Im-OFDP: An Improved OpenFlow-based Topology Discovery Protocol for Software Defined Network

Yongpu Gu[1,2)], Dong Li[*1)], Junqing Yu[1,2)]
[1)](Network and Computation Center,Huazhong University of Science and Technology,Wuhan Hubei,430074,China)
[2)](Department of Computer Science,Huazhong University of Science and Technology,Wuhan Hubei,430074,China)
(guyongpu,lidong,yjqing)@hust.edu.cn

***Abstract*—Current controllers use the OpenFlow Discovery Protocol(OFDP) to explore the underlying network topology. For improving the performance and safety of OFDP, an improved protocol Im-OFDP is proposed by taking advantage of minimum vertex cover algorithm. Experiments demonstrate that Im-OFDP outperforms OFDP and OFDPv2 in terms of LLDP packet number, controller CPU load and safety.**

***Index Terms*—Software-Defined Networking, Topology Discovery, OpenFlow**

I. Introduction

Topology discovery service is utmost important for software-defined network(SDN) in terms of network monitoring, diagnosing, and resource management etc[1]. OFDP is a basic protocol for discovering network topology and adopted by most SDN controllers. However, some research works find that OFDP is vulerable because of low efficiency and poor safety[2, 3].

Aiming at improving the performance of OFDP, OFDPv2 can reduce the number of LLDP *Packet_Out* messages by installing pre-defined flow rules and modifying *Packet_In* event handler on the controller[3]. This paper proposes an improved topology discovery protocol Im-OFDP which is based on OFDP and OFDPv2. It divides the process of topology discovery into two stages: initializing and updating. In the first stage, controller will collect topology information via OpenFlow messages, explore the network structure via minimum vertex cover algorithm, and install pre-defined flow rules on switches to forward LLDP packets for topology discovery; In the second stage, controller will send out a small number of LLDP *Packet_Out* messages periodically to update the network topology information in real-time.

Experiments show that compared to OFDP and OFDPv2, Im-OFDP can achieve a great reduction in the total number of LLDP *Packet_Out* and *Packet_In* messages with up to 70% over OFDP and 50% over OFDPv2 respectively. In addition, it can effectively defend against link fabrication attacks due to innovations on topology discovery mechanism.

II. ALGORITHM AND PROCEDURE

In this paper, the basic hypothesis is that all links between switches are bidirectional, which is conform to actual situation. Im-OFDP can reduce the number of *Packet_Out* and *Packet_In* messages with LLDP packets significantly under this condition.

A. Supporting Switch Exploring Algorithm

For formulating, supposing graph $G(V, E)$ represents the SDN network topology, V is the set of vertices (switches): $V = \{S_i | i \in N\}$, and E is the set of edges(links): $E = \{l_{ij} = (S_i, S_j) | S_i, S_j \in V, i \neq j\}$.

Definition 1: *(Covering set of switches): $V' \subseteq V$, $\forall l_{ij} \in E$,if $S_i, S_j \in V'$, then we define V' is a covering set of switches for V.*

Definition 2: *(Minimum covering set of switches): For a covering set V', if $\forall$ covering set $V'' \subseteq V$, $|V'| \leq |V''|$, we define V' is the minimum covering set of switches for V.*

Definition 3: *(Supporting Switch): If switch $S \in V'$, V' is the minimum covering set of switches for V, then we define S is a supporting switch of network $G(V, E)$.*

In this paper, we use a hybrid greedy algorithm to figure out it. A theorem can be deduced based on the above definitions:

Theorem 1: *Algorithm 1 yields the least number of Packet_Out messages with LLDP packets.*

According to *Definition 1* we can infer that all the links in the network can be detected via supporting switches. According to *Definition 2 and 3*, it can be concluded that supporting switch set has the least number of links that cover the entire network within one hop.

B. Flow Rules for Topology Discovery

For topology discovery, Im-OFDP utilizes three specific flow rules for controller to pre-install on switches. Table I shows the attributes of these rules. *Rule I* is installed on the supporting switches, *Rule II* and *Rule III* are installed on all switches. In *Rule I*, the output ports are assigned by controller.

Rule I is high priority, means if switch receives *Packet_Out* messages from controller with LLDP packets with *dl_src=0*, it will forward it on ports assigned by controller. *Rule II* is middle priority, means if switch receives LLDP packets with *dl_src=0*, it will rewrite the *dl_src* field with mac address of the port which receives the LLDP packets and sends it back on the same port. *Rule III* is low priority, means if switch receives LLDP packets, it will forward it to controller within *Packet_In* messages.

*Corresponding author.
Annex to ISBN 978-3-903176-28-7© 2020 IFIP.
This work was supported by National Key R&D Program of China (2017YFB0801703).

Algorithm 1 : Supporting Switch Exploring Algorithm

INPUT: *Network topology* $G(V, E)$
OUTPUT: *Supporting switch set* V'
Step 1: *In* $G(V, E)$*, generating switch set* $\bar{V}$*. Let* $d^* = max\{d(v)|\forall v \in V\}$,$\bar{V} = \{v|\forall v \in V, d(v) = d^*\}$.
// $d(\bullet)$ means the degree of vertex.
Step 2: *If* $|\bar{V}| = 1$,*for* $v^* \in \bar{V}$*, goto Step 4; If* $|\bar{V}| > 1$,*then goto Step 3;*
//$|\bullet|$ means the cardinality of a set.
Step 3: *Let* $\bar{E} = (v1, v2)|v1, v2 \in \bar{V}$*, in* $G(\bar{V}, \bar{E})$*, let* $d^{**} = max\{d(v)|\forall v \in \bar{V}\}$,$\bar{\bar{V}} = \{v|\forall v \in \bar{V}, d(v) = d^{**}\}$,*for* $v^* \in \bar{\bar{V}}$*, goto Step 4;*
Step 4: *Add* v^* *to set* V_S*;*
Step 5: $E = E - \{(v, v^*)|\forall v \in V\}, V = V - \{v^*\}, if|E| = 0$*, then goto Step 6; or goto Step 1;*
Step 6: *Output supporting switch set* V'

TABLE I
FLOW RULES FOR TOPOLOGY DISCOVERY

No.	Priority	Match-Fileds	Actions
Rule I	High	in_port=CONT dl_type=0x88cc dl_src=0	OUTPUT=assigned PORT
Rule II	Middle	dl_type =0x88cc dl_src=0	mod_dl_src=mac of *in_port* OUTPUT=*in_port*
Rule III	Low	dl_type=0x88cc	OUTPUT=CONT

C. Topology Updating

For tracking the status of network topology, Im-OFDP establishes link information table to save realtime links between switches, and port mapping table to save the realtime mapping relationship between MAC address and port ID. When the status of link, device or port is changed, the table will be updated, and supporting switches will be re-explored, flow rules to forward LLDP packets will be redeployed on switches to enter a new round of network topology discovery,otherwise it means the status of network is unchanged,Im-OFDP will enter a new updating cycle directly.

D. Procedure of Im-OFDP

Figure 1 depicts the workflow of the Im-OFDP for discovering bidirectional link *(S1, Port1)-(S2, Port3)*. The detail is as follows:

Step 1: Controller installs the *Rule I II III* described in II(B) on *S1*, and installs *Rule II III* on *S2*.

Step 2: Controller sends out LLDP *Packet_Out* message in which the *ChassisID* is *S1*, the *PortID* is 0, and the *SrcMAC* is 0 to *S1*.

Step 3: *S1* receives the LLDP *Packet_Out* message from controller, then unencapsulating and distributing the LLDP packet to regulated ports in *Rule I*.

Step 4: *S2* receives the LLDP packet on Port3, then processes it as *algorithm 2*. If *SrcMAC=0*, then rewrites the *SrcMAC* field with MAC address of *(S2,Port3)*, and sends back to *S1* according to *Rule II*.

Step 5: *S1* receives the LLDP packet sent back from *S2* on Port1, then forwards it to the controller via *Packet_In* message.

Step 6: Controller receives the LLDP *Packet_In* message from *S1*, it can infer the existence of link *(S1, Port1)-(S2, Port3)*.

We can find that Im-OFDP only needs to parse a *Packet_In* message to learn a bidirectional link, while both OFDP and OFDPv2 require two *Packet_In* messages.

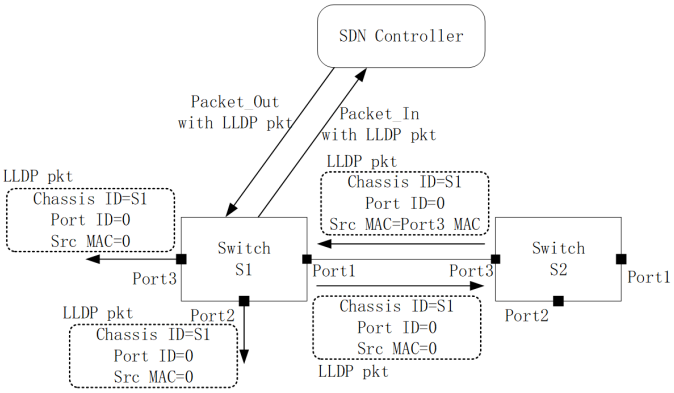

Fig. 1. Im-OFDP topology discovery procedure

Algorithm 2 : LLDP frame processing algorithm

```
for received packet pkt do
    if pkt.etherType = LLDP and pkt.inPort = CONT and pkt.srcMac = 0 then
        for all switch ports P in Rule I do
            forward pkt copy on port P
        end for
    else if pkt.etherType = LLDP and pkt.srcMac = 0 then
        pkt.srcMac = P.MACaddr
    else if pkt.etherType = LLDP then
        forward pkt to Controller
    end if
end for
```

III. THEORETICAL ANALYSIS

A. Performance Analysis

Assuming that in a SDN network the data plane is composed of switches and hosts,with $N(N \geq 2)$ Switches, $H(H \geq 0)$ Hosts, and $L(L \geq 1)$ Links.

TABLE II
NUMBER OF MESSAGES IN OFDP, OFDPV2 AND IM-OFDP

Protocol	*Packet_Out*	*Packet_In*	$\sum$
OFDP	$H + 2L$	$2L$	$H + 4L$
OFDPv2	N	$2L$	$H + 2L$
Im-OFDP	N_x	L	$N_x + L$

According to the principle of topology discovery protocol, we can calculate the number of LLDP *Packet_Out* and *Packet_In* messages in Table II.

For evaluating the efficiency of Im-OFDP, the performance metric is the efficiency gain G, G_1 and G_2 represent the relative number of control message reduction of Im-OFDP versus OFDP and OFDPv2 respectively. They can be caculated as equation (1) and (2) as in [3].

$$G_1 = \frac{(H+4L)-(N_x+L)}{H+4L} \geq \frac{3-\frac{N-1}{L}}{4} \geq \frac{1}{2} \quad (1)$$

$$G_2 = \frac{(N+2L)-(N_x+L)}{N+2L} \geq 1-\frac{2}{3+\frac{1}{L}} \geq \frac{1}{2} \quad (2)$$

Through theoretical analysis, we can find that the efficiency gain of Im-OFDP is not less than 1/2 against OFDP and OFDPv2.

IV. EVALUATION

A. Experiment Description

we use Pox as controller, Mininet as simulator, Openvswitch as SDN switch. All experiments were run on a DELL R730 server with i7-4770 CPU of 16 cores and 64 GB of RAM. The structure and parameter are shown in Table III. N_x denotes the number of supporting switches.

TABLE III
NUMBER OF MESSAGES IN OFDP, OFDPV2 AND IM-OFDP

Name	***Topology***	***N***	***P***	***H***	***L***	N_x
Topo1	Tree,d=4,f=4	85	424	256	84	17
Topo2	Tree,d=9,f=2	511	1532	512	510	170
Topo3	Linear,N=500	500	1498	500	499	250
Topo4	Fat Tree,k=6	45	270	54	108	18

B. Performance analysis

TABLE IV
NUMBER OF LLDP MESSAGES IN OFDP, OFDPV2 AND IM-OFDP

Topology	***OFDP***	***OFDPv2***	***Im-OFDP***	G_1	G_2
Topo1	592	253	101	82.9%	60.1%
Topo2	2552	1531	680	73.3%	55.6%
Topo3	2496	1498	749	70.0%	50.0%
Topo4	486	261	126	74.1%	51.7%

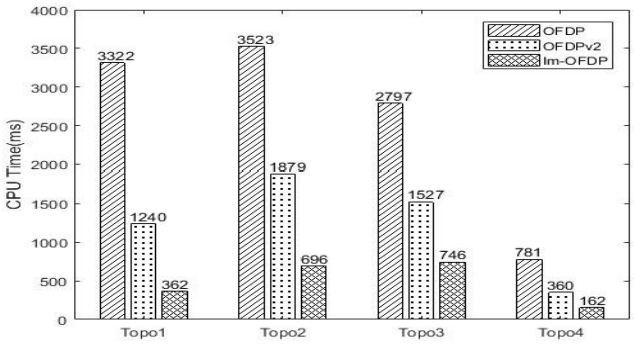

Fig. 2. Cumulative CPU time in OFDP, OFDPv2 and Im-OFDP

In the experiment, every topology is carried out by 60 times, and the polling interval is set to 5 seconds. The efficiency gain results shown in Table IV demonstrate that Im-OFDP can acheive a great reduction in the total number of LLDP messages. The CPU load analysis is shown in Figure 2, which can conclude that Im-OFDP is able to reduce the CPU load of controller up to 90% and 60% over OFDP and OFDPv2 respectively.

C. Security

Through investigating relevant works, attacks to SDN network topology are mainly divided into two categories: host location hijacking attack and link fabrication attack[1]. We simulate them and test the safety of Im-OFDP. Link fabrication attack can be implemented via two methods: LLDP forgery and LLDP replay. Each method is implemented by 50 times in every topology. The ratio of LLDP packets accepted by controller is shown in Table VI. 100% means all the LLDP packets forged/replayed by host are accepted by controller. 0 means controller doesn't receive the malicious LLDP packets.

TABLE V
NUMBER OF MESSAGES IN OFDP, OFDPV2 AND IM-OFDP

Attack	***Protocol***	***Topo1***	***Topo2***	***Topo3***	***Topo3***
LLDP Replay	OFDP	100%	100%	100%	100%
	OFDPv2	100%	100%	100%	100%
	Im-OFDP	0	0	0	0
LLDP Forgery	OFDP	100%	100%	100%	100%
	OFDPv2	100%	100%	100%	100%
	Im-OFDP	0	0	0	0

The results of experiment demonstrate that Im-OFDP can defend against the two topological attacks while both OFDP and OFDPv2 are vulnerable to them.

V. CONCLUSION

In this paper, we put forward an improved OpenFlow-based topology discovery mechanism Im-OFDP, which can achieve a great reduction in the total number of LLDP messages and controller CPU load compared to OFDP and OFDPv2,. In addition, Im-OFDP is immune to link fabrication attacks due to enhanced safety.

REFERENCES

[1] Eduard Marin, Nicola Bucciol, and Mauro Conti. An in-depth look into sdn topology discovery mechanisms: Novel attacks and practical countermeasures. In *acm conference on computer and communications security*, pages 1101–1114, 2019.

[2] Suleman Khan, Abdullah Gani, Ainuddin Wahid Abdul Wahab, Mohsen Guizani, and Muhammad Khurram Khan. Topology discovery in software defined networks: Threats, taxonomy, and state-of-the-art. *IEEE Communications Surveys and Tutorials*, 19(1):303–324, 2016.

[3] Farzaneh, Pakzad, Marius, Portmann, Wee, Lum, Tan, Jadwiga, and Indulska. Efficient topology discovery in openflow-based software defined networks. *Computer Communications*, 77:52–61, 2016.