In-Lab Replay Testing under Real Traffic with a Case Study on WLAN Routers

Ying-Dar Lin[1], Ren-Hung Hwang[2], Chun-Nan Lu[1], Jui-Tsun Hung[1]
[1]Department of Computer Science, National Chiao Tung University, Taiwan
[2]Department of Computer Science and Information Engineering, National Chung Cheng University, Taiwan
ydlin@cs.nctu.edu.tw, rhhwang@cs.ccu.edu.tw, {cnlu, jhung}@cs.nctu.edu.tw

Abstract

Replaying artificial or real-world traffic is a method used to test networking devices. Using real-world traffic is desirable as it uncovers more realistic properties such as traffic diversity and complicated user behaviors. In this paper, we propose an in-lab replay testing (ILRT) framework, composed of a monitor, a traffic replayer and a library of real-world traffic traces, to replay captured packet traces to test networking devices. To demonstrate its effectiveness, a total of 28 WLAN routers were tested to evaluate whether they could work stably for an extended time. In our experiments, 53.57% and 100% of devices under test (DUTs) were triggered critical (L1) failures and tolerable (L2) failures, respectively. 21.43% and 100% of DUTs were triggered more than one L1 and L2 failures, respectively. Among high-spec DUTs, there was 25% of DUTs which were triggered L1 failures. Furthermore, among 458 test results, chance of pass, L1 failure, and L2 failure is 2.84%, 7.86%, and 89.3%, respectively. The results showed that even though these devices have passed traditional lab tests, there is still a good chance for them to fail under real-world traffic.

Keywords: Traffic capture, Traffic replay, Stability testing.

1 Introduction

Testing is a method used to discover defects in the development stage of a product, and aims at reducing the number of after-sale failures found by customers [1]. Even though they pass a series of tests during their development processes, networking devices may still have customer found defects (CFDs) when they are used under real-world conditions. This could be explained by the fact that laboratory testing often uses artificial traces or traces containing simple activities which have less realistic properties, and are less diverse and complex in terms of applications and protocols [2]. For instance, applications such as peer-to-peer (P2P), video streaming services, and on-line games often have proprietary protocols or diverse behaviors. The action of an exploit or a malware may be partitioned into several stages to try to shield itself from detection. All of these behaviors are hard to mimic under artificial conditions. Because there are more unexpected usages in real-world user behaviors, replaying real-world traffic usually can trigger more product failures not found by stimulating artificial traffic.

However, replaying real-world traffic is non-trivial. A primary challenge in traffic replaying is that the activity intended for the original host would likely not be accepted by another without modification. For example, the activity contained may include or depend on states specific to the original host, such as its network address, transport layer port number, etc. In such cases, a straightforward traffic replay to a different host with a state that is incompatible with that of the new host would likely fail. Therefore, the state-dependent fields must be altered to correspond to the different replay scenarios and participants for replay to succeed.

In this work, we propose an in-lab replay testing (ILRT) framework to test networking devices, and implement a stateful traffic replayer, NATreplay, to test devices which support Network Address Translation (NAT) mechanism. ILRT consists of three different components, a *monitor*, a *traffic replayer* and a *library* of captured traffic traces. The monitor, called CheckDev, is a probing tool that regularly sends ARP, ICMP and HTTP probes to the DUT and monitors its responses. The traffic replayer is responsible for parsing the traces, separating the traces into two groups and sending the traffic to the DUT from different interfaces of the replayer. The traffic replayer is dedicated to the DUT and can be replaced for different test plans and criteria. The library of packet traces are captured and updated from hundreds of volunteers' daily usage of all kinds of network applications, which contains numerous categories and activities.

We capture real-world network traffic generated by the operations of hundreds of volunteers without interruption and then use the captured traffic traces to test networking devices. Each DUT is tested and monitored by a dedicated replayer and CheckDev with the distance of one-hop separately. Therefore, every time a failure occurs, we can ensure that its happening is because the DUT fails. In this manner, we can avoid the failures caused by connectivity. Besides, a CheckDev can monitor multiple replayers at a time and dynamically adjust replay throughput based on the feedback of DUTs. When the test is completed, CheckDev

*Corresponding author: Chun-Nan Lu; E-mail: cnlu@cs.nctu.edu.tw
DOI: 10.6138/JIT.2017.18.2.20130924

can sort out the reports based on the probe results during replay. Compared to other traffic replayer which can only be used to replay traffic, our framework is definitely the better choice.

To demonstrate the appropriateness and the effectiveness of the ILRT framework, 28 WLAN routers were used as the DUTs and were evaluated whether they could work stably for an extended time. The triggered failures were classified into two groups, *critical* failures (L1) and *tolerable* failures (L2), based on the severity of damage. For a networking device, it is unacceptable if triggered failures disrupts its ability to access the Internet, a failure which would be definitely be identified as a critical-level failure; if the triggered failures only degrades the performance or slows down the throughput but is still be able to access the Internet, they would be identified as tolerable failures.

This paper is organized as follows. In the next section, some important related literature is surveyed. Section 3 describes terminologies and discusses related issues. Our proposed algorithm is introduced in Section 4 and their performance is evaluated in Session 5. Finally, conclusions are given in Session 6.

2 Related Works

2.1 Traffic Capture

Testing a networking device with real-world traffic requires the capture of traces in a controlled environment. The quality of captured traces is a determining factor of the accuracy and efficiency of tests performed on products or experimental studies [3]. There is a beta site testing environment at National Chiao Tung University campus where the network traffic generated by volunteers can be captured [1]. This environment, called Beta Site, evaluates product performance with real-world traffic. More than 800 students are invited as volunteers and their daily traffic is captured in files in PCAP format. There are usually hundreds of types of protocols and applications in the traffic, including audio/video streaming, peer-to-peer applications, mail protocols, etc. Beta Site provides a good way to evaluate DUTs in an in-line live mode or an off-line replay mode.

2.2 Traffic Replay

There have been several studies which address the problem of network traffic replay. Cui et al. [4] proposed a heuristic-based mechanism to identify and update related fields in an application session. They decoupled application semantics, identified state-specific fields and updated them, based on the semantics of each field type. The decoupling required the identification of application-dependent fields which had to be modified for successful replay. Accuracy was not guaranteed and new heuristics had to be developed to handle new related fields. Newsome et al. [5] focused on application layer and proposed a static analysis approach that takes a binary program as input and outputs the set of inputs that the program accepts. In addition to the fact that it was not able to statistically determine the complete set of inputs of a program in advance, this approach also suffered from scalability issues. Therefore, this method was only able to use the very simple protocol and small prototype applications that they developed.

A number of projects developed tools or plug-ins to solve the problem of traffic replay. For example, Tcpreplay [6] intuitively replays traffic captured and stored in a PCAP file format, based on the timestamps of each packet. Because the purpose of Tcpreplay is to send the captured traffic back to the test network, it can’t respond to services running on a DUT. Therefore, Tcpreplay can’t be used to test stateful networking devices because it doesn’t timely update the states of connection during traffic replay.

SocketReplay [3] supports stateful replay in the network and transport layers because many DUTs, including NAT devices, proxies and security appliance, and may modify TCP/IP headers. SocketReplay could update the response states to prevent from replaying blocked connections.

Some tools can maintain the states of the network and transport layers for traffic replay. TCPOpera [7] uses four heuristics to follow the TCP/IP stack. Monkey [8] replays web application traffic by emulating the TCP stack to reproduce network conditions. Monkey infers delays caused by the client, the applications, the server, and the network in each captured flow and replays each flow according to them. Although [4, 5] support the states of the application layer, they still failed on replaying traffic captured from a large network.

In this work, we develop a stateful traffic replayer, NATreplay, which supports stateful traffic replay for NAT devices because it is able to identify and update the mapping of the address translation between the old and the new IP addresses. By timely updating related states during replay process, the captured application activities can be reproduced and the test configurations can be repeatedly used to test products.

TCPreplay is a commonly used traffic replayer. However, compared to our framework, TCPreplay has two insufficiencies. First, TCPreplay simply replays the packets of the captured traces in the order of the packet timestamps at a pre-specified rate and can’t dynamically adjust replay throughput based on the feedback of the DUT, which may cause many subsequent packets being dropped. Secondly, the connection states of the replayed traffic by TCPreplay does not necessarily conform to the TCP protocol (e.g., TCPreplay is unable to synchronize SYN-ACKs to create valid TCP sessions), which is harmful to replay test. Therefore, our framework has better flexibility.

3 In-Lab Replay Issues

Successfully replaying an application activity may require modifying parts of traffic contents specific to the original host or scenario. However, blindly changing the trace may result in an inconsistent activity, e.g., network addresses and checksums on packets may be incorrect. Therefore, how to identify which parts of fields to change, what value to modify, and subsequently making the state consistent in an automated fashion is not trivial and is difficult.

3.1 Effectiveness of Replay

The replay accuracy affects the effectiveness of testing a DUT and reproduction of potential events. To effectively replay the traffic traces recognized as valid traffic by a DUT, the replay must follow the state machines of applications/protocols and send out the corresponding packets in the correct direction and order to the DUT. Besides, the replay may consistently modify the outgoing packets in response to fit the test scenario at that time. Otherwise, the packets may be dropped or discarded by the DUT, which makes the replay failed.

Each DUT is tested and monitored by a dedicated replayer and CheckDev with the distance of one-hop separately. Therefore, every time a failure occurs, we can ensure that its happening is because the DUT fails. In this manner, we can avoid the failures caused by connectivity. Therefore, the number and topology of DUTs would not influence the overall performance and effectiveness.

3.2 Efficiency of Replay

The efficiency affects the time spent during the traffic replay and the effort required in result analysis. The efficiency of replay relies on two things, one is the volume of replayed traffic and the other is the time required to reproduce an event. For event analysis and debugging, the lower the volume of the traffic traces required to reproduce an event, the quicker the testers and the developers can analyze and solve them. However, reducing the volume of replayed traffic might unintentionally drop critical events of interest, so there are usually tradeoffs between the accuracy and the efficiency of traffic replay.

4 In-Lab Replay Framework and NATreplay

4.1 In-Lab Replay Framework

In this work, we propose a replay testing framework, In-Lab Replay Testing, and a traffic replayer, NATreplay, to test network devices. ILRT consists of three different roles, the monitor, the traffic replayer, the library of captured traffic traces, and the DUT. Figure 1 illustrates the ILRT framework.

The monitor, called CheckDev, is a probing tool that regularly sends ARP, ICMP and HTTP probes to the DUT to diagnose it during the test. ARP and ICMP probes test the reachability of a networking device hosted on the local network, and HTTP probes check if the device still responds to users' requests. Furthermore, CheckDev would work together with the traffic replayer to adjust the replay throughput based on the response of the DUT.

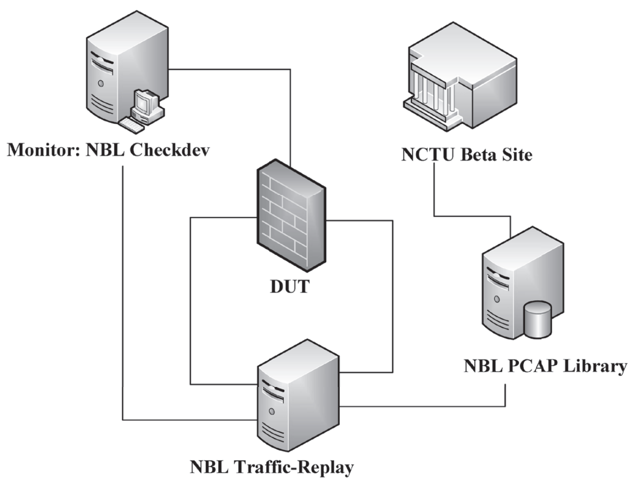

Figure 1 In-Lab Replay Test Framework

The traffic replayer is dedicated to the DUT and can be replaced to qualify the test plans and test requirement. The dedicated traffic replayer is responsible for parsing captured traffic traces, separating the traffic into two groups, updating the states of packets, and sending them to the DUT, The traffic replayer should also log the related statistics about the characteristics of replayed traffic and the responses from the DUT to help later debugging and analysis.

The library of real-world traffic traces stores the traffic traces captured from Beta Site [1]. The traffic from hundreds of volunteers' daily usage is captured and contains numerous application categories and transactions. The traffic traces are updated and profiled regularly, (1) to keep up-to-date all kinds of application behaviors and, (2) to double-check the responses and handling results of DUTs.

4.2 NATreplay

We developed a stateful traffic replayer, NATreplay, to test NAT devices. NATreplay is a modification of TCPreplay and designed to support NAT mechanism. TCPreplay provides a variety of features for replaying traffic both passive sniffer devices and in-line devices such as routers, firewalls, and Intrusion Prevention System (IPS). During the usage, the IP addresses can be rewritten or randomized, MAC addresses can be rewritten, transmission speeds can be adjusted, the truncated packets can be repaired, and the packets can be selectively sent or dropped. However, TCPreplay cannot maintain application-dependent states, which limits its applications. In order to be able to test NAT devices, NATreplay is designed to maintain the mapping between the original and the new network addresses during replay. Related fields, such as networks addresses and payload checksums, need timely updating during replay. Furthermore, the modified values should be kept in subsequent packets during the same replay process.

The mechanism of how NATreplay replay packets is shown in Figure 2. At the beginning, a packet with (L_{ip}, L_{port}, E_{ip}, E_{port}) as its source IP address, port number, destination IP address, and port number, respectively, is sent from the LAN interface of the replayer to the DUT. When the packet passes through the DUT, its source IP and port number are modified by the DUT. A mapping entry (L_{ip}, L_{port}, W_{ip}, W_{port}) is created in a NAT table. Upon receiving the packet at the WAN interface, the replayer at the WAN interface sends a response packet (destination W_{ip}, destination port: W_{port}) to the DUT. Once the response packet reaches the DUT, the DUT will look up the NAT table and check if a mapping entry (L_{ip}, L_{port}, W_{ip}, W_{port}) exists to allow the packet to pass. If yes, the packet will be forwarded to the LAN interface after replacing its destination address (W_{ip}, W_{port}) with (L_{ip}, L_{port}).

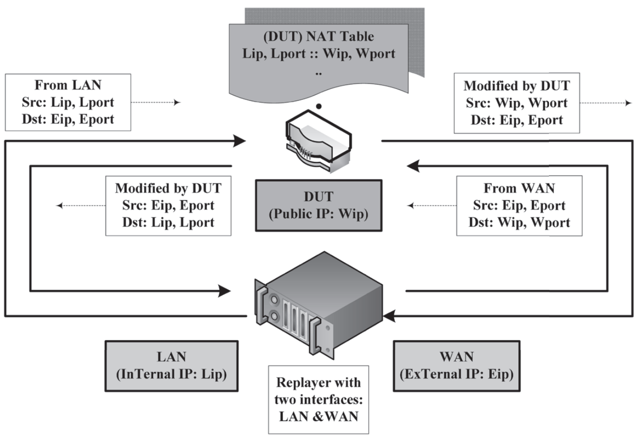

Figure 2 The Mechanism How NATreplay Works

5 Experiment Studies and Case Study

NBL [9] utilizes ILRT framework with real-world traffic to test many kinds of networking devices, including WLAN routers, switches, and application layer proxies. In the following, 28 retailed WLAN routers are used as the DUTs and are evaluated whether they can stably work for an extended time to demonstrate the correctness and the effectiveness of the ILRT framework and NATreplay.

5.1 Trace Selection and Experimental Testbed

To conduct the experiments, we used the network traffic captured from NCTU Beta Site [1]. During peak hour traffic, the captured network traffic volume could reach 60 GB in 30 minutes, and even during regular hours around midnight, the traffic volume could still reach at least 20 GB in 30 minutes. The total volume of the traffic captured ranges from 1.0 to 1.4 TB a day, and the common application profile of the traffic is listed in Table 1. There are always hundreds of applications detected and only the applications with a volume ratio larger than 0.01% are listed, including P2P applications, video streaming applications, file transferring, on-line games, instance messaging, web mails, and DNS queries. Lin et al. [1] suggest that one month and one year of beta site testing are minimum test durations for low-end and high-end products, respectively. The usage of HTTP and video streaming applications dominates the captured network traffic, and among those applications detected, the top 10 mostly belongs to Web and P2P applications. The 28 retailed WLAN routers that were used as the DUTs were numbered from 1 to 28.

5.2 Testbed Configuration

Throughout our experiments the initial replay throughput was set to 50 Mbps while NATreplay would dynamically adjust the throughput according to the responses of a DUT during the test. CheckDev was configured with the information of MAC and IP addresses of the DUT and NATreplay, the LAN and WAN subnets and the *timeout*. The timeout is the maximal allowable interval between the probe and the corresponding response from the DUT. If CheckDev could not collect the responses from the DUT in time, the issued probes would be judged to have failed. The traffic captured in PCAP files is separated into two parts, called primary and secondary, and the network addresses contained are bound to different interfaces of NATreplay. In this way, the DUT can see the traffic that a host in the primary side initiating to another host in the secondary side, will go through it one way and the response will go through it the other way.

Furthermore, multiple instances of Tcpdump [10] were

Table 1 The Application Profile of the Original Raw Traces

Name	Ratio (%)	Name	Ratio (%)	Name	Ratio (%)
Http`	31.4	QVOD	0.95	VNC	0.06
PPStream	11.76	Skype	0.56	qingyule	0.06
FTP	10.33	HTTPS	0.48	KuGoo	0.06
TCP	9.84	Garena	0.43	Telnet	0.05
PPLive	9.44	SSH	0.3	Direct Play 7	0.05
RDP	4.61	RTSP	0.23	Gnutella	0.05
BitTorrent	3.78	tudou	0.21	Webmail	0.05
Funshion	3.69	Gmail	0.2	YouKu	0.04
Xunlei	3.52	iTunes	0.16	DNS	0.04
UDP	2.91	Fs2you	0.14	Dropbox	0.03
YouTuBe	1.9	RTP	0.13	Shoutcast	0.03
Flashcom	1.4	TeamViewer	0.12	Warcraft	0.02
MSN	1.35	Ku6speedup	0.12	SMB	0.02
QQTV	1.32	Steam	0.07	TTPlayer	0.01
eDonkey	0.96	FlashGet	0.06		

deployed between every two roles of ILRT framework during traffic replay to monitor activities in the links. The logs recorded by Tcpdump can be used to verify the status of DUT and NATreplay to check if they work correctly.

5.3 Profile of DUTs

The 28 retailed WLAN routers were used as the DUTs and were evaluated whether they can stably work for an extended time. Table 2 shows the profile of the 28 retailed WLAN routers, including CPU types, RAM size, and Flash RAM size. Over 70% of DUTs support gigabit link and IEEE 802.11 b/g/n technology. For most DUTs, the sizes of RAM and Flash are limited except the only one equipped with large sizes of RAM and Flash.

Table 2 The Profile of 28 Retailed WLAN Routers

Counts	CPU Types (MHz)	RAM (MB)	Flash (MB)
2	133	8	2
2	133	16	2
6	220, 266, 320	16	4
10	266, 384, 400	32	4
4	266, 400	32	8
1	470	64	8
1	384	64	16
1	400	64	32
1	480	128	32

The triggered failures were classified into two groups, critical level (L1) and tolerable failures (L2), based on the severity of damage. Table 3 shows the classification, impact and possible errors of failures occurred on WLAN routers. Because the connectivity of accessing the network or Internet is the fundamental requirement of WLAN routers, we separated the NAT functionality from other functionalities. If the failures disrupted the connectivity with the Internet, they would be identified as critical failures, while if the failures just degraded the performance or slow down the throughputs but still be able to access the Internet, they were identified as tolerable failures. Each occurred failure was double-checked by the logs of NATreplay and CheckDev. The logs recorded the order, the time, and the interface of packets that were sent and received.

To assess the stability of DUTs, we developed the metric *stability score* to differentiate the stability of device by the equation, *stability score* = $V_t / (1 + p_{L1} \times n_{L1} + p_{L2} \times n_{L2})$, where n_{L1}, n_{L2} are the individual number of L1 and L2 failures occurred, p_{L1}, p_{L2} are the relative weights of L1 and L2 failures and Vt is the accumulated volume of replayed traffic (GB). In this experiment, p_{L1} was set to 10 times p_{L2} to emphasize the importance of proper functioning [11].

In our experiments, we tried to apply the same number of test rounds except hardware breakdown. However, because the replayer would dynamically adjust the throughput according to the response of the DUT, different replayed traffic per test round and different accumulated replayed traffic may be obtained.

Figure 3 shows the results of stability testing. The horizontal axis shows individual DUTs, and the vertical axis shows the corresponding stability score and replayed traffic

Table 3 Triggered Failures Severity Classification

Level	Failure	Impact	Possible Errors
L1	Hanging	DUT crashes or hangs	Ran out of memory or memory leak
	Reboot	DUT shutdown and reboot again	Ran out of memory or memory leak
	NAT failure	NAT functionality fails	NAT maintenance fails
	Connection failure	TCP connections fail to build	Ran out of memory or memory leak
L2	Resources exhausted	Fail to accept requests during replay but recover while stopping replay	Ran out of memory or resources
	Functionality failure	Other functionalities except connecting fail	Functionality implementation error
	Slowdown	Throughput degradation	Memory leak or too small memory size

volume. In order to be more easily to compare intuitively, the unit of the replayed volume was set to 10 GB. Take DUT_1 as an example. DUT_1 was tested with the amount of 2051.7 GB traffic, and the stability score was 2564.23. From Figure 3, three interesting observations could be found that (1) even though the DUTs passed laboratory tests, there was still a good chance to fail the test, (2) the stability score was not in proportional to the replayed volume, and (3) even if the hardware spec of a DUT was more powerful, the stability score was not necessarily high.

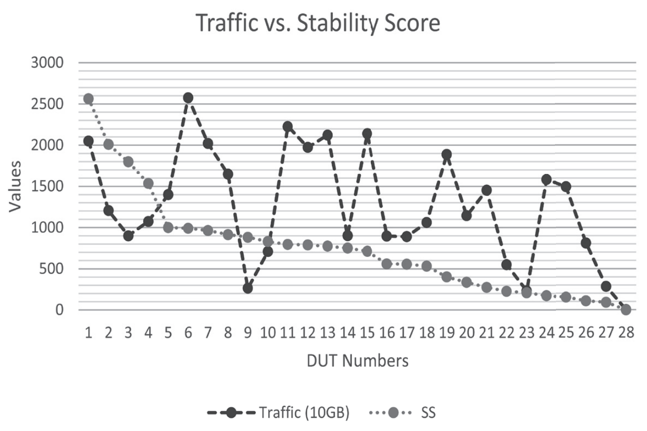

Figure 3 Stability testing results of 28 retailed WLAN routers

Figure 4 showed the relationship of the replayed traffic volume and the triggered failures. The horizontal axis shows individual DUTs, and the vertical axis shows the corresponding number of triggered failures. Take DUT_{21} as an example. DUT_{21} was tested with the amount of 14.35 TB traffic, and the number of L1 and L2 failures were 2 and 32, respectively. Among 458 test results, chance of pass, L1 failure, and L2 failure, is 2.84%, 7.86%, and 89.3%, respectively.

Figure 5 illustrated the relationship of the stability score and triggered failures. The horizontal axis shows individual DUTs, and the vertical axis shows the corresponding stability score and the number of triggered failures. Take

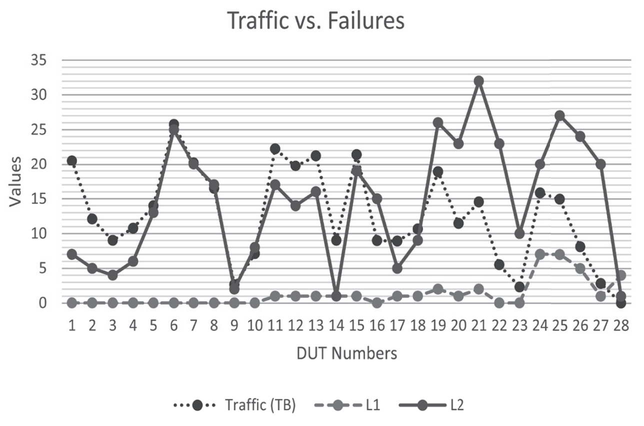

Figure 4 The Relationship of Replayed Traffic and Triggered Failures

DUT_{21} as an example. The stability score of DUT_{21} was 274.5, and the number of L1 and L2 failures were 2 and 32, respectively. From Figure 5, it could be found that more number of triggered failures caused lower stability score.

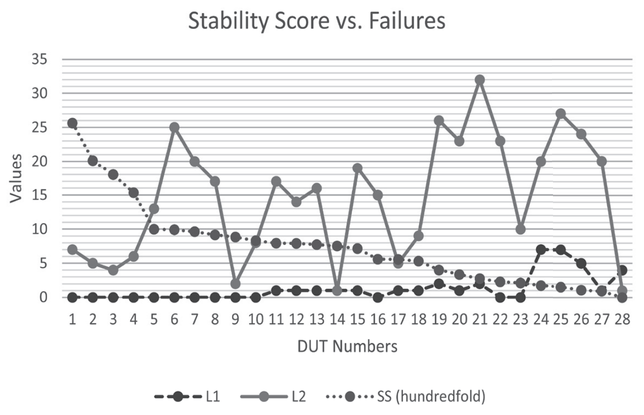

Figure 5 The Relationship of Stability Score and Triggered Failures

From above figures, the quality of a DUT could be easily differentiated from each other even though we had no details of how the mechanism of packet processing was implemented. Among 28 retailed WLAN routers, 53.57% and 100% of DUTs were triggered critical (L1) failures and tolerable (L2) failures, respectively. 21.43% and 100% of DUTs were triggered more than one L1 and L2 failures, respectively. In these experiments, the most L2 failure of a DUT is to not to respond to any request for a while although the DUT was still alive to process incoming and outgoing packets, which could be observed by Tcpdump. Among high-spec DUTs, there was 25% of DUTs which were triggered L1 failures. Therefore, it might be able to conclude that although high-spec products are more powerful, what really matters is the quality of the firmware.

In order to evaluate their performance, we used Spirent SmartBits 600 and WebSuite v2.6 [12] to test the maximum session rate of the 28 DUTs. A fixed number of requests issued to establish TCP sessions per test round can be configured to evaluate the performance of a DUT. Here, the fixed amount requested of TCP sessions/round was set to 15,000 and the experiment for a DUT lasted for a whole day.

The final results were averaged and are shown in Figure 6. Two interesting observations were made: that (1) for each test round, even the rate that a DUT could handle was much lower than the fixed testing rate, but did not crash or stop the DUT, and (2) the hardware specification was not closely linked to the results because some low-spec DUTs outperformed some high-spec ones.

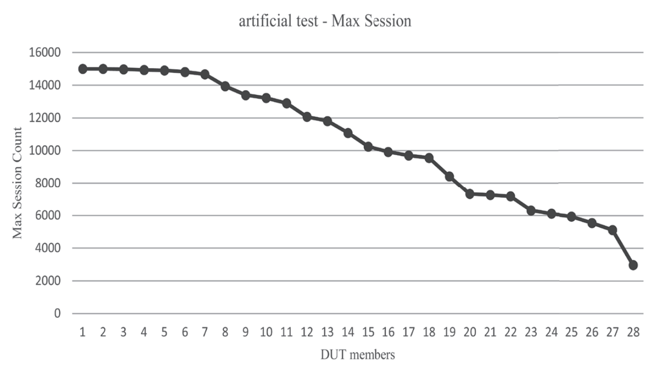

Figure 6 The Maximal Session Test Results Using SmartBits 600 and WebSuite v2.6

6 Conclusions

Testing networking devices before releasing them onto the market is a way of ensuring their quality and robustness. Replaying artificial or real-world traffic is a method used to test networking devices. Using real-world traffic is desirable as it uncovers more realistic properties such as traffic diversity and complicated user behaviors.

In this work, we developed an In-Lab Replay Testing (ILRT) framework, composed of a monitor, a dedicated traffic replayer and a library of real-world traffic traces. We also developed a stateful traffic replayer, NATreplay, which was designed to test NAT devices while keeping the original captured transaction scenarios. The monitor, called CheckDev, is a probing tool that regularly sends ARP, ICMP and HTTP probes and monitors the status of the DUT. The dedicated traffic replayer is responsible for parsing the captured traffic traces, separating the traffic into two sides and sending them to different interfaces of the DUT. The traffic replayer can be replaced to qualify the test requirement, and the library of real-world traffic traces should contain numerous application categories and transactions for comprehensive verification.

28 retailed WLAN routers were used as the DUTs, and were evaluated as to whether they were able to work stably for an extended time, so as to demonstrate the correctness and the effectiveness of the ILRT framework. The triggered failures were classified into two groups, *critical* level (L1) and *tolerable* level (L2) based on the severity of the damage. If the failures disrupted the ability to access the Internet, they would be identified as critical failures, while the triggered failures that only degraded performance but were still able to access the Internet, were identified as tolerable failures. By comparing the logs, observed DUT status, and the triggered failures, the effectiveness of ILRT framework can be ensured because IRLT framework is very suitable to test networking devices.

In our experiments, 53.57% and 100% of DUTs were triggered critical (L1) failures and tolerable (L2) failures, respectively. 21.43% and 100% of DUTs were triggered more than one L1 and L2 failures, respectively. Among high-spec DUTs, there was 25% of DUTs which were triggered L1 failures. Furthermore, among total 458 test results, three types of results, namely pass, L1 failures, L2 failures, were 2.84%, 7.86%, and 89.3%, respectively. The results showed that (1) even though the DUTs have passed traditional lab tests, there is still a good chance for them to fail under real-world traffic, (2) although high-spec products are more powerful, what really matters is the quality of the firmware, (3) even though having no details of the implementation of a DUT, ILRT could still be used to evaluate the quality of a DUT and (4) in artificial test, even the rate that a DUT could handle was much lower than the fixed testing rate, the DUT still didn't crash or stop working.

References

[1] Y.-D. Lin, I-W. Chen, P.-C. Lin, C.-S. Chen and C.-H. Hsu, On Campus Beta Site: Architecture Designs, Operational Experience, and Top Product Defects,

IEEE Communications Magazine, Vol. 48, No. 12, December, 2010.
[2] M. K. Daskalantonakis, A Practical View of Software Management and Implementation Experiences within Motorola, *IEEE Transactions on Software Engineering*, Vol. 18, No. 11, pp. 998-1010, November, 1992.
[3] Y.-D. Lin, P.-C. Lin, T.-H. Cheng, I-W. Chen and Y.-C. Lai, Low-Storage Capture and Loss-Recovery Selective Replay of Real Flows, *IEEE Communication Magazine*, Vol. 50, No. 4, pp. 114-121, April, 2012.
[4] W. Cui, V. Paxson, N. C. Weaver and R. H. Katz., Protocol-Independent Adaptive Replay of Application Dialog, *Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, 2006, pp. XX-XX.
[5] J. Newsome, D. Brumley, J. Franklin and D. Song, Replayer: Automatic Protocol Replay by Binary Analysis, *Proceedings of the 13th ACM Conference on Computer and Communications Security*, Alexandria, VA, 2006, pp. 311-321.
[6] Tcpreplay, *Important: Tcpreplay development is now being done by AppNeta*, http://tcpreplay.synfin.net
[7] S.-S. Hong and S. F. Wu, On Interactive Internet Traffic Replay, *Proceedings of the 8th International Conference on Recent Advances in Intrusion Detection (RAID)*, Seattle, WA, 2005, pp. 247-264.
[8] Y.-C. Cheng, U. Holzle, N. Cardwell, S. Savage and G. M. Voelker, Monkey See, Monkey Do: A Tool for TCP Tracing and Replaying, *Proceedings of the 2004 USENIX Annual Technical Conference*, Boston, MA, 2004, pp. 87-98.
[9] *Network Benchmarking Lab*, http://www.nbl.org.tw.
[10] *Tcpdump & Libpcap*, http://www.tcpdump.org/
[11] *NBL RealFlow Certification Testing*, http://web2010.nbl.org.tw/en/services/real_flow_certi.php
[12] Spirent, *Spirent TestCenter*, http://www.spirent.com/Products/Smartbits

Biographies

Ying-Dar Lin is a Distinguished Professor at National Chiao Tung University. He received his PhD from UCLA in 1993. He is the director of Network Benchmarking Lab, a certified test lab of the Open Networking Foundation (ONF). He is an IEEE Fellow, IEEE Distinguished Lecturer and ONF Research Associate.

Ren-Hung Hwang received his PhD from University of Massachusetts. He is distinguished professor of the department of Computer Science and Information Engineering and the Dean of the Engineering College at National Chung Cheng University, Taiwan. His research interests include wireless networks, software defined networking, and mobile edge computing.

Chun-Nan Lu received his PhD from National Chiao Tung University in 2014. His researches focus on network security and traffic measurement/analysis.

Jui-Tsun Hung received his PhD from the State University of New York in 2003. His current research interests include computer networks, computer architecture, wireless communications, and signal processing.