Journal of Network and Systems Management manuscript No.
(will be inserted by the editor)

Lightweight and Informative Traffic Metrics for Data Center Monitoring

Kuai Xu · Feng Wang · Haiyan Wang

the date of receipt and acceptance should be inserted later

Abstract In recent years, thousands of commodity servers have been deployed in Internet data centers to run large scale Internet applications or cloud computing services. Given the sheer volume of data communications between servers and millions of end users, it becomes a daunting task to continuously monitor the availability, performance and security of data centers in real-time operational environments. In this paper, we propose and evaluate a lightweight and informative traffic metric, *streaming frequency*, for network monitoring in Internet data centers. The power-series based metric that is extracted from the aggregated IP traffic streams, not only carries temporal characteristics of data center servers, but also helps uncover traffic patterns of these servers. We show the convergence and reconstructability properties of this metric through theoretical proof and algorithm analysis. Using real data-sets collected from multiple data centers of a large Internet content provider, we demonstrate its applications in detecting unwanted traffic towards data center servers. To the best of our knowledge, this paper is the first to introduce a streaming metric with a unique reconstruction capability that could aid data center operators in network management and security monitoring.

Keywords Cloud computing · Data center traffic · Streaming frequency · Unwanted traffic

K. Xu
Arizona State University, 4701 W. Thunderbird Road, Glendale, AZ 85306
E-mail: kuai.xu@asu.edu

F. Wang
Arizona State University, 4701 W. Thunderbird Road, Glendale, AZ 85306
E-mail: fwang25@asu.edu

H. Wang
Arizona State University, 4701 W. Thunderbird Road, Glendale, AZ 85306
E-mail: wangh@asu.edu

1 Introduction

In recent years, thousands of servers have been deployed in Internet data centers to support large scale online applications and cloud computing services. To ensure high availability, security and performance, real-time monitoring of data center servers becomes an important component of data center operations and management. However, given the size of data centers and the vast amount of network traffic, it is a daunting task to continuously monitor and analyze the behaviors of thousands of servers from IP traffic streams in real-time. The challenge of scalable data center monitoring calls for efficient metrics to summarize IP traffic streams.

The goal of this paper is to propose and evaluate *lightweight* and *informative* metrics from IP traffic streams in data center networks. *Lightweight* metrics enable fast processing and reporting of network traffic from high-speed links [1,2], while *informative* metrics provide appropriate thresholds for generating alerts, examining historical observations for forensic analysis, and pinpointing the root causes for operators to act upon. Many of the existing traffic metrics are either expensive or non-actionable. For example, collecting the incoming and outgoing packet statistics of each data center server over time requires a significant amount of computing as well as memory and storage resources due to the sheer amount of traffic data and the number of servers. In addition, real-time data center monitoring tools, e.g., MRTG [3], generate a variety of traffic graphs and statistics in each time interval that are often difficult for the operators to act upon without informative interpretation.

The massive data collected by traditional traffic measurement methods presents a significant challenge in storage, processing, and analysis. Therefore, there is a pressing need for simple yet effective metrics for data center monitoring. This paper introduces *streaming frequency (SF)*, a power-series based traffic metric, to characterize temporal traffic patterns of data center servers and to differentiate these servers with distinct patterns. Most existing metrics collected by network monitoring tools focus on observations on the current time window, rather than providing long-term temporal information. Through theoretical analysis, we demonstrate the informative aspects of this metric by showing two important properties of this metric: *convergence* and *reconstructability*. The streaming frequency metric has the convergence property since the value is always bounded between 0 and a certain thereshold, and the reconstructability property comes from the metric's capability of restoring the original traffic statistics. In general, the convergence property could help define and evaluate appropriate thresholds in the monitoring tools [3,4], while reconstruction is very important for in-depth analysis as it provides historical traffic patterns for trend analysis and change detections [5–7]. The simple streaming frequency metric not only has unique convergence and reconstruction properties that could aid data center operators in network management and security monitoring, but also significantly reduces the amount of storage and time for storing and processing traffic data compared with traditional traffic measurement methods.

The streaming frequency metric, carrying meaningful and actionable information, is very useful for real time monitoring, such as detecting unwanted traffic and troubleshooting network events. It also aids in understanding temporal traffic patterns given its ability to restore the original data traffic for in-depth analysis. In this paper, we propose and implement the reconstruction algorithm, and evaluate the metric using real network flow traces collected from three data centers in a large Internet content provider [8]. In addition, we demonstrate the effectiveness of this metric in detecting unwanted traffic towards data center networks with these data sets. The recent attacks towards GMail [9] and other cloud services illustrate the urgency and importance of protecting data center servers for ensuring the security and high-availability of cloud computing.

The contributions of this paper include

- proposing a new streaming frequency metric for real-time data center monitoring;
- performing a theoretical analysis of the metric and proposing an algorithm to reconstruct the traffic patterns from the streaming frequency;
- proposing a lightweight approximation for streaming frequency to reduce the memory cost of the metric and giving a theoretical analysis of its accuracy;
- demonstrating the applications of the streaming frequency metric using real data-sets collected from three data centers of a large Internet content provider.

The remainder of this paper is organized as follows: Section 2 describes the background and formulates the research problem. Section 3 introduces the *streaming frequency* metric, and Section 4 presents theoretical analysis. Section 5 implements the algorithm and discusses its computational complexity and memory consumptions. Section 6 demonstrates the applications of the metric in data center monitoring. Section 7 concludes this paper and outlines the future work.

2 Background and Problem Formulations

Data center networks have recently drawn much attention from research and industry communities. Most of the prior work has focused on reducing the cost of data centers and building scalable and robust data center architectures [10–15]. There has been little attempt to understand traffic characteristics and detect anomalous behaviors in data center networks.

To support large scale Internet applications, such as information search and online social networks, today's data centers host hundreds, even thousands, of servers that continuously communicate with a large number of distributed end users. At the same time, the servers also receive unwanted traffic from attackers in the forms of scanning, worms, viruses and the denial of service attacks. In [16], Benson et al. examined SNMP logs from data centers to study temporal

and spatial variations of the traffic load and packet loss at the aggregated link level. However, studying network traffic at the individual server level is often desired for a deep understanding of anomaly behaviors towards each data center server. In addition, monitoring the servers over a long period of time is very important for forensic and trending analysis. For example, sudden traffic increase or decrease of one server could indicate interesting network or system events, e.g., congestions or attacks. Cloud service providers (such as Amazon Web Services) monitor the inbound and outbound Internet data transfer of cloud customers' servers for billing purposes [17]. This accounting-related monitoring application mostly collects aggregated measurements on the total data received by and sent from the server; however, such coarse-grained monitoring does not reveal the underlying traffic patterns and network behaviors of these servers.

Monitoring data center traffic and detecting anomaly behavior from network data streams is not a trivial task given the large number of servers, their traffic, and the infinite duration of the monitoring period. A lot of existing techniques have been developed to collect various network metrics; such as packets, bytes, IP addresses, and ports from network flows or packet payloads. One shortcoming of these metrics is that such statistics is non-actionable or non-informative. In addition, efficiently storing and analyzing such information is also challenging. The database provides efficient processing and query for these metrics, however, the growth of data collections over time makes the database impractical. An alternative method is typically to store metrics into files saved on disks; however, reading data files through disk operations in real-time is time consuming.

In this paper we would like to answer the following research problem: *given the traffic streams from data center networks, how do we summarize the traffic with meaningful and actionable metrics that will aid operators in network management and security monitoring?* Specifically, we are interested in developing simple and lightweight metrics that could distinguish traffic patterns and reveal insight on the behavior of data center servers. We evaluate the metrics with real network flow traces collected from three major data centers a large Internet content provider [8]. Each network flow includes the well-known 5-tuple information (i.e., the source IP address, destination IP address, source port number, destination port number, and protocol), statistics of packets and bytes in the flow, and the start and end timestamps.

The solutions towards this goal are very useful for a variety of applications, such as performance monitoring and traffic engineering, since these metrics provide actionable information for the operators to find the root cause and perform correlation and forensic analysis. It is important to note that our proposed traffic metrics in this paper not only could be applied in data center traffic, but also could be extended to other types of network traffic such as Internet backbone traffic and enterprise network traffic. One of our future works is to study the feasibility of applying the proposed metrics on the traffic streams in backbone networks and enterprise networks.

3 Method

In this section, we first describe the traditional cumulative frequency and discuss its limitations. Subsequently, we propose an informative metric called streaming frequency to capture the underlying temporal traffic patterns.

3.1 Cumulative Frequency

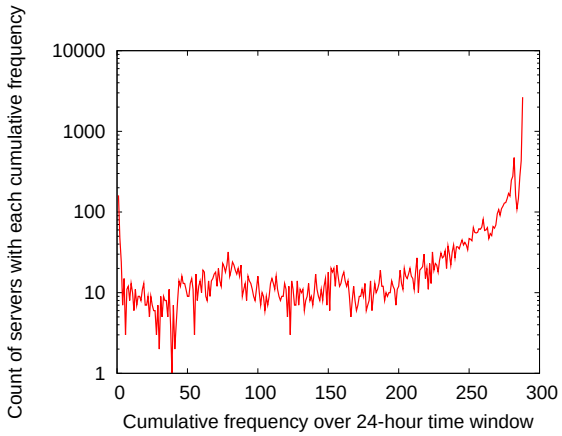


Fig. 1 The cumulative frequency of data center servers over a 24-hour time window

The basic on-off temporal metric cumulative frequency (CF) of data center servers captures the number of occurrences of a server in the aggregated IP traffic streams from all the source IP addresses towards the same server as the destination over a certain time period. Suppose each data center server is associated with χ, which is an indication function to represent whether there is traffic towards the server at a certain time interval. $\chi_i = 1$ if there is traffic towards the server at time interval i. Otherwise $\chi_i = 0$. The cumulative frequency, c, of a server is $c = \sum_{i=0}^{k} \chi_i$, where k is the index of the current interval. This cumulative frequency metric basically counts the number of occurrences for each data center server, which is an important statistic. However, it does not reveal any occurrence pattern of the server. For example, all of the following three temporal patterns have a cumulative frequency of 5:

Temporal Pattern 1: 0 0 0 0 0 1 1 1 1 1

Temporal Pattern 2: 0 1 0 1 0 1 0 1 0 1

Temporal Pattern 3: 1 1 1 1 1 0 0 0 0 0

Clearly, they exhibit different occurrence patterns. Thus, given a cumulative frequency of a server, it is difficult to restore or reconstruct when the

server is observed during the previous time periods. In other words, the cumulative frequency metric lacks the information on *when* and *in what pattern* the server is observed. Such information could be very useful, especially in network security monitoring and forensic analysis.

Figure 1 illustrates the cumulative frequency of over 82,000 data center servers during a 24-hour time window, which is divided into 288 time intervals with each interval being 5 minutes. The maximum cumulative frequency of a server is 288 if the server is observed with traffic during each time interval. From this plot, we find that a number of servers are not constantly observed, since their cumulative frequencies are less than 288. However, cumulative frequency only keeps track of the aggregated occurrence numbers of a server from the aggregated IP traffic streams and discards the information on *when* the server is observed; it is difficult to infer the original traffic pattern for the server. We introduce a new temporal frequency metric, namely, *streaming frequency (SF)*, which is able to not only restore the previous temporal patterns of the servers, but also summarize the temporal observations of data center servers with a bounded number.

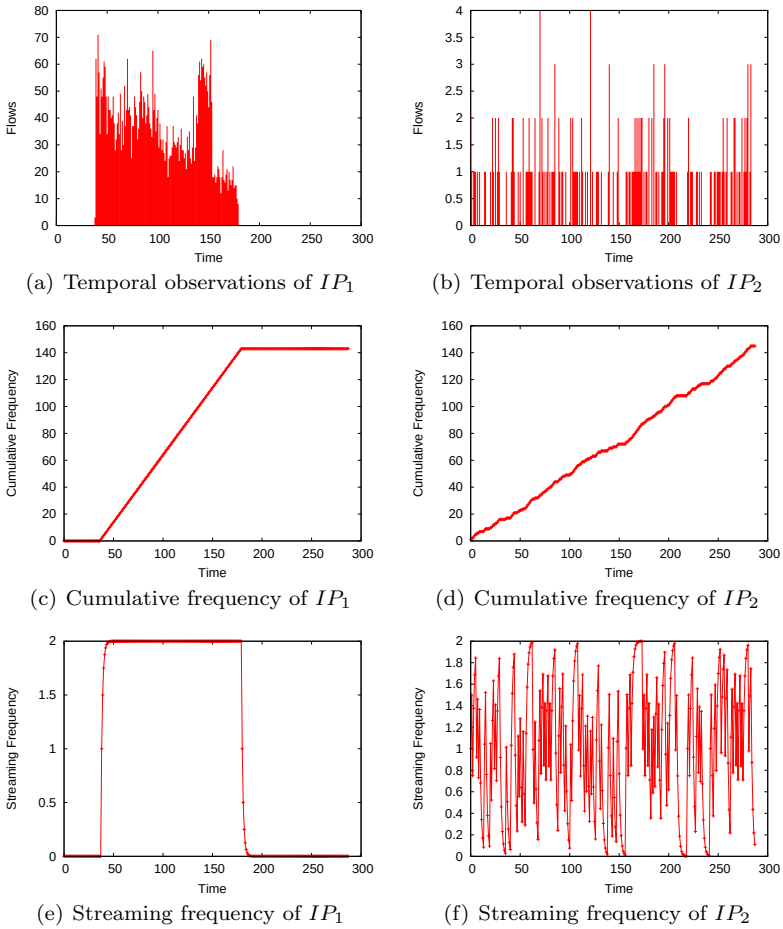
3.2 Streaming Frequency

Unlike the cumulative frequency metric which simply counts the number of time intervals during which a server is observed, the *stream frequency (SF)* assigns more weight to the recent time intervals, and assigns less weight towards "old" units. This new weight assignment is intuitive, since the current observation is more influenced by recent observations than historical ones. For a server, its streaming frequency s at the kth time interval is calculated with a power-series function as follows:

$$s_k = \chi_k + \frac{1}{\alpha} s_{k-1}, \tag{1}$$

where χ is the indication function which represents the on-off temporal occurrence of traffic towards the server. α is the decay parameter to reflect how fast the old observations decay, so $\alpha \geq 1$. In addition, we require α to be an integer for reconstruction purposes.

Equation 1 can be further expanded into



(a) Temporal observations of IP_1 (b) Temporal observations of IP_2

(c) Cumulative frequency of IP_1 (d) Cumulative frequency of IP_2

(e) Streaming frequency of IP_1 (f) Streaming frequency of IP_2

Fig. 2 Case studies of two IP addresses with middle temporal frequency

$$
\begin{aligned}
s_k =& \chi_k + \frac{1}{\alpha} s_{k-1} \\
=& \chi_k + \frac{1}{\alpha}\chi_{k-1} + \frac{1}{\alpha^2} s_{k-2} \\
=& \chi_k + \frac{1}{\alpha}\chi_{k-1} + \frac{1}{\alpha^2}\chi_{k-2} + \frac{1}{\alpha^3} s_{k-3} \\
=& \chi_k + \frac{1}{\alpha}\chi_{k-1} + \frac{1}{\alpha^2}\chi_{k-2} + \cdots + \frac{1}{\alpha^k}\chi_0 \\
=& \sum_{j=0}^{k} \frac{1}{\alpha^{k-j}} \chi_j
\end{aligned}
$$

As we can see from the above equations, streaming frequency is represented by a power series. In a special case where $\chi_i = 1$ for $i = 0, 1, \ldots, k$, the streaming frequency represents a geometric series. Streaming frequency has several interesting properties that are desirable for network monitoring, including *convergence* and *reconstructability*. We will demonstrate these properties through theoretical proof and experiments in the next two sections. In the rest of this section, we discuss the benefits of the convergence property.

First we revisit three earlier cases that have the same cumulative frequency of 5 but very different temporal patterns and we calculate their streaming frequencies. Based on equation 1, the streaming frequency metrics of these three patterns with $\alpha = 2$ are 1.9375, 1.3281, and 0.0605, respectively. This indicates that the streaming frequency metric can distinguish these temporal patterns with very different values, even though all of them have the same cumulative frequency. More importantly, these streaming frequencies also carry interesting information that the cumulative frequency does not carry. For example, the first value 1.9375, which is close to 2 (as we prove in Section IV where the streaming frequency of a server when $\alpha = 2$ is bounded by 2) tells us that the server is observed in most of the recent time intervals, while the value 0.0605 indicates the server is observed at the very beginning of the time period and does not appear in the latest time intervals.

Figures 2[a-f] further illustrate the advantages of the streaming frequency metric over the cumulative frequency metric through case studies. Figures 2[a][b] show the traffic patterns of two data center servers, IP_1 and IP_2 during nearly 300 time units, respectively. It is obvious that these two servers have similar numbers of occurrences but different traffic patterns. As shown in Figures 2[c][d], the two servers have the same cumulative frequency values at the 288 time unit, since they occur during the same number of time units. However, these two servers have very different temporal patterns, as IP_1 has a continuous traffic pattern, while IP_2 has a very random on-off pattern. Unlike the cumulative frequency metric, the streaming frequency metric with $\alpha = 2$ illustrated in Figure 2[e][f] clearly separates these two scenarios. This advantage is largely contributed by the convergence property of the streaming frequency metric whose value is always bounded between 0 and a certain thereshold.

Several prior works have exploited frequency measures and streaming analysis to study network traffic for anomaly detection. For example, in [18] Zhou et al. applied Discrete Fourier transform (DFT) to collect the frequency information for intrusion detection in a simulated network. In addition, [19–21] have applied signal processing, wavelet analysis, graph theory, and other techniques for network anomaly detections. In parallel, data streaming techniques have recently been used in a number of studies in the areas of real-time network monitoring [22–26]. In contrast to these work, the objective of our work is to develop lightweight and effective metrics based on a power-series that measures general traffic patterns.

In summary, we have introduced a power-series based traffic metric, namely streaming frequency, to summarize the traffic patterns of data center servers. Compared with the traditional cumulative frequency, which counts the number

of occurrences of a server, the streaming frequency metric carries the information on the on-off temporal patterns of the servers. In the next section, we will demonstrate some important properties of the streaming frequency metric through theoretical analysis.

4 Theoretical Analysis

In this section, we present the proof of the convergence and reconstructability properties of the streaming frequency metric.

Theorem 1 (Property of Convergence). For any server with indication function χ, its streaming frequency that is defined recursively as $s_k = \chi_k + \frac{1}{\alpha}s_{k-1}$, converges to $\frac{\alpha}{\alpha-1}$, when $\alpha > 1$ and α is an integer.

Proof: Since we are interested in the on-off traffic pattern of a server, χ_i can only be 0 or 1 for $i = 0, 1, ..., k$, we have

$$\begin{aligned} s_k =& \sum_{i=0}^{k} \frac{1}{\alpha^{k-i}}\chi_i \\ \leq& 1 + \frac{1}{\alpha} + \frac{1}{\alpha^2} + \cdots + \frac{1}{\alpha^k} \\ =& \frac{1 - \frac{1}{\alpha^k}}{1 - \frac{1}{\alpha}} \\ <& \frac{1}{1 - \frac{1}{\alpha}} \\ =& \frac{\alpha}{\alpha - 1} \end{aligned}$$

If $\alpha = 2$, the streaming frequency is bounded by $[0, 2)$. □

Before we prove the reconstructability property, we need to prove the following lemma. To simplify the proof, we introduce a function χ'. Suppose the current time interval is k, $\chi'_i = \chi_{k-i}$, that is $\chi'_0 = \chi_k, \chi'_1 = \chi_{k-1}, ..., \chi'_k = \chi_0$. Now $s = \sum_{i=0} k\chi'_i \frac{1}{\alpha^i}$. Now reconstructing the series χ_k, ..., χ_0 for a given s becomes solving the series χ'_0, ..., χ'_k and $X = \sum_{q=0}^{p-1} \chi'_q \frac{1}{\alpha^q}$.

Lemma 1: If at a certain time interval p, $\chi'_p = j$, then $s \in [X + \frac{j}{\alpha^p}, X + \frac{j}{\alpha^p} + \mu_p)$ for every $p = 0, 1, ..., k$, and $j \in \{0, 1\}$, where $X = \sum_{i=0}^{p-1} \chi'_i \frac{1}{\alpha^i}$ and $\mu_p = \frac{1}{\alpha^{p+1}} \frac{1}{1-\frac{1}{\alpha}}$.

Proof: At any time interval p where $p = 0, 1, ..., k$, we have

$$
\begin{aligned}
s=&\sum_{i=0}^{k} \chi'_i \frac{1}{\alpha^i} \\
=&\sum_{i=0}^{p-1} \chi'_i \frac{1}{\alpha^i} + \sum_{i=p}^{k} \chi'_i \frac{1}{\alpha^i} \\
=&X + \sum_{i=p}^{k} \chi'_i \frac{1}{\alpha^i}.
\end{aligned}
$$

It is clear that if $\chi'_p = j$, then $s \geq X + \frac{j}{\alpha^p}$ and

$$
\begin{aligned}
s=&X + \frac{j}{\alpha^p} + \sum_{i=p+1}^{k} \chi'_i \frac{1}{\alpha^i} \\
<&X + \frac{j}{\alpha^p} + \frac{1}{\alpha^{p+1}} \sum_{i=0}^{\infty} 1 \frac{1}{\alpha^i} \\
=&X + \frac{j}{\alpha^p} + \frac{1}{\alpha^{p+1}} \frac{1}{1-\frac{1}{\alpha}} \\
=&X + \frac{j}{\alpha^p} + \mu_p.
\end{aligned}
$$

Combining the two inequalities together, we have, if $\chi'_p = j$, then $s \in [X + \frac{j}{\alpha^p}, X + \frac{j}{\alpha^p} + \mu_p)$ for every $p = 0, ..., k$, and $j \in \{0, 1\}$. □

Theorem 2 (Property of Reconstructability). Given the streaming frequency s, we can uniquely decide the indication function χ with no ambiguity.

Proof: Note that χ'_p can only be 0 or 1. From Lemma 1, we have that if $\chi'_p = 0$, then $s \in [X, X + \mu_p)$ for every $p = 0, ..., k$. If $\chi_p = 1$, then $s \in [X + \frac{1}{\alpha^p}, X + \frac{1}{\alpha^p} + \mu_p)$, where $X = \sum_{i=0}^{p-1} \chi'_i \frac{1}{\alpha^i}$, and $\mu_p = \frac{1}{\alpha^{p+1}} \frac{1}{1-\frac{1}{\alpha}}$. Because $\alpha \geq 2$, it follows that $\frac{1}{\alpha^p} \geq \mu_p$. Thus $X + \mu_p \leq X + \frac{1}{\alpha^p}$, which means that the two ranges $[X, X + \mu_p)$ and $[X + \frac{1}{\alpha^p}, X + \frac{1}{\alpha^p} + \mu_p)$ do not overlap. This implies that at each time interval p, where $p = 0, 1, ...k$, we can uniquely decide the χ'_p based on the range in which $s - X$ falls. Therefore, χ can be decided uniquely. □

Theorem 2 shows that we can use the streaming frequency, s, to restore the indication function χ unambiguously. In the next section, we present the

reconstruction algorithm that takes streaming frequency as input and outputs the sequence of χ.

5 Implementation and Analysis of A Reconstruction Algorithm

In this section, we describe an algorithm to reconstruct the original *on-off* traffic pattern of a server from the streaming frequency, and analyze its computational complexity and memory requirement.

Algorithm 1 illustrates the reconstruction algorithm. It takes the incrementally updated streaming frequency s of a data center server at the current time interval k as input, and reconstructs the sequence of χ as output that represents the original *on-off* occurrence pattern of the data center server during the past k time intervals. As we proved in the previous section, there exists one and only one χ_i, $i = 0, 1, ...k$, given the streaming frequency s.

Algorithm 1 Server traffic pattern reconstruction algorithm

1: INPUT: $s = \sum_{i=0}^{k} \chi_i \frac{1}{\alpha^{k-i}}$, which is calculated for the current time interval k and $\chi_i = j$, where $j \in \{0, 1\}$;
2: OUTPUT: a sequence of χ_i, $i = 0, 1, ..., k$;
3: $s' = s$;
4: for $i = 0$ to k
5: $\quad \mu_i = \frac{1}{\alpha^{i+1}} \frac{1}{1-\frac{1}{\alpha}}$;
6: $\quad$ if $s' \in [0, \mu_i)$
7: $\quad\quad \chi_{k-i} = 0$;
8: $\quad$ else if $s' \in [\frac{1}{\alpha^i}, \frac{1}{\alpha^i} + \mu_i)$
9: $\quad\quad \chi_{k-i} = 1$;
10: $\quad$ end if
11: $\quad s' = s' - \chi_{k-i} \frac{1}{\alpha^i}$;
12: end for

The time complexity of this algorithm is $O(k)$ because the `for` loop (LINES 4 - 12) runs k times and each run takes $O(1)$ time.

During our experiments of running the algorithm against the real data-sets, one challenge lies in the limitation of significant digits in the popular programming languages. For example, the number of significant digits for double floating point numbers in `C`, and `Java` programming languages are 15. However, several existing libraries, e.g., the `java.math` package, support arbitrary precision floating point values for financial applications and cryptography.

Utilizing these packages, we are able to calculate s for any arbitrary k. In other words, our algorithm could restore the traffic patterns over an arbitrary length of time. On the other hand, the reconstruction comes with the cost of memory consumptions of storing s with an arbitrary precision. Figure 3 illustrates the relationship between the length of reconstruction history and memory assumptions for the streaming frequency metric. Similar to capturing the original temporal patterns in binary representations (cf. Section 3),

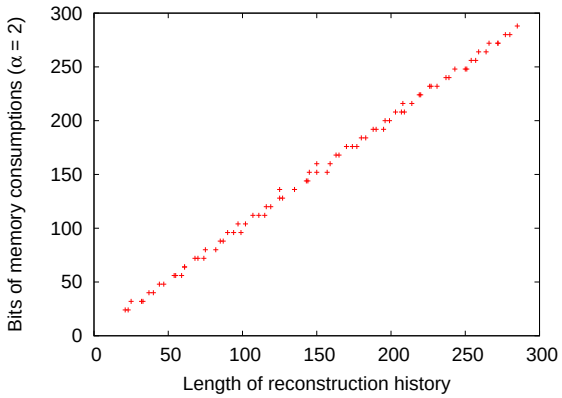
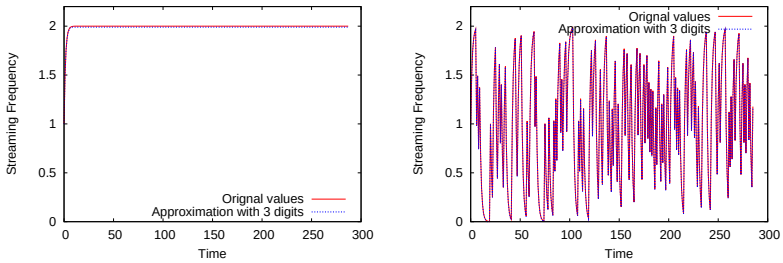


Fig. 3 Relationship between the length of the history and the memory consumptions

a longer history of traffic observations leads to more memory or storage, and the required space grows linearly as the length of the time series. For example, we need nearly 16 bytes (128 bits) for the streaming frequency when $k = 120$, while 32 bytes (256 bits) are required for $k = 240$. However, the binary representation of temporal patterns lacks the properties of convergence and reconstruction. In addition, we could use a fixed amount of memory space to represent the approximate value of streaming frequency for data center servers with a bounded error, as proved in Theorem 3.



(a) Case 1 (b) Case 2

Fig. 4 The difference of streaming frequency between original values and approximation values with three digits

As we can see, although streaming frequency is an informative metric that can distinguish traffic patterns and recover traffic history, the memory cost is undesirable. In order to reduce the memory consumption and calculation, we limit the number of digits representing the streaming frequency and show that this lightweight approximation can still achieve high accuracy. Specifically, the following theorem proves that we could use several digits to approximate

the streaming frequency metric over an arbitrarily long time period with a bounded error.

Theorem 3 (Approximation with a Bounded Error). If the number of digits that represent the streaming frequency is limited by x digits, the calculated s value is no more than $\frac{2}{10^{x-1}}$ apart from the precise value.

Proof. Since only x digits are used to represent the streaming frequency, all the digits starting from the $(x+1)$-th digit will be dropped at each step when the streaming frequency is updated. Thus at each step, the introduced error is at most $\frac{1}{10^{x-1}}$. For the subsequent step, the error introduced by the previous drop of digits is decreased by half, which is $\frac{1}{2}\frac{1}{10^{x-1}}$ and the newly introduced error is $\frac{1}{10^{x-1}}$. Thus, if we add all the errors up, the total error, ϵ, is

$$\begin{aligned}
\epsilon=&\frac{1}{10^{x-1}}+\frac{1}{2}\frac{1}{10^{x-1}}+\frac{1}{2^2}\frac{1}{10^{x-1}}+\frac{1}{2^3}\frac{1}{10^{x-1}}+...\\
=&\frac{1}{10^{x-1}}(1+\frac{1}{2}+\frac{1}{2^2}+\frac{1}{2^3}+...)\\
\leq&\frac{1}{10^{x-1}}*\frac{1-(1/2)^{\infty}}{1-1/2}\\
=&\frac{2}{10^{x-1}}.
\end{aligned}$$

That is, if we use only three digits to represent streaming frequency, the calculated s is less than 0.02 apart from the precise streaming frequency no matter how large k is. If four digits are used, the error is bounded by only 0.002. Figure 4 illustrates the streaming frequencies for two sample data center servers over time. In both figures, the red solid lines represent the precise values with an arbitrary size of memory, while the blue dotted lines denote the approximation values with only 3 digits. Figure 4 confirms that the approximations are very close to the original values, thus we could indeed use a small amount of memory in our experiments and real-time systems to represent the streaming frequency of data center servers over a long time interval. Such observations hold for other servers as well.

In summary, the time complexity of the reconstruction algorithm is $O(k)$ for reconstruction of traffic pattern over a period of k time intervals. Furthermore, our theoretical analysis and experiment results show that the approximation method with a small number of digits could achieve a very high accurate estimation on the original value. Therefore, we could use approximation methods in real-time systems if the memory constraint exists.

6 Applications

Lightweight and informative streaming frequency metrics have a variety of applications in real-time data center network monitoring given their convergence and reconstructability properties. In this section, we demonstrate the applications in detecting unwanted traffic towards data center servers.

As the number of data center servers grows due to the rapid growth of cloud computing, unwanted traffic towards these servers continues to increase in size and diversity [27,28]. The sheer volume of network traffic makes it challenging to examine unwanted traffic directly from continuous traffic streams, thus streaming frequency provides an efficient mechanism for identifying unwanted traffic given its lightweight computation and unique convergence and reconstruction abilities.

As discussed in the previous sections, the streaming frequency of a given host h is $1 \leq s \leq 2$ if the host h is observed at the time unit t. Clearly, if $s = 1$, it indicates that the host h is the first to be observed at the unit time, while if $s \approx 2$, it indicates that the host h is consistently observed during all the previous time units. On the other hand, any other value (i.e., $1 < s < 2$), suggests that the host is not *always* observed over time and thus, it is very interesting to examine its detailed temporal patterns over the time.

To further understand the correlation of the streaming frequency and the temporal traffic patterns, we compute the mean and standard deviation of the streaming frequency for all data center servers. The mean of the streaming frequency for the host h is computed as

$$\mu(h) = \frac{1}{t}\sum_{i=1}^{t} s_i,$$

while the standard deviations of the streaming frequency is calculated as

$$\sigma(h) = \sqrt{\frac{1}{t}\sum_{i=1}^{t}(s_i - \mu(h))^2}.$$

Figure 5 illustrates a scatter plot of the mean and standard deviation for the hosts in three data centers with $1 < s < 2$ and the same cumulative frequency of 144. This graph shows two well isolated groups among these hosts, which motivates us to apply simple clustering algorithms for dividing them into separate clusters for further analysis. Through k-means clustering algorithms [29] with $k = 2$, we further find the centroids of these two clusters: c1 (1.529, 0.090) and c2 (1.960, 0.027). The in-depth analysis shows that the top-left cluster are *inactive* servers or unused IP addresses that are the targets of random scanning and exploit behaviors from outside attackers, since most of the traffic is towards ports that are associated with well known vulnerabilities. The bottom-right cluster are normal *active* data center servers, and most of the traffic is towards legitimate services. Note that some of the traffic towards active servers could also be unwanted traffic from malicious sources exploiting

active servers. Although the capability of differentiating active and inactive servers in data centers is not new, the unique benefits of streaming frequency lie in its low cost in metric computation and storage over continuous traffic streams.

To uncover this anomalous traffic, network operators could analyze all the traffic towards active servers and find unusual ports or access patterns. Based on these empirical observations, we calculate its distances[1] to the centroid $d_1 = |s - \mu_{c1}|$, and $d_2 = |s - \mu_{c2}|$ for each new host h, and use the following rule to classify the hosts observed from the aggregated data center traffic streams:

$$type\ of\ server = \begin{cases} inactive, & \text{if } d_1 \leq d_2, \\ active, & \text{if } d_1 > d_2, \end{cases} \tag{2}$$

Prior works [30–32] suggested that hosts that talk to inactive servers are suspicious, as the majority of such data communications is due to prevalent random scanning and malicious activities [33,34]. Using additional traffic features such as the number of packets and bytes, we find that hosts talking with inactive servers only send a single packet to these servers on destination ports associated with well-known vulnerabilities, which confirms the conjectures in previous studies. More importantly, [31,32] have also found that the attackers that communicate with inactive servers are also interested in scanning active servers as they randomly select potential targets or victims during the initial exploit phases. If these active servers are compromised, the normal Internet applications or cloud computing services could be disrupted. Therefore, it is very intuitive and important to further study all the data traffic sent from the anomalous sources that talk with those inactive servers. Such a recursive detection technique is simple yet effective to detect unwanted traffic towards data center servers.

After identifying inactive servers, we then locate suspicious sources that talk with them and study all network traffic from such sources towards active and inactive servers respectively. The analysis of anomalous behavior could augment intrusion detection solutions deployed in Internet data centers by detecting stealthy or low-volume behavior, such as port 22 activity after scanning traffic. In other words, the deployment of this proposed method could protect servers with popular services and ensure the normal operations of Internet data centers. Figures 6[a][b] illustrate the difference of traffic volumes towards active and inactive servers from the same anomalous sources, respectively. In both plots, the x axis represents the number of servers exploited by each source, while the y axis represents the total number of network flows towards these servers from each source. As shown in Figures 6[a][b], each anomalous source tends to communicate with active servers with more than one network flow, while each source tends to communicate with the majority of inactive servers with just one network flow. The additional flows toward the active servers

[1] We do not include the variance in calculating the distance, since calculating the variance requires reconstructions at each time unit.

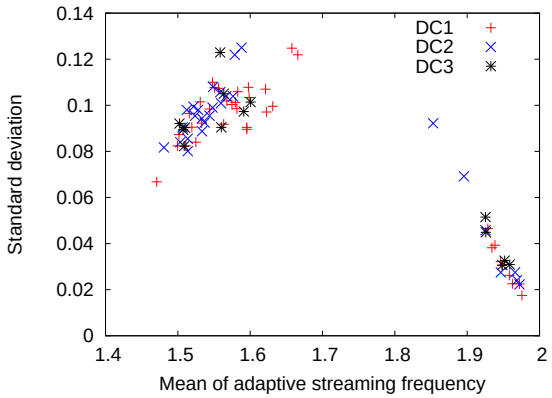
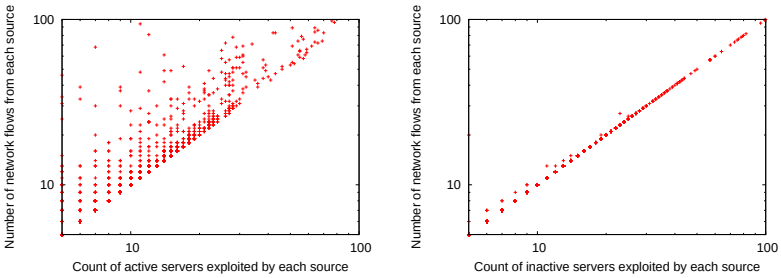


Fig. 5 The clustering of the mean and standard deviation of streaming frequencies for the hosts with $1 < s < 2$ and the same cumulative frequency

are mostly follow-up activities of these anomalous sources, after the active servers become the potential targets for further activities. Although streaming frequency does not identify specific exploit patterns, its capability of differentiating active and inactive servers provides valuable and critical input in detecting suspicious activities from anomalous sources. In other words, identifying inactive servers is a first and important step in detecting and reducing unwanted traffic in Internet data centers.

Due to the difficulty of establishing the ground truth on the attacks in the real traffic datasets, we rely on the simulated scanning, worm propagation and DDoS behaviors to study the false positive rate and false negative rate. We use known unwanted traffic from 3728 anomalous events identified in our previous studies [35,36] and combine them with real data center traffic. To generate the synthetic traffic with consistent data center IP addresses, we replace the destination addresses of unwanted traffic with random IP addresses in data center networks. Subsequently, we apply streaming frequency to identify inactive servers and find suspicious sources that send the unwanted traffic. Our simulation results show that our proposed method is able to discover over 95% of anomalous behaviors, i.e., a false negative rate of less than 5%. The major reason for false negatives is that the sources of unwanted traffic target mostly active data centers server without communicating with inactive servers.

We also study the distributions of legitimate Internet services and unwanted traffic by examining the destination ports in network traffic flows. Table 1 lists the top 10 destination ports and transport protocols towards the active and inactive servers during one 5-minute observation time window. For active servers, the destination ports towards them are widely used Internet services, e.g., HTTP, SMTP, POP3, DNS. On the other hand, the destination ports towards inactive servers are mostly those associated with well-known vulnerabilities, e.g., 1434 and 1026-1027, ports for direct accesses, e.g., SSH as well as unknown ports. The traffic towards unknown ports is likely from



(a) Traffic towards active servers from each source

(b) Traffic towards inactive servers from each source

Fig. 6 The difference of traffic volumes towards active and inactive servers originating from the same anomalous sources

the aggressive attackers who attempt to exploit all service ports. These findings suggest that the proposed method can identify emerging unwanted traffic. Although packet payload is not available for in-depth investigation, the observations on application ports provide significant evidences of traffic variations towards active and inactive servers.

Table 1 Top 10 destination ports towards active and inactive servers.

Towards active servers			Towards inactive servers		
Rank	Port/Protocol	Services	Rank	Port/Protocol	Services
1	80/TCP	HTTP	1	22/TCP	SSH
2	25/TCP	SMTP	2	21/TCP	FTP
3	5510/TCP	VoIP	3	7212/TCP	Unknown
4	5505/TCP	Messenger	4	8000/TCP	Unknown
5	443/TCP	HTTP over SSL	5	25/TCP	SMTP
6	995/TCP	POP3 over SSL	6	1026/UDP	Windows Messenger
7	110/TCP	POP3	7	1027/UDP	Windows Messenger
8	143/TCP	IMAP	8	1434/UDP	Microsoft SQL Server
9	53/UDP	DNS	9	80/TCP	HTTP
10	465/TCP	SMTP over SSL	10	4899/TCP	Remote Adminstration

In addition, we also correlate unwanted traffic across data centers. Such correlation is very important, since it could reveal more insight on unwanted traffic patterns, and help detect the behaviors that are difficult to discover from single vantage points. The focuses of the correlations are on the anomalous sources and the destination ports. For the anomalous sources, we study whether there exists aggressive attackers that are exploiting targets across data centers. For the destination ports, we are interested in the similarity and dissimilarity of the port distribution of unwanted traffic across data centers. By correlating the source IP addresses of the anomalous traffic, we find that there are indeed many aggressive attackers that exploit various servers across data centers. The traffic from these sources could be blocked through an access control list deployed on the firewalls. For the destination port distribution across data centers, we find that the observations on the port distribution towards inactive servers in Table 1 hold for all three data centers. During the

early phases of a worm outbreak, the correlation analysis of unwanted traffic across data centers could provide early warning to network operators.

7 Conclusions and Future Work

This paper proposes an informative and actionable traffic metric for monitoring the temporal frequency and traffic patterns of data center servers. Through theoretical analysis and algorithm implementations, we demonstrate the characteristics of convergence and reconstructability of these metrics. More importantly, this simple yet effective metric provides insights of the temporal patterns of data centers due to these characteristics. Using real data-sets collected from data centers of a large Internet content provider, we demonstrate the applications of this metric on detecting unwanted traffic towards data centers servers. As part of our on-going research, we are currently exploring the applications of streaming metrics on additional traffic features such as traffic volume or application ports for multi-dimensional traffic analysis.

References

1. H. Dreger, A. Feldmann, V. Paxson and R. Sommer, in *Proc. of International Symposium On Recent Advances In Intrusion Detection* (2008)
2. S. Bhatia, A. Kumar, M. Fiuczynski, and Larry Peterson, in *Proc. of USENIX Symposium on Operating Systems Design and Implementation* (2008)
3. MRTG. Multi Router Traffic Grapher. `http://oss.oetiker.ch/mrtg/`
4. C. Estan, S. Savage, G. Varghese, in *Proc. of ACM SIGCOMM* (2003)
5. S. Kornexl, V. Paxson, H. Dreger, A. Feldmann, and R. Sommer, in *Proc. of ACM SIGCOMM conference on Internet Measurement* (2005)
6. B. Krishnamurthy, S. Sen, Y. Zhang, Y. Chen, in *Proc. of ACM/USENIX IMC* (2003)
7. T. Karagiannis, K. Papagiannaki and M. Faloutsos, in *Proc. of ACM SIGCOMM* (2005)
8. Yahoo! Labs. Datasets. `http://labs.yahoo.com/`
9. Wall Street Journal, `http://online.wsj.com/article/BT-CO-20100112-716798.html` (2009)
10. M. Al-Fares, A. Loukissas, A. Vahdat, in *Proc. of ACM SIGCOMM* (2008)
11. C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, S. Lu, in *Proc. of ACM SIGCOMM* (2008)
12. D. Joseph, A. Tavakoli, I. Stoica, in *Proc. of ACM SIGCOMM* (2008)
13. PC World. Microsoft: Datacenter Growth Defies Moore's Law. `http://www.pcworld.com/article/130921/article.html`
14. A. Greenberg, J. Hamilton, D. Maltz, and P. Patel, ACM SIGCOMM Computer Communication Review **39**(1), 68 (2009)
15. S. Nedevschi, J. Padhye, and S. Ratnasamy, in *Proc. of USENIX OSDI Workshop on Power Aware Computing and Systems* (2008)
16. T. Benson, A. Anand, A, Akella, and Ming Zhang, in *Proc. of SIGCOMM WREN Workshop* (2009)
17. Amazon. Amazon Web Services. `http://aws.amazon.com/`
18. M. Zhou and S.-D. Lang, Systemics, Cybernetics and Informatics **2** (2004)
19. P. Barford, J. Kline, D. Plonka, A. Ron, in *Proc. of ACM SIGCOMM IMW* (2002)
20. W. Lu and A. Ghorbani, EURASIP Journal on Advances in Signal Processing (2009)
21. M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese, in *Proc. of ACM SIGCOMM Internet Measurement Conference* (2007)
22. G. Cormode, F. Korn, S. Muthukrishnan and D. Srivastava, in *Proc. of VLDB* (2003)
23. A. Kumar, J. Xu, L. Li, and J. Wang, in *Proc. of ACM SIGCOMM Conference on internet Measurement* (2003)

24. S. Muthukrishnan, *Data Streams: Algorithms and Applications* (Now Publishers, 2005)
25. E. Cohen, N. Duffield, H. Kaplan, C. Lund and M. Thorup, in *Proc. of ACM SIGCOMM Internet Measurement Conference* (2007)
26. A. Ramachandran, S. Seetharaman, N. Feamster and V. Vazirani, in *Proc. of ACM SIGCOMM Internet Measurement Conference* (2008)
27. M. Armbrust, A. Fox, Armando, R. Griffith, A. D. Joseph, R. Katz, Randy, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, Communications of the ACM **53**(4), 50 (2010)
28. T. Ristenpart, E. Tromer, H. Shacham, S. Savage, in *Proc. of ACM Conference on Computer and Communication Security (CCS)* (2009)
29. J. Mcqueen, in *Proc. of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (1967)
30. Y. Jin, E. Sharafuddin, and Z.-L. Zhang, in *Proc. of International Conference on emerging Networking EXperiments and Technologies (CoNext)* (2007)
31. Y. Jin, G. Simon, K. Xu, Z.-L. Zhang, and V. Kumar, in *Proc. of USENIX NSDI Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML)* (2007)
32. Y. Jin, Z.-L. Zhang, K. Xu, F. Cao, and S. Sahu, in *Proc. of ACM SIGMETRICS MineNet Workshop* (2007)
33. V. Yegneswaran , P. Barford and J. Ullrich, in *Proc. of ACM SIGMETRICS* (2003)
34. R. Pang, V. Yegneswaran, P. Barford, V. Paxson and L. Peterson, in *Proc. of ACM SIGCOMM IMC* (2004)
35. K. Xu, Z.L. Zhang, S. Bhattacharyya, in *Proceedings of Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI)* (2005)
36. K. Xu, Z.-L. Zhang, and S. Bhattacharyya, IEEE/ACM Transactions on Networking **16**, 1241 (2008)

Author Biographies

Kuai Xu is currently an assistant professor at Arizona State University. He received his Ph.D. degree in computer science from the University of Minnesota in 2006, and his B.S. and M.S. degrees from Peking University, China, in 1998 and 2001. His research interests include network security and cloud computing. He is a member of ACM and IEEE.

Feng Wang received the BS degree from Wuhan University in 1996, MS degree from Beijing University in 1999, and PhD degree from the University of Minnesota, Twin Cities in 2005, all in computer science. She is currently an assistant professor at Arizona State University. Her research interests include wireless networks, social networks, computational biology, and combinatorial optimization. She is a member of the IEEE.

Haiyan Wang received his B.S in mathematics from Northwest Normal University, China, 1985. He received M.S. in Computer Science and Ph.D. in Mathematics from Michigan State University, East Lansing, in 1997. He is an Assistant Professor in the Division of Mathematical and Natural Sciences at Arizona State University. His current research interests include applied mathematics, differential equations and mathematical biology.