Volume 2, Issue 12, December-2015, pp. 1142-1146 ISSN (O): 2349-7084

International Journal of Computer Engineering In Research Trends

Available online at: www.ijcert.org

# Automatic and Methodical approach for Test Packet Generation

[1]BEJJANKI PRADEEPKUMAR, [2] D.RAVI KIRAN

[1]*M.Tech (CSE), Priyadrshini Institute of Technology & Management*

[2]*Professor (Dept.of CSE), Priyadrshini Institute of Technology & Management*

**Abstract:** Recently networks are growing wide and more complex. However administrators use tools like ping and trace route to debug problems. Hence we proposed an automatic and Methodical approach for testing and debugging networks called Automatic Test Packet Generation (ATPG). This approach gets router configurations and generates a device-independent model. ATPG generate a few set of test packets to find every link in the network. Test packets are forwarded frequently and it detect failures to localize the fault. ATPG can detect both functional and performance (throughput, latency) problems. We found, less number of test packets is enough to test all rules in networks.

**Keywords:** Fault Localization, Test Packet Selection, Network Debugging, Automatic Test packet Generation (ATPG).

———————————— ◆ ————————————

## 1. INTRODUCTION

Detects and finding faults in differently and exhaustively testing all forwarding entries security rules and any packet processing rules in the network model generated algorithmically from the device configuration files with the minimum number of packets required for complete locations Test packets are different the network so that every conditions directly from the data sources its full coverage guarantees testing of every link in the network It can also be indicate to resurge a small set of packets that merely test every link for network likeness At end of in this basic form we feel that some different technique is fundamental to networks model of reacting to errors many network operators such as Internet2 proactively check the health of their network using pings between two of sources all-pairs guarantee testing of all links Networks generate larger and more different results in improper maintenance of the networks administrators rely on rudimentary tools like ping and trace route to debug errors despite it being getting lesser and lesser reliable It is notoriously hard to debug networks today network engineers wrestle with router miss configurations fiber cuts software errors and myriad of problems that cause the networks to different and fail completely An framework which is taken model testing and debugging networks automatically generates a minimal set of packets to exercise every link in the network as well as different rule in the network ATPG detects errors by different and indivgually testing all forwarding security firewall models and any packet processing model in the network test packets are generated algorithmically from the device configuration files based with the minimum number of data required for complete security It used for testing the insipient data of the underlying topology and the congruence between data plane state and configuration models It also complements work in static checking and errors localization The tool can automatically generate packets to test results assertions such as packet loss

## II. RELATED WORK

The test packets which generate automatically by configuration is not aware by earlier techniques. The very often related works we are familiar is offline tools which test invariants in networks. In control plane, NICE [7] tries to comprehensively cover code path symbolically in a controller applications with support of simplified switch and host models. In the data plane, Anteater [25] models invariants as a Boolean satisfiability problem which tests them against configurations with a SAT solver. Header Space Analysis [16] use geometric model for checking reachability, detecting loops, and for verifying slicing. Recently, SOFT [1] put forward to check uniformity between different Open Flow agent implementations which is responsible for bridging control and data planes in SDN context. ATPG supplement these checkers directly by verifying the data plane and exercising a important set of dynamic or performance errors which could not be captured. The major contribution of ATPG is not fault localization, but deciding a compact set of end-to-end measurements which could exercise every rule and every link. The mapping in between Min-Set-Cover and network monitoring was been explored previously in [3] and [5]. ATPG progress the detection granularity to rule level by working router configuration and data plane information. ATPG not limited to liveness testing, but it can be applicable for checking higher level properties like performance. Our work was closely related to work in programming languages and symbolic debugging. We made a preliminary tries to use KLEE [6] and find it to be 10 times slower than the unoptimized header space framework. We speculate this is basically because in our framework we directly simulate the forward path of a packet in addition of solving constraints using an SMT solver. However, more work is needed to understand the differences and potential opportunities.

## II. EXISTING SYSTEM

Testing liveness of a network is a fundamental problem for ISPs and large data center operators. Sending probes between every pair of edge ports is neither exhaustive nor scalable . It suffices to find a minimal set of end-to-end packets that traverse each link. However, doing this requires a way of abstracting across device specific configuration files, generating headers and the links they reach, and finally determining a minimum set of test packets (Min-Set-Cover). To check enforcing consistency between policy and the configuration.

**1. Packets**
A packet is taken by number of the denotes a packet's places in the network at any time and locations Instant; every physical node in the network is assigned a same id number.

**2. Switches** A switch transfer function models is taken a network device such as a switch Each network device contains a set of forwarding conditions that determine packets are processed An arriving packet is associated with different rule by matching it against each rule in descending order in different locations and is dropped if no rule matches in different models.

**3. Rules** A rule generates a list of one or more output packets corresponding to the output port to which the packet is sent and defines how packet fields are changed The rule abstraction models all real world conditions we know including IP forwarding and ACLs Essentially a rule defines how a region of header space at the ingress is transformed into regions of header space at the egress

**4. Rule History** Take any node each packet has conditions history and ordered list of rules the packet matched and traveling the network. Rule histories are fundamental problems provided provide the basic raw material

**5. Topology** The topology different function models the network models by specifying which pairs of ports and links are rules that forward packets from to without changing. If no models rules match an input port the port is an edge port and the packet has reached its destination

**A. Test Packet Generation** Algorithm We take a set of test nodes in the network model send and receive test packets Our aim is to generate a set of test packets and changes every conditions in every switch objects so that every fault will be observed by at least one test packet This is scanner type software test models that try to test every possible branch models The broader goal can be limited to testing every link every queue When generating test packets ATPG must respect two key constraints **(1) Port:** ATPG must be test terminals that are available **(2) Header:** ATPG must be use headers that each test terminal is permitted to send For the network administrator may only allow using a specific model of VLAN

```
GOOGLE-QUERY-DELAY($tcp\_segments$)
  $s \leftarrow tcp\_segments[1]$
  $p \leftarrow s$
  $c \leftarrow tcp\_segments[2]$
  while $c \neq NIL$
     do if $c.snd\_time > p.snd\_time$ and
           $c.snd\_time > p.ack\_time$
           then return ($s.snd\_time, c.snd\_time$)
           else
               if $c.size < MSS$
                  then return ($s.snd\_time, c.snd\_time$)
        $p \leftarrow c$
        $c \leftarrow c.next$

```

Fig. 2: Segments Methods Test packets are taken into the network in which that every rule is covered directly from the data plane with different locations treats links like normal forwarding conditions its full coverage provides testing of every link in the network model It can also be specialized to form a minimal no of of packets that obviously test every link for network likeness At least one basic form we would feel that ATPG 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 all-pairs of network provide testing of all links and has been found for large networks model such as Planet Lab.

## IV. PROPOSED SYSTEM

Contender framework generates minimum no of packets automatically to debug the false occurring in the network model This tool could automatically generate packets for checking performance assertions such as like packet loss finds and determines errors by independently testing all forwarding entries any packet processing rules and security models in network test packets are generated algorithmically from device configuration files and from FIBs which requires minimum number of packets for complete coverage Test packets are fed into the network in which that every rule is covered directly from the data plane Since treats links like normal forwarding conditions its full coverage provides testing of every link in the network model It can also best specialized to form a minimal set of packets that obviously test every link for network likeness At least in this basic form, we would feel that some different technique is fundamental to networks Instead of reacting to failures many network operators such as proactively check the health of their network using pings between all pairs of sources all-pairs does not provide testing of all links and has been found to be unsalable for large networks such as Planet Lab

## V. METHODOLOGY

The proposed system can be divided into following modules: 1. Failures and root causes of network operators 2. Data plane analysis 3. Network troubleshooting 4. ATPG system 5. Network Monitor

**1. Failure and Root Causes of Network Operators** Network traffic is represented to a specific queue in router but these packets are drizzled because the rate of token bucket low It is difficult to troubleshoot a network for three different models First the forwarding state is shared to multiple routers and security and is determined by the forwarding data filter conditions and configuration parameters Second the forwarding state is difficult to watch because it requires manually logging into every box in the network model Third the forwarding state is edited simultaneously by different programs protocols and humans.

**2. Data Plane Analysis** Automatic Test Packet Generation framework which automatically generates a minimum set of packets to check the likeness of underlying network models and congruence different data plane state and configuration specifications These model can automatically generate packets to test performance assertions like packet latency ATPG find faults by independently and exhaustively checking all security rules forwarding entries and packet processing conditions in network. The test packets are generated algorithmically from the device configuration different files and FIBs, with less number of packets needed for whole coverage Test packets are fed in the network so that every rule is covered directly from the data plane This tool can be customized to check only for reach ability or for its performance

**3. Network Troubleshooting**

The cost of network debugging is captured by two metrics one is the number of network-related tickets per month and another is the average time taken to resolve a ticket there are 35% of networks which generate more than 100 tickets per month. Of the respondents, 40.4% estimate takes under 30 minutes to resolve a ticket if asked what the ideal tool for network debugging it is would be, 70.7% reports automatic test generation to check performance and correctness. Some of them added a desire for long running tests to find jitter or intermittent real-time link capacity monitoring and monitoring tools for network state. In short, while our survey is small, it helps the hypothesis that network administrators face complicated symptoms and causes.

**4. ATPG Systems**

Depending on network model ATPG generates less number of test packets so that every forwarding rule is exercised and covered by at least one test packet When an error is found, ATPG use different localization algorithm to ascertain the failing rules in network model

**5. Network Monitor**

To send and receive test data packet network monitor assumes special test agents in the network The network monitor gets the database and builds test packets and instructs each different to send the proper packets Recently test agents partition test packets by IP Proto field and TCP/UDP port number but other fields like IP option can be used If any tests fail the monitor chooses extra test packets from booked packets to find the faults The process gets repeated till the fault has been identified To communicate with test agents monitor uses and SQL it string matching to lookup test packets efficiently

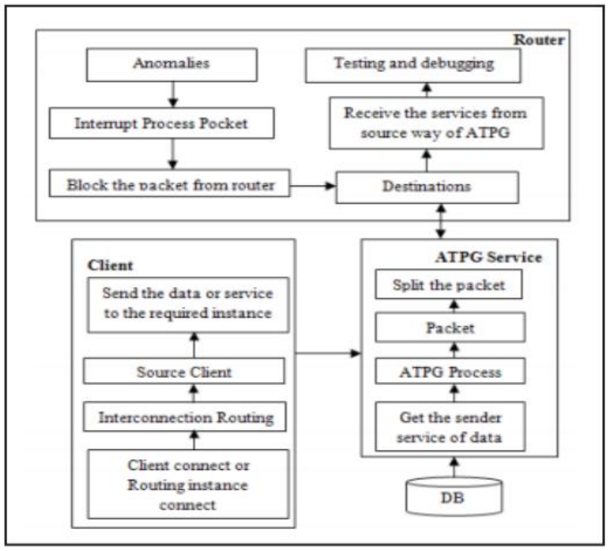
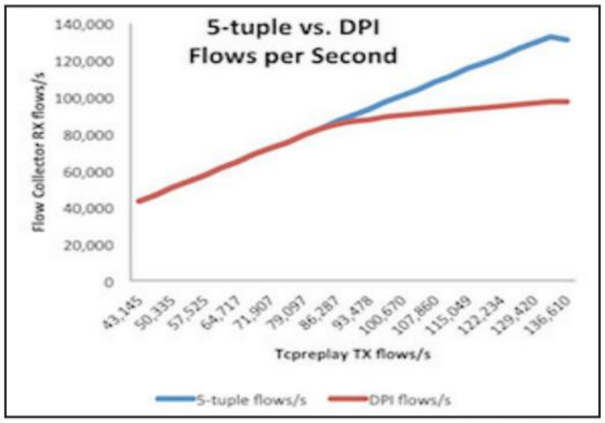

Router
Anomalies
Interrupt Process Pocket
Block the packet from router
Testing and debugging
Receive the services from source way of ATPG
Destinations
Client
Send the data or service to the required instance
Source Client
Interconnection Routing
Client connect or Routing instance connect
ATPG Service
Split the packet
Packet
ATPG Process
Get the sender service of data
DB

Fig. 2. ATPG system block diagram

## VI. PERFORMANCE

The principal component overhead for ATPG are polling the network periodically for forwarding state and performing two reachable While one can reduce overhead by running the offline ATPG calculation less frequently this runs the risk of using out-ofdate forwarding information we reduce overhead in two ways First we have recently fast up the all-pairs reach ability calculation using a fast multithreaded. Second, instead of extracting the complete network state every time ATPG is triggered an incremental state updater can significantly reduce both the retrieval time and the time to calculate reach ability We are working on a real life version of ATPG that incorporates both techniques Test agents within terminals incur negligible overhead because they merely de multiplex test packets addressed to their IP address at a modest rate compared to the link speeds gb most modern CPUs are capable taken.



## V. CONCLUSION

In current System it uses a method that is neither exhaustive nor scalable different it reaches all pairs of edge nodes it could take detect faults in likeness properties ATPG goes much further than likeness testing with different framework ATPG could test for reach ability model and performance methods Our implementation also enlarges testing with simple errors localization scheme also build using header space framework

## REFERENCES

[1] Hongyi Zeng, Peyman Kazemian, George Varghese, ACM, and Nick McKeown, ACM,“Automatic Test Packet Generation”.

[2] M. Jain, C. Dovrolis,“End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput”, IEEE/ACM Trans. Netw., Vol. 11, No. 4, pp. 537–549, Aug. 2003.

[3] Kompella, R. R., Greenberg, A., Rexford, J., Snoeren, A. C., Yates, J. Cross-layer Visibility as a Service", In Proc. of fourth workshop on Hot Topics in Networks (HotNet-IV) (2005)

[4] D. Maltz, G. Xie, J. Zhan, H. Zhang, G. Hjalmtysson, A. Greenberg,"Routing design in operational networks: A look from the inside", In Proc. ACM SIGCOMM, 2004.

[5] Mark Stemm, Randy Katz, Srinivasan Seshan,“A network measurement architecture for adaptive applications”, In Proceedings of the nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, pp. 285 - 294, 2000.

[6] Verma, D.,"Simplifying Network Administration using Policy based Management".

[7] P. Yalagandula, P. Sharma, S. Banerjee, S. Basu, S.-J. Lee, “S3: A scalable sensing service for monitoring large networked systems".

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest,"Introduction To Algorithms", Eastern Economy Edition.New Delhi: Prentice-Hall of India Private Limited 1998.

[9] E Balagurusamy,"Programming in ANSI C", Second edition. New Delhi: Tata McGraw-Hill, 1997.

[10] Deitel & Deitel, C++ How To Program, Second Edition, New Jersey, NJ Prentice Hall, 1997

[11] Tom Christiansen, Nathn Torkington,"Perl Cookbook: Solutions and Examples For Perl Programmers", 1st Edition, Sebastopol, CA: O’Reilly & Associates, Inc., 1999.

[12] Nathan Patwardhan, Ellen Siever and Stephen Spainhour, PERL In A Nutshell, 2nd Edition, Sebastopol, CA: O’Reilly & Associates, Inc., 2002.

[13] Tom Milligan, “Using Perl With Rational ClearCase Automation Library”, 2004Retrieved Nov10, 2010 from