Flow Table Congestion in Software Defined Networks

Tauseef Jamal∗, Pedro Amaral∗‡, Khurram Abbas
∗Instituto de Telecomunicações, Lisboa, Portugal
‡Dept de Eng Electrotecnica, Faculdade de Ciencias e Tecnologia, Universidade Nova de Lisboa, Caparica, Portugal
tauseef.jamal@lx.it.pt, pfa@fct.unl.pt, abbaskhurram98@gmail.com.

***Abstract*—Security is a major concern for today's networks and network applications. Denial of Service (DoS) is major threat to availability of service. DoS is easy to detect but hard to mitigate. There are several types of DoS attacks, such as flooding etc. Software Defined Networks (SDN) inherit security threats from traditional networks along with threats specific to them. Flow table congestion is an example of such problem specific to SDN networks. The attacker generates multiple packets as messages to the controller. Because of this, the switch's Ternary Content Addressable Memory (TCAM) is flooded with controller replies. TCAMs are very expensive and power hungry. To avoid this type of attack, different aggregation strategies have been proposed. These techniques save TCAMs at the cost of lost statistics in OpenFlow flow table. In this paper, we used an improved version of Optimal Routing Table Construction (ORTC) to perform flow aggregation similar to Fast Flow Table Aggregation (FFTA). Some disadvantages of FFTA include loss of flow table statistics and counts. Our proposed technique reduces the number of flows to solve Flow Table Congestion Problem along with maintaining the consistency of statistics.**

Keywords- congestion; flow table; optimal routing.

I. INTRODUCTION

Software Defined Networking (SDN) is an innovative paradigm for computer networking [1]. Traditional networks are hardware oriented, while SDN shifts the focus from hardware to software. It makes computer networks more maintainable by providing logically centralized control and separating them from the forwarding functionality. It offers network programmability, agility, central management, open standards and vendor neutral solutions.

On the other hand, this separation of network functionality opens a more targeted attack surface. Potential attackers can target control or data plane more accurately than ever before. In addition, if attackers can gain control of the control plane, the entire network is compromised. SDN inherits traditional network security issues along with issues specific to it. SDN centralized monitoring can be used to cope with security issues more efficiently [1].

One of these security issues is Denial of Service, which targets the availability of the system and disrupts the legitimate user to available system services. The attacker sends a huge amount of useless traffic or exploits some vulnerability on the target system. This causes the system to stop responding, crash or reboot. Sources of the DoS can be single or multiple. The scale of this attack varies i.e. it can employ a single system to thousands of systems, known as bots. These types of attacks are a real threat to systems which are supposed to be providing online services because these can result into loss of revenue, loss of customer's trust and loss of reputation.

This article revolves around the following problem.

A) How flow tables in OpenFlow can be used to avoid flow table saturation attack.

B) How a secure system can be designed around SDN control plane to avoid traditional attacks.

The rest of the paper is organized as follows. In Section II, related work is described; Section II models the problem statement and discusses the solution, while Section IV details the implementation.

II. LITERATURE ANALYSIS

A brief description of literature survey is given below.

A) Optimal Routing Table Construction

The backbone routers of the Internet are populated with routes by Border Gateway Protocol (BGP). Optimal Routing Table Construction (ORTC) calculates the minimal number of equivalent routes locally with modifying BGP [1]. First, the binary tree representation of IP routing table is constructed. Then, the resultant table is obtained by performing the following two generalized steps of netting in sequence on the original table.

1. Sub Netting
2. Super Netting

This causes a huge decrease in flows but ORTC is applied locally to every router because modifying BGP along with other protocol on every network device is a challenging task [1].

B) Fast Flow Table Aggregation

Flow tables of OpenFlow are TCAM hungry because they contain comparatively more fields in the header than traditional switches and routers. The flow table of OpenFlow v1.0.0 contains 12 matching fields for headers, whose size is more than 237 bits per flow entry. Newer versions of OpenFlow have increasing number of matching fields to cope with different dynamic requirements. On the other hand, TCAM's are very expensive and power hungry. Because of this OpenFlow, switches suffer from Flow Table Congestion Problem (FTCP).

Aggregation of flows can be used to reduce the demand of TCAM's by OpenFlow switches. Flow aggregation is an economical solution because it is software based. It best matches the nature of SDN. FFTA provides an improved technique for aggregation of flows. This technique consists of three steps [2].

 ISBN: 978-1-61208-615-6

1. Rule list is partitioned into permutable prefixes
2. Modified aggregation of prefixes
3. Merging of bits iteratively

This aggregation is very efficient, but there is a problem with this approach. It mixes up the entries which results in coarser grain statistics. It requires some statistical estimates to fine grain the statistics so that original flow counts and other parameters can be obtained from the aggregated flow [3].

C) Flow Table Reduction Scheme

In traditional networks switches perform actions according to rigid protocols. SDN overcomes this inflexibility by separating the control function from packet transmission. Using SDN, the dynamic policies can be easily implemented. On the other hand, this dynamic nature can cause redundant entries in flow tables. Flow tables are very important to SDN. However, due to limited size of TCAM, this redundancy leads to the problem of congestion of flow table. Some solutions are proposed to divide the flows into mice flows and giant flows. FRTS puts three constraints which every flow table reduction scheme should follow for proper functioning of OpenFlow enabled networks. These constraints are

Consistency: Same actions should be allotted to the flows after the reduction.

Absoluteness: Rules that are manually added should match and execute on priority.

Accuracy: The statistics should be accurate at any interval of time.

III. PROBLEM MODELLING

Attack trees [4] are conceptual diagrams well known for security assessment of a system. These show how an asset can be attacked. There exist two generic types of security modeling for systems.

- System oriented
- Attacker oriented

The former system models itself and performs security analysis on the system. The later one models the attack on the system with a focus on attacker objectives. An attack tree contains a single root which represents the overall objective of the attacker. This objective is iteratively decomposed into finer grained and quantitative objectives, which form the child and leaves of the attack tree.

The problem is modeled using attack tree with denial of service as a root node. Denial of service can be performed in SDN by attacking controller, switch or any of the hosts. To conduct a denial of service, the attacker should flood or exploit any one of these components. This type of relation is called “OR” relation because the attacker can achieve its parent objective by performing any one of child objectives.

Figure 1 shows first level of attack tree. It shows that denial of service can be achieved by attacking any one of the child components i.e. OpenFlow switch, controller or host.

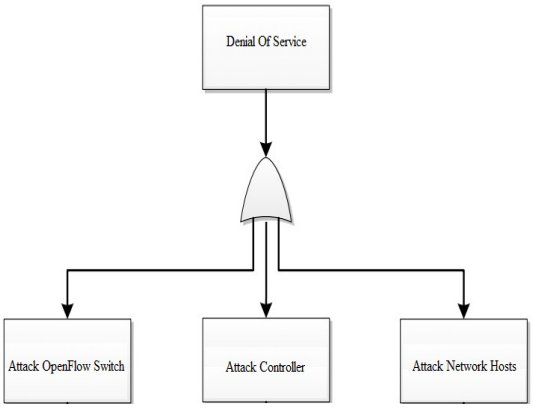

Figure 1. Level one attack tree for DoS.

A simplified attack tree for attack on controller is shown in Figure 2. This attack tree models the objective of the attacker having intent of attacking SDN controller. SDN controller can be exposed to DoS by flooding it or by performing logical or vulnerability attack on it. Vulnerability DoS requires three activities i.e. finger printing of OS, finding vulnerability of that OS and writing exploit along with payload. These all activities must be carried out to perform logical DoS on controller. On the other hand, flooding DoS can be performed by carrying out any one of the leaf activities, as shown in Figure 2.

The attack tree in Figure 3 models the actual threat against which we have proposed a solution. Flow tables store flow entries in TCAMs. TCAM is now a de facto industry standard [5][6]. TCAMs are very expensive and have limited amount of memory. Any attacker can generate a large amount of Packet_In messages to controller. In the response of those messages the switch is flooded with policy replies against messages from controller. However, due to TCAM’s limited memory, new rules are not installed. Legal new packets suffer from such attack because policy against them cannot become part of flow table. To avoid this type of attack, an efficient and consistent mechanism of flow aggregation is required. On the other hand, aggregation of flows results in modification and generalization of counters. Statistics are mixed up due to aggregation.

A. Proposed Solution

An improved version of Optimal Routing Table Construction is used to minimize the flow entries in the flow table. Moreover, to ensure consistency of statistics improved ORTC is applied to core switches, which results in removal of redundant entries without modifying the statistics. Details of these techniques are provided below. First of all, we will discuss ORTC. After that, we will use Bit Weaving [7] approach to create a binary tree representation of non-prefix OpenFlow entries because ORTC works only at prefixes. After that, we will discuss where this technique should be applied along with

 ISBN: 978-1-61208-615-6

considering the consistency of statistics of OpenFlow flow table.

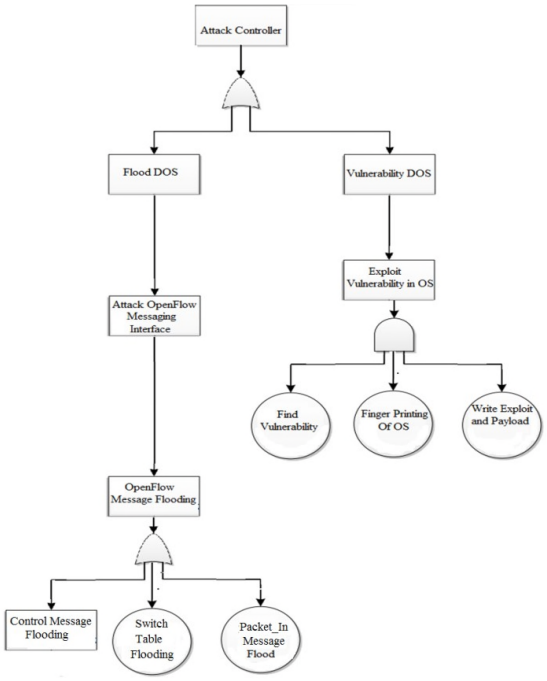

Figure 2. Attack tree for attack on controller.

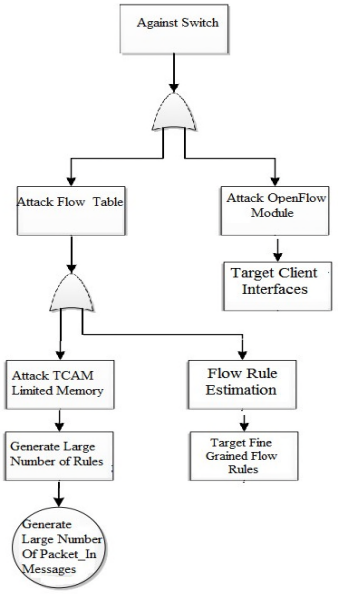

Figure 3. Attack tree for Flow Table Congestion Attack on OpenFlow switch.

a) Optimal Routing Table Construction

ORTC [1] is an optimal solution to minimize IP routing entries or prefixes. It contains 3 steps to get optimal reduced tree.

Step 1: In the first step, a tree representation of table in binary format is normalized by setting zero or two children for every node. This is done by initializing the next hop for newly created leaf node by the next hop of nearest ancestor. After the first step, Table I is converted into Figure 5.

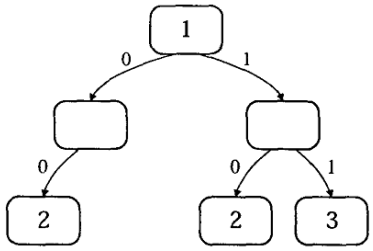

Figure 4. Normalized tree [1].

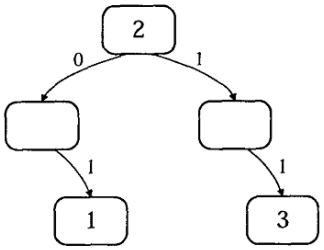

Figure 5. Results of ORTC application [1].

Table I. EXAMPLE ROUTING TABLE

Destination IPs (Binary	Action
*	1
00*	2
10*	2
11*	3

Step 2: The second step of ORTC performs calculation for next hop which are prevalent for every level of the given table. The following operation symbolically defined by "#", shown in Figure 6, is used to define next hops up the tree.

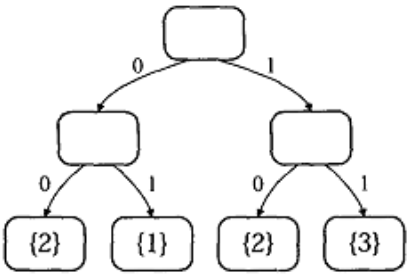

Figure 6. Table after step 1 [1].

Step 3: This step uses sub netting to remove redundant entries from the tree by selecting next hops. Pre order traversal can be used. Also, one can traverse by level down the root. If the nearest ancestor has a next hop, than it will be inherited by the node. The state of table during step 3 is shown in Figure 7, while Figure 4 shows the resultant binary tree representation. The equivalent table for routes is shown in Table II.

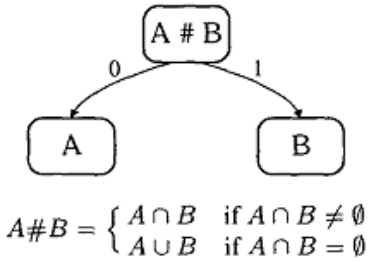

Figure 7. Prevalent hop calculation.

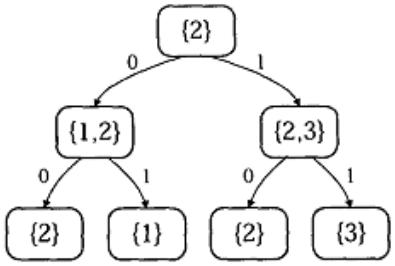

Figure 8. Table after step 2 [1].

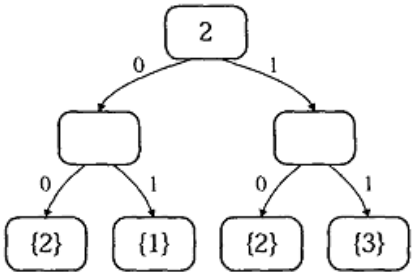

Figure 9. Table during step 3.

TABLE II. RESULTANT TABLE AFTER ORTC

Destination IPs (Binary	Action
*	2
01*	1
11*	3

b) Bit Weaving

ORTC is prefix based approach. So, it in not directly applicable to OpenFlow based flows because they can include wild card anywhere. Therefore, we need to get a binary tree representation of OpenFlow based flows by applying bit weaving initial permutation. The output of this permutation can be now directly utilized by ORTC engine.

c) Avoiding loss of statistics in OpenFlow Table

Consider the topology shown in Figure 8. In this topology S1 is the first level core switch. Similarly, S5 and S2 are second level core switches. While S3, S4, S6 and S7 are end switches. Consider Host 1 with IP 10.0.0.1. Different statistics and counts related to Host 1 may be available on every switch. But actual aggregated statistics of Host 1 in this case will be available at switch S3. On all other switches the statistics regarding Host 1 are redundant entries. Same is the case with all other hosts. Generally, we can say that end switches contain total statistics related to the connected host, while when we move from end switches to core switches, the statistics found here are redundant.

By comparing statistics for Host 1 at different switches, we can conclude that total statistics regarding Host 1 are available at end switch connected to it i.e. switch S1. While core switches contain only redundant statistics regarding to different hosts in network. Hence, it is safe to apply flow aggregation on core switches without the fear of losing statistics. Core switches also contain more flows than end switches in real life cases.

B. Implementations

There exist two types of development techniques for SDN application and policy enforcement.

Reactive Approach:

When a new packet arrives at OpenFlow switch, its header is matched against flow table entries. Packet is sent to controller if no match occurs in flow table. In reaction to that controller installs appropriate policy in flow table against that packet. This approach is called reactive approach, which is event based.

Proactive Approach:

In this approach all necessary policies are installed proactively. REST API's are widely used for this approach.

We used both approaches in our solution appropriately. For rate limiting and filtering reactive approach was used. To install reduced flows and remove the original flows from core switches proactive approach was used [8].

C. Data Collection

We classified data into two types.

a) Input Data:

There were two types of input data.

1. Flow rules for OpenFlow table were generated by using a wrapper around ClassBench [9] tool. This wrapper used same seed files generated and used by ClassBench.

2. To overflow switch TCAM storage with flow rules huge amount of *Packet_In* messages needed to generate. These messages were generated by Scapy [10]. Scapy is used to craft custom packets.

b) Output Data:

Output data was collected from three sources.

- CLI of different tools such as Mininet.
- Data written by custom utilities in log files.
- Web interface of SFlow tool was also used to collect some data.

IV. EVALUATION

In this section, statistics and results are presented against attacks carried out in the SDN environment, using attack trees. There were two types of attacks through which an attacker can achieve the objective of DoS in SDN. The first type of attack is a traditional attack which SDN inherits from traditional networks i.e., attack against host. While the second attack is specific to SDN network i.e., attack against switch. The evaluation presents the later type of attack.

A. Attack Against Switch

To avoid this type of attack, a solution is proposed in design and methodology. We used switch as learning switch to conduct the desired behavior. Script for POX learning switch [11] is available as learningswitch.py while for Opendaylight learning switch [27] is implemented using Service Abstraction Layer SAL with the name of MD-SAL Layer 2 switch. CBench is a tool used for benchmarking of controller. It measures throughput and latency of different SDN controllers. It connects to controller and simulates millions of devices sending messages to the controller. Throughput of Opendaylight, Floodlight and POX [12][13] are shown in Figure 10. The figure shows throughput of controllers in terms of flows per second for a given number of switches. These statistics are collected from a machine with Ubuntu 14.04 LTS and single core with 4 gb ram.

If considerable number of packets are generated on some given switch interface using scapy than packet_in messages will be generated and because of controller's response switch memory will be exhausted. After that, new incoming legitimate packets will be dropped at that switch, because rules against these packets will not be installed. Also, when several times a switch fails to install a flow, controller starts to ignore the packet in messages from that controller. This causes packet loss too. Figure 11 shows packet loss after flow table congestion for different controllers with link bandwidth set to 100 Mbps.

To cope with this type of problem, we used flow aggregation. Figure 12 shows the results of reduced set of rules along with original rule sets against number of switches and hosts. This shows that for core switches, where aggregation does not disturb statistics in our scenario, we were successfully able to reduce rule from 23% to 41%.

This technique also causes reduced bandwidth consumption between controller and switch, because now switch communicates with the controller less frequently in case of Packet_In messages.

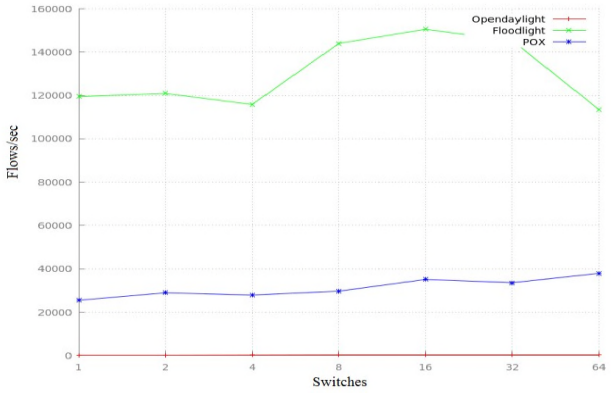

Figure 10. Throughput analysis.

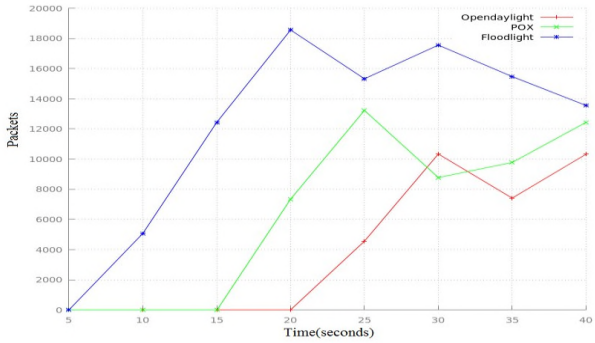

Figure 11. Packets loss analysis.

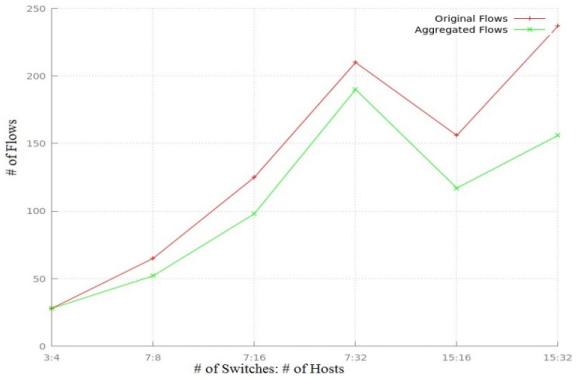

Figure 12. Reduced flows analysis.

V. CONCLUSIONS

Flow table congestion is a problem specific to SDN. Optimal routing table construction was used with enhancement to prevent this type of attack. Existing aggregation techniques also prevent flow table congestion, but these also cause in loss of counts and statistics. A simple strategy was used to identify redundant statistics. Next, aggregation was applied on those identified flow tables to avoid loss of statistics in OpenFlow table successfully. This caused about 23% to 40% percent compression in different cases.

 ISBN: 978-1-61208-615-6

A simple idea was also proposed to reconstruct redundant statistics for our sample topology. But that idea needs to be generalized with the use of different degree of equation along with statistical estimation. This technique can also be matured to be applied even on original flows and their reconstruction, so that aggregation can also be useful for edge switches. As a future work we aim to apply cooperative mechanism [14][15][16] to identify early congestion to avoid inefficiency in routers.

ACKNOWLEDGMENT

This work is funded by FCT/MEC through national funds and when applicable co-funded by FEDER – PT2020 partnership agreement under the project UID/EEA/50008/2013.

REFERENCES

[1] Richard P. Draves, Christopher King, Srinivasan Venkatachary and Brian N. Zill, " Constructing Optimal IP Routing Tables " 1998.

[2] Shouxi Luo "Fast incremental flow table aggregation in SDN" Aug. 2014.

[3] B. Leng, L. Huang and Y. Zhang, "A Mechanism for Reducing Flow Tables in Software Defined Network " 2015.

[4] V. Saini and V. Paruchuri, "Threat modeling using attack trees" J.Comput. Sci. Coll., 23(4):124{131, April 2008.

[5] K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary, “Algorithms for advanced packet classification with ternary CAMs,” in Proc.ACM SIGCOMM, Aug. 2005, pp. 193–204.

[6] “Content addressable memory,” Integrated Device Technology, Inc.,

[7] Chad R. Meiners, Alex X. Liu, and Eric Torng "Bit Weaving: A Non-Prefix Approach to Compressing Packet Classifiers in TCAMs" April 2012.

[8] T. Jamal and P. Mendes, "Cooperative relaying in user-centric networking under interference conditions", Communications Magazine IEEE, vol. 52, pp. 18-24, 2014, ISSN 0163-6804.

[9] T. Jamal, P. Mendes, and A. Zúquete, “Wireless Cooperative Relaying Based on Opportunistic Relay Selection,” International Journal on Advances in Networks and Services, vol. 05, no. 2, pp. 116–127, Jun. 2012.

[10] S Kaur and J Singh, "Network Programmability Using POX Controller" 2014.

[11] T. Jamal, M Alam and M Umair, “Detection and Prevention against RTS Attacks in Wireless LANs” in Proc of IEEE C-CODE, Islamabad, Pakistan 2017.

[12] Jeremy M. Dover "A denial of service attack against the SDN controller" 2013.

[13] Open Floodlight The OpenDaylight Platform | OpenDaylight https://www.opendaylight.org, Accessed on 7 October 2015.

[14] T. Jamal, P. Mendes, A. Zuquete, “Analysis of hybrid relaying in cooperative WLAN” In proc. of IEEE/IFIP Wireless Days 2013.

[15] T. Jamal and P. Mendes, “Relay Selection Approaches for Wireless Cooperative Networks,” in Proc. of IEEE WiMob, Niagara Falls, Canada, Oct. 2010.

[16] T. Jamal, P. Mendes, and A. Zúquete, “RelaySpot: A Framework for Opportunistic Cooperative Relaying,” in Proc. of IARIA ACCESS, Luxembourg, Jun. 2011.

 ISBN: 978-1-61208-615-6