Bits Learning: User-adjustable Privacy versus Accuracy in Internet Traffic Classification

Zhenlong Yuan, Jie Xu, Yibo Xue, and Mihaela van der Schaar

***Abstract*—During the past decade, a great number of machine learning (ML) based methods have been studied for accurate traffic classification. Flow features such as the discretizations of *the first five packet sizes* (*PS*) and *flow ports* (*FP*) are considered the best discriminators for per-flow classification. For the first time, this letter proposes to treat *the first n-bits of a flow* (*BitFlow*) as features and compares its overall performance with the well-known *ACAS* (*Automated Construction of Application Signatures*) that takes *the first n-bytes of a flow* (*ByteFlow*) as features. The results show that *BitFlow* achieves not only a higher classification accuracy but also 1∼3 orders of magnitude faster speed than *ACAS* in training and classifying. More importantly, this letter also proposes to treat *the first n-bits of each of the first few packet payloads* (*BitPack*) as features, which enables a user-adjustable trade-off between user privacy protection and classification accuracy maximization. The experiments show that *BitPack* can significantly outperform *BitFlow*, *PS* and *FP*.**

***Index Terms*—Traffic classification, ML, bits as features.**

I. Introduction

TRAFFIC classification is fundamental to a broad range of network operations and management, e.g. quality-of-service (QoS) control and intrusion detection [11], [12]. Due to the prevalence of asymmetric routing, traffic classification is required to support per-flow identification (i.e. deal with unidirectional flows). Early work in ML-based traffic classification by Moore and Zuev [10] proposed to apply the supervised *Naïve Bayes* techniques using discriminators derived from packet headers for categorizing traffic. Afterwards, a variety of ML based methods using various sets of characteristics were investigated to get a higher classification accuracy. The discretizations of *PS* and *FP* are considered by far the best discriminators for ML-based traffic classification, regardless of what ML algorithms are used [9]. Besides that, *the first n-bytes of a flow* (*ByteFlow*) were also used as features to Automatically Construct Application Signatures (*ACAS*) [7]. Compared with payload-based approaches (e.g. [7]), statistics-based approaches (e.g. [4]) can deal with encrypted traffic and do not have to entail the privacy or legal concerns associated with traffic classification. However, payload-based approaches are still playing the most important role in today's traffic classification since they are considered more reliable [5].

Previous research [9], [13] found that, in traffic classification problems, different ML algorithms using the same set of features can achieve similar classification accuracy. Therefore, feature selection has been considered far more important than the selection of ML algorithms in improving the classification accuracy. In addition, due to the requirements of real-world deployment, computational performance (i.e. training time and classification speed) has become another important consideration for ML-based traffic classification. For the first time, this letter proposes two novel discriminators that use bits of payload as features for traffic classification, which improve both classification accuracy and computational performance. The contributions of this letter are as follows.

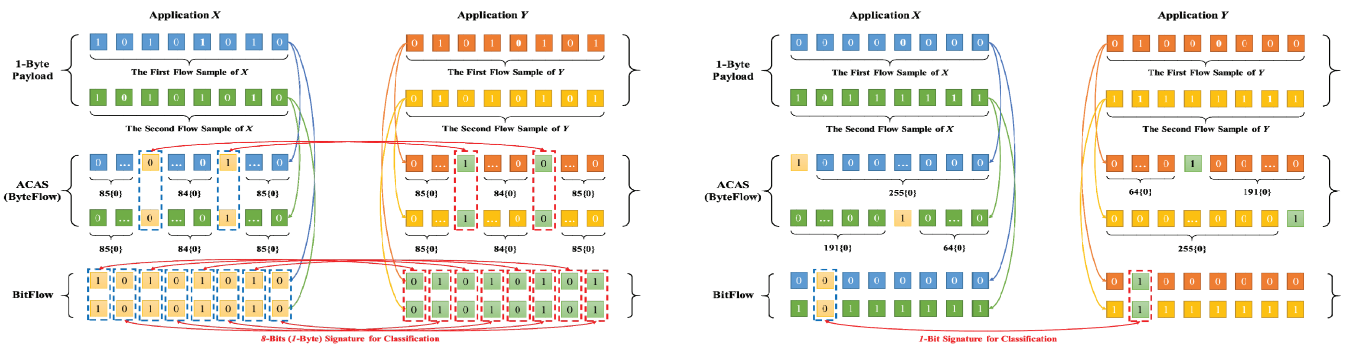
Firstly, this letter proposes to treat *the first n-bits of a flow* (*BitFlow*) as features for ML-based traffic classification. The experiments show that this leads to a significant improvement by using the bit-level information (i.e. *BitFlow*) instead of the byte-level information (i.e. *ByteFlow*) when constructing the features, which not only greatly improves the classification accuracy but also performs 1∼3 orders of magnitude faster than *ACAS* in training models and classifying traffic.

Secondly, this letter proposes an enhancement to *BitFlow*, namely *BitPack*, that uses *the first n-bits of each of the first few packet payloads* as features for ML-based traffic classification. The experiments show that *BitPack* significantly outperforms *PS* and *FP* [9] in terms of classification accuracy. The results also show that the classification accuracy increases with the amount of payload bits used, which indicates that *BitPack* enables a user-adjustable trade-off between user privacy protection and classification accuracy maximization.

Thirdly, this letter proposes to combine *BitPack* with *PS* and *FP* for ML-based traffic classification. This is different from conventional methods that use either statistics derived from packet headers or payload-based signatures. The experiments show that such a combination can achieve nearly 100% classification accuracy on both TCP and UDP traces. This suggests that combining *BitPack* with *PS* and *FP* can be a general solution for accurate traffic classification.

II. BitFlow and BitPack

BitFlow and *BitPack* both encode the raw application level data of a unidirectional flow as a feature vector, which take each bit as a feature for ML-based traffic classification. As illustrated in Fig.2, *BitFlow* takes every bit of *the first n-bits of a flow* as a feature where n is a tunable parameter. Note that using *BitFlow* as features for ML algorithms is essentially a way to automatically construct bit-level signatures for applications. This is similar to *ACAS* which uses *ByteFlow* as features for automatically constructing byte-level signatures [7]. As an enhancement to *BitFlow*, *BitPack* takes every bit of *the first n-bits of each of the first few packet payloads* as a feature where n is a tunable parameter as well. Moreover, if any bit of the first n-bits of *BitPack* does not exist, it will be replaced with '-1' in the feature vector. Consequently, each feature has three possible vaules (i.e. '0', '1' and '-1'). Since


Application X
Application Y
1-Byte Payload
The First Flow Sample of X
The Second Flow Sample of X
The First Flow Sample of Y
The Second Flow Sample of Y
ACAS (ByteFlow)
BitFlow
8-Bits (1-Byte) Signature for Classification
1-Bit Signature for Classification

(a) Example 1: *BitFlow* has the same effect as *ACAS*. (b) Example 2: *BitFlow* and *ACAS* have the different effects.

Fig. 1: Features Comparisons: *BitFlow* vs. *ACAS*

BitPack only uses the first *n-bits* of each of the first few packet payloads without accessing the whole payload content, it is particularly suitable for dealing with packet payloads that have been truncated to the first few bytes (bits) due to the privacy and legal concerns from network operators or governments. More importantly, *BitPack* only uses the first few packets of a flow and hence, it enables early identification of traffic, which is crucial for timely network operations and management.

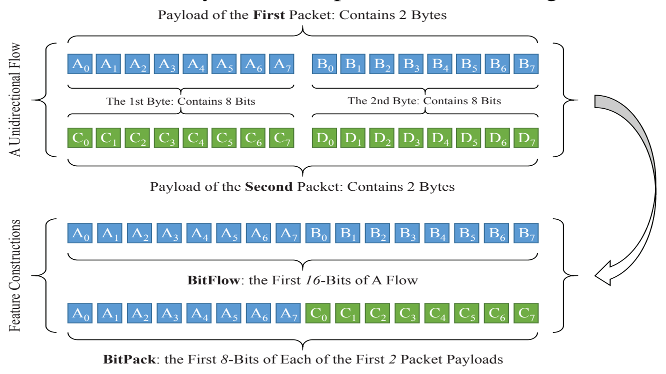

Payload of the First Packet: Contains 2 Bytes
A Unidirectional Flow
The 1st Byte: Contains 8 Bits
The 2nd Byte: Contains 8 Bits
Payload of the Second Packet: Contains 2 Bytes
Feature Constructions
BitFlow: the First 16-Bits of A Flow
BitPack: the First 8-Bits of Each of the First 2 Packet Payloads

Fig. 2: Feature Constructions: *BitFlow* and *BitPack*

The rationale of using *bits* instead of *bytes* of the payload as features is that *bits* are more fine-grained than *bytes* in application traffic and may contain signatures that cannot be detected by methods using byte-level features. As a key observation, the data-transfer formats of more and more protocols and applications have been encoded at the bit-level granularity instead of the byte-level granularity (e.g. DNS operates at the bit-level granularity), and as a consequence, there might be only bit-level signatures existing in their traffic without any byte-level signatures. As illustrated in Fig.1, this letter presents two simple examples of constructing features for *BitFlow* and *ACAS*, respectively. As described in [7], *ACAS* uses the first *n* (64 or 256) bytes of a flow as features and encodes them into a feature vector *v* with *n*∗256 elements (all components of *v* are initialized to 0, then for each byte of payload, the component *i*∗256 + c[*i*] is set as 1, where *i* represents the position of a byte with value c[*i*] in a flow). Different from *ACAS*, *BitFlow* treats a bit as a feature and thus, the same first *n* bytes (*n*∗8 bits) can be encoded into a feature vector with only *n*∗8 elements, which is 1/32 of *ACAS*. Fig.2 (a) shows that when two applications have one distinct **byte** value (i.e. "10101010" and "01010101") in their traffic samples, both *ACAS* and *BitFlow* can properly construct the *1*-byte (*8*-bits) signature for distinguishing their traffic. However, as shown in Fig.2 (b), when two applications only have one distinct **bit** value in their traffic samples (i.e. the value of the second bit-position is '0' or '1'), only *BitFlow* can properly differentiate their traffic by constructing the *1*-bit signature because the other 7 bits with randomly distributed values make the byte-level methods (e.g. ACAS) impossible to construct the 1-byte signature. It is evident that constructing features at the bit-level (*BitFlow*) should perform equally as or better (cannot be worse) than doing that at the byte-level (*ACAS*).

III. Evaluation

A. Experimental Setup

Both publicly available and privately captured traffic traces are used to evaluate the classification performance with various features. The statistics of the datasets are summarized in Table I. The *TCP Dataset* contains traffic traces from the publicly available WITS Project [3]. In this dataset, packet payloads following the transport header have been truncated to 4 bytes due to the privacy concerns. Six TCP applications (i.e. *FTP-Control*, *SSH*, *SMTP*, *HTTP*, *Pop3* and *HTTPS*) are extracted based on their well-known ports as well as in [4], [6], [7],

TABLE I: Two Datasets for Performance Evaluation

Dataset	Application	Port	Training Set (# of Flows)			Test Set (# of Flows)			Payload
			Client→Server	Server→Client	Total #	Client→Server	Server→Client	Total #	
TCP Dataset	FTP Control	21	119	210	329	119	210	329	4 Bytes
	SSH	22	952	778	1,730	952	778	1,730	4 Bytes
	SMTP	25	422	441	863	422	441	863	4 Bytes
	HTTP	80	197	4,962	5,159	197	4,962	5,159	4 Bytes
	Pop3	110	960	2,536	3,496	960	2,536	3,496	4 Bytes
	HTTPS	443	543	1,147	1,690	543	1,147	1,690	4 Bytes
UDP Dataset	Skype	–	280	220	500	280	220	500	Full
	Xunlei	–	302	198	500	302	198	500	Full
	PPTV	–	250	250	500	250	250	500	Full
	RTMFP	–	250	250	500	250	250	500	Full
	eMule	–	250	250	500	250	250	500	Full
	QQDownload	–	250	250	500	250	250	500	Full

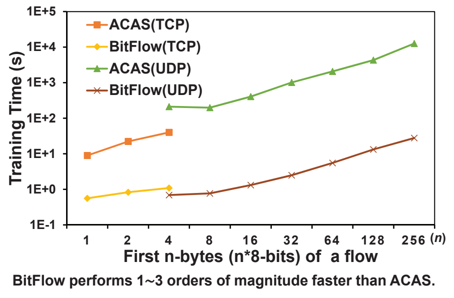


Fig. 3: Training Time Comparisons

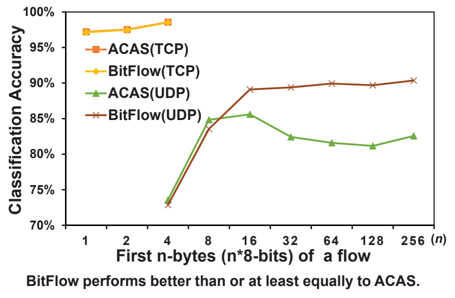


Fig. 4: Accuracy Comparisons

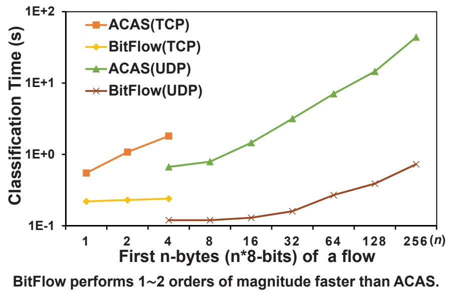


Fig. 5: Classification Time Comparisons

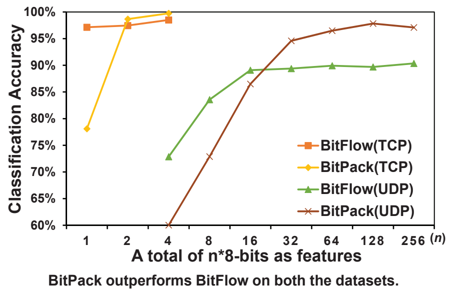


Fig. 6: Accuracy Comparisons

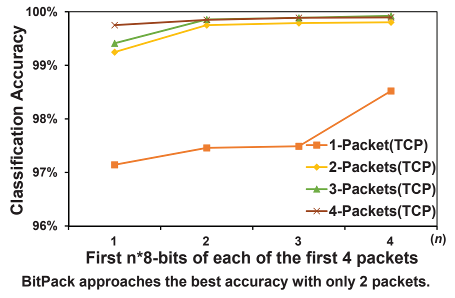


Fig. 7: *BitPack* on *TCP Dataset*

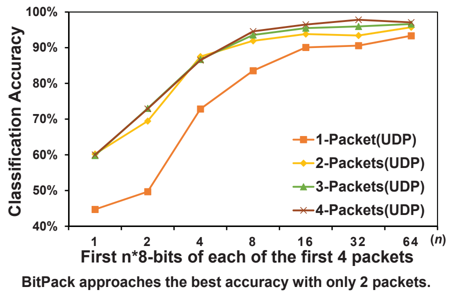


Fig. 8: *BitPack* on *UDP Dataset*

[13] since these applications are still mainly using their default port numbers. Since only very limited payload data in public datasets can be accessed, the letter also builds a *UDP Dataset* that contains privately captured traffic traces. These traces are from 6 popular P2P applications (i.e. *Xunlei*(*Thunder*), *Skype*, *PPTV*(*PPLive*), *RTMFP*, *eMule* and *QQDownload*) with full packet payloads (i.e. the whole payload content of each packet is preserved and not truncated). Their traffic samples are captured and manually labeled in three different network scenarios (including two university campuses and a networking company) located in the US and China. We used a number of machines for generating the real-world application traffic. For both datasets, the letter uses half of the unidirectional flows of each application for training and the other half for testing. The experiments are repeated 10 times by randomly shuffling the datasets. The average results of these experiments are reported in the subsequent sections. This letter uses the WEKA [2] machine learning software suite, which is widely adopted in existing works [6], [8], [9], [10], [13], [14] on traffic classification. To make the comparison fair, this letter uses the AdaBoost algorithm (with C4.5 being the base classifier [1]) as in ACAS for all the experiments. Once a training set and a test set in ARFF format are imported into the Weka tool, we select Adaboost (with C4.5 being the base classifier) as the classification algorithm and use all the other default configuration parameters for the classification tasks.

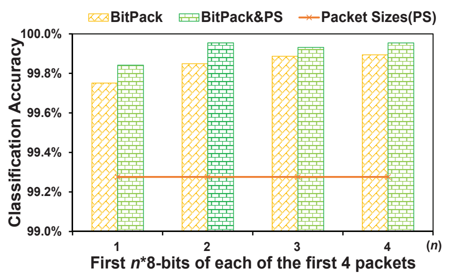
B. BitFlow vs. ACAS

This letter first compares the overall performance of *ACAS* and *BitFlow*. Fig.3 shows the training time of *ACAS* and *BitFlow* on the *TCP Dataset* and *UDP Dataset*, respectively. Note that the *TCP Dataset* has been truncated to only 4 bytes in each packet payload. Therefore, at most the first 4 bytes (32 bits) of a flow can be used to construct features. We can see that *BitFlow* is 1∼3 orders of magnitude faster than *ACAS* in training the ML models. This significant improvement is a result of the reduced size of the feature vectors of *BitFlow*, which is *1/32* of *ACAS*. Fig.4 shows the classification accuracy of *ACAS* and *BitFlow*, although the benefit of using *BitFlow* is not obvious on the *TCP Dataset*, *BitFlow* outperforms *ACAS* on the *UDP Dataset*. The reason for the tie on the *TCP Dataset* is that traditional TCP applications can be easily distinguished by byte-level signatures without the need for more fine-grained bit-level signatures. However, applications in the *UDP Dataset* are more sophisticated and often operate at the finest bit-level granularity. Therefore, only bit-level signatures exist in the traffic of these applications. Interestingly, while the accuracy of *BitFlow* always increases with more payload data, the classification accuracy of *ACAS* drops when more payload data is used. The latter is consistent with the observation in [7]. In addition, Fig.5 shows the classification time of *BitFlow* and *ACAS*. As can be seen, *BitFlow* performs 1∼2 orders of magnitude faster than *ACAS* in classifying the traffic, which is crucial for a real-world traffic classification system.

C. BitPack vs. BitFlow

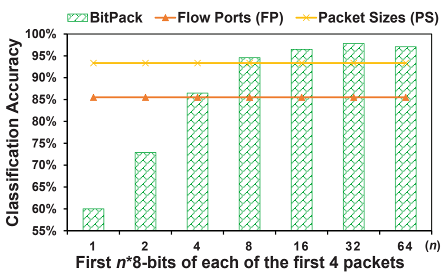
In Fig.6, this letter compares the classification accuracy of *BitPack* and *BitFlow* by using a total of *n**8-bits as features. For *BitPack*, this letter uses the first 4 packets and hence, for each packet, *BitPack* uses the first 2*n-bits so that the total number of bits used is *n**8. As we can see, *BitFlow* performs better than *BitPack* when the amount of payload data used is small. However, when accessing more payload data, *BitPack* performs much better than *BitFlow* on both the *TCP Dataset* and *UDP Dataset*. In particular, on the *UDP Dataset*, the best classification accuracy of *BitPack* reaches 97.8% while *BitFlow* can only achieve 90.4%. Note that *BitPack* has realized an improvement of 12.2% in classification accuracy compared with *ACAS* which was shown in Fig.4.

In addition, this letter shows the impact of using different numbers of packets on the classification accuracy for *BitPack*. Fig.7 and Fig.8 show the results by using the first



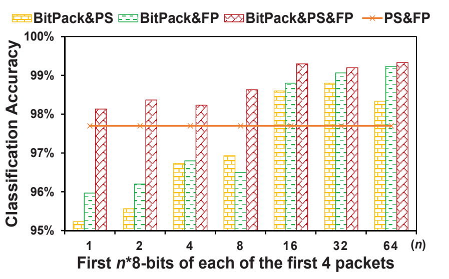
BitPack outperform PS and their combination performs the best.

Fig. 9: Comparisons on *TCP Dataset*



BitPack outperforms PS and FP while using more than 64 bits.

Fig. 10: Comparisons on *UDP Dataset*



The combination of BitPack, PS and FP always performs the best.

Fig. 11: Comparisons of Combinations

one to four packets for the *TCP Dataset* and *UDP Dataset*, respectively. As can be seen, *BitPack* can already approach the best accuracy with only two packets. Moreover, *BitPack* performs better with more bits of each packet payload. This suggests that a higher classification accuracy can be achieved by *BitPack* by using more payload data. Thus, *BitPack* enables a user-adjustable trade-off between user privacy protection and accuracy maximization by tuning the amount of payload data accessed. It is worth nothing that although the *TCP Dataset* has been truncated to 4 bytes in each packet payload, *BitPack* can still achieve 99.9% accuracy by using the few packets.

D. BitPack vs. PS and FP

The discretizations of *PS* and *FP* have been considered by far the best discriminators for ML-based traffic classification, regardless of what ML algorithms are used [9]. Therefore in this subsection, this letter compares the classification accuracy of *BitPack* (uses the first four packets) and *PS* (uses the first five packets)/*FP* on the *TCP Dataset* and *UDP Dataset*, as shown in Fig.9 and Fig.10. Since the *TCP Dataset* is built through filtering application traffic based on their default ports, the letter only performs *FP* on the *UDP Dataset*. From Fig.9, we can see that *BitPack* outperforms *PS*, even with only the first *8* bits. Moreover, as shown in Fig.10, *BitPack* performs much better than *PS* and *FP* by using more than *64* bits of the payloads on the more difficult *UDP Dataset*. This improvement is up to 4.5% against *PS* and 12.3% against *FP*.

Different from existing methods using either payload-based signatures or header-derived characteristics for accurate traffic classification, this letter proposes to integrate them together for ML-based traffic classification. This letter first combines the features (e.g. BitPack, PS and FP) to construct a new feature set and then uses the new feature set for the ML algorithm. From Fig.9, we can see that *BitPack* and *PS* combined can perform better than just using any single discriminator, and achieves almost 100% classification accuracy. As shown in Fig.11, this letter evaluates the classification accuracy of all the possible combinations of *BitPack*, *PS* and *FP* on the *UDP Dataset*. We can see that the combinations of *BitPack* and *PS* or *FP* can outperform the well-known combinations of *PS* and *FP* [9] by using more than 16 bytes (128 bits). Moreover, the combination of the three discriminators (i.e. *BitPack*, *PS* and *FP*) always performs the best among all the combinations, which achieves 99.33% classification accuracy. More importantly, as *BitPack*, *PS* and *FP* all use at most the first five packets of a flow, their combination can be a general solution for accurate and timely traffic classification.

IV. Conclusion

For the first time, this letter proposes to use the bits of payload as features for ML-based traffic classification. In particular, this letter designs two novel discriminators, i.e. *BitFlow* and *BitPack*, which are essentially two ways for automatically constructing application signatures at the finest bit-level granularity. The experiments show that *BitFlow* significantly outperforms *ACAS* in terms of both classification accuracy and training and classification speed. Moreover, as an enhancement to *BitFlow*, *BitPack* enables a user-adjustable trade-off between user privacy protection and classification accuracy maximization. The experiments show that *BitPack* performs better than the two well-known discriminators, i.e. *PS* and *FP*. Furthermore, the letter also shows that the combination of *BitPack* with existing discriminators (i.e. *PS* and *FP*) can further improve the classification accuracy, which can be a general solution for accurate and timely traffic classification.

References

[1] Adaboost algorithm with c4.5 being the base classifier. http://weka.sourceforge.net/doc.dev/weka/classifiers/meta/AdaBoostM1.html.

[2] Weka 3: Data mining software in java. http://www.cs.waikato.ac.nz/ml/weka/.

[3] Wits: Waikato internet traffic storage. http://wand.net.nz/wits/waikato/8/waikato_viii.php.

[4] R. Alshammari and A. N. Zincir-Heywood. Can encrypted traffic be identified without port numbers, ip addresses and payload inspection? *Computer Networks*, 55(6):1326–1350, 2011.

[5] T. Bujlow, V. Carela-Español, and P. Barlet-Ros. Independent comparison of popular dpi tools for traffic classification. *Computer Networks*, 76:75–89, 2015.

[6] J. Erman, M. Arlitt, and A. Mahanti. Traffic classification using clustering algorithms. In *ACM SIGCOMM MineNet Workshop*, 2006.

[7] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. Acas: automated construction of application signatures. In *ACM SIGCOMM MineNet Workshop*, 2005.

[8] H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee. Internet traffic classification demystified: myths, caveats, and the best practices. In *ACM CoNEXT*, 2008.

[9] Y.-s. Lim, H.-c. Kim, J. Jeong, C.-k. Kim, T. T. Kwon, and Y. Choi. Internet traffic classification demystified: on the sources of the discriminative power. In *ACM CoNEXT*, 2010.

[10] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *ACM SIGMETRICS*, 2005.

[11] B. Park, J. W.-K. Hong, and Y. J. Won. Toward fine-grained traffic classification. *IEEE Communications Magazine*, 49(7):104–111, 2011.

[12] P. Reviriego, S. Pontarelli, and J. A. Maestro. Energy efficient exact matching for flow identification with cuckoo affinity hashing. *IEEE Communications Letters*, 18(5):885–888, 2014.

[13] N. Williams, S. Zander, and G. Armitage. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. *ACM SIGCOMM CCR*, 2006.

[14] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, and Y. Guan. Network traffic classification using correlation information. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 24(1):104–117, 2013.