UANM: a platform for experimenting with available bandwidth estimation tools

Giuseppe Aceto, Alessio Botta, Antonio Pescapé
University of Napoli Federico II, Italy
{giuseppe.aceto,a.botta,pescape}@unina.it

Maurizio D'Arienzo
Second University of Napoli, Italy
maudarie@unina.it

***Abstract*— In the field of network monitoring and measurement, the *efficiency* and *accuracy* of the adopted tools is strongly dependent on (i) structural and dynamic characteristics of the network scenario under measure and (ii) on manual fine tuning of the involved parameters. This is, for example, the case of the *end-to-end available bandwidth estimation*, in which the constraints of the measurement stage vary according to the use of the final results. In this work we present UANM (Unified Architecture for Network Measurement), a novel measurement infrastructure for an automatic management of measurement stages, tailored to the end-to-end available bandwidth estimation tools. We describe in details its architecture, illustrating the features we introduced to mitigate the problems affecting available bandwidth estimation in heterogeneous scenarios. Moreover, to provide evidences of UANM benefits, we present an experimental validation in three selected scenarios deployed over a real network testbed: (i) we show how UANM is able to alleviate the interferences among concurrent measures; (ii) we quantify the overhead introduced by the use of UANM; (iii) we illustrate how UANM is capable to provide more accurate results thanks to the knowledge of the network environment.**

I. Introduction

The size of the current networks, their heterogeneity, the presence of overlays, the use of multi-path routing are only some of the examples that make the work of measurement tools hard. Over current networks, the measurement process is not a straightforward task, and the availability of measurement infrastructures to support complex measurement experiments is of paramount importance, mainly because single tools can return unreliable results or may not converge. This is the case of end-to-end available bandwidth (AB) estimation tools. Almost every year, the list of these tools is enriched with new proposals, many of them requiring active injection of traffic. This witnesses the interest of research community towards the measure of this metric, used by different kinds of applications (P2P applications, overlay nets, ...) and for different purposes (billing, QoS, ...). Unfortunately, despite the interesting proposals coming from both universities and industries, the efficiency and accuracy of such tools is still strongly dependent on structural and dynamic characteristics of the network under measure, and on manual fine tuning of the parameters related to the adopted measurement method. Moreover, the measurement constraints (accuracy, time duration, intrusiveness, etc.) vary according to the purpose of the measure, and a tool working fine in certain conditions is often highly unreliable in others. Even worst, active estimation methods can interfere with each other [1]. As a consequence, applications that share a measurement end point can suffer a degradation of the measure or can lead to its complete invalidation. For all these reasons, an expert operator is currently needed in order to properly measure the AB in heterogeneous networks. To this end, we designed and implemented ***UANM*** (*Unified Architecture for Network Measurement*), a novel measurement infrastructure to automatically manage end-to-end available bandwidth estimation tools. UANM is based on measurement servers capable to provide (through a simple API) complex and mutually exclusive end-to-end available bandwidth estimations to any interested applications. It can work with dynamically loadable measurement plugins or with standalone measurement tools; in both cases it can also interact with third party measurement tools. A decision engine selects the most suitable method according to the measurement environment, also avoiding the interference between concurrent measurements. In this paper, we describe UANM architecture in details, and we present a preliminary experimental analysis aimed at illustrating the benefits achievable. UANM is geared towards wide adoption among developers of network applications and researchers in the field of network measurement, and a prototype is released under GPL license and with a LGPL API for developing measurement plugins. We believe that UANM allows to overcome the main limitations of current tools, allowing accurate AB estimation in heterogeneous environments.

II. Related Work

Many works have analyzed and compared the available bandwidth tools. As a first need [2] argues in favour of a better design stage of measurement experiments to avoid frequent mistakes, first of all those due to the imperfections of tools. Indeed, a perfect measurement tool does not exist, and every tool should be released with an indication of its accuracy, i.e. the deviation from the true value under different conditions. This does not necessarily mean that a tool is better than another, but rather that a calibration is needed to detect and correct some errors. Among the other studies, [3] [4] tested pathload [5], pathchirp [6], IGI [7] and spruce

[0]The research leading to these results has received partial funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n. 224263-OneLab2 and from the project LATINO of the FARO programme financed by the Compagnia di San Paolo and by the Polo delle Scienze e delle Tecnologie of the University of Napoli Federico II.

on real testbeds and report several pitfalls in which they can end. The work in [8] evaluates the performance of the same tools under different kinds of network traffic - i.e. on/off, bursty, multiple TCP streams - reporting that pathchirp and IGI achieve the smallest standard deviation. The works [9] and [10] present a performance evaluation of different tools on a very high speed network. The existence of biased results due to current implementation technology is showed in [11] and [12]; the latter also proposes a simulated environment for a fair and unbiased comparison of tools. Finally, [13] presents the results of a composition of different techniques. e.g. capacity estimation joined to available bandwidth estimation, provides an improvement of 15% in the measure accuracy.

Although it is not specifically designed for the measurement of the available bandwidth, NetQuest [14] is one of the first examples of an architecture that takes into the right account all the variables related to the measurement on a large scale network. It operates in two steps: it first tailors the experiments to the environment, and then builds a global view of the network. The design step relies on Bayesian techniques and on a subset of active measurements to limit network starvation. In the second step, inference algorithms are applied on the available subset of real data to compute a global map of the network status. In UANM, we outline the same design requirements as: i) the use of different techniques on different part of the network in order to achieve the best result; ii) the augmented accuracy obtained with additional measurements given existing information; iii) the support of multiple users who are interested in different areas of the network. The accuracy of NetQuest is strictly dependent on the amount of data collected. In spite of UANM, NetQuest is not mainly interested in the achievement of the best performance from the measurement of the available bandwidth, since it is more oriented to a wider knowledge of the general network status. Similarly to our approach, in [15] the authors do not propose a new method to measure the available bandwidth, but they propose an architecture called YAZ, whose main goal is to calibrate the existing tools in order to retrieve the best results from the measurements. In contrast with UANM, YAZ does not provide support for concurrent measurement experiments, it does not consider the status of the network under study, and it does not support third party tools. At the end of these considerations, an architecture able to select and calibrate the best tool for the specific measurement, providing users or applications with a result and its error range, was still needed. Such an architecture would also provide a fair environment for the comparison of available bandwidth estimation tools. Although many techniques are already available, they are still not conveyed on a common platform. UANM intends to fulfill this gap thanks to a new paradigm that is described in the next section.

III. UANM Architecture

UANM is a distributed platform whose main features are:

- support for different measurement techniques and tools in a fair environment;

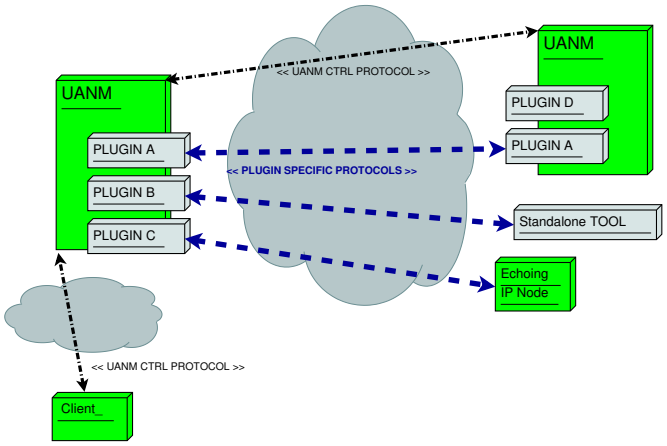

Fig. 1. Deployment Diagram

- full compliance and open interaction with existing tools;
- mutual exclusion of concurrent measurements;
- automatic selection and calibration of the most suitable tool.

It is made up of four different components: daemons, clients, measurement plugins and third-party probes. The main component is the daemon that, in order to fulfill the measurement requests issued by the clients, orchestrates the measurement plugins and interacts with the other daemons and third-party probes. The daemons communicate with the other daemons and with the clients by means of a dedicated control protocol (*uanmProtocol*), and with the third-party probes by using their specific control protocol. The interactions between clients and daemons happens in the classic client-server fashion, while daemons are arranged on a peer-to-peer basis. UANM is written in C language and runs on Unix operating systems, currently supporting Linux platforms. The main module of the platform is *uanmDaemon*, which runs on one or both edges of the path under test, waiting for measure requests from clients. To issue a measure request, an application has to connect to *uanmDaemon* using the *uanmProtocol*, whose API is provided in a C library called *libuamn*. *uanmDaemon* implements multiple measurement tools in the form of dynamically loadable modules, hereafter called "measurement plugins" or, in short, *plugins*. *uanmDaemon* can interact with standalone versions of third party measurement tool, and it can use them as one of the measurement edges. A deployment diagram that depicts various possible settings is reported in Fig.1.

A. uanmDaemon

uanmDaemon is in charge of managing client requests and the plugin set. It is designed to run as a daemon, with little direct interaction with the user, since it is possibly started by boot scripts and stays up all the time waiting for requests from clients. Running as a daemon, the server is detached from the terminal, and it is controlled either by means of a client issuing management commands through the network, or locally by using *posix* signals. When a measurement request is received, *uanmDaemon* first performs a feasibility check aimed at determining if it can actually perform the measurement and

what tool is best fitted to the current *context*. The *context* is a model that comprises both a description of the path status and a description of measurement constraints (information about the measurement, optionally specified by the client). The path status refers to the information about critical characteristics that may affect the measurement process, both structural (e.g. presence of wireless hops or of broadband access links along the path, presence of UANM instances or known third party estimation tools on the other edge, ...) and behavioral (e.g. already congested path, rapidly changing routes, ...). This information is provided by a module called *Knowledge Base* that will be soon described. The measurement constraints are set by the client, as needed for the intended purpose of the measure. As an example, an application may request a group of quick-and-dirty estimations to choose in a set of eligible communication partners (e.g. for server selection), or it could request a non intrusive sampling (e.g. for network monitoring). The server offers a set of measurement choices referred to as *measurement profiles*, each aimed at maximizing some characteristics, trading off the others. If no constraints are specified, the measure will be set in order to reach a trade-off between accuracy and low-intrusiveness, according to the context. An application that is aware of AB estimation parameters may specify detailed constraints such as the average timescale, the standard deviation, the number of estimation tries, the total probe load, or even the exact measurement technique, all to best tailor the measure to its own particular needs. More information about measurement profiles and constraints are provided in Section III-C.

Once the measurement parameters are configured, the daemon sets up the communication between the edges of the path under evaluation and schedules the measure. The scheduler processes the measurements according to the policies. The policy currently implemented is a FCFS queue, but measurement constraints and information about convergence time of the chosen measurement method allow for more complex scheduling criteria. Thanks to the scheduler, it is possible to avoid interference due to concurrent measurements, as explained in the following. In order to gain flexibility and to ease future incremental enhancements, the daemon is divided in the following modules.

- **Plugin Manager** is in charge to control the measurement plugins. At startup, and at each configuration-reload request, it scans the plugin directory for the presence of plugins, it loads each of them (updating the knowledge base with the related information), and monitors their status.
- **Knowledge Base** collects all the information needed by the daemon to process the requests, which are:
  - *plugin list* - the list of measurement plugins known to the daemon, each with its own characteristics and current status;
  - *edge list* - the list of known edges (UANMdaemons, thirdparty probes), each with its own information (type of edge, available methods, time of latest communication, ...);
  - *network scenario* - information affecting the planning of the measurement (first hop characteristics such as capacity, symmetrical/asymmetrical, wireless/wired, ...);
  - *client list* - the list of the clients that contacted the daemon.

  The information in the *knowledge base* is consulted and updated by every other module when it has new informations or needs; it can also be explicitly set in the configuration file (providing the *uanmDaemon* with external knowledge).
- **Decision Engine - Scheduler** checks the incoming requests for feasibility, integrates their specifications and schedules the measurements. Upon the arrival of a new measurement request, the module refers to the *Knowledge Base* to check the feasibility of the measurement, which is fulfilled if the following conditions are met: i) the measurement constraints can be satisfied in the current network context by at least one available plugin; ii) the plugins selected in the previous step are available at both edges; iii) the timing constraints can be satisfied.
  Since all the communications with clients are performed on the network, the Scheduler manages also the timing of the control messages, and the synchronization with commands issued by signals. The scheduler enforces the policy to manage concurrent incoming requests: the measure starts only when both the actors, that we will call in the following `WAITER` and the `INITIATOR`, are in an idle state. This is needed because, to the best of our knowledge and experience with existing available bandwidth tools, these tools are designed to execute single measurements, and they are not reliable in case of concurrent measurements that may be executed from different systems, while sharing a portion of the network. In section IV we also show the effect of the interference among concurrent measures in our laboratory testbed.

B. The measurement plugins

The plugins perform the actual measurement part. If the measurement technique implemented by the plugin needs both the edges to be controlled, there will be two components: a `SENDER` plugin, that generates probe packets, and a `RECEIVER` plugin, that receives the probe packets (and usually performs the computations to estimate the measure). In this case, the two components usually perform different roles: one of them being run as a daemon, waiting for orders (we identify this behavior as `WAITER`), and the other one in charge of initiating the measurement (`INITIATOR`). This information is needed by *uanmDaemon* in order to decide if the command to start the measure must be issued to the local plugin or requested to the other edge. Some techniques allow for the use of an unmanaged edge, so there is only one plugin under control: the plugins using this kind of methods are denoted as `SINGLESIDE`. This information is needed by *uanmDaemon* in order to check the correct matching of available plugins

TABLE I

LIST OF IMPLEMENTED *measurement plugins*.

Plugin Name	Protocol Version	Measure	Sender/ Receiver	Provided profiles
pathChirp	2.4.1	ABW	both	MONITOR
pathLoad	1.3.2	ABW	both	SELECTION, QOS
IGI	2.1	ABW	both	MONITOR
abget	1.0	ABW	Sender	SELECTION

TABLE II

LIST OF PREDEFINED *measurement profiles* WITH CHARACTERIZATION.

Profile Name	Response Time	Accuracy	Repetition Frequency	Averaging interval	Probe Load
SELECTION	wide	wide	once	wide	high
MONITOR	wide	wide	medium	wide	low
QOS	strict	strict	low	strict	high
DEFAULT	medium	medium	once	medium	medium

TABLE III

LIST OF SUPPORTED *measurement constraints*.

Measurement Characteristic	Response Time	Uncertainty	Averaging interval	Probe Load
Type of Bound	upper	upper	exact	upper
Unit	ms	Mbps	ms	Mbps

on the edges of the path while assessing the feasibility of the measurement.

Each plugin can implement three kinds of measurement: fully-specified, constraint-based, and profile-based. They correspond to the kind of measurement requested by the clients through the API (see API Measurement Functions in section III-C). For fully-specified measurements, the clients have to specify the entire parameter string to be used by the plugin. In the profile-based measurements, clients select a predefined profile for the measurements (e.g. minimally intrusive measurement). *uanmDaemon* then calculates the parameter values most suited to this request, using the information in the *Knowledge Base*. For constraint-based measurements, the clients specify the constraints on the measurements (e.g. the maximum duration). Using this information and that in the *Knowledge Base*, the *uanmDaemon* calculates the most appropriate values for the parameters. Each receiver-based estimation method has its own control protocol, used to synchronize the edges and to set the measurement parameters. UANM manages the plugins as gray-boxes, using only high level methods such as initPlugin(), startMeasure(), ignoring underlying details. This eases the transformation of third party tools in UANM-plugins with minimal or no changes in the original code. This also leaves full compatibility of the UANM-plugin version of the tool and its original standalone version. The list of the measurement methods currently implemented as UANM plugins is reported in Tab. I.

C. The Clients and the UANM API

Every application that is able to communicate with a *uanmDaemon* by using the uanm-protocol is a *uanmClient*. A communication API is provided with UANM for this aim. Every application in need of network measurement can benefit of a local centralized orchestration point (*uanmDaemon*) offering enhanced measurement services through a highly abstracted interface. The API provides different measure-request functions ranging from the simplest (e.g. specifying only the endpoints and the type of measure) up to a completely detailed one (e.g specifying the technique to use and its parameters). In the following, we report the kinds of measurement that can be requested by a client.

1) **Measure using Profile**: The general use case for the UANM architecture is represented by an application in need of an available bandwidth estimation, with loose constraints, that can just pick one “ready-cooked” *measurement profile* (see Tab. II). The *measurement profiles* are an abstract classification of use cases in terms of measurement characteristics[1], and they are meant as a choice of a preset combination of values for the actual measurement parameters. They are also used by the *Decision Engine* of the daemon as selection criteria in the choice of the most suitable method among the available ones. For these reasons, the characteristics of the measurement have been chosen as being useful in tool comparison and evaluation (as described in [12]).
2) **Measure with Constraint**: In case the client provides explicit constraints to the purpose of the measure, it is possible to inform the *uanmDaemon* of such constraints by means of this function, specifying the bounding characteristics of the measurement and the value of the acceptable limit. The supported measurement constraints are reported in Tab. III.
3) **Fully Specified Measure**: The maximum control on the measurement process is given to clients through this function, that allows to specify the measurement tool and all the parameters allowed by the tool. The syntax of parameter specification is the same as the command-line syntax of the specific tool, as described in the tool documentation.

To develop ”smart“ clients and administration interfaces, some administration functions are also provided. According to permissions set for the *uanmDaemon*, some of these functions could be forbidden to some clients. In order to perform additional tasks, ancillary functions are also provided. These functions can be used as building blocks to create a complex *uanmClient* or a manager, or to implement the interface needed to integrate UANM in another architecture, such as a distributed measurement infrastructure, an anomaly detection system, or a network monitoring application.

IV. UANM VALIDATION

The availability of different tools embedded in the UANM architecture allows us to execute some first comparative tests on a laboratory testbed. The main purpose of these experiments is to verify the selection of the best tool with the configuration

[1]The measurement profiles have been designed using the available bandwidth estimation as reference, but the same principles and implementations can be used for every supported type of measure.

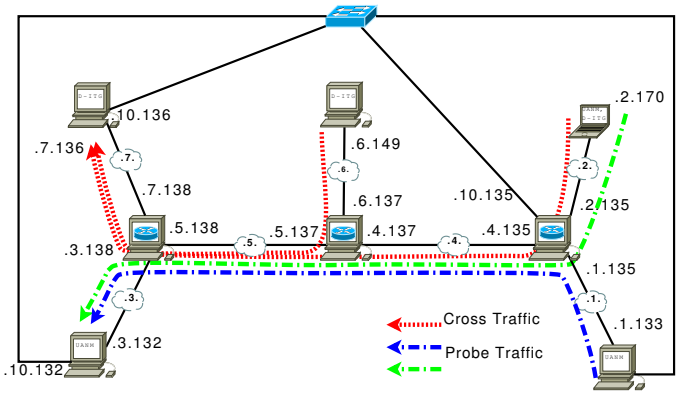

Fig. 2. Testbed used for the experiments.

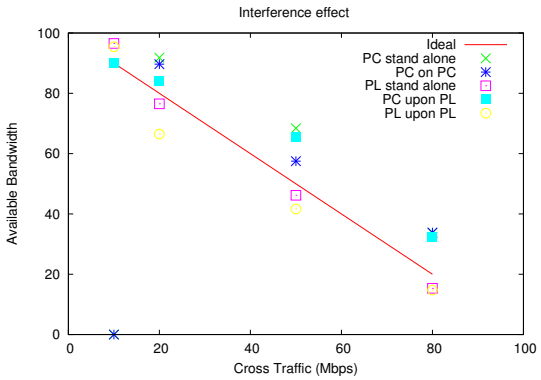

Fig. 3. Effect of the interference among available bandwidth tools.

of the most appropriate parameters operated by the UANM architecture, but also, and not less important, to check the reliability of the measurement tools in different use cases that comprise the interference of two or more tools in execution on the same measurement path at the same time. To validate the UANM architecture we also run some comparative tests with the tools pathchirp and pathload used as stand alone applications or as UANM plugin.

A. Testbed and Tools

For our experiments we set up a laboratory testbed composed of 8 linux-based hosts showed in Fig. 2. The three intermediate hosts act as routers, and the end systems are provided with *pathchirp* and *pathload* tools, as well as with UANM. To emulate different load conditions, we also provide the three end systems on top of the testbed scheme with a traffic generator called D-ITG [16]. Both the measurement and the cross traffic are generated from left towards right.

B. Preliminary results

In our preliminary experiments we aim at demonstrating the basic benefits introduced by the UANM architecture with respect to a trivial use of available bandwidth tools. In this first stage, we focus the attention on the following three issues:

1) the avoidance of the effect of the interference among concurrent measurement processes;
2) the evaluation of the overhead introduced by the adoption of the UANM architecture with respect to the regular available bandwidth tools;
3) the results achieved by UANM when autonomously selecting a tool and its parameters with respect to a basic use of the available bandwidth tools.

Starting from the first point, we show the problem a measure may encounter when more uncontrolled measurement processes share even one single part of the network for a long or short time interval. This problem may occur since the current methods and tools do not provide coordination among measurement stations or any kind of alert feedback from the network. We set all the links of the testbed to 100 Mbps, and we start two series of concurrent measurement processes from sender .170 and .133 towards receiver .132. We also overlap an adjunctive cross traffic from .149 to .136 at different rate of 10, 20, 50, and 80 Mbps so that there is a bottleneck link between router .137 and .138.

Fig. 3 shows a diagram with x axis representing cross traffic and y axis the measured available bandwidth both expressed in Mbps. To increase the readability of such a diagram, we report a line related to the ideal values. All the results are calculated as an average of the outcomes of 10 experiments. We use the same outline for all the experiments. The situations considered are: two concurrent pathload (PL upon PL), pathchirp upon pathload (PC upon PL), and two concurrent pathchirp (PC on PC). Moreover, we also report the results related to a stand alone use of pathload (PL stand alone) and pathchirp (PC stand alone). As reported, there is an effect due to the interference that affects both the tools, as the difference between the stand alone status and the concurrent measures is significant. We also noticed in some experiments with pathload upon pathload that the tool did not converge to a final stage. This result is due to the approach adopted in pathload, which leads the network towards a congestion state for short time interval. The design of UANM daemon avoids the interference effect thanks to the scheduler that coordinates the different clients and activates the measurements on a FCFS basis.

In the following test we show how the adoption of UANM does not introduce significant overhead with respect to the original tools. Fig. 4 reports the results of two series of measurements conducted under different load conditions. In the first measurement we execute pathload, whose results are returned as usual in a range (PL min and PL max), and patchirp (PC). We repeat the same measurement with the same tools embedded in UANM under a *fully specified* profile (UANM PL, as an average on min and max, and UANM PC). As shown in Fig. 4, the difference between the two situations is negligible in all the cases.

The third aspect we highlight is the lack of measurement accuracy that may happen when the available bandwidth tools are used without knowledge of the basic network configuration. For instance, pathchirp gives wrong result on high speed networks unless some of its parameters are correctly set up. The decision engine in UANM is able to automatically select

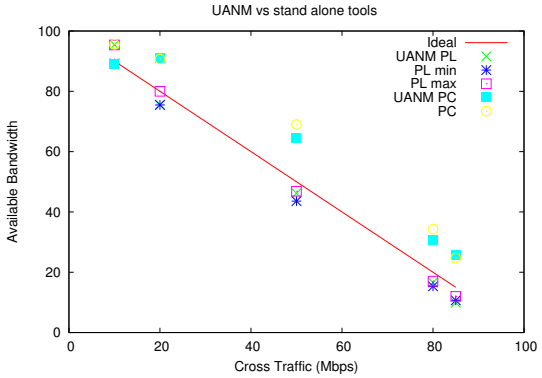

Fig. 4. Overhead introduced by the UANM architecture.

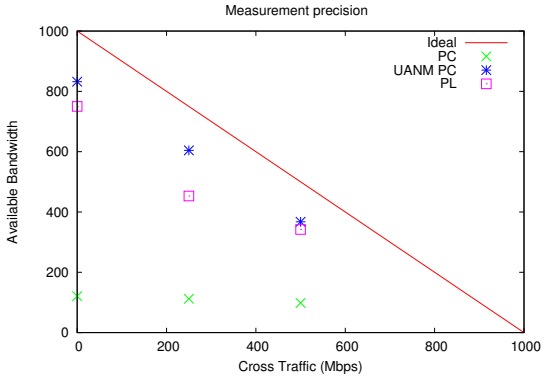

Fig. 5. Selection and trim of the best available bandwidth tool.

the best tool related to the particular network configuration and to recognize some basic configuration in order to set up the most appropriate parameters for the required measure. In this last experiment we show the difference between a straightforward use of pathchirp or pathload with respect to the adoption of UANM onto the same testbed with all the links set to 1Gbps. We made three series of experiments with cross traffic set to 0, 250, and 500 Mbps. As from the Fig. 5, a basic usage of pathchirp returns the worse results, and pathload still underestimates the ideal values. In such a situation, the UANM *decision engine* selects pathchirp according to the strict accurate QoS *measurement profile*, thus preparing the measure with a different configuration of the packet trains. The results of this last series of experiments appear to be the best in all the tested load conditions. The set of plugins implemented in UANM is still in progress and we are confident in a quick extension of the list in order to test the platform in a more challenging scenario that also include wireless links.

V. Conclusions and Future Work

In this paper we presented UANM, a novel platform for supporting and orchestrating complex measurement of end-to-end available bandwidth. Thanks to an open API, existing tools or new ones can be easily pluginized in this platform, still keeping a full compliance with original tools. UANM is able to manage concurrent measurement and to avoid interference, thus increasing the accuracy and reliability of any measurement. We described in details the architecture of the platform, illustrating all the features we introduced in order to mitigate the problems arising in available bandwidth estimation over heterogeneous networks. Comparative experiments carried out on a laboratory testbed showed how UANM can actually provide accurate results in typical situations in which the other tools fail. In the future we plan to implement and evaluate other plugins, widening both the set of supported measures and the number of estimation techniques for each measure. The scheduling policy will be extended from FCFS to other criteria including time constraints or the availability of independent paths on multi-homed daemons. We are considering to modify the work-flow of the daemon to follow the autonomic paradigm: by leveraging the current modular structure of the daemon we are planning to turn UANM in an autonomic architecture. We believe that UANM can be useful for a fair comparison of measurement techniques, even on a WAN scale thanks to the use of overlay network.

References

[1] D. Croce, M. Mellia, and E. Leonardi. The quest for bandwidth estimation techniques for largescale distributed systems. In *ACM HotMetrics*, 2009.

[2] V. Paxson. Strategies for sound internet measurement. 2004.

[3] A. A. Ali, F. Michaut, and F Lepage. End-to-end available bandwidth measurement tools : A comparative evaluation of performances. 2007. http://arxiv.org/abs/0706.4004.

[4] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *3rd ACM SIGCOMM conference on Internet measurement*, 2003.

[5] M. Jain and C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput. *IEEE/ACM Trans. Netw.*, 11(4):537–549, August 2003.

[6] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cot. pathchirp: Efficient available bandwidth estimation for network paths. In *Passive and Active Measurement Workshop*, 2003.

[7] N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE Journal on Selected Areas in Communications*, 21:879–894, 2003.

[8] L. Angrisani, S. D'Antonio, M. Esposito, and M. Vadursi. Techniques for available bandwidth measurement in ip networks: a performance comparison. *Comput. Networks*, 50(3):332–349, 2006.

[9] A. Shriram, M. Murray, Y. Hyun, N. Brownlee, A. Broido, M. Fomenkov, and K. Claffy. Comparison of public end-to-end bandwidth estimation tools on high-speed links. In *PAM*, 2005.

[10] M. Murray, S. Smallen, O. Khalili, and M. Swany. Comparison of end-to-end bandwidth measurement tools on the 10gige teragrid backbone. In *The 6th IEEE/ACM International Workshop on Grid Computing.*, 2005.

[11] G. Urvoy-Keller, T. En-Najjary, and A. Sorniotti. Operational comparison of available bandwidth estimation tools. *ACM SIGCOMM Comput. Commun. Rev.*, 38(1):39–42, January 2008.

[12] A. Shriram and J. Kaur. Empirical evaluation of techniques for measuring available bandwidth. In *IEEE INFOCOM 2007*, pages 2162–2170, 2007.

[13] A. Botta, S. D'Antonio, A. Pescapé, and G. Ventre. Bet: a hybrid bandwidth estimation tool. In *International Conference on Parallel and Distributed Systems*, volume 2, pages 520–524 Vol. 2, 2005.

[14] H Song and Q Zhang. Netquest: A flexible framework for large scale network measurements. *IEEE/ACM Transactions on Networking*, 17(1):106–119, 2007.

[15] J. Sommers, P. Barford, and W. Willinger. A proposed framework for calibration of available bandwidth estimation tools. 2006.

[16] A. Botta, A. Dainotti, and A. Pescapé. Multi-protocol and multi-platform traffic generation and measurement. In *IEEE INFOCOM 2007 DEMO Session, May*, 2007.