SDN-based DDoS Attack Detection with Cross-Plane Collaboration and Lightweight Flow Monitoring

Xiangrui Yang, Biao Han*, Zhigang Sun, Jinfeng Huang
School of Computer, National University of Defense Technology, Changsha, P. R. China
{yangxiangrui11, nudtbill, sunzhigang, hjf}@nudt.edu.cn

***Abstract*—Distributed Denial of Service (DDoS) attack is one of the biggest concerns for security professionals. Traditional DDoS attack detection mechanisms are based on middle-box devices or SDN controllers, which either lack network-wide monitoring information or suffer with serious southbound communication overhead and detection delay. In this paper, we propose a SDN-based DDoS attack detection framework with cross-plane collaboration called OverWatch, which performs a two-stage granularity filtering procedure between coarse-grained detection data plane and fine-grained detection control plane for abnormal flows. It leverages computational capabilities that currently underutilized on OpenFlow switches to shrink the detection range for fine-grained DDoS attack detections. In OverWatch, we propose a lightweight flow monitoring algorithm to capture the key features of DDoS attack traffics on the data plane by polling the values of counters in OpenFlow switches. Experiments are conducted in an evaluating network with a FPGA-based OpenFlow switch prototype and the Ryu controller, which reveal that our proposed OverWatch framework and flow monitoring algorithm can greatly improve the detection efficiency, as well as reduce the detection delay and southbound communication overhead.**

Keywords—DDoS Attack Detection, OpenFlow switch, Software Defined Networking, Cross-Plane Collaboration

I. INTRODUCTION

Distributed Denial of Service (DDoS) attacks are typically explicit attempts to disrupt legitimate users' access to services, which are often launched by botnet computers that are simultaneously and continuously sending a large number of service requests to the victims. The victims either respond so slowly as to be unusable or crash completely. According to Arbor Networks, which offers services to protect against DDoS attacks, they observed over 124,000 DDoS attacks per week since 2016 and they believe this number is growing rapidly [1]. Besides, since breaking the 100Gbps barrier in 2010, DDoS attacks are also increasing in size, making them more and more difficult to defend against. Therefore, protecting network-wide resources from these frequent and large DDoS attacks necessitates the research community to focus on developing high-efficient and accurate DDoS attack detection mechanisms that can be appropriately deployed in time.

Traditional DDoS attack detection involves the use of middle-box devices, which are generally complicated integration of customized hardware and software. Although they are superior in detection performance, it is found that middle-box based DDoS attack detection is inflexible with network evolution, for example, hard to support new network architectures or protocols. Moreover, these devices are usually independently deployed in a network and have different communication interfaces. This hinders them from a holistic perception of the network status, which is becoming extremely critical for network-wide detection of increasingly frequent and large volume DDoS attacks [2].

Recently, extensive research efforts have been conducted to apply software-defined networking (SDN) in diagnosing DDoS attacks in a global point of view [3] [4] [5]. The central controller of SDN can quickly install measurement rules on all switches and run a controller-based DDoS attack detection application without additional cost of middle-box devices. This makes SDN an ideal platform for DDoS attack detection. However, as most SDN-based approaches leverage the controller to poll flow statistics or packets from each switch on the data plane [3] [6], the detection delay significantly increases compared with middle-boxes based methods. Moreover, in large scale networks with a great number of SDN switches connecting to a central controller, the communication overhead of the southbound interface for DDoS attack detection can be greatly increased as the traffics between data plane and control plane are becoming enormous. The above-mentioned problems make the southbound interface become a potential bottleneck while applying SDN controller for DDoS attack detection, which has a significant impact on the detection performance. More seriously, this bottleneck may result in SDN controller saturation attack, as discussed in [7].

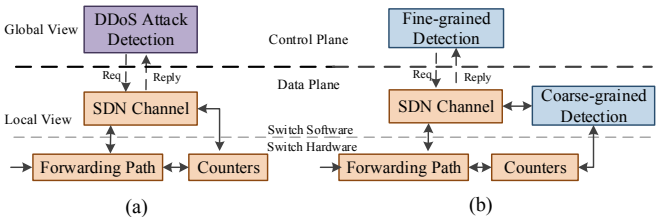

Fig. 1. Design schematic of SDN-based DDoS attack detection framework: (a) controller-based framework; (b) our proposed cross-plane framework.

In this paper, we develop a SDN-based DDoS attack detection framework with cross-plane collaboration, called OverWatch, which overcomes the problems of SDN controller based DDoS attack detection methods by a novel two-stage detection procedure between data plane and control plane, as shown in Fig. 1. In the proposed OverWatch framework, a

* Corresponding author.

978-1-5090-5019-2/17/$31.00 ©2017 IEEE

coarse-grained lightweight flow monitoring algorithm on the data plane and open interfaces for fine-grained detection mechanisms on the control plane are proposed to utilize OpenFlow switches to filtrate abnormal flows on the data plane before applying control plane DDoS attack detection methods. Specifically, we focus on two key features of DDoS attack traffic: volume feature and asymmetry feature [3]. In order to extract the key features, the proposed lightweight flow monitoring algorithm filters abnormal flows by polling the values of counters in OpenFlow switches. Besides, the proposed OverWatch framework also provides open interfaces for the control plane to conduct fine-grained DDoS attack detections for the abnormal flows. This framework improves the DDoS attack detection efficiency and reduces the detection overhead by a novel cross-plane design with two-stage granularity filtering and lightweight flow monitoring. We implement and evaluate OverWatch in a SDN scenario with a FPGA-based OpenFlow switch prototype [8] and the Ryu controller [11]. Experimental results reveal that the proposed framework greatly improves SDN-based DDoS attack detection performance in terms of detection delay and communication overhead.

The main contributions of this paper can be summarized as follows:

- We propose a SDN-based DDoS attack detection framework with cross-plane collaboration, which performs a two-stage granularity filtering procedure between coarse-grained detection data plane and fine-grained detection control plane for abnormal flows.
- We design a lightweight flow monitoring algorithm to capture the key features of DDoS attack traffics on the data plane by polling the values of counters in OpenFlow switches. An open interface for the control plane to conduct fine-grained DDoS attack detections is provided for the abnormal flows filtered by the data plane.
- We conduct extensive DDoS attack detection experiments on a FPGA-based OpenFlow switch prototype with a Ryu controller. Through performance evaluation, we validate that our proposed cross-plane framework and flow monitoring algorithm can greatly improve the detection accuracy, as well as reduce the detection delay and communication overhead between data plane and control plane.

The rest of this paper is organized as follows. Section II covers the related works. Section III presents an overview of our framework. The proposed flow monitoring algorithm is presented in Section IV. Section V presents our evaluation results. This paper is concluded in Section VI.

II. Related Works

Characteristics of DDoS attacks have been widely studied and researchers have proposed various methods to detect DDoS attacks [3] [4] [5]. In the field of network monitoring, many researches also focus on how to reduce monitoring overhead while ensuring accuracy [5] [6]. Among them, Braga *et.al* in [3] proposes using Support Vector Machine (SVM) to detect DDoS attacks on the SDN controller. Chowdhury.S.R *et al.* in [6] proposes Payless, which includes a SDN-based flow monitoring method based on an adaptive statistics collection algorithm that delivers highly accurate statistics without significant overhead. Zhang Y. in [5] proposes a prediction-based method to control the granularity of measurement in abnormal traffic detection in order to reduce the monitoring overhead. Although we share the similar objective with them, our method, which involves switch resources to conduct coarse-grained detection, is quite different from theirs.

Former attempts to achieve a better overhead/accuracy balance in DDoS attack detection are through traffic sampling [10]. However, studies like [24] have shown them to be inaccurate, as they may easily miss small flows. Although there are some new sampling algorithms to overcome the problem of being inaccurate, most of them are difficult to be deployed on general OpenFlow switches. At the same time, there are some newly proposed methods such as AVANT-GUARD [7] and OFX [14], which attempt to offload functions onto data plane to gain performance and reduce overhead. The major difference between those works and ours is that instead of addressing the issue from a single-plane perspective, we use a cross-plane collaboration method to optimize SDN-based DDoS attack detection methods.

III. System Description

A. Motivation of Cross-Plane Collaborative Detection

In SDN, in order to improve the performance of DDoS attack detection and reduce detection delay, it reasonable to deploy traffic filtering functions on the data plane [7] [14]. However, typical SDN-based DDoS attack detection methods require constantly polling statistics from switches to maintain a holistic view of the network on the controller. This greatly increases the communication overhead of southbound interface. Fortunately, , most OpenFlow switches or hybrid switches that support OpenFlow specification consist of one or more CPUs running operating system with abundant computational resource that are currently far from fully utilized [12]. This inspires us to exploit the underutilized computing capabilities in switches to perform lightweight traffic monitoring for DDoS attacks.

On the other hand, a well-designed SDN-based DDoS attack detection method is required to analyze flow statistics from a number of switches to determine where the attackers are located, which paths the DDoS attack flows pass through and who is the exact target. This demands fine-grained DDoS attack detection functions running on the centralized controller. In addition, with the help of programmable characteristics of SDN, it is feasible to implement advanced and complex DDoS attack traffic monitoring applications on the controller. It inspires us to deploy fine-grained and holistic network-wide DDoS attack detection functions on the controller.

B. Architecture of the Proposed OverWatch Framework

We depict the architecture of our proposed SDN-based collaborative DDoS attack detection framework (OverWatch) in Fig. 2. It inherits the decoupling characteristic of SDN, which consists of data plane (i.e. OpenFlow switch), control plane (i.e. SDN controller) and the southbound channel (i.e. OpenFlow

protocol). OverWatch performs a two-stage granularity filtering procedure. It firstly conducts coarse-grained flow monitoring on the data plane switches to filter abnormal flows. Once abnormal flows are detected, a fine-grained DDoS attack detection is triggered on the control plane to deeply analyze the abnormal packets.

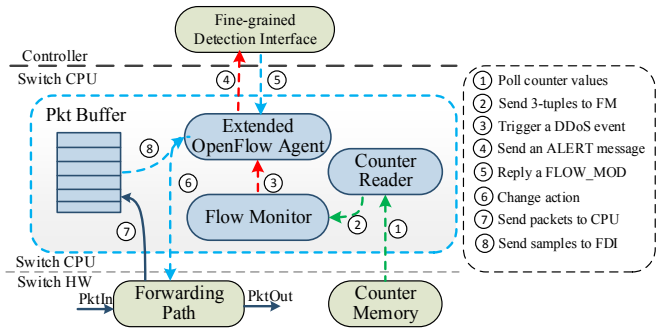

Fig. 2. Architecture and workflow of OverWatch framework.

In OverWatch, each OpenFlow switch is comprised of a **Flow Monitor (FM)** module and a **Counter Reader (CR)** module, together with an **Extended OpenFlow Agent (EOA)** module which is a modified OpenFlow agent to support southbound communication between the data plane and the control plane. On the control plane, we design a **Fine-grained Detection Interface (FDI)** module that aims to support any controller-based fine-grained DDoS attack detection methods. It provides filtered abnormal flow status and API functions that a controller-based application can use for fine-grained DDoS attack detections. Then, we describe the workflow of OverWatch briefly.

Counters of an OpenFlow switch typically count the number of packets and bytes passing through an element. OpenFlow specification defines a set of counters for each flow entry, flow table and switch port, etc. since specification 1.0. In OverWatch, CR module polls counter values from hardware successively, and sets these values as outputs to FM module. Then, in the FM module, these counter values will be processed by a flow monitoring algorithm to filter abnormal flows. Once abnormal flows are detected, the EOA module will be notified and send an `ALERT` message, which carries general status of the specific abnormal flow, to FDI module through southbound channel (i.e. OpenFlow channel).

When FDI module receives the `ALERT` message, it will store the abnormal flow status in the database and trigger an event to notify controller-based DDoS attack detection applications to conduct deep detection for the specific abnormal flow at the same time. According to our research [2] [24], a fine-grained DDoS attack detection usually requires deep packet inspections for those abnormal flows. Thus, in OverWatch, once an `ALERT` message is received, FDI module will send a standard `OFPT_FLOW_MOD` message (defined in OpenFlow specification since 1.0) to the specific switch. This message is able to instruct an OpenFlow switch to buffer packets from the specific abnormal flow, which will be sent to FDI module and stored in the database for fine-grained analysis.

C. Key Modules in OverWatch

Counter Reader (CR): The CR module on switch-local CPU is mainly responsible for successively polling counter values from switch hardware, and passing these values to FM module for flow monitoring.

TABLE I. PARAMETERS IN THE 3-TUPLE

Parameter(s)	Explanation
`CounterTypeFlag`	Indicating the counter types.
`SerialNumber`	Serial number of the specific port, flow entry, etc.
`CounterValues`	Counter values of the specific port, flow entry, etc.

The counter values related to the same pair-flow [3], port, or switch are represented by a {`CounterTypeFlag`, `SerialNumber`, `CounterValues`} 3-tuple, where `CounterTypeFlag` and `SerialNumber` are both used to identify a group of counter values that belongs to the same flow (port, flow entry, etc.).

There are three different aggregation levels, which are flow entry level, port level, and switch level respectively, for the CR module to poll counter values. The choice among these aggregation levels is determined by two metrics: monitoring strategy and switch performance. On the one hand, different monitoring granularity requires polling different types of counter values (e.g., considering a scenario where network admins plan to monitor each switch port for flow monitoring on the data plane, then the aggregation level should be set to port level.). Thus, the choice should consistent with the monitoring strategy. On the other hand, as an OpenFlow switch is able to support more than thousands of flow entries in its flow table currently, network admins should choose port level or switch level for low-end switches. Otherwise, with the number of flow entries grows, the CPU resources will soon be depleted.

Flow Monitor (FM): FM module is mainly responsible for preliminarily filter abnormal flows according to volume and asymmetry features of DDoS attack traffic. It consists of three parts: 1) Tuple Parser, which extracts counter values from 3-tuple. 2) Monitor Thread, which filters abnormal flows by utilizing specific counter values. 3) Flow Status Collector, which manages detected abnormal flow status. We present the working process of FM module below.

Firstly, 3-tuples sent by CR module are constantly received by Tuple Parser. Then, the counter values included in a 3-tuple are passed to a specific Monitor Thread according to which flow (or port) they belongs to. Next, these Monitor Threads leverage a flow monitoring algorithm, which we will discuss in great details in Section IV, to filter abnormal flows from each flow on the switch.

Once abnormal flows are detected on the data plane, an `Abnormal_Flow_Event` that records the status of the filtrated flow will be sent to the Flow Status Collector. From this point on, the Flow Status Collector delivers collected information to EOA module, in which the general information of the specific abnormal flow will be sent to FDI module as an `ALERT` message through southbound interface immediately.

Fine-grained Detection Interface (FDI): FDI module located on the control plane is mainly responsible for providing

general status of abnormal flows and API functions for fine-grained DDoS attack detection.

```
{"Abnormal_Flow_Status":{
  "Index": ["index"],
  "Time": ["current_time"],
  "Switch_ID": ["data_path_ID"],
  "Aggregation": ["Flow_Entry"|"Switch_Port"|"Switch"],
  "Serial_Num": ["Flow_ID"|"Port_ID"|"data_path_ID"],
  "Rate": ["Byte_Rate"]
}}
```

Fig. 3. The format of an abnormal flow status record in FDI module.

TABLE II. EXEMPLARY API FUNCTIONS PROVIDED IN OVERWATCH FOR FINE-GRAINED DDOS ATTACK DETECTION METHODS

Function	Description
`ReadARecord(Index)`	Obtain an abnormal flow status record by recording index in the database.
`DeleteARecord(Index)`	Delete an abnormal flow record from the database by recording index.
`GetPacketSamples(Index)`	Obtain sampled packets of a specific abnormal flow by recording index of a specific record.

In order to enable controller-based DDoS attack detection applications to obtain a holistic view of DDoS attacks, we define an `Abnormal_Flow_Status` (Fig. 3) object to record and store the status of every abnormal flow detected on the data plane. Controller-based DDoS attack detection applications can monitor abnormal network behaviors through this type of object. Additionally, FDI module also provides API functions that a controller-based DDoS attack detection application can use to collect abnormal packets and list history abnormal flow records, etc. Table II presents a few exemplary functions and descriptions.

IV. A LIGHTWEIGHT FLOW MONITORING ALGORITHM

In this section, we present a lightweight flow monitoring algorithm that is utilized in Flow Monitor module to detect abnormal flows caused by DDoS attacks Unlike many other flow monitoring algorithms, this lightweight algorithm aims to extract the key features of DDoS attack traffic by means of polling counter values from a general OpenFlow switch.

A. Feature Extraction of DDoS Attack Traffic

In this part, we present a brief explanation of the volume feature and asymmetry feature of DDoS attacks, together with a method of utilizing counters in OpenFlow switches to extract the two features of DDoS attack traffic.

There are some fundamental differences between a typical DDoS attack and normal network behaviors. As we have discussed in Section I, it is apparent that large traffic rate is one important feature for DDoS attacks. Moreover, during a DDoS attack, usually there is huge rate difference between flows coming into a victim server and flows going out of the server. Fortunately, the two features can both be determined by directly polling OpenFlow switch counter values. We define $C_{t_n}^{Byte}$ and $C_{t_n}^{Pkt}$ as the byte count and the packet count of a specific flow, port, or switch at time t_n respectively, the two DDoS attack traffic features can be expressed as followings:

1) Byte Count per Second at time t_n (B_{t_n}): describes the average byte rate of a flow, port or switch between time $t_{n\text{-}1}$ and t_n :

$$B_{t_n} = \frac{C_{t_n}^{Byte} - C_{t_{n-1}}^{Byte}}{t_n - t_{n-1}} \tag{1}$$

2) Packet Count per Second at time t_n (P_{t_n}): describes the average packet rate of a flow, port, or switch between time $t_{n\text{-}1}$ and t_n:

$$P_{t_n} = \frac{C_{t_n}^{Pkt} - C_{t_{n-1}}^{Pkt}}{t_n - t_{n-1}} \tag{2}$$

3) Byte Count Asymmetry ($A_{t_n}^{Byte}$) at time t_n: describes the average byte rate asymmetry of pair-flow, port or switch between time $t_{n\text{-}1}$ and t_n:

$$A_{t_n}^{Byte} = \frac{B_{t_n}^{in}}{B_{t_n}^{out}} \tag{3}$$

4) Packet Count Asymmetry ($A_{t_n}^{Pkt}$) at time t_n: describes the average packet rate asymmetry of pair-flow, port or switch between time $t_{n\text{-}1}$ and t_n:

$$A_{t_n}^{Pkt} = \frac{P_{t_n}^{in}}{P_{t_n}^{out}} \tag{4}$$

In the four metrics above, B_{t_n} and P_{t_n} describe the volume feature from the perspective of byte count and packet count respectively, while $A_{t_n}^{Byte}$ and $A_{t_n}^{Pkt}$ describe the asymmetry feature of DDoS attack traffic from the same perspectives.

B. Proposed Flow Monitoring Algorithm

In this part, we propose a lightweight algorithm that can be used to capture the great changes of the above four metrics. This algorithm leverages previous metric samples from a specific flow to estimate a future value. If the actual values for the four metrics all fall into the ranges of the predication, it indicates the current flow is normal. When all four observed values fall out of the acceptance range, we determine it as an abnormal flow caused by DDoS attacks.

More specifically, we maintain a list of n records for each metric, which is represented by $\mathbf{v}_i^a$ (consists of $v_{i,j}^a$, *j=1,2,3,4*). The predicted vector $\mathbf{v}_{n+1}^p$ for these metrics in time t_n is calculated in Eq. 5.

$$\mathbf{v}_{n+1}^p = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i^a \qquad \sum_{i=1}^{n} \lambda_i = 1 \tag{5}$$

In order to eliminate the impact of traffic spikes that may produce the similar changes as DDoS attacks in above four metrics, we use WMA (Weighted Moving-Average) to define λ_i for each $\mathbf{v}_i^a$.

Then, as shown in Eq. 6, for every element $v^a_{n+1,i}$ in $\mathbf{v}^a_{n+1}$, we use the ratio metric to compare the predicted output $\mathbf{v}^p_{n+1}$ with the actual vector $\mathbf{v}^a_{n+1}$.

$$R(v^p_{n+1,i}) = \left| \frac{v^a_{n+1,i}}{v^p_{n+1,i}} \right| \quad i = 1,2,3,4 \qquad (6)$$

Finally, to demonstrate the deviation between history records and the actual vector $\mathbf{v}^a_{n+1}$ in t_{n+1}, we utilize Pauta criterion in Gaussian distribution to define the acceptable range for the ratio metric. The definitions of upper and lower limits of the acceptance range are shown as Eq. 7 and Eq. 8.

$$R^u_a = R(avg_a + 3 \times std_a) \qquad (7)$$

$$R^l_a = R(avg_a - 3 \times std_a) \qquad (8)$$

The two parameters in Eq. 7 and Eq. 8, which are avg_a and std_a, represent the average value and standard deviation of $v^a_{i,j}$ in the n-record list respectively.

If one or more of the four ratio metrics $R(v^p_{n+1,i})$ fall into the acceptable range, it indicates no DDoS attack in the specific flow currently. Conversely, if all the four metrics fall out of the acceptable range, it shows it is an abnormal flow caused by DDoS attacks. Once an abnormal flow is detected in Monitor Thread of Flow Monitor module, the data plane will send an `ALERT` message to notify the controller for a fine-grained DDoS attack detection.

V. Performance Evaluation

In this section, we set up an evaluating network to evaluate OverWatch. To demonstrate the benefits of our framework, we also implement a typical controller-based flow monitoring method as a baseline, in which the controller periodically polls the counter values from switches at a constant time interval and uses the same algorithm to detect abnormal flows. In both methods. The time interval between every polling is set to 1s.

A. Experiment Setup

Fig. 4 illustrates the testbed of our experiments. It consists of a FPGA-based OpenFlow switch prototype [8] that runs Ubuntu 14.04 on the CPU of it, a control platform, a computer, running Ryu controller and up to eight hosts, which represent DDoS attackers, victims and normal traffic generators.

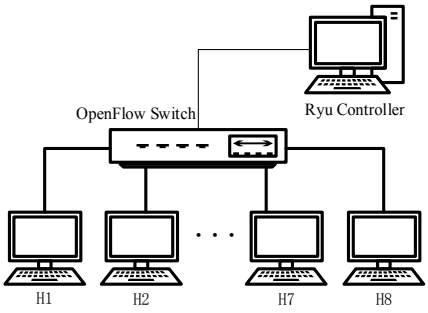

Fig. 4. Diagram of our testbed network.

We deployed OverWatch on this testbed. DDoS attack flows among hosts are generated using hping3 [9]. Fig. 5 is the timing diagram showing the start time and the end time for each attack flow in a scenario.

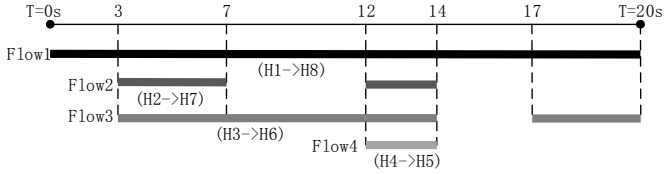

Fig. 5. Timing diagram of DDoS attack flows.

B. Efficiency of the Proposed Algorithm

In order to evaluate the efficiency of our algorithm deployed on the data plane. Firstly, we use hping3 to generate DDoS attack traffic from H1 to H3 within 20s. The changes in four metrics in the proposed algorithm and volume and asymmetry features of flow (H1→H3) are depicted in Fig. 6. These results demonstrate that the four metrics in our algorithm are able to capture both two features of the abnormal flow as soon as DDoS attack happens, which evidently shows the effectiveness of the algorithm.

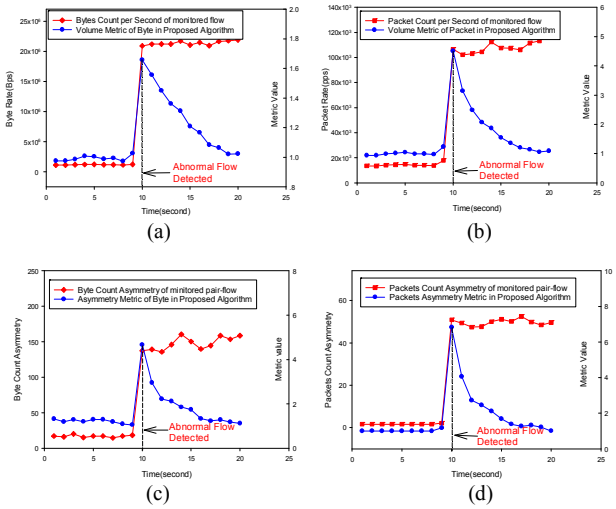

Fig. 6. The changes of four metrics in the proposed algorithm and volume and asymmetry features during a DDoS attack: (a) Byte rate and volume metric of byte in monitored flow; (b) Packet rate and volume metric of packet in monitored flow; (c) Byte count asymmetry and asymmetry metric of byte in monitored pair-flow; (d) Packet count asymmetry and asymmetry metric of packet in monitored pair-flow.

C. Effectiveness of the Proposed Framework

To benchmark the performance improvement in OverWatch, we use hping3 to conduct an attack from H1 to H3. Before diving into the result, there are two things need to be stressed out. The first is in order to eliminate the impact of installing new flow rules caused by spoofed attacker source address, we use non-spoofed source address during the attack. Secondly, to simulate large-scale networks where other switches may keep interacting with the controller, we leverage H2 and H4 to keep generating `ICMP` packets to the controller at a constant rate by means of AnySend [16].

The relation between detection time and southbound interface data rate is shown in Fig. 7. As the data rate on southbound interface grows, both versions take longer to obtain the status of abnormal flows for the controller. However, instead of periodically polling flow statistics, OverWatch only sends an `ALERT` message to notify the controller once abnormal flows are detected, which let it spend roughly half time less than the baseline method.

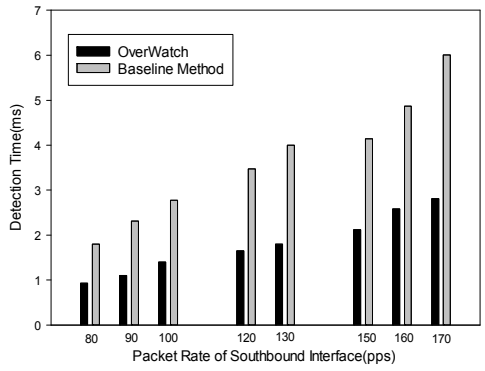

Fig. 7. The amount of time takes to detect the abnormal flow of OverWatch and baseline method.

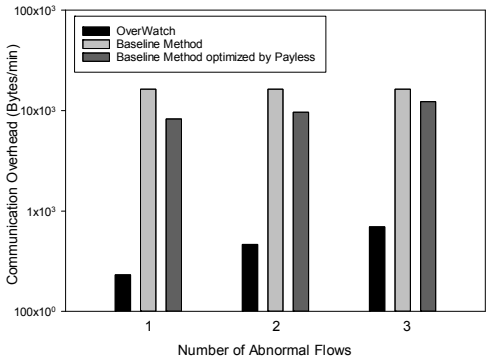

Fig. 8. Overhead of southbound interface in OverWatch, baseline method and baseline method optimized by Payless within 1minute.

To demonstrate the advantage of communication overhead reduction, we implement another method, which optimizes the baseline method by utilizing an adaptive polling algorithm proposed in Payless [6] to reduce communication overhead of southbound interface. Then, we implement these three methods in a scenario where different number of DDoS attack flows are generated from these hosts within 1 minute. We use Wireshark to capture all the packets of southbound interface during that time. The results are shown in Fig. 8. It is found that OverWatch has orders of magnitude less overhead than the other two methods. This is achieved by offloading coarse-grained flow monitoring mechanism onto data plane.

VI. CONCLUSIONS

To conclude, we introduced OverWatch, a SDN-based DDoS attack detection framework with cross-plane collaboration. OverWatch overcomes challenges of detection performance and communication overhead by performs two-stage granularity filtering procedure between coarse-grained detection data plane and fine-grained detection control plane for abnormal flows. Moreover, we also proposed a lightweight flow monitoring algorithm to filtrate abnormal flows before involving the control plane. Through experiments, we found that OverWatch can greatly achieve goals above. All these outcomes demonstrate the feasibility of OverWatch in large-scale networks.

ACKNOWLEDGEMENT

This research was supported in part by grants from Startup Research Grant of National University of Defense Technology (No.JC15-06-01), Chinese National Programs for High Technology Research and Development (863 Programs) (No.2013AA013505) and the project of National Science Foundation of China (No.61601483).

REFERENCES

[1] DDoSResearch,https://www.tripwire.com/state-of-security/security-data-protection/cyber-security/5-significant-ddos-attacks-2016/

[2] Zargar, Saman Taghavi, James Joshi, and David Tipper. "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks." *IEEE communications surveys & tutorials* 15.4 (2013): 2046-2069.

[3] Braga, Rodrigo, Edjard Mota, and Alexandre Passito. "Lightweight DDoS flooding attack detection using NOX/OpenFlow." *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*. IEEE, 2010.

[4] Xu, Yang, and Yong Liu. "DDoS attack detection under SDN context." *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*. IEEE, 2016.

[5] Zhang, Ying. "An adaptive flow counting method for anomaly detection in SDN." *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, 2013.

[6] Chowdhury, Shihabur Rahman, et al. "Payless: A low cost network monitoring framework for software defined networks." *Network Operations and Management Symposium (NOMS), 2014 IEEE*. IEEE, 2014.

[7] Shin, Seungwon, et al. "AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks." *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013.

[8] Netmagic,http://www.netmagic.org/

[9] Hping3,"Hping3 (8)-Linux Man Page." (2005).

[10] Claise, Benoit. "Cisco systems netflow services export version 9." (2004).

[11] Tomonori, F. U. J. I. T. A. "Introduction to ryu sdn framework." *Open Networking Summit* (2013).

[12] Mao, Jianbiao, et al. "Efficient mismatched packet buffer management with packet order-preserving for OpenFlow networks." *Computer Networks* 110 (2016): 91-103.

[13] Version, OpenFlow Switch Specification. "1.5. 1 (Protocol version 0x06), December, 2014."

[14] Sonchack, John, et al. "Enabling practical software-defined networking security applications with ofx." *Proceedings of the 2016 Network and Distributed System Security Symposium (NDSS)*. 2016.

[15] Mai, Jianning, et al. "Is sampled data sufficient for anomaly detection?." *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. ACM, 2006.

[16] Anysend,https://anysend.en.softonic.com