International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 03 Issue: 01 | Jan-2016 www.irjet.net p-ISSN: 2395-0072

Automatic Analyzing System for Packet Testing and Fault Mapping

Mr. Shrikant B. Chavan[1] , Prof. Soumitra S. Das[2]

[1] Researcher, Department of Computer Engineering, Dr. D. Y. Patil School of Engineering, Pune, India
[2] Guide and HOD, Department of Computer Engineering, Dr. D. Y. Patil School of Engineering, Pune, India

---

Abstract - **now a day's Networks are getting larger and more complex, hence network admin depend on normal tools such as ping and to traceroute debug the problems. Automated and systematic approach for testing and debugging networks called "Automatic Analyzing System for Packet Testing and Fault Mapping". This system read router configurations and generates a device-independent model. This model is used to generate a minimum set of test packets to check every link in network check every rule in the network. Test packets are sent periodically, and detected failure trigger a separate mechanism to localize the fault. This model can detect both functional testing and performance testing problems. This proposed to use symbolic execution a technique prevalent in compilers to check network properties more general than basic reachability. the key idea is to track the possible values for specified fields in the packet as it travels through a network.**

Keywords:*Dataplane analysis, network troubleshooting, test packet generation, Network, Monitoring Functional Testing, Performance Testing*

1.INTRODUCTION

It is difficult to debug systems. Day by day system architects grapple with switch misconfiguration, fiber cuts, flawed interfaces, mislabeled links, programming bugs, irregular connections, and different reasons that cause systems come up short totally. System architects to execute down bugs utilizing the most well-known devices (e.g., Ping , Traceroute ,SNMP,and Tcpdump) and track main drivers utilizing a mix of accumulated shrewdness . Debugging of systems is just getting to be difficult to systems are getting greater (current server farms may contain 10 000 switches, a grounds system may serve 50 000 clients, a 100-Gb/s long term connection may convey 100 000 flows) and are getting more confounded (with in excess of 6000 RFC, switch programming is focused around a large number of lines of source code, and system chips likewise contain billions of entryways). It is a ponder that system specialists have been marked "experts of many sided quality". For that Consider a same.[1]

Example 1: Suppose a router with a faulty line card starts dropping packets silently. Admin, who administers 100 routers, receives an ticket from several unhappy users complaining about connectivity. First Admin examines each router to see if the configuration was changed recently and concludes that the configuration was untouched [2].

Next, Admin utilizes his insight into topology to follow the broken gadget with ping and tracerout charge. At long last, he calls an associate to supplant the link. That the two most regular reasons for system disappointment are equipment disappointments and programming bugs, and those issues distinguished themselves both as achieve capacity disappointments and throughput/inertness debasement. to test the liveness to give backing to topology .The instrument can likewise naturally produce parcels to test execution attestations, for example, parcel inactivity.[2]

2.RELATED WORK

Present day machine systems can be isolated into the information plane and the control plane. The information plane comprises of various interconnected switches, each one contains sending decides that focus the flow of parcels. For instance, the sending lead in an Ethernet switch takes a gander at a bundle's end of the line MAC address, and chooses its next port. [6].

On top of the information plane is the control plane that runs directing conventions, for example, OSPF or BGP. The control plane populates the information plane with sending tenets focused around its worldwide system learning.

The objective of information plane testing incorporates two sections: 1) checking sending guidelines rightness given topology and strategy (e.g., the back-end database can't converse with the front-end web server straightforwardly); 2) confirming system execution given the administration level assertion, which is an agreement between the system administration supplier and its clients.

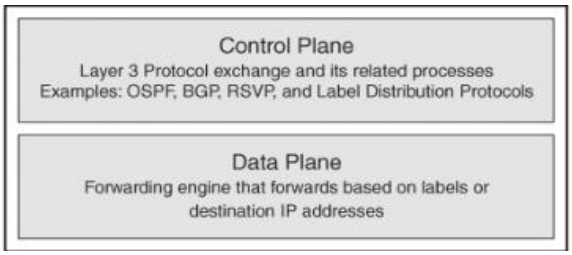

Fig. -1 Plane Testing

The disastrous substances of system operation make programmed, precise information plane troubleshooting a need. Nonetheless, there is more than one approach to

approach this issue. Besides, distinctive systems may call for diverse methodologies. Any information plane analyzer outline ought to answer the accompanying three inquiries.

- **Method:** Do we just read and break down sending tables (static examination), or do we really convey test bundles to watch the system's conduct (dynamic investigation)?
- **Knowledge:** How much we think about the system under test? Do we know all the sending tables and topology, simply a piece of them, or none of them?
- **Coverage:** Which arrange parts do we cover, connections or principles? How would we attain to 100% scope?

3. ALGORITHM

We take a set of test nodes in the network model send and receive test packets Our aim is to generate a set of test packets and changes every conditions in every switch objects so that every fault will be observed by at least one test packet This is scanner type software test models that try to test every possible branch models The broader goal can be limited to testing every link every queue When generating test packets ATPG must respect two key constraints (1) Port: ATPG must be test terminals that are available
(2) Header: ATPG must be use headers that each test terminal is permitted to send For the network administrator may only allow using a specific model of VLAN

```
GOOGLE-QUERY-DELAY(tcp_segments)
 1  s ← tcp_segments[1]
 2  p ← s
 3  c ← tcp_segments[2]
 4  while c ≠ NIL
 5     do if c.snd_time > p.snd_time and
 6           c.snd_time > p.ack_time
 7           then return (s.snd_time, c.snd_time)
 8           else
 9                 if c.size < MSS
10                   then return (s.snd_time, c.snd_time)
11        p ← c
12        c ← c.next
13
```

Fig. -2: Segments Methods

Test packets are taken into the network in which that every rule is covered directly from the data plane with different locations treats links like normal forwarding conditions its full coverage provides testing of every link in the network model It can also be specialized to form a minimal no of of packets that obviously test every link for network likeness At least one basic form we would feel that ATPG similar technique is fundamental to networks Instead of reacting to failures many network operators

4.METHODOLOGY

The proposed system can be divided into following modules:

1. Failures and root causes of network operators
2. Data plane analysis
3. Network troubleshooting
4. ATPG system
5. Network Monitor

1. Failure 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 different models First the forwarding state is shared to multiple routers and security and is determined by the forwarding data filter conditions and configuration parameters Second the forwarding state is difficult to watch because it requires manually logging into every box in the network model Third the forwarding state is edited simultaneously by different programs protocols and humans.

2. Data Plane Analysis

Automatic Test Packet Generation framework which automatically generates a minimum set of packets to check the likeness of underlying network models and congruence different data plane state and configuration specifications These model can automatically generate packets to test performance assertions like packet latency ATPG find faults by independently and exhaustively checking all security rules forwarding entries and packet processing conditions in network. The test packets are generated algorithmically from the device configuration different 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.

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 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.

4. ATPG Systems

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 different localization algorithm to ascertain the failing rules in network model

5. Network Monitor

To send and receive test data packet network monitor assumes special test agents in the network The network monitor gets the database and builds test packets and instructs each different to send the proper packets

5.PERFORMANCE

The principal component overhead for ATPG are polling the network periodically for forwarding state and performing two reachable While one can reduce overhead by running the offline.

ATPG calculation less frequently this runs the risk of using out-of-date forwarding information we reduce overhead in two ways First we have recently fast up the all-pairs reach ability calculation using a fast multithreaded. Second, instead of extracting the complete network state every time ATPG is triggered an incremental state updater can significantly reduce both the retrieval time and the time to calculate reach ability We are working on a real life version of ATPG that incorporates both techniques Test agents within terminals incur negligible overhead because they merely de multiplex test packets addressed to their IP address at a modest rate compared to the link speeds gb most modern CPUs are capable taken.

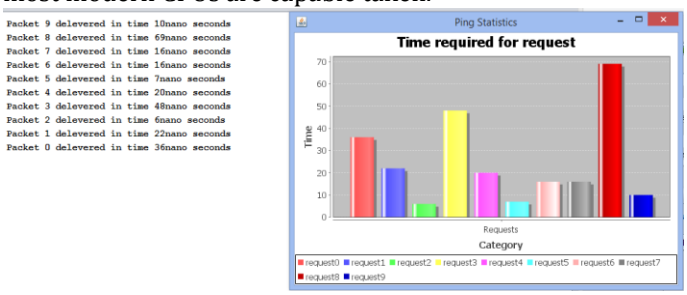

Chart 1:Result for 10 packets send and time required for reaching the destination of each packet.

CONCLUSION

System director use primitive instruments, for example, Ping and tracerroute. My study results show they are esager for more advanced devices. Other field of building show that yearnings are not outlandish: for instance, programming configuration commercial ventures are buttressed by billion dollar apparatus organizations that supply strategies for both static (e.g., outline standard) and element (e.g., timing) verification.

That demonstrating stateful systems could be possible in an adaptable manner Model headers as variables and utilization typical execution to catch fundamental system properties; middlebox stream state can likewise be effectively demonstrated utilizing such header variables.

ACKNOWLEDGMENT

I wish to express my genuine on account of the aide and HOD Prof. Soumitra Das and also our primary Dr. S.S. Sonawane, additionally Grateful on account of our PG Coordinator Prof. Pankaj Agarkar and to wrap things up, the departmental staff parts for their backing.

REFERENCES

[1] Hongyi Zeng, Peyman Kazemian,George Varghese,and Nick McKeown,"Automatic Test Packet Generation",VOL. 22, NO. 2, APRIL 2014.

[2] M. Jain and C. Dovrolis, End-to-end available bandwidth: Measure-ment methodology, dynamics, and relation with TCP throughput, IEEE/ACM Trans. Netw., vol. 11

[3] Daniel Turner, Kirill Levchenko, Alex C. Snoeren, and Stefan Savage "California Fault Lines: Understanding the Causes and Impact of Network Failures"

[4] Peyman Kazemian George Varghese Nick McKeown, "Header Space Analysis: Static Checking For Networks" R. Nicole.

[5] Maciej Ku zniar, Peter Pere s ni, Dejan Kosti "ProboScope: Data Plane Probe Packet Generation" EPFL Technical Report EPFL-REPORT-20182

[6] N. Dufeld, Network tomography of binary network performance characteristics, IEEE Trans. Inf. Theory, vol. 52, no. 12, pp. 53735388, Dec.2006