International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 09, September 2015
Available at http://internationaljournalofresearch.org

Spontaneous Investigation packet Creation

Vempati Sudheshna[1] & V. Sujatha[2]

[1]M.Tech (C.S.E), Roll no: 13NN1D5812, Vignan's Nirula Institute of Technology & Science for Women, Pedapalakaluru, Guntur-522005,AP.

[2]Assistant Professor, Vignan's Nirula Institute of Technology & Science for Women, Pedapalakaluru, Guntur-522005,AP.

ABSTRACT

Networks are getting larger and more complex, yet administrators rely on rudimentary basic tools to debug problems. We offer an automated and methodical tactic for investigationing and debugging networks, it is called "Spontaneous Investigation Package Creation" (SIPC). SIPC reads router configurations and creates a device-independent model. The model is used to create a least set of investigation packages to use every link in the network or exercise every rule in the network. Investigations are done occasionally, and detected failures trigger a separate mechanism to restrict the fault. SIPC can detect both functional like incorrect firewall rule and performance problems like congested queue. SIPC complements but goes beyond earlier work in static checking which cannot detect live ness or performance faults or fault localization which only restricts faults given live ness results). We describe our prototype SIPC implementation and results on two real-world data sets: AQ & T University's backbone network and Internet2. We find that a small number of investigation packages suffice to investigation all rules in these networks: For instance, 4000 packages can cover all rules in AQ & T backbone network, while 54 are enough to cover all links. Sending 4000 investigation packages 10 times per second consume less than 1% of link capacity. SIPC code and the data sets are publicly available.

Index Terms—Data plane analysis; network repairing; and investigation package creation

INTRODUCTION

It is scandalously hard to fix networks. Every day, network engineers struggle with router misconfigurations, fiber cuts, defective interfaces, mislabeled cables, software viruses, sporadic links, and countless other reasons that cause networks to play up or fail wholly. Network engineers hunt down bugs using the most basic tools like SNMP, and track down root causes using a mixture of accrued wisdom and insight. Repairing networks is only becoming harder as networks are getting *bigger* (modern data centers may contain 12,000 switches, a campus network may serve 70000 users, a 100-Gb/s long-haul link may carry 123000 flows) and are getting *more complicated* (with over 6200 RFCs, router software is based on millions of lines of source code, and network chips often contain billions Gates).

Example 1: Presume a router with a defective line card starting to drop packages silently. Ameer, who manages 120 routers, receives a coupon from numerous unfortunate users complaining about connectivity. Ameer inspects each router to see if the configuration was changed recently and determines that the configuration was intact. Next, Ameer uses his awareness of the topology to

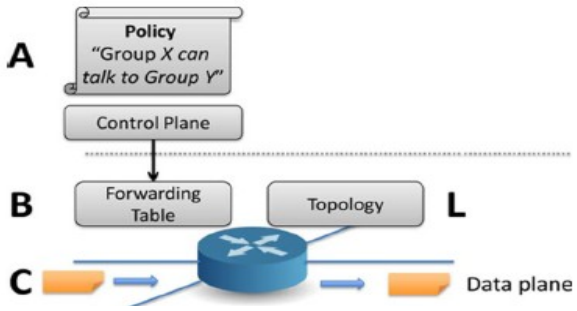
International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 09, September 2015
Available at http://internationaljournalofresearch.org

triangulate the faulty device with ping and trace rout. Finally, he calls a associate to replace the line card.

Example 2: Suppose that video traffic is mapped to a specific queue in a router, but packages are fell because the token bucket rate is too low. It is not at all clear how Ameer can track down such a *performance fault* using ping and trace rout.

Repairing a network is difficult for three reasons. First, the progressing state is distributed across multiple routers and firewalls are defined by their progressing tables, filter rules, and other configuration parameters. Second, the progressing state is hard to observe because it typically needs manually logging into every box in the network. Third, there are many diverse programs, protocols, and humans updating the progressing state concurrently. When Ameer uses ping and trace rout, he is spending a rudimentary lens to investigate the current advancing state for clues to track down the disaster.

The procedure include: "Security group X is isolated from security Group Y," "Use SIPC for routing," and "Video traffic should get 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 progressing behavior of each package (C). To ensure the network behaves as designed, all three steps should remain stable at all times, i.e.,A = B = C . In addition, the topology, shown to the bottom right in the figure, should also satisfy a set of live ness properties. Minimally L requires that sufficient links and nodes are working; if the control plane postulates that a laptop can access a server, the desired outcome can fail if links fail. It can also specify performance guarantees that detect flaky links.



In fact, we learned from a survey of 61 network operators that the two most mutual causes of network failure are hardware failures and software bugs, and that problems manifest themselves *both* as reachability failures and throughput / latency degradation. Our goal is to automatically detect these types of failures. The main contribution of this paper is what we call a (SIPC) framework that *spontaneously* creates a minimal set of packages to investigation the live ness of the underlying topology *and* the congruence between data plane state and configuration specifications. The tool can also recurrently make packages to investigation *performance* assertions such as package latency. In Example 1, instead of Ameer manually deciding which ping packages to send, the tool does so periodically on her behalf. In Example 2, the tool determines that it must send packages with certain legends to "exercise" the video queue, and then determines that these packages are being dropped. SIPC detects and diagnoses errors by independently and fully *investigating* all progressing entries, firewall rules, and any package processing rules in the network. In SIPC, investigation packages are created algorithmically from the device configuration files and FIBs, with the least number of packages required for complete coverage. Investigation packages are fed into the network so that every rule is drilled directly from the data plane. Since SIPC treats links just like normal progressing rules, its full coverage guarantees

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 09, September 2015
Available at http://internationaljournalofresearch.org

investigating of every link in the network. It can also be specialized to generate a least set of packages that merely investigation every link for network aliveness. At least in this basic form, we feel that SIPC or some similar technique is important to networks: Instead of *reacting* to letdowns, many network operators such as Internet2 [14] *proactively* check the health of their network using pings between all pairs of sources. Though, Planet Lab [30] organizations can customize SIPC to meet their needs; for specimen, they can choose to merely check for network live ness or check every rule to ensure security policy. SIPC can be modified to check only for reachability or for performance as well. SIPC can adapt to constraints such as requiring investigation packages from only a few places in the network or using special routers to generate investigation packages from every port. SIPC can also be tuned to allocate more investigation packages to exercise more critical rules. For sample, a healthcare network may dedicate more investigation packages to Firewall rules to ensure HIPPA compliance.

Category	Avg	% of ≥ 4
Reachability Failure	3.67	56.90%
Throughput/Latency	3.39	52.54%
Intermittent Connectivity	3.38	53.45%
Router CPU High Utilization	2.87	31.67%
Congestion	2.65	28.07%
Security Policy Violation	2.33	17.54%
Forwarding Loop	1.89	10.71%
Broadcast/Multicast Storm	1.83	9.62%

(a)

Category	Avg	% of ≥ 4
Switch/Router Software Bug	3.12	40.35%
Hardware Failure	3.07	41.07%
External	3.06	42.37%
Attack	2.67	29.82%
ACL Misconfig.	2.44	20.00%
Software Upgrade	2.35	18.52%
Protocol Misconfiguration	2.29	23.64%
Unknown	2.25	17.65%
Host Network Stack Bug	1.98	16.00%
QoS/TE Misconfig.	1.70	7.41%

(b)

It is scandalously hard to fix networks. Every day, network engineers struggle with router misconfigurations, fiber cuts, defective interfaces, mislabeled cables, software viruses, sporadic links, and countless other reasons that cause networks to play up or fail wholly. Network engineers hunt down bugs using the most basic tools like SNMP, and track down root causes using a mixture of accrued wisdom and insight. Repairing networks is only becoming harder as networks are getting *bigger* (modern data centers may contain 12,000 switches, a campus network may serve 70000 users, a 100-Gb/s long-haul link may carry 123000 flows) and are getting *more complicated* (with over 6200 RFCs, router software is based on millions of lines of source code, and network chips often contain billions Gates).

Example 1: Presume a router with a defective line card starting to drop packages silently. Ameer, who manages 120 routers, receives a coupon from numerous unfortunate users complaining about connectivity. Ameer inspects each router to see if the configuration was changed recently and determines that the configuration was intact. Next, Ameer uses his awareness of the topology to triangulate the faulty device with ping and trace rout. Finally, he calls a associate to replace the line card.

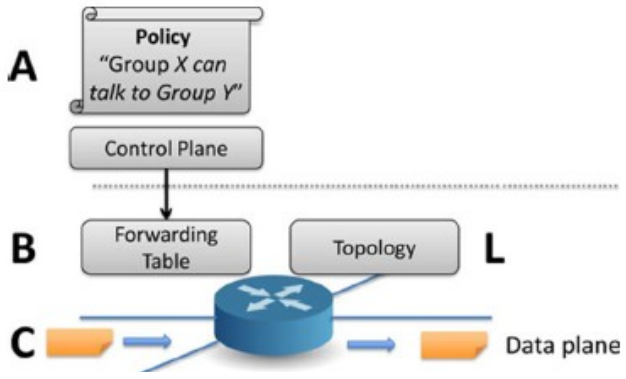
Example 2: Suppose that video traffic is mapped to a specific queue in a router, but packages are fell because the token bucket rate is too low. It is not at all clear how Ameer can track down such a *performance fault* using ping and trace rout.

Repairing a network is difficult for three reasons. First, the progressing state is distributed across multiple routers and firewalls are defined by their progressing tables, filter rules, and other configuration parameters. Second, the progressing state is hard to observe because it typically needs manually logging into every box in the network. Third, there are many diverse programs, protocols, and humans updating the progressing state

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 09, September 2015
Available at http://internationaljournalofresearch.org

concurrently. When Ameer uses ping and trace rout, he is spending a rudimentary lens to investigate the current advancing state for clues to track down the disaster.

The procedure include: "Security group X is isolated from security Group Y," "Use SIPC for routing," and "Video traffic should get 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 progressing behavior of each package (C). To ensure the network behaves as designed, all three steps should remain stable at all times, i.e.,A = B = C . In addition, the topology, shown to the bottom right in the figure, should also satisfy a set of live ness properties. Minimally L requires that sufficient links and nodes are working; if the control plane postulates that a laptop can access a server, the desired outcome can fail if links fail. It can also specify performance guarantees that detect flaky links.



In fact, we learned from a survey of 61 network operators that the two most mutual causes of network failure are hardware failures and software bugs, and that problems manifest themselves *both* as reachability failures and throughput / latency degradation. Our goal is to automatically detect these types of failures. The main contribution of this paper is what we call a (SIPC) framework that *spontaneously* creates a minimal set of packages to investigation the live ness of the underlying topology *and* the congruence between data plane state and configuration specifications. The tool can also recurrently make packages to investigation *performance* assertions such as package latency. In Example 1, instead of Ameer manually deciding which ping packages to send, the tool does so periodically on her behalf. In Example 2, the tool determines that it must send packages with certain legends to "exercise" the video queue, and then determines that these packages are being dropped. SIPC detects and diagnoses errors by independently and fully *investigating* all progressing entries, firewall rules, and any package processing rules in the network. In SIPC, investigation packages are created algorithmically from the device configuration files and FIBs, with the least number of packages required for complete coverage. Investigation packages are fed into the network so that every rule is drilled directly from the data plane. Since SIPC treats links just like normal progressing rules, its full coverage guarantees investigating of every link in the network. It can also be specialized to generate a least set of packages that merely investigation every link for network aliveness. At least in this basic form, we feel that SIPC or some similar technique is important to networks: Instead of *reacting* to letdowns, many network operators such as Internet2 [14] *proactively* check the health of their network using pings between all pairs of sources. Though, Planet Lab [30] organizations can customize SIPC to meet their needs; for specimen, they can choose to merely check for network live ness or check every rule to ensure security policy. SIPC can be modified to check only for reachability or for performance as well. SIPC can adapt to constraints

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 09, September 2015
Available at http://internationaljournalofresearch.org

such as requiring investigation packages from only a few places in the network or using special routers to generate investigation packages from every port. SIPC can also be tuned to allocate more investigation packages to exercise more critical rules. For sample, a healthcare network may dedicate more investigation packages to Firewall rules to ensure HIPPA compliance.

Category	Avg	% of ≥ 4
Reachability Failure	3.67	56.90%
Throughput/Latency	3.39	52.54%
Intermittent Connectivity	3.38	53.45%
Router CPU High Utilization	2.87	31.67%
Congestion	2.65	28.07%
Security Policy Violation	2.33	17.54%
Forwarding Loop	1.89	10.71%
Broadcast/Multicast Storm	1.83	9.62%

(a)

Category	Avg	% of ≥ 4
Switch/Router Software Bug	3.12	40.35%
Hardware Failure	3.07	41.07%
External	3.06	42.37%
Attack	2.67	29.82%
ACL Misconfig.	2.44	20.00%
Software Upgrade	2.35	18.52%
Protocol Misconfiguration	2.29	23.64%
Unknown	2.25	17.65%
Host Network Stack Bug	1.98	16.00%
QoS/TE Misconfig.	1.70	7.41%

(b)

NETWORK MODEL

SIPC uses the *legend space* framework—a symmetrical model of how packages are processed we described in [16] used in [31]). In legend space, protocol-specific meanings related with legends are ignored: A legend is viewed as a flat sequence of ones and zeros. A legend is a point and a flow is a region in the space, where is an upper bound on legend length. By using the legend space framework, we obtain a unified, vendor-independent, and protocol- model of the network2 that simplifies the package creation process significantly.

A. Definitions

Packages: A package is defined by a port and legend tuple, where the port denotes a package's position in the network at any time instantly; each physical port in the network is assigned a unique number.

Switches: A *switch transfer function*, T, models a network device, such as a switch or router. Each network device contains a set of progressing rules that determine how packages are processed. An arriving package is associated with exactly one rule by matching it against each rule in descending order of priority, and is dropped if no rule matches.

Rules: A *rule* creates a list of one or more output packages, corresponding to the output port(s) to which the package is sent, and defines how package fields are modified. The rule perception mockups all real-world rules we know including IP progressing adapts port, checksum, and TTL, but not IP address. VLAN classification adds VLAN IDs to the legend and ACLs block a legend, or map to a queue. Basically, a rule describes how a region of legend space at the ingress the set of packages matching the rule is distorted into regions of legend space at the egress [16].

Rule Antiquity: At any point, each package has a rule *antiquity*: an ordered list of rules the package matched so far as it navigated the network. Rule histories are vital to SIPC, as they provide the basic raw material from which SIPC hypotheses investigations.

Topology: The *topology transfer function*, T, models the network topology by specifying which pairs of ports are connected by links. Links are rules that forward packages from pk1 to pk2 without alteration. If no topology rules match an input port, the port is an edge port, and the package has touched its destination.

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 09, September 2015
Available at http://internationaljournalofresearch.org

Lifecycle of a Package

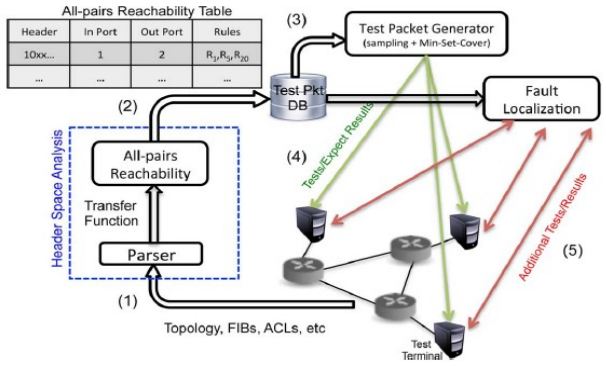
The life of a package can be viewed as applying the switch and topology *transfer functions* repeatedly. When a package pk1 arrives at a network port, the switch function that covers the input port is applied to pk1, creating a list of new packages. If the package reaches its destination, it is recorded. Otherwise, the topology function is used to invoke the switch function containing the new port. The process repeats until packages reach their destinations or dropped.

SIPC SYSTEM

Based on the network model, SIPC creates the nominal number of investigation packages so that every progressing rule in the network is trained and covered by at least one investigation package. When an error is detected, SIPC uses a fault localization algorithm to regulate the failing rules or links. The system first collects all the progressing state from the network. This usually involves reading the FIBs, ACLs, and config files, as well as gaining the topology. SIPC uses Legend Space Analysis [16] to compute reachability between all the investigation terminals. The result is then used by the investigation package selection algorithm to compute a nominal set of investigation packages that can investigation all rules. These packages will be sent occasionally by the investigation terminals. If an error is sensed, the fault localization algorithm is invoked to narrow down the cause of the error.

Bit	$b = 0\|1\|\mathrm{x}$
Header	$h = [b_0, b_1, \ldots, b_L]$
Port	$p = 1\|2\|\ldots\|N\|\mathtt{drop}$
Packet	$pk = (p, h)$
Rule	$r : pk \rightarrow pk$ or $[pk]$
Match	$r.matchset : [pk]$
Transfer Function	$T : pk \rightarrow pk$ or $[pk]$
Topo Function	$\Gamma : (p_{src}, h) \rightarrow (p_{dst}, h)$

function NETWORK($packets, switches, \Gamma$)
  for $pk_0 \in packets$ **do**
    $T \leftarrow$FIND_SWITCH($pk_0.p, switches$)
    for $pk_1 \in T(pk_0)$ **do**
      if $pk_1.p \in EdgePorts$ **then**
        #Reached edge
        RECORD(pk_1)
      else
        #Find next hop
        NETWORK($\Gamma(pk_1), switches, \Gamma$)



Investigation Package creation

Algorithm: We assume a set of *investigation* terminals in the network can send and receive investigation packages. Our goal is to create a set of investigation packages to exercise *every* rule in *every* switch function, so that *any* fault will be observed by at least one investigation package. This is analogous to software investigation suites that try to investigation every possible branch in a program. The wider goal can be limited to investigating every

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 09, September 2015
Available at http://internationaljournalofresearch.org

link or every queue. When creating investigation packages, SIPC must respect two key checks:

1) ***Port:*** SIPC must only use investigation terminals that are available;

2) ***Legend:*** SIPC must only use legends that each investigation terminal is permitted to send. For example, the network admin may only allow using a specific set of VLANs. Officially, we have the following problem.

Problem 1 (Investigation Package Selection): For a network with the switch functions, T , and topology function, t , define the least set of investigation packages to work out all accessible rules, subject to the port and legend constraints. SIPC chooses investigation packages using an algorithm we call *Investigation Package Selection* (TPS). TPS first finds all *equivalent classes* between each pair of available ports. An equivalent class is a set of packages that works on the same combination of rules. It then samples each class to choose investigation packages, and finally compresses the resulting set of investigation packages to find the minimum covering set.

EXECUTION

We employed a prototype system to spontaneously parse router configurations and generate a set of investigation packages for the network. The code is publicly available [1].

A. Investigation Package creator. The investigation package creator, written in Python, contains a Cisco IOS configuration parser and a Juniper Junos parser. The data plane info, including router configurations, FIBs, MAC learning tables, and network topologies, is collected and parsed through the command line interface (Cisco IOS) or XML files (Junos). The creator then uses the Hassel [13] legend space analysis library to construct switch and topology functions. All-pairs reachability is computed using the parallel-processing module shipped with Python. Each process studies a subset of the investigation ports and finds all the accessible ports from each one. After reachability investigations are complete, results are collected, and the master process executes the Min- Set-Cover algorithm. Investigation packages and the set of investigation rules are stored in a SQLite database.

B. Grid Observer

The network grid observer accepts there are special investigation agents in the network grid that are able to send/receive investigation packages. The network grid observer reads the database and constructs investigation packages and coaches each agent to send the right packages. Presently, investigation agents separate investigation packages by IP Proto field and TCP/UDP port number, and IP option can also be used. If some of the investigations fail, the observer selects other investigation packages from reserved packages to pinpoint the problem. The procedure recurrences until the fault have been known. The observer uses JSON to communicate with the investigation agents, and uses SQLite's string matching to lookup investigation packages efficiently.

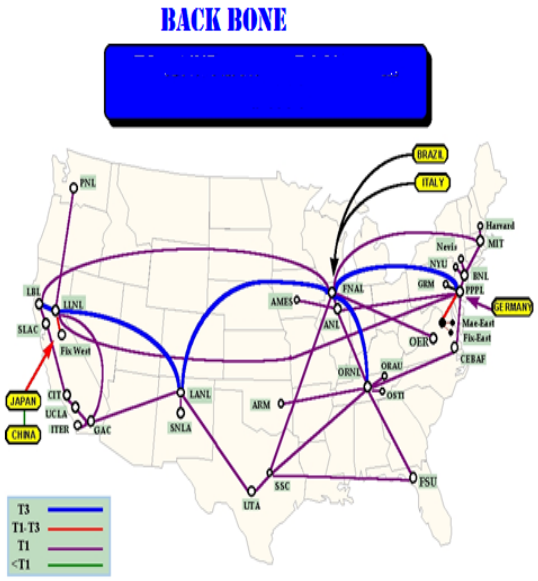
C. Alternate Implementations

Our prototype was designed to be slightly aggressive, requiring no changes to the network except to add termini at the edge. In network grids requiring faster analysis, the following extensions are possible.

Cooperative Routers: A new feature could be added to switches/routers, so that a central SIPC system

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 09, September 2015
Available at http://internationaljournalofresearch.org

can teach a router to send/receive investigation packages. In fact, for manufacturing investigating determinations, it is likely that almost every commercial switch/router can already do this; we just need an open interface to control them.



SDN-Based Investigation: In a software defined network (SDN) such as Open Flow [27], the controller could directly coach the switch to send investigation packages and to detect and forward established investigation packages to the control plane. For performance investigation, investigation packages need to be time-stamped at the routers.

Stanford (298 ports)	10%	40%	70%	100%	Edge (81%)
Total Packets	10,042	104,236	413,158	621,402	438,686
Regular Packets	725	2,613	3,627	3,871	3,319
Packets/Port (Avg)	25.00	18.98	17.43	12.99	18.02
Packets/Port (Max)	206	579	874	907	792
Time to send (Max)	0.165ms	0.463ms	0.699ms	0.726ms	0.634ms
Coverage	22.2%	57.7%	81.4%	100%	78.5%
Computation Time	152.53s	603.02s	2,363.67s	3,524.62s	2,807.01s
Internet2 (345 ports)	10%	40%	70%	100%	Edge (92%)
Total Packets	30,387	485,592	1,407,895	3,037,335	3,036,948
Regular Packets	5,930	17,800	32,352	35,462	35,416
Packets/Port (Avg)	159.0	129.0	134.2	102.8	102.7
Packets/Port (Max)	2,550	3,421	2,445	4,557	3,492
Time to send (Max)	0.204ms	0.274ms	0.196ms	0.365ms	0.279ms
Coverage	16.9%	51.4%	80.3%	100%	100%
Computation Time	129.14s	582.28s	1,197.07s	2,173.79s	1,992.52s

ASSESSMENT:

A. Data Sets: AQ & T and Internet2

We assessed our sample system on two sets of network configurations: the AQ & T backbone and the Internet2 backbone, representing a large size enterprise network and a nationwide backbone network, individually.

***AQ & T Backbone*:** With a population of over 15500 employees, 2500 Consultant, and five/16 IPv4 subnets, AQ & T represents a large enterprise network. There are 17 operational zone (OZ) Cisco routers connected via 10 Ethernet switches to two backbone Cisco routers that in turn connect AQ & T to the outside world. Overall, the network has more than 859000 progressing entries and 1500 ACL rules. Data plane arrangements are collected through command line interfaces. AQ & T has made the entire configuration rule set public [1].

Internet2: Internet2 is a nationwide backbone network with nine Juniper T1600 routers and 100-Gb/s interfaces, supporting over 88000 institutions in US. There are about 125000 IPv4 progressing rules. All Internet2 configurations and FIBs of the core routers are publicly available [14], with the exception of ACL rules, which are removed for security concerns. Though IPv6 and MPLS entries are also accessible, we only use IPv4 rules in this paper.

B. Investigation Package creation

We ran SIPC on a quad-core Intel Core i7 CPU 3.2 GHz and 16 GB memory using eight threads. For a given number of investigation terminals, we made the minimum set of investigation packages needed to investigate all the reachable rules in the AQ & T and Internet2 backbones. For example, the first column tells us that if we attach investigation

terminals to 10% of the ports, then all of the reachable AQ & T rules (22.2% of the total) can be investigated by sending 725 investigation packages.

If every edge port can act as an investigation terminal, 100% of the AQ & T rules can be investigated by sending just 3,871 investigation packages. The Period row indicates how long it took SIPC to run; the worst case took about an hour, the majority of which was devoted to calculating all-pairs reachability.

To put these results into viewpoint, each investigation for the AQ & T backbone requires sending about 997 packages per port in the worst case. If these packages were sent over a single 1-Gb/s link, the entire network could be investigated in less than 1 ms, supposing each investigation package is 100 B and not as the propagation delay. Put another way, investigating the entire set of progressing rules 10 times every second would use less than 1% of the link bandwidth. Similarly, all the progressing rules in Internet2 can be investigated using 4677 investigation packages per port in the worst case. Even if the investigation packages were sent over 10-Gb/s links, all the progressing rules could be investigated in less than 0.5 ms, or 10 times every second using less than 1% of the link bandwidth.

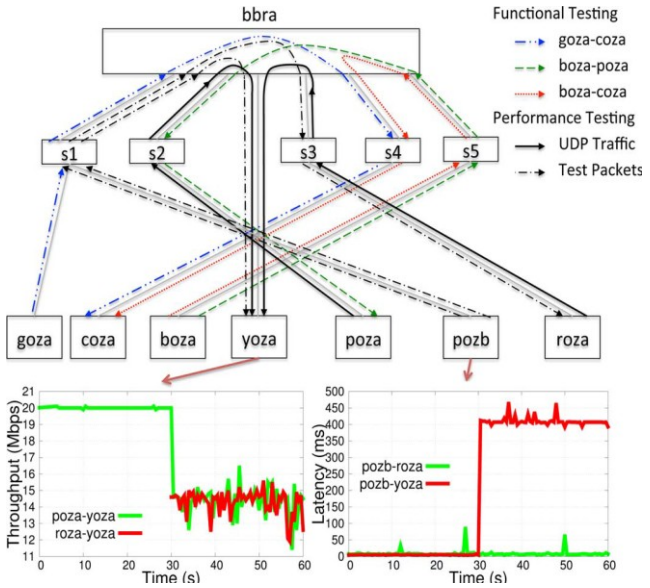
We also found that 100% link coverage (instead of rule coverage) only needed 54 packages for AQ & T and 20 for Internet2.

Rule Structure: The reason we need so few investigation packages is because of the structure of the rules and the routing policy. Most rules are part of an end-to-end route, and so multiple routers cover the same rule. Also, multiple devices contain the same ACL or QoS configuration as they are part of a network-wide policy. Thus, the number of discrete regions of legend space grows linearly, not exponentially, with the diameter of the network. We can validate this structure by clustering rules in AQ & T and Internet2 that match the same legend patterns. In both networks, 60%–70% of matching patterns appear in more than one router. We also discovered that this recurrence is correlated to the network topology. In the AQ & T backbone, which has a two-level hierarchy, similar patterns usually appear in two (52.4%) or four (19.3%) routers, which signifies the length of edge-to-Internet and edge-to-edge routes. In Internet2, 77.1%of all distinct rules are replicated nine times, which is the number of routers in the topology.

C. Investigating in an Emulated Network

To evaluate the network grid observer and investigation agents, we replicated the AQ & T backbone network in [20], a container-based network emulator. We used Open v Switch (OVS) [29] to emulate the routers, using the real port configuration info, and connected them according to the real topology. We then interpreted the progressing entries in the AQ & T backbone network grid into equivalent Open Flow [27] rules and connected them in the OVS switches with Beacon [4]. We used matched hosts to send and receive investigation packages created by SIPC. The part of network that is used for tests in this section. We now present diverse investigation scenarios and the corresponding results.

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 09, September 2015
Available at http://internationaljournalofresearch.org



Limitations: As with all investigating policies, SIPC has limitations.

1) Vibrant boxes: SIPC cannot model boxes whose internal state can be changed by investigation packages. For example, an NAT that vigorously assigns TCP ports to outgoing packages can complicate the online observer as the same investigation package can give different results.

2) Nondeterministic boxes: Boxes can load balance packages based on a hash function of package fields, usually joint with a random seed; this is common in multipath routing such as ECMP. When the hash algorithm and parameters are unknown, SIPC cannot properly model such rules. But, if there are known package patterns that can repeat through all possible outputs, SIPC can create packages to traverse every output.

***3) Invisible rules*:** A failed rule can make a backup rule active, and as a result, no changes may be observed by the investigation packages. This can happen when, despite a failure, an investigation package is routed to the predictable destination by other rules. In addition, an error in a backup rule cannot be noticed in normal operation. Another example is when two drop rules appear in a row: The failure of one rule is untraceable since the effect will be disguised by the other rule.

4) Momentary network states: SIPC cannot uncover errors whose lifetime is shorter than the time between each round of investigations. For example, cramming may disappear before an available bandwidth probing investigation concludes. Finer-grained investigation agents are needed to capture abnormalities of short duration.

***5) Specimen*:** SIPC uses specimen when creating investigation packages. As a result, SIPC can miss match faults since the error is not uniform across all matching legends. In the worst case (when only one legend is in error), exhaustive investigation is required.

CONCLUSION

Investigating live ness of a network is a important problem for ISPs and large data center operators. Sending probes between every pair of edge ports is neither comprehensive nor scalable [30]. It serves to find a minimal set of end-to-end packages that traverse each link. Though, doing this requires a way of abstracting across device specific configuration files like legend space, creating legends and the links they reach such as all-pairs reachability, and finally defining a minimum set of investigation packages (Min-Set-Cover). The fundamental problem of spontaneously creating investigation packages for efficient live ness investigation requires techniques akin to SIPC.

SIPC, though, goes much further than live ness investigation with the same framework. SIPC can investigate for reachability policy by investigating all rules including drop rules and

performance health by associating performance measures such as latency and loss with investigation packages. Our application also augments investigations with a simple fault localization scheme also constructed using the legend space framework. As in software investigation, the formal model helps exploit investigation reporting while reducing investigation packages. Our results show that all progressing rules in AQ & T backbone or Internet2 can be trained by a astonishingly small number of investigation packages.

REFERENCES

[1] "SIPC code repository," [Online]. Available: http://eastzone.github.com/SIPC

[2] "Automatic Investigation Pattern Generation," 2013 [Online]. Available: http://en.wikipedia.org/wiki/Automatic_investigation_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] "Beacon," [Online]. Available: http://www.beaconcontroller.net/

[5] Y. Bejerano and R. Rastogi, "Robust observering 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 investigations 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 investigation OpenFlow applications," in Proc. NSDI, 2012, pp. 10–10.

[8] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, "Netdiagnoser: Repairing 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 linkloss 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 Legend Space Library," [Online]. Available: https://bitbucket. org/peymank/hassel-public/

[14] Internet2, Ann Arbor, MI, USA, "The Internet2 observatory data collections," [Online]. Available: http://www.internet2.edu/observatory/archive/data-collections.html

[15] 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.

[16] P. Kazemian, G. Varghese, and N. McKeown, "Legend 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 investigationing," 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.