International Journal of Computer Science
ISSN:2348-6600
Oddity...Probe...Reviste...
Published by SK Research Group of Companies(SKRGC)

http://www.ijcsjournal.com Volume 3, Issue 1, No 3, 2015. ISSN: 2348-6600
Reference ID: IJCS-083 PAGE NO: 473-478.

AUTOMATIC TEST PACKET GENERATION FOR NETWORKS

Mr. S.MD. Asif , (M.tech Student),
Department of Computer Science and technology
Madina Engineering College,
Andhra Pradesh, Kadapa, India.
syedmohammed548@yahoo.com

Sri. K. Sreenivasulu, Professor&H.O.D,
Department of Computer Science and technology
Madina Engineering College,
Andhra Pradesh, Kadapa, India.
sreenu.kutala@gmail.com

Abstract*— *In today's Environment network are increasing day by day to maintain the situations is complex because an administrator couldn't be find the bugs of bugs primitive kind of tools. To handle this type of situations proposed a new technology called automatic test packet generation (ATPG) is a systematic approach for testing and debugging networks like LAN & WAN and it reads the router configuration and generates the independent device model.

It generates minimum test packets and examines every protocol in network. These test packets are flow in sequence and find out the failures of a localized separated mechanism. ATPG can detected both incorrect firewall rule and perform problem. ATPG have static checking or fault localization.

Index Terms: Fault localization, Firewall policies, Static & Dynamic checking, automatic test packet configuration.

I. INTRODUCTION

In IT technology networks are maintain by the network administrators or network engineer generally network are manipulate the CISCO configuration of router, connecting of misbehave cables, software bugs, faulty interface and faulty fibers to find out troubleshooting problems network engineers are use this some system tools (e.g. PING, tracer outer, SNMP , and TCPdump). Debugging Networks is only becoming harder as networks are getting bigger (modern data centers may contain 10 000 switches, a campus network may serve 50 000 users, a 100-Gb/s long-haul link may carry 100 000 flows) and are getting more complicated (with over 6000 RFCs, router software is based on millions of lines of source code, and network chips often contain billions of gates). It is a small wonder that network engineers have been labeled "masters of complexity" . For example suppose a router with a faulty line card starts dropping packets silently. Alice, who administers 100 routers, receives a ticket from several unhappy users complaining about connectivity. First, Alice examines each router to see if the configuration was changed recently and concludes that the configuration was untouched. Next, Alice uses her knowledge of the topology to triangulate the faulty device with and finally, she calls a colleague to replace the line card.

Suppose that video traffic is mapped to a specific queue in a router, but packets are dropped because the token bucket rate is too low. It is not at all clear how Alice can track down such a performance. Troubleshooting a network is difficult for three reasons. First, the forwarding state is distributed across multiple routers and firewalls and is defined by their forwarding tables, filter rules, and other configuration parameters. Second, the forwarding state is hard to observe because it typically requires manually logging into every box in the network. Third, there are many different programs, protocols, and humans updating the forwarding state simultaneously when Alice uses and, she is using a crude lens to examine the current forwarding state for clues to track down the failure.

ATPG treats links just like normal forwarding rules, its full coverage Guarantees testing of every link in the network. It can also be specialized to generate a minimal set of packets that merely test every link for network livens. At least

International Journal of Computer Science
ISSN:2348-6600
Oddity...Probe...Reviste...
Published by SK Research Group of Companies(SKRGC)
http://www.ijcsjournal.com Volume 3, Issue 1, No 3, 2015. ISSN: 2348-6600
Reference ID: IJCS-083 PAGE NO: 473-478.

in this basic form, we feel that ATPG or some similar technique is fundamental to networks: Instead of reacting to failures, many network operators such as Internet2 proactively check the health of their network using pings between all pairs of sources. However, all-pairs does not guarantee testing of all links and

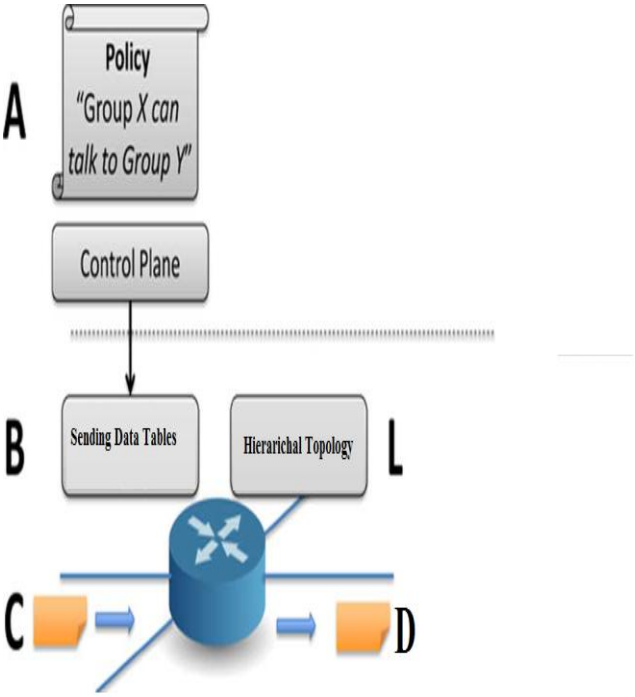
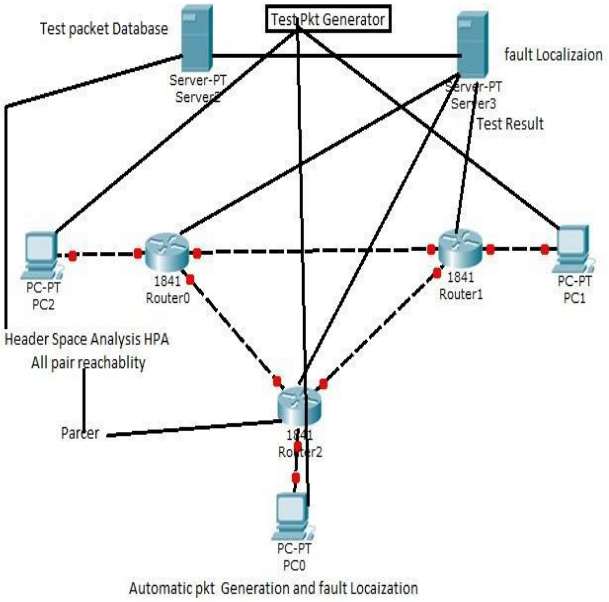


Fig: Static versus dynamic checking

II.FAULT LOCALIZATION

Now a day's Networks are getting larger and more complex, hence network admin depend on normal tools such as ping and to trace route debug the problems. We are proposing automatic and systematic approach for testing and debugging networks called "Automatic Test Packet Generation and Fault Localization". ATPG read router configurations and generates a unique model. This model is generating a minimum set of test packets to exercise every link in network exercise every rule in the network. Test packets are sent periodically and detected failure trigger a separate mechanism to localize the fault. ATPG can detect both functional testing and performance testing problems. ATPG complements but goes beyond earlier work in static checking or fault localization. We describe our prototype ATPG implementation and results on two real-world data sets applications: like Stanford University's backbone network and Internet2. We find that small number of test packets suffices test all rules in these networks.



STEP 1- This involves reading the FIBs, ACLs, and config file, and obtaining the topology. ATPG uses Header Space Analysis to compute reach ability between all the test terminals.

STEP 2- The result is then used by the test packet selection algorithm to compute a minimal set of test packets that can test ll rules.

STEP 3 - These packets will be sent periodically by the test terminals

STEP 4 - If an error is erected, the fault localization algorithm is down the cause of the error.

A general survey of network admin provides information about common failures and root causes in network. A fault localization algorithm is to quarantine faulty devices and its rules and configurations. ATPG performs various testing like functional and performance testing to improve accuracy. Evaluation of a prototype ATPG system using rule sets collected from the Stanford and Internet2 backbones.

International Journal of Computer Science
ISSN:2348-6600
Oddity...Probe...Reviste...
Published by SK Research Group of Companies(SKRGC)
http://www.ijcsjournal.com Volume 3, Issue 1, No 3, 2015. ISSN: 2348-6600
Reference ID: IJCS-083 PAGE NO: 473-478.

III.ALGORITHM

1**) FAULT MODEL**: A rule fails if its observed behavior differs from its expected behavior. ATPG keeps track of where rules

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

fail using a result function. For a rule, the result function is defined as
We divide faults into two categories: action faults and match
FAULTS: An action fault occurs when every packet matching the rule is processed incorrectly. Action faults include unexpected packet loss, a missing rule, congestion, and miswiring. On the other hand, match faults are harder to detect because they only affect some packets matching the rule: for example, when a rule matches a header it should not, or when a rule misses a header it should match.
We will only consider action faults because they cover most likely failure conditions and can be detected using only one test packet per rule.
2) PROBLEM 2 (FAULT LOCALIZATION): Given a list of (pk0, (R(pk0), (pk1, (R(pk1)) … tuples, find all that satisfies $^{\exists}$pki,R(pki,r)=0.
STEP 1: Consider the results from sending the regular test packets. For every passing test, place all rules they exercise into a set of passing rules, P. Similarly, for every failing test, place all rules they exercise into a set of potentially failing rules F. By our assumption, one or more of the rules F are in error. Therefore F-P, is a set of suspect rules.
STEP 2: ATPG next trims the set of suspect rules by weeding out correctly working rules. ATPG does this using the reserved packets. ATPG selects reserved packets whose rule histories contain exactly one rule from the suspect set and sends these packets. Suppose a reserved packet p exercises only rule r in the suspect set. If the sending of p fails, ATPG infers that rule r is in error; if p passes; r is removed from the suspect set. ATPG repeats this process for each reserved packet chosen in
STEP 3: In most cases, the suspect set is small enough after Step 2, which ATPG can terminate and report the suspect set.

If needed, ATPG can narrow down the suspect set further by sending test packets that exercise two or more of the rules in the suspect set using the same technique underlying Step 2. If these test packets pass, ATPG infers that none of the exercised rules are in error and removes these rules from the suspect set.

If our Fault Propagation assumption holds, the method will not miss any faults, and therefore will have no false negatives.
FALSE POSITIVES: Note that the localization method may introduce false positives, rules left in the suspect set at the end of Step 3. Specifically, one or more rules in the suspect set may in fact behave correctly. False positives are unavoidable in some cases.

When two rules are in series and there is no path to exercise only one of them, we say the rules are indistinguishable; any packet that exercises one rule will also exercise the other. Hence, if only one rule fails, we cannot tell which one. For example, if an
ACL rule is followed immediately by a forwarding rule that matches the same header, the two rules are indistinguishable.

Observe that if we have test terminals before and after each rule (impractical in many cases), with sufficient test packets, we can distinguish every rule. Thus, the deployment of test terminals not only affects test coverage, but also localization accuracy.

III.TAXONOMY OF CHART

Parameter / References	Fault Propagation	Fault Activation	Hardware Failure	Software Bug	QOS (B/W,Latency,, Throughput
Automatic Test Pattern Generation	✓	✓	✓	✓	✓
Robust monitoring of link delays and faults in IP networks	✓	✓	x	x	x
Network tomography of binary network	✓	✓	✓	x	x
All-pairs ping service for PlanetLab	✓	✓	x	x	x

IV.FIREWALL POLICES

A fault model of firewall policies is an explicit hypothesis about potential faults in firewall policies.

International Journal of Computer Science
ISSN:2348-6600
Oddity...Probe...Reviste...
Published by SK Research Group of Companies(SKRGC)
http://www.ijcsjournal.com Volume 3, Issue 1, No 3, 2015. ISSN: 2348-6600
Reference ID: IJCS-083 PAGE NO: 473-478.

Our proposed Fault model includes five types of faults.

1. WRONG ORDER: This type of fault indicates that the order of rules is wrong. Recall that the rules in a firewall policy follow the first-match semantics due to conflicts between rules.

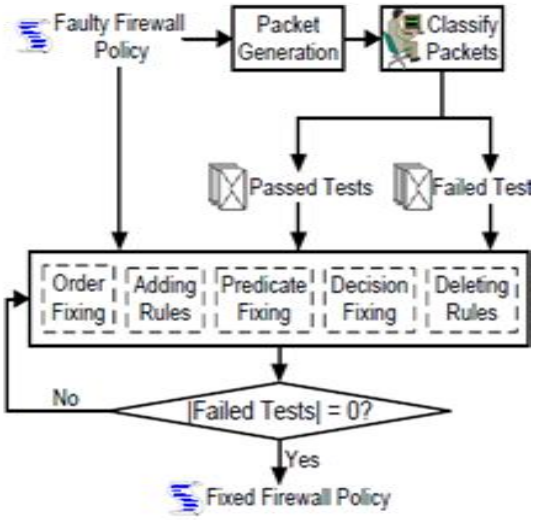
Disordering firewall rule scan misconfigure a firewall policy. Wrong order of rules is a common fault caused by adding a new rule at the beginning of a firewall policy without carefully considering the order between the new rule and the original rules. For example, if we misorder r1 and r2 in Figure 1, all packets will be discarded.

2. MISSING RULES: This type of fault indicates that administrators need to add new rules to the original policy. Usually, administrators add a new rule regarding a new security concern. However, sometimes they may forget to add the rule to the original firewall policy.

3. WRONG PREDICATES: This type of fault indicates that predicates of some rules are wrong. When configuring a firewall policy, administrators define the predicates of rules based on security requirements. However, some special cases may be overlooked.

4. WRONG DECISIONS: This type of fault indicates that the decisions of some rules are wrong.

5. WRONG EXTRA RULES: This type of fault indicates that administrators need to delete some rules from the original policy. When administrators make some changes to a firewall policy, they may add a new rule but sometimes forget to delete old rules that filter a similar set of packets as the new rule does.



V.ATPG

Let's consider a scenario where an administrator maps video traffic to a specific queue in a router, and packets are dropped because the token bucket rate is low. What would the network administrator do in such case?

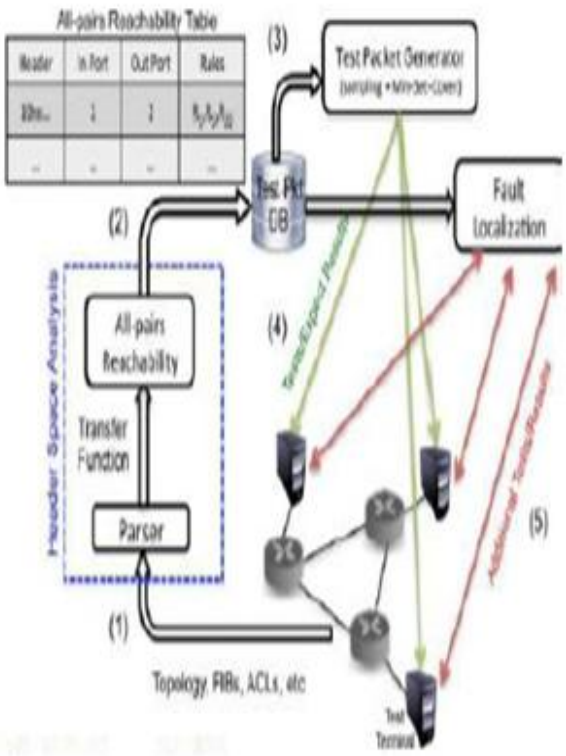
CURRENT SYSTEM:

The administrator manually decides which ping packets to send. Here, the approaches designed can prevent software logic errors but fails to detect failures caused by failed links and routers.

ATPG SYSTEM:

Instead of the administrator, the ATPG tool would do so periodically on his or her behalf. Whereas here, ATPG automatically detects the failures by testing the liveness of the underlying topology

When an error is detected, ATPG goes through the following



STEPS:

- ✓ The system first collects all the forwarding state from the network

International Journal of Computer Science
ISSN:2348-6600
Oddity...Probe...Reviste...
Published by SK Research Group of Companies(SKRGC)
http://www.ijcsjournal.com Volume 3, Issue 1, No 3, 2015. ISSN: 2348-6600
Reference ID: IJCS-083 PAGE NO: 473-478.

- ✓ ATPG uses Header Space Analysis to compute reach ability between all the test terminals.
- ✓ The result is then used by the test packet selection algorithm to compute a minimal set of test packets that can test all rules.
- ✓ These packets will be sent periodically by the test terminals.
- ✓ If an error is detected, the fault localization algorithm is invoked to narrow down the cause of the error.

Step 1: Collect all forwarding states: Forwarding table which usually involves reading the FIBs.

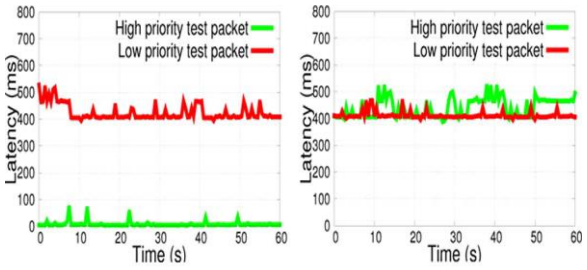
Step 2: Generate All-Pairs Reach ability Table: ATPG Start's by computing the complete set of packet headers that can be sent from each test terminal to every other test terminal. For each such header, ATPG finds the complete set of rules it exercises along the path. To do so, ATPG applies the all-pairs reach ability algorithm as follows:

1. Header constraints are applied.

For example, if traffic can be sent on VLAN A, then instead of starting with an all- x header, the VLAN tag bits are set to A.

2. Set of rules that match the packet are recorded

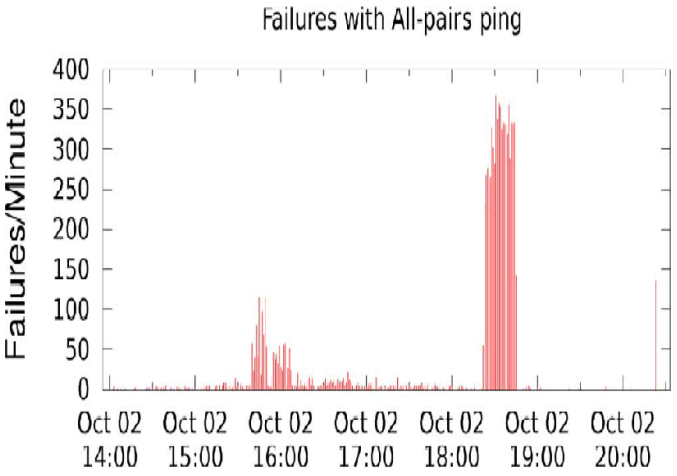
VI.RESULT ANALYSIS

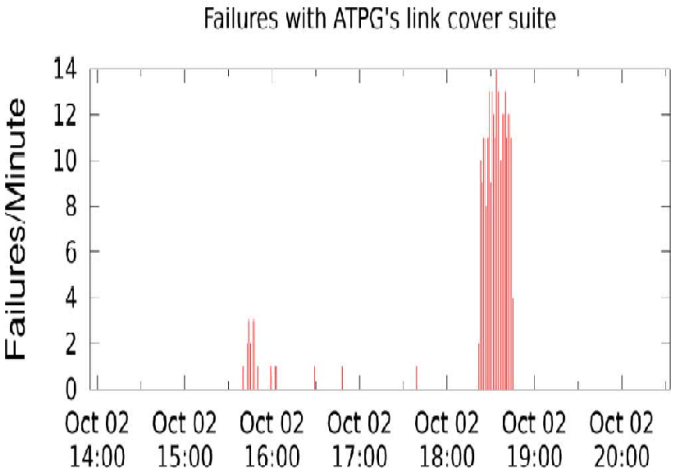


(a) (b)

Congested Slice	Terminal	Result/Mbps
High	High	11.95
	Low	11.66
Low	High	23.22
	Low	11.78

(c)





VII.CONCLUSION AND FUTURE ENHANCEMENT

In this proposed System we use a method which is neither exhaustive nor scalable. Even though it reaches all the pairs of edge nodes it fails to detect faults in liveness properties. ATPG, however, goes much further than liveness testing with the same framework. ATPG can test for reach ability policy (by testing all rules including drop rules) and performance health (by associating performance measures such as latency and loss with test packets).

International Journal of Computer Science
ISSN:2348-6600
Oddity...Probe...Reviste...
Published by SK Research Group of Companies(SKRGC)

http://www.ijcsjournal.com Volume 3, Issue 1, No 3, 2015. ISSN: 2348-6600
Reference ID: IJCS-083 PAGE NO: 473-478.

Our implementation also augments testing with a simple fault localization scheme also constructed using the header space framework.

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..
[9] N. Duffield, "Network tomography of binary network performance characteristics," IEEE Trans. Inf. Theory, vol. 52, no. 12, pp. 5373–5388, Dec. 2006.
[10] N. Duffield, F. L. Presti, V. Paxson, and D. Towsley, "Inferring link loss using striped unicast probes," in Proc. IEEE INFOCOM, 2001, vol. 2, pp. 915–923.
[11] N. G. Duffield and M. Grossglauser, "Trajectory sampling for direct traffic observation," IEEE/ACM Trans. Netw., vol. 9, no. 3, pp. 280–292, Jun. 2001.
[12] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," in Proc. ACM SIGCOMM, 2011, pp. 350–361.
[13] "Hassel, the Header Space Library," [Online]. Available: https://bitbucket. org/peymank/hassel-public/
[14] Internet2, Ann Arbor, MI, USA, "The Internet2 observatory data collections," [Online]. Available: http://www.internet2.edu/observatory/ archive/data-collections.html