Scalable Traffic Sampling using Centrality Measure on Software-Defined Networks

Seunghyun Yoon, Taejin Ha, Sunghwan Kim, and Hyuk Lim

***Abstract*— With regard to cyber security, pervasive traffic visibility is one of the most essential functionalities for complex network systems. A traditional network system has limited access to core and edge switches on the network; on the other hand, software-defined network (SDN) technology can provide flexible and programmable network management operations. In this paper, we consider the practical problem concerning how to achieve scalable traffic measurement using SDN functionalities. Less-intrusive traffic monitoring can be achieved by using a packet-sampling technique that probabilistically captures data packets at switches, and the sampled traffic is steered towards a traffic analyzer such as an intrusion detection system (IDS) on SDN. We propose the use of a centrality measure in graph theory for deciding the traffic sampling points among the switches. In addition, we discuss how to decide the traffic sampling rates at the selected switches. The results of the simulation and SDN testbed experiments indicate that the proposed sampling point and rate decision methods enhance the intrusion detection performance of an IDS in terms of malicious traffic flows in large-scale networks.**

I. Introduction

As the population of Internet users continues to grow, the number of devices connected to the Internet is increasing rapidly. Because of this explosive expansion of the network scale, there exists huge demand for flexible and scalable network management. Moreover, cyber security for home and enterprise networks has become more important because our daily data applications and access to services such as banking, shopping, business, education, and transportation are provided via the Internet. One of the rapidly increasing threats is ransomware, which is computer malware that executes a cryptovirology attack and demands a ransom payment to restore the damage [1, 2]. These malware propagations via the Internet can be prevented by capturing suspicious packets on the network and inspecting them by using security application solutions such as an intrusion detection system (IDS). Usually, IDSs are deployed on networks in data centers. These systems are connected to edge or core switches, and are used to monitor network traffic patterns and inspect data packets in order to detect malicious activities.

A software-defined network (SDN) [1] is a promising technology that can provide flexibility, robustness, and programmability. Briefly, the principal concept of SDN is to decouple the network control plane from the data-forwarding plane. Forwarding decisions in traditional networks are made by a routing algorithm in each switch; on the other hand, the controller on an SDN is responsible for controlling the forwarding operations of the switches in a centralized manner. In this regard, the OpenFlow (OF)[2] protocol is one of the most popular protocols used for communication between the SDN controller (for the control plane) and SDN-enabled switches (for the data plane). Owing to this flexibility and programmability, SDN technology can be applied to cyber security network applications as well as high-performance data center networks. For example, the SDN controller can enable a switch to duplicate the traffic flow of interest and steer the traffic towards a traffic collector by simply updating the forwarding table of the switch via an OF protocol. Network programmability such as traffic duplication and rerouting can facilitate the implementation of traffic-monitoring operations for cyber security. In [1], a ransomware mitigation method that exploits SDN functionalities was proposed. It uses the SDN functionality for forwarding/steering traffic in order to inspect all the DNS traffic packets (that may include the DNS query for ransomware proxy servers) and the traffic duplication functionality to implement the DNS packet inspection without incurring delays in the DNS response time.

Another important problem associated with network traffic monitoring is the acquisition of network traffic packets in a less-intrusive manner. If every packet belonging to a traffic flow is captured and forwarded to a traffic collector, the amount of traffic that is newly generated for traffic monitoring purposes would be the same as that of the original traffic, and may cause significant congestion on the network. For less-intrusive monitoring, it would be desirable to selectively capture traffic packets rather than capturing every packet at the switches. This method is known as **packet sampling** and can be implemented in several ways. Given information about the network topology and traffic flows, it is essential to decide where to sample traffic and how much traffic is sampled at each of the selected switches when the total amount of sampled traffic is bounded for less-intrusive traffic monitoring. In terms of deciding the sampling points, we propose the use of a centrality measure in graph theory for exploiting the network topology and flow information. The use of a centrality measure enables us to select switches with relatively higher importance. Once a subset of switches is chosen, packets passing through each switch can be sampled at a certain rate in a fair manner. The results of the simulation and SDN testbed experiments show that the proposed approach can significantly reduce the sampling points

The authors are with the Gwangju Institute of Science and Technology (GIST), Korea.

[1] SDN: http://www.opennetworking.org/sdn-resources/sdn-definition

[2] OpenFlow: http://www.openflow.org

on the network while retaining the traffic inspection performance for malicious traffic such as that containing viruses and ransomware in a large network.

II. Network Traffic Monitoring

Network traffic monitoring includes various administrative operations to acquire network traffic information such as the routing paths of traffic flows, the traffic volume, the network/transport layer protocol types of the traffic flows, and the payload size distribution of the packets in each flow. The use of these various types of traffic information makes it possible to infer the network topology and the network resource status including the packet loss rates and congestion levels at the switches and routers on the network. Once information about the network traffic is gathered, it can be analyzed using statistical and information-theoretical methodologies for network operation and management. The analysis of the dynamic patterns resulting from changes in the traffic statistics allows a network administrator to detect abnormal operations of network links and nodes and to exactly identify faulty links and nodes. Subsequently, an appropriate management operation can be conducted to resolve the networking problems. For example, OpenNetMon was proposed to acquire the flow-level network status such as throughput, delay, and packet loss rate using the OpenFlow protocol for fine-grained traffic engineering [3]. In addition, network traffic monitoring is also an essential function of cyber security because malicious network activities such as DDoS and port scanning generate their own unique traffic patterns. Thus, security applications can use the traffic patterns to defend the network system against malicious threats. In [4], it was proposed to combine the OpenFlow and sFlow[3] protocols to gather flow-level traffic statistics in order to detect and identify network anomalies such as DDoS, worms, and port scanning attacks.

The use of network traffic monitoring requires the data packets belonging to each traffic flow to be captured in order to acquire information about these traffic flows. We consider two approaches: **partial-packet capturing** (PPC) and **full-packet capturing** (FPC). The former approach involves capturing and storing the header of a packet because it contains the most useful information about the flows. Optionally, PPC may capture the first several bytes of the packet payload as well as the header to obtain additional information about the contents of the packet payload. sFlow and NetFlow[4] are the most popular monitoring solutions for PPC. Because PPC captures at most the first two hundred bytes of each packet, the size of each report for the captured packets is much smaller than that of the original packets. If the reports of multiple packets are compressed before delivery to the traffic collector, the amount of additional traffic generated for network monitoring is further reduced. However, PPC can be limitedly used for malware inspection such as the detection of a computer virus, malicious code injection, and ransomware detection. On the other hand,

[3] sFlow: http://www.sflow.org

[4] NetFlow: http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html

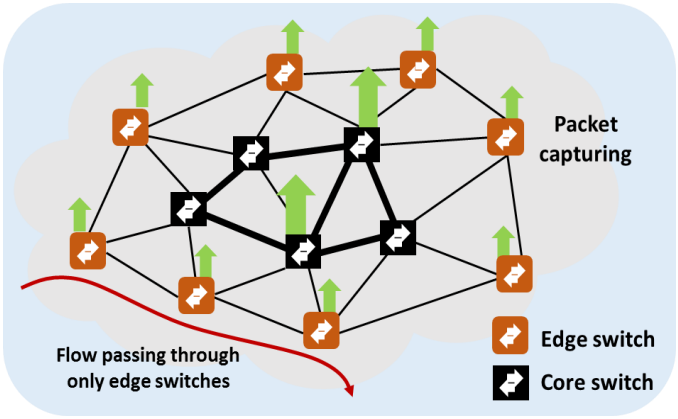


Fig. 1 Packet capturing at edge switches versus core switches.

FPC captures both the header and payload of a packet. Because it is possible to analyze the payload as well as the information specified in packet headers, a variety of network threats can be inspected and prevented. For example, an IDS can be deployed to inspect the payloads of data packets to detect potential malware using a signature-based matching method, which compares the payload of input packets with a number of known malicious binary patterns [5]. However, FPC causes the amount of traffic on the network to double and the resulting network congestion may disrupt the traffic if every packet is captured, and the header and payload of each packet are duplicated.

For network traffic monitoring, there exists a tradeoff relationship between the network resource overhead and the amount of information acquired for traffic inspection. As cyber threats and attacks become more complicated, it is desirable that network traffic monitoring systems are able to support the FPC functionality for cases when traffic monitoring is required for malware inspection.

III. Network Traffic Monitoring Location

The location at which network traffic monitoring is performed by capturing traffic packets is an important factor that has a significant impact on the monitoring performance. Ideally, any point on a network (i.e., all the switches and routers) would need to be accessible to capture the network flows for traffic inspection. In traditional networks, the number of traffic monitoring devices (e.g., network tap and port mirror) is quite limited, and feasible locations at which the devices are deployed are also restricted.

Figure 1 shows two possible locations for traffic monitoring, edge switches and core switches. Since the edge switches are directly connected to client devices, the number of flows at each switch is relatively small in comparison with that in switches on the network backbone. As a result, it is easy to achieve fine-grained traffic monitoring for the individual flows that pass through each switch. However, since one capturing device must be deployed at each edge switch, the number of capturing devices required for network-wide traffic monitoring would be considerable. This increase in the number of required capturing devices would incur management overhead, and may not be a feasible solution for a large network. As shown in Fig. 1, the core switches constitute the backbone of the network, and can be connected to the network of an Internet service provider (ISP). Since each core switch serves a number of flows at a high

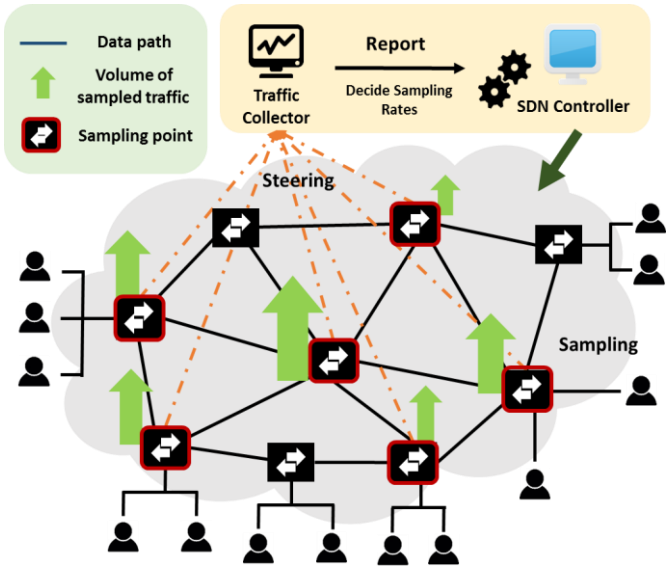


Fig. 2 Packet capturing on SDN with OF-enabled switches.

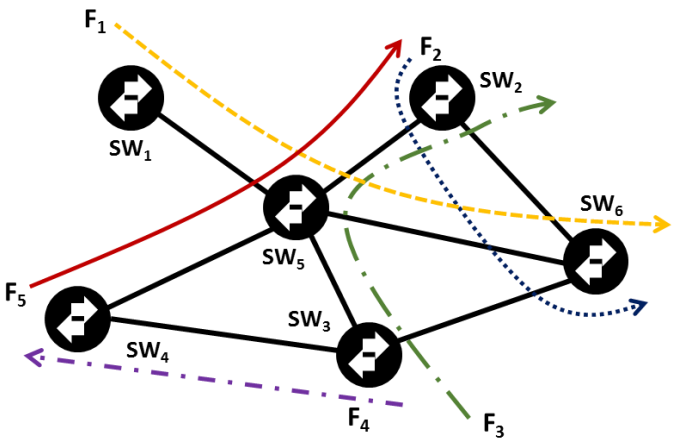
rate, high-performance capturing devices should be used for full-rate traffic capturing. However, it would not be possible to identify each traffic flow at the core switches without missing packets in real-time. Note that the number of devices used for traffic capturing is usually smaller than that required for traffic capturing at edge switches, and they can be more easily managed in a centralized manner. In addition, as shown in Fig. 1, at the core switches, it would not be possible to capture those traffic flows that only pass through edge switches. Moreover, some traffic flows may be unnecessarily captured more than once at intermediate switches.

Figure 2 shows SDN-based traffic capturing. SDN technology provides more flexible traffic monitoring by mirroring and rerouting the traffic flows of interest. SDNs do not necessitate discrimination between edge switches and core switches. SDN-based traffic capturing obviates the need to exploit hardware-based capturing devices; this capturing method relies on the OF protocol to capture traffic packets. In practice, this simply involves updating the flow table of each switch using the OF protocol. In [6], a fast traffic monitoring architecture (named Planck), which leverages the port mirroring feature of switches, was proposed. A high sampling rate is achieved by dividing N ports in a switch into m monitor ports and (N-m) data ports, and data traffic is forwarded to one of the monitoring ports. However, instead of using the mirroring feature of the switch, if it is necessary to monitor a specific traffic flow and capture data packets belonging to the particular flow on SDN, the controller can simply duplicate the traffic flow and forward the flow to a specific port of the switch by using the "OUTPUT" action of the OF protocol. Once the duplicated traffic is forwarded to the port, it can be steered to a traffic collector by the SDN controller as for a normal traffic flow.

IV. Probabilistic Packet Sampling

Probabilistic packet sampling is widely used in both PPC and FPC for traffic monitoring to avoid excessive traffic overhead on the network due to traffic duplication. Instead of capturing every packet passing through a switch, it selectively captures a certain number of packets from the traffic flows according to a sampling policy. In [7], several sampling policies were proposed and discussed in detail. Among them, the **systematic packet sampling** (SPS) method captures every packet for a sampling duration from a starting point in time, and **probabilistic packet sampling** (PPS) is a method to selectively capture packets from traffic flows with a uniform or non-uniform probability p. For example, if p is 10 % for PPS, only 10 % of packets are captured and the remaining 90 % are simply discarded. In [8], an OpenFlow extension (named FleXam) was proposed to support both SPS and PPS methods with a variety of sampling options. In [9], OpenSample was proposed to exploit the probabilistic packet sampling of the sFlow protocol to achieve a fast flow-level traffic statistics measurement on SDNs. Figure 2 shows an example of probabilistic packet sampling on an SDN. For a given set of sampling rates, the SDN controller updates the flow table of each switch using the OF protocol. Then, the sampled traffic is steered towards the traffic collector. As mentioned, since the sampled traffic flows are newly generated flows, the SDN controller has to compute less-congested flow paths to the traffic collector and update the flow table of switches using the computed routing paths.

The use of traffic-sampling techniques can lead to a significant reduction in network overhead because sampled traffic duplication can be significantly decreased. However, there exists the risk of useful information not being acquired from the discarded packets. Therefore, it is crucial to appropriately decide the sampling rates for traffic monitoring on the network. Two different approaches can be considered to decide the sampling rate: **per-flow (PF) sampling** and **per-switch (PS) sampling**. First, a different sampling rate can be applied to each flow. It is possible to adjust the sampling rate for each flow; therefore, fine-grained packet sampling is possible by increasing and decreasing the sampling rate. However, per-flow sampling of this nature may not be scalable with respect to the number of flows on the network because frequent updating of the flow table consumes substantial network resources, and the memory space reserved for the flow table in a switch is limited. In addition, the implementation and running complexities for per-flow operations are also significant [10]. Alternatively, each switch could use the same sampling rate for every flow that passes through the switch. Since the number of switches on a network does not change dynamically, the operational complexity is much lower than that of per-flow sampling. However, if the sampling rates are not properly determined, some flows are either excessively sampled at multiple switches or not sampled at all. The sampling rate decision for the per-switch sampling rates that approximate per-flow sampling rates is a challenging problem with high computational complexity.



Flow Information Matrix (*M*)

	SW_1	SW_2	SW_3	SW_4	SW_5	SW_6
F_1	1	0	0	0	1	1
F_2	0	1	0	0	0	1
F_3	0	1	1	0	1	0
F_4	0	0	1	1	0	0
F_5	0	1	0	1	1	0

Fig. 3 Simple network topology and the corresponding flow information matrix.

For malicious traffic inspection, we consider a probabilistic full-packet sampling on SDNs. If a larger number of traffic packets on the network were to be sampled with large sampling rates, it would be possible to acquire a higher level of traffic information for cyber security. However, it is more desirable to restrict the value of sampling rates for the following reasons:

Network overhead: As the sampling rates increase, the amount of sampled traffic increases proportionally. Because the sampled traffic is forwarded to the traffic collector, it consumes network resources and may incur network congestion, which interferes with the data delivery of normal traffic flows. Therefore, the number of duplicated packets for traffic sampling should be maintained as small as possible for less intrusive traffic monitoring.

Limited analysis capability: Traffic inspection is conducted by analyzing the sampled traffic by security applications such as IDS, which usually has limited processing capability. If the rate of incoming traffic to the IDS exceeds its capability, the IDS cannot process the incoming traffic without dropping packets. Therefore, the amount of sampled traffic should preferably not exceed the processing capacity of the IDS.

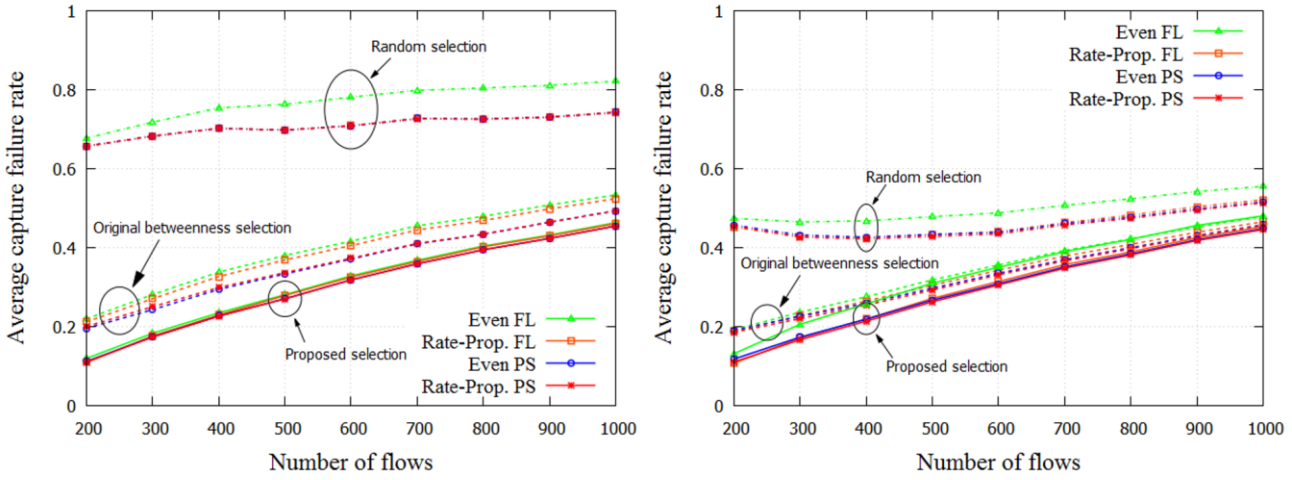
V. Sampling Point and Rate Decision

A. *Sampling point decision*

We propose a scalable packet-sampling point decision scheme using a graph theoretic centrality measure, which qualifies the relative importance of switches on the network. Although packet samplings in every OF-enabled switch are possible using OF on SDN, it is not desirable in a large-scale network, because it may incur overhead caused by flow-table processing at the SDN controller. Instead, a subset of switches can be selected to perform packet sampling. In graph theory, centrality measures such as degree, closeness, and betweenness centrality indicate the relative importance of vertices in the graph. The importance of a particular vertex can be quantified based on its topological relationship among the other vertices such as the number of neighboring vertices and the number of edges required to reach each of the other vertices. The concept of centrality measures has been applied to various areas such as social networking to find the most influential person.

When computing the centrality measures, one may use the same network topology as for physical links. However, that approach may reflect the physical network topology itself rather than the characteristics of flows. For traffic monitoring, it would be more effective to use only active links that currently serve the traffic flows rather than all the physical links in the entire network. On SDNs, information about the flow paths is readily available by using SDN northbound APIs. Furthermore, it is more computationally efficient to calculate the centrality measures using the number of active links rather than all the physical links.

We propose the use of the betweenness centrality for sampling point decision, which is defined as the number of shortest paths that pass through the switch for all the node pairs [11]. If any pair of nodes has k possible shortest paths, each possible path is counted by $1/k$ rather than 1 in the computation of the betweenness centrality. The betweenness centrality of a switch is simply obtained by the number of flows that pass through the switch. Figure 3 shows a simple network topology with six switches and six flows and its corresponding flow information matrix. The betweenness centrality can be easily computed by the flow information provided by the SDN controller. Let $\boldsymbol{M}$ denote the flow information matrix $\boldsymbol{M} = [m_{ij}]$, where m_{ij} is a binary number. If the i-th flow passes through the j-th switch, $m_{ij} = 1$; otherwise, $m_{ij} = 0$. Note that the dimension of M is the number of flows by the number of switches. The betweenness centrality for the j-th switch$c_j = \sum_i m_{ij}$, (i.e., the summation of the elements at the j-th column). The betweenness centralities for the switches are given by 1, 3, 2, 2, 3, and 2.



(a) Transit-stub topology (b) Mesh topology

Fig. 4 Average capture failure rate on transit-stub and mesh topologies.

In addition to the original betweenness centrality, we propose a new extended betweenness centrality to avoid an excessive sampling of traffic flows passing through a few bottleneck switches with high betweenness centrality. For example, if a number of flows pass through multiple bottleneck switches, the switches on the path would have a high betweenness centrality and the flows would be unnecessarily sampled multiple times at the switches. Note that, in general, switches located near the core of the network have a high betweenness centrality. The extended betweenness centrality is iteratively computed. Given $\boldsymbol{M}$, a single switch with a highest betweenness centrality is selected. Then, all the flows that go through the selected switch are excluded from $\boldsymbol{M}$, and this procedure is repeated with the updated $\boldsymbol{M'}$ until there are no remaining flows on the flow information matrix.

B. *Sampling rate decision*

Once the sampling points are selected, each switch has to be allocated a sampling probability. It is also worthy to note that the aggregated volume of sampled packets is retained below the maximum IDS capability of C in bits/s. Let $\boldsymbol{x}$ denote the switch sampling probability vector, where the k-th element of $\boldsymbol{x}$ is the sampling probability of the k-th switch. In addition, let $\boldsymbol{r}$ and $\boldsymbol{A}$ denote the flow-rate vector $\boldsymbol{r} = [r_i]$, where r_i is the data rate of the i-th flow and the flow-rate information matrix $\boldsymbol{A} = [a_{ij}]$, where $a_{ij} = r_i \cdot m_{ij}$, respectively. We consider two possible choices for per-switch sampling. First, $\boldsymbol{x}$ can be set by $x_k = \frac{1}{\sum_i a_{ij}} \cdot \frac{C}{\text{\# of switches}}$ such that each switch may sample traffic packets evenly at the same rate, which is equal to C divided by the number of switches. Second, every switch may have the same sampling probability, (i.e., $x_k = \frac{C}{\sum_i \sum_j a_{ij}}$). In this case, the rate of sampled traffic at each switch is proportional to the aggregated traffic rate passing through the switch.

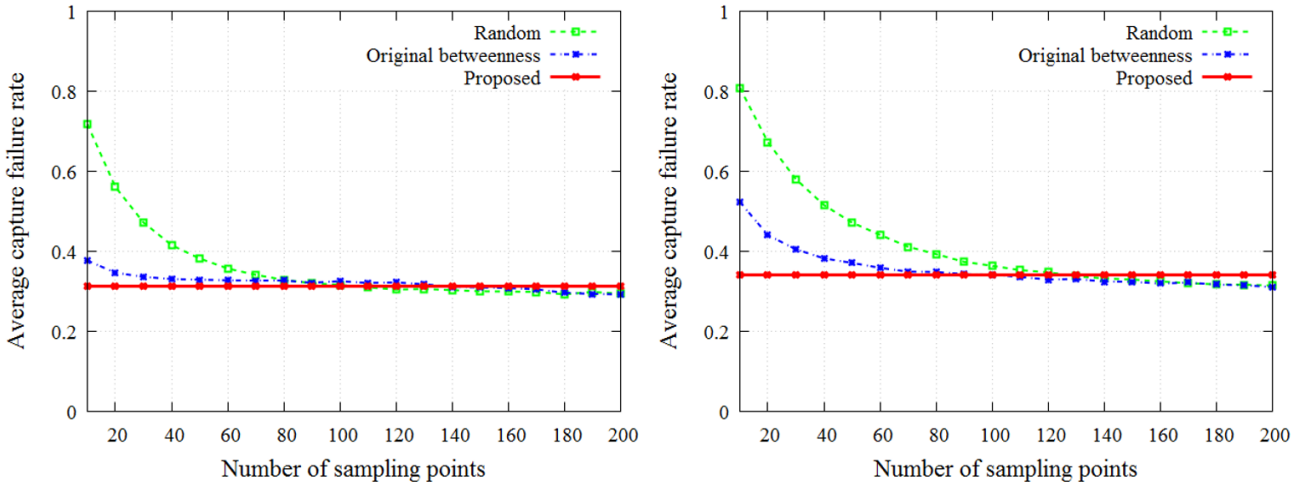
In addition, it is also possible to arbitrarily control the sampling of each individual flow by assigning a different sampling probability to each switch. This **flow-level sampling** is considered as a per-switch sampling technique that can approximate per-flow sampling. Let $\boldsymbol{b}$ denote the target sampling rate vector for the flows for the flow-level sampling. Then, the sampling rate vector $\boldsymbol{x}$ is obtained by a solution that satisfies $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$. As in the above per-switch sampling, each flow is sampled either evenly at the same rate by setting $\boldsymbol{b} = \frac{C}{\text{\# of flows}} \cdot \vec{1}$ or at a rate that is proportional to its traffic rate by setting $\boldsymbol{b} = \frac{C}{\sum_j r_j} \cdot \boldsymbol{r}$. For the flow-level sampling, the solution $\boldsymbol{x}$ can be obtained by using the pseudo-inverse of $\boldsymbol{A}$, (i.e. $\boldsymbol{x} = (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}$).

VI. Simulation

To evaluate the traffic sampling performance, we conducted simulations using BRITE[5] for generating a large-scale topology, and Network Simulator 2 (ns-2) to simulate the network flows for malicious packet detection scenarios. We created two types of topologies, i.e., clustered transit-stub and grid-type mesh topologies. Each network has 200 switches, and the number of flows varies from 200 to 1000. Each flow has a random data rate from 1 to 100 Mb/s, and it is assumed that 3% of total flows are malicious. The IDS capacity for malicious packet inspection is fixed at 1 Gb/s. The reported values are the averages of 1000 simulation runs. Regarding the sampling point decision, we compare the extended betweenness centrality-based sampling point decision method with the original betweenness centrality-based sampling point decision and random point decision methods. For each sampling rate decision, four sampling rate decision methods are evaluated: even per-switch (PS) sampling, rate-proportional PS sampling, even flow-level (FL) sampling, and rate-proportional FL sampling.

Figure 4 shows the average capture failure rates of malicious flows on transit-stub and mesh topologies. The capture failure rate is computed under the assumption that the traffic rates of normal and malicious flows are given [12]. The number of sampling points is given by the proposed extended betweenness centrality-based method, and the other methods select the same number of sampling points. As the number of flows increases,

[5] BRITE (Boston University Representative Internet Topology Generator): https://www.cs.bu.edu/brite



(a) Transit-stub topology (b) Mesh topology

Fig. 5 Performance comparison with respect to the number of sampling points.

the capture failure rate increases because the number of captured packets belonging to each flow decreases. Note that the aggregate rate of sampled traffic is fixed at the inspection capacity of 1 Gb/s. The number of sampling points only includes 15 switches in the transit-stub topology and varies from 35 to 60 switches in the mesh topology. The results show that the proposed sampling point selection algorithm provides the lowest capture failure rates among the three sampling point decision methods. Regarding the sampling rate decision method, the flow-level sampling methods perform more effectively than the per-switch sampling methods in most cases. However, when the sampling points are determined by the proposed extended betweenness centrality-based method, the three sampling rate decision methods show almost the same capture failure rate.

Figure 5 shows the average capture failure rate for even FL sampling when the number of sampling points varies for the random selection and original betweenness centrality-based methods. The number of sampling points for the proposed method was 15 in the transit-stub topology and 55 out of 200 switches in mesh topology. The number of flows is 700 in each topology. Since the probability that a flow will pass through core switches connecting mesh topologies in the transit-stub topology is high, this topology has fewer sampling points. The result indicates that the proposed algorithm can achieve almost the same performance with a much smaller number of sampling points compared to the other methods.

VII. Experiment

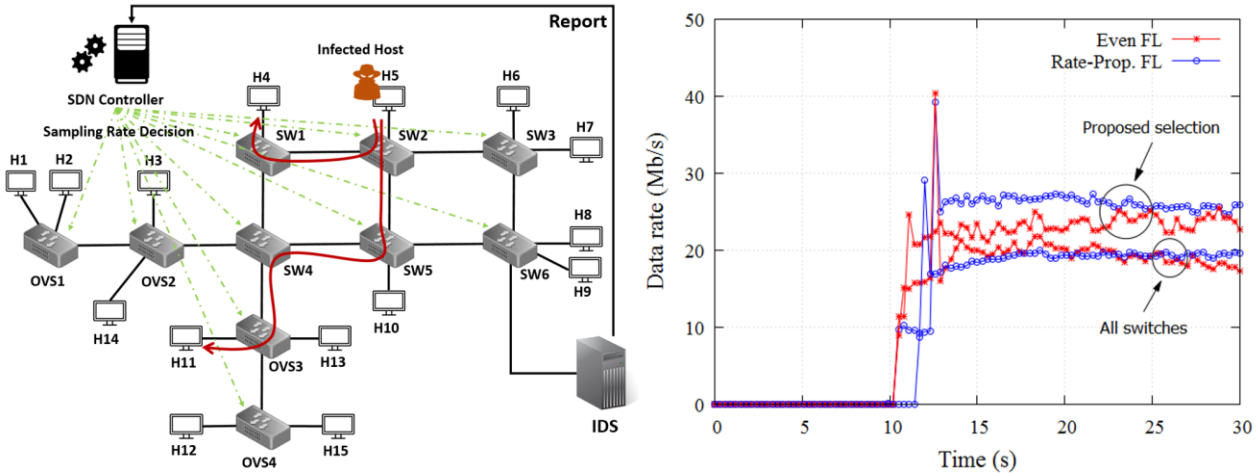
We constructed an SDN-enabled testbed to evaluate the traffic sampling performance. We consider signature-based ransomware propagation detection using IDS. Because some ransomware attacks use malicious toolkits such as Angler, Neutrino, and RIG [13], their propagation can be detected by inspecting whether the captured packets include the malicious toolkit.

Figure 6(a) illustrates the topology of our SDN-enabled testbed. It consists of six HP 2920 OF-enabled switches (Open-Flow 1.3 supported), four Open vSwitches (OVSs) running on Linux embedded boxes, two HP workstations (one HP workstation for the SDN controller and the other for IDS), and 15 host PCs. The SDN controller is configured with the helium version of OpenDaylight. Snort IDS is used to inspect the traffic for detecting malicious attacks. Snort is an open-source IDS that inspects network traffic using a variety of rulesets and generates security alarms when suspicious network activities are detected. In the experiment, the number of flows is 15, and their data rate varies from 5 to 50 Mb/s. Two malicious flows are added with a rate of 30 and 40 Mb/s, respectively. The IDS detects the ransomware propagation using Snort rulesets for malicious toolkit signatures. The SDN controller updates the sampling probabilities of OF-enabled switches either when it receives an 'OFPT_PACKET_IN' message for newly added flows or when it detects changes in the data rate of current data flows in service.

Figure 6(b) shows the rates of malicious traffic forwarded to the IDS. Initially, there are 15 flows. After 10 seconds, two malicious flows are generated by attackers. The SDN controller detects the two new flows and calculates the sampling points and rates for the switches. The traffic sampling is stabilized in 2 seconds as shown in Fig. 6(b). The proposed sampling point decision method captures malicious flows at higher data rates than the method that samples from every switch. It is also observed that rate-proportional FL sampling achieves higher data rates than even FL sampling in this network configuration.

VIII. Conclusion

Network traffic monitoring plays an increasingly important role in cyber security. Unlike traditional networking systems, the SDN technology provides programmable functionalities that enable OF-enabled switches to perform probabilistic traffic sampling and to steer the sampled traffic towards a traffic collector for network traffic inspection. In this article, we focused on a scalable traffic sampling point and rate decision scheme that uses a centrality measure to allow the network traffic to be secured with low monitoring overhead. The simulation and experimental results demonstrated that the proposed method achieves less-intrusive monitoring and decreases the malicious flow capture failure rate in a scalable manner.



(a) SDN testbed topology (b) Captured malicious traffic rate

Fig. 6 SDN testbed topology and the rates of captured malicious traffic.

REFERENCES

[1] K. Cabaj and W. Mazurczyk, Using Software-Defined Networking for Ransomware Mitigation: The Case of CryptoWall, *IEEE Network*, November/December 2016, pp. 12-19.

[2] N. Scaife *et al.*, CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data, *Proc. IEEE Int'l Conf. Distrib. Comp. Sys.*, June 2016, pp. 303-12.

[3] N. L. M. van Adrichem *et al.*, OpenNetMon: Network Monitoring in Openflow Software-Defined Networks, *Network Operations and Management Symposium*, May 2014.

[4] K. Gioties *et al.*, Combining OpenFlow and sFlow for an Effective and Scalable Anomaly Detection and Mitigation Mechanism on SDN Environments, *Elsevier Computer Networks*, vol. 62, December 2013, pp. 122–36.

[5] C. Xu *et al.*, A Survey on Regular Expression Matching for Deep Packet Inspection: Applications, Algorithms, and Hardware Platforms, *IEEE Commun. Surveys & Tutorials*, vol. 18, May 2016, pp. 2991-3029.

[6] J. Rasely *et al.*, Planck: Millisecond-scale Monitoring and Control for Commodity Networks, *Proc. ACM conf. SIGCOMM*, August 2014, pp. 407-18.

[7] M. Korczynski *et al.*, An Accurate Sampling Scheme for Detecting SYN Flooding Attacks and Portscans, *IEEE Int'l. Conf. Commun.*, June 2011, pp. 1-5.

[8] S. Shirali-Shahreza and Y. Ganjali, FleXam: Flexible Sampling Extension for Monitoring and Security Applications in OpenFlow, *Proc. ACM SIGCOMM Wksp. Hot Topics in Software Defined Networking*, August 2013, pp. 167–8.

[9] J. Suh *et al.*, OpenSample: A Low-latency, Sampling-based Measurement Platform for Commodity SDN, *Proc. IEEE Int'l Conf. Distrib. Comp. Sys.*, July 2014, pp. 228-37.

[10] Y. Liu *et al.*, On the Resource Trade-off of Flow Update in Software-Defined Networks. *IEEE Commun. Mag.*, vol. 54, June 2016, pp. 88–93.

[11] L. C. Freeman, A Set of Measures of Centrality based on Betweenness, *Sociometry*, vol. 40, March 1977, pp. 35–41.

[12] T. Ha *et al.*, Suspicious Traffic Sampling for Intrusion Detection in Software-Defined Networks, *Elsevier Computer Networks*, vol. 109, November 2016, pp 172-82.

[13] R. Brewer, Ransomware Attacks: Detection, Prevention and Cure, *Elsevier Network Security*, vol. 2016, September 2016, pp. 5-9.