An SDN based hopping multicast communication against DoS attack

Zheng Zhao, Fenlin Liu and Daofu Gong
Zhengzhou Science and Technology Institute
Zhengzhou, Henan 450001 - China
[e-mail: diyigemsn@hotmail.com, liufenlin@vip.sina.com, gongdf@ailiyun.cn]
Zhengzhou, Henan 450001 - China
*Corresponding author: Fenlin Liu

Received September 20, 2016; revised December 14, 2016; accepted January 3, 2017; published April 30, 2017

Abstract

Multicast communication has been widely used in the Internet. However, multicast communication is vulnerable to DoS attack due to static router configuration. In this paper, HMC, a hopping multicast communication method based on SDN, is proposed to tackle this problem. HMC changes the multicast tree periodically and makes it difficult for the attackers to launch an accurate attack. It also decreases the probability of multicast communication being attacked by DoS and in the meanwhile, the QoS constrains are not violated. In this research, the routing problem of HMC is proven to be NP-complete and a heuristic algorithm is proposed to solve it. Experiments show that HMC has the ability to resist DoS attack on multicast route effectively. Theoretically, the multicast compromised probability can drop more than 0.6 when HMC is adopt. In addition, experiments demonstrate that HMC achieves shorter average multicast delay and better robustness compared with traditional method, and more importantly, it better defends DoS attack.

Keywords: Software defined network (SDN), Moving target defense (MTD), Multicast tree, Minimum hotness tree, Game theory

This work is supported by the National Natural Science Foundation of China (No. 61401512, 61379151), Fundamental Research Funds for the Central Universities (No. 2016TJJBKY021). National Natural Science Foundation of China (No. 61572519), National Basic Research Program of China (973) (Grant No. 2012CB315901, 2013CB329104), National Natural Science Foundation of China (Grant No. 61309019, 61372121).

https://doi.org/10.3837/tiis.2017.04.020 **ISSN : 1976-7277**

1. Introduction

Multicast is designed to deliver data to multicast destination hosts and is widely used in IPTV, real-time data transmission, multi-media conferences etc. Compared with unicast, it greatly improves the efficiency of data transmission and utilization of network resource. However, multicast communication also faces serious security problems. Higher data transmission makes multicast communication more vulnerable to DoS attack on route [1, 2]. In this kind of DoS attack, attackers choose target nodes or links to flood the packets and congest, thus the communication will be interrupted.

Congestion detection is a traditional method to defense DoS attack on route. If a DoS attack is detected, the compromised nodes or links will be identified [3], and route will be adjusted to avoid attack [4, 5], or the attack traffic will be restricted [6]. However, these methods mitigate DoS attack after the communication has been seriously affected and will become ineffective if the attacker changes target nodes or links frequently. In recent years, moving target defense (MTD) [7-10] is proposed to enhance communication security using dynamicity and randomness. With MTD, the configuration of network will change to achieve attack avoidance or defense. In multicast communication, if the defender changes multicast route in real time, the current route will be difficult to be found by the attacker. Thus secure multicast communication could be achieved since the attacker cannot launch an exact attack. However, dynamic network configuration depending on routing protocols will cause service interruption or route expansion due to distributed route management in traditional IP network. Emerging software-defined network (SDN) [11] brings new possibilities to dynamic network configuration. SDN decouples the control plane and the data plane, and applies logic centralized control. The powerful network management and control ability of SDN achieve the flexibility of dynamic network configuration.

To solve the problem brought by the hysteretic and static nature of traditional defense methods against DoS, in this paper, an SDN based hopping multicast communication (HMC) is proposed using the idea of MTD. In this method, instead of using passive detection, a proactive defense approach is adopted to avoid delayed feedback. In addition, the multicast tree changes dynamically with time to increase the difficulty of launching DoS attack. The major contribution of our work includes:

- A method of hopping multicast tree is proposed and built as a HMC model. We prove that the routing problem of HMC is NP-complete.
- A heuristic algorithm for building hopping multicast tree is proposed.
- The attacker and defender are analyzed using game theory in order to determine the hopping period by which the defense benefit can be maximized.
- We conduct a series of experiments to validate the effectiveness, performance and robustness of HMC. Results show that the present approach can resist DoS attack in multicast route effectively.

The rest of the paper is organized as follows. In section 2, related work is discussed. Section 3 describes the DoS attack model as well as the defense model of HMC. In Section 4, a method to determine the multicast tree hopping period is proposed. Section 5 presents basic prototype system of HMC and simulation experiments. In addition, the security, performance and robustness of HMC are analyzed. Section 7 concludes the paper.

2. Related work

Efforts have been taken by researchers on multicast communication in SDN. Aakash et al. proposed Avalanche [12] which enables multicast in data centers on commodity switches and minimizes the size of the routing tree for the multicast group. Based on SDN, Alexander et al. [13] designed a controller framework for multicast communication, which can balance the traffic load of multicast. Zhang et al. [14] presented a multicast mechanism based on network function virtualization, in which flows of multicast are processed by NFV. In addition, a routing algorithm to build an appropriate multicast topology was proposed. Shen et al. [15] presented a reliable multicast tree (RST) for SDN. They proved that RST problem is NP-hard and proposed Aware Edge Reduction Algorithm (RAERA) to solve it. These methods mentioned above achieve multicast communication with SDN, but the security of multicast communication is not considered. Aiming at the problem that multicast communication is vulnerable to DoS attacks, HMC is proposed in this paper to protect the multicast flows by hopping the multicast route.

The further deployment of multicast is limited by its security risks. Researchers attempt to improve the security and reliability of multicast in SDN. Zou et al. [16] proposed a multicast scheme based on SDN, which can improve the security and controllability of multicast communication. Multicast events management, multicast tree calculation and user authentication are achieved on controller, which can effectively prevent the illegal users from joining multicast group. Thomas et al. [17] proposed a robust multicast method based on SDN framework, which utilizes fast-failover to achieve one-link fault tolerance with limited packet loss. However, these work mentioned above cannot defense against DoS attack. The most similar work with HMC is RTM [18], which can defense against sniffer and DoS attack using moving target defense (MTD). They constructed a random multicast tree in network topology which can be modified with time. An extend Dijkstra's shortest path algorithm is used for generating multicast tree and the edge weights are updated with time to generate different multicast trees. The attack cost is increased and the security of multicast communication is enhanced. However, the deployment cost of this method is high because the terminals need to be modified. Moreover, the QoS and resource constraints are not considered in RTM.

MTD has higher demand for networks. In traditional network, it is difficult to achieve quick cooperation for distributed route management. However, SDN can be used to achieve MTD more easily due to its centralized control and programmability. An active random route mutation (RRM) method [19] was proposed by Qi Duan et al. and was applied in SDN environment. Routes of multiple flows in the network are randomly changed simultaneously to defense against eavesdropping and DoS attacks. Jafar et al. [20] established a game of attacker-defender for RRM and the defender's benefit is maximized. The route selection is modeled as a constraint satisfaction optimization problem and the practical routes are obtained using Satisfiability Modulo Theories (SMT). Based on the centralized control and programmability of SDN, DHC was proposed in literature [21], which can resist sniffer attack by changing multiple network configuration periodically. The above methods apply the idea of MTD in unicast communication. However, HMC, proposed in this paper, takes advantage of MTD to defense against DoS attack effectively on multicast communication.

3. Model construction

This research focuses on the DoS attack which floods packets to the nodes on multicast route. In traditional multicast, a static multicast tree is built in the network. Since the multicast tree is

static, attackers can find the nodes on the multicast tree accurately after limited number of trials and explorations. If the multicast tree is dynamic, it will increase the difficulty for attackers to find current active multicast tree and launch an attack. In this section, a model is constructed to simulate the behavior of the attacker. Then a formalized model of MHC is built and a heuristic algorithm is given. At last, an update method of multicast tree based on SDN is proposed.

3.1 Attacker model

We classify DoS attackers into two categories based on the behavior of attacks: static DoS attacker and dynamic DoS attacker.

(1) The static DoS attacker randomly selects a target node set in the network to launch an attack and the node set to be attacked does not change during the attack.

(2) The dynamic DoS attacker selects a target node set in the network to attack as well. If the DoS attack is successful, the attacker will keep the node set, otherwise the attacker will try to attack other nodes.

For multicast communication, dynamic attackers are more challenging to tackle. They can adjust their targets with the feedback of attacks. In this case, if multicast tree is static, attackers will find the multicast tree gradually as time goes on and attack the multicast communication effectively. We focus on stopping dynamic attackers and defending multicast communication through hopping multicast tree.

3.2 HMC model

3.2.1 Definition

In HMC, the multicast tree hops periodically for changing multicast route. Thus, it is difficult for the attacker to launch an accurate attack. Different multicast trees are generated by HMC to avoid long time stay for a certain node in multicast tree. A multicast group is defined by a 2-tuple $g=(s,D)$ where s is the multicast source and D is the destination node set. For a multicast group g, the hotness is defined for each forwarding node in the network to evaluate the frequentness that the node is selected into multicast tree.

Definition 1: Given a network G and a multicast group g, the hotness of $v \in G$ is defined as h_v.

The hotness of a path in network can be defined as the sum of hotness of all nodes on the path. With the Definition 1, we can redefine the network G for multicast group g.

Definition 2: A network can be defined as $G=(V,E,H)$, where V is the node set in the network, E is the edge set and H is the hotness set of nodes. For $\forall v \in V$, there is $\exists h_v \in H$.

For network G and multicast group g, a multicast tree T_g is built in each hopping period. We defined the hotness of T_g as follows.

Definition 3: For a multicast tree T_g, $Hotness(T_g)=\sum_{v \in T_g} h_v$ is defined as the hotness of T_g.

Lower value of $Hotness(T_g)$ indicates lower usage frequency of the nodes selected into T_g. The hotness of each node is updated with the hopping of multicast tree. The update function of a node is defined as follows.

Definition 4: The update function of a node is defined as $h_v^i = \varphi(h_v^{i-1})$ ($i \geq 1$), in which h_v^{i-1} and h_v^i indicates the hotnesses of node v in $i-1$th and i th hopping period, respectively.

3.2.2 Minimum hotness tree

Multicast is often used in real-time communication, such as IPTV and multimedia conference, therefore the QoS of communication is demanded. The communication delay and bandwidth are the key factors that influence the QoS. Hence, the construction of multicast tree is constrained by these two factors in HMC.

For $\forall e \in E$, the delay on edge e is defined as $Delay(e): E \to R^+$. For $\forall v \in V$, the bandwidth load capacity of v is defined as $L(v): V \to R^+$. For a multicast tree T_g, the path from multicast source s to the destination node $d \in D$ is marked as $P_{T_g}(s,d)$. The following formulas hold.

$$Hotness(T_g) = \sum_{v \in T_g} h_v \tag{1}$$

$$\text{For } \forall d \in D, \quad Delay\left(P_{T_g}(s,d)\right) = \sum_{e \in P_{T_g}(s,d)} Delay(e) \tag{2}$$

$$Bandwidth(T_g) = \min\left\{Bandwidth(v) \mid v \in T_g\right\} \tag{3}$$

Eq. (1) describes the definition of hotness of multicast tree. Eq. (2) calculates the delay from the multicast source node to a destination node. Eq. (3) defines the maximum bandwidth which can be forwarded by the multicast tree. If a node is present in a multicast tree for a long time, the multicast tree will be more likely to be exposed. To avoid long time stay of the nodes in the multicast tree, the hotness of multicast tree should be as small as possible. Considering the QoS constraints of multicast communication, the constrained minimum hotness tree is defined as follows.

Definition 5: Given the network $G = (V, E, H)$, multicast group $g(s, D)$, delay upper bound Δ_d and bandwidth lower bound Δ_l, T_g is called the constrained minimum hotness tree of $g(s, D)$, if T_g covers $\{s\} \cup D$ and satisfies the constraints (4) ~ (6). Solving T_g is considered as a constrained minimum hotness tree problem.

$$\mathcal{H}(T_g) = \min Hotness(T_g) \tag{4}$$

$$\text{For } \forall d \in D, \ Delay\left(P_{T_g}(s,d)\right) \leq \Delta_d \tag{5}$$

$$\text{For } \forall d \in T_g, \ Load(T_g) \geq \Delta_l \tag{6}$$

To bring randomness to the multicast tree, the hotness of each node is updated with randomness using hotness update function, which is shown in Eq. (7). If a node v with hotness h_v is selected into the multicast tree in one hopping period, its hotness will be updated as $\varphi(h_v) = h_v \times \gamma$, $\gamma > 1$. In other word, the hotness of v will be increased. If v is not selected

into the multicast tree, its hotness is updated as $\varphi(h_v) = h_v \times \lambda$, $0 < \lambda < 1$, and the hotness of v will decrease.

$$\varphi(h_v) = \begin{cases} h_v \times \gamma \quad (\gamma > 1) & v \text{ is selected in this hopping period} \\ h_v \times \lambda \text{ with probability } P\ (0 < \lambda < 1) & \text{otherwise} \end{cases} \tag{7}$$

It can be proved that the constrained minimum hotness tree problem is NP-complete (shown in appendix I). Therefore, there is no algorithm can solve this problem in polynomial time. To solve this intractable problem in reasonable time, we proposed a heuristic algorithm CMHT (Constrained Minimum Hotness Tree). Inspired by KPP algorithm [22], the constrained minimum hotness tree is solved using the strategy of greedy. KPP algorithm is used to solve constrained Steiner tree [23]. Both of Steiner tree and minimum hotness tree cover a certain node set in the graph. However, Steiner tree is a tree with minimum weight sum of edges, while minimum hotness tree is a tree with minimum hotness sum of nodes. Therefore, we convert the hotness of nodes to the hotness of edges and take the hotnesses of edges as the edge weights. Then we can solve the constrained minimum hotness tree problem using KPP algorithm. If there exists an edge $e_{vv'} \in G$ connecting v and v' with hotness h_v and $h_{v'}$, the hotness of $e_{vv'}$ can be obtained by Eq. (8).

$$h_{e_{vv'}} = (h_v + h_{v'})/2 \tag{8}$$

Algorithm 1. The heuristic algorithm of constrained minimum hotness tree problem

Input:

$G = (V, E, H)$ describes the network topology

$g = (s, D)$ describes the multicast group

Δ_d denotes delay constraint

Δ_l denotes bandwidth constraint

Output:

T_g as a constrained multicast tree

CMHT (G, g, Δ_d, Δ_l)

(1) Compute h_e for each $e \in G$ using the hotnesses of the nodes at each end of e

(2) Delete the nodes from G that have a bandwidth load capacity $< \Delta_l$

(3) For each $s_1, s_2 \in g$, computes the minimum hotness path satisfying $Delay(P_T(s_1, s_2)) \le \Delta_d$ using Yen algorithm

(4) Construct the closure group G' for node set $\{s\} \cup D$ with the paths obtained from step (3)

(5) Construct a delay constrained minimum hotness spanning tree of G' using Prim algorithm

(6) Expand the edges of constrained minimum hotness spanning tree into the constrained minimum hotness path in G

(7) Remove any loops caused by step (6) to generate T_g

(8) For $v \in G$, update bandwidth load capacity and hotness

(9) Return T_g

Known from Eq. (8), the larger hotness of v is, the larger hotnesses of the edges connecting it will be. Given the hotnesses of edges, CMHT can be achieved, as shown in **Algorithm 1**. Step (1): the hotnesses of nodes are converted to the hotnesses of edges. Step (2): the nodes with bandwidth load capacity less than Δ_l are deleted. Step (3): the minimum hotness paths between any two nodes in g satisfying Δ_d are calculated using the k-shortest algorithm Yen [24]. Step (4): a closure graph G' of node set $\{s\} \bigcup D$ is constructed based on the paths obtained in step (3). Step (5): the minimum spanning tree of G' satisfying constrains is constructed using Prim algorithm [25]. Step (6,7): the edges of this spanning tree are expanded into the constrained minimum hotness paths in G. If loops exist in the outcome, delete relevant edges to remove loops then T_g is generated. Step (8,9): the bandwidth load capacity and hotnesses of nodes are updated and the constrained minimum hotness tree is returned.

It should be pointed out that the ability of CMHT to randomly generate multiple multicast tree is to some extent dependent on the connectivity of network. More specifically, better network connectivity renders higher performance, as multiple paths exist between two nodes in the network, which can be utilized to construct multiple multicast trees for a multicast group. Therefore, good network connectivity is favored if you would like to implement CMHT. However, we do not consider this as a flaw of this method since networks in reality often have a connectivity that is good enough to be used to improve communication performance and security[19-21, 26, 27] [28, 29].

3.3 Multicast tree update method

In HMC, when the multicast tree hops, the flow entries on switches need to be updated. Moreover, it should be guaranteed that the flow entries update is consistent and no packet is lost. To meet these requirements, flow labeling method [30] is used for updating multicast tree. For the multicast group $g(s,D)$, T_g^o denotes the old multicast tree and T_g^n denotes the new multicast tree in network. The multicast tree is updated by the following steps.

1) Add flow entries to deploy the new multicast tree in the network except the multicast source node, i.e. deploying $T_g^n - \{s\}$. These flow entries forward flow labeled with $\mathcal{L}$.
2) Modify the flow entry on the multicast source node s. The new flow entry labels multicast flow with $\mathcal{L}$. The new multicast tree T_g^n is constructed.
3) Wait for the maximum delay in the network.
4) Delete the flow entries of $T_g^o - \{s\}$.
5) Delete the label $\mathcal{L}$ in flow entries in T_g^n from leaves node to source node layer by layer.

The multicast tree update method described above guarantees that the multicast traffic can be routed by the old flow entries during updating without packet loss. On the other hand, traffic is routed by the new flow entries after updating, which maintains per-packet consistency.

4. The determination of hopping period

The key to hopping multicast communication is how to decide the hopping period in order to maximize the defense benefit. Intuitively, if the multicast tree hops with higher frequency, the

communication would be more secure, thus higher defense benefit can be obtained. However, with the increase of hopping frequency, the cost of route calculation and flow entry setup will increase correspondingly. Therefore, there is a tradeoff between defense benefit and cost when selecting the hopping period of multicast tree.

Due to limited resource, the attacker can only attack limited nodes in the network. In the meanwhile, the defender can detect limited compromised nodes because of limited detection ability. The behaviors of the attacker and defender are described as follows.

The behavior of attacker: the attacker selects r nodes in the network to attack within hopping period T_a. If the multicast flow is compromised, the attacker keeps attacking these nodes until the multicast tree is hopped. Otherwise the attacker will change the nodes to attack, i.e., the hopping attack.

The behavior of defender: the defender updates the multicast tree with constrained minimum hotness tree within the period of T_d. The defender can detect the compromised nodes with speed of f. These compromised nodes will be set to larger hotnesses so that they will not appear in the multicast tree, until the attack to these nodes stops.

Intuitively, if the attack hops more frequently, more traffic will be compromised. While if the multicast tree hops more frequently, the time during which compromised nodes are present in the multicast tree will be shorter. Assuming that both the attacker and defender have complete information of each other, their strategies can be defined as a static game of complete information. Both players determine their strategies based on the Nash Equilibrium of this game.

The game is defined as $\Gamma = \{I, S, U\}$, where $I = \{a, d\}$ is the player set. a denotes the attacker and d denotes the defender. $S = \{T_a, T_d\}$ is the strategy set of attacker and defender, i.e. the attack period and defense period (hopping period of multicast tree). $U = \{u_a, u_d\}$ is the utility function set of attacker and defender. T_c denotes the communication time of multicast. We define CT as the proportion of compromised traffic for evaluating the security of HMC.

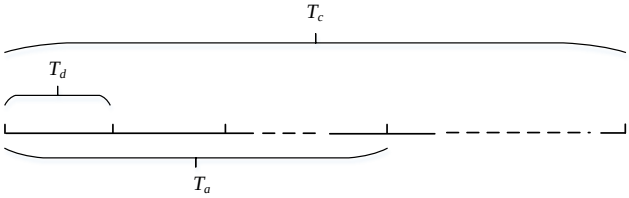


Fig. 1. The case of $T_a \geq T_d$

1) The case of $T_a \geq T_d$

The case of $T_a \geq T_d$ is shown in **Fig. 1**. Let $z = \lceil T_a / T_d \rceil$, i.e. T_a contains z T_d. Because the attack does not hop over a single period T_a, the number of compromised nodes that can be detected by the defender during hth defense period is $R(h)$, as shown in Eq. (9) ($1 \leq h \leq z$).

$$R(h) = f \times T_d \times (h-1) \tag{9}$$

The defender can get rid of $R(h)$ compromised nodes during h defense periods. Assuming that there are n nodes in network, the probability of each node compromised being $x(h)$, as shown in Eq. (10).

$$x(h)=\frac{r-R(h)}{n-R(h)} \tag{10}$$

The multicast communication fails if there exists at least one node on the multicast tree that is compromised. Assuming that there are at most L nodes on the multicast tree and p_i is the probability that there are i nodes in the multicast tree, then the probability of the multicast tree being compromised during hth defense period is $X(h)$, as shown in Eq. (11).

$$X(h)=\sum_{i\in[1,L]} p_i\left(1-\left(1-x(h)\right)^i\right) \tag{11}$$

The probability of a successful attack in hth T_d over a single period T_a is $X(h)$. If the attack is successful, the proportion of compromised traffic is T_d/T_c. Therefore, the expectation of compromised traffic in T_a is $\sum_{h\in[1,z]} X(h)T_d/T_c$. Assuming that the compromised traffic in each T_a is independent to each other, the compromised traffic obeys binomial distribution $B\left(\lceil T_c/T_a\rceil, \sum_{h\in[1,z]} X(h)T_d/T_c\right)$. Therefore, the expectation of the traffic that the attacker can compromise in the multicast communication can be obtained by Eq. (12).

$$CT(T_a,T_d)=\lceil T_c/T_a\rceil\times\sum_{h\in[1,z]} X(h)T_d/T_c \tag{12}$$

2) The case of $T_a<T_d$

The case $T_a<T_d$ is shown in **Fig. 2**. Assume that $z'=\lceil T_d/T_a\rceil$, i.e. there are z' attack periods in one defend period. The attacker selects r nodes to launch the attack during each attack period. The multicast tree will stay unchanged during z' attack periods because the attack hops more frequently than the defense. If the attacker compromises the multicast communication successfully in a certain T_a, the target node set will stay unchanged, until the multicast tree hops. If the attack fails, the attacker will reselect the target nodes. Those attack-fail nodes will not be selected, until the multicast tree hops.

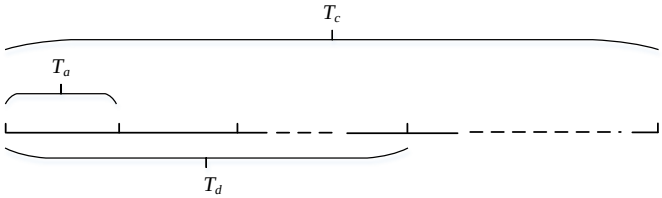


Fig. 2. The case of $T_a<T_d$

Within one T_d, the attacker will preclude $r(k-1)$ nodes during k th attack period, if the attack fails during first $k-1$ defense periods. Then in the remaining $n-r(k-1)$ selectable nodes, the probability of each node being selected by the attacker is shown in Eq. (13).

$$y(k)=\frac{r}{n-r(k-1)} \tag{13}$$

The probability that the multicast tree is compromised during k th attack period is $\sum_{i\in[1,L]} p_i\left(1-\left(1-y(k)\right)^i\right)$. If the attack fails during first $k-1$ attack periods of one defense period and succeeds during k th attack period, the probability is shown in Eq. (14).

$$Y(k)=\prod_{j\in[1,k-1]}\left(1-\sum_{i\in[1,L]} p_i\left(1-\left(1-y(j)\right)^i\right)\right)\times\sum_{i\in[1,L]} p_i\left(1-\left(1-y(k)\right)^i\right) \tag{14}$$

If the attacker compromises the multicast traffic successfully during k th attack period, the compromised traffic will be $(z'-k+1)T_a/T_c$ during this defense period T_d and the expectation of compromised multicast traffic in T_d will be $\sum_{k\in[1,z']} Y(k)(z'-k+1)T_a/T_c$. Suppose that the compromised traffic is independent within each T_d. Thus, the compromised traffic in the multicast communication obeys binomial distribution $B\left(\lceil T_c/T_d\rceil, \sum_{k\in[1,z']} Y(k)(z'-k+1)T_a/T_c\right)$. Therefore, the compromised traffic in the multicast communication is shown in Eq. (15).

$$CT(T_a,T_d)=\lceil T_c/T_d\rceil\times\sum_{k\in[1,z']} Y(k)(z'-k+1)T_a/T_c \tag{15}$$

Combining Eq. (12) and Eq. (15), the expectation of compromised traffic can be obtained, as shown in Eq. (16). If the periods of attack and defense are the same, and $T_a=T_d=\frac{1}{\theta}T_c$ ($\theta\in\mathbb{Z}^+$), we can get $CT(T_a,T_d)=\theta\times X(1)\times 1/\theta=X(1)$.

$$CT(T_a,T_d)\begin{cases}\lceil T_c/T_a\rceil\times\sum_{h\in[1,z]} X(h)T_d/T_c & T_a\geq T_d\\ \lceil T_c/T_d\rceil\times\sum_{k\in[1,z']} Y(k)(z'-k+1)T_a/T_c & T_a<T_d\end{cases} \tag{16}$$

The utility function of the defender u_d is related to the protected traffic and cost of defense. If the strategies of attacker and defender are T_a and T_d, respectively, the traffic that defender can protect is $1-CT(T_a,T_d)$. The cost of defender consists of two parts: cost of hopping multicast tree calculation on the control plane and the cost of flow entries setup on the data plane. If the multicast tree hops more frequently, the cost of defender will increase. The utility function of the defender is shown in Eq. (17), where B_d and C_d denote the benefit function and cost function of defender, respectively. For the attacker, the utility function is related to the compromised traffic $CT(T_a,T_d)$ and the cost of attack. If the attack hops more frequently, it is more likely to be detected. Moreover, the attacker will need more resource to probe the nodes in network, causing higher attack cost. The utility function of the attacker is shown in Eq. (18), where B_a and C_a denote the benefit function and cost function of the attacker, respectively.

$$u_d(T_a,T_d)=B_d\left(1-CT(T_a,T_d)\right)-C_d(T_d) \tag{17}$$

$$u_a(T_a,T_d)=B_a\left(CT(T_a,T_d)\right)-C_a(T_a) \tag{18}$$

The utility functions of the attacker and defender change with the state of network and strategies of both players are determined based on the utility functions. Under the assumption that both players have the complete information of each other, Nash Equilibrium indicates the

best strategy of both players. The attacker and defender can determine the Nash Equilibrium strategy combination $\left(T_a^*, T_d^*\right)$ via Eq. (17) and Eq. (18). Both of them will not deviate from this strategy combination.

5. Prototype system and experiments

To verify the effectiveness of HMC, a prototype based on SDN controller is implemented and the performance and security of HMC are analyzed in this section.

5.1 Prototype system

As shown in **Fig. 3**, the prototype system consists of two parts: control plane and data plane. The control plane runs on the SDN controller and is responsible for multicast group management and the multicast tree calculation. In addition, the control plane also manages the data plane via Openflow protocol. The data plane consists of Openflow switches which forward traffic based on the commands from the controller and forward the multicast protocol packets to the controller via Packet-in packets [16].

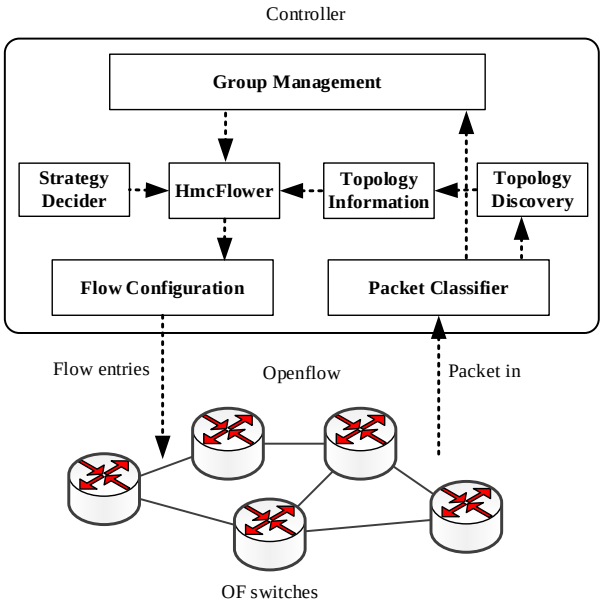


Fig. 3. HMC prototype system based on SDN

On the control plane of HMC, Packet Classifier classifies the Packet-in packets. And then the multicast protocol packets are delivered to the Group Management and the other Packet-in packets are delivered to the Topology Discovery. According to multicast protocol packets, Group Management manages the multicast events and authenticates the multicast users. While Topology Discovery calculates the network topology and stores it in the Topology Information database. Strategy Decider calculates the best strategy of defender using the utility functions of attacker and defender. Based on network topology and hopping strategy, HmcFlower, the core module of HMC, calculates the constrained minimum hotness tree for multicast group. HmcFlower generates flow entries and sends them to the data plane through Flow Configuration in order to update the multicast tree.

5.2 Experiments and analyses

We utilize Mininet [31] to build our experimental environment based on SDN, where Openflow1.0 [32, 33] is used as southbound Interface and NOX [34] is used as controller. We use 2.53 GHz Intel Xeon and 32G RAM 64 bit computing platform in the experiments.

BRITE [35] is used as the network topology generator in the experiments, by which random topologies are generated based on Waxman model [35] with parameters $\alpha = 0.2$, $\beta = 0.15$. The parameter P in the hotness update function φ (Eq. (7)) is set as $P = 0.5$ and the initial hotnesses of all the nodes in network are set to 1. The delay constraint is set to 10ms and we suppose that all the nodes in the network satisfy bandwidth constraint. Robustness of HMC is also evaluated under different constraint conditions.

5.2.1 Parameter selection

In Eq. (7), γ and λ are two system parameters for updating the hotnesses of nodes in the network. To determine the optimal value of the two parameters, related experiments are conducted. We set $\gamma = 1.1$ and five values within the interval $[1/1.1, 1/1.001]$ are selected to investigate the effect of λ. The defender can detect 10 compromised nodes in unit time, i.e. $f = 10$. The detected nodes will be precluded in multicast tree during the next defense period. The attack period and defense period are set as 10 unit time and 1 unit time, respectively. A random topology of 1000 nodes is generated in the experiment. The scale of multicast group is set as $|g| = 5$ and the multicast communication time is set as $T_c = 100$. We utilize the compromised traffic proportion (CT) to evaluate the performance of parameters, as shown in **Fig. 4**.

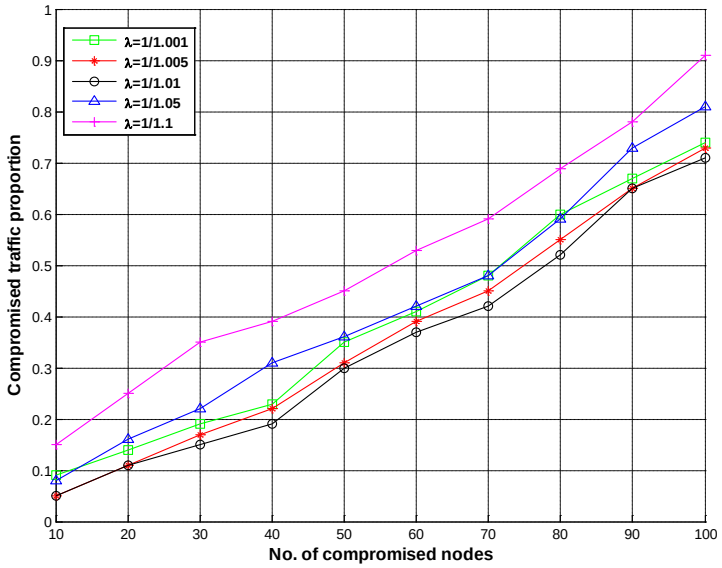


Fig. 4. Comparison of CT under different system parameter values

The horizontal coordinate stands for the number of nodes that can be compromised by the attacker at the same time, which indicates different attack power. The vertical coordinate stands for compromised traffic proportion. The least compromised traffic proportion is

achieved when $\lambda = 1/1.01$. When λ is small, the hotnesses of nodes decline quickly, thus some nodes will be selected into multicast tree multiple times during one attack period. Therefore, the attacker can compromise more traffic on these nodes. When λ is large, the hotnesses of nodes cannot be randomly declined effectively and the randomness of multicast tree will decrease. Thus, the attacker can exclude the attacked nodes when target traffic is compromised in recent attack periods. Therefore, the exploration space of the attacker decreases and more traffic can be compromised by the attacker. In the following experiments, system parameters γ and λ are set as $\gamma = 1.1, \lambda = 1/1.01$, respectively.

5.2.2 Evaluation of security

To evaluate the security of HMC, a random topology of 1000 nodes is generated in the experiments and the scale of multicast group is set as $|g| = 5$. The attacker can compromise 10 nodes with DoS attack at the same time, while the defender can detect 1 compromised node in unit time, i.e. $f = 1$. The theoretical probability of multicast communication being compromised is shown in **Fig. 5**, where horizontal coordinate stands for the time of multicast communication. When the attack period is 1 unit time and multicast tree is static ($T_a = 1, T_d = \text{Inf}$), the compromised probability increases with time. In this case, if one attack fails, the attacker can preclude some nodes out of multicast tree narrowing down the exploration space of the attacker along with the attacks. Thus, the probability of multicast being compromised will increase with the communication time. If the hopping periods of both players are the same ($T_a = 1, T_d = 1$), the probability of multicast being compromised stays unchanged. When the attack hops, the multicast tree also hops, so that the attacker can't preclude any node out of multicast tree. Therefore, the compromised probability stays unchanged. When the defense period is 1 unit time and attack does not hop ($T_a = \text{Inf}, T_d = 1$), the defender can gradually detect the compromised nodes and avoid them from being present on multicast tree. Therefore, the compromised probability of multicast decreases with time. After 10 unit time, there are no compromised node in the multicast tree, so the compromised probability is 0. Compared with the case of static defense ($T_a = 1, T_d = \text{Inf}$), the dynamic defense ($T_a = 1, T_d = 1$) that is adopted by HMC can mitigate DoS attack effectively. If the communication time is long enough, the multicast compromised probability decreases by more than 0.6.

A random network of 1000 nodes is generated and the multicast communication time is set as $T_c = 100$. The defender can detect 10 compromised nodes in 1 unit time. i.e. $f = 10$. The theoretical value of CT, experimental value of CT and experimental values of RTM [18] are shown in **Fig. 6**. In the case that both periods of attack and defense are 1 unit time, compromised traffic proportion will increase as the attack power increases. When $T_a = 1, T_d = 10$, in other words, the attack hops faster than multicast tree, the attacker can preclude some nodes out of multicast tree and the compromised traffic proportion will arise. As we can see from **Fig. 6**, the compromised traffic proportion increases significantly. On the contrary, when $T_a = 10, T_d = 1$, i.e., multicast tree hops faster than the attack, the defender can detect a part of compromised nodes and exclude them from multicast tree. Therefore, the compromised traffic will decrease. However, when there are a large number of compromised nodes, the protective effect of HMC is still limited. As shown in **Fig. 6,** the theoretical values of CT are very close to the experimental values. Our theoretical analyses can correctly reflect the situation of multicast communication being attacked under different attack powers and

hopping periods. In addition, it can be noticed that the values of *CT* obtained by RTM is higher than that of HMC. The reason is that HMC constructs constrained minimum hotness tree in the network, which avoid long time stay of the nodes in the multicast tree, thus it's more difficult for attacker to compromise multicast traffic.

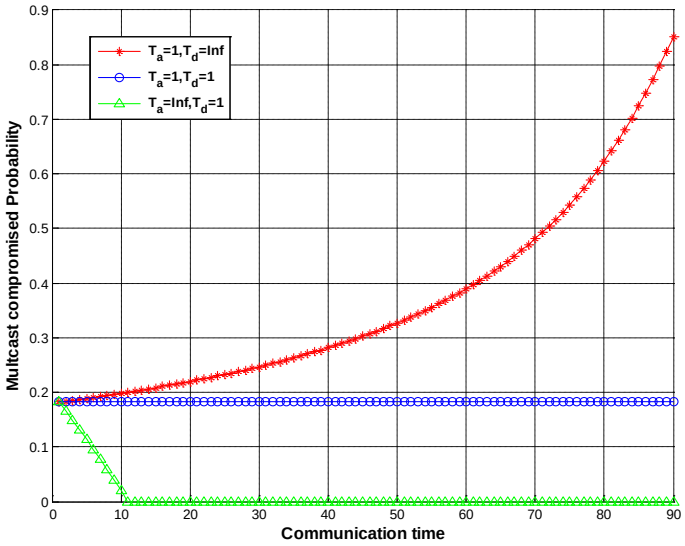


Fig. 5. The probability of multicast communication being compromised

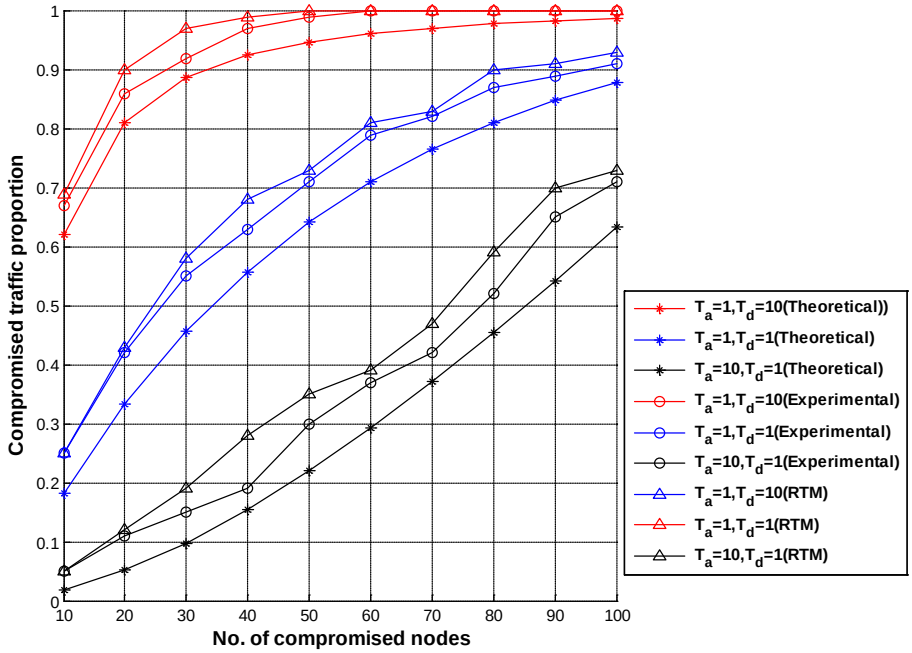


Fig. 6. Comparison among HMC's theoretical *CT* (line with stars), HMC's experimental *CT* (line with circles) and RTM's experimental *CT* (line with triangles)

5.2.3 Evaluation of performance

The communication delay is one of the most important measurement of multicast QoS. The average delay of traditional multicast route algorithm KPP, HMC and RTM are compared in this experiment. For multicast group $g=(s,D)$, the multicast average delay AD is defined as Eq. (19). In the experiment, a random network of 100 nodes is generated. From this network, 5~20 nodes are chosen randomly as a multicast group, then the multicast tree hops continuously for 100 periods and the average value of AD is calculated.

$$AD=\frac{1}{|D|}\sum_{d\in D}Delay\left(P_{T_g}(s,d)\right) \tag{19}$$

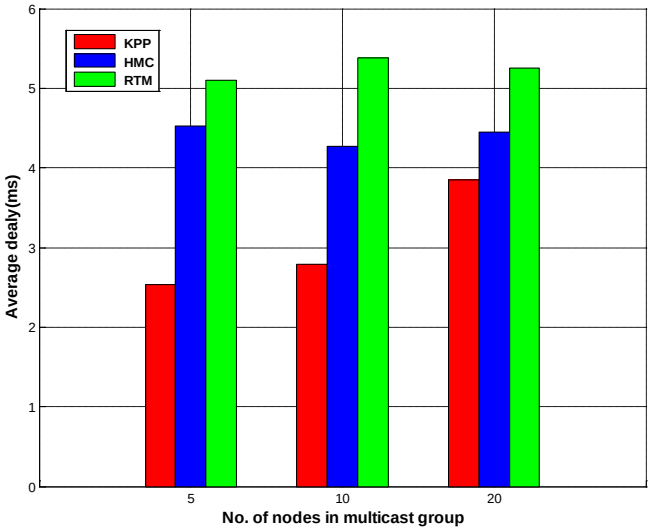


Fig. 7. Comparison among multicast average delay of KPP algorithm, HMC and RTM

It can be seen from **Fig. 7** that average delay of HMC is larger than KPP. In KPP algorithm, a Steiner tree is built for multicast group, in which minimal sum of delay of multicast communication is required. While in HMC, a minimum hotness tree is built, which requires the sum of hotnesses of nodes in multicast tree is minimized. HMC only requires that the delay of multicast communication satisfies delay constraint. Therefore, the average delay of HMC is larger than that of KPP but still under delay constraint. As we can see in **Fig. 7**, the average delay of RTM is the largest among the three methods. This is because RTM method constructs random multicast tree regardless of communication delay and it is likely to present some paths in the multicast tree with long communication delay.

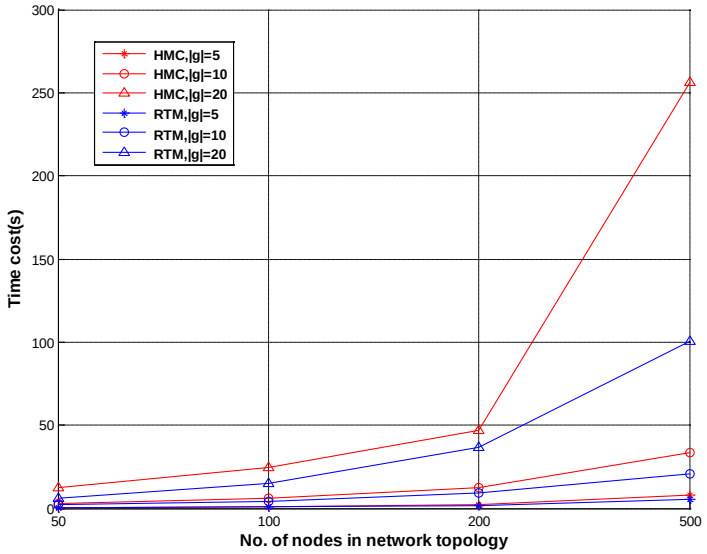


Fig. 8. The time cost of calculating multicast tree

During each defense period, the multicast tree hops and the route are updated in HMC. Therefore, HMC needs more time cost and space cost to calculate and update multicast route. In this experiment, time cost of calculating multicast tree between HMC and RTM is compared, as shown in **Fig. 8**, where the horizontal coordinate stands for the number of nodes in network and the vertical coordinate stands for the time cost. As can be seen, the time cost of both methods increases with the scales of network and multicast group. The time cost of calculating multicast tree for HMC is larger than that for RTM, because the QoS constraints is considered in HMC, which is time-consuming. When the number of nodes in network is 500 and $|g| = 20$, there exist a large number of cases in which the delay from multicast source node to destination node does not satisfy the constraint when calculating constrained minimum hotness tree. As a result, the k-shortest path need to be calculated multiple times and the time cost increases dramatically.

In Openflow1.0, a single flow entry can contain multiple actions and we take advantage of this character to realize traffic replication and multi-port forwarding. For a multicast tree T_g, only one flow entry is required on one node in T_g. Assuming that there are L nodes on the multicast tree at most and p_i denotes the probability that there exists i nodes on the multicast tree, the expectation of the number of flow entries N for deploying multicast tree can be shown as Eq. (20). The staged update method of multicast tree is used in HMC, as illustrated step 3) in section 3.3, the flow entries of both new and old (except the multicast source node) multicast tree exist in the network. Therefore, the expectation of flowtable space for HMC is $2N-1$.

$$N = \sum_{i \in [1,L]} p_i \times i \tag{20}$$

5.2.4 Evaluation of robustness

The robustness of HMC reflects its ability to maintain effectiveness under various network states and user demands. We consider not only the user's delay demand but also nodes'

bandwidth load capacity to evaluate the robustness of HMC. In the experiments, a random topology of 100 nodes is generated and the scale of multicast group is set as $|g|=5$. Firstly, we assume that all the nodes in the network have enough bandwidth and evaluate the fraction of legal multicast tree. The comparison between fraction of legal multicast trees that satisfy delay constraint generated by HMC and RTM is shown in **Fig. 9**, where the horizontal coordinate stands for the user's delay demand. As we can see, all the multicast trees generated by HMC satisfy the delay demands, because the delay constraint is considered when constrained minimum hotness trees are constructed. However, some multicast trees generated by RTM do not satisfy the delay demands as RTM only considers the randomness of multicast tree and delay constraint is not satisfied. When the delay demand becomes larger, more multicast trees generated by RTM can satisfy delay constraint. Therefore, the fraction of legal multicast trees increases as the user's delay demand increases.

In this experiment, the fractions of multicast trees satisfying bandwidth constraint generated by HMC and RTM are compared. We take the longest delay in the network topology as delay demand. The results are shown in **Fig. 10**, where the horizontal coordinate stands for the fraction of nodes that have insufficient bandwidth. As we can see, all the multicast trees generated by HMC satisfy bandwidth constraint, because HMC eliminates the nodes without enough bandwidth load capacity when generating multicast tree. However, RTM randomly generates multicast trees regardless of the node bandwidth load capacity. Therefore, nodes without enough bandwidth load capacity exist in some multicast trees. The larger the fraction of nodes without enough bandwidth, the more illegal multicast trees will be generated by RTM. This is obvious since the nodes without enough bandwidth load capacity have a higher probability to be selected into multicast trees by RTM when there are more such nodes. However, it is worth mentioning that HMC may fail to construct available multicast trees when the proportion of nodes without enough bandwidth load capacity is large enough, as there may be not enough available nodes to be selected into a multicast tree.

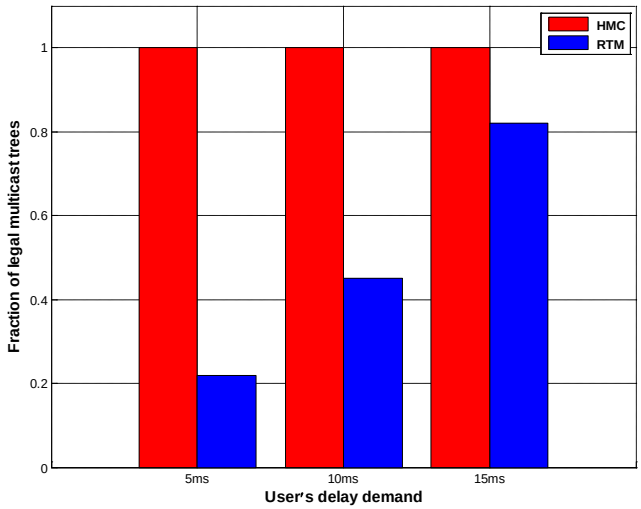


Fig. 9. The comparison of fraction of multicast trees satisfying delay constraint generated by HMC and RTM

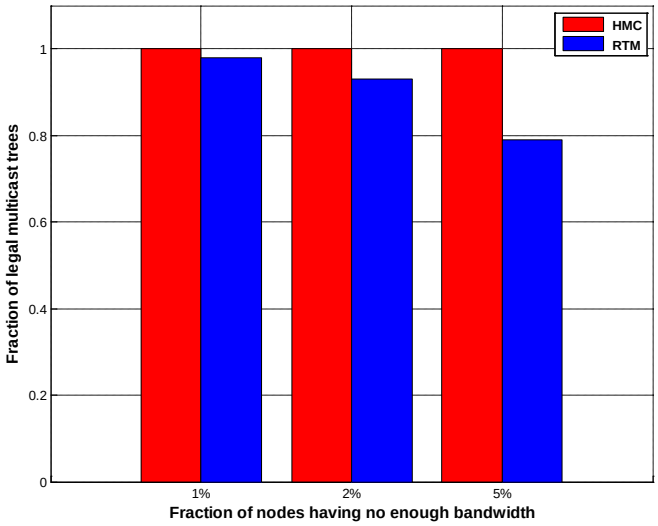


Fig. 10. The comparison of fraction of multicast trees satisfying bandwidth constraint generated by HMC and RTM

6. Conclusion and future work

In this paper, hopping multicast communication framework HMC based on SDN was proposed. It takes advantage of moving target defense to resist DoS attack on multicast route. In HMC, the multicast tree hops with time, bringing difficulty for the attacker to discover the route of multicast, so that the probability of multicast communication being compromised is decreased. We also proved that the routing problem in HMC is NP-complete and then a heuristic algorithm is proposed to solve it. The game of attacker and defender is analyzed and Nash equilibrium is used as the best hopping period. Experiments show that HMC can protect multicast communication against DoS attack effectively. In addition, HMC achieves better performance and robustness compared with traditional methods.

References

[1] Studer, A. and A. Perrig, "The coremelt attack," *Computer Security–ESORICS 2009*, Springer. pp. 37-52, 2009. Article (CrossRef Link)

[2] Kang, M.S., S.B. Lee, and V.D. Gligor, "The Crossfire Attack," in *Proc. of IEEE Symposium on Security and Privacy*, pp, 127-141, 2013. Article (CrossRef Link)

[3] Athreya, A.P., X. Wang, Y.S. Kim, Y. Tian, and P. Tague, "Resistance Is Not Futile: Detecting DDoS Attacks without Packet Inspection," in *Proc. of Information Security Applications: 14th International Workshop* , pp. 19-21, 2013. Article (CrossRef Link)

[4] Gkounis, D., V. Kotronis, and X. Dimitropoulos, "Towards Defeating the Crossfire Attack using SDN," *arXiv preprint*, arXiv:1412.2013, 2014. Article (CrossRef Link)

[5] Xue, L., X. Luo, E.W.W. Chan, and X. Zhan. "Towards detecting target link flooding attack," in *Proc. of Usenix Conference on Large Installation System Administration*, pp. 81-96, 2014. Article (CrossRef Link)

[6] Lee, S.B. and V.D. Gligor, "FLoc: Dependable Link Access for Legitimate Traffic in Flooding Attacks," in *Proc. of IEEE International Conference on Distributed Computing Systems*, pp. 327-338, 2010. Article (CrossRef Link)

[7] National Cyber Leap Year Summit 2009 co-chairs' report, "Networking and information technology research and development," *Technical report*, Sept. 2009. Article (CrossRef Link)
[8] Cyberspace, T., "Strategic Plan for the Federal Cybersecurity Research and Development Program," *Executive Office of the President National Science and Technology Council*, 2011. Article (CrossRef Link)
[9] Jajodia, S., A.K. Ghosh, V. Swarup, C. Wang, and X.S. Wang, "Moving target defense: creating asymmetric uncertainty for cyber threats," *Springer Science & Business Media*, vol. 54, 2011. Article (CrossRef Link)
[10] Al-Shaer, E., "Toward Network Configuration Randomization for Moving Target Defense," *Springer New York*, pp. 153-159, 2011. Article (CrossRef Link)
[11] McKeown, N., "Software-defined networking," *INFOCOM keynote talk 2009*, vol. 17, no. 2, pp. 30-32, 2009. Article (CrossRef Link)
[12] Iyer, A., P. Kumar, and V. Mann. "Avalanche: Data center Multicast using software defined networking," in *Proc. of Sixth International Conference on Communication Systems and Networks*, pp. 1-8, 2014. Article (CrossRef Link)
[13] Craig, A., B. Nandy, I. Lambadaris, and P. Ashwood-Smith, "Load balancing for multicast traffic in SDN using real-time link cost modification," in *Proc. of IEEE International Conference on Communications*, pp. 5789-5795, 2015. Article (CrossRef Link)
[14] Zhang, S.Q., Q. Zhang, H. Bannazadeh, and A. Leon-Garcia, "Routing Algorithms for Network Function Virtualization Enabled Multicast Topology on SDN," *IEEE Transactions on Network & Service Management*, vol., 12, no.4, pp. 580-594, 2015. Article (CrossRef Link)
[15] Shen, S.H., L.H. Huang, D.N. Yang, and W.T. Chen, "Reliable multicast routing for software-defined networks," pp. 181-189, 2015. Article (CrossRef Link)
[16] Zou, J., G. Shou, Z. Guo, and Y. Hu, "Design and implementation of secure multicast based on SDN," in *Proc. of Broadband Network & Multimedia Technology (IC-BNMT)*, in *5th IEEE International Conference*, pp. 124-128, 2013. Article (CrossRef Link)
[17] Pfeiffenberger, T., J.L. Du, P.B. Arruda, and A. Anzaloni, "Reliable and flexible communications for power systems: Fault-tolerant multicast with SDN/OpenFlow," in *Proc. of International Conference on New Technologies, Mobility and Security*, pp. 1-6, 2015. Article (CrossRef Link)
[18] Huang K, C.Y., Lan J, H, jia, "Random Tree Multicast Communications in Reconfigurable Network," *International Journal of Future Generation Communication and Networking*, vol. 9, no.1, pp. 1-10, 2016. Article (CrossRef Link)
[19] Duan, Q., E. Al-Shaer, and H. Jafarian, "Efficient Random Route Mutation considering flow and network constraints," in *Proc. of Communications and Network Security (CNS)*, pp. 260-268, 2013. Article (CrossRef Link)
[20] Jafarian, J., E. Al-Shaer, and Q. Duan, "Formal Approach for Route Agility against Persistent Attackers," in *Proc. of Computer Security – ESORICS 2013*, pp. 237-254, 2013. Article (CrossRef Link)
[21] Zhao, Z., D. Gong, B. Lu, F. Liu, and C. Zhang, "SDN-based Double Hopping Communication against sniffer attack," *Mathematical Problems in Engineering*, 2016. Article (CrossRef Link)
[22] Kompella, V.P., J.C. Pasquale, and G.C. Polyzos, "Multicast routing for multimedia communication," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 286-292, 1993. Article (CrossRef Link)
[23] Hwang F K, R.D.S., Winter P, "The Steiner tree problem," *Elsevier*, 1992. Article (CrossRef Link)
[24] Yen, J.Y., "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712-716, 1971. Article (CrossRef Link)
[25] Prim, R.C., "Shortest connection networks and some generalizations," *Bell Labs Technical Journal*, vol. 36, no. 6, pp. 1389-1401, 2010. Article (CrossRef Link)
[26] Dolev, S. and S.T. David, "SDN-Based Private Interconnection," in *Proc. of Network Computing and Applications (NCA) International Symposium*, pp. 129-136, 2014. Article (CrossRef Link)

[27] Gkounis, D., V. Kotronis, and X. Dimitropoulos, "Towards Defeating the Crossfire Attack using SDN," *Computer Science*, 2014. Article (CrossRef Link)

[28] Ronald, V.D.P., S. Boele, F. Dijkstra, A. Barczyk, G. Van Malenstein, J.H. Chen, and J. Mambretti, "Multipathing with MPTCP and OpenFlow," *High Performance Computing, Networking, Storage and Analysis (SCC)*, pp. 1617-1624, 2012. Article (CrossRef Link)

[29] Egilmez, H.E., S.T. Dane, K.T. Bagci, and A.M. Tekalp, "OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks," in *Proc. of Signal & Information Processing Association Summit and Conference*, pp. 1-8, 2012. Article (CrossRef Link)

[30] Qazi, Z.A., C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," in *Proc. of the ACM SIGCOMM conference on SIGCOMM*, vol. 43, no. 4, pp. 27-38, 2013. Article (CrossRef Link)

[31] Lantz, B., B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proc. of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, pp. 19, 2010. Article (CrossRef Link)

[32] McKeown, N., T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69-74, 2008. Article (CrossRef Link)

[33] OpenFlow Specification, online available http://www.OpenFlow.org/wk/index.php/Main_ Page. Article (CrossRef Link)

[34] Gude, N., T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. Mckeown, and S. Shenker, "NOX: towards an operating system for networks," *Acm Sigcomm Computer Communication Review*, vol. 38, no. 3, pp. 105-110, 2008. Article (CrossRef Link)

[35] Medina, A., I. Matta, and J. Byers, "BRITE: a flexible generator of Internet topologies," *Technical Report*, 2000. Article (CrossRef Link)

[36] Garey, M.R. and D.S. Johnson,"Computers and Intractability: A Guide to the Theory of NP-Completeness," in *Proc. of W.H. Freeman and Company*, 1979. Article (CrossRef Link)

Appendix I

Lemma 1: Minimum hotness tree problem is NP-complete.

Given graph $G(V,E,H)$ and a node subset $R \subseteq V$, the minimum hotness tree for R in G is a tree covering all the nodes in R with minimum hotness. Its decision problem is described as follows.

For a certain $K \in Z^+$, if there exists a tree $T=(V_1,E_1,H_1)$ with hotness no larger than K, which satisfies $R \subseteq V_1 \subseteq V$, $E_1 \subseteq E$?

Proof:

Tree T contains at most $|V|$ nodes. Obviously, whether the hotness of T is not larger than K can be verified in polynomial time. Therefore, minimum hotness tree problem is actually in NP.

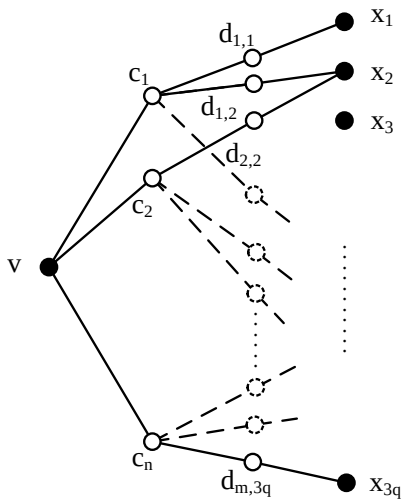


Fig. 11. The graph constructed for the transformation process
(The nodes in R are marked with black)

X3C problem (Exact Cover by 3-Sets) is a known NP-complete problem [36]. Then, we propose a reduction from X3C to minimum hotness tree. Then we prove that the reduction is executable in polynomial time.

Given an instance of X3C defined by the set $X=\{x_1,x_2,\cdots,x_{3q}\}$ and a collection of 3-element sets $C=\{c_1,c_2,\cdots,c_n\}$, we build an instance of minimum hotness tree.

A graph $G=(V,E,H)$ is built as shown in **Fig. 11**, where $V=\{v_0\}\cup C\cup D\cup X$, $D=\{d_{ij}\mid c_i\in C\wedge x_j\in c_i\}$ denotes the node set and $E=\{(v,c_i)\mid 1\le i\le n\}\cup\{(c_i,d_{ij})\mid 1\le i\le n, x_j\in c_i\}\cup\{(d_{ij},x_j)\mid 1\le i\le n, x_j\in c_i\}$ denotes the edge set. In graph G, node c_i and x_j ($x_j\in c_i$) are linked by node d_{ij}. Set the hotnesses of all nodes to 1, $R=\{v\}\cup X$ and $K=7q+1$. Obviously, the building process can be finished in polynomial time.

If $C'\subseteq C$ is an exact 3-cover of X, then $T=(V_1,E_1,H_1)$, where

$$V_1 = \{v_0\} \cup \{c_i \mid c_i \in C'\} \cup X \cup \{d_{ij} \mid c_i \in C', x_j \in c_i\},$$
$$E_1 = \{(v_0, c_i) \mid c_i \in C'\} \cup \{(c_i, d_{ij}) \mid c_i \in C', x_j \in c_i\} \cup \{(d_{ij}, x_j) \mid c_i \in C', x_j \in c_i\}$$

For $\forall x_j \in X$, x_j is present in a 3-element of C', we can say T is connected and acyclic, i.e. T is a tree. With $|X| = 3q$ as we define, we can get $|C'| = q$ because C' is an exact 3-cover of X. Assume $D' = \{d_{ij} \mid c_i \in C', x_j \in c_i\}$, then, node $\forall x_i \in X$ connects with node $c_j \in C'$ through node $d_{ij} \in D'$. It's easy to deduce $|D'| = |X| = 3q$ from X3C. Therefore, the number of nodes in T is $|V_1| = |X| + |D'| + |C'| + 1 = 7q + 1$, i.e. the hotness of T is $7q+1$. In other words, if there exists an exact 3-cover, there is a tree covering R with hotness no more than $7q+1$.

On the contrary, suppose that $T = (V', E', H')$ is a subtree of $G = (V, E, H)$ with hotness $7q+1$ and $R \subseteq V'$. Because the hotness of each node is 1, the number of nodes in T is $7q+1$. And there are $3q+1$ nodes in R, so that $|V' - R| = 4q$, i.e. the total number of c-nodes and d-nodes is $4q$. There are $3q$ x-node in T, and each x-node connects with one d-node at least to ensure the connectivity of T. While one d-node can connect only one x-node at most, therefore, there are $3q$ at least d-nodes in T. Similarly, each d-node connects with a c-node and each c-node connects with 3 d-nodes. Therefore, there are q c-nodes in T at least. The total number of c-nodes and d-node is $4q$ at least, therefore, it can be concluded that the total number of c-nodes and d-node is $4q$ exactly. Therefore, there are $3q$ d-nodes and q c-nodes in T. In graph G, $3q$ x-nodes are connected by q c-nodes through d -nodes and each c-node is connected by at most 3 x-nodes. Therefore, any c-node is connected by 3 x-nodes and the x-node sets connected with different c-nodes do not overlap with each other. Let $C' = \{c_i \mid (c_i, d_{ij}) \in E, (d_{ij}, x_j) \in E, 1 \le j \le 3q\}$, then C' is an exact 3-cover of X.

In conclusion, X3C problem can be reduced to the minimum hotness tree problem in polynomial time, so that the minimum hotness tree problem is NP-complete.

Theorem 1: Constrained minimum hotness tree problem is NP-complete.

Given graph $G = (V, E, H)$, where node set $R \subseteq V$ is the covered node set, the minimum hotness tree should satisfy delay upper bound Δ_d and bandwidth lower bound Δ_l.

Proof:

Given the node set and edge set, we can verify in polynomial time that

1) If the node and edge sets can compose a tree.
2) If the node set can cover the node set R.
3) If the tree can satisfy the delay upper bound Δ_d and the bandwidth lower bound Δ_l.

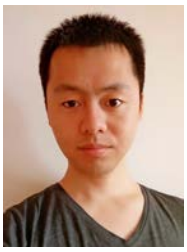
Therefore, this problem is NP.

Suppose that there exists a polynomial time algorithm A. Then for any minimum hotness tree problem, we can construct a constrained minimum hotness tree as follows.

Node set R is the covered node set, Δ_d is the length of longest simple path in the graph and Δ_l is the smallest bandwidth in the network. Then algorithm A can solve the minimum hotness tree problem exactly. Since A is a polynomial time algorithm, we can get the conclusion that the minimum hotness tree problem is not NP-complete, which is contradictory

with Lemma 1. Therefore, the supposition is invalid. In other words, there does not exist a polynomial time algorithm to solve constrained minimum hotness tree problem.

In conclusion, constrained minimum hotness tree problem is NP-complete.



Zheng Zhao received his B.S. degree from Dalian University of Technology in 2010 and received his M.S. degree from Zhengzhou Science and Technology Institute in 2013. Currently, he is a Ph.D candidate of Zhengzhou Science and Technology Institute. His research interests include network security and next generation internet.

Fenlin Liu is currently a professor of Zhengzhou Science and Technology Institute. His research interest includes information security, software protection theory and next generation internet **.**

Daofu Gong received his B.S. degree, M.S. degree and Ph.D from Zhengzhou Science and Technology Institute in 2006, 2009 and 2013 respectively. He is currently a lecturer in Zhengzhou Science and Technology Institute. His research interests include information hiding, software protection theory and next generation internet.