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 4, April 2017

Detecting Liveness and Performance Faults in a Network

R.Rajani, P.Sumathi, I.V.S.Sai Srinivas

Associate Professor & HOD, Department of MCA, Narayana Engineering College, Nellore, AP, India

Student, Department of MCA, Narayana Engineering College, Nellore, AP, India

Student, Department of MCA, Narayana Engineering College, Nellore, AP, India

ABSTRACT: Networks are getting larger and more complex, yet administrators rely on rudimentary tools such as and to debug problems. We propose an automated and systematic approach for testing and debugging networks called "Automatic Test Packet Generation"(ATPG).ATPG reads router configuration and generates a device-independent model. The model is used to generate a minimum set of test packets to exercise every link in the network or exercise every rule in the network. Test packets are sent periodically, and detected failures trigger a separate mechanism to localize the fault. ATPG can detect both functional and performance problems. ATPG complements but goes beyond earlier work in static checking or fault localization.

KEYWORDS: Test packet generation, network trouble shooting

I. INTRODUCTION

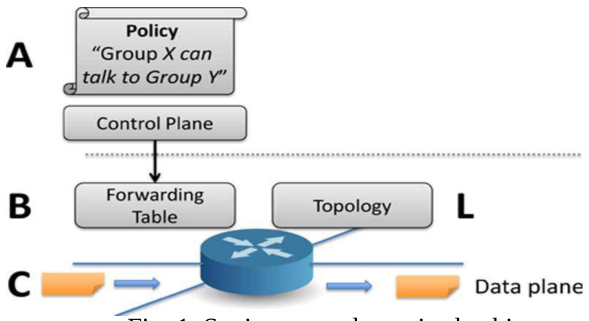


Fig. 1. Static versus dynamic checking

It is very hard to debug networks. Every day, network engineers struggle with router misconfigurations, fiber cuts, faulty interfaces, mislabeled cables, software bugs, intermittent links, and a numerous other reasons that cause networks to misbehave or fail completely. Debugging networks is becoming harder as networks are getting bigger and are getting more complicated.

II. RELATED WORK

The closest related works we know of are offline tools that check invariants in networks. In the data plane, NICE attempts to cover the code paths symbolically in controller applications with the help of simplified switch/host models. In the data plane, Anteater models invariants as Boolean satisfiability problems and checks them against configurations with a SAT solver. Header Space Analysis uses a geometric model to check reachability, detect loops, and verify slicing. ATPG complements these checkers by directly testing the data plane and covering a significant set of dynamic or performance errors that cannot otherwise be captured. By contrast, the primary contribution of ATPG is not fault localization, but determining a compact set of end-to-end measurements that can cover every rule or every link.

Disadvantages of existing system:

- Not designed to identify liveness failures, bugs, router hardware or software, or performance problems.
- The two most common causes of network failure are hardware failures and software bugs, and that problems manifest themselves both as reach ability failures and throughput/latency degradation.

III. SCOPE OF RESEARCH

The two most common symptoms (switch and router software bugs and hardware failure) are best found by dynamic testing. Two metrics capture the cost of network debugging—the number of network-related tickets per month and the average time consumed to resolve a ticket (Fig. 2). There are 35% of networks that generate more than 100 tickets per month. Of the respondents, 40.4% estimate it takes under 30 min to resolve a ticket. However, 24.6% report that it takes over an hour on average.

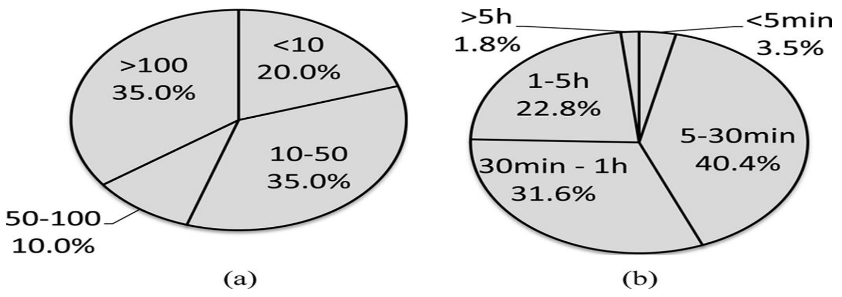


Fig. 2. Reported number of (a) network-related tickets generated per month and (b) time to resolve a ticket.

IV. PROPOSED ALGORITHM

Automatic Test Packet Generation (ATPG) framework automatically generates a minimal set of packets to test the liveness of the underlying topology and the congruence between data plane state and configuration specifications. 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. The block diagram of ATPG is as follows:

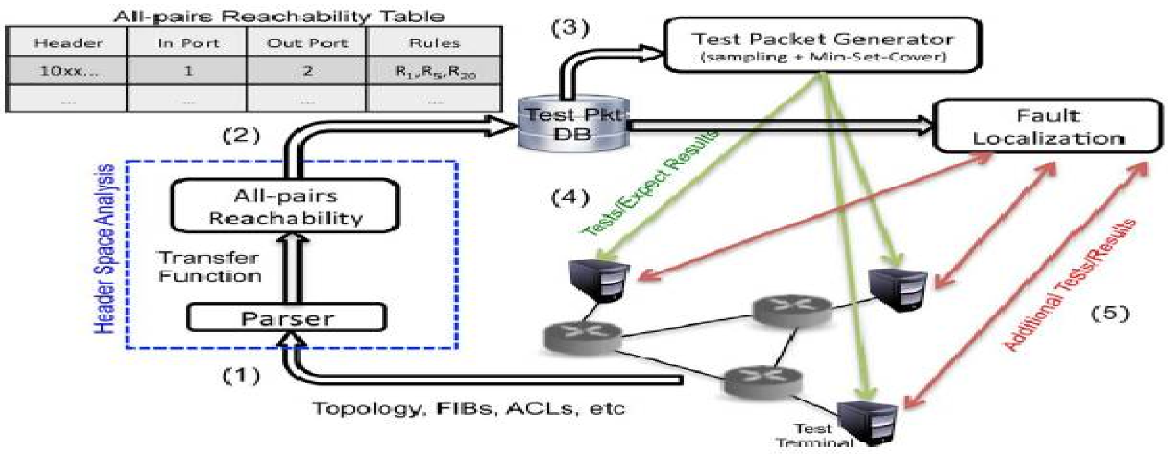


Fig:3 ATPG System Block Diagram

This project is organized as follows. First, we introduced the test packet generation, All -pairs reachability table, fault localization.

V. IMPLEMENTATION

1. Test Packet Generation

We assume a set of test terminals 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 observed by atleast one test packet. This is analogous to software test suites that try to test every possible branch in a program. The broader goal can be limited to testing every link or every queue.

When generating test packets, ATPG must respect two key constraints: 1) Port: ATPG must only use test terminals that are available; 2) Header: ATPG must only use headers that each test terminal is permitted to send.

ATPG chooses test packets using an algorithm we call Test Packet Selection (TPS). TPS first finds all equivalent classes between each pair of available ports. An equivalent class is a set of packets that exercises the same combination of rules. And finally compresses the resulting set of test packets to find the minimum covering set.

2. Generate All-Pairs Reachability Table Module

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 complete set of rules it exercises along the path.

TABLE -I
TEST PACKETS FOR THE EXAMPLE NETWORK DEPICTED IN FIG. 3. p_6 IS STORED AS A RESERVED PACKET

	Header	Ingress Port	Egress Port	Rule History
p_1	dst_ip=10.0/16, tcp=80	P_A	P_B	r_{A1}, r_{B3}, r_{B4}, link AB
p_2	dst_ip=10.1/16	P_A	P_C	r_{A2}, r_{C2}, link AC
p_3	dst_ip=10.2/16	P_B	P_A	r_{B2}, r_{A3}, link AB
p_4	dst_ip=10.1/16	P_B	P_C	r_{B2}, r_{C2}, link BC
p_5	dst_ip=10.2/16	P_C	P_A	r_{C1}, r_{A3}, link BC
(p_6)	dst_ip=10.2/16, tcp=80	P_C	P_B	r_{C1}, r_{B3}, r_{B4}, link BC

ATPG picks at least one test packet in an equivalence class to exercise every (reachable) rule. The simplest scheme is to randomly pick one packet per class. This scheme only detects faults for which all packets covered by the same rule experience the same fault (e.g., a link failure). At the other extreme, if we wish to detect faults specific to a header, then we need to select every header in every class. ATPG therefore selects a minimum subset of the packets such that the union of their rule histories covers all rules. The cover can be chosen to cover all links (for liveness only) or all router queues (for performance only). This is the classical Min-Set -Cover problem. We call the resulting (approximately) minimum set of packets, the *regular test packets*. The remaining test packets not picked for the minimum set are called the *reserved test packets*. In Table IV, $\{p_1 p_2 p_3 p_4 p_5\}$are regular test packets, and $\{p_6\}$is a re-served test packet. Reserved test packets are useful for fault localization.

3. Fault Localization

ATPG periodically sends a set of test packets. If test packets fail, ATPG pinpoints the fault(s) that caused the problem.

- Fault Model: A rule fails if its observed behavior differs from its expected behavior. ATPG keeps track of where rules fail using a result function R .For a rule r ,the result function is defined as “Success” and “failure” depend on the nature of the rule

  R(r, pk)={0, if fails at rule r
  1, if succeeds at rule r

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

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

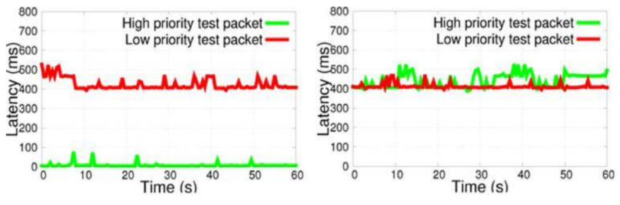
Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

- We divide faults into two categories: action faults and match faults. An action fault occurs when every packet matching the rule is processed incorrectly. Examples of action faults include unexpected packet loss, a missing rule, congestion, and miswiring. On the other hand, match faults are harder to detect because they only affect some packets matching the rule.

VI. EXPERIMENTAL RESULTS

- My prototype was designed to be minimally invasive, requiring no changes to the network except to add terminals at the edge. A new feature could be added to switches/routers, so that a central ATPG system can instruct a router to send/receive test packets.
- In a software defined network (SDN) such as Open Flow, the controller could directly instruct the switch to send test packets and to detect and forward received test packets to the control plane. For performance testing, test packets need to be time-stamped at the routers.
- We detect congestion by measuring the one- way latency of test packets. In My emulation environment, all terminals are synchronized to the host's clock so the latency can be calculated with a single time-stamp and one-way communication.



(a) (b)

Congested Slice	Terminal	Result/Mbps
High	High	11.95
	Low	11.66
Low	High	23.22
	Low	11.78

(c)

Fig.4 Priority testing: Latency measured by test agents when (a) low- or (b) high-priority slice is congested. (c) Available bandwidth measurements when the bottleneck

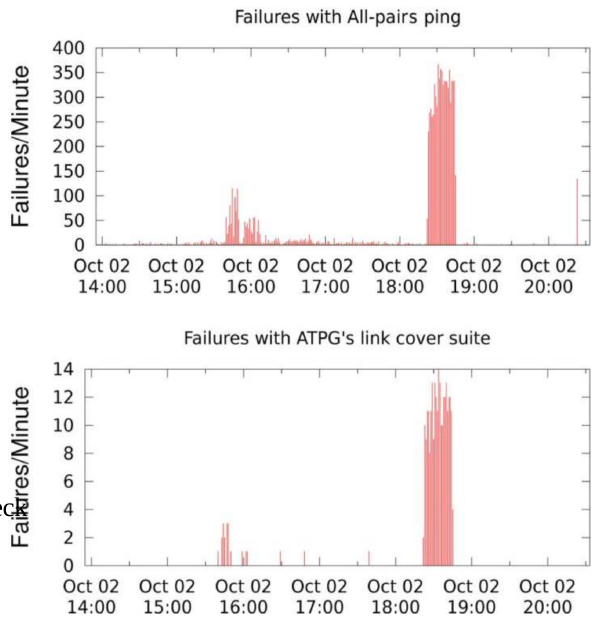


Fig. 5. October 2, 2012 production network outages captured by the ATPG system as seen from the lens of *(top)* an inefficient cover (all -pairs) and *(bottom)* an efficient minimum cover.

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

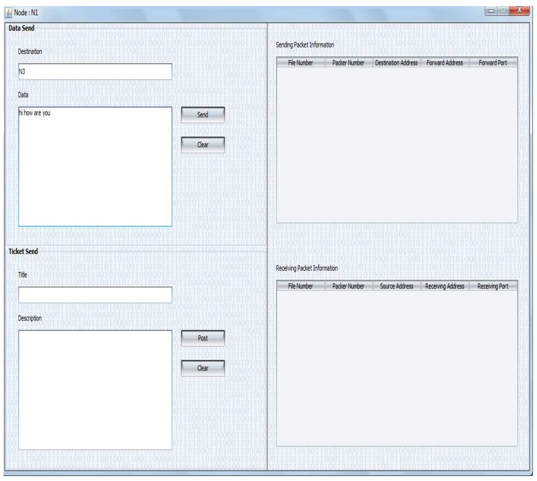
International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

Packet sending:


Node : N1

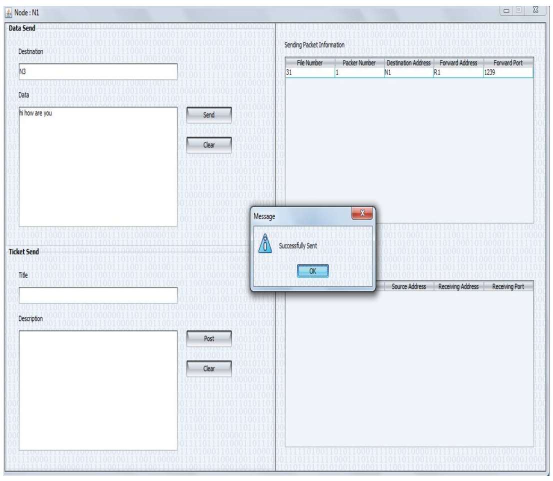
Data Send

Destination: N3

Data: hi how are you

File Number	Packer Number	Destination Address	Forward Address	Forward Port

File Number	Packer Number	Source Address	Receiving Address	Receiving Port


Node : N1

Destination: N3

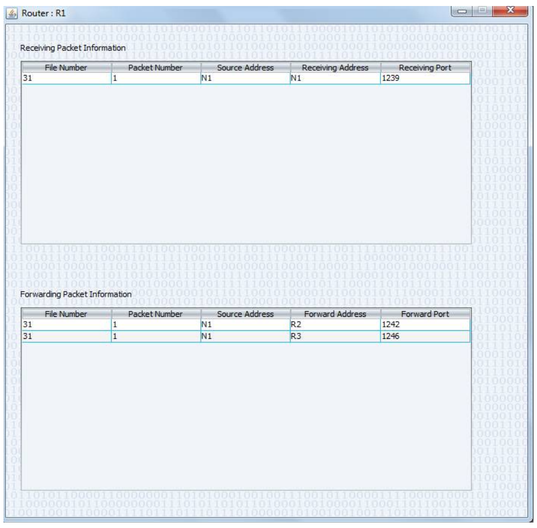
Data: hi how are you

Sending Packet Information

File Number	Packer Number	Destination Address	Forward Address	Forward Port
31	1	N1	R1	1239

Message: Successfully Sent

Packet receiving:

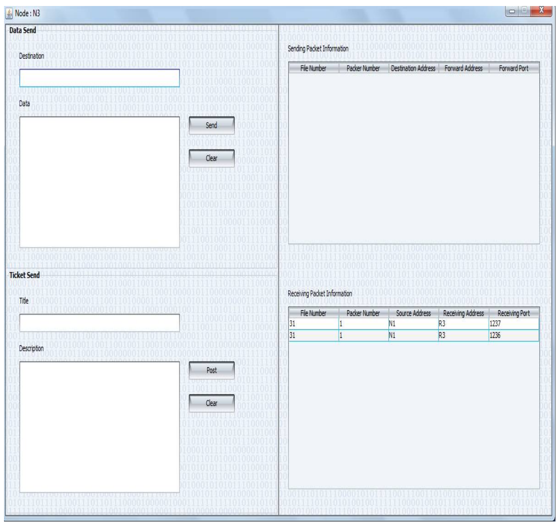

Router : R1

Receiving Packet Information

File Number	Packet Number	Source Address	Receiving Address	Receiving Port
31	1	N1	N1	1239

Forwarding Packet Information

File Number	Packet Number	Source Address	Forward Address	Forward Port
31	1	N1	R2	1242
31	1	N1	R3	1246


Node : N3

Sending Packet Information

File Number	Packer Number	Destination Address	Forward Address	Forward Port

Receiving Packet Information

File Number	Packer Number	Source Address	Receiving Address	Receiving Port
31	1	N1	R3	1237
31	1	N1	R3	1236

VII. CONCLUSION

Testing liveness of a network is a fundamental problem for ISPs and large data center operators. Sending probes between every pair of edge ports is neither exhaustive nor scalable. It suffices to find a minimal set of end-to-end packets that traverse each link. However, doing this requires a way of abstracting across device specific configuration files (e.g., header space), generating headers and the links they reach (e.g., all -pairs reach-ability), and finally determining a minimum set of test packets. Even the fundamental problem of automatically generating test packets for efficient liveness testing requires techniques akin to ATPG. My implementation also augments testing with a simple fault localization scheme also constructed using the header space framework. As in software testing, the formal model helps maximize test coverage while minimizing test packets.

Network managers today use primitive tools such as `Ping` and `traceroute`. My survey results indicate that they are eager for more sophisticated tools. I discovered to our surprise that ATPG was a well-known acronym in

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 4, April 2017

hardware chip testing, where it stands for Automatic Test *Pat-tern* Generation. I hope network ATPG will be equally useful for automated dynamic testing of production networks.

REFERENCES

References for the project Development Were Taken From The Following Books And Websites:
1. ATPG code repository, [Online]. Available: http://eastzone.github. com/atpg/
2. Automatic Test Pattern Generation. 2013 [Online].
Available: http://en.wikipedia.org/wiki/Automatic_test_pattern_generation
3. P. Barford, N. Duffield, A. Ron, and J. Sommers, “Network perfor-mance anomaly detection and localization,” in *Proc. IEEE INFOCOM*, Apr. , pp. 1377–1385.
4. “Beacon,” [Online]. Available: http://www.beaconcontroller.net/
5. 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.
6. C. Cadar, D. Dunbar, and D. Engler, “Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs,” in *Proc. OSDI*, Berkeley, CA, USA, 2008, pp. 209–224.
7. 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.
8. 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..
9. N. Duffield, “Network tomography of binary network performance characteristics,” *IEEE Trans. Inf. Theory*, vol. 52, no. 12, pp. 5373–5388, Dec. 2006.
10. 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.
11. 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.
12. 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.
13. “Hassel, the Header Space Library,” [Online]. Available: https://bit-bucket.org/peymank/hassel-public/
14. Internet2, Ann Arbor, MI, USA, “The Internet2 observatory data col-lections,” [Online]. Available: http://www.internet2.edu/observatory/ archive/data-collections.html
15. 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, no. 4, pp. 537–549, Aug. 2003.
16. P. Kazemian, G. Varghese, and N. McKeown, “Header space analysis: Static checking for networks,” in *Proc. NSDI*, 2012, pp. 9–9.
17. R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren, “IP fault localization via risk modeling,” in *Proc. NSDI*, Berkeley, CA, USA, 2005, vol. 2, pp. 57–70.
18. M. Kuzniar, P. Peresini, M. Canini, D. Venzano, and D. Kostic, “A SOFT way for OpenFlow switch interoperability testing,” in *Proc. ACM CoNEXT*, 2012, pp. 265–276.
19. K. Lai and M. Baker, “Nettimer: A tool for measuring bottleneck link, bandwidth,” in *Proc. USITS*, Berkeley, CA, USA, 2001, vol. 3, pp. 11–11.
20. B. Lantz, B. Heller, and N. McKeown, “A network in a laptop: Rapid prototyping for software-defined networks,” in *Proc. Hotnets*, 2010, pp. 19:1–19:6.
21. F. Le, S. Lee, T. Wong, H. S. Kim, and D. Newcomb, “Detecting network-wide and router-specific misconfigurations through data mining,” *IEEE/ACM Trans. Netw.*, vol. 17, no. 1, pp. 66–79, Feb. 2009
22. H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Kr-ishnamurthy, and A. Venkataramani, “iplane: An information plane for distributed services,” in *Proc. OSDI*, Berkeley, CA, USA, 2006, pp. 367–380.

BIOGRAPHY

Mrs. R. RAJANI is an Associate Professor and heading the department of MCA, Narayana Engineering College, Nellore, AP, India. She is pursuing her Ph.D from Sri Padmavathi Mahila University. She guided many projects for B.Tech and PG students. Her research interests include Datamining, Query Optimization, Computer Networks and Software Engineering etc.

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 4, April 2017

Ms. P. Sumathi is a student pursuing MCA at Narayana Engineering College, Nellore, AP, India. During My final semester project work we prepared this paper for publishing it in an international journal.

Mr. I .V.S. Sai Srinivas is a student pursuing MCA at Narayana Engineering College, Nellore, AP, India. During My final semester project work we prepared this paper for publishing it in an international journal.