Network Fault Detection Using Test Packet Generation: A Survey

Mrs. R. Radheesha
Research Scholar, Department of CSE
Bharath University, Chennai, India

Mr. A. R. Arunachalam
Research Guide, Department of cse,
Bharath University, Chennai, India

***Abstract*—** Networks are becoming larger and a lot of advanced, yet directors think about various tools like ping and traceroute to correct issues. Instead of using different tools to debug the network problems, we introduced an automatic and systematic scheme for testing and debugging networks known as Automatic test Packet Generation (ATPG). This automated approach fetches router configurations to generate a device-independent model. The model is employed to get a minimum set of test packets to analyse each link in the network. The detected failures trigger a separate mechanism to localize the fault by sporadically sending test packets. ATPG will notice each operational (e.g., incorrect firewall rule) and performance problems. ATPG complements however goes on the far side earlier add static checking (which cannot observe functional or performance faults) or fault localization (which solely localizes faults given liveness results).

***Keywords*—** *Network Monitor, Transfer Function, Header Space analysis, Forwarding Error, Fault Detection*

*****

I. .INTRODUCTION

When network comes into mind, the request that we get are about "How to protect your network? Is my network safe? What makes my network safer?" Network security do not have limitations by installing new firewall optimizing techniques or to protect the information, rather it could includes observing the packets, forwarding entries etc. Now, it will arise question of how this would help to safe the network. The response to this is, security can easily breached by tampering the rules and exploiting the faults.Troubleshooting a network is tough for 3 reasons. First, the forwarding state is distributed across multiple routers and firewalls and is outlined by their forwarding tables, filter rules, and alternative configuration parameters. Second, the forwarding state is difficult to look at as a result of it generally needs manually logging into each box up the network. Third, there are various different programs, protocols, and humans change the forwarding state at the same time.

The main contribution of this paper is what we tend to decision associate Automatic Test Packet Generation (ATPG) framework that mechanically generates a smallest set of packets to check the animateness of the underlying topology and therefore the congruousness between information plane state and configuration specifications. The tool also can mechanically generate packets to check performance assertions such as packet latency. 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, check packets are generated algorithmically from the device configuration files and FIBs, with the minimum range of packets needed for complete coverage. Check packets area unit fed into the network thus that every rule is exercised directly from the information plane. Organizations will customize ATPG to satisfy their needs.

The contributions of this paper square measure as follows:

1) A survey of system operators revealing common failures and root causes ;
2) A take a look at packet generation algorithmic rule;
3) A fault localization algorithmic rule to isolate faulty devices and rules;
4) ATPG use cases for practical and performance testing ;
5) Analysis of image ATPG system mistreatment rule sets collected from Stanford and Internet2 backbones

II. Network Model

ATPG uses the header house framework—a geometric model of how the packets area get unit processed. In header house, protocol-specific meanings associated with headers area unit ignored: A header is viewed as a flat sequence of ones and zeros. A header may be a purpose within the {0, 1} L house, wherever L is associate edge on header length. By victimization the header house framework, we obtain a unified, vendor-independent, and protocol-diagnostic model of the network2 that simplifies the packet generation method significantly. Let us get some familiar index words.

Packets: A packet is outlined by a (port header) tuple, where the port denotes a packet's position within the network at any time instant; every physical port within the network is allotted a singular number.

Switches: A switch transfer perform T, models a network device, like a switch or router. Every network device contains a set of forwarding rules that determine however packets square measure processed. An inbound packet is associated with precisely one rule by matching it against every rule out descending order of priority, and is born if no rule matches.

 ISSN: 2321-8169
1589 - 1592

Rules: A rule generates an inventory of 1 or a lot of output packets, relevant to the output port(s) to that the packet is shipped, and defines however packet field's square measure changed. The rule abstraction models all real-world rules we all know together with scientific discipline forwarding (modifies port, checksum, and TTL, however which is not scientific discipline address); VLAN and ACLs.

Bit	$b = 0\|1\|\mathrm{x}$
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)

Fig.1. Network model: Outline the definitions in our model.

Rule History: At any purpose, every packet includes a rule history: an ordered list of rules [r0,r1,…] the packet matched to date as it is exercised to the network. Rule histories are elementary to ATPG, as they supply the fundamental stuff from that ATPG constructs tests.

Topology: The topology transfer perform, , models the network topology by specifying that pairs of ports (Psrc,Pdst)square measure connected by links. Links are procedures that forward packets from two while not modification. If no topology procedure match associate input port, the port is a grip port, and therefore the packet has reached its destination.

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)

Fig.2. Network model: switch transfer function.

III. ATPG System Architecture

Based on the network model, ATPG generates the borderline number of take a look at packets in order that each forwarding decree the network is exercised and lined by a minimum of one take a look at packet. When an error is detected, ATPG uses a fault localization algorithmic program to determine the failing rules or links.

Fig. 3 is a system diagram of the ATPG system. The system 1st collects all the forwarding state from the network (step 1). This usually involves reading the FIBs, ACLs, and config files, as well as getting the topology. ATPG uses Header house Analysis [16] to reason reachability between the entire take a look at terminals (step 2). The result's then employed by the take a look at packet choice algorithmic program to reason a borderline 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-up is detected, the fault localization algorithm is invoked to slender down the explanation for the error (step 5).

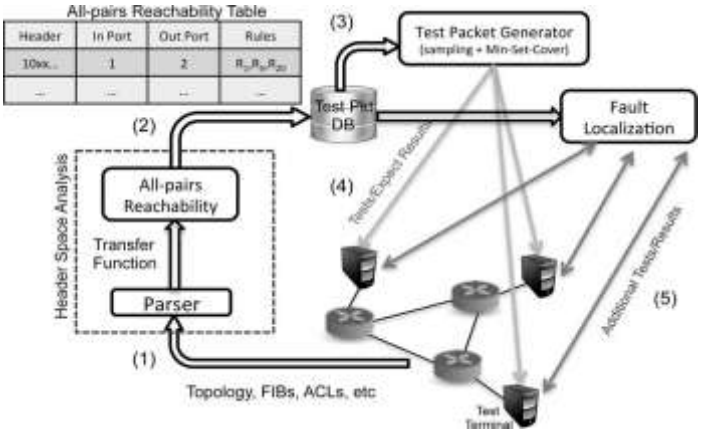

Fig. 3. ATPG block diagram

A. *Packet Generation*

1) Algorithm: we tend to assume a collection of check terminals within the network can send and receive check packets. Our goal is to come up with a set of check packets to exercise each rule each switch perform, so that any fault are going to be discovered by a minimum of one check packet. This is analogous to package check suites that attempt to check each attainable branch during a program. The broader goal will be restricted to testing every link or each queue. When generating check packets, ATPG should respect 2 key constraints: 1) Port: ATPG should solely use check terminals that are available; 2) Header: ATPG must solely use headers that every check terminal is permissible to send. As an example, the network administrator may solely enable employing a specific set of VLANs. Formally, we have the subsequent downside.

Problem 1 (Test Packet Selection): For a network with the switch functions,{T1,…,Tm} , and topology operate, , determine the minimum set of take a look at packets to exercise all accessible rules, subject to the port and header constraints. ATPG chooses check packets mistreatment associate formula we tend to decision check Packet choice (TPS). TPS 1st finds all equivalent categories between every try of accessible ports. Identical category may be a set of packets that exercises identical combination of rules. It then samples every category to settle on check packets, and eventually compresses the ensuing set of check packets to seek out the minimum covering set. When an error is detected, ATPG goes through the following steps:

 ISSN: 2321-8169
1589 - 1592

Step 1: *Generate All-Pairs Reachability Table*: ATPG starts by computing the whole set of packet headers that may be sent from every take a look at terminal to each alternative take a look at terminal. For each such header, ATPG finds the whole set of rules it exercises on the trail. To do so, ATPG applies the all-pairs reachability rule delineate. On each terminal Port, associate degree all- header is applied to the transfer operate of the primary switch connected to each take a look at terminal. Header constraints area unit applied here. As every packet Pk traverses the network victimization the network operate, the set of rules that match pk area unit recorded in pk. history. Working this for all pairs of terminal ports generates an all-pairs reachability table as shown in Table I.

Header	Ingress Port	Egress Port	Rule History
h_1	p_{11}	p_{12}	$[r_{11}, r_{12}, \ldots]$
h_2	p_{21}	p_{22}	$[r_{21}, r_{22}, \ldots]$
...	...	...	...
h_n	p_{n1}	p_{n2}	$[r_{n1}, r_{n2}, \ldots]$

Table I

All-pairs reachability table: all needed headers from every terminal to every other terminal, along with rules they exercise

Therefore all packets matching this class of header will examine the set of switch rules.

	Header	Ingress Port	Egress Port	Rule History
p_1	dst_ip=10.0/16, tcp=80	P_A	P_B	r_{A1}, r_{B3}, r_{B4}, link AB
p_2	dst_ip=10.1/16	P_A	P_C	r_{A2}, r_{C2}, link AC
p_3	dst_ip=10.2/16	P_B	P_A	r_{B2}, r_{A3}, link AB
p_4	dst_ip=10.1/16	P_B	P_C	r_{B2}, r_{C2}, link BC
p_5	dst_ip=10.2/16	P_C	P_A	r_{C1}, r_{A3}, link BC
(p_6)	dst_ip=10.2/16, tcp=80	P_C	P_B	r_{C1}, r_{B3}, r_{B4}, link BC

Step 2: *Sampling*: Next, ATPG picks a minimum of one check packet in associate equivalence category to exercise each (reachable) rule. The best theme is to at random choose one packet per class. This theme solely detects faults that all packets covered by identical rule expertise identical fault (e.g., a link failure). At the opposite end, if we would like to tend to sight faults specific to a header, then we'd like to pick out each header in each category.

Step 3: *Compression*: many of the take a look at packets picked in Step two exercise an equivalent rule. ATPG so selects a minimum subset of the packets chosen in Step two such the unions of their rule histories cowl all rules. The quilt is chosen to cover all links (for animateness only) or all router queues (for performance only) this can be the classical Min-Set-Cover downside. While NP-Hard, a greedy O (N2) algorithmic rule provides a decent approximation, wherever is that the variety of take a look at packets. We call the ensuing (approximately) minimum set of packets, the regular test packets.

IV. Implementation

Our algorithmic rule for pinpointing faulty rules assumes that a check packet can succeed given that it succeeds at each hop. For intuition, a succeeds only when all the forwarding Rules on the trail behave properly. Similarly, if a queue is congested, any packets that travel through it'll incur higher latency and will fail Associate in nursing end-to-end check. Formally, we've got the following.

Assumption 1 (Fault Propagation): R(pk) = 1 if and as long as Vr Є pk history, R(R,pk) = 1

ATPG pinpoints a faulty rule by 1st computing the token set of probably faulty rules. Formally, we've drawback a pair of. Problem a pair of (Fault Localization): Given an inventory of (pk0, R(pk0), (pk1, R(pk1),…tuples, realize all r that satisfies pki, R(pki, r) = 0.We solve this drawback opportunistically and in steps.

Step 1: contemplate the results from causation the regular take a look at packets. For each passing take a look at, place all rules they exercise into a set of passing rules . Similarly, for each failing take a look at, place all rules they exercise into a group of doubtless failing rules. By our assumption, one or additional of the foundations in area unit in error. Therefore, may be a set of suspect rules.

Step 2: ATPG next trims the set of suspect rules by weeding out properly operating rules. ATPG will this victimization the reserved packets (the packets eliminated by Min-Set-Cover). ATPG selects reserved packets whose rule histories contain specifically one rule from the suspect set and sends these packets. Suppose a reserved packet exercises solely rule the suspect set. If the sending of fails, ATPG infers that rule is in error; if passes, is far from the suspect set. ATPG repeats this method for each reserved packet chosen in Step two.

Step 3: In most cases, the suspect set is little enough when Step 2, that ATPG will terminate and report the suspect set. If needed, ATPG will slender down the suspect set more by Sending check packets that exercise 2 or a lot of the foundations within the suspect set victimization a similar technique underlying Step two. 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, then the strategy won't miss any faults, and so can haven't any false negatives.

V. CONCLUSION

In current System it uses a method that is neither exhaustive nor scalable. Though it reaches all pairs of edge nodes it could not detect faults in liveness properties. ATPG goes much further than liveness testing with same framework. ATPG could test for reachability policy (by checking all rules including drop rules) and performance measure (by associating performance measures such as latency and loss of test packets). Our implementation also enlarges testing with simple fault localization scheme also build using header space framework.

REFERENCES

[1] Hongyi Zeng, Member, IEEE, Peyman Kazemian, Member, IEEE, George Varghese, Member, IEEE, Fellow, ACM, and Nick McKeown, Fellow, IEEE, ACM, "Automatic Test Packet Generation".

[2] 2. "Ballani, H., and Francis, P. Complexity Oblivious Network Management: A step towards network manageability. Tech.Rep. cul.cis/TR2006-2026, Cornell University, Ithaca, NY, US,2006.

[3] Carsten Schmidt. Interface Tra_c Monitor Pro.http://software.ccschmidt.de/.

[4] Feamster, N., and Balakrishnan, H. Detecting BGP Configuration Faults with Static Analysis. In Proc. of 2nd Symp. on Networked Systems Design and Implementation (NSDI) (2005).

[5] A. Feldmann and J. Rexford. IP Network Configuration for Intradomain Traffic Engineering.In *IEEE Network Magazine*, 2001.

[6] T. V. Lakshman et al. The SoftRouter architecture. In *Proc. ACM HotNets Workshop*, 2004.

[7] Ratul Mahajan et al. Mutually Controlled Routing with Independent ISPs. In *Proc. NSDI*, 2007.

[8] Hung Xuan Nguyen, Patrick Thiran, "Active Measurement for Multiple link Failures Diagnosis in IP Networks", In Proceedings of the International Workshop on Passive and Active Network Measurement, pp. 185-194, 2004.

[9] Valent Carela-Espa, Pere Barlet-Ros, Albert Cabellos-Aparicio, "Analysis of the impact of sampling on NetFlow traffic classification", Computer Networks, vol.55, no.5, pp.1083-1099 ,2011.