

Automatic Profiling of Network Event Sequences: Algorithm and Applications

Xiaoqiao Meng, Guofei Jiang, Hui Zhang, Haifeng Chen and Kenji Yoshihira
 NEC Laboratories America, Princeton, NJ 08540
 Email: {xqmeng,gfj,huizhang,haifeng,kenji}@nec-labs.com

Abstract—The behavior of network entities, such as flows, sessions, hosts, and users, can often be described by communication event sequences in the time domain. For the purpose of many network measurement and monitoring tasks, it is desirable to have an accurate yet information-compact profiling of the behavior of massive event sequences. This paper proposes a new method to achieve this goal. On a given set of event sequences, the proposed method automatically learns a mixture model which fully captures the sequence behavior including both event pattern and duration between events. The learned mixture model is information-compact as it classifies sequences into a set of behavior templates, each of which is described by a Markov Chain. The model parameters are estimated in an iterative procedure which is developed from the Expectation Maximization algorithm. Two network management applications are proposed based on the method: a visualization tool for network administrators to conduct exploratory traffic analysis, and an efficient anomaly detection mechanism. In the evaluation, we validate the method accuracy as well as the usefulness of the two applications by using three networking datasets with different types: TCP packet traces, VoIP calls, and syslog traces in wireless networks.

I. INTRODUCTION

Understanding the behavior of networking entities such as TCP flows, sessions, hosts, or users are essential in many applications of network measurement and monitoring. Such behavior can often be described by event sequences, where by event sequence we refer to a series of events that *i)* they are affiliated with the same entity, *ii)* each event is identified by a symbol, *iii)* the possible symbol values are limited. While many event sequences can be found in a variety of networking scenarios, we describe three examples below as well as in Figure 1:

TCP SYN/FIN/RST sequences: The TCP protocol signals the start and end of a TCP connection by packets that are distinguished by flags in the packet header. The start packet has a SYN flag set and the end packet usually has a FIN or RST flag set. The arrivals of SYN, FIN and RST packets in a TCP connection form an event sequence.

SIP-based VoIP calls: SIP (Session Initiation Protocol) is the *de facto* signaling protocol for VoIP. Each VoIP call contains SIP control messages (e.g., INVITE, ACK and BYE) which set up or tear down the call. All such control messages in a call form an event sequence.

Wi-Fi user sessions: In Wi-Fi wireless networks, when a user establishes a session with the nearby access point, a set of Wi-Fi management frames are exchanged between the user

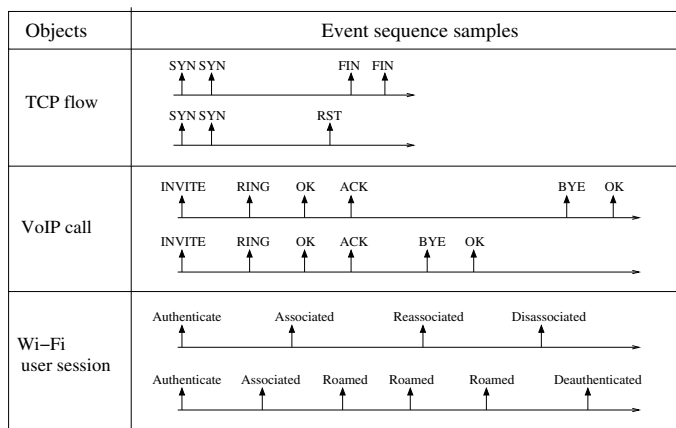


Fig. 1. Three event sequence examples

and the access point to establish, secure and terminate the session. All such management frames within a user session form an event sequence.

As a common problem in network measurement and monitoring tasks, operators often seek simple yet effective ways to help them understand the diverse behavior hidden in massive amount of event sequences - including both the massive normal behavior and potentially the behavior of a small proportion of anomalies. The goal in this paper is to develop a general methodology for a complete profiling of massive event sequences. The primary challenge in achieving this goal is the multi-dimensional behavior exhibited by event sequences, i.e., the behavior can not be fully described by a single variable or distribution. Instead, it exhibits at least two important behavior aspects: sequential patterns comprising of events, and duration between events. In practice, even one type of properties might be difficult to be described precisely. E.g., the aforementioned TCP SYN/FIN/RST sequence has only three event types, yet we can identify more than 3000 patterns with the longest pattern having 136 events (based on a 16-hour packet trace collected at an edge router [1]), even if we ignore the duration information and only look at sequential patterns of symbols. On the other hand, if we are only interested in duration, a precise profiling is still challenging. Taking the VoIP sequence in Figure 1 as an example, the duration between ACK and BYE is the actual call duration. This duration is usually heavy-tailed and ranges from seconds to hours [2]. When

both sequential patterns and duration are in consideration, the profiling difficulty is conceivably even higher.

In this paper we present a novel method to fully capture the behavior of massive amount of sequences by a compact model. Given a large sequence dataset, our method automatically learns a mixture model which consists of multiple Markov Chains. The model is general in the sense that it requires little knowledge about the targeting sequences except for the total number of discrete symbol values. The model has strong descriptive and inference power because it simultaneously captures two aspects of sequence behavior: the sequential symbol pattern and duration between symbols. Also, the model is flexible in terms of adapting its sensitivity with respect to either behavior aspect. In the model learning procedure, we apply Maximum Likelihood Estimation to estimate model parameters. The estimation procedure is developed from the Expectation-Maximization (EM) algorithm [3]. Although our method does not require much prior domain knowledge, we show that such knowledge can be utilized to improve accuracy and to reduce computational complexity. We evaluate the method by applying it to the three aforementioned sequence examples. Evaluation experiments are based on real networking traces, including a TCP packet trace, VoIP calls logged at a VoIP testbed that we set up on PlanetLab, and syslog traces in Wi-Fi wireless networks. The results validate the accuracy of the proposed method.

The proposed method possesses two merits potentially useful for many network measurement and management tasks. First, the method produces a convenient manifestation of massive sequence behavior by classifying sequences into so-called elephants and mice categories. This opens a door for a variety of exploratory data analysis tasks. As a showcase, we develop a tool which can visualize the resulting mixture model in an intuitive manner. The tool allows network operators to comprehend behavior patterns hidden in a chunk of traffic data. Secondly, because the method provides a way to describe the activity of network entities, it is applicable for distinguishing between significant sequence behavior and insignificant behavior. Such a capability is particularly useful for anomaly detection design. We provide a proof of concept by using the proposed method to design a simple anomaly detection scheme.

The contributions of this paper are summarized as following:

- A general method is proposed to automatically profile the behavior of massive event sequences.
- A visualization tool is developed from the method to help exploratory traffic analysis.
- A new anomaly detection scheme is proposed based on the method.

The rest of the paper is organized as follows. In Section II we provide details of our method. In Section III we demonstrate that our method can be applied in two network monitoring tasks: exploratory traffic analysis and anomaly detection. We discuss related work in Section V and conclude the paper by Section VI.

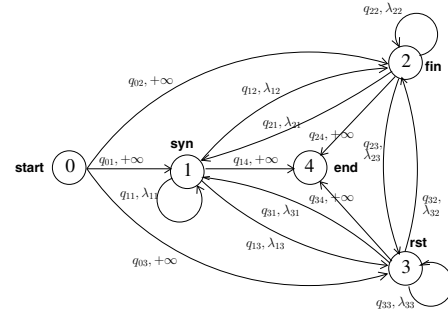


Fig. 2. A Markov chain for TCP SYN/FIN/RST sequences. Each transition is labeled by its probability and rate. The rate for any transition adjacent to the “start” and “end” states are infinite because such transitions are considered to take zero time.

II. PROCEDURE OF EVENT SEQUENCE MODELING

In this section we provide details of our method. We first introduce a mixture model, the basis of our method. We then describe the procedure for model parameter estimation. Finally, we discuss computational complexity and the model sensitivity with respect to parameters.

A. A mixture model approach

Our goal is to learn a model from input sequence data such that the model characterizes the sequence behavior. Our approach is based on a mixture model which contains multiple Markov Chains (MCs). We assume that each event sequence is independently generated from one MC. Each MC mimic one type of sequence behavior. Thus the mixture model is able to characterize the diverse behavior hidden in massive sequence data.

Now we describe the structure of MCs. All MCs in the mixture model share the same structure. For every discrete symbol value ever appearing in sequences, there is a corresponding state in the MCs. In addition, each MC has two special states: “start” and “end”. An event sequence can be uniquely mapped to a path from the “start” state to the “end” state. MCs are full meshed, i.e., a state can transit to any other state, including the state itself. There are two exceptions: the “start” state has no incoming transitions and the “end” state has no outgoing transitions. For a transition between two states, both its probability and duration are modeled. For the transition from state u and v , its probability is denoted by q_{uv} ; its duration is assumed to follow an Exponential distribution with a parameter λ_{uv} , which is also referred to as transition rate. As an example, Figure 2 shows the MC used for modeling the aforementioned TCP SYN/FIN/RST sequences.

To formally describe our problem, we need to introduce several notations. \mathcal{S} denotes the input sequence set. s denote one sequence instance in \mathcal{S} . N denotes the size of \mathcal{S} . A complete description of s is a tuple $\{b_1(s), t_{1,2}(s), b_2(s), t_{2,3}(s), b_3(s), \dots, b_{l(s)}(s)\}$, where $b_i(s)$ is the i -th symbol in s , $t_{i,i+1}(s)$ is the duration between the i -th and $i+1$ -th symbol, and $l(s)$ is the sequence length. We use M to denote the number of states in each MC. M includes

the “start” and the “end” state. We use Θ to denote the entire parameter set of the mixture model. We use θ^c to denote the parameter set of MC c ($c = 1, \dots, C$). θ^c includes all those transition related parameters q_{uv} and λ_{uv} . When necessary, we add a superscript to such transition parameters to specify which MC we refer to. For example, q_{uv}^c denotes the transition probability from state u to v in MC c .

Given the above notations, the probability of observing a sequence instance s is computed by

$$p(s|\Theta) = \sum_{c=1}^C \pi_c p(s|\theta^c)$$

where π_c is called mixture probability. It is also a parameter within Θ . π_c shall be interpreted as the *a priori* probability of assigning a sequence instance to MC c . π_c must satisfy the constraint $\sum_{c=1}^C \pi_c = 1$.

Clearly, constructing the above mixture model only requires prior knowledge on the total number of discrete symbol values. All the other parameters - including the number of MCs, transition probability and delay - will be estimated by input sequence data. Due to its generality, the model is applicable for various event sequence types. In the next section, we will present the procedure for parameter estimation.

B. Procedure for parameter estimation

We use the Expectation Maximization (EM) algorithm [3] to estimate parameters in the mixture model. EM algorithm is a general method for finding the maximum likelihood estimates for model parameters when the observed data is incomplete. The procedure of EM algorithm is presented below (Figure 3).

1. Let $l = 0$ and initialize Θ_l
2. do
3. $Q(\Theta, \Theta_l) = \sum_{m \in \mathcal{S}} \log p(\mathcal{S}, m|\Theta) p(m|\Theta_l)$
4. Compute $\Theta_{l+1} = \operatorname{argmax}_{\Theta} Q(\Theta, \Theta_l)$
5. $l = l + 1$
6. until (convergence criterion is satisfied)

Fig. 3. EM algorithm

In the EM algorithm, the estimate for Θ is updated in an iteration manner. In the $l + 1$ -th loop, a new estimate for Θ is obtained and denoted by Θ_{l+1} . Θ_{l+1} is computed based on training data and Θ_l , which is the estimate obtained in the l -th loop. The iteration procedure is guaranteed to converge to a solution Θ that yields a local maximum value for the target likelihood function $\log(\Theta|\mathcal{S})$. The key for applying the EM algorithm is to determine $Q(\Theta, \Theta_l)$ and to maximize $Q(\Theta, \Theta_l)$ in each loop, which are the so-called expectation step and maximization step respectively. Below we describe how to realize these two steps in our scenario.

From [4], we know the following equation holds in any

mixture model

$$Q(\Theta, \Theta_l) = \sum_{c=1}^C \sum_{s \in \mathcal{S}} \log[p_c(s|\Theta)] p(c|s, \Theta_l) + \sum_{c=1}^C \sum_{s \in \mathcal{S}} \log(\pi_c) p(c|s, \Theta_l) \quad (1)$$

where $p(c|s, \Theta_l)$ is the *posterior* probability of assigning s to MC c . Based on Bayes' theorem, $p(c|s, \Theta_l)$ is computed by $p(c|s, \Theta_l) = \frac{\pi_c p(s|\theta^c)}{\sum_{k=1}^C \pi_k p(s|\theta^k)}$ where θ^k and π_k are known from Θ_l .

In (1), $p_c(s|\Theta)$ is the probability of generating s from MC c which takes parameters from Θ . Its mathematical form can be derived as

$$p_c(s|\Theta) = p_c(s|\theta^c) = \prod_{j=1}^{l(s)-1} q_{s_j, s_{j+1}}^c p(\lambda_{b_j(s), b_{j+1}(s)}^c, t_{j, j+1}(s)) \quad (2)$$

where $p(\lambda_{b_j(s), b_{j+1}(s)}^c, t_{j, j+1}(s))$ is the *a priori* probability of observing a transition delay $t_{j, j+1}(s)$ given that the delay follows an Exponential distribution with a parameter $\lambda_{b_j(s), b_{j+1}(s)}^c$. Such a probability is determined by the following function

$$p(\lambda, t) = \begin{cases} e^{-\lambda \frac{t}{\Delta t}} (1 - e^{-\lambda}), & t \leq t_{max} \\ e^{-\lambda \frac{t_{max}}{\Delta t}} (1 - e^{-\lambda}), & t > t_{max} \end{cases} \quad (3)$$

where Δt and t_{max} are two user-defined constants. $p(\lambda, t)$ returns a fixed yet small probability when the transition delay t is larger than t_{max} . When t is less than or equal to t_{max} , $p(\lambda, t)$ returns the accumulated Exponential density probability over the time interval $[\frac{t}{\Delta t}, \frac{t}{\Delta t} + 1]$. In fact, alternative function forms can be chosen for $p(\lambda, t)$ based on application requirements. The parameters t_{max} and Δt should be carefully chosen because they affect the model accuracy and its sensitivity with respect to different properties of sequences. We will elaborate on this issue in Section II-G.

Now we need to maximize $Q(\Theta, \Theta_l)$ in (1). The two parts on the right side are independent and can be maximized separately. We firstly maximize the first part. By substituting (2)(3) into the first part, we rewrite it as follows

$$\sum_{c=1}^C \sum_{s \in \mathcal{S}} \sum_{j=1}^{l(s)-1} (\log q_{b_j(s), b_{j+1}(s)}^c - \lambda_{b_j(s), b_{j+1}(s)}^c t_{j, j+1}^*(s)) + \log \lambda_{b_j(s), b_{j+1}(s)}^c p(c|s, \Theta_l) \quad (4)$$

where $t_{j, j+1}^*(s) = \frac{t_{j, j+1}(s)}{\Delta t}$ if $t_{j, j+1}(s) \leq t_{max}$; otherwise $t_{j, j+1}^*(s) = \frac{t_{max}}{\Delta t}$. Note that the derivation of (4) uses an approximation $1 - e^{-\lambda} \approx \lambda$.

To maximize (4) with respect to each transition parameter, we consider the transition from state u to state v in MC c . It is possible that one transition appears in a sequence s for multiple times. Thus we use $I(s, u, v)$ to denote the number of times that the transition appears in s . We also use $\Gamma(s, u, v)$ to denote the summed duration of all such transitions. Again, when we compute $\Gamma(s, u, v)$, we use the

previously defined $t_{u,v}^*(s)$ whenever $t_{u,v}(s)$ is involved. Since transition probabilities are constrained by $\sum_{v=1}^M q_{uv}^c = 1$ for every u , we apply the Lagrange multiplier method to maximize (4) with respect to $q_{u,v}^c$ and $\lambda_{u,v}^c$. Without much difficulty we obtain

$$q_{u,v}^c = \frac{\sum_{s \in \mathcal{S}} I(s, u, v) p(c|s, \Theta_l)}{\sum_{i=1}^M \sum_{s \in \mathcal{S}} I(s, u, i) p(c|s, \Theta_l)} \quad (5)$$

$$\lambda_{u,v}^c = \frac{\sum_{s \in \mathcal{S}} I(s, u, v) p(c|s, \Theta_l)}{\sum_{i=1}^M \sum_{s \in \mathcal{S}} \Gamma(s, u, i) p(c|s, \Theta_l)} \quad (6)$$

Next, we maximize the second part of the right side in (1) with respect to π_c . Again the Lagrange multiplier method is applied because of the constraint $\sum_{c=1}^C \pi_c = 1$. π_c is solved as

$$\pi_c = \frac{\sum_{s \in \mathcal{S}} p(c|s, \Theta_l)}{N} \quad (7)$$

The three equations, (5) (6) and (7), describe how the parameters are estimated in the $l + 1$ -th loop of the iteration procedure.

C. Using protocol knowledge to fix certain parameters

It is possible to fix certain parameters by domain knowledge on involved networking protocols. This not only improves the model accuracy, but also reduces the computational complexity since those fixed parameters no longer participate in the parameter estimation procedure. Both transition probabilities and rates can be fixed. Below are two commonly used rules:

- if we ensure that a transition from state u to state v is impossible in reality, we could assign zero to the corresponding transition probability. For example, in the TCP SYN/FIN/RST scenario, it is clear that a FIN packet is never followed by a SYN in one TCP connection. Accordingly, the transition probability from the “FIN” state to the “SYN” state is set to zero, and the corresponding transition rate is zero as well.
- if we ensure that two events occur almost simultaneously, we could assign a large constant to the corresponding transition rate. Considering the VoIP example, RING and OK messages are always sent together by the VoIP server. Thus their corresponding transition rate could be set to a large constant.

A subtle issue arises in this case. While we fix the transition probability (or rate) between two states, we need to decide how many MCs in the mixture model are affected. In principle, such a decision depends on the confidence level of the rule we obtained from domain knowledge. A high confidence level implies that the rule always holds and no exceptions will exist; if this is true, the parameter fixing operation should be applied to all the MCs. Otherwise, only a subset of MCs should be affected as the rest of MCs are expected to be able to accommodate exceptions.

D. Assignment of initial parameter values

After certain parameters are fixed by domain knowledge, the rest of parameters will be estimated in the iterative procedure.

Their initial values need to be assigned before the iteration procedure. In practice, we generate random values as their initial values. A normalization operation is imposed to ensure that those initial values satisfy those constraints such as $\sum_{c=1}^C \pi_c = 1$ and $\sum_{j=1}^M q_{ij}^c = 1$. As the EM algorithm only guarantees local optimality, the accuracy of parameter estimation is affected by the initial values. Therefore, we repeat the parameter estimation multiple times (10 in our experiment) with different initial values. We then choose the model yielding the best compliance with the convergence criterion (Line 6 in Figure 3). Regarding the convergence criterion, we compute $Q(\Theta, \Theta_l)$ upon the finish of the l -th loop. If $\frac{|Q(\Theta, \Theta_l) - Q(\Theta, \Theta_{l-1})|}{Q(\Theta, \Theta_{l-1})} < \delta$ (δ is a user-defined threshold which is set to 0.05% in our experiment), the iteration is terminated.

E. Selection of C

Before constructing the mixture model, we need to specify C , the total number of MCs. Selecting an appropriate value for C is critical because it gives a tradeoff between accuracy and complexity. In our method, we use the Akaike Information Criterion (AIC) [5] which is defined as

$$AIC(\Theta) = -2 \log p(\mathcal{S}|\Theta) + 2U$$

where U is the number of parameters being estimated. The total parameter count is $C(2M^2 - 7M + 6) - 1$. Nevertheless, U should be much smaller than this number after fixing certain parameters by domain knowledge. The introduced AIC measures the goodness of fit for a model. A chosen value of C that minimizes $AIC(\Theta)$ is interpreted as striking a balance between model accuracy and complexity. When applying AIC, we first specify an upper bound for C , we then search below the upper bound for a value for C that minimizes $AIC(\Theta)$. How to choose the upper bound depends on the number of states in MCs, and also our prior knowledge on how diverse sequence behavior could be. A high upper bound should be chosen if MC has many states or the sequence behavior is diverse.

F. Computational complexity

The major computational complexity is incurred by the three equations (5) (6) and (7). Their required computation is to multiply N by U , which is bounded by $O(CM^2)$ according to Section II-E. Prior to solving these three equations, (2) and (3) must be computed. While the complexity of (2) is $O(CN)$, computing (3) involves an Exponential function which can be replaced by a lookup table in implementation. So the complexity for (3) is $O(M^2)$. The summed computational complexity in one iteration is upper bounded by $O(NCM^2 + CN + M^2) \approx O(NCM^2)$. This complexity, multiplied by the number of iterations, is the total complexity. In all our experiments, the number of iterations is less than twenty.

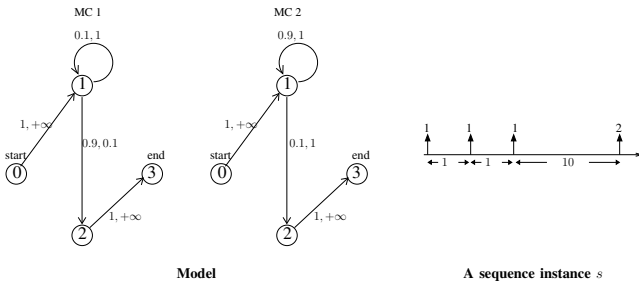


Fig. 4. An example

G. Analysis of model sensitivity

As mentioned early, the proposed model is expected to characterize two aspects of sequence behavior: the sequential pattern constituted by symbols, and durations between symbols. It is important to understand the model sensitivity with respect to each aspect because this decides the capability of the model in distinguishing sequences with different behavior. In this section we analyze this model sensitivity issue. We first use an example to explain how the model sensitivity is affected by parameter selection. We then discuss how the model sensitivity can be tailored for application requirements.

Figure 4 shows a mixture model comprising of two MCs. The two MCs are assumed to have equal mixture probability, that is, the *a priori* probability for an sequence instance generated by either MC is equal. Note that in the two MCs only those edges with non-zero transition probabilities are depicted. Each edge is labeled by its transition probability and rate. We first compare the likelihood for a sequence instance s being generated by MC 1 and 2, respectively. In MC 1, the expected transition delay from state 1 to itself is 1, and the expected delay from state 1 to 2 is $1/0.1 = 10$. These two values are consistent with our observation on s . Therefore, if transition delay is our primary concern, s should be considered more likely modeled by MC 1 since the aspect of transition delay in MC 1 explains the sequence instance well. On the other hand, if the symbol order is our primary concern, s should be better modeled by MC 2 because in MC 2 the self transition probability at state 1 is 0.9, much higher than in MC 1, and this property is consistent with the observed multiple consecutive symbol “1” in s . This simple example shows that there is a tradeoff between being more sensitive to the symbol order and being more sensitive to transition delay.

We now show that the model sensitivity is controlled by two parameters: t_{max} and Δt (see (3)). From (3), it is clear that the difference of transition delay only matters when the duration is less than or equal to t_{max} . If t_{max} is small, the model tends to be insensitive to variance of transition delay. In our last example, if we let $t_{max} = 100$, which is larger than any transition delay in s , and let $\Delta t = 1$, then we compute the probability for s being generated from each MC as follows

$$\begin{aligned} p(s|\theta^1) &= [0.1 \times (e^0 - e^{-1})]^2 \times 0.9 \times (e^{-1} - e^{-1.1}) \\ p(s|\theta^2) &= [0.9 \times (e^0 - e^{-1})]^2 \times 0.1 \times (e^{-10} - e^{-11}) \end{aligned}$$

There is $p(s|\theta^1) \gg p(s|\theta^2)$. Since the two MCs have equal mixture probabilities, we know the model has a higher *posterior* probability to assign s to MC 1 than to MC 2. On the other hand, if $\Delta t = 10$ and t_{max} is unchanged, there is $p(s|\theta^1) \ll p(s|\theta^2)$, the assignment becomes different. This example gives the following insight: while a smaller Δt makes the model more sensitive to transition delay, a larger Δt makes it more sensitive to the order of symbols. In an extreme scenario, we might be only concerned about whether the order of symbols follows certain protocol logic, then we should let $t_{max} = 0$. In the other extreme scenario where we are only interested in duration, we will let t_{max} larger than any possible duration values and let Δt close to zero.

Lastly, we note that it is possible to specify different t_{max} and Δt for multiple transitions. This feature is particularly useful when transitions have unequal importance. E.g., if a transition is not within our concern, we can adjust its t_{max} and Δt to let the mixture model ignore the impact of its variance.

III. EMPIRICAL EVALUATION

In this section, we empirically evaluate the performance of the proposed method. In Section III-A we describe the dataset we used. In Section III-B we present the evaluation methodology and results.

A. Dataset

In our experiments we still consider the aforementioned three event sequence types: TCP SYN/FIN/RST sequences, SIP-based VoIP calls and Wi-Fi user sessions. The used dataset are three network traces, including a TCP packet trace collected from an edge router, a VoIP call traces from our own testbed, and a syslog trace in a campus-wide Wi-Fi wireless network.

The TCP trace is publicly available at [1]. It was collected from an edge router at UCLA during August 2001. The trace is about 16 hours long. From the trace we extract all the SYN, FIN, RST packets and group them into event sequences by flow identity, i.e., the tuple (source IP, destination IP, source port, destination port). In total we extract 505K TCP SYN/FIN/RST sequences. The VoIP trace was collected from a VoIP testbed that we set up on PlanetLab. We use the open-source software SIPp [6] as VoIP clients. We use another open-source software SER [7] as the SIP server. SIPp is deployed on twenty PlanetLab nodes. They generate calls to the SIP server. We then use tcpdump to capture packets at the server and extract all the call sessions. To make the traffic more realistic, we intentionally create packet loss and delay at both call senders and receivers. The trace is about 1.5 hour long, contains 16K call sessions. The third dataset is a syslog trace collected from the campus-wide wireless networks in Dartmouth College [8]. The syslog trace contains all the Wi-Fi management frames exchanged between wireless users and their associated access points. In our experiment, we only use the trace for one access point in the first two weeks of 2003. This trace subset has 2.2M management frames which are

Data	Δt	t_{max}	C	Significant MCs	Run-time
TCP	1 sec	10 min	6	4	7 min
VoIP	1 sec	10 min	12	2	90 sec
Wi-Fi syslog	30 sec	2 hour	26	5	8 min

TABLE I

EXPERIMENT SETTINGS AND BASIC RESULTS (“SIGNIFICANT MCs”: THE NUMBER OF MCs WITH MIXTURE PROB. LARGER THAN 5%)

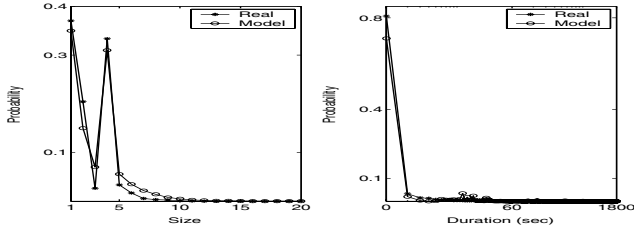


Fig. 5. Histogram of TCP sequence size and duration

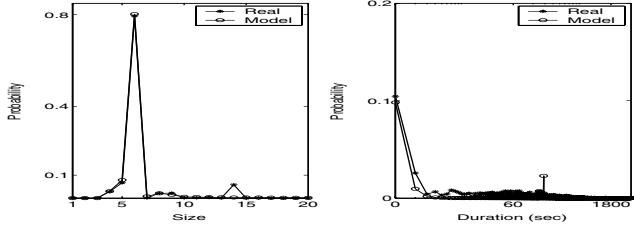


Fig. 6. Histogram of VoIP call size and duration

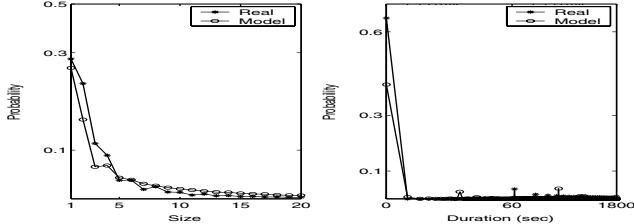


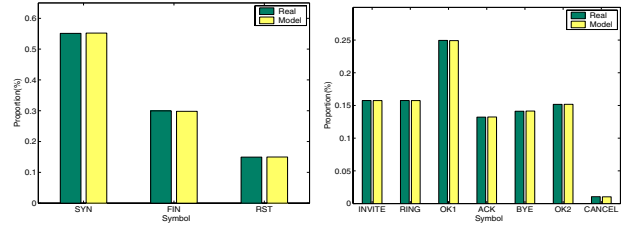
Fig. 7. Histogram of Wi-Fi user session size and duration

grouped into 122K sessions based on MAC address and user activity.

B. Model accuracy

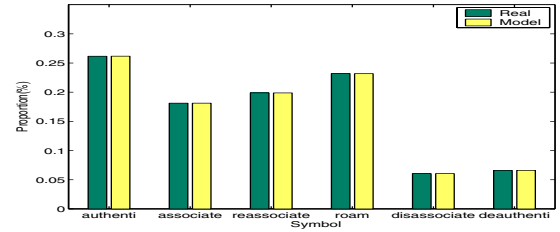
We test our method against the three traces on a Intel Xeon(R) 1.60GHz processor. Table I summarizes the experiment settings and basic results. The column “Significant MCs” is particularly interesting. It reveals how many popular behavior templates the input sequences have. We find that such significant MCs are closely related to the network protocol underlying the data. We will explain this more in later of this section.

We use two metrics to evaluate the accuracy of our method: *histogram of sequence size* and *histogram of sequence duration*. Both metrics are self-evident. For either metric, we compare it between the input data and the fitted model. A consistency reported by the comparison indicates that the model does capture the input sequence behavior. While computing the histograms for the input data is straightforward, it is difficult to



8.1: TCP sequences

8.2: VoIP calls



8.3: Wi-Fi user sessions

Fig. 8. Comparison of histogram for discrete symbols

analytically derive the histograms from the created models. We overcome this problem by applying Monte Carlo simulations. The comparison is conducted for all the three datasets and the results are provided in Figure 5-7. In all the figures we use a logarithmic scale for the sequence duration axis.

Figure 5 compares the two histograms for the TCP dataset. We notice that the sequence size histogram has two spikes at size 1 and 4 (see the left figure). The spike at size 1 is caused by the fact that many TCP connections have a single SYN without any FIN or RST. The spike at size 4 is caused by the most common case wherein a TCP connection has two SYNs and two FINs. For these two spikes, our model seems to give an accurate explanation. The right side of Figure 5 shows the histogram of sequence duration. Both real data and the fitted model show that more than 70% of TCP connections last less than one second. Figure 6 plots the histograms for VoIP call sessions. The VoIP call size histogram has a sharp spike at size 6. This corresponds to the most typical VoIP call pattern shown in Figure 1. On the other hand, the VoIP call duration has a relatively flat distribution. In either case, the model reflects similar statistics to the real data. Last, we look at the histograms about Wi-Fi user sessions in Figure 7. We notice that the histogram of Wi-Fi user session size is relatively flat, yet the histogram of session duration is uneven. Such an observation is consistent with the results reported in [8].

We further break down the histogram of sequence size to examine each discrete symbol value. The results are shown in Figure 8 which demonstrates that the model matches the real data extremely well in terms of the histogram of each discrete symbol value.

IV. APPLICATIONS

In this section we present two application showcases based on our method. The first showcase is a visualization tool that allows network operators to conduct exploratory data analysis (Section IV-A); the second showcase is a simple anomaly detection mechanism (Section IV-B).

A. A model visualization tool

By the nature of mixture models, our method classifies input event sequences into different MCs. Each MC is a cluster that profiles a group of sequences with similar behavior. This feature is attractive to various exploratory data analysis tasks such as monitoring dominant traffic trend, detecting outliers, and developing traffic model.

As a showcase, we developed a tool to visualize the clusters produced by our method. Its purpose is to use graphic display to aid network operators in rapidly understanding the information conveyed by the mixture model. Figure 9 is the tool's display of the four most significant clusters for the TCP dataset. The tool has one unique feature: it encodes transition delay by the thickness of the corresponding edge, and it encodes the transition rate by the scattering of the corresponding edge. This feature, along with the mixture probability numbers appearing on the display, enables the operators to have a coarse, yet immediate estimate of the status of the overall data. For example, by examining MC 1 in Figure 9, we infer that 42% of TCP connections either only have a SYN, or have a SYN and a RST. Now if we examine MC 2, we infer that 35% of TCP connections have both SYN and FIN. Since the SYN and FIN have non-negligible self transition probabilities in this case, such 35% of TCP connections may have multiple consecutive SYNs (or FINs). Moreover, if consecutive SYNs occur, their intermediate duration is short. While for consecutive FINs, the duration is relatively long. We further examine MC 3 and 4: MC 3 shows that 13% of TCP connections have a single or multiple consecutive SYNs. MC 4 shows that 6% of TCP connections may have multiple consecutive SYNs followed by zero or one FIN.

In the above scenario, the tool displays only the significant MCs by pruning those insignificant ones. This enables operators to promptly catch the dominant trend hidden in the data and take actions if necessary. For example, it is well known that most TCP RSTs are caused by network outages, malicious attacks, or implementations of certain Web browsers [9]. If our visualization tool reveals that the popularity of TCP RSTs is increasing sharply, operators could be alarmed to check whether a TCP reset attack [10] is launched. On the other hand, our tool can also display those insignificant, yet possibly unusual behavior. Whether enabling this option is determined by operators' requirements.

B. Anomaly detection

Because our modeling approach provides one way to describe the behavior of network objects, it is readily applicable to anomaly detection whose duty is essential to distinguish abnormally-behaved objects from normally-behaved

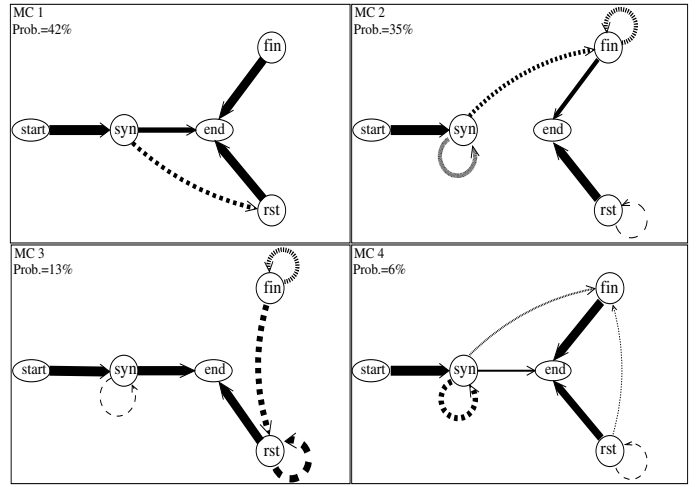


Fig. 9. Visualization of the four most significant MCs after building models for the TCP SYN/FIN/RST dataset. Transition probabilities in each MC are encoded by thickness of the edges. A thicker edge indicates a higher probability and vice versa. Edges with transition probabilities smaller than 5% are pruned in the figure. Transition rates are encoded by the ratio of dash length and dash gap of the edges. A more scattered dashed line indicates a smaller transition rate, i.e., a longer expected transition delay.

ones. There could be many ways to design an anomaly detection method. Nevertheless, in this section, we propose a simple method by using our modeling method and the visualization tool mentioned earlier.

In anomaly detection we usually have two event sequence sets: \mathcal{S} and $\hat{\mathcal{S}}$. While \mathcal{S} is known to be clean without anomalies, $\hat{\mathcal{S}}$ is the test dataset that we do not know whether it contains anomalies. In reality, $\hat{\mathcal{S}}$ could be generated by one host or user, or the data collected in a monitoring period. The goal of our anomaly detection technique is to determine whether the behavior of $\hat{\mathcal{S}}$ is different from \mathcal{S} by certain criterion. If they are different, $\hat{\mathcal{S}}$ is reported to contain anomalies.

We first use our presented method to build a model Θ for \mathcal{S} . Θ is considered to represent the normal behavior. In principle we can build another model for $\hat{\mathcal{S}}$ and compare it with Θ . If the two models have significantly different parameter values, we detect $\hat{\mathcal{S}}$ contains anomalies. Despite its seemingly simplicity, this approach suffers from one drawback: training two models are time-consuming and $\hat{\mathcal{S}}$ is usually small and insufficient for training a reliable model. Therefore, we take the following alternative approach. After obtaining Θ , we extract the mixture probabilities π_c from Θ . π_c are the *a priori* probability for a sequence coming from MC c . Such a distribution is denoted by $P(MC\ c) = \pi_c$. We will use this distribution to represent the normal behavior. Next, we apply Θ to calculate the so-called membership probability $p(c|s, \Theta)$ for every s in $\hat{\mathcal{S}}$. $p(c|s, \Theta)$ is the probability for assigning s to MC c . It satisfies the constraint $\sum_{c=1}^C p(c|s, \Theta) = 1$. We further compute

$$E_c = \sum_{s \in \hat{\mathcal{S}}} p(c|s, \Theta), \quad c = 1, 2, \dots, C$$

where E_c shall be interpreted as the number of sequences in $\hat{\mathcal{S}}$ that are assigned to MC c . It is also called the observed frequency count for MC c .

Clearly, if the sequence behavior in $\hat{\mathcal{S}}$ and in \mathcal{S} are similar, the observed frequency counts E_1, \dots, E_C should follow the *a priori* distribution $P(MC\ c) = \pi_c$. Thus we use this fact as the basis for anomaly detection. Specifically, we apply χ^2 (chi-square) statistics to measure the fitness between the observed frequency counts and the distribution, that is, we compute

$$\chi^2 = \sum_{c=1}^C \frac{(|\hat{\mathcal{S}}|\pi_c - E_c)^2}{E_c}$$

where $|\hat{\mathcal{S}}|$ is the size of $\hat{\mathcal{S}}$, $|\hat{\mathcal{S}}|\pi_c$ is the expected frequency count for MC c . Because the value of χ^2 is affected by $|\hat{\mathcal{S}}|$, we normalize it in the following way

$$C_c = \sqrt{\frac{\chi^2}{\chi^2 + |\hat{\mathcal{S}}|}}$$

The above C_c is usually called *contingency coefficient* [11]. Since it is normalized to be between zero and one, it is readily used to measure how likely an anomaly is detected. It equals zero when $\hat{\mathcal{S}}$ exhibit identical behavior as \mathcal{S} ; and it equals one when their behavior do not resemble at all. In practice, a high value of C_c indicates $\hat{\mathcal{S}}$ might be anomalies and require a closer examination.

Overall, the presented technique requires to build a mixture model for a clean set of sequences. This only needs to be done occasionally under a reasonable assumption that the normal behavior does not change frequently. For each testing sample set $\hat{\mathcal{S}}$, our technique only needs to compute the membership probability for each sample. The technique is light-weighted and can be performed either online or offline.

For the purpose of evaluation, we use our technique to conduct an offline anomaly detection on the aforementioned TCP dataset. Since we do not have a clean dataset, we use the mixture probability distribution computed from the entire dataset to represent normal behavior. We believe that this is valid as long as the proportion of anomalies in the entire dataset is small. We then divide the 16-hour trace into 30-minute intervals and compute the contingency coefficient for every interval. The results are shown in Figure 10 which shows a contingency coefficient larger than 0.4. This value is relatively high and indicates a suspicious interval. Although our anomaly detection technique does not require to model sequences in the suspicious interval, in this experiment we have done so to reveal more information. We train a model by using the 13K TCP sequences in the suspicious interval. We then apply our visualization tool to display the modeling results which are shown in Figure 11. It is interesting to make a visual comparison between Figure 11 and Figure 9, which is assumed to reflect the normal behavior. By reading the mixture probability numbers and estimating the thickness of edges on the two figures, we easily realize that in the suspicious interval a much higher proportion of TCP connections contains only a

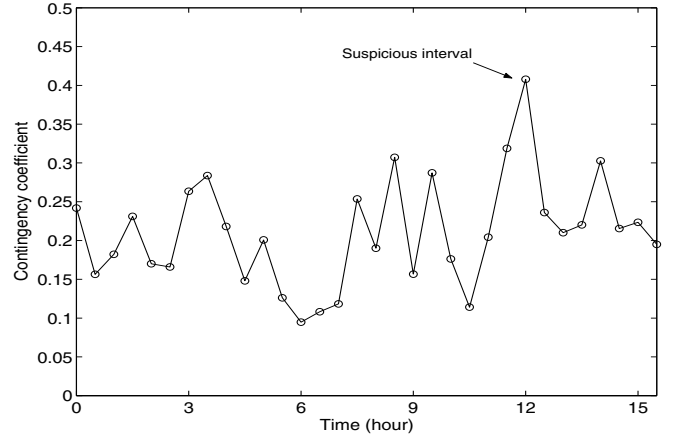


Fig. 10. Anomaly score for TCP sequence dataset

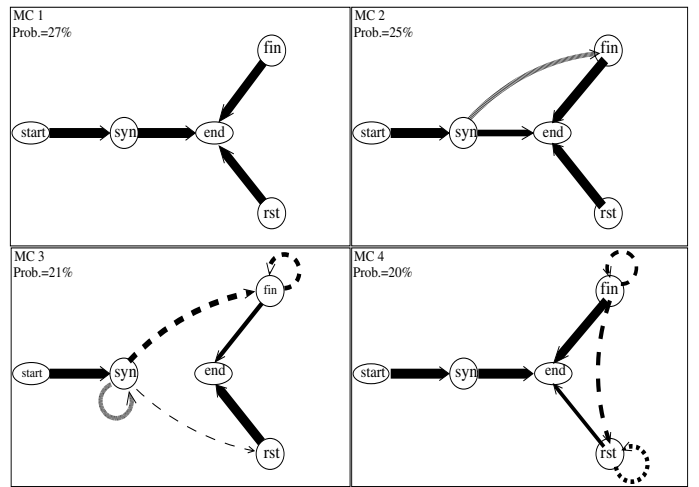


Fig. 11. Visualization of the most significant MCs for the suspicious 30-min period in Figure 10

single SYN without FIN or RST. Specifically, in the suspicious interval this percentage is about 60%. While in the normal behavior model the percentage is about 33%. As our future work, we plan to further investigate whether a SYN attack occurs in the suspicious interval.

V. RELATED WORK

There are a large number of prior work on modeling or mining sequence-alike data in areas such as speech recognition, bio-informatics, database and database mining. While it is infeasible to list all such work, we name a few representative ones. In speech recognition area, [12], [13] apply Hidden Markov Model (HMM) to cluster a string of acoustic units. In database and database mining area, mining sequential patterns has been a hot topic in the last decade. Many of these work focus on discovering rules (e.g., [14], [15], [16]) instead of modeling. [17] is perhaps the most relevant work. [17] clusters WWW users' navigation records by using a mixture model consisting of first-order Markov Chains. However, they do not

take into account the time information as we did. In security area, several work detect anomalies (or intrusion) by mining sequential system calls [18], [19]. In contrast to all these work, our method can capture both the sequential pattern and time information for sequential data. This is particularly suitable for networking problems.

Surprisingly, in networking area there are few studies on sequence-alike data analysis and applications. Some recent efforts (e.g., [20], [21], [22], [23]) are devoted to inferring properties of traffic flow. Despite the seemingly similarity between their flow concept and the event sequence in this paper, a clear difference exists: our method is targeting those network behavior that can be described by a sequence of symbols which take values from a limited number of discrete values. Our method is not intended to be applied to some of their studied flows (e.g., TCP packet sequences). So it is inappropriate to make a direct comparison between our work and the aforementioned work. Nevertheless, we notice that most of these work capture only one single property of traffic flows. It remains an open question on how to have a complete characterization of flow properties.

Some recent work ([24], [25], [26], [27], [28], [29]) apply various data mining techniques to identify significant patterns or insignificant anomalies from various traffic entities such as flows, Internet backbone traffic and host communication patterns. All these work share a common goal that the operators need tools to conveniently monitor massive traffic. Our method can achieve this goal via a different approach.

VI. CONCLUSION

Automatic profiling of massive event sequences is critical for a variety of network measurement and monitoring tasks. In this work we propose a novel approach that automatically captures the behavior hidden in massive event sequences. Our approach is based on a mixture model - a collection of Markov Chains. The most salient feature of the model is that it simultaneously captures both the order of events and duration between events. Prior domain knowledge on the event sequences can be seamlessly integrated into our model to improve accuracy and to reduce complexity. To estimate parameters of the model, we develop an iterative algorithm based on the Expectation Maximization algorithm. We evaluate our method on multiple network traces, including a TCP packet trace, a VoIP call collection and Wi-Fi syslog. The experimental results demonstrate that our method yields a high consistency with the real data. Furthermore, we explore the applicability of our method to two network monitoring tasks. Specifically, we develop a visualization tool for conducting exploratory traffic analysis. We also propose an anomaly detection scheme that is light-weighted and can be conducted online or offline.

REFERENCES

- [1] Packet traces from UCLA Computer Science Department, "http://fimg-www.cs.ucla.edu/ddos/traces/".
- [2] R. Birke, M. Mellia, M. Petracca, and D. Rossi, "Understanding VoIP from backbone measurements," in *IEEE INFOCOM*, 2007.
- [3] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. 39, no. B, pp. 1–38, 1977.
- [4] J. A. Bilmes, "A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," *ICSI-TR-97-021*, 1997.
- [5] H. Akaike, "Information theory and extension of the maximum likelihood principle," in *2nd International Symposium on Information Theory*, 1973, pp. 267–281.
- [6] SIPp, "http://sipp.sourceforge.net."
- [7] SIP Express Router, "http://www.iptel.org/ser/."
- [8] D. Kotz, T. Henderson, and I. Abyzov, "CRAWDAD data set dartmouth/campus (v. 2007-02-08)," Downloaded from <http://crawdad.cs.dartmouth.edu/dartmouth/campus>, Feb. 2007.
- [9] M. Arlitt and C. Williamson, "An analysis of TCP reset behaviour on the internet," *ACM Computer Communication Review*, vol. 35, no. 1, pp. 37–44, February 2005.
- [10] P. A. Watson, "Slipping in the window: TCP reset attacks," 2003, technical whitepaper.
- [11] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical recipes in C: the art of scientific computing*. New York, NY, USA: Cambridge University Press, 1988.
- [12] L.R.Rabiner, C.H.Lee, B.H.Juang, and J.G.Wilpon, "HMM clustering for connected word recognition," in *proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 1989.
- [13] X.D.Huang, Y.Ariki, and M.Jack, *Hidden Markov models for speech recognition*. Edinburgh University Press, 1990.
- [14] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Int. Conf. Very Large Data Bases VLDB*, 1994, pp. 487–499.
- [15] R.Agrawal and R.Srikant, "Mining sequential patterns," in *proceedings of IEEE ICDE*, March 1995.
- [16] R. Srikant and R. Agrawal, "Mining sequential patterns: generalizations and performance improvements," in *Inc. Conf. on Extending Database Technology EDBT*, 1996, pp. 3–17.
- [17] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White, "Model-based clustering and visualization of navigation patterns on a web site," *Data Mining Knowledge Discovery*, vol. 7, no. 4, pp. 399–424, 2003.
- [18] W. Lee and S. J. Stolfo, "Data mining approaches for intrusion detection," in *Proc. of the 7th USENIX Security Symposium*, January 1998.
- [19] P.Smyth, "Hidden Markov monitoring for fault detection in dynamic systems," *Pattern Recognition*, vol. 27, no. 1, pp. 149–164, 1994.
- [20] V. Paxson, "Empirically-derived analytic models of wide-area TCP connections," *IEEE/ACM Transactions on Networking*, vol. 2, no. 4, August 1994.
- [21] N. Duffield, C. Lund, and M. Thorup, "Estimating flow distributions from sampled flow statistics," in *proceedings of ACM SIGCOMM*, 2003.
- [22] A. Kumar, M. Sung, J. xu, and J. Wang, "Data streaming algorithms for efficient and accurate estimation of flow size distribution," in *ACM SIGMETRICS*, 2004.
- [23] X. Meng, S. Wong, Y. Yuan, and S. Lu, "Characterizing flows in wireless data networks," in *ACM MOBICOM*, 2004.
- [24] C.Estan, S.Savage, and G.Varghese, "Automatically inferring patterns of resource consumption in network traffic," in *proceedings of ACM SIGCOMM*, 2003.
- [25] F.Hernandez-Campos, A.B.Nobel, F.D.Smith, and K.Jeffay, "Statistical clustering of Internet communication patterns," in *Proc. of Symposium on the Interface of Computing Science and Statistics*, 2003.
- [26] K. Xu, Z. Zhang, and S. Bhattacharyya, "Profiling Internet backbone traffic: behavior models and applications," in *ACM SIGCOMM*, 2005.
- [27] A.Lakhina, M.Crovella, and C.Diot, "Mining anomalies using traffic feature distributions," in *ACM SIGCOMM*, 2005.
- [28] T.Karagiannis, K.Papagiannaki, and M.Faloutsos, "BLINC: multilevel traffic classification in the dard," in *ACM SIGCOMM*, 2005.
- [29] A. Soule, K. Salamatian, N. Taft, R. Emilion, and K. Papagiannaki, "Flow classification by histograms: or how to go on safari in the internet," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 49–60, 2004.