REAL TIME AND DYNAMIC VERIFICATION OF DATA PLANE IN SOFTWARE DEFINED NETWORKS

[1]SUCHITA NAVPUTE, [2]MININATH NIGHOT, [3]SANJEEV WAGH

[1,2,3]K. J. College of Engineering and Management Research, Pune
E-mail: [1]suchita.navpute@gmail.com, [2]imaheshnighot@gmail.com, [3]sjwagh1@yahoo.co.in

Abstract— Today's networks are very large in size as well as complex by design. Validating today's networks against any problem is really cumbersome task with traditional approaches like ping, traceroute, tcpdump, etc. Networks are generally susceptible to problems like loop, black-hole, drop, software bugs and physical failures. Protecting networks from these kind of problems using traditional approaches is complicated and time consuming which is not at all affordable in any enterprise scale networks.
To avoid loss caused due to network problems, there is need of tools which can validate network against above said problems in less time as soon as they are introduced in network. In this paper we propose a system which is capable of detecting link failure problems in software defined network (SDN) dynamically.

Index Terms— Network verification, dynamic validation, static validation, software defined network.

I. INTRODUCTION

In the beginning, the work of networking devices was quite simple. Work of networking devices was just to see the forwarding table entries and on the basis of destination address decide where to send packet next. Number of hosts previously were also less so it was not so hard to handle network. On the other hand today's situation of network has changed. Network grew in all dimensions like size, complexity, number of hosts, size of data which we want to transfer on network and many more.
Hosts grew exponentially which leads in crossing the address limit of IPv4. To overcome this problem middle boxes like NAT, firewall came in picture. Consequence of which is routing become little bit complex. Solution on this is new mechanisms invented like VLAN, MPLS which help to make routing flexible. Detecting errors like reachability problem, loop and drop in such large and complicated network with old tools is not efficient as well as it is time consuming process.
There are some typical problems which generally occur in networks like causing packet to loop indefinitely, packet drop before it is reached to destination, fall into black-hole, error occurred due to faulty line card, packet drop due to buffer overflow, etc. As we have no choice for large and complex network then it is must to face the problems and finding solution. For voluminous network errors are unavoidable.
There are mainly two types of network validation
1) Static Validation: It validates network using networks forwarding states snapshot representation. It assumes forwarding state snapshot is consistent with underneath network. We will see more details about static validation in section III.
2) Dynamic Validation: It validates network using actual underneath network because of which it could detect problems caused due to physical failure and software bug. We will see more details about dynamic validation in section III.
Trying new analyzing mechanism or protocol on network is quite hard and take more time to settle down the network because it need to manage two things at a time that are control functionality and data transfer functionality of network devices.
Software Define Network is start of new era of computer network industry. It separates control plane and data plane so controlling the network devices become easy. Due to SDN, network devices like switch, router have become simple data transfer entities and these entities are controlled by SDN controller centrally.
Some common network problems are discussed in section II whereas section III represents approaches for validating network.

II. NETWORK PROBLEMS

Network problems arises mainly due to two causes, first is forwarding state inconsistencies and second is hardware failure. As there is no any central entity controlling and configuring network devices, network administrator has to configure each network entity separately using its own communication mechanism. Forwarding state of network is a cumulative, cooperative entity among all network data forwarding entities. Configuring thousands of routers and switches at their individual level could possibly cause conflicting entries across different forwarding entities which is main cause of forwarding state inconsistency. Hardware failures mainly include network card failures and network link failures.
Error in forwarding table causes packet to loop indefinitely, packet drop before it reach to destination, packet fall in to black-hole. Identifying these type of errors in complex network is hard because in such network, administrator has to deal with thousands of switches and millions rule.

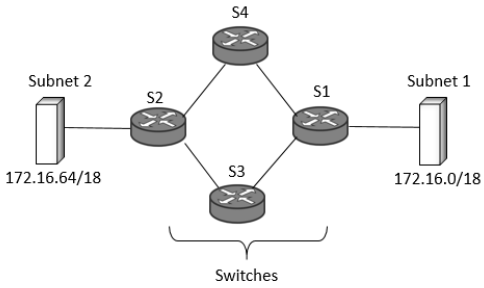


Figure 1: Example Network.

Consider small network example shown in fig1. This network has 4 switches and 2 subnets. Our example network uses multipath routing. The node having more than one outgoing edges represent multipath routing in the network. It has 8 rules as shown in fig2. Packet arriving at S1 from subnet 172.16.0/18 has two paths S3 and S4 to reach to the destination address 172.16.64/18.

Rules:

S1: 172.16.64/18 → S3,
S4 172.16.0/18 → direct
172.16.0/18 → S3,
S4 172.16.64/18 → direct
S3: 172.16.64/18 → S2
172.16.0/18 → S1
S4: 172.16.64/18 → S2
172.16.0/18 →S1

Graphical representation of example network is shown fig 2. Network behavior of subnet can be best represented using directed forwarding graph.

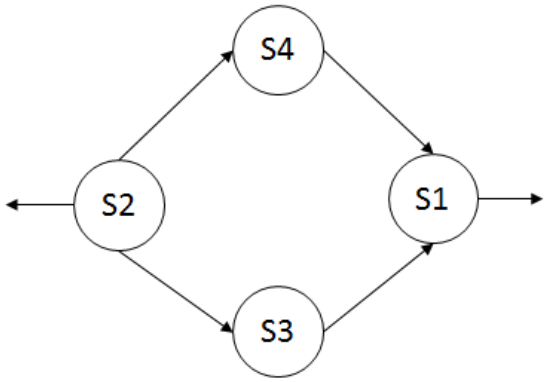


Figure 2: Normal Network

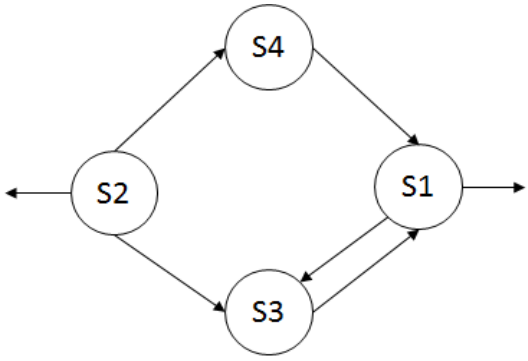


Figure 3: Loop in Network

In Fig. 2 arrow from S2 to S4 means packets from 172.16.64/18 be forwarded from S2 to S4.

Loop: Fig. 3 shows how an error in forwarding table can lead to loop in the network.

172.16.64/18 → S3,
S4 172.16.0/18 → S3

Consider above forwarding entries at switch S1. From the forwarding state at S1 it is clear that switch S1 will forward packet destined to 172.16.0/18 to switch S3 causing loop as shown in fig4. Network has one loop S1S3-S1 and packets for 172.16.0/18 will never reach to their intended destination. It will keep infinitely looping between S1 and S3.

Black-hole: Fig. 4 shows black-hole situation in network. Consider that entry 172.16.0/18 -¿ direct is dropped. In this scenario packet destined to subnet 172.16.0/18 will be dropped at S1 as it has no forwarding entry to decide its outgoing path.

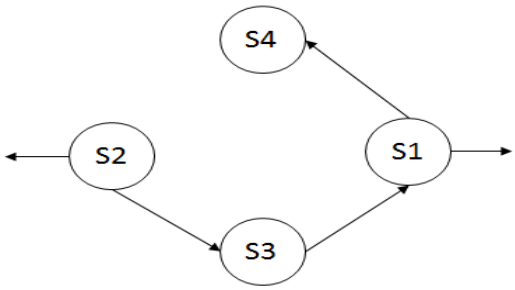


Figure 4: Black Hole in Network

Detecting hardware failure in large network is hard because network devices are typical black-boxes i.e. switches and routers. To find location of the failure administrator has to login into each device manually and check all configuration of the device. Detecting simple line-card failure in large network take time because administrator has to decide manually which ping packet to send and after observing forwarding states they find clue of failure.

Drop due to physical failure: Consider that physical link from S1 to S4 is failed due to from physical breakage of network line is shown in Fig. 5. Packet destined to 172.16.0/18 from S2 will be spread across S3 and S4. Packets following route from S2 to S4 will be dropped due to physical failure of line.

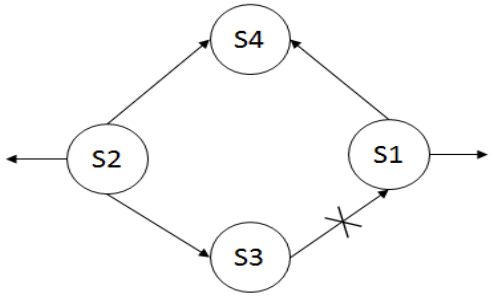


Figure 5: Physical Failure in Network

III. APPROCHES

Network error should be caught before too much traffic is lost and security is in danger. To analyze network many ideas came forward, among them only few stand in performance test. Though they succeed in analyzing network as well as finding errors, some of them are failed in test of time consumption, some are failed in performance. Need varies as per type of network and also as per user. But most commonly, expectations from analyzing tool are that it should analyze network correctly and it should catch error in less time as it possible. To satisfy these requirement many tools are implemented but sadly some have succeeded in analyzing network correctly but simultaneously time consuming. Some have shown result in very less time but are not guaranteed about correctness.

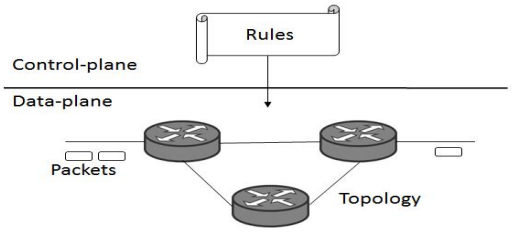


Figure 4: Simplified View of Network

Fig. 6 shows general network scenario it shows data plane and control plane. Forwarding state required by data plane is written by control plane. Control plane can be local in case of traditional or non SDN network whereas control plane will be remote in case of SDN [1]. Control plane should correctly implement network policies into data plane.
Controller in control plane compiles policy into device specific configuration file. Forwarding behavior of network is decided by configuration files. To work network properly, all three aspects i.e. policy, configuration files, forwarding behavior should be in consistent state with each other. In addition to this sufficient physical links and nodes should be in working state.
Basically there are two types of validation.

1) Static Validation

It validates network using networks forwarding states snapshot representation. Static validation tool calculates representational model (snapshot) e.g. HSA [1] from forwarding state of network. As these snapshots are static representation of network forwarding state, tools based on analysis of such static snapshot are referred as static validation tools. Static validation tools are capable of confirming consistency between network policy and configuration files. Most of the static validation tools work offline on built snapshot, they may miss any consistencies caused in underline network after calculation of snapshot. Some of the examples of static validation tools are HSA [1], Anteater [4], and Netplumber [5].

As static validation tools validates network based on representational model, they are not capable of detecting problem caused due to physical failure or software bugs. And also not able to confirm consistency between forwarding configuration and actual forwarding behavior in data plane.

2) Dynamic Validation

It validates network using actual underneath network because of which it could detect problems cause due to physical failure and software bug. Dynamic validation may use snapshot approach to help tuning and reducing overhead while validating network. As actual network is test bed in dynamic validation, keeping validation overhead small is an important factor in dynamic validation otherwise validation itself could hamper performance of network. Example of dynamic validation include ATPG [2].

IV. RELATED WORK

To catch routing error before too much data loss and security breached there is of need fast and scalable tool. Many tools have been proposed to fulfill networks need like Netplumber [5], Header Space Analysis (HSA) [1], Anteater [4], Veriflow [3], Automatic Test Packet Generation (ATPG) [2]. These all tools use different model for analyzing the network. Most of them do static validation. We will see one by one these tool and their approaches.
HSA validates network statically. To represent network, HSA uses geometric model i.e. it consider a packet as a point in the 0,1L space where L is the length of the header. And packet will be transferred by the network devices from one point to other point in the space. HSA detect some class of failure like reachability failure, loop in the network, detecting slice isolation and detect leakage. HSA is protocol independent model. As HSA is static checker it is not able to recognize problem in running network or it will not be possible to catch the failure before it happened. It is not possible to HSA to predict problem by looking the forwarding table inconsistency.
Netplumber [5] is based on HSA but it is real time checker. Netplumber take leverage of Software Define Network (SDN). Netplumber seats in-line with controller. Netplumber uses graphical model to represent the network. Real time checking of updates and flexible way to add new complex policy queries without writing new ad hoc code for each policy check are advantages over HSA.
Netplumber take more processing time to verify link updates so it is not suitable for network which has high rate of link up and down.
Anteater also follow static validation approach but difference is that it analyses data plane. As anteater statically checks data plane it is possible to catch bugs which are in visible at the level of configuration

files. Anteater translate high level network invariants into instances of boolean satisfiability problems, check these problems against network state using SAT solver and report if violations have been found.
Anteater is unable to find bugs in network devices such as router which interprets and act on configuration files. Second it can only focus on validating single protocol like BGP or firewall.
VeriFlow is a layer between SDN controller and the data plane. It is static checker but it perform real-time network validation in the context of SDN. VeriFlow gets each update from SDN controller before it is forwarded to the data plane. VeriFlow uses trie data structure for its internal representation. The main task of VeriFolw is tracking each forwarding state change event as is passes by SDN, apply that new update on its internal model, check if any violation. VeriFlow is faster than earlier tools because it needs to check only that subpart of a trie which gets affected due to update. It can prevent system before any failure.
Limitation of VeriFlow is that it is static checker so it is unable to prevent network from hardware failure e.g. physical link down.
ATPG is one of the example of dynamic validation. ATPG collects all information about network like forwarding state, configuration file, topology etc. so that it can create its internal representational model. With the help of header space analysis, it checks all reachability between test terminals. ATPG creates minimal set of test packet. These set of packets are sent to the test terminals and checked for failure. If any error detected, the fault localization is done.
Detecting physical failure is one of its big advantage.

V. PROPOSED WORK

This tool uses an automated and systematic approach to test the network. In this mechanism there are two main functionalities; first, it checks each new update on internal model and then forward to data plane. Second for physical validation purpose it SDN network status message packets.
Controller: Software Define Network provides new mechanism for the network. It separates the control part and the data forwarding part of the network. We can also say the separation of the configuration mechanism and the actual data forwarding elements. So SDN has two main parts which are, control plane and data plane.

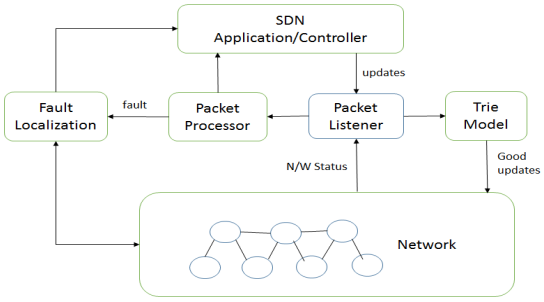


Figure 7: Project Design

Switches in the network works as basic packet forwarding devices. They contain flow tables containing forwarding rules. Switches are managed by a remote controller entity, which securely communicates with switches and routers.
The agent placed in between the controller and the data plane in our design model plays an important role. Each configuration as set from the controller, an agent first read and apply it on internal model. If it finds any bad update it can generate the alarm. So that network get saved from bad update before it harms the system.

Internal Model: We create our internal model by using trie structure. A trie is a data structure that is a tree of nodes, where each node is an array of M elements. Trie is very flexible to use. We can form trie as per our requirements. There are some basic types of trie unibit trie, multibit trie, fixed stride trie and variable stride trie. In our project design case, the network gets represented in the form of trie. Each configuration first gets checked in to trie model and then decide to forward to data plane or not.
It is flexible in a sense that for each new configuration, there is no need to check whole internal model. The sub-part of a trie which will get affected by new update that only need to check. Due to this it take very less time to decide, the incoming update is good or bad for the network.

Packet Processor: The test packet processor is an important functionality of our tool. It processes packets captured by packet listener. Packet processor detects kind of problem by processing packet. If packet does not belong to network status, packet is rerouted to SDN controller as it is.
Network status packet representing link failure is sent to fault localization module to find out exact location of failure in the network. Fault localization module finds out exact edge in the network by using network structure file and network status fields. Fault localization module informs network representation module about failure location along with required information.
Network representation module represents to administrator about exact status of network. Network is represented in the form of graph. If everything is all well, network is shown in the form of graph with all solid edges. If network representation module get request to represent any failed link, it updates network graph and shows failed link with the help of dotted edge with red color. Alongside this representation, it also generates network data file which has network representation in the form of text which programmer can user to represent network as per administrator's requirement.

VI. IMPLEMENTATION

Implementation has two main parts. One is module to intercept SDN events and analyze change for good or

bad. Second is testing network by listening and processing network status packets.

We are extending implementation of Veriow [3] to enable it to test network by processing network status packets. Implementation is built with VeriFlow with POX. POX API facilitated to implement independent rule verifier sitting in between openflow application and switches and routers of network.

There are five types of flow table modification messages generated by openflow application. OFPFC ADD (add new flow), OFPFC MODIFY STRICT (modify entry strictly matching wildcards and priority), OFPFC DELETE STRICT (delete entry strictly matching wildcard and priority), OFPFC MODIFY (modify all matching flows) and OFPFC DELETE (delete all matching flows).

Multidimensional prefix tree (trie) is used to represent all equivalence classes. A trie is an ordered tree data structure. Trie associates the set of packets matched by a forwarding rule with the forwarding rule itself. Each trie node has three branches, corresponding to three possible values that the rule can match: 0, 1, and * (wildcard). The trie can be seen as a composition of several sub-tries or dimensions, each corresponding to a packet header field. For each EC, we generate a forwarding graph. A forwarding graph represents, how packets within an EC will be forwarded through the network. A node in the graph represents an EC at a particular network device and a directed edge represents forwarding decision for a particular (EC, device) pair. To build a forwarding graph we traverse our trie to find out devices and rules that match packets from that EC.

Packets listener module listens for network status message from SDN switches. Packet processor module process packets and find out whether there is any problem in network. Administrator user interface display real time network status to administrator. In case of any problem in network, administrator is instantly notified about problem with its location through network representation diagram.

Algorithm:

1) Packet listener module listens for packets from SDN switch to SDN controller.

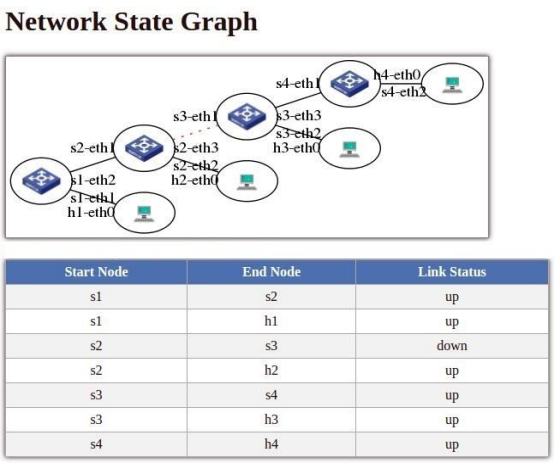
2) Out of listened packets, network status packets are identified and sent to packet processor module. Other packets are forwarded to SDN Controller.

3) Packet processor module processes network status packet and identifies if there is any problem.

4) If problem is found, required packet fields are sent to fault localization module.

5) Fault localization module finds exact location of fault in the network by comparing various fields sent by packet processor module against network representation.

6) Once the fault location is localized, location is sent to network representation module, which then updates network UI representation and updates screen shown to the administrator.



Start Node	End Node	Link Status
s1	s2	up
s1	h1	up
s2	s3	down
s2	h2	up
s3	s4	up
s3	h3	up
s4	h4	up

Figure 8: Output Screen

7) By looking at UI, administrator easily gets visual of problem in the network can reach to location to take necessary steps to correct problem.

CONCLUSION

We presented testing tool which able to check network dynamically. Tool is placed in between the data-plane and control-plane therefore every new configuration is passed through it. It is able to prevent violating rule before it enter into the network. Also it is able to find out if there is any physical failure occurred in the network.

This tool may not know when inconsistency in network state is due to intermediate state during which the violation is acceptable by the operator. So, deciding whether to check or not will be the future work.

REFERENCES

[1] Peyman Kazemian, George Varghese, and Nick McKeown. 2012. Header space analysis: static checking for networks. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI'12). USENIX Association, Berkeley, CA, USA, 9-9. [2] Hongyi Zeng, Peyman Kazemian, George Varghese, and Nick McKeown. 2012. Automatic test packet generation. In Proceedings of the 8th international conference on Emerging networking experiments and technologies (CoNEXT '12). ACM, New York, NY, USA, 241-252. DOI=http://dx.doi.org/10.1145/2413176.2413205

[2] B. Ramkumar, H.M. Kittur, and P. M. Kannan, "ASIC implementation of modified faster carry save adder," *Eur. J. Sci. Res.*, vol. 42, no. 1, pp.53–58, 2010.

[3] Ahmed Khurshid, Xuan Zou, Wenxuan Zhou, Matthew Caesar, and P. Brighten Godfrey. 2013. VeriFlow: verifying network-wide invariants in real time. In Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation (nsdi'13), Nick Feamster and Jeff Mogul (Eds.). USENIX Association, Berkeley, CA, USA, 15-28.

[4] Haohui Mai, Ahmed Khurshid, Rachit Agarwal, Matthew Caesar, P. Brighten Godfrey, and Samuel Talmadge King. 2011. Debugging the data plane with anteater. In Proceedings of the ACM SIGCOMM 2011 conference (SIGCOMM '11). ACM, New York, NY, USA, 290-301. DOI=http://dx.doi.org/10.1145/2018436.2018470

[5] Peyman Kazemian and Michael Chang and Hongyi Zeng and George Varghese and Nick McKeown and Scott Whyte. 2013. Real Time Network Policy Checking Using Header Space Analysis. Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13).

[6] S.Shenker, The future of networking, and the past of protocols, 2011 [Online].Available:http://opennetsummit.org/archives/oct11/shenkertue.pdf

[7] L.Yuan, J.Mai, Z.Su, H.Chen, C-N.Chuah, and P. Mohapatra, FIREMAN: A Toolkit for Firewall Modeling and Analysis, 2006 IEEE Symposium on Security and Privacy, pp. 213-228

[8] Y. Bartal, A. J. Mayer, K. Nissim, and A. Wool. Firmato: A novel firewall management toolkit, Proc. 20th IEEE Symposium on Security and Privacy, 1999.

[9] A.Mayer, A.Wool, and E. Ziskind, Fang: A firewall analysis engine, Proc. IEEE Symposium on Security and Privacy, 2000.

★★★