Zhou, Donghao; Yan, Zheng; Liu, Gao; Atiquzzaman, Mohammed

An Adaptive Network Data Collection System in SDN

Published in:
IEEE Transactions on Cognitive Communications and Networking

DOI:
10.1109/TCCN.2019.2956141

Published: 01/06/2020

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Zhou, D., Yan, Z., Liu, G., & Atiquzzaman, M. (2020). An Adaptive Network Data Collection System in SDN. *IEEE Transactions on Cognitive Communications and Networking*, *6*(2), 562-574. Article 8915764. https://doi.org/10.1109/TCCN.2019.2956141

An Adaptive Network Data Collection System in SDN

Donghao Zhou, Zheng Yan, *Senior Member*, *IEEE*, Gao Liu, and Mohammed Atiquzzaman, *Senior Member*, *IEEE*

***Abstract*—Network data collection is a vital part in the process of network monitoring, traffic billing, network management and intrusion detection. As a new kind of network architecture, Software Defined Network (SDN) provides a possibility of intelligent and adaptive network data collection with centralized control and programming. However, existing literatures lack a concrete solution to economically collect network data, while satisfying the quality of data processing and analytics. Current data collection methods are not sufficiently adaptive and intelligent in terms of network context awareness. In this paper, we propose an adaptive network data collection system in SDN by automatically selecting proper data collection nodes based on network status in a dynamic way. During data collection, network traffic is sampled by considering flow characteristics in order to effectively reduce the amount of collected data while ensuring the accuracy of later data analysis, e.g., malicious traffic detection. A series of experiments are conducted to test and verify the data collection system and show its advantages through comparison with existing works in terms of CPU/memory consumption, storage usage, flow size recovery, and threat perception.**

***Index Terms*—SDN, network data collection, traffic characteristics.**

I. Introduction

With the development of the Internet, 5G and IoT, more and more network devices connect to networks, which create massive network data. Without any doubt, these data greatly assist network administrators to understand the network environment and evaluate its Quality of Service (QoS) [1]. For example, they can help understanding the temporal and spatial distribution of network traffic, analyzing the performance of network links and nodes, and knowing the distribution of bottlenecks and compromised nodes. Therefore, network fault location, traffic visualization [2], network routing optimization [3], [4], attack detection and mitigation [5]–[9] can be realized accordingly. Network data collection has become a crucial and essential part in network management and attack detection.

Although there are many big data processing techniques (e.g., machine learning and deep learning) helping massive network traffic data processing, current network data collection is still confronting challenges. If all original data are collected directly at every network node, the efficiency of data processing and analytics is hard to be achieved. Compressing and sampling data can release this problem, but the accuracy of further analysis might be influenced. Therefore, it is particularly significant to study efficient and effective data collection methods to collect as few as possible network traffic data in real time and at the same time to ensure the quality of later data processing and analytics.

However, current data collection methods are not sufficiently adaptive and intelligent [16], [25]. Complex manual modification and configuration lead to high error rate in network monitoring, controlling, evaluating and traffic accounting. They can hardly identify network situation and targeted data according to packet rate, network security level, attack types, and the purpose of data usage. While network data collectors execute the tasks of data collection, the security and availability of the network must be ensured. Unfortunately, proper security policies and data collection strategy to detect and mitigate attacks are still open to be investigated. Traditional networks lack network awareness and do not have the capability of programming for adaptive and intelligent data collection [10], [11].

Concretely, there are still some open issues in the process of network data collection. First, traditional networks cannot be aware of global topology and adaptively adjust data collection strategy in real time [26]. Corresponding policies are managed and changed by a network manager manually. Second, prior data collection methods do not balance between collection and resource consumption [14]–[16]. Third, the requirement of adaptivity of network data collection to network and traffic status is hard to meet by prior arts [17]–[21]. Fourth, the process of network data collection may influence normal network traffic, the quality of networking service could be impacted by it. How to avoid such an influence of data collection on the quality of networking service should be well considered. Traditional network is lack of network situation awareness and does not have the capability of programming. Therefore, accurate, real-time and effective data collection is difficult to achieve.

Software Defined Network (SDN) achieves the separation of traffic control and forwarding, abstracts the control functions of a network from data plane and infrastructure, and enables network managers to concentrate on the logical realization of network functions. At the same time, the logical centralization of SDN architecture makes network management more flexible and efficient in terms of network perception, management and resource allocation. SDN provides logically centralized control and open programming interfaces, which facilitates the adaptability of network data collection and enables the collection to be performed according to traffic and network

D. Zhou and G. Liu are with the State Key Laboratory on Integrated Services Networks and the School of Cyber Engineering, Xidian University, Xi'an, 710126 China

Z. Yan is with the State Key Laboratory on Integrated Services Networks and the School of Cyber Engineering, Xidian University, Xi'an, 710126 China, and with the Department of Communications and Networking, Aalto University, Espoo, Finland e-mail: zyan@xidian.edu.cn.

M. Atiquzzaman is with the School of Computer Science, The University of Oklahoma, Norman, OK 73019, USA

status. However, based on our thorough surveys [17], [26], the literature still lacks a concrete solution to economically collect network data, while satisfying the needs of data processing and analysis at the same time.

In this paper, we propose an adaptive network data collection system in SDN, aiming to answer such questions as what data, where, how and when to collect them. In this paper, we propose an adaptive network data collection system in SDN, aiming to answer such questions as what data, where, how and when to collect them. Through network status evaluation, the system adaptively selects proper collection nodes for the purpose of reducing the redundancy of collected data and ensuring complete traffic perception at the same time. During data collection, it employs dynamic probability generation and in-flow sampling to determine sampling and collection rules for further reducing the redundancy of collected traffic data, but simultaneously ensuring the accuracy of attack detection.

We propose a node selection module to decide the nodes for network data collection. According to a network topology and the availability of nodes and links, we find all blocked nodes in the network topology in order to figure out root nodes of blocked traffic, thus we can decide data collection nodes (i.e., blocked root nodes) in abnormal status. In case that the network status is normal, the node selection module selects central nodes of link intersection and access nodes of an island as data collection nodes. As an intelligent and distributed method for data collection, our system helps collecting, detecting and mitigating malicious traffic in blocked root nodes. It avoids collecting data in all influenced nodes, thus improving collection efficacy. We further design an adaptive data collection algorithm to reduce collected data volume at an individual collection node. The algorithm first divides an online flow into a number of segments. The length of the segment is increased with the increase of its order number. Then, sampling with a probability adjusted based on the flow length is applied in data collection. The sampling probability is decreased with the length of flow increasing. Thus, our collection method can reduce redundant collected data for elephant flows and increase perception on mice flows. This design is effective in time-sensitive circumstances, e.g., when DDoS attacks occur. It also considers the content information of a flow in order to achieve accurate and high-quality data collection by identifying the differences between flows. In summary, by perceiving network status, data collection nodes are selected adaptively based on network topology in order to decide where to collect data. Then, they sample packets from traffic flows with dynamic probability in order to reduce the volume of collected data, thus we figure out how and when to collect data and what data should be collected.

To be specific, the contributions of this paper can be summarized as below.

1) We design and develop an adaptive network data collection system in SDN, which is one of the first to realize network-status-aware data collection in SDN. It can dynamically and adaptively select a small number of nodes as data collectors based on network status. It can also properly sample segments from flows to achieve high efficiency and low resource consumption of data collection. Meanwhile, it helps detecting sensitive network events (e.g., link flooding attacks) with high accuracy.

2) We propose a network measurement and evaluation model to quantify the status of a network node and a link.

3) We present a node selection algorithm to choose appropriate nodes for network data collection based on network topology and traffic distribution, as well as the status of a network node and a link. Instead of collecting data from all nodes, our system avoids collected data redundancies and incomplete traffic perception.

4) We further propose dynamic probability generation and in-flow sampling that can determine appropriate sampling and collection rules according to the time distribution and flow distribution of traffic. It not only effectively reduces the redundancy of data in collection, but also has time-sensitive capability and the ability of timely sensing traffic in a malicious environment, e.g., during DDoS attacks.

The rest of this paper is organized as follows. Section II briefly overviews background and related work. Section III presents our proposed system architecture and detailed system design, followed by system test and evaluation in Section IV. Section V concludes the paper.

II. Background and Related Work

In this section, we briefly describe the current state of art of network data collection.

Network Data Collection

Libpcap is a powerful open source software library, which provides a system-independent and user-level network data packet capture interface. It is the most common functional component in network data collection. Many network monitoring software (e.g., Wireshark, Tcpdump) and packet-based collection methods rely on libpcap. However, Libpcap cannot solve the open problems listed in the introduction.

NetFlow, a classical network data collection method, provides the capability of packet-based collection on switches. However, with the increase of network data, NetFlow overwhelms switches due to resource constraints. Although some mitigation methods were proposed [14]–[16], they cannot achieve ideal balance between resource consumption and data collection, especially when DDoS attacks occur. NetFlow reduces the scale of collected data by sampling. However, with the increase of peak rate of DDoS, this method is prone to overwhelm switches.

Sketch-based data collection [17]–[21] is widely used in network data collection and network measurement. In [19], Yu et al. proposed a data collection framework named OpenSketch. It records the number of packets per flow with multiple hash tables instead of collecting packets or flows directly. It provides three unique channels for hashing, filtering and counting three kinds of messages and realizes fast storage. In [18], Huang et al. combined a sketch-based mechanism and a counter-based mechanism to adapt to the flow distribution of elephant flows for such flows' detection. According to the characteristics of SDN, Huang et al. improved the adaptability of overload flow detection [20] and flow size estimation [21] in sketch-based data collection.

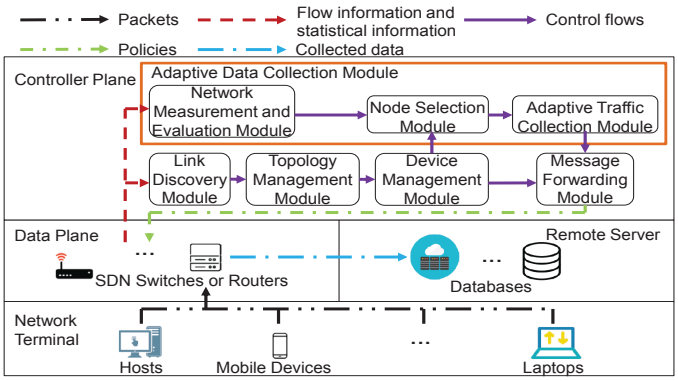


Fig. 1. System Architecture.

Sketch-based methods not only aggregate network data, but also help detecting DDoS attacks caused by elephant flows with high accuracy. However, they have three main flaws. First, they are not available and effective for other network attacks. Second, sketches are usually deployed in some fixed nodes such as gateway, which is hard to sense the global traffic status in a network. Third, sketches lack capacity of tracing, which make this kind of methods take the risk of the false positive due to hash collisions. Jing et al. [27], [28] proposed novel reversible sketch-based methods based on Chinese Remainder Theorem for collecting and storing network data. They used the compressed data to detect and mitigate amplification attacks and DDoS flooding attacks. But their methods do not solve the questions on where, how and when to collect data in an adaptive and automatic way.

Flow-based data collection is another common way of collecting network data. Based on the prominent role played by elephant flows in network monitoring, Jose et al. detected and identified convergent flows with a small number of flow tables in switches [22]. As a result, the adaptivity of the distribution of elephant flow was achieved during data collection. However, only intersection nodes are chosen to collect network data, which ignores adaptation of collection positions. In [23], Flowsense was proposed. It relies on passive detection at switches. According to each Flow-Removed message from OpenFlow switches, a data collection module in controller is triggered. It collects all network data when flows come in, which might lead to a heavy overload. In addition, hysteresis is another problem in real-time scenarios. In [6], [8], two network data collection methods based on active measurement were proposed. However, both methods only filter flow, which are coarse-grained. Meanwhile, they consume many computing and storage resources in a high-speed network and in attack scenarios due to a high sampling frequency. Chowdhury et al. proposed PayLess [29], which adaptively aggregates statistics collection. However, a high volume of network data is collected in the case that a large amount of traffic occurs, since the distribution of traffic flows is not considered. Therefore, PayLess brings about many collected data redundancies.

Based on the above literature review and many literature surveys [35]–[37], we can see that it lacks an effective data collection method in SDN that can dynamically and adaptively select a small number of nodes as data collectors based on network status in order to collect as few as possible data, but at the same time, the collection still helps detecting sensitive network events with high accuracy. Therefore, in this paper, we aim to propose such a data collection system in SDN.

III. System Design

This section describes the system model, the system overview, the model of network measurement and evaluation, the mechanisms of node selection and adaptive data collection.

A. System Model

SDN is an important network architecture that simplifies network management. In SDN, the network control is separated from the data forwarding plane. The former is logically centralized, and the latter performs strategies from the former. Thus, we can implement new control functions in SDN by writing software logic in the control plane. As a result, the control plane can help SDN to support different services and achieve various applications [12], [13]. As shown in Fig. 1, the system is composed of four parts: network terminal, data plane, controller plane and remote server. The SDN controller analyzes and processes network packets and flow information.

Network Terminal: It is the user to visit a network. During visiting, SDN switches in data plane will receive and forward its packets.

Data Plane: It is responsible for matching and forwarding packets passing through SDN networks. The data plane consists of SDN switches. Switches record flow information and statistical information in flow tables. The controller plane can request these records and ask switches to execute their policies.

Controller Plane: It is the core part of an SDN network. All control policies are sent from this plane. In the proposed system, adaptive data collection is realized by link discovery module, topology management module, device management module, adaptive data collection module and message forwarding module. Collection nodes in open networks are not trusted, since they might be compromised or behave maliciously. The data collected from untrusted collection nodes might disturb adaptive data collection and collected data analytics. In order to achieve trustworthy data collection, the controller can adopt some trust evaluation and sustainment mechanisms [38]–[40] to evaluate the credibility of collection platforms and assure their trustworthiness as expectation for ensuring the trust of collected data. Once the trusted environments in collection nodes are destroyed, the controller can discover these nodes and apply correct collection strategies to exclude the data provided by them. However, trustworthy data collection is beyond the scope of this paper.

Link Discovery Module: This module is responsible for link discovery and maintenance of link state in networks. The module asks switches alive to respond its requests. By this way, this module keeps and updates the state of links.

Topology Management Module: It is responsible for discovering network connections and island partitions based

on topology computing. SDN is not only deployed in all SDN-based networks. There are many hybrid application scenarios with SDN and non-SDN devices. There exist islands that consist of directly connected SDN routers. The router connecting with non-SDN node is the access router (AR). It's the entrance and export of an SDN island that communicates with non-SDN networks. The topology manager module computes the distance of SDN routers to judge if there is an island and what the topology is.

Device Management Module: It identifies Virtual Local Area Network (VLAN) and Media Access Control (MAC) address for network devices and defines device addresses.

Network Measurement and Evaluation Module: Firstly, the module measures the factors of link availability: packet loss, delay and occupied bandwidth, and obtains the factors of node availability from the historical data of nodes: the lasting time of denial of service (DoS) attack and malicious packet rate. Secondly, through the availability factors of links and nodes, the module evaluates the availability of links. Then, corresponding evaluation results can be obtained.

Node Selection Module: This module finds all central nodes and access nodes of the network according to the island partition and link topology. Then, the module judges the availability of nodes and links. As a result, this module applies the node selection algorithm to find appropriate data collection nodes in order to reduce the amount of data collected and ensure the integrity of traffic as far as possible.

Adaptive Traffic Collection Module: This module depends on the node selection module for data collection. The module only collects data from the selected nodes. According to the characteristics of traffic, an adaptive data collection algorithm is applied to sample the flow in order to reduce the amount of redundant data. The characteristics of traffic are various. They can be packet rate, flow distribution, and so on.

Message Forwarding Module: This module is responsible for generating the policies to forward packets between devices.

Remote Server: It is used to store the collected data for further processing.

B. System Overview

The network terminals (e.g., hosts, laptops and mobile devices) enter SDN networks through network connection. An SDN switch in the data plane reports its own statistical information and stores flow information into the SDN controller. The monitoring mechanism of PACKET_IN message in SDN provides the controller good awareness about flows. The system obtains the arrival time of flow from the PACKET_IN message and the statistical message from the switch.

In the controller plane, the link discovery module, the topology management module and the device management module get the information of network topology, equipment and links from PACKET_IN messages, which helps the controllers to sense network topology changes. The network measurement and evaluation module is responsible for matching specific PACKET_IN messages to sense topological traffic and traffic changes. Then it realizes the measurement and evaluation on time delay, packet loss rate and bandwidth. According to the network topology information and network evaluation results, the node selection module determines whether the network is in a blocking state. If it is in this state, the module finds the blocking roots in the blocking topology as collection nodes. Otherwise, the central nodes of the link intersection and access nodes of network islands are selected. When the node selection is finished, the adaptive network traffic collection module adopts different sampling algorithms on new and old flows in order to collect data with low redundancy. The module sends generated traffic sampling strategies to the forwarding module. The forwarding module generates flow table modification information, which is then sent to SDN switches through southward interface. The switches collect and save network data in a remote server.

In the following subsections, we introduce the core modules of the system, namely network measurement and evaluation module, node selection module and adaptive traffic collection module, since a traditional SDN has other function modules. We assume that the parameters (e.g., the low bound of a factor and its weight) can be set based on experiences or by using some methods, e.g., gradient descent, and we do not consider involved security and privacy issues, since our goal is to propose an adaptive data collection system in SDN.

C. Network Measurement and Evaluation Model

This subsection presents a link availability model and a node availability model that play important roles in evaluating network status.

Link Availability Model

Due to the continuity of network state, the current link availability is highly related to historical link availability. However, the longer the interval is, the less the correlation of current and historical link availability. The threshold number of interval slots is defined as K. Given time $t_i, t_j, j \geq i$, we can ignore the correlation when $j - i \geq K$ such that the correlation is less than a threshold δ. Q_t is the link availability at time t, and Q_{G_t} is defined as the global link availability at time t, which is related to previous K time slots. We use the global link availability to replace the availability at a single time slot to make decision. Because the link availability relates to current and historical availability with different weights. It gradually changes and is not affected by system instability. In addition, the procedures conducted by other modules are based on link availability. However, the modules cannot parallel their work, so that their processing time is not synchronized. The global link availability can be predicted, which helps the synchronization of different modules and improves the accuracy of their processing results. For getting the global link availability Q_{G_t} at time t, we need to evaluate the impact factors of instantaneous link availability.

1) Impact Factor Evaluation

Time delay (td), packet loss (pl) and bandwidth (bd) are the essential factors to evaluate link availability [24], [35]–[37]. Assume M_ω is the measurement value of factor ω. Thus, we can define evaluation value of td, pl and bd as M_{td}, M_{pl} and

M_{to}. Moreover, their superior bounds are M_{td_sl}, M_{pl_sl} and M_{to_sl} and their low bounds are M_{td_ll}, M_{pl_ll} and M_{to_ll}, respectively.

When the measurement value of td, pl and bd of a link is large than its superior bounds, the link is overloaded. When the measurement value of td, pl and bd of a link is smaller than its low bounds, the change of this factor has little influence on the link availability.

We define the evaluation value of td, pl and bd as E_{td}, E_{pl} and E_{bd}, where $E_\omega \in [0, 1]$.

$$\begin{cases} E_{td} = 1, M_{td} < M_{td\_ll} \\ E_{pl} = 1, M_{pl} < M_{pl\_ll} \\ E_{bd} = 1, M_{bd} < M_{bd\_ll} \end{cases} \tag{1}$$

$$\begin{cases} E_{td} = 0, M_{td} > M_{td\_sl} \\ E_{pl} = 0, M_{pl} > M_{pl\_sl} \\ E_{bd} = 0, M_{bd} > M_{bd\_sl} \end{cases} \tag{2}$$

When $M_{\omega\_ll} \le M_\omega \le M_{\omega\_sl}$, E_ω is decreased with the increase of M_ω. M_{td}, M_{pl} and M_{bd} can be normalized into $[0, 1]$ by using a formula $(M_\omega - M_{\omega\_ll})/(M_{\omega\_sl} - M_{\omega\_ll})$.

In previous studies, many algorithms were proposed to evaluate link availability with td, pl and bd. Generally, the evaluation algorithm based on absolute values is a common and effective evaluation method. It has gotten some good results in practice. But herein, Gaussian function is used instead of absolute values as the attenuation factor of exponential function. In theory and practice, it is proved to have better results than the absolute values in the following aspects:

1) The Gaussian function based evaluation value is more stable near the low and superior bounds. The evaluation value attenuation near the low and superior bounds is smaller compared to the absolute value based evaluation.

2) The Gaussian function based evaluation value decays sharply when it is far away from the lower bound, and the decaying rate is more obvious than that of absolute value based evaluation value. This feature can well present actual situation of network links.

As a result, we adopt the Gaussian function as our evaluation model when $M_{\omega\_ll} \le M_\omega \le M_{\omega\_sl}$, $E_\omega = e^{-2[(M_\omega - M_{\omega\_ll})\delta_\omega/(M_{\omega\_sl} - M_{\omega\_ll})]^2}$, where ω is equal to td, pl or bd, and δ_ω is a fixed value to control the variance of E_ω.

2) Instantaneous Link Availability

The instantaneous QoS is evaluated with td, pl and bd as follows. There are different requirements for td, pl and bd in different network and business scenarios. For instance, td and bd are important for game server clusters while pl is not so essential for them. However, the clerking business services require low pl but td while they are not the first thing to fulfill. Therefore, our system gives coefficients f_{td}, f_{pl}, f_{bd} to balance the importance of each factor, $f_{td} + f_{pl} + f_{bd} = 1$. Then we have

$$\begin{aligned} Q_t = {} & J_{td\_sl} f_{td} e^{-2J_{td\_ll}[(M_{td} - M_{td\_ll})\delta_{td}/(M_{td\_sl} - M_{td\_ll})]^2} \\ & + J_{pl\_sl} f_{pl} e^{-2J_{pl\_ll}[(M_{pl} - M_{pl\_ll})\delta_{pl}/(M_{pl\_sl} - M_{pl\_ll})]^2} \\ & + J_{bd\_sl} f_{bd} e^{-2J_{bd\_ll}[(M_{bd} - M_{bd\_ll})\delta_{bd}/(M_{bd\_sl} - M_{bd\_ll})]^2}, \end{aligned} \tag{3}$$

where

$$J_{\omega\_sl} = \begin{cases} 0, & M_\omega > M_{\omega\_sl} \\ 1, & M_\omega \le M_{\omega\_sl} \end{cases} \tag{4}$$

and

$$J_{\omega\_ll} = \begin{cases} 0, & M_\omega < M_{\omega\_ll} \\ 1, & M_\omega \ge M_{\omega\_ll} \end{cases} \tag{5}$$

We have calculated the evaluation of link availability at time t and the time before it. In the evaluation model with memory characteristics, there are two common evaluation methods, namely the evaluation method based on memory attenuation factor and the evaluation method based on time window. The memory attenuation factor based evaluation usually weights the historical time by an exponential function, which can realize the fast attenuation in continuous time. The time window based evaluation adopts linear weighting for evaluation values in previous slots. Compared with the exponential method, it is more stable and can provide a better evaluation with strong historical correlation. Because the time to collect data about the link is discrete, a great amount of link evaluation information is mastered, and the global availability is highly related to historical evaluation values. Therefore, the time window method is used to obtain the global availability value Q_{G_t} at time t. According to the link evaluation values Q_{t-K+1},Q_{t-K+2},Q_{t-1}, …, Q_t at the previous K time slots, we can get $Q_{G_t} = \sum_{k=0}^{k=K-1} (K-k) Q_{t-k} / \sum_{k=0}^{k=K-1} (K-k)$. Q_{G_t} is the global evaluation value of the link, which represents the availability of the link.

Node Availability Model

Denial of Service (DoS) and malicious behaviors are the most significant issues influencing the availability of network nodes. To evaluate the availability of network nodes, we must consider these issues to react security threats. Therefore, the time of DoS (ts) and malicious packet rate (mr) are proposed to evaluate the availability of network nodes. Similar to the evaluation of link availability, we evaluate the availability of a node S_t and its global availability S_{G_t} at time t. S_{G_t} is related to the availability of the node at previous K' time slots.

M_{ts} and M_{mr} are the measurement values of ts and mr, respectively. The superior bound of ts and mr is M_{ts_sl} and M_{mr_sl}, and their low bounds are M_{ts_ll} and M_{mr_ll}, respectively.

If the evaluation E_{ts} of ts and the evaluation E_{mr} of mr are in the scope of $[0, 1]$, we can obtain

$$\begin{cases} E_{ts} = 1, M_{ts} < M_{ts\_ll} \\ E_{mr} = 1, M_{mr} < M_{ts\_ll} \end{cases} \tag{6}$$

$$\begin{cases} E_{ts} = 0, M_{ts} > M_{ts\_sl} \\ E_{mr} = 0, M_{mr} > M_{mr\_sl} \end{cases} \tag{7}$$

When the measurement values of ts and mr and the capability of judgement are accurate enough, the low bound that is provided to adjust fault tolerance can be reduced. $M_{ts\_ll} = M_{mr\_ll} = 0$ is acceptable. However, the superior bound is not recommended to set as 1 in order to ensure a good capability to detect unavailable network nodes, security events and threats happened on the nodes as early as possible.

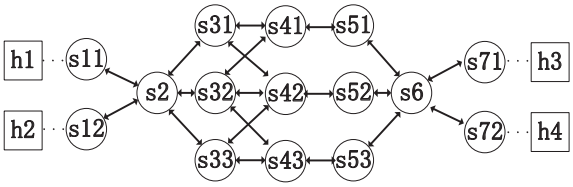


Fig. 2. Abstracted Graph of Network Topology.

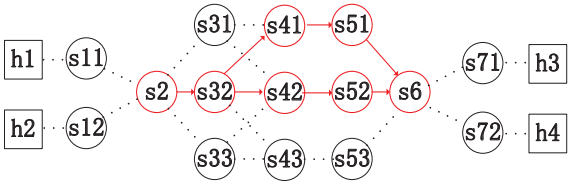


Fig. 3. Topology with Blocked Links.

Similar to the availability of network links, the evaluation of each factor is obtained by using the Gaussian function, $E_\omega = e^{-2[(M_\omega - M_{\omega\_ll})\delta_\omega/(M_{\omega\_sl}-M_{\omega\_ll})]^2}$, where ω is either ts or mr.

As mentioned above, the node availability evaluation is mainly based on two factors, i.e., ts and mr. Similar to link availability, the requirements of accuracy and sensitivity of node availability are diverse in different scenarios.

For the most scenarios of enterprise services, ts is a very important factor, which determines the stability and QoS. However, there are few concerns on mr. In other scenarios, mr is vital due to strict security requirements, such as the entrance of data center, firewalls, and Intrusion Detection Systems (IDS). To satisfy the requirements of different scenarios, we set adjustable weights for adapting to different scenarios, i.e., $f_{ts} + f_{mr} = 1$, where f_{ts} and f_{mr} are the weights of ts and mr, respectively. Thus, we can get the node evaluation S_t, $S_t = f_{ts}E_{ts} + f_{mr}E_{mr}$. Based on E_ω, we can obtain

$$\begin{aligned} S_t = {} & J_{ts\_sl} f_{ts} e^{-2J_{ts\_ll}[(M_{ts}-M_{ts\_ll})\delta_{ts}/(M_{ts\_sl}-M_{ts\_ll})]^2} \\ & + J_{mr\_sl} f_{mr} e^{-2J_{mr\_ll}[(M_{mr}-M_{mr\_ll})\delta_{mr}/(M_{mr\_sl}-M_{mr\_ll})]^2} \end{aligned} \tag{8}$$

where

$$J_{\omega\_sl} = \begin{cases} 0, & M_\omega > M_{\omega\_sl} \\ 1, & M_\omega \le M_{\omega\_sl} \end{cases} \tag{9}$$

and

$$J_{\omega\_ll} = \begin{cases} 0, & M_\omega < M_{\omega\_ll} \\ 1, & M_\omega \ge M_{\omega\_ll} \end{cases} \tag{10}$$

Based on the evaluation $S_{t-K'+1}$,$S_{t-K'+2}$, ..., S_{t-1}, S_t of previous K' time slots, we can get the final evaluation value that represents the availability of nodes, $S_{G_t} = \sum_{k=0}^{k=K'-1}(K'-k)S_{t-k} / \sum_{k=0}^{k=K'-1}(K'-k)$.

D. Node Selection Module

On basis of the network awareness and evaluation offered by the network measurement and evaluation module, the node selection module can obtain a complete traffic topology. Based on the topology, it selects multiple nodes to collect data. Thus, an intelligent and adaptive distributed data collection system can be realized in order to reduce redundant data collection and mitigate the awareness limitation of network traffic.

The topology is abstracted as a directed graph, and the direction of edges is the direction of traffic. The direction of traffic is usually bidirectional, and thus the direction of edges is marked bidirectionally. As a result, we aim to select nodes in a topology. Given a directed topology graph $\vec{G} = (\vec{V}, \vec{E})$, where $\vec{V}$ represents the set of all nodes in the graph, and $\vec{E}$ represents the set of all directed edges. Fig. 2 is an abstracted network structure. Circular nodes represent switches and routers in the network. Square nodes are hosts. There are bidirectional links between switches or routers, marked as bidirectional solid lines. The link between the host and the switch or router is marked as a dotted line.

According to the blocking state of network links, the network node selection module runs in two modes, namely normal mode and abnormal mode. In the normal mode, the module first identifies all central nodes and access nodes, and then SDN controller collects flow information from these nodes. Central nodes are the intersection points of multiple links. Because the switch or router is rarely connected to the host in the core network. Theoretically, the traffic passing through the switch or router also passes through the central or access nodes. In Fig. 3, s2 and s6 are central nodes with a large amount of flow, while s11, s12, s71 and s72 are access nodes that connect to the routers, switches or hosts of non-SDN networks. External large-scale or malicious traffic flows into SDN networks must pass through these access nodes. When any central or access node is blocked, the network will change into the abnormal data collection mode, where our system locates the blocked roots by applying a blocked root discovery algorithm. Only the flows passing through the blocked roots will be collected in order to reduce the volume of collected network data.

Due to the requirements of reflecting network congestion, we need to figure out the traffic topology about network blocking. Directed edges in the topology represent all links that are blocked. These edges are still oriented, and the direction of the edges is the direction of large flows. However, if there is no large flow in the link, corresponding edges will not appear in the traffic topology. Furthermore, if a node does not connect with any blocking edges, it disappears from the blocking traffic topology. Therefore, the original topology can be simplified to a blocking traffic topology. Fig. 3 shows an blocking traffic topology. The directed edges labeled by red solid lines are blocked edges. The circles connected with these blocking edges are blocked nodes. The edges labeled by black dotted lines are non-blocked edges. These edges are not considered in blocking analysis.

Moreover, in order to enhance the robustness of the system, some special conditions should be taken into consideration as follows.

1. Some loops might exist in the blocking traffic topology. For example, when two malicious hosts continuously send a large amount of traffic to another host, a loop appears in the topology. In fact, there are many loops in the blocking traffic

Input: Nonempty blocked graph $\vec{G} = (\vec{V}, \vec{E})$.
Output: A set C of nodes, which contains all blocked roots in graph $\vec{G}$.
Step 1: Let $C = \emptyset$;
Step 2: **If** $\vec{V} = \emptyset$ **then**
return C;
end if
Step 3: **If** we have found all nodes v whose in-degree is 0 **then**
put all discovered nodes into set C, and remove them from $\vec{G}$ and corresponding links; Get new $\vec{G} = (\vec{V}, \vec{E})$;
else
If $\vec{G} = \emptyset$ **then**
return C;
else // Loops exist in $\vec{G}$.
go back to **Step 1**;
end if
end if
Step 4: Search the node which has the max degrees in $\vec{V}$; Put it into C and remove it and its links from $\vec{G}$; Go back to **Step 3**.

Algorithm 1: Finding Blocked Root Nodes

topology.

2. When both directions have large traffic, a blocked edge is bidirectional.

3. The blocking of network links is continuous. When a link is blocked by large traffic, its downstream link is also blocked until the traffic is diverted to several different links. The blocking depends on the bandwidth of switches, which is applicable in peer-to-peer and downlink networks.

The process of node selection is described as below.

Step 1: The controller discovers the network topology and computes the availability of network topology, and find out all central and access nodes.

Step 2: The availability of all edges can be calculated. If the evaluation value of an edge is lower than a threshold, this link is marked blocked, Otherwise, the link is marked as an unblocked one. If all links are available, the normal flow sampling mode is turned on, the central and access nodes are selected as data collection nodes; otherwise, go to Step 3.

Step 3: The abnormal traffic mode is triggered, and the node selection module discovers the blocked root node.

Step 4: The blocked node collects and forwards the specified network traffic to a remote server according to the flow table policy specified by the controller.

The algorithm for finding blocked root nodes is described in **Algorithm 1**.

E. Adaptive Traffic Collection Module

With the development of 5G and big data, the volume of traffic is huge in the process of network data collection. How to control the volume of data collected from the selected collection nodes, reduce the data redundancy and ensure the recoverability of information as far as possible is a big challenge in the process of data collection. In order to solve these problems, traffic sampling is used in data collection. At present, many flow sampling algorithms have been proposed to guide data collectors to realize the restorable ability of flow distribution when the collected data is compressed. In addition, some algorithms [23], [27] reduce the redundant information of flows based on in-flow sampling. Packet sampling methods are time-sensitive data collection methods, which usually adjust collection strategies at time intervals. However, flow sampling is the time-insensitive sampling method, which cannot adapt to real-time situations, i.e., DDoS and other real-time network attacks. At the same time, it heavily depends on the response speed and detection accuracy of servers.

Current packet sampling methods usually depend on time intervals and do not consider large traffic or ordinary network scenarios so that the accuracy of data collection cannot be ensured. Besides, these methods lack the adaptivity to flow distribution, which becomes worse in elephant flows and loses the perception of mice flows. The main reasons that result in aforementioned problems are two aspects. On one hand, many packet sampling methods are applied in specific network situations. On the other hand, the lack of network situation awareness and the lag of network processing in traditional networks limit the ability of sampling algorithms. With the advent of 5G, SDN becomes its mainstream network architecture. An SDN-based data collection algorithm is urgently needed and simultaneously overcomes the flaws of traditional sampling algorithms. We propose some requirements of SDN based sampling algorithms.

1. The algorithm should be time-dependent in order to react quickly in time-sensitive networks.

2. The algorithm should realize the ability of flow awareness.

3. The algorithm has a certain in-flow sampling mechanism for reducing the size of elephant flows and its influence.

The following algorithm that satisfies the aforementioned requirements is proposed to adapt to flows, which is described in **Algorithm 2**.

Step 1: When a packet m passes through a network data collector, the flow tables in the collector attempt to match the packet according to the flow information of m.

Step 2: If m matches a flow table (i.e., it belongs to an old flow), the count $ocount_{m-1}$ of the old flow will be increased by 1. Its corresponding segment threshold will be checked. When the segment threshold is reached, the data collector performs collection operations. Otherwise, the packet belongs to a new flow. Assume the current new flow count is $ncount_{m-1}$. According to the dynamic probability generation in the controller, the dynamic probability p is generated. If the new flow count $ncount_m$ reaches $ncount_{m-1} + 1 = \lfloor 1/p \rfloor$, m is collected and the count is returned to 0. Otherwise, the new flow count is increased by 1 to $ncount_m = ncount_{m-1} + 1$, and m is not collected. End this operation until another new packet comes in.

Most operations, such as counting, comparison and collection are carried out in the data plane. Only the update of p in Step 2 and the generation of collection threshold by in-flow sampling in Step 3 are carried out in the controller plane. Thus, collection and processing efficiency can be achieved.

The dynamic probability generation and the in-flow sampling are described below.

Dynamic Probability Generation

In order to realize real-time awareness of network attacks such as DDoS and malicious network behaviors, collectors

Input: A packet m that passes through a network data collector, $m' = \emptyset$, $count_m = \{ocount_m, ncount_m = \emptyset\}$, $ocount_{m-1}, ncount_{m-1}$.
Output: m' and $count_m = \{ocount_m, ncount_m\}$
Step 1: Employ flow tables to match m;
Step 2: **If** flow tables match m **then** // m belongs to an old flow.
$ocount_m = ocount_{m-1} + 1$;
If $ocount_m$ reaches a threshold **then**
$m' = m$;
end if
else // m belongs to a new flow.
Generate p with **Dynamic Probability Generation**;
If $ncount_m$ reaches $ncount_{m-1} + 1 = \lfloor 1/p \rfloor$ **then**
$m' = m$;
$ncount_m = 0$;
else
$ncount_m = ncount_{m-1} + 1$;
end if
end if
Step 3: Return m' and $count_m = \{ocount_m, ncount_m\}$.

Algorithm 2: Adaptive Traffic Collection

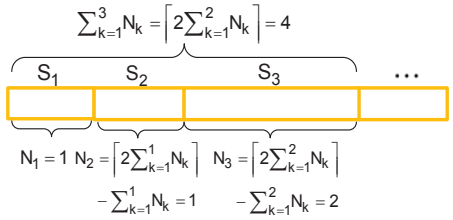


Fig. 4. Sample Diagram of Segment Partition for Flow Sampling.

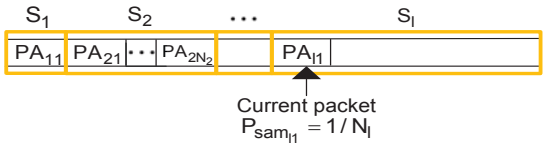


Fig. 5. An Example of Sampling with A Fixed Probability: Arrival of The First Packet.

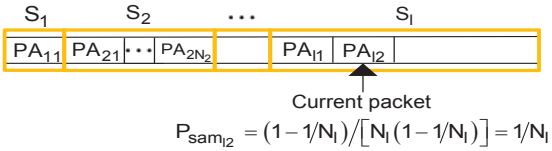


Fig. 6. An Example of Sampling with A Fixed Probability: Arrival of The Second Packet.

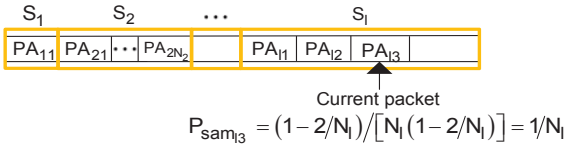


Fig. 7. An Example of Sampling with A Fixed Probability: Arrival of The Third Packet.

should make corresponding behavior changes according to changing traffic.

The Poisson sampling algorithm is an important and adaptive flow sampling algorithm recommended by RFC2330. It provides some ideas for the design of probability generation algorithm. The algorithm shows that in a time-dependent system, the factors at current period can be generated according to the factors at previous periods. The factors can be the packet interval of sampling, the time interval of sampling, etc. However, the Poisson sampling algorithm is not suitable for flow sampling, since the former depends on sampling intervals but the latter relies on the probability. Due to the correlation of flows between adjacent periods, Poisson sampling is used to adjust the sampling probability.

Suppose the sampling probability at time t is $P(t)$. If time t_1 and t_2 belong to a same sampling period, they have the same sampling probability. In other words, if $t_1, t_2 \in [(n-1)T, nT]$, $n = 1, 2, \ldots$, and $t_1 \le t_2$, we can get

$$P(t_1) = P(t_2), \tag{11}$$

where T is a fixed time interval. According to the Poisson sampling algorithm, the sampling probability is related to the last sampling time slots. Therefore, the average change rate of the last periodic probabilities can be calculated as follows.

$$P'(t) = \frac{1}{\lfloor \frac{N}{2} \rfloor^2} \sum_{i=1}^{\lfloor \frac{N}{2} \rfloor} P(t - iT) - P(t - iT - \lfloor \frac{N}{2} \rfloor T). \tag{12}$$

In fact, the probability change rate $\Delta(P(t)) = P(t) - P(t-T)$ is related to not only the average change rate at the last N slots but also the probability at the last time slot. In the investigation of flow sampling algorithm, it can be found that the change rate of the sampling probability decreases with the increase of the number of packets and the sampling probability of the flow sampling algorithm. Therefore, the change rate of probability at time t is computed as follows.

$$\Delta(P(t)) = P(t) - P(t-T) \sim \frac{b'}{a'P(t-T)^2 + 1}. \tag{13}$$

a', b' are the coefficients to adjust the change rate. With the relation $\Delta(P(t)) \sim P'(t)$, we can obtain $\Delta(P(t)) = k'b'P'(t)/[a'P(t-T)^2 + 1]$, where k' is the proportionality coefficient ensuring that the equation is true. Furthermore, we get $P(t) = P(t-T) + bP'(t)/[aP(t-T)^2 + 1]$, where $b = b'$ and $a = a'$. Finally, we can obtain

$$P(t) = P(t-T) + \frac{b \sum_{i=1}^{\lfloor \frac{N}{2} \rfloor} P(t - iT) - P(t - iT - \lfloor \frac{N}{2} \rfloor T)}{\lfloor \frac{N}{2} \rfloor^2 (aP(t-T)^2 + 1)} \tag{14}$$

In-Flow Sampling

Effectively handling elephant flows is an important objective in network data collection. Due to the Pareto's Law of real traffic in networks, a great number of collection resources are consumed when the elephant flows happen. However, an elephant flow is full of redundant information because of the high correlation of packets in the same flow. Therefore, in-flow sampling is proposed based on the correlation of packets in the same flow in order to reduce the volume of collected data. It consists of two stages, namely flow partition and sampling with a probability. The sampling probability decreases with the increase of the flow size.

In flow partition, an online flow whose current length is L can be divided into several segments. The length of each

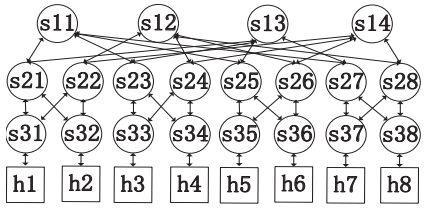


Fig. 8. An Example of Fat-Tree Topology.

segment is N_i, $i = 1, 2, \ldots, l$, where l is the number of segments. With the increase of i, the length of N_i increases.

In each segment, packets are sampled by a fixed probability. For the i-th segment of a flow, its length is N_i. The probability of each packet being sampled is $1/N_i$. With the increase of i, N_i increases, thus the sampling rate is reduced, so that less data are collected in the situation of elephant flows.

Given a real number $\mu > 1$, an online flow with length L can be divided into l segments. The length of the i-th segment can be denoted as

$$N_i = \begin{cases} 1, \ i = 1 \\ \lceil \mu \sum_{k=1}^{i-1} N_k \rceil - \sum_{k=1}^{i-1} N_k, \ 1 < i < l \\ min\{L, \lceil \mu \sum_{k=1}^{i-1} N_k \rceil - \sum_{k=1}^{i-1} N_k\}, \ i = l \end{cases} \tag{15}$$

Note that l and N_i are unique when L and μ are determined.

Fig. 4 shows N_i and the summation of the length of the last l intervals. Moreover, the bottom of Fig. 4 shows a special case to illustrate the regular length N_i of the i-th segment S_i when $\mu = 2$. In the sampling process, μ is usually set as a value slightly larger than 1 in order to avoid the excessive growth of N_i. In our design, only one data packet is collected in each segment. Therefore, when i increases, the sampling rate of the algorithm decreases. Specifically, for the first i segments whose total length is $\lceil \mu \sum_{k=1}^{i-1} N_k \rceil$, only i packets are collected, and the sampling rate in each segment is $1/N_i$. This reduces the number of collected packets compared to dividing flows with the same length and sampling a packet from a segment. In addition, the larger μ is, the lower the sampling rate.

The packets in S_i are denoted as $PA_{ij}, j = 1, 2, \ldots, N$ in order. PA_{ij}'s direct collection probability $P_{dir_{ij}}$ is computed, $P_{dir_{ij}} = 1/N_i[1 - (j-1)/N_i]$.

When a packet in S_i is collected, remaining packets in this segment are not collected. Therefore, the probability P_{sam} of PA_{ij} being sampled from S_i satisfies $P_{sam_{ij}} = (1 - \sum_{s=1}^{j-1} P_{sam_{is}}) P_{dir_{ij}}$. According to the calculation of recursive formula, the final probability of PA_{ij} being sampled can be computed, $P_{sam_{ij}} = 1/N_i$, which means that for any $k \in [1, N_i]$, the sampling probability is fixed and equal to $1/N_i$.

In order to illustrate the sampling process, Fig. 5-Fig. 7 show a special case in which packets are sampled from an online flow. As shown in Fig. 5, S_i is the current segment of an online flow, and its length is N_i. However, since the flow is online, its last segment length cannot be guaranteed to reach N_i. As shown in Fig. 5, only one packet belongs to the last segment of the flow. If the packet is not collected, collectors wait for the next packet belonging to this segment.

As shown in Fig. 6, the next packet of the flow enters. The packet is the second packet of S_i. Its direct probability $P_{dir_{i2}} = 1/N_i(1 - 1/N_i)$. Since the probability that PA_{i1} is not sampled is $1 - 1/N_i$, the probability that PA_{i2} is eventually sampled is $P_{sam_{i2}} = 1/N_i$. If PA_{i2} is still not sampled, the collector waits for the next packet.

Fig. 7 shows the scenario of the third packet coming in. According to the same calculation as the second packet entering, $P_{sam_{i3}} = 1/N_i$. If PA_{i3} is not sampled, subsequent packets are sampled with a less direct probability. Thus, the probability of all packets being sampled in this segment is fixed.

Estimation of Flow Size and Maximum Error

The flow size is the number of packets in a flow. We attempt to estimate the size of original flow based on the number l of sampled packet, which is an important factor to evaluate the availability of the sampling algorithm.

Suppose $f^n(x)$ is an n-order function of $f(x)$, namely $f^n(x) = \overbrace{f(f(\ldots f(x)))}^{n}$. Given a real number $\mu > 1$, we define $f(x) = \lceil \mu x \rceil$, and $L(l) = f^l(N_1) = f^l(1)$. In addition, we use a common estimation (i.e., mathematical expectation $E(L)$) of a real flow. When the number of sampled packets is l, L belongs to $[f^{l-1}(1) + 1, f^l(1)]$ and the sampling probability is fixed at $1/[f^l(1) - f^{l-1}(1)]$. Therefore, the mathematical expectation of the real flow is $E(L) = [f^l(1) + f^{l-1}(1))]/2$. In theory, the maximum deviation rate ε of the estimated value can be obtained, $\varepsilon = [f^l(1) - E(L)]/f^l(1) = [\lceil \mu f^{l-1}(1) \rceil - f^{l-1}(1)]/2\lceil \mu f^{l-1}(1) \rceil = \lceil (\mu - 1) f^{l-1}(1) \rceil / 2 \lceil \mu f^{l-1}(1) \rceil$. In fact, μ is usually less than 1.05, and thus the estimated ε is generally less than 2.5%.

IV. Test and Evaluation

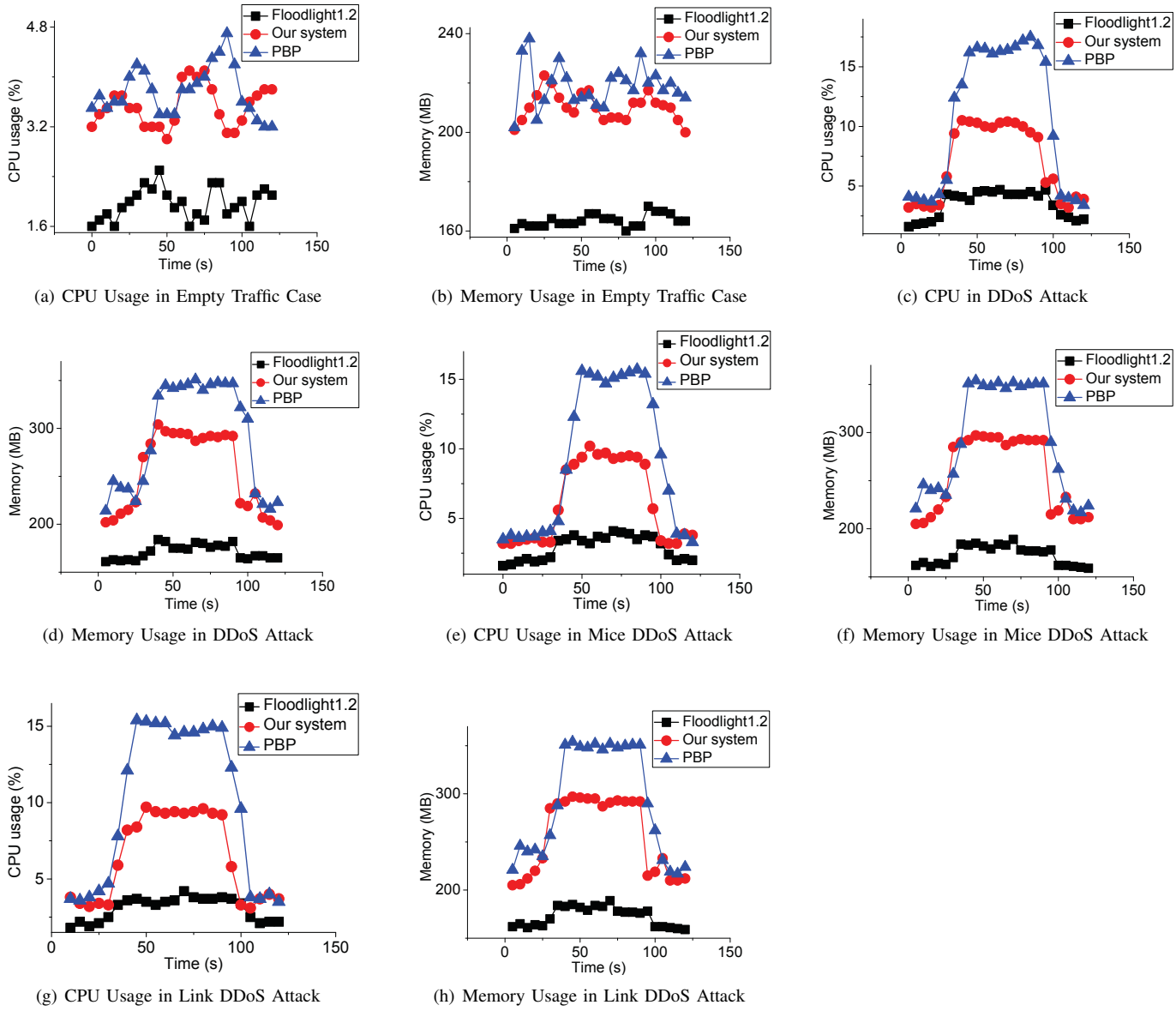
We evaluated the performance of our proposed system with a proof-of-concept prototype implementation. Then, we compared our system to other flow-based methods, e.g., Payless-Based Plan (PBP) [29] and sketch-based method [32], since current data collection methods can be categorized into flow and packet based ones and our system belongs to flow based data collection methods.

A. Metrics

We evaluated our system based on the following metrics. (1) The CPU consumption change with the increase of time; (2) The memory consumption change with the increase of time; (3) The storage usage change with the increase of the number of input packets; (4) The flow size recovery: the ability that the number of all packets of a flow can be predicted by using the packets sampled in this flow; (5) The threat perception: the ability of perceiving mainstream DDoS attacks.

B. Experimental Settings

Our implementation was completed in a laptop that runs the Ubuntu operating system 14.04 with Intel Core i7-8550



(a) CPU Usage in Empty Traffic Case

(b) Memory Usage in Empty Traffic Case

(c) CPU in DDoS Attack

(d) Memory Usage in DDoS Attack

(e) CPU Usage in Mice DDoS Attack

(f) Memory Usage in Mice DDoS Attack

(g) CPU Usage in Link DDoS Attack

(h) Memory Usage in Link DDoS Attack

Fig. 9. CPU and Memory Consumption.

CPU @1.8GHz and 8G RAM. We used Floodlight 1.2 as SDN controller in four kinds of traffic cases as described below. Our implemented virtual network adopted a fat-tree topology, which was built by the Mininet 2.2.1 simulation tool. An example of fat-tree topology is shown in Fig. 8. Based on our testing environment, we set $f_{td} = f_{pl} = f_{bd} = 1/3$, M_{td_sl} = 10ms, M_{td_ll} = 1000ms, M_{pl_sl} = 1%, M_{pl_ll} = 20%, $M_{bd_sl} = 10*8$bit, M_{bd_ll} = 100Mbit, $N = 6$, T = 34ms, $\mu = 1.005$, $a = b = 1$, and the initial sampling probability as $P = 0.5$.

C. Traffic Cases

Jing et al. claimed that DDoS attacks mainly include direct DDoS attacks, reflective amplification DDoS attacks and link flooding attacks [30]. The data packets of direct DDoS attacks and reflective amplification DDoS attacks are similar, and many data packets come from the same flow. Therefore, we can detect a large volume of flows for recognizing direct DDoS attacks and reflective amplification DDoS attacks. Although these three types of attacks cause a large volume of traffic, link flooding attacks are different from direct DDoS attacks and reflective amplification DDoS attacks due to various traffic characteristics. A small volume of flows can also lead to the unavailability of networks [31]. As a result, we can categorize current DDoS attacks into three classes due to the type and distribution of traffic flows, namely elephant flow, mice flow and link flooding based DDoS attacks. In our experiments, we considered the following four traffic cases.

1. Empty: No traffic in the network.

2. DDoS by elephant flow: Multiple elephant flows flood into a specific network host that leads to overwhelming.

3. DDoS by mice flow: A large volume of mice flows flood into a specific network host that leads to overwhelming.

4. Link DDoS: A large volume of traffic flows pass through

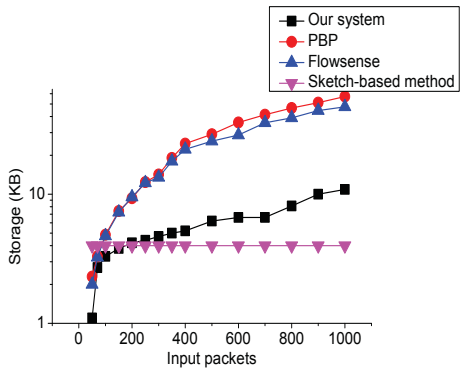


Fig. 10. Storage.

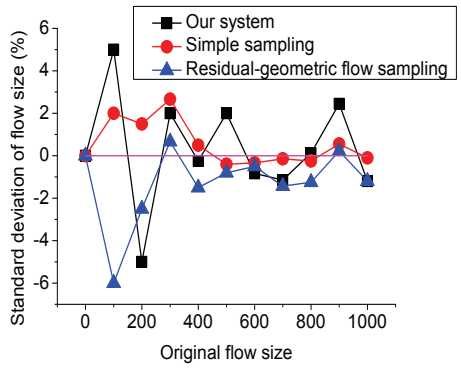


Fig. 11. Flow Recovery.

some attacking nodes so that some network links are overwhelmed.

D. CPU and Memory Consumption

We tested the CPU and memory consumption of PBP [29] and our system under the aforementioned four cases.

Fig. 9(a) and Fig. 9(b) show the CPU usage and memory consumption at the controller when links have no traffic. The CPU usage in our system is about 2% higher than that of the original floodlight controller. Although our collector consumes about 40MB more memory than floodlight, this is acceptable for a network data collection system. We can clearly see that our system outperforms PBP in this idle state, because it only checks the states of flows in central nodes when the network is free from blocking. Therefore, it saves CPU and memory resources.

In Case 2-4, as shown in Fig. 9(c)-Fig. 9(h), the CPU usage and memory consumption are quite different between PBP and our system. Obviously, PBP has a higher CPU usage and memory consumption than our system, which reduces at least 40% CPU resource consumption when there is flooding traffic. Because our system checks traffic states in central nodes but all nodes. Regarding memory, our system can save at least 40MB memory by comparing with PBP, since our system adopts a sampling flows method. In short, our system achieves economic CPU and memory consumption with regard to data collection in different network traffic scenarios.

E. Storage

We also tested the storage resource consumption of our system, PBP, Flowsense, and a sketch-based collection method [32]. The testing flows were distributed according to the ratio of elephant flows to mice flows. Several cases were tested: 1) 80% elephant flows and 20% mice flows; 2) 50% elephant flows and 50% mice flows; 3) 20% elephant flows and 80% mice flows.

As shown in Fig. 10, we can observe that the storage consumption of our system is much less than PBP and Flowsense, since sampling flows in our system reduces much storage consumption. The sketch-based method performs better than our system in terms of storage consumption. Because the sketch-based method only records the number of packets and stores it in a hash table. It cannot provide sufficient network information and hardly reflects a complete network traffic situation. In addition, it suffers from a large distortion phenomenon.

F. Flow Size Recovery

Flow size recovery is one of important criteria in evaluating a flow sampling algorithm. The flow size recovery capability of our system was evaluated and compared with the simple flow sampling algorithm and the geometric residual sampling algorithm [33], [34]. The latter two algorithms are very sensitive to elephant flow and have strong flow recovery ability due to their low convection collection rate. As shown in Fig. 11, the error rate of flow size recovery of our system is ranged from -5% to 5%, which is slightly worse than that of the latter two algorithms. However, when the compression factor in our system is 1.05 to collect 1000 data packets in the same flow, the size of collected data in the latter two methods is 10000 data packets. This fact implies that our system can achieve a similar error rate of flow size recovery to others, but collecting much less data.

G. Threat Perception

We further evaluate the threat perception ability of our system in terms of elephant flow, mice flow and link flooding based DDoS attacks. Based on the recovery of flow size by the data in Table 1, the threat perception ability of our system can be evaluated in Table 2 by restoring the number of flow packets and flows based on packet information collected under these three attacks.

In the elephant flow-based DDoS attack, the packets of elephant flow can be detected by our system. The PBP method can perceive the attack by recording the number of packets. The sketch-based method can also perceive the DDoS attack with multiple hash tables. Although the Flowsense based method can record the flow information, it cannot accurately and timely perceive the elephant flow based DDoS attack only when the flows are established and expired.

When the mice flow based DDoS attack occurs, a large number of packets belonging to new flows will enter switches or routers, and these flows usually do not have subsequent packets to maintain the survival of the flows. Therefore, the number of new flows counted by our prototype system

TABLE I
COMPARISON OF COLLECTED DATA

Network conditions	Our system		Flowsense	PBP	Sketch-based method
	The number of packets of elephant flows	The number of flows	The number of flows	The number of flows	The number of packets of elephant flows
Elephant flow based DDoS attacks	4963	1134	1207	5325	4972
Mice flow based DDoS attacks	575	5245	5212	5278	583
Link flood based DDoS attacks	4901	1083	1125	5332	232

TABLE II
CAPABILITY OF THREAT PERCEPTION

Network conditions	Our system	Flowsense	PBP	Sketch-based method
Elephant flow based DDoS attacks	Yes	No	Yes	Yes
Mice flow based DDoS attacks	Yes	Yes	Yes	No
Link flood based DDoS attacks	Yes	No	Yes	No

can be easily detected. The PBP method records the flow state, and thus it is also efficient to sense this attack. The Flowsense method uses the method of collecting packets from multiple network nodes. Although it can detect the attack, its performance is poor. The sketch-based method can only detect the elephant flow, but cannot detect mice flows. Thus, it cannot perceive this kind of DDoS attack.

The link flooding based attack is similar to the elephant flow based DDoS attack, which can result in blocking some nodes in the network. Therefore, both our system and the PBP method can sense it. However, The Flowsense method is unable to recognize this attack. The sketch-based method is usually deployed at a gateway or host nodes, thus it cannot identify the attack on links.

V. CONCLUSION

In this paper, we proposed an adaptive network data collection system in SDN. It employs a network measurement and evaluation model to quantify network node and link states for selecting proper nodes for data collection. The selected nodes adaptively perform flow sampling based on flow characteristics, thus ensuring the economy, efficiency, efficacy and low resource consumption of data collection. We conducted a series of experiments based on a proof-of-concept prototype to evaluate our system and compare its performance with other methods in terms of CPU/memory consumption, storage usage, flow size recovery, and threat perception. Experimental results show that our system outperforms other methods as a whole. Although our system can help detecting mainstream DDoS attacks, it might not perceive other types of attacks, e.g., impersonation attacks. Future work includes real deployment in a 5G test-bed system to support 5G network management and intrusion detection.

ACKNOWLEDGMENT

The work is supported in part by the National Natural Science Foundation of China under Grants 61672410, 61802293 and U1536202, the Academy of Finland under Grants 308087 and 314203, the Key Lab of Information Network Security, Ministry of Public Security under grant No. C18614, the National Postdoctoral Program for Innovative Talents under grant BX20180238, the Project funded by China Postdoctoral Science Foundation under grant 2018M633461, the open grant of the Tactical Data Link Lab of the 20th Research Institute of China Electronics Technology Group Corporation, P.R. China under grant CLDL-20182119, the Shaanxi Innovation Team project under grant 2018TD-007 and the 111 project under grant B16037.

REFERENCES

[1] M.K. Karray, "Analytical evaluation of QoS in the downlink of OFDMA wireless cellular networks serving streaming and elastic traffic," IEEE Transactions on Wireless Communications, vol. 9, no. 5, pp. 1799-1807, 2010.

[2] P.H. Isolani, J.A. Wickboldt, C.B. Both, et al., "Interactive monitoring, visualization, and configuration of OpenFlow-based SDN," in Proc. of Ifip/ieee International Symposium on Integrated Network Management, 2015.

[3] D. Marconett and S.J.B. Yoo, "FlowBroker: A software-defined network controller architecture for multi-domain brokering and reputation," Journal of Network and Systems Management, vol. 23, no. 2, pp. 328-359, 2015.

[4] J.M. Wang, W. Ying, X. Dai, et al., "SDN-based multi-class QoS-guaranteed inter-data center traffic management," in Proc. of IEEE International Conference on Cloud Networking, 2014.

[5] M. Belyaev and S. Gaivoronski, "Towards load balancing in SDN-networks during DDoS-attacks," in Proc. of Science & Technology Conference, 2015.

[6] R.T. Kokila, S.T. Selvi, and K. Govindarajan, "DDoS detection and analysis in SDN-based environment using support vector machine classifier," in Proc. of 6th International Conference on Advanced Computing, 2015.

[7] K. Bhushan and B.B. Gupta, "Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment," Journal of Ambient Intelligence & Humanized Computing, vol. 10, no. 5, pp. 1985-1997, 2019.

[8] S.M. Mousavi and M. Sthilaire, "Early detection of DDoS attacks against SDN controllers," in Proc. of International Conference on Computing, 2015.
[9] W. Miao, F. Agraz, S. Peng, et al., "SDN-enabled OPS with QoS guarantee for reconfigurable virtual data center networks," Journal of Optical Communications and Networking, vol. 7, no. 7, pp. 634-643, 2015.
[10] D. Kreutz, F. Ramos, P. Verissimo, et al., "Software-defined networking: A comprehensive survey," Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, 2015.
[11] S. Sezer, "SDN security: A survey," In Proc. of IEEE SDN for Future Networks and Services, 2013.
[12] B.B. Gupta, "Computer and cyber security: principles, algorithm, applications, and perspectives," CRC Press, 2018.
[13] Y. Jararweh, M. Alsmirat, M. Al-Ayyoub, et al., "Software-defined system support for enabling ubiquitous mobile edge computing," The Computer Journal, vol. 60, no. 10, pp. 1443-1457, 2017.
[14] N. Bandi, A. Metwally, D. Agrawal, et al., "Fast data stream algorithms using associative memories," in Proc. of ACM Sigmod International Conference on Management of Data, 2007.
[15] G. Cormode, F. Korn, S. Muthukrishnan, et al., "Finding hierarchical heavy hitters in streaming data," ACM Transactions on Knowledge Discovery from Data, vol. 1, no. 4, pp. 1-48, 2008.
[16] D. Zhou, Z. Yan, Y. Fu, et al., "A survey on network data collection," Journal of Network and Computer Applications, vol. 116, pp. 9-23, 2018.
[17] H. Nan, B. Lin, J. Xu, et al., "BRICK: a novel exact active statistics counter architecture," in Proc. of ACM/IEEE Symposium on Architectures for Networking and Communications Systems, 2008.
[18] J. Huang, P. Lee, R. Li, et al., "SketchVisor: Robust network measurement for software packet processing," in Proc. of the 2017 ACM Sigmod Conference,, 2017.
[19] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with OpenSketch," in Proc. of USENIX Symposium on Networked Systems Design and Implementation, 2013.
[20] P. Huang, P. Lee, and Y. Bao, "SketchLearn: Relieving user purdens in approximate measurement with automated statistical inference," In Proc. of the 2018 ACM Sigmod Conference. 2018.
[21] T. Yang, J. Jiang, P. Liu, et al., "Elastic sketch: Adaptive and fast network-wide measurements," in Proc. of the 2018 ACM Special Interest Group, 2018.
[22] L. Jose, M. Yu, and J. Rexford, "Online measurement of large traffic aggregates on commodity switches," In Proc. of Usenix Conference on Hot Topics in Management of Internet, 2011.
[23] C. Yu, C. Lumezanu, Y. Zhang, et al., "FlowSense: Monitoring network utilization with zero measurement cost," in Proc. of International Conference on Passive & Active Network Measurement, 2013.
[24] M. Karakus and A. Durresi, "Quality of Service (QoS) in Software Defined Networking (SDN): A survey," Journal of Network and Computer Applications, vol. 80, pp. 200-218, 2017.
[25] H.Q. Lin, Z. Yan, Y. Chen, et al., "A survey on network security-related data collection technologies," IEEE Access, vol. 6, no. 1, pp. 18345-18365, 2018.
[26] H.Q. Lin, Z. Yan, Y.L. Fu, "Adaptive security-related data collection with context awareness," Journal of Network and Computer Applications, vol. 126, pp. 88-103, 2019.
[27] X.Y. Jing, Z. Yan, X.Q. Liang, et al., "Network traffic fusion and analysis against DDoS flooding attacks with a novel reversible sketch," Information Fusion, vol. 51, pp. 100-113, 2019.
[28] X.Y. Jing, J.J. Zhao, Q.H. Zheng, et al., "A reversible sketch-based method for detecting and mitigating amplification attacks," Journal of Network and Computer Applications, vol. 142, pp. 15-24, 2019.
[29] S.R. Chowdhury, M.F. Bari, R. Ahmed, et al., "PayLess: A low cost network monitoring framework for Software Defined Networks," in Proc. of IEEE/IFIP Network Operations and Management Symposium, 2014.
[30] X.Y. Jing, Z. Yan, and W. Pedrycz, "Security data collection and data analytics in the Internet: A survey," IEEE Communications Surveys and Tutorials, vol. 21, pp. 586-618, 2019.
[31] P. Dong, X. Du, H. Zhang, et al., "A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows," in Proc. of IEEE International Conference on Communications, 2016.
[32] M. Rejmanek, "A theory of seed plant invasiveness: The first sketch," Biological Conservation, vol. 78, no. 1, pp. 171-181, 1996.
[33] V. Paxson, G. Almes, J. Mahdavi, et al., "Framework for IP performance metrics," IETF RFC 2330, 1998.
[34] X. Wang, X. Li, and D. Loguinov, "Modeling residual-geometric flow sampling," IEEE/ACM Transactions on Networking, vol. 21, no. 4, pp. 1090-1103, 2013.
[35] H. Xie, Z. Yan, Z. Yao, et al., "Data collection for security measurement in wireless sensor networks: A survey", IEEE Internet of Things Journal, vol. 6, no. 2, pp. 2205-2224, 2019.
[36] L. He, Z. Yan, and M. Atiquzzaman, "LTE/LTE-A network security data collection and analysis for security measurement: A survey", IEEE Access, vol. 6, pp. 4220-4242, 2018.
[37] G. Liu, Z. Yan, and W. Pedryczc, "Data collection for attack detection and security measurement in mobile Ad Hoc Networks: A survey", Journal of Network and Computer Applications, vol. 105, pp. 105-122, 2018.
[38] Z. Yan, Y. Chen, and Y. Shen, "Percontrep: A practical reputation system for pervasive content services," The Journal of Supercomputing, vol. 70, no. 3, pp. 1051-1074, 2014.
[39] Z. Yan and P. Cofta, "A mechanism for trust sustainability among trusted computing platforms," in Proc. of International Conference on Trust and Privacy in Digital Business, 2004.
[40] Z. Yan, "Trust modeling and management in digital environments from social concept to system development," Information Science Reference, IGI Global, Hershey, Pennsylvania, USA, 2010.

Donghao Zhou received the B.Sc. degree in telecommunications engineering from Zhengzhou University, Zhengzhou, China, in 2016. He is currently pursuing the master degree in information security in Xidian University, Xi'an, China. His research interests are in security, Software-Defined-Network (SDN) and security measurement.

Zheng Yan received the BEng degree in electrical engineering and the MEng degree in computer science and engineering from the Xi'an Jiaotong University, Xi'an, China in 1994 and 1997, respectively, the second MEng degree in information security from the National University of Singapore, Singapore in 2000, and the licentiate of science and the doctor of science in technology in electrical engineering from Helsinki University of Technology, Helsinki, Finland. She is currently a professor at the Xidian University, Xi'an, China and a visiting professor at the Aalto University, Espoo, Finland. Her research interests are in trust, security, privacy, and security-related data analytics. Prof. Yan serves as a general or program chair for 30+ international conferences and workshops. She is a steering committee co-chair of IEEE Blockchain international conference. She is also an associate editor of many reputable journals, e.g., IEEE Internet of Things Journal, Information Sciences, Information Fusion, JNCA, IEEE Access, SCN, etc.

Gao Liu is currently a Ph.D. candidate in Information Security, School of Cyber Engineering, Xidian University. His research interest includes network security measurement, data collection, blockchain, deep learning, machine learning, e-voting, e-lottery, micropayment, data aggregation in Smart Grid, and authentication in Smart Grid and VANETs.

Mohammed Atiquzzaman received the MS and PhD in electrical engineering and electronics from the University of Manchester. He holds the Edith Kinney Gaylord Presidential professorship at the University of Oklahoma. He is the editor-in-chief of Journal of Network and Computer Applications, founding editor-in-chief of Vehicular Communications, associate editor of many journals including IEEE Transactions on Mobile Computing, IEEE Journal on Selected Areas in Communications and IEEE Communications Magazine. His research interests include IoT, wireless and mobile networks, and satellite networks. His research has been funded by NSF, NASA, US Air Force, Cisco, Honeywell, etc. His publications can be found at www.cs.ou.edu/ atiq.