Computer Communications xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Lightweight capacity measurements for mobile networks

Q1 Foivos Michelinakis[a,b,*], Nicola Bui[a,b], Guido Fioravantti[c], Joerg Widmer[a], Fabian Kaup[d], David Hausheer[d]

[a] IMDEA Networks Institute, Spain
Q2 [b] Universidad Carlos III de Madrid, Spain
[c] CartoDB Inc., Spain
[d] P2P Systems Engineering, Technische Universität Darmstadt, Germany

ARTICLE INFO

Article history:
Received 5 September 2015
Revised 6 February 2016
Accepted 13 February 2016
Available online xxx

Keywords:
Mobile capacity
Measurement
Packet dispersion

ABSTRACT

Mobile data traffic is increasing rapidly and wireless spectrum is becoming a more and more scarce resource. This makes it highly important to operate mobile networks efficiently. In this paper we are proposing a novel lightweight measurement technique that can be used as a basis for advanced resource optimization algorithms to be run on mobile phones. Our main idea leverages an original packet dispersion based technique to estimate per user capacity. This allows passive measurements by just sampling the existing mobile traffic. Our technique is able to efficiently filter outliers introduced by mobile network schedulers and phone hardware. In order to asses and verify our measurement technique, we apply it to a diverse dataset generated by both extensive simulations and a week-long measurement campaign spanning two cities in two countries, different radio technologies, and covering all times of the day. The results demonstrate that our technique is effective even if it is provided only with a small fraction of the exchanged packets of a flow. The only requirement for the input data is that it should consist of a few consecutive packets that are gathered periodically. This makes the measurement algorithm a good candidate for inclusion in OS libraries to allow for advanced resource optimization and application-level traffic scheduling, based on current and predicted future user capacity.

1. Introduction

Even though spectrum efficiency is improving thanks to the fifth generation [1] of mobile networks, the wireless medium is becoming a scarcer and scarcer resource, due to the ever increasing demand for mobile communication. Recently, a number of papers addressed improved resource allocation mechanisms based on capacity prediction techniques. For instance, [2–4] propose to use resources when they are more abundant and cheap, and to refrain from or to limit communication when it is more expensive (e.g., lower spectral efficiency, higher congestion, etc.) by exploiting perfect knowledge of the future capacity.

In [5], we surveyed the state of the art on mobile capacity prediction techniques and built a model for both short and medium to long term prediction errors in order to be able to quantify the impact of prediction uncertainties in resource allocation. Most short term prediction techniques [6,7] rely on time series filtering solutions, such as moving average and autoregressive (ARMA) or autoregressive conditional heteroskedasticity (ARCH) modeling. Thus, in order to allocate resources on a given time granularity, prediction must be available with the same granularity and, consequently, mobiles must be able to measure capacity with the same granularity [8].

Mobile capacity measurement is a well investigated topic in the literature, but, to the best of our knowledge, no lightweight or passive technique allows mobiles to collect frequent measures of their capacity. To fill this gap, this paper proposes a simple technique which is able to measure the fast variations of the per user capacity and, from those, the expected end-to-end throughput.

In order to do so we adapt packet train dispersion techniques by applying an adaptive filtering mechanism, which we show is effective in removing the impact of outliers due to bursty arrival and jitter, which are very prevalent in mobile environments. We validate the effectiveness of the solution through extensive simulation and "real world" measurement campaigns: our technique can achieve an accurate throughput estimate with as few as 5% of the packets needed by other solutions, while making an error smaller than 20%.

* Corresponding author at: IMDEA Networks Institute, Spain. Tel.: +34633119487.
E-mail addresses: michelinakis.foivos@gmail.com, foivos.michelinakis@imdea.org (F. Michelinakis), nicola.bui@imdea.org (N. Bui), guido@cartodb.com (G. Fioravantti), joerg.widmer@imdea.org (J. Widmer), fkaup@ps.tu-darmstadt.de (F. Kaup), hausheer@ps.tu-darmstadt.de (D. Hausheer).

http://dx.doi.org/10.1016/j.comcom.2016.02.005
0140-3664/© 2016 Published by Elsevier B.V.

Our goal is to provide a simple tool that evaluates passively or with minimum impact the per user capacity variations over time in a mobile environment. This enables filter based prediction techniques and, consequently, prediction based resource allocation optimization. Source code for the tool can be found in the repository of the EU project eCOUSIN.[1]

In the following sections we propose a lightweight measurement technique of the per user cell capacity. Our proposal adapts earlier packet train dispersion techniques and allows to collect reliable measurements on a mobile device despite the complexities introduced by the wireless link and the phone hardware. Also, we have evaluated our technique on both simulations and actual mobile network data collected during a measurement campaign.

The rest of the paper is structured as follows. Related work and some mobile network fundamentals are discussed in Sections 2 and 3, respectively. We present our measurement technique in Section 4, in Section 5 a first evaluation of our technique based on simulations, and in Section 6 we describe how we collected “real world” data to validate it. The results are discussed in Section 7. Finally, Section 8 summarizes our conclusions.

2. Related work

A number of approaches exist to estimate mobile capacity. The most popular of which is Ookla’s mobile application, Speedtest [9], which computes the maximum end-to-end throughput achievable by two long lived TCP connections with the closest measurement server (according to our tests the measurement lasts for either 20 s or after 30 MB have been downloaded, whichever happens first). Then, it derives throughput samples and aggregates them into 20 bins (each one has about 5% of the samples), applies some post processing to remove measurement artifacts and, finally, estimates the average of the bins. Huang et al. [10] proposed to use 3 parallel TCP connections in order to remove the effects of packet losses, TCP receive window limitations and overloaded servers, while ignoring any data collected during the slow-start phase of TCP. The calculated throughput is given by the median of the collected samples, in order to reduce the effect of outliers. Recently, Xu et al. [11] analyzed the use of UDP to compute the end-to-end throughput availability, also accounting for packet interarrival times and the impact of mobile scheduling. All these techniques are active, use long data transfers and thus, incur a high overhead.

Conversely, passive monitoring techniques aim at estimating similar information by analyzing ongoing mobile communications, without triggering any dedicated activity. Gerber et al. [12] achieved quite accurate results just by relying on selected types of applications (i.e., video streaming), which provide more reliable throughput measurements as they are more likely to exploit the full cell capacity. In order to study transport protocols in LTE, [13] developed a passive measurement scheme, which monitors the sending rate over a given time window that ensures the full exploitation of the capacity. PROTEUS [14] combines passive monitoring with linear prediction to estimate the achievable throughput. Other solutions worth mentioning in this category are [15], where the authors try to identify bottleneck links in the core network of an operator by conducting large scale passive measurements of TCP performance parameters and [16], where network “footprints” (generated by counting the number of packets and the number of retransmissions of all the users of a network) were used to identify capacity bottlenecks. However, these solutions cannot be directly applied to mobile phones. We conclude that none of the aforementioned solutions allow for frequent throughput measurements, nor do they provide estimates of the per user cell capacity on the client side (mobile device) to allow for effective capacity prediction and resource allocation.

Lai [17] attempts to actively measure the link capacity (which in [17] is called bandwidth) of a path by taking advantage of the packet pair property of FIFO-queuing networks. Dovrolis [18] further refines the packet pair technique and demonstrates that packet pair dispersion rate has a multimodal distribution, whose modes in turn depend on the capacity and the cross traffic at each of the links composing the sender-receiver path. Also, the authors devise a method to estimate the capacity of the bottleneck link in the path, based on the fact that the average throughput measured by packet trains converges to the asymptotic dispersion rate, from which an estimate of the bottleneck capacity can be computed. As we will discuss later though, it is unsuitable for use over mobile networks. CapProbe [19] proposed a technique based on packet pairs dispersion and delays to devise a reliable capacity estimation technique, aimed at mobile networks. Both techniques are meant to measure the capacity of the bottleneck link of a path. Instead, we are interested in measuring the per user capacity at a given moment.

We have recently proposed a passive technique that is able to provide an estimation of the per user capacity range by monitoring the packet arrival pattern that takes place during the TCP slow start phase [20]. In this current work, we are interested in a more accurate per user capacity measurement that is based on periodic samples of the exchanged traffic, taken during the whole duration of the flow.

3. Mobile networks characteristics

In this section we provide a brief overview of the components and characteristics of mobile networks that have an effect on capacity measurement. In the rest of the paper, we will use terminology and network architecture components of LTE, but the ideas and the algorithm can be applied to any recent mobile network technology like 3G.

The user equipment (UE), which can be any device with mobile communication capabilities, connects to the operator network through any of the multiple base stations (BS) that the operator controls, as shown in Fig. 1. BSs are in turn connected to the core network (CN) of the operator. This set of BS can be collectively called Radio Access Network (RAN). They form the interface between the UE and the operator.

The transmission of data from the BS to the multiple UEs connected to it is regulated by a scheduler, which periodically allocates resources and transmits packets to the associated UEs. This period, called Transmission Time Interval, (TTI) largely differs among mobile telecommunication systems, with more recent

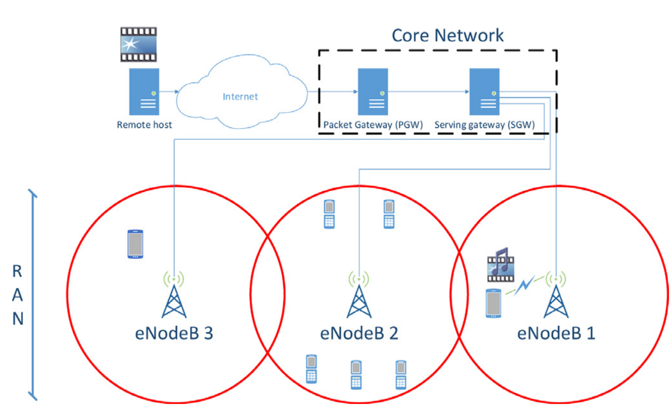
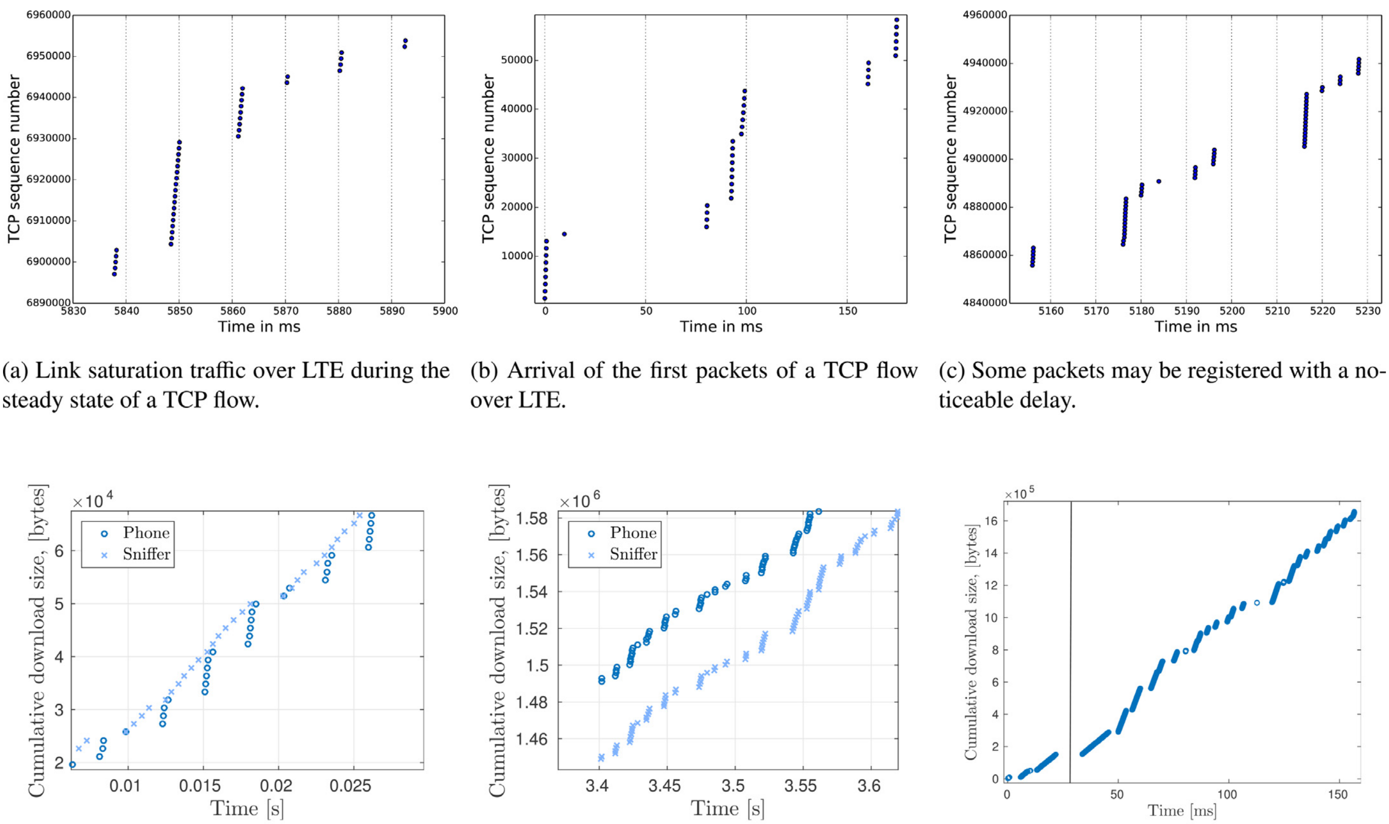


Fig. 1. Some of the LTE network components that a file has to traverse in order to reach a mobile client.

[1] https://ecousin.cms.orange-labs.fr/sites/ecousin/files/lightmeasure.zip .

Please cite this article as: F. Michelinakis et al., Lightweight capacity measurements for mobile networks, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.02.005



(a) Link saturation traffic over LTE during the steady state of a TCP flow.

(b) Arrival of the first packets of a TCP flow over LTE.

(c) Some packets may be registered with a noticeable delay.

(d) Phone with infrequent polling of the NIC

(e) Phone unaffected by polling

(f) Arrival of high speed UDP CBR traffic.

Fig. 2. Time-sequence graphs presenting the arrival of packets to a smartphone, as they were captured by the traffic sniffing tool tcpdump. The time values represent time since the first packet of the download arrived and when the related packets were captured by tcpdump.

technologies having lower values. It can be as short as 1 ms for LTE or at least 10 ms for UMTS. Thus, the UEs receive data in a way such that a burst of data is transmitted to them, during TTIs in which they have been allocated resources and receive nothing during TTIs in which they have not been allocated resources. The scheduling process is usually based on a fairness scheme that takes into account the data requirements and channel quality of all the UEs served by the same BS. A very popular such scheme is the “proportionally fair” scheduling [21]. It tries to weight the past allocation of resources and the current potential throughput of all the competing users. This way it finds a balance between providing adequate resources to all users, regardless of their channel quality, and maximizing the overall throughput of the base station. Thus, in contrast to wired networks, which usually serve traffic based on a FIFO scheme, the incoming traffic at the antenna is distributed to user specific queues and the outgoing is shaped by the scheduler. So, the nature of the competing traffic (UDP/TCP or short/long flows) does not greatly affect the speed of each user. On the other hand, factors that may have an effect include policies (e.g., whether a user is a virtual or host network subscriber [22]) and the specific service that generates the traffic (e.g., VoLTE traffic has the highest priority in an LTE network).

When a packet is transmitted to a UE, it travels from the Internet to the operator’s core network which forwards it to the base station that the UE is connected to. The packet is then stored at the base station in a buffer dedicated to the recipient UE. The packet remains in the dedicated buffer until the scheduler decides to allocate resources to the recipient UE. Upon allocation and depending on the signal quality, it is either grouped alongside other packets present in the buffer to a Transport Block (TB) or, in cases of a bad signal and/or a small amount of allocated resources, a segment of it is encapsulated in a TB. The TB is then sent to the UE.

The mechanisms above are illustrated in Fig. 2a, which shows the arrival of packets to an LTE smartphone, as captured by the sniffing tool tcpdump. In this experiment we are saturating the link and observe its behavior during TCP steady state. Note that the TTI of LTE is fixed to 1 millisecond. It is easily observable that the packets arrive in groups that have about the same duration as the TTI. Between these groups of packets, the smartphone is not allocated resources, thus nothing is received. The size and temporal spacing of the groups depend on the channel quality of the UE and the congestion level at the BS.

3.1. Measurement artifacts

In our traces we frequently observed measurement artifacts that are unrelated to the scheduler and are due to the following reasons.

3.1.1. Small congestion window values during the slow start

The servers that transmit data over TCP send bursts of packets to the client and wait for the related acknowledgments before sending more. This behavior is very prominent during the slow start phase of the transmission when the congestion window has small values. The gap in the transmission at the server side may cause an analogous gap in the transmission at the base station. During this time, the base station is not sending data to the recipient UE, because there are not data in the dedicated buffer. This is visible in Fig. 2b, which illustrates the delivery of the first packets of a TCP flow over LTE. In two occasions, consecutive TBs are received with a delay on the order of tens of ms. We also observe in this example, that the total number of packets delivered in the groups that arrive at about 75 ms is bigger than the number of packets in the first set of groups (the second group has just one packet) at 0 ms. This is caused by the exponential growth of the congestion window. Eventually, the congestion window is large enough that the we observe a continuous stream of incoming packets and this effect diminishes. Since the Round Trip Time (RTT) is larger in 3G networks, the impact of this TCP behavior is slightly more pronounced.

3.1.2. Infrequent polling for incoming packets

IP packets arrive at the UE as part of a TB alongside other IP packets. An ideal method to measure the downlink speed then would require the registering of the exact size and timestamp of each TB. However, this is unfeasible. The related information is only available at the eNodeB, to which a client side tool as the one we propose has no access, or at the Network Interface Card (NIC) of the mobile device. Accessing such NIC information would require specialized drivers, that vendors are very hesitant to release for public usage. The lowest level from which we can extract network information is the kernel, where we register the time and size of all the IP packets. Thus, our view of the network is limited to what is known to the kernel. The exact timing of packet arrivals at the kernel is affected by the capabilities of the phone and the capture software.[2] Usually packets are registered at the kernel with a noticeable delay, compared to their arrival at the NIC. In [23] the delay between the WiFi interface and the kernel is measured, which the authors believe should be comparable with the "Mobile NIC-kernel" delay. They note that the TCP data packets, the packets we are interested in, have the lowest possible delay, compared to ICMP and other TCP packets. The delay, which depends on the NIC ranges from being insignificant to being a few ms. According to [11], both delays are related to the polling frequency of the NIC from the OS.

We have conducted a small scale experiment to assess the effect of polling on several phones, when both the WiFi and the LTE interface are used. When the LTE interface is active, packets are reported in groups similar to the ones visible in Fig. 2a, in all of the phones. The pattern is always similar with some minor variations on the size and spacing of the groups, depending on how powerful the hardware is. For the WiFi experiment we use 802.11g without packet coalescing, to ensure that each MAC frame encapsulates exactly one IP packet and there is no grouped transmission of packets. We also set up a sniffer, which provides more accurate timestamps to monitor the exchanged traffic and provide the groundtruth. In Fig. 2d and e, we show the traces captured by the sniffer and the phones during high speed downloads. We observe that different phones may exhibit a very different behavior. The sniffer always reports a continuous delivery of packets "in the air". Some phones report the packets in the same grouped fashion as above, whereas others report continuous delivery of packets. Based on these observations, we conclude that the pattern of packet arrival on WiFi seems to be greatly dependent on the phone specifications. The arrival pattern in the LTE case is determined by the grouped delivery of packets in the physical layer, but the timestamping accuracy of each packet is related to the phone hardware. More powerful phones are less affected by the polling problem, but even in this case, the delay shows slight variations. Since this delay is very small, it is not significantly affecting our technique, whose adaptive and statistical nature tries to countermeasure it.

3.1.3. Weak or busy phone hardware

It is quite common for packets to be delivered to the phone but not delivered to the higher layers until several milliseconds later, alongside all the other packets that have been received in the meantime. This is usually observed in cases of high capacity and/or high CPU utilization. This behavior is very evident in Fig. 2, which depicts the TCP steady state of a 3G download. According to the server side trace of this download, the server transmitted all the packets that are visible in the figure almost "back-to-back". Also, the phone trace showed a steady rate in the delivery of packets. But at times 5175 and 5215 ms we observe a gap in the delivery of packets and then the delivery of an impossibly large group. Packets were actually delivered during these gaps, but were registered all together when the CPU was able to process them.

3.1.4. Slower speed during the first packets of a flow

We have noticed that when a UE may achieve very high speed, there is a significant difference in the arrival rate of the first few hundred packets of a flow and the arrival rate of the rest of that flow's packets. The difference is present even if we take into account the reduced rate of the slow start phase of TCP, in case the flow is TCP. We have observed this phenomenon in traces gathered in the networks we used to evaluate our tool, as well as other European mobile networks. In order to get more insight, we have done a small experiment in a Spanish LTE network, where we send constant bit-rate UDP traffic and monitor the arrival rate as reported by the mobile. When the server transmits traffic at a rate smaller than 25 Mbps, there is no difference in the arrival rate at different parts of the flow. If the rate of the server is higher than 25 Mbps, the first part of the flow (usually the first 150 to 300 packets) has an arrival rate 25–50% lower compared to other parts of the same flow. For the flow presented in Fig. 2, the arrival rate of the packets located on the left side of the vertical line (first 178 packets) is almost half the rate of the rest of the packets on the right side of the vertical line. If the transmission pauses for a few tens of ms, the same effect is observed upon restart. Even though we did not perform a dedicated experiment for a 3G network, our traces indicate that this phenomenon is even more prominent in 3G. An independent team of researchers [24], who conducted measurements in the same German network we used to collect our traces, observed that the first packets of a flow experience a considerably higher delay compared to the rest, when the rate at the server is higher than 20 Mbps. This effect causes reduced speed during the first part of the flow. While we are unable to investigate this phenomenon further, due to we lack of physical layer or mobile network specific information, we believe that it can be attributed to an operator configuration.

3.2. Packet pairs issue

The previous characteristics of mobile networks and phone hardware make the use of traditional packet pair techniques infeasible. Any two packets that would make a packet pair are in either of the following cases.

Transmitted in the same TB. In this case the packets arrive more or less at the same time to the UE, since all the information included in the TB is transmitted in parallel using multiple carrier frequencies. The lower protocol layers of the UE ensure that they are delivered to the higher layers in the right order, while also assigning them slightly different timestamps. Consequently, sniffing tools like tcpdump perceive them as arriving with a tiny time difference, in the order of a few hundreds of microseconds. A capacity estimation based on these packet pairs would greatly over-estimate the real value of the capacity.

Transmitted in different TBs. In this case, the packet pair consists of the last packet of a TB and the first packet of the following TB. Thus, the capacity value is greatly underestimated, since the measured dispersion is the dispersion between the TBs and each TB is very likely to be able to encapsulate more than one IP packet,

[2] http://www.tcpdump.org/faq.html#q8 [Last access: 2015-03-24].

which is not reflected in the measurement. If there is exactly one packet per TB, then an accurate estimation is possible, but we observed that in the majority of the cases each TB contains multiple packets.

3.3. Packet trains issue

Packet trains are also problematic. They cannot be used in a passive scenario because the server transmits packets on the receipt of ACKs and the application requirements, so the trains will have variable length. The number of packets in each TB may be different, which results in similar problems to the ones described in the "packet pair" scenario. On some occasions all the packets will be transferred in the same TB and on others in multiple TBs.

It is clear that long-established packet dispersion techniques that were developed to detect the bottleneck link capacity in wired networks are not suitable for mobile networks, especially in regards to detecting the per user capacity. In the sequel, we will present the necessary modifications to this approach for it to provide reliable capacity estimations in mobile scenarios.

4. Mobile capacity estimation

In the literature, the term "link capacity" refers to the transmission rate of a link, "path capacity" is the minimum transmission rate among all the links of the path and finally "link available bandwidth" refers to the spare link capacity (capacity not used by other traffic) [18]. Instead, we are interested in estimating the maximum capacity that the scheduler of an eNodeB could allocate to a target user if he requested saturation traffic under a specific bearer. This metric is specific to cellular networks, we call it "per user capacity" and we symbolize it as C_U. For brevity, in the rest of the paper we refer to it as "capacity". To the best of our knowledge, traffic flow templates are not used for generic browsing and multimedia traffic, which is the scope of this work. Thus, we can safely assume that all the measured traffic is using the default bearer, allowing us to ignore this variable. As we will analyze in the sequel, in practice, the measured C_U will often be less than the maximum capacity a user could be allocated. For this reason, the measured value represents the greatest lower bound of the user's capacity. We will show that this value is very close to the actual maximum, thus causing a slight underestimation of the true maximum per user capacity.

The wireless link is the last hop of a downlink path and the C_U of all the connected users is dependent on the cell congestion, the channel quality, the channel's bandwidth and the scheduling algorithm. It is usually the link of a path with the lowest capacity, that also contributes the most to the delay. On the other hand, the average end-to-end TCP throughput R, depends on the capacities and the cross traffic of all the links in the path, as well as possible rate adaptations at the server side, caused by the TCP mechanisms. The end-to-end TCP throughput is primarily determined by the link with the minimum spare link capacity, which in a mobile scenario is usually the RAN. We are interested in measuring C_U, since it is the metric that affects all the connections that the user is going to have in the future and is usually the bottleneck.

Fig. 3 illustrates the packet dispersion due to the transmission over links at different link capacities. This example is based on LTE, but similar effects are observed in various mobile technologies. Initially, (1) the server sends a burst of IP packets (A-H in the example) back to back. The number of packets in the burst varies since it depends on a number of factors like the state of TCP connection, the specifics of the application and the server that generates it. Subsequently, (2) the base station (eNodeB) receives the packets, which have suffered variable delays due to the different link capacities and cross traffic encountered along the path. When the scheduler allocates a TB (marked with horizontal arrows in the plot) to the receiving UE (3), as many packets as possible are encapsulated in it. Therefore, all the packets that are scheduled together arrive within the same TTI at the UE. As a consequence, the inter-packet interval can be greatly reduced (packets A and B) or greatly magnified (packets B and C).

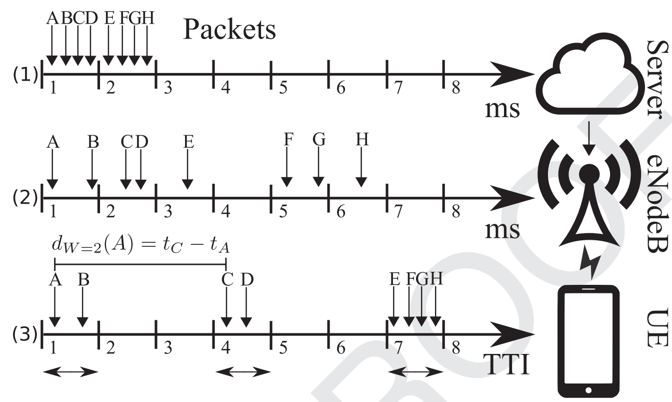


Fig. 3. Dispersion of IP packets over the Internet. First, they are sent back-to-back from the server (1). After experiencing dispersion on the Internet, they arrive on the BS (eNodeB) (2). Finally, they are received in groups by the UE (3). The timelines (1–3) happen sequentially, one after the other, not in parallel. The horizontal arrows represent TBs allocated to the recipient UE.

Considering the set of "back-to-back" transmitted packets crossing the path in Fig. 3, we can distinguish their arrival rate R_A at the antenna from their transmission rate from the antenna to the user, which can have a maximum value of C_U. Both metrics are dynamic and are affected by the same parameters that affect R. Thus, if we sample them for a specific period of time, we may notice the following relationship between them. If $R_A > C_U$, the set of packets arrives at the BS with a delay which is inversely proportional to R_A and shorter than the average time needed for the BS to serve all but the last packet. Since the arrival rate is higher than the departing rate at the base station, the dispersion of the set is caused by the last link. Also, depending on the scheduling strategy, the set may be served within the same transport block or multiple transport blocks by the BS. Conversely, if $R_A < C_U$ the set of packets arrives at the BS separated by a delay which is longer than the average serving time of the BS. We thus have three cases (excluding the problematic cases of Section 3):

(i) Bursty arrival [11,13] (e.g.: set of packets E-F), if $R_A > C_U$ and packets are in the same transport block.
(ii) Wireless link capacity, if $R_A > C_U$ and packets are in different transport blocks (e.g.: set of packets A-D).
(iii) The bottleneck link being in the server-BS path and/or the server transmitting at a very low rate (e.g. TCP slow start), if $R_A < C_U$.

In order to estimate C_U, we have to filter both *i*) and *iii*) cases, as well as take into account the behavior of sets of packets when transmitted over mobile networks as presented in Section 3. In brief, our approach has two components: (a) generating capacity estimation samples which are not significantly affected by the above and (b) the statistical processing of those samples in order obtain a C_U value.

4.1. Capacity estimation samples

The input data for our passive measurement tool are the timestamps and sizes of all the received data packets of a smartphone. We ignore packets related to connections establishment such as TCP and TLS handshakes, since they can not saturate even momentarily the wireless link. This information can be collected on

Please cite this article as: F. Michelinakis et al., Lightweight capacity measurements for mobile networks, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.02.005

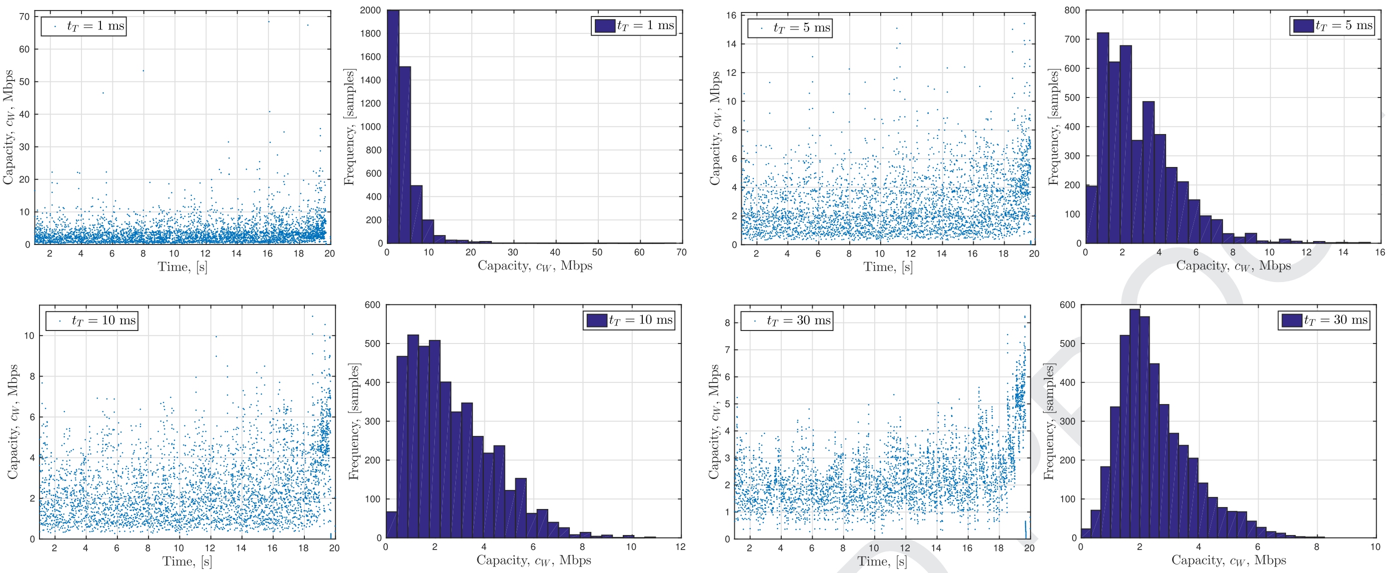


Fig. 4. Scatterplots of c_W (left of each pair) and its statistical distribution (right of each pair) computed for $t_T = \{1, 5, 10, 30\}$ ms from left to right. When the dispersion time is computed on windows larger than the TTI, $t_T > t_S$, the distribution gets more stable.

the OS level by monitoring the stack. In our experiments, we use rooted Android smartphones and tcpdump to capture all the incoming traffic. Ultimately this functionality could be included in the mobile OS as an on-demand lightweight measurement service.

We consider a set of N packets sent from a server and received at the UE so that the ith packet is received at time t_i, with $i = \{1, \ldots, N\}$. A key metric used by our algorithm is the "inter-packet interval", the time difference between the arrival of two consecutive packets $(t_{i+1} - t_i)$. Obviously, in a group containing N packets, there are $N - 1$ intervals. W represents the unit-less number of such intervals that we take into account when we generate the capacity estimation samples. For each packet in the set we define the dispersion time $d_W(i) = t_{i+W} - t_i$, and the per user capacity sample $c_W(i) = (\sum_{j=i}^{i+W-1} L_j)/d_W(i)$, for a given value of W, where L_i is the length of ith packet.

In detail, the $c_W(i)$ value of packet i is derived by adding the sizes of W consecutive packets, starting from i and then dividing by the time duration of W consecutive inter-packet intervals, starting from $[t_{i+1} - t_i]$. Packet $i + W$ contributes only to the denominator. For example, in Fig. 3, $c_{W=2}(A)$ is computed by dividing the sum of sizes of the packets A and B by the dispersion time $d_{W=2}(A) = t_C - t_A$.

The three arrival cases above contribute to the distribution of the capacity samples in different ways. Arrivals of type (i) cause a tiny d_W and, thus, skew the distribution to the right (over-estimation of C_U). At the same time, type (iii) events, which show larger d_W (under-estimation of C_U) skew the distribution towards the left. To better visualize what is discussed next, Fig. 4 shows a set of scatterplots of c_W and histograms of its distribution computed on a single download performed using the Speedtest application [9] over a HSPA connection. The X-axis of the scatterplots represents the arrival time of packet i and the Y-axis its c_W value.

The impact of type i) arrivals can by limited by setting W appropriately. The idea is to include in each measurement packets belonging to different TBs in order to make sure that the highest throughput c_W we can measure is only related to the cell capacity and not to bursty packet arrivals, as it would have happened had we chosen $W = 1$ in the example of Fig. 3. In order to achieve that, it is sufficient to study groups that, starting from any packet i, contain W_i intervals so that the minimum dispersion time $d_W(i)$ is longer than the maximum TTI of the scheduler, abbreviated t_S:

$$W_i = \{\min(W) \mid \min_W(d_W(i)) > t_S\} \tag{1}$$

This guarantees that at least two packets within the W_i window are scheduled in two different transport blocks, since $t_{i+W_i} - t_i = d_{W_i}(i) > t_S$. In other words, we are averaging the burstiness over two transport blocks. An effect of Eq. (1) is that each packet i has a different W_i value, depending on the spacing of packets that were received after it.

It is important to select the minimum value of W for the creation of the $c_{W_i(i)}$ value for packet i that has the property $\min(d_{W_i}(i)) > t_S$. As discussed in Section 3, the "slow start" behavior of TCP introduces noticeable gaps in packet delivery. Thus, samples that include these gaps in their calculation of d_W, generate c_W values that are significantly smaller and not representative of the C_U. A high value of W increases the probability of a sample to include such gaps.

4.2. Statistical processing of the samples

Now that type (i) events are filtered, we ensure that each set spans across at least two TBs. The minimum dispersion time $\min d_{W_i}(i)$ for every packet i of the flow cannot be smaller than the minimum time needed for a set of packets to cross the wireless link, which corresponds to the maximum per user cell capacity. Thus, C_U can be found as the maximum of the distribution of c_W, which is equivalent to the maximum value of c_W.

$$C_U = \max_{i \in [1,\ldots,P]} c_{W_i}(i) \tag{2}$$

P is the total number of data packets of a flow. Note that, with Eq. (1) we are filtering the effect of type (i) arrivals (min) and with Eq. (2) the delays introduced by type (iii) arrivals (max).

Ideally, we would like to sample c_W until its distribution is stable, but C_U is varying because of both user movements and fast fading. Hence we can only obtain an estimate $C_U{}^{(p)}$ of it from a set of p consecutive estimation samples, where $p < P$. Although estimating the distribution from a limited number of samples reduces the accuracy of our measurement, we can at least guarantee that

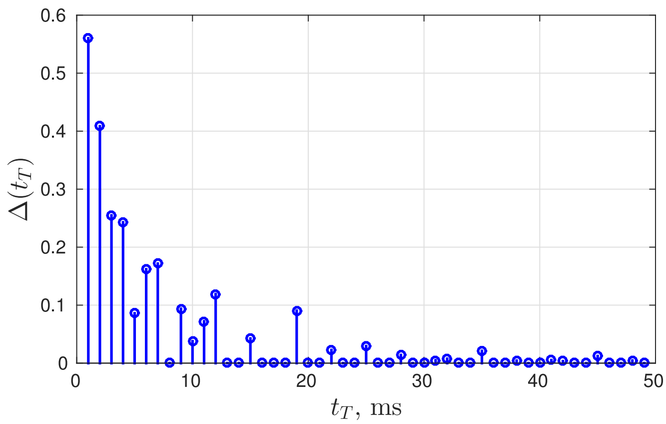


Fig. 5. Ratio $\Delta(t_T)$, varying $t_T \in [2, \ldots, 50]$ ms. The measurements get stable from $t_T > t_S = 10$ ms.

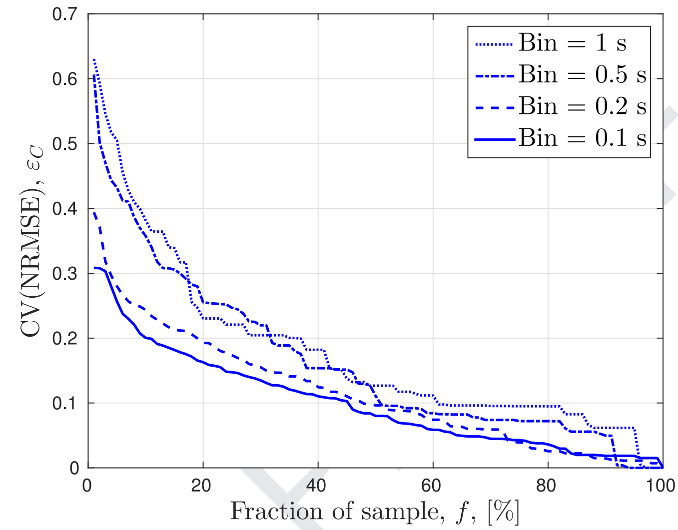


Fig. 6. Coefficient of variation of the normalized root mean square error ε_C of the capacity estimate computed over a fraction $f = k/K$ of continuous samples for varying bin sizes ({0.1 s, 0.2 s, 0.5 s, 1 s}).

we are not overestimating C_U:

$$C_U{}^{(p)} = \max_{i \in [1, \ldots, p]} c_{W_i}(i) \leq \max_{i \in [1, \ldots, P]} c_{W_i}(i) = C_U \tag{3}$$

This follows from the probability of the distribution of a sampled random process to contain the maximum of the theoretical distribution of the process, which is increasing with the number of collected samples:

$$\lim_{p \to \infty} C_U{}^{(p)} = C_U \tag{4}$$

4.3. Capacity measurement

This section describes the feasibility of lightweight active and passive measurements of per user capacity C_U based on dispersion samples of packet sets. It also explores the effect different values of some parameters have on our technique. We compute the dispersion time by using an adaptive window W_i intervals long for every packet i such that:

$$W_i = \{\min(W) \mid t_{i+W} - t_i > t_T\}, \tag{5}$$

where $t_T \in [1, \ldots, 50]$ ms, for all the values of t_T. The estimation sample of the i^{th} packet is composed of all packets following i until the first packet which arrived at least t_T ms later than i. This allows to satisfy Eq. (1) a posteriori if the TTI duration is not known.

We exemplify the dispersion time in Fig. 4 based on data obtained by time-stamping the arrival time of the packets of a 6 MB HSPA download. The figure presents the evolution of the scatter-plots of c_W and the corresponding histograms of the c_W distribution for various characteristic values of t_T.

During the slow start phase of a TCP connection an increasing number of packets are sent back to back from the server, and after a few RTTs the congestion window is large enough to allow the transmission of packet trains long enough to measure capacity as high as 100 Mbps. In fact, C_U should be proportional to the maximum number of packets that can be scheduled in a single transport block and, if Eq. (1) is satisfied and $t_T > t_S$, the impact of outliers due to bursty arrivals is removed. With reference to Fig. 4, it can be seen that the maximum of c_W is approaching a stable value of about 10 Mbps when $t_T \geq 15$ ms. Due to limited space, we do not present the related plots of other downloads. Based on the rest of our dataset, a stable value is reached for values of t_T between 10 and 20 ms.

Moreover, Fig. 5 shows the stability of the maximum of the capacity by plotting the ratio $\Delta(t_T)$, computed between the maximum value obtained with windows of $[t_T]$ and $[t_T - 1]$:

$$\Delta(t_T) = \frac{|C_{W|t_T} - C_{W|t_T - 1}|}{C_{W|t_T - 1}} \tag{6}$$

Ideally, the ratio $\Delta(t_T)$ should stabilize to 0 as soon the scheduling outliers are filtered ($t_T > t_S$) and further increasing t_T should only make the distribution smoother. However, in actual experiments increasing t_T makes it more difficult to obtain a sample of the maximum capacity which is consistent over different transport blocks. In this preliminary example, we can see that $\Delta(t_T)$ becomes stable for $t_T > 20$ ms, which is in line with the HSPA TTI of 2–10 ms.

Next, we divide the time duration of a download into fixed sized bins. We apply the above method taking into account only a percentage $f = k/K$ of consecutive capacity samples in each bin. In this case, K is the total number of samples inside each bin and k is the number of consecutive samples that we consider for every bin. Fig. 6 shows the coefficient of variation of the normalized root mean square error – CV(NRMSE) – of the estimate ε_C, by varying f:

$$\varepsilon_C = \sqrt{\frac{\sum_{\text{bins}} (C^{(k)} - C^{(K)})^2}{N_b \mathrm{E}[C^{(K)}]^2}}, \tag{7}$$

where N_b is the number of bins in a flow. The computations have been repeated for different bin sizes varying in {1, 0.5, 0.2, 0.1} seconds (dotted, dash-dotted, dashed and solid lines, respectively). It can be seen that the error decreases below 20% when more than 20% of the samples are used.

Fig. 6 can also be interpreted as the width of the probability distribution of having an exact measurement using f% of the samples. In particular, it is easy to see that when we use all the samples, the distribution should collapse into a delta function (zero width), while the fewer samples we use, the wider the distribution. The real value can only be larger than the measured one, because of Eq. (3) that shows $\max_{i \in [1, \ldots, k]} c_{W_i}(i) \leq \max_{i \in [1, \ldots, K]} c_{W_i}(i)$. Thus, this distribution has non-zero width for values smaller than the actual measurement only.

To complete this preliminary evaluation of our measurement technique, Fig. 7 shows the variation of the per user capacity $C_U{}^{(K)}(t)$ measured every 500 ms and its estimates $C_U{}^{(k)}(t)$

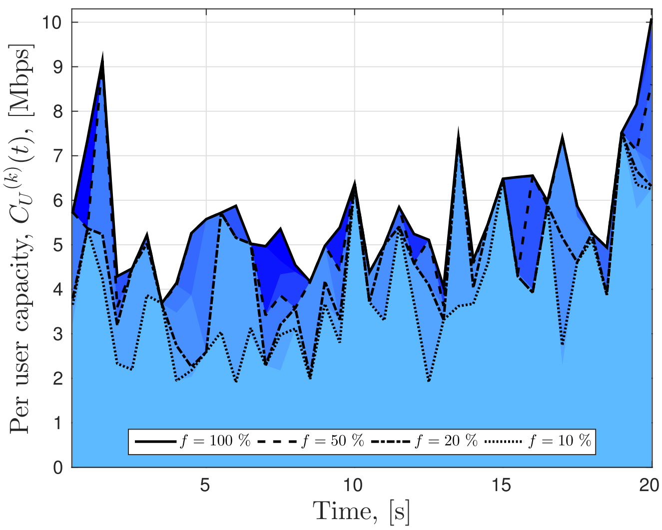


Fig. 7. Time plot of the capacity variation $C_U^{(k)}(t)$ computed every 500 ms and its different estimates computed with $f = \{10, 20, 50, 100\}$%.

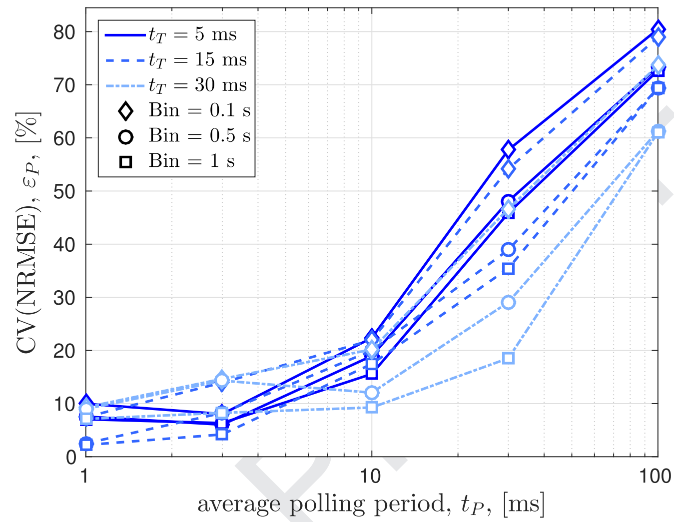


Fig. 8. CV(NRMSE) ε_P of the capacity estimate between ideal arrivals ($t_P = 0$) and arrivals that suffer from polling ($t_P \neq 0$), for varying bin sizes and minimum dispersion times t_T.

Table 1
Simulation parameters.

Parameter	Value
Number of resource blocks (Mhz)	25 (5), 50 (10), 75 (15), 100 (20)
Number of competing UEs in the cell	$[0, 1, 2 \ldots, 10]$
Distance between UE and BS in m	$[0, 50, 100 \ldots, 450]$
Number of interfering BS	$[0, 1, 2 \ldots, 6]$
Type of scenario	"static", "urban walking", "vehicular"

computed with $f = k/K = \{10, 20, 50, 100\}$ % (dotted, dash-dotted, dashed and solid lines, respectively). Although with 10% of samples the estimates are quite different from the actual capacity values, we will be showing next that it is possible to exploit these coarse estimates to obtain a sufficiently accurate capacity estimate.

5. Simulation campaign

We have performed an extensive simulation campaign in order to evaluate our proposed technique in a controlled environment. We use a modified version of ns-3.23 [25] and its LTE module LENA [26]. We focus on LTE due to its increasing popularity. In all simulations the monitored user uses TCP, since it is both the most challenging and the most popular [13] transport layer protocol of mobile phones. The variable parameters of the simulations are presented in Table 1. The fixed parameters are: (1) the simulation lasts for 22 seconds and (2) the BS uses a proportionally fair scheduler. For each set of parameters we run the simulation multiple times with a different seed, generating in total 18,570 flows.

Next we investigate the effect of polling on the accuracy of the measurements. The simulation results do not suffer from polling, thus the packet arrival time reported in the logs is the actual arrival time at the NIC. In order to simulate the polling effect we manipulate the logs so that we check for incoming packets every $t_P \pm 10\%$, where $t_P \in [1, 3, 10, 30, 100]$ ms. We add the 10% deviation in the timing of each polling because based on our traces and the literature, polling does not have a fixed frequency. We also add a tiny inter-packet delay (in the range of 0.1 ms) between the packets that are reported together by the polling function, in a fashion similar to the one we observe in our "real life" traces. Please note that the polling delay (if present) is usually within 10 ms under normal circumstances.

Fig. 8 shows the CV(NRMSE) ε_P between traces that have the original timestamps and processed ones. We calculate the ε_P as we did for the ε_C in Eq. (7).

$$\varepsilon_P = \sqrt{\frac{\sum_{\text{bins}}(C^{(t_P)} - C^{(0)})^2}{N_b \mathrm{E}[C^{(0)}]^2}} \tag{8}$$

It can be seen that the error is at most 20% for most cases (up to 10 ms of delay).

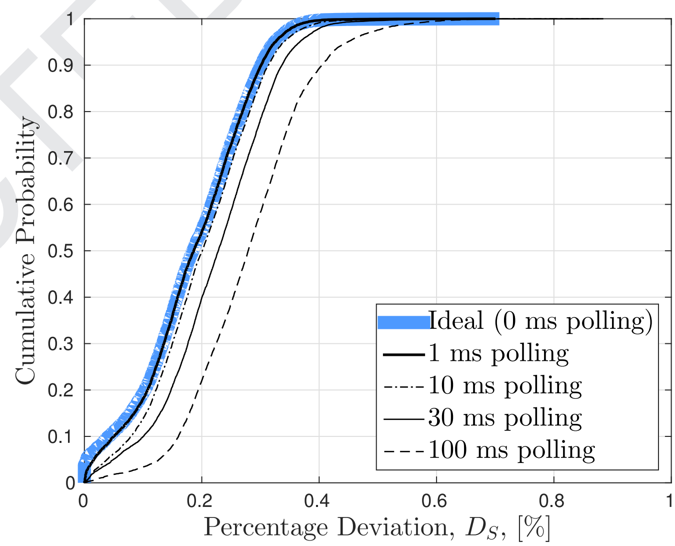


Fig. 9. Deviation of the sampling estimations ($k = 5\%$) for various average polling periods t_P from the ideal case ($k = 100\%$, $t_P = 0$).

Subsequently, we examine how the combination of sampling only 5% of the available estimators and polling affects the accuracy of the results. We divide every flow to 100 ms bins and for every bin we calculate the $C_U{}^{(100\%)}$ and the $C_U{}^{(5\%)}$ for various t_P values. The speed of each flow is the average of the measured capacity of all its bins $\mathrm{E}[C_U{}^{(k)}]$. As a groundtruth, against which we compare the rest of the results, we suppose the case where $t_P = 0$ (ideal polling) and $k = K$. Fig. 9 depicts the Empirical CDF of the percent Deviation D_S computed by the formula:

$$D_S = \frac{|\mathrm{E}[C_U{}^{(5\%)(t_P)}] - \mathrm{E}[C_U{}^{(100\%)(0)}]|}{\mathrm{E}[C_U{}^{(100\%)(0)}]} \tag{9}$$

By comparing the ideal line of $t_P = 0$ with the rest, we conclude that even though polling does have a negative effect in the mea-

Please cite this article as: F. Michelinakis et al., Lightweight capacity measurements for mobile networks, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.02.005

surements, the dominant cause of error is the sampling. Also, we observe that for the most common t_P values (t_P < 10 ms) the deviation for 90% of the cases is less than 30%.

6. Measurement campaign

In order to validate our measurement technique over many different "real life" scenarios and configurations, we organized a measurement campaign that covers two cities in two different countries, Darmstadt (Germany) [27] and Madrid (Spain), for 24 h a day lasting 7 days. During this time, 5 people per city moved around as they normally do, carrying one measuring device each and performing their usual tasks involving mobile networking on the measuring devices. In order to be able to compare results of both passive and active measurements, we also perform automated periodic file downloads.

All the devices were running a simple Android application, which was periodically sampling the available capacity by starting two download types: *short* downloads of 500 KB to study the TCP slow start phases and *long* downloads of 2 MB to measure TCP steady state throughput. The two types were organized in a sequence with a long download, preceded by two small downloads and later succeeded by another two. We use tcpdump on the measurement devices to monitor the arrival time and size of all incoming packets. The download sequence was repeated every 50 min. Additionally, we log other related phone parameters: GPS, cell ID, Channel Quality Indicators (ASU, dBm) and network technology (2G, 3G, LTE).

The phones used in the campaign were the following: 5 Nexus 5, located in Germany, and 4 Sony Xperia Miro and 1 Samsung Galaxy S3, located in Spain. Also, while the Nexus 5 phones are LTE capable, the other phones only support radio technologies up to HSPA.

7. Results and discussion

We verified our measurement technique by analyzing more than 3000 unique TCP flows extracted from the communication of the phones participating in the campaign. As before, we split each flow into 100 ms bins and calculate the ${C_U}^{(100\%)}$ and ${C_U}^{(5\%)}$ metrics, and assume that their average is the speed of each flow. Note that in these measurements we neither have control over the polling, nor we can distinguish it from the scheduling behavior.

Fig. 10 shows a scatterplot where the abscissa and the ordinate of each rectangular point are the sampled and non-sampled versions of C_U, respectively. Further we add in the same plot the related simulation results for $t_P = 3$ ms as diamonds. As expected from Eq. (3) all the data points are above the $y = x$ line. Thus, we verify that our algorithm may only underestimate the capacity. The fact that all the points are so close to the $y = x$ line proves that the values derived by just 5% of the samples are good estimators of ${C_U}^{(100\%)}$. As a consequence, this measurement can be safely used as a lower bound in resource optimization problems. We also plot the linear regression of only the actual measurement results as a dashed line. The regression line would allow us to build an even better estimator with lower error.

The figure is plotted in double logarithmic scale in order to emphasize that the relationship between ${C_U}^{(100\%)}$ and ${C_U}^{(5\%)}$ can be observed over all the measured connection rates and there is an almost constant ratio between the estimate and the actual value. Although outliers are visible, we can obtain quite an accurate estimate of C_U by exploiting as few as 5% of the packets sent during a TCP connection. This allows for quite an effective passive monitoring technique as, even by monitoring small data exchanges, it is possible to obtain frequent and accurate mobile per user capacity measurements necessary for user throughput prediction and re-

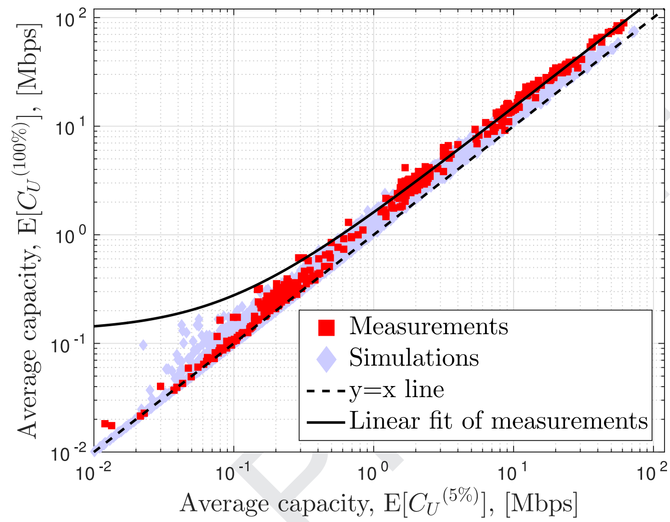


Fig. 10. Scatterplot of the average estimate of per user capacity computed using all available information E[${C_U}^{(K)}$] against the estimate computed 5% of the available information E[${C_U}^{(k)}$], $k = K/20$.

source allocation. The linear regression line seems to deviate from the measurement "cloud" for low values of capacity, because of the double logarithmic scale used in the plot, which highlights the regression offset for low values (500 Kbps and less). Further, we observe that for high values, the regression line has an almost fixed vertical distance from the $y = x$ line (constant percentage error). This represents the error of the estimate and, since it is constant, in the double logarithmic plot, appears as a fixed deviation on the y-axis from the $y = x$ line.

Unfortunately, using very low rate background traffic is impossible. The rates of such traffic are on the order of 4 packets over 100 ms, which do not allow for reliable capacity measurements. Also, a big number of the APPs use the Google Cloud Messaging (GCM) service, which minimizes their notification related traffic. In the case of GCM, if there is an update a few packets are sent just to generate a notification. When the user interacts with the notification, a larger number of packets are downloaded. In this scenario, we can use that download to get an estimation.

In the experiments, we use rooted Android phones and tcpdump to perform the measurements. Given the very low complexity and resources that are required by our approach, the C_U estimation is generated at virtually no cost. Therefore, we believe that it may be included in the OS as a service to applications that may opt-in to use it. For example, the flow-id, the timestamp and the size of a packet could be registered as part of the standard kernel packet processing procedure. Since these values do not contain any sensitive information, there are no privacy concerns and after a short period to time, when this information is irrelevant it can be deleted. Upon application request, the OS could generate a C_U estimation, if there are sufficient data stored. The knowledge of the flow-id can help distinguish the state of a TCP flow (slow-start, steady-state etc.). If it is possible to use small values of t_T, it is possible to generate accurate estimators even during the late part of slow start, when the congestion/receive windows have relatively high values, since then the dispersion time can be smaller than the time required by the antenna to transmit a server burst. In case of a TCP flow that stops very early, it can be difficult to remove both the slow start and the scheduling artifacts. In such cases, the resulting value will be significantly lower than the truth, but this is easy to detect and filter (e.g., requiring a flow to generate at least 75 downlink packets in order to be used).

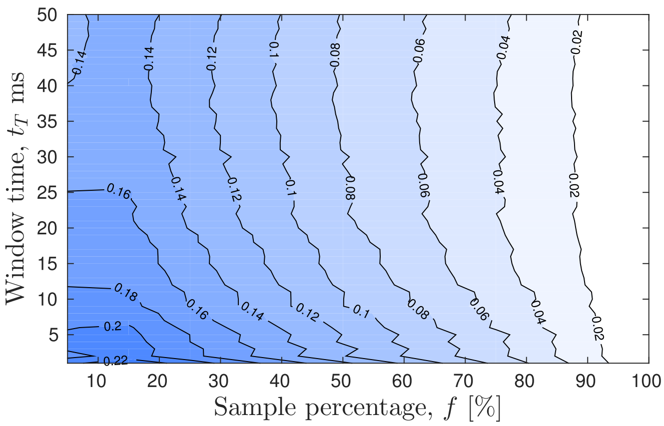


Fig. 11. Contour graph of ε_C varying t_T and f for a bin size of 200 ms.

Table 2
Average C_U and average optimal t_T per technology.

Technology	UMTS	HSPA	HSPA+	LTE
C_U (Mbps)	10.83	1.4	10.74	24.3
Optimal t_T (ms)	19	23	17	16

As a side note, our technique is also able to estimate fast per user capacity variations. However, it obtains a lower accuracy since a larger fraction of samples are needed to estimate the maximum of the c_W distribution. Nonetheless, it is often sufficient to use 20% of the samples collected in a bin to achieve a reasonable estimate of C_U. In fact, with the smallest bin size and as few as 20% of the samples have an error $\varepsilon_C < 0.2$, which means the actual capacity should not be larger than 120% of the estimated value.

In addition, t_T must be taken slightly longer than the TTI to avoid the measurement being impacted by many bursty arrivals. In line with Eq. (1) of Section 4, $\Delta(t_T)$ approaches zero for $t_T >$ 15 ms for most of the recorded flows.

Fig. 11 shows the CV(NRMSE) for various combinations of t_T and f of the measurement campaign flows. The bin size is set to 200 ms to give an example of this technique's results when it collects very frequent measurements. As expected ε_C decreases when t_T and f increase. For values of $t_T \geq 15$ ms and $f \geq 20\%$, the error is small enough for the model to give trustworthy results ($\varepsilon_C \leq 15\%$).

Finally, Table 2 shows some of the overall evaluation of the traces obtained by the measurement campaign with $f = 25\%$ averaged over the bin size and using the optimal t_T ($\min t_T|\Delta(t_T) \to 0$). Optimal t_T and C_U are computed as described in Section 4 and then averaged over all the traces. While some of the flows are transmitted using 2G EDGE data, the results are not included since there are too few such flows for statistical significance.

The measurements are based on the data reported by the Android OS. Note that HSPA and HSPA+ are a family of enhancements to UMTS, that greatly increase its speed. The high average speed of UMTS is related to networks that support the HSDPA enhancement for improved downlink speed, but not all the enhancements that would classify them as HSPA or HSPA+. The very big differences in speed between the HSPA, HSPA+ and LTE technologies can be explained by the following reasons. More recent technologies can achieve higher speeds. Smartphones tend to use the best technology possible for their channel quality. Thus, they use HSPA only when their signal is too bad to use a better technology and in turn the bad signal greatly affects speed.

Our approach is designed for downlink measurements, which account for the vast majority of the smartphone generated traffic [13]. Recent trends, though, show an increase in uplink related user activity and therefore we will briefly discuss the uplink case. Our algorithm cannot be directly applied to the uplink due to uplink communication characteristics. For instance, if we attempt to perform a measurement on the phone side we can gather very limited information. Without accessing the transceiver firmware, we can only observe how fast packets appear in the kernel, instead of how fast the NIC successfully transmits them at the medium, which is the metric we are interested in. It is possible that packets may remain in the buffer of the NIC for a relatively long time after they appear in the kernel, leading to wrong estimations. On the other hand, applying our algorithm to measurements collected on the server side will fail to measure the cell capacity, since many intermediate hops may be between the eNodeB and the server. An alternative approach would be to infer clues of the speed indirectly at the phone side. If a UDP socket is blocking, it can be an indication that the rate at which an application is generating packets (which we can detect) is higher than the link capacity, thus deriving an upper limit of the speed. In the case of TCP traffic, the ACKs can be analyzed to infer whether the rate that the application is generating traffic is above or below the link capacity. Further analyzing the uplink scenario is beyond the scope of the present paper and we leave it for future work.

8. Conclusions

We presented a lightweight measurement technique that leverages adaptive filtering over the packet dispersion time. This allows to estimate the per user capacity in mobile cellular networks. Accurate estimates can be achieved exploiting as few as 5% of the information obtained from TCP data flows. Given that this solution can support dense throughput sampling, it is ideal for capacity prediction and optimized resource allocation. In fact, if the future capacity availability is known, it is possible to predict when it is best to communicate by doing so when it is cheaper (i.e., more capacity available). In addition, our solution is able to estimate the fast capacity variations from a mobile terminal by monitoring the traffic generated under normal daily usage.

We validated our technique over a week-long measurement and an extensive simulation campaign. We achieved good estimation accuracy even when using only short lived TCP connections. Since our technique is based on simple post-processing operations on the packet timestamps, it is possible to easily integrate it in background processes or OS routines.

We are planning to extend our measurement application with filter based prediction capabilities in order to provide mobile phones with a complete capacity forecasting tool, which, in turn, will allow for advanced resource allocation mechanisms. Finally, we are planning additional measurement campaigns in order to further extend these encouraging results on passive and lightweight measurement tools.

Acknowledgments

This article is partially supported by the Madrid Regional Government through the TIGRE5-CM program (S2013/ICE-2919), the European Union H2020-ICT grant 653449 and SmartenIT (EU-FP7-317846), the Ramon y Cajal grant RYC-2012-10788 and grant TEC2014-55713-R from the Spanish Ministry of Economy and Competitiveness, and the German DFG (CRC 1053, MAKI).

References

[1] S. Wang, Y. Xin, S. Chen, W. Zhang, C. Wang, Enhancing spectral efficiency for lte-advanced and beyond cellular networks [guest editorial], IEEE Wirel. Commun. 21 (2) (2014) 8–9, doi:10.1109/MWC.2014.6812285.

[2] Z. Lu, G. de Veciana, Optimizing stored video delivery for mobile networks: the value of knowing the future, in: IEEE INFOCOM 2013, 2013, pp. 2706–2714.

[3] H. Abou-zeid, H. Hassanein, S. Valentin, Energy-efficient adaptive video transmission: Exploiting rate predictions in wireless networks, IEEE Trans. Veh. Technol. 63 (5) (2014) 2013–2026, doi:10.1109/TVT.2014.2314646.
[4] N. Bui, J. Widmer, Mobile network resource optimization under imperfect prediction, in: Proceedings of IEEE WoWMoM, 2015.
[5] N. Bui, F. Michelinakis, J. Widmer, A model for throughput prediction for mobile users, in: European Wireless, 2014. Barcelona, Spain.
[6] Y. Qiao, J. Skicewicz, P. Dinda, An empirical study of the multiscale predictability of network traffic, in: Proceedings IEEE HDCP, 2004.
[7] N. Sadek, A. Khotanzad, Multi-scale high-speed network traffic prediction using k-factor Gegenbauer ARMA model, in: Proceedings of IEEE ICC, 2004.
[8] N. Bui, I. Malanchini, J. Widmer, Anticipatory admission control and resource allocation for media streaming in mobile networks, in: Proceedings of ACM MSWIM, 2015.Cancum, Mexico
[9] Ookla, Ookla Speedtest Mobile Apps, http://www.speedtest.net/mobile/(last accessed June 2014). Q3
[10] J. Huang, F. Qian, A. Gerber, Z.M. Mao, S. Sen, O. Spatscheck, A close examination of performance and power characteristics of 4G LTE networks, in: ACM MobiSys, 2012, pp. 225–238. Low Wood Bay, Lake District, United Kingdom.
[11] Y. Xu, Z. Wang, W.K. Leong, B. Leong, An end-to-end measurement study of modern cellular data networks, in: Passive and Active Measurement, Springer, 2014, pp. 34–45.
[12] A. Gerber, J. Pang, O. Spatscheck, S. Venkataraman, Speed testing without speed tests: estimating achievable download speed from passive measurements, in: ACM IMC, 2010, pp. 424–430. Melbourne, Australia.
[13] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z.M. Mao, S. Sen, O. Spatscheck, An in-depth study of LTE: effect of network protocol and application behavior on performance, in: ACM SIGCOMM, 2013, pp. 363–374.Hong Kong, China.
[14] Q. Xu, S. Mehrotra, Z. Mao, J. Li, PROTEUS: network performance forecast for real-time, interactive mobile applications, in: ACM MobiSys, 2013, pp. 347–360. Taipei, Taiwan.
[15] F. Ricciato, F. Vacirca, M. Karner, Bottleneck detection in UMTS via TCP passive monitoring: a real case, in: ACM CoNEXT, 2005, pp. 211–219. Toulouse, France.
[16] P. Svoboda, F. Ricciato, Analysis and detection of bottlenecks via TCP footprints in live 3G networks, in: IFIP WiOPT, 2008, pp. 37–42. Hammammet, Tunisia.
[17] K. Lai, M. Baker, Measuring link bandwidths using a deterministic model of packet delay, in: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, ACM SIGCOMM '00, New York, NY, USA, pp. 283–294. doi:10.1145/347059.347557
[18] C. Dovrolis, P. Ramanathan, D. Moore, Packet-dispersion techniques and a capacity-estimation methodology, IEEE/ACM Trans. Netw. 12 (6) (2004) 963–977.
[19] R. Kapoor, L.-J. Chen, L. Lao, M. Gerla, M. Sanadidi, CapProbe: a simple and accurate capacity estimation technique, ACM SIGCOMM Comput. Commun. Rev. 34 (4) (2004) 67–78.
[20] F. Michelinakis, G. Kreitz, R. Petrocco, B. Zhang, J. Widmer, Passive mobile bandwidth classification using short lived tcp connections, in: WMNC, 2015.
[21] R. Kwan, C. Leung, J. Zhang, Proportional fair multiuser scheduling in LTE, Signal Process. Lett. IEEE 16 (6) (2009) 461–464.
[22] F. Zarinni, A. Chakraborty, V. Sekar, S.R. Das, P. Gill, A first look at performance in mobile virtual network operators, in: Proceedings of the 2014 Conference on Internet Measurement Conference, ACM, 2014, pp. 165–172.
[23] W. Li, R.K. Mok, D. Wu, R.K. Chang, On the accuracy of smartphone-based mobile network measurement, in: IEEE INFOCOM, 2015. Hong Kong.
[24] N. Becker, A. Rizk, M. Fidler, A measurement study on the application-level performance of LTE, in: IFIP Networking Conference, 2014, pp. 1–9.
[25] The network simulator - ns-3, (http://www.nsnam.org/)Last accessed September 2015. Q4
[26] LENA - ns-3 LTE module, (http://lena.cttc.es/manual/, Last accessed September 2015.
[27] F. Kaup, F. Michelinakis, N. Bui, J. Widmer, K. Wac, D. Hausheer, Behind the NAT – a measurement based evaluation of cellular service quality, in: CNSM, 2015.