© November 2015 | IJIRT | Volume 2 Issue 6 | ISSN: 2349-6002

AN ACCURATE AUTOMATIC TEST PACKET GENERATION AND FAULT LOCALIZATION

Kattubadi Abdulla1, A. V. Ramakrishna Reddy2
[1]*M.Tech student,CSE,Kottam college of Engineering,*
[2]*Assistant Professor,CSE ,Kottam college of Engineering,*
Chinnatekuru(V),Kurnool,Andhra Pradesh, India

***Abstract*- Now a day's Networks are getting larger and more complex, hence network admindepend on normal tools such as ping and to trace route debug the problems. We are proposingautomatic and systematic approach for testing and debugging networks called "Automatic TestPacket Generation and Fault Localization". ATPG read router configurations and generates aunique model. Test packets are sent periodically, and detectedfailures trigger a separate mechanism tolocalize the fault. ATPG can detect bothfunctional (e.g., incorrect firewall rule) andperformance problems (e.g., congested queue).ATPG complements but goes beyond earlierwork in static checking (which cannot detectliveness or performance faults) or faultlocalization (which only localize faults givenliveness results).**

***Index Terms*- Fault Localization, Test Packet Selection, Network Debugging, Automatic Test packet Generation (ATPG).**

I. INTRODUCTION

Detects and finding faults in differently and exhaustively testingall forwarding entries security rules and any packet processingrules in the network model generated algorithmically from thedevice configuration files with the minimum number of packetsrequired for complete locations . Test packets are different thenetwork so that every conditions directly from the data sources itsfull coverage guarantees testing of every link in the network It canalso be indicate to resurge a small set of packets that merely testevery link for network likeness. At end of in this basic form we feelthat some different technique is fundamental to networks modelof reacting to errors many network operators such as Internet2proactively check the health of their network using pings betweentwo of sources all-pairs guarantee testing of all links.

Consider two examples:

Example 1: Suppose a router with a faulty line card starts dropping packets silently. Admin, whoadministers 100 routers, receives a ticket from several unhappy users complaining aboutconnectivity. First Admin examines each router to see if the configuration was changed recentlyand concludes that the configuration was untouched [2].Next, Admin uses his knowledge of topology to trace the faulty device with ping and traceroutcommand. Finally, he calls a colleague to replace the cable. Two most common causes ofnetwork failure are generally hardware failures and software bugs, and that problems detectedthemselves both as reach ability failures and throughput/latency degradation. Our goal is toautomatically detect these types of failures The main contribution of a paper is what we call anAutomatic Test Packet Generation [ATPG] framework that automatic generates a minimal set ofpackets to test liveness that provide support for topology. The tool can also automaticallygenerate packets to test performance assertions such as packet latency.

In Example 2, instead of Admin manually decide which packets to send, the tool doesperiodically on his behalf. ATPG detects and diagnoses errors by independently and testing allforwarding entries, firewalls rules, and any packet processing rules in network.In ATPG, test packets are create algorithmically from the configuration files and FIB, withminimum 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 likenormal forwarding rules, it's full testing of every link in the network [2].Fig. 1 is a simplified view of network state. Bottom of thefigure is the forwarding state to forward each packet, consistof L2 and L3 forwarding information base (FIB), accesscontrol lists, etc.

Fig. 1 is a simplified view of network state. Bottom of thefigure is the forwarding state to forward each packet, consistof L2 and L3 forwarding information base (FIB), accesscontrol lists, etc. The forwarding state was written by thecontrol plane (that could be local or remote) and shouldcorrectly implement the network administrator's scheme.Examples of the scheme include: "Security group X wasisolated from security Group Y," "Use OSPF for routing,"and "Video traffic received at least 1 Mb/s." We could thinkof the controller compiling the scheme (A) into devicespecific configuration files (B), which in turn determine theforwarding behavior of each packet (C).To ensure thenetwork behave as designed, the three steps should remainconsistent every times. Minimally, requires that sufficientlinks and nodes are working; the control plane identifies thata laptop can access a server, the required outcome can fail iflinks fail. The main reason for network failure is hardwareand software failure, and this problem is recognizedthemselves as reachability failures and throughput/latencydegradation. Our intention is to automatically find thesekinds of failures.

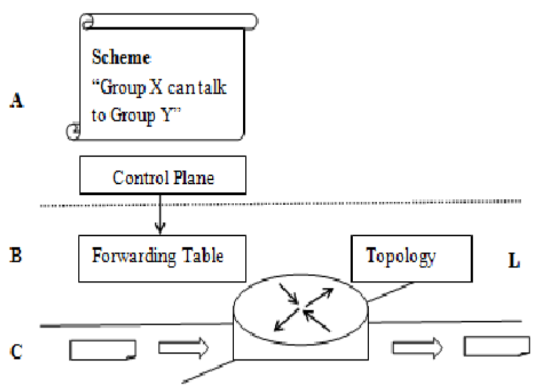


Fig. 1. Static versus dynamic checking: A scheme iscompiled to forwarding state, and it is executed by theforwarding plane.

The intention of this paper is to generate a minimum set ofpackets automatically to cover every link in the network..This tool can automatically generate packets to testperformance assertions like packet latency. ATPG detectserrors independently and exhaustively testing forwardingentries and packet processing rules in network. In this tool,test packets are created algorithmically from the deviceconfiguration files and First information base, withminimum number of packets needed for complete coverage.Test packets are fed into the network in which every rulewas exercised directly from the data plan. Since ATPGtreats links just like normal forwarding rules, the fullcoverage provides testing of every link in network. It couldbe particularized to generate a minimal set of packets thattest every link for network liveness. For reacting to failures,many network operators like Internet proactively test thehealth of the network by pinging between all pairs ofsources.

The life of a packet can be viewed as applying the switchand topology transfer functions repeatedly (Figure 4). Whena packet pk arrives at a network port p, the switch functionT that contains the input port pk.p is applied to pk, producing a list of new packets [pk1, pk2, . . .]. If the packet reachesits destination, it is recorded.

```
function network(packets, switches, Γ)
    for pk0 ∈ packets do
        T ←find_switch(pk0.p, switches)
        for pk1 ∈ T(pk0) do
            if pk1.p ∈ EdgePorts then
                #Reached edge
                record(pk1)
            else
                #Find next hop
                network(Γ(pk1), switches, Γ)
```

II. EXISTING SYSTEM

Testing liveness of a network is a fundamentalproblem for ISPs and large data center operators.Sending probes between every pair of edge ports isneither exhaustive nor scalable . It suffices to find aminimal set of end-to-end packets that traverseeach link. However, doing this requires a way ofabstracting across device specific configurationfiles, generating headers and the links they reach,and finally determining a minimum set of testpackets (Min-Set-Cover). To check enforcing consistency between policy andthe configuration.

Disadvantages Of Existing System: Not designed to identify liveness failures, bugsrouter hardware or software, or performanceproblems.The two most common causes of network failureare hardware failures and software bugs, and thatproblems manifest themselves both as reachabilityfailures and throughput/latency degradation

III. PROPOSED SYSTEM AND METHODOLOGY

Automatic Test Packet Generation (ATPG)framework that automatically generates a minimalset of packets to test the liveness of the underlyingtopology and the congruence between data planestate and configuration specifications. The tool canalso automatically generate packets to testperformance assertions such as packet latency.It can also be specialized to generate a minimal setof packets that merely test every link for networkliveness.Figure 2 shows the block diagram of ATPG system. Thesystem first collects all the forwarding states from the network (step 1).
This usually involves reading the FIBs, ACLsor config files and obtaining the topology. ATPG uses HeaderSpace Analysis [12] to find reachability between all the testterminals (step 2). The result is then used by the test packetselection algorithm to find a minimal set of test packets necessary for complete testing of all the rules in the network(step 3). These packets will be sent periodically in the network by the test terminals (step 4). Once an error is detected, the fault localization algorithm is invoked to narrowdown the cause of the error (step 5).

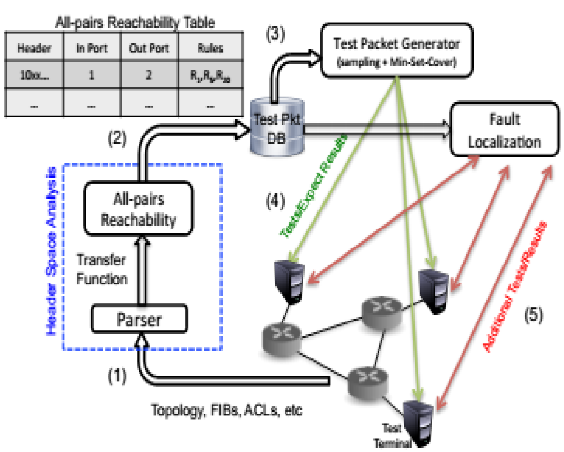


Figure 2: ATPG system block diagram.

A survey of network operators revealingcommon failures and root causes.

- A test packet generation algorithm.
- A fault localization algorithm to isolatefaulty devices and rules.
- ATPG use cases for functional andperformance testing.

  Evaluation of a prototype ATPG systemusing rule sets collected from the Stanford andInternet2 backbones.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. 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 tokenbucket low. It is difficult to troubleshoot a network for threereasons.

a) First, the forwarding state is shared to multiplerouters and firewalls and is determined by the forwardingtables, filter rules, and configuration parameters.

b) Second, theforwarding state is difficult to watch because it requiresmanually logging into every box in the network.

c) Third, theforwarding state is edited simultaneously by differentprograms, protocols and humans.

2. Data plane analysis: These model canautomatically generate packets to test performance assertionslike packet latency ATPG find faults by independently andexhaustively checking all security rules forwarding entries andpacket processing conditions in network.

3. Network Troubleshooting: Some of them added a desire for long runningtests to find jitter or intermittent real-time link capacity monitoringand monitoring tools for network state. In short, while our surveyis small, it helps the hypothesis that network administrators facecomplicated symptoms and causes.

4. ATPG Tool :ATPG generates the minimal number of testpackets so that every forwarding rule in thenetwork is exercised and covered by at least onetest packet. When an error is detected, ATPG usesa fault localization algorithm to determine thefailing rules or links.

5. Network Monitor : To send and receive test data packet network monitor assumesspecial test agents in the network The network monitor gets thedatabase and builds test packets and instructs each different to send the proper packets.

IV. SIMULATION RESULTS

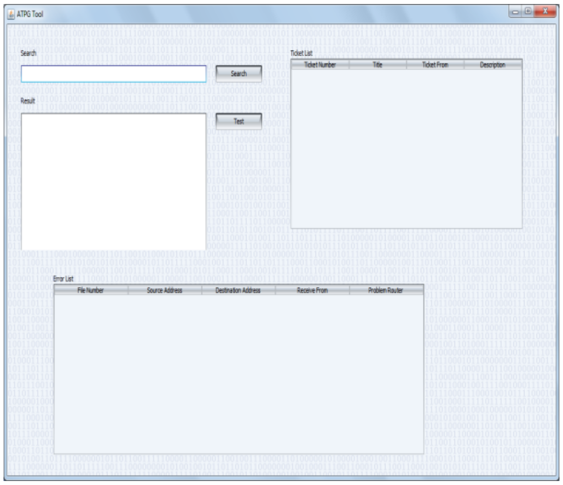


Figure 3. Atpg tool

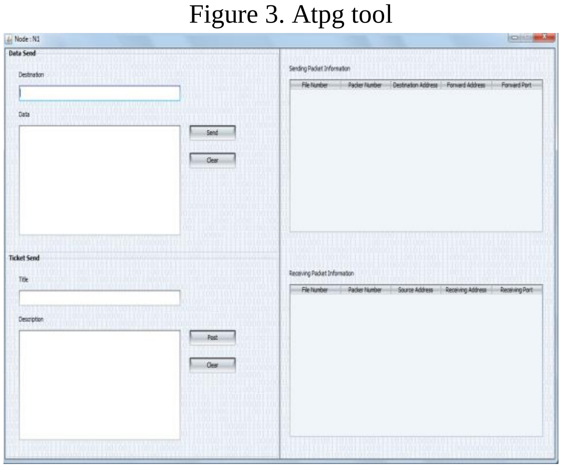


Figure 4 Node1

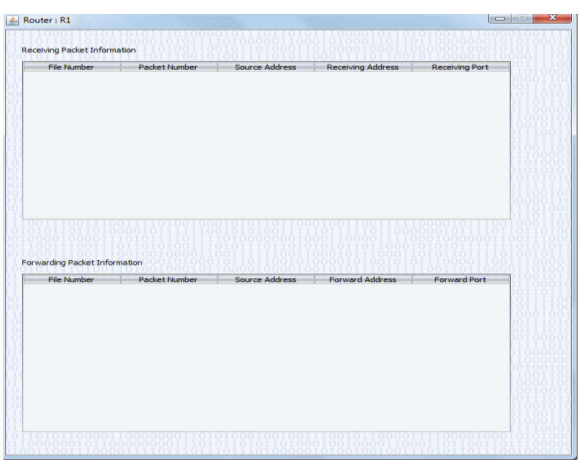


Figure 5 Router 1

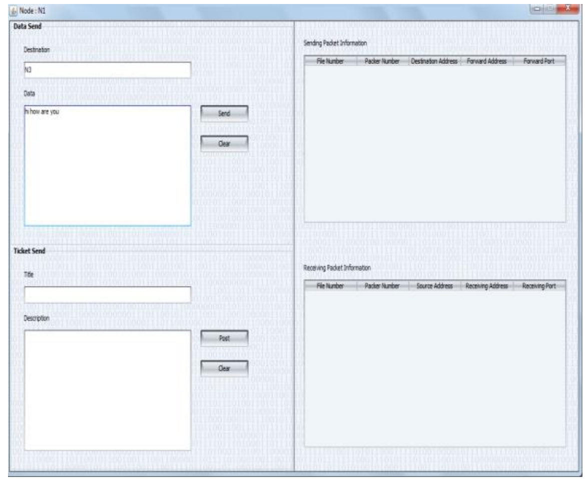


Figure 6 Packet Send

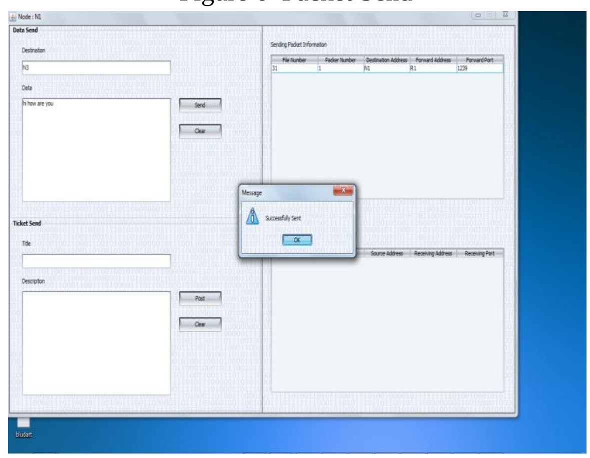


Figure 7

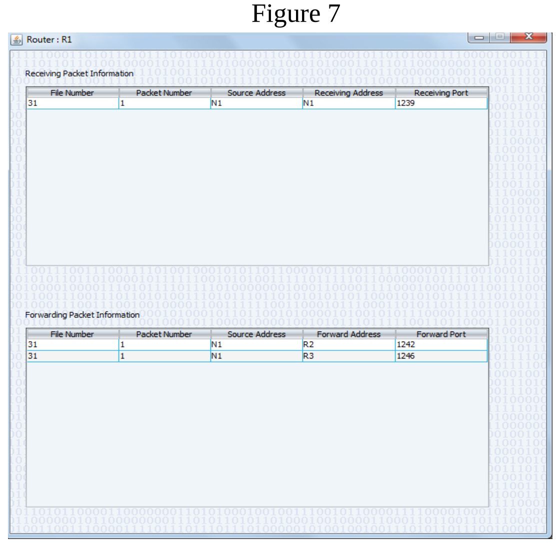


Figure 8 Router R1

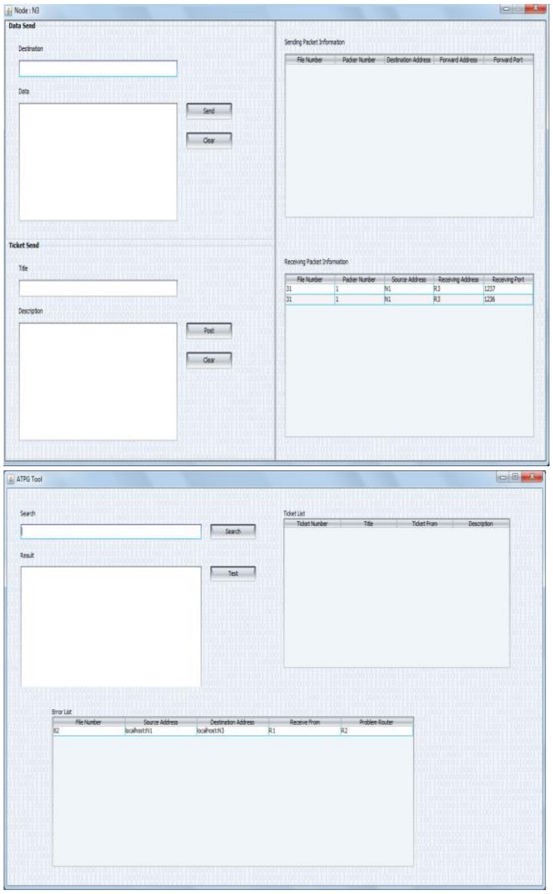
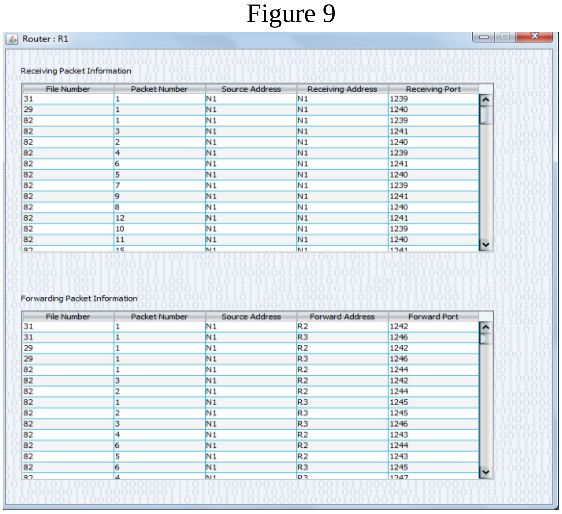


Figure 9



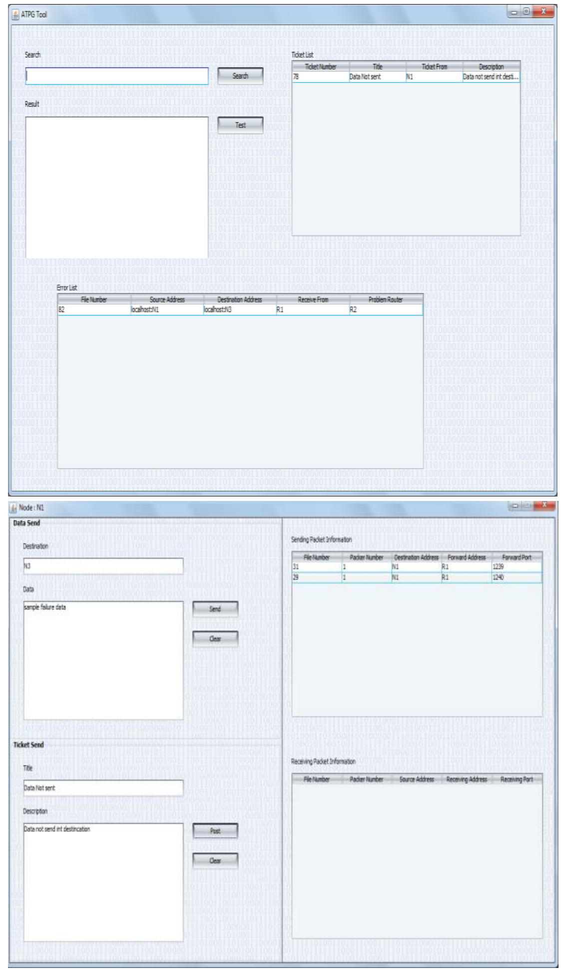


Figure 10

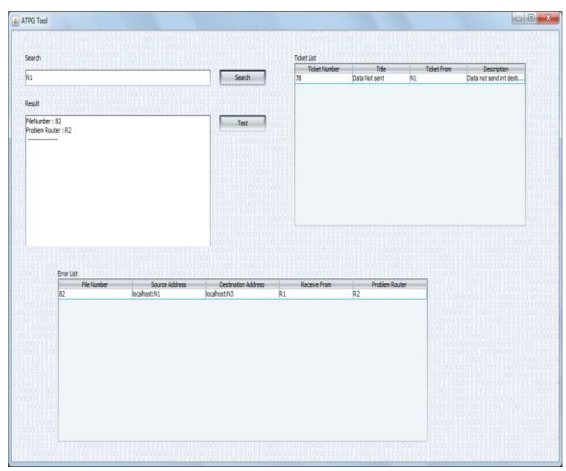


Figure 11

V. CONCLUSION

In present System it uses a method that is neither comprehensivenor scalable. Though it reaches all pairs of edge nodes itcould not detect faults in liveness properties.ATPG, however, goes much further than livenesstesting with the same framework. ATPG can testfor reachability policy (by testing all rulesincluding drop rules) and performance health (byassociating performance measures such as latencyand loss with test packets). Our implementationalso augments testing with a simple faultlocalization scheme also constructed using theheader space framework.

REFERENCES

[1] Hongyi Zeng, PeymanKazemian, George Varghese,ACM, and Nick McKeown, ACM,“Automatic Test Packet Generation”.

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

[3] Kompella, R. R., Greenberg, A., Rexford, J., Snoeren, A.C., 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, 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.

BIODATA

Author

Kattubadi Abdulla presently pursuing his M.Tech CSE in Kottam College of Engineering, Kurnool, Andhra Pradesh, India.

CoAuthor

A.V.Ramakrishna Reddy received M.Tech. Presently working as Assistant Professor in Kottam College of Engineering, Chinnatekuru(V), Kurnool,Andhra Pradesh, India.