Fault Diagnosis Using Automatic Test Packet Generation

Mrs. R.Radheesha,
Research scholar, Department of cse,
Bharath University, Chennai, India

Mr. A.R. Arunachalam
Research Guide, Department of cse,
Bharath University, Chennai, India

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

It is popularly known us, very difficult to troubleshoot or identify and remove errors in networks. Every day, network engineers fight with mislabeled cables, software bugs, router misconfigurations, fiber cuts, faulty interfaces and other reasons that cause networks to drop down. Network engineers hunt down bugs with various tools (e.g., Ping, trace route, SNMP) and track down the reason for network failure using a combination of accrued wisdom and impression. Debugging networks is becoming more harder as networks are growing larger (modern data centers may contain 10 000 switches, a campus network may serve 50 000 users, a 100-Gb/s long-haul link may carry 100 000 flows) and are getting complicated (with over 6000 RFCs, router software was based on millions of lines of source code, and network chips contain billions of gates.

Fig. 1 is a simplified view of network state. Bottom of the figure is the forwarding state to forward each packet, consist of L2 and L3 forwarding information base (FIB), access control lists, etc. The forwarding state was written by the control plane (that could be local or remote) and should correctly implement the network administrator’s scheme. Examples of the scheme include: “Security group X was isolated from security Group Y,” “Use OSPF for routing,” and “Video traffic received at least 1 Mb/s.” We could think of the controller compiling the scheme (A) into device-specific configuration files (B), which in turn determine the forwarding behavior of each packet (C). To ensure the network behave as designed, the three steps should remain consistent every times. Minimally, requires that sufficient links and nodes are working; the control plane identifies that a laptop can access a server, the required outcome can fail if links fail. The main reason for network failure is hardware and software failure, and this problem is recognized themselves as reachability failures and throughput/latency degradation. Our intention is to automatically find these kinds of failures.

The intention of this paper is to generate a minimum set of packets automatically to cover every link in the network. This tool can automatically generate packets to test performance assertions like packet latency. ATPG detects errors independently and exhaustively testing forwarding entries and packet processing rules in network. In this tool, test packets are created algorithmically from the device configuration files and First information base, with minimum number of packets needed for complete coverage. Test packets are fed into the network in which every rule was exercised directly from the data plan. Since ATPG treats links just like normal forwarding rules, the full coverage provides testing of every link in network. It could be particularized to generate a minimal set of packets that test every link for network liveness. For reacting to failures, many network operators like Internet proactively test the health of the network by pinging between all pairs of sources.

Organizations can modify ATPG to face their needs; for example, they can select to test for network liveness (link cover) or test every rule (rule cover) to make sure security policy. ATPG could be modified to test reachability and performance. ATPG can adapt to constraints such as taking test packets from only a few places in the network or using particular routers to generate test packets from every port.

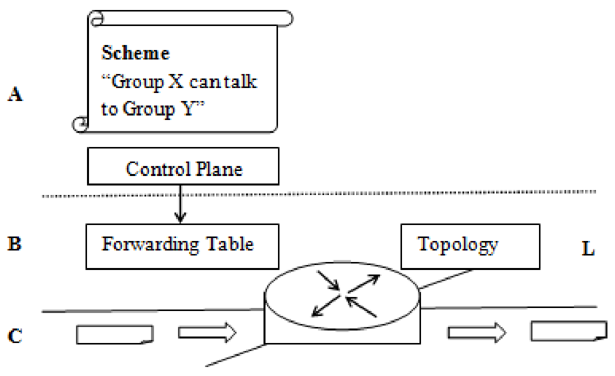

Fig. 1. Static versus dynamic checking: A scheme is compiled to forwarding state, and it is executed by the forwarding plane.

The contributions of this paper are as follows:
1) A survey of network operators exposing common failures and root causes.
2) A test packet generation algorithm.
3) A fault localization algorithm to separate faulty devices and Rules.
4) ATPG usecases for functional and throughput testing.
5) Evaluation of prototype ATPG system using rulesets gathered from the Stanford and Internet2 backbones.

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

Fig.2 shows the architecture of proposed system. In this paper,

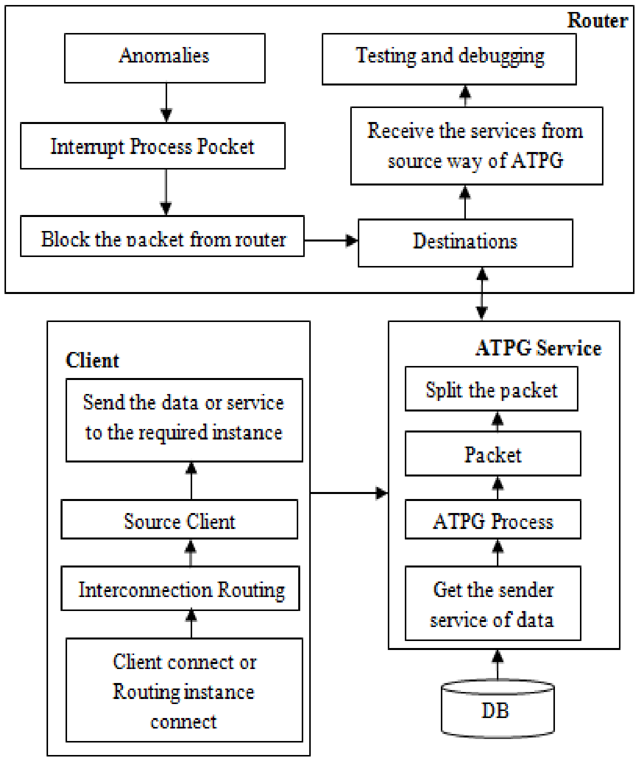
Router
Anomalies
Testing and debugging
Interrupt Process Pocket
Receive the services from source way of ATPG
Block the packet from router
Destinations
Client
Send the data or service to the required instance
Source Client
Interconnection Routing
Client connect or Routing instance connect
ATPG Service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

Fig. 2. ATPG system block diagram

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 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:

5.1 Failures and root causes of network operators
5.2 Data plane analysis
5.3 Network troubleshooting
5.4 ATPG system
5.5 Network Monitor

5.1 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.

5.2 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 reachability or for its performance

5.3 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 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.

5.4 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.

5.5 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.

VI. CONCLUSION

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 reachability 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.

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”.

[2] M. Jain and 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., and 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, and 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. IEEE Network Magazine (March 2002).

[7] 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.