International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 12, December 2015
Available at http://internationaljournalofresearch.org

A Novel Approach for Minimizing Attacks Using Automatic Test Packet Generation on Networks

Pulavarthi Srinivas[1]& R.Nagaraju [2]
[1]PG Scholar,Dept Of CSE, Nova's Institute Of Tecvhnology, Eluru ,Andhra Pradesh
[2]Associate Professor,Dept Of CSE Nova's Institute Of Tecvhnology, Eluru,Andhra Pradesh

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. For example, 4000 packets can cover all rules in Stanford backbone network, while 53 are much enough to cover all links.

Keywords: Fault Localization; Test Packet Selection; Network Debugging; Automatic Test packet Generation (ATPG); Forwarding Information Base (FIB)

I.INTRODUCTION

Whenever networking comes into picture, questions that we come across are about “How to secure your network? Is my network secure? What do I need to do make network secure?” But network security does not limit only by implementing new firewall optimizing techniques or to secure the information, rather it also includes monitoring the packets, forwarding entries etc.

Now, this would arise the question of how this would help to secure the network. The answer to this is, the security could be easily breached by tampering the rules and exploiting the errors.

Until now it is the network administrator’s problem to tackle with such issues. Troubleshooting a network is difficult for three reasons. First, the forwarding state is distributed across multiple routers and firewalls and is defined by their forwarding tables, filter rules, and other configuration parameters. Second, the forwarding state is hard to observe because it typically requires manually logging into every box in the network. Third, there are many different programs, protocols, and humans updating the forwarding state simultaneously. (See Fig.1)

But creating a tool using ATPG algorithm would automate the entire process.

Therefore our goal is “To build a system which would automatically monitor functional and performance faults in network. To detect and diagnose errors by independently and exhaustively testing all forwarding entries, firewall rules, and any packet processing rules in the network. To check the liveness and fault localization of the network.”

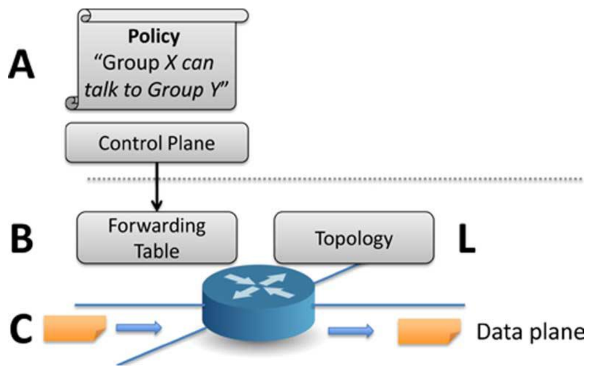

Fig. 1. Static versus dynamic checking: A policy is compiled to forwarding state, which is then executed by the forwarding plane. Static checking (e.g., confirms that A=B . Dynamic checking (e.g., ATPG in this paper) confirms that the topology is meeting liveness properties (L) and that B=C.

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.

III.PROBLEM DEFINITION

In current system, the administrator manually decides which ping packet to be sent. Sending programs between every pair of edge ports is neither extensive nor scalable. This system is enough to find minimum set of end-to-end packets that travel each link. However, doing this need a way of abstracting across device specific configuration files generating headers and links they reach and finally calculating a minimum set of test packets. It is not designed to identify failures caused from failed links and routers, bugs caused from faulty router hardware or software, and performance problems. The common causes of network failure are hardware failures and software bugs, in which that problems manifest both as reachability failures and throughput/latency degradation. To overcome this we are proposing new system.

IV. PROPOSED SYSTEM

In current system, the administrator manually decides which ping packet to be sent. Sending programs between every pair of edge ports is neither extensive nor scalable. This system is enough to find minimum set of end-to-end packets that travel each link. However, doing this need a way of abstracting across device specific configuration files generating headers and links they reach and finally calculating a minimum set of test packets. It is not designed to identify failures caused from failed links and routers, bugs caused from faulty router hardware or software, and performance problems. The common causes of network failure are hardware failures and software bugs, in which that problems manifest both as reach ability failures and throughput/latency degradation. To overcome this we are proposing new system. Fig.2 shows the architecture of proposed system. In this paper, ATPG framework generates minimum set of packets automatically, to debug the failures occurring in the network. This tool could automatically generate packets for checking performance assertions such as like packet latency. ATPG finds and determines errors by independently testing all forwarding entries, any packet processing rules and firewall rules in network.

Here, 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 ATPG treats links like normal forwarding rules, its full coverage provides testing of every link in the network. It can also be specialized to form a minimal set of packets that obviously test every link for network liveness. At least in this basic form, we would feel that ATPG or some 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. However, all-pairs do not provide testing of all links and has been found to be unsalable for large networks such as Planet Lab

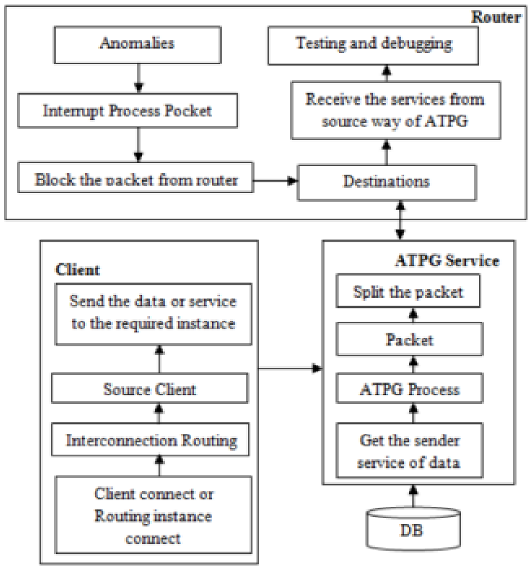
Router
Anomalies
Testing and debugging
Interrupt Process Pocket
Receive the services from source way of ATPG
Block the packet from router
Destinations
Client
Send the data or service to the required instance
Source Client
Interconnection Routing
Client connect or Routing instance connect
ATPG Service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

Fig.2. ATPG system block diagram.

The proposed system can be divided into following modules:

- Failures and root causes of network operators
- Data plane analysis
- Network troubleshooting
- ATPG system
- Network Monitor

A.Failures 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 reasons. First, the forwarding state is shared to multiple routers and firewalls and is determined by the forwarding tables, filter rules, and configuration parameters. Second, the forwarding state is difficult to watch because it requires manually logging into every box in the network. Third, the forwarding state is edited simultaneously by different programs, protocols and humans.

B. Data Plane Analysis

Automatic Test Packet Generation framework which automatically generates a minimum set of packets to check the likeness of underlying topology and congruence between data plane state and configuration specifications this tool can automatically generate packets to test performance assertions like packet latency. ATPG find errors by independently and exhaustively checking all firewall rules, forwarding entries and packet processing rules in network. The test packets are generated algorithmically from the device configuration 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

C.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 is the ideal tool for network debugging it 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 issues, 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.

D. ATPG System 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 fault localization algorithm to ascertain the failing rules or links.

E. Network Monitor To send and receive test packets, network monitor assumes special test agents in the network. The network monitor gets the database and builds test packets and instructs each agent 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 problem. The process gets repeated till the fault has been identified. To communicate with test agents, monitor uses JSON, and SQLite's string matching to lookup test packets efficiently

V. CONCLUSION

Network administrator use primitive tools such as Ping and trace route. My survey results indicate they are sager for more sophisticated tools. Other field of engineering indicates that desires are not unreasonable. 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 reach ability 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

VI. 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‖, IEEE/ACM Transactions on Networking, Vol. 22, No. 2, April 2014.

[2] ―ATPG code repository,‖ [Online]. Available: http:// eastzone.github. Com/atpg/. [3] ―Automatic Test Pattern Generation,‖ 2013 [Online].Available:http://en.wikipedia.org/wiki/Automatic_test_patter n_ generation.

[4] P. Barford, N. Duffield, A. Ron, and J. Sommers, ―Network performance anomaly detection and localization,‖ in Proc. IEEE INFOCOM, Apr, pp. 1377–1385.

[5] ―Beacon,‖ [Online]. Available: http://www.beaconcontroller.net./.

[6] 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.

[7] C. Cadar, D. Dunbar, and D. Engler, ―Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs,‖ in Proc. OSDI, Berkeley, CA, USA, 2008, pp. 209–224.

[8] M. Canini, D.Venzano, P. Peresini, D.Kostic, and J. Rexford, ―A NICE way to test Open Flow applications,‖ in Proc. NSDI, 2012, pp. 10–10.

[9] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, ―Netdiagnoser: Troubleshooting network un-reach abilitiesusing end-to-end probes and routing data,‖ in Proc. ACM Context, 2007, pp. 18:1–18:12.

[10] N. Duffield, ―Network tomography of binary network performance characteristics,‖ IEEE Trans. Inf. Theory, vol. 52, no. 12, pp. 5373–5388, Dec. 2006.

[11] N. Duffield, F. L. Presti, V. Paxson, and D. Towsley, ―Inferring link loss using striped uni-cast probes,‖ in Proc. IEEE INFOCOM, 2001, vol. 2, pp. 915–923.

[12] 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.