International Journal of Science Engineering and Advance Technology

# Habitual Test Packet Generation And Fault Localization

**KSNM Chaitanya [1], U.Vinod Kumar [2]**
[1]PG Scholar, Pydah College of Engineering, Kakinada, AP, India, E-mail: ksnmc522@gmail.com.
[2]Assistant Professor, Pydah College of Engineering, Kakinada, AP, India.

**Abstract—Networks are getting larger and more complex, yet administrators rely on rudimentary tools such as and to debug problems. We propose an automated and systematic approach for testing and debugging networks called "Automatic Test Packet Generation" (ATPG). ATPG reads the router configurations and generates a device-independent model. The model is used to generate a minimum set of test packets to (minimally) exercise every link in the network or (maximally) exercise every rule in the network. Test packets are sent periodically, and detected failures trigger a separate mechanism to localize the fault. ATPG can detect both functional (e.g., incorrect firewall rule) and performance problems (e.g., congested queue). ATPG complements but goes beyond earlier work in static checking (which cannot detect liveness or performance faults) or fault localization (which only localize faults given liveness results).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**

**Key words: test packet generation, Data plane analysis, network troubleshooting,**

## I. Introduction

Networking is the word basically relating to computers and their connectivity. It is very often used in the world of computers and their use in different connections. The term networking implies the link between two or more computers and their devices, with the vital purpose of sharing the data stored in the computers, with each other. The networks between the computing devices are very common these days due to the launch of various hardware and computer software which aid in making the activity much more convenient to build and use.

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 the 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. 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.

## II.Proposed System

- Automatic Test Packet Generation (ATPG) framework that automatically generates a minimal set of packets to test the liveness of the underlying topology and the congruence between data plane state and configuration specifications. The tool can also automatically generate packets to test performance assertions such as packet latency.
- It can also be specialized to generate a minimal set of packets that merely test every link for network liveness.

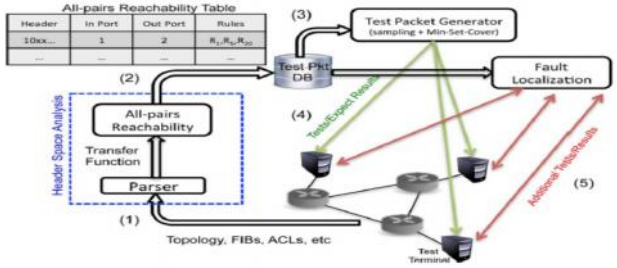



Fig.1: system architecture

**Advantages Of Proposed System:**

- A survey of network operators revealing common failures and root causes.
- A test packet generation algorithm.
- A fault localization algorithm to isolate faulty devices and rules.
- ATPG use cases for functional and performance testing.
- Evaluation of a prototype ATPG system using rule sets collected from the Stanford and Internet2 backbones.

### *A. Test Packet Generation*

*1)* **Algorithm:**We assume a set of *test terminals* in the networkcan send and receive test packets. Our goal is togenerate aset of test packets to exercise*every* rule in *every* switch function,so that *any* fault will be observed by at least one test packet. Thisis analogous to software test suites that try to test every possiblebranch in a program. The broader goal can be limited to testingevery link or every queue.When generating test packets, ATPG must respect two keyconstraints: 1) *Port:* ATPG must only use test terminals that areavailable; 2)*Header:* ATPGmust only use headers that each testterminal is permitted to send. For example, the network administratormay only allow using a

specific set of VLANs. Formally,we have the following problem.*Problem 1 (Test Packet Selection):* For a network with the switch functions, and topology function, determinethe minimum set of test packets to exercise all reachablerules, subject to the port and header constraints.ATPG chooses test packets using an algorithm we call *TestPacket Selection* (TPS). TPS first finds all *equivalent classes*between each pair of available ports. An equivalent class is aset of packets that exercises the same combination of rules.

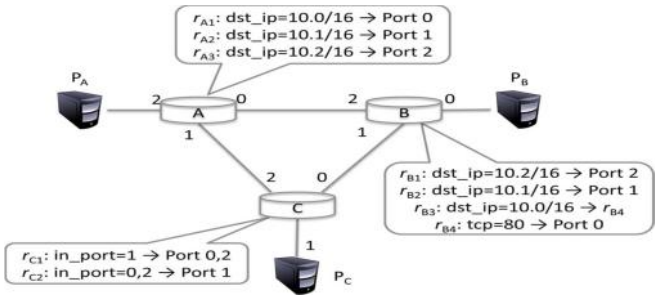



Fig2. Example topology with three switches

**IV. Implementation**

**Modules:**

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

**Modules Description:**

**Test Packet Generation:**

We assume a set of test terminals in the network can send and receive test packets. Our goal is to generate a set of test packets to exercise every rule in every switch function, so that any fault will be observed by at least one test packet. This is analogous to software test suites that try to test every possible branch in a program. The broader goal can be limited to testing every link or every queue. When generating test packets, ATPG must respect two key constraints First Port (ATPG must only use test terminals that are available) and Header (ATPG must only use headers that each test terminal is permitted to send).

**Generate All-Pairs Reachability Table:**

ATPG starts by computing the complete set of packet headers that can be sent from each test terminal to every other test terminal. For each such header, ATPG finds the complete set of rules it exercises along the path. To do so, ATPG applies the all-pairs reach ability algorithm described. On every terminal port, an all- header (a header that has all wild carded bits) is applied to the transfer function of the first switch connected to each test terminal. Header constraints are applied here.

**ATPG Tool:**

ATPG generates the minimal number of test packets so that every forwarding rule in the network is exercised 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.

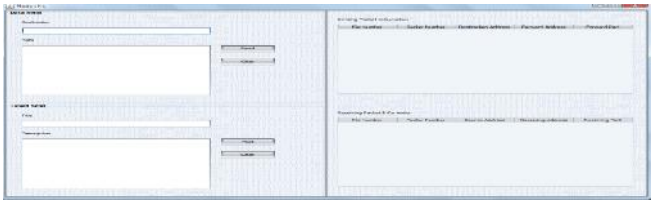
**Fault Localization:**

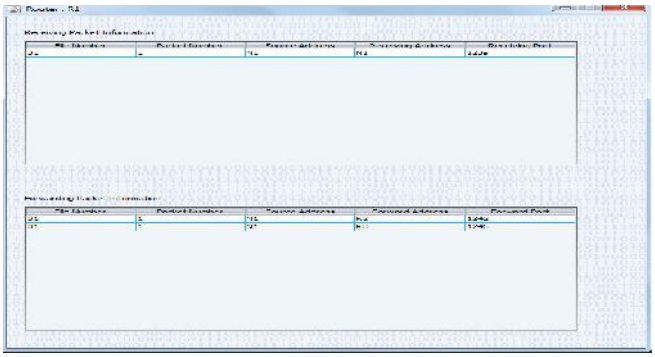
ATPG periodically sends a set of test packets. If test packets fail, ATPG pinpoints the fault(s) that caused the problem. A rule fails if its observed behaviour differs from its expected behaviour. ATPG keeps track of where rules fail using a result function “Success” and “failure” depend on the nature of the rule: A forwarding rule fails if a test packet is not delivered to the intended output port, whereas a drop rule behaves correctly when packets are dropped. Similarly, a link failure is a failure of a forwarding rule in the topology function. On the other hand, if an output link is congested, failure is captured by the latency of a test packet going above a threshold.

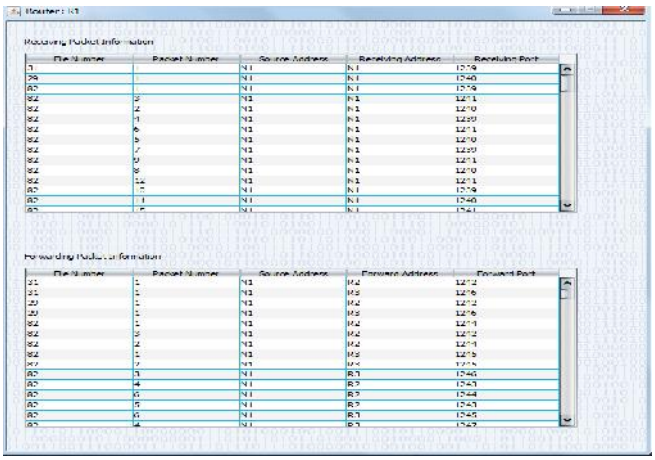
**Results**

**Screen Shots**

Node 1:







**V. Conclusion**

Testing liveness of a network is a fundamental problem for ISPs and large data center operators. However, doing this requires a way of abstracting across device specific configuration files (e.g., header space), generating headers and the links they reach (e.g., all-pairs Reachability), and finally determining a minimum set of test packets (Min-Set-Cover). Even the fundamental problem of automatically generating test packets for efficient liveness testing requires techniques akin to ATPG. Our implementation also augments testing with a simple fault localization scheme also constructed using the header space framework. As in software testing, the formal model helps maximize test

coverage while minimizing test packets. Our results show that all forwarding rules in Stanford backbone or Internet2 can be exercised by a surprisingly small number of test packets ( for Stanford, and for Internet2).

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. ATPG, however, goes much further than liveness testing with the same framework. ATPG can test for Reachability policy (by testing all rules including drop rules) and performance health (by associating performance measures such as latency and loss with test packets). Other fields of engineering indicate that these desires are not unreasonable: For example, both the ASIC and 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, many months after we built and named our system, we discovered to our surprise that ATPG was a well-known acronym in hardware chip testing, where it stands for Automatic Test *Pattern* Generation. We hope network ATPG will be equally useful for automated dynamic testing of production networks.

Network managers today use primitive tools such as and. Our survey results indicate that they are eager for more sophisticated tools.

**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

[10] “ATPG code repository,” [Online]. Available: http://eastzone.github.com/atpg/

[11] “Automatic Test Pattern Generation,” 2013 [Online]. Available:http://en.wikipedia.org/wiki/Automatic_test_pattern_generation

[12] P. Barford, N. Duffield, A. Ron, and J. Sommers, “Network performanceanomaly detection and localization,” in *Proc. IEEE INFOCOM*,Apr. , pp. 1377–1385.

[13]“Beacon,”[Online].Available:http://www.beaconcontroller.net/

[14] Y. Bejerano and R. Rastogi, “Robust monitoring of link delays andfaults in IP networks,” *IEEE/ACM Trans. Netw.*, vol. 14, no. 5, pp.1092–1103, Oct. 2006.