MITATE: Mobile Internet Testbed for Application Traffic Experimentation

Utkarsh Goel, Ajay Miyyapuram, Mike P. Wittie(✉), and Qing Yang

Department of Computer Science,
Montana State University, Bozeman, MT 59717, USA
mwittie@cs.montana.edu

Abstract. This paper introduces a Mobile Internet Testbed for Application Traffic Experimentation (MITATE). MITATE is the first programmable testbed to support the prototyping of application communications between mobiles and cloud datacenters. We describe novel solutions to device security and resource sharing behind MITATE. Finally, we show how MITATE can answer network performance questions crucial to mobile application design.

Keywords: Mobile networks · Testbed · Application · Performance

1 Introduction

Innovative mobile applications, such as multiplayer games and augmented reality, will require low message delay to provide a high quality of user experience (QoE) [1,2]. Low message delay, in turn, depends on low network latency and high available bandwidth between mobile devices and cloud datacenters, on which application back-end logic is deployed. Unfortunately, mobile network performance can change rapidly [3]. Worse, traffic shaping mechanisms in cellular networks, such as as cap-and-throttle, traffic redundancy elimination, and deep packet inspection (DPI), can delay application messages without being reflected in standard metrics of network performance [4–6].

If innovation in the mobile space is to achieve broad adoption, new applications must deliver a high QoE across a range of network conditions. In other words, application communication protocols must be smart enough to adapt to changing network performance to keep message delay low. Such adaptations might include changing packet size, or moving between server endpoints to deliver best traffic performance for a given client [3,7].

To design and validate adaptive communication protocols developers need to prototype their implementations in production networks. The research community has produced several testbeds capable of application prototyping in the wired Internet [8–20]. To date, however, cellular network measurement platforms are not *programmable* in that they do not provide an foreign code execution environment [21–26]. Instead applications are evaluated in network simulators

I. Stojmenovic et al. (Eds.): MOBIQUITOUS 2013, LNICST 131, pp. 224–236, 2014.
DOI: 10.1007/978-3-319-11569-6_18

configured to reflect measurements of network performance [3]. While measurement-based simulation allows repeatable experiments, it misses the dynamic effects of competing traffic in cellular schedulers and of traffic shaping mechanisms.

The technical problem we address in this paper is a lack of a programmable testbed for mobile application prototyping in production cellular networks. We have identified two challenges to building such a testbed. First, the personal nature of mobile devices creates user concerns over privacy, accountability for actions of foreign code being prototyped, and abuse of limited data plan and battery resources. Striking a balance between a flexible application prototyping environment and the safe execution of foreign code has been a difficult problem even in the more permissive wired environment [10,18]. Second, because mobile battery and data plan resources are limited, testbed participants need adequate incentives to share them. Difficulty in enlisting mobile users has limited measurement studies to small samples [26], high cost of testbeds based on dedicated hardware [27], and collection of only high level network performance metrics [24].

In this paper we describe MITATE – a Mobile Internet Testbed for Application Traffic Experimentation made possible by novel solutions to the problems of security and mobile resource sharing. MITATE is unique in that it allows programmable application traffic experiments between mobile hosts and backend server infrastructure. MITATE provides strong client security by separating application code execution from traffic generation. MITATE also provides incentives and protections for mobile resource sharing through tit-for-tat mechanisms. MITATE's specialized traffic experiments can help developers answer questions crucial to mobile application design such as: "What is the largest game state update message that can be reliably delivered under 100 ms?," "Does my application traffic need to contend with traffic shaping mechanisms?," or "Which CDN provides fastest downloads through a particular mobile service provider's network peering points?"

The remainder of this paper is organized as follows. Section 2 covers related research. In Sect. 3 we describe MITATE's architecture. Section 4 shows MITATE application prototyping capabilities. Finally, we conclude and present directions for future work in Sect. 5.

2 Related Work

The research community has produced several testbeds capable of application prototyping in the wired Internet [8–20]. To date, however, cellular network measurement platforms are not *programmable* in that they do not provide a foreign code execution environment [21–26]. The result is a functionality gap: new applications are either evaluated on a small number of mobile devices, or in network simulators [3,28]. While small scale studies capture real application performance, they miss variation across geographic areas, carriers, and devices. On the other hand, simulation studies configured to reflect aggregate measures of network performance miss the dynamic effects of traffic shaping and cellular schedulers [3–6].

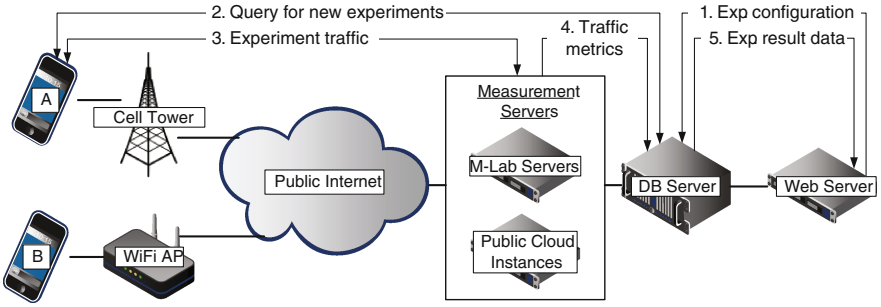

Fig. 1. MITATE architecture and steps of a network traffic experiment.

Existing testbeds share some features with MITATE, such as criteria-based filtering of testbed devices [26], (limited) evaluation of application layer mechanisms such as HTTP and DNS [23], and an M-Lab[1] back-end [24]. Closest to our approach is Dasu, which provides a custom execution environment within an extension to a PC BitTorrent client [18]. SatelliteLab is also similar to MITATE in that prototyped application logic is not executed on edge devices [13].

One mobile testbed with programmable features is PhoneLab, which provides 200 participants with mobile phones and discounted data plans [27]. In exchange, participants agree to network experiments executed on their phones. However, PhoneLab relies on a custom OS, which limits its deployment to dedicated hardware, since installing an OS is a significant barrier to entry for most users [13].

3 MITATE

MITATE goes beyond current work and allows application prototyping on mobile devices in production cellular networks. MITATE offers the flexibility of Dasu and SatelliteLab, but without the security vulnerabilities of mobile code [13,18]. To achieve wider adoption and easier access than the dedicated hardware model of PhoneLab, we adapt proven resource sharing incentives [27,29]. In this section, we describe MITATE's architecture, application prototyping capabilities, and address the challenges of security and resource sharing on mobile devices.

3.1 Architecture and Traffic Experiments

To register a device with MITATE, a user downloads our mobile application and starts it as a background service with her login credentials, obtained by creating a MITATE account. Once her device is registered, a user can conduct traffic experiments, referring to Fig. 1, as follows: In Step 1, a user creates an experiment by uploading a configuration file, described in Sect. 3.2, via the Web

[1] http://measurementlab.org

```
<experiment>
 <transfer>
  <id>t1</id>
  <src>client</src>
  <dst>54.243.176.74</dst>
  <prot>UDP</prot>
  <dstport>5060</dstport>
  <bytes>32</bytes>
 </transfer>
 <transfer>
  <id>t2</id>
  <src>54.243.176.74</src>
  <dst>client</dst>
  <prot>UDP</prot>
  <srcport>5060</srcport>
  <bytes>512</bytes>
 </transfer>
```

```
 <criteria>
  <id>c1</id>
  <latlong>"45.666 -111.046"<\latlong>
  <radius>5000<radius>
  <networktype>cellular</networktype>
  <starttime>12:00</starttime>
  <endtime>13:30</endtime>
 </criteria>
 <transaction count="10">
  <criteria>
   <criteriaid>c1</criteriaid>
  </criteria>
  <transfers>
   <transferid>t1</transferid>
   <transferid delay="40">t2</transferid>
   <transferid>t1</transferid>
  </transfers>
 </transaction>
</experiment>
```

Fig. 2. MITATE XML configuration file.

interface. In Step 2, MITATE devices query the database for new experiments, whose criteria they meet. To reduce resource contention, as in SatelliteLab, we allow only one experiment at a time on a device [13]. If device A, for example, meets the geographic location and network type criteria of an experiment, A will begin, in Step 3, to transfer data defined by the experiment to the measurement servers. Experiment transfer traffic is timed at each endpoint (mobiles and measurement servers) and network performance metrics, together with metadata, are reported back to the database in Step 4. Finally in Step 5, a user may access the Web interface again to visualize, or download the experiment data collected by multiple devices. Based on the collected data, the user may refine her experiment and restart the process from Step 1.

3.2 Programmable Network Traffic Experiment Configuration

MITATE offers a flexible programming environment that supports evaluation and optimization of existing application traffic traces, as well as prototyping of adaptive application communication protocols. Existing network testbeds support such flexibility through mobile code, whose potential security vulnerabilities result in designs based on dedicated testbed hardware [8,27], or execution environments constrained by custom APIs [10,18]. Neither solution is satisfactory. While the dedicated hardware limits adoption, custom APIs require application reimplementation in restricted, or non-standard programming environments.

We propose a secure and flexible network testbed design that eliminates the drawbacks of mobile code. MITATE experiments use multiple rounds of statically defined traffic transmissions. Processing between the rounds, i.e. mobile application logic, is implemented offline. Offline processing allows for the execution of unmodified application code inside an emulator[2] with message transmissions

[2] http://developer.android.com/tools/help/emulator.html

delegated to MITATE. Offline processing can also optimize communication protocol parameters, such as packet size, through binary parameter search, or a more powerful approach, such as CPLEX.[3] Finally, static experiment definitions allow static verification, which simplifies resource management (Sect. 3.3) and testbed security design (Sect. 3.4) and leads to a more accessible testbed.

Application Traffic Trace Experiments can help answer questions such as "What is the largest game state update message that can be reliably delivered under 100 ms?" An abbreviated MITATE experiment configuration XML file in Fig. 2 specifies two transfers, `t1` and `t2`. The transfers transmit the specified number of `bytes` between a MITATE mobile `client` and a datacenter server IP with MITATE backend logic.

The configuration file also specifies criteria definitions that `client` endpoints must meet before executing an experiment. In the Fig. 2 example, criteria `c1`, requires that a mobile be within 5000 m of geographic coordinates 45.666 -111.046 (Bozeman, MT), be connected to a cellular network, and that device time be between noon and 1:30PM. MITATE will allow experimenters to specify a wide set of criteria, for example radio signal strength, location (eg. radius, bounding box, or set of ZIP codes), availability of GPS (indoor/outdoor), or device travel speed (for example over 55mph).

Finally, configuration files specify one, or more transactions that group criteria and transfers. In the Fig. 2 example, there is one transaction, which conceptually reflects a user request (transfer `t1`), game state update (transfer `t2`) after 40 ms of server processing `delay`, and an acknowledgement (transfer `t1`). This transaction will be executed by a mobile device if the device satisfies transaction criteria when polling MITATE servers, fewer than `count` devices have completed the transaction, and the user issuing the experiment has sufficient test data credit (see Sect. 3.3) to execute the entire transaction.

To find the largest game state update that can be delivered under 100 ms, multiple experiment rounds can perform binary parameter search, with MITATE reporting individual transfer and overall transaction delays. MITATE can also be used with sophisticated optimization tools, such as CPLEX, where performance of intermediate solutions are the reported metrics in each experiment round. Because MITATE traffic experiments use production networks they are not necessarily repeatable, and so decision metrics should be averaged over multiple trials. Finally, a `repeat` attribute can indicate that a transfer, or a transaction, should be executed multiple times. These `repeat` and `delay` attributes can be combined to configure periodic traffic, for example polling every 10 min for 24 h.

Programmable Application Traffic Experiments can help answer questions such as "Which CDN provides fastest downloads through a particular mobile service provider's peering points?" To measure download times an experiment needs to issue a DNS lookup, followed by a download from the resolved

[3] www.ibm.com/software/commerce/optimization/cplex-optimizer/

```
<transfer>
 <id>dns_req</id>
 <src>client</src>
 <dst>DNS</dst>
 <dstport>53</dstport>
 <prot>UDP</prot>
 <bytes><![CDATA[0x0100be07de55...]]></bytes>
 <response>1</response>
</transfer>
```

Fig. 3. DNS query in MITATE.

server addresses. MITATE supports such experiments with two mechanism: explicit packet content and device-specific scheduling.

Figure 3 shows a configuration of transfer `dns_req` that represents a DNS lookup for a CDN server. The `bytes` tag contains the explicitly specified bytes of a well-formed DNS lookup request. When the `response` tag is set to `1`, the DNS reply packet will be included in the result data set, from which a user can parse out the resolved IP addresses.

To measure the download time of an image hosted on a particular CDN network, the user would configure a second experiment with a well-formed HTTP `GET` request to each resolved server IP. To make sure that each mobile device contacts only the IP addresses it resolved, each MITATE measurement contains the unique ID of the device that collected the result. That ID can be subsequently used as an endpoint address instead of the "`client`" keyword.

One downside of our approach is a potential for delay between each round of transmissions as experiments wait to be scheduled on mobile devices. We are working on integrating MITATE with the Android emulator to make the process of experiment configuration as easy as writing to a socket. Our integration will carefully modify emulator clocks, so that they advance only by measured transmission delay, excluding experiment scheduling delay. This mechanism will allow studies of adaptive communication mechanisms, such as server-host switching in online games, implemented in native application code running inside the emulation with only traffic transmissions being delegated to MITATE.

3.3 Deployment Incentives

One of the challenges faced by mobile network measurement platforms is how to assure sufficient resource capacity for scheduled experiments. The limiting resource is mobile data, subject to monthly caps.[4] To assure a supply of mobile bandwidth that matches the demand, a mobile testbed must, first, entice users to contribute resources and, second, protect contributed resources from abuse. MITATE jointly addresses both problems using a data *credit* exchange system inspired by BitTorrent tit-for-tat mechanisms [29].

The insight behind BitTorrent's tit-for-tat mechanisms is that they reward users for contributing bandwidth, as well as for merely being willing to do so.

[4] While battery power is also limited, it can be more easily replenished by charging.

While in BitTorrent users make this assessment vis-a-vis each other, MITATE accounts for contribution and willingness to contribute with respect to the system as a whole. A MITATE user earns bandwidth credit for her experiments by allowing others' experiments to run on her device. A user is considered willing to contribute when her devices reliably ping MITATE servers for new experiments. The credit earned by the user, x_{earned}, is computed daily as:

$$x_{\text{earned}} = \alpha \times x_{\max} \times \min\left(\frac{x_{\text{contributed}}}{x_{\max}} + \frac{p_{\text{actual}}}{p_{\text{expected}}}, 1\right),$$

where $x_{\max}$ is the remaining amount of mobile data a user is willing to contribute during a monthly billing cycle divided by remaining number of days, $x_{\text{contributed}}$ is the volume of mobile data used by MITATE experiments on the user's data plan, p_{actual} is the number of pings reaching MITATE servers within 24 hours, and p_{expected} is the expected number of pings based on a system wide ping frequency setting. The parameter $\alpha < 1$ creates a mismatch between contributed resources and earned credit intended to ensure high experiment completion rates in areas with fewer participating devices, such as rural states. We recalculate user credit every 24 hours to prevent users from accumulating credit that, if used all at once, could deplete system resources on any given day. We expect that some participants will use MITATE sporadically and others on ongoing basis. Similar user participation takes place in BitTorrent, yet the system as a whole is able to maintain a sustained capacity [29].

Thus, MITATE credits users for contributed bandwidth, which allows them to use the bandwidth of others, keeping the two in a state of equilibrium. A final element of the mechanism to prevent resource abuse is that daily experiment bandwidth requirements are computed at submission time, a process facilitated by the static XML experiment definition, and checked against submitting user's credit before being admitted to the system. We believe this approach is more predictable than resource caps enforced at run time that can lead to low experiment completion rates [10]. We also believe MITATE's credit based approach is simpler and more democratic than the delegated trust approach proposed in NIMI [30].

3.4 Security and Privacy

MITATE's goal of open-access necessitates a well thought out security design. With the contributed data plan resources protected by the incentive mechanisms, the security goals focus on protection of user privacy, the volunteered devices, and non-MITATE Internet resources.

Protecting User Privacy. MITATE runs on personal mobile devices, which has the potential for violations of privacy if a device owner's activity and personally identifiable information were to become public. For example, user network and calling activity is not only private, but may itself contain personally

identifiable information. Similarly GPS data becoming public can lead to legal challenges if traffic laws (speeding), or property laws (trespassing) were violated.

We have designed multiple levels of protection to preclude violations of user privacy. First, MITATE can only be used for active traffic experiments and cannot monitor non-MITATE traffic on a device. Second, while MITATE does collect GPS and accelerometer readings as metadata to accompany network performance metrics, users are asked to opt-in before starting the MITATE mobile app. Finally, third, we separate all data collected on devices from personally identifiable user account information. Each device registered with MITATE receives two random IDs: one to label traffic metrics collected on the device, the other to keep track of credit data earned by the device for its owner. The dual ID system means that collected experiment data are never linked to a device owner's identifiable information.

Protecting User Devices. Users who volunteer their devices for MITATE agree to cede some control over them. It is imperative that MITATE limit other user's actions on volunteered devices to within the bounds of that agreement. MITATE protects user devices with three mechanisms.

First, a user can set usage limits for mobile data, WiFi data, and battery level on their devices. These limits are consulted during experiment scheduling to disallow experiments that exceed remaining device resource allowance. Second, users never directly interact with others' devices. To submit an experiment, or download data, users authenticate and communicate with MITATE servers over encrypted connections. Mobile devices download experiments and upload collected metrics to MITATE servers also using encryption. Finally, third, our XML experiment configuration is static in that it does not allow conditional, nor jump statements. Such static definitions enforce the separation between the on-device functionality of data transmission and off-device processing. This separation allows for static checking of XML configurations using mature schema verification tools, which is simpler than dynamic code analysis and more lightweight than mobile code sandboxing. Static experiment definition also allows for the volume of each transfer in the XML file to be added up and compared against user credit and device resource limits.

Protecting Non-MITATE Resources. Our final goal is to protect non-MITATE resources, for example from DDoS attacks configured as MITATE experiments. ScriptRoute, designed from the ground up as a secure Internet measurement system, considers two types of malicious experiments: *magic* packets and traffic amplification [10]. *Magic* packets can disrupt legitimate traffic, for example, when a spoofed `FIN` packet closes a TCP connection. Because MITATE allows experiments with explicitly defined packet content, we will make sure that these packets do not pose threats to other systems by matching them against signatures of known exploits using intrusion detection mechanisms.

Traffic amplification takes place when a malicious user leverages testbed nodes to monopolize the resources of a legitimate service, for example through

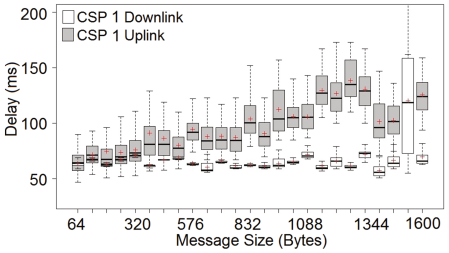

Fig. 4. Message delay vs. message size at 10 AM on CSP 1 to a CA datacenter.

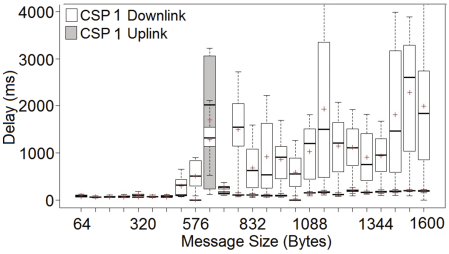

Fig. 5. Message delay vs. message size at 2 PM on CSP 1 to a CA datacenter.

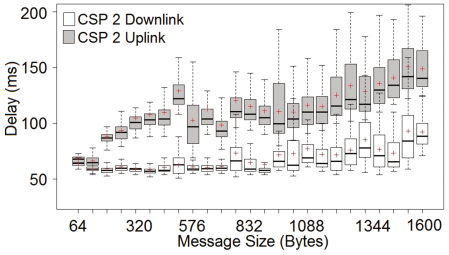

Fig. 6. Message delay vs. message size at 10 AM on CSP 2 to a CA datacenter.

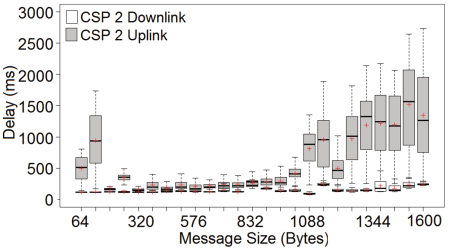

Fig. 7. Message delay vs. message size at 10 AM on CSP 2 to a VA datacenter.

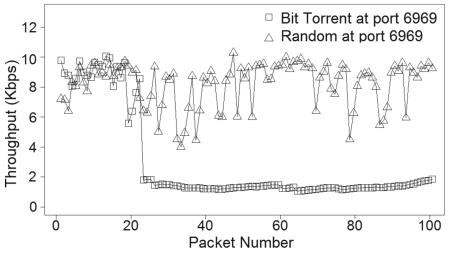

Fig. 8. Per packet throughput of BitTorrent and random payloads on CSP 1.

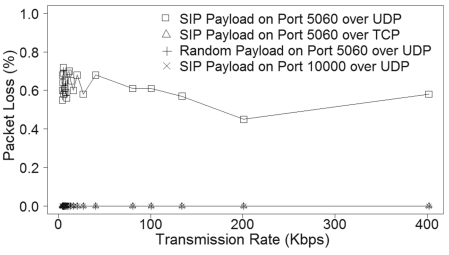

Fig. 9. Packet loss of SIP and random payloads vs. flow data rate on CSP 1.

a Smurf attack. Existing testbeds limit traffic amplification by placing a rate limit on the volume of data that can be generated by an experiment, which also constrains legitimate load testing. Instead, MITATE limits the total volume of experiment data to a user's earned credit. Although a MITATE user may request that multiple devices send data simultaneously, the user's credit will be rapidly depleted, and so even if the transmissions are malicious, they will be short-lived.

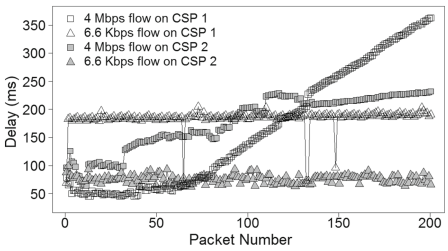

Fig. 10. Delay of different data rate flows vs. on CSP 1 and CSP 2.

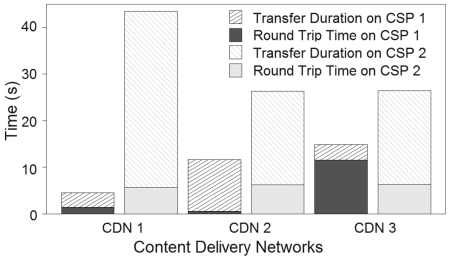

Fig. 11. Round trip time and transfer time of 3 MB image from three CDNs.

4 MITATE Application Traffic Prototyping Capability

To demonstrate MITATE's traffic emulation capabilities we present a set of network experiments and collected data. We show that MITATE can elicit various network performance phenomena useful to developers in answering a wide range questions about application traffic performance. The collected data includes traffic performance metrics and associated metadata. Prior to sending experiment traffic, MITATE calculates the clock offset between the mobile and measurement servers, which allows us to time unidirectional (unacknowledged) UDP transfers [31]. Experiments were performed on several Android phones and two different cellular service providers (CSP) networks in Bozeman, MT, and over connections to two different cloud datacenters. We anonymize the identities of CSPs and CDNs.

4.1 Effect of Packet Size on Message Delay

In gaming applications game state updates need to be delivered while their content is relevant. And so, game developers may want to know: "What is the largest game state update message that can be reliably delivered under 100 ms?" To answer that question we configure a MITATE experiment with transfers of increasing size (`bytes`). We plot the results in Figs. 4, 5, 6, 7, which show message delay as a function of message size during different times of day.

Our results show that message delay increases with message size and does so more rapidly on the uplink, likely due to asymmetric network provisioning. We also observe in Fig. 5 a high delay for larger messages on the downlink, likely due to mid-day network congestion. Figure 6 shows a higher sensitivity of message delay to size on CSP 2. That effect is especially pronounced on connections to a datacenter located in Virginia, shown in Fig. 7.

From these experiments a developer might conclude that a message of 320 B can be delivered under 100 ms with high confidence to customers in Bozeman, MT on CSP 1, but a smaller message might be needed on CSP 2. Also, to keep message delay low, requests from Bozeman should not be directed to the Virginia datacenter.

4.2 Effect of Traffic Shaping

The degree to which FCC net neutrality rules apply to CSPs continues to be debated [32]. And so, application developers may want to ask: "Does my application traffic need to contend with CSP traffic shaping mechanisms?" To answer that question we configure a series of MITATE experiments, in which transfers of specific content, on specific ports, and at different rates are used to detect traffic shaping [5,33].

Figure 8 shows downlink throughput on CSP 1 of consecutive BitTorrent and random payloads transmitted over UDP on tracker port 6969. Our results show a drop in throughput for well-formed BitTorrent packets relative to random content, which likely indicates the presence of DPI mechanisms. We did not detect similar throughput drops on CSP 2. These results show that embedding of explicit packet payloads allows MITATE to detect content based traffic shaping.

Figure 9 shows downlink percent packet loss on CSP 1 of 1000 SIP packets transmitted on port 5060 over UDP and TCP versus transmission rate. Our results show that while SIP packets over TCP are undisturbed, same packets over UDP experience close to 60 % loss rate. Because loss remains nearly constant across transmission rates, we believe that SIP packet loss over UDP is due to traffic policing, rather than traffic shaping.

Figure 10 shows per packet delay of uplink UDP flows transmitted at 4 Mbps and 6.6 Kbps on CSP 1 and CSP 2 versus packet number. The 4 Mbps flows experience an increase in delay, likely from queueing that results from the mismatch between sending and token bucket service rate limits [5]. The 6.6 Kbps flows, on the other hand, are sent below the service rate and avoid self-induced congestion. Testing different transmission rates allows developers to determine the maximum sending rate that will fall below token generation rate and avoid queuing delays. The experiments are useful for configuration of adaptive video stream encoding.

4.3 Measurement Based CDN Selection

Finally, dynamic content applications customize content for each user and have the opportunity to adapt to user's network conditions, for example, by embedding links to static content in different CDNs. And so, application developers may want to ask: "Which CDN provides fastest downloads through a particular mobile service provider's network peering points?" To answer that question we configure a MITATE experiment that sends a well-formed HTTP `GET` requests, configured in the `bytes` tag, for an image hosted in three different CDNs.

Figure 11 shows the CDN response time for the first bit, or round trip time (RTT), and last bit, or transfer duration, of a 3 MB image delivered over the two CSP networks. Our results show a lower last bit delay for requests in CSP 1, but a higher RTT variation between CDNs, likely due to different CSP peering points that lead to CDN servers. From these experiments a developer might conclude that for users in Bozeman, MT CDN 2 provides the best combination of performance across the two CSP networks.

5 Discussion and Future Work

In this paper we described MITATE, the first public testbed that supports prototyping of application communications between mobiles and cloud datacenters. MITATE separates application logic from traffic generation, which simplifies security and resource sharing mechanisms. We have presented data collected with MITATE experiments that demonstrates the system's capability in eliciting effects of cellular network performance on mobile application message delay.

Future work on the project involves deploying the current implementation onto M-Lab servers. In the meantime, we invite the community to use publicly available MITATE code[5] in private deployments. We also welcome community participation in evolving MITATE functionality in the areas of resource sharing models, GPS and accelerometer data anonymization, data visualization, and tools based on the MITATE platform.

References

1. Chen, K.-T., Huang, P., Lei, C.-L.: Effect of network quality on player departure behavior in online games. Parallel Distrib. Syst. **20**, 593–606 (2009)
2. Geerts, D., Vaishnavi, I., Mekuria, R., van Deventer, O., Cesar, P.: Are we in sync?: synchronization requirements for watching online video together. In: SIGCHI Conference on Human Factors in Computing Systems, May 2011
3. Winstein, K., Sivaraman, A., Balakrishnan, H.: Stochastic forecasts achieve high throughput and low delay over cellular networks. In: USENIX NSDI, Apr 2013
4. Zohar, E., Cidon, I., Mokryn, O.O.: Celleration: loss-resilient traffic redundancy elimination for cellular data. In: Workshop on Mobile Computing Systems (HotMobile), Feb 2012
5. Kanuparthy, P., Dovrolis, C.: ShaperProbe: end-to-end detection of ISP traffic shaping using active methods. In: ACM IMC, Nov 2011
6. Lu, X., Cao, W., Huang, X., Huang, F., He, L., Yang, W., Wang, S., Zhang, X., Chen, H.: A real implementation of DPI in 3G network. In: Global Telecommunications Conference (GLOBECOM), Dec 2010
7. Wittie, M.P., Pejovic, V., Deek, L., Almeroth, K.C., Zhao, B.Y.: Exploiting locality of interest in online social networks. In: ACM CoNEXT, Nov 2010
8. Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., Bowman, M.: PlanetLab: an overlay testbed for broad-coverage services. SIGCOMM Comput. Commun. Rev. **33**, 3–12 (2003)
9. Andersen, D.G., Balakrishnan, H., Kaashoek, M.F., Morris, R.: Experience with an evolving overlay network testbed. SIGCOMM CCR **33**, 13–19 (2003)
10. Spring, N., Wetherall, D., Anderson, T.: Scriptroute: a public Internet measurement facility. In: USENIX Symposium on Internet Technologies and Systems (USITS), Mar 2003
11. Simpson Jr., C.R., Riley, G.F.: NETI@home: a distributed approach to collecting end-to-end network performance measurements. In: Barakat, C., Pratt, I. (eds.) PAM 2004. LNCS, vol. 3015, pp. 168–174. Springer, Heidelberg (2004)

[5] http://github.com/msu-netlab/MITATE

12. Bavier, A., Feamster, N., Huang, M., Peterson, L., Rexford, J.: In VINI veritas: realistic and controlled network experimentation. In: ACM SIGCOMM, Aug 2006
13. Dischinger, M., Haeberlen, A., Beschastnikh, I., Gummadi, K.P., Saroiu, S.: SatelliteLab: adding heterogeneity to planetary-scale network testbeds. In: ACM SIGCOMM, Aug 2008
14. Choffnes, D.R., Bustamante, F.E., Ge, Z.: Crowdsourcing service-level network event monitoring. In: ACM SIGCOMM, Aug 2010
15. Manweiler, J., Agarwal, S., Zhang, M., Roy Choudhury, R., Bahl, P.: Switchboard: a matchmaking system for multiplayer mobile games. In: ACM MobiSys, June 2011
16. Geni, Oct 2012. http://www.geni.net/
17. FIRE: Future Internet Research and Experimentation, July 2013. http://www.ict-fire.eu/
18. Sánchez, M.A., Otto, J.S., Bischof, Z.S., Choffnes, D.R., Bustamante, F.E., Krishnamurthy, B., Willinger, W.: Dasu: pushing experiments to the Internet's edge. In: USENIX NSDI, Apr 2013
19. Archipelago measurement infrastructure, June 2013. http://www.caida.org/projects/ark/
20. Ripe atlas, July 2013. http://atlas.ripe.net/
21. Wittie, M.P., Stone-Gross, B., Almeroth, K.C., Belding, E.M.: MIST: cellular data network measurement for mobile applications. In: Conference on Broadband Communications, Networks and Systems (BROADNETS), Sept 2007
22. Austin, M., Wish, M.: The official story on AT&T Mark the Spot, Oct 2010. http://www.research.att.com/articles/featured_stories/2010_09/201009_MTS.html
23. Huang, J., Xu, Q., Tiwana, B., Mao, Z.M., Zhang, M., Bahl, P.: Anatomizing application performance differences on smartphones. In: ACM MobiSys, June 2010
24. MobiPerf, Welcome to MobiPerf. http://www.mobiperf.com/home, Feb 2012
25. ROOT Metrics. http://www.rootmetrics.com/, July 2013
26. Gember, A., Akella, A., Pang, J., Varshavsky, A., Caceres, R.: Obtaining in-context measurements of cellular network performance. In: ACM IMC, Nov 2012
27. Baldawa, R., et al.: PhoneLab: A large-scale participatory smartphone testbed. In: USENIX NSDI poster session, Apr 2012
28. Gao, J., Sivaraman, A., Agarwal, N., Li, H., Peh, L.: DIPLOMA: Consistent and coherent shared memory over mobile phones. In: International Conference on Computer Design (ICCD), Sept 2012
29. Cohen, B.: Incentives build robustness in BitTorrent. In: International Workshop on Peer-To-Peer Systems (IPTPS), Feb 2003
30. Paxson, V., Mahdavi, J., Adams, A., Mathis, M.: An architecture for large scale Internet measurement. IEEE Commun. **36**, 48–54 (1998)
31. Mills, D.L.: Network Time Protocol (version 2) specification and implementation. Network Working Group Request for Comments: 1119, Sept 1989
32. Ammori, M.: The next big battle in Internet policy. http://www.slate.com/articles/technology/future_tense/2012/10/network_neutrality_the_fcc_and_the_internet_of_things_.html, Oct 2012
33. Dischinger, M., Marcon, M., Guha, S., Gummadi, K.P., Mahajan, R., Saroiu, S.: Glasnost: enabling end users to detect traffic differentiation. In: USENIX NSDI, Apr 2010