Research of Topology Discovery technology in Software-Defined-Network

Lei Yan, 10405833

NIS-800, 2016 fall

***Abstract* - Software Defined Networking (SDN) is physical separating control and data planes network devices, which leads to an adaptable and manageable network. To effectively manage SDN, topology discovery is an essential application. At the security aspect, attackers can easily poison the network to steal information or destroy it. Thus, security solutions are needed to prevent attacks. In this paper, I also mention several future challenges of Topology discovery mechanism in SDN.**

Keywords: SDN; Topology Discovery; Security problems and solutions

I. INTRODUCTION

SDN, standing for Software Defined Network, decouples network control and forwarding functions, enabling network control to become directly programmable and the underlying infrastructure to be abstracted from applications and network services. In this scheme, the controller is the brain of a SDN network, users can design their own applications to control the network and collect all the information about this network. Among these applications, topology discovery is essential. If we have the complete and accurate topology information of a network, we can perform a lot of network management tasks including monitoring, troubleshooting and resource allocation.

In this paper, I demonstrate what I did on this topic. The Section II is about the topology discovery technologies in physical and network layer of traditional networks. The Section III is about the topology discovery technologies in SDN. In Section IV , I state the security problems and possible solutions of topology discovery in SDN. In the Section V, I state a simple experiment I have done. In the Section VI, I describe several future challenges of topology discovery in SDN. Finally, I make a conclusion for this semester's research of topology discovery in SDN.

II. TOPOLOGY DISCOVERY IN TRADITIONAL NETWORKS

Traditional IP network is comprised by autonomous systems (ASs), a border router separates Ass. In my research, I mainly focus on the topology discovery within ASs. I separate this discovery task to two parts: layer 2 discovery and layer 3 discovery. Layer 2 discovery means discovering switches and links in subnets; Layer 3 discovery means discovering topology of router level.

A. Layer 2 Discovery

In general, layer 2 discovery aims to discover topology of subnets divided by routers. In [1], the authors introduce the discovery method in subnets. First, we discover network elements in the network, i.e. routers, switches, hubs. Second, we discover edges/links in the network.

To do that, we use the information in Address Forwarding Tables of devices in the network. We denote the *j*th interface of a switch S*i* by S*ij*. For each interface S*ij*, the set of addresses that have been learned on that interface is referred to as the AFT corresponding to S*ij* and is denoted by A*ij*. Therefore, A*ij* is the set of MAC addresses that have been considered as source addresses on packet frames received at S*ij*. Furthermore, in topology discovery, we mainly use two lemmas.

Lemma 1. Interfaces Sij and Skl are directly connected to each other if and only if $Aij \cup Akl = U$ *and* $Aij \cap Akl = \emptyset$.

Lemma 2. A leaf interface of a switch Si is an interface that is not connected to an interface of any other switch

To distinguish different elements in the network, we use the value of variables in switch MIB (*ipForwarding* and *system.sysServices*) to determine if a node is a switch or router. Particularly, we may find hubs when we are looking for leaf interfaces. To be more specific, we will start with finding an address forwarding table A*ij* that has the smallest number of elements. If this address forwarding table has only one element, S*ij* is a leaf interface; If this table has more than one elements, then we add one hub connected to S*ij*.

To discover edges/links of networks, we traverse every switch port in this network, then use these two lemmas to judge every possible link.

After these work, we generate a graph representing the topology of this network.

B. Layer 3 Discovery

In layer 3 discovery, we are going to generate a router level topology of the network. In [2], authors state 5 main challenges of this task. (1) Reduce the number of probes. (2) Ensure the accuracy. (3) Discover anonymous routers. (4) Alias resolution. (5) Identify subnets correctly.

In [3], the authors propose a hybrid approach to da TD for network layer. The authors combine active and passive discover methods together. Passive methods rely on the use of SNMP and DNS. They are fast and reliable but not always usable. Active methods rely on the use of tools such 'ping' and 'traceroute'. They are neither reliable nor fast, but they are widely usable. Therefore, this paper combines them together as Hybrid network topology discovery (HyNeTD) method.

Following is the architecture of HyNeTD:

1) Identify all the active addresses. HyNeTD sends a ping to all IP addresses in the discovery domain. Then adds the addresses that respond to the ping to an active addresses list.

2) Identify SNMP information resources. To narrow down the range of addresses to be tested, HyNeTD sends one UDP packet to active addresses, then removes addresses responding with ICMP "*Host Unreachable-Port*". Then, HyNeTD sends an SNMP request to the remaining addresses and inserts responding addresses to the SNMP list.

3) Perform passive discovery process. For each address belonging to the SNMP list, HyNeTD sends a SNMP request for just one time to obtain information from SNMP MIB. After this operation, HyNeTD uses DNS to obtain the addresses' names.

4) Perform active discovery process. HyNeTD uses traceroute with ICMP Echo-Request message and ping with record route to do active discovery.

5) Reconstruct the topology of the discovery domain. Reconstruct links and subnets in discovery domain and solve the alias problem.

III. TOPOLOGY DISCOVERY IN SDN

In this section, I describe the theoretical approach to discover the topology of SDN. At first, I introduce several basic concepts of SDN, i.e. Open Flow protocol, Link Layer Discovery Protocol (LLDP) and Open Flow Discovery Protocol (OFDP).

A. Open Flow protocol

Open Flow [4] is a protocol that provides SDN southbound interface, which connects control layer and data forwarding layer. Open Flow is produced by Open Network Foundation (ONF), it is currently the dominant standard south interface protocol.

An Open Flow switch will be configured with the IP address and the TCP port number of its controller. Thus, when a switch wants to connect to the network, it will contact its controller on the corresponding IP address and port number, and establish a Transport Layer Security (TLS) link with the controller.

With Open Flow, controller can access and manipulate the forwarding rules and flow tables of SDN switches. Switches may be allowed to rewrite

the packet headers, but they cannot rewrite the content of packets.

A switch can also forward data packets that it has received to the connected controller by an Open Flow Packet-In message. In the other direction, the controller send Packet-Out message to switches to instruct them to do something.

B. Link Layer Discovery Protocol

Link Layer Discovery Protocol (LLDP) [5] is implemented in LAN to tell other nodes their capability and their neighbors. This protocol is used by Ethernet switches to communicate with their neighbors. And LLDP packets are sent only across a single hop.

As shown in *Figure.1*, an LLDP payload is encapsulated in an Ethernet frame with the EtherType set to 0x88cc. The shading area in the following figure is LLDPDU, which contains the key information of the LLDP packets. The Classic ID (switch identity) and Port ID TLV are important in Topology Discovery technology of SDN.

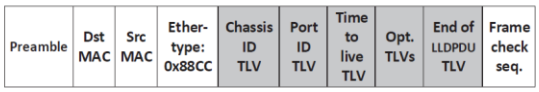

Figure.1

C. Open Flow Discovery Protocol

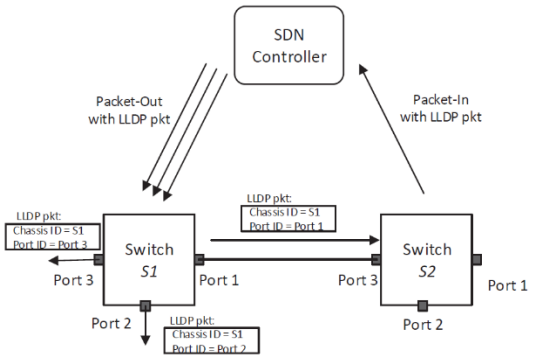

Figure.2

An example of OFDP [5] process is shown in *Figure.2*.

1) The SDN creates an individual LLDP packet for each port on each switch. These three LLDP packets have different Port ID information.

2) The SDN controller sends these packets to switch 1 through three separate Packet-Out messages. Moreover, all switches have a pre-installed rule, LLDP packets received from any port except the controller port are to be forwarded to the controller through Packet-In message.

In the example of *Figure.2*, an LLDP packet sent out from port 1 of switch 1 is received by switch 2 via port 3. Thus, the Packet-In message indicates the connection between port 1 of switch 1 and port 3 of switch 2 will be sent to the controller.

In [6], authors also propose OFDPv2, which reduces the overhead by merging the multiple Packet-Out messages to single Packet-Out message. This Packet-Out message will ask the switch to flood out LLDP packets via its all ports except the port connected to the controller. And OFDPv2 indicates the source port of the LLDP packet by using MAC address of this port instead of port ID TLV in the LLDPDU.

D. Discovery Process in SDN

In section II I introduce the layer 2 topology discovery method, similarly, in SDN, we should also discover elements and links in the network. In this part I will introduce three main tasks [7] of TD in SDN.

1. Host Discovery

With host discovery, the controller can identify the exact location of hosts in the network. Then controller can perform a bunch of tasks such as traffic monitoring, assisting in traffic routes and determining where packets are from. To realize this function, the controller maintains a host profile table for each host in the network. When a host enter the network, it will request the connected switch to encapsulate the host profile table information into a Packet-In message. Then send this message to the controller to generate this host's profile. The host

profile contains IP address, MAC address and Meta information (DPID, port number, and last timestamp). When a host migrate from one switch to another, its port and switch IDs will change due to the new location. These changes will be reported to the controller through the Packet-In message.

2. Switch Discovery

The location of switches is important to the controller, because the controller and switches will communicate frequently. The switches are discovered in the initial handshake process with the controller. Once a switch has entered the network, the controller gets the existence and key information of the switch such as number of ports and MAC address.

3. Internal links discovery

In the part of OFDP, I describe the process of discovering links between Open Flow switches. However, LLDP has the limitation that it only discovers links between adjacent switches. This drawback makes LLDP packet misses when there are links with traditional switches, i.e. switches do not support Open Flow protocol. One or more traditional switches may exist between two Open Flow switches. This situation causes indirect links between Open Flow switches.

To discover indirect links, BDDP (Broadcast Domain Discovery Protocol) is proposed. The BDDP messages present the same structure of LLDP packets. The key difference between these two protocols is that the destination MAC address field will be replaced by the broadcast address (*ff:ff:ff:ff:ff:ff*). Moreover, the EtherType area is changed to 0x8999 for BDDP.

The discovery process is similar with the method using LLDP. I only replace LLDP by BDDP. After the Open Flow switch processing BDDP packets, it forwards the message to neighbors. Neighbors may be Open Flow switches or traditional switches. On the one hand, when one neighbor is Open Flow switch, we will use the same approach as OFDP. On the other hand, when on neighbor is traditional switch, it will flood the packet by all it ports. Suppose that at least one of its neighbors supports Open Flow, it will send the Packet-In message to the controller. Then an indirect link is discovered.

IV. Security Problems and Solutions

SDN is an innovative technology, while gives us a more scalable and effective network structure, it also brings us some security threats. In this part, I introduce topology poisoning attacks on SDN [7], i.e. Host Location Hijacking Attack and Link Fabrication Attack.

A. Host Location Hijacking Attack

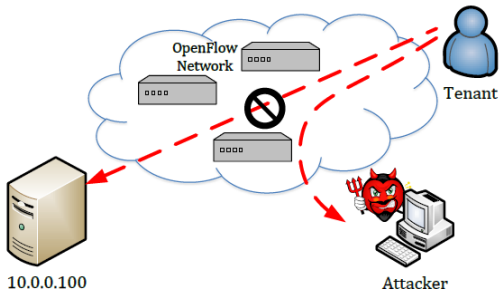

Figure.3

As described earlier that the host profile table in the controller contains key information of hosts, i.e. IP address, MAC address and Meta information (DPID, port number, and last timestamp). They indicate to the controller the location of the host and the connected Open Flow switch. Furthermore, when a host roam in the network, it may update its host profile. However, because SDN lacks strong authentication mechanism, the integrity of information updates cannot be guaranteed. An attacker can use the same authentication key as his/her target to hijack all the packets supposed to be sent to the target host.

B. Link Fabrication Attack

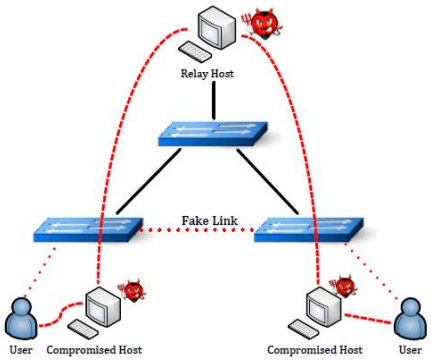

Figure.4

In this part, I describe how an attacker make the link fabrication attack. Because fake links are injected to the network, the topology will be changed accordingly. Then some normally using ports may be killed, because Spanning Tree Protocol is implemented to remove redundant links. Furthermore, the shortest paths calculation will be disturbed. Thus, the link fabrication attack is a disaster for our network.

Attackers can launch link fabrication attacks using two different methods, fake LLDP packets injection and genuine LLDP packets relay (*Figure.4*).

In current SDN system, the integrity of LLDP packets cannot be guaranteed, i.e. switches cannot authenticate the origin of LLDP packets. Thus, the adversary can generate fake LLDP packets to announce non-exist links between two switches, obtain genuine LLDP packets from the fake link.

On the other hand, the adversary can only repeat the LLDP packets to another switch without any modification. Using this method, the adversary can make a link invisible to the controller and create fake links.

C. Current solutions

Until now, there are two kinds of solutions for security problems in SDN, authentication and verification. First, for host hijack attacks, we can authenticate a host by adding additional public-key infrastructure. Particularly, when a host needs to change its location, it encodes the new location information into an unused field of packet with the encryption using its private key. We can also verify the legitimacy of migration by checking pre- and post-condition of a migration activity. Before the migration, the controller should receive a *PORT_DOWN* message from the host. After the migration, host will become unreachable in the previous location.

Second, for link fabrication attacks, we can add a controller-signed TLV into the LLDP packet and check the signature when receiving the LLDP packets. We can also add some extra logic to track the traffic coming from each switch port to decide which device is connected to the port.

V. Software Simulation

To understand topology discovery better, I make some research on currently popular controller operating systems (e.g. Ryu, NOX/POX, OpenDayLight). I have performed a topology discovery experiment on the Ubuntu operating system with an Open Flow simulation platform, Mininet and a controller operating system, Ryu.

First, I draw a topology in Mininet platform MiniEdit, a software in Mininet used to create an Open Flow network manually. The topology I drawn is shown in the *Figure.5*. Its topology is a 3-layer tree.

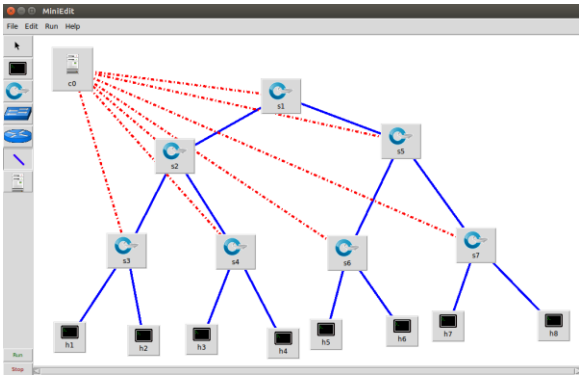

Figure.5

Second, I use the topology discovery function, "*gui_topology.py*", in Ryu to view the topology of the network I created. The topology is shown in the *Figure.6*, we can see switches and internal links among them.

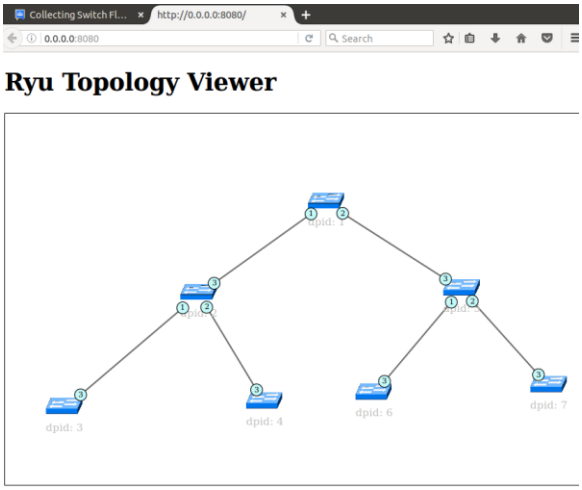

Figure.6

Furthermore, in the discovered topology, controller and hosts are invisible. Because we have not considered the problem of multiple controllers, only one controller is in the network, and it will connect to every switch. For hosts, Ryu does not have the function to discover hosts, this will be the part I am going to work in the future.

VI. FUTURE CHALLENGES

In [8], the authors describe several challenges we may encounter in the future. (1) Multiple SDN Domain: The SDN controllers controlling different domains create complication in sharing topology discovery information. (2) Topology discovery through Open Flow switches： Reduce the working burden of controllers. (3) Identify fake links: Know about status of every link. (4) Frequent host migration: make the work more sophisticated. (5) Safety of topology information: prevent the attackers to access the topology information. (6) Controller upgradation: make discovery mechanism upgraded at the same time.

VII. Conclusion

Topology discovery is an essential technology for networks. A lot of network management applications are based on it. In SDN, all of information is centralized to controller. Thus, it is important for the controller to have the completed topology view of the network. However, SDN will suffer from attacks if we do not improve the security mechanism. In the future, we should pay attention to the security aspect of SDN. Moreover, we should follow the development of SDN and solve occurring problems to make it better.

This semester, I read a lot of papers on this topic. I also complete some hands-on experiments. In the future, I will keep my attention on this topic. I will improve the topology discovery software and read more on the future challenges.

REFERENCES

[1] Y. Bejerano, Y. Breitbart, M. Garofalakis, and R. Rastogi, “Physical topology discovery for large multi-subnet networks,” in Proc. IEEE INFOCOM, 2003, pp. 342–352.

[2] M. Gunes and K. Sarac, “Inferring subnets in router-level topology collection studies,” in Proc. ACM/USENIX Internet Measurement Conference (IMC), November 2007.

[3] Emma, A.Pescape and G.Ventre. Discovering Topologies at Router Level. IPOM 2005, LNCS 3751, pp. 118–129, 2005.

[4] https://www.opennetworking.org/sdn-resources/openflow

[5] GENI Wiki. [Online]. Available: http://groups.geni.net/geni/wiki/OpenFlowDiscoveryProtocol

[6] Pakzad, M. Portmann, W. L. Tan, and J. Indulska, "Efficient topology discovery in OpenFlow-based Software Defined Networks," Computer Communications, 2015.

[7] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures," in NDSS, 2015.

[8] S. Khan, A. Gani, A. Wahab, M. Guizani, M. Khan, “Topology Discovery in Software Defined Networks:Threats, Taxonomy, and State-of-the-art,” DOI 10.1109/COMST.2016.2597193, IEEE Communications Surveys & Tutorials