Computing (2019) 101:339–361
https://doi.org/10.1007/s00607-018-0619-4

Multivariate network traffic analysis using clustered patterns

Jinoh Kim[1] · **Alex Sim**[2] · **Brian Tierney**[3] · **Sang Suh**[1] · **Ikkyun Kim**[4]

Received: 29 May 2017 / Accepted: 19 April 2018 / Published online: 28 April 2018

Abstract Traffic analysis is a core element in network operations and management for various purposes including change detection, traffic prediction, and anomaly detection. In this paper, we introduce a new approach to online traffic analysis based on a pattern-based representation for high-level summarization of the traffic measurement data. Unlike the past online analysis techniques limited to a *single* variable to summarize (e.g., sketch), the focus of this study is on capturing the network state from the *multivariate* attributes under consideration. To this end, we employ *clustering* with its benefit of the aggregation of multidimensional variables. The clustered result represents the state of the network with regard to the monitored variables, which can also be compared with the observed patterns from previous time windows enabling intuitive analysis. We demonstrate the proposed method with two popular use cases, one for estimating state changes and the other for identifying anomalous states, to confirm its

✉ Jinoh Kim
jinoh.kim@tamuc.edu

Alex Sim
asim@lbl.gov

Brian Tierney
bltierney@es.net

Sang Suh
sang.suh@tamuc.edu

Ikkyun Kim
ikkim21@etri.re.kr

1 Texas A&M University, Commerce, TX 75428, USA

2 Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

3 ESnet, Berkeley, CA 94720, USA

4 ETRI, Daejeon 305-700, Korea

feasibility. Our extensive experimental results with public traces and collected monitoring measurements from ESnet traffic traces show that our pattern-based approach is effective for multivariate analysis of online network traffic with visual and quantitative tools.

Keywords Network traffic analysis · Clustered patterns · Change detection · Anomaly detection · Multivariate analysis

Mathematics Subject Classification 68Uxx Computing methodologies and applications

1 Introduction

Monitoring network traffic is an integral part of network operations and management. The basic requirement for network monitoring is to effectively capture the traffic dynamics in a timely manner. For example, some anomalies are the indication of performance bottlenecks with a huge number of simultaneous connections, which may be caused by several reasons such as flash crowds, denial of service attacks, or network component failures. The monitored results are then used for reconfiguring the network to optimize performance or to reinforce security with the historical information.

A crucial challenge for traffic analysis for monitoring is the exponential increase in the volume of data [1,2]. This should be more critical to online monitoring with a large number of incoming/outgoing packets to be processed within a limited time interval. For example, it can be over tens of millions of packets per second for a single 10 Gbps link. For this reason, in-depth analysis (e.g., on a per-flow basis) would not be feasible for online monitoring, particularly in a large-scale network (e.g., ISP or enterprise networks). In response to this, data streaming computation has been studied to provide a summary of network traffic within a time interval. For instance, sketch is a statistical summary technique based on counting with hashing to analyze the network traffic data [3–7]. While decent for streaming computation without keeping extravagant per-flow data, a sketch is limited to producing the statistics for a single traffic variable. The probabilistic density information was also considered as the snapshot of the network traffic in a fixed amount of time [8]. However, the techniques in this class have the same problem, and are limited to one dimensional variable to analyze. As a result, the individual variables need to be analyzed independently, and how to combine the observations from multiple variables is left to the administrators.

In this paper, we propose a new approach that offers a *high-level summary* of the online network traffic in regard to the multivariate attributes under consideration. With this approach, what the administrators see is an abstract *pattern* compiled from the traffic variables being monitored, rather than observing a bunch of the independent statistical information for the variables. By doing so, the detection problems in traffic analysis (e.g., change detection and anomaly detection) can be reduced to the pattern comparison problem. For ease of exposition, we use a term "network state" defined as the recap of the network traffic with respect to the monitored variables. The observed pattern represents the network state for the associated time interval, and it can be compared with the patterns in the archive to interpret the traffic dynamics.

To represent the network state, we employ *clustering* with its benefit of the aggregation of multidimensional attributes. The clustering result is taken as a pattern that tells the network state for a given time interval. In addition, the connected clustering statistics (e.g., centroid positions) can be used to compare patterns to provide a tool for the quantitative analysis. Note that the focus of this paper is fundamentally different from the past studies employed clustering for anomaly detection [9–12], the primary concern of which is to test individual data points to classify into either normal or anomalous. The main interest of our research is to develop a method to define network states from the given multivariate traffic variables.

In this paper, we present our new approach along with two use cases: *change detection* and *anomaly detection*. The first use case is the estimation of traffic changes between two different time intervals. The network state is captured with a static number of clusters, and the change is measured with the "degree of changes" established based on the movement of the centroid coordinates. The second use case we demonstrate is anomaly detection. In this use case, we assume a dynamic number of clusters for a time window, and the abnormality of a cluster is estimated with the patterns of suspicious activities clusters using a new measure of "likelihood" constructed based on the centroid positions and the sum of squares information.

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

- We present a new multivariate traffic analysis technique using clustered patterns to provide a high-level summary of the traffic data to enable intuitive network monitoring with visual patterns;
- We establish a set of quantitative measures ("degree of changes" and "likelihood of suspicious activities") as supportive means to the visual patterns for comprehensive analysis;
- We demonstrate the proposed technique with two common network traffic analysis applications of change detection and anomaly detection with two public data sets (UNIBS data [13] and KDDCup 1999 data [14]);
- We apply our technique to the traffic measurement collection from Energy Sciences Network (ESnet) [15] recently collected (May 2016) to confirm the feasibility.

This paper is organized as follows. Section 2 summarizes the related studies with respect to network monitoring techniques based on graphical tools, probabilistic distributions, hashing-based sketch, and clustering techniques. We then discuss a traditional approach to network traffic analysis using probabilistic distributions as preliminary studies in Sect. 3, and introduce our new approach of clustered pattern for multivariate traffic analysis with the use case of change detection in Sect. 4. Section 5 presents how the clustered patterns can be used to identify anomalous network states using the pattern-based approach. Section 6 demonstrates the analysis of the ESnet [15] `tstat` trace using the clustered pattern technique. Finally, we conclude our presentation in Sect. 7.

2 Related work

Monitoring network traffic using graphical means have been studied in several past studies. In the BLINC work [16,17], the authors focused on the patterns of host behav-

ior at the transport layer, using four tuples of source and destination IP addresses and port numbers, particularly for traffic classification. The BLINC idea was further extended using a traffic dispersion graph (TDG) that models "social behavior of hosts" using graph notations, which can capture patterns of node interactions [16]. A benchmark tool known as NeTraMark [18] implements BLINK and TDG, as well as machine learning algorithms including C4.5 Decision Tree, k-Nearest Neighbors, and Support Vector Machines, for the purpose of network traffic classification. Recently, the authors in [19] developed a visualization tool (ReView) to discover traffic dependency. The focus of this study is on detecting malicious events based on the causality analysis. Compared to the past work, we take a different approach that aggregates multivariate variables into clustered patterns to show a high-level summary of network states to enable intuitive analysis.

With the heavy increase in traffic volumes, flow-level traffic analysis would not be feasible for high-speed network monitoring. The sketch technique has been extensively studied to build high-level summaries of the traffic data (rather than maintaining expensive per-flow information) to deal with the change detection and heavy-hitter detection problems [3–7]. Sketch is a probabilistic summary technique using a set of hash functions. A single variable is aggregated in the sketch data structure based on hashing for a time interval. The change is then measured by comparing the summary with the predicted summary using forecasting techniques (e.g., ARIMA). A recent study [20] enhances the sketch technique to catch up the rising line rates (10–100 Gbps per port over 10–100 ports) using a concept of HashPipe that maintains counts of heavy flows in a pipeline of hash tables to track the k heaviest flows (i.e., for heavy-hitter detection).

Probabilistic models and samplings have also been considered to summarize the network traffic data by employing frequency counting [21], histogram [22], sliding windows [23], random sampling [24], wavelets [25], and dimensionality reduction [26]. One of the promising studies in [8] proposed a dynamic sampling technique to reduce the size of streaming data based on the difference of the probabilistic density. The authors employed the Kolmogorov–Smirnov test (KS test) to calculate the distance of two summaries observed in different time intervals to measure the change. The sketch and probabilistic modeling techniques are however limited to capture a single traffic variable only, and individual variables should be independently analyzed. The key difference of the proposed approach is the ability to capture the multivariate traffic variables to provide a comprehensive view of the traffic data.

A large body of studies employed clustering for intrusion/anomaly detection [27–31]. The main focus of many of the past studies was on identifying intrusive events *individually* with a trained model consisting of normal or anomalous behaviors. The first use of clustering was by [9], in which the authors made an assumption that anomalous events would be much smaller than the normal events in quantity. The clustering information is then used to test whether the input is anomalous or not, based on this assumption. Similarly, a recent work [32] proposed a method using co-clustering to improve the detection performance for individual connections. The main focus of our work is fundamentally different from the past work. Our interest in this work is in building aggregated summaries to enable the traffic analysis process in a collective way.

Clustering is the most widely used tool for unsupervised learning. Several different types of clustering have been developed, including centroid-based, hierarchical, density-based, and model-based method. According to a recent comparison study of clustering techniques with a large data set in smart grid [33], the centroid-based method (e.g., k-means) works better than the other clustering methods with respect to compactness (i.e., the elements in a cluster are very close to one another). The authors in [34] stress the speed and simplicity of k-means technique for clustering in practice, which is the main reason why it has maintained its popularity. In addition, it can be more scalable through optimizations including parallelism (e.g., a parallel version of the k-means++ [34]). In this work, we employ k-means clustering to create clustered patterns with its simplicity and popular use. As we will discuss shortly, the centroid-based clustering is also beneficial with a low complexity for the representation of a pattern, which is a vector of cluster centroid position and density that is derived from the population and the sum of squares in the cluster.

3 Preliminary study using distributions

A simple form of network traffic monitoring that keeps track of the volume of incoming/outgoing traffic in the network would not be comprehensive, and can only provide a partial characteristic of the network state. One of the traditional approaches is to scrutinize probabilistic distributions of essential variables related to traffic statistics, obtained from the data set collected within a predetermined time interval (known as a time window). For example, the distribution of packet lengths can be referred to the characterization of the network state in the current time window. However, a non-trivial challenge with this approach is an exponential increase of computational complexity with additional variables to be monitored and larger time windows [8]. Since the network administrator may want to include multiple traffic-related attributes in the monitoring process, multivariate analysis is commonly needed for network monitoring, making it less attractive to employ such distribution-based monitoring.

For this study, we employed a 16-h trace excerpted from the UNIBS traffic trace, between 10 a.m. on September 30, 2009 and 2 a.m. on October 1, 2009 [13]. The data set contains the information for network flows[1] with timing, and the ground-truth data with the associated application for each connection is provided [35]. The average number of flows is 789 flows/hour with a high degree of variance (min = 20, max = 7052).

Figure 1 shows cumulative distributions for two variables (flow duration and average number of packets in flows) over 16 monitoring time windows, each of which has 1-h in length. From the figure, we observed that some time windows show somewhat similar plots, while some others look different one another. For instance, we can see that the two CDF plots at 10 and 11 a.m. look largely different. In contrast, for example, the plots from 11 a.m. to 5 p.m. are fairly similar each other although not quite clear.

[1] A flow is identified with five tuples of source IP address, source port number, destination IP address, destination port number, and protocol in TCP/IP header.

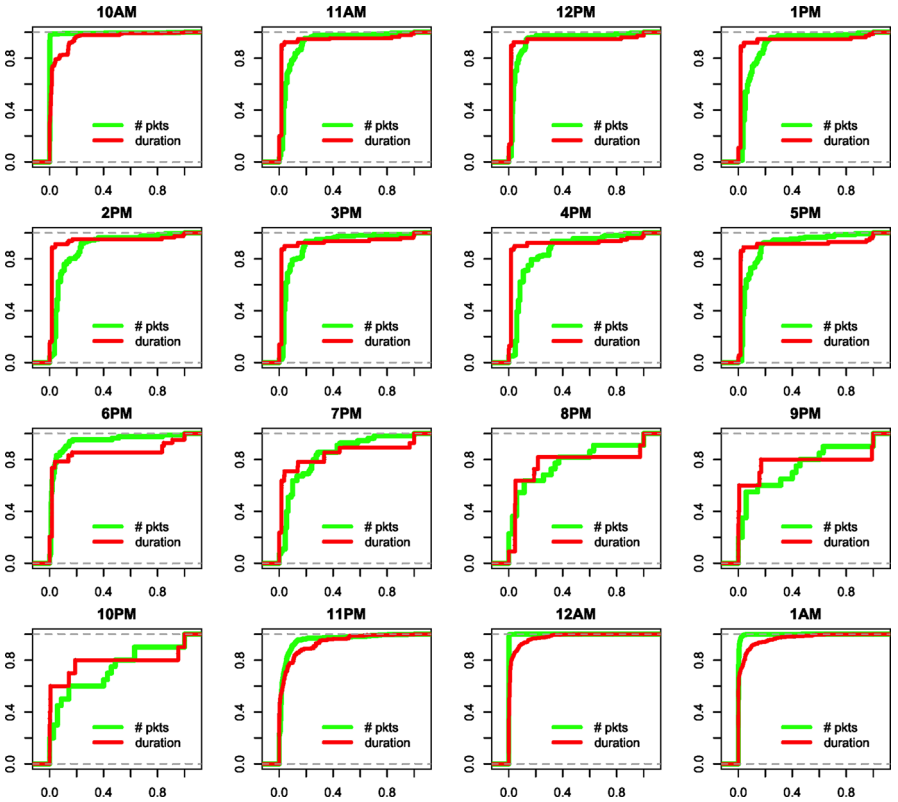

Fig. 1 Normalized cumulative distributions of two network traffic attributes (flow duration and average number of packets in flows) over 16 consecutive time windows, on UNIBS trace between 10 a.m. on Sep. 30, 2009 and 2 a.m. on Oct. 1, 2009

To better understand the characteristics of the given traffic data set, we analyzed the composition of applications in each window. The breakdown of applications was performed using the associated ground-truth data provided together with the traces [35]. We agree that it is hard to explain the network state only through the compositions of applications. However, we believe that it is a good reference to infer network states. Figure 2 shows the compositions of applications over the time windows. It shows that some time frames are highly related, such as 11 a.m.–5 p.m. and 8–10 p.m., with respect to application compositions. In contrast, some show strongly unrelated breakdowns with other windows, such as in 10 a.m., 11 p.m., 12 p.m., and 1 a.m..

From the distributions shown in Fig. 1, those related time windows in the application breakdowns also show similar patterns (e.g., 11 a.m.–5 p.m.) in the plots. However, some plots do not seem to agree on the breakdown information. For example, two windows for 12 and 1 a.m. show a very close pattern although the compositions of applications for those windows are highly distinctive each other. From the result, network monitoring using distributions would be useful but only to some extent. In addition to visual monitoring, one may want to measure the difference of the distributions of time windows using distribution comparison methods such as KS test [36].

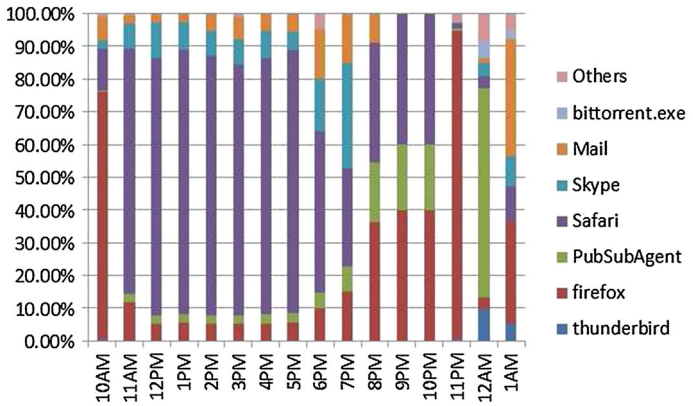

Fig. 2 Breakdown of applications for time windows (10–1 a.m.), compiled from the ground-truth data

In general, this traditional approach is optimized with a single variable but hard to extend it to the analysis of multiple variables [37,38].

These challenges motivate us to explore possible alternative approaches that may have greater potential in scalability and feasibility. As will be introduced, clustering, with its strong benefit of aggregation, enables to yield a predetermined number of clusters regardless of the number of variables considered in monitoring. Therefore, network monitoring using clustering would provide a consistent way to keep track of patterns for network states.

4 Clustered patterns for traffic analysis

In this section, we present our approach to network traffic analysis using clustered patterns. The key idea of the proposed approach is to capture the network state from the monitored variables and represent it with a clustered pattern. Again, "network state" is defined as the recap of the network traffic with respect to the monitored variables. What makes our new approach unique and more efficient than traditional approaches, such as sketches and distribution-based analysis, is that the clustering method has the ability to combine multivariate attributes in a straightforward manner without an excessive extra cost, which is a critical challenge when relying on the probabilistic distributions as discussed in the previous section. In addition, the calculated patterns can be utilized to evaluate "similarity" of network states over time for the purpose of network monitoring. Thus comparing similarity of given network states can be reduced to a straightforward pattern comparison problem.

Figure 3 demonstrates the basic idea of clustered patterns. Each window shows the snapshot of the network state with a set of clusters. The number of clusters for each window in the example is preset to three. In the figure, each cluster has the centroid position and size of the cluster. A partition-based clustering such as k-means algorithm provides a set of properties for a cluster, such as the centroid coordinate, the number of data points within the cluster, the sum of squared errors, and so forth, which could be

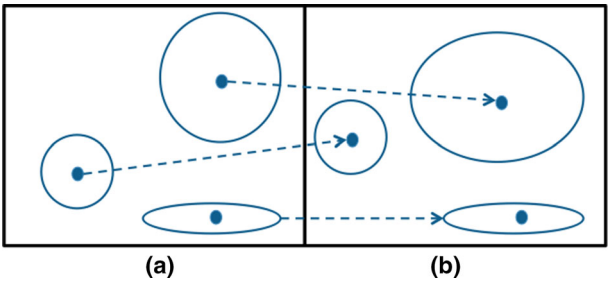

Fig. 3 An example of clustered patterns in two different time windows. Each window shows the snapshot of the network state with a set of clusters and the number of clusters for each window in the example is three. A pattern is a vector of clusters with cluster properties, such as the centroid coordinate, the number of data points within the cluster, the sum of squared errors, etc. The patterns can be compared with the defined cluster properties. For example, the centroid position moves between two patterns (as the arrow lines in the figure) can be considered as a measure to quantitatively compare two clustered patterns (which is discussed in Sect. 4.2)

used to define the property of the cluster (i.e., the centroid position and size of cluster). Thus, a pattern is simply a vector of clusters with the defined cluster properties. The two patterns can then be compared based on the cluster property information. In this study, we chose a conventional partitioning-based clustering (k-means) with its manageable complexity and scalability [34].

We next discuss our approach of the clustered patterns with the UNIBS data discussed in the previous section. The degree of changes measure will then be introduced as a quantitative analysis tool.

4.1 Clustered patterns of the UNIBS data

In our preliminary work, we used the cumulative distribution function to analyze the UNIBS data trace, with two traffic variables of flow duration and the average number of packets in flows (as shown in Fig. 1). We apply the clustering technique against the same data set to construct multivariate patterns. Figure 4 demonstrates the clustering patterns over 16 time windows. We set the number of clusters to 4 ($k = 4$) for a single time window, which is derived from the total sum of squares using an "elbow" method. As shown in the figure, the clustered results show the captured network states, as well as the correlated patterns (Δ in the figure will be explained shortly). For example, the pattern for 10 a.m. time window is quite different from the one for 11 a.m. time window. In contrast, the clustered patterns from 11 a.m. to 5 p.m. are visually similar. The three patterns for 8–10 p.m. time windows are also resembling, whereas the last three time windows (11 p.m.–1 a.m.) have somewhat distinctive patterns.

From Figs. 2 and 4, we can see a strong correlation. For example, the breakdown graph (Fig. 2) shows a high degree of similarity from 11 a.m. to 5 p.m. and from 8 p.m. to 10 p.m., respectively, which agrees with similarity of the clustered patterns in Fig. 4. On the other hand, there is a high degree of difference in the breakdown graph between 10 and 11 a.m.. Similarly, we can see huge differences from the windows of 11 p.m.–1 a.m. in Fig. 2.

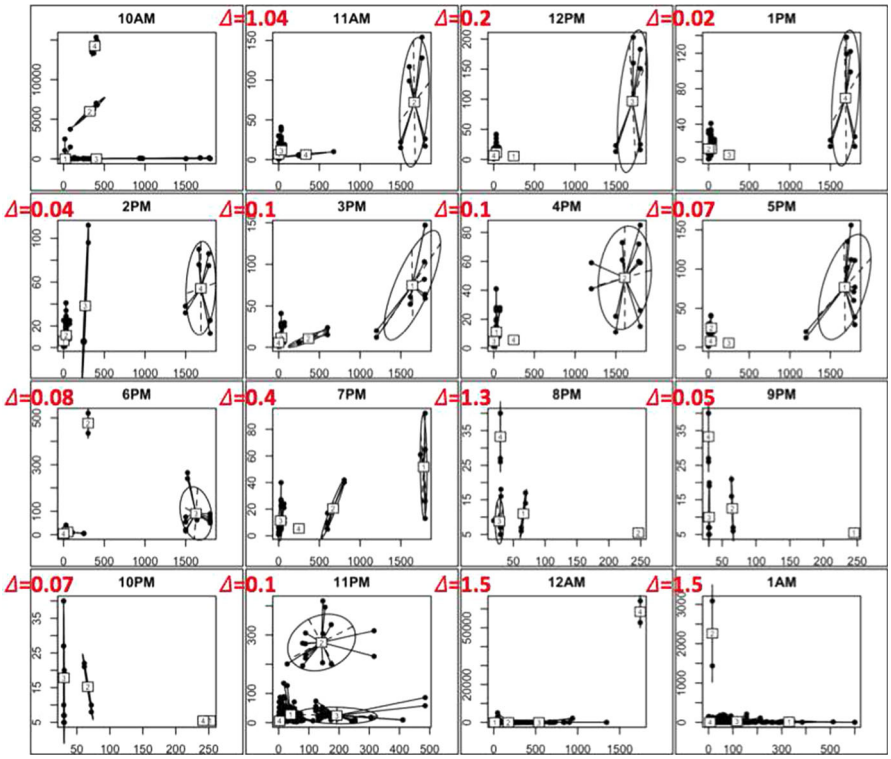

Fig. 4 Clustered patterns for 16 consecutive hourly time windows on UNIBS data trace for flow duration on x-axis and the average number of packets per flow on y-axis, between 10 a.m. on Sep. 30, 2009 and 2 a.m. on Oct. 1, 2009. The number of clusters is set to 4, based on the sum of squares in groups. Note that cluster IDs were randomly selected by the clustering tool. Δ stands for degree of changes calculated based on the centroid positions move between two clustered patterns, to quantitatively measure the similarity of the patterns. The smaller, the more similar the patterns are (see details in Sect. 4.2)

The clustering-based monitoring is helpful to intuitively identify similarity of patterns among time windows. However, some patterns may not be clear enough to determine the similarity of them only through the visual monitoring. Any quantitative measures would be helpful to perceive the degree of similarity with greater confidence if provided. For this purpose, we define a metric based on the centroid position move (in the Euclidean space) between two compared windows in the next section.

4.2 Quantitative measure: degree of changes (Δ)

We introduce a new measure of "degree of change" (Δ) that computes the difference of two patterns based on the movement of the centroid positions of the clusters. The basic intuition behind this is that the centroid coordinates of two patterns would be close without a heavy change if the network states in comparison are tightly related. That is why we fixed the number of clusters in this use case.

Suppose two time windows W_i and W_j, and the associated cluster sets $C_i = \{c_i^0, c_i^1, ..., c_i^k\}$ and $C_j = \{c_j^0, c_j^1, ..., c_j^k\}$, respectively, where k is the number of clus-

ters. Each cluster c_x^y has its centroid p_x^y. Without knowing which cluster in W_i is mapped with one in W_j, we find a set of pairs showing the minimal move. Suppose a distance function $D : C_i \times C_j \rightarrow \mathbb{R}$. Then the problem is reduced to the assignment problem that finds a bijection $f : C_i \rightarrow C_j$ with the minimal distance function:

$$\Delta_{i,j} = \sum_{l \in C_i} D(l, f(l))$$

Hungarian algorithm is a well-known method for this type of problem with $O(k^3)$ of the computational complexity [39]. As k (the number of clusters) is generally not large ($k = 4$ in Fig. 4), the complexity overhead would not be unacceptable.

The Δ values in Fig. 4 represent the degree of changes for two adjacent windows. From the Δs, we see relatively small Δs from 11 a.m. to 6 p.m. and from 8 p.m. to 10 p.m. ($\Delta \leq 0.2$). On the other hand, large Δs were observed for the adjacent windows of (10 a.m., 11 a.m.), (7 p.m., 8 p.m.), (11 p.m., 12 a.m.) and (12 a.m., 1 a.m.) ($\Delta > 1.0$). While strong correlations are observed overall, the Δ(10 p.m., 11 p.m.) is the only exception with a small value ($\Delta = 0.1$) although the two windows are slightly different visually. This indicates that the visual patterns and quantitative measures are complementary and need to be considered together to properly read the state of the network.

4.3 Evaluation with another day trace

To further investigate the effectiveness of identifying patterns and the correlation between the clustered patterns and the metric of the degree of changes, we experiment on a data set collected in a different day in the UNIBS data set. In this experiment, we did not consider application breakdowns, but compared the plotted patterns with the associated quantitative information. The new data set is a part of the UNIBS data trace, collected in a different day of October 2, 2009. Since the data set includes a relatively large number of flows, we performed clustering with every 5 min trace from 8:00 a.m. to 9:20 a.m.. The average number of flows is 973 flows/hour with a low variance (min = 599, max = 1350). The configuration for clustering is the same as the previous experiment with four clusters, based on the sum of squares within groups.

Figure 5 demonstrates the clustering results on the 5-min time window data set. For ease of presentation, we identify individual time windows with the unique ID in addition to their beginning time. As in Fig. 4, the figure shows clustered pattern changes over time with discernible similarity/dissimilarity between windows. We can see that several pairs of time windows such as (3,4), (6,7), (8,9) and (10,11) look pretty similar. On the other hand, some others such as (1,2), (2,3), (12,13) and (15,16) show highly distinctive patterns.

Figure 6 shows the corresponding degree of changes (Δs) computed. In the figure, x-axis shows a pair of adjacent windows compared to one another, and y-axis shows the calculated Δ for the associated pair of windows (normalized). For the clustered results with two attributes of flow duration and the number of packets in flow ("2 attributes"), pairs of windows of (1,2), (4,5), (5,6), (9,10), (12,13) and (15,16) show relatively

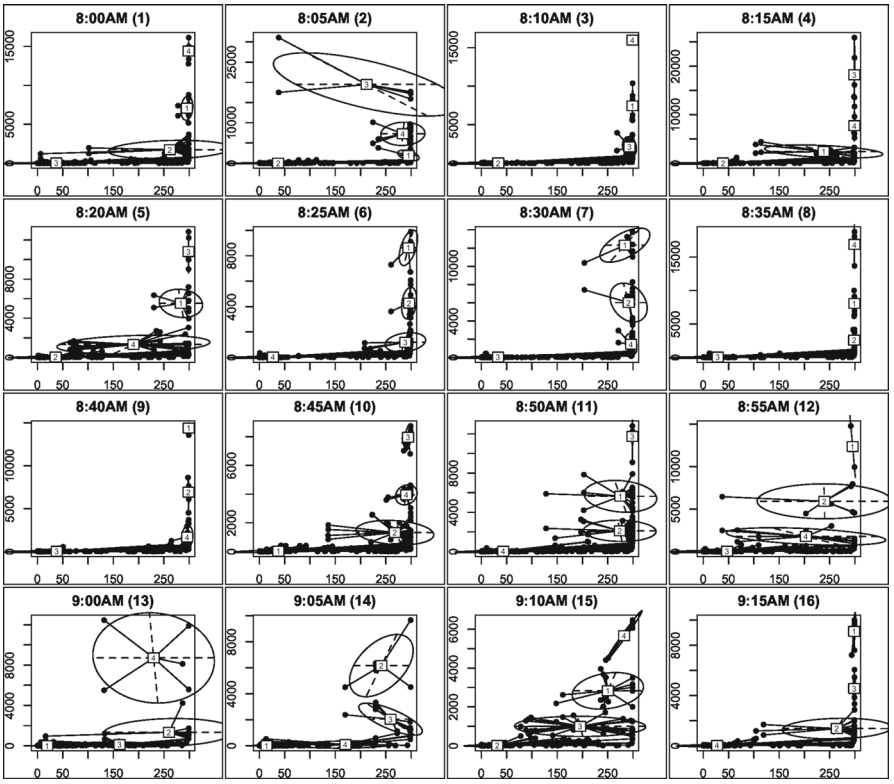

Fig. 5 Clustered patterns for 16 consecutive 5-min time windows on UNIBS data trace, between 8:00 and 9:20 a.m. on October 2, 2009

high Δs, whereas windows of (3,4), (6,7), (8,9), (10,11) and (13,14) reported minor changes, overall agreeing on the visual patterns. Although not adjacent, two windows of 1 and 4 look alike in Fig. 5, and its calculated degree of changes is fairly low with $\Delta_{(1,4)} = 0.34$ (not shown in Fig. 6). Also, the windows of (5,11) has a moderate difference of $\Delta_{(5,11)} = 0.42$, supporting the soundness of our approach.

The figure also includes a plot ("3 attributes") with an additional variable of the number bytes in flow for clustering, showing concurring trends with the other ("2 attributes"). We observed very negligible variations ($\sigma \approx 0$) in this experiment with a relatively large number of flows in each window, indicating that the impact of randomness introduced by the k-means algorithm would not be significant with a sufficiently large number of samples.

To sum it up, we observed a strong correlation between the clustered patterns and traffic composition that confirms the feasibility of the proposed method for change detection of network states. We also introduced the measure of "degree of changes" (Δ) to quantitatively estimate the similarity of two patterns. In the next section, we apply the clustered pattern technique to identify anomalous network states as another use case model.

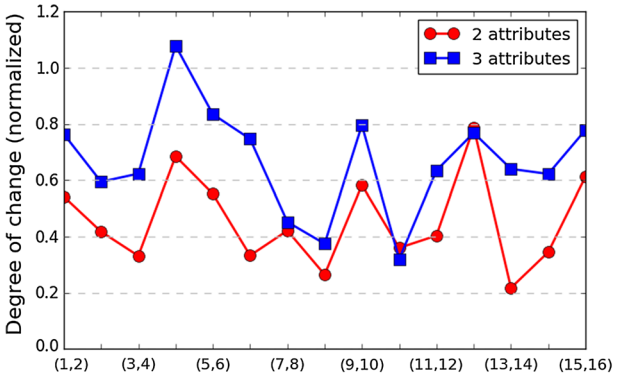

Fig. 6 Degree of changes (Δ) based on the centroid positions move with the trace from 8 a.m. to 9:20 a.m. on Oct. 2, 2009. All the observed variances are negligible and almost zeros ($\sigma \approx 0$)

5 Anomalous state detection using clustered patterns

Clustering has been employed for network anomaly detection in previous studies [27, 29,31,32]. The main focus of the past studies is on determining whether *individual* connections are anomalous or not. For example, a connection in question would be anomalous if it is closer to an anomaly cluster than a normal cluster. We are not interested in the binary classification of individual connections in this study. Rather, our primary interest is to help the network operator to determine whether the current network state is anomalous (due to many reasons such as intrusions and failures) or not in a *collective* way.

For this use case, we employed the KDDCup 1999 data ("kddcup.data_10_percent_corrected") [14] that has been widely used in the anomaly detection study. A data instance in the data set is a record of a single TCP connection. The individual record also contains a label for: a normal connection ("NORMAL"), a denial of service attack ("DOS"), an unauthorized access from a remote host ("R2L"), an unauthorized access to root functions ("'U2R"), and one employed for probing for vulnerabilities ("Probe"). Since no timing information is provided in the data set, we formed 16 partition windows ("AA"–"AP") serially from the beginning of the data file, each of which contains 1000 connections.

Two connection-related attributes were considered in this case study: "src_bytes" is the number of bytes from the source to destination host and "dst_bytes" is the number of bytes from the destination to source. Due to a high degree of skewness, we normalized the data using a *log* function. For example, the summary of src_byte *before* normalizing is: 0 (min), 45 (1st quartile), 520 (median), 3026 (mean), 1032 (3rd quartile), 693,400,000 (max); the summary *after* the normalization is: 0.00 (min), 1.65 (1st quartile), 2.72 (median), 2.16 (mean), 3.01 (3rd quartile), and 8.84 (max). The normalization makes sense since a 10-byte difference is still critical to the connections in the 1st quartile group ($\leq$ 46 bytes), while it is insignificant to the right-skewed connections such as one with 693,400,000 bytes (max).

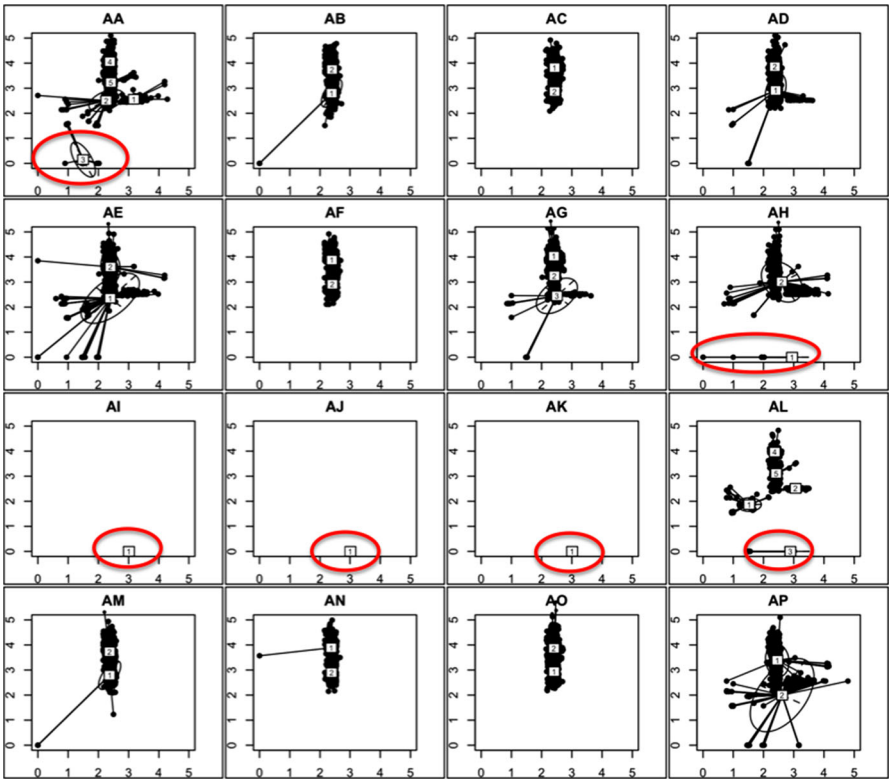

Fig. 7 Clustering results: Each window contains 1000 connections without overlapping. The k-means algorithm is used and the number of clusters is estimated using the partitioning around medoids technique. The clusters in a thick circle are not found from a cluster with normal connections only

5.1 Representation using clustered patterns

As in the previous section, we employ the k-means clustering to obtain the patterns. One difference is that the number of clusters is dynamically determined for each time window, while it was fixed in the first use case that focuses on the degree of changes. The number of clusters for a time window is estimated by the optimum average silhouette width. The procedure for clustering is as follows: For each window W_i, the number of clusters (k_i) is estimated with the data instances in that window (D_i and $|D_i| = 1000$). Then the clustering algorithm is executed with an input of k_i to obtain a clustered pattern for W_i.

Figure 7 shows the clustering result and Table 1 provides a summary of the windows with labels. From Table 1, we can see the windows from AH to AL have a lot of DOS connections and three windows of AA, AE and AP include some malicious connections. From the figure, we can see noticeable similarity from many clean windows (AB–AD, AF, AM–AO). The windows containing malicious events (AA, AE, AH–AL, and AP) yield quite different shapes from the ones observed in the clean windows. We can see that DOS makes a specific pattern as shown in the windows of

Table 1 Traffic composition and likelihoods of attacks, with two threshold values of 0.5 (italicized) and 0.7 (bold-fonted)

Window	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP
Normal	997	1000	1000	1000	998	1000	1000	792	0	0	0	511	1000	1000	1000	979
DOS	0	0	0	0	0	0	0	208	1000	1000	1000	489	0	0	0	20
U2R	2	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
R2L	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Probe	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L_{pv} (2D)																
DOS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	*0.51*	**1.00**	**1.00**	**1.00**	0.22	0.00	0.00	0.00	0.00
U2R	**0.71**	*0.60*	*0.62*	**0.70**	0.47	*0.68*	0.48	0.43	0.00	0.00	0.00	**0.71**	*0.60*	**0.72**	**0.70**	0.31
R2L	**0.85**	*0.51*	*0.56*	*0.60*	**0.71**	*0.59*	**0.72**	0.40	0.17	0.17	0.17	*0.59*	*0.51*	*0.62*	*0.60*	0.29
Probe	0.44	0.48	0.42	0.40	0.23	0.41	*0.66*	0.28	0.00	0.00	0.00	0.33	0.49	0.41	0.38	*0.50*
L_{dr} (2D)																
DOS	0.32	0.00	0.00	0.00	0.00	0.00	0.00	**0.81**	**1.00**	**1.00**	**1.00**	**0.70**	0.00	0.00	0.00	0.00
U2R	0.38	0.48	0.32	*0.67*	*0.53*	0.42	0.09	0.51	0.00	0.00	0.00	0.44	0.49	0.49	*0.51*	0.13
R2L	**0.72**	0.27	0.20	*0.58*	**0.88**	0.27	0.39	**0.74**	0.00	0.00	0.00	*0.60*	0.27	0.25	0.46	*0.54*
Probe	0.13	0.00	0.00	0.00	0.28	0.00	0.48	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	*0.57*
L_{pv} (3D)																
DOS	0.11	0.00	0.00	0.00	0.00	0.00	0.00	*0.51*	**1.00**	**1.00**	**1.00**	0.22	0.00	0.00	0.00	0.00
U2R	0.40	0.38	0.37	0.37	0.40	0.37	0.38	0.36	0.02	0.02	0.02	0.40	0.38	0.37	0.38	0.39
R2L	0.42	0.18	0.13	0.12	0.41	0.13	0.27	0.37	0.29	0.29	0.29	0.45	0.19	0.14	0.26	0.15
Probe	0.17	0.10	0.09	0.09	0.16	0.09	0.12	0.05	0.00	0.00	0.00	*0.66*	0.11	0.09	0.12	0.11
L_{dr} (3D)																
DOS	**0.70**	0.00	0.00	0.00	0.00	0.00	0.00	**0.81**	**1.00**	**1.00**	**1.00**	**0.70**	0.00	0.00	0.00	0.00
U2R	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
R2L	0.23	0.00	0.00	0.00	*0.59*	0.00	0.22	0.17	0.00	0.00	0.00	0.28	0.00	0.00	0.00	0.45
Probe	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.19	0.00	0.00	0.00	0.00

AH–AL. Window AA has a cluster around the position of (1.5,0) (red-circled); it does not include any of DOS connections but some of U2R attack connections. AE and AP both contain several malicious connections and look similar each other, although the windows have different types of attacks. AG has no attack connection and looks partly similar to AE and AF, thus making it somewhat unclear to figure out.

5.2 Measuring likelihood of attacks

We next present a metric of "likelihood" as a quantitative tool for operators to test the potential of a cluster as an attack class. One simple way to compare two clusters would be to use the Euclidean distance $d(c_1, c_2)$, where c_i is the centroid position of cluster C_i in a multi-dimensional space; hence, $d \in \mathbb{R}$ and $d \geq 0$. Then we can assume a greater likelihood for two clusters with a smaller distance. When $d = 0$, it implies that the given clusters have the same centroid position, and it will result in the greatest likelihood by definition. While simple, the limitation of this method is that none of d other than zero may not be adequately interpreted to estimate the likelihood since it can be any positive real number. For instance, it could be $d = 1$ or $d = 100$; what do they mean in terms of the likelihood? To establish a metric of the likelihood, we want to scale it between zero (the least likely) and one (the most likely) for the relative comparison.

We assume that a cluster is represented by two parameters of $\langle c, v \rangle$, where c is the centroid position and v is a degree of variation in the cluster. A larger v thus indicates that the elements in the cluster are further dispersed in the space. Obtaining the centroid position is straightforward with the k-means clustering as any cluster has a centered point that minimizes the total squared sum of the elements ("within-SS" or simply wss). We utilize this parameter (wss) to characterize a degree of variation, which is defined as $v = \frac{wss}{n}$, where n is the number of elements in the cluster. With this representation of a cluster, we establish two measures for the likelihood defined, one is based on a *percentile* in the Gaussian distribution (l_{pv}), and the other is based on a *diameter* of a cluster (l_{dr}).

To define the first measure for the likelihood (l_{pv}), we employ a concept of the statistical percentile in a Gaussian distribution using p value. The first measure (l_{pv}) is derived from the statistical percentile, by assuming that the centroid position of the compared cluster as the mean and the $\sqrt{v}$ as the standard deviation. Suppose a cluster $C' = \langle c', v' \rangle$ and an attack cluster $C = \langle c, v \rangle$, and we would like to determine the likelihood of an attack of C for C'. The normal score (z score) is calculated by $\frac{c-c'}{\sqrt{v}}$, where $(c - c')$ is the distance in the Euclidean space. With the calculated normal score, l_{pv} is defined as the two-fold percentile using the associated p value. Note that we assumed $v = \min(v, 0.01)$ to prevent the divide-by-zero error.

Then we test a p value with the centroid position of the comparing cluster The basic idea for this measure is that we assume a cluster The attack cluster is represented with (c, n, e), where c is the centroid coordinate, n is the number of data points, and e is the minimum of either the total sum of squares in the cluster or 0.01 (an arbitrary small value to avoid divide-by-zero for calculating normal score below). Then we assume e/n as its variance. Thus, the smaller cluster size, the smaller variance is, and

vice versa. To test likelihood of the attack for a cluster with centroid position c', we compute p value by calculating the normal score by $z = \frac{c-c'}{\sqrt{v}}$, where $c - c'$ is simply a Euclidean distance. Finally, we define l_{pv} is the two-sided p value of z.

We next introduce the second measure (l_{dr}) based on the centroid positions and the radius of the cluster. While l_{pv} tests how close the center of the cluster is to the attack cluster's center, we consider a cluster as a circle with a radius that is assumed equal to the deviation of the cluster to establish l_{dr}. Thus, cluster $C' = \langle c', v' \rangle$ is represented as a circle with the center of c' and the radius of $r' = \sqrt{v'}$. To test the likelihood of an attack represented with $C = \langle c, v \rangle$, we compare the two center points c and c' with the radius r'. In the comparison step, we compute l_{dr} as follows:

$$l_{dr} = \max\left(1 - \frac{c - c'}{\sqrt{v'}}, 0\right)$$

Figure 8 demonstrates two examples for l_{dr}. In the figure, the large circle is a cluster in question, and the small circle is an attack cluster. As shown in the left figure, we compute the likelihood by $l_{dr} = 1 - \frac{c-c'}{\sqrt{v'}}$, since the center of the attack cluster is located within the large circle. In contrast, it is a zero ($l_{dr} = 0$) in the right figure as the two circles have no overlap, and the center of the attack cluster is located outside the large circle.

To evaluate the defined measures, we precalculated the patterns of the attack clusters for each attack class. Table 2 shows the number of clusters for each attack type obtained in the precalculation process, with two variables of <*sbytes*,*dbytes*> and three variables of <*sbytes*,*dbytes*,*dur*>. The precalculated attack patterns will then be compared with the clustered patterns obtained in each time window to measure the likelihood of attack. Using the patterns for individual attacks, we compute likelihood using l_{pv} and l_{dr} for each window. As there can be multiple clusters for a single attack type and a window can have more than a single cluster, we define a collective measure L that takes the max from the set of the likelihood values. In detail, suppose a window

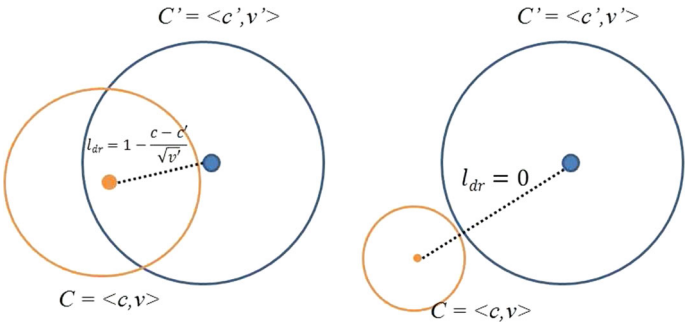

Fig. 8 Estimating likelihood based on the radius of the circle (l_{dr}): The large circle is a cluster in question and the small circle is an attack cluster. In the left figure, the center of the attack cluster is located within the large circle and the likelihood is defined as the relative distance compared to the radius of the large circle ($l_{dr} = 1 - \frac{c-c'}{\sqrt{v'}}$). The right figure show an example that the two circles have no overlap since the center of the attack cluster is located outside the large circle ($l_{dr} = 0$)

Table 2 Number of clusters observed in the precalculated attack patterns, with two variables of <*sbytes*,*dbytes*> and three variables of <*sbytes*,*dbytes*,*dur*>

Attack	2D <*sbytes*,*dbytes*>	3D <*sbytes*,*dbytes*,*dur*>
DOS	6	6
U2R	4	2
R2L	10	9
Probe	10	10

Table 3 Evaluation of measures (number of cases and percentage)

Measure	$L \geq 0.5$		$L \geq 0.7$	
	FP	FN	FP	FN
L_{pv} (2D)	8 (50%)	3 (19%)	6 (38%)	4 (25%)
L_{dr} (2D)	1 (6%)	2 (13%)	0 (0%)	3 (19%)
L_{pv} (3D)	0 (0%)	4 (25%)	0 (0%)	5 (31%)
L_{dr} (3D)	0 (0%)	3 (19%)	0 (0%)	3 (19%)

with a set of clusters $\{C'_1, C'_2, ..., C'_m\}$, and an attack class (e.g., DOS) with another set of clusters $\{C_1, C_2, ..., C_n\}$. Then the collective likelihood L_α for attack type α is defined as:

$$L_\alpha = \max_{i,j} l(C'_i, C_j), \quad \text{where } 0 < i \leq m \text{ and } 0 < j \leq n$$

Table 1 also shows L_{pv} and L_{dr} when using two variables (2D) and three variables (3D), respectively. To evaluate the measures, we considered two threshold values of 0.5 (italicized in the table) and 0.7 (bold-fonted). As seen from the table, the windows with attack connections have high values for the likelihood. In contrast, the normal windows without any attack connections show relatively small values for the likelihood measure. The following illustrates the steps to evaluate the quality of the likelihood measure (L), with respect to anomaly detection:

1. If the window has no attack and none of L is greater than or equal to the threshold, it is a *true negative* (TN);
2. If the window has any types of attacks and any of the associated L is greater than or equal to the threshold, it is a *true positive* (TP);
3. If the window has any types of attacks and any of the associated L is less than the threshold, it is a *false negative* (FN);
4. Otherwise, it is a *false positive* (FP).

We examined with two threshold values, $L \geq 0.5$ and $L \geq 0.7$, using two attributes (src_bytes and dst_bytes) and using three attributes (plus "connection duration"). We refer to the former as "2D" and the latter as "3D". Table 3 shows the calculated FPs and FNs. From the result, we can see using three features reduces the number of the false decisions but with a slight increase of false negatives. Also we can see that L_{dr} works conservatively compared to L_{pv}.

5.3 Evaluation with another trace

We also conducted the same experiment with another data set ("dataset B") from KDDCup 1999 for estimating the likelihood of attacks with the three variables (3D). All the evaluation methods are equal to the ones used in the past section, and the only difference is the data set itself.

Like the result with the first dataset, we can see that the estimated measures largely agree with the network states, showing greater likelihood for anomalous windows. As shown in Table 4, L_{pv} made 3 FNs and 2 FPs with $L \geq 0.5$ and 3 FNs and 1 FP with $L \geq 0.7$. With L_{dr}, we obtained 3 FPs and 2 FPs respectively for the two thresholds with no FN. From the results, the proposed measures are overall working well, although there is also a room for future optimizations.

In sum, the clustering result with dataset B yielded somewhat distinctive patterns between normal and anomalous states, and the normal patterns are closely similar to the ones observed in the first data set (i.e., dataset A). From the experimental results with two independent datasets, we can see that the patterns can be helpful for network operators to intuitively estimate the network state whether it is anomalous or not.

6 Evaluation with ESnet trace

ESnet [15] provides the high-bandwidth, reliable network connections that link scientists at national laboratories, universities and other research institutions. ESnet is funded by the DOE Office of Science, and managed and operated by the Scientific Networking Division at Lawrence Berkeley National Laboratory. The `tstat` data used in this paper comes from the test ESnet Data Transfer Nodes (DTNs)[2]. Data Transfer Nodes are an important part of ESnet's Science DMZ network design pattern [40]. ESnet is collecting `tstat` logs in order to analyze how various network tuning settings impact TCP behavior and network throughput.

To collect TCP instrumentation data for each flow, the `tstat` tool is used [41]. `tstat` can be used to analyze either real-time or captured packet traces. It rebuilds each TCP connection by looking at the TCP header in the forward and reverse direction. `tstat` reports the source and destination host/port information, and the number of bytes/packets sent and received. It also reports a number of useful TCP statistics, including congestion window size and number of packets retransmitted, which can be used to analyze the health and performance of the link.

In this study, we analyze the `tstat` data to monitor the network state change over time. We selected a subset of the measurement collection between 12:00 p.m. and 2:40 p.m. on May 9, 2016, to form sixteen 10-min windows. The windows have 367 connections for each on average. We chose two variables of <number of packets, max throughput> to keep track of changes. From the distribution of the variables,

[2] The details of DTNs are described at https://fasterdata.es.net/performance-testing/DTNs/.

Table 4 Traffic composition & likelihoods of attacks (dataset B)

Window	BA	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM	BN	BO	BP
Normal	906	1000	1000	1000	1000	999	1000	1000	1000	1000	1000	1000	1000	722	275	979
DOS	0	0	0	0	0	1	0	0	0	0	0	0	0	276	724	0
U2R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R2L	0	0	0	0	0	0	0	0	0	0	0	0	0	2	1	2
Probe	94	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
L_{pv} (3D)																
DOS	0.00	0.01	0.00	0.00	0.00	0.39	0.00	0.00	0.05	0.00	**0.87**	0.00	0.00	0.47	**0.93**	**0.92**
U2R	*0.50*	0.39	0.38	0.37	0.37	0.38	0.39	0.38	0.38	0.40	0.38	0.38	0.37	0.40	0.39	0.46
R2L	0.42	0.23	0.26	0.16	0.11	0.58	0.15	0.26	0.25	0.08	0.32	0.19	0.15	0.26	0.35	0.45
Probe	0.09	*0.59*	0.12	0.10	0.08	0.06	0.28	0.12	0.11	0.10	*0.68*	0.10	0.09	0.07	*0.58*	**0.92**
L_{dr} (3D)																
DOS	**0.71**	0.36	0.00	0.00	0.00	**0.92**	*0.55*	0.00	**0.77**	0.00	**0.98**	0.00	0.00	**0.95**	**0.98**	**0.91**
U2R	**0.77**	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	*0.59*	*0.50*	0.00
R2L	*0.60*	0.00	0.00	0.00	0.00	*0.78*	0.00	0.00	0.07	0.39	*0.57*	0.00	0.00	0.46	0.32	0.46
Probe	**0.73**	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	*0.68*	*0.64*	**0.91**

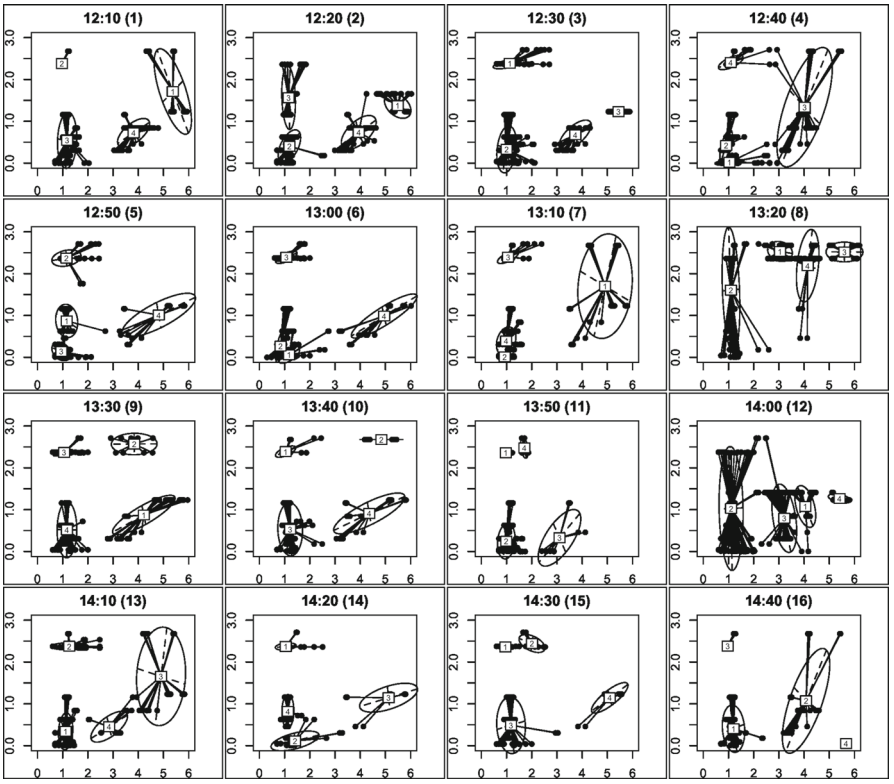

Fig. 9 Clustering results over 16 consecutive 10-min time windows on ESnet data trace, with two variables of the number of packets on x-axis and max throughput on y-axis. We randomly selected data between 12:00 p.m. and 2:40 p.m. on May 2, 2016. The number of clusters is set to 4, based on the sum of squares in groups. Note that cluster IDs were randomly selected by the clustering tool

we observed a very high degree of skewness. Thus, we normalized data using a *log* function. We set the number of clusters to 4 ($K = 4$) using an elbow method, as in the previous sections.

Figure 9 demonstrates the clustered patterns for the 16 windows. We can see some patterns look closely similar each other, e.g., (5,6), (9,10), and (14,15), while some adjacent patterns are visually distinctive, e.g., (3,4), (7,8), and (11,12).

Figure 10 shows the associated degree of changes between the adjacent windows. Since the ESnet data set does not contain any annotation information, we conducted KS test for the two traffic variables. The first two plots in the figure show the KS test statistic for (a) the number of packets and (b) max throughput. Lower statistic values indicate greater similarity in terms of probabilistic distribution. The next plot (Fig. 10c) shows the degree of changes (Δ). The horizontal lines in each figure represent the averages over the 16 time windows. From the plots in Fig. 10, the probabilistic distribution similarity largely agree on our degree of changes measure for this data set.

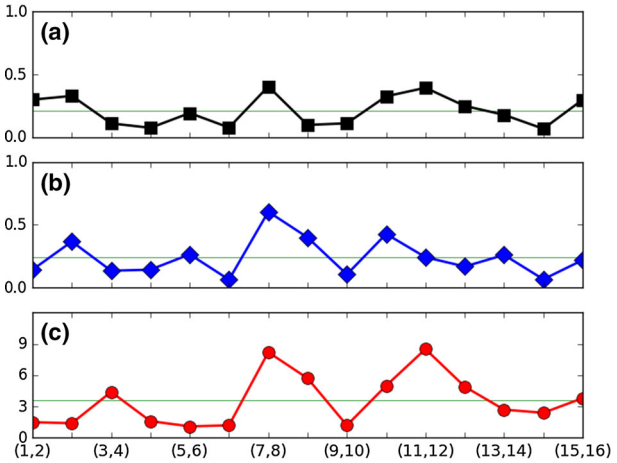

Fig. 10 Comparison between KS test similarity for individual variables and degree of changes

7 Conclusion

This paper proposes a new approach to the high-level online network monitoring using clustered patterns. The main goal of this study is to enable network traffic analysis with multivariate attributes. We demonstrated the feasibility of the proposed technique for two popular network monitoring applications. For the detection of state changes, we demonstrated our analysis on the clustering results with the associated ground-truth information, with the established measure of "degree of changes". We also presented the network anomaly detection as the second use case. We defined a new metric to estimate the likelihood of a cluster to be an attack class and set up two measures with the clustering information, one based on percentile in the Gaussian distribution and the other based on the radius of a cluster. The presented method was also applied to a recent ESnet traffic trace to keep track of state changes. The evaluation results support the effectiveness of the new approach with the pattern-based representation of network states and the quantitative measures.

This research is an on-going project, and there are several interesting challenges to be addressed in future. Scalable analysis is one of the key requirements in network operation. For example, we observed that it takes approximately 10 seconds to construct clusters with 16,000 data points in a commodity PC with the simple k-means that is known as a scalable method for clustering. Sampling could be an option for scalability, but we observed that a simple sampling may result in somewhat deviated ones compared to the output with the entire data points. In addition, clustering can be affected by a small set of noises. Thus, developing a noise-resistant method for representing network states would be an interesting topic to explore for greater robustness.

Acknowledgements This work was supported in part by the Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, and by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea

government(MSIP) (No. 2016-0-00078, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

References

1. Cisco white paper: Cisco vni forecast and methodology, 2015–2020. http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf
2. Cho K, Fukuda K, Esaki H, Kato A (2008) Observing slow crustal movement in residential user traffic. In: Proceedings of the 2008 ACM conference on emerging network experiment and technology, CoNEXT 2008, Madrid, Spain, December 9–12, p 12
3. Tong D, Prasanna V (2016) High throughput sketch based online heavy hitter detection on FPGA. ACM SIGARCH Comput Archit News 43(4):70–75
4. Yu M, Jose L, Miao R (2013) Software defined traffic measurement with opensketch. In: Proceedings of the 10th USENIX conference on networked systems design and implementation, NSDI'13, pp 29–42
5. Liu Z, Manousis A, Vorsanger G, Sekar V, Braverman V (2016) One sketch to rule them all: Rethinking network flow monitoring with univmon. In: Proceedings of the 2016 conference on ACM SIGCOMM 2016 conference, Florianopolis, Brazil, August 22–26, pp 101–114
6. Li B, Springer J, Bebis G, Gunes MH (2013) Review: a survey of network flow applications. J Netw Comput Appl 36(2):567–581
7. Krishnamurthy B, Sen S, Zhang Y, Chen Y (2003) Sketch-based change detection: methods, evaluation, and applications. In: Proceedings of the 3rd ACM SIGCOMM conference on internet measurement, IMC '03, pp 234–247
8. Choi J, Hu K, Sim A (2013) Relational dynamic bayesian networks with locally exchangeable measures. LBNL Technical Report, LBNL-6341E
9. Portnoy L, Eskin E, Stolfo S (2001) Intrusion detection with unlabeled data using clustering. In: Proceedings of ACM CSS workshop on data mining applied to security (DMSA), pp 5–8
10. Khan L, Awad M, Thuraisingham B (2007) A new intrusion detection system using support vector machines and hierarchical clustering. VLDB J 16(4):507–521
11. Leung K, Leckie C (2005) Unsupervised anomaly detection in network intrusion detection using clusters. In: Proceedings of the twenty-eighth Australasian conference on computer science, vol 38, ACSC '05, pp 333–342
12. Garcia-Teodoro P, Díaz-Verdejo JE, Maciá-Fernández G, Vázquez E (2009) Anomaly-based network intrusion detection: techniques, systems and challenges. Comput Secur 28(1–2):18–28
13. Dusi M, Este A, Gringoli F, Salgarelli L (2009) Using GMM and SVM-based techniques for the classification of SSH-encrypted traffic. In: Proceedings of IEEE international conference on communications, ICC, pp 1–6
14. KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
15. ESnet. https://www.es.net/
16. Iliofotou M, Pappu P, Faloutsos M, Mitzenmacher M, Singh S, Varghese G (2007) Network monitoring using traffic dispersion graphs (TDGS), IMC '07, pp 315–320
17. Karagiannis T, Papagiannaki K, Faloutsos M (2005) Blinc: multilevel traffic classification in the dark. SIGCOMM Comput Commun Rev 35(4):229–240
18. Lee S, Kim H, Barman D, Lee S, Kim C, Kwon T, Choi Y (2011) Netramark: a network traffic classification benchmark. SIGCOMM Comput Commun Rev 41(1):22–30
19. Zhang H, Sun M, Yao DD, North C (2015) Visualizing traffic causality for analyzing network anomalies. In: Proceedings of the 2015 ACM international workshop on international workshop on security and privacy analytics, IWSPA '15, New York, NY, USA, pp 37–42. ACM
20. Sivaraman V, Narayana S, Rottenstreich O, Muthukrishnan S, Rexford J (2017) Heavy-hitter detection entirely in the data plane. In: Proceedings of the symposium on SDN research, SOSR '17, New York, NY, USA, pp 164–176. ACM
21. Das S, Antony S, Agrawal D, Abbadi AE (2009) Cots: a scalable framework for parallelizing frequency counting over data streams. In: IEEE international conference on data engineering (ICDE), pp 1323–1326
22. Guha S, Koudas N, Shim K (2001) Data-streams and histograms. In: ACM symposium on theory of computing, pp 471–475

23. Datar M, Gionis A, Indyk P, Motwani R (2002) Maintaining stream statistics over sliding windows. In: ACM-SIAM symposium on discrete algorithms, pp 635–644
24. Motwani R, Raghavan P (1995) Randomized algorithms. Cambridge University Press, Cambridge
25. Manku GS, Motwani R (2002) Approximate frequency counts over data streams. In: VLDB, pp 346–357
26. Papadimitriou S, Sun J, Faloutsos C (2007) Dimensionality reduction and forecasting on streams. Data Streams Models Algorithms 31:261–288
27. Baek S, Kwon D, Kim J, Suh SC, Kim H, Kim I (2017) Unsupervised labeling for supervised anomaly detection in enterprise and cloud networks. In: 4th IEEE international conference on cyber security and cloud computing, CSCloud 2017, New York, NY, USA, June 26–28, pp 205–210
28. Fernandes G, Carvalho LF, Rodrigues JJPC, Proença ML (2016) Network anomaly detection using ip flows with principal component analysis and ant colony optimization. J Netw Comput Appl 64(C):1–11
29. Ahmed M, Mahmood AN, Hu J (2016) A survey of network anomaly detection techniques. J Netw Comput Appl 60:19–31
30. Qin T, Guan X, Li W, Wang P, Huang Q (2011) Monitoring abnormal network traffic based on blind source separation approach. J Netw Comput Appl 34(5):1732–1742
31. Li B, Liu P, Lin L (2016) A cluster-based intrusion detection framework for monitoring the traffic of cloud environments. In: 3rd IEEE international conference on cyber security and cloud computing, CSCloud 2016, Beijing, China, June 25–27, pp 42–45
32. Papalexakis EE, Beutel A, Steenkiste P (2012) Network anomaly detection using co-clustering. In: Proceedings of the 2012 international conference on advances in social networks analysis and mining (ASONAM 2012), ASONAM '12, pp 403–410
33. Jin L, Lee D, Sim A, Borgeson S, Wu K, Spurlock CA, Todd A (2017) Comparison of clustering techniques for residential energy behavior using smart meter data. In: AAAI workshops—artificial intelligence for smart grids and buildings, March 2017, San Francisco, CA
34. Bahmani B, Moseley B, Vattani A, Kumar R, Vassilvitskii S (2012) Scalable k-means++. Proc VLDB Endow 5(7):622–633
35. Rgringoli F, Salgarelli L, Dusa M, Cascarano N, Risso F, Claffy K (2009) Gt: picking up the truth from the ground for internet traffic. ACM SIGCOMM Comput Commun Rev 39(5):12–18
36. Kolmogorov-Smirnov Goodness-of-Fit Test. http://www.itl.nist.gov/div898/handbook/eda/section3/eda35g.htm
37. Justel A, Pena D, Zamar R (1997) A multivariate Kolmogorov–Smirnov test of goodness of fit. Stat Probab Lett 35:251–259
38. O'Neilla TJ, Sterna SE (2012) Finite population corrections for the Kolmogorov–Smirnov tests. J Nonparametr Stat 24(2):497–504
39. Mills-Tettey GA, Stentz A, Dias SMB (2007) The dynamic hungarian algorithm for the assignment problem with changing costs. Technical report, Carnegie Mellon University, East Lansing, Michigan
40. Dart E, Rotman L, Tierney B, Hester M, Zurawski J (2013) The science dmz: a network design pattern for data-intensive science. In: Proceedings of the international conference on high performance computing, networking, storage and analysis, SC '13, pp 85:1–85:10
41. Mellia M, Cigno RL, Neri F (2005) Measuring IP and TCP behavior on edge nodes with tstat. Comput Netw 47(1):1–21