A Capacious Scrutiny on Automatic Test Packet Generation

Shastri Saumit C.
M Tech Student,
CSE Department, SGGSIE&T,
Nanded-431606.

U. V. Kulkarni, Ph.D
Professor,
CSE Department, SGGSIE&T,
Nanded-431606.

ABSTRACT
These days networks are not getting any smaller, they are increasing in size and it is becoming tedious job for network administrators to debug the network, since they rely on traditional tools such as ping and traceroute for this job. This paper puts forward an automated and systematic approach to test and debug a network called Automatic Test Packet Generation (ATPG). ATPG produces a model which is not dependent on devices after reading configuration from routers. The model is used to generate minimum number of test packets to cover every link in a network and each rule in network. ATPG is capable of investigating both functional and performance problems. Test packets are sent at regular intervals and separate technique is used to localize faults. The working of few offline tools which automatically generate test packets are also given, but ATPG goes beyond the earlier work in static checking (Checking liveness and fault localization).

General Terms
Header space analysis, Test packet generation algorithm, Fault localization algorithm.

Keywords
ATPG system, TPS algorithm.

1. INTRODUCTION
It is not at all easy task to debug a network. The network administrators face problems like router misconfiguration, Fiber cut, mislabeled cables, software bug, Faulty interfaces etc. Network administrators try to overcome these problems using mostly used tools such as ping and trace route. Debugging networks is getting more and more difficult as not only size of networks but also their level of complexity is increasing day by day. Let us consider few examples of different types of problems network administrators face in day to day life.

Consider a router with a line card having a fault, so that it silently drops test packets, as a result, many users straggling for connections complain to network administrator. Now if that administrator is administrating 100 routers he has to go to each router to see if configuration is not altered, and if the answer is no, he uses his knowledge of topology to search faulty device using techniques like ping and trace route [1].

Consider another example where video traffic is put in a particular queue, and token bucket ratio is low is the reason why packets are dropped. Such performance faults are not possible for network administrators to investigate [1].

To make out what difficulties network administrators face and at present how they overcome these difficulties, a survey is made in 2011.All responses to that survey is given in [2].From the survey it is clear that administrators have to fight with complex symptoms and causes. Many problems associated with networks occur frequently and it takes much time to come out of them, so the cost of debugging a network becomesinsignificant. Pure tools like ping and trace route are largely used, but now network administrators wish more refined tools.

This paper put forward an automated and systematic approach to test and debug a network called Automatic Test Packet Generation (ATPG). ATPG produces a model which is not dependent on devices after reading configuration from routers. Another advantage of ATPG system is that it covers each link and every rule in network with minimum number of test packets. Uniformly the test packets are send, and if any fault is detected, it is triggered by separate mechanism namely fault localization. ATPG can solve both of the above problems, hence it can cover both functional and performance faults [1].

The figure 1 is uncomplicated view of network state. In lower half of the figure there is forwarding table. The function of forwarding table is to forward each packet. Packet is consisting of forwarding information base (FIB), access control lists etc. It is control plane which writes forwarding state.

Figure 1 can be decomposed in three parts as A, B and C. We can consider the policy (A), which is compiled by controller into configuration files which are device specific (B), which then shows the forwarding behavior of every packet (C). To ensure the network behaves as per requirement, all the three steps at all times should remain consistent, that is same as A=B=C. At the same time, the topology, shown at the bottom right in the figure, should also be able to satisfy a set of liveness properties shown by L [1].

It is not too long ago when scientists come up with tools showing compactness between policies and configuration files A=B [3], [4], [5], [6], but these tools can't deal with performance problems which requires checking of liveness property L or B=C. ATPG can do this job efficiently [1].

The outline for the rest of the paper is as given below.

1) First take a look at some earlier works related to automatic test packet generation, some offline tools.

2) Followed by Header Space Analysis [4] used in ATPG system.

3) Next ATPG System [1] is presented for readers

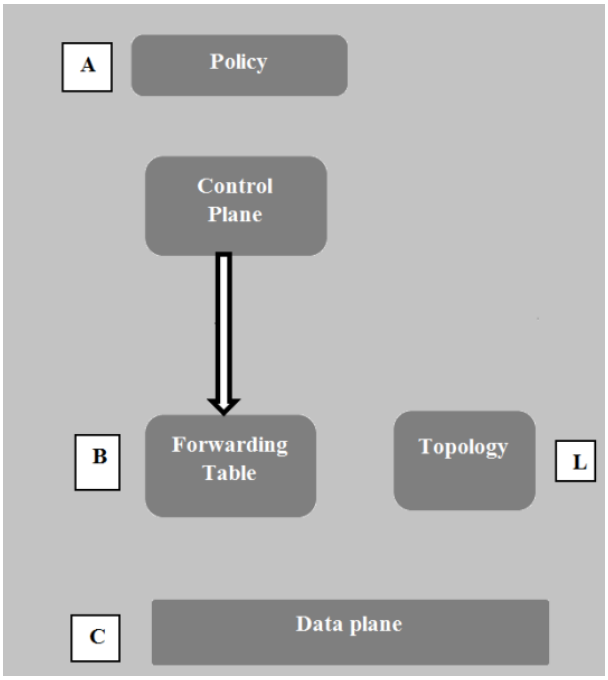

Fig 1: Network state

2. RELATED WORK

In this section some of the earlier techniques used for automatically generating test packets are given. Nearest technologies known are few offline tools.

2.1 Offline Tools Supporting Automatic Test Packet Generation

One of the offline tools which have been used for generating test packets automatically in control plane is NICE [3]. NICE stands for no bugs in controller execution. NICE is an offline tool, which brings the bugs in controller program to user's notice more efficiently with the help of model checking and symbolic execution in open flow system. Working with open flow system programmer have to deal with challenges like large space of switch state, large space of input packets, large space of event ordering etc. To overcome these challenges NICE [3] is of good use.

Working of NICE is shown in Figure 2. NICE programmer has to supply controller program along with topology of network which consist of specification of switches and hosts. The programmer can ask NICE for general correctness of properties such as, program is not having any forwarding loop or program is without any black holes. The NICE according to fixed plan looks into the possible system behavior and checks it with correctness properties supplied by the programmer. The programmer has the freedom to configure search strategy which is desired by him. Finally NICE gives the traces of property violation or properties which are not up to the mark with their indications as output [3].

The tool NICE works in control plane similarly in the data plane there is another offline tool that can be used namely Anteater [5]. Anteater gathers the network topology and forwarding information bases (FIBs) of devices, and describes them as boolean functions. Then an error to be checked is specified by operator against the network, such errors can be consistency of forwarding rules between routers, reachability or loop free forwarding. Anteater makes the combination of these errors and converts them into examples of Boolean satisfiability problem (SAT), and makes use of a SAT solver to perform analysis. If the network state disobeys an invariant, Anteater provides a specific counterexample, for instance a packet header, FIB entries, and path thatbrings about the potential bug.

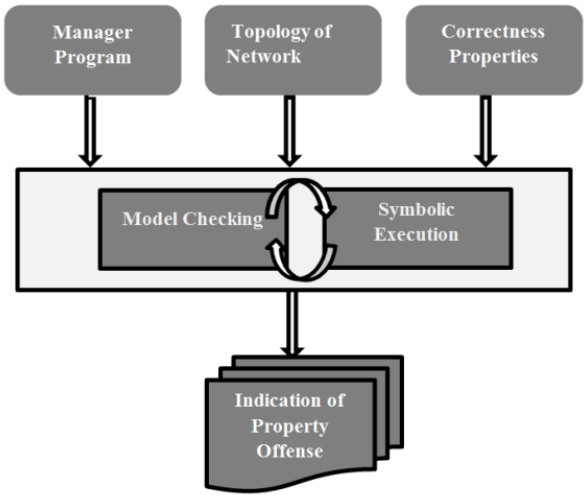

Fig 2:No bugs In Controller Execution (NICE)

Anteater finds errors through various steps. First of all, Anteater gathers the contents of FIBs from networking equipment through terminals, SNMP, or control sessions maintained to routers. These FIBs can be either simple IP longest prefix match rules, or more complex actions like access control lists or modifications of the packet header. Secondly, the operator forms new invariants or selects from a menu of standard invariants are to be checked against the network. This can be done via bindings in Ruby or in a declarative language that we designed to reorganize the expression of invariants. Third action is that, Anteater interprets both the FIBs and invariants into examples of SAT, which are resolved by SAT solver. Lastly, if the results from the SAT solver show that the provided invariants are violated, Anteater will obtain a counterexample to support recognition [5].

Only a short time ago researchers have come up with SOFT [7] used to prove the uniformity between various open flow agents which are liable for linking control and data plane in the context of SDN [1].

2.2 Other Related Works

Since long time, to examine faults in networks they are examined on end to end basic. Currently researchers are passionate about mining inferior quality unorganized data for example, we can consider router configuration and network tickets. On the other hand, main offering of ATPG system is giving dense set of end to end estimate that can occupy each rule or every link, and not just fault localization [1].

Many examiners have come with different measurement kindly schemas [8], [11], [9], and [10]. Our approach is additional to all these. Group by input along with port compulsions ATPG produces test packets and point of injection for these packets with the help of distribution of estimate devices [1].

2.3 Header Space Analysis

The automatic test packet generation uses the framework of Header space analysis [4], in which it uses a geometric model, which allows the ATPG system to statistically check the

network specifications and configurations to deal with important classes of failures such as forwarding loops, reachability failures, traffic isolation and linkage problem.

Another advantage of header space analysis is capability to do slicing. Slicing assures isolation between system hosts, users or traffic.

Consider virtual LAN as an example of slicing. Once the virtual LAN is configured correctly it gives guarantee thattraffic from one slice cannot leak into other slice, so it provides more security. In this example slice is virtual LAN.

At the same time by using geometric model of header space analysis, after enabling the static analysis of network sliced in more general way the opinion of isolation can be taken further [4].

A slice is made up of number of different header fields and consisting of topology of number of switches and links. There is set of headers on each link and its share of capacity corresponding to each header. Each slice has the separate control plane, and it is up to its owner to decide how packets are routed and processed in that slice.

In header space, the meaning of header which is specific to protocols in not taken into account: A header is seen as unbroken arrangement of binary representation i.e. zeros and ones. A header is a point and flow can be seen as region in a set containing 0 and 1 as elements, that is{0, 1} to the power L space where, L is upper limit on length of header. By making use of header space framework one can achieve new, vector free and protocol unbeliever model of network which facilitate the process of packet generation by a great deal [4].

3. NETWORK DESIGN

As mentioned in the last section, the automatic test packet generation (ATPG) system makes use of geometric model of header space analysis [4]. This section explains some of the key terms associated with geometric framework of header space analysis.

3.1 Packet

Packet in a network can be described as a tuple of the form (port, header) in such a way that, it is the job of port to show position of packet in a network at instantaneous time. Each one of the port is allotted with one and only one unique number [1].

3.2 Switch

Another term used in geometric model of header space analysis is switches. It is the job of switch transfer Function T, to model devices in a network. Example of devices can be switches or routers. There is a set of forwarding rules contained in each device, which decides how the packets should be processed. When a packet comes at a switch, a switch transfer function comperes it with each rule in descending order of priority. If packet does not match withany of the rule then it is dropped. Each incoming packet is coupled with exactly single rule [1].

3.3 Rules

Piece of work for rules is generation of list of one or more output packets associated with those output ports to which the packet is transferred, and explain how fields of port are modified. In other words, rules explains how the region of header space at entrance in changed into region of header space at exit [1].

3.4 Rule History

At any moment, every packet has its own rule history, which can be described as ordered list of rules packet have matched up to that point as it covers the network. Rule history provides necessary and important unprocessed material for automatic test packet generation (ATPG). That is the reason why it is fundamental to ATPG [1].

3.5 Topology

The network topology is modeled by topology transfer function. The topology transfer function gives the specification about which two ports are joined by links. Links are nothing but rules that forwards a packet from source to destination with no modification. If there is not a single topology rule matching an input port, the port is situated at edge of a network and packet has come to its desired destination [1].

3.6 Life of a Packet

One can see life of a packet as carrying out or executing switch transfer function and topology transfer function at length. When a particular packet comes in a network port p, firstly a switch function is applied to that packet. Switch transfer function also contains input port pk.p of that packet. The result of applying switch function is list of new packets [pk1, pk2, pk3,]. If the packet reached its destination it is recorded, and if that is not the case, topology transfer function is used to call upon switch function of new port. This process is done again and again unless packet is at its destination [1].

4. ATPG THEORY

Stand on the system standard analyzed above; Automatic test packet generation system makes use of least possible number of test packets to study whole forwarding rules in a network, on the condition that each forwarding rule is capped by at least one test packet. When the fault is encountered, ATPG is equipped with fault localization algorithm to resolve the declining rules or links.

Figure 3 represents the work flow of automatic test packet generation (ATPG) system.

1) The ATPG system begins by gathering forwarding state from network, which is represented as first step in the figure. Work covered in this step is normally not only retrieving topology of network but also learning forwarding information base and configuration files etc.

2) The second step follows the first, in which header space analysis is used by ATPG system to figure out scope of each terminal.

3) The outcome of second step is taken as input by test packet generation algorithm to gauge smallest number of test packets sufficient to test all rules. This completes third step.

4) These test packets are sent regularly by the test terminals as a penultimate step.

5) Lastly, if an error is disclosed ATPG appeals to fault localization algorithm to curtail root of error [1].

Readers can see other version of figure 3 in figure 5 given in [1].

4.1 Origination of Test Packets

The ATPG system can be roughly divided into two parts namely test packet generation and fault localization. While developing an algorithm for test packet generation a supposition is that, set of test terminals may transmit or take

in test packets. The target for algorithm is generating minimum number of test packets to practice every rule in every switch function, as a result if a fault occurs, it will be watched by at least one test packet. ATPG system makes use of test packets selection algorithm (TPS) to generate test packets.

ATPG must only make use of test terminals that are available and ATPG must utilize headers that each test terminal is authorized to send are two important restrictions of which ATPG must take a notice of at the time of generating test packets.

1) ATPG system begins by estimating entire set of test packet headers that can be forwarded from each test terminal to every other test terminal. ATPG achieves this by detecting full set of rules it can work out in entire journey. Thus, ATPG refers to all pair reachability algorithm [4] to perform this task.

2) Afterwards, ATPG selects greater than or equal to one test packet from identical class of test packets to use every rule which is within reachable distance. Automatic test packet generation can complete this with ease by haphazardly selecting single packet in each class. This method is capable of finding only those faults for which all packets screened by same rule suffer the same fault. Example of such faults includes link failure. On the other hand if someone desired to find out faults which are particular to headers, then he has to select every header in every class. This process is called sampling.

3) Lastly in the process of generating test packets ATPG goes to compression. Most of the times while using test packet selection algorithm there come situation such that same rule can be used by numerous test packets. Consequently ATPG chooses smallest family of test packets selected in above step in such a way that alliance of their rule histories cover total rules [1].

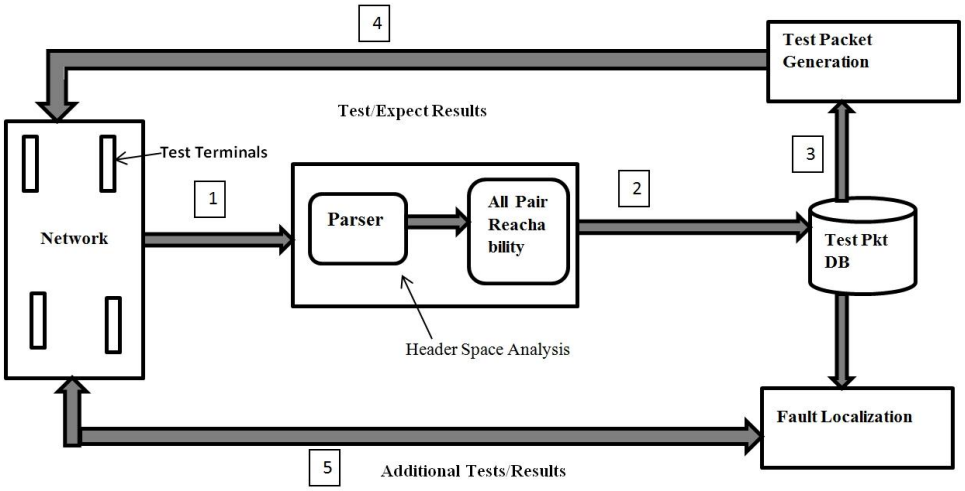

Fig 3: Working of Automatic Test Packet Generation

4.2 Error Fixing

ATPG sends a set of test packets at regular intervals. If in case test packets fail to reach their desired target, ATPG is capable of identifying errors that induced the problem.

If watched performance of a rule is not the same as its normal behavior then a rule is neglected, in other words it fails.

ATPG monitors where rules fail by applying a result function R on rule r in a packet pk. A result function takes value 1 if packet pk follows rule r, if not it takes value 0.

A forwarding of a rule fails if a test packet is not provided to its planned output port on other extreme.Forwarding of a rule is successful if either a test packet is provided to its planned output port, or in case it is a drop rule, it is addressed rightly ifit is dropped. A link collapse can be characterized by failure of forwarding rule in the topology function. Further, if output link is jammed, failure can be determined by waiting time of a test packet going above a threshold.

Algorithm that discovers defective rules is described below, makes hypothesis that a test packet will prosper if and only if it succeeds at every short trip.

1) Let's begin by thinking about the outcome of steadily sending the test packets. For each passing test, put all the rules used by them in a set of passing rules P. Likewise, for each failing test put all the rules used by them in a set of probably failing rules F. As per hypothesis, minimally one or more of the rules in F are defective. Hence, F-P becomes set of suspect rules.

2) Next responsibility of ATPG is to reduce the size of set of suspect rules by clearing out that rule in suspect set, which is working properly. ATPG fulfills this responsibility with the help of reserved packets as given below. Reserved packets are those packets which are eliminated by ATPG, during selection of minimum number of test packets to cover each rule in a network.

a) ATPG picks those reserved packets from the set of suspect rules whose rule histories hold one and only one rule and transmit these packets.

b) Suppose that a reserved packet p contains only one rule r which is also in a suspect set. If transmitting reserved packet p is not successful, ATPG concludes that rule r is in error. Otherwise if transmitting reserved packet p is successful, rule r is removed from suspect set.

c) ATPG goes on repeating this process for each reserved packet selected in this step.

3) Most of the times, after second step suspect set become much smaller that ATPG is finished with suspect set. If situation demands, ATPG can cut down suspect set further by transmitting those reserved packets that exercise two or more rules contained in suspect set in same way as mentioned above in second step. If transmitting reserved packets is successful ATPG comes to the conclusion that none of the used rules are defective and removes them from suspect set [1].

4.3 Superiorities of ATPG System

These are some advantages of using automatic test packet generation system above conventional tools as given below

1) The set of test packets generated in ATPG system can cover each reachable rule in a network, taking into account all port and headers restrictions.

2) By making use of test packet selection algorithm ATPG generates minimumnumber of test packets to cover every link and each reachable rule in a network.

3) The time complexity of ATPG has polynomial runtime.

4) To find faulty test terminals with its rule as well as configurations, fault localization algorithm is used in ATPG.

5) With the help of ATPG system exactness can be improved by testing functional as well as performance problems [12].

5. CONCLUSION

Network supervisors these days mostly depend on old tools such as ping and traceroute to correct a network. It is observed that they are in need of more refined tool for this work.

In day to day life, internet service providers as well as big data center operators face problems in testing liveness of a network. On the other hand, conducting tests between each pair of border ports is not only incomplete but also unappreciable. One can come out of this problem by digesting on device specific configuration files, creating headers and links reached by them. Lastly finding least number of test packets to cover each link. To overcome all these problems require method like ATPG.

By testing all rules inclusive of all drop rules ATPG is able to test reachability strategy. That is not all; by using performance scales such as delay and loss of test packets ATPG can calculate performance soundness of a network. ATPG uses simple fault localization method constructed with the help of header space analysis [4] to localize faults. Regular model of ATPG system helps to cover maximum links or rules in a network with minimum number of test packets.

6. FUTURE SCOPE

ATPG provides better solution for network organizers hanging on old tools for computing a network. ATPG has a favorable future opportunity considering,ATPG is blessed with ascendency of overcoming not only functional but also execution blemish.

Combined with all these upper hands there are few issues which remained to be addressed in the future such as, ATPG cannot cope with routers with a change in internal state; ATPG has restriction that it can rightly model rules only when parameters along hash function are known; Another problem is of dealing with rules that are out of sight; ATPG fails to reveal errors which exists for time less than time between two consecutive tests; While using sampling at times ATPG can fail to reach some flaws.

7. ACKNOWLEDGMENTS

Authors are grateful to SGGSIE&T College for its encouragement and support to write this paper. Authors also like to thank authors of all the papers which have been referred in writing this paper.

8. REFERENCES

[1] Hongyi Zeng, Peyman Kazemian, George Varghese and Nick McKeown," Automatic Test Packet Generation,"IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 22, NO. 2, APRIL 2014 [Online].Available:http://yuba.stanford.edu/~nickm/papers/atpg-ton.pdf

[2] "Troubleshooting the network survey," 2012 [Online]. Avaible:http://eastzone.github.com/atpg/docs/NetDebugSurvey.pdf

[3] 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[Online].Available:http://infoscience.epfl.ch/record/170618/files/nsdi-final

[4] P. Kazemian, G. Varghese, and N. McKeown, "Header space analysis: Static checking for networks," in Proc. NSDI, 2012, pp. 9–9.

[5] H. Mai, A. Khurshid, R. Agarwal, M. Caesar, P. B. Godfrey, and S. T. King, "Debugging the data plane with Anteater," Comput. Commun. Rev., vol. 41, no. 4, pp. 290–301, Aug. 2011.

[6] M.Reitblatt,N.Foster, J. Rexford, C. Schlesinger, and D.Walker, "Abstractions for network update," in Proc. ACM SIGCOMM, 2012, pp. 323–334.

[7] M. Kuzniar, P. Peresini, M. Canini, D. Venzano, and D. Kostic, "A SOFT way for OpenFlow switch interoperability testing," in Proc. ACM CoNEXT, 2012, pp. 265–276.

[8] 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.

[9] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iplane: An information plane for distributed services," in Proc. OSDI, Berkeley, CA, USA, 2006, pp. 367-380.

[10] P. Yalagandula, P. Sharma, S. Banerjee, S. Basu, and S.-J. Lee, "S3: A scalable sensing service for monitoring large networked systems," in Proc. INM, 2006, pp. 71–76.

[11] "OnTimeMeasure,"[Online].Available: http://ontime.oar.net/

[12] Shrikant B. Chavan, Soumitra Das,"Review paper of Automatic Test Packet Generation and Fault Localization," Multidisciplinary Journal of Research in Engineering and Technology, Volume 2, Issue 2, Pg.419-423.