An experimental evaluation of the impact of heterogeneous scenarios and virtualization on the available bandwidth estimation tools

Giuseppe Aceto*,◇, Valerio Persico*,◇, Antonio Pescapé*,◇
*University of Napoli Federico II (Italy) and ◇NM2 srl (Italy)
{giuseppe.aceto, valerio.persico, pescape}@unina.it

***Abstract*—Mobile Broadband (MBB) access networks are becoming more and more used worldwide, and the devices adopted to access them are increasing in number and complexity (smartphones, mobile hotspots, vehicular infotainment systems). The highly dynamic nature of such scenarios calls for continuous monitoring and measurement of the network. To this aim, the Available Bandwidth is a performance metric of the utmost importance, albeit hard to estimate in uncontrolled scenarios.**
Shared experimental testbeds such as MONROE are becoming available to offer in-the-field MBB experimenting facilities. In this context, the SOMETIME project is focused on providing MONROE and similar testbeds with the tools to measure Available Bandwidth in MBB scenarios, also taking advantage of the benefit of SDN to perform active and passive measurements. In line with experimental activities planned by the project roadmap, in this paper we discuss the suitability of a number of publicly released ABw estimation tools when run in heterogeneous scenarios. Experimental results confirm that (i) the experimental scenario in which the tools run heavily impacts their performance in terms of accuracy; (ii) the entity of cross-traffic may have different effects on some of the tools, unacceptably undermining the estimation accuracy, depending on limitations of both the specific tools and the setup.

***Index Terms*—Network measurement, Available Bandwidth**

I. Introduction

Mobile terminals are becoming more and more complex systems, hosting multiple network applications with different requirements in terms of QoS. Often, mobile broadband (MBB) access networks are shared among multiple devices by means of mobile hotspots or mobile wireless router (e.g., *Mi-Fi*) as well as wireless networks are used as a backhaul in smart city scenarios [1, 2]. Vehicles themselves are often equipped with network applications for different goals, such as entertainment, travelling assistance, comfort, or maintenance. In addition, at both server- and client-side, virtualization techniques are growingly adopted and will be leveraged more and more in the near future. For instance, Android devices natively run a Java virtual machine, multitenancy is commonly implemented through virtualization in a number of measurement platforms as well as cloud infrastructures [3, 4]. A MBB measurement procedure that does not account for such communication-resource sharing is subjected to limitations by design.
The adoption of software-defined networking (SDN) approach—where the control plane is logically centralized and the control protocol is abstracted and standardized—can help implement scenarios with multiple communications. Besides being a very promising approach with ongoing lively research, SDN provides both high flexibility and standardization, ideal for network measurement studies. End-to-end Available Bandwidth (ABw), that is the maximum rate that a new packet flow can impose on a path without affecting *cross-traffic* (i.e. other flows sharing path resources), is one of the most useful and adopted metrics.
For the reasons above, we considered the implementation of ABw estimation techniques in an SDN scenario a significant advance of the state of the art. In the framework of MONROE[1] this led us to design the SOMETIME project[2]. The MONROE testbed [6] has been designed purposely to experiment with MBB access networks, and provides the suitable infrastructure to implement and evaluate a prototype of the measurement system we devised. SOMETIME plans to leverage the MONROE testbed to perform ABw estimation in an SDN environment from MBB nodes [7].
With reference to the aforementioned testbed, the main intent of SOMETIME is to provide experimenters with a highly valuable tool to measure the ABw in MBB scenarios. In more details, SOMETIME aims at providing the estimation of ABw by active or hybrid measurements, leveraging the SDN paradigm both to tune the technique considering interference with node-local processes (that is a more realistic scenario compared with mutually exclusive measurements), and to mitigate such interference.
According to the SOMETIME project roadmap [7], the evaluation of the suitability of publicly released ABw tools is a primary and critical step, as literature on ABw estimation tools has found their performance to depend upon the measurement context [8]. It is worth to notice that, although our experimentations are tailored on the MONROE platform, the outcomes carried by the SOMETIME analyses are of general interest, also due to the common characteristics and the typical issues related to MBB platforms. The outcome of this analysis is functional to an SDN-based implementation of active or hybrid ABw estimation (demanded to future work)

[1]The MONROE project [5] is an European Union's Horizon 2020 funded research project, aimed building and operating a large-scale experimental platform, targeting MBB and WiFi networks, distributed over multiple European countries.

[2]The SOMETIME project (SOftware defined network-based available Bandwidth MEasuremenT In MONROE) was accepted in the 1^{st} MONROE Open Call.

as described in [7, 9].

In this paper we propose an experimental suitability analysis of the state-of-art tools proposed in the literature. In more details, we discuss the performance of the different tools in terms of accuracy when they are leveraged in heterogeneous scenarios, involving both wired and wireless networks and even host virtualization. The remainder of the paper is organized as follows: Section II introduces the background concepts; Section III details the methodology we followed in our experimental analysis; Section IV reports the main results and the related discussion; Section V ends the paper with the concluding remarks and future work.

II. Related Work

The *available bandwidth* of a network path is a useful metric. Defined on a single link, it is the average of unused capacity during the considered time interval. More formally, *available bandwidth* in the time interval $(t-\tau, t)$ for the i-th link, with capacity C_i, is

$$A_i(t-\tau, t) \equiv \frac{1}{\tau}\int_{t-\tau}^{t} C_i(1-u_i(x))dx \quad (1)$$

$$= C_i(1-\bar{u}_i(t-\tau,t)) \quad (2)$$

where τ is the *averaging timescale* and $\bar{u}_i(t-\tau,t)$ is the average utilization of link i during τ. The available bandwidth on a path is the minimum value of available bandwidth of the links composing the path.

The implied assumptions are that during interval τ the path is fixed and unique (not subject to routing changes or multipath forwarding), and the capacity of each link is constant. Moreover, a common assumption for ABw estimation tools is that the routers operate according to FIFO discipline. These assumptions are not easily met when wireless links are involved: in these cases the measure is heavily affected by τ and the overall time of measurement [8, 10–12]. As a consequence, ABw estimation is not a trivial task, and many tools and techniques have been proposed.

The general approach of active ABw estimation tools is to send probe packet streams characterized by a carefully designed pattern of packet sizes and inter-departure times. The receiver side collects the inter-arrival times, and knowing the pattern that has been sent, evaluates the impact of network traversal, then requires the sender another iteration (with modified parameters for the probe stream), or produces an overall estimation of ABw. The different algorithms differ in the patterns that are used (and thus, for the inference method applied) and for the filtering they adopt to mitigate the many sources of noise. Compared to TCP Achievable Throughput tests ("speed tests"), ABw estimation tools are much less intrusive on the network in terms of probe traffic, and create congestion on the bottleneck links only for a negligible fraction of time (besides having the goal of estimating a different metric). The most cited tools are clearly `pathload` [13] (using equally-spaced packets), and `pathchirp` [14] (using exponentially-spaced packets), with several others less frequently considered as improvements or for comparisons.

Goldoni and Schivi [15] compare various well known tools on a real testbed equipped with 100 Mbps links. They find that the highest accuracy is obtained by `pathload` and `yaz`, although with significant time to convergence and high intrusiveness. Among the outstanding tools, `assolo` exhibits high accuracy, short time to convergence, and low intrusiveness.

TABLE I
Considered Available Bandwidth Estimation Tools.

Tool	Approach	Ref.
`pathload`	Self-Loading Periodic Streams	[13]
`pathchirp`	Self-Loading Packet Chirps	[14]
`yaz`	Self-Loading Periodic Streams (improving `pathload`)	[19]
`assolo`	Self-Loading Packet Chirps (improving `pathchirp`)	[20]

An evaluation of the performance of various tools on a very high speed network is provided by the works of Shriram and Kaur [16] and Murray et al. [17]. In Angrisani et al. [18] the authors assess the performance of `pathload`, `pathchirp`, `IGI/PTR`, and `spruce` with different types of network traffic, i.e. multiple TCP streams and on/off bursts. Results reveal that the lowest standard deviation is achieved by `pathload` and `pathchirp`. We refer to [8, 15] for a comparison and analysis of different ABw estimation tools on wired and wireless paths. A tool is not necessarily better than another, but rather that a calibration is needed to discover and solve possible errors.

Based on the big picture offered by the related literature, we have chosen a number of tools in order to perform the estimation of the ABw in our scenario. The criteria used to select the candidate tools are: (i) the availability of the source code and the possibility to correctly compile it for recent linux-based systems (namely Debian/Ubuntu distributions); (ii) the enhancement technique adopted by each tool to improve accuracy and to mitigate intrusiveness (aiming at extending the variability of techniques tested in our environment). These criteria have led us to choose: *(i) `pathload`* [13], because it has proven to be the reference for accuracy, being also stable with the default parameter values over the many different scenarios in which it has been compared with other tools; *(ii) `pathchirp`* [14], designed as a quicker and less intrusive tool compared with `pathload`; *(iii) `yaz`* [19] that has been specifically designed to improve accuracy and convergence time of `pathload` better tuning its algorithm; *(iv) `assolo`* [20] because it has been implemented with specific focus on timing accuracy.

A summary of the considered tools is reported in Table I.

III. Methodology

In this section we describe the methodology we followed to evaluate the performance of four different ABw estimation tools in heterogeneous scenarios.

For our experimental analysis, we leveraged **3 hosts**—namely Node A, Node B, and Node C—whose hardware and software characteristics are briefly summarized in Table II. The hosts differ by their CPU, operating system, kernel version, and memory size. In addition, while Node C is equipped with both wired and wireless network adapters, Node A and Node B are equipped with a wired Gigabit Ethernet NIC only. Leveraging the characteristics of the nodes, we set **three different scenarios** up, as reported in Figure 1 and described in the following. In the *wired scenario* (Figure 1a), Node A and Node C have been connected through a 100 Mbps LAN. For the *wireless scenario* (Figure 1b), the Node C took advantage of its wireless adapter to connect to a wireless access-point, while Node A was still connected through its wired adapter. Finally, in the *virtualized scenario* (Figure 1c), on both Node A and Node B—that were connected through the switched LAN—lightweight virtualization environments (i.e. Docker containers [21]) were configured.

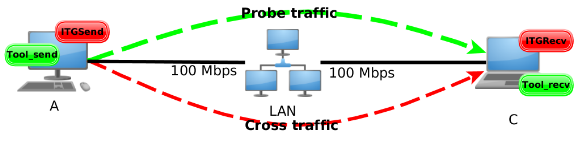
The **four tools** we considered in this analysis are `pathload`, `pathchirp`, `yaz`, and `assolo` (see Section II). Each tool is implemented in a client-server fashion, being made up of two components, namely the sender and the receiver counterparts. In our analysis, both nodes alternately hosted both counterparts to either generate or receive the probe traffic (green dashed lines in Figure 1a, Figure 1b, and Figure 1c).

With the aim of investigating the sensitiveness of ABw estimation tools to **cross-traffic** (i.e., traffic flowing between the two hosts other the probe traffic), the nodes also hosted D-ITG traffic generator [22], in order to emulate cross-traffic flowing between hosts at different rates (red dashed lines in Figure 1a and Figure 1b). We configured D-ITG to generate UDP cross-traffic at constant bitrate, namely 0.001 (negligible but involving the same setup of the others), 8, 16, 32, and 64 Mbps. It is worth noticing that, also due the specific hardware of the hosts, the cross-traffic actually generated by D-ITG may slightly differ from the targeted values. For this reason, we took into account the *received* cross-traffic rate for the calculation of the ABw and hence of the estimation error.

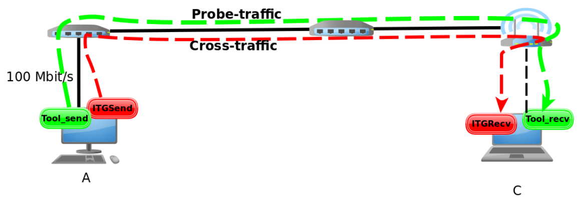
While in both the wired and virtualized scenarios all the setup parameters are known, for the wireless one the path capacity is not known (as it is impacted by the status of the radio channel, which is not under our control). To obtain this information—that is needed for evaluating the performance of the tool—we adopted `iwconfig`, able to provide an upper bound for this parameter.

Measurement procedure

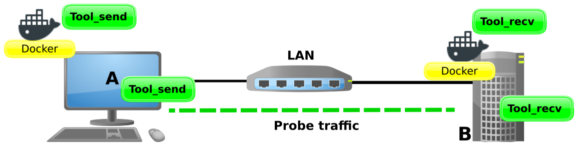
To assess the consistency of our results, we performed multiple experiments in the same experimental conditions. In more details, we performed three runs, if not stated otherwise. The duration of each tool run was set to 45 seconds. In more details, D-ITG was set to generate UDP cross-traffic for the whole duration of the run, while each of the tools was instructed to measure the available bandwidth over a shorter time frame (20 seconds) placed in the middle of the run. Figure 2 reports an example of one experimental run with `assolo`. As also shown in the figure, common operation of active ABw estimation tools is to perform several estimations (e.g., one per second), and then report the average of results. In our experimentation, each run of each tool consisted of around 20 estimations, whose average is reported as result of the run and plotted.



(a) Wired scenario.



(b) Wireless scenario.



(c) Virtualized scenario.

Fig. 1. Considered testbed configurations.

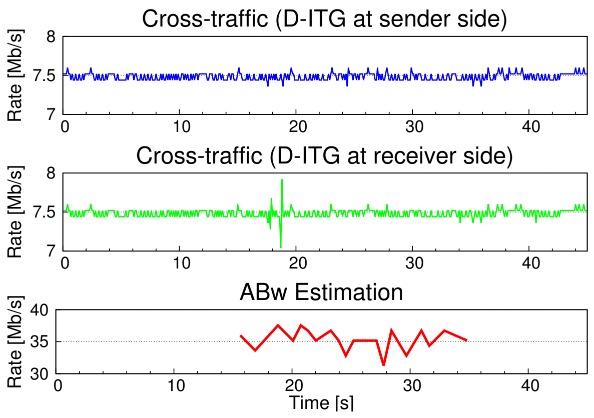


Fig. 2. Example for an experimental run.

It is worth noting that we have accounted for possible uncontrolled interference on the wireless link by alternating the tools in a round-robin scheme (for three rounds, at least). Therefore the measurement points reported in the graphs for each tool in a given experimental condition are actually interleaved with all the other tools. The coherence among measurements of each tool confirms that the differences are unlikely to be due to varying experimental conditions, and are instead characteristic of the tool.

IV. Results

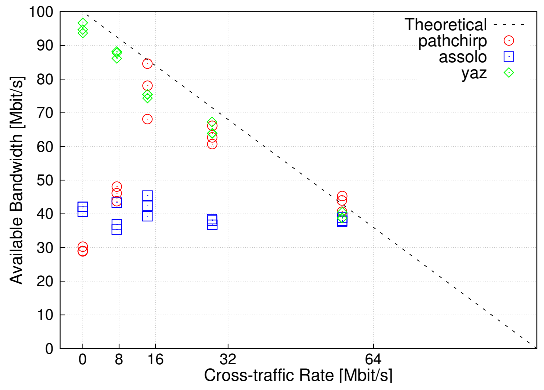
In this section we discuss the performance of the tools in different experimental conditions. In the following, we will discuss the performance in the wired scenario (Section IV-A), in the wireless scenario (Section IV-B), and in the virtualized scenario (Section IV-C).

A. Wired scenario

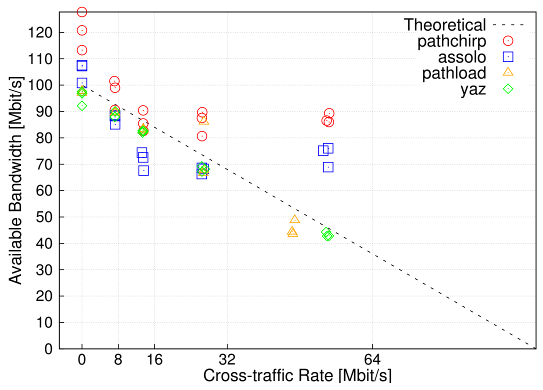
Figure 3 reports the results obtained in the wired scenario. Figure 3a reports how the ABw estimated by each of the

TABLE II
EXPERIMENTAL SETUP DETAILS.

	NODE A	NODE B	NODE C
CPU	I3-3200@3.3 GHz x 4 CPU	E5-2640 v2 @ 2.00GHz x 16	T4200@2.00GHz x 2 CPU
OS	Ubuntu 14.04.5 LTS 64 bit	Ubuntu 14.04.2 LTS 64 bit	Ubuntu 16.04 LTS 32 bit
Kernel	Linux 3.19.0-73-generic	Linux version 3.13.0-24-generic	Linux 4.4.0-66-generic
RAM	4 GiB	16 GiB	2 GiB
NIC	Gigabit Ethernet Adapter	Gigabit Ethernet Adapter	Gigabit Ethernet Adapter Wireless Network Adapter



(a) Traffic direction: NODE A → NODE C.



(b) Traffic direction: NODE C → NODE A.

Fig. 3. ABw estimation results reported by the different tools in the wired scenario.

tools varies with the cross-traffic between the hosts, when the bandwidth is estimated placing the sender on NODE A and receiver on NODE C. While the dashed black line points at the theoretical available bandwidth (computed as the spare capacity on the path, i.e., as *capacity* minus *cross-traffic*), the colored markers report the (average) results for each run according to the different tools. Note how the markers are not perfectly aligned to the values of the targeted cross traffic (i.e., 0.001, 8, 16, 32, and 64 Mbps), due to the existing discrepancies between target rates and those attained with D-ITG.

The main outcome of this analysis is that the rate of the cross-traffic may have a non-negligible impact on the results provided by the tools.

The results computed by `assolo` float around 40 Mbps, as this tool appears to be unable to detect changes in the available bandwidth, notwithstanding the changing cross-traffic rate. As a result, `assolo`'s markers are placed far from the theoretical available bandwidth, with the only exception of experimental scenarios with targeted cross-traffic as high as 64 Mbps. The results obtained with `pathchirp` are also heavily affected by the cross-traffic rates: for this tool we obtained values around the theoretical available bandwidth only when cross-traffic comes at rates higher than 8 Mbps. A possible explanation for the observed phenomena is the traffic generation capability at the sender node, which is solicited in different ways by the different algorithms and their implementations. Only `yaz` reported values evolving with the expected ones. In most of the cases, the tools under-estimate the available bandwidth.

When considering the accuracy of the tools, expressed as the relative error (RE) with respect to the theoretical available bandwidth, i.e., $RE = \frac{O-(C-Xt)}{C-Xt}$, where O is the outcome of the tool, C is the capacity of the path, and Xt is the bitrate of the cross-traffic, we found that RE for `pathload` and `yaz` is always lower than 15%.

The worst result in terms of accuracy have been obtained by `pathchirp` when the estimation is provided in absence of cross-traffic (RE higher than 70%). While the the accuracy of `pathchirp` improves when cross-traffic increases (as the tool is able to provide results comparable to `pathload` and `yaz` in accuracy), the same is not valid for `assolo`, whose RE is always higher than 45% when the requested cross-traffic is lower than 64 Mbps. Note that results for `pathload` are not reported because the tool did not provide results (i.e. did not converge) in this experimental setup.

Interestingly, the results above do not hold when the sender and the receiver components are swapped (see Figure 3b). In this configuration, all the tools provided results. In addition, the lower the rate of the cross-traffic is, the higher the accuracy. Investigating this counter-intuitive behavior, we analyzed in-depth the reason for the lack of convergence of `pathload`, and found that the issue was lying in the discrepancy between the bit rate requested by the algorithm and the one actually generated. Moreover, we found a high correlation between this phenomenon and the number of voluntary context switches: with a voluntary context switches frequency over a given threshold (272 Hz) the accuracy degrades rapidly, and for frequencies higher than 678 Hz the estimation tool even ends in an loop, failing to converge. The frequency thresholds were dependent on both hardware and software configuration. These findings have led to the reimplementation of the `pathload` algorithm with a sender tool (D-ITG) that is able to generate

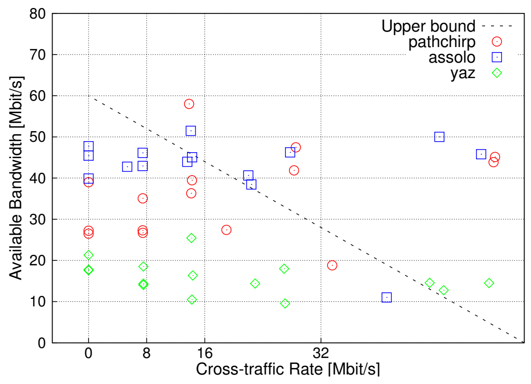


Fig. 4. ABw estimation results reported by different tools in the wireless scenario (traffic direction: NODE A → NODE C).

accurately paced packets over a wider range of hardware and software environments.

Due to lack of space, the details of this analysis and the resulting reimplementation will be left to future publications.

B. *Wireless scenario*

When investigating the performance of the ABw estimation tools in the wireless scenario, it is possible to draw further considerations, as shown in Figure 4 reporting the results related to this analysis. In this analysis we estimated the upper-bound of the capacity of the path relying on the information provided by `iwconfig` command-line tool, and calculate the theoretical available bandwidth accordingly.

Overall, we obtained worse results, as the tools appear to be less sensitive to the changing cross-traffic rate (which implies also a variation of the theoretical available bandwidth). The provided estimation values are more scattered, showing a higher variability across consecutive measurements performed with the same tool. As shown in Figure 4, differently than in the wired scenario, in this case it cannot be told the most accurate tool overall, as tools accuracy rank is heavily impacted by cross traffic. In more details, while `assolo` and `pathchirp` report the most accurate estimation for lower cross-traffic rates, results by `yaz` show the opposite tendency.

C. *Virtualized scenario*

When considering the virtualized scenario, we took into account all the available combinations for client and server running onto either the virtualized (*Guest*) or non-virtualized (*Host*) environments. Therefore, we also considered the cases in which only the sender or the receiver counterpart of the tool is placed into a virtualized environment. As this analysis is specifically focused on investigating the impact of the virtualization on tools' accuracy, we only considered this scenario with no cross-traffic generated by D-ITG.

Figure 5a and Figure 5b report performance results of the considered tools, for both directions. When considering direction from NODE A to NODE B, the tools reported low variability in their results, providing consistent results across different runs. Looking at Figure 5a, only `assolo` showed to be sensitive to virtualization (as the Guest-Guest configuration reported results affected by a slightly higher error).

However, changing the measurement direction, the results worsened in accuracy, with the exception of those provided by `yaz`. As for this direction we observed highly variable but clusterizable behaviors for some of the tools, in this case we performed a higher number of runs (10) to better characterize the observed phenomena. As shown in Figure 5b, the performance of `assolo` and `pathchirp` significantly differs from that experienced in the opposite direction. For the former, higher variability was observed, leading to estimation errors floating between around -10% and $+18\%$ (interestingly, occurrences of under- and over-estimation appear to be strongly related to the specific setup). For what concerns `pathchirp`, two behaviors have been identified: (i) the tool heavily over-estimates the ABw (always settling to around 170 Mbit/s, i.e. providing estimates affected by a 70% relative error); (ii) the tool provides estimates with RE in the range $\pm15\%$. Interestingly, experimental results show how the occurrence share of behavior (i) and (ii) is dramatically impacted by the virtualization setup.

V. CONCLUSION

In line with the roadmap planned for the SOMETIME project, in this paper we have proposed an experimental evaluation of the accuracy of state-of-the-art ABw estimation tools in heterogeneous scenarios involving wired and wireless communication, as well as cutting-edge virtualization technologies, today largely adopted.

Experimental results reported how in the wired scenario the impact that cross-traffic has on accuracy cannot be neglected, although it does not affect the accuracy some of the tools (such as `yaz` and `pathload`). Introducing wireless communication made performance worse, leading to more variable results, to the extent that it is harder to identify a tool with acceptable performance in all the cases investigated. Finally, the results in virtualized setup show how container-based virtualization, according to the running environment of both the sender and the receiver, as well as the direction of the measurement, may lead to dramatic changes in the performance of specific tools (such as `pathchirp`).

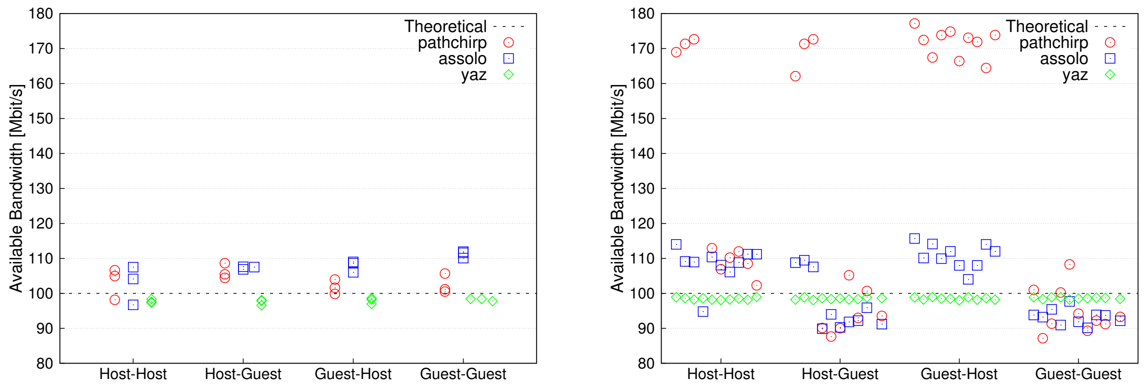
As preliminary investigations suggest that context-switch occurrences is related to cases with poor accuracy, future work aims at investigating the root causes of the observed phenomena, also evaluating the impact of other virtualization solutions and of the size of packets leveraged for active measurements, in order to mitigate their impact of these factors on ABw accuracy in MBB scenarios.

ACKNOWLEDGMENT

The research described in this paper has been partially funded by the SOMETIME project of the 1^{st} MONROE Open Call, funded from the European Union in the Horizon 2020 research and innovation programme under grant agreement No 644399 and art. 11 DM 593/2000 for NM2 srl. The views expressed are solely those of the authors.

REFERENCES

[1] Roger P Karrer, Istvan Matyasovszki, Alessio Botta, and Antonio Pescapé. Magnets-experiences from deploying a



(a) Traffic direction: NODE A →NODE B.   (b) Traffic direction: NODE B →NODE A

Fig. 5. ABw estimation results reported by different tools in the virtualized scenario.

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.

[2] Roger P Karrer, Istvan Matyasovszki, Alessio Botta, and Antonio Pescapé. Experimental evaluation and characterization of the magnets wireless backbone. In *Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization*, pages 26–33. ACM, 2006.

[3] Giuseppe Aceto, Alessio Botta, Walter De Donato, and Antonio Pescapè. Cloud monitoring: definitions, issues and future directions. In *IEEE CLOUDNET*, 2012.

[4] Valerio Persico, Alessio Botta, Pietro Marchetta, Antonio Montieri, and Antonio Pescapé. On the performance of the wide-area networks interconnecting public-cloud datacenters around the globe. *Computer Networks*, 112: 67 – 83, 2017.

[5] The monroe project website. https://www.monroe-project.eu.

[6] Özgü Alay, Andra Lutu, Rafael García, Miguel Peón-Quirós, Vincenzo Mancuso, and et al. Measuring and assessing mobile broadband networks with MONROE. In *IEEE WoWMoM*, 2016.

[7] Giuseppe Aceto, Valerio Persico, Antonio Pescapé, and Gioggio Ventre. Sometime: SOftware defined network-based Available Bandwidth MEasuremenT In MONROE. In *IFIP/IEEE TMA*, 2017.

[8] Giuseppe Aceto, Alessio Botta, Antonio Pescapé, and Maurizio D'Arienzo. Unified architecture for network measurement: The case of available bandwidth. *Journal of Network and Computer Applications*, 35(5), 2012.

[9] Péter Megyesi, Alessio Botta, Giuseppe Aceto, Antonio Pescapé, and Sándor Molnár. Challenges and solution for measuring available bandwidth in software defined networks. *Computer Communications*, 2016. ISSN 0140-3664.

[10] Karthik Lakshminarayanan, Venkata N. Padmanabhan, and Jitendra Padhye. Bandwidth estimation in broadband access networks. In *ACM IMC*, 2004.

[11] L Angrisani, A Botta, A Pescape, and M Vadursi. Measuring wireless links capacity. In *IEEE Wireless Pervasive Computing*, 2006.

[12] F. Chen, H. Zhai, and Y. Fang. Available bandwidth in multirate and multihop wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 28(3), 2010.

[13] Manish Jain and Constantinos Dovrolis. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput. *IEEE/ACM Trans. Netw.*, 11(4), 2003.

[14] Vinay Joseph Ribeiro, Rudolf H Riedi, Richard G Baraniuk, Jiri Navratil, and Les Cottrell. pathchirp: Efficient available bandwidth estimation for network paths. In *PAM*, 2003.

[15] Emanuele Goldoni and Marco Schivi. End-to-end available bandwidth estimation tools, an experimental comparison. In *TMA*, 2010.

[16] Alok Shriram and Jasleen Kaur. Empirical evaluation of techniques for measuring available bandwidth. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*. IEEE, 2007.

[17] M. Murray, S. Smallen, O. Khalili, and M. Swany. Comparison of end-to-end bandwidth measurement tools on the 10GigE TeraGrid backbone. In *IEEE/ACM GRID*, 2005.

[18] Leopoldo Angrisani, Salvatore D'Antonio, Marcello Esposito, and Michele Vadursi. Techniques for available bandwidth measurement in IP networks: A performance comparison. *Computer Networks*, 50(3), 2006.

[19] Joel Sommers, Paul Barford, and Walter Willinger. Laboratory-based calibration of available bandwidth estimation tools. *Microprocessors and Microsystems*, 31(4), 2007.

[20] Emanuele Goldoni, Giuseppe Rossi, and Alberto Torelli. Assolo, a new method for available bandwidth estimation. In *Internet Monitoring and Protection, 2009. ICIMP'09. Fourth International Conference on*, pages 130–136. IEEE, 2009.

[21] Docker website. https://www.docker.com/.

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