Computer Networks 120 (2017) 56–70

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Global Flow Table: A convincing mechanism for security operations in SDN

Xiaofeng Qiu, Kai Zhang*, Qiuzheng Ren

Beijing Key Laboratory of Network System Architecture and Convergence, Beijing University of Posts and Telecommunications, Beijing, China

ARTICLE INFO

Article history:
Received 28 October 2016
Revised 13 March 2017
Accepted 3 April 2017
Available online 6 April 2017

Keywords:
Security operations
Software defined networking
Global flow
Weak vertex cover

ABSTRACT

One of the key challenges of network security is that security middle boxes, such as firewalls and Intrusion Detection Systems (IDSs), only have local view of the network. This lowers the efficiency of security detection and makes it difficult to locate the sources of the threats. There have been growing demands for security operations and appliances that are aware of the distribution and behavior of flows in the whole network; logically centralized control ability of Software-Defined Network (SDN) makes it possible for the network controller to acquire the global view of the network. In this paper, we propose a mechanism named Global Flow Table (GFT) which can provide security appliances and operators with paths of all the flows in SDN network, in addition to their sources, destinations, setup and terminate time, traffic volume and directions. A weak vertex cover based GFT algorithm which sacrifices less than 5% accuracy is also provided to improve scalability. Tests with different network topologies of cloud computing center and enterprise networks show promising performance. Utilizing the Global Flow Table, we built several applications to illustrate how GFT could benefit the security operations.

1. Introduction

Recently, profound interest has been developed in automating network control, particularly with the emergence of software-defined networks (SDN). SDN is a new software-based network architecture and technology. Loose coupling between control plane and data plane, centralized control of network state and transparency of underlying network facilities for upper-layer applications are the most important features of SDN. Network intelligence and state are logically centralized by network controller, which provides a global view of the whole network [1,2]. SDN switches are centrally managed through a well-defined southbound interface such as OpenFlow [3] which allows control plane to write forwarding rules to switches. OpenFlow is the only certified southbound interface by ONF (Open Networking Foundation), which has a pivotal position in the development of SDN.

Similar to the design philosophy of SDN, network security operation is also on its way to automation and intelligence. Besides, it also requires self-healing and real-time anomaly detection [4]. However, unlike their counterparts in network operation, security operators are still lack of tools or applications to obtain the global view of network behaviors.

In this paper we propose a new mechanism—Global Flow Table (GFT). It generates and stores the complete path of each flow containing the direction, passing switches, forwarding ports and traffic volume in the current network. It can make up the deficiency in security detection. For example:

(1) With the assistance of GFT, path abnormal attacks in network can be easily detected with simpler and faster algorithms. Although detection based only on GFT will provide relatively high false positive rate for some kinds of attacks, the GFT will greatly help to reduce the burden of security detection as only the suspicious flows or nodes will be reported to more sophisticated security devices for further fine grained detection. In Section 6.1, we demonstrate how suspicious Man-in-the -middle attack (MITM) that intercepts and inserts into the flow path can be intuitively detected utilizing the GFT. Besides, the source address forgery attack could also be detected by comparing the global flow pathes in GFT with information of links obtained by the northbound API of the SDN controller. In addition, Denial of Service (DoS) attackers [5] that drop packets maliciously in data plane of SDN can also be coarsely detected by checking the forwarding port item of the GFT.
(2) GFT could help to improve the efficiency of security protection against Distributed Denial of Service (DDoS) attack. By learning the passing path of each network flow provided by

* Corresponding author.
E-mail address: kaibao@bupt.edu.cn (K. Zhang).

http://dx.doi.org/10.1016/j.comnet.2017.04.002
1389-1286/

GFT, the Intrusion Detection Systems(IDS) can not only handle the flow in detection point, they could also notify the switches closer to the malicious sources to drop malicious flows in order to suppress attacking traffic.

(3) GFT could help to detect attacking flows with low traffic failed to trigger the threshold of IDS. For example, slow scanner scans tenant networks in a round-robin fashion and DDoS attackers launch attacks targeted to the victim from lots of Bots in order to keep the IDS not triggered. GFT will help the security appliances to detect these kinds of attackers that elaborately designed to defeat the threshold based detection algorithm. In Section 6, we demonstrate two applications upon GFT, the Global Flow Monitor and Global Flow Path visualization, both provide security operation with roughly suspicious flows or nodes that are eligible for further sophisticated detection.

GFT not only helps security operations, but also makes network operation and management more intelligent and efficient. Commercial network troubleshooting tools provide visibility through packet sampling [6,7], configurable packet duplication [8–10], or log analysis [11], etc. Most of the work mentioned above lack network-wide visibility and packet-level state consistency which are provided by GFT.

GFT is built based on flow tables of the SDN network. In the SDN architecture, control and data planes are decoupled. The network controller chooses optimal network path for application traffic through issuing flow table entries to each switch. All packets processed by a switch are compared against the flow table entries in this switch. Flow table entry contains a set of matching entries, activity counters, and a set of zero or more actions that the switch will conduct on the matching packets. The matching entry acts as identity of a flow in a flow table. Through identifying flow table entries, we can find the specific flow passing different switches along its path [12]. Therefore, flow table entries existing in SDN networks form the basis for building GFT.

The key challenge in creating GFT lies in flow information collection and path computation of all the flows in the whole network. The first step of creating GFT is to gather flow table entries. Flow table entries in switches could be read by the SDN controller through OpenFlow protocol between the controller and switches. GFT could further get these flow table entries through controller's northbound API. As introduced previously, during the flow setup, the controller could record most items of a flow table entry before issuing it to the switch. However, in order to acquire complete visibility in a large SDN network, the controller has to collect real-time statistic information in counters of flow table entries at each switch. Such fine-grained control and visibility come with two kinds of costs: switch-implementation cost involving the switch's control-plane and distributed-system cost involving the controller. As a result, collecting flow information from all the switches via the pull-based Read-State mechanism can actually create dreadful control-plane load. Meanwhile, GFT need timely access to statistics to accommodate the dynamic network. Moreover, statistics gathering competes for limited bandwidth between switch and controller for flow setup. The more frequently statistics are gathered, the fewer flows a switch can set up. Hence it is necessary to reduce the cost of collecting flow table entries while building the GFT [13].

In this paper, we proposed a weak vertex cover based GFT algorithm to find the optimum Collecting Set of switch nodes instead of gathering all the flow table entries from every switch using the Vertex Cover (VC) problem with flow-conservation law [14] to collect flow table entries. A VC is a set of nodes in a graph that every edge of the graph has at least one endpoint in the set. A minimum cover is the VC which has the smallest number of nodes for a given graph. According to the traffic of each edge associated with the VC set, we can determine the flow of arbitrary edge in the network. Since VC is a NP-hard problem, accordingly, it is required to reduce the number of effective measured set in order to find the minimum effective measuring set. Using a VC of the network graph to determine the set of nodes on which the controller collects the flow tables can obviously result in a substantial reduction in the number of measured nodes. Moreover, it is possible to be more effective by exploiting the flow-conservation law. Based on the flow table entries gathered from the Collecting Set of network nodes, all the flows are visible and the path of each flow can be computed. Further, it greatly reduce the cost of statistics gathering.

Path computation is also a key factor in the GFT. Flow table entries collected are only a set of flows, which cannot reflect the path information and traffic distribution. In this paper, we design a global path calculation algorithm to compute the complete path of all the flows existing in the current network. GFT exposes a restful API for applications to specify, receive, and act upon global flow information of interest. Based on the GFT, we developed several applications such as the Global Flow Graph, Global Flow Path visualization, MITM Detection application and Global Flow Monitor, which can significantly promote efficiency and performance of security operations. Details of these applications will be introduced in Section 6.

The contributions of this paper are as follows:

1. We propose a new mechanism named GFT and corresponding algorithms to build the global information of the entire network flows, which makes the security appliances aware of network behaviors and the security operations intelligent.
2. To reduce the cost of gathering the real-time information of flows, we harness the Weak Vertex Cover problem to find the optimum Collecting Set of measured switch nodes. Statistics of all the flow are only pulled by the controller from the switches in the collection set which greatly cut the cost of GFT.
3. Based on the collected flow table entries, we propose a global path calculation algorithm to recover and compute the path of each network flow. We also design a data structure to store GFT for the purpose of searching information on specific flows conveniently.
4. We developed several applications based on GFT, such as Global Flow Graph, Global Flow Path visualization MITM Detection application and Global Flow Monitor. These applications help network security managers to detect and trace the source network attacks. Meanwhile, they can also guarantee the correctness of the network behaviors.
5. In weak vertex based GFT, scalability is at the cost of accuracy, to apply GFT to different topologies of cloud computing center, campus and enterprise network, we carried out experiments in the scale of these networks. In consequence, the accuracy ratio of GFT is always more than 95% with different network topologies and scales when adding marginal nodes. The time delay is less than 1000 ms when there are 300 switch nodes in the network and each of the nodes contains 100 flow table entries.

The remainder of this paper is structured as follows: Related work is presented in Section 2. Global Flow Table is introduced in Section 3. Section 4 presents weak vertex cover based GFT algorithm. Section 5 is the experimental results and evaluation. The applications developed upon GFT are described in Section 6 and we conclude the paper in Section 7.

2. Related work

In this section, we review three categories of work. First, we discuss recent work on network security operation. Second, we compare the method of network data collecting. Last, we focus on the reduction of statistics gathering when collecting essential network forwarding information.

2.1. Network security operations

The decoupling of data and control plane as well as the centralized control and management characteristics, make network security operations in SDN different from those in traditional network. In recent works, most researches center upon network operations in SDN. For example, the VeriFlow [15] focuses on network invariant checks, including network loops, suboptimal routing and black holes, etc. NetSight [16,17] utilizes the packet histories to troubleshoot networks. Common bugs like reachability error and incorrect packet modification can be diagnosed by NetSight. NetPlumber [18] is a real time policy checking tool, which is applied to check global network properties like loop-freedom and reachability for nearly arbitrary devices regardless of the protocols being used. GFT mainly devotes to the network security operation in SDN. By utilizing the global path and essential flow information, the GFT can easily detect and screen the security faults and threats. Meanwhile, GFT is seated as a SDN application. Upon GFT, targeted security applications can easily be developed through the restful API afforded by GFT. Besides, the source code of GFT is accessible in GitHub, and the version of GFT will be updated irregularly. In addition, GFT is also able to diagnose network invariants, such as network loops and routing holes.

2.2. Method of network data collecting

Collecting network data is the first step of security operation and troubleshooting networks. In [16,17], every switch creates a postcard by duplicating the passing packet, then NetSight turns postcards into packet histories. This forces switches perform similar packet duplication actions to divert packets for monitoring. Meanwhile, the duplicating process occupies the bandwidth of switch, which may influence and reduce the performance of data transmission. A NetPlumber agent is created to communicate with the data and control plane in SDN [18]. Meanwhile, the agent updates events to NetPlumber and receives check results from it. There are many procedures in the entire communication process, which may causes inefficiency and faults. VeriFlow [15] sits as a layer between the controller and the devices, and checks the validity of invariants as each rule is inserted, which may cause the data forwarding process delayed. The whole network is conducive to be crashed if VeriFlow is out of order because of forceful attack or fatal bugs.

GFT utilizes the centralized management and control capability of SDN control plane. Unnecessary to modify any of the rules in network switches or duplicate network packets, our approach uses the north API released by the SDN controller to obtain all network forwarding rules. Besides, GFT runs on commodity hardware and requires no special support fromswitches other than the base OpenFlow 1.0 protocol implementation, which also removes the burden of duplicating network packets. In addition, GFT sitting as a network application makes no impact on the original SDN architecture.

2.3. Reduction of statistics gathering

As mentioned in the Section 1, the first step of creating GFT is to gather flow table entries. However, collection of a large amount of traffic can reduce the performance of network. To reduce the cost of gathering flow table entries, selecting a subset of measured switches is a necessity for covering every link in the network, which can be considered as an example of the well-known Vertex Cover (VC) problem over the network graph [14].

Using a VC of the network graph to determine the set of nodes on which the controller collects the flow tables can obviously result in a substantial reduction in the number of measured nodes. Moreover, it is possible to be more effective by exploiting the flow-conservation law [19]. Considering a router or switch v in the directed network graph G, one of the key observations found that each router or switch v satisfies the flow-conservation law which has been simply defined as the approximate equivalence relation between the sum of the traffic flowing into v and that out of v. The importance of the flow-conservation law for selecting the measured nodes lies in that it's no longer indispensable to ensure that all edges of a switch are covered. If the fact that a switch has k links and the traffic of k-1 of the links is known, then the traffic of the remaining links can be derived from the flow conservation equation for that switch.

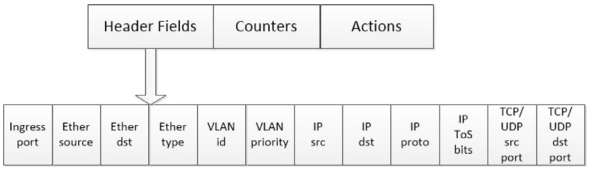


Fig. 1. The flow table entry of OpenFlow1.0.

3. Design of global flow table

3.1. The flow table mechanism of SDN

The design of GFT is based on the flow table mechanism of SDN, which is the fundamental basis of handling data packet in SDN switches. An OpenFlow switch contains one or more flow tables. All packets processed by the switch are compared against the flow table entries. Flow table entry is generated by the centralized controller based on the whole network topology utilizing the LLDP(Link Layer Discovery Protocol), and distributed to the switches on packet path. Taking OpenFlow1.0 for example, flow table entry consists of header fields containing a set of matching entries shown in the Fig. 1, activity counters, and a set of zero or more actions to match packets. The flow table mechanism of SDN breaks the hierarchy conception in traditional network. Encapsulations in different layers of protocol stack are listed in the flow table entry, whether the source or destination MAC, VLAN ID and other layer 2 network information, or the layer 3 network information such as source and destination IP addresses, or the source and destination TCP / UDP port numbers in the transport layer. Firstly, when a packet is forwarded to a switch, the packet header will be compared with the matching entries of the flow table entries in this switch. If a matching entry is found, any actions for that entry are performed on the packet. Otherwise, the packet is forwarded to the controller over the secure channel. The controller is responsible for determining how to handle packets without valid flow table entries, for most data packets, a new flow table entry will be generated and distributed to switches on packet forwarding path. The controller is also in charge of adding, removing and modifying flow table entries.

Briefly in summary, action field of a flow table entry in a switch determines the next actions that the switch will take on a specific flow, such as dropping or forwarding to next hop. The matching entry in the header field, as the identity of a flow, differenti-

Table 1
The structure of GFT.

Flow ID	Matching Entry Set	Packet count	Byte count	Create time	Finish time	Global path

ates one flow from another. Counters in the flow table entry stores statistics of the flow, for example, the number of packets.

3.2. The structure of GFT

Considering security operations, there is intense need to obtain the global information on the network traffic. Therefore, we calculate the GFT on the basis of the SDN flow table. In addition to the necessary information in the SDN flow table, such as IP src, IP dst, Ether source, the GFT contain the global path of each network flow, as well as its creation and termination time, which are necessary for the analysis of malicious behavior in the network.

As shown in the Table 1, a data structure of GFT is created to store and record the global information of network flows. Each global flow table entry contains:

- *FlowID*: to represent distinctive identity of each network flow.
- *MatchingEntrySet*: to record the essentials of network flows, such as IP src, IP dst, Ether source.
- *Packetcount*: to accumulate total packets of each network flow.
- *Bytecount*: to calculate total bytes of each network flow.
- *Createtime*: to record when the flow was created.
- *Finishtime*: to record when the flow was finished.
- *Globalpath*: to record the input and forwarding port, switch id of all the switches that a flow passed sequentially.

3.3. Steps of generating GFT

The first step of generating GFT is to collect flow table entries. In SDN network, applications like GFT could ask for flow table information through northbound interface of controller, who will read flow table of switches via southbound protocol, such as Openflow. However, collecting a large amount of flow table entries from all network nodes may bring heavy burden for both controller and switch. As tested in the DevoFlow[6], even if there is no other load on the switch, the statistic-gathering latency of the 5406zl (representative of the current generation of Ethernet switches) will be more than one second when its flow table has more than 5600 entries. In addition, statistics-gathering and flow setup compete for the limited switch-controller bandwidth,the more frequently statistics are gathered, the fewer flows the switch can set up.

To tackle this issue, the weak vertex cover problem is utilized to obtain the Collecting Set, the switch nodes set that are used to gather flow table entries. Therefore, flow information will be collected only through switches of the Collecting Set rather than all switch nodes. Minute principle and algorithm for generating the optimum Collecting Set will be introduced in the following Section 4.1.

The second step of generating GFT is to calculate the global path of each network flow. To obtain ordered, complete path of each network flow by using the flow table entries, we proposed a global path calculation algorithm, which will be detailed in the following Section 4.2.

A data structure is created to store and record the global information of network flows efficiently. Firstly, MD5 of the matching entry in the flow is calculated. It stands for the flow id, the key of the HashMap. HashMap is a compound structure which contains the set of the matching entry, the global path of the flow and other fields shown in the Table 1.

For security operation, time information of flows is also very useful. So, we add Create time and Finish time in GFT to make it easier for the traffic analysis and anomaly detection. We designed relevant algorithms to calculate the create and finish time of each flow. Network flow tables are collected periodically from switches inCollecting Set, and the GFTs are calculated by the GFT algorithm consequently. All global flow tables will be stored in database once they are generated. The create time of a network flow is recorded once the flow is stored for the first time. The finish time of a flow is calculated through packet timeout mechanism as in conventional other IP networks.

4. Weak vertex cover based GFT alogrithm

In this section, the algorithms employed in the step of generating GFT will be detailed individually. Firstly, the weak vertex cover based GFT algorithm will be introduced. Then, Section 4.1 details the algorithm on the selection of the Collecting Set when gathering flow table entries. Afterwards, Section 4.2 gives a comprehensive introduction of the algorithm on calculating global flow path.

Given a directed network graph G, a set S of nodes is defined to be a Weak Vertex Cover of G if it is possible to mark every node in G by iteratively performing the following two steps after initially marking each node in S as covered [20]:

- Mark every edge in G which is linked to a covered node;
- For any non-leaf node v in G, mark v if $deg(v)$-1 edges linked to v are marked.

The weak vertex set S in graph G is the effective set of measures with the flow conservation law constraints(Section 2.3). It is sufficient for GFT to derive the flow path and traffic on every link in the network by collecting flow tables only from the nodes in a Weak Vertex Cover. Firstly, the flows of switches associated with S can be obtained directly. Then, if $v \in S$ and the flows of $deg(v)$-1 edges have been obtained, the flows of the rest edge linked to v can be calculated according to the flow conservation law. Recursively, flows of all edges in graph$G$ can be derived.

Note that, the flow conversation law holds only approximately since there can be (a) traffic directed to/from switches (b) multicast traffic that is replicated along many output interfaces and (c) dropped packets in switches. In the first case, the traffic directed to/from switches are normally management traffic, which are far less than our concerned data traffic. Then, the GFT is originally designed for unicast. How to support multicast will be one of our future work. Besides, there are probably network faults or attacks when switch drops packets. As shown in the Section 5, the accuracy rate of GFT algorithm remains around 95%, which indicates the above undesirable cases are acceptable and tolerable.

Thus, given flow conservation at each switch of G, our selection of the collecting set becomes equivalent to determining a minimum Weak VC for G.

4.1. Greedy algorithm to create collecting set

Collecting Set includes marginal nodes whose degree is 1 and Weak Vertex Cover. To determine a minimum weak VC set is a NP-hard problem. One of the approximation algorithms is shown in the following greedy algorithm.

Given an undirected graph $G = (V, E)$, it is essential to add all the nodes whose degree is 1 to the Collecting Set and delete them

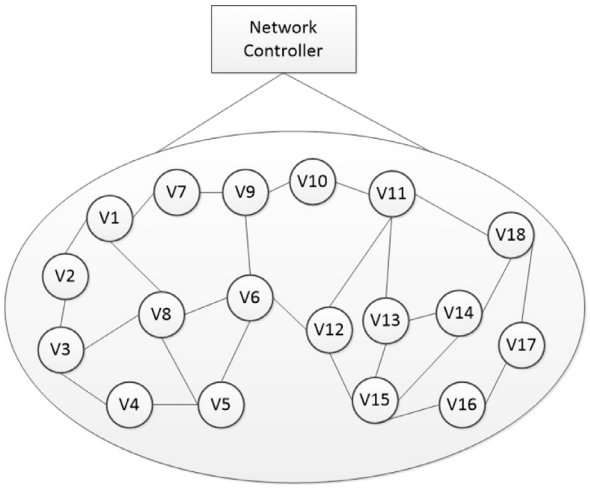


Fig. 2. The graph structure of the network topology.

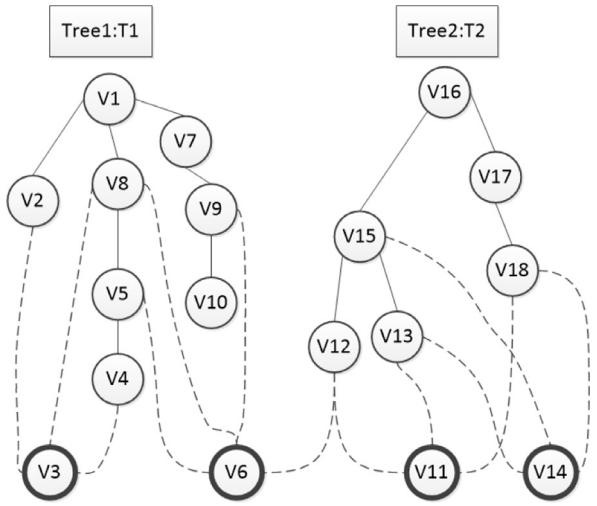


Fig. 3. The forest of the topology which removes the weak vertexes.

from the node set V at the beginning of greedy algorithm, because these nodes are the input and output nodes that cannot satisfy the flow conservation law.

In the iterative process of greedy algorithm, the chosen nodes are added to the Weak Vertex Cover and the edges linked to the nodes are deleted from E. The algorithm terminates when the edge set E is empty, which means that information of flows on all the links could be gathered by collecting flow table entries from this set of chosen nodes, named Collecting Set.

Notice that due to the nonuniqueness of weak vertex cover selection, the calculated Collecting Set may not be unique. Besides, if the condition in line 8 is loosened, a Collecting Set with more and different nodes will be produced.

4.2. Global path calculation algorithm

The data collected from the Collecting Set through the northbound API of SDN controller are flow table entries representing network forwarding rules instead of flow path expected in GFT. Global path calculation algorithm is designed to separate all network flows and calculate the passing path of each flow from these flow table entries.

The procedure of global path calculation algorithm is divided into three parts: network topology graph construction, flow classification and path computation. Firstly, network topology is abstracted as a graph structure as the basis of the subsequent work. Then, flow classification distinguishes each flow in the current network and find out which switches of the Collecting Set the flow goes through. Finally, path computation computes the accurate passing path of that flow.

4.2.1. Network topology graph construction

SDN flow is unidirectional, a bidirectional connection is represented in two unidirectional SDN flows. So, as shown in the Fig. 2, we model a SDN network as an undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots v_n\}$ denotes the set of network nodes (i.e., switches) and $E = \{e_1, e_2, \ldots e_m\}$ represents the set of edges (i.e., physical links) connecting to the switches. Detailed definitions of these parameters are provided below:

- *Vertex:* v_j. Each vertex v_j represents the unique switch datapath(dpid) as the switch identification which has a set of port numbers related to the links.
- *Edge:* $e(v_i, v_j, p_m, q_n)$. The edge stands for link connection with a pair of port numbers attached. Such as $e(v_i, v_j, p_m, q_n)$ represents a link e joining the node v_i and v_j with the port number p_m belonging to v_i and q_n belonging to v_j.
- *Tuple of vertex and port.* (v_i, p_m) represents one end of a link connecting the switch v_i through port p_m.

To compute the global path of each flow in current network, we use the lemma 1 of weak VC [21].

- *Lemma*1.: For any node v in an undirected network graph $G = (V, E)$, it is satisfied with the condition that its degree is $Degree(v) >= 2$, the set $S(S \subset G)$ is one of G's weak vertex covers, if and only if $G' = (V', E)$ is a forest which is satisfied with the condition that $V' = V/S$, $E = \{(u, v) | (u, v) \in E \wedge u \in V' \wedge v \in V'\}$.

Lemma 1 of weak VC [21] indicates that if the weak vertex cover set and their associated edges are removed, the remaining nodes and edges can form a forest. The weak vertex cover of the network topology graph in the Fig. 2 is $\{v_3, v_6, v_{11}, v_{14}, \}$ according to the greedy algorithm. Therefore, a forest F can be generated by removing the weak vertex cover set and associated edges of the topology graph.

So, as shown in the Fig. 3, in this step, original network topology graph in Fig. 2 is reconstructed into a Weak Vertex Cover, a forest F constituted of two trees $T = \{T_1, T_2\}$ and connections between the two trees and weak vertexes represented as the dotted lines.

- *Nodes*: $((n_i, p), parent)$. Every node of the tree contains two aspects: the node and the position of its parent node. Further, the node is described as the tuple of vertex and port (n_i, p). n_i represents the switch dpid while p represents the port connected to its parent node. *Parent* is the position of its parent node.
- *Root*: r. Root is the root node of the tree and the position of its parent node is -1.

4.2.2. Flow classification

The purpose of flow classification is to distinguish each flow in the network and find out which switches of the Collecting Set the flow goes through. Table of HashMap is created to effectively store the flow and its relevant switch set in the form of key and value. The field of matching entry which stands for flow identity is regarded as the key while the tuple list of switch dpid and port numbers is the value. The process is as follows.

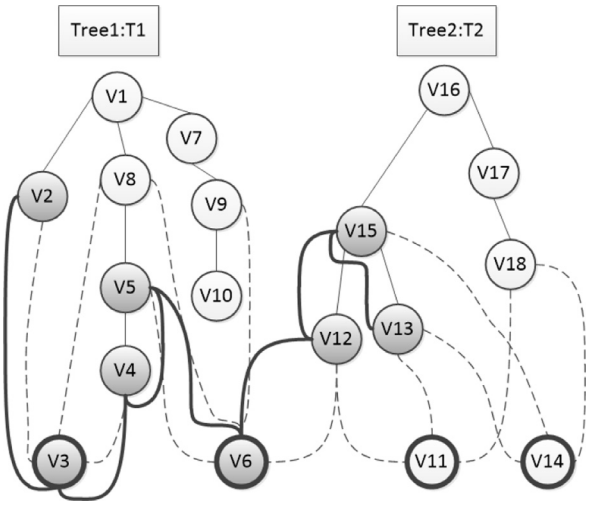


Fig. 4. The example of a flow which passes the forest and weak vertexes.

Algorithm 1 Greedy algorithm for the weak vertex cover.

input: (1) $G = (V, E)$
output: Collecting Set C

```
function GREEDY
    C ← φ
    while deg(v) = 1 in G_i = (V_i, E_i) do
        C ← C + v_i
        V_c ← V_i − v_i
    end while
    while E ≠ φ do
        Choose v_i whose deg(v_i) is the max in G_i = (V_i, E_i)
        C ← C + v_i
        V_c ← V_i − v_i
        E_c ← E_i − Adj(v_i)
        while deg(v) = 1 in G_i = (V_i, E_i) do
            Delete v and all edges adjacent to v
        end while
    end while
end function
```

The first step is traversing the first flow table entry obtained from the Collecting Set, and regarding its matching entry as a key of the HashMap. Then the switch dpid together with the input and output ports of the flow entry are added to the tuple list as the value.

Further, the next flow table entry will be traversed meanwhile determining whether the keys of the HashMap contain its matching entry. If the matching entry has existed in the keys of the HashMap, then the tuple of its corresponding switch dpid and port numbers will be added to the value list of the existing key. If not, a new key will be created for the matching entry in the HashMap. Similarly, the tuple of the switch dpid and its input and forwarding ports will be one member of the tuple lists as the value.

After traversing all the flow table entries gathered from the Collecting Set, all network flows and their corresponding passing switches from the Collecting Set can be classified.

For example, assume there is a flow in the network graph, as depicted in Fig. 4. The grey vertexes represent the nodes the flow goes through. The black bold lines represent the flow path which is $\{v_2, v_3, v_4, v_5, v_6, v_{12}, v_{15}, v_{13}\}$.

After the procedure of flow classification, the Collecting Set $\{v_3, v_6\}$ which this flow passes is obtained.

4.2.3. Path computation

After flow classification, what we get is not the complete flow path but a list of unordered dpid, port pair representing the weak vertexes each flow passed. Therefore, path computation is designed to compute the complete flow path of each network flow.

Time performance is the key point we considered in path computation. According to 4.2.1, network topology graph is divided into trees by the weak vertexes. Correspondingly, the global path of each flow will be divided into several segments in different trees. The starting and end nodes of each segment connected with the weak vertexes are in the same tree. In addition, the path between any two nodes from the same tree is assured and exclusive based on the properties of tree. Therefore, each segment of the flow path can be computed if the starting and end nodes of the segment are found.

Subsequently, the global flow path will be constituted by connecting all the flow segments with weak vertexes. The pseudo code of path computation is shown in the Algorithm 2.

In our example, according to the previous step, as is shown in Fig. 4, the weak vertexes the flow passed are $\{v_3, v_6\}$. Besides, the start and end nodes of the whole path is necessary to be found in order to remodel the complete flow path. Based on the source and destination addresses from the flow's matching entry and the link connections between network users and switches obtained from northbound API of SDN controller, we can easily find the source node of the flow is v_2 while the destination is v_{13}. In addition, the previous and the next nodes of the two weak vertexes of the flow could be found as $\{v_2, v_4, v_5, v_{12}\}$, which stand for the end and the start node of flow segments respectively.

Through the process, we get a node lists for the flow in each tree, containing start and end nodes $\{v_2, v_4, v_5\}$ and $\{v_{12}, v_{13}\}$.

The next step is to judge which two nodes are the starting and end node of a segment. Firstly, compute the paths from root node to every node in the node list in every tree and store these paths in the path list. The path list of tree T1 in the Fig. 4 is $\{v_1, v_2\}$, $\{v_1, v_8, v_5, v_4\}$, $\{v_1, v_8, v_5\}$.

Then, traverse the path list, compare and select the two paths in which the number of the same nodes is the highest as a couple. The last nodes of the two paths are the starting and end nodes in a segment. Then delete the two paths in the path list and continue to find another couple in the same way. Until the path list is empty, the starting and end nodes of all the segments distributed in the tree can be computed. For example, in the tree T_1, v_4 and v_5 are the two ends of a segment for the flow. Then the paths of each segment are $\{v_2\}$, $\{v_4, v_5\}$. Similarly, in the tree T_2, the path of each segment are $\{v_{12}, v_{15}\}$ and $\{v_{13}\}$.

By means of the aforementioned procedures, all the segments of the flow divided by the weak vertexes are found. Thus, to recover the global path of the flow, the weak vertexes are used to connect the segments from the source node to the destination node. As a result, the global path of the flow is $\{v_2, v_3, v_4, v_5, v_6, v_{12}, v_{15}, v_{13}\}$ after combing all the segments, which is in full accord with the actual path of the flow.

5. Performance analysis and evaluation

To further validate the accuracy and time performance of the algorithm under different topologies, we established a SDN simulation network. As shown in Fig 5, Mininet [22] is installed on PC1 to create various virtual network with different topologies that may appear in cloud computing center, campus and enterprise networks. FloodLight as SDN controllers [23] is installed on Server. Our global flow table related JAVA software is implemented on PC3. The hardware of PC1 and PC2 is a ThinkPad X200, while Server1 is a DELL PowerEdge R720.

Algorithm 2 Path Computation.

input: (1) Weak vertexes set $U(c)$: Each vertex $c(in, v, out)$ contains the node v and the input port number in and the output port out. (2) Graph G (3) Forest $f(t)$: ft contains a set of trees t. Each tree contains a list of nodes.

output: The ordered link list $L(t(u, p))$ which represents the path of the flow going through

function PATH COMPUTATION
  $V \leftarrow \phi$; $pathlist \leftarrow \phi$; $seglist \leftarrow \phi$; $N(t) \leftarrow \phi$
  for each c in $U(c)$ **do**
    Get pervious switch p and next switch n of c
    $V \leftarrow V + p$; $V \leftarrow V + n$
  end for
  for each t in $f(t)$ **do**
    for each c in $V(c)$ **do**
      if c in the t **then**
        $N(t) \leftarrow N(t) + c$
      end if
    end for
  end for
  for each t in $f(t)$ **do**
    find the path p between root node and the recorded node in the $N(t)$
    $pathlist \leftarrow pathlist + p$
  end for
  while $pathlist \neq \phi$ **do**
    find two paths:$path1, path2$ which the number of the same nodes is highest in the $pathlist$;
    $startnode \leftarrow path1.last$;$endnode \leftarrow path2.last$
    find the path seg between $startnode$ and $endnode$.
    $seglist \leftarrow seglist + seg$
    Delete $path1, path2$ from $pathlist$
  end while
  combine all the seg in the $seglist$ and weak vertexes from source to destination
  put the combined seg in L
end function

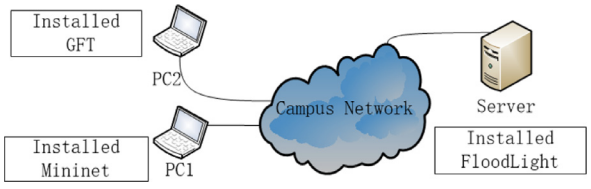


Fig. 5. The example of a flow which passes the forest and weak vertexes.

Table 2
Parameters used in the algorithm.

Parameters	Number
The total number of all switches in the network	n
The average number of flow table entries in each switch	p
The total number of all weak vertexes in the network	q
The average number of links connected with each switch	k
The average weak vertex number of each flow passing by	w
The total number of all flows in the network	s
The total number of all flows in all the weak vertexes	u
The total number of all the links in the network	t

5.1. Time and space complexities of proposed GFT algorithm

Parameters used in our proposed GFT algorithm is shown in the Table 2. Then, the time complexities of the procedures for generating GFT is shown in Table 3.

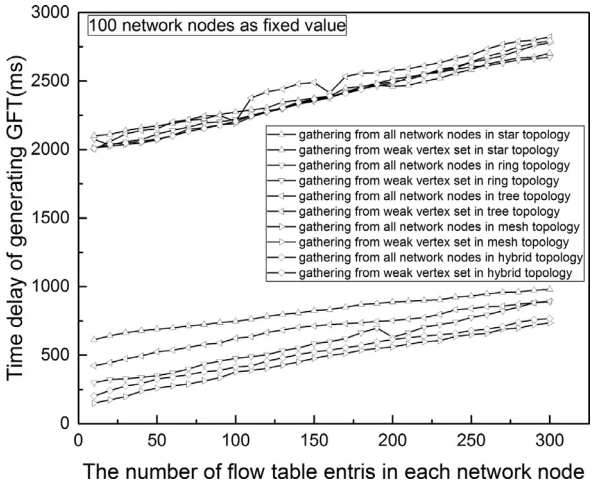


Fig. 6. Time delays with expansion in network flows.

The total time complexity of the algorithm is $T(n) = T_1(n) + T_2(n) + T_3(n) + T_4(n)$. In the best case, the time complexity is $T(n) = O(t + n + u + w * s)$, while $T(n) = O(t + n^2 + u + w * s)$ in the worst case. It can be deduced that the major contributors affecting the performance of the algorithm are derived from two aspects–the scale of network topology and the size of network flow. Further, the scale of network topology implies the number of switches and links in the network as well as different types of network topology; while the size of network flow indicates the total number of flow table entries in all weak vertexes and the amount of network flows. Analogously, the space complexity analysis of the algorithm, which is accord with the time complexity, is $S(n) = O(t + n + u + w * s)$ in the best case and $O(t + n^2 + u + w * s)$ in the worst case. In the follow-up sections of this paper, experiments were carried out to analyze these two dimensions which impact the performance of the algorithm.

To test the time performance of the GFT algorithm, we tested the time delay of generating GFT when collecting flow table entries from all network switch nodes and only from the collecting set. According to the previous analysis, the main factors influencing the time complexity of the GFT algorithm are the scale of the network topologies and traffic. Therefore, analyzing the time performance of the algorithm is mainly from these two aspects.

As shown in Fig. 6, X-axis is the number of flow table entries from each switch node in five diverse network topologies which are star, ring, tree, mesh and hybrid topologies. The hybrid topologies we used is a mixture of "star- ring" topologies. Each topology contains 100 switch nodes to make sure the normal functioning of network in cloud computing center, campus and corporation. While Y-axis is the time delay of generating global flow tables. The total number of the generated GFT are approximately 30000. From the trend of curve, it is visibly shown that the time delay of generating GFT only from the collecting set is less than one-third of that from all the network switches.

Simultaneously, it also can be seen from the picture that there remain discrepancies in the time delay when generating GFT from collecting set in different network topologies. Of those, star topology has the maximum time delay, and close behind is the tree topology, while the time delays of the mesh and hybrid topologies are lower.

It can be inferred from the principle of the proposed greedy algorithm that when searching for the collecting set, all the nodes whose degree is 1 will be added to the set. Also, the number of the nodes whose degree are 1 in star topologies are larger than that of mesh and hybrid topologies, so the time delays of generating global flow tables are higher in star topologies.

Table 3
The time complexity in the procedures of generating GFT.

Time complexity	Procedures
$T1(n) = O(n)$ in the best case and $O(n2)$ in the worst case	Collecting Set creation
$T2(n) = O(n * k + q * k) = O(n * k) = O(t)$	Network topology graph construction
$T3(n) = O(p * q) = O(u)$	Flow classification
$T4(n) = O(w * s)$	Path computation

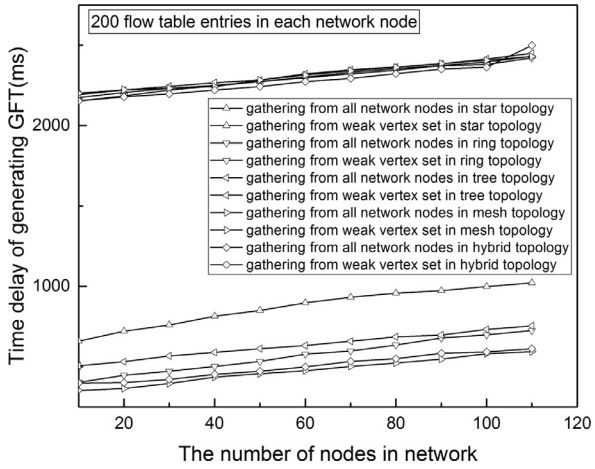


Fig. 7. Time delays with expansion in network topology scales.

With regard to the Data Computing Center, campus and enterprise, in which the flow table entries from each switch are no more than 300 typically, the time delay of generating GFT is less than 1000 milliseconds based on experimental results. It showed a good time performance of the algorithm.

Similarly, to verify the time performance of the algorithm with the incremental complexities of network topologies, we carried out experiments under five different network topologies, where the flow table entries from each switch are set to 200, and the corresponding time delays were shown in Fig. 7.

The result concludes that the weak vertex cover based GFT algorithm can dramatically reduce the time delay of generating global flow tables. Besides, as for the cloud computing center, campus and enterprise networks, the numbers of network nodes in which are not large, the proposed GFT algorithm has low time latency. Moreover, considering hardware limits, the time performance of the GFT algorithm can be affirmatively improved if running on a server with higher capabilities.

5.2. The accuracy of GFT algorithm

There are several contributing factors that may influence the accuracy performance of GFT algorithm in the generation of GFT. Firstly, in the step of creating collecting set through the proposed greedy algorithm, the number of marginal nodes added to the collecting set may influence the accuracy of GFT algorithm. Second, there may be little management traffic that directed to or from switches and dropped packets caused by network faults or attacks that violate the flow conservation law in the process of global path calculation. Besides, the marginal nodes and collecting set can be variant in divert network topologies. The complexities of network topologies and traffic may also cause inaccuracy of GFT algorithm. To satisfy the using cases in cloud computing center, campus and corporation, the high accuracy performance of GFT algorithm should be guaranteed in the expected network scales of topologies and traffic.

The accuracy performance of GFT algorithm is tested from two aspects: (1) whether the path of each network flow is correctly calculated (2) whether network flows are completely collected.

Firstly, to verify whether the path of each network flow is calculated with no errors, a number of specific flows which are recorded with the sources, destinations and paths were injected to the network. Comparing the actual flow paths with the paths computed by the GFT algorithm, it showed that all flow paths could be correctly calculated expect the following situation:

- If the packet header of a network flow, which is used to compare with the matching entries, is modified during the process of flow forwarding. That is, the flow id will be changed after this procedure. Therefore, the flow whose packet header is modified will be regarded as a different one from its previous flow before modified. However, it's suitable to treat them as the same flow. The solution for this case is one of our future works.

Then, to test whether network flows are completely collected through the GFT algorithm, we evaluated the amount of the actual flows and the flows generated by GFT algorithm.

As mentioned, the number of marginal nodes added to the collecting set may influence the accuracy of GFT. Adding these marginal nodes to the collecting set can increase the number of nodes used to collecting network data, further reduce the time performance of generating GFT. However, the accuracy performance may be reduced without adding marginal nodes, because the flow conservation law may not be completely satisfied. We tested the two extreme cases to analyze the effect of marginal nodes on the accuracy of the GFT algorithm. Firstly, the accuracy performance of GFT algorithm was tested when adding all marginal nodes to collecting set. Then, we tested the accuracy performance of GFT algorithm without adding any marginal nodes in the same experimental environment. Besides, to test the accuracy performance influenced by diverse topologies, we launched our experiment in 5 typical topologies, which are star, ring, tree, mesh and hybrid topologies. Meanwhile, the complexities of network topologies and traffic are used as variables to verify the robustness of GFT and whether the GFT algorithm can meet the requirements of network in cloud computing center, campus and corporation.

Firstly, aimed at verifying the accuracy of the GFT algorithm when expanding the network scales under various topologies, subsequent experiments were tested. We initially set 200 as a fixed value for the number of flow table entries in each switch node of the whole network, which can largely meet our normal request for cloud computing center, campus and enterprise networks. The accuracy ratio line chart of the flow paths generated by GFT algorithm is depicted in Fig. 8 when network nodes increased from 10 to 110. As shown in the picture, the accuracy ratio of the GFT algorithm remains more than 95% with a gradual increase of network nodes.

Secondly, in order to evaluate the accuracy of our proposed GFT algorithm with the increase of network traffic under various topologies, we launched our experiment in 5 typical topologies, which are star, ring, tree, mesh and hybrid topologies, each containing 100 switch nodes. Such scales of network topologies can surely support the normal functioning of network in cloud computing center, campus and corporation. The line chart drawn from

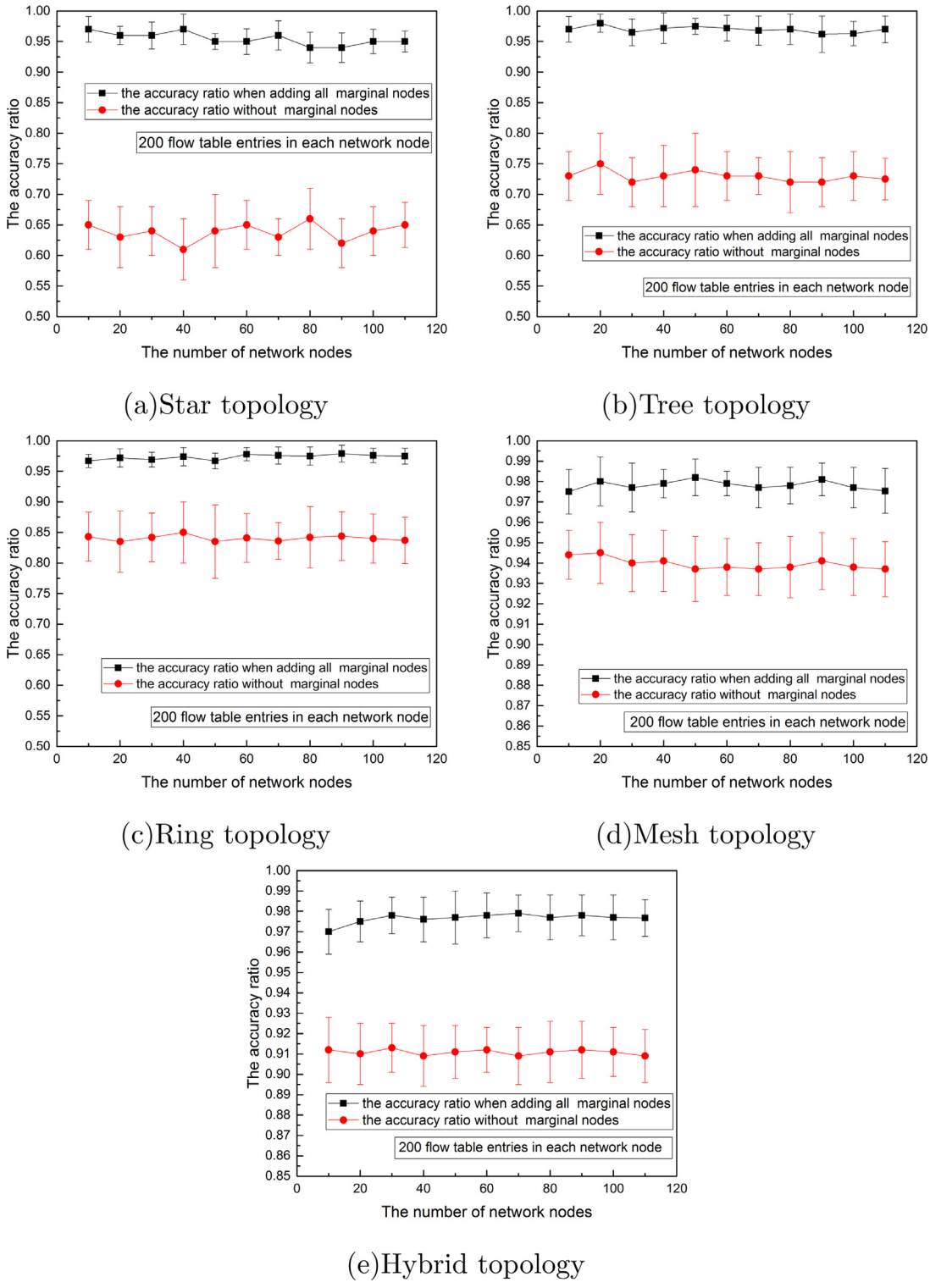


Fig. 8. The accuracy ratio of measured flows to the actual flows with expansion in network topology scales(note that the starting point of Y-axes in pictures (d) and (e) is 0.85 for better illustrating).

Fig. 9 demonstrates the accuracy ratio of the flow paths generated by GFT algorithm compared with the actual network flow paths when the number of flow table entries in each switch node of the whole network increased from 10 to 300. It states clearly that the average accuracy ratio of GFT algorithm is always over 95% in different topologies with the increase of network traffic.

To further validate the impact of marginal nodes, we compared the accuracy ratio when adding all marginal nodes and without them. As a result, the average accuracy ratio of the cases without adding marginal nodes is remarkably lower than that adding all marginal nodes. Meanwhile, the offset of accuracy ratio is larger without adding marginal nodes. And the accuracy ratio is more vulnerable to be affected by the different types of networks in the circumstances without adding marginal nodes.

The above experiments manifested that our proposed GFT algorithm can obtain high accuracy in diverse network topologies and

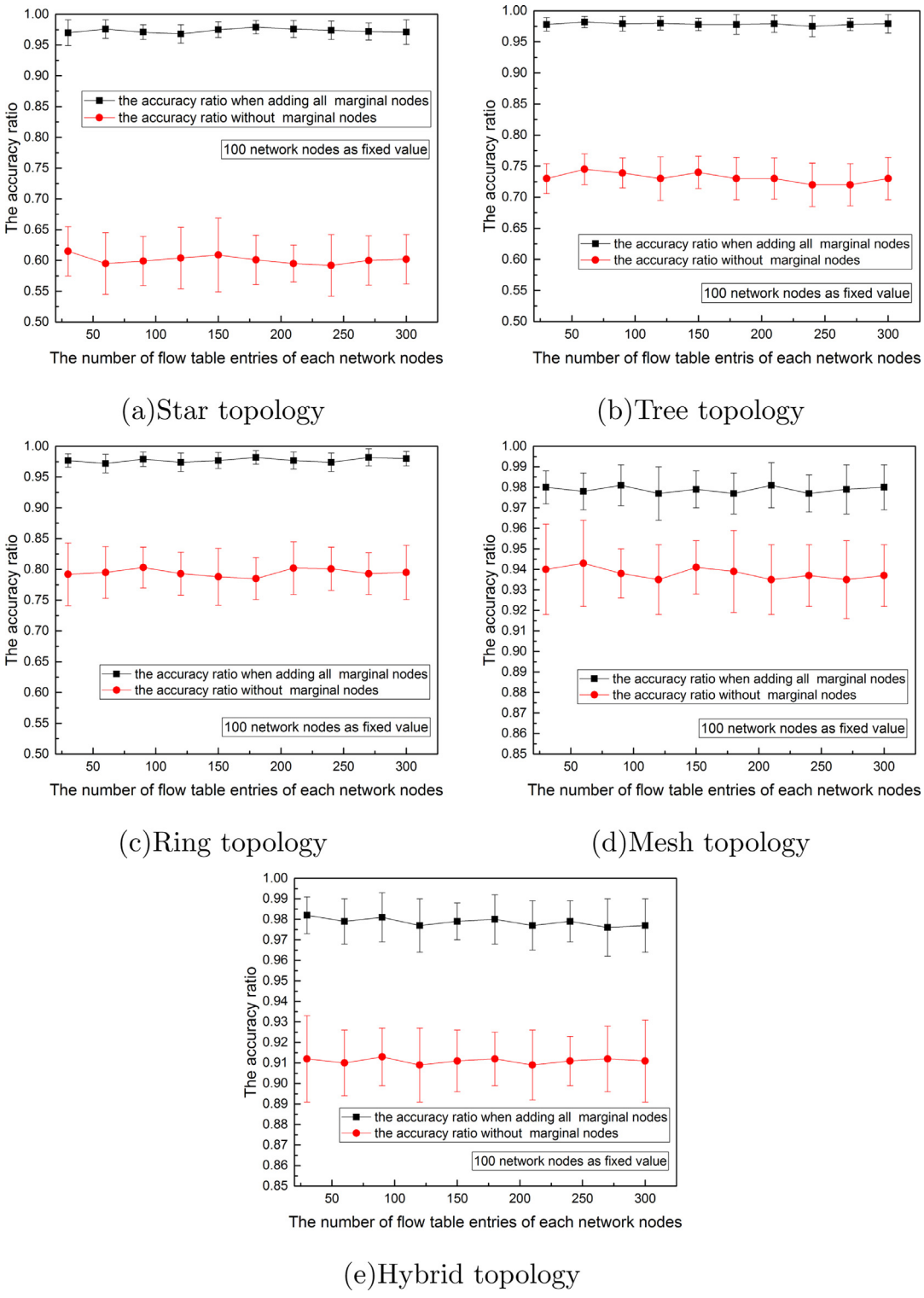


Fig. 9. The accuracy ratio of measured flows to the actual flows with expansion in network flows(note that the starting point of Y-axes in pictures (d) and (e) is 0.85 for better illustrating).

adapt to different scales of topologies and flows, so it boasts robustness.

6. Applications

This section mainly introduces the application developed from Global Flow Table. As is shown in the Fig. 10, the Global Flow Table seats in the application layer of SDN, and provides the global information of each network flow via its REST API for security applications. Python and JAVA were used to develop applications of GFT. In order to examine practicability of the applications, we utilized OpenvSwitch and Floodlight to simulate an SDN.

6.1. MITM detection application

The man in the middle attack(MITM) is a common and hardly detectable attack in network. Most of the current MITM detection

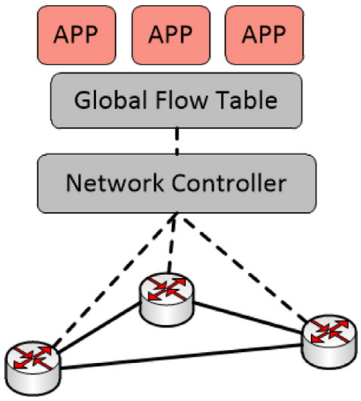


Fig. 10. GFT Architecture overview.

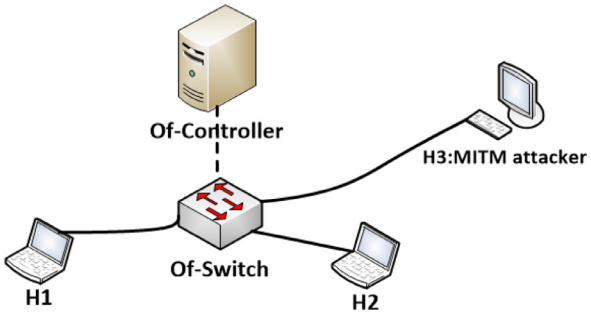


Fig. 11. Experimental topology of the MITM attack.

mechanism are inefficient and limited, such as [24] is limited to TCP connection and restricted to the chosen end-users but not the whole users in network [25]; [26] only apply to ARP based MITM attack. However, using the characteristics of network flow provided by GFT, the MITM can be easily detected. We developed an application to simply grasp this attack.

The fundamental principle is trying to find the abnormal nodes (except switch nodes) that are the source nodes of certain network flows as well as the destination nodes. The detection method is traversing the database of network flows, and searching for flows that the source MAC address or the destination MAC address of the flows is the same node. Judging whether the create time of these flows are similar. Concurrently, the communication time and traffic maintain beyond certain thresholds. Nodes with such characteristics are regarded as the MITM attackers. Once the abnormal nodes are detected, an alarm and essential information of the nodes will be sent out to the security operators.

To test the correctness and practicability of this application, we simulated a MITM attack, as shown in Fig. 11. H1 and H2 were treated as the victim hosts, while H3 which installed Ettercap, an MITM attack simulation software, was used to manufacture Bidirectional ARP deception to the victim hosts. As consequence, the attack was detected within 2 seconds, and an alarm with the information of the attackers' IP and MAC address as well as the location was set off.

However, it should be noted that the MITM Detection application is designed to better illustrate the significance of GFT, not for accurate detection. This application is suitable for the first line of MITM detection, because the detected attackers also might be network proxies or dedicated servers. Thus, next step is to differentiate MITM attackers with these specific network nodes. For example, it is useful to set white lists to avoid such misjudgment. In addition, the detection results can be further analyzed and validated by the method based on packet content detections.

To explicitly present the characteristics of network traffic, we used the Global Flow Graph application to retrieve network flows with certain fields. The Global Flow Graph application is able to provide the global traffic with paths of network flows. In addition, it also satisfies the enhancements of retrieving the specified flows according to the conditionally filter fields including source IP, destination IP, source MAC, destination MAC, source port, destination port, application layer protocols. From the Fig. 12, it is apparent that the times of flows sent to and from the MITM attacker are sufficiently close, and the traffic is aggregated. Meanwhile, using visualization functions provided by the Global Flow Path application this MITM attack can be emerged more intuitively, as is shown in the Fig. 13.

The above mentioned Global Flow Path application was developed to provide the global vision of data flows in the network. The path information of each packet in the network will be observable, and different colors are used to represent different network traffic levels. It is accessible to visualize the topology of current network, including the link connections of switches and hosts, the port numbers and other information. Simultaneously, network traffic will be displayed dynamically in the topology when there is traffic in the network. As shown in the Fig. 13, arrow with color indicates the path of flow. Each color represents different scales of network traffic. It indicates that the flow has exceeded the normal level when the color becomes red, and the flow will be marked as suspicious traffic. Moreover, to facilitate the analysis of the abnormal traffic, specified flow path can be visualized in the topology through the flow fields.

6.2. Global flow monitor

The global flow monitor application (GFM) is a live network security monitor, which allows network security operators to specify desired network behaviors in the form of invariants, and triggers alarms whenever a packet violates any invariant, such as the routing loops and Black Holes which are common problems in the network.

To check out a routing loop, the GFM examines the global paths of network flows to identify whether the flow has ever passed through the same network nodes. If it has, a loop-detected callback function will be invoked. Similarly, a black hole is automatically detected when there is no output port in the path of a flow. In this case, a black-hole-detected callback function is invoked.

Further, we applied GFT to detect anomaly traffic and attackers in SDN. GFT can provide the traffic and path of each network flow. Meanwhile, it also provides the function of flow retrieving. Specific information such as the source, destination IP address andMAC address is required to input and we can derive the details of a specific flow for further analysis of abnormal behavior in the network.

The experimental environment is a real SDN network in our laboratory, which consists of nine student-hosts and twelve server-hosts connected in this SDN network. We collected nearly ten thousands GFT records from 17:03:10 on November 28th 2015 to 22:53:31 on December 4th 2015. There are 283 source IP addresses and 311 destination IP addresses included in this data set. The data was imported to a software called WEAK and mined from different angles.

First, we analyzed the communication state between each IP. The result is shown in Fig. 14. X-axis represents the source IP address and Y-axis represents the destination IP address. If the points corresponding to an IP value of X-axis are dense in the figure, it indicates that the source IP is active and accesses a large number of external IP addresses. Equally, if the points corresponding to an IP value of Y-axis are dense, it shows that the IP is frequently accessed by a large number of other IP addresses. Thus, it's obvious to find out the active users over a period from the figure. There

Fig. 12. Network flows whose source MAC or destination MAC addresses are from the attacker shown by the global flow graph application.

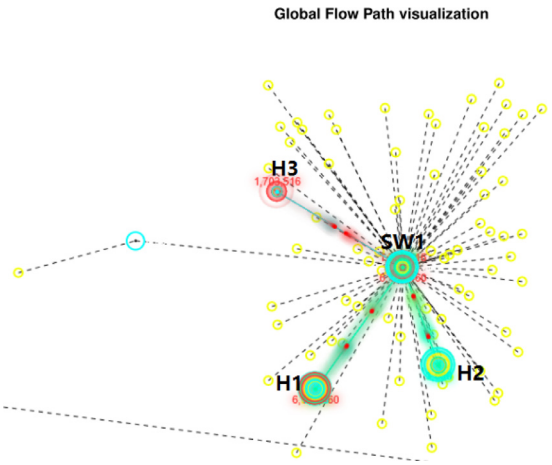


Fig. 13. The visualization emerged by global flow path application.

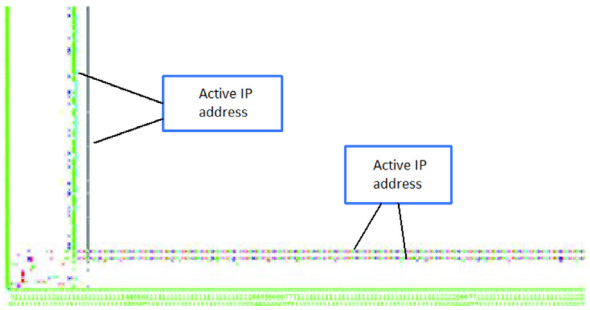


Fig. 14. The communication state between each IP.

are four hosts accessing many other destinations frequently. Besides, they got these external networks' response. That is to say, they are also accessed by other IP addresses frequently. The statistics of communication state between users' IP can be used as the basis for detecting user abnormal behavior.

In order to further determine whether these active users are inoffensive to the network, we can analyze from several aspects such as the view of these active IP traffic. The relationship between source IP and the byte counts of the flows during a period is shown in the Fig. 15. The X-axis is expressed as the source IP ad-

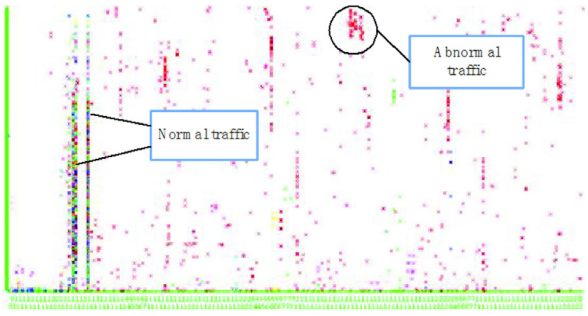


Fig. 15. The relationship between source IP and the byte counts of its flows during a period.

Fig. 16. The communication between source IP and the port number that it scans.

dresses while the Y-axis is expressed as byte counts of the IP flows. It is shown that the byte counts of these source IP addresses keep a high level from the part which is circled in the graph. A normal active IP address should has all levels of byte counts like the IP addresses which are marked in left of the graph. So it is likely that these abnormal IP addresses are transporting big files within a short time.

Nevertheless, abnormal behaviors may not be detected just through byte counts or packet counts, such as port scan. Hence, analyzing the IP and its related port numbers is needed. The communications between source IP and the port numbers that it scans are shown in Fig. 16. X-axis represents the source IP address and Y-axis represents the port numbers of the destination. If the corresponding points of an IP address are distributed densely, we can conclude the host of the IP was scanning a large number of ports, which is usually the precursor of a network attack. It is revealed that there are two hosts scanning all other ports from the statistic results, which obviously is not the normal access. To comprehend which IP is scanned, just view the destination IP and its port numbers.

Via GFT, it is effortless to get the statistics of all IP activities within a day, so as to analyze the user's Internet behaviors. In the statistics of Fig. 17, where X-axis is the time and Y-axis is IP source address, the corresponding point distribution of Y-axis represents the time of users accessing the Internet. The figure shows the three most active IP addresses. One IP started from 6:50 and took a break in the middle. Then he used the internet to work from 9:11 am to 5:45pm. Another IP (111.0.0.119) had been on the Internet from 9:11am to 12:55am, and then he intermittently accessed the network until 2:00 pm. Time analysis facilitates the analysis of network flow correlation.

In general, the common flow data tends to microcosmic analysis. It pays more attention to the packets between IP addresses every second. So it is in the need for real-time detection or response with attacks such as DDOS. However, the GFT is more inclined to the macro analysis. It reflects the global distribution of the flows and the packets transportation during a period (hours). So it's easier to analyze the user network behaviors and find out the slow and inconspicuous attacks. For example, it has a great advantage in port scanning.

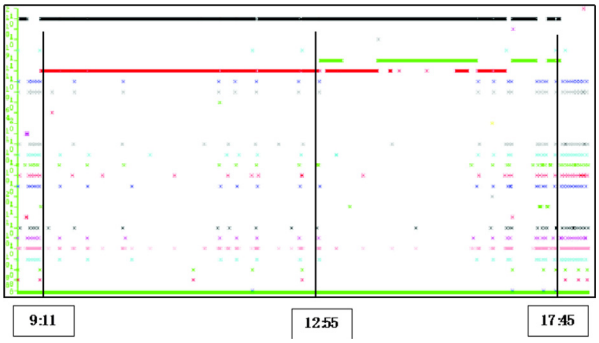


Fig. 17. The time distribution of the users on the internet.

Besides, GFT records the global path and traffic volume of each network flow, so the use of network resources can be observed from the perspective of the whole network. Therefore, dynamic routing strategies can be issued according to the current situation of network resource utilization to fully route network resources.

7. Conclusion

In SDN, even if network flows can be dispatched and controlled globally by network controller, there is no such a mechanism that provides clear visibility of the distribution, path and traffic of all the network flows for the security appliances. In this paper, we proposed a new mechanism named GFT and corresponding algorithms to realize global flow collection, computation of all the paths in the whole network and global flow storing. This method provides an extensive variety of information of flows in the current network such as the passing path of each flow with the matching entry of each flow as the identity. In order to reduce the network load posed by network statistics gathering, we use the Weak Vertex Cover problem to find the best and smallest set of measured switches instead of gathering flow tables of all the switches in the network. We also evaluated time and accuracy performances of the GFT algorithm. The result shows the algorithm is feasible and has a good time and accuracy performance. GFT is applied to originally enhance the security of the SDN network, especially in cloud computing center, campus network and enterprise network. At last, we presented applications built upon GFT and applied them to the analysis of network behavior and abnormal detection by methods of data mining.

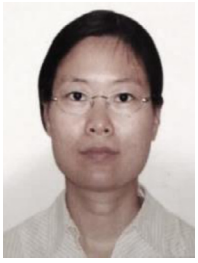
Our next step is to adapt GFT to various special situations of network such as multicast and NAT. Moreover, the GFT is getting ready to be deployed in real corporation networks presently. We will further develop more security applications based on GFT to make the security operation be more automatic and intelligent. In addition, we expect to establish a multilevel global flow state table based on security parameters to support a security mechanism of global vision and make the distributed security devices work together.

Acknowledgement

This research was supported by National High Technology Research and Development Program of China(863 Program)(No.2015AA016003) and Beijing Laboratory of Advanced Information Networks.

References

[1] Chung-Shen, L.L. Wanjiun, Software defined networks [guest editorial], in: Communications Magazine, 51(2), IEEE, 2013, p. 113.
[2] Z. QY, M. Chen, Z. GS, X. CY, Z. GM, J. PC, Openflow-based SDN technologies[j], RuanjianXuebao /Journal of Software (2013) 1078–1097.
[3] OpenFlowSpecication. http://www.openflow.org/wp/docum.
[4] S. Bian, P. Zhang, Z. Yan, A survey on software-defined networking security[c]// eai international conference on mobile multimedia communications, in: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, pp. 190–198.
[5] Y.A. Khimabhai, V. Rohokale, SDN control plane security in cloud computing against DDos attack[c]The International Conference, 2016, 1–5
[6] B. Claise, RFC 5101: Specification of the IP flow information export (IPFIX) protocol for the exchange of IP traffic flow information, http://www.tools.ietf.org/html/rfc5101.
[7] P. Phaal, sFlow version 5. http://www.sflow.org/sflow_version_5.txt.
[8] Endace Inc. http://www.endace.com/.
[9] Gigamon. http://www.gigamon.com/.
[10] Net optics: Architecting visibility into your network http://www.netoptics.com/.
[11] Splunk: Operational Intelligence, Log Management, Application Management, Enterprise Security and Compliance. http://www.splunk.com.
[12] K.S. Sahoo, S. Mohanty, M. Tiwary, et al., A comprehensive tutorial on software defined network: the driving force for the future internet technology[c], in: International Conference on Advances in Information Communication Technology & Computing, ACM, 2016, p. 114.
[13] A.R. Curtis, J.C. Mogul, J. Tourrilhes, et al., Devoflow: scaling flow management for high-performance networks[c], ACM SIGCOMM Comput. Commun. Rev. ACM 41 (4) (2011) 254–265.
[14] F.C. Gomes, C.N. Meneses, P.M. Pardalos, et al., Experimental analysis of approximation algorithms for the vertex cover and set covering problems[j]., Comput. Oper. Res. 33 (12) (2006) 3520–3534.
[15] A. Khurshid, X. Zou, W. Zhou, M. Caesar, P.B. Godfrey, Veriflow: Verifying Network-Wide Invariants in Real Time, NSDI (2013).
[16] N. Handigol, B. Heller, V. Jeyakumar, D. Mazires, N. McKeown, Where is the debugger for my software-defined network? in hotSDN, 2012,
[17] N. Handigol, B. Heller, V. Jeyakumar, D. Mazieres, N. McKeown, I know what your packet did last hop: Using packet histories to troubleshoot networks., NSDI (2014).
[18] P. Kazemian, M. Chang, H. Zeng, et al., Real time network policy checking using header space analysis, Usenix Conference on Networked Systems Design and Implementation (2013) 99–112.
[19] Y. Breitbart, C.Y. Chan, M. Garofalakis, et al., Efficiently Monitoring Bandwidth and Latency in IP Networks[c], in: INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 2, IEEE, 2001, pp. 933–942.
[20] D.B. West, Introduction to Graph Theory[M], Prentice Hall, Upper Saddle River, 2001.
[21] X.H. Liu, J.P. Yin, L.L. Tang, et al., Analysis of efficient monitoring method for the network flow[j], J. Software 14 (2) (2003) 300–304.
[22] R.C. Chen, P.G. Jessel, R.A. Patterson, MININET: A microprocessor-controlled “mininetwork”[j], Proc. IEEE 64 (6) (1976) 988–993.
[23] B. Switch, Developing floodlight modules. Floodlight, 2012, http://www.projectfloodlight.org/floodlight/.
[24] V.A. Vallivaara, M. Sailio, K. Halunen, Detecting man-in-the-middle attacks on non-mobile systems[c], in: ACM Conference on Data and Application Security and Privacy, 2014, pp. 131–134.
[25] J.H. Cox, R.J. Clark, H.L. Owen, Leveraging SDN for ARP security[c], in: Southeastcon, IEEE, 2016, pp. 1–8.
[26] S. Hong, L. Xu, H. Wang, et al., Poisoning network visibility in software-defined networks: new attacks and countermeasures[c], Network and Distributed System Security Symposium, 2015.



Xiaofeng Qiu is an associate professor in Beijing University of Posts and Telecommunication since 1994, working on network security and data mining. She has invented 8 patents, and published dozens of articles and books. Also, she has led her team accomplish a number of national and enterprise projects.

Kai Zhang received the bachelor's degree in communication engineering from Chongqing University in 2015. Now he is studying on network security, and Software Defined Networking in Beijing University of Posts and Telecommunication.

Qiuzheng Ren received the master degree in information and communication engineer ing from Beijing University of Posts and Telecommunication in 2016. Now she is wor king in the Lenovo Research&Technology as the software defined networking researcher.