Network Traffic Classification in Encrypted Environment: A Case Study of Google Hangout

Jayeeta Datta, Neha Kataria, Neminath Hubballi
Department of Computer Science and Engineering
Indian Institute of Technology Indore
{cs1000129, cs1100123, neminath}@iiti.ac.in

***Abstract*—Traffic classification is an important task for providing differentiated service quality to applications and also for security monitoring. With the advent of peer-to-peer applications and tunneling techniques it is becoming increasingly difficult to identify the traffic without going to the application semantics. Several approaches have been proposed (with varied success) which use machine learning techniques to identify the application traffic. In this paper we propose a novel technique based on application behavior based feature extraction and classification. We experiment with Google Hangout as a case study and report its detection results. Google Hangout is a semi peer-to-peer application allowing two parties to do video chat online. We performed experiments with a dataset consisting of several hours of network traffic consisting of 2.5 million packets and report results on 3 classification algorithms namely Naive Base, decision tree and AdaBoost. We conducted 3 sets of experiments with different combinations of data and performed 10 fold cross validation in each case to assess the classification performance.**

I. Introduction

Application identification is at the center of many security operations. An organization can have security policies to permit certain type of application and block others. This could be solely driven by security or can be of Quality of Service (QoS) issue. In QoS certain types of applications traffic may get preferential treatment compared to others. For example in a office environment email traffic can be given preference over video download. Often the very nature of traffic demands such preference be given; for example audio traffic is sensitive to delay and video content is sensitive to jitter, etc.

Historically such application traffic was identified using port numbers [1], [2] as most applications used standard port numbers defined by IANA. In that scenario it was just sufficient to either allow, block or apply QoS on a packet based on port number. With the emergence of peer to peer applications like eDonkey [3], emule [4] and Internet traffic anonymizers like UltraSurf [5], Tor [6] and tunnelling of traffic under non conventional port numbers this method is no longer effective [7]. To overcome this Deep Packet Inspection (DPI) was introduced which uses application protocol content and semantics to detect type of application. Unfortunately this requires traffic in plain text format and currently majority of applications communicating over internet uses encryption. Thus deep packet inspection is not possible [8] by a third party sitting in between unless the connection is completely intercepted and deciphered. Further given the growing network bandwidth of today's networks, DPI is expensive to perform [9], [10], [11]. Further there is a lack of publicly available datasets and tools for performance comparison. Recently research community has started working towards having an open platform [12] for traffic classification.

978-1-4799-6619-6/15/$31.00

There are several commercial grade tools performing traffic classification based on different techniques. For example Sandvine [13] uses the site certificate exchanges in the beginning and classify the subsequent communications to the IP address of certificate.

Port based and payload based classification techniques does not work well in encrypted environment [9]. There are several attempts to use machine learning based classification methods [14], [15], [16], [17] to identify encrypted traffic. PacketShaper [18] is another application QoS instrument available publicly.

Rest of this paper is organized as follows. In section II, works on traffic classification are described. In section III, Google Hangout communication behavior is explained. In the next section IV set of features used for classification and motivation for selecting those features are elaborated. Experiments done and performance of proposed method is described in section V and finally paper is concluded in section VI.

II. Related Work

Traffic classification techniques found in literature falls into signature based, statistical methods and flow based methods. Following 3 subsections describe each of these categories.

A. Signature Matching

As the name suggests these methods use signatures written on application protocol behavior and invariant content or pattern. Some of the commercial Unified Thread Mitigation appliances like Palo Alto Networks [19] uses decryption of encrypted traffic and signatures to identify Gtalk and other applications. Taking motivations from the fact that signature based deep packet analysis is time consuming and error prone; in [20] an automated signature extraction method which is capable of discovering new applications is proposed. Signatures are based on packet payload content and are generated with invariant parts of payloads in different flows. Further these signatures can be encoded as text-based or binary data.

B. Statistical Methods

These methods uses statistics derived out of packets [21] (like inter arrival timings of packets, average packet size) and traffic distribution of applications as features to identify application.

Philip Branch [22] uses inter arrival timing of packets and packet length as features to train a machine learning algorithm and identify traffic class. The choice of inter-arrival time serves as a parameter because real time applications have stringent timing requirements and for the same reason even packet length is also an useful feature in detecting such applications.

C. Flow Classification

Flow based methods use characteristics of flows like duration of flow, average number of packets, average packets size, etc [23], [24]. to identify applications. In [25] multilevel flow characteristics are used to identify flows belonging to different classes.

A hybrid classifier including K-nearest neighbor and k-means clustering algorithms are used to classify flows [15] using 17 attributes. In a similar approach [26] flows are classified using a machine learning algorithm with features extracted from the packet headers. One of the limitation of this work is it uses a public dataset which is anonymized with an assumption that port numbers in packet headers correctly identify applications which is not true in case of peer to peer applications.

Dorfinger et. al [27] used entropy of the first packet in a flow to classify a flow as encrypted or not. Here the first packet means first packet with payload excluding the TCP 3-way handshake packets. This work does not identify the applications but only whether the flow is encrypted or non encrypted.

Alshammari and Zincir-Heywood [16] also used NetMate to derive statistics from flows and used machine learning algorithms namely Naive Base, SVM and C4.5 and Ripper to classify SSH and Skype traffic with a reported accuracy ranging between 84% to 97%.

Zhang et. al [17] use correlation information among flows to classify network traffic into different applications. The correlation is defined over bag of flows. BoF is formed using a 3 tuple heuristics comprising of destination IP address, destination port and protocol. A probabilistic model created to use the correlation information among BoF is used as a feature for Nearest Neighbor Classifier. Experiments on two large real network traces yielded accuracy ranging from 60% to 85% for different applications.

III. Google Hangout Behaviour

Google Hangout is an application allowing users to text chat, perform voice over IP communication and video chat. Google Hangout is not a pure peer-to-peer application but possess characteristics of a peer-to-peer communications as it allows two clients to communicate in real-time through a conference server chosen dynamically. For the same reason it can be called as a semi peer-to-peer application. The sequence of operations in a Hangout communication is shown in Figure 1. Typically a user supplies her account credentials and these credentials are verified by Google account server which is a common account server for all Google services. After the account credentials are verified, Hangout client connects to one of the several connection[1] servers. The client application uses TCP packets to connect and exchange data with connection server. To our understanding the details of peer users who are online and their status is periodically updated through this TCP connection. When the user selects one or more peer users and starts a video chat; a conference server is identified and connected. To carry video content generated during a video chatting session Google Hangout prefers UDP as a transport layer protocol.

TCP packets uses TSL protocol to encrypt the packets and hence are sent to destination port 443 of connection server. The number of TCP connections to connection server will be more than 1 (ranging from 1-14 in our experiments) all of them using a different source port number. In most of the cases data is exchanged using only one of the TCP connections. Remaining connections will be either closed after a while or used occasionally.

A. DNS Name Resolution

Before a Hangout client is connected and is online there are sequence of DNS queries made. The sequence of DNS name resolutions is shown in Figure 2. The first DNS query is to the *accounts.Google.com* name server to which user credentials are supplied. Once the credentials are verified the next DNS query is to one of the client servers. These client servers are named *client*.goole.com* where * takes numeric value. In our experiments we saw its value being 4, 5 and 6 in most cases. The reply to this DNS query is an IP addresses indicating the connection server to which client can connect and acquire a list of peer users who are online and offline. When the user initiate a video chat, another DNS query stun.l.goole.com is sent to get the list of available conference servers. This name resolution always yielded a list of 12 IP addresses in our experiments. Our current understanding is these conference servers are chosen based on location of two clients engaging in online communication.

B. Data Exchange

Google Hangout uses STUN packets at regular intervals to maintain the connectivity between client and conference server. Google Hangout uses Session Initiation Protocol to maintain the session between client and conference server. As in most of the cases client will be behind a NAT can not directly interact with the server. STUN is protocol designed for a NATed client to query the connection between itself and server. STUN packets are exchanged with the conference server to query whether client can connect to conference server. There are different types of STUN packets and these are sent at particular events. Video data in Hangout is transmitted

[1]The name connection server is used here for the lack of a better name

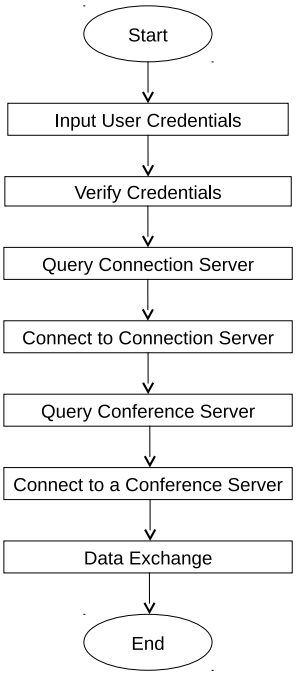

Fig. 1. Hangout Operation Sequence

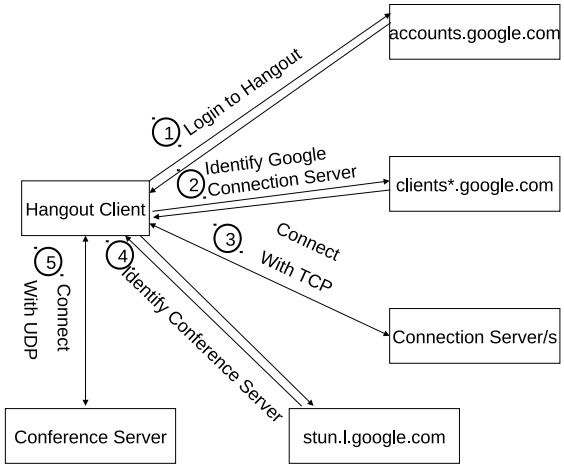

Fig. 2. DNS Name Resolution Sequence in Google Hangout

using UDP as a prefereed transport layer protocol however it can use TCP too. UDP packets are sent to the same server as STUN packets with different source port numbers (3-4) and share a common destination port with STUN. These UDP source port numbers alternate at irregular intervals.

IV. Feature Extraction and Classification

A. Feature Extraction

Our method of identification is per packet based identification rather than flow based or session based found in literature. We use a hybrid approach of Google Hangout communication and follow the connection streams and subsequently feed to a trained classifier for decision making[2].

As discussed in the previous section connection is made to one of the conference server offered by the Google servers. As a first step we identify the DNS resolution sequence in network traffic and by virtue of that we can identify the conference server. Subsequently we follow the connections made to the conference server. Based on the Hangout connection behavior we use 7 features listed in Table I to classify the packets.

TABLE I
Feature Set

SN	Feature	Value
1	Is Packet is to/from One of the conference Server Offered in DNS reply to clients*.Google.com	Yes/No/NA
2	Layer 4 Protocol	UDP/TCP
3	Is UDP using Same Source Port as STUN	Yes/No/NA
4	Destination Port Number	Integer
5	Is Packet is to/from One of the Conference Server Offered in DNS reply to stun.l.Google.com	Yes/No/NA
6	Packet Length	Integer
7	Type of STUN Packet	Binding Request, Binding success response, Binding response to server, Binding success response user from server, Binding request user with id from server, NA

B. Classification Algorithms

We experiment with 3 different classification algorithms namely Naive Base, AdaBoost and J48 decision tree algorithms which are explained below in brief.

1) Naive Base Algorithm: Naive Base is a statistical classification algorithm based on Bayes theorem. It predicts the posterior probability of instance belonging to a particular class using prior probabilities of various attributes and their probability of belonging to a particular class. For calculating prior probabilities of various attributes it assumes class conditional Independence i.e., every attribute is independent of every other attribute so that probability of a attribute and its probability of belonging to a class can be easily calculated by just counting the occurrences of attribute and class together.

2) AdaBoost: AdaBoost is a weighted ensemble classification method where several classification models are used to decide the class label of an instance. Given a dataset D of size n belonging to one of the C classes it samples the dataset and use it for creating a classification model. Totally K such training models are created by sampling the dataset. The algorithm adjusts the weights of samples after assessing the effectiveness of model M_i before creating the model M_{i+1}.

[2]classification starts after the DNS resolution sequence is completed

3) J48: It is a decision tree algorithm and hence generates decision tree from the input training samples. To begin with it selects an attribute which best classifies the classes i.e., an attribute with highest information gain. If any of the values for this attribute if there is no ambiguity for class assignment then the branch is terminated otherwise another attribute is chosen with highest gain and class assignment procedure is repeated.

V. Experiments

In this section we describe the experiments done to verify the proposed approach of identifying Google Hangout packets. The first part of the experiments is to collect labeled dataset with ground truth assigned. We collected dataset from few home computers connected to Internet using ADSL connectivity with Wireshark [28] on a windows machine. We collected data in two parts first is data of non Google Hangout traffic and second is exclusively of Google Hangout traffic. In both the cases due care is taken not to contaminate traffic. In the first case, all Google services are manually terminated and traffic was recorded. In this duration none of the Google services including Google's search engine was used. In the second part exclusively Hangout traffic is collected by running only Hangout client and terminating all other Google services. Both the traces are manually analyzed for the presence of contamination. Table II shows the details of dataset collected and their approximate size and duration. We do experiments in 3 stages in the first case we classify Goggle Hangout traffic with non Google service traffic. This is a two class classification problem and is discussed in section V-A. In the second phase we address the problem of classifying Hangout traffic with other Google service traffic and also with rest of traffic these experiments are discussed in section V-B. We also do a third variation with 4 class classification with some common behaviors of Google services being classified into a fourth class and these experiments are detailed in section V-C.

A. Two Class Classification of Hangout and Others

Once the traces for both cases are collected next step was to extract features from packets and appropriately label them. We wrote a java program using jnetpacp[29] library to read traces and extract features mentioned in section IV. Once the labelled feature vectors are obtained next step was to use classification algorithms mentioned above and train the classifier. In our experiments we used Weka tool [30] to train and classify the feature vectors. We did 10 fold cross validation to infer the performance of classifier. Performance of each type of classifier is reported as a confusion matrix. Table III, IV and V shows the confusion matrix for Naive Base, J48 and AdaBoost algorithms. Recall of Hangout packets for various algorithms is shown in Table VI. We can notice from these 3 tables that Naive Base algorithm performed poorly and j48 algorithm has a better classification accuracy.

B. Classifying Hangout and Gmail Packets Using 3 class Classification

In the second set we conducted experiments to accurately classify Hangout packets with other Google services traffic.

TABLE III
Confusion Matrix for NAVE BAYES Classification

Actual Label	Classified as Not Hangout	Classified as Hangout
Not Hangout	1984948	0
Hangout	14368	674657

TABLE IV
Confusion Matrix for J48 Classification

Actual Label	Classified as Not Hangout	Classified as Hangout
Not Hangout	1984949	5
Hangout	10	689015

TABLE V
Confusion Matrix for AdaBoost

Actual Label	Classified as Not Hangout	Classified as Hangout
Not Hangout	1984945	9
Hangout	5	689020

TABLE VI
Recall for various Classification Algorithms

Algorithm	Recall
Naive Base	97.91%
AdaBoost	99.99%
J48	99.99%

We used gmail traffic in the experiments as a sample case. We collected gmail traffic separately by following the similar steps as in the case of Hangout. The details of the datset used for this experiment is shown in Table VII. Same set of features are extracted and feature vectors are labeled manually as in previous case. The classification performance of various algorithms for this experiment is tabulated in Table VIII, Table IX and Table X. One of the issue with this experiment is, Google services use only one account type for all services and there are many steps which are common particularly initial account verification and login connection to connection server is found in both Hangout and Gmail services. Figure 3 shows the sequence of DNS resolution operations in gmail. We can notice the first two steps are same as in Hangout this leads to difficulty in classification and as a result there are few misclassifications. For example j48 classifier which performed better with very few misclassifications previously also now has more missed detection cases. For the other two algorithms too similar trend was observed. The reason for increased misclassification is attributed to the common behavior found in gmail and Hangout (which is addressed using a 4 class classification detailed in next sub section).

TABLE VIII
Confusion Matrix for NAVE BAYES Classification

Actual Label	Classified as Not Hangout	Classified as Hangout	Classified as Gmail
Not Hangout	1979714	0	5240
Hangout	94	344289	1853
Gmail	0	0	331963

TABLE II
DATASET DETAILS

Packet Type	Packet Count	Percentage	Duration of Data Collection	Size of Data
Google Hangout	689025	25.76%	7 Hours	350 MB
Non Hangout	1984954	79.81%	12 Hours	1.5 GB

TABLE VII
DATASET DETAILS FOR HANGOUT AND GMAIL TRAFFIC IDENTIFICATION

Packet Type	Packet Count	Percentage	Duration of Data Collection	Size of Data
Google Hangout	346236	13.0%	4 Hours	200 MB
Gmail	331963	12.4%	4 Hours	400 MB
Non Hangout	1984954	74.6%	12 Hours	1.5 GB

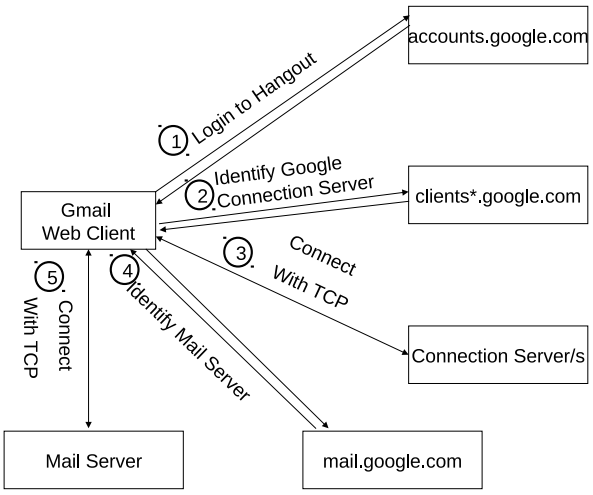

Fig. 3. Gmail Operation Sequence

TABLE IX
CONFUSION MATRIX FOR J48 CLASSIFICATION

Actual Label	Classified as Not Hangout	Classified as Hangout	Classified as Gmail
Not Hangout	1984954	0	0
Hangout	0	346194	42
Gmail	0	0	331963

TABLE X
CONFUSION MATRIX FOR ADABOOST CLASSIFICATION

Actual Label	Classified as Not Hangout	Classified as Hangout	Classified as Gmail
Not Hangout	1984954	0	0
Hangout	0	344383	1853
Gmail	0	0	331963

TABLE XI
RECALL FOR 3 CLASS CLASSIFICATION

Algorithm	Recall
Naive Base	99.98%
AdaBoost	99.99%
J48	99.46%

C. Classifying Hangout and Gmail Packets using 4 class Classification

This set of experiments use commonalities among Hangout and Gmail as a separate class into Google Plus specifically steps 1 to 3 in name resolution are labeled as Google Plus and trained. Thus we have 4 classes and details of the dataset used for this experiment is shown in Table XII. Classification accuracy of various algorithms are also shown in Table XIII, Table XIV and Table XV. We can notice from these tables that classification accuracy improved for all the algorithms in comparison to the previous case while improving or retaining the recall of Hangout packets as shown in Table XVI.

TABLE XIII
CONFUSION MATRIX FOR NAVE BAYES CLASSIFICATION

Actual Label	Classified as Not Hangout	Classified as Hangout	Classified as Gmail	Classified as Google Plus
Others	1979706	0	2473	2775
Hangout	84	344299	0	0
Gmail	0	0	238961	0
Google Plus	0	0		94855

TABLE XIV
CONFUSION MATRIX FOR J48 CLASSIFICATION

Actual Label	Classified as Not Hangout	Classified as Hangout	Classified as Gmail	Classified as Google Plus
Others	1984954	0	0	0
Hangout	0	344383	0	0
Gmail	0	0	238961	0
Google Plus	0	0	0	94855

TABLE XV
CONFUSION MATRIX FOR ADABOOST CLASSIFICATION

Actual Label	Classified as Not Hangout	Classified as Hangout	Classified as Gmail	Classified as Google Plus
Others	1984954	0	0	0
Hangout	0	344383	0	0
Gmail	0	87602	151359	0
Google Plus	0	84987	9868	0

VI. CONCLUSIONS

Network traffic classification is an important task to provide differentiated service quality to various applications and also in

TABLE XII
DATASET DETAILS FOR HANGOUT, GOOGLE PLUS AND GMAIL TRAFFIC IDENTIFICATION

Packet Type	**Packet Count**	**Percentage**	**Duration of Data Collection**	**Size of Data**
Google Hangout	344383	12.9%	4 Hours	200 MB
Gmail	238961	8.9%	4 Hours	400 MB
Non Hangout	1984954	74.6%	12 Hours	1.5 GB
Google plus	94855	3.6%	Taken from Hangout and gmail traces	NA

TABLE XVI
RECALL FOR 4 CLASS CLASSIFICATION

Algorithm	Recall
Naive Base	99.98%
AdaBoost	100.0%
J48	100.0%

security monitoring. Several organizations and Internet Service Providers perform network traffic classification. In this paper we used application semantics to identify a set of features which are subsequently used by a classification algorithm to identify the class to which a packet belongs. We experimented with a dataset collected for Google Hangout and reported its detection performance using few classification algorithms. Specifically we used Naive Base, j48 and AdaBoost classification algorithms to assess the detection performance and reported results. Since Google services uses common behavior between them we find the classification where the common behavior is identified separately performs better in comparison to others. The ability to identify packets at this granularity permits to apply QoS at ease. In the future we would like to extend this work with other Google services and also for other peer-to-peer application identification.

REFERENCES

[1] D. Moore, K. Keys, R. Koga, E. Lagache, and k. claffy., "The coralreef software suite as a tool for system and network administrators," in *LISA '01:Proceedings of the 15th USENIX Conference on System Administration*, pp. 133–144, USENIX Association, 2001.

[2] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *PAM'05: Proceedings of the 6th International Conference on Passive and Active Network Measurement*, pp. 41–54, Springer-Verlag, 2005.

[3] "http://en.wikipedia.org/wiki/edonkey2000."

[4] "http://www.emule-project.net/."

[5] "http://ultrasurf.us/."

[6] "https://www.torproject.org/."

[7] T. Karagiannis, A. Broido, M. Faloutsos, and K. claffy, "Transport layer identification of p2p traffic," in *IMC '04: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, pp. 121–134, 2004.

[8] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, "Revealing skype traffic: When randomness plays with you," in *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '07, pp. 37–48, 2007.

[9] J. a. V. Gomes, P. R. M. Inácio, M. Pereira, M. M. Freire, and P. P. Monteiro, "Detection and classification of peer-to-peer traffic: A survey," *ACM Computing Surveys*, vol. 45, no. 3, pp. 1–40, 2013.

[10] A. Dainotti, A. Pescape, and K. Claffy, "Issues and future directions in traffic classification," *Netwrking Magazine of Global Internetwoking*, vol. 26, no. 1, pp. 35–40, 2012.

[11] N. Hubballi and V. Suryanarayanan, "False alarm minimization techniques in signature-based intrusion detection systems: A survey," *Computer Communications*, vol. 49, no. 1, pp. 1–17, 2014.

[12] W. Donato, A. Pescap, and A. Dainotti, "Issues and future directions in traffic classification," *IEEE Network*, vol. 28, no. 12, pp. 56–64, 2014.

[13] "https://www.sandvine.com/technology/traffic-classification.html."

[14] C. McCarthy and A. N. Z. Heywood, "An investigation on identifying ssl traffic," in *CISDA'11:Proceedings of the Fourth IEEE International Conference on Computational Intelligence for Security and Defense Applications*, pp. 1–8, IEEE, 2011.

[15] R. B. Yanai, M. Langberg, D. Peleg, and L. Roditty, "Realtime classification for encrypted traffic," in *SEA'10:*, pp. 1–13, LNCS, 2010.

[16] R. Alshammari and A. N. Heywood, "Machine learning based encrypted traffic classification: Identifying ssh and skype," in *CISDA'09:Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*, pp. 289–296, IEEE, 2009.

[17] 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*, vol. 24, no. 1, pp. 104–117, 2013.

[18] "https://www.bluecoat.com/products/packetshaper."

[19] "https://www.paloaltonetworks.com/."

[20] A. Tongaonkar, R. Torres, M. Iliofotou, R.Keralapura, and A. Nucci, "Towards self adaptive network traffic classification," *Computer Communications (In Press http://dx.doi.org/10.1016/j.comcom.2014.03.026)*, 2014.

[21] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 5–16, 2007.

[22] P. Branch, "Using machine learning to identify realtime traffic classes."

[23] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *Proceedings of the 2006 ACM CoNEXT Conference*, ACM CoNEXT '06, pp. 1–12, 2006.

[24] 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.

[25] T. T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel traffic classification in the dark," in *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ACM SIGCOMM '05, pp. 229–240, 2005.

[26] S. Zander, T. Nguyen, and G. Armitage, "Automated traffic classification and application identification using machine learning," in *LCN05: Proceedings of the IEEE Conference on Local Computer Networks 30th Anniversary*, pp. 1–8, IEEE, 2005.

[27] P. Dorfinger, G. Panholzer, and W. John, "Entropy estimation for real-time encrypted traffic identification," in *TMA'11:*, pp. 164–171, LNCS, 2011.

[28] "https://www.wireshark.org."

[29] "http://www.http://jnetpcap.com/."

[30] "http://www.cs.waikato.ac.nz/ml/weka."