Gaming with the Throughput and the Latency Benchmarking Measurement Procedures of RFC 2544

G. Lencse, Á. Kovács and K. Shima

***Abstract*—In this paper, we investigate three potential issues of the benchmarking measurement procedures defined in RFC 2544 and also used in RFC 5180 and RFC 8219. One of them is the lack of proper timeout usage. We use a Linux box, which can selectively delay a specified ratio of the packets. Using carefully selected parameters based on our preliminary measurements, we demonstrate that the experienced speed of the HTTP download is much less, than what could have been expected on the basis of the throughput results of the RFC 2544 tests. The other critical issue is the strict, absolutely zero loss criterion. We use a Linux box, which drops a specified small ratio of the packets. Whereas the RFC 2544 throughput results tend to be zero, the experienced speed of the HTTP download is quite good. The third problem is the lack of requirement for statistically relevant number of tests in the RFC 2544 benchmarking procedures. We demonstrate its severity with the latency benchmarking procedure defined in RFC 2544 and kept unchanged in RFC 5180 but redefined in RFC 8219.**

***Keywords*—benchmarking, frame loss rate, latency, number of tests, RFC 2554, throughput, timeout.**

I. Introduction

According to our understanding, the aim of benchmarking is to "accurately measure some standardized performance characteristics in order to obtain reasonable and comparable results" [1]. RFC 2544 [2] has been successfully serving this purpose from 1999 by defining a comprehensive benchmarking methodology for network interconnect devices. Its basic measurement procedure is the throughput measurement. In the simplest case, the measurement setup consists of two devices, the Tester and the DUT (Device Under Test). During the throughput measurement, the Tester sends frames through the DUT at a constant rate for at least 60 seconds, and counts the number of received frames. The tester keeps receiving for 2 more seconds after finishing the sending of the frames. If the number of the received frames is equal with the number of the sent frames, then the frame rate is increased, and the test is rerun. If the number of the received frames is less than the number of the sent frames, then the frame rate is decreased, and the test is rerun. The throughput is the highest rate, at which the number of the received frames was equal with the number of the sent frames. (In practice, a binary search is used, where its initial upper limit is the maximum frame rate for the media, and its initial lower limit is zero.) Of course, the measured throughput value depends on the frame size used, thus RFC 2544 defines some standard frame sizes to be used for testing.

RFC 5180 [3] was published in 2008, and it addressed some IPv6 specificities and also defined some further maximum frames rates for the contemporary media types, but it kept the throughput benchmarking procedure unchanged. RFC 5180 has declared IPv6 transition technologies out of its scope. RFC 8219 [4] defined a benchmarking methodology for IPv6 transition technologies in 2017. It has also kept the throughput benchmarking procedure unchanged.

In our most current research [5], we have performed benchmarking measurements of different SIIT [6] (also called stateless NAT64) implementations according to RFC 8219. We have pointed out three problems with the throughput measurement procedure.

1. As no per frame timeout was defined, some frames may arrive with several seconds delay and they are still accepted. (E.g. some frames from the first second may arrive with 61 seconds delay.) These frames are very likely handled as lost ones by TCP implementations and also by real-time UDP applications. Thus, the measured throughput result may be very far from the one experienced by the users.
2. If the DUT has a non-zero but low frame loss rate, then communication is possible with significant throughput, but the absolute zero frame loss requirement of RFC 2544 results in a near zero throughput measurement result.
3. Having no requirement for multiple (statistically relevant number of) tests, the measurement results may be very different and thus unreliable.

Our current effort aims to investigate and deliberately demonstrate them. As for the first two problems, we examine, how much the RFC 2544 throughput results may differ from the throughput experienced in real-life situation. As for the third one, we cannot avoid dealing with it during the throughput measurements, and in addition to that, we demonstrate its seriousness with RFC 2544 latency measurements.

Manuscript received February 12, 2020, revised June 3, 2020.

G. Lencse is with the Department of Telecommunications, Széchenyi István University, Egyetem tér 1, Győr, H-9026, Hungary (phone: +36 96 613-665; fax: +36 96 613-646; e-mail: lencse@sze.hu).

Á. Kovács is with the Dept. of Telecommunications, Széchenyi István University, Egyetem tér 1, Győr, H-9026, Hungary (akos.kovacs@sze.hu).

K. Shima is with IIJ Innovation Institute Inc., Iidabashi Grand Bloom, 2-10-2 Fujimi, Chiyoda-ku, Tokyo, 102-0071, Japan (keiichi@iijlab.net)

The remainder of this paper is organized as follows. In Section II, we clarify benchmarking methodology and terminology. In Section III, we examine how delaying all the packets influences the result of the RFC 2544 throughput test and the file download time using the HTTP protocol. In Section IV, we study how frame loss affects the results of the before mentioned two tests. In Section V, we investigate the effect of the selective delay to the real-life measurements. In section VI, we (further) demonstrate the need for statistically relevant number of experiments by latency tests and we also expose the fact that using average alone as the summarizing function of multiple experiments can be an oversimplification. In section VII, we discuss our findings and disclose our recommendations as well as our plans for further research. Section VIII concludes our paper.

II. Methodology and Terminology

As its title suggests, RFC 2544 defines a benchmarking methodology for network interconnect devices, which can be individual elements of a network, e.g. switches, routers, etc. On the one hand, ISPs (Internet Service Providers) rely on the results of RFC 2544 compliant tests, when they choose the appropriate devices to build up their networks, however, on the other hand, their ultimate concern is to achieve users' satisfaction and cost effectivity at the same time. Service level agreements between ISPs and their customers usually contain QoS (Quality of Service) parameters, like throughput, packet loss rate, transmission delay, etc. The advantage of such parameters is that they can be easily checked, but they do not describe the QoE (Quality of Experience) of the users. RFC 6349 [7] aims to asses user experience by defining a methodology for measuring end-to-end TCP throughput.

Unfortunately, the terminology is not consistent. The term *goodput* is in use for more than two decades. It is mentioned already in 1998 as "real user level good throughput" [8], and it is also called as "user seen throughput" in the same paper. TCP goodput was also called as "useful throughput of TCP" [9]. Gootput is also called as "effective throughput" [10]. Another definition for goodput is "TCP throughput at the end nodes" [11]. It is also called as "application throughput" [12]. However, IETF documents including RFC 6349 usually omit the term. In the rest of our paper, we simply call it "user experience".

RFC 2544 measurements should be performed in an isolated environment containing only the Tester and the DUT, whereas RFC 6349 measurements can be performed only by using the entire communication system. (For example, a web server, a client and the network between them, which has load from other users, too.) Being aware of this significant difference between the two, we contend that RFC 2544 benchmarking results are useful for an ISP if they give a kind of warranty that if there are no other bottlenecks in the system, the given device will not deteriorate user experience. Moreover, we aim to point out some possible opportunities for amending RFC 2544 with further or modified tests, the results of which are more in line with the user experience.

In the rest of this paper, we demonstrate situations, when the results of RFC 2544 benchmarking tests are not in line with the user experience.

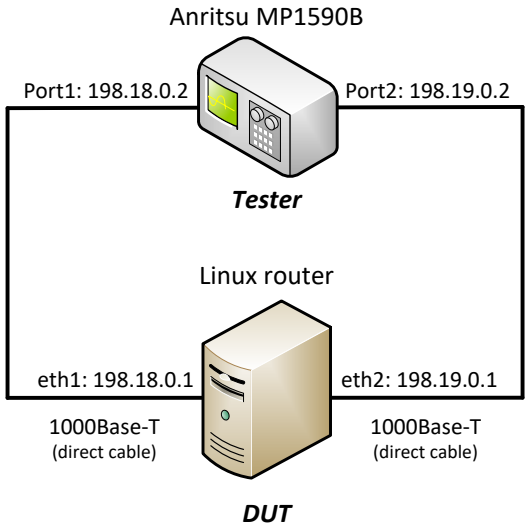

Fig. 1 Measurement setup for RFC 2544 throughput and latency tests

III. Investigation of the Effect of Delay

Delay really matters in real-life situation, but according to our understanding of the RFC 2544 *throughput*[1] benchmarking procedure, it is overly delay tolerant. To check, if our interpretations is correct, first, we performed throughput tests with a commercial RFC 2544 tester.

Then, we have examined, how the increase of delay, which is completely invisible for the RFC 2544 tester, can degrade the throughput of a communication system in real-life situations.

As we mentioned before, RFC 2544 does not state how many times the experiments have to be executed to achieve reliable throughput results. It only mentions (in its Section 26.2) that the Latency measurements should be repeated at least 20 times and the reported values should be the average of the recorded values.

We believe that the necessary number of the repetitions deserves a careful study for *all* benchmarking procedures, and we also plan to do so. Currently, we have chosen to execute all tests 20 times.

A. RFC 2544 Compliant Measurements

We used the measurement system shown in Fig. 1. Our Tester was a commercially available, RFC 2544 compliant Anritsu MP1590B Network Performance Tester. It had a four port Anritsu MU210212A 10/100/1000M Ethernet Module, and we used Port1 and Port2 of the module.

The DUT was a contemporary PC from the Info-communications Laboratory of the Department of Telecommunications, Széchenyi István University, Győr, Hungary with the following parameters: Gigabyte H310M H mother board, 6 core Intel Core i5-8400 2.80GHz CPU, two Kingmax DDR4 2133GHz 8GB memory modules, two Intel OEM Gigabit CT Desktop Adapters (for experimentation), Kingston SA400S3 120GB SSD. Debian Linux 10.2 operating system with 4.19.0-6-amd64 kernel was installed.

[1] The same applies to the *frame loss rate* measurement procedure, too.

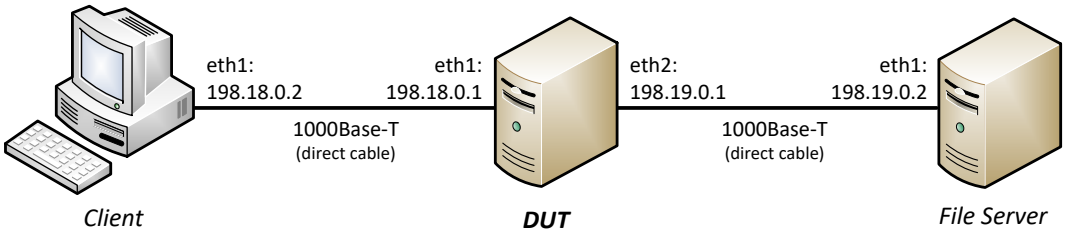

Fig. 2 Test network for our real-life investigations

As otherwise the CPU clock frequency was varying, which could have caused unstable measurement results according to our earlier benchmarking experience [13], we have set it to fixed 2.8GHz by using the **`tlp`** Debian package. The **`/etc/default/tlp`** file contained the following lines:

```
CPU_SCALING_MIN_FREQ_ON_AC=2800000
CPU_SCALING_MAX_FREQ_ON_AC=2800000
```

We note that the above frequency values are interpreted in kHz unit.

Although RFC 2544 recommends using several standard frame sizes from 64 bytes to 1518 bytes, we decided to use only 1518 bytes, because our aim was not the complete performance analysis of a given router, but rather the *investigation of certain special problems*[2], and the 1518 bytes frame size was the natural choice for the file transfer applications we used for real-life testing.

For delaying the IP datagrams, we used the **`tc`** command of the **`iproute2`** Debian package. We used the following two command lines to delay all packets by 1s:

```
tc qdisc add dev eth1 root netem \
  delay 1s limit 100000
tc qdisc add dev eth2 root netem \
  delay 1s limit 100000
```

Remarks:

- The specified delay is applied only to the outgoing packets through the specified interface, this is why we used two commands.
- We did not intend to limit the packet rate at all, but we had to set it to some appropriate value, because otherwise it was automatically set to 1000. (The used 100,000 value was chosen to be greater than 81,274, the maximum frame rate for Gigabit Ethernet using 1518bytes long frames[3].)

As required by RFC 2544, we used bidirectional traffic and the duration of the tests was 60s.

To be able to disclose all the details of the measurements, we must mention some properties of the Anritsu tester. Although throughput is usually specified in frames per second (fps), the user of the Anritsu tester has to specify the required frame rates in a certain proportion of the maximum frame rate for the media at the given frame size.

The user can set the following values:

- *Initial Rate*: frame rate for the very first test. If it is set to 100% and the test is passed, then no binary search is executed.
- *Maximum Rate*: The initial upper bound for the binary search.
- *Minimum Rate*: The initial lower bound for the binary search.
- *Resolution*: The binary search is finished, if the difference between the upper bound and lower bound of the binary search is less than or equal with this value.
- *Loss Tolerance*: The Tester allows the user to specify some tolerated loss rate. It must be set to 0 to perform an RFC 2544 compliant test. (We always did so, unless we stated otherwise.)

First, we performed the throughput test without delaying the test frames by the DUT. The initial frame rate was set to 100% in the Tester and the tests were passed all 20 times. The reported frame rate was 162,548fps, which is exactly the double of the maximum frame rate of Gigabit Ethernet. (The Tester reports the number of all transmitted frames, not frames per direction.)

Then the tests were repeated by applying the above mentioned **`tc`** commands to delay all the test frames in both directions. As expected, we have received the same results, 162,548fps for all 20 times.

B. Measurements in a Real-life Situation

We used the measurement system shown in Fig. 2. All three computers had the same hardware as before, except that Client and File Server had only a single Intel OEM Gigabit CT Desktop Adapter for experimentation, each. (For controlling the experiments, the integrated NICs were used.)

We wanted to experiment in a realistic environment and share the experience of an average user, thus we did not do any tuning on the TCP settings of the Linux kernel, but left them as they were. For the repeatability of our experiments, we have checked and document the most important settings as follows. The TCP congestion algorithm was set to CUBIC, SACK (Selective Acknowledgment) and window scaling were enabled on both the Client and the File Server.

For benchmarking, we downloaded a 10,000MiB size file using the HTTP protocol. The file size was chosen large enough so that its downloading through the Gigabit Ethernet link last more than one minute, and thus the initial transient (TCP connection establishment, and "slow start" phase of the congestion control protocol) may be amortized and the

[2] In section III.A, we investigate, how the delay of the packets effects the RFC 2544 results, and in Section III.B, we examine, how it effects the real life user experience. In the upcoming sections, we study different problems, but we use the same measurement setup with some minor changes.

[3] Please refer to Appendix A.1 of RFC 5180 for the theoretical maximum frame rates for different media.

```
#!/bin/bash
for D in 0ms 10ms 100ms 1000ms
do
  ssh root@dut "tc qdisc add dev eth1 root netem limit 100000 delay $D"
  for i in {1..20}
  do
    /usr/bin/time -f "%E" -o wget_$D -a wget 198.19.0.2/test.img \
      -o wget_log_${D}_$i.log -O /dev/null
  done
  ssh root@dut "tc qdisc del dev eth1 root netem limit 100000 delay $D"
done
```

Fig. 3 Measurement script for benchmarking with HTTP download

downloading process be dominated by the steady state behavior of the system. The **`wget`** Linux command was used for downloading, and its execution time was measured by the GNU **`time`** command. First, we used no delay to have a reference. Then 10ms, 100ms and 1000ms delay was set at the DUT, but only on its **`eth1`** interface, which influenced only the outgoing packets of the interface. It was done so, because the message flow in the two directions were independent during the RFC 2544 test, but in the current test the frames carrying the data packets from the File Server to the Client and the frames carrying the acknowledgements from the Client to the File Server belong to the same TCP session and applying the delay also for the outgoing packets through the **`eth2`** interface would have resulted in a double measure delay for the entire communication. For the easy replication of our experiments, we disclose our measurement script in Fig. 3. As it can be seen from the script, all experiments were executed 20 times. The results are shown in Table I. Whereas the 10ms delay caused only a hardly noticeable increase in the download time, the 100ms delay resulted in a more than 4 times increase and *the 1s delay caused a more than 40 times increase of the download time*.

Therefore, we can lay down that whereas the RFC 2544 throughput test failed to point out any difference between the two routers (the one without delay and the one with a 1s delay seemed to operate at full frame rate of the Gigabit Ethernet) the user of the two routers experiences a radical difference.

IV. Investigation of the Effect of Frame Loss

Frames loss is present in our networks from the very beginning, and TCP can handle some low to moderate frame loss rates like 0.01% or even 0.1% quite well. The RFC 2544 throughput benchmarking procedure tolerates no frame loss. Even if only a single frame is lost during the 60s measurement interval, the test is qualified as failed.

TABLE I
Download Time of a 10,000MiB Size File as a Function of the Delay at the DUT (Lower is Better)

Delay at the DUT	0ms	10ms	100ms	1000ms
median (s)	89.11	89.19	377.5	3708
1st percentile (s)	89.10	89.18	376.9	3695
99th percentile (s)	89.12	89.36	378.8	3710

Commercial RFC 2544 Testers usually have an option that allows the user to set some tolerated frame loss rate.

A. RFC 2544 Compliant Measurements

For experimenting with RFC 2544 compliant measurements, we used the test system shown in Fig. 1. The only deviation of the DUT settings from that of in Section III.A was that now we used the following two command lines to achieve 0.01% frame loss in each directions:

```
tc qdisc add dev eth1 root netem \
  loss 0.01% limit 100000
tc qdisc add dev eth2 root netem \
  loss 0.01% limit 100000
```

We have set the values of both the “Initial Rate” and the “Maximum Rate” parameters to 1%. It actually meant 812.74fps, which was more than high enough considering that at this frame rate 48,764 frames can be transmitted during the required 60s testing time, and the 0.01% frame loss means that 1 frame is lost from every 10,000 frames on average, but of course, the dropping of the frames is pseudo random. (It means that the first two steps of the binary search are expected to fail, and later steps should sometimes succeed sometimes fail due to the loss of a single frame from every 10,000 frames on average. It also means that the results are expected to be rather scattered.) For “Minimum Rate” and “Resolution”, we set 0.01%, because it was the lowest possible value to be set. The “Loss Tolerance” parameter was set to 0, to achieve RFC 2544 compliant measurements. The tests were executed 20 times. The results reported by the Tester are shown in Table II. As expected, the results of the 20 measurements are very much scattered. The extreme values are “-” and 0.50. We have checked the meaning of “-”, the result of the 8-th test, in the measurement log. The Tester first executed a test with 1% throughput and it failed. Then it performed a test with 0.01% throughput (Minimum Rate) and it also failed, therefore it did not perform a binary search in the [Minimum Rate, Maximum Rate] interval. Thus “-” means “less than 0.01% throughput”. The other extreme value is 0.5% throughput, which has also happened only once. (We consider it a rare event, as 0.25% occurred also only once.)

We contend that here it is inappropriate to calculate that the average of the 20 results is 0.12% of the maximum frame rate, because *it oversimplifies the results*. We believe that it is desirable to state the extreme values to reflect the rather

TABLE II
REPORTED RESULTS OF THE RFC 2544 COMPLIANT THROUGHPUT TEST, WHEN 0.01% OF THE FRAME WERE DROPPED BY THE DUT
(THE VALUES ARE GIVEN AS PERCENTAGES OF THE MAXIMUM FRAME RATE FOR GIGABIT ETHERNET)

Frame Size	Average	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1,518	0.12	0.10	0.13	0.06	0.04	0.13	0.13	0.06	-	0.25	0.14	0.07	0.13	0.07	0.08	0.05	0.50	0.14	0.04	0.08	0.13

TABLE III
DOWNLOAD TIME OF A 10,000MIB SIZE FILE AS A FUNCTION OF THE FRAME LOSS RATE AT THE DUT (LOWER IS BETTER)

Frame loss rate	0%	0.01%	0.1%	1%
median (s)	89.11	89.14	89.44	188.7
1st percentile (s)	89.10	89.11	89.34	180.6
99th percentile (s)	89.13	89.15	89.56	197.5

scattered nature of the results. Anyway, *the results are extremely low*, we can state that the throughput is close to zero in the case of 0.01% frame loss.

We have not performed more RFC 2544 compliant throughput test with higher loss rates (e.g. 0.1%, etc.) as it was deliberate that they would result in practically zero throughput, which we could not reliably measure due to the allowed lowest values of the "Minimum Rate" and "Resolution" parameters of the Tester.

As we have mentioned before, commercial Testers allow the user to specify some tolerated loss rate. The Anritsu Tester calls it as "Loss Tolerance". We have performed an experiment, when it was set to 0.011%. (It was chosen to be somewhat higher than 0.01%, the frame drop rate set at the DUT.) This time, the results were totally different: 100% throughput was reported all 20 times. Please note that this measurement did not comply with the requirements of RFC 2544!

B. *Measurements in a Real-life Situation*

We used the test system shown in Fig. 2, and the measurement script was also very similar to the one in Fig. 3, just the keyword "delay" was replaced by "loss" and the following loss rates were used: 0%, 0.01%, 0.1%, 1%. As before, we applied the frame loss only for the outgoing traffic through the **`eth1`** interface, that is, the packet flow from the client to the server was lossless.

Results are shown in Table III. The distribution of the results of the measurements using 0.01% frame loss overlaps with the distribution of the results of the lossless measurements. The 0.33s increase of the median download time caused by the 0.1% frame loss is still unnoticeable for the user. And even though 1% frame loss has doubled the download time, communication is still possible with an acceptable speed (unlike in the case of 1s delay).

Therefore, we can lay down that whereas the RFC 2544 throughput tests showed nearly zero throughput even with 0.01% frame drop rate, the users experience either no difference (with 0.01% or 0.1% frame drop rate) or an acceptable communication speed (with 1% frame drop rate) in real life.

V. INVESTIGATION OF THE EFFECT OF SELECTIVE DELAY

We found it an interesting question, what happens, when only a small fraction of the frames is delayed. In Section III, we have identified 100ms and 1000ms as relevant delay

```
tc qdisc add dev eth1 handle 1: root htb
tc class add dev eth1 parent 1: classid 1:11 htb rate 1000Mbps
tc qdisc add dev eth1 parent 1:11 handle 11: netem delay 100ms # or 1000ms
tc filter add dev eth1 parent 1:0 prio 1 protocol ip handle 11 fw flowid 1:11
```

Fig. 4 DUT settings for delaying a certain proportion of the frames, which have been marked before (please see the **`iptables`** rule in Fig. 5)

```
#!/bin/bash
for N in 1000 100
do
  ssh root@dut "iptables -t mangle -A FORWARD -m statistic --mode nth --every $N \
    --packet 0 -j MARK --set-mark 11"
  for i in {1..20}
  do
    /usr/bin/time -f "%E" -o wget_$N -a wget http:// 198.19.0.2/test.img \
      -o wget_log_${N}_$i.log -O /dev/null
  done
  ssh root@dut "iptables -t mangle -D FORWARD -m statistic --mode nth --every $N \
    --packet 0 -j MARK --set-mark 11"
done
```

Fig. 5 Measurement script for benchmarking with HTTP download

TABLE IV
DOWNLOAD TIME OF A 10,000MIB SIZE FILE AS A FUNCTION OF BOTH THE LATENCY AND THE PROPORTION OF THE LATE FRAMES (LOWER IS BETTER)

Latency	100ms		1000ms	
Proportion	0.1%	1%	0.1%	1%
median (s)	89.26	218.1	89.26	218.0
1st percentile (s)	89.23	214.2	89.24	212.8
99th percentile (s)	89.30	221.4	89.64	221.1

values, and we used them in this experiment. As for the proportion of the delayed frames, we used 0.1% and 1%, which were identified in Section IV as relevant frame loss values. We used their possible 2x2=4 combinations as parameters for the measurements with selectively delayed frames.

To achieve the selective delaying of 0.1% or 1% of the frames, we used the combination of two tools. The statistics module of **`iptables`** was used to mark every 1000th or every 100th frames, and then 100ms or 1000ms delay was applied for the marked frames. The commands in Fig. 4 were issued at the DUT. The bash shell script in Fig 5 was executed by the client computer.

We note that this time we did not perform any RFC 2544 compliant measurements, because we have already seen that delay makes no effect and the applied frame drop rates result in a practically zero throughput.

We performed the previously used file download tests and the results are shown in Table IV. It is well visible that the measure of the delay (100ms or 1000ms) made no difference in the download time. Its explanation can be that TCP congestion control handled the late segments as lost ones.

However, the late TCP segments could be even worse, than lost ones. Let us compare the last columns of Table III and Table IV. If 1% of the frames was really lost, then the median of the download time was 188.7s, whereas if 1% of the frames suffered 1s delay, then the median of the download time was 218s. Thus we can see that it is worse if 1% of the frames arrives 1000ms late, than if they are completely lost. We surmise that those TCP segments that were first considered as lost and then arrived late could mislead the congestion control algorithm and thus the situation resulted in a higher increase of the download time than in the case, when the same amount of frames were dropped.

VI. LATENCY MEASUREMENTS

To demonstrate the need for a statistically relevant number of tests, we used a single directional test and applied 1s delay for every 10th frame using the script shown in Fig. 6.

We have performed two RFC 2544 compliant Latency tests according to the test setup shown in Fig. 1. As it is required by RFC 2544, first, we determined the throughput by a 60s long measurement and then we performed the 120s long Latency tests. Of course, the throughput was found to be 100% in both cases. The only difference between the two tests was the number of repetitions: 20 and 50. RFC 2544 states that the final result is the average of the measured latency results of the (at least 20) measurements. The final results of the two tests were: 0.000270863s and 0.200289483s, which are very much different. Its reason is very simple. RFC 2544 Latency test measures only the latency of a single tagged[4] frame sent after 60s. In the first test, none of the 20 tagged frames was delayed. In the second test, exactly 10 from the 50 tagged frames was delayed. (We have counted them in the measurement log file.) Of course, other measurements could give further different results. However, these two tests are enough to demonstrate the following three things:

1. Statistically relevant number of tests are needed.
2. The Latency measurement procedure of RFC 2544 is rather questionable.
3. Using only the average as summarizing function of the results is a serious oversimplification.

We note that whereas RFC 8219 has kept the Throughput and Frame Loss Rate measurement procedures, it redefined the Latency measurement procedure, which addresses all three problems. (Please see Section VII for the details)

VII. DISCUSSION, RECOMMENDATIONS AND FUTURE WORK

We believe that the results of the different benchmarking measurements procedures should be in agreement with the experience of the users. Let us consider, what can be done in this regard.

As for updating the throughput measurement procedure, we have already recommended the usage of *frame timeout* in [5], but then we did not have an appropriate tester yet. Since then we have implemented **`siitperf`**, an RFC 8219 compliant SIIT (also called stateless NAT64) tester, which was released as a free software under the GPLv3 license, and it is available from GitHub [14]. Although its primary purpose is the benchmarking of SIIT gateways, it may also be used for benchmarking IPv4 or IPv6 routers. It has a part called **`siitperf-pdv`**, which implements packet delay variation measurements. It numbers every single test frame, records their sending and receiving times, and then calculates their delays. This program accepts an optional parameter called *frame timeout*. If this parameter is present and its value is positive, then the program reclassifies all frames having higher latency than frame timeout as *lost*, when it reports the number of received frames [15]. Thus, *it is now feasible to carry out throughput benchmarking measurements by checking the timeout for every single frame individually*. As for the proposed value for timeout, we have seen in Section V of the current paper that frames with 100ms latency were considered as lost, thus we propose that *the required frame timeout value should be in the order of a few milliseconds*. Of course, its actual value is still subject of research.

As for the absolutely zero frame loss rate criterion, we think that it has been already long relieved in benchmarking practice. For example, Kevin Tolly mentioned 0.001% as his "zero loss criterion" in 2001. And we have seen that the Anritsu tester also allowed the user to specify his own "Loss Tolerance" rate. What we consider important, is that *the applied loss tolerance rate must be well visibly disclosed*

[4] A frame is marked by an implementation dependent way so that the Tester may recognize it, when it arrives back through the DUT.

```
tc qdisc add dev eth1 handle 1: root htb
tc class add dev eth1 parent 1: classid 1:11 htb rate 1000Mbps
tc qdisc add dev eth1 parent 1:11 handle 11: netem delay 1000ms limit 100000
tc filter add dev eth1 parent 1:0 prio 1 protocol ip handle 11 fw flowid 1:11
iptables -t mangle -A FORWARD -m statistic --mode nth --every 10 --packet 0 \
  -j MARK --set-mark 11
```

Fig. 6 DUT settings for delaying every 10th frame by 1s

together with the results.

As for the Latency measurement procedure of RFC 2544, it was redefined by RFC 8219. The new procedure requires to tag at least 500 frames for latency measurement, which addresses the first two problems we mentioned in Section VI. It also uses different summarizing functions instead of average: *Typical Latency* is the median of the measured latencies, and *Worst Case Latency* is the 99.9th percentile of the measured latencies. Thus, the issue of oversimplification is also addressed. The only problem is that the scope of RFC 8219 is IPv6 transition technologies, and it does not update RFC 2544 for benchmarking network interconnect devices in general.

According to our understanding, the worst problem is the lack of requirement for statistically relevant number of tests. Whereas we used Latency measurements for an easy demonstration of the problem, it still persists in the case of Throughput and Frame Loss Rate tests, which were kept unchanged in RFC 8219. As we have mentioned in [5], we plan to develop and algorithm which can decide during the measurements, if more repetitions are needed or not.

We have published the "00" version of an Internet Draft [16], which recommends to update RFC 2544 as we described above. All Internet Drafts are to be referred to as "work in progress", and our one should definitely be considered only as a thought provoking writing and it will require a lot of work to precisely define new or modified benchmarking procedures.

VIII. Conclusion

We have investigated and demonstrated three issues of the RFC 2544 benchmarking procedures.

We have shown that the 2 seconds global timeout of the throughput measurement procedure is inappropriate, and recommended the usage of per frame timeout, which means the checking of the latency of every single frame. We have also shown that it is now possible to carry out using our free software tool **`siitperf-pdv`**.

We have demonstrated that the absolutely zero loss criterion of RFC 2544 is too strict and it may lead to very different throughput results than experienced by the user in real life. We proposed to "canonize" the wide spread practice of using a non-zero loss criterion together with the mandatory indication of its value, when disclosing the results.

We have also exposed the lack of requirement for statistically relevant number of tests. As a future solution, we plan to develop an algorithm, which can decide during the measurements, if more repetitions are needed or not.

Acknowledgment

The authors thank the National Media and Info-communications Authority (NMHH) of Hungary for landing us the Anritsu MP1590B Network Performance Tester.

The authors thank István Pilisi, NMHH, for reading and commenting the manuscript.

References

[1] G. Lencse, "Benchmarking methodology for IPv6 transition technologies", *IIJ Lab seminar*, Tokyo, Oct. 10, 2017, [Online]. Available: https://seminar-materials.iijlab.net/iijlab-seminar/iijlab-seminar-20171010.pdf

[2] S. Bradner and J. McQuaid, "Benchmarking methodology for network interconnect devices", *IETF RFC 2544*, 1999. DOI: 10.17487/RFC2544

[3] C. Popoviciu, A. Hamza, G. Van de Velde, and D. Dugatkin, "IPv6 benchmarking methodology for network interconnect devices", *IETF RFC 5180*, 2008. DOI: 10.17487/RFC5180

[4] M. Georgescu, L. Pislaru L, and G. Lencse, "Benchmarking methodology for IPv6 transition technologies, *IETF RFC 8219*, 2017. DOI: 10.17487/RFC8219

[5] G. Lencse, K. Shima, "Performance analysis of SIIT implementations: Testing and improving the methodology", *Computer Communications*, vol. 156, no. 1, pp. 54-67, April 15, 2020, DOI: 10.1016/j.comcom.2020.03.034

[6] C. Bao, X. Li, F. Baker, T. Anderson, and F. Gont, "IP/ICMP translation algorithm", *IETF RFC 7915*, 2016. DOI: 10.17487/RFC7915

[7] B. Constantine, G. Forget, R. Geib, R. Schrage, "Framework for TCP Throughput Testing", *IETF RFC 6349*, 2011. DOI: 10.17487/RFC6349

[8] P. Lettieri and M. B. Srivastava, "Adaptive frame length control for improving wireless link throughput, range, and energy efficiency" *Proc. IEEE INFOCOM '98*, San Francisco, CA, USA, 1998. DOI: 10.1109/INFCOM.1998.665076

[9] Y. Bai, A.T. Ogielski, G. Wu, "Interactions of TCP and radio link ARQ protocol" *Proc. IEEE VTS 50th Vehicular Technology Conference*, Amsterdam, The Netherlands, 1999. DOI: 10.1109/VETECF.1999.801596

[10] V. Tsaoussidis, H. Badr, X. Ge, K. Pentikousis, "Energy/throughput tradeoffs of TCP error control strategies", *Proc. ISCC 2000*, Antibes-Juan Les Pins, France, 2000. DOI: 10.1109/ISCC.2000.860618

[11] N. Celandroni, F. Potorti, "Maximizing single connection TCP goodput by trading bandwidth for BER", *International Journal of Communication Systems*, vol. 16, no. 1, pp. 63-79, February 18, 2003. DOI: 10.1002/dac.580

[12] Peng Zhang ; Hongbo Wang ; Shiduan Cheng, "Shrinking MTU to Mitigate TCP Incast Throughput Collapse in Data Center Networks", *Proc. 2011 Third International Conference on Communications and Mobile Computing*, Qingdao, China, 2011. DOI: 10.1109/CMC.2011.68

[13] G. Lencse and Y. Kadobayashi, "Benchmarking DNS64 implementations: Theory and practice", *Computer Communications*, vol. 127, no. 1, pp. 61-74, September 1, 2018. DOI: 10.1016/j.comcom.2018.05.005

[14] G. Lencse, "Siitperf: an RFC 8219 compliant SIIT (stateless NAT64) tester written in C++ using DPDK", source code, [Online]. Available: https://github.com/lencsegabor/siitperf

[15] G. Lencse, "Design and implementation of a software tester for benchmarking stateless NAT64 gateways", accepted for publication

in *IEICE Transactions on Communications*, revised version is available: http://www.hit.bme.hu/~lencse/publications/IEICE-2020-siitperf-revised.pdf

[16] G. Lencse, K. Shima, "An upgrade to benchmarking methodology for network interconnect devices", individual Internet Draft, May 20, 2020. available: https://tools.ietf.org/html/draft-lencse-bmwg-rfc2544-bis-00

Gábor Lencse received his MSc and PhD in computer science from the Budapest University of Technology and Economics, Budapest, Hungary in 1994 and 2001, respectively.

He has been working full time for the Department of Telecommunications, Széchenyi István University, Győr, Hungary since 1997. Now, he is a Professor. He has been working part time for the Department of Networked Systems and Services, Budapest University of Technology and Economics as a Senior Research Fellow since 2005. His research interests include the performance and security analysis of IPv6 transition technologies.

Ákos Kovács received his BSc and MSc in Electrical Engineering from the Széchenyi István University, Győr, Hungary in 2008 and 2013, respectively.

He has been working full time for the Department of Telecommunications, Széchenyi István University, Győr, Hungary since 2008 as Laboratory Engineer. Now he works as an Assistant Lecturer. He is also a PhD student, and his research field includes the Cloud infrastructure and performance analysis of Multipath communication techniques.

Keiichi Shima is a deputy director at the Research Laboratory of IIJ Innovation Institute, Inc.

His research field is the Internet, including designing and implementing communication protocols, computer networking technologies, computer network security, AI-based anomaly detection, and so forth. He also works as a board member of the WIDE project operating a nation wide research network in Japan.