ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

Automatic Test Packet Generation to Detect Problems in Computer Network Nodes and Routers

Neha Rathod[1], Pratiksha Surve[2], Shraddha Chaudhari[3], N.V Satya Naresh Kalluri[4]

Bachelors of Engineering, Dept. of Computer, Alamuri Ratnamala Institute of Engineering & Technology, University of Mumbai, India[1,2,3]

Assistant Professor, Dept. of Computer, Alamuri Ratnamala Institute of Engineering & Technology, University of Mumbai, India [4]

ABSTRACT: Now-a-days our network administrators are been depended on the traditional tools as ping and traceroute. Their job is getting more and more complex as the network this days is just going on increasing at a very high speed. This paper both functional and performance problems. This model is capable proposed algorithm shows efficient utilization and increased network lifetime. Test packets are sent periodically and detect failures triggering a separate mechanism to localize the fault. It proposes an automatic testing and debugging procedure for verifying the security in various network conditions and provide the safety reaching of packets from one node to another. The model produced by ATPG model is capable for investigating complements but goes beyond earlier work in static checking (which cannot detect liveness or performance faults).

KEYWORDS: Data plane analysis; Network monitor; troubleshooting; Test packet.

I. INTRODUCTION

It is been well known that debugging and troubleshooting network has become a very difficult task this days. Due to increase in number of networks day-by-day the network engineers have to go through many problems such as router misconfiguration, mislabelled cables, software bugs, faulty interfaces and other such reasons that lead to drop down in networks. As networks are getting bigger it is becoming harder task to debug a network. Lets look for example,
Example:- Modern data centres might contain 10,000 switches a campus network given services to 50,000 users and 100-Gbps long link might carry 1000,000 flow and get more complicated with over 6,000 RFC's network chips which also contain thousands of gates and router software which is based upon thousands of lines of source codes It is been the small wonder where the network engineers have been labelled as the "Masters of Complexity".

Here we consider that if the video traffic is mapped to specific queue in a router, but because of low token bucket rate the packets are dropped. It is unclear that how alice can track down such a performance fault using ping & traceroute.To trace the faulty device with ping & traceroute command admin uses his knowledge of topology; finally to replace the cable he calls colleague. The hardware failure and software bugs are the most common cause to the network failure. Network troubleshooting is found to be difficult for the few reasons, (1) The distribution of forwarding state table across multiple routers and firewalls which is defined by their forwarding tables and configuration parameters. (2) Due to manually logging requirement the forwarding state is hard to observe. (3) Different programs, protocols and human update the forwarding state simultaneously.

Thinking of controller compiling the scheme(A) into device-specific configuration files (B) which in turn determine the forwarding behavior of each packet (C). To ensure that the network behave as designed, the three steps should remain consistent every times. Requires that the link and nodes are sufficiently working; the laptop accessing the server is identified by the control plane. If the link fails the required outcome can also be failed.

The contributions of this paper is as follow,

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 use-cases for functional and throughput testing.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

II. Related work

In recent system the ping packet to be sent was decided manually by the administrator. The program sending between each and every pair of edge ports is neither scalable nor extensive. The offline tools is the often related works to which we are familiar which test invariants in network. This system was enough to find minimum set of end-to-end packet which travel every link. Let's begin by solving simple white box dynamic testing problem,

Example: **-** Considering a router with a faulty line-card that starts dropping packets silently. A ticket is been received by the Alice from an unhappy users complaining about connectivity who administrate 100 routers. After which the first thing Alice do is that she checks the configuration of each router if it was been changed recently and after that conclude that the configuration was untouched. Next she do is that she uses her knowledge of the topology to triangulate the faulty device with traceroute, ping and at end she finally calls a colleague to replace the line-card.

ATPG exhaustively test, detect and diagnoses all forwarding entries, packet processing rules and firewall rules, in the network errors by independently. In ATPG with minimum number of packets which are required for complete coverage test packets are generated automatically by the device configuration files and FIB's. IN ATPG, its full coverage guarantees testing of every link in the network as it treats links just like normal forwarding rules. In it testing of all link is not guarantee by all-pairs ping, to meet the need of ATPG organization can customize; such as they choose to check for liveness of network or every rule which ensure security policy. It can only be customized to check reachability or for performance, can adopt for constraining required test packets from only few places in the network or for using special routers to generate test packets from every port. It can also be tuned to allocate more test packets which exercise more of critical rules.

III. Proposed algorithm

A. *Design Considerations:*

1) Create a set *sptSet* (shortest path tree set) that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is empty.

2) Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.

3) While *sptSet* doesn't include all vertices

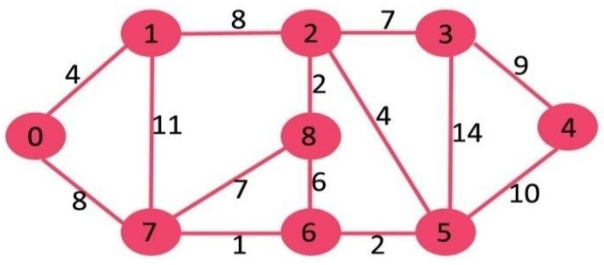
a) Pick a vertex u which is not there in *sptSet* and has minimum distance value.

b) Include upto *sptSet*.

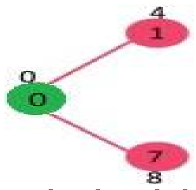
c) Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

B. *Description of the Proposed Algorithm:*

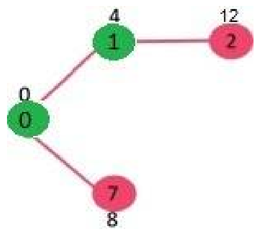
Let us understand with the following example:



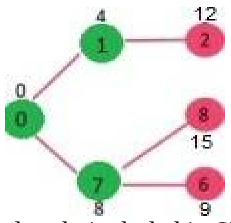
The set *sptSet* is initially empty and distances assigned to vertices are {0, INF, INF, INF, INF, INF, INF, INF} where INF indicates infinite. Now pick the vertex with minimum distance value. The vertex 0 is picked, include it in *sptSet*. So *sptSet* becomes {0}. After including 0 to *sptSet*, update distance values of its adjacent vertices. Adjacent vertices of 0 are 1 and 7. The distance values of 1 and 7 are updated as 4 and 8. Following subgraph shows vertices and their distance values, only the vertices with finite distance values are shown. The vertices included in SPT are shown in green color.



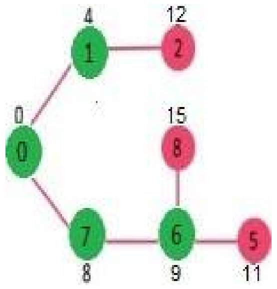
Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). The vertex 1 is picked and added to sptSet. So sptSet now becomes {0, 1}. Update the distance values of adjacent vertices of 1. The distance value of vertex 2 becomes 12.



Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). Vertex 7 is picked. So sptSet now becomes {0, 1, 7}. Update the distance values of adjacent vertices of 7. The distance value of vertex 6 and 8 becomes finite (15 and 9 respectively).



Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). Vertex 6 is picked. So sptSet now becomes {0, 1, 7, 6}. Update the distance values of adjacent vertices of 6. The distance value of vertex 5 and 8 are updated.



We repeat the above steps until *sptSet* doesn't include all vertices of given graph. Finally, we get the following Shortest Path Tree (SPT).

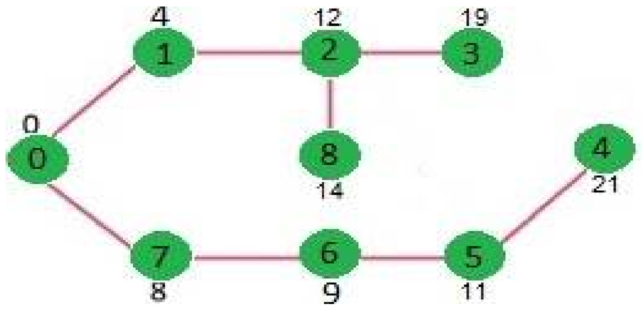
ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017



IV. Pseudo code

```
Step 1: Generate all the possible routes.
Step 2:  Calculate the $TE_{node}$ for each node of each route using eq. (1).
Step 3:  Check the below condition for each route till no route is available to transmit the packet.
            if ($RBE <= TE_{node}$)
                  Make the node into sleep mode.
            else
                  Select all the routes which have active nodes
            end
Step 4:  Calculate the total transmission energy for all the selected routes using eq. (2).
Step 5: Select the energy efficient route on the basis of minimum total transmission energy of the route.
Step 6:  Calculate the RBE for each node of the selected route using eq.  (3).
Step 7: go to step 3.
Step 8: End.
```

V. Simulation Results

In Automatic test packet generation minimal nodes are allowed to be send for one end to another(INFO - ode192.168.10.2: Sending data to 192.168.10.14), the relevant packet when sent from one end to another the successful acknowledgement is sent from the end user i.e. the data was successfully received by the user. If the sender wants to send data then it first searches(INFO - Node192.168.10.2: Searching for 192.168.10.14) for the shortest path/ nearest node to send data so that there will be less time required in sending data(what actually our project is about) and accordingly send such information. When the nearest node is fetched by the system the data is sent immediately(INFO - MapManager Sending broadcast packet From 192.168.10.2 to 192.168.10.3). During the sending of data if their is any kind of error found then it gives back the acknowledgement as(INFO - Node192.168.10.3: Jamming attack new route to 192.168.10.2 through 192.168.10.2 added). The following result shows the output for our packet when sent through the path.

OUTPUT:

```
INFO - Node192.168.10.2: Sending data to 192.168.10.14
INFO - Node192.168.10.2: discovery initiated to 192.168.10.14
INFO - Node192.168.10.2: Searching for 192.168.10.14
INFO - Node192.168.10.2: Route to 192.168.10.14Attack found!
INFO - Node192.168.10.2: Sending Broadcast Packet
INFO - MapManager Sending broadcast packet From 192.168.10.2 to 192.168.10.3
INFO - MapManager Sending broadcast packet From 192.168.10.2 to 192.168.10.1
INFO - Node192.168.10.3: Jamming attack new route to 192.168.10.2 through 192.168.10.2
added
INFO - Node192.168.10.1: Jamming attack new route to 192.168.10.2 through 192.168.10.2
added
INFO - Node192.168.10.3: RREQ_Recieved from 192.168.10.2 through 192.168.10.2
INFO - Node192.168.10.1: RREQ_Recieved from 192.168.10.2 through 192.168.10.2
```

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

```
INFO - Node192.168.10.3: Searching for 192.168.10.2
INFO - Node192.168.10.1: Searching for 192.168.10.2
INFO - Node192.168.10.3: Jamming attack new route to 192.168.10.2 through 192.168.10.2
added
INFO - Node192.168.10.1: Jamming attack new route to 192.168.10.2 through 192.168.10.2
added
INFO - Node192.168.10.3: Searching for 192.168.10.14
INFO - Node192.168.10.1: Searching for 192.168.10.14
INFO - Node192.168.10.3: Route to 192.168.10.14Attack found!
INFO - Node192.168.10.1: Route to 192.168.10.14Attack found!
INFO - Node192.168.10.3:recieved RREQPacket from 192.168.10.2 which handded from
192.168.10.2: but it is not the destination
INFO - Node192.168.10.1:recieved RREQPacket from 192.168.10.2 which handded from
192.168.10.2: but it is not the destination
INFO - Node192.168.10.3: Sending Broadcast Packet
INFO - Node192.168.10.1: Sending Broadcast Packet
INFO - MapManager Sending broadcast packet From 192.168.10.2 to 192.168.10.5
INFO - Node192.168.10.5: Jamming attack new route to 192.168.10.2 through 192.168.10.2
added
INFO - Node192.168.10.5: RREQ_Recieved from 192.168.10.2 through 192.168.10.2
INFO - Node192.168.10.5: Searching for 192.168.10.2
INFO - Node192.168.10.5: Jamming attack new route to 192.168.10.2 through 192.168.10.2
added
INFO - Node192.168.10.5: Searching for 192.168.10.14
INFO - Node192.168.10.5: Route to 192.168.10.14Attack found!
INFO - Node192.168.10.5:recieved RREQPacket from 192.168.10.2 which handded from
192.168.10.2: but it is not the destination
INFO - Node192.168.10.5: Sending Broadcast Packet
INFO - MapManager Sending broadcast packet From 192.168.10.2 to 192.168.10.6
INFO - MapManager Sending broadcast packet From 192.168.10.1 to 192.168.10.6
INFO - MapManager Sending broadcast packet From 192.168.10.1 to 192.168.10.5
INFO - Node192.168.10.6: Jamming attack new route to 192.168.10.2 through 192.168.10.2
added
INFO - Node192.168.10.5: Jamming attack new route to 192.168.10.1 through 192.168.10.1
added
INFO - Node192.168.10.6: Jamming attack new route to 192.168.10.1 through 192.168.10.1
added
INFO - Node192.168.10.6: RREQ_Recieved from 192.168.10.2 through 192.168.10.2
INFO - Node192.168.10.6: Searching for 192.168.10.2
INFO - Node192.168.10.6: Jamming attack new route to 192.168.10.2 through 192.168.10.2
added
INFO - Node192.168.10.6: Searching for 192.168.10.14
INFO - Node192.168.10.6: Route to 192.168.10.14Attack found!
INFO - Node192.168.10.6:recieved RREQPacket from 192.168.10.2 which handded from
192.168.10.2: but it is not the destination
INFO - Node192.168.10.6: Sending Broadcast Packet
INFO - MapManager Sending broadcast packet From 192.168.10.2 to 192.168.10.16
```

VI. Conclusion and Future Work

In our proposed system it uses a method which is neither exhaustive nor scalable, it reaches all different pairs of edge nodes and it would detect faults in likeness properties. ATPG also test for reachability model with performance methods. In our implementation it also checks simple error localization schemes and enlarges testing using header space framework

Even one of the requirements gathered through the voice of customers and feedback different users are implemented there are always opportunities to enhance model this tool and take it to the one level by automating different steps involved upon any level of code changes Explore automatically generating the unit tests results specific to the project without different the platform and save them to the output PDF Explore automatically generating the code coverage report and integrate in to the code review packet generation process Provide users used to upload the file directly to the given network location.

REFERENCES

[1] C. Cadar, D. Dunbar, and D. Engler. "Klee: unassisted and automatic generation of high-coverage tests for complex systems programs." In Proceedings of OSDI'08, pages209–224, Berkeley, CA, USA, 2008.USENIX Association.
[2] M. Canini, D. Venzano, P. Peresini, D. Kostic, and J. Rexford. A NICE way to test open flow applications. Proceedings of NSDI'12, 2012.
[3] P.Kazemian, G.Varghese, and N.McKeown, "Header space analysis: Static checking for networks,"in Proc. NSDI, 2012
[4] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, "Net diagnoser: Trouble shooting network unreachabilities using end-to-end probes and routing data,"inProc.ACMCoNEXT, 2007, pp.18:1–18:12.
[5] H. Zeng, P. Kazemian, G.Varghese, "Automatic test Packet generation," In Proc.IEEE,April 2014.
[6] H.Mai, A.Khurshid, R.Agarwal, M.Caesar, P.B.Godfrey, and S.T. King, "Debugging the data plane with Anteater," Comput. Commun. Rev., vol. 41, no. 4, pp. 290–301, Aug. 2011.
[7] M.Reitblatt, N.Foster, J.Rexford, C.Schlesinger, and D.Walker, "Abstractions for network update," in Proc. ACM SIGCOMM, 2012, pp. 323–334
[8] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," in *Proc. ACM SIGCOMM*, 2011, pp. 350–361.
[9] M. Shrikant, B. Chavan1, S. Das, "Review Paper on AUTOMATIC TEST PACKET GENERATION AND FAULT LOCALIZATION".

BIOGRAPHY

Neha Rathod[1], **Pratiksha Surve**[2], **Shraddha Chaudhari**[3], a students of computer department pursing degree in computer engineering from Alamuri Ratnamala Institute Of Engineering And Technology**.**

Prof .N.V Satya Naresh Kalluri[4] ; HOD of Information Technology Department of ARMIET, Mumbai University ,Mumbai**.**