Hindawi
Journal of Computer Networks and Communications
Volume 2019, Article ID 5758437, 10 pages
https://doi.org/10.1155/2019/5758437

Research Article

Using Burstiness for Network Applications Classification

Hussein Oudah[1,2], **Bogdan Ghita**[1], **Taimur Bakhshi**[3], **Abdulrahman Alruban**[1,4] **and David J. Walker**[5]

[1]*Centre for Security, Communications and Network Research, University of Plymouth, Plymouth, UK*
[2]*Department of Mathematics and Computer Applications, Al-Muthanna University, Samawah, Iraq*
[3]*National University of Computer & Emerging Sciences, Lahore, Pakistan*
[4]*Department of Information Technology, Computer Sciences and Information Technology College, Majmaah University, Al-Majmaah, 11952, Saudi Arabia*
[5]*Centre for Robotics and Neural Systems, University of Plymouth, Plymouth, UK*

Correspondence should be addressed to Bogdan Ghita; bogdan.ghita@plymouth.ac.uk

Received 29 March 2019; Revised 19 June 2019; Accepted 25 July 2019; Published 20 August 2019

Academic Editor: Djamel F. H. Sadok

Network traffic classification is a vital task for service operators, network engineers, and security specialists to manage network traffic, design networks, and detect threats. Identifying the type/name of applications that generate traffic is a challenging task as encrypting traffic becomes the norm for Internet communication. Therefore, relying on conventional techniques such as deep packet inspection (DPI) or port numbers is not efficient anymore. This paper proposes a novel flow statistical-based set of features that may be used for classifying applications by leveraging machine learning algorithms to yield high accuracy in identifying the type of applications that generate the traffic. The proposed features compute different timings between packets and flows. This work utilises tcptrace to extract features based on traffic burstiness and periods of inactivity (idle time) for the analysed traffic, followed by the C5.0 algorithm for determining the applications that generated it. The evaluation tests performed on a set of real, uncontrolled traffic, indicated that the method has an accuracy of 79% in identifying the correct network application.

1. Introduction

In the context of the ever-increasing network activity and reliance on the Internet, monitoring and characterising web traffic is critical for network administrators for operational and security activities. A number of directions were explored, such as establishing what websites the users are interested in, how much traffic is generated by specific applications, and whether these applications or services can be controlled in terms of network resource demands [1]. The research community proposed a number of alternatives, with earlier studies focusing on port-based approaches and deep packet inspection. However, these methods failed to identify applications as they currently use dynamic/well-known ports such as port 80 or encrypted methods such as SSL/TLS [2, 3]. In recent years, studies have focused on using statistical features approach for identifying traffic associated with applications based on machine learning algorithms [4]. This method relies on the characteristics of IP flows such as the number of packets in a flow and size and duration of a flow which reflect unique patterns for applications. The aforementioned method was considered flexible for emerging traffic as it utilises the network level (packet header) with promising results rather than the application level (packet contents) [5].

For better classification decisions, the prior art proposed either new machine learning algorithms (MLA) or novel feature-based approaches. The majority of previous studies in the domain of traffic classification focused upon introducing new MLA, and little attention has been given to the extraction of new features. To this end, this study aims to explore the potential of extracted new features subset and find out whether these features have a positive impact on the system performance. The new features need to be sufficiently

discriminative in order to distinguish between applications. The proposed approach focuses on classifying web-based applications such as Facebook and YouTube, which use the network/Internet to manage requests from a client to an application server rather than network-specific utilities/protocols such as SMTP and FTP. Identifying the optimal features for applications reduces the potentially large dimensionality and might be useful to improve the system performance [6]. The proposed features were extracted from real data, which were collected from a group of 20 users at the University of Plymouth for two months, using tcptrace tool and evaluated with the C5.0 algorithm [7]. The data were labelled based on DNS queries and IP addresses for each examined application, which was identified from our previous studies [8, 9].

The rest of the paper is organised as follows: Section 2 discusses the state-of-the-art traffic classification approaches in more detail to provide a review of the limitations of present techniques. Section 3 highlights the proposed method and analysis and introduces the feature set. Section 4 presents the results using C5.0 algorithm, and Section 5 concludes the paper with a summary of achievements.

2. Related Work

As mentioned in the introductory section, the classification of applications has received significant interest from the research community in recent years. This section summarises previously proposed classification techniques and their limitations.

2.1. Traffic Classification Techniques. In the early days of the Internet, its applications were identified easily based only upon port number [10]. IANA [11] assigned protocols to well-known transport layer ports; therefore, the identification process was merely based upon matching the port number in the packet header with the table containing the port applications. Due to the continuous growth of Internet applications, standard ports are no longer used; instead, they have been moved towards a web-based front-end or have used dynamic ports [2]. Consequently, this method becomes inaccurate when identifying applications and typical performance ranges between 30 and 70%, depending on the mix of traffic, and this includes a number of applications to be identified [12]. Following from the improvement in processing power, deep packet inspection (DPI) [13] was then the preferred choice, as it identifies signatures of applications or protocols based on the content of the packets. DPI also became inefficient as most traffic nowadays is encrypted. Moreover, it breaches the privacy of the users and did not scale well from a computational perspective with the increase in core network speed [3, 14].

The research community therefore introduced two techniques to avoid these limitations, focusing on host behaviour and statistical methods. The former technique is based on the idea that hosts generate different communication patterns at the transport layer; by extracting these behavioural patterns, activities and applications can be classified. Although the method showed acceptable performance (over 90%) [15] and it can detect the application type, it could not correctly identify the application names, classifying both Yahoo and Gmail as e-mail [16]. In contrast, high accuracy was achieved (over 95%) by applying the latter approach of statistical methods [17–20], using statistical features derived from the packet header, such as number of packets, packet size, interarrival packets time, and flow duration with the aid of machine learning algorithms. The advantage of using ML algorithms is that they can be used in real-time environments to provide rapid application detection with high accuracy. For instance, the authors of [21] used the Naïve Bayes techniques with the statistical features to identify traffic. Other ML algorithms utilised were Bayesian neural networks, support vector machines, and decision trees [7, 22, 23]. In [7], the author used a C5.0 decision tree algorithm to classify seven applications with average accuracy over 99%. However, the process of feature selection, which must be flexible to the network circumstances, is a critical point in the construction of a classifier [6].

Given this classification, the statistical differences between interarrival times of packets and flows approach outlined in this work strengthen the behavioural and statistical methods by considering arrival times of packets and flows as discriminating features among applications. The authors in [24] proved that there is variability (burstiness) in network traffic by using a measure called index of variability. The hypothesis that timing can be used to discriminate between applications was also put forward in [25], which highlighted that applications generate different behaviour based on statistical features relating to the timing of packets arriving. More details about burstiness were proposed in [26], which defines it in two levels. The first level was called a small-time scale flight (STF) which means that the interarrival times of packets occur within a predefined time T (a constant threshold in the range of 5–10 milliseconds). The second level is a large-time scale flight (LTF), defined as larger interarrival times of packets with a value of 40–1,000 milliseconds. A different number of bursts would be generated for each category based on the value of the threshold.

From an efficiency perspective, it should be noted that the statistical approach is appropriate for traffic classification as it can deal with encrypted traffic, which nowadays has become dominant, and it can adapt to real-time traffic. In our previous studies [8, 9], we proposed a set of attributes based on burstiness and idle time; six applications were used to evaluate our features with the aid of C5.0 method, the results showed high accuracy in identifying six applications over 97%.

2.2. Splitting Traffic Based on DNS Requests. Given the diminishing success of port analysis and DPI, traffic classification can also be based on DNS analysis. The authors of [27, 28] focused on the volume and variety of DNS

queries generated from both clients and servers, aiming to observe the effect of caching mechanisms on the client side. Other studies, such as [29, 30], exploited the DNS information to reveal malware activities. Furthermore, the authors in [31] used DNS queries to classify traffic by matching keywords in the domain names table with the collected flows of traffic. These labelled flows were categorised based on domain name similarity, and the aim was to break down the traffic volume. Similarly, the study [32] utilised DNS to tag flows by capturing the first packet of each flow and exploiting the domain name which was separated into keywords to form vectors for each application. Also, they used the port number and transport application name as features to classify applications. They claimed that the provided DNS information could be useful to identify more than 30% of traffic. In [33, 34], the authors used DNS to label flows based on the keywords available after resolving IP addresses. Otherwise, the flows would be classified based on selected attributes and with the aid of machine learning to improve accuracy. Another study [35] collected large traffic from University of Auckland and it found that about 10% of their observed TCP traffic did not use DNS lookups and neither did about 85% of UDP lookups.

Using a similar scenario, the authors in [36] argued that traffic could be classified based on the IP address and hostname. Although the results showed that up to 55% of web traffic could be identified based on the proposed method, it also had a high accuracy in identifying applications such as WhatsApp, Twitter, and Dropbox. Based on the long-term monitoring, the authors concluded that the IP addresses of servers associated with each application remain stable for short periods, but they change over long periods. The study recommended updating and checking the IP addresses frequently for the methods that rely on IP addresses as a key feature. Similarly, the authors in [37] proposed a method to label websites based on server IP addresses. Firstly, they collected data from different users working on the same website to ensure that the server IP address belongs to the same application, and then they built a ground truth of IP addresses for specific applications and used it to classify a mix of traffic flows. The method showed good results when considering DNS queries. Following the same line of research, the authors in [38] used IP server addresses to group traffic applications to study the user activities, and the authors in [39, 40] claimed that the IP address represents an informative feature. In [1], the authors claimed that traffic could be classified based on DNS, and they proved that a majority of traffic could be resolved, such as HTTP and HTTPS generated traffic, except P2P applications.

From previous studies, we conclude that DNS information and IP addresses could be active factors in classifying applications. We need to look into these attributes for each application and check if they are unique and robust when presented with the variable network environment. The next section focuses on the concept of burstiness and idle time and how burstiness-related features may be generated from tcptrace [41].

3. Proposed Method and Features

As concluded from the prior art, the statistical approach is a robust and reliable approach, allowing efficient network-based (packet header) rather than application-based (packet contents) analysis. Extracting the most discriminative features that characterise applications remains the key to success in this approach without being biased by either user or network circumstances such as congestion and delay.

This paper aims at identifying an additional set of features that can be used to discriminate between applications, based on the statistical differences between interarrival times of packets and flows. We focus in particular on burstiness, which defines closely spaced data exchanges, such as objects on the same page, and idle periods, which separate longer-term transactions, such as moving from one page to another when the user is browsing a website. The assumption that we make is that different applications produce different distributions of packet size, duration, distribution of the bursts, and idle time parameters. Consequently, Internet applications behave inherently different, generating different amounts of data, creating various connections and timing patterns between the generated packets and flows, beyond the generic distribution of connections for overall traffic [42]. For instance, streaming a video on Netflix versus e-mail checking or using social media could lead to significantly different packet arrival patterns and hence a slightly different burstiness signature.

The following example explains the concept of burstiness and how it may be used to discriminate the behaviour of Internet applications. When a user is browsing an application, for instance, the BBC news website (https://www.bbc.com/news), the session would consist of some pages that the user chooses to visit. Within each page, the browser will be requesting and downloading the objects embedded in the page, some on the same site and some hosted on other sites. From a timing perspective, the download of objects on a page would appear as a burst of connections, followed by a period of inactivity (idle time) while the user reads the page until he/she decides to click on a link and load another page. Figure 1 shows how the group of packets forms a burst based on interpacket arrival time and inactivity of time between bursts. This burstiness phenomenon could be happen within packets or flows. In this study, the burstiness concept will be defined in two levels, the first level is in the context of packet analysis and the second level is in the context of flow analysis.

3.1. Packet Analysis. In a packet-based analysis, the bursts and idle times would be formed based on the interarrival times for packets during the connection between the client and server. This level was defined in [43] as a group of consecutive packets with shorter interarrival delays than the packets arriving before or after them. Given one of the two unidirectional data flows within a connection, a burst_threshold (T) is defined as a maximum time delay between the arrivals of two consecutive packets that belong

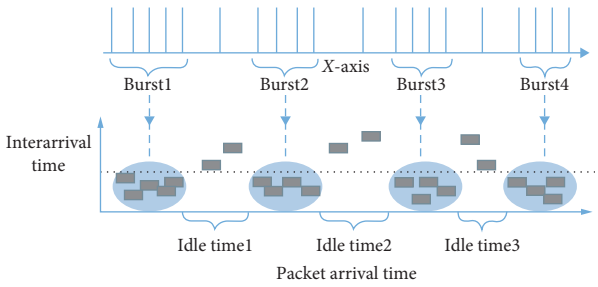

Figure 1: Definition of bursts and idle time.

to the same burst. Similarly, idle_threshold (I) is defined the distance between groups of packets of interarrival time at which could be identified the idle time that separates two consecutive data exchanges and could be defined as I. In order to provide a meaningful description of the interactions, the analysis must establish the values for T and I and whether they should be constant or dynamic. A previous study [26] proposed two ranges for T, of 5 ms–10 ms and 40 ms–1 s. Another study [43] proposed two different scenarios for the value of T; the first one was dynamic, which means different values could be for T, while the second scenario was fixed without proposing any values for T. In order to get an image of the range of time values for the protocol interaction, Figure 2 shows the interpacket arrival time for five applications. Most distributions of the interpacket arrival time fall under 1 second, except for YouTube that falls under 0.5 seconds. Accordingly, the burst_threshold could be set to 1 second, while the idle threshold was set to 10 seconds. While the application does indeed exhibit a different signature in terms of packet arrival distribution, user behaviour may also influence this distribution, particularly in relation to long-term activity, as idle times are a factor of user behaviour too. The idle time could be varied according to the behaviour of the user when he/she moves from one page to another. As shown in previous studies, the distribution of timing for user connections may be used as a discriminant among users [38, 44]. However, while users may introduce a level of noise in the distribution, a sufficiently large data sample would allow determining the benefits and limitations of the method. Prior studies, such as [45], utilised idle time values typically ranging from 10 seconds to 5 minutes, the idle_threshold (I) proposed to set 10 seconds. As shown in Figure 1, many features could be extracted from each flow and each direction such as a total number of bursts in the direction a-b/b-a, the total number of packets within bursts for each direction, and the total size of bursts in bytes in each direction. The pseudocode in Algorithm 1 summarises the estimation of bursts and idle time between packets and for each flow. For each packet arrival, the interarrival time is compared with the two burstiness thresholds to determine whether the packet is part of a new burst or session. The possible features that could be extracted from the pseudocode are described in Table 1, each of the inputs in the table is a pair of variables, one for the a-to-b direction and one for the b-to-a direction, where a refers to the client side and b refers to the server side.

3.2. Flow Analysis. The same concept was applied to calculate the burst and idle time between flows. The calculation was measuring time differences between the initial times of flows and subsequent flows, which are calculated from the first packet of each flow. The timestamp of the first packet time of the first flow is subtracted from the timestamp of the first packet time of the second flow; if the time difference is less than 1 s, then a burst is formed. Otherwise, if the time difference is more than 10 s, then the period is considered an idle time. Table 2 summarises burst-based features.

3.3. Conventional Analysis. In our experiment, the proposed features are compared with the previous ones to show the effects of the proposed method in distinguishing between applications. These features were calculated for each direction of flow as shown in Table 3.

3.4. Splitting Traffic. In our earlier studies [8, 9], data collection was based on controlled application usage, with users being given instructions of what to do, which applications to use, and for how long. The users were asked to browse these applications separately. Hence, the data were collected per application and dumped in labelled files. Accordingly, from each application file, the destination IP address was extracted and dumped in separated files. For this paper, a real data traffic was collected and the DNS requests were used to identify which application is requested. We acknowledge that DNS may not be the most accurate method, but it allowed testing the accuracy of our method. In addition to the automatic allocation to applications, after each request, IP addresses were extracted for three seconds to update the IP address files, finally matching between traffic flows and the IP address files and storing the matched traffic in files based on each application. These two mechanisms were to label data with the applications which will be the input to the classifier and to examine the proposed features.

4. Experimental Methodology

A high-level architecture of the proposed system is presented in Figure 3, which highlights the key components of the application identification scheme. Firstly, a real traffic was collected from twenty hosts in an office environment for over two months. Afterwards, the packet trace was parsed to read the captured DNS requests, and the name was used to identify the application type, such as Google or BBC news. Also, the packet trace was analysed by tcptrace to generate flows with proposed and conventional features. Finally, the resulting flows were labelled based on IP address, which was then fitted into a C5.0 classifier. The following subsections provide more detail on the methodology.

4.1. Data Collection. The raw data traffic was collected from a lab in the University of Plymouth between May and July 2018 from a group of 20 PhD students. The data were collected using a tcpdump tool via a network-based

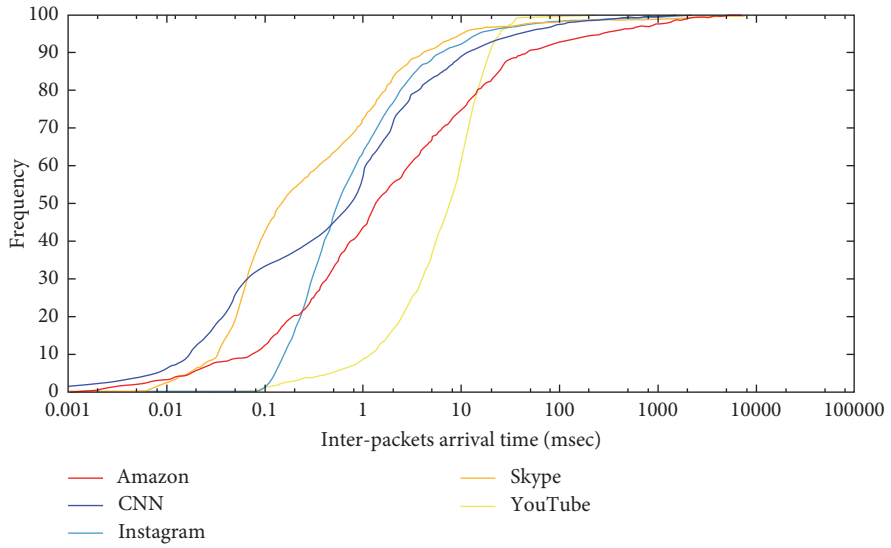

Figure 2: The distribution of interpacket arrival times for five applications.

```
burst_threshold = 1 s
idle_threshold = 10 s
   initialise burst and idle time parameters
   while packets arriving
   do
      calculate interarrival_time
      if interarrival_time < burst_threshold
         current_burst ++
         current_session ++
      else
         burst_counter ++
         current_burst = 1
         if interarrival_time > idle_threshold
            current_session = 1
            session_counter ++
            idle_time + = interarrival_time
         fi
      fi
   done
```

Algorithm 1: Estimation of packet bursts and idle time.

Table 1: Burstiness and idle time features amongst packets (for each direction).

Feature	Description
Bursts	Total number of bursts between packets for each direction
Packets-in-burst	Total number of packets in bursts for each direction
Burst-size	Total bytes for bursts for each direction
Burst-size-b/Burst-size-a	The ratio between burst-size-b and burst-size-a
Burst-duration	The time duration of bursts for each direction
Burst-duration-a/Packet-a	The ratio between burst duration and number of all packets in bursts
Burst-duration-b/Packet-b	The ratio between burst duration and number of all packets in bursts
Idle-time	The accumulation of inactive time for all packets
Idle-time-data	The accumulation of inactive time for only data packets

Table 2: Burstiness and idle time features for flow analysis.

Feature	Description
Burst-no	Total number of bursts between flows for each session
Flows-no	Total number of flows within all bursts for each session
Packets-no	Total number of packets within all bursts for each session
Packets-data-no	Total number of data packets within all bursts for each session
Burst-size	The total size of all bursts in bytes for each session
Average-burst-size	The average size of all bursts for each session
Burst-duration	The total time duration for all bursts
Burst-duration/burst-no	The ratio between burst duration and the total number of bursts for each session
Burst-idle-time	The total inactive time between flows for each session

Table 3: Conventional features proposed by previous studies.

Features	Description
Packets	Total number of packets
Data_packets	Total number of data packets
Flags_packets	Total number of TCP flag packets
First_packet	The size of the first packet
Flow_duration	The time of the last packet subtracted by the time of the first packet
Inter_arrival_time	The time duration of each direction divided by the total number of packets
Packets_b/packets_a	A ratio of received packets to transmitted packets
Data_packets_b/data_packets_a	A ratio of receiving of data packets to transmitted data packets
Flags_packets_b/Flags_packets_a	A ratio of received of flags packets to transmitted of flags packets

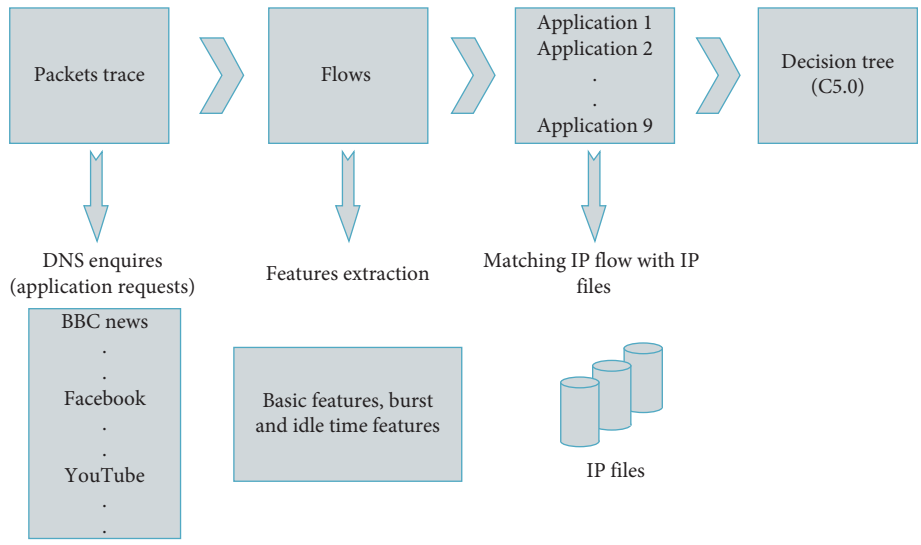

Figure 3: Proposed traffic classification methodology.

method and were divided into 24 samples per day; each sample represented a one-hour traffic of pcap format which is suitable to be input to the tcptrace tool; this division reduces the size and processing time of each sample.

4.2. DNS Enquires. The collected data were packet-based, which contain DNS queries; the content of the DNS requests is used to identify applications. In each DNS packet request, there is a keyword that refers to the requested application. For instance, the keywords for BBC news and YouTube are bbc.co.uk

and youtube.com, respectively. A Python script is used to read the DNS request from the user and tag the time of the application requested. Three seconds after each request usually belong to the same application as noticed from the monitoring the traffic. Therefore, the IP addresses for this period were tagged as well for the same requested application. This process partitions the traffic into many applications considering the specific time stamp of each request which is essential in the next stage.

4.3. Feature Extraction (Packet Analysis). The collected Internet traffic was analysed using the tcptrace tool [41]. The tool takes packets as input and output flows that are sharing the same five tuples (source IP address, source port number, destination IP address, destination port number, and protocol). The concept of burstiness and idle time was implemented in this tool to generate the desired features; the script was written inside the tool to extract the packet features. Moreover, features that were proposed by other studies were extracted from the same tool. In this stage, three types of flows were classified; firstly, all flows which were extracted with packet-based features, secondly, some flows were tagged with time and name of applications, and finally, most flows were tagged as unknown flows.

4.4. IP Address Matching. The uncontrolled data were analysed as shown in the previous subsection into flows that contained known flows based on reading the DNS requests plus the three seconds after the requests. Therefore, the matching process started with reading the known flows to determine the application name and afterwards fetching the specified file of the IP addresses for that application. Secondly, matching the unknown flows with the specified file until the end of the flow trace and tag them as known flows. Finally, dumping known flows in separated files and labelling them according to the application name. Based on previous studies, the IP files are subjected to change continuously by the owners of applications for security reasons. Therefore, updating these addresses is essential, but it must be automatic and during the identification process. The results are nine applications with details in Table 4. As seen from the results, the application that was most frequently used by the users was the Gmail, against very low usage for Yahoo mail. The application identification accuracy of the proposed features versus traditional ones was evaluated using three feature sets. The first set included features suggested by previous studies as introduced in Table 3; the second set contained the burstiness and idle time features proposed by this paper, as presented in Tables 1 and 2, while the third set included the full list of inputs from the other two. Cross-validation technique was used for training and testing a model with three folds as ratio 2/1, respectively. Cross validation is a statistical method that divides data into equal folds, one fold used for validating the model, and the others used for training it. In each new round, a different fold is used for validating the model so that the training can be shown to be effective across different datasets. During the process, each fold will be used for validation once. This technique is used to evaluate the performance of machine learning model by testing the model on unseen data to avoid overfitting and underfitting problems. This technique partitioned the data into three equal parts, and the model was trained on two parts of the data and tested on the remaining part. The process was repeated three times, and the error was calculated by taking the average of all errors. The classification algorithm was applied to all three feature sets with six different boosting values (0, 10, 15, 20, 50, and 100). The boosting refers to algorithms that apply weak classifiers to build a strong classifier by combining the results. The algorithm gives all records the same weight and applies a sequence of iterations of classification; the misclassified records increase their weight, while the weight of the right classified records is reduced. Finally, a strong classifier is created from incorporating the individual ones with the best tuning for the parameters to avoid overfitting. The results in Table 5 indicate low accuracy for set1 compared to set2 as the burstiness features to increase the efficiency of the classifier in discriminating the different applications. Combining both sets showed considerable improvement in classification accuracy peaking at 79.68% at boost 10.

Table 4: Overall results for classification of the observed data.

Application	Flows	Duration (h)	Number of samples
BBC news	3,150	1.6	6
Facebook	98,210	33.1	287
Google	59,422	88.5	892
Yahoo mail	6,795	0.8	9
YouTube	66,500	76.5	714
Gmail	1,448,392	143	870
Amazon	23,975	6.6	34
Plymouth.ac.uk	24,225	42.5	286
Bing	10,324	17.2	110

The proposed features show the ability to better discriminate between the applications in comparison with the other parameters, which enhances the classifier capability. Table 6 shows the comparison between basic and burstiness features for the most attributes that were used by C5.0 classifier. The burstiness attributes reported maximum usage in segregating the applications. This is another indicator that the classifier strongly relied on the proposed features because they provide high discrimination between applications.

4.5. Confusion Matrix. The accuracy, as presented in the previous section, represents only the ratio of correctly classified samples versus all samples. In order to further analyse the performance of the classifier, Table 7 presents a confusion matrix for the observed data, which is presented in Table 4. Rows of the matrix represent predicted samples of each application and columns represent the actual samples. For example, the actual samples for Gmail are 289, which are a summation of a first column, 198 samples are predicated correctly, while the other samples are predicted as false-positives for different applications. On the contrary, the row for Gmail represented a predication of other applications and classified as false-negative for Gmail application. The overall performance of the classifier is high for all

Table 5: Average accuracies with different feature sets using 3-fold cross validation.

Boosting	0	10	15	20	50	100
Set1	47.77	56.56	58.05	58.54	60.30	60.31
Set2	49.30	58.75	60.21	61.11	64.23	65.51
Set3	52.55	79.73	73.99	67.78	68.10	67.13

Table 6: Attribute usage in C 5.0 classifier.

Basic features usage (75-100)%
data_packets[min, max], flow_duration[mean, min], flags_packets[mean, min, max], inter_arrival_time_data[sd, min]
Burstiness features usage (75-100)%
burst_size_bytes[md, min, mean], burst_no[sd, min], idle_time_data[mean, min, sd], pkt_data_count[min, mean], pkt_count[min, sd], inter_arrival_time_burst_conns[min, sd], inter_arrival_time_burst[mean, max], burst_size_bytes_data[max, min, mean], burst_duration [sd, mean], burst_data_no[min]

Table 7: Confusion Matrix results for optimal classifier3.

Applications	Gmail	Ymail	Amazon	BBC	Bing	Facebook	Google	UoP site	YouTube
Gmail	198	0	0	0	3	6	14	5	13
Ymail	0	4	0	0	0	0	0	0	0
Amazon	0	0	9	0	0	0	0	0	0
BBC	0	0	0	2	0	0	0	0	0
Bing	4	0	0	0	20	2	16	0	7
Facebook	14	0	0	0	5	82	0	0	8
Google	32	0	2	0	2	2	247	0	12
UoP site	11	0	0	0	0	3	0	90	1
YouTube	30	0	0	0	4	0	20	0	198

applications except for Google applications (i.e., Gmail, YouTube, and Google search engine). Out of the total tested samples, it was observed that Yahoo mail, Amazon, and BBC news recorded optimal accuracy and that University of Plymouth website recorded lowest rate of false-negatives. The reason for obtaining high classification accuracy for these applications could be attributed to the fact that they have unique behaviours, distinct from the others. On the contrary, Google applications (Gmail, YouTube, and Google) performed the worst in terms of classification, as they belong to the same company and they were misclassified as each other. Overall, the accuracy of all applications was satisfactorily high.

5. Conclusion

This study proposed a novel set of features based on interarrival times between packets and flows, most specifically burstiness and idle time, for application identification. From the experimental results, the proposed features outperform the traditional features that were proposed by previous studies; furthermore, higher accuracy is achieved when combining both proposed and traditional features. A modified tcptrace tool was used to extract the new features, and a C5.0 classifier was used to detect applications based on real data collection. Overall, accuracy was more than 79%; however, some applications resulted in low accuracies such as Google, Gmail, and YouTube as they belong to the same owner. One of the limitations of this work was that constructing the truth table for application membership of flows relied on IP addresses and DNS. Unfortunately, due to the underlying CDN hosting of different applications, this classification led to inaccurate results. Moreover, the data traffic was collected at the University of Plymouth and from managed computers owned by the university and included many web-based services that introduced noise in the collected data. On the contrary, comparing results with previous studies that reported high accuracy, most had classified traffic either according to network protocols such as FTP, IMAP, and HTTP or as per the application class such as e-mail, P2P, and streaming. Protocol, port number, or class-based traffic is generally easy to identify, and hence, the reported accuracy is also usually high. However, reviewing literature, very few studies such as [46] have classified traffic according to modern applications (i.e., Facebook and Google services). Moreover, these studies relied on the DPI method for labelling traffic in which they used a supervised approach for traffic classification. DPI had been considered trustworthy by such studies [47, 48] until 2009 where a study in [49] claimed that libraries of DPI are unreliable. Nowadays, current applications are web-based and become almost encrypted; therefore, the DPI method cannot cope with modern services as it is based on matching payload patterns, IP address, and port number [46]. Future work will primarily consider larger datasets with different types of applications and more end users in order to fully investigate the performance of the proposed work. Moreover, future work will also focus on recognizing new applications that emerge over time by applying the proposed

method. Finally, a more accurate approach for labelling the traffic should also be incorporated to ensure the robustness of the method.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] I. N. Bermudez, M. Mellia, M. M. Munafo, R. Keralapura, and A. Nucci, "DNS to the rescue: discerning content and services in a tangled web," in *Proceedings of the 12th ACM SIGCOMM Conference on Internet Measurement, (IMC'12)*, pp. 413–426, Vienna, Austria, November 2012.

[2] A. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Proceedings of the Passive and Active Measurement Workshop*, Boston, MA, USA, March 2005.

[3] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, and K. Hanssgen, "A survey of payload-based traffic classification approaches," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1135–1156, 2014.

[4] S. Valenti, D. Rossi, A. Dainotti, A. Pescapè, A. Finamore, and M. Mellia, "Reviewing traffic classification," *Data Traffic Monitoring and Analysis*, vol. 7754, pp. 123–147, 2013.

[5] Y. Wang, Y. Xiang, J. Zhang, W. Zhou, G. Wei, and L. T. Yang, "Internet traffic classification using constrained clustering," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 11, pp. 2932–2943, 2014.

[6] A. Hajjar, J. Khalife, and J. Díaz-Verdejo, "Network traffic application identification based on message size analysis," *Journal of Network and Computer Applications*, vol. 58, pp. 130–143, 2015.

[7] T. Bujlow, T. Riaz, and J. M. Pedersen, "A method for classification of network traffic based on C5.0 machine learning algorithm," in *Proceedings of the International Conference on Computing, Networking and Communications (ICNC)*, pp. 237–241, Maui, HI, USA, March 2012.

[8] H. Oudah, B. Ghita, and T. Bakhshi, "Network application detection using traffic burstiness," in *Proceedings of the World Congress on Internet Security (WorldCIS-2017)*, Cambridge, UK, December 2017.

[9] H. Oudah, B. Ghita, and T. Bakhshi, "A novel features set for internet traffic classification using burstiness," in *Proceedings of the (ICISSP 5th International Conference on Information Systems Security and Privacy)*, pp. 397–404, Prague, Czech Republic, July 2019.

[10] N. Al Khater and R. E. Overill, "Network traffic classification techniques and challenges," in *Proceedings of the 2015 Tenth International Conference on Digital Information Management (ICDIM)*, pp. 43–48, Jeju, South Korea, October 2015.

[11] M. S. Joe Touch, E. Lear, A. Mankin et al., *Service Name and Transport Protocol Port Number Registry*, IANA, Playa Vista, CA, USA, 2016, http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.

[12] R. Zou, T. Xu, and H. Hou, "An enhanced Netflow data collection system," in *Proceedings of the 2012 Second International Conference on Instrumentation, Measurement, Computer, Communication and Control (IMCCC)*, pp. 508–511, Harbin, China, December 2012.

[13] A. Boukhtouta, S. A. Mokhov, N.-E. Lakhdari, M. Debbabi, and J. Paquet, "Network malware classification comparison using DPI and flow packet headers," *Journal of Computer Virology and Hacking Techniques*, vol. 12, no. 2, pp. 69–100, 2016.

[14] T. Bujlow, V. Carela-Español, and P. Barlet-Ros, "Independent comparison of popular DPI tools for traffic classification," *Computer Networks*, vol. 76, pp. 75–89, 2015.

[15] A. Bashir, C. Huang, B. Nandy, and N. Seddigh, "Classifying P2P activity in netflow records: a case study on BitTorrent," in *Proceedings of the 2013 IEEE International Conference on Communications (ICC)*, pp. 3018–3023, Budapest, Hungary, June 2013.

[16] B. Park, Y. Won, J. Chung, M.-S. Kim, and J. W.-K. Hong, "Fine-grained traffic classification based on functional separation," *International Journal of Network Management*, vol. 23, no. 5, pp. 350–381, 2013.

[17] A. Vlăduţu, D. Comăneci, and C. Dobre, "Internet traffic classification based on flows' statistical properties with machine learning," *International Journal of Network Management*, vol. 27, no. 3, article e1929, 2017.

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

[19] R. Alshammari and A. N. Zincir-Heywood, "How robust can a machine learning approach be for classifying encrypted VoIP?," *Journal of Network and Systems Management*, vol. 23, no. 4, pp. 830–869, 2015.

[20] A. Ulliac and B. V. Ghita, "Non-intrusive identification of peer-to-peer traffic," in *Proceedings of the 2010 Third International Conference on Communication Theory, Reliability, and Quality of Service*, pp. 175–183, Athens, Greece, June 2010.

[21] A. W. Moore, D. Zuev, A. W. Moore, and D. Zuev, "Internet traffic classification using bayesian analysis techniques," *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1, p. 50, 2005.

[22] 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, 2007.

[23] A. Este, F. Gringoli, and L. Salgarelli, "Support vector machines for TCP traffic classification," *Computer Networks*, vol. 53, no. 14, pp. 2476–2490, 2009.

[24] G. Y. Lazarou, J. Baca, V. S. Frost, and J. B. Evans, "Describing network traffic using the index of variability," *IEEE/ACM Transactions on Networking*, vol. 17, no. 5, pp. 1672–1683, 2009.

[25] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement—IMC '04*, pp. 135–148, Boston, MA, USA, July 2004.

[26] S. Shakkottai, N. Brownlee, and K. C. Claffy, "A study of burstiness in TCP flows," *Lecture Notes in Computer Science*, vol. 3431, pp. 13–26, 2005.

[27] R. Liston, S. Srinivasan, and E. Zegura, "Diversity in DNS performance measures," in *Proceedings of the Second ACM SIGCOMM Workshop on Internet Measurment—IMW '02*, vol. 19, Marseille, France, November 2002.

[28] J. Jung, E. Sit, H. Balakrishnan, and R. M. Morris, "DNS performance and effectiveness of caching," in *Proceedings of*

the First ACM SIGCOMM Workshop on Internet Measurement Workshop—IMW '01, vol. 10, no. 5, pp. 589–603, San Francisco, CA, USA, November 2001.

[29] D. Wessels, "Is your caching resolver polluting the internet?," in *Proceedings of the ACM SIGCOMM Workshop on Network Troubleshooting Research, Theory and Operations Practice Meet Malfunctioning Reality—NetT '04*, pp. 271–276, Portland, OR, USA, September 2004.

[30] D. Whyte, E. Kranakis, and P. Van Oorschot, "DNS-based detection of scanning worms in an enterprise network," in *Proceedings of the 12th Annual Network And Distributed System Security Symposium*, vol. 1, pp. 1–17, San Diego, CA, USA, January 2005.

[31] D. Plonka and P. Barford, "Flexible traffic and host profiling via DNS rendezvous," in *Proceedings of the SATIN*, Teddington, UK, April 2011.

[32] P. Foremski, C. Callegari, and M. Pagano, "DNS-Class: immediate classification of IP flows using DNS," *International Journal of Network Management*, vol. 24, no. 4, pp. 272–288, 2014.

[33] N. F. Huang, C. C. Li, C. H. Li, C. C. Chen, C. H. Chen, and I. H. Hsu, "Application identification system for SDN QoS based on machine learning and DNS responses," in *Proceedings of the 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 407–410, Seoul, South Korea, September 2017.

[34] G. Mamidisetti and G. T. Varma, "Performance issues of internet protocol versions," *International Journal of Soft Computing and Engineering*, vol. 3, no. 6, pp. 30–32, 2014.

[35] M. Janbeglou, H. Naderi, and N. Brownlee, "Effectiveness of DNS-based security approaches in large-scale networks," in *Proceedings of the 28th International Conference on Advanced Information Networking and Applications Workshops*, pp. 524–529, Victoria, Canada, May 2014.

[36] M. Trevisan, I. Drago, M. Mellia, and M. M. Munafo, "Towards web service classification using addresses and DNS," in *International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 38–43, Paphos, Cyprus, September 2016.

[37] L. M. Torres, E. Magana, M. Izal, and D. Morato, "A popularity-aware method for discovering server IP addresses related to websites," in *Proceedings of the Global Information Infrastructure Symposium—GIIS 2013*, Trento, Italy, October 2013.

[38] T. Bakhshi and B. Ghita, "Traffic profiling: evaluating stability in multi-device user environments," in *Proceedings of the 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 731–736, Crans-Montana, Switzerland, May 2016.

[39] A. N. Mahmood, C. Leckie, and P. Udaya, "An efficient clustering scheme to exploit hierarchical data in network traffic analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 6, pp. 752–767, 2008.

[40] D. Tammaro, S. Valenti, D. Rossi, and A. Pescapé, "Exploiting packet-sampling measurements for traffic characterization and classification," *International Journal of Network Management*, vol. 22, no. 6, pp. 451–476, 2012.

[41] S. Ostermann, "tcptrace—Official Homepage," 2016, http://www.tcptrace.org/.

[42] B. V. Ghita, S. M. Furnell, B. M. Lines, and E. C. Ifeachor, "Endpoint study of internet paths and web pages transfers," *Campus-Wide Information Systems*, vol. 20, no. 3, pp. 90–97, 2003.

[43] R. Krzanowski, "Burst (of packets) and burstiness," in *Proceedings of the 66th IETF Meeting*, Montreal, Quebec, Canada, July 2006.

[44] T. Bakhshi and B. Ghita, "User traffic profiling," in *Proeedings of the 2015 Internet Technologies and Applications (ITA)*, pp. 91–97, Wrexham, UK, September 2015.

[45] R. Hofstede, P. Celeda, B. Trammell et al., "Flow monitoring explained: from packet capture to data analysis with NetFlow and IPFIX," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[46] Z. Aouini, A. Kortebi, Y. Ghamri-Doudane, and I. L. Cherif, "Early classification of residential networks traffic using C5.0 machine learning algorithm," in *Proceedings of the Wireless Days (WD)*, pp. 46–53, Dubai, UAE, April 2018.

[47] L. Bernaille, R. Teixeira, L. Bernaille et al., *Early Recognition of Encrypted Applications to Cite This Version*, Springer, Berlin, Germany, 2007.

[48] R. Alshammari and A. N. Zincir-Heywood, "Machine learning based encrypted traffic classification: identifying SSH and Skype," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, Canada, July 2009.

[49] G. Maier, A. Feldmann, V. Paxson, and M. Allman, "On dominant characteristics of residential broadband internet traffic," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference—IMC '09*, p. 90, Chicago, IL, USA, November 2009.