaded to the central server when a device is charging. When their experiment completes, users receive an archive of data that matches experiment tags from all devices that participated in their experiment.

Resource Incentives and Protection: Unlike MITATE and Seattle, which rely on volunteered devices, PhoneLab provides phones with discounted data plans to its participants. This approach results in high scaling costs, although the RIPE Atlas project has successfully build a large network testbed from inexpensive LAN-attached devices provided free of charge to hosting institutions [65]. In spite of this incentive scheme, the PhoneLab team has faced significant participant attrition, with only 43 of 191 first year volunteers continuing in the second year [27]. Also unlike MITATE, PhoneLab allows multiple active experiments at a time on a device. PhoneLab limits the number of simultaneously active of experiments on each device to balance device utilization against interference between experiments.

Privacy Protection: To protect user privacy, experiments submitted to PhoneLab need IRB approval, or exemption. PhoneLab participants also choose to participate in a particular experiment after reviewing what information will be collected. Participants can opt-out of an experiment at any time.

Remaining Challenges and Limitations: PhoneLab's use of volunteers potentially limits the scalability of the testbed. Also if phones are not replaced frequently, testbed hardware will eventually lag behind phone models used by the general public. Finally, PhoneLab code is not publicly available, which precludes the possibility of private deployments [66].

3.1.4 LiveLabs

LiveLabs, designed by Archan Misra *et al.* at Singapore Management University, is a mobile testbed intended to evaluate location-based services, such as commercial promotions to shopping mall customers [30]. LiveLabs has been tested on campus of the Singapore Management University (SMU) and is currently being deployed at a large shopping mall near SMU campus, Singapore Changi International Airport terminal, and the Sentosa resort island. The testbed is available to the three partnering venue operators, but not the general public.

Functionality: To facilitate evaluation of location-based services, LiveLabs supports device location discovery in indoor settings as well as characterization of user behavior. LiveLabs is designed for continual operation, thus the design has focused on low energy usage, for example by allowing multiple experiments to concurrently use sensor readings such as GPS, or WiFi signal strength.

Researchers and participating companies use LiveLabs to evaluate location-based applications, for example real-time promotions to users at a shopping mall. LiveLabs is available for Android and iOS systems.

Data Collection: Unlike other testbeds discussed in this section, LiveLabs does not collect network performance metrics, but instead focuses on discovering user behavior, by recording device ID and a variety of sensor readings. The LiveLabs backend then supports higher level functions to detect and record user behavior, such as history of movement, group size, user physical queue length, and activity such as standing, walking, or sitting. LiveLabs also records information about participating users, such as their nationality.

Resource Incentives and Protection: LiveLabs has several mechanisms for garnering user participation. First, users are offered a rebate on their monthly data bill. Second, users are recruited by context-based apps, which offer rebates on commercial services in deployment locations [67]. Finally, LiveLabs runs a "lucky draws" lottery, though details of frequency and prizes are not specified [30].

Privacy Protection: Data collected by LiveLabs has the potential to disclose private user information, such as location, shopping patterns, and nationality. As such, experiments launched on LiveLabs go through SMU's IRB approval process [68]. Users are also asked to opt-in to data collection on their devices.

Remaining Challenges and Limitations: LiveLabs is not designed for mobile network measurement (does not collect network metrics) and so it offers functionality distinct from MITATE, Seattle, and PhoneLab. At the same time, LiveLabs supports experimentation with new services in the mobile environment similarly to PhoneLab and has attracted participation of 30,000 users through its incentive model and business partnerships.

3.1.5 Emerging Systems

In addition to MITATE, Seattle, PhoneLab, and LiveLabs we are aware of two systems in different stages of planning that will support mobile application prototyping.

PortoLan is a network experiment testbed based on volunteered mobile devices that executes experiments submitted to back-end servers [69]. PortoLan is designed by Enrico Gregori *et al.* at Istituto di Informatica e Telematica, to discover Internet topology and build wide scale mobile network signal quality maps. The Android application for PortoLan is available on Google Play and allows users to run measurement tests like ping, traceroutes, maximum throughput, and detection of traffic shaping of BitTorrent traffic [70]. The PortoLan team intends to add capability to support active network experiments and access to mobile sensor data such as network signal strength, device location, network name, cell type, and roaming status. PortoLan relies on user altruism to build testbed capacity and support measurement. The PortoLan mobile application limits the device cellular bandwidth usage to 2 MB/day and postpones experimentation when battery drops below 40%. Finally, the application does not collect personally identifiable information from the device and anonymously stores measurement data on backend servers.

NetSense is a mobile testbed, designed by Aaron Striegel *et al.* at the University of Notre Dame, focused on characterising the impact of mobile information technology on social tie creation [29]. The project is similar to PhoneLab in that two hundred rooted Nexus 4S Android devices with fully subsidized data plans were given to incoming freshmen. NetSense has also experienced high participant attrition, mitigated by subsidized device repair and replacement plans. As of August 2013, NetSense is winding down its two-year data collection effort and will not continue as a testbed.

3.2 Monitoring Network Performance

Mobile network performance characterization requires wide scale and ongoing measurement from a variety of devices across different networks and locations. Tools in this space, developed by industry, research, and regulatory communities, differ in how they obtain network metrics and how they select devices for measurement. Although network measurement tools presented in this section are not testbeds, in that they only support a fixed set of experiments, these tools do support long-term and wide-scale network monitoring, which offers important insights to developers, researchers, and regulators.

3.2.1 FCC Speed Test

The FCC Speed Test app, released in November 2013, was designed to provide insight to regulators and the general public on the performance of mobile networks across geographic areas in the US [44]. Developed in collaboration with SamKnows and major wireless service providers, the free application is available on Google Play Store for Android smartphones [71]. An iOS version of the application is also slated for release, though limitations of the iOS API mean that the iOS version will not collect some metadata collected by the Android version [14], [72].

Functionality: At the start of a measurement, the FCC Speed Test app pings available measurement servers to identify one with lowest round trip time (RTT) to the mobile device. The selected server then sends a list of measurement instructions to the mobile device. If the mobile device is currently using less than 64 Kbps of bandwidth for other tasks, it starts the measurements, otherwise the device postpones measurement until its bandwidth usage drops.

The FCC Speed Test app supports active traffic measurements over four types of connections: single connection HTTP GET and POST, as well as multi-connection

GET and POST. Multi-connection transfers test multithreaded download performance over three parallel downloads of 256KB data chunks. To measure packet loss and RTT, the FCC Speed Test app exchanges a series of UDP packets with the nearby server. Following a measurement, the mobile device uploads measurement data and associated metadata to an FCC server.

Data Collection: The FCC Speed Test app reports upload and download rates, packet loss, and RTTs based on HTTP and UDP transfers. Packet loss on a path is inferred based on failure to receive a UDP packet on that path within three seconds. The app records the number of packets sent each hour, the average RTT, total packet loss for performed tests, and throughput in 5-second intervals [73].

The FCC Speed Test app also collects metadata related to the mobile device as well as the network. The collected metadata includes signal strength reported by the device, type of connection (3G/4G/Wi-Fi), location and ID of cell towers, GPS location, device model, OS version, network country code, SIM's operator ID, SIM's country code, network carrier, phone type (GSM/CDMA), and the device's roaming status.

Resource Incentives and Protection: To build nationwide measurement capacity the FCC Speed Test app relies on user curiosity about their own network performance. Instrumental to the app's popularity and success was a press campaign [72], [74]–[77], which was followed by application installation and measurements from more than 50,000 devices in about 1.5 years. Unfortunately, these numbers have declined over the life of the system, so the effectiveness of a publicity-driven approach to support long-term network monitoring remains to be seen.

Privacy Protection: The FCC app collects measurement data on the mobile device in the application sandbox, as opposed to through the standard Android logging interface, so data is not visible to other applications. The collected data are uploaded to FCC servers over encrypted connections. Once the data are uploaded, or become stale, they are automatically deleted from the application's sandbox storage. The FCC Speed Test app does not collect personally identifiable information, such as phone number or IMEI [78].

Remaining Challenges and Limitations:

The FCC Speed Test app executes only experiments configured by the FCC, i.e., it does not support custom network measurement. As of October 2014, the configured tests do not detect traffic shaping in mobile networks, which is of increasing interest to regulators and the general public [15], [48]–[55]. With respect to resources used on the device, the FCC application runs at startup and prevents the phone from sleeping, which can drain the phone battery.

3.2.2 MobiPerf

The MobiPerf mobile application was developed as a collaboration of University of Michigan, Northeastern University, University of Washington, and M-Lab to measure network performance and diagnose problems with application content delivery on mobile devices [31]. To allow the community to understand the impact of collected data across geographic locations, network carriers, and devices, MobiPerf allows a comparative study of past network measurements made by different users. This approach limits contention for testbed resources by preventing multiple users from running the same measurement. New measurements are executed only if a query for previously collected data comes back empty. The latest version of MobiPerf, released in August 2014, is based on Mobilizer – an open-source Android library for network measurement announced at AIMS 2014 [34].

Functionality: MobiPerf supports several types of network performance measurement, which can execute serially or in parallel [17]. Mobilyzer provides measurement isolation (only one measurement experiment is active at a time), which avoids bandwidth contention among measurements, and also prevents side-effects such as radio power state transitions across experiments. To measure throughput, Mobiperf transmits random data to and from a nearby M-Lab server for 16 seconds and computes throughput from uplink and downlink packet traces.

MobiPerf supports latency measurements on both IPv4 and IPv6 network paths, using ICMP ping when available, with fallback to a Java ping implementation and latency estimates from three-way TCP handshakes in HTTP transfers. Mobiperf measures the delay of DNS lookup using the default DNS server configured for the device. The constraint of experimenting with only one DNS server, limits the ability to measure performance of third-party open DNS infrastructure.

MobiPerf also supports measurement of uplink and downlink UDP packet loss, out-of-order delivery, and variation of one-way latency. To obtain these metrics on the uplink, a client device sends a group of UDP packets to a nearby M-Lab server, where the server calculates network metrics from packet arrival time and order. The same transmission repeats from server to client to calculate downlink metrics.

MobiPerf performs more complex measurements to discover fine-grained network policies and their effect on data plane performance. For example, MobiPerf measures radio resource control (RRC) state information of cellular networks to estimate the impact on packet latency [79]. Finally, MobiPerf measurements can execute in the background to support long-term monitoring of network performance.

Data Collection: Similar to other measurement tools, the MobiPerf application collects measurement related data such as TCP uplink and download throughput,

HTTP download latency and throughput, traceroutes, path latency, and DNS lookup delay. Researchers and vendors may also want to know how variation in mobile hardware affects application performance, so MobiPerf also collects device-related data such as manufacturer, model, operating system version, Android API level, carrier, salted hash of device IMEI, coarse-grained cell ID location information, cell tower ID and signal strength, Location Area Code (LAC), local IP address, IP address seen by the remote server, GPS coordinates, ports blocked by cellular provider and network connection type (HSPA/LTE) [80], [81].

Resource Incentives and Protection: MobiPerf relies on user curiosity to support measurement, but users can limit the resources they contribute. Specifically, measurements do not execute when the device battery consumption, or MobiPerf application monthly data usage, exceed user-set thresholds.

Privacy Protection: MobiPerf currently records the users' e-mail address, if they choose to provide one, to access their historical measurement results. This information is secured by Google's account authentication mechanisms and is not made publicly available. To minimize any risk of exposing this potentially personally identifiable information, future versions of MobiPerf will store a salted hash of users' e-mail addresses instead.

Remaining Challenges and Limitations: MobiPerf allows users only to choose from predefined measurements, which limits testbed flexibility. For example, MobiPerf does not support transfers of custom content on arbitrary ports to detect network traffic shaping.

3.2.3 WindRider

Content based traffic discrimination has recently been considered a threat to mobile application performance [15], [48]–[55]. WindRider, a measurement tool developed in 2009 by Ionut Trestian *et al.* at Northwestern University, detects application and service-based traffic discrimination by mobile ISPs [32]. WindRider's mobile client supports a number of traffic shaping experiments against M-Lab servers.

Functionality: WindRider supports active and passive measurement of traffic shaping [45]. Active measurements exchange traffic between a user's mobile devices and a randomly chosen M-Lab server. The mobile device initiates a series of uploads and downloads and records their observed performance. To detect port-based traffic shaping, WindRider compares delay of identical transfers to different ports on M-Lab servers. Passive measurements record packet latency to well-known Web servers during normal user browsing activity. To detect content-based traffic shaping, WindRider compares the observed packet delay to that reported by other devices in different carrier networks and locations. Active measurement results are stored on M-Lab servers, while passive measurement data, collected with user permission, are stored on WindRider servers.

Data Collection: The WindRider mobile application collects experiment-related data such as connection start time, connection establishment time, connection finish time, and number of inbound and outbound bytes [32]. WindRider also collects metadata such as device IMEI, device location (as ZIP code), network carrier, and browsing history. WindRider also collects device hardware performance metrics that can help interpret observed traffic delays, such as CPU execution time, virtual memory size, page faults per minute, and other metrics as permitted by the OS API.

Resource Incentives and Protection: WindRider relies on user curiosity for its network measurements.

Privacy Protection: WindRider optionally collects device IMEI, which can be linked with a user's browsing history. To protect user privacy, users can choose whether to make this information available to the application.

Remaining Challenges and Limitations: Although WindRider supports detection of traffic shaping in mobile networks, it has two significant limitations. First, active measurement traffic is sent only to M-Lab servers, and developers may want to evaluate whether traffic shaping is present on paths to their servers. Second, WindRider only detects content-based traffic shaping as discrimination based on traffic sources, i.e., well-known Web servers, rather than type of traffic, for example BitTorrent.

3.2.4 MySpeedTest

The MySpeedTest mobile application, launched in June 2012 by Sachit Muckaden *et al.* at Georgia Institute of Technology, measures network performance of mobile users with the goal of explaining patters of user behavior observed by mobile ISPs and application developers [82], [83]. Such analysis may allow developers and service providers to tune application performance. MySpeedTest mobile application is available on Google Play and has more than 900 active users from 115 different countries, as of February 2013 [84]. As of April 2013, MySpeedTest is in the process of sharing a subset of their data with Google's M-Lab to help researchers benefit from data collected by each others' experiments [82].

Functionality: MySpeedTest performs passive and active measurements. Passively, MySpeedTest records the total number of bytes sent and received by each active application since the device booted. Information such as package name, bytes transmitted and received, application status (active vs. background) helps users know which applications consume the most data and power,

and which applications may affect performance of other applications on the device.

Active measurements include a recurring test to measure TCP uplink and downlink throughput, inter-packet delay, and packet loss. MySpeedTest also measures network latency with 40 parallel ICMP pings against five servers in the US and Europe. These tests store the minimum, average, and maximum latency for each of the five servers. The collected data help researchers and developers understand the performance of paths to potential application servers.

TCP-based experiments can reduce bandwidth available to other applications on the device, so MySpeedTest performs TCP-based experiments only on user request, in a single thread for about 20 seconds, and using the maximum-sized packets that will not be fragmented. MySpeedTest also gauges streaming data quality by measuring packet loss and jitter of UDP traffic flows. The server generates a stream of 64-byte UDP packets, transmission at Poisson-sampled intervals, with timestamps and sequence numbers in the payload. The server sends 500 packets with a data rate less than 1 Kbps to avoid congestion. On the client side, packet loss and jitter are calculated from every 10 packets received.

The client compiles all data collected on mobile device into the JSON format and sends it to the server for storage. Finally, the application scans for available Wi-Fi connections and their signal strength once every 15 minutes, which allows insight into the stability of the wireless network at the link layer.

Data Collection: The MySpeedTest mobile application collects experiment-related data such as TCP upload and download throughput, ping latency, UDP jitter, UDP packet loss, and time to acquire a dedicated channel for data transmission [82]. MySpeedTest also collects device level data, such as cellular service provider, Android version, device manufacturer, connection type, radio firmware, hashed phone number, hashed IMEI, software version, SIM card state and serial number, latitude and longitude of base station, network operator ID, CDMA system ID, CDMA network ID, battery technology, status of battery charging, battery health, battery voltage, battery temperature, and device location.

Resource Incentives and Protection: Similar to the FCC Speed Test app, MySpeedTest relies on user curiosity about their own network performance. MySpeedTest allows users to limit the amount of contributed mobile resources through a monthly data cap. To protect battery resources, MySpeedTest postpones network tests until the battery is above 5% and the device is attached to a network.

Privacy Protection: MySpeedTest collects personally identifiable information (phone number, IMEI, device location), which may expose private information, such as a user's location when a measurement occurred.

Remaining Challenges and Limitations: Similar to MobiPerf and WindRider, MySpeedTest provides users a limited network measurement capability between mobile devices and testbed servers. MySpeedTest does not support transmission of custom traffic, such as tools to detect traffic-shaping based on content or port.

3.2.5 *Akamai Mobitest and WebPageTest*

Akamai's Mobitest application and Web service, released in March 2012 by Akamai, measures the performance of mobile Web sites [12]. The application uses the WebPageTest framework and is available for Android, iOS, Blackberry based smartphones, tablets and simulators [85].

Functionality: Mobitest platform relies on user participation to install Mobitest software on their mobile devices. Each Mobitest installation on a device acts as an agent to the WebPageTest framework, where such device executes experiments requested by other users through the Mobitest Web service [86]. To measure the page load time on a mobile device, a user enters a URL through the Akamai Mobitest Web interface and selects the mobile device hardware that will perform the download [12]. Mobile devices running Mobitest periodically poll WebPageTest servers to obtain pending URL download requests entered by Mobitest users. Each requested URL is then accessed from the default browser on each device over the Wi-Fi, or cellular network, depending on how the device is connected at the time.

Data Collection: Akamai Mobitest collects the total time to load a Web page, individual request headers, average Web page size, as well as screen shots and optionally video of the loading page as it happened [86]. The tool produces waterfall charts of requests and delays, and an HTTP archive (HAR) file [87], [88]. The collected data helps researchers and developers gain insight into the responsiveness of Web servers and browser rendering of different site implementations [89]. Mobitest allows users to reuse previously collected data by linking them to user accounts on Akamai Mobitest's site.

Resource Incentives and Protection and Privacy: The Akamai Mobitest app allows its users to set the frequency at which pending experiments are downloaded from WebPageTest servers to be executed on their mobile devices. Additionally, Akamai Mobitest allows users to control device resource utilization through a number of configuration options. Specifically, users can set whether the app should poll for new experiments after restart, whether to restart the app after every experiment, whether to capture network traffic, and the frequency at which screenshots for loading pages are taken [90].

Remaining Challenges and Limitations: Akamai Mobitest evaluates the webpage load time on mobile devices,

but does not allow more general experiments with non-browser-based application traffic, including how to characterize traffic shaping of non-Web traffic. The WebPageTest framework requires rooted phones, which limits the applicability of the tool outside of dedicated test farms.

3.2.6 ALICE

A Lightweight Interface for Controlled Experiments (ALICE) is a programmable network measurement testbed for Android devices developed by John Rula *et al.* at Northwestern University [43]. ALICE extends Dasu, a rule-based network testbed built as an add-on to the Vuze BitTorrent client [39], by enabling experiment definition in Javascript [91].

Functionality: The ALICE measurement library supports active and passive experiments on mobile devices. ALICE provides a programmable interface for the configuration of active network measurements, such as DNS resolution, ping, iPerf, etc.. Tests can execute sequentially or in parallel. Although the sequence of tests and value passing between them is organized through a Javascript experiment definition, ALICE does not support custom traffic generation, and so is primarily a network measurement library. For serially scheduled experiments, ALICE allows only one experiment on a device at a time; for parallel execution, ALICE allows only a limited number of experiments to run at the same time – new experiments scheduled for a given device enter a queue until the device becomes available. ALICE chooses its test devices based on user-specified time of day, network provider, and network type (Wi-Fi/Cellular).

Data Collection: ALICE collects device location, radio signal strength (WiFi and cellular), WiFi access point name, device hardware address, IP address on each network interface, number of bytes sent and received by other applications on the device. ALICE also collects performance metrics, including HTTP GET request time, DNS lookup time, ping times, available bandwidth. ALICE also records network diagnostic information provided by traceroute and NDT (Section 3.3.1).

Resource Incentives and Protection: ALICE is a measurement library that can be included in Android applications. As of September 2014, ALICE has been included in three different applications developed at Northwestern University and available through the Google Play store: Namehelp Mobile[1], Application Time (AppT)[2], and NU Signals v2[3]. The Northwestern team's deployment model of growing the testbed through application deployments allows ALICE to benefit from popularity spikes of new applications. To protect device resources, developers can set quotas for bandwidth usage of individual measurements.

Privacy Protection: Unlike other tools, ALICE records hardware addresses of available network interfaces, which are unique to each device. In combination with the ability to record sent and received traffic of other applications, for example location reporting, ALICE creates a potential for privacy exposure, if user location, or other private data, is correlated to unique device ID.

Remaining Challenges: Currently ALICE does not support repeatable experiments on the same device, or set of devices, through device selection criteria. ALICE also does not support peer-to-peer experiments, or custom traffic transmissions, which limits the testbed's support for application prototyping.

3.3 Diagnosing Network Configurations

While most of the previous projects focus on measuring end-to-end performance of mobile application traffic, the following tools allow developers and researchers to learn more about the state of network infrastructure and configurations that affect transmission of application traffic. Pertinent features include the presence of proxy servers and other middleboxes, or complex multi-level DNS resolutions.

3.3.1 NDT (Mobile Client)

The Network Diagnostic Test (NDT) system, developed by Internet2, evaluates the performance of mobile connections to diagnose problems that limit network bandwidth [35], [95]. NDT also detects problems associated with device misconfiguration and network infrastructure. NDT (Mobile) is currently hosted on Google's M-Lab and allows access to its backend through an Android mobile application.

Functionality: NDT measurements are performed from a mobile Web browser that issues requests to NDT servers, hosted by M-Lab. The server-specific tests diagnose network problems encountered by connected clients. After the measurement experiment completes, the server analyzes the results and returns them to the client device.

Data Collection: The NDT mobile application collects traffic performance information such as upload and download speed, round trip network latency (minimum, average, and maximum), jitter, TCP receive window size (current and maximum), packet loss, TCP retransmission timer, and number of selective acknowledgements received. The application also detects router cable faults, incorrectly set TCP buffers in the device, duplex mismatch conditions on Ethernet links, presence of NAT, and capacity limits.

1. Namehelp Mobile measures the DNS performance of Cellular ISPs and public DNS resolvers, including the performance of CDN replicas [92]
2. Application Time allows users to track their application usage on their mobile device [93].
3. NU Signlas allows users to diagnoses Wi-Fi problems [94].

Resource Incentives and Protection: The incentive model for the NDT mobile client is based on providing network diagnostic information in exchange for users running traffic tests on their mobile devices. One issue for users who volunteer their device resources is that NDT requires permission to prevent the phone from going into power save mode, which may drain the battery quickly.

Privacy Protection: By default NDT records experimental data separately for each user, which allows users to privately diagnose their network problems. Data isolation also prevents malicious users from learning of open ports and interfaces in others' networks.

Remaining Challenges and Limitations: The NDT mobile client executes experiments that evaluate network traffic only between a mobile device and its closest M-Lab server. NDT creates a potential risk to user privacy by recording the remote phone number of caller, if the device is on a call during a network test.

3.3.2 Netalyzr

Netalyzr (mobile version), developed as a collaboration of ICSI Berkeley, UC Berkeley, HIIT, and Aalto University, is a diagnostic tool that characterizes connectivity, performance anomalies, and network security issues [36], [96]. The tools measures network latency and bandwidth to reveal insight into not only performance to cloud servers, but also how middleboxes in the path affect the performance of traffic. As of March 2014, Netalyzr has run over 15000 times to diagnose 290 operators in 90 countries. Netalyzr is accessible via an Android mobile application available on the Google Play Store.

Functionality: Netalyzr identifies the presence of Network Address Translations (NATs), proxy servers along a route, IP fragmentation, size of bottleneck buffers, reachability of services, and presence of HTTP proxies. When the Netalyzr application starts, it contacts the Netalyzr's Web server, which issues a DNS lookup request to redirect the user's request randomly to one of the twenty Netalyzr's back-end servers hosted on the Amazon cloud. Each back-end server supports twelve concurrent measurement sessions.

Netalyzr detects the presence of a NAT based on a difference between a user's local and public IP addresses. For clients behind a NAT, Netalyzr identifies how the network renumbers addresses and ports, i.e., whether the NAT uses fixed associations of local IP addresses to different public IP addresses, or if the NAT uses load-balancing.

To detect support for IP fragmentation, Netalyzr sends a 2 KB UDP packet (larger than 1500 B Ethernet maximum transmission unit (MTU)) to the server – a response from the server indicates the network supports fragmentation. If there is no response Netalyzr uses binary search to find the maximum packet size it can deliver without the packet being fragmented at the IP layer. The same test repeats from server to client to detect network support for fragmentation on the reverse path.

The sizing of bottleneck buffers affects user-perceived latency, and is measured based on the difference in latency during inactivity and during path throughput tests. Finally, queue drain time indicates the size of the buffer. To perform service reachability related experiments, the application attempts to connect to 25 different well known ports on a back-end server.

Netalyzr infers the presence of HTTP proxies if the public IP address in the request received by the back-end server is not the same as the client's public IP address. To detect the presence of in-path HTTP proxy, the client first sends an HTTP request to the server, the server then returns the request headers it received in the request back to the client. The client then compares the headers it sent and the headers the server sent to the client for any added, deleted or modified fields. To detect the presence of caching policies, the application relies on the HTTP 304 Not Modified response from the server.

To detect the presence of a DNS-proxy server or firewall, the application sends a DNS request to Netalyzr's back-end server. If the client detects any change in the response (different transaction ID, or public IP address), then Netalyzr assumes an in-path DNS proxy exists. Netalyzr then makes invalid DNS requests to the back-end server. If the client receives an invalid response from the server, nothing is detected, but if the request is blocked, Netalyzer assumes a DNS-aware middlebox blocking invalid DNS requests from leaving the network.

Data Collection: The Netalyzr mobile app records the presence of network interfaces, gateways, NAT detection, port renumbering, path MTU, packet fragmentation, DNS resolver, extension mechanisms for DNS (EDNS) support, port randomization, IPv6 support, hidden proxies, in-path caches, header manipulation, image transcoding, compression, HTTP type filtering, port filtering, traffic differentiation, IP fragmentation, signal-to-noise ratio, Wi-Fi/cellular configuration, network topology through traceroute, TLS handshake, UPnP vulnerabilities on Wi-Fi APs, clock drift, and TLS default certificates [96].

Resource Incentives and Protection: Netalyzr provides network diagnostic and troubleshooting information to users. Netalyzr requests user permission to modify system settings and to terminate other running applications in order to increase measurement accuracy. The Netalyzr mobile application asks users for permission to execute IP traceroutes, since ICMP packet transmission on a mobile device requires access to raw sockets.

Privacy Protection: Netalyzr asks users to opt in to data collection. With permission, the application can modify or delete the contents of USB storage, use GPS to get device location, read phone status and identity.

Remaining Challenges and Limitations: Although Netalyzr provides a robust diagnostic set of end-to-end network measurements and helps users troubleshoot networks, unlike MITATE, or WindRider, Netalyzr does not detect traffic shaping in mobile ISPs.

3.4 Experimenting with Cellular Infrastructure

Improving the performance of data delivery mechanisms internal to cellular networks requires privileged access to cellular infrastructure. Such access would enable researchers to experiment with novel protocols in cellular networks, free from the constraints of production deployment.

3.4.1 SciWiNet

Science Wireless Network (SciWiNet), being developed by Jim Martin *et al.* at Clemson University, is a NSF-funded re-seller of network infrastructure, based on Mobile Virtual Network Operator (MVNO) model, which provides the research community with a service on Sprint's cellular network infrastructure (and T-Mobile's infrastructure by late 2014) [37]. SciWiNet supports experimentation over 3G and 4G cellular networks, but without support for SMS, MMS, or voice services. SciWiNet provides additional infrastructure to the research community in the form of a shared pool of wireless devices (smartphones and USB LTE dongles), a common set of Android applications (WiFi hotspot, VPN tunnels, performance monitoring programs), and a set of wireless network services (VPN tunnel termination, secure database backend, performance monitor servers and backend).

Deployment: The SciWiNet project has two proposed project phases and is in phase-I as of September 2014. In phase-I, the project aims to determine the potential user community for SciWiNet infrastructure and investigate capabilities that it should support. In phase-II, the project will develop, deploy and operate the functional SciWiNet network infrastructure based on what was learned in phase-I.

Device support: Since SciWiNet uses Sprint's cellular network as its back-end cellular infrastructure, Sprint maintains a whitelist of mobile devices that are authorized to access SciWiNet's network and therefore eliminates the need to install a SIM card in every mobile device. Although, SciWiNet records device MAC address, it does not make the device MAC publicly available. SciWiNet maintains a list of popular devices and blacklisted devices. iOS devices are excluded because they do not support re-seller networks [97]. SciWiNet helps researchers access testbed resources by providing them with 1-2 mobile devices and a prepaid data plan for a limited time, typically six months. Alternatively researchers can access SciWiNet from their own devices and SciWiNet covers part of the data usage costs.

Data Collection: Currently, SciWiNet Android app collects the following network measurements over cellular and Wi-Fi networks: throughput for TCP and UDP traffic flows, packet loss, and ping latency. It can also detect location-based services such as base station identity, location, and wireless signal strength.

Privacy policy: Users can login to their account to check their data usage, or data contributed by others to their experiments. Data usage is limited by a leaky bucket rate limiter, where a user receives a number of tokens, which he can share among multiple devices. Once the data rate is exceeded, the device is temporarily restricted from accessing the SciWiNet network.

Remaining Challenges and Limitations: As of September 2014, it is unclear how SciWiNet will provide access to its devices and network resources to the research and developer community. One possibility is to offer incentives for user participation by providing free or discounted device access.

3.4.2 PhantomNet

PhantomNet, being developed by Jacobus Van der Merwe *et al.* at University of Utah, is an emerging testbed based on a network of small-cell base stations connected through a software-defined network (SDN) backbone [38]. Users will be able to not only experiment with end-to-end services, but also modify backbone traffic forwarding for their experiments. PhantomNet devices will have dual-radio interfaces, which will allow integration with a reseller network, for example through SciWiNet. PhantomNet also leverages management tools from other systems, notably Emulab and Seattle. Currently, PhantomNet remains under development.

4 Conclusions

This survey provides a comprehensive overview of the existing and emerging end-to-end mobile network measurement testbeds. In spite of the relative maturity of existing tools, several functionality gaps remain with respect to the needs of developers, researchers, and regulators in assessing mobile network performance. First, existing tools do not adequately support detection of traffic shaping. As depicted in Table 1, testbeds such as MITATE, Seattle, PhoneLab, PortoLan, and WindRider can detect the presence of traffic shaping mechanisms in mobile ISPs, whereas, other testbeds do not. Second, device churn inherent in tools based on ad-hoc user participation means that existing tools are not well-suited for long-term network performance monitoring. In fact, the popularity of tools such as PhoneLab and FCC has declined over time. Third, several tools enable

developers to prototype the performance of their applications ahead of deployment. However there is significant disparity in how testbeds provide that functionality in terms of execution models and APIs. Finally, exchange of P2P traffic, network diagnostics, ICMP traceroutes, device selection criteria, and NAT traversal are not only selectively supported by different testbeds.

Based on the surveyed work, we believe the mobile network measurement community needs a more concerted effort among developers, researchers, and regulators to produce network measurement tools that meet the needs of all three communities. A more concerted effort would lead to greater adoption of (perhaps fewer) tools, as well as large-scale and long-term network monitoring. At the same time, funding agencies should support development of new measurement approaches and capabilities, especially when such improvements are aimed at enhancement of existing testbeds.

Acknowledgments

The authors would like to thank Justin Cappos, Geoffrey Challen, David Choffness, Nick Feamster, Walter Johnston, Valerio Luconi, Jim Martin, John Rula, Mario Sanchez, Qing Yang, and Hongyi Yao for suggested improvements to an early version of this manuscript.

References

[1] "USTREAM," http://www.ustream.tv/, Jul. 2014.
[2] J. Codorniou, "Whats next for social mobile games?" http://techcrunch.com/2012/12/22/whats-next-for-social-mobile-games/, Dec. 2012.
[3] K. Rogers, "What's next after WhatsApp: a guide to the future of messaging apps," http://www.theguardian.com/technology/2014/feb/21/whatsapp-facebook-messaging-apps-viber-kik, Feb. 2014.
[4] P. Marupaka, "The future looks bright for augmented reality," http://www.siggraph.org/discover/news/future-looks-bright-augmented-reality, May 2014.
[5] K. Shubber, "Microsoft kinect used to live-translate sign language into text," http://www.wired.co.uk/news/archive/2013-07/18/sign-language-translation-kinect, Jul. 2013.
[6] I. Rimington, "Leave your wallet at home and pay with your profile picture," https://www.paypal.co.uk/Blog/Leave-your-wallet-at-home-and-pay-with-your-profile-picture/, Aug. 2013.
[7] I. Grigorik, *High Performance Browser Networking*. O'Reilly, 2013.
[8] C. Zhang, C. Huang, P. A. Chou, J. Li, S. Mehrotra, K. W. Ross, H. Chen, F. Livni, and J. Thaler, "Pangolin: speeding up concurrent messaging for cloud-based social gaming," in *ACM CoNEXT*, December 2011.
[9] S. Agarwal and J. R. Lorch, "Matchmaking for online games and other latency-sensitive P2P systems," in *ACM SIGCOMM*, August 2009.
[10] J. Erman, V. Gopalakrishnan, R. Jana, and K. K. Ramakrishnan, "Towards a SPDY'Ier Mobile Web?" in *ACM CoNEXT*, Dec. 2013.
[11] J. Butler, W. Lee, B. McQuade, and K. Mixter, "A Proposal for Shared Dictionary Compression over HTTP," http://lists.w3.org/Archives/Public/ietf-http-wg/2008JulSep/att-0441/Shared_Dictionary_Compression_over_HTTP.pdf, Sep. 2008.
[12] Akamai, "Free Mobile Web Performance Measurement Tool," http://mobitest.akamai.com/m/index.cgi, 2012.
[13] "Perfecto Mobile," http://www.perfectomobile.com/, 2014.
[14] W. Johnston, "Measuring Broadband America," http://www.caida.org/workshops/aims/1403/slides/aims1403_wjohnston.pdf, Mar. 2013.
[15] Readwrite, "Net Neutrality: What Happens Now That Verizon Has Vanquished The FCC," http://readwrite.com/2014/01/15/net-neutrality-fcc-verizon-open-internet-order, Jan 2014.
[16] B. Zevenbergen, I. Brown, J. Wright, and D. Erdos, "Ethical Privacy Guidelines for Mobile Connectivity Measurements," http://www.oii.ox.ac.uk/research/projects/?id=107, 2013.
[17] J. Huang, C. Chen, Y. Pei, Z. Wang, Z. Qian, F. Qian, B. Tiwana, Q. Xu, Z. M. Mao, M. Zhang, and P. Bahlc, "MobiPerf: Mobile Network Measurement System (Technical report)," University of Michigan and Microsoft Research, Tech. Rep., 2011.
[18] k. claffy, D. D. Clark, and M. P. Wittie, "The 6th Workshop on Active Internet Measurements (AIMS-6) Report," *Sigcomm CCR*, October 2014.
[19] "What is Measurement Lab?" http://www.measurementlab.net/about, 2014.
[20] U. Goel, A. Miyyapuram, M. P. Wittie, and Q. Yang, "MITATE: Mobile Internet Testbed for Application Traffic Experimentation," in *Mobile and Ubiquitous Systems: Computing, Networking and Services (Mobiquitous)*, Dec. 2013.
[21] K. Winstein, A. Sivaraman, and H. Balakrishnan, "Stochastic forecasts achieve high throughput and low delay over cellular networks," in *USENIX NSDI*, Apr. 2013.
[22] S. Higginbotham, "Traffic Shaping Coming to a Mobile Network Near You," https://gigaom.com/2011/04/05/traffic-shaping-coming-to-a-mobile-network-near-you/, Apr 2011.
[23] M. Marcon, M. Dischinger, K. P. Gummadi, and A. Vahdat, "The Local and Global Effects of Traffic Shaping in the Internet," in *Communication Systems and Networks (COMSNETS)*, Jan 2011.
[24] Librato, "One Platform. Unlimited Metrics. Monitoring Zen." http://metrics.librato.com/, Sep. 2014.
[25] "Google Analytics," http://www.google.com/analytics/, Sep. 2014.
[26] Y. Zhuang, A. Rafetseder, and J. Cappos, "Experience with seattle: A community platform for research and education," in *GENI Research and Educational Workshop (GREE)*, Mar. 2013.
[27] A. Nandugudi, A. Maiti, T. Ki, F. Bulut, M. Demirbas, T. Kosar, C. Qiao, S. Y. Ko, and G. Challen, "PhoneLab: A Large Programmable Smartphone Testbed," in *Workshop on Sensing and Big Data Mining*, Nov. 2013.
[28] A. Faggiani, E. Gregori, L. Lenzini, V. Luconi, and A. Vecchio, "Network sensing through smartphone-based crowdsourcing," in *Embedded Networked Sensor Systems (SenSys)*, Nov. 2013.
[29] A. Striegel, S. Liu, L. Meng, C. Poellabauer, D. Hachen, and O. Lizardo, "Lessons learned from the netsense smartphone study," in *ACM Workshop on HotPlanet*, ser. HotPlanet '13, Aug. 2013.
[30] R. K. Balan, A. Misra, and Y. Lee, "Livelabs: Building an in-situ real-time mobile experimentation testbed," in *Workshop on Mobile Computing Systems and Applications (HotMobile)*, Feb. 2014.
[31] MobiPerf, "Welcome to MobiPerf," http://www.mobiperf.com/, 2014.
[32] I. Trestian, R. Potharaju, and A. Kuzmanovic, "Closing the Loop: Feedback at Your Fingertips," http://www.cs.northwestern.edu/~ict992/docs/draft.pdf, 2009.
[33] K. Claffy, "The 5th Workshop on Active Internet Measurements (AIMS-5) Report," in *ACM SIGCOMM Computer Communication Review, Volume 43, Number 3*, July 2013.
[34] H. Yao, A. Nikravesh, Y. Jia, D. R. Choffnes, and Z. M. Mao, "Mobilyzer: A Network Measurement Library for Android Platform," in *Workshop on Active Internet Measurements (AIMS)*, Mar 2014.
[35] MLAB, "NDT (Mobile Client)," http://www.measurementlab.net/tools/ndt-mobile, 2013.
[36] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, "Netalyzr: Illuminating the edge network," in *ACM SIGCOMM Conference on Internet Measurement*, Nov. 2010.
[37] "SciWiNet," http://sciwinet.org/, 2014.
[38] K. V. d. Merwe, "PhantomNet: An end-to-end mobile network testbed," http://www.caida.org/workshops/aims/1403/slides/aims1403_jvandermerwe.pdf, Mar. 2014.
[39] M. A. Sánchez, J. S. Otto, Z. S. Bischof, D. R. Choffnes, F. E. Bustamante, B. Krishnamurthy, and W. Willinger, "Dasu: Pushing experiments to the Internet's edge," in *USENIX NSDI*, Apr. 2013.
[40] MLAB, "WindRider," http://www.measurementlab.net/tools/windrider, 2009.
[41] K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating latency between arbitrary Internet end hosts," in *SIGCOMM Workshop on Internet Measurement*, Nov. 2002.

[42] "M-Lab Tests," http://www.measurementlab.net/tests, 2014.
[43] J. Rula, "ALICE - Mobile Experiment Engine," http://aqualab.cs.northwestern.edu/projects/alice, Aug. 2014.
[44] FCC, "Measuring Broadband America," http://www.fcc.gov/measuring-broadband-america/mobile, Nov 2013.
[45] WindRider, "WindRider A Mobile Network Neutrality Monitoring System," http://www.cs.northwestern.edu/~ict992/mobile.htm, Oct 2009.
[46] C. Osborne, "The state of LTE 4G networks worldwide in 2014 and the poor performance of the US," http://www.zdnet.com/the-state-of-lte-4g-networks-worldwide-in-2014-and-the-poor-performance-of-the-us-7000026594/, Feb. 2014.
[47] J.D. Power, "Overall wireless network problem rates differ considerably based on type of service," http://www.jdpower.com/press-releases/2013-us-wireless-network-quality-performance-study-volume-2, Aug. 2013.
[48] T. Karr, "Verizon's Plan to Break the Internet," http://www.savetheinternet.com/blog/2013/09/18/verizons-plan-break-internet, Sept 2013.
[49] C. Aaron, "Net Neutrality Is Dead. Here's How to Get It Back," http://www.savetheinternet.com/blog/2014/01/14/net-neutrality-dead-heres-how-get-it-back, Jan 2014.
[50] M. Fahey, "Why Gamers Should Care About Net Neutrality," http://kotaku.com/5512448/why-gamers-should-care-about-net-neutrality, Apr 2010.
[51] S. Buckley, "Cogent and Orange France fight over interconnection issues," http://www.fiercetelecom.com/story/cogent-and-orange-france-fight-over-interconnection-issues/2011-08-31, Aug 2011.
[52] ——, "France Telecom and Google entangled in peering fight," http://www.fiercetelecom.com/story/france-telecom-and-google-entangled-peering-fight/2013-01-07, Jan 2013.
[53] A. Lynn, "Cable Companies' Big Internet Swindle," http://www.freepress.net/blog/2009/11/24, Nov 2009.
[54] N. Anderson, "Huge ISPs want per-GB payments from Netflix, YouTube," http://arstechnica.com/tech-policy/2011/01/huge-isps-want-per-gb-payments-from-netflix-youtube/, Jan 2011.
[55] R. Singel, "Mobile Carriers Dream of Charging per Page," http://www.wired.com/business/2010/12/carriers-net-neutrality-tiers/2/, Dec 2010.
[56] B. Cohen, "Incentives build robustness in BitTorrent," in *Workshop on Peer-to-Peer Systems (IPTPS)*, Feb. 2003.
[57] J. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson, "Seattle: a platform for educational cloud computing," in *ACM SIGCSE Bulletin*, Mar 2009.
[58] MITATE, "MITATE : Mobile Internet Testbed for Application Traffic Experimentation (User Manual)," http://mitate.cs.montana.edu/sample/MITATE_Documentation_v1.0.pdf, Nov 2013.
[59] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: an overlay testbed for broad-coverage services," *SIGCOMM CCR*, vol. 33, no. 3, pp. 3–12, Jul. 2003.
[60] C. Siaterlis, A. Garcia, and B. Genge, "On the use of emulab testbeds for scientifically rigorous experiments," *IEEE Communications Surveys Tutorials*, vol. 15, no. 2, pp. 929–942, Feb. 2013.
[61] "Geni," http://www.geni.net/, Oct. 2012.
[62] yanyan, "Using Sensors In Seattle," https://seattle.poly.edu/wiki/UsingSensors, Apr 2012.
[63] "Sensibility testbed," http://sensibilitytestbed.com, Jul. 2014.
[64] PhoneLab, "PhoneLab A Programmable Smartphone Testbed," http://www.phone-lab.org/, 2013.
[65] "RIPE Atlas," http://atlas.ripe.net/, Jul. 2013.
[66] PhoneLab, "Overview," http://participate.phone-lab.org/info/, 2013.
[67] LiveLabs, "Participation," http://livelabs.smu.edu.sg/participant/, Mar. 2013.
[68] "LiveLabs Registration," http://athena.smu.edu.sg/livelabs_register/, Oct 2014.
[69] E. Gregori, L. Lenzini, V. Luconi, and A. Vecchio, "Sensing the Internet through crowdsourcing," in *Proceedings of the Second IEEE PerCom Workshop on the Impact of Human Mobility in Pervasive Systems and Applications (PerMoby)*, May 2013.
[70] "Portolan network tools," https://play.google.com/store/apps/details?id=it.unipi.iet.portolan.traceroute&hl=en, May 2014.
[71] FCCAPPs, "FCC Speed Test App," https://play.google.com/store/apps/details?id=com.samknows.fcc&hl=en, Dec 2013.
[72] J. Clover, "FCC Launches 'FCC Speed Test' iPhone App to Measure Mobile Broadband Performance," http://www.macrumors.com/2014/02/25/fcc-speed-test/, Feb 2014.
[73] FCC, "Measuring Mobile Broadband Methodology - Technical Summary," http://www.fcc.gov/measuring-broadband-america/mobile/technical-summary, Nov. 2013.
[74] J. Kastrenakes, "FCC releases Android speed test app to gather data on cell carrier performance," http://www.theverge.com/2013/11/14/5105090/fcc-launches-android-mobile-speed-test-app, Nov 2011.
[75] S. Silbert, "FCC launches speed test app for Android, looks to collect mobile broadband performance data," http://www.engadget.com/2013/11/14/fcc-launches-speed-test-app-android/, Nov 2011.
[76] Z. Honig, "FCC Speed Test app for iOS lets the government track your iPhone's network performance," http://www.engadget.com/2014/02/25/fcc-speed-test-app-ios/, Feb 2014.
[77] K. Bell, "FCC Launches iOS 'Speed Test' App," http://mashable.com/2014/02/25/fcc-speed-test-app-ios/, Feb 2014.
[78] FCC, "FCC Speed Test App Tip Sheet," https://www.fcc.gov/guides/mobile-speed-test-tip-sheet, 2014.
[79] S. Rosen, H. Luo, Q. A. Chen, Z. M. Mao, J. Hui, A. Drake, and K. La, "Discovering Fine-grained RRC State Dynamics and Performance Impacts in Cellular Networks," in *ACM Mobicom*, Sep. 2014.
[80] MobiPerf, "Data Collection and Privacy Policy," http://www.mobiperf.com/privacy, 2014.
[81] Y. Zhou, "Mobiperf," http://www.caida.org/workshops/isma/1302/slides/aims1302_yyzhou.pdf, Feb 2013.
[82] S. Muckaden, "MySpeedTest: active and passive measurements of cellular data networks," Ph.D. dissertation, Georgia Institute of Technology, 2013.
[83] N. Feamster, "My Speed Test Mobile Performance Measurement Tool Released," http://noise-lab.net/2012/06/02/my-speed-test-mobile-performance-measurement-tool-released/, Jun. 2012.
[84] S. Muckaden, "MySpeedTest: Active and Passive Measurements of Cellular Data Network Performance," http://www.caida.org/workshops/isma/1302/slides/aims1302_smuckaden.pdf, Feb. 2013.
[85] Webpagetest, "Test a website's performance," http://www.webpagetest.org/, 2008.
[86] Guy Podjarny, "Open-Sourcing Mobitest," https://blogs.akamai.com/2012/03/open-sourcing-mobitest.html, Mar 2012.
[87] A. Dyke, "What is a HAR File and what do I use it for?" http://www.speedawarenessmonth.com/what-is-a-har-file-and-what-do-i-use-it-for/, Aug 2012.
[88] Akamai, "Test Your Website Performance On A Mobile Device," http://www.akamai.com/html/awe/login.html?campaign_id=F-MC-16282&curl=/html/awe_auth/mobitest.html, 2012.
[89] M. Piatek, "Measurement @ Google," http://www.caida.org/workshops/aims/1403/slides/aims1403_mpiatek.pdf, Mar. 2014.
[90] "Akamai Mobitest: Mobile Web Performance Measurement Agents," https://code.google.com/p/mobitest-agent/source/browse/trunk/mobitest-agent/Android/BZAgent/README?r=2, Mar 2012.
[91] J. Rula, "ALICE - Technical Description," http://aqualab.cs.northwestern.edu/262-details-alice, Aug. 2014.
[92] Aqualab, "Namehelp," https://play.google.com/store/apps/details?id=edu.northwestern.aqualab.namehelp&hl=en, Apr 2014.
[93] ——, "Application Time (AppT)," https://play.google.com/store/apps/details?id=edu.northwestern.aqualab.behavior.research, Apr 2014.
[94] NUStudents, "NU Signals v2," https://play.google.com/store/apps/details?id=edu.northwestern.nux, May 2014.
[95] Internet2, "Network Diagnostic Tool (NDT)," http://software.internet2.edu/ndt/, 2013.
[96] N. Vallina-Rodriguez, N. Weaver, C. Kreibich, and V. Paxson, "Netalyzr for Android: Challenges and opportunities," in *Workshop on Active Internet Measurements (AIMS)*, Mar 2014.
[97] "Devices Supported by SciWiNet," http://sciwinet.org/SciWiNet-Devices.html, 2014.