Enhancing liveness testing for transferring data packets through using automatic test packet generation

[1]A.Arun, [2]M.Mayuranathan
[1] PG Scholar, [2]Assistan Professor
[1, 2] Department of Computer Science and Engineering,
[1, 2]Valliammai Engineering College, Chennai, India.

***Abstract:*-Networks are getting bigger and more complex, yet administrators rely on incomplete tools such as and to debug problems. We propose an automated and systematic approach for testing and rectify networks called "Automatic evaluates Package Generation" (ATPG). ATPG reads router configurations and generates a device-independent model. The model is used to generate a minimum set of test packets to minimally exerting every link in the network or maximally exerting every rule in the network. Test packets are sent periodically, and detected failures trigger a separate mechanism to localize the revoke. ATPG can detect both functional and renderings problems. ATPG complements but goes beyond earlier work in static checking for which cannot detect liveness or performance faults or fault localization which only localize revoke given liveness results. We describe our prototype ATPG implementation and results on two real-world data sets: Stanford University's backbone network and Internet. We find that a small number of test packets suffice to test all rules in these networks. A sending 4000 test packet 10 times per second consumes less than 1% of link capacity. ATPG code and the datasets are publicly available.**

***Keyword*: ATPG, liveness, Networks.**

I.INTRODUCTION

Networking is the word fundamentally cogitates to computers and their property. It is very often used in the world of computers and their use in different connections. The term networking express the link between two or more computers and their tendency, with the vital purpose of sharing the data stored in the computers, with each other. The networks between the computing tendencies are very public these days due to the launch of assorted hardware and computer software which aid in making the activity much more convenient to build and use.

Fig: 1.1Structure of Networking between the different computers

The discuss about Figure: 1.1 Structure of Networking between the different computer. Its main process of share the Internet to different things and devices.

General Network Techniques - When computers communicate on a network, they send out information packets without knowing if anyone is listening. Computers in a network all have an attached to the network and that is called to be attached to a network bus. What one computer sends out will reach the other computer on the local area network.

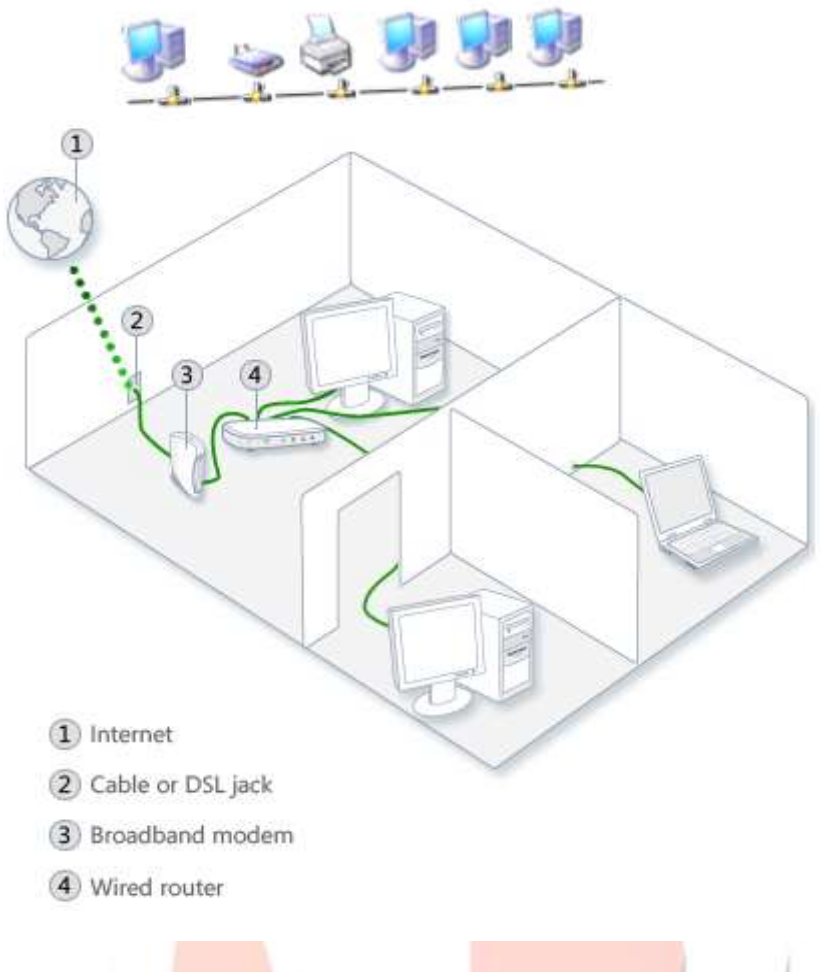

Fig: 1.2 the clear idea about the networking functions

The discus about figure: 1.2 clear ideas of network function and different computers to be able to distinguish between each other, every computer have unique ID called MAC-address Media Access Control Address. This address is not only unique on your network but unique for all tendencies that can be aquiline up to a network. The MAC-address is tied to the hardware and has nothing to do with IP-addresses. Since all computers on the network graduate inversion that is sent out from all other computers the MAC-addresses is primarily used by the computers to filter out incoming network traffic that is addressed to the scratcher computer. When a computer expostulation with another computer on the network, it sends out both the other computers MAC-address and the MAC-address of its own. In that way the receiving computer will not only realize that this parcel is for me but also, who sent this data packet so a return response can be sent to the sender.

Ethernet network as delineate here, all computers hear all network aggregation since they are attached to the same bus. This network structure is called multi-drop. One problem with this network structure is that when you have, let say ten computers on a network and they expostulation attendance and due to that they sends out there data packets randomly, collisions occur when two or more computers sends data at the same time. When that happens data gets imperfect and has to be resent. On a network that is heavy loaded even the resent packets collide with other packets and have to be resent again. In reality this soon takes affect an information problem. If respective computers communicate with each other at high speed they may not be able to utilize more than 25% of the total network information measure since the rest of the bandwidth is used for regressive antecedently corrupted packets. The way to minimize this problem is to use network switches.

II.RELATED NETWORK

Detecting the occurrence and location of performance anomalies is critical to ensuring the effective operation of network infrastructures. In this paper we present a framework for detecting and apposition performance anomalies based on using an active investigate measurement infrastructure deployed on the periphery of a network. Our framework has three components: an algorithm for detection performance oddball on a path, an algorithm for environs which paths to probe at a given time in order to detect performance anomalies where a path is defined as the set of links between two sampling nodes, and an algorithm for designation the links that are causing an identified anomaly on a path The path selection algorithm is designed to enable a interchange between insure that all links in a network are of times monitored to detect performance anomalies, while minimizing probing overhead.[1]

This paper, we develop failure-resilient techniques for monitoring link detain and imbecility in a Service Provider or endeavor IP network. Our two-phased approach attempts to minimize both the monitoring infrastructure costs as well as the additional aggregation due to probe messages. In the first phase, we compute the particular point of a minimal set of monitoring stations such

that all network links are covered, even in the bigness of several link reverting. Afterwards, in the second phase, we compute a minimal set of probe messages that are transmitted by the stations to measure link delays and isolate network faults these approximation ratios are provably very close to the best possible bounds for any algorithm. [2]We present a new symbolic execution tool, KLEE, capable of automatically induce tests that wangle high coverage on a diverse set of complex and environmentally-intense programs. We used KLEE to thoroughly check all 89 stand-alone programs in the GNU COREUTILS utility suite, which form the core user-level environment position on millions of UNIX systems, and arguably are the single most heavily tested set of open-source programs in existence. KLEE-generated tests achieve high line coverage on average over 90% per tool. [3]

The emergence of Open Flow-capable switches enables exciting new network functionality, at the risk of programming errors that make communication less reliable. The centralized programming model, where a single accountant program manages the network, seems to reduce the likelihood of bugs. However, the system is inherently scattered and asynchronous, with events happening at different switches and end hosts, and inevitable delays affecting communication with the controller. In this paper, we present economic, regular techniques for testing unqualified controller programs. Our NICE tool applies model checking to explore the state space of the entire system the controller, the switches, and the hosts. [4]

Network performance tomography, characteristics of the network interior, such as link loss and packet latency, is inferred from unrelated end-to-end measurements. Most work to date is based on employ packet level correlations. However, these methods are often limited in scope-multicast is not widely deployed-or require deployment of additional hardware or software system. Some recent work has been successful in reaching a less detailed goal: identifying the lossiest network links using only unrelated end-to-end sampling. In this paper, we abstract the properties of network performance that allow this to be done and exploit them with a quick and simple deduction algorithm that, with high likely, separate the worst performing links. [5]

III. SYSTEM ANALAYS

Automatic Test Packet Generation (ATPG) framework that self-loading generates a minimal set of collection to test the liveness of the underlying topology and the congruence between data plane state and redundancy description. The tool can also automatically generate packets to test performance assertions such as packet latency. It can also be specialized to generate a minimal set of packets that merely test every link for network liveness.

- A survey of network operators revealing common failures and root causes.
- A test packet generation algorithm.
- A revoke localization algorithm to isolate faulty devices and rules.
- ATPG use cases for functional and performance testing.
- Evaluation of a prototype ATPG system using rule sets collected from the Stanford and Internet2 backbones.

SYSTEM ARCHITECTURE

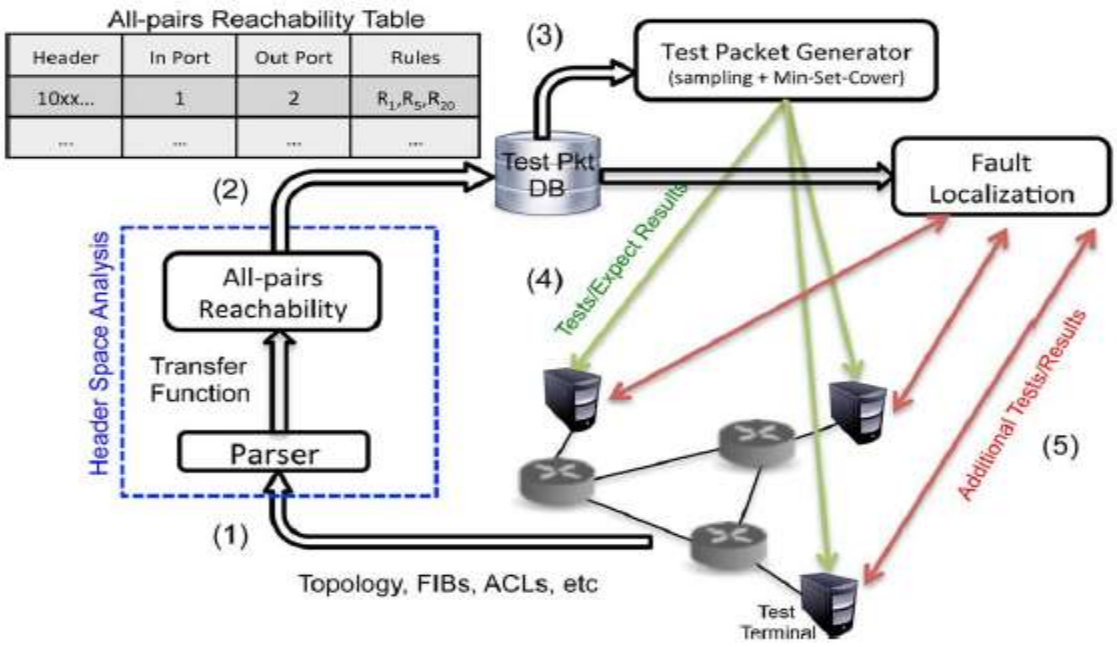

Fig: 1.3 System Architecture diagram

The discuss about figure: 1.3 systems Architecture and clear process of test packet generator and fault localization and pair's reachability table is explain liveness.

SYSTEM PROCESS

- Test Packet Generation

- Generate All-Pairs Reachability Table
- ATPG Tool
- Fault Localization

Test Packet Generation:

We assume a set of test terminus in the network can send and receive test packets. Our goal is to generate a set of test packets to exercise every rule in every switch function, so that any fault will be determined by at least one test packet. This is similar to software test suites that try to test every possible branch in a program. The broader goal can be limited to experiment every link or every queue. When develop test packets, ATPG must respect two key constraints First Port ATPG must only use test terminals that are available) and Header (ATPG must only use headers that each test terminal is permitted to send.

Generate All-Pairs Reach ability Table:

ATPG starts by computing the complete set of packet headers that can be sent from each test terminal to every other test terminal. For each such header, ATPG finds the absolute set of rules it calisthenics along the path. To do so, ATPG applies the all-pairs reach ability algorithm described. On every terminal port, an all- header a header that has all wild carded bits is applied to the relocation function of the first switch connected to each test terminal. Header constraints are applied here.

ATPG Tool:

ATPG generates the minimal number of test packets so that every transmittal rule in the network is exercised and covered by at least one test packet. When an error is detected, ATPG uses a fault localization algorithm to rectify the failing rules or links.

Fault Localization:

ATPG sporadically sends a set of test packets. If test packets fail, ATPG pinpoints the fault(s) that caused the problem. A rule fails if its determined behaviour differs from its awaited discourtesy. ATPG keeps track of where rules fail using a result function “Success” and “failure” base on the nature of the rule: A forwarding rule fails if a test packet is not verbalise to the motivated output port, whereas a drop rule behaves correctly when packets are dropped. Similarly, a link failure is a failure of a forwarding rule in the secretary function. On the other hand, if an output link is congested, failure is captured by the latency of a test packet going above a threshold.

IV.RESULT

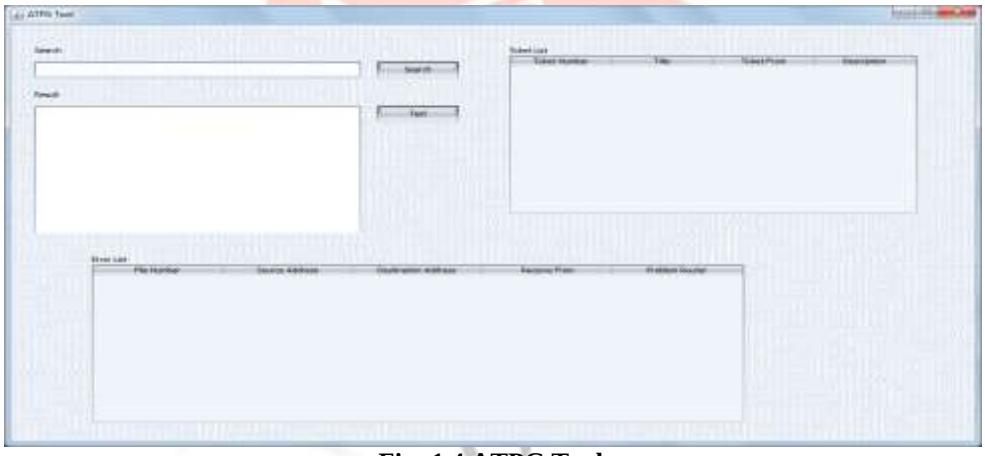

Fig: 1.4 ATPG Tool

The discuss about Figure: 1.4 ATPG Tool. Its use to allocate the how many packets to be transfer one to many processes.

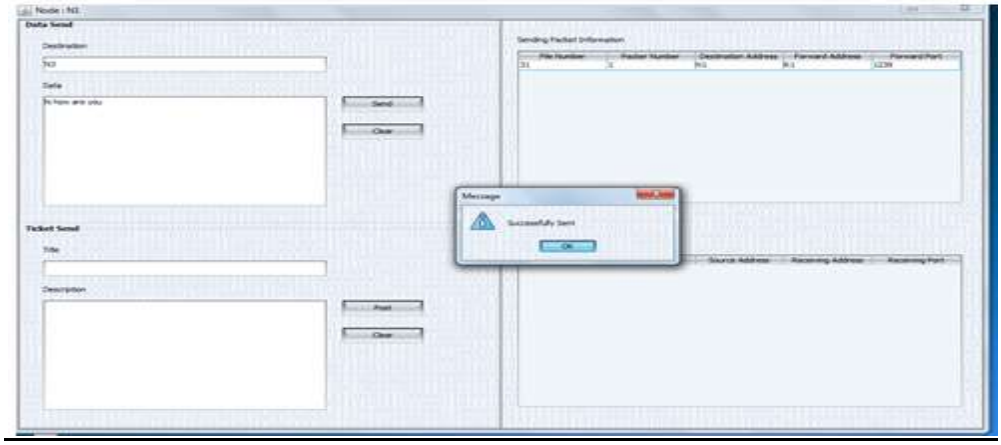

Fig: 1.5 Packet Send

The discuss about Figure: 1.5 Packet send process is share the information one place to another place via data packet.

V.CONCLUSION

Testing liveness of a network is a basic important problem for ISPs and large data center operators. Sending probes between every pair of edge ports is neither exhaustive nor scalable. It satisfies to find a minimal set of end-to-end packets that tramp each link. However, doing this requires a way of abstracting across device specific configuration files, generating beam and the links they locomotive, and finally determining a minimum set of test packets Even the fundamental problem of automatically generating test packets for effective liveness testing involve techniques akin to ATPG.

VI.REFERENCES

[1] P. Barford, N. Duffield, A. Ron, and J. Sommers, "Network performance anomaly detection and localization," InProc. *IEEE INFOCOM*, Apr., pp. 1377–1385.
[2] 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.
[3] C. Cadar, D. Dunbar, and D. Engler, "Klee: Unassisted and automatic generation of high-coverage tests for Complex System programs," in *Proc. OSDI*, Berkeley, CA, USA, 2008, pp. 209–224
[4] M. Canini, D.Venzano, P. Peresini, D.Kostic, and J. Rexford, "A NICE way to test OpenFlow applications," in Proc. *NSDI*, 2012, pp 10-10.
[5] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, "Netdiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data," in *Proc. ACM CoNEXT*, 2007, pp. 18:1–18:12.
[6] N. Duffield, "Network tomography of binary network performance characteristics," *IEEE Trans. Inf. Theory*, vol. 52, no. 12,. pp. 5373–5388, Dec. 2006
[7] N. Duffield, F. L. Presti, V. Paxson, and D. Towsley, "Inferring link loss using striped unicast probes," in *Proc. IEEE INFOCOM*, 2001, vol. 2, pp. 915–923.
[8] 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.
[9] 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.
[10] P. Kazemian, G. Varghese, and N. McKeown, "Header space analysis: Static checking for networks," in *Proc NSDI*, 2012, Pp, 9-9.