INTERNATIONAL JOURNAL OF RESEARCH
In Advanced Engineering Technologies

Volume 4, Issue 5 SEP 2015

Using Fault Localization Algorithm to Determine the Failing Rules in Automatic Test Packet Generation

[1] CH. RAMESH, [2] Mr. N. JAYAKRISHNA

[1]M.Tech Student, Department of CSE

ramramesh1212@gmail.com

[2] Assistant Professor, Department of CSE.

jaya1238@gmail.com

ABSTRACT—networks are becoming larger and additional advanced,yet directors relay on rudimentary tools like ping and traceroute to correct issues. We tend to propose an automatic and systematic approach for testing and debugging networks referred to as "Automatic Test Packet Generation" (ATPG). ATPG reads router configurations and generates a device-independent model. The model is employed to come up with a minimum set of take a look at packets to (minimally) exercise each link within the network or (maximally) exercise every rule the network. Take a look at packets ar sent sporadically, and detected failures trigger a separate system to localize the fault. ATPG will discover each purposeful (e.g., incorrect firewall rule) and performance issues (e.g., engorged queue). ATPG complements however goes on the far side earlier add static checking (which cannot discover physiological property or performance faults) or fault localization (which solely localize faults given physiological property results). We describe our example ATPG implementation and results on 2 real-world information sets: Stanford University's backbone network and Internet2. We discover that a little range of take a look at packets suffices to test all rules in these networks: as an example, 4000 packets will cover all rules in Stanford backbone network, whereas fifty four are enough to cover all links. Causing 4000 take a look at packets ten times per second consumes but a hundred and twenty fifth of link capability. ATPG code and also the information sets ar in public out there.

1.INTRODUCTION

It Is notornetwork engineers wrestle with router misconiously arduous to right networks. Every day,network engineers wrestle with router misconfigurations, fiber cuts, broken interfaces, misbranded cables, software system bugs, intermittent links, and a myriad alternative reasons that cause networks to act or fail utterly. Network engineers seek out bugs victimization the foremost elementary tools (e.g.,ping, traceroute, SNMP, and) and hunt down root causes employing a combination of accumulated knowledge and intuition. Debugging networks is just turning into tougher as networks are becoming larger (modern information centers could contain ten thousand switches, a field network could serve 50000 users, a 100-Gb/s long-hau link could carry a hundred thousand flows) and are becoming additional difficult. It's a mall marvel that network engineers are labelled "masters of complexity".

The main improvement of this paper is what we tend to decision associate in nursing automatic take a look at Packet Generation (ATPG) scheme that mechanically generates a lowest set of packets to check the animateness of the underlying topology and therefore the harmony between information plane state and con figuration specifications. The tool may also mechanically generate packets to check

performance assertions such as packet latency. In Example one, rather than Alice manually deciding that packets to send, the tool will thus sporadically on her behalf. In Example a pair of, the tool determines that it should send packets with bound headers to "exercise" the video queue, and then determines that these packets area unit being born.

ATPG detects and diagnoses errors by severally and thoroughly testing all forwarding entries, firewall rules, and any packet process rules within the network. In ATPG, take a look at packets are generated algorithmically from the accessory configuration files and FIBs, with the minimum variety of packets needed for complete coverage. Take a look at packets area unit fed into the network thus that every rule is exercised directly from the information plane. Since ATPG treats links a bit like traditional forwarding rules, its full coverage guarantees testing of each link within the network. It may also be specialized to get a lowest set of packets that simply test each link for network animateness. A minimum of during this basic kind, we feel that ATPG or some similar technique is key to networks: rather than reacting to failures, several network operators like Internet2 proactively check the health of their network victimization pings between all pairs of sources. However, all-pairs doesn't guarantee testing of all links and has been found to be unscalable for giant networks like PlanetLab.

2.RELATED WORK

We square measure unaware of earlier techniques that mechanically generate take a look at packets from configurations. The nearest connected works we all know of square measure offline tools that check invariants in networks. Within the management plane, NICE makes an attempt to thoroughly cover the code methods symbolically in controller applications with the assistance of simplified switch/host models. Within the information plane, Anteater models invariants as mathematician satisfiability troubles and audit them against configurations with a Saturday solver. Header house Analysis uses a geometrical model to check reachability, notice loops, and verify slicing. Recently, SOFT was projected to verify consistency between totally different OpenFlow agent implementations that square measure accountable for bridging management and information planes within the SDN context. ATPG complements these checkers by directly testing the information plane and covering a compelling set of dynamic or performance errors that cannot well be captured. End-to-end probes have long been employed in network fault diagnosing. Recently, mining low-quality, unstructured information, like router configurations and network tickets, has attracted interest. By distinction, the first contribution of ATPG isn't fault localization, however decisive a compact set of end-to-end measurements which will cowl each rule or each link. The mapping between Min-Set-Cover and network observation has been explored. ATPG improves the detection granuality to the rule level by using router configuration and information flat data. What is more, ATPG isn't restricted to animateness testing, however are often applied to checking higher level properties like performance.

There square measure several proposals to develop a measurement-friendly planning for networks. Our approach is complementary to those proposals: By incorporating input and port constraints, ATPG will generate take a look at packets and injection points victimization existing readying of measuring devices.

3. ATPG SYSTEM

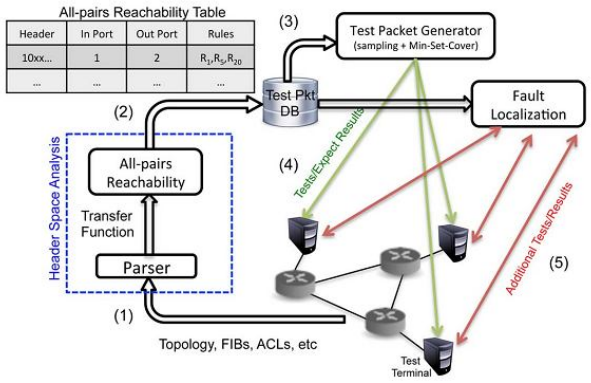

Fig. 5. ATPG system block diagram.

From the figure 5, the system 1st collects all the forwarding state from the network (step 1). This usually involves reading the FIBs, ACLs, and con fig files, as well as getting the topology. ATPG uses Header house Analysis to calculate reachability between all the take a look at terminals (step 2).

The result's then utilized by the take a look at packet choice algorithmic program to calculate a bottom set of take a look at packets that may take a look at all rules (step 3). These packets are sent sporadically by the test terminals (step 4). If a slip is detected, the fault localization algorithmic program is invoked to slender down the reason behind the error (step 5).

A. Test Packet Generation:

We assume a group of check terminals within the network will send and receive check packets. Our goal is to come up with a group of check packets to exercise each rule out each switch operate, so any fault are going to be ascertained by a minimum of one check packet. this can be analogous to package check suites that attempt to check each doable branch in an exceedingly program. The broader goal may be restricted to testing each link or each queue.

When generating audit packets, ATPG should respect 2 key constraints: 1) Port: ATPG should solely use check terminals that area unit available; 2) Header: ATPG should solely use headers that every check terminal is allowable to send. for instance, the network administrator might solely permit employing a specific set of VLANs. Formally, we've got the subsequent downside.

B. Fault Localization:

1) Fault Model:

A rule fails if its determined behavior differs from its expected behavior. ATPG keeps track of wherever rules fail employing a result perform . For a rule , the result perform is outlined as

$$R(r, pk) = \begin{cases} 0, & \text{if } pk \text{ fails at rule } r \\ 1, & \text{if } pk \text{ succeeds at rule } r. \end{cases}$$

"Success" and "failure" depend upon the character of the rule: A forwarding rule fails if a check packet isn't delivered to the supposed output port, whereas a drop rule behaves properly once packets ar born. Similarly, a link failure could be a failure of a forwarding rule the topology perform. On the opposite hand, if an output link is engorged, failure is captured by the latency of a test packet going higher than a threshold. We divide faults into 2 categories: action faults and match faults. An action fault happens once each packet matching the rule is processed incorrectly. Samples of action faults embrace unexpected packet loss, a missing rule, congestion, and miswiring. On the opposite hand, match faults ar more durable to sight as a result of they solely have an effect on some packets matching the rule: as an example, once a rule matches a header it mustn't, or when a rule misses a header it ought to match. Match faults will solely be detected by a lot of complete sampling such a minimum of one check packet exercises every faulty region. We can generally solely observe a packet at the sting of the network once it's been processed by each matching rule. Therefore, we have a tendency to DE fine AN end-to-end version of the result perform

$$R(pk) = \begin{cases} 0, & \text{if } pk \text{ fails} \\ 1, & \text{if } pk \text{ succeeds.} \end{cases}$$

4. USE CASES

We can use ATPG for each useful and performance testing, because the following use cases demonstrate.

A. Functional Testing:

We can take a look at the purposeful correctness of a network by testing that every approachable forwarding and drop rule the network is behaving properl.

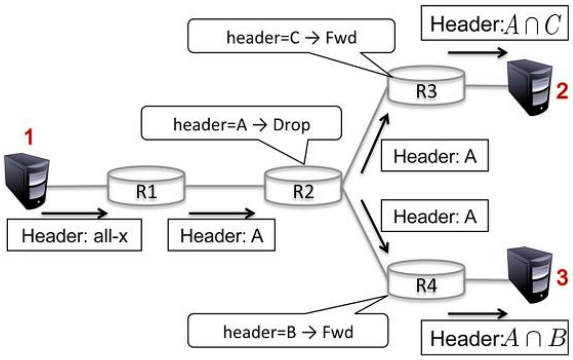

Fig. 7. Generate packets to test drop rules: "flip" the rule to a broadcast rule in the analysis.

As associate in nursing example, contemplate Fig. 7. to check the drop decree, we inject the all- check packet at Terminal one. If the drop rule was instead a broadcast rule, it'd forward the packet to any or all of its output ports, and therefore the check packets would reach Terminals 2 and 3.

Finally, we tend to send and expect the 2 check packets to not appear since their arrival would indicate a failure of 's drop rule.

B. Performance Testing:

We can conjointly use ATPG to observe the performance of links, queues, and QoS categories within the network, and even monitor SLAs.

Congestion: If a queue is full, packets can expertise longer queuing delays. This may be thought of as a (performance) fault. ATPG lets North American nation generate a way congestion tests to measure the latency between each try of take a look at terminals; once the latency passed a threshold, fault localization can pinpoint the full queue, like regular faults. With applicable headers, we are able to take a look at links or queues as in Alice's second problem.

Available Bandwidth: equally, we are able to live the obtainable information measure of a link, or for a selected service category. ATPG will generate the take a look at packet headers required to check each link, or each queue, or each service class; a stream of packets with these headers will then be wont to live information measure.

5. IMPLEMENTATION

A. Test Packet Generator:

The check packet generator, written in Python, contains a Cisco IOS con figuration program and a Juniper Junos program. The data plane info, as well as router configurations, FIBs, MAC learning tables, and network topologies, is collected and parsed through the statement interface (Cisco IOS) or XML files (Junos). The generator then uses the Odd Hassel header house analysis library to construct switch and topology functions.

B. Network Monitor:

The network monitor assumes there area unit special check agents within the network that area unit able to send/receive check packets. The network monitor reads the information and constructs check packets and instructs every agent to send the acceptable packets. Currently, check agents separate check packets by informatics early field and TCP/UDP port range, however different fields, like informatics possibility, can even be used. If a number of the tests fail, the monitor selects further check packets from reserved packets to pinpoint the matter. The method repeats till the fault has been identified. The monitor uses JSON to speak with the check agents, and uses SQLite's string matching to search check packets with efficiency.

6. OVERHEAD AND PERFORMANCE

The principal sources of overhead for ATPG area unit polling the network sporadically for forwarding state and performing arts allpairs reachability. Whereas one will cut back overhead by running the offline ATPG calculation less oftentimes, this runs the danger of victimisation outdated forwarding data. Instead, we tend to cut back overhead in 2 ways in which. First, we've recently sped up the all-pairs reachability calculation employing a quick multithreaded/multimachine header area library. Second, rather than extracting the complete network state on every occasion ATPG is triggered, an progressive state updater will considerably cut back each the retrieval time and also the time to calculate reachability. We tend to area unit engaged on a period of time version of ATPG that comes with each techniques. Test agents inside terminals incur negligible overhead as a result of they just demultiplex check packets self-addressed to their IP address at a modest rate (e.g., one per millisecond) compared to the link speeds Gb/s most recent CPUs area unit capable of receiving.

7. EVALUATION

We ran ATPG on a quad-core Intel Core i7 processor three.2 GHz and 6 GB memory exploitation eight threads. For a given range of check terminals, we tend to generated the minimum set of check packets required to test all the approachable rules within the Stanford and Internet2 backbones.

TABLE V
TEST PACKET GENERATION RESULTS FOR STANFORD BACKBONE (TOP) AND INTERNET2 (BOTTOM), AGAINST THE NUMBER OF PORTS SELECTED FOR DEPLOYING TEST TERMINALS. "TIME TO SEND" PACKETS IS CALCULATED ON A PER-PORT BASIS, ASSUMING 100 B PER TEST PACKET, 1 GB/S LINK FOR STANFORD, AND 10 GB/S FOR INTERNET2

Stanford (298 ports)	10%	40%	70%	100%	Edge (81%)
Total Packets	10,042	104,236	413,158	621,402	438,686
Regular Packets	725	2,613	3,627	3,871	3,319
Packets/Port (Avg)	25.00	18.98	17.43	12.99	18.02
Packets/Port (Max)	206	579	874	907	792
Time to send (Max)	0.165ms	0.463ms	0.699ms	0.726ms	0.634ms
Coverage	22.2%	57.7%	81.4%	100%	78.5%
Computation Time	152.53s	603.02s	2,363.67s	3,524.62s	2,807.01s
Internet2 (345 ports)	10%	40%	70%	100%	Edge (92%)
Total Packets	30,387	485,592	1,407,895	3,037,335	3,036,948
Regular Packets	5,930	17,800	32,352	35,462	35,416
Packets/Port (Avg)	159.0	129.0	134.2	102.8	102.7
Packets/Port (Max)	2,550	3,421	2,445	4,557	3,492
Time to send (Max)	0.204ms	0.274ms	0.196ms	0.365ms	0.279ms
Coverage	16.9%	51.4%	80.3%	100%	100%
Computation Time	129.14s	582.28s	1,197.07s	2,173.79s	1,992.52s

Table V shows the amount of check packets required. For example, the primary column tells USA that if we tend to

attach check terminals to 100% of the ports, then all of the approachable Stanford rules (22.2% of the total) are often checked by causation 725 test packets. If each edge port will act as a check terminal, 100% of the Stanford rules are often tested by causation simply three,871 check packets. The "Time" row indicates however long it took ATPG to run; the worst case took regarding associate hour, the majority of that was dedicated to shrewd all-pairs reachability. To put these results into perspective, every check for the Stanford backbone needs causation regarding 907 packets per port within the worst case. If these packets were sent over one 1-Gb/s link, the entire network may well be tested in but one ms, assuming each check packet is a hundred B and not considering the propagation delay. place otherwise, testing the whole set of forwarding rules 10 times each second would use but one hundred and twenty fifth of the link information measure. Similarly, all the forwarding rules in Internet2 are often tested using 4557 check packets per port within the worst case. notwithstanding the check packets were sent over 10-Gb/s links, all the forwarding rules could be tested in but 0.5 ms, or ten times each second using but 100 and 25th of the link information measure.

8. CONCLUSION

Testing animateness of a network could be a basic drawback for ISPs and enormous information center operators. causing probes between every try of edge ports is neither thoroughgoing nor ascendable. It suffices to seek out a bottom set of end-to-end packets that traverse each link. However, doing this needs some way of abstracting across device speci fic con figuration files (e.g., header space), generating headers and therefore the links they reach (e.g., all-pairs reachability), and at last crucial a minimum set of take a look at packets (Min-Set-Cover). Even the elemental drawback of mechanically generating take a look at packets for economical animateness testing needs techniques like ATPG.

ATPG, however, goes abundant additional than animateness testing with the same framework. ATPG will take a look at for reachability policy (by testing all rules together with drop rules) and performance health (by associating performance measures like latency and loss with take a look at packets). Our implementation additionally augments testing with an easy fault localization theme additionally made mistreatment the header area framework. As in package testing, the formal model helps maximize take a look at coverage whereas minimizing take a look at packets. Our results show that every one forwarding rules in Stanford backbone or Internet2 is exercised by a astonishingly little number of take a look at packets (for Stanford, and for Internet2). Network managers nowadays use primitive tools like and . Our survey results indicate that they're eager for a lot of subtle tools. Different fields of engineering indicate that these wishes aren't unreasonable: as an example, both the ASIC and package style industries area unit supported by billion-dollar tool businesses that provide techniques for each static (e.g., style rule) and dynamic (e.g., timing) verification. In fact, several months when we have a tendency to designed and named our system, we have a tendency to discovered to our surprise that ATPG was a well known word form in hardware chip testing, wherever it stands for Automatic take a look at Pattern Generation. We have a tendency to hope network ATPG are going to be equally useful for machine-controlled dynamic testing of production networks.

REFERENCES

[1] "ATPG code repository," [Online]. Available: http://eastzone.github.com/atpg/

[2] "Automatic Test Pattern Generation," 2013 [Online]. Available:http://en.wikipedia.org/wiki/Automatic_test_pattern_generation

[3] P. Barford, N. Duffield, A. Ron, and J. Sommers, "Network performance anomaly detection and localization," in Proc. IEEE INFOCOM, Apr. , pp. 1377–1385. [4]"Beacon,"[Online].Available:http://www.beaconcontroller.net/

[5] Y. Bejerano and R. Rastogi, "Robust monitoring of link delays and faults in IP networks," IEEE/ACM Trans. Netw., vol.14,no.5,pp.1092–1103,Oct.2006.

[6] C. Cadar, D. Dunbar, and D. Engler, "Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs," in Proc. OSDI, Berkeley, CA, USA, 2008,pp.209–224.

[7] M. Canini, D. Venzano, P. Peresini, D. Kostic, and J. Rexford, “A NICE way to test OpenFlow applications,” in Proc.NSDI,2012,pp.10–10.

[8] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, “Netdiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data,” in Proc. ACM CoNEXT,2007,pp.18:1–18:12..