NORA: AN INTEGRATED NETWORK MEASUREMENT TOOL FOR END-TO-END CONNECTIONS

Robert Lübke, Peter Büschel, Daniel Schuster, Alexander Schill
Computer Networks Group, Technische Universität Dresden, Germany
{robert.luebke, daniel.schuster, alexander.schill}@tu-dresden.de

ABSTRACT

Current tools and methods for measuring packet-based end-to-end network connections are facing two problems. First, no tool or concept covers all characteristics that are necessary to describe such an end-to-end connection. Second, even specialized tools and concepts lack the required accuracy when used in scenarios with network emulators. Therefore, we developed our own cooperative and active network measurement tool called NORA based on existing and well-established concepts and published it as open-source software. Our evaluation shows that NORA covers all relevant network parameters and exceeds existing tools in terms of accuracy, especially when used with network emulation. In this paper we describe the general concepts, implementation and evaluation of NORA, which has the ability to analyze rates relating to bandwidth, packet delay, loss, duplication and reordering in a single measurement for UDP and TCP.

KEYWORDS

active network measurement; bandwidth; loss; delay; duplication; reordering.

1. INTRODUCTION

Much research has been done in the area of network measurement in the last decades. It is still required to perform network measurements in many scenarios, for example for quality of service verifications and traffic engineering. In our use case we need accurate measurements for testing and verifying network simulation and emulation environments to evaluate their accuracy and performance [Lübke et al, 2014].

The research community and industry have developed several methodologies to measure certain characteristics of a network connection. Existing approaches are usually classified according to the following aspects. Network measurement can either be performed on the link level or end-to-end. Link level measurements are used to characterize direct connections between two network nodes. The results are mostly very accurate, but it also requires access to the corresponding routers. With end-to-end measurements, one can describe the characteristics of a whole network path. Another aspect is the way the measurement is performed. Active methods insert additional traffic into the inspected network. Passive methods do not inject probe packets, but they only observe the existing traffic. The third possibility is inline measurement. These methods only use existing traffic but they modify it, for example by putting additional measurement information into the header fields of IPv6 packets that pass by. Then, other network nodes can read and evaluate this data. The last criterion is the number of used measurement points. Cooperative methods require multiple measurement spots that access the probe packets. Non-cooperative methods, however, do not need any counterpart for the measurement. They mostly use ICMP messages or TCP SYN packets as probe traffic.

Many specialized tools that implement these concepts have been developed. Therefore, it is possible to determine all relevant parameters in a network experiment. However, no current tool is able to determine all relevant parameters of end-to-end network connections in one measurement. Currently network experimenters have to use several tools to find out the complete network characteristics. This is tedious work, especially if the measurements are repeated for multiple network technologies. Furthermore, we found that the accuracy of some of the existing tools is not sufficient if used with network emulators like in our test bed.

We therefore developed an own cooperative integrated tool named NORA (<u>N</u>etwork-<u>O</u>riented <u>R</u>ates <u>A</u>nalyzer) and present it in this paper. In conclusion, the main contributions of this paper are:

- We developed the open-source active network measurement tool NORA that covers all relevant network metrics in one single measurement for UDP and TCP packets.
- We used and adapted established measurement concepts, but also developed new methods for measuring packet loss, duplication and reordering within TCP traffic.
- NORA achieves good accuracy compared to established network measurement tools, especially when used with network emulators.

The requirements of a network measurement tool depend on the scenario where it should be used. In our use case of evaluating the accuracy of network emulators and simulators we analyzed the following requirements. First of all, the network measurement tool must cover all relevant parameters and metrics (see Section 2). The determination of these metrics should be done in one single measurement. This facilitates the experimental setup and allows the repetition of experiments in a time-saving manner. Furthermore, the measurement results must be accurate enough to allow comparisons between the different network emulators and simulators. A good configurability is also required. This includes setting the measured metrics, the used ports, the measurement type (one-way, round trip, bidirectional), the packet sizes and intervals as well as the used transport protocols (TCP or UDP). Also, the tool must be usable on any platform and it should be easy to install and use.

In the remainder of this paper we first discuss relevant metrics of end-to-end network connections and then analyze existing research work and measurement tools for these metrics according to the stated requirements. We then present the concepts of our measurement tool called NORA. The evaluation section discusses the accuracy of NORA compared to existing solutions. The last section concludes the paper.

2. RELEVANT NETWORK METRICS

There are six relevant network parameters that form the characteristics of each end-to-end network connection. These parameters and their metrics are discussed in the following.

Obviously one of the most important parameters of a network connection is the **bandwidth** as it determines how much data can be transferred per time unit. The capacity C is a metric for the maximum bandwidth. It depends on the physical medium of the transmission channel, but neither on time nor on the current traffic. In contrast to C, the available bandwidth A depends on the background traffic (utilization u with $0 \leq u \leq 1$) that is currently sent in parallel over the same connection. The available bandwidth can be determined with $A = C \cdot (1 - u)$. Regarding end-to-end connections one often wants to find out how much payload can be transferred in a given amount of time. This can be specified with the bulk TCP thoughput *BTT* that is based on the transport protocol TCP.

The **packet delay** specifies the time between sending and receiving a message. One-Way Delay OWD is used for measurements of only one communication direction whereas Round Trip Delay *RTD* is the time between sending a message and receiving the corresponding answer. The overall delay of a packet in an end-to-end connection consists of the routing delay D_R (processing and queueing inside the router), the transmission delay D_T (placing the packet onto the link) and the propagation delay D_P (passing the packet from one end of the link to the other). [Crovella and Krishnamurthy, 2006] In real networks packet delay is usually not constant. The variability of packet inter-arrival times is often called **packet jitter** ΔOWD. Based on the definition of packet delay of two subsequent packets (*j* and *j-1*) ΔOWD^j can be determined as follows: $\Delta OWD^j = \left(D_R^j - D_R^{j-1}\right) + \left(D_T^j - D_T^{j-1}\right) + \left(D_P^j - D_P^{j-1}\right)$. Assuming that packets of the same length are sent over the same path the transmission and propagation delays of packets j and *j-1* equalize. In this case ΔOWD only depends on the different routing delays and therefore on the varying load and queue sizes of the routers due to parallel traffic. Packet jitter consequently occurs in all non-exclusively used computer networks.

Packet loss can occur for multiple reasons. Especially in error-prone transmission channels packet contents can get changed. If these bit errors are detected by error recognition mechanisms (e.g. checksums) and cannot be corrected, the whole packet is discarded and counted as loss. Another reason is congestion of the involved network elements. If the queue of the output link of a router is full, the scheduling algorithm either has to drop incoming or already queued packets. [Crovella and Krishnamurthy, 2006] Depending on the direction there are two metrics for packet loss: One-Way Loss Rate (*OWLR*, [Almes et al, 1999]) and Round Trip Loss Rate (*RTLR*). $OWLR$ is more accurate because it considers asymmetric connections. It is computed with the number of sent packets P_{send} and actually received packets P_{rec} over a given measurement period as shown in the following equation: $OWLR = 1 - P_{rec}/P_{send}$. If the network changes the order of two packets, this is called **packet reordering**. Reasons for this effect are parallelism within the routers, load sharing between multiple paths and path changes during transmission. The One-Way Packet Reordering Rate (*OWRR*) specifies the ratio of packets that arrive in wrong order and all sent packets. **Packet duplication** occurs if a packet that was sent once reaches the receiver multiple times. It can also be seen as a special form of reordering [Jaiswal et al, 2007], but in the following we discuss duplication as a separate effect. The One-Way Duplication Rate (*OWDR*) specifies the ratio of duplicated packets that reach the receiver and all sent packets.

3. RELATED WORK

The integrated measurement of multiple metrics was already investigated in two research areas. On the one hand there are network measurement frameworks like *NetQuest* [Song et al, 2009] and *Atmen* [Krishnamurthy et al, 2005]. These systems provide architectures for distributed measurements in large-scale networks. As these systems should be publicly available for network experimenters, they focus on architectural aspects like security and prevention of attacks. The measurement methodology itself is not the main scope. On the other hand there are network monitoring systems like *Anemos* [Danalis and Dovrolis, 2003] and *Flame* [Anagnostakis et al, 2006]. These systems focus on the observation of changing parameters in a whole network. Users are able to create monitoring rules that can trigger alarms.

Although most of the mentioned systems follow a flexible scripting approach, no system supports all relevant metrics that were discussed in the previous section. Furthermore these systems are not easy to use for comprehensive end-to-end measurements, because they focus on how to perform the measurement in whole networks and not on the methodology itself. Therefore, we further investigated on established active measurement methods that could serve as a basis for our integrated measurement tool.

Bandwidth: Existing techniques for estimating the capacity *C* of a path are Variable Packet Size (VPS) probing and Packet Train Dispersion (PTD). VPS probing estimates *C* of each single link inside a network path. The tool *Pathchar* implements the VPS approach. It measures the round-trip delay from the source to each link with the help of the time-to-live (TTL) field inside the IP header and uses this delay as a function of the size of the probe packets. PTD estimates *C* for the complete path with the limitation, however, that there must be no cross-traffic. PTD is implemented in *Pathrate* [Dovrolis et al, 2004]. This tool analyzes the distribution of the inter-arrival times of long packet trains and uses statistical methods to estimate *C*.

Commonly used principles for available bandwidth estimation are the Probe Gap Model (PGM) and the Probe Rate Model (PRM) [Guerrero and Labrador, 2010]. An example of a PGM tool is *abing* [Navratil and Cottrell, 2003] which sends packet pairs with a known initial dispersion and determines *u* by measuring the final dispersion of the received packets. PRM is based on the exploitation of self-induced congestions and relies on the fact, that one-way delays of probe packets increase, if their sending rate is higher than *A*. To find the turning point at which delay starts to increase, several techniques were developed, differing in structure of the probe stream and the way *A* is derived from the received stream. Example implementations are *Pathload* [Jain and Dovrolis, 2002], *Yaz* [Sommers et al, 2006] and *Assolo* [Goldoni et al, 2009]. *Pathload* uses a technique called Self-Loading Periodic Streams (SLoPS) and tries to find the exact turning point by sending constant bit-rate streams at varying rates. After each stream, *Pathload* changes the sending rate based on a one-way delay trend analysis at the receiver. *Assolo* uses the same technique, but it sends variable bit-rate streams consisting of variously spaced packets to test a wide range of rates with one stream.

The *BTT* metric can be determined with *Iperf*. It measures the amount of sent TCP payload for a fixed period of time and calculates the *BTT*.

OWLR, OWRR and OWDR: The IP Performance Metrics Working Group defines the One-way Active Measurement Protocol (OWAMP, [Shalunov et al, 2006]) that is able to determine loss, duplication and reordering. Other tools covering these metrics are *Iperf* and *QoSMet*. Instead of *OWDR*, they only provide *OWRR*; duplicated packets are treated as reordered. The measurement concept for all three metrics is simply based on the analysis of the packet sequence numbers. A packet is deemed as lost, if its sequence number is not recorded by the receiver. It is duplicated if its sequence number was already recorded. Finally, it is marked as reordered, if its sequence number does not equal the expected value.

RTD and ΔOWD: Techniques determining *RTD* are often based on ICMP echoes, the protocol behavior of TCP SYN-ACK/RST pairs or on cooperative processing similar to client-server systems. Tools concentrating on the first two techniques operate without the need to access the remote station. *Ping* belongs to that category and uses timestamps within ICMP echoes. Other tools like *hping* or *tcpping* can also measure the *RTD* with the help of the TCP SYN-ACK/RST mechanism, in which the sender times the outgoing SYN packet and the incoming ACK (or RST) packet. However, firewalls usually block that behavior to avoid SYN-ACK attacks. The principle of the third technique works in a similar way to that, but requires access on both endpoints of a connection. In this way the *RTD* can be measured under even more realistic conditions. *Abing* uses exactly this technique. It consists of a reflector running on the remote machine and a main program which sends UDP probe packets to the reflector and analyzes the received packets. The computation of ΔOWD can be done in different ways. Tools like *QoSMet* measure the arrival time of all packets in a stream and calculate the inter-arrival time of each consecutive packet pair. Other tools extend this approach by applying additional filters. An example is *Iperf* which implements the jitter calculation of RTP (Real-Time Transport Protocol).

After the discussion of related work we conclude that no current network measurement tool covers all presented metrics in one single measurement as it is required in our scenario. Although, there are many established measurement concepts and methods that can serve as a basis of an integrated measurement tool. In the following we discuss our own concepts and their implementations that match the stated requirements.

4. NORA MEASUREMENT TOOL

Building on the preceding sections, the main concepts and also some implementation details of our measurement tool NORA will be explained in this section. NORA is a client-server based multiprocessing and multithreading tool which includes mostly active but also some passive measurement methods. Its general structure is illustrated in Figure 1. The architecture of NORA is designed in such a way, that it can act either as a server or as a client. The measurement server sends the probe streams to the requesting clients. Measurement parameters like the amount of streams, the period of time and the source probe port are sent through the control port. If a client connects to the server towards the control ports, the server sends the probe streams towards the measure port to the client and the measurement takes place.

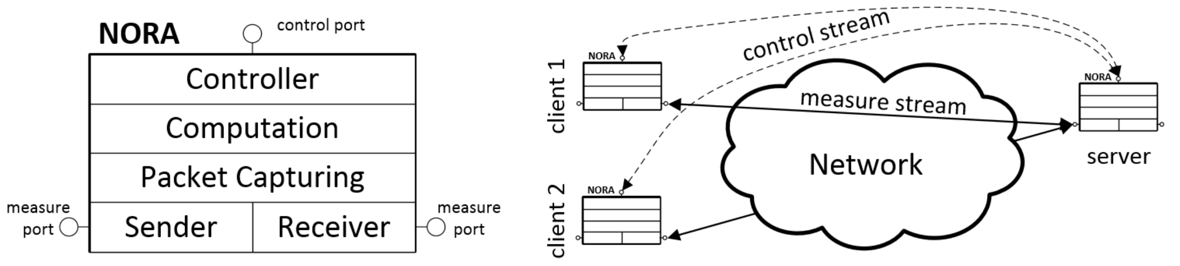

Figure 1. Structure of NORA and its use as server and client

The sender (*Tx*) represents the active part regarding the measurement methods and sends probe streams of precisely crafted packet trains through the probes port. Timing is a decisive factor inside the sending process. To get accurate results, especially for bandwidth estimations, the compliance of the inter-departure time between consecutive packets, trains or streams has to be as accurate as possible. To achieve this NORA implements busy sleep with state-of-the-art timers and sets itself to high priority for Unix-like platforms. The used timers are based on the timestamp counter and provide high time stability with accuracy in the range of microseconds.

On the receiver-side the packet capturing represents the passive measurement part. It captures the probe streams through the measure port and extracts the needed packet information. For this purpose it uses *libpcap* on Unix-like systems and *WinPcap* on windows based systems. This gives NORA the advantage of capturing a timestamp of a packet when the network interface's device driver handles it. With an accuracy in the range of microseconds these timestamps are more precise than timestamps created at application level. Another advantage of the passive method is its ability to capture the behavior of TCP directly on the network interface, without the flow control. NORA captures each packet at OS kernel level and can therefore detect reordering and duplication of TCP packets. Normally, the OS kernel covers all TCP mechanisms and user level applications cannot observe such packet effects.

As a last step, the computation instance gets all information from the sniffer, calculates the required values and sends them to the controller, who in turn shares these results with the server and/or client. So both sides get all results. The computation instance can also detect and handle interrupt coalescence.

In the following we describe how NORA measures and obtains the values for the different metrics.

Bandwidth: The values of BTT and A can be estimated for each one-way measurement. By definition BTT is based on TCP traffic and calculated according $BTT = (\sum_{j=2}^{p} L_j)/(t_{end} - t_{start})$, where L_j is the size of the j-th packet without the TCP overhead and p is the amount of packets received between the timestamps t_{start} and t_{end}. A timestamp is taken after receiving the last bit of a packet. Duplicated packets are included in the calculation, whereas lost and control packets are not taken into account. Thus, only the received payload will be summed up.

In contrast to *BTT*, the values of *A* will be estimated with the help of UDP packets. The estimation consists of two phases. In the first phase, a slightly modified *PTD* technique is used. The original *PTD* rate is calculated without the consideration of UDP packet fragmentation. At the end of this phase, NORA uses the *PTD* result as a starting rate for the second phase, which is based on *SLoPS* and follows the principles of *Pathload* with some alterations. NORA also iterates over different rates to find an upper and lower bound of *A*. As described in Section 3, the basic idea behind this technique is to find the turning point, where the one-way delay starts to increase at a given data rate. NORA uses packet pair dispersions (*PPD*) to analyze increasing trends (I-trend) or non-increasing trends (N-trend). The algorithm starts by sending a stream of packet trains from the server to the client. To derive an I-trend from *PPD* our tool analyzes each packet train in a stream with the help of *Pathload*'s Pairwise Comparison Test (*PCT*) and a modified version of its Pairwise Difference Test (*PDT*). Both test procedures are used, because there are cases in which only one of them can detect an I-trend. To determine the final trend of a train, NORA considers both, *PCT* and *PDT*. If both approaches have complementary results or show ambiguous trends, the train is discarded.

As a next step NORA counts all I-trends and N-trends in a stream. If one of them is in the majority, NORA marks the stream accordingly. Otherwise the stream will be discarded. So, in contrast to *Pathload*, NORA makes hard decisions and has no need of a "grey region" [Jain and Dovrolis, 2002]. Further improvements like a variable threshold for packet loss and an alternative adjustment of the next probe rate make NORA more robust and let it return results more quickly.

OWLR, OWRR and OWDR: The computation of these rates is based on sequence numbers and is different for UDP and TCP. As UDP has no acknowledgements, the concepts for UDP measurements are quite simple. The algorithm starts by sending a stream of packet trains. After *Tx* has finished and has waited a predefined period of time, it sends an additional control message to the receiver (*Rx*). Reordered, lost and duplicated packets can be found by analyzing the sequence numbers of the received packet train. The corresponding rates OWLR, OWRR and OWDR are then calculated with the size of the packet train.

Using TCP for the measurements requires consideration of lost, reordered and duplicated acknowledgements. Hence, we developed an adapted measurement version. The main difference is that *Rx* and *Tx* maintain complete lists of all sent and received packets. After the actual measurement is performed by sending multiple packet trains, these lists are exchanged and both sides use them to analyze which packets got lost, reordered and duplicated and to calculate the corresponding metrics.

RTD and ΔOWD: The *RTD* values are computed at sender-side with the help of timestamps of outgoing and incoming packets captured by the sniffer. Therefore, a packet will be sent to *Rx* and the sniffer at sender-side captures the timestamp of the outgoing packet. Also, a timer will be started. If the time-out is reached, the packet counts as loss. Immediately upon the receiving of a packet, the sniffer at receiver-side recognizes that packet and sends a copy of this back to the sender through the same port. The sniffer at sender-side recognizes the returning packet and captures its timestamp as well as its sequence number. If the sequence number is the expected one, *Tx* subtracts both captured timestamps. The result is the *RTD* of a packet and its copy. The measurements with UDP and TCP packets do not differ significantly. To measure the *RTD* of TCP traffic, NORA uses only packets carrying payload, so control packets (e.g. *ACK*, *FIN*) are discarded.

The values for ΔOWD are again obtained using the list approach. This time *Tx* maintains a list of all sent packet numbers and the corresponding timestamps. After the measurement, the list is transferred to Rx, where the ΔOWD values are calculated by subtracting the inter-arrival times of consecutive packets (j and j-1) from their inter-departure times: $\Delta OWD^j = \left|\left(t_{sent}^j - t_{sent}^{j-1}\right) - \left(t_{rec}^j - t_{rec}^{j-1}\right)\right|$. Lost and duplicated packets are skipped and reordered packets are sorted for these calculations.

5. EVALUATION

To evaluate our concepts and their implementation within NORA we show that all analyzed requirements (see Section 1) are fulfilled. In Section 2 we discussed the characteristics of end-to-end network connections and found the corresponding metrics: available bandwidth *A*, bulk TCP throughput *BTT*, one-way delay *OWD* and its variation ΔOWD, packet loss *OWLR*, reordering *OWRR* as well as duplication *OWDR*. All of these metrics are supported by NORA and it can also determine these characteristics in one single measurement. As NORA is a command line tool, it is configured via command line parameters. Among others the user can define the measured metrics, the measurement type, the used ports, the packet sizes and intervals as well as the employed transport protocols. Thus, NORA also matches the requirement of good configurability. Due to its implementation in Python, NORA can be used on any platform that can provide a Python interpreter. Despite all configuration possibilities NORA is still easy to use, because all parameters are optional and have reasonable default values and no additional installation step is required. In the following we want to proof that NORA also provides the required accuracy of the measurement results.

In order to evaluate our tool, we use a **testbed** consisting of two low-cost computers running Debian 6.0 (Kernel 2.6.32), which are connected through a "Linktropy 7500pro" hardware network emulator manufactured by Apposite. The hardware emulator enables us to configure all of the relevant network characteristics with a high precision and it can also emulate cross traffic.

Each metric describes a single network characteristic. In the context of the evaluation of NORA, we preferred to examine them separately. Additionally, we compared NORA with well-known tools specialized on the specific metric. The **measurement methodology** and the corresponding **results** for the different metrics are discussed in the following.

Bandwidth: Experiments of *A* and *BTT* were performed in 12 different testbed configurations, whereby one configuration stands for a predefined capacity *C* with one of three fixed utilizations *u*. The values of *C* (384 kbps, 7.2 Mbps, 54 Mbps and 120 Mbps) are inspired by known technologies that provide Internet access. The fixed values of *u* were 10 %, 25 % and 60 %. For each configuration we measured 100 samples. Also, we compared NORA's results with *Yaz*, *abing* and *Assolo* for *A* and with *Iperf* for *BTT*. These tools were chosen, because they produced the best results from all discussed tools in Section 3.

Table 1. Predefined values for the bandwidth and measured values for the Available Bandwidth *A* at different utilization levels as well as for the Bulk TCP Throughput *BTT* without utilization

Predefined Values		**Measured Values**					
		A (in kbps) [relative deviation of expected value]				**BTT** (in kbps)	
bandwidth	**utilization**	Nora	Yaz	Assolo	Abing	Nora	Iperf
	10 %	308 [-10.88 %]	-	-	-		
384 kbps	25 %	236 [-18.06 %]	-	-	-	369	366
	60 %	180 [+17.19 %]	-	-	-		
	10 %	6438 [-0.65 %]	5566 [-14.10 %]	5969 [-7.89 %]	6984 [+7.78 %]		
7200 kbps	25 %	5191 [-3,87 %]	3065 [-43.24 %]	5082 [-5.89 %]	6979 [+29.24 %]	6898	6870
	60 %	3537 [+22,81 %]	-	3930 [+36.46 %]	6960 [+141.67 %]		
	10 %	47748 [-1,75 %]	47394 [-2.48 %]	44170 [-9.12 %]	52911 [+8.87 %]		
54000 kbps	25 %	38058 [-6,03 %]	38471 [-5.01 %]	40548 [+0.12 %]	52199 [+28.89 %]	50993	51500
	60 %	24294 [+12,47 %]	21485 [-0.53 %]	36632 [+69.59 %]	50535 [+133.96 %]		
	10 %	105456 [-2,36 %]	104845 [-2.92 %]	100269 [-7.16 %]	115889 [+7.30 %]		
120000 kbps	25 %	87350 [-2,94 %]	86104 [-4.33 %]	93985 [+4.43 %]	113945 [+26.61 %]	82939	114000
	60 %	47263 [-1,54 %]	47301 [-1.46 %]	80235 [+67.16 %]	108576 [+126.20 %]		

Table 1 shows the results of our bandwidth experiments for the metrics *A* and *BTT*. Regarding the available bandwidth the first obvious finding is that only NORA was able to measure the small bandwidth of 384 kbps. All other tools either got stuck during the measurements or produced completely inapplicable results. Furthermore, it can be seen that *abing* measures the same values for all three utilization values. Our explanation for this behavior is that *abing* does not work with the artificially created background traffic of our hardware network emulator. NORA, *Yaz* and *Assolo* mostly match the expected values with a relative error of about ±3 %, while NORA shows less outliers than the other tools. Regarding *BTT* we compared NORA with *Iperf*, because it also measures *BTT* with TCP payload. This comparison ends even, because both tools approximately deliver the same appropriate results. Only for the highest bandwidth of 120 Mbps NORA seems to determine a *BTT* value that is too small. We did not explicitly calculate the expected *BTT* values because of the unknown number and sizes of the TCP acknowledgments and protocol headers.

OWLR, OWDR and OWRR: To be able to measure rates with a precision of 0.01 %, we measured these metrics with trains of 10,000 packets. The respective metric was analyzed with two different settings, whereby we gathered 100 samples for each setting. The rates for these settings were 0.01 % and 1.95 %. The first one reflects a typical value and the second should show the precision of NORA.

Loss, reordering and duplication results are shown in Figures 2a and 2b. The mean of each rate and used protocol is nearly identical with the predefined value (*expectation*). Outliers can be found when using TCP. The hardware emulator and of course a real end-to-end connection do not differentiate between control packets and payload packets on which NORA focuses. The inner statistic of the hardware emulator also counts acknowledgements. Thus, the rates based on TCP streams are often under the expected values.

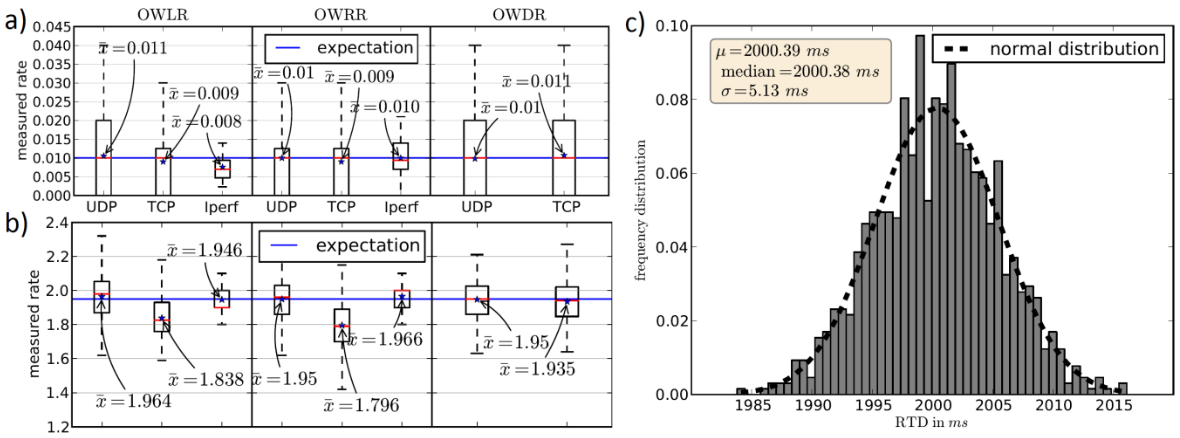

Figure 2. The left side shows the results for *OWLR*, *OWRR* and *OWDR* measurements based on UDP/TCP packets in comparison with *Iperf* with a presetting of 0.01 % (a) and 1.95 % (b) for each rate. The right side (c) shows the frequency distribution of 10,000 *RTD* measurements based on UDP.

RTD and ΔOWD: The delay values were measured with 1,000 samples on a connection modelled with a normal distribution. The minimum value and the mean value of this distribution were 0.2 ms or rather 2000 ms with a standard deviation of 5 ms. The high mean value is based on the default time-out for round trip measurements in NORA.

The frequency distribution (see Figure 2c) of our UDP *RTD* measurements is very close to the predefined normal distribution (dashed line). The values of the standard deviation σ and the mean μ are equal to the predefined settings. So, also for this case NORA shows its high reliability. Results for measurements with TCP are similar to that. The concrete results regarding OWD are left out for this paper, because they are similar to the results of RTD. NORA shows the right tendency of the packet delay distribution function.

Findings: The estimations and measurement results in comparison with well-known tools show the reliability and accuracy of NORA. In some cases NORA is even more precise than the established tools and its results are closer to the expected values.

6. CONCLUSION

In this paper we argue that current tools for network measurement mostly specialize in measuring one or maybe a few network parameters, but no solution covers the overall characteristics in one measurement. We have tackled this issue and contribute with our integrated network measurement tool NORA that covers all relevant parameters of end-to-end network connections. Its development was driven by the analyzed requirements of our scenario in which we measure the accuracy of current network emulators. But the resulting concepts and their implementation are so generic that it can be used in multiple other use cases by other researchers as well.

The implemented measurement concept for the estimation of *A* is mostly based on existing and well-established methods, which we adapted and improved. As a result the estimation becomes faster, more robust to packet loss and can handle a larger range of bandwidth. But we also contribute with newly developed concepts mainly for measuring TCP traffic. In short, we implemented a *pcap*-based solution to detect *OWLR*, *OWRR* and *OWDR* at kernel-space. New methods are implemented for their computations. Also the captured timestamps are more precise than those taken with conventional timing methods.

The discussion of related work and our evaluation showed that existing solutions have deficits in accuracy when used with network emulators and simulators. This emphasizes the third main contribution of our work that is NORA's good accuracy especially in these scenarios.

Nevertheless, some open issues are remaining for future work. NORA could for example be enhanced with support of inline measurements (IPv6), a separation of control and measurement traffic or some kind of visualization of the measurement results. Although all important network characteristics are already supported, NORA could easily be extended to also meet future requirements. Due to the component-based design the measurement methods could also be replaced by other algorithms.

The NORA project is released as open-source software[1] and we encourage other researchers to not only use it for their own experiments, but also to collaborate with us in improving NORA for future releases.

ACKNOWLEDGEMENT

This work results from the NESSEE project that is financed by Citrix Systems, Online Services Division.

REFERENCES

Anagnostakis, K. G., 2006. Flexible network monitoring with flame. *Computer Networks*, vol. 50, no. 14, pp. 2548–2563.

Almes, G. et al, 1999. *A One-way Packet Loss Metric for IPPM*. RFC 2680 (Proposed Standard), IETF.

Crovella, M. And Krishnamurthy, B., 2006. *Internet Measurement: Infrastructure, Traffic and Applications*. John Wiley & Sons, Chichester, England.

Danalis, A. and Dovrolis, C., 2003. Anemos: An autonomous network monitoring system. *In Proc. of 4th Passive and Active Measurements Workshop*.

Dovrolis et al, 2004. Packet-dispersion techniques and a capacity-estimation methodology. *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, pp. 963–977.

Goldoni, E. et al, 2009. Assolo, a new method for available bandwidth estimation. *Fourth International Conference on Internet Monitoring and Protection (ICIMP'09)*, pp. 130–136.

Guerrero, C. D. and Labrador, M. A., 2010. On the applicability of available bandwidth estimation techniques and tools. *Computer Communications*, vol. 33, no. 1, pp. 11–22.

Jain, M. and Dovrolis, C., 2002. Pathload: A measurement tool for end-to-end available bandwidth. *In Proceedings of Passive and Active Measurements (PAM) Workshop*

Jaiswal, S. et al, 2007. Measurement and classification of out-of-sequence packets in a tier-1 ip backbone. *IEEE/ACM Trans. Netw.*, vol. 15, no. 1, pp. 54–66.

Lübke, R. et al, 2014. Measuring Accuracy and Performance of Network Emulators. *Second International Black Sea Conference on Communications and Networking (BlackSeaComm 2014)*

Navratil, J. and Cottrell, R. L., 2003. Abwe: A practical approach to available bandwidth estimation. *In Proceedings of the 4th Passive and Active Measurement Workshop PAM 2003*.

Shalunov, S. et al., 2006. A one-way active measurement protocol (owamp). *Network Working Group-RFC*, vol. 4656.

Sommers, J. et al, 2006. A proposed framework for calibration of available bandwidth estimation tools. *11th IEEE Symposium on Computers and Communications (ISCC'06)*, pp. 709–718.

Song, H. 2009. Netquest: A flexible framework for large-scale network measurement. *IEEE/ACM Trans. Netw.*, vol. 17, no. 1, pp. 106–119.

[1] https://github.com/nora-network