Research Article

Detecting Persistent User Behavior Using Probabilistic Counting in Network-Wide View

Aiping Zhou ^{1,2}, Jin Qian ¹ and Hang Yu ¹

¹School of Computer Science and Technology, Taizhou University, Taizhou, China

²Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, Nanjing, China

Correspondence should be addressed to Aiping Zhou; apzhou@njnet.edu.cn

Received 20 June 2021; Accepted 23 September 2021; Published 12 October 2021

Academic Editor: Shangce Gao

Copyright © 2021 Aiping Zhou et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Persistent user behavior monitoring, which deals with finding users that occur persistently over a measurement period, is one hot topic in traffic measurement. It is significant for many applications, such as anomaly detection. Former works concentrate on monitoring frequent user behavior, such as users occurring frequently either over one measurement period or on one monitor. They have paid little attention to detect persistent user behavior over a long measurement period on multiple monitors. However, persistent users do not necessarily appear frequently in a short measurement period, but appear persistently in a long measurement period. Due to limited resource on monitors, it is not practical to collect a tremendous amount of network traffic in a long measurement period on one single monitor. Moreover, since network attackers deliberately send packets flowing through the entire managed network, it is difficult to detect abnormal behavior on one single monitor. To solve the above challenges, a novel method for detecting persistent user behavior called DPU is proposed, and it contains both online distributed traffic collection in a long measurement period on multiple monitors and offline centralized user behavior detection on the central server. The key idea of DPU is that we design the compact distributed synopsis data structure to collect the relevant information with users occurring in a long measurement period on each monitor, and we can reconstruct user IDs using simple calculations and bit settings to find users with persistent behavior on the basis of estimated occurrence frequency of users on the central server when user IDs are unknown in advance. The experiments are conducted on real traffic to evaluate the performance of detecting persistent user behavior, and the experimental results illustrate that our method can improve about 30% estimation accuracy, 40% detection precision, and accelerate about 3 times in comparison with the related method.

1. Introduction

Frequent item and persistent item are two fundamental problems of data stream mining [1–3]. Frequent items mining is to find items with large occurrences over a measurement period. Frequent items mining has been extensively used in many applications such as the keyword query through the search engine. Persistent item mining is a special case of frequent item mining, which only counts once when an item occurs repeatedly over a measurement period. This study focuses on the problem of finding persistent items in the network-wide view. For an item, its occurrence frequency is the number of timeslots in which it appears. The problem of persistent items mining is converted to find

items with large occurrence frequency. We consider that items span across the whole managed network consisting of a central server and multiple monitors, where each monitor serves as one data collection point and the central server serves as one user behavior detection point. Each monitor stores the summary of network traffic and periodically sends the summary to the central server. The server estimates occurrence frequency of users using the received summary and reconstructs user IDs. Here, user IDs can be some identifiers such as source IP, source IP and source port pair, destination IP, destination IP and destination port pair, and the five tuple of network packet [4].

The task of persistent user behavior detection is to find users with persistent behavior, and it is essentially the

problem to mine persistent items. Persistent user behavior detection has been extensively applied in many fields such as click fraud and Botnet detection. For click fraud detection, identifying persistent user behavior helps to find attackers of click frauds who are periodically clicking on ads to increase payment for advertisers in pay-per-click online advertising systems [5]. For Botnet detection, persistent user behavior can be used to find attackers (C&C servers) who control a large number of bots to attack important servers by communication between C&C servers and bots periodically [6]. However, methods of frequent item mining cannot be directly used to solve the problem of finding persistent items, since frequent items are essentially different from persistent items.

There are challenges for persistent user behavior detection. First, a large amount of network traffic usually arrives at data collection devices at high rates for many real-world systems. For instance, users post and comment on social networks so as to generate massive data in the form of stream. Due to limited computation and memory resources on data collection devices, it is not feasible to store the whole user IDs over a long measurement period. To solve this challenge, we need to store user IDs by summary data structure which consumes a small amount of memory. Second, to evade detection, network attackers usually access important servers by utilizing multiple intermediate nodes. Due to attack traffic distributed on many nodes in a managed network system, network attacks are not detected by the single monitor efficiently. To solve this challenge, we need to aggregate the received data from multiple monitors to find network-wide attacks effectively. Third, it is difficult to accurately reconstruct user IDs using the encoded bits from multiple monitors, since there is high communication cost among the server and each monitor. To solve this challenge, we construct reversible hash functions based on Chinese remainder theorem and recover user IDs by simple computation when the entire user IDs are not stored. Fourth, due to the false positive rate, the user IDs reconstructed may not be users with persistent behavior. To solve this challenge, we consider filtering the reconstructed user IDs with low estimated occurrence frequency to improve the detection accuracy.

To the best of our knowledge, little research work has been done on persistent user behavior detection in network-wide view. The straightforward solution to find users with persistent behavior is that each monitor sends the whole observed packets to the server and the server finds users with persistent behavior by analyzing the received packets. However, this solution is not feasible to store the whole packets due to limited computing and memory resource of monitors. Besides, this solution incurs too much communication overhead between multiple monitors and the server. Existing methods have been not able to detect persistent user behavior over a long measurement period, such as traffic statistics estimation [7–9]. To address these challenges, each monitor can generate a compact summary of network traffic over each timeslot online and send the summary to the server for detecting persistent user behavior over a long

measurement period offline. Our main contributions are summarized as follows.

First, we develop a novel compact summary data structure which is suitable to detect persistent user behavior in a managed network consisting of a central server and multiple monitors. User IDs are extracted from network traffic and compressed to the summary data structure online in each timeslot over one measurement period for each monitor. Several bits are only stored for each distinct user IDs. Therefore, there is small memory and calculation.

Second, for the task of detecting persistent user behavior, we need to find IDs of users with persistent behavior. User IDs are not known in advance, and it is not practical to maintain the whole IDs of users in massive network traffic. Therefore, we develop an invertible method to reconstruct IDs of users with persistent behavior. The method only needs simple computing and to set several bits for each user with persistent behavior. Nevertheless, it does not store the whole IDs of users for detecting persistent user behavior so as to save memory.

Finally, we analyze the estimation accuracy of DPU in theory. We also conduct extensive experiments on two real network traffic traces. The experimental results demonstrate that DPU is superior to them compared to the relative ones in terms of estimation accuracy, detection precision, and compressing time.

The rest of this study is organized as follows. Section 2 summarizes related works. Section 3 formulates the problem. Section 4 presents our method. Section 5 evaluates performance and conducts experiments. Section 6 concludes.

2. Related Works

Detecting abnormal behavior has been one hot topic of network traffic measurement and monitoring, such as frequent items [10, 11], superspreaders [12, 13], and persistent items. Frequent items occur many times over one short measurement period, and superspreaders connect a large number of distinct hosts over one measurement period. Persistent items continuously occur over one long measurement period. Superspreaders and persistent items are the special cases of frequent items. Nevertheless, the methods to detect frequent items and superspreaders are not directly able to be used by persistent items. Users with persistent behavior are actually persistent items in this study. Existing approaches to find users with persistent behavior can be mainly divided into two categories, whose performance is summarized in Table 1.

The first category identifies users with persistent behavior at one single monitor. The straightforward methods maintain all users by hash table over a long measurement period to detect users with persistent behavior. They generate large memory overhead, and it is not feasible to be a function module in the network equipment. Chen et al. [14] proposed a data streaming method to track continuous long-duration flows (CLDF) which occur continuously over a long measurement period. It uses two bloom filters to store the candidates of continuous long-duration flows occurring

TABLE 1: Comparison of algorithms.

Reference	Algorithms	Accuracy	Memory
[14]	CLDF	Low false positive and false negative rate	Small memory
[15]	NLDF	Without false negative error	Small memory
[16]	PIE	Low false negative rate	Small memory
[17]	UE, RUE	High recall and precision	Small memory
[18]	IW, SW	Low false positive and false negative rate	Small memory
[19]	DISPERSE	Low false positive and false negative rate	Small memory
[20]	LDF	High estimation accuracy	Small memory

over the previous and current timeslots, respectively. Continuous short duration flows are filtered by sampling to improve its performance. Lee et al. [15] proposed a data streaming method to detect long-duration flows without false negatives (NLDF). It uses two bloom filters to maintain flows by hash functions. It does not generate false negative. Research shows that the online method exactly tracks persistent items using large memory and is not run on one monitor. Dai et al. [16] proposed a persistent items identification scheme (PIE), which solves the problem to identify persistent items and estimate the occurrence frequency of each persistent item. It uses Raptor codes to encode the ID of each item occurring and only stores several bits of the encoded ID over a measurement period. Wang et al. [17] proposed user embedding (UE) and reversible user embedding (RUE) to detect persistent frequent behaviors. They only need to randomly select bits from one bit array and set the selected bits to one for each user occurring over each timeslot. Then, they estimate the occurrence frequency of each user to detect persistent users.

The second category identifies users with persistent behavior on multiple monitors. Singh and Tirthapura [18] proposed the distributed methods to track persistent items approximately under infinite window and sliding window (IW and SW). They have low communication overhead, false positive rate and false negative rate. Dai et al. [19] proposed a probabilistic distributed persistent item identification method (DISPERSE) to find persistent items in distributed datasets. Each monitor compresses the ID of each item in a lossy fashion and sends the set of lossy compressed IDs of items to the server; then, the server recovers the IDs of persistent items.

Besides, some methods estimate the persistent spread, which is the number of distinct hosts it connects persistently over a long measurement period. Xiao et al. [21] proposed a persistent spread estimator to help detect long-term stealthy behaviors. It generates a virtual bitmap randomly selected from the bitmap to estimate the persistent spread of each flow. Zhou et al. [22] proposed a highly compact and efficient virtual intersection HyperLogLog architecture for persistent spread estimator of each flow to help detect long-term stealthy behaviors. It uses register intersection and maximum likelihood estimation to measure the persistent cardinality of each flow. It obtains far better memory efficiency. Huang et al. [23] proposed an efficient and accurate k -persistent estimator. It uses SUM to join the information collected from different periods to estimate the k -persistent spread of each flow.

The above methods are essentially different from the persistent user behavior detection studied in this study, and they are good at detecting abnormal behavior over one short measurement period at one monitor. Therefore, they are inapplicable to detect network-wide persistent user behavior over a long measurement period. Shin and Yoon [20] proposed a long-duration flow method (LDF) to detect persistent items which occur over one long measurement period. It uses an integer array to construct summary of network traffic. Based on the constructed summary, it generates each flow's virtual vector consisting of several bits randomly selected from the integer array using hash functions when we inquiry about the occurrence of one flow. Then, it estimates the occurrence of one flow using the generated virtual vector. Compared to LDF, our algorithm is more efficient to detect persistent user behavior.

3. Problem Statement

In this study, we consider a managed network system composed of a central server and a set of monitors. At the beginning of a measurement period, each monitor extracts user IDs from network traffic and compresses user IDs into synopsis data structure. At the end of the measurement period, each monitor sends the generated summary data structure to the central server. The central server merges the generated summary data structure from each monitor and detects persistent user behavior by probabilistic counting. Let r be the number of monitors. Moreover, the user ID called u can be an arbitrary value, such as source IP, source port, destination IP, destination port, and protocol. The central server does not know user IDs in advance.

We are interested in measuring how long the u occurs. The issue is how to quantitatively define the occurrence frequency of u . The traditional definition of occurrence frequency is to find the number of consecutive timeslots, in which the u appears over the long measurement period. This definition cannot solve the problem of concealed network attacks. For example, we divide one hour into 60 timeslots whose size is one minute. Supposing that u occurs in 50 timeslots, there are only 48 consecutive timeslots. Based on the above definition, the occurrence frequency of u is 48. If the given threshold is 50, the network attacks cannot be detected. Therefore, we define the occurrence frequency of u as the number of timeslots. The occurrence frequency of u is 50, and u is considered to be a concealed network attack behavior.

Generally, we specify the occurrence frequency of u by dividing a long measurement period into many timeslots and measure the number of timeslots, in which u appears persistently. We give a more formal definition as follows.

To formulate the problem of persistent item detection, we introduce some notations. We divide the long measurement period T into many timeslots T_k with the same size. Let U be the set of users occurring over the long measurement period T . Each user has a user identification, and the set of user identifications is denoted as U , belonging to $\{0, 1, \dots, N-1\}$, where N is the size of U . For example, we divide T ($=60$) minutes into 60 timeslots with 1 minute. Supposing that the user identification consists of source IP and destination IP, we have $N=2^{64}$.

For each user $u \in U$, let $o_u^{i,t}$ denote the occurrence indicator of u in timeslot t on one monitor i . When u occurs in timeslot t on one monitor i , $o_u^{i,t}$ is one, and otherwise, $o_u^{i,t}$ is 0. Therefore, the occurrence indicator of u for timeslot t is denoted as $o_u^t = \cup_{i=1}^r o_u^{i,t}$. When user u occurs at least in one monitor in timeslot t , o_u^t is one, and otherwise, o_u^t is 0. The occurrence frequency of the user u is defined as $f_u = \sum_{t \in T} o_u^t$. Persistent user behavior is defined as users whose occurrence frequency f_u is more than the given threshold F , namely, users occur at least in F timeslots over the measurement period T .

The proposed framework for estimating the occurrence frequency of u is intended to be generic, and its parameters are configured according to the application requirements. Specially, the size of timeslot depends on the specific applications. Besides, the threshold is set for abnormal user behavior detection to be more than the measured number of timeslots in which users appear, and there is divergence among different networks. Similarly, the parameters of abnormal user behavior detection are set on the basis of the specific applications and normal user behavior. Consider the example of detecting concealed network attacks by measuring the occurrence frequency of u . When the measurement period is one day, we set the size of timeslot to be one hour. We may find persistent user behavior in normal traffic, since it is probable that legal users access their e-mail and webs regularly. When we set the size of timeslot to be 5 minutes, we may still find persistent user behavior in normal traffic, since it is probable that connections to service may appear intermittently over many timeslots. However, we choose the proper timeslot size and threshold, and it is unlikely for normal user behavior demonstrating the same occurrence frequency to access the servers as persistent user behavior.

The goal of this study is to design an occurrence frequency estimation framework that is composed of an online module and an offline module. The former stores all users in real time using synopsis data structures on each monitor. The latter estimates the occurrence frequency of users based on the summary data structures from multiple monitors and detects persistent user behavior. We will evaluate the performance of our design based on memory overhead and estimation accuracy.

4. Our Algorithm

The framework of our method is shown in Figure 1. First, each monitor processes the arriving packets and updates the summary data structure online. Second, each monitor sends

FIGURE 1: Framework of our method.

the generated summary data structure to the central server at the end of one measurement period. Finally, on the basis of the received data structures, the central server estimates occurrence frequency of users by the probabilistic counting approach and then reconstructs IDs of users with persistent behavior to detect persistent user behavior offline.

4.1. Data Structure. We construct a bit vector B [24] with size n , in which users are mapped to by one group of hash functions $h_j(u)$, $1 \leq j \leq k$. Hash function $h_j(u)$, $1 \leq j \leq k$ is defined as $h_j(u): \{0, 1, \dots, N-1\} \rightarrow \{0, 1, \dots, n_j-1\}$, $1 \leq j \leq k$, where N is the size of user space, and m_i and m_j are coprime to each other, $1 \leq i < j \leq k$. Another group of hash functions $f_j^t(u)$ for one timeslot t is defined as $f_j^t(u): \{0, 1, \dots, N-1\} \rightarrow \{0, 1, \dots, n-1\}$, $1 \leq j \leq k$, where n is the size of bit vector B . User IDs are mapped to the bits vector B by hash functions $f_j^t(u)$, $1 \leq j \leq k$. Therefore, the corresponding bits in B are set to ones, that is, $B[f_j^t(h_j(u))] = 1$, $1 \leq j \leq k$. The data structure is shown in Figure 2. At the beginning of each timeslot, each monitor initializes B . B is updated by hash functions $f_j^t(u)$, $1 \leq j \leq k$ during each timeslot. At the end of each timeslot, each monitor sends its bit vector B to the central server and resets B . Updating data structure is shown in Algorithm 1.

For each user, we need $2k$ hash calculations and k bit setting operations.

4.2. Estimating Occurrence Frequency. There are r monitors in the managed network system, and they may be any computing devices such as the computer, firewall, and router. The occurrence indicator of u for one timeslot t is denoted as $o_u^t = \cup_{i=1}^r o_u^{i,t}$, and it is estimated as $\hat{o}_u^t = \cup_{i=1}^r \hat{o}_u^{i,t}$. When one user u appears at least in one monitor i for one timeslot t , o_u^t is one. Let $U_c^{i,j}$ be the set of users whose hash values $h_j(u)$ are c , on one monitor i , and $U_c^{i,j}$ is denoted as $U_c^{i,j} = \{u | h_j(u) = c, u \in U^i\}$, where U^i indicates the set of

FIGURE 2: Data structure.

Input: initialize B_i , $1 \leq i \leq r$
 Output: the updated B_i , $1 \leq i \leq r$
 (1) for each timeslot t do
 (2) for each packet do
 (3) extract user ID u
 (4) compute $h_j(u)$, $1 \leq j \leq k$
 (5) compute $f_j^t(h_j(u))$, $1 \leq j \leq k$
 (6) set the $f_j^t(h_j(u))$ bit of B_i to one
 (7) end for
 (8) end for

ALGORITHM 1: Updating data structure.

users on one monitor i . We define the occurrence indicator of $U_c^{i,j}$ as $o_c^{i,j,t} = \bigcup_{u \in U_c^{i,j}} o_u^{i,j,t}$ for one timeslot t . When at least one user in $U_c^{i,j}$ appears in one timeslot t , $o_c^{i,j,t}$ is one and zero otherwise. Therefore, $o_c^{i,j,t}$ is estimated as $\hat{o}_c^{i,j,t} = B_i[f_j^t(c)]$. When $o_c^{i,j,t}$ is one, $o_c^{i,j,t}$ can be correctly estimated as $\hat{o}_c^{i,j,t} = 1$. However, when $o_c^{i,j,t}$ is zero, $o_c^{i,j,t}$ may be erroneously estimated as $\hat{o}_c^{i,j,t} = 1$. Since hash functions $f_j^t(u)$, $1 \leq j \leq k$, are random and independent each other, different users may be mapped to the same bits in B_i ($1 \leq i \leq r$). For one user u on one monitor i , $o_u^{i,t}$ is estimated as

$$\hat{o}_u^{i,t} = \bigcap_{j=1}^k \hat{o}_{h_j(u)}^{i,j,t}. \quad (1)$$

When $o_u^{i,t}$ is one, $\hat{o}_u^{i,t}$ is one, and we obtain

$$P(\hat{o}_{h_j(u)}^{i,j,t} = 0 | o_{h_j(u)}^{i,j,t} = 0) = \frac{n_0^{i,t}}{n}, \quad (2)$$

where $n_0^{i,t}$ is the number of zero bits of B_i in one timeslot j on one monitor i . Let $U^{i,t} = \{u | o_u^{i,t} = 1, u \in U\}$ denote the set of users appearing in one timeslot t on one monitor i . We obtain

$$P(o_{h_j(u)}^{i,j,t} = 0 | o_u^{i,j,t} = 0) = \left(1 - \frac{1}{n_j}\right)^{|U^{i,t}|}. \quad (3)$$

Thus, we obtain the following equation based on (2) and (3).

$$P(\hat{o}_{h_j(u)}^{i,j,t} = 0 | o_u^{i,t} = 0) = \frac{n_0^{i,t}}{n} \left(1 - \frac{1}{n_j}\right)^{|U^{i,t}|}. \quad (4)$$

Then, we obtain

$$\begin{aligned} P(\hat{o}_u^{i,t} = 1 | o_u^{i,t} = 0) &= \prod_{j=1}^k P(\hat{o}_{h_j(u)}^{i,j,t} = 1 | o_u^{i,t} = 0) \\ &= \prod_{j=1}^k \left(1 - \frac{n_0^{i,t}}{n} \left(1 - \frac{1}{n_j}\right)^{|U^{i,t}|}\right). \end{aligned} \quad (5)$$

Therefore, we obtain the following equation based on (5).

$$\begin{aligned} q_t^i &= P(\hat{o}_u^{i,t} = 0 | o_u^{i,t} = 0) \\ &= 1 - P(\hat{o}_u^{i,t} = 1 | o_u^{i,t} = 0) \\ &= 1 - \prod_{j=1}^k \left(1 - \frac{n_0^{i,t}}{n} \left(1 - \frac{1}{n_j}\right)^{|U^{i,t}|}\right). \end{aligned} \quad (6)$$

When n_j is larger than $|U^{i,t}|$, q_t^i is approximately denoted as (7) based on the limit theory.

$$q_t^i = 1 - \left(1 - \frac{n_0^{i,t}}{n}\right)^k. \quad (7)$$

Therefore, we obtain

$$q_t = \prod_{i=1}^r \left(1 - \left(1 - \frac{n_0^{i,t}}{n}\right)^k\right). \quad (8)$$

Thus, each zero bit o_u^t , $t \in T_u^0$ is selected with probability q_t . Therefore, we estimate the occurrence frequency f_u of one user u as

$$\hat{f}_u = |T| - \sum_{t \in T} \frac{0(\hat{o}_u^t)}{q_t}, \quad (9)$$

where $0(\hat{o}_u^t)$ denotes a function, and $0(\hat{o}_u^t)$ is one as $\hat{o}_u^t$ is zero and zero otherwise. Estimating occurrence frequency is shown in Algorithm 2.

Then, the mathematical expectation and variance of the estimation $\hat{f}_u$ for f_u are computed as

$$E(\hat{f}_u) = f_u, \quad (10)$$

$$\text{Var}(\hat{f}_u) = \sum_{t \in T_u^0} \frac{1 - q_t}{q_t}, \quad (11)$$

where T_u^0 denotes the timeslots in which one user u does not appear.

The mathematical expectation and variance of the estimation $\hat{f}_u$ is proofed as follows.

Equation (9) is transformed into the following equation.

$$\hat{f}_u = f_u - \sum_{t \in T_u^0} \left(1 - \frac{0(\hat{o}_u^t)}{q_t}\right). \quad (12)$$

Then, we obtain

Input: the updated B_i , $1 \leq i \leq r$
Output: the estimated occurrence frequency

- (1) for each user u do
- (2) for each timeslot t do
- (3) compute $h_j(u)$, $f_j^t(h_j(u))$
- (4) estimate $\hat{o}_{h_j(u)}^{i,j,t} \leftarrow B_i[f_j^t(h_j(u))]$
- (5) estimate $\hat{o}_u^{i,t} \leftarrow \cap_{j=1}^k \hat{o}_{h_j(u)}^{i,j,t}$
- (6) estimate the occurrence of u as $\hat{o}_u^t \leftarrow \cup_{i=1}^r \hat{o}_u^{i,t}$
- (7) count the number $n_0^{i,t}$ of zero bits of B_i , $1 \leq i \leq r$
- (8) compute the probability $q_t \leftarrow \prod_{i=1}^r 1 - (1 - (n_0^{i,t}/n))^k$
- (9) end for
- (10) end for
- (11) estimate $\hat{f}_u \leftarrow |T| - \sum_{t \in T} (0(\hat{o}_u^t)/q_t)$

ALGORITHM 2: Estimating occurrence frequency.

$$E(\hat{f}_u) = f_u - \sum_{t \in T_u^0} \left(1 - \frac{E(0(\hat{o}_u^t))}{q_t} \right). \quad (13)$$

Since the estimator $\hat{o}_u^t$ obeys 0-1 distribution with probability q_t , $E(0(\hat{o}_u^t))$ is q_t . Therefore, we have

$$E(\hat{f}_u) = f_u. \quad (14)$$

The estimator $\hat{f}_u$ is unbiased.

We obtain

$$\text{Var}(\hat{f}_u) = \sum_{t \in T_u^0} \frac{\text{Var}(0(\hat{o}_u^t))}{q_t^2}. \quad (15)$$

Since $\text{Var}(0(\hat{o}_u^t))$ equals $q_t - q_t^2$, we obtain

$$\text{Var}(\hat{f}_u) = \sum_{t \in T_u^0} \frac{1 - q_t}{q_t}. \quad (16)$$

4.3. Detecting Persistent User Behavior. Since U is not known in advance, we consider the reconstruction of user IDs. Let U_c^j be the set of users whose hash values $h_j(u)$ are c , and U_c^j is denoted as $U_c^j = \{u \mid h_j(u) = c, u \in U\}$. We define the occurrence indicator of U_c^j as $o_c^{j,t} = \cup_{u \in U_c^j} o_u^{j,t}$ for one timeslot t . The corresponding occurrence frequency $f_c^j = \sum_{t \in T} o_c^{j,t}$. When U_c^j contains users with persistent behavior, the estimation $\hat{f}_c^j$ of occurrence frequency is more than the threshold F_c that is the occurrence frequency of U_c^j . Therefore, we consider these users whose occurrence frequency estimation is more than $\hat{f}_c^j$ as users with persistent behavior. Let P_j be the set to which users with persistent behavior are mapped using hash functions $h_j(u)$, $1 \leq j \leq k$, and it is expressed as

$$P_j = \left\{ c \mid \hat{f}_c^j > F_c, 0 \leq c \leq n_j - 1 \right\}, \quad 1 \leq j \leq k. \quad (17)$$

Each user with persistent behavior is mapped to k sets P_j , $1 \leq j \leq k$. Therefore, the problem of detecting persistent user behavior is transformed to find users mapped to P_j , $1 \leq j \leq k$. We reconstruct IDs of users with persistent behavior using hash functions $h_j(u)$, $1 \leq j \leq k$, and they are expressed as

$$h_j(u) \equiv k_j u + b_j \pmod{n_j}, \quad 1 \leq j \leq k, \quad (18)$$

where n_j , $1 \leq j \leq k$, is coprime to each other, $1 \leq k_j \leq n_j - 1$, and $0 \leq b_j \leq n_j - 1$. Let c_j equal $h_j(u)$, $1 \leq j \leq k$. Consider the simple situation. When $k_j = 1$ and $b_j = 0$, (18) is converted into the following equation.

$$u \equiv c_j \pmod{n_j}, \quad 1 \leq j \leq k. \quad (19)$$

We solve (19) using the Chinese remainder theory [25], and its solutions are expressed as

$$u \equiv \sum_{j=1}^k c_j N_j N_j^{-1} \pmod{N}, \quad (20)$$

where $N = \prod_{j=1}^k n_j$, $N_j = N/n_j$, and N_j^{-1} is obtained using $N_j N_j^{-1} \equiv 1 \pmod{n_j}$.

Consider the general situation. Based on the Euler-Fermat theorem, (20) is converted into the following equation.

$$u \equiv k_j^{n_j-2} (c_j - b_j) \pmod{n_j}, \quad 1 \leq j \leq k. \quad (21)$$

Let $a_j = k_j^{n_j-2} (c_j - b_j)$, and we obtain

$$u \equiv a_j \pmod{n_j}, \quad 1 \leq j \leq k. \quad (22)$$

The solution of (22) is expressed as follows.

$$u \equiv \sum_{j=1}^k a_j N_j N_j^{-1} \pmod{N}. \quad (23)$$

Since $c_j \in P_j$, $1 \leq j \leq k$, the number of combinations c_j , $1 \leq j \leq k$ is $\prod_{j=1}^k |P_j|$. Besides, the reconstructed IDs of users may be false positives due to the collision of hash function $f_j(u)$, $1 \leq j \leq k$, that is, different users are mapped to the same positions by hash function $f_j(u)$, $1 \leq j \leq k$. Therefore, we estimate the occurrence frequency $\hat{f}_c^j$ of U_c^j before IDs are reconstructed. If the estimation $\hat{f}_c^j$ is less than F_c , the corresponding IDs are not considered to be users with persistent behavior. Otherwise, we reconstruct IDs of users with persistent behavior by (23). The estimation of their occurrence frequency is expressed as

$$\hat{f}_u = \sum_{t \in T} \cap_{j=1}^k \delta_{c_j}^{j,t}. \quad (24)$$

5. Experiment and Evaluation

In this section, we introduce two public datasets. Our solution DPU was implemented in C++. For comparison purposes, we also implemented LDF [20] in C++. All experiments are performed on a server equipped with four-core Intel Xeon E-2224, 3.40 GHz CPU, and 32 GB RAM.

5.1. Datasets. To evaluate the performance of our algorithm, we use two real network traffic traces CDD and CND, respectively. CDD [26] is network traffic trace without packet header over 8 minutes, which is published by CAIDA. CND [27] is another network traffic trace without packet header collected at high link over 60 minutes, which is published by CERNET. CDD is divided into 480 timeslots with 1 second, and CND is partitioned into 360 timeslots with 10 seconds. There are almost the same number of packets during each timeslot. Table 2 provides the number of packets, source IP address (SIP for short), destination IP address (DIP for short), the number of source IP address, and destination IP address pairs for CDD and CND.

In this study, we use SIP and DIP pairs as user IDs. There is approximately 615K, 35K distinct source IP address, 268K, 46K distinct destination IP address, and 823K, 78K distinct source IP address and destination IP address pairs during each timeslot with 1 minute for CDD, CND, respectively. When the length of each timeslot is 2 minutes, there are about 724K, 48K distinct source IP address, 373K, 65K distinct destination IP address, and 1M, 126K distinct source IP address and destination IP address pairs during each timeslot for CDD, CND, respectively. As shown in Figure 3, most of users have low occurrence frequency, and a small number of users have high occurrence frequency, that is, the occurrence frequency of users approximately follows heavy-tailed distribution for CDD and CND.

In the experiments, we randomly allocate packets in the datasets for each monitor. For detecting persistent user behavior, we evaluate the performance of our algorithm compared with LDF [20]. LDF constructs a virtual vector consisting of bits randomly selected from an integer array by different hash functions for each flow. It uses an occurrence estimator to detect long-duration flows. We compare our method with LDF under the same memory usage in terms of estimation accuracy and detection precision. PDU and LDF have two configurable parameters: the data structure size n and the number k of hash functions. The experiments are performed under $n = 1 \times 10^5$, 2×10^5 , 3×10^5 , 4×10^5 , 5×10^5 , respectively. The number k of hash functions is set to 286 according to the proposed method [28]. We allocate the same amount of 1 MB memory to each method. We also need to know actual persistent users under different timeslots in advance, which is given in Table 3.

5.2. Estimation Accuracy. We use source IP address and destination IP address pairs as IDs of users for CDD and CND. We evaluate accuracy of occurrence frequency

estimation by the weighed mean difference (WMD for short) [29], which is defined as

$$\text{WMD} = \frac{\sum_{u \in U} |f_u - \hat{f}_u|}{\sum_{u \in U} f_u}, \quad (25)$$

where f_u denotes the occurrence frequency of one user u , and $\hat{f}_u$ denotes the estimation of occurrence frequency of one user u . The smaller the WMD is, the more accurate the estimation of occurrence frequency is.

Figure 4 compares the occurrence frequency estimation of users with their actual occurrence frequency for our algorithm and LDF using CDD and CND. Each point in the plots indicates a user ID, whose coordinate x denotes the actual occurrence frequency and the coordinate y denotes the corresponding estimation. We evaluate the accuracy of occurrence frequency estimation by the diagonal line. The closer the points are to the diagonal line, the more accurate the estimation of occurrence frequency is. As shown in Figures 4(a)-4(b), most of points generated by our algorithm are closer to the diagonal line than LDF using CDD. As shown in Figures 4(c)-4(d), we obtain similar experimental results. Therefore, our algorithm is superior to LDF in terms of occurrence frequency estimation.

Figures 5(a)-5(b) show the experimental results of two algorithms under different timeslot sizes for CDD. We can observe that the WMD of two algorithms decreases as the size n increases and the WMD of two algorithms increases as the timeslot size increases for CDD. The WMD of our algorithm is about 2 times smaller than that of LDF under $\Delta T = 1$ for CDD as the size n is 2×10^5 . Similarly, Figures 5(c)-5(d) demonstrate the experimental results of two algorithms under different timeslot sizes for CND. Therefore, our algorithm outperforms LDF in terms of WMD. In a word, our algorithm is superior to LDF in estimating the occurrence frequency of users.

Besides, as shown in Figure 6, two algorithms represent different estimation errors for CDD and CND when the number of monitors increases. We can see that WMD of two algorithms under the different numbers of monitors at each timeslot is 10 seconds. From Figure 6(a), we know that the WMD of two algorithms decreases as the number of monitors increases, and the WMD of our algorithm is smaller than LDF, since we randomly assign packets of datasets to each monitor, and then, the average number of users arriving on each monitor decreases as increasing the number of monitors. We obtain the similar experimental results for CND, as shown in Figure 6(b). Therefore, our algorithm outperforms LDF in estimating the occurrence frequency of users.

5.3. Detection Precision. For detecting persistent user behavior, we evaluate the performance of our algorithm compared with LDF. For each user u in the entire user ID space, we estimate the occurrence frequency of u to determine whether u is a user with persistent behavior. Obviously, the detected users may contain users with persistent behavior, and the users with persistent behavior may not be also detected. Therefore, we use false positive rate and false negative

TABLE 2: Statistics of datasets.

Dataset	Packet number	SIP number	DIP number	(SIP, DIP) number
CDD	220M	3M	1M	5M
CND	76M	822K	869K	2M

FIGURE 3: Distribution of occurrence frequency. (a) Distribution under $\Delta T = 1$ second for CDD. (b) Distribution under $\Delta T = 10$ seconds for CND.

TABLE 3: Actual persistent user under different timeslots.

Timeslot	1	2	3	4	5
CDD	375	316	288	225	173
Timeslot	10	20	30	40	50
CND	286	235	214	187	128

FIGURE 4: Continued.

FIGURE 4: Estimation accuracy. (a) Estimation accuracy of LDF for CDD. (b) Estimation accuracy of DPU for CDD. (c) Estimation accuracy of LDF for CND. (d) Estimation accuracy of DPU for CND.

FIGURE 5: Estimation accuracy under different n . (a) WMD under $\Delta T=1$ second for CDD. (b) WMD under $\Delta T=2$ seconds for CDD. (c) WMD under $\Delta T=10$ seconds for CND. (d) WMD under $\Delta T=20$ seconds for CND.

FIGURE 6: Estimation accuracy under different numbers of monitors. (a) WMD for CDD. (b) WMD for CND.

FIGURE 7: Detection precision. (a) False positive rate for CDD under different timeslots. (b) False negative rate for CDD under different timeslots. (c) False positive rate for CDD under different n . (d) False negative rate for CDD under different n .

FIGURE 8: Processing time. (a) Online updating time for CDD under different timeslots. (b) Online updating time for CDD under different n . (c) Offline detecting time for CDD under different timeslots. (d) Offline detecting time for CDD under different n .

rate to evaluate the detection precision of our algorithm. The false positive rate (FPR for short) denotes the ratio of the number of users with persistent behavior not identified correctly to the number of users with persistent behavior identified, and the false negative rate (FNR for short) indicates the ratio of the number of users with persistent behavior not identified to the number of actual users with persistent behavior [25]. The FPR and FNR are expressed as

$$\text{FPR} = \frac{|B - A|}{|B|}, \quad (26)$$

$$\text{FNR} = \frac{|A - B|}{|A|}, \quad (27)$$

where A is the set of actual users with persistent behavior, and B is the set of users with persistent behavior identified. We conduct extensive experiments to detect users with persistent behavior. We treat source IP address and destination IP address pairs as user IDs. Figure 7 shows the detection precision under different timeslot sizes and data

structure sizes for CDD. As shown in Figures 7(a)-7(b), we can see that the false positive rate and the false negative rate decrease as timeslot size increases for two algorithms. The false positive rate and the false negative rate of our algorithm are significantly lower than that of LDF. Similarly, as shown in Figures 7(c)-7(d), we can see that the false positive rate and false negative rate decrease as data structure size increases for LDF and PDU. The positive rate and false negative rate of PDU are apparently lower than that of LDF. PDU also generates some false positive and false negative, since different users may be mapped to the same bits in B . In brief, our algorithm is superior to LDF in detection precision.

5.4. Processing Time. We evaluate the processing time of our algorithm to detect persistent user behavior in comparison with LDF. Figures 8(a)-8(b) show the average online updating time of each packet under different timeslot sizes and data structure sizes. As shown in Figure 8(a), the average online updating time of each packet hardly changes as the timeslot size increases for LDF and PDU. As shown in

Figure 8(b), the average online updating time of each packet also hardly changes as the data structure size increases. The average online updating time of each packet for PDU is about 6 times faster than that of LDF. Figures 8(c)-8(d) show the offline detecting persistent user behavior time under different timeslot sizes and data structure sizes. As shown in Figure 8(c), the detecting persistent user behavior time barely changes as the timeslot size increases for two algorithms. As shown in Figure 8(d), the detecting persistent user behavior time also barely changes as the data structure size increases. The detecting persistent user behavior time for PDU is about 3 times faster than that of LDF. In a word, our algorithm PDU outperforms LDF in processing time.

6. Conclusion

In this study, we propose an algorithm DPU for detecting persistent user behavior from the network-wide view. DPU is computationally efficient to construct compact summary data structures of users occurring in a measurement period, since it only needs to randomly select and update some bits from the summary data structures by hash functions for each incoming packet online. DPU also has low communication cost to collect a large number of network-wide packets, since the monitors only send the generated summary data structures with small memory to the central server. On the basis of generated summary data structures over a long measurement period, DPU accurately estimates occurrence frequency of users by the probabilistic counting approach and then reconstructs user IDs by simple computation to detect users with persistent behavior when users occurring in a measurement period are unknown in advance. Its capability of quickly processing and consuming small memory can become a functional module of modern routers or firewall. The theoretical analysis and experimental results illustrate that our algorithm significantly outperforms LDF in terms of estimation accuracy, detection precision, and processing time. The experimental results illustrate that our method can improve about 30% estimation accuracy and 40% detection precision. The online updating time per packet and offline detecting time of our algorithm are about 3 and 6 times faster than that of LDF. In the future, we will deploy our algorithm to real distributed systems and study the problems on advanced persistent user behavior identification and parameter optimization [30–33]. In addition, we will generate the data by simulating abnormal user behavior to validate the performance of our algorithm [34, 35].

Data Availability

The data used to support the findings of the study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61802274), Open Project Foundation

of Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, China (K93-9-2017-01), Scientific Research Foundation for Advanced Talents of Taizhou University, China (QD2016027), and Innovation and Entrepreneurship Training Program of Jiangsu Students, China (201812917011Y).

References

- [1] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," *Theoretical Computer Science*, vol. 312, no. 1, pp. 3–15, 2004.
- [2] G. Cormode and M. Hadjieleftheriou, "Finding frequent items in data streams," *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1530–1541, 2008.
- [3] B. Lahiri, S. Tirthapura, and J. Chandrashekar, "Space-efficient tracking of persistent items in a massive data stream," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 7, no. 1, pp. 70–92, 2014.
- [4] Q. Xiao, S. Chen, Y. Zhou, and J. Luo, "Estimating cardinality for arbitrarily large data stream with improved memory efficiency," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 433–446, 2020.
- [5] H. Haddadi, "Fighting online click-fraud using bluff ads," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, pp. 21–25, 2010.
- [6] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, vol. 45, pp. 100–123, 2014.
- [7] M. Durand and P. Flajolet, "Loglog counting of large cardinalities," in *Proceedings of the Springer European Symposium on Algorithms*, pp. 605–617, Budapest, Hungary, September 2003.
- [8] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier, "Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm," in *Proceedings of the Conference on Analysis of Algorithms (AofA'07)*, pp. 137–156, Juan des Pins, France, June 2007.
- [9] Q. Xiao, Y. Zhou, and S. Chen, "Better with fewer bits: improving the performance of cardinality estimation of large data streams," in *Proceedings of the IEEE Conference on Computer Communications*, pp. 1–9, Atlanta, GA, USA, May 2017.
- [10] F. Wang and L. Gao, "Simple and efficient identification of heavy hitters based on bitcount," in *Proceedings of the IEEE 20th International Conference on High Performance Switching And Routing*, pp. 1–6, Xi'an, China, May 2019.
- [11] L. Tang, Q. Huang, and P. P. Lee, "MV-Sketch: a fast and compact invertible sketch for heavy flow detection in network data streams," in *Proceedings of the IEEE Conference on Computer Communications*, pp. 2026–2034, Paris, France, April 2019.
- [12] P. Wang, X. Guan, D. Towsley, and J. Tao, "Virtual indexing based methods for estimating node connection degrees," *Computer Networks*, vol. 56, no. 12, pp. 2773–2787, 2012.
- [13] L. Tang, Q. Huang, and P. P. Lee, "SpreadSketch: toward invertible and network-wide detection of superspreaders," in *Proceedings of the IEEE Conference on Computer Communications*, pp. 1608–1617, Toronto, ON, Canada, July 2020.
- [14] A. Chen, Y. Jin, J. Cao, and L. E. Li, "Tracking long duration flows in network traffic," in *Proceedings of the IEEE*

- Conference on Computer Communications*, pp. 1–5, San Diego, CA, USA, March 2010.
- [15] S. Lee, S.-H. Shin, and M. Yoon, “Detecting long duration flows without false negatives,” *IEICE-Transactions on Communications*, vol. E94-B, no. 5, pp. 1460–1462, 2011.
 - [16] H. Dai, M. Shahzad, A. X. Liu, and Y. Zhong, “Finding persistent items in data streams,” *Proceedings of the VLDB Endowment*, vol. 10, no. 4, pp. 289–300, 2016.
 - [17] P. Wang, P. Jia, J. Tao, and X. Guan, “Detecting a variety of long-term stealthy user behaviors on high speed links,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 10, pp. 1912–1925, 2019.
 - [18] S. A. Singh and S. Tirthapura, “Monitoring persistent items in the union of distributed streams,” *Journal of Parallel and Distributed Computing*, vol. 74, no. 11, pp. 3115–3127, 2014.
 - [19] H. Dai, M. Li, A. X. Liu, J. Zheng, and G. Chen, “Finding persistent items in distributed datasets,” *IEEE/ACM Transactions on Networking*, vol. 28, no. 1, pp. 1–14, 2019.
 - [20] S.-H. Shin and M. Yoon, “Virtual vectors and network traffic analysis,” *IEEE Network*, vol. 26, no. 1, pp. 22–26, 2012.
 - [21] Q. Xiao, Y. Qiao, M. Zhen, and S. Chen, “Estimating the persistent spreads in high-speed networks,” in *Proceedings of the IEEE 22nd International Conference on Network Protocols*, pp. 131–142, Raleigh, NC, USA, October 2014.
 - [22] Y. Zhou, Y. Zhou, M. Chen, and S. Chen, “Persistent spread measurement for big network data based on register intersection,” *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 1, pp. 1–29, 2017.
 - [23] H. Huang, Y.-E. Sun, C. Ma et al., “An efficient k-persistent spread estimator for traffic measurement in high-speed networks,” *IEEE/ACM Transactions on Networking*, vol. 28, no. 4, pp. 1463–1476, 2020.
 - [24] C. Estan, G. Varghese, and M. Fisk, “Bitmap algorithms for counting active flows on high speed links,” in *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, pp. 153–166, Miami Beach, FL, USA, October 2003.
 - [25] P. Wang, X. Guan, T. Qin, and Q. Huang, “A data streaming method for monitoring host connection degrees of high-speed links,” *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 1086–1098, 2011.
 - [26] “CAIDA,” <http://www.caida.org/data/overview>.
 - [27] “CERNET,” <http://iptas.edu.cn>.
 - [28] M. Yoon, T. Li, S. Chen, and J.-K. Peir, “Fit a compact spread estimator in small high-speed memory,” *IEEE/ACM Transactions on Networking*, vol. 19, no. 5, pp. 1253–1264, 2011.
 - [29] J. Wang, W. Liu, L. Zheng, Z. Li, and Z. Liu, “A novel algorithm for detecting superpoints based on reversible virtual bitmaps,” *Journal of Information Security and Applications*, vol. 49, pp. 1–9, 2019.
 - [30] S. Gao, Y. Yu, Y. Wang, J. Wang, J. Cheng, and M. Zhou, “Chaotic local search-based differential evolution algorithms for optimization,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 6, pp. 3954–3967, 2021.
 - [31] Y. Wang, S. Gao, M. Zhou, and Y. Yu, “A multi-layered gravitational search algorithm for function optimization and real-world problems,” *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 1, pp. 94–109, 2021.
 - [32] Y. Wang, S. Gao, Y. Yu, Z. Cai, and Z. Wang, “A gravitational search algorithm with hierarchy and distributed framework,” *Knowledge-Based Systems*, vol. 218, Article ID 106877, 2021.
 - [33] Y. Wang, Y. Yu, S. Cao, X. Zhang, and S. Gao, “A review of applications of artificial intelligent algorithms in wind farms,” *Artificial Intelligence Review*, vol. 53, no. 5, pp. 3447–3500, 2020.
 - [34] T. Jin, S. Gao, H. Xia, and H. Ding, “Reliability analysis for the fractional-order circuit system subject to the uncertain random fractional-order model with caputo type,” *Journal of Advanced Research*, vol. 32, pp. 15–26, 2021.
 - [35] T. Jin, H. Ding, H. Xia, and J. Bao, “Reliability index and asian barrier option pricing formulas of the uncertain fractional first-hitting time model with caputo type,” *Chaos, Solitons & Fractals*, vol. 142, Article ID 110409, 2021.