RESEARCH ARTICLE OPEN ACCESS

AUTOMATIC TEST PACKET GENERATION

[1]Mrs. Vidhya A.,[2] Ms. Abarna N.,

*[1] M.Phil Research Scholar, PG & Research Department of Computer Science & Information Technology, Arcot Sri Mahalakshmi Women's College, Villapakkam, Vellore, Tamil Nadu, India

*[2]Assistant Professor, PG & Research Department of Computer Science & Information Technology, Arcot Sri Mahalakshmi Women's College, Villapakkam, Vellore, Tamil Nadu, India

--**********************--------------------------------------

Abstract:

Networks are getting larger and more complex, yet administrators rely on rudimentary tools such as ping and trace route to debug problems. We propose an automated and systematic approach for testing and debugging networks called "Automatic Test Packet 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) exercise every link in the network or (maximally) 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 (e.g., incorrect firewall rule) and performance problems (e.g., congested queue). ATPG complements but goes beyond earlier work in static checking (which cannot detect liveness or performance faults) or fault localization (which only localize faults given liveness results). We describe our prototype ATPG implementation and results on two real-world data sets: Stanford University's backbone network and Internet2. We find that a small number of test packets suffice to test all rules in these networks: For example, 4000 packets can cover all rules in Stanford backbone network, while 54 are enough to cover all links. Sending 4000 test packets 10 times per second consume less than 1% of link capacity. ATPG code and the data sets are publicly available.

Keywords— **Automatic Test Packet Generation (ATPG, small number of test packets).**

--**********************--------------------------------------

I. INTRODUCTION

It is notoriously hard to debug networks. Every day, network engineers wrestle with router misconfigurations, fiber cuts, faulty interfaces, mislabelled cables, software bugs, intermittent links, and a myriad other reasons that cause networks to misbehave or fail completely. Network engineers hunt down bugs using the most rudimentary tools (e.g., Ping, trace route, tcpdump, SNMP, and) and track down root causes using a combination of accrued wisdom and intuition. Debugging networks is only becoming harder as networks are getting *bigger* (modern data centers may contain 10 000 switches, a campus network may serve 50 000 users, a 100-Gb/s long-haul link may carry 100 000 flows) and are getting *more complicated* (with over 6000 RFCs, router software is based on millions of lines of source code, and network chips often contain billions of gates). It is a mall wonder that network engineers have been labeled "masters of complexity". Consider two examples. *Example 1:* Suppose a router with a faulty line card starts dropping packets silently. Alice, who administers 100 routers, receives a ticket from several unhappy users complaining about connectivity.

It is not an easy task to debug a network. The network engineers face problems like router misconfigurations, Fiber cut, mislabeled cables, software bug, Faulty interfaces etc. Network engineers try to solve these issues using mostly used tools such as ping and trace route. Debugging networks is getting more and more complex as not only size of networks but also their level of complexity is increasing day by day. ATPG produces a model which is not dependent on devices after reading configuration from routers. Another advantage of ATPG system is that it covers each link and every rule in network with minimum number of test packets.

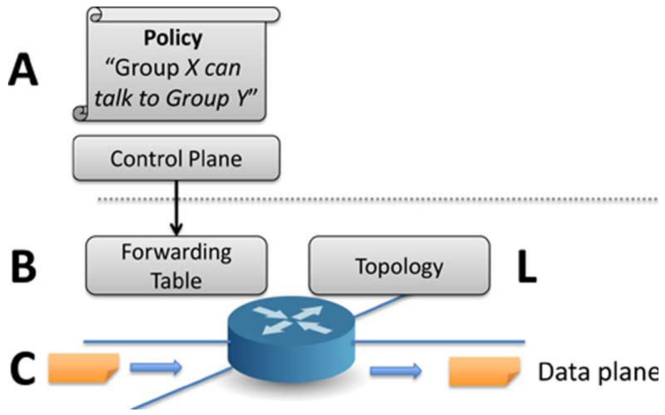


Figure 1.1 Static versus dynamic checking

Fig. 1 is a simplified view of network state. At the bottom of the figure is the forwarding state used to forward each packet, consisting of the L2 and L3 forwarding information base (FIB), access control lists, etc. The forwarding state is written by the control plane (that can be local or remote as in the SDN model [32]) and should correctly implement the network administrator's policy. Examples of the policy include: "Security group X is isolated from security Group Y," "Use OSPF for routing," and "Video traffic should receive at least 1 Mb/s." We can think of the controller compiling the policy (A) into device-specific configuration files (B), which in turn determine the forwarding behavior of each packet (C).

1.1 Problem Definition

- Most common causes of network failure
  1) Hardware failures
  2) Software bugs
- Others failures include reach ability failures and throughput/latency degradation.
- Our goal is to automatically detect these types of failures.

1.2 Objective

- Project is concerned with analyzing all packets across LAN
- Here filtering of packets is carried out by means of filtering rules
- In first rule filtering is carried out by means of IP address.
- In second rule filtering is by means of size of the packet.
- In third rule filtering is carried out by means of data inside the packet.

1.3 Goals

- Automatically generate test packets to test the network state, faults before being noticed by application.
- Augment human wisdom and intuition.
- Reduce the downtime.
- Save money.

II. RELATED WORK

2.1 Network Tomography of Binary Network Performance Characteristics

In network performance tomography, characteristics of the network interior, such as link loss and packet latency, are inferred from correlated end-to-end measurements. Most work to date is based on exploiting packet level correlations, e.g., of multicast packets or unicast emulations of them. However, these methods are often limited in scope-multicast is not widely deployed or requires deployment of additional hardware or software infrastructure.

Some recent work has been successful in reaching a less detailed goal: identifying the lossiest network links using only uncorrelated end-to-end measurements. In this paper we abstract the properties of network performance that allow this to be done and exploit them with a quick and simple inference algorithm that, with high likelihood, identifies the worst performing links. We give several examples of real network performance measures that exhibit the required properties. Moreover, the algorithm is sufficiently simple that we can analyze its performance explicitly.

Advantage:

- Tomographic methods are that they require no participation from network elements other than the usual forwarding of packets.
- This distinguishes them from well-known tools, such as trace route and ping, that require ICMP responses from routers in order to function.
- In practice, ICMP response is restricted by some network administrators
- Identifying the lossiest network links using only uncorrelated end-to-end measurements

Disadvantages

- The disadvantage of high complexity.
- Needing to probe individual routers

2.2 Inferring Link Loss Using Striped Unicast Probes

In this paper we explore the use of end-to-end unicast traffic as measurement probes to infer link-level loss rates. We leverage off of earlier work that produced efficient estimates for link-level loss rates based on end to-end multicast traffic measurements. We design experiments based on the notion of transmitting stripes of packets (with no delay between transmissions of successive packets within a stripe) to two or more receivers. The purpose of these stripes is to ensure that the correlation in receiver observations matches as closely as possible what would have been observed if the stripe had been replaced by a notional multicast probe that followed the same paths to the receivers. Measurements provide good evidence that a packet pair to distinct receivers introduces considerable correlation which can be further increased by simply considering longer stripes. We then use simulation to explore how well these stripes translate into accurate link level loss estimates. We observe good accuracy with packet pairs, with a typical error of about 1%, which significantly decreases as stripe length is increased to 4 packets.

Advantage:

- Measurements provide good evidence that a packet pair to distinct receivers introduces considerable correlation which can be further increased by simply considering longer stripes.

Disadvantages:

- These measurements do not rely on administrative access to internal nodes since the inference can be calculated using only information recorded at the end hosts.
- The key intuition for inferring packet loss is that the arrival of a packet at a given internal node can be directly inferred

III. ALGORITHMS AND TECHNIQUES USED

3.1 Algorithm:

We have a tendency to assume a collection of take a look at terminals within the network will send and receive take a look at packets. Our goal is to come up with a collection of take a look at packets to exercise each rule each switch perform, in order that any fault are determined by a minimum of one take a look at packet. This can be analogous to software package take a look at suites that attempt to take a look at each potential branch in a very program. The broader goal will be restricted to testing each link or each queue. Once generating take a look at packets, ATPG should respect 2 key constraints:

a) Port: ATPG should solely use take a look at terminals that square measure available;
b) Header: ATPG should solely use headers that every take a look at terminal is allowable to transfer. For instance, the network administrator might solely enable employing a specific set of VLANs.

3.2 Fault Localization Algorithm

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. For a rule, the result function is defined as

$$R(r, pk) = \begin{cases} 0, & \text{if } pk \text{ fails at rule } r \\ 1, & \text{if } pk \text{ succeeds at rule } r. \end{cases}$$

We divide faults into two categories: action faults and match Faults. An action fault occurs when every packet matching the rule is processed incorrectly. 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: for example, when a rule matches a header it should not, or when a rule misses a header it should match.

We will only consider action faults because they cover most likely failure conditions and can be detected using only one test packet per rule.

Problem 2 (Fault Localization): Given a list of (pk0, (R (pk0), (pk1, (R (pk1)) ... tuples, find all that satisfies $^{\exists}$pki,R(pki,r)=0.

Step 1: Consider the results from sending the regular test packets. For every passing test, place all rules they exercise into a set of passing rules, *P*. Similarly, for every failing test, place all rules they exercise into a set of potentially failing rules *F*.
Step 2: ATPG next trims the set of suspect rules by weeding out correctly working rules. ATPG does this using the *reserved packets* (the packets eliminated by Min-Set-Cover). ATPG selects reserved packets whose rule histories contain *exactly one* rule from the suspect set and sends these packets.
Step 3: In most cases, the suspect set is small enough after Step 2, which ATPG can terminate and report the suspect set.

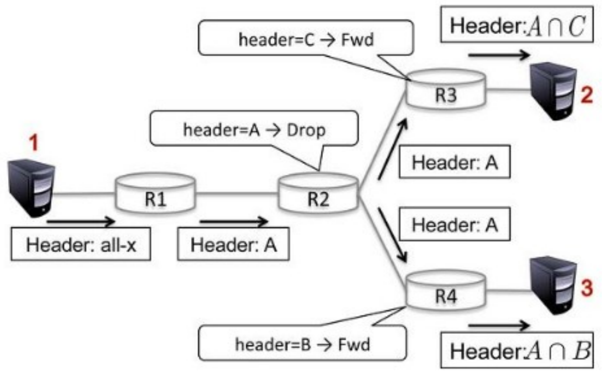


Fig 3.1 Generate packets to test drop rules: "flip" the rule to a broadcast rule in the analysis

3.3 PODEM:

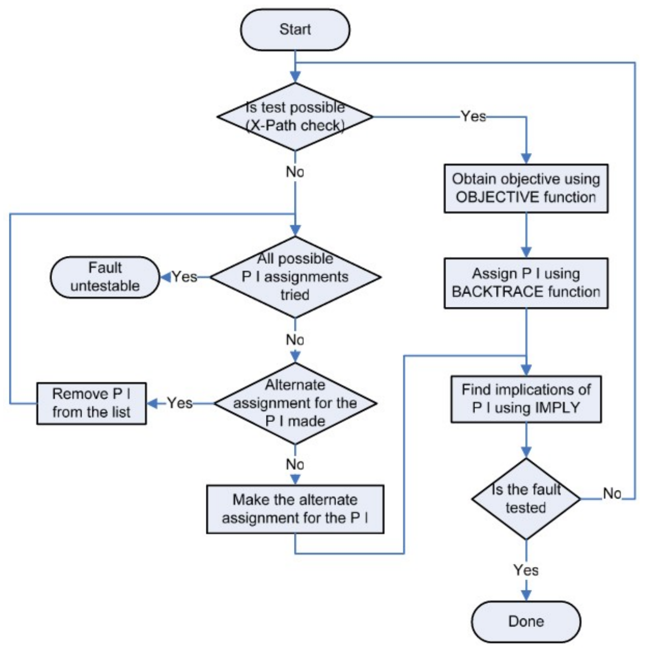

Start
Is test possible (X-Path check)
Yes
No
Obtain objective using OBJECTIVE function
All possible P I assignments tried
Fault untestable
Yes
No
Assign P I using BACKTRACE function
Alternate assignment for the P I made
Remove P I from the list
Yes
No
Find implications of P I using IMPLY
Make the alternate assignment for the P I
Is the fault tested
No
Yes
Done

Fig 3.2 PODEM

IV. ATPG SYSTEM

Based on the network model, ATPG generates the minimal number of test packets so that every forwarding 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 determine the failing rules or links.

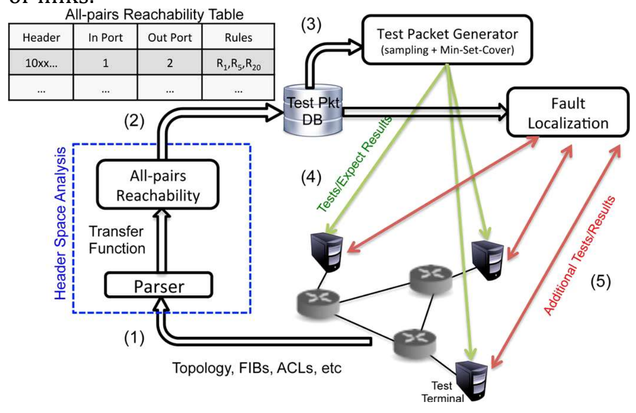


Fig. 4.1 ATPG system block diagram.

ATPG is one in every of the systematic approach uses for debugging network. ATPG generates stripped variety of take a look at packets in order that each forwarding rule the network is exercised and coated by a minimum of 1 take a look at packet.

4.1 ATPG Methods and Algorithm

- ATPG enables testers to distinguish between the correct circuit behavior and the faulty circuit behavior based on inputs patters
- The generated patterns are used to test semiconductor devices for defects after manufacturing
- A defect is an error introduced into a device during the manufacturing process

  - The effectiveness of ATPG is measured by the amount of modeled defects, or fault models, that are detected and the number of generated patterns.
  - The effectiveness of ATGP gives an estimate of test quality
  - A fault model is a mathematical description of how a defect alters design behavior
  - A fault is aid to be detected by a test pattern if, the faulty circuit output differs from the original circuit output

There are two steps that ATPG should lake to detect fault.

Fault activation: Establishes a signal value at the fault model site that is opposite of the value produced by the fault model

Fault propagation: Moves the resulting signal value, or fault effect, forward by sensitizing a path from the fault site to a primary output.

4.2 ATPG: D-Algorithm

- An error is observed due to differing values at a line in the circuit with or without failure. Such a divergence is denoted by values D or D´ to mark differences 1/0 or 0/1, respectively.
- Instead of Boolean values, the set {0, 1, D, D´} is used to evaluate gates and carry out implications.
- A gate that is not on a path between the error and any output does never have a D-value.
- A necessary condition for testability is the existence of a path from the error to an output, where all intermediate gates either have a D-value or are not assigned yet. Such a path is called a potential D-chain.

4.3 ATPG Algorithm Types

Exhaustive Algorithm: For n-input circuit, generate all 2n input patterns. Infeasible, unless circuit is partitioned into cones for logic, with < 15 inputs.

Random Pattern Generation Used to get tests for 60-80% of the faults. The D-algorithm or other ATPG algorithms used for the rest. Fault simulation is necessary in order to select useful patterns. Weighted random patterns: 0 and 1 are not equally likely.

4.4 Automatic Test packet generation and fault localization

Automatic take a look at Packet Generation (ATPG) structure that consequently produces a negligible arrangement of bundles to check the basic's livener's topology and also the coinciding between data plane state and style determinations. The equipment will likewise naturally produce bundles to check execution affirmations, for instance, parcel dormancy. It will likewise be specific to provide a negligible arrangement of parcels that solely take a look at every association for system liveners.

- A survey of network operators revealing common failures and route causes take a look at packet generation algorithmic program.
- A fault localization algorithmic program to isolate faulty device and rules.
- ATPG use cases for useful &amp; performance testing

Evaluation of an example ATPG system victimization rule sets collected from the Stanford and internet2 backbones.

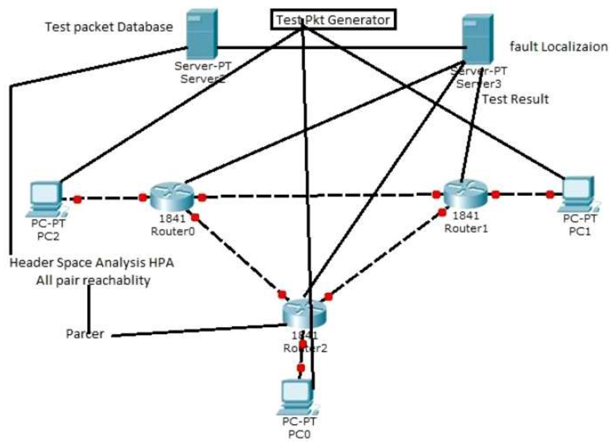


Fig 4.2 Automatic Test packet generation and fault localization

V. NETWORKING

Networking is the word basically relating to computers and their connectivity. It is very often used in the world of computers and their use in different connections. The term networking implies the link between two or more computers and their devices, with the vital purpose of sharing the data stored in the computers, with each other. The networks between the computing devices are very common these days due to the launch of various hardware and computer software which aid in making the activity more convenient to build and use.

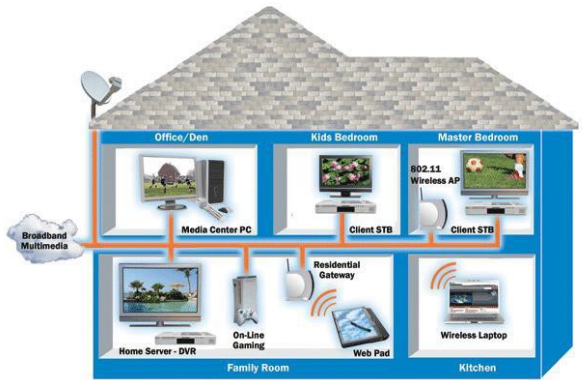


Figure 5.1: Structure of Networking

When computers communicate on a network, they send out data packets without knowing if anyone is listening. Computers in a network all have a connection to the network and that is called to be connected to a network bus. What one computer sends out will reach all the other computers on the local network. Above diagrams show the clear idea about the networking functions.

5.1 Characteristics of Networking

The following characteristics should be considered in network design and ongoing maintenance:

1. Availability is typically measured in a percentage based on the number of minutes that exist in a year. Therefore, uptime would be the number of minutes the network is available divided by the number of minutes in a year.
2. Cost includes the cost of the network components, their installation, and their ongoing maintenance.
3. Reliability defines the reliability of the network components and the connectivity between them. Mean time between failures (MTBF) is commonly used to measure reliability.
4. Security includes the protection of the network components and the data they contain and/or the data transmitted between them.
5. Speed includes how fast data is transmitted between network end points (the data rate).
6. Scalability defines how well the network can adapt to new growth, including new users, applications, and network components.
7. Topology describes the physical cabling layout and the logical way data moves between components.

5.2 Advantages of Networking

- Easy Communication
- Ability to Share Files, Data and Information
- Sharing Hardware
- Sharing Software
- Security
- Speed

5.3 NETWORK DESIGN

As mentioned in the last section, the automatic test packet generation (ATPG) system makes use of geometric model of header space analysis. This section explains some of the key terms associated with geometric framework of header space analysis.

5.3.1 Packet

Packet in a network can be described as a tuple of the form (port, header) in such a way that, it is the job of port to show position of packet in a network at instantaneous time. Each one of the port is allotted with one and only one unique number.

5.3.2 Switch

Another term used in geometric model of header space analysis is switches. It is the job of switch transfer Function T, to model devices in a network. Example of devices can be switches or routers. There is a set of forwarding rules contained in each device, which decides how the packets should be processed.

5.3.3 Rules

Piece of work for rules is generation of list of one or more output packets associated with those output ports to which the packet is transferred, and explain how fields of port are modified. In other words, rules explains how the region of header space at entrance in changed into region of header space at exit

5.3.4 Rule History

At any moment, every packet has its own rule history, which can be described as ordered list of rules packet have matched up to that point as it covers the network. Rule history provides necessary and important unprocessed material for automatic test packet generation (ATPG). That is the reason why it is fundamental to ATPG.

5.3.5 Topology

The network topology is modeled by topology transfer function. The topology transfer function gives the specification about which two ports are joined by links. Links are nothing but rules that forwards a packet from source to destination with no modification.

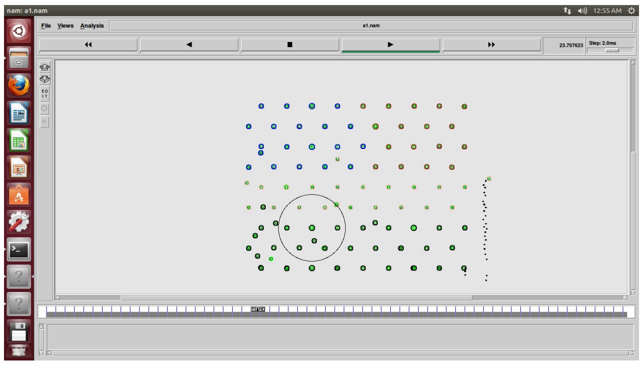
5.3.6 Life of a Packet

One can see life of a packet as carrying out or executing switch transfer function and topology transfer function at length. When a particular packet comes in a network port p, firstly a switch function is applied to that packet.

V EVALUATION RESULT

In this section, three experiments were designed to evaluate the performance of the improved clustering algorithm. The simulating program was developed by our team using NS2. In the following experiments, Most of studies only consider that wireless sensor networks are equipped with only Omni-directional antennas, which can cause high collisions. It is shown that the per node throughput in such networks is decreased with the increased number of nodes. Thus, the transmission with multiple short - range hops is preferred to reduce the interference. However, other studies show that the transmission delay increases with the increased number of hops. Found that using directional antennas not only can increase the throughput capacity but also can decrease the delay by reducing the number of hops.

Contains both merits and limitations when people use it to simulate WSNs. To the merits, firstly as a non-specific network simulator, NS-2 can support a considerable range of protocols in all layers. For example, the ad-hoc and WSN specific protocols are provided by NS-2. Secondly, the open source model saves the cost of simulation, and online documents allow the users easily to modify and improve the codes. However, this simulator has some limitations. Firstly, people who want to use this simulator need to familiar with writing scripting language and modeling technique; the Tool Command Language is somewhat difficult to understand and write. Secondly, sometimes using NS-2 is more complex and time-consuming than other simulators to model a desired job. Thirdly, NS-2 provides a poor graphical support, no Graphical User Interface (GUI) the users have to directly face to text commands of the electronic devices. Fourthly, due to the continuing changing the code base, the result may not be consistent, or contains bugs.

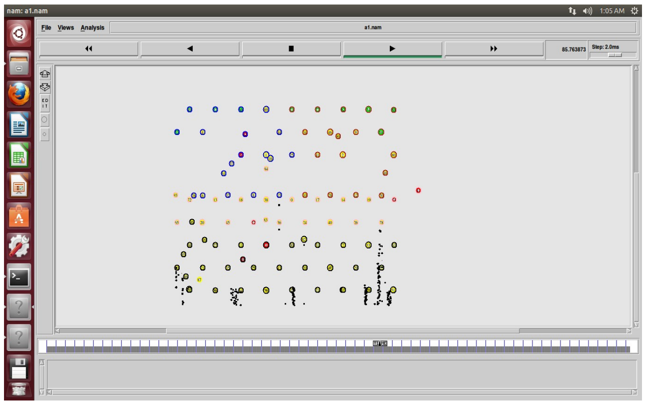


5.1 Packet Redundancy check

For the experimental results presented in this section, a network of sensors randomly distributed in a 1000m * 1000m field is considered. The number of sensors in the network, i.e. the network size, is kept at 100, 200, 300 and so on. Each sensor has an initial energy of 1.Joule and the base station is located at (250, 330). Each sensor generates packets of size 1000 bits. The energy model for the sensors is based on the first order radio model

Parameter	Description
RoutingProtocols	(Proposed)
Channel Type	Wireless Channel
Link Layer type	LL
Interface Queue type	DropTail/priQueue
Max interface priority queue	50
Network Interface Type	Phy/wirelessphy
MAC type	802.11
Max packet in IFQ	50
No of mobile nodes	100
Simulator	NS2.35
DataRate	4Packets/S
TCP/IPLayer	NetworkLayer
Node to NodeDistance	Random
PropagationModel	Tworay Ground
groundInitial EnergyofNode	100J

Table5.1: Simulation Table



5.2 Automatic Packet filtering

X- Axis :Maximum number of retransmission

Y- AXIS :Success RateEnd-to-end reliability achieved by options. Each line represents number of redundant code words for 8 original messages. RF means route fixis use Above graph shows the reliability each option (link-level retransmission, erasure code, and route fix) achieves. The x-axis shows the number of retransmissions and whether route fix isused. Each curve represents how many redundant.

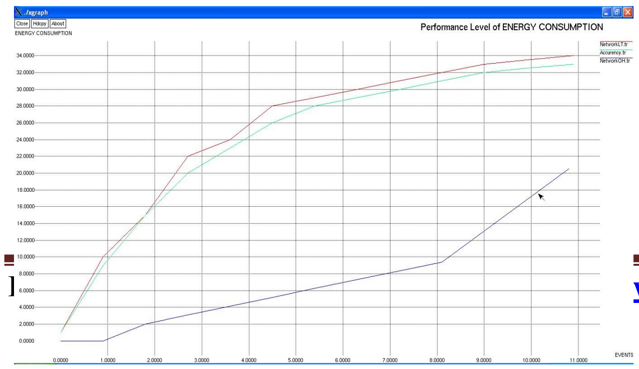


Fig 5.3: X- Axis :Maximum number of retransmission

Above graph shows the normalized overhead for each option with same x-axis andlegends.First observation is that link level retransmissions should be used in any case. With no transmissions, the reliability isso low that the effect of redundancy is negligible (in spite of adding overhead, as in graph 2). The low number (less than 30%) seen in graph1, is close to the expected for a five hop transmission over links with 26.28% loss rate. When using at most 1 per-link retransmission, not only does the success rate go substantially up, but also does the effect of adding redundancy increase.

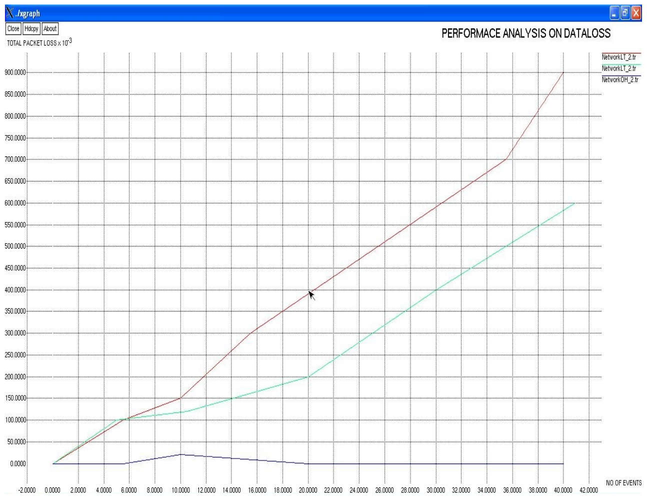


Fig 5.4: Performance Analysis

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 (Min-Set-Cover). Even the fundamental problem of automatically generating test packets for efficient liveness testing requires techniques akin to ATPG.

ATPG, however, goes much further than liveness testing with the same framework. ATPG can test for reach ability policy (by testing all rules including drop rules) and performance health (by associating performance measures such as latency and loss with test packets). Our 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. Our results show that all forwarding rules in Stanford backbone or Internet2 can be exercised by a surprisingly small number of test packets (40,000 for Stanford, and 40000 for Internet2).

Network managers today use primitive tools such as and trace route. Our survey results indicate that they are eager for more sophisticated tools. Other fields of engineering indicate that these desires are not unreasonable: For example, both the ASIC and software design industries are buttressed by billion- dollar tool businesses that supply techniques for both static (e.g., design rule) and dynamic (e.g., timing) verification. In fact, many months after we built and named our system, we discovered to our surprise that ATPGwas awell-known acronym in hardware chip testing, where it stands for Automatic Test Pattern Generation. We hope network ATPG will be equally useful for automated dynamic testing of production networks.

FUTURE WORK:

ATPG provides better solution for network organizers hanging on old tools for computing a network. ATPG has a favorable future opportunity considering, ATPG is blessed with ascendency of overcoming not only functional but also execution blemish. Combined with all these upper hands there are few issues which remained to be addressed in the future such as, ATPG cannot cope with routers with a change in internal state; ATPG has restriction that it can rightly model rules only when parameters along hash function are known; Another problem is of dealing with rules that are out of sight; ATPG fails to reveal errors which exists for time less than time between two consecutive tests; While using sampling at times ATPG can fail to reach some flaws.

REFERENCES:

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 performance anomaly detection and localization," in Proc. IEEE INFOCOM, Apr. , pp. 1377–1385.
4. 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.
5. 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.
6. M. Canini, D.Venzano, P. Peresini, D.Kostic, and J. Rexford, "A NICE way to test Open Flow applications," in Proc. NSDI, 2012, pp. 10–10.
7. [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..
8. N. Duffield, "Network tomography of binary network performance characteristics," IEEE Trans. Inf. Theory, vol. 52, no. 12, pp. 5373–5388, Dec. 2006.
9. 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.
10. 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.
11. 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.
12. "Hassel, the Header Space Library," [Online]. Available: https://bitbucket.org/peymank/hassel-public/
13. Internet2, Ann Arbor, MI, USA, "The Internet2 observatory data collections," [Online]. Available: http://www.internet2.edu/observatory/archive/ data-collections.html
14. M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," IEEE/ACM Trans. Netw., vol. 11, no. 4, pp. 537–549, Aug. 2003.