MJRET
Open Access
Multidisciplinary Journal of Research in Engineering and Technology
Journal homepage: www.mjret.in
ISSN:2348 - 6953

REVIEW PAPER ON AUTOMATIC TEST PACKET GENERATION AND FAULT LOCALIZATION

Mr. Shrikant B. Chavan[1], Soumitra Das [2],
Dr. D. Y. Patil School of Engineering, (Affiliated to Savitribai Phule Pune University),Pune, India
[1]shrikantbchavan@gmail.com , [2]hod_comp@dypic.in

Abstract: *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.*

Keyword: *Data plane analysis, network troubleshooting, test packet generation.*

1. INTRODUCTION

It is very hard to debug network. Daily network engineers wrestle with router mis-configuration, fiber cuts, faulty interfaces, mislabelled cables, software bugs, intermittent links, and other reasons that cause networks fail completely. Network engineers to kill down bugs using the most common tools (e.g., *Ping, Traceroute, SNMP and Tcpdump*) and track root causes using a combination of accrued wisdom. Debugging of networks is hard to networks are getting bigger for example modern data centres may contain 10 000 switches, a campus network given

service to 50000 users and 100-Gbps long link may carry 100000 flow and got more complicated with over 6000 RFCs, router software is based on thousands of lines of source code, and network chips also contain thousands of gates. It is wonder that network engineers have labelled “masters of complexity”.

For this see the examples.

Example 1: Suppose a router with a faulty line card starts dropping packets silently. Admin, who administers 100 routers, receives a ticket from several unhappy users complaining about connectivity. First Admin examines each router to see if the configuration was changed recently and concludes that the configuration was untouched [2].

Next, Admin uses his knowledge of topology to trace the faulty device with *ping* and tracerout *command.* Finally, he calls a colleague to replace the cable. Two most common causes of network failure are generally hardware failures and software bugs, and that problems detected themselves both as reach ability failures and throughput/latency degradation. Our goal is to automatically detect these types of failures The main contribution of a paper is what we call an Automatic Test Packet Generation [ATPG] framework that automatic generates a minimal set of packets to test liveness that provide support for topology. The tool can also automatically generate packets to test performance assertions such as packet latency.

In Example 1, instead of Admin manually decide which packets to send, the tool does periodically on his behalf. ATPG detects and diagnoses errors by independently and testing all forwarding entries, firewalls rules, and any packet processing rules in network.

In ATPG, test packets are create algorithmically from the configuration files and FIB, with minimum number of packets required completing test. Test packets are provide into the network so that every rule is checked directly from the data plane. Since ATPG treats links just like normal forwarding rules, it’s full testing of every link in the network [2].

It can also specialize to generate a minimal set of packets that test every link for network liveness. At least in this basic form, we feel that ATPG or some similar technique is fundamental to networks: Instead of reacting to failures.

3. PROPOSED SYSTEM

Based on the network model, ATPG generates the minimal number of test packets so that every forwarding rule in the network is check and covered by at least one test packet. When an error is detected, ATPG uses a fault localization algorithm to determine the failing rules or links [1].

Fig.1 is a block diagram of the ATPG system. The system first collects all the forwarding state from the network then all below test perform on network.

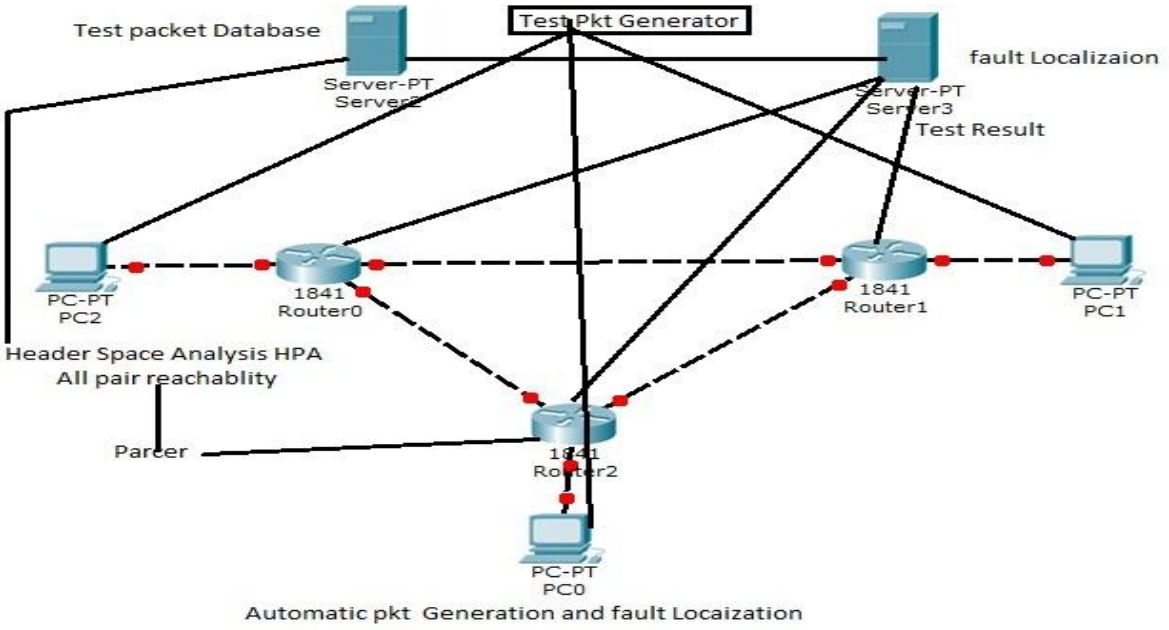

Fig.1: Automatic packet generation and fault localization

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.

3.1 ADVANTAGES OF PROPOSED SYSTEM:

- 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.

4. MATHEMATIC MODEL

We can think of the controller compiling the policy (A) into device-specific configuration files (B), which in turn determine the forwarding behavior of each packet (C). To ensure the net-work

behaves as designed, all three steps should remain consistent at all times i.e. In addition, the topology, shown to the bottom right in the figure, should also satisfy a set of liveness properties. Minimally, requires that sufficient links and nodes are working; if the control plane specifies that a laptop can access a server, the desired outcome can fail if links fail. Can also specify performance guarantees that detect flaky links.

Bit	$b = 0\|1\|\mathtt{x}$
Header	$h = [b_0, b_1, \ldots, b_L]$
Port	$p = 1\|2\|\ldots\|N\|\mathtt{drop}$
Packet	$pk = (p, h)$
Rule	$r : pk \rightarrow pk$ or $[pk]$
Match	$r.matchset : [pk]$
Transfer Function	$T : pk \rightarrow pk$ or $[pk]$
Topo Function	$\Gamma : (p_{src}, h) \rightarrow (p_{dst}, h)$

(a)

function $T_i(pk)$
  #Iterate according to priority in switch i
  for $r \in ruleset_i$ **do**
    if $pk \in r.matchset$ **then**
      $pk.history \leftarrow pk.history \bigcup \{r\}$
      return $r(pk)$
  return $[(\mathtt{drop}, pk.h)]$

(b)

4.1 TAXONOMY CHART

This chart gives information about existing system and their functionalities

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	✓	✓	×	×	×
Network tomography of binary network	✓	✓	✓	×	×
All-pairs ping service for PlanetLab	✓	✓	×	×	×

5. CONCLUSION

Network administrator use primitive tools such as *Ping* and *traceroute*. My survey results indicate they are esager for more sophisticated tools. Other field of engineering indicate that desires are not unreasonable: For example, software design industries are buttressed by billion-dollar tool businesses that supply techniques for both static (e.g., design rule) and dynamic

(e.g., timing) verification. In fact, that ATPG was a well-known series of checking hardware chip testing, where it stands for Automatic Test Pattern Generation [4]. We hope network ATPG will be equally useful for automated dynamic testing of production networks.

ACKNOWLEDGMENT

I wish to express my sincere thanks to the guide and HOD Prof. Soumitra Das as well as our principal Dr. S.S. Sonawane , also Grateful thanks to our PG Coordinator Prof. Pankaj Agarkar and last but not least, the departmental staff members for their support.

REFERENCES

[1] *Zeng , Kazemian, Varghese,and Nick "Automatic Test Packet Generation",VOL. 22, NO. 2, APRIL 2014*
[2] *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*
[3] *N. Duffield, "Network tomography of binary network performance characteristics," IEEE Trans. Inf. Theory, vol. 52, no. 12, pp. 5373–5388, Dec. 2006.*
[4] *N. Duffield, F. L. Presti, V. Paxson, andD.Towsley,"Inferringlink loss using striped unicast probes," in Proc. IEEE INFOCOM, 2001, vol. 2, pp. 915–923.*
[5] *B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in Proc. Hotnets, 2010, pp. 19:1–19:6.*
[6] *"OnTimeMeasure," [Online]. Available: http://ontime.oar.net/*
[7] *"Open vSwitch," [Online]. Available: http://openvswitch.org/*
[8] *H. Weatherspoon, "All-pairs ping service for PlanetLab ceased," 2005 [Online]. Available: http://lists.planet-lab.org/pipermail/users/2005-July/001518.htm*
[9] *S. Shenker, "The future of networking, and the past of protocols," 2011 [Online].Available: http://opennetsummit.org/archives/oct11/shenker-tue.pdf*