Project Proposal

for

Packet Tracer : SDN tool to trace ICMP packet path using 2-Colored graph

Version 1.0

Prepared By:
Nishant Gill (924001574)
Akash (823008707)
Anish Dubey(523002969)

Advanced Networking And Security
Instructor: Dr. Guofei Gu

Department of Computer Science
Texas A&M University
10/07/2014

Table of Contents

Problem Statement/Motivation:

Software-defined networking (SDN) is an approach to networking in which control plane is detached from the hardware and given to a software application called a controller. When a packet arrives at a switch in a conventional network, rules are built into the switch's firmware tell the switch where to forward the packet. These rules are installed by the controller itself . Now, the switch sends every packet going to the same destination along the same path and treats all the packets the exact same way. However , this flexibility brings added complexity, which requires new debugging tools that can provide insights into network behavior. When troubleshooting a problem, SDN programmers and network operators must grapple with many possibilities including bugs in controller logic, switches, individual SDN applications, and their composition. This makes it imperative to have tools that can provide visibility into how different packets are handled by the network at any given time.

One of the many applications mainly used for debugging the path of the packet is traceroute. But it has its own disadvantages, when it is tracing the route , it consumes lot of bandwidth of the network by continuously sending the packet back and forth to the host. But now with the controller , traceroute approach can be optimised by sending its per hop information to the controller instead of sending back to the host as discussed in SDN Traceroute[3]. So this way , bandwidth utilization of the network is reduced and controller will have all the necessary information regarding the trace route.

What we propose will even optimise the above approach and reduce the packet flow communication between switches and the controller. We will develop a tool in which alternate nodes will transfer the information to the controller, instead of every nodes, and thus reducing this overhead by up to 50% as compared to SDN Traceroute[3] proposed.

Proposed Technique / Solution:

Packet Tracer works in two phases. In the first phase, we create a graph to represent the topology of the network and color each switch (or vSwitch) in the network using a graph 2-coloring algorithm, in which each node is colored either black or white, such that no two white nodes are adjacent to each other and number of black nodes in the graph is minimum. Each node stores info about its color , switch egress port and neighbours' switch ids, connected to it. Then, using the results of the coloring, we install a high-priority rule in every switch that is a black color node in the network.

The second phase starts when an ICMP packet is pushed into the network. The installed high priority rules at the black nodes allows the packet to progress either one or two hops (depending on the node color) before being returned to the controller. The controller receives the packet and extracts the current and previous switch details and adds it to the trace path and re-injects the packet into the switch. When the packet reaches the destination, the switch will transfer its last information to the controller and controller will check if the destination host is local to the switch . If it is local , the controller will save its trace route information in a list and this information will be exposed as a Rest Api. The main idea behind this is to save the overhead of sending information to the controller every time the packet passes through switches. In our approach, only black colored switch will send the packet to controller, thus saving the overhead by up to 50% .
In the end, our project will expose a Rest Api which will provide the trace information, to the network administrator, about the latest ICMP packets in the network. The information will include Switch IDs through which the packets passes and the order in which it goes through the switches.

Survey of related work and comparison:

A lot of tools have been proposed to troubleshoot SDNs, including Anteater[5], VeriFlow[6], Libra[7], Pathlet Tracer[4], SDN TraceRoute[3] etc. Our current network tool shares a similar goal to SDN Traceroute as we trace the path for all the ICMP packets in the network and our future work shares some ideas from Pathlet Tracer.
SDN TraceRoute used a special packet for tracing path using vlan bits in the packet to identify the special packet. On the other hand, our work does not use some special packet for the tracing path , instead it does this for all the ICMP packets in the system and provides an API to the network administrator for each packet. Also, Traceroute uses coloring algorithm to identify neighbours using 7 colors, whereas our system uses 2 colors, Black and White, to represent the whole network with both colors having some special meaning. Our tool keeps the whole information about the color of nodes in the controller only and does not pass this information using any vlan bits in the data packet like Traceroute does.
Pathlet Tracer is another tool to trace only certain predefined paths in the network. Our future work will try to incorporate some ideas from Pathlet Tracer. Like, performing actions on the certain bits in the packet based on the rule matching. The bits used to do such actions can be ToS field or IP ID field as suggested by the paper. But, unlike only setting action used by the Pathlet our work will perform an operation on the bits which will be achieved by modifying the action behaviour of the OpenFlow enabled switches, Open vSwitch[2]. Also, unlike Pathlet which uses the value of Tos field to match to precalculated path in the network, our system will use the IP ID field to store the information about all the switches through which the packet passes. Our tool for this future work will use 7 color system like used in the SDN Traceroute instead of 2 colors, and each switch will perform operation on the IP ID bits to embed its color in the data packet.

Project Plan:

Listed below are the project tasks with their timelines. First six tasks will be equally divided between 3 team members based on complexity. Project report will be a combined effort based on respective tasks.

Seq No.	**Task**	**Description**	**Timeline**
1	Topology Discovery	Creating graph to represent the whole topology of th network with each node storing info about its color an neighbours connected to it.	11/13
2	Graph Coloring	2-color (Black, White) nodes coloring, edge nodes o network as black, avoiding adjacent white nodes	11/13
3	Color Based Rule Installation	Installing high priority rules on all black colored nodes forward all ICMP packets to the controller for path tracing.	11/13
4	ICMP - Actual Path Tracing	Tracing path at the controller using ICMP packet information.	11/13
5	Exposing Rest API for Network Admin	Providing the administrator a facility to trace the path o certain number of new ICMP packets in the network, with information about switches it travelled through	11/13
6	Evaluation and Testing	Testing the path tracing on various network topologie using Mininet[9] or other available tools and comparin results against the SDN Traceroute tool [3]	11/30
7	Project Report	Final project report with experimental results.	11/30

References:

[1] OpenFlow Switch Specification. http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf
[2] Open vSwitch. http://openvswitch.org/
[3] K. Agarwal, E. Rozner, C. Dixon and J. Carter. SDN traceroute: Tracing SDN Forwarding without Changing Network Behaviour. In HotSDN'14
[4] H. Zhang, C. Lumezanu, J. Rhee, N. Arora, Q. Xu and G. Jiang. Enabling Layer 2 Pathlet Tracing through Context Encoding in Software-Defined Networking. In HostSDN'14
[5] Mai, H., Khurshid, A., Agarwal, R., Caesar, M., Godfrey, P. B., and King, S. T. Debugging the data plane with anteater. In Proceedings of the ACM SIGCOMM 2011 Conference (2011)
[6] Khurshid, A., Zou, X., Zhou, W., Caesar, M., and Godfrey, P. B. Veriflow: Verifying network-wide invariants in real time. In Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (2013), USENIX.
[7] Zeng, H., Zhang, S., Ye, F., Jeyakumar, V., Ju, M., Liu, J., McKeown, N., and Vahdat, A. Libra: Divide and conquer to verify forwarding tables in huge networks. In Proceedings of NSDI'14 (2014), USENIX.
[8]Floodlight Controller http://www.openflowhub.org/display/floodlightcontroller/Floodlight+Documentation
[9] Miniet: Virtual Network Emulator. http://mininet.org/