Evaluation of SDN-based bandwidth estimation in Mobile Broad Band networks

Giuseppe Aceto[⋆,†], Fabio Palumbo[⋆], Valerio Persico[⋆,†], Haiming Chen[‡], and Antonio Pescapé[⋆,†]
[⋆]University of Napoli Federico II (Italy) and [†]NM2 srl (Italy) and [‡]Ningbo University (China)
{giuseppe.aceto, fabio.palumbo, valerio.persico, pescape}@unina.it, chenhaiming@nbu.edu.cn

***Abstract*—Mobile Broad Band (MBB) networks and Software-Defined Networking (SDN) are expected to strongly characterize the future evolution of global communications envisioned by the Fifth Generation mobile networks (5G). Although SDN has seen adoption and wide experimentation in data-center networks, its benefits and challenges in MBB has not received comparable coverage. In this work we experiment with a state-of-art SDN-based approach for passive monitoring available bandwidth and throughput with an OpenFlow switch in the mobile node. We evaluate the approach on a real-world commercial 4G network (leveraging the MONROE platform), considering two deployments (with an SDN controller local to the mobile node, and a remote one, whose control messages traverse the radio access network) and compare the results of the experiments against analogous deployments in a fully-wired testbed. For both the local and remote deployments, different polling periods, in different traffic conditions, are considered. Results show that, while further research is needed to investigate the variability of the relative error (standard deviation ranges between 1.21 and 8.65% in the worst case), its mean is very low, confirming the feasibility of the proposed estimation approach.**

I. Introduction

In the evolution towards the Fifth Generation of mobile communications (5G), aimed at enabling a fully mobile and connected society, several infrastructural innovations have been considered [1]. Among these, one of the most important is the deployment of Mobile Broad Band (MBB) networks, that have an already established coverage in the form of 4G (with less strict requirements on bandwidth and latency compared to 5G) and are expected to witness public availability as 5G by 2020. At the same time, in order to manage the increased complexity of networks and foster their evolvability towards future applications and scenarios, the novel network implementation and management paradigm named Software Defined Networking (SDN) has seen increasing adoption.

The earlier real-world SDN implementations have been motivated by data-center network fast-paced evolution [2], but mobile terminals arguably will soon become another application scenario benefiting from SDN in the evolution towards 5G. MBB access networks already show abundant cases of access sharing among multiple devices, e.g. by means of mobile hotspots or mobile wireless router (so-called *Mi-Fi*); similarly, wireless networks are used as a backhaul for smart cities [3, 4], and modern vehicles are often equipped with network applications for different goals, such as entertainment, traveling assistance, comfort, or maintenance. All these examples can be modeled as mobile nodes acting as a gateway for a number of network applications and network devices, sharing one or more Radio Access Networks (RANs) towards the Internet.

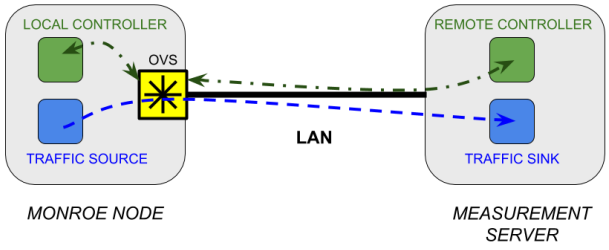
In this scenario SDN offers several advantages related to its promise of flexibility and standardization (and thus, "future-readiness"), including new monitoring possibilities. On the other hand, due to the novelty of both the technologies and the application scenarios (also characterized by resource-limited devices), SDN in MBB is subject to new challenges and issues, demanding for experimental evaluation and analysis in realistic testbeds or in-the-wild deployments. This need can be answered by experimental testbeds such as that delivered by the MONROE project[1] in which we have operated the SOMETIME experiment[2]. The MONROE testbed [5] has been designed purposely to provide an experimental platform on real MBB access networks, offering the suitable infrastructure to implement and evaluate our measurement efforts. More specifically, the SOMETIME experiment has focused on Available Bandwidth estimation in an SDN environment, over commercial 4G connections.

In this paper, we experimentally evaluate a state-of-art SDN-based approach recently proposed in [6] for monitoring bandwidth (in terms of both available bandwidth and throughput) in a real-world MBB scenario (leveraging the MONROE platform). In a nutshell, the approach under investigation—running onto an SDN controller—takes advantage of the network abstractions provided by the SDN southbound API to query the switches in the mobile network and gather the available counters in order to perform passive measurements.

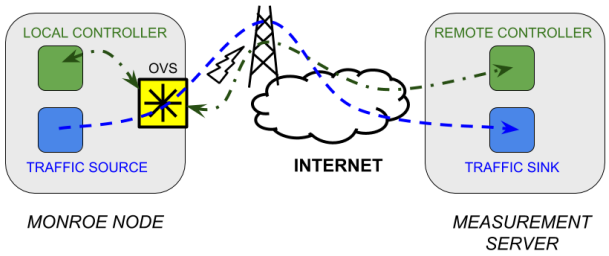
The contribution of this work consists in the first—to the best of our knowledge—experimental evaluation of an SDN-based bandwidth estimation method in a commercial MBB scenario, with setups comprising both a local and a remote SDN controller, compared with an analogous fully wired setup (Figure 1). Hence, we provide an accuracy assessment of the considered monitoring approach when varying measurement parameters (polling period), deployment aspects (position of the SDN controller), as well as the characteristics of the monitored traffic (its average throughput). Results show that accuracy is very high on average, with mean error close to zero in all the investigated scenarios. However, experimental

[1]The MONROE project (https://www.monroe-project.eu) is an European Union's Horizon 2020 funded research project.

[2]The SOMETIME project (SOftware defined network-based available Bandwidth MEasuremenT In MONROE) is part of the 1^{st} MONROE Open Call for Experiments.



(a) Wired LAN deployment.



(b) RAN/Internet-crossing deployment.

Fig. 1. Measurement setups. Dashed blue lines show the application traffic, while dotted-dashed green lines report OpenFlow messages exchanged between the controller and the switch, for both deployments the *local* and *remote* controller setups are shown.

campaigns show that some of the investigated factors (e.g., polling period) impact the accuracy of the method.

The paper is organized as follows. In Section II we provide background for SDN, the MONROE platform used for the experiments, and available bandwidth and throughput definitions and measurement approaches. In Section III we briefly summarize the working principle of the investigated method. We then show the experimental analysis in Section IV and discuss the main outcomes in Section V. Finally, we draw the main conclusions and discuss future work in Section VI.

II. Background

In this section we provide the background to understand the basic working principle of the investigated approach as well as the context and experimental environments the measurements have been conducted into, while citing alternative approaches to position our contribution against the existing literature.

Software-Defined Networking: Software-Defined Networking (SDN) is a recent common name for earlier approaches in network devices management, rooted in works in the fields of active networks and network virtualization. SDN is based on the whole separation of control and data planes. Indeed, network devices are simple forwarding elements—*SDN switches* or simply *switches*—whose forwarding rules are based on a set of fields in the packet headers, while the control plane is relocated onto an external (logically centralized) entity—*SDN controller*. In the current form, SDN attracted significant research and deployment after the introduction of OpenFlow [7] standard protocol for controller-switch communication—*southbound interface* in OpenFlow terms. For more details and a broad discussion on SDN we refer to [8]. Network monitoring with SDN has been considered in several works, the most similar to ours is considering OpenFlow monitoring messages to calculate the flow throughput averaged on the whole flow duration [9], while others indirectly exploit control communications to infer network metrics [10]. Besides being focused on different metrics and adopting different approaches, to the best of our knowledge no previous work has investigated bandwidth measurement with OpenFlow on MBB networks (we refer to [6] for further details on the related works).

MONROE testbed: MONROE is an EU project whose objective is to design and operate an European transnational open platform for independent, large-scale monitoring and assessment of performance of MBB networks in heterogeneous environments. The platform is composed of the following main components: (i) distributed standardized hardware appliances (*MONROE nodes*, also referred to as *mobile nodes* in the following) running the experiment software; (ii) the software—core components and user-defined experiments—running on MONROE nodes in virtualized environments; (iii) the management system, allowing users to access, schedule experiments, and import data; (iv) a database holding experimental results. For further details on MONROE we refer to [5] and project deliverables.

Throughput and Available Bandwidth: Network throughput refers to the amount of information transferred from a source to a destination over a time interval. If maximum application-layer throughput is considered, it is upper-bounded by the network-layer throughput, and also depends on the transport protocol (e.g., TCP or UDP) adopted. A related network metric, Available Bandwidth (ABw) *of a hop* is the average of unused capacity during a considered time interval (equal to hop capacity subtracting throughput). The ABw *of a path* is defined as the minimum value of ABw of the links composing the path, and can be considered as the maximum network-layer throughput that can be imposed on that path without affecting the other flows sharing part of it.

Throughput and ABw can be estimated both through *active* and *passive* measurement approaches. While we have analyzed ABw estimation with active measurement methods in MBB scenario in recent works [11], in this work we focus on passive approaches, along the lines described in [6], but applying it to real MBB scenario, using only standard OpenFlow messages.

III. Measuring Throughput and Available Bandwidth with SDN

The measurement method under investigation leverages the API exposed by SDN switches to gather information about traffic flows of interest. In more details, the measurement application running on the SDN controller queries at different points in time the volume counters—that an SDN switch stores for each flow in its flow table—for a flow of interest. Querying the information related to traffic flows crossing specific interfaces, the controller can obtain a view related to the overall traffic crossing a link. Evaluating the value of volume counters at two different points in time, the measurement application is able to estimate the network throughput B_i on the link as

$$B_i = \frac{V_i - V_{i-1}}{T_i - T_{i-1}} = \frac{\Delta V_i}{\Delta T_i} \quad (1)$$

where V_i and T_i are the data volume counter and the timestamp of the i-th query, respectively. When the capacity of the link is known for the whole path, by measuring the throughput for each link the measurement application can evaluate the ABw on the link as its spare capacity, and on the path as the minimum among such values.

Considering the OpenFlow implementation of this method, queries consist of `FlowStats Request` messages, while volume counters are carried back to the controller within `FlowStats Reply` messages. For the detailed message exchange during the measurement process we point to [6]. Notably as OpenFlow does not store in the message the *actual* timestamp associated to volume counters (according to the most recent standard version, 1.5), the ability of the controller of associating reliable timestamps to these counters is crucial: in fact, errors in evaluating ΔT_i may significantly impact the accuracy of the throughput estimate (see [6] for a detailed analysis and possible solutions extending the OpenFlow protocol). In this work, among the aspects we take in consideration, we investigate the impact of two crucial parameters: (i) the polling period and (ii) the position of the controller with respect to the switch. According to the considerations above, *polling period* (i.e. the time distance between two consecutive `FlowStats Request` messages) is an important parameter of the measurement process: lower polling periods allow to obtain finer measurement granularity, but on the other hand, smaller values for ΔT_i would amplify the impact on measurement accuracy of errors and uncertainties of the timestamping process. The need for placing the controller at different points in the network may be dictated by different architectural and deployment constraints and therefore we believe the impact of this deployment choice is worth to be investigated.

IV. Experimental analysis

Measurement scenario: For the experimental analysis we considered two different scenarios, namely a *wired-LAN deployment* and an *RAN/Internet-crossing deployment* (see Figure 1). Both scenarios share the configuration of the end hosts (i.e. of the mobile node and of the measurement server) but differ by the way they are connected. In the former case (see Figure 1a), the mobile node is deployed in our laboratory at the University of Napoli, directly connected to the measurement server through a 100 Mbps Ethernet LAN. In the latter case (see Figure 1b), while the measurement server is still in Napoli, the mobile node is deployed in Karlstad (Sweden) and connected to the public Internet through a 4G access network (operated by Telia). For both scenarios: (i) a software OpenFlow switch (Open vSwitch, OVS) was deployed onto the mobile node, configuring it as a gateway for all the traffic exiting from and entering the node; (ii) a synthetic traffic generator (DITG [12]) was deployed onto the mobile node, in charge of generating constant-bitrate traffic at different rates towards the measurement server; (iii) we have alternatively considered experiments with the SDN controller (Ryu) placed either on the mobile node (*local controller*) or on the measurement server (*remote controller*). According to the MONROE project design, all the software modules running on the mobile node were run inside a lightweight virtualization environment (*containerization* using Docker[3]). Hardware and software characteristics of the mobile node and the measurement server are reported in Table III.

Experiment Design: In our experimental campaigns, we have considered as experimental variables (i) the placement of the controller (local or remote); (ii) the rate of the traffic flows generated by D-ITG (0.1 Mbit/s, 0.5 Mbit/s, 1 Mbit/s, 5 Mbit/s, 10 Mbit/s and 50 Mbit/s); (iii) and the polling period at which the controller queries the information from the switch (0.5 s, 1 s, 2 s, 5 s, 10 s). It is worth noting that results shown in Section IV are all related to experiments where actual requests are made by controllers every 0.5 s. Indeed, after a two-hour preliminary campaign where we set the controllers to query the switch at different periodicities, we chose to reconstruct results related to larger polling periods sampling the sequences obtained with the 0.5 s polling period with different pace (i.e. $1/2$, $1/4$, $1/10$, $1/20$). This design choice allows us to fairly compare results obtained leveraging different polling periods with no bias generated e.g., by possibly encountered network transient variability. For each of the experimental scenarios taken into account, the obtained statistics refer to 100 samples.

In our experiments, while D-ITG was generating traffic the throughput was monitored by the controller (OpenFlow `FlowStats Request`/`FlowStats Reply` message exchange). The traffic exchanged between the switch and the controller is captured on the switch, thus allowing to compare the inter-departure times of the replies sent by the switch (which constitute our ground truth), to the ΔT seen by the controller, where the timestamps associated to the replies (`FlowStats Reply` messages) are considered. Therefore, for accuracy evaluation purposes, we define the ΔT relative error as the difference between the interval estimated by the controller minus the interval seen on the switch, normalized to the ground truth. In this work, we always report this value in percentage. The error we report has a direct relationship with the error in estimating network throughput, since this is calculated as the difference in the counter values returned by the switch divided by the time interval separating the two replies. As the traffic counters on the switch are collected with no sampling nor precision degradation, there is no measurement error on the volumes, therefore the error on throughput and ABw is linearly (inversely) dependent on ΔT estimation error only (see equation 1).

Experimental results: A common result for all considered scenarios is that the relative error on ΔT is concentrated around zero, and for the vast majority of cases, its standard deviation is smaller than 1%. In the following we detail the results of the experimental campaigns, first for the wired-LAN deployment and then for the RAN/Internet-crossing one, discussing edge cases.

1) Wired LAN deployment: Details of results for this scenario are summarized in Tables I and II, where mean and standard deviation of the relative error are shown for local and remote controller, respectively, varying the values for polling period and generated bitrate. Indeed, in both local and remote

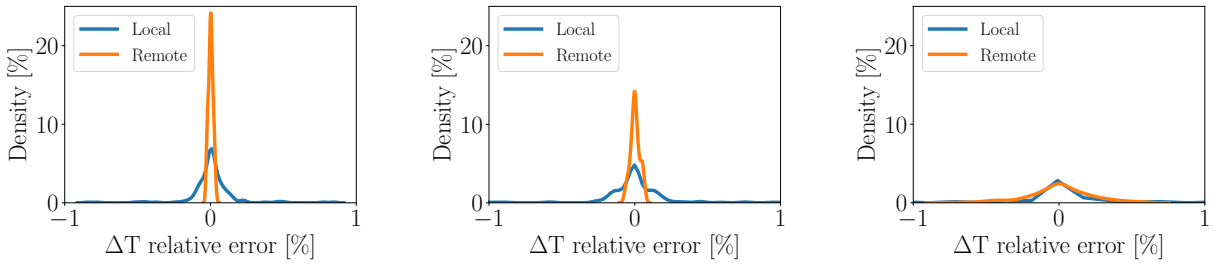
[3]http://www.docker.com.

TABLE I
MEAN AND STANDARD DEVIATION (%) FOR ΔT *relative error* IN WIRED-LAN DEPLOYMENT USING LOCAL CONTROLLER.

		Requested bitrate (Mbps)					
		0.1	**0.5**	**1**	**5**	**10**	**50**
Period (s)	**0.5**	0.0078 ± 0.4590	0.0076 ± 0.3586	0.0064 ± 0.5999	0.0035 ± 0.6755	0.0056 ± 1.6695	0.0039 ± 1.9629
	1	0.0079 ± 0.2505	0.0070 ± 0.2055	0.0067 ± 0.3315	0.0050 ± 0.4108	0.0070 ± 1.2465	0.0045 ± 1.3440
	2	0.0072 ± 0.1156	0.0067 ± 0.1121	0.0068 ± 0.1814	0.0054 ± 0.2313	0.0078 ± 0.7372	0.0070 ± 0.5825
	5	0.0060 ± 0.0500	0.0065 ± 0.0535	0.0064 ± 0.0755	0.0064 ± 0.1021	0.0076 ± 0.3062	0.0111 ± 0.3243
	10	0.0062 ± 0.0272	0.0062 ± 0.0303	0.0062 ± 0.0386	0.0066 ± 0.0522	0.0068 ± 0.1557	0.0116 ± 0.1712

TABLE II
MEAN AND STANDARD DEVIATION (%) FOR ΔT *relative error* IN WIRED-LAN DEPLOYMENT USING REMOTE CONTROLLER.

		Requested bitrate (Mbps)					
		0.1	**0.5**	**1**	**5**	**10**	**50**
Period (s)	**0.5**	0.0059 ± 0.1462	0.0106 ± 0.0992	0.0100 ± 0.1279	0.0066 ± 0.1749	0.0110 ± 0.2206	0.0071 ± 0.2157
	1	0.0069 ± 0.0841	0.0086 ± 0.0676	0.0091 ± 0.0824	0.0080 ± 0.0995	0.0104 ± 0.1196	0.0073 ± 0.1187
	2	0.0080 ± 0.0530	0.0077 ± 0.0379	0.0081 ± 0.0567	0.0081 ± 0.0540	0.0101 ± 0.0678	0.0093 ± 0.0669
	5	0.0066 ± 0.0251	0.0071 ± 0.0184	0.0073 ± 0.0248	0.0066 ± 0.0267	0.0086 ± 0.0281	0.0061 ± 0.0361
	10	0.0052 ± 0.0114	0.0049 ± 0.0106	0.0053 ± 0.0123	0.0043 ± 0.0110	0.0067 ± 0.0158	0.0035 ± 0.0170



(a) Polling period: 10 s. (b) Polling period: 5 s. (c) Polling period: 0.5 s.

Fig. 2. Distribution for the relative error in the wired-LAN deployment for different polling periods when D-ITG traffic is generated at 50 Mbit/s. Probability density function (PDF) is compared for local and remote controllers. With higher polling periods the error distributions are more concentrated around the mean.

controller cases the average relative error is always lower than 0.015%. However, considering the distribution of the error, by looking at Figure 2 it can be noted that it is symmetrical around the mean, and decreasing the polling period causes standard deviation to significantly raise. Similar behavior is observed for increasing generated bitrate, especially in the case of local controller. A potential explanation for this phenomenon is the sharing of computing resource among the controller, the switch, and the application generating traffic, all deployed in the same Docker container. We plan to further investigate this phenomenon in our ongoing work.

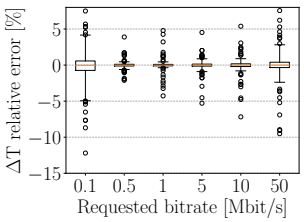
2) RAN/Internet-crossing deployment: In Figure 3, the relative error of ΔT is shown for different polling periods (different rows) both when considering local and remote controller (left and right column, respectively). In more details, the figure reports as boxplots the distribution for different experiments, showing 5^{th} percentile, 25^{th} percentile, median, 75^{th} percentile, and 95^{th} percentile. Values below the 5^{th} percentile and above the 95^{th} percentile are plotted as outliers (black circles).

Considering the *local* controller, Figures 3a, 3c, 3e, 3g, 3i show that the variability (see outliers and whiskers length) is higher when generated traffic has higher throughput. An exception to this behavior, the lowest considered bitrate (0.1Mbps) shows higher variability. Our hypothesis for this unexpected result is that the virtualization environment hosting at the same time the SDN controller, the SDN switch, and the application generating the traffic affects the measurements (e.g. due to scheduling policy and queueing of node-local communications). Further investigation is planned to verify such hypothesis.

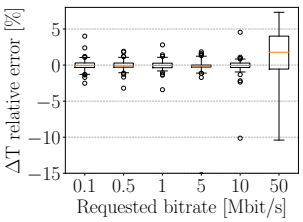
Instead, when the controller is deployed on the remote server (Figures 3b, 3d, 3f, 3h, 3j) the variability of the relative error does not significantly change when increasing requested bitrate, and also outliers are more limited. The exceptional cases are related to 50Mbps requested traffic: inter-quartile ranges increases more than linearly with respect to the case of 10Mbps, and both 5^{th} and 95^{th} percentiles are more distant (crossing $+5\%$ and -10% relative error levels when considering a 0.5 s polling period). We have found the reason for this behavior, which is shown for all the different polling periods, considering that monitoring replies from switch to controller share the same link used by background traffic: at the highest rate the latter saturates the bandwidth (as confirmed by logs from D-ITG, which show an achieved throughput less than half the requested rate), making the effect on the measured delays clearly visible. Therefore, the position of the

TABLE III
HARDWARE AND SOFTWARE CHARACTERISTICS OF THE MOBILE NODE AND THE MEASUREMENT SERVER.

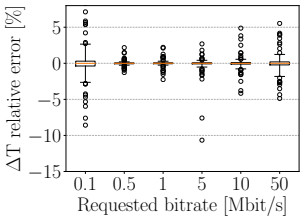
HW/SW spec.	Measurement server	Mobile node
CPU	i7-4710MQ @ 2.50GHz x8	AMD G-T40E @ 1 GHz x2
RAM	12 GB DDR3 @ 1600 MHz	4 GB DDR3 @ 1066 MHz
NIC	Gigabit Ethernet	3xGigabit Ethernet ports
4G	—	3xZTE MF910 MiFi
OS	Ubuntu 16.04 64 bit	Debian-Jessie
Kernel	4.4.0-119-generic	4.9.0-6-amd64



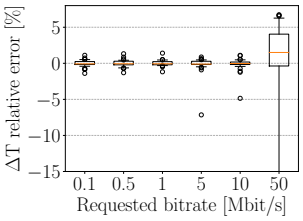
(a) Polling period: 0.5 s.



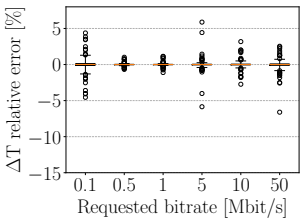
(b) Polling period: 0.5 s.



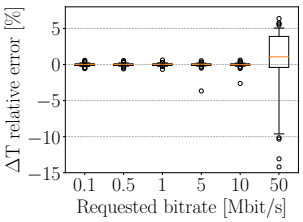
(c) Polling period: 1 s.



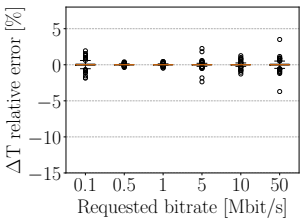
(d) Polling period: 1 s.



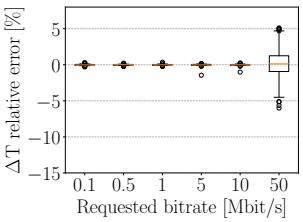
(e) Polling period: 2 s.



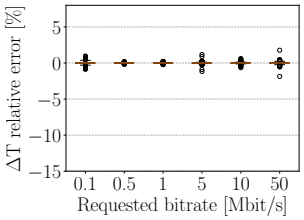
(f) Polling period: 2 s.



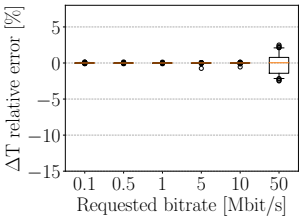
(g) Polling period: 5 s.



(h) Polling period: 5 s.

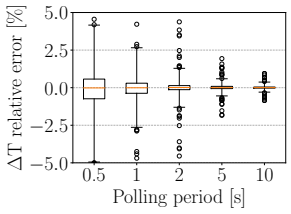


(i) Polling period: 10 s.

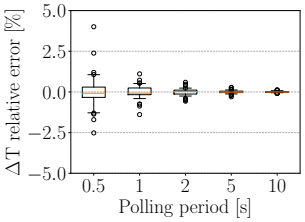


(j) Polling period: 10 s.

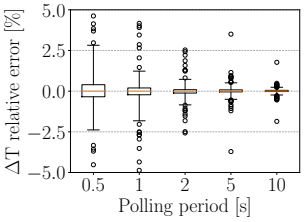
Fig. 3. Relative error when varying requested bitrate using a local (left) and remote (right) controller in the RAN/Internet-crossing deployment.



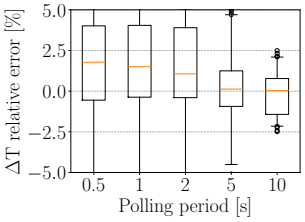
(a) Requested rate: 0.1 Mbps.



(b) Requested rate: 0.1 Mbps.



(c) Requested rate: 50 Mbps.



(d) Requested rate: 50 Mbps.

Fig. 4. Relative error when varying polling period using a local (left) and remote (right) controller in the RAN/Internet-crossing deployment. Y-axes range is restricted to emphasize the differences between the different periods.

controller may have a significant impact on the error when the network used for the in-band communication between switch and controller is loaded. To highlight the effect of polling period given a requested background traffic rate, we refer to Figures 4a–4d. Both with a local (left) and a remote (right) controller, the variability of the relative error—expressed as standard deviation or inter-quartile range—decreases when the polling period increases, as expected. This behavior is consistent on all considered scenarios, therefore we reported only a representative subset. Using a higher polling period, indeed, the uncertainty about the delays is averaged over a wider interval and therefore becomes negligible. This comes to the cost of monitoring the network less frequently, being less reactive to sudden changes in network traffic. It should be also noted that when the polling period is lower, less computational resources are needed, and this aspect should be taken into account, particularly when the controller is deployed on the same node as the switch.

V. Discussion

The results of our investigation have shown that passive measurements using an SDN switch co-located with the mobile terminal (run in the same *container* of applications generating traffic towards the RAN) are both feasible and show a mean relative error close to zero, with a limited standard deviation. In our experimental evaluation we have verified that even if the average of the relative error tends to zero, each measurement carries estimation errors that can be non-negligible, specially for high traffic rates and low polling periods. In order to average the measurement errors, several subsequent measurements can be used, at the expense of the time granularity of the estimation.

To infer the ABw of the radio access link from the measurement of the throughput, the knowledge of capacity of the link is needed. Such information can be derived either from passive measurements and device-provided link status data [13], or by active measurements along the lines of [11]. Both kinds of measurements arguably introduce further and different forms of error, to be compared with the one we estimated in the current paper. We leave these further experimental evaluations to future works. From a more practical standpoint, a sensible upper bound on ABw can be derived from throughput measurements by considering the nominal capacity for a given transmission technology, as obtained by the mobile device

monitoring metadata [13]. Such upper bound would be useful to enforce an *optimistic* admission control, with the advantage of no excess rejection.

The measurement setup considered in this work and implemented on the MONROE testbed is modeling a more complex application scenario. First of all, as the OVS that is located on the mobile node acts as a gateway for the radio access link, it constitutes a monitoring and management point for the local network. The traffic to and from the local network that traverses the shared radio access link is in our experiments emulated using a network traffic generator, but this does not imply lack of generality for the SDN-based measurement we explored in this work. In fact, the aggregate volume of such diverse traffic would be accounted for at the same way, and without any measurement error, by the SDN switch.

On the other hand, the possibility to configure the local SDN network in response to the estimated bandwidth on the RAN link opens new possibilities, such as network admission control for applications requiring a given bandwidth, or traffic engineering based on different priority associated with the application generating the traffic.

VI. Conclusion

We have presented and discussed the experimenting with a passive SDN-based approach for estimating available bandwidth and throughput, applied to the case of MBB networks (4G). More specifically, we analyzed the inaccuracy on time interval estimation, that is the source of error in the proposed bandwidth estimation approach, as the traffic volumes are collected with no approximation nor sampling. In the experimental design we focused on the effect of controller deployment (local to the mobile node, or instantiated on the other end of a path comprising a RAN link), while varying the periodicity at which the monitoring information is collected, in different traffic conditions (emulated with a traffic generator in the mobile node). We have compared the results with a fully controlled testbed in which the network path between the node and the controller is substituted with a wired LAN to highlight the effect of network conditions variability on the estimation procedure. Results confirm that the proposed solution is viable in mobile nodes, showing signs of computing resources exhaustion only in the most demanding case (the SDN controller instantiated on the mobile node in the same container with the SDN switch and the application generating high-throughput traffic). In all the cases, the relative error on the time interval estimation is very low, averaging at zero with standard deviation ranging between 1.31 and 8.65% when the requested bitrate is highest and the controller is remote. Such encouraging results inspire further research goals. For what concerns the effect of the position of the controller, the characteristics of the network path interconnecting the controller to switch is worth being investigated (in terms of e.g., delay, jitter, and packet loss) as each of these QoS indexes can have a different impact on the measurement process. A model summarizing such characterization could be used to predict or improve the estimation accuracy. Moreover, the feasibility of both passive measurement methods and active ones for bandwidth estimation paves the way for a hybrid method, using both kinds of measurements to improve either the accuracy or the efficiency of the estimation procedure.

References

[1] A. Annunziato. 5g vision: Ngmn - 5g initiative. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, May 2015.

[2] B. Dai, G. Xu, B. Huang, P. Qin, and Y. Xu. Enabling network innovation in data center networks with software defined networking: A survey. *Journal of Network and Computer Applications*, 94, 2017.

[3] R. P. Karrer, I. Matyasovszki, A. Botta, and A. Pescapè. Magnets-experiences from deploying a joint research-operational next-generation wireless access network testbed. In *Testbeds and Research Infrastructure for the Development of Networks and Communities, 2007. TridentCom*. IEEE, 2007.

[4] R. P. Karrer, I. Matyasovszki, A. Botta, and A. Pescapè. Experimental evaluation and characterization of the magnets wireless backbone. In *Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization*. ACM, 2006.

[5] Ö. Alay, A. Lutu, R. García, M. Peón-Quirós, V. Mancuso, and et al. Measuring and assessing mobile broadband networks with MONROE. In *IEEE WoWMoM*, 2016.

[6] P. Megyesi, A. Botta, G. Aceto, A. Pescapè, and S. Molnár. Challenges and solution for measuring available bandwidth in software defined networks. *Computer Communications*, 2016.

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Computer Communnication Review*, 38 (2), March 2008.

[8] D. Kreutz, F. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1), Jan 2015.

[9] N. L. Van Adrichem, C. Doerr, and F. A. Kuipers. Opennetmon: Network monitoring in openflow software-defined networks. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*. IEEE, 2014.

[10] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha. Flowsense: Monitoring network utilization with zero measurement cost. In *International Conference on Passive and Active Network Measurement*. Springer, 2013.

[11] G. Aceto, V. Persico, A. Pescapè, and G. Ventre. Sometime: Software defined network-based available bandwidth measurement in monroe. In *Network Traffic Measurement and Analysis Conference (TMA), 2017*. IEEE, 2017.

[12] A. Botta, A. Dainotti, and A. Pescapè. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56(15), 2012.

[13] X. Xie, X. Zhang, and S. Zhu. Accelerating mobile web loading using cellular link information. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '17, New York, NY, USA, 2017. ACM.