AUTOMATED TEST PACKET GENERATION

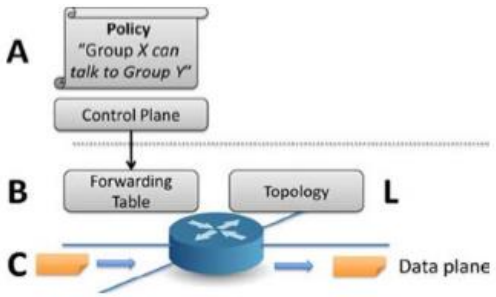
[#1]**Mrs.SHAGUFTHA BASHEER,** *Assistant Professor*
[#2]**Ms.KAMARAPU JEEVITHA,** *Assistant Professor*
Department of Computer Science and Engineering,
SREE CHAITANYA INSTITUTE OF TECHNOLOGICAL SCIENCES, KARIMNAGAR, TS.

ABSTRACT:
This study presents the ground-breaking Automatic Test Packet Generation (ATPG) technique, which offers an automated and methodical method for testing and debugging network problems. The ATPG tool's ability to parse and analyze router properties enables the building of models that are independent of any particular device in use. The model is utilized to provide either a minimal collection of test packets capable of testing each network connection in an effective manner or a maximal set of test packets capable of testing each network rule in an exhaustive manner. Both of these options are made available to the user. Regular test packets are transmitted, and mistakes are found by triggering a process that is intended to pinpoint the origin of the problem. ATPG, which stands for Automatic Test Pattern Generation, has the ability to identify performance concerns such as backed-up queues. Existing methods of static testing are not only improved by the ATPG technique, but they are also surpassed by this approach in terms of locating liveness and performance issues. On the other hand, approaches to defect localization rely heavily on the findings of liveness analysis in order to identify problems in the system. In this work, we give a comprehensive analysis of our prototype Automatic Test Pattern Generation (ATPG) implementation. ATPG stands for Automatic Test Pattern Generation. In order to evaluate how effective our system is, we make use of the backbone network at Stanford University as well as Internet2. It has come to light that just a relatively small number of test packets are required in order to adequately analyze each rule in these networks. It has been discovered, for instance, that 54 packets are sufficient to cover all connections in the Stanford backbone network, however 4,000 packets are necessary to cover all rules. This discrepancy is due to the fact that covering all rules requires more packets. When 4000 test packets are transmitted 10 times per second, less than one percent of the bandwidth that is available on the network is utilised. Both the ATPG code and the associated data collections can be accessed by anybody interested.

Index Terms—Dataplane analysis, network trouble shooting, test packet generation.
DOI Number: 10.48047/nq.2021.19.11.NQ21267 **NeuroQuantology 2021; 19(11): 619-622**

1. INTRODUCTION

Debugging a network is typically considered to be a challenging undertaking. Network engineers confront a variety of challenges on a regular basis, some of which include misconfigured routers, fiber breaks, damaged interfaces, cables with wrong labeling, unreliable software, intermittent connection, and other problems that can cause networks to malfunction or fail entirely. The method was evaluated using data from two different real-world datasets, namely the backbone network of Stanford University in Stanford, California, and Internet2, which, respectively, are examples of an enterprise network and a national Internet service provider.



The results of the study provide some cause for optimism. As a result of the effectiveness of real-world rule sets, an unexpectedly low number of test packets are required. On the Stanford network, which has over 757,000 rules and over 100 VLANs, only 4000 packets are needed to validate all of the forwarding rules and ACLs. On Internet2, testing and evaluating all IPv4 forwarding rules with a sample size of 35,000 packets is sufficient. An additional choice is to conduct a thorough investigation of each rule

implemented in each router connected to the Stanford backbone at a rate of ten rules per second. Sending test packets that make use of less than 1% of the network's available capacity is what is required to accomplish this. The link coverage at Stanford University is restricted to approximately 50 packets, which is a relatively small area. When there is this much network coverage, it is possible to perform proactive liveness testing every millisecond while utilizing only 1% of the network's capacity.

This publication includes the following contributions from its contributors:

The findings of a study of network operators are presented in Section II, which focuses on common failures and the primary causes of those failures

- An algorithm for producing test packets is presented in Section IV.A of this document.
- The defect localization algorithm is discussed in Section IV-B. This algorithm can be used to locate defective regulations and components.
- A number of different applications are provided after the discussion of Automatic Test Pattern Generation (ATPG) in functional and performance testing that can be found in Section V.
- A prototype ATPG system is evaluated in Sections VI and VII by utilizing rule sets from the Stanford and Internet2 backbones.

2. EXISTING SYSTEM

Internet service providers (ISPs) and the administrators of data centers face a significant challenge when it comes to measuring the liveness of their networks. Transmitting probes between every pair of edge ports is not the way to achieve scalability and exhaustiveness because of this. It is sufficient to identify a small number of end-to-end transmissions passing across each link. However, in order to finish this work, a method for generalizing device-specific configuration files must be devised. This approach must additionally generate headers and associated connections, as well as identify the least amount of test packets required (known as Min-Set-Cover).

To confirm that the policy and configuration are compatible.

Disadvantages Of Existing System

This approach is not meant to detect liveness failures, router hardware or software problems, or performance concerns.

Hardware and software issues are the most typical causes of network failure. These defects can result in two types of issues: a decrease in data transfer speed and a delay in data transmission.

3.PROPOSED SYSTEM

The Automatic Test Packet Generation (ATPG) framework is meant to automatically create a suitable quantity of packets. These packets are used to evaluate the network topology's operational status and to assure that the data plane state and configuration requirements are consistent. The application can produce packets autonomously in order to analyze and validate performance claims such as packet latency.

Furthermore, the system's functionality can be altered to generate a condensed set of packets that just evaluate the operational state of each network link.

Advantages Of Proposed System:

- A study of network operators was done to identify common failures and their underlying causes.
- The author presents a way for constructing test packets. A defect localization algorithm that discovers and separates incorrect rules and devices.
- The use case scenarios of Automatic Test Pattern Generation (ATPG) include functional and performance testing.
- The purpose of this study is to analyze a prototype Automatic Test Pattern Generation (ATPG) system.

The evaluation is carried out with the use of rule sets derived from the Stanford and Internet2 backbones.

SYSTEM ARCHITECHURE

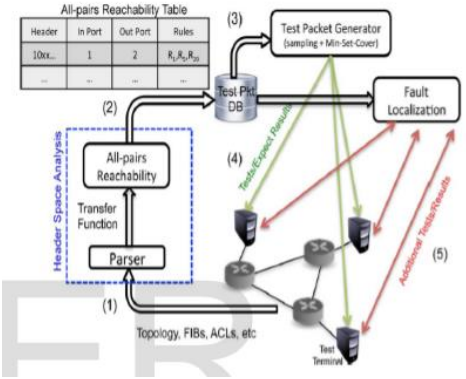


Fig.1The method of producing test packets in an automated fashion.

4. IMPLEMENTATION

JavaTechnology

Both a platform and a programming language are included in the Java technology package.

The Java Programming Language

The Java programming language is considered to be a high-level language and is characterized by the following traits, which are frequently cited in descriptions of the language:

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic

In the vast majority of programming languages, a computer program can only be executed once it has either been compiled or interpreted. A program written in the Java programming language can be compiled in addition to being interpreted, which is one of the defining characteristics of this language. During compilation, a program is changed from its original form into Java byte codes, which are an intermediate language. These codes can be interpreted and assessed by the Java platform's interpreter since they are platform independent. The interpreter is responsible for decoding and carrying out the operation of each Java binary code instruction on the computer. In contrast, interpretation must be performed each time a program is executed, whereas compilation only needs to be done once. The flow of this method is illustrated in the provided diagram, which can be found below. ATPG should only utilize the IDs that are allowed to be broadcast by each test terminal. This is the only set of identifiers that should be used.

Generate All-Pairs Reach ability Table:

The first thing that has to be done in order to use the ATPG method is to determine the complete list of packet headers that can be transmitted between two test terminals. Specifies the exhaustive collection of guiding principles that ATPG employs for each header it encounters along its path. This is accomplished by ATPG through the utilization of the all-pairs reachability algorithm, which was discussed before. On each terminal port of the first switch's transfer function, which is connected to each test terminal, all-headers, also known as headers with random carded bits, are inserted. These headers are also known as all-headers. In this instance, the limitations relating to the header are in force.

IMPLEMENTATION MODULES:

- Test Packet Generation
- Generate All-Pairs Reach ability Table
- ATPG Tool
- Fault Localization

MODULES DESCRIPTION:

Test Packet Generation:

It is to be anticipated that the network will feature a collection of test terminals that are in a position to both send and receive test packets. The objective is to generate a large number of test packets in order to perform an in-depth analysis of the rule set associated with each switch function. This technique guarantees that any potential problems will be found in at least one test transmission that is performed. This is analogous to software test suites, which attempt to assess every possible alternative branch of the program in great detail. It's possible that the goal is as straightforward as inspecting each queue or link. When it comes to the creation of test packets, Automatic Test Pattern Generation (ATPG) needs to be very careful to comply to two essential limitations. Automatic test pattern generating (ATPG) systems should only make use of test terminals that are already accessible during their initial deployment. Along with this,

ATPG Tool:

The ATPG approach is designed to examine each forwarding rule in the network by using at least one test packet as a basis for the analysis. When an issue is found, the Automatic Test Pattern Generation (ATPG) technology employs an approach called fault localization to figure out which rules or links did not work properly.

Fault Localization:

The Automatic Test Pattern Generation (ATPG) strategy involves routinely transmitting a number of test packets in a predetermined pattern. The ATPG (Automatic Test Pattern Generation) program accurately identifies the weaknesses or faults that led to the failure of a test packet when the packet is subjected to an unsuccessful test. When the behavior that is observed does not match the behavior that is expected, a rule is said to have failed. In order to maintain a record of rule violations, the ATPG algorithm makes use of a result function. The characteristics of the system being used to define what constitutes success and failure are taken into consideration. A forwarding rule is considered to have been unsuccessful if a test payload is unable to reach the output port for which it was intended. On the other hand, in situations where packets are dropped on purpose, it is presumed that the drop rule is operating well. In the same vein, a link failure takes place whenever a forwarding rule contained inside the topology function is unable to operate in the appropriate manner. Failure, on the other hand, is only identified when a test packet reveals that the delay on an output link has reached a level that is greater than the threshold that was previously established.

5. CONCLUSION

Internet service providers (ISPs) and the operators of data centers face a significant challenge when it comes to measuring the liveness of their networks. It is considered insufficient to send probe signals between every pair of edge terminals since it does not provide sufficient coverage and it cannot be scaled up. The goal at this point is to discover a collection of end-to-end packets that make use of the minimum number of links possible. However, in order to fulfill this goal, a method for generalizing device-specific configuration files, such as a header space, will need to be conceived of and developed. In addition, all-pairs reachability requires the development of headers in addition to the links that these headers construct. The last criterion is to figure out the minimum set cover, often known as the least amount of test packets. The application-specific integrated circuit (ASIC) and software design industries are aided by multibillion-dollar tool companies that offer ways for static (such as design rule) and dynamic (such as timing) verification. A few months after we built and christened our system, we were taken aback by the discovery that the acronym ATPG, which stands for Automatic Test Pattern Generation, is a well-known name in the field of hardware semiconductor testing. This was an unexpected discovery for us. Additionally, the automated dynamic testing of production networks is anticipated to benefit from the utilization of the network ATPG..

REFERENCES

1. ATPGcoderepository,[Online].Available:http://eastzone.github.com/atpg/
2. AutomaticTestPatternGeneration,2013
3. Networkmanagerstoday use primitive toolssuchas and.Our survey results indicate that they areeager for more sophisticated tools. Other fields of engineering indicate that these desires are not unreasonable:Forexample, both P.Barford, N.Duffield, A.Ron, and J.Sommers, Network performance anomaly detection and localization, in Proc.IEEEINFOCOM,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, inProc. OSDI, Berkeley, CA, USA, 2008, pp. 209–224.
7. Canini, D.Venzano, P.Peresini,D.Kostic,and J. Rexford, A NICE way to test Open Flow applications, inProc.NSDI,2012,pp.10–10.
8. Dhamdhere, R. Teixeira, C. Dovrolis, and C.Diot, Net diagnoser: Trouble shooting 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 unicastprobes, in Proc. IEEE INFOCOM, 2001, vol. 2,pp. 915–923.