

Edge Concierge: Democratizing Cost-Effective and Flexible Network Operations using Network Layer AI at Private Network Edges

Anan Sawabe^{*†}, Takanori Iwai^{*}, Kozo Satoda^{*}, and Akihiro Nakao[†]

^{*} System Platform Research Laboratories, NEC Corporation

1753 Shimonumabe, Nakahara-ku, Kawasaki, Kanagawa, Japan

[†] Graduate School of Interdisciplinary Information Studies, the University of Tokyo

7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan

Email: { a-sawabe@cb, t-iwai@hx, k-satoda@cb }.jp.nec.com, nakao@nakao-lab.org

Abstract—We observe two major revolutionary trends in network operations: democratization of cost-effective and flexible communication means for vertical players, such as public safety, by private mobile networking combined with edge computing, and automatic and autonomic network operations empowered by Artificial Intelligence (AI). Further innovations are required for making private networking readily available for vertical players that are reluctant to acquire expertise in complex network operations. We propose *Edge Concierge*, of which concept is to democratize cost-effective and flexible network operations using *network layer AI* at private network edges. *Edge Concierge* assists smart network operations for private mobile network operators and energy saving by changing working state of AI-empowered anomaly detection applications by *network layer AI*. We also employ unsupervised machine learning using Hidden Markov Model (HMM) for estimating contexts by solely observing network traffic at mobile edge computing (MEC) middle boxes. In detail, we design a system of real-time and self-learning context estimation by a multi-level probabilistic state transition model trained by unsupervised learning, which is implemented in a commodity PC. In order to evaluate our proposed system, we take public safety context of smart cities as an example use case and show the benefits.

I. INTRODUCTION

We observe two major revolutionary trends in network operations. First, *democratization* of telecommunications catches attentions all over the world, where the latest mobile network technologies can be licensed to non-operators, such as Private 5G in Germany and Local 5G in Japan. Combined with edge computing, private networking is considered *cost-effective* and *flexible* communication means for vertical players, such as local governments and smart cities resolving various regional issues, e.g., public safety and environmental sensing without incurring public telecommunication cost. Second, Artificial Intelligence (AI) is considered a powerful tool enabling *automatic* and *autonomic* network operations in general.

However, in the context of making private networking readily available for vertical players, we posit that the existing solutions in each of these trends are still premature and require further innovations. Specifically, we observe two challenges:

first, while vertical players increasingly demand cost-effective solutions, they are reluctant to acquire expertise in complex network operations, which is both time-consuming and costly in terms of manual labors. In other words, verticals are looking for “out of box” unattended solutions for network operations. Therefore, AI-empowered network solution is not just a useful tool, but a necessity for private network operations. Second, AI-aided private network operations must also be inexpensive as otherwise it would defeat the original purpose of cost effective supportive solutions, e.g., for AI-empowered anomaly detection with network connectivity for thousands of devices.

In the light of these observations, we propose *Edge Concierge* at your service and attempt to democratize cost effective network operations for private mobile networking. *Edge Concierge* is a system to assist smart network operations for private mobile network operators and energy saving by changing working state of AI-empowered anomaly detection applications via AI-empowered traffic analysis at the network layer, namely *network layer AI*. We also employ *unsupervised machine learning* using *Hidden Markov Model (HMM)* for identifying application states, namely *contexts*, by solely observing network traffic at mobile edge computing (MEC) middle boxes, so that vertical players may enjoy an unattended solution as much as possible.

We believe our proposed system are different from the existing work in that it has two innovative features: first, *network layer AI* brings further benefit to vertical players in addition to low-cost private connectivity, that is, low energy consumption. AI-empowered anomaly detection applications consume a vast amount of electricity because deep learning, which is a representative AI technique, generally requires rich hardware such as a GPU to run. Because of this, energy consumption of AI-empowered applications can be saved by changing working state to work only while detecting suspicious behaviors by *network layer AI* at low computing cost. We aim this feature to be as simple a solution as possible that can be implemented using only a commodity PC.

Second, our proposed system automatically interprets context by a multi-level probabilistic state transition model using

Hidden Markov Model (HMM), and thus by unsupervised machine learning. Our system is applicable to encrypted traffic as the input to our model is only time-series of packets and also executable in real-time and online. Some have attempted to understand application information from encrypted traffic without inspecting packets by supervised learning [1]–[3]. Others identify applications from statistics of traffic patterns, i.e., packet size distributions. These methods require labeled traffic to train the analysis model and also need to be tailored to a specific private mobile networking environment. To the best of our knowledge, no other system can interpret application context without collecting labeled traffic in advance.

The rest of this paper is structured as follows. We first discuss the technical challenges of context estimation in Section II, and we describe related work on traffic analysis in Section III. After that, we design a context estimation system and formulate problems in each function in Section IV. The formulated problems are solved by the proposed traffic analysis model based on HMM in Section V. In order to evaluate our proposed system, we take *public safety* context of local governments and smart cities as an example use case and show the benefits in Section VI. Finally, we conclude this paper in Section VII.

II. TECHNICAL CHALLENGES

In this section, we describe challenges we address in our proposed system. *Edge Concierge* mainly addresses traffic analysis problems to estimate context changes in the application layer by analyzing traffic features in the network layer. To provide an unattended and inexpensive solution for vertical players, we face the following three challenges.

The first challenge is *real-time*. If context changes are estimated in real time, *Edge Concierge* can provide connectivity of many devices and low energy consumption of AI-empowered applications. As for connectivity, real-time estimating context changes, e.g., smartphone user activities, enables network operations to shift and flatten traffic with contexts of each device. As for energy consumption in case of surveillance cameras, AI-based recognition applications can change the working state, e.g., active while detecting anomaly behavior by *Edge Concierge* and idle while no movement.

The second challenge is *self-learning*. To provide unattended solutions for vertical players, self-learning capability is required. In other words, context estimation should be executable without manual labelling or task supervision.

The third challenge is *inexpensiveness*. While addressing above challenges, estimating context changes should be implemented in commodity PC in order not to defeat the original purpose of inexpensiveness.

Finally, we describe two assumptions regarding our traffic analysis in this paper. The first assumption is that traffic patterns change with context changes. For example, smartphone traffic patterns depend on the user activity, e.g., watching videos. In addition, video traffic patterns of a network camera change depending on objects in its shooting coverage, e.g., no movement and people moving. The second assumption is

that contexts cannot be separated by only using 5-tuple. For example, multiple flows from a smartphone are generated by doing an activity. In addition, video traffic transmitted by a network camera via a single flow changes with contexts.

III. RELATED WORK

With the spread of machine learning (ML) that effectively categorizes statistical features into labels, many researchers have attempted to understand application information, e.g., protocol and application, from traffic features. We here discuss related work from viewpoint of real-time and self-learning.

Real-time estimation: Some researchers [4]–[10] analyze flow statistics, e.g., duration and flow size, which requires a finished flow set to calculate traffic features. To identify applications before flow finishing, other researchers have analyzed several packets of head of flow (HoF) [11], [12]. Although flow based methods are effective in identifying application, we do not adopt these methods as a context change estimator because context changes cannot be separated by 5-tuple as discussed in Section II. To estimate context changes, we find straightforward methods [13]–[17] that first extract traffic features, e.g., packet size and interval statistics, from packets sampled in a Sliding Window (SW) and then identify application via supervised ML, e.g., Random Forest (RF), referred to as SW+RF [13]. Although this method can be adopted to context estimation, estimation accuracy degrades due to statistical problems in following two cases. The first case is to use traffic features mixing multiple contexts in a large window. The second case is to use traffic features from only few packets in a small window. In this paper, we construct traffic analysis model for accurate estimation and compare estimation accuracy of our system and SW+RF in Section VI.

Self-learning estimation: These methods require collecting labeled data for training their traffic analysis model. Some researchers have proposed collecting training data set automatically. In the straightforward method, network operators collect raw application information by decoding encrypted traffic in a specific proxy server. Moreover the method in [12] collects packets with application tags from customized smartphones. These methods require to customize each device or applications for labelling for network operators, so that these methods do not meet our expectations for unattended solutions. On the other hand, although unsupervised traffic classification methods are proposed in the fields of anomaly detection [18], pre-training baseline dataset is also required for detecting unlearned traffic patterns. To the best of our knowledge, there is no method of estimating context in real-time without labelling and supervising.

IV. SYSTEM DESIGN FOR REAL-TIME AND SELF-LEARNING CONTEXT ESTIMATION

In this section, we first design a real-time and self-learning context estimation system that has several functions, i.e., online estimation, model training, and model updates. After that we formulate problems solved in each function.

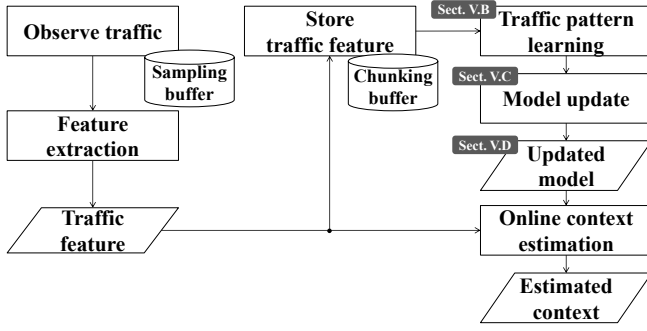


Fig. 1. Real-time and self-learning context estimation system.

A. System design of traffic analysis

We design context estimation system as illustrated in Figure 1. The workflow is as follows. Observed packets are first stored in a buffer with Δ_1 -second timer, referred to as the sampling buffer. When the timer expires, we calculate four traffic features per uplink and downlink: traffic volume [Mb], average packet size [B], average packet intervals [s], and number of packets [packet]. The extracted traffic features are stored in another buffer with Δ_2 -second timer, referred to as the chunking buffer. When the timer for the chunking buffer expires, the traffic features stored in the buffer are analyzed by using the traffic analysis model we constructed (Section V-A) and the model is trained by unsupervised learning (Section V-B). The traffic analysis model is created regularly, and all models are optimized to observed traffic features that have Δ_2 -length time series sequences. After training, the model is regularly updated by using the past model (Section V-C). A detailed self-learning workflow is written in Algorithm 1. To avoid waiting time during the model training time, we propose an online estimation function (in Section V-D) that estimates contexts by using the updated model when it observes traffic features in parallel while the model is being trained and updated.

B. Problem formulation

Traffic analysis model training: The traffic analysis model Θ is trained by using only observed traffic patterns O for realizing self-learning estimator. We use an unsupervised learning in which the model is trained for classifying traffic patterns into a class with similar features. This can be formulated as the following equation.

$$\Theta = \arg \max_{\Theta} \Pr[O = \{o_1, \dots, o_T\} | \Theta] \quad (1)$$

Traffic analysis model updates: When new contexts are observed, the model is updated to add contexts and its traffic patterns. On the other hand, when we observe existing context, its traffic patterns in the past model are updated to the most recent patterns.

Online context estimation: The function estimates context after obtaining traffic features at time t by using the updated

Algorithm 1 Self-learning algorithm

```

// Initialize
Sampling buffer  $Q_1 \leftarrow \phi$ 
Chunking buffer  $Q_2 \leftarrow \phi$ 
Traffic analysis model  $\Theta \leftarrow \phi$ 
// Packet capture loop
for  $i = 0, 1, \dots$  do
   $p_i \leftarrow$  observed packet
  Store  $p_i$  into  $Q_1$  by  $Q_1 \leftarrow Q_1 \cup p_i$ 
  if  $\Delta_1$  elapsed then
     $o_t \leftarrow$  traffic features calculated from packets in  $Q_1$ 
    Store  $o_t$  into  $Q_2$  by  $Q_2 \leftarrow Q_2 \cup o_t$ 
    Clear  $Q_1$  by  $Q_1 \leftarrow \phi$ 
    if  $\Delta_2$  elapsed then
      Train  $\Theta_t$  by using  $Q_2$  via unsupervised learning
      Update  $\Theta$  by comparing contexts in  $\Theta_t$  and  $\Theta$ 
      Clear  $Q_2$  by  $Q_2 \leftarrow \phi$ 
    end if
  end if
end if
end for

```

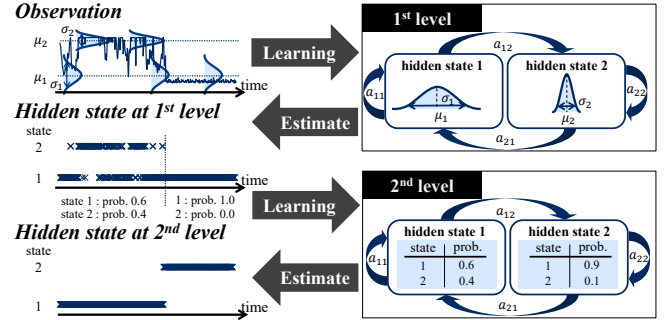


Fig. 2. Training model parameters ($N = 2$).

model. We stochastically estimate context by using the model as the following equation.

$$s_t = \arg \max_{s \in S} \Pr[o_t | \Theta], \quad (2)$$

where s_t is an estimated context at time t , o_t is an observed traffic features at time t , s is a context in a context set S , and Θ is an updated model.

V. HMM BASED TRAFFIC ANALYSIS MODEL

In this section, we first construct a traffic analysis model based on HMM [19]. After that, we discuss detailed algorithms of each function in the proposed context estimation system.

A. ML-HMM: Multi-level probabilistic state transition model

We have found that HMM based speech recognition techniques in Reference [19] are similar to our target. As opposed to speech recognition, traffic pattern changes with not only context but also network quality changes. Because of that, we here enhance HMM for estimating context changes accurately while suppressing noisy patterns.

We construct a multi-level probabilistic state transition model, referred to as ML-HMM, based on the concept that is first to quantize observed continuous values such as traffic volume. After that, we hierarchically categorize quantized time-series traffic patterns into similar patterns as illustrated in Figure 2. The model is defined as Θ whose state transition model at the n -th level is defined as $\theta^n (1 \leq n \leq N)$. N is the deepest level. The observed time-series traffic features are defined as O and the d -dimensional observation at time t is defined as o_t . The hidden state sequence at the n -th level is defined as S^n and hidden state at t is defined as s_t^n . The state transition model at the n -th level is defined as $\theta^n = \{\Pi^n, A^n, B^n, P^n\}$. Let $\pi_i^n \in \Pi^n$ be the set of the initial state probabilities of state i , $a_{ij}^n \in A^n$ be the set of state transition probabilities from state i to j , $b_i^n(s_t^{n-1}) \in B^n$ be the set of probabilities that state i at the n -th level outputs s_t^{n-1} , and $p_i^n(\tau) \in P^n$ be set of the duration probability density of τ consecutive observations in state i . Note that at the first level ($n = 1$), the output probability is set as $b_i^1(o_t)$ and this follows the normal distribution of $N(\mu_i^d, \sigma_i^d)$.

B. Unsupervised training ML-HMM

The goal of model training is to estimate model Θ to solve Eq. (1) and context sequence $S^N = \{s_1^N, \dots, s_{\Delta_2}^N\}$ by analyzing time-series traffic feature $O = \{o_1, \dots, o_{\Delta_2}\}$ stored in the chunking buffer.

At the first level, estimation of model parameters is defined as $\hat{\theta}^1$ and calculated by selecting parameter θ^1 maximizing $\Pr[O|\theta^1]$. Estimation of the hidden state sequence is defined as $\hat{S}^1$ and calculated by selecting hidden state sequence S^1 maximizing $\Pr[O, S^1|\hat{\theta}^1]$. By using estimated sequence $\hat{S}^1$, estimation of the model parameter $\hat{\theta}^n$ and hidden state sequence $\hat{S}^n$ at deeper levels ($1 < n \leq N$) are recursively calculated as follows.

$$\hat{\theta}^n = \arg \max_{\theta^n} \Pr[\hat{S}^{n-1}|\theta^n] \quad (3)$$

$$\hat{S}^n = \arg \max_{S^n} \Pr[\hat{S}^{n-1}, S^n|\hat{\theta}^n] \quad (4)$$

Eq. (3) and (4) are solved using an Expectation-Maximization (EM) algorithm in three steps: (i) evaluating the probability of the traffic features given the model, (ii) estimating the hidden state sequence by using the model, and (iii) updating model parameters to maximize the probability of the traffic features given the model. In previous study in [20], efficient forward and backward algorithms is proposed for training $\theta = \{\Pi, A, B, P\}$ to calculate the probability in step (i). We enhance these algorithms for ML-HMM. The probability of hidden state i at time t at the n -th level in the forward algorithm, $\alpha_t^n(i) = \Pr[s_t^n = i | s_t^{n-1}, \dots, s_1^{n-1}, \theta^n]$, can be calculated by marginalizing $\alpha_t^n(i, \tau)$ defined by

$$\begin{aligned} \alpha_t^n(i, \tau) &= \alpha_{t-1}^n(i, \tau+1) b_i^n(s_t^{n-1}) \\ &+ \left(\sum_{j \neq i} \alpha_{t-1}^n(j, 1) a_{ji}^n \right) b_i^n(s_t^{n-1}) p_i^n(\tau). \end{aligned} \quad (5)$$

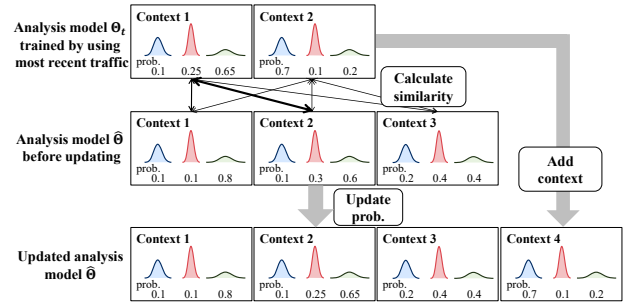


Fig. 3. Update analysis model.

Therefore, the probability of the traffic features given the model, $\Pr[S^n|\theta^n]$, can be calculated by $\sum_i \alpha_{\Delta_2}^n(i)$. In backward algorithm, the probability of i at time t at the n -th level, $\beta_t^n(i) = \Pr[s_{t+1}^{n-1}, \dots, s_{\Delta_2}^{n-1} | s_t^n = i, \theta^n]$, can also be calculated by marginalizing $\beta_t^n(i, \tau)$ defined by

$$\beta_t^n(i, \tau) = b_i^n(s_t^{n-1}) \beta_{t+1}^n(i, \tau-1). \quad (6)$$

Note that, at the first level ($n = 1$), s_t^{n-1} is replaced with o_t in Eq. (5) and (6). In step (ii), the Viterbi algorithm [19] is used to compute Eq. (4) to estimate hidden state sequences $\hat{S}^n$ maximizing $\alpha_t^n(i)$. To update θ^n at each level in step (iii), the parameter update algorithm [20] is used to calculate θ^n maximizing $\Pr[S^n|\theta^n]$. We recursively compute estimations at all levels, $\hat{\theta}^n$ and $\hat{S}^n$, which are calculated by repeating steps (i), (ii), and (iii) until the probability $\Pr[S^n|\theta^n]$ converges.

Finally, we select the number of depth, N , fitting observed traffic features while reducing the number of model parameters. We here use the Bayesian Information Criterion (BIC), which is a likelihood criterion penalized by the model complexity [21] as defined by

$$\text{BIC} = -2\ln(L) + k\ln(|O|), \quad (7)$$

where L is the likelihood of the model, $L = \Pr[S^n|\theta^n]$, k is the number of parameters in deepest levels, and $|O|$ is the length of the observed traffic features. Consequently, we select k minimizing the BIC as N .

C. Updating ML-HMM

We next propose a method to update ML-HMM as shown in Figure 3. Traffic patterns of each context are modeled by several parameters: normal distributions, output probabilities, state transition probabilities, and duration. These parameters of context i in the most recent model and of context j in the past model are summarized as γ_i and ζ_j , respectively. We use cosine similarity as distance between contexts i and j , and the similarity is defined by

$$D(i, j) = \frac{\gamma_i \zeta_j}{|\gamma_i| |\zeta_j|}. \quad (8)$$

The contexts are identified as new contexts when there is no context in the past model with higher similarity than the threshold value δ_{CS} . On the other hand, the contexts are identified as existing contexts in the past model when there is a context with higher similarity than δ_{CS} .

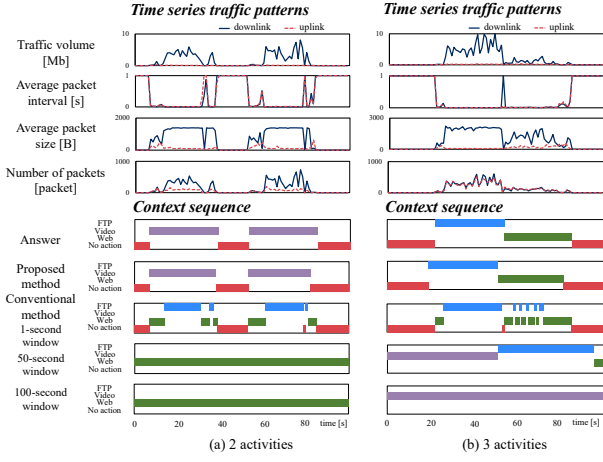


Fig. 4. Results of smartphone activity estimation.

D. Online estimation

The goal of the online estimation is to estimate context when traffic features at time t is given. In ML-HMM, we can use the forward probability at the N -th level, α_t^N , as context probabilities. Therefore, we select context maximizing probability by

$$s(t) = \arg \max_i \alpha_t^N, \quad (9)$$

where i is context index, and α_t^N is calculated by Eq. (5).

VI. PERFORMANCE EVALUATION

In this section, we evaluate our proposed system by taking two anomaly detection use cases in public safety solutions: activity estimation of smartphone users and human movement estimation in network cameras for video surveillance system. The proposed context estimation system is developed by using Java and runs on a traffic monitoring PC with 3.6 GHz CPU and 16 GB RAM connected to a Wi-Fi router.

A. Evaluation in offline estimation

As preliminary experiments, we evaluate the proposed system in terms of estimation accuracy through offline smartphone experiments by comparing the proposed system with SW+RF [13]. The estimation accuracy is calculated by $N_{\text{TRUE}}/|O|$, where N_{TRUE} is the amount of correct estimations and $|O|$ is the length of a traffic sequence. Note that we here use same identifier, i.e., RF, for identifying contexts in the proposed system.

Figure 4 illustrates the results of estimating the user's activities. Estimations in SW+RF are too sensitive to traffic pattern changes in case of the 1-second window. On the other hand, estimations are too robust in case of the 50- and 100-second windows. In more detail, estimation accuracy is quite different for each window size as shown in Figure 5. On the other hand, as shown in Figures 4 and 5, the proposed context estimation system can accurately estimate the activities. The proposed system estimates state transition probabilities before activity identification, so it does not require setting the window

TABLE I
ESTIMATION ACCURACY IN 20 SCENARIOS: 10 SCENARIOS FOR 2 ACTIVITIES AND THE OTHER 10 SCENARIOS FOR 3 ACTIVITIES.

Num. of activities	Statistics	SW+RF			Proposed method
		1 s	50 s	100 s	
2	maximum	0.90	0.67	0.67	0.99
	average	0.74	0.38	0.32	0.91
	minimum	0.34	0.00	0.00	0.75
3	maximum	0.65	0.50	0.37	0.95
	average	0.62	0.30	0.26	0.83
	minimum	0.58	0.00	0.00	0.83

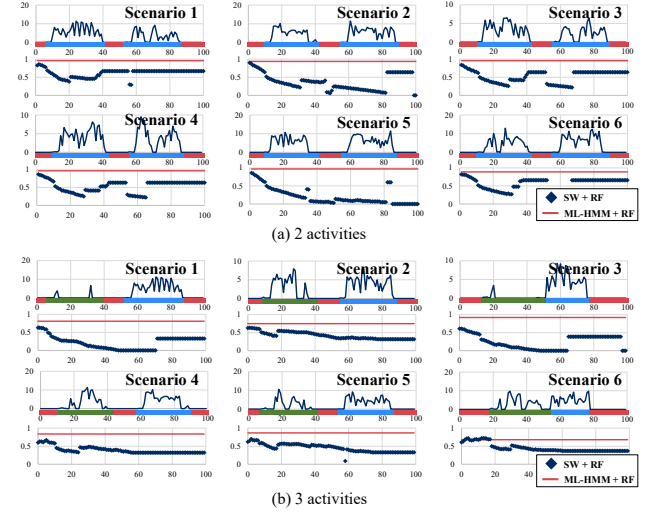


Fig. 5. Relationship between estimation accuracy and window size of sliding window in SW+RF. Top sequence in each scenario is traffic sequence (only traffic volume[Mbps]) and the bottom of the figure shows ground truth context. Red, green, and blue show no action, video watching, and large file downloading, respectively. Bottom figure in each scenario shows estimation accuracy in each window size in SW+RF and accuracy in ML-HMM + RF.

size in contrast to SW+RF. Therefore, as shown in Table I, the proposed system attains 0.91 and 0.83 accuracy on average in the 2- and 3-activity scenarios, respectively.

B. Evaluation in online estimation

The proposed system is evaluated in terms of real-time and self-learning estimation through two online estimation experiments: smartphone activity estimation and network camera state estimation.

1) *Smartphone activity estimation*: Figure 6a and Table II show the results of estimating activity of a smartphone user. The traffic analysis model is initialized from the first Δ_2 -second sequence, which has only two activities: no action and video watching. Therefore, the initial model misestimates unlearned activities until the model update, which results in 0.39 accuracy. After the first model update, the proposed system can estimate activities with 0.76 accuracy because the model has already learned traffic patterns of three activities including web browsing. Similarly, the model can estimate all four activities including file downloading with over 0.81 accuracy after the third update.

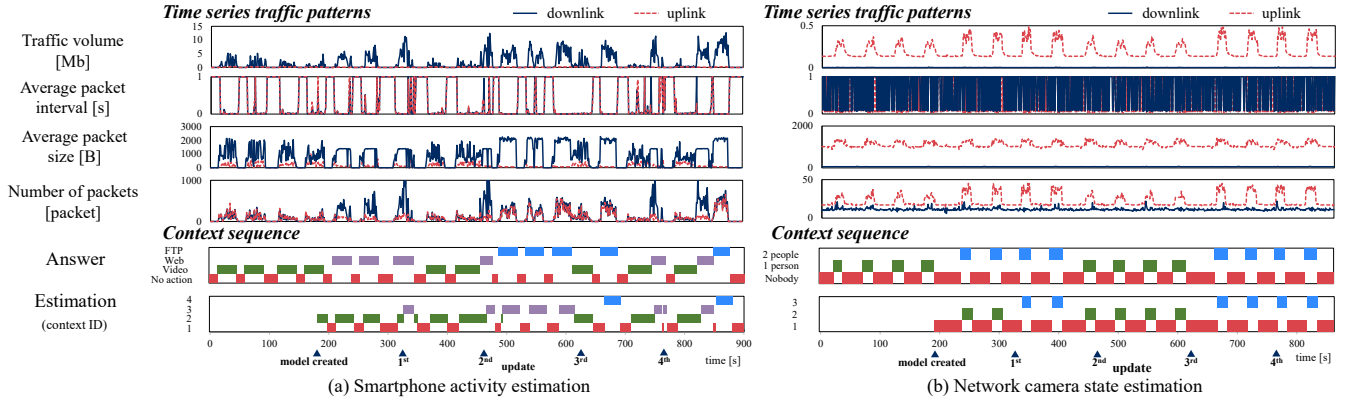


Fig. 6. Results in on-line estimation scenarios. $\Delta_1 = 1$ s. $\Delta_2 = 150$ s. $r = 1$. $\delta_{CS} = 0.2$.

TABLE II
ESTIMATION ACCURACY IN EACH SCENARIO.

	from model created to 1st update	from 1st update to 2nd update	from 2nd update to 3rd update	from 3rd update to 4th update	from 4th update
Smartphone	0.39	0.76	0.27	0.81	0.82
Network camera	0.70	0.89	0.95	0.91	0.88

TABLE III
STATISTICS OF COMPUTING TIME [s].

	Learning	Updating	On-line estimation
average	2.83×10^1	1.43×10^{-5}	3.26×10^{-2}
s.d.	1.47×10^0	1.43×10^{-5}	9.85×10^{-3}
maximum	3.15×10^1	2.85×10^{-5}	7.03×10^{-2}

2) *Network camera state estimation*: Figure 6b and Table II show the results of estimating the state of the network camera. The model initialized from the first Δ_2 -second sequence has only two states: no action and one person moving. Therefore, the initial model misestimates the two-people-moving state as the other state until the model update; therefore, the proposed context estimation system estimates states in 0.70 accuracy. In similar manner with the smartphone experiment, the proposed system finally estimates states over 0.88 accuracy because of self-learning capability by model updates. Through these two online experiments, we have found that the proposed system detects and learns unknown traffic behavior, namely anomaly behavior, by automatically updating ML-HMM.

Finally, Table III shows the computing time in each function: model parameter learning (Section V-B), analysis model updating (Section V-C), and online context estimation (Section V-D). From these results, our system estimates context in 32.6 ms on average from Δ_1 -second ($\Delta_1 = 1$ s) traffic features while learning and updating the analysis model in 28.3 s on average from Δ_2 -second ($\Delta_2 = 150$ s) sequence.

VII. CONCLUSION AND FUTURE WORK

We have proposed *Edge Concierge* targeted at the vertical players utilizing private mobile networking enabling *unattended and self-evolve-able, yet inexpensive* network opera-

tions, which is based on key concept we proposed that assists smart network operations and energy saving by changing working state of AI-empowered anomaly detection applications via traffic analysis by *network layer AI*.

Our contributions are three-fold. First, we define three technical challenges regarding context estimation by *Edge Concierge*, namely real-time, self-learning, and inexpensive. Second, to address the challenges above, we propose a real-time and self-learning context estimation system to estimate application contexts in real-time and online without labelling and supervising. To the best of our knowledge, our proposed system is the first to utilize a *multi-level probabilistic state transition model* for estimating application contexts for the precise traffic control for even emerging applications with unknown behaviors. In addition, we show our proposed system can be implemented in commercial off-the-shelf (COTS) components such as PCs so that verticals without special hardware may easily use *Edge Concierge*. Finally, our performance evaluations show that *Edge Concierge* can accurately identify as many as four contexts in given use cases, e.g., smartphones and network cameras, by *unsupervised learning* even if the contexts are not known a priori.

Although we plan additional discussions such as further evaluations of the increased number of contexts, the estimation performance under the actual environment, and the operational cost reduction by taking specific network operations for near future work, we strongly believe that we have paved the way for enabling unattended and inexpensive network operations by our initial proposal of *Edge Concierge* through unsupervised learning with state transition probability using Hidden Markov Model (HMM).

REFERENCES

- [1] A. Callado, C. Kamienski, G. Szabo, B. P. Gero, J. Kelner, S. Fernandes, and D. Sadok, "A survey on internet traffic identification," *IEEE Communications Surveys Tutorials*, vol. 11, no. 3, pp. 37–52, 2009.
- [2] N. Nandev, S. Agrawal, and S. Silkari, "Recent advancement in machine learning based internet traffic classification," *Procedia Computer Science*, vol. 60, pp. 784–791, 2015.
- [3] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.
- [4] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: Myths, caveats, and the best practices," in *Proc. of the ACM CoNEXT*, 2008, pp. 11:1–11:12.
- [5] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 5–16, 2006.
- [6] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1, pp. 50–60, Jun. 2005.
- [7] T. Auld, A. W. Moore, and S. F. Gull, "Bayesian neural networks for internet traffic classification," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 223–239, Jan. 2007.
- [8] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proc. of the SIGCOMM Workshop on Mining Network Data*, 2006, pp. 281–286.
- [9] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "Blinc: Multilevel traffic classification in the dark," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 229–240, Aug. 2005.
- [10] A. Nakao and P. Du, "Toward in-network deep machine learning for identifying mobile applications and enabling application specific network slicing," *IEICE Transactions on Communications*, vol. E101.B, no. 7, pp. 1536–1543, 2018.
- [11] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 2, pp. 23–26, Apr. 2006.
- [12] T. Iwai and A. Nakao, "Adaptive mobile application identification through in-network machine learning," in *Proc. of the APNOMS*, 2016.
- [13] T. T. T. Nguyen and G. Armitage, "Training on multiple sub-flows to optimise the use of machine learning classifiers in real-world ip networks," in *Proc. of the 31st IEEE Conference on Local Computer Networks*, Nov 2006, pp. 369–376.
- [14] F. Zhang, W. He, X. Liu, and P. G. Bridges, "Inferring users' online activities through traffic analysis," in *Proc. of the Fourth ACM Conference on Wireless Network Security*, 2011, pp. 59–70.
- [15] J. Liu, Y. Fu, J. Ming, Y. Ren, L. Sun, and H. Xiong, "Effective and real-time in-app activity analysis in encrypted internet traffic streams," in *Proc. of the 23rd ACM SIGKDD*, 2017, pp. 335–344.
- [16] S. Nawata, H. Koto, T. Kitahara, and S. Ano, "Analysis and practicality of traffic flow classification method using binned time-series data," in *Proc. of the APSITT*, 2015, pp. 1–3.
- [17] A. Sawabe, T. Iwai, and K. Satoda, "Identification of smartphone applications by encrypted traffic analysis," in *Proc. of the IEEE CCNC*, 2019, pp. 1–2.
- [18] M. Usama, J. Qadir, A. Raza, H. Arif, K. A. Yau, Y. Elkhatib, A. Hussain, and A. Al-Fuqaha, "Unsupervised machine learning for networking: Techniques, applications and research challenges," *IEEE Access*, vol. 7, pp. 65 579–65 615, 2019.
- [19] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *In Proc. of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
- [20] S. Z. Yu and H. Kobayashi, "An efficient forward-backward algorithm for an explicit-duration hidden markov model," *IEEE Signal Processing Letters*, vol. 10, no. 1, pp. 11–14, 2003.
- [21] S. S. Chen and P. S. Gopalakrishnam, "Speaker, environment and channel change detection and clustering via the bayesian information criterion," in *Proc. of the DARPA Broadcast News Transcription and Understanding Workshop*, 1998, pp. 127–132.