Measuring mobile performance in the Tor network with OnionPerf

Ana Custura
University of Aberdeen

Iain Learmonth
Tor Project

Gorry Fairhurst
University of Aberdeen

***Abstract*—The Tor network is the largest public deployed anonymity network using the Internet. While there has been a longitudinal study into the performance of the network in progress since 2009, it has only used vantage points in data centre networks. In this paper we propose modifications to the performance measurement tool, OnionPerf, to enable its use for measuring performance from a mobile end-user's perspective. We provide initial findings on simulated mobile networks, using two types of emulated links, using both the public Tor network and in a private test Tor network.**

I. Introduction

The Tor anonymity network [1] is the largest deployed public anonymity network on the Internet. It is used by more than 2 million users each day [2]. On mobile devices it is most commonly used through Tor Browser for Android[1], which has between 1 million and 5 million installs. Its precursor, Orfox[2], has over 10 million installs. This is evidence that many Tor clients are using mobile networks.

Despite the popularity and importance of Tor on mobile networks, there is no study of the performance operating over mobile networks, largely because of a lack of suitable tools. This paper explores how OnionPerf[3], a performance measurement tool for the Tor network, can be used to measure mobile networks. Ahead of measurements using deployed mobile infrastructure, we provide results on simulated mobile networks, using two emulation methods. Finally, we discuss the implications of adapting OnionPerf to measure mobile networks and identify changes needed to ensure the measurement methodology conforms to the best practices for measuring capacity-constrained networks.

II. Background and Related Work

A. The Tor network

The Tor network comprises over 6500 relays [3] that can forward traffic for many users at the same time, and can be part of an arbitrary number of circuits. Each Tor circuit is a tunnelled connection through three Tor relays, respectively called the guard, middle and exit. User connections pass, encrypted, through these three relays before exiting from the Tor network onto the public Internet [4].

[1]https://play.google.com/store/apps/details?id=org.torproject.torbrowser_alpha

[2]https://play.google.com/store/apps/details?id=info.guardianproject.orfox

[3]https://gitweb.torproject.org/onionperf.git

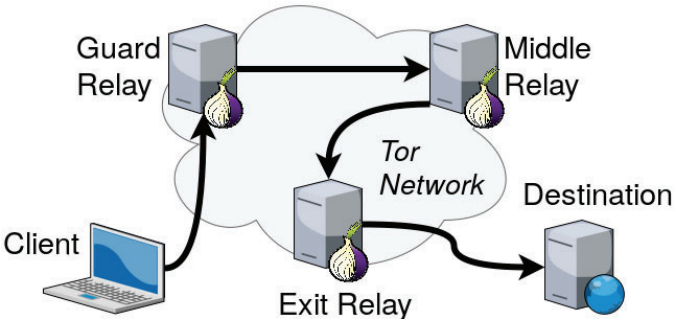

Fig. 1. An illustration of a connection through the Tor network

The user software, or client, first contacts the Tor Directory Authorities to obtain a list of available relays. These Directory Authorities are a group of semi-trusted servers that keep track of all running healthy relays. Directory authorities periodically publish a list of all relay addresses. A Tor client will choose a set of relays from this list to build their circuits when an application, such as Tor Browser, wishes to make a connection (Figure 1). The client then incrementally builds a circuit of encrypted connections through these relays, extending it one hop at a time. The client negotiates separate sets of encryption keys for each hop. No individual relay ever knows the complete path that a data packet takes. After passing (encrypted) through the Tor circuit, the traffic exits unencrypted and travels to the destination.

Tor uses the transmission control protocol (TCP) for connections from the Tor client to the guard relay, connections between the relays in the circuit, and the connection from the exit to the destination. Within the Tor network (including from the client to the guard) the transport layer security (TLS) protocol is used between nodes and a Tor-specific protocol is used end-to-end [5].

Tor also allows users to run, and connect to, Onion Services. These are self-published services, made available through Tor relays that act as "rendezvous points". A Tor user can connect to these Onion Services, without knowing the other network's identity by establishing a connection to the rendezvous point, which joins it to the circuit of the Onion Service.

978-3-903176-17-1 / © 2019 IFIP

B. Passive Measurement

The goals of a privacy and anonymity network like Tor are not easily combined with extensive data gathering, but at the same time data is needed for monitoring, understanding, and improving the network. The Tor Metrics project [6] aggregates passive measurements of capacity used in the Tor network. Relays in the network collect statistics over a period of 24 hours [7]. Noise is added to some statistics to provide differential privacy. This data helps understand the rates achievable within the Tor network, but can only determine the performance of an "average" client.

When a part of the Tor circuit is carried on a mobile network, the network characteristics can be different and the performance therefore also is not the same. The existing data and tools do not capture mobile network performance.

C. Simple Bandwidth Scanner (sbws)

sbws[4] is an active measurement tool designed to measure the performance of individual relays in the Tor network. It uses a "two-hop" circuit rather than the traditional "three-hop" to assign weights to relays to balance load across the Tor network. sbws therefore does not try to measure the absolute performance of a relay, but rather the speed of a relay compared to other relays. While this serves a useful role in supporting operation of the Tor network, and has an interesting methodology, it is not suitable for evaluating end-user performance.

D. Tor and Mobile networks

Many Tor users rely on mobile connectivity, but the service available varies around the world. Despite efforts by service operators to improve performance to support streaming media content and interactive applications, the mobile network service in many world regions is still limited by second generation access technologies. As discussed by [8], these parts of the world are correlated with low political instability, where anonymity is critical for an end-user.

Mobility scenarios [9] introduce variations in network conditions, which can result in packet loss and higher end-to-end delays. Service performance over Tor is sensitive to this loss and is also impacted by network latency, a key performance parameter for users. Latency can be different in a mobile context, however, there are currently no latency-related metrics gathered by the passive measurement.

A correlation was found between packet loss and mobility and changes in Radio Access Technology, which could induce temporary loss of service. Typically, in stationary scenarios, packet loss is less than 1%, but this if often not achieved for mobile services.

Each Tor relay establishes a single TCP connection to another relay when a circuit is built. Traffic from all client connections that take a path over the same two consecutive relays will be carried using a single TCP connection between the relays. Consequently, any packet loss, delay or disruption to the TCP packets that comprise this flow will affect all connections sharing that part of the path, cascading the effect of network impairments down to impact the performance experienced by users.

Tor treats application-level data as a bytestream, and avoids dealing with TCP segments. This avoids the inefficiencies arising from interacting congestion control timers seen when tunneling TCP over TCP [10]. Tor relays also act like routers. They have buffers and are subject to bufferbloat, introducing latency in the Tor network. Bufferbloat is also an issue with significant impact on tunnels in general [11]. Latency across mobile networks varies by access technology[12], and is influenced by many factors. 5G mobile infrastructure promises to reduce the mobile segment latency, but this is an area where more experimentation will be required. In another example, latency can be increased when roaming, because operators tunnel user traffic back to their home country, increasing delay proportional to the geographic distance [13].

Encapsulation of an end-to-end transport using TCP involves an additional layer of state, shared by all traffic inside the tunnel, resulting in performance considerations that need to be understood. Recent work [11] in the IETF is exploring the impacts of such TCP encapsulation.

E. Tor Services

A Tor client can make information requests from the public Internet, or using an Onion Service. In the first case, Tor treats application connections as bytestreams attached to circuits. In the second case, both the client and Onion Service circuits connect at a common rendezvous point, adding another layer of complexity and overhead to the process. A study of Onion Service performance [8] in capacity-constrained networks found the bootstrap process for setting-up Onion Services takes significantly longer in such networks due to the data overheads associated with publishing Onion Services, which needs to be downloaded before building a circuit.

While historically used for Internet browsing, Tor is also being increasingly used to support real-time interactive applications, like WhatsApp and Facebook Messenger. These applications also demand a lower network latency.

III. Experiment Design

We ran OnionPerf experiments in two emulated scenarios, modelling EDGE, 3G and LTE network profiles. These experiments were repeated without the link emulation to provide a baseline test for comparison. The characteristics for the two different emulated profiles is summarised in Table I.

Parts of these experiments use the public Tor network. We have been careful to ensure our experiments do not degrade the security or anonymity properties of the Tor network for other users. Our experiments do consume bandwidth from the network, although we operate a relay at the University of Aberdeen to offset this. We are only measuring traffic that we have generated ourselves and only from the client, not from relays in the network.

[4]https://sbws.readthedocs.io/

Work is in progress in the IETF to document considerations for safe measurement of live networks [14]. Our measurements with OnionPerf implement these considerations, and have contributed to further refining them.

The following subsections describe the mode of operation of OnionPerf and its traffic generation components, as well as the setup of the emulated links.

A. OnionPerf

OnionPerf is a software tool currently used to measure Tor network performance. It uses multiple processes and threads to download random data through Tor, while tracking the performance of the downloads, fetched on localhost using traffic generator processes, and is transferred through Tor using Tor client processes and an ephemeral Tor Onion Service.

The traffic generator in OnionPerf uses TGen [5], a C application that models traffic behavior using an action-dependency graph. OnionPerf will run a TGen client/server pair that transfer traffic through Tor and through an ephemeral Onion Service started by OnionPerf. Each TGen node takes a graphml-formatted file as a parameter, and then begins transferring data to/from other nodes by following a path through the action graph. Currently, OnionPerf downloads 50 KB, 1 MB and 5 MB files using a probabilistic weighted action graph.

Tor control information and TGen performance statistics are logged to disk, and are analyzed once per day to produce a json stats database and files that can be used to visualize changes in Tor client performance over time[6].

OnionPerf (Figure 2) uses a *measure* component that controls the flow of all measurements performed by the tool. There are two measurement options:

- Files hosted on the local machine are downloaded via the Tor network using only a Tor client. This emulates access to the Internet via Tor.
- Files hosted as an Onion Service using a Tor server are downloaded via the Tor network using a Tor client. This emulates access to an Onion service via Tor.

B. OnionPerf Data

The Tor Metrics team has been operating OnionPerf instances since April 2017, and before that, Torperf instances since July 2009. The results of these measurements are released as open data via CollecTor[7].

The vantage points used for these instances have been in datacenters typically with the newest addition, known as `op-ab`[8], is operated at the University of Aberdeen. These vantage points have much higher capacity and lower latency than available to a typical mobile user.

In the earlier days of the Tor network, performance would change from day to day, as can be seen in Figure 3. Since around 2014, there has been less variation as the network matured. Recent measurements from the four current OnionPerf deployments are shown in Figure 4.

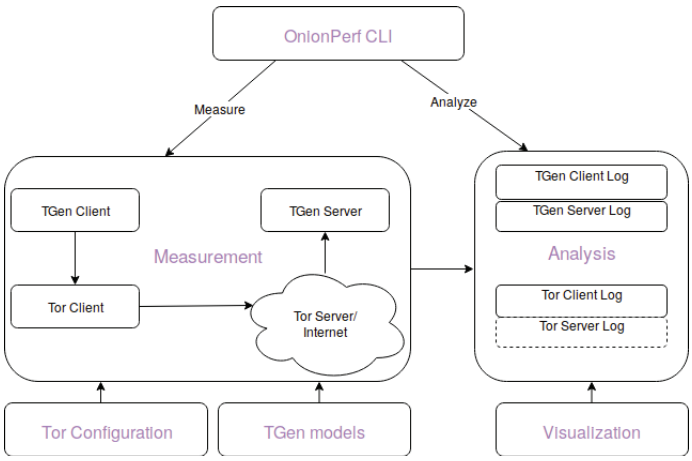

Fig. 2. The main OnionPerf components. Measurements are started via the CLI, and produce logs for analysis. As a part of the measurement process, OnionPerf stars a Tor client, a TGen client, a TGen server and optionally a Tor server.

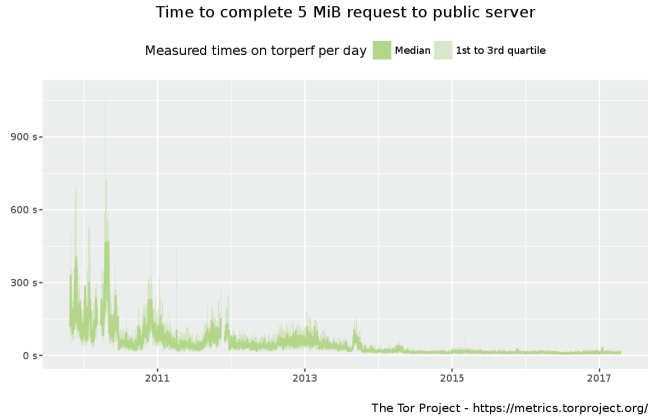

Fig. 3. Time to complete a 5MB download via a Tor exit circuit between 2009 and 2017.

C. Link emulation

In the first experiment, we ran OnionPerf from within chutney, a tool for creating test (private) Tor networks complete with emulated links between the test nodes.

The second experiment used netem to emulate connection parameters between the OnionPerf measurement host at the University of Aberdeen and the public Tor network.

1) Chutney: Chutney[9] is a Tor project tool for creating test networks, it handles configuring and launching of configurable Tor networks and the execution of tests on these networks.

Chutney emulates an entire Tor network by using a network configuration file which describes the Tor network nodes to be created. For our setup, we used the simplest Tor network,

[5]https://github.com/shadow/tgen

[6]https://metrics.torproject.org/torperf.html

[7]https://metrics.torproject.org/collector.html#torperf

[8]"ab" in this case is a pseudo-country code for Scotland, based on the Scottish Gaelic name, *Alba*.

[9]https://github.com/torproject/chutney/blob/master/README

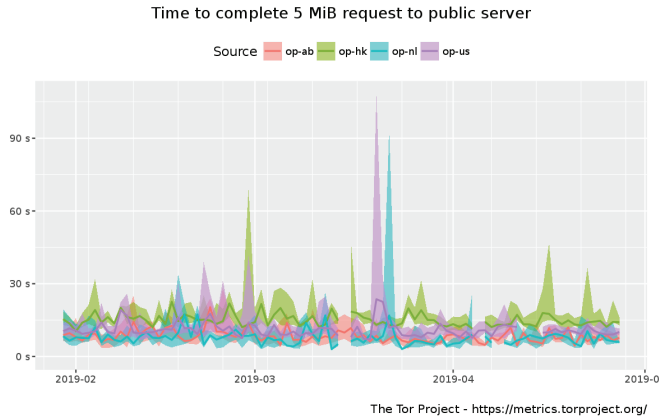

Fig. 4. Time to complete a 5MB download via a Tor exit circuit between February and May 2019.

Name	Downlink bandwidth	Units	Downlink delay (ms)	Uplink bandwidth	Units	Uplink delay (ms)
EDGE (2.5G)	240	kbps	400	200	Kbps	440
3G	780	kbps	100	330	Kbps	100
LTE	50	Mbps	50	10	Mbps	65

TABLE I
MOBILE NETWORK LINK CHARACTERISTICS BASED ON NETWORK LINK TECHNOLOGY

created from two directory authorities, one relay and one client node. The tor instance associated with each node starts in its own network namespace, meaning each tor process launched with Chutney has its own network stack that can emulate different link characteristics.

Chutney allows for different characteristics to be described as "profiles" at configuration setup. The link characteristics supported for emulation purposes are drop rate, bandwidth, and delay. These can be varied to emulate different types of links. For example, a Chutney configuration file can describe a 3G profile (Figure 5). This comprises three Directory Authorities and five Exit relays. There was no additional load traffic added to the test network. The setup used network profiles for emulating a range of links (Table I). The OnionPerf client ran from within Chutney to perform 10 downloads for each profile in the test network.

2) Netem: Chutney emulates not just the link properties of connections between clients and nodes in a Tor network, but also creates the nodes themselves. As such, it is a great testing tool but limited in predicting performance in a live, public network. We used Netem to emulate different link properties for the connection between the OnionPerf client and the public Tor network. Netem[10] uses Linux traffic control (TC) facilities to emulate network characteristics to the outgoing packets from a selected network interface. Netem is built using the existing Quality Of Service and Differentiated Services facilities in the Linux kernel. Similarly to `veth` interfaces,

[10]http://man7.org/linux/man-pages/man8/tc-netem.8.html

```
# 3G profile
profile = {
    "uplink":{"drop_rate":0, "bandwidth":330, "delay":100},
    "downlink":{"drop_rate":0, "bandwidth":780, "delay":100}
}

# By default, Authorities are not configured as exits
Authority = Node(tag="a", authority=1, relay=1,
                 torrc="authority.tmpl")
ExitRelay = Node(tag="r", relay=1, exit=1, torrc="relay.tmpl")
Client = Node(tag="c", client=1, torrc="client.tmpl",
                 network_profile=profile)

NODES = Authority.getN(3) + ExitRelay.getN(5) + Client.getN(1)

ConfigureNodes(NODES)
```

Fig. 5. Chutney emulated 3G network. A network profile is added to the client template. The parameters in this profile are used by the network emulator to enforce these network characteristics.

netem can virtualise properties of a real network interface, including variable delay, loss, duplication and re-ordering.

The outgoing interface of the OnionPerf measurement host, a VM hosted by the University of Aberdeen, was modified with netem to perform network emulation for this experiment. This interface was used by the OnionPerf client to connect to the public Tor network and perform 10 measurements with each of the profiles shown in Table I.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

Measurements using real network stacks can help understand the impact of access technology on connection performance, with the potential to improve current and future designs of the Tor protocol stack. In this section we present the results obtained for each of the simulated scenarios described above. In addition, we compare these to an initial study using a real cellular network stack. Figures 6 and 7 present the median time to download a 5MB object over the Tor network using the emulated network profiles. Each figure also presents baseline results for downloads on a data centre network link, emulated with each respective tool as a 1GB connection with no delay or loss.

To perform an initial study using a real cellular network, a Tor client was configured with guards disabled and a series of 5 MiB downloads were attempted via the public Tor network from a server hosted in the Amsterdam Digital Ocean datacenter. Of the set of downloads that completed, the average time to complete a download was 381 seconds, resulting in an average rate of 110kbps. The time to complete a download varied considerably, between 275 and 537 seconds, shown as a boxplot of the download times in Figure 8. This figure also presents the real cellular results side-by-side with the emulated results for EDGE networks. The median download time is higher in the real network versus its emulated counterparts, with more download time variability. The simulated measurements behave in a more deterministic fashion, and present optimistic download times compared to the measurement taken in a real environment. The emulated network measurements can therefore be regarded as 'best-case scenarios'. In the case of Chutney, the Tor network has no load, while netem can

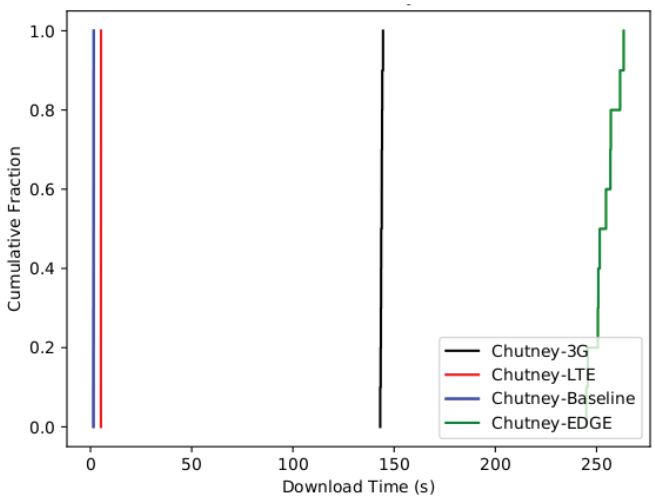

Fig. 6. Time to download 5242880 bytes, all downloads

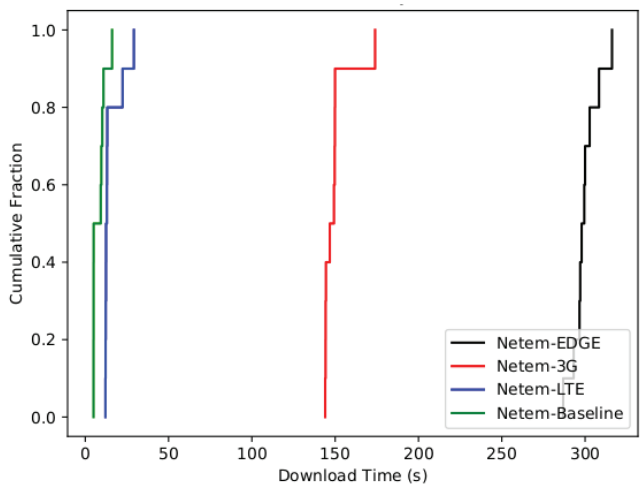

Fig. 7. Time to download 5242880 bytes, all downloads

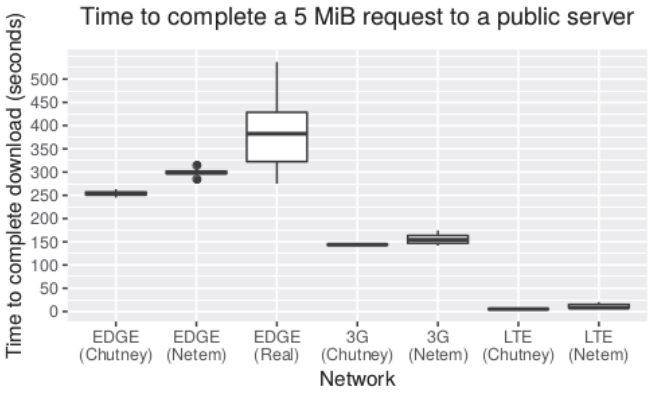

Fig. 8. Time to download 5242880 bytes, via an EDGE network (completed downloads only)

only simulate delay, capacity limits and packet loss using averages - real mobile networks present a higher variation across their characteristics, with results depending on time of day, cell load, propagation conditions, etc. While the simple methodology described above allows us to directly compare download times with the simulated EDGE network OnionPerf data, it does not provide the rich logging of the latter. In the future, measurements on real networks will be performed by OnionPerf, which provides detailed logging of Tor client and server behaviour, allowing for a nuanced analysis of the penalty on performance when building and extending Tor circuits, in addition to measuring the download performance.

A. Adapting OnionPerf for mobile platforms

The primary limitation to using OnionPerf on mobile platforms is that the tool consumes significant capacity when used to perform measurements, over 20GB of data per month. Mobile operators (and consequently, mobile measurement platforms) usually apply data volume limitations as a part of the purchased data plan. The authors therefore recognise that the methodology needs to be updated to reduce the volume of test data required to provide acceptable measurements using deployed mobile clients. A different size of transfer could be considered, as well as adding a delay between measurements. To achieve this, we plan to modify the TGen specifications for the download graph used by OnionPerf to tune both download sizes and time to wait between transfers. An alternative option could be to utilise a single larger download and to record a series of partial completion times as the download progresses, although this may result in less measurement diversity. On average, the 5M measurements represent 65% out of the total bytes Onionperf downloads as part of its normal operation, but only 5% of attempted downloads. For example, recording partial completion times on 5M files could result in a reduction of roughly 7GB a month - whereas, skipping these large downloads altogether and recording patial completion times from 1M downloads would bring the consumed data to a more manageable 5GB per month.

OnionPerf is inherently limited by how it operates - when performing downloads via the Internet, the traffic generator component requires measurement hosts to use a public IP address in a network that allows access to a port exposed by OnionPerf. This is not feasible for a typical mobile scenario, because the address of a measurement host would either be NAT-ed or limited by a firewall that is not under the control of the person making the measurements.

OnionPerf could be modified to listen for incoming connections on a different interface on measurement hosts connected via multiple interfaces. This can be implemented by modifying the traffic generation component, Tgen, to serve data on a separate interface. This software modification, will allow OnionPerf to run on mobile measurement platforms using dedicated multi-networked hardware, such as the MONROE platform [15]. This distributed measurement platform comprises over 200 nodes using over 10 mobile European broadband operators,and is made available to researchers on

an Experiment-as-a-Service basis. The platform nodes run a stripped-down version of Debian Linux and Docker to run mobile measurement experiments in a containerized fashion. OnionPerf is a self-contained tool which could be easily adapted to run in a Docker container, making it a perfect candidate for use on MONROE.

Finally, OnionPerf is designed to match a real client's behavior as far as possible. In this way it is similar to a black box and doesn't allow for observation of how the access link, the individual relays, or buffers, are contributing to the performance measurement. As it currently exists, OnionPerf can only validate that a change made, such as varying the access technology, results in either an improvement or reduction in performance but cannot identify where exactly that improvement or reduction has been made. This approach allows for safe measurement of the live Tor network without posing risks of harm to other users of the Tor network, with the downside that degradation in mobile network performance be distinguished from degradation in the Tor network.

B. Exploring Loss Recovery and Latency

While Tor performs tunelling of bytestream application data over TCP, multiple bytestreams may be multiplexed over a TCP stream. The Tor protocol implements both circuit-level and stream-level congestion control maintaining windows for the maximum number of data cells that can be in-flight without the receiver requesting more cells be sent. Stream-level congestion control is required because streams may have different destinations once they leave the Tor network, even though they are carried on the same circuit while within the network. As this congestion control is built on top of the existing congestion control mechanisms of TCP, any disruption in one of the flows can affect the end-to-end performance for the circuit. A new proposal revolving around multiplexing over QUIC [16], MASQUE (Multiplexed Application Substrate over QUIC Encryption) has the potential remove some of the head-of-line blocking and impact of loss recovery created by layering a tunnel over TCP/TLS, but has yet to be evaluated. Its design could inform future Tor protocol design and could in future make it more resilient on mobile networks.

V. Conclusion and next steps

The Tor network presents a unique point of study where mobile networks are concerned, due to its encrypted and layered nature. There have been very few studies on how the access technology used by this network impacts performance from an end-user perspective. We have explored the challenges in using OnionPerf, a tool historically used to measure download performance through Tor from data centres, for measuring mobile networks and provided suggestions for its improvement. The authors plan an implementation of an updated tool and methodology that will enable a longitudinal measurement study to explore performance from several mobile vantage points, using a range of access technologies, and traffic-modelling of popular content browsing.

VI. Acknowledgements

This work is funded by The University of Aberdeen. This support does not imply endorsement.

References

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[2] Tor Metrics, "Relay Users," https://metrics.torproject.org/userstats-relay-country.html.

[3] ——, "Relays and bridges," https://metrics.torproject.org/networksize.html.

[4] A. Debiasi, R. Dingledine, A. Edelstein, A. F. røy, M. Finkel, D. Goulet, K. Hanna, M. Haughey, G. Kadianakis, I. R. Learmonth, A. Macrina, and M. Xynou, "Addressing denial of service attacks on free and open communication on the internet," The Tor Project, Tech. Rep. 2018-11-001, November 2018. [Online]. Available: https://research.torproject.org/techreports/dos-censorship-report1-2018-11-19.pdf

[5] Tor Project, "Tor protocol specification," https://spec.torproject.org/tor-spec.

[6] K. Loesing, S. J. Murdoch, and R. Dingledine, "A case study on measuring statistical data in the Tor anonymity network," in *Proceedings of the Workshop on Ethics in Computer Security Research (WECSR 2010)*, ser. LNCS. Springer, January 2010.

[7] Tor Project, "Tor directory protocol, version 3," https://spec.torproject.org/dir-spec.

[8] J. Lenhard, K. Loesing, and G. Wirtz, "Performance measurements of tor hidden services in low-bandwidth access networks," in *Applied Cryptography and Network Security*, M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 324–341.

[9] D. Baltrunas and A. Elmokashfi and A. Kvalbein and O. Alay, "Investigating packet loss in mobile broadband networks under mobility," in *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, May 2016, pp. 225–233.

[10] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda, "Is It Still Possible to Extend TCP?" in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '11. Berlin, Germany: ACM, 2011, pp. 181–194. [Online]. Available: http://doi.acm.org/10.1145/2068816.2068834

[11] T. Pauly and E. Kinnear, "TCP Encapsulation Considerations," Internet Engineering Task Force, Internet-Draft draft-pauly-tsvwg-tcp-encapsulation-00, Jul. 2018, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-pauly-tsvwg-tcp-encapsulation-00

[12] M. Laner, P. Svoboda, P. Romirer-Maierhofer, N. Nikaein, F. Ricciato, and M. Rupp, "A comparison between one-way delays in operating hspa and lte networks," in *2012 10th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, May 2012, pp. 286–292.

[13] A. M. Mandalari, A. Lutu, A. Custura, A. Safari Khatouni, O. Alay, M. Bagnulo, V. Bajpai, A. Brunstrom, J. Ott, M. Mellia, and G. Fairhurst, "Experience: Implications of roaming in europe," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '18. New York, NY, USA: ACM, 2018, pp. 179–189. [Online]. Available: http://doi.acm.org/10.1145/3241539.3241577

[14] I. Learmonth, "Guidelines for Performing Safe Measurement on the Internet," Working Draft, IETF Secretariat, Internet-Draft draft-learmonth-pearg-safe-internet-measurement-01, December 2018, work in Progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-learmonth-pearg-safe-internet-measurement-01.txt

[15] MONROE, "Measuring Mobile Broadband Networks in Europe," https://www.monroe-project.eu/.

[16] D. Schinazi, "The MASQUE Protocol," Internet Engineering Task Force, Internet-Draft draft-schinazi-masque-00, Feb. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-schinazi-masque-00