2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress

# Early Stage Internet Traffic Identification Using Data Gravitation Based Classification

Lizhi Peng*, Haibo Zhang†, Bo Yang*, Yuehui Chen* and Xiaoqing Zhou*
*Shandong Provincial Key Laboratory of Network Based Intelligent Computing,
University of Jinan, Jinan, 250022, P. R. China
yangbo@ujn.edu.cn
†Department of Compute Science, University of Otago, Dunedin, New Zealand
haibo@cs.otago.ac.nz

*Abstract*—**Traditional machine learning traffic identification techniques usually use the features of a whole Internet flow, which makes such techniques few sense in engineering practices. Therefore, recent years, an increasing number of researchers turn to build effective machine learning models to identify traffics with the few packets at the early stage. In this paper, data gravitation based classification (DGC) model, a supervised learning approach inspired by Newton's universal gravitation law, is applied for early stage traffic identification. In the empirical study, two open data sets and a data set collected in our campus network are employed. Eight widely used supervised learning algorithms are compared with our approach in the identification experiments. Accuracy and Cohen's kappa coefficient are applied to evaluate the performances of compared methods. The experimental results suggest that DGC outperformed the other algorithms for most cases considering both of accuracy and kappa. Thus, DGC is effective for early stage traffic identification.**

*Index Terms*—**Traffic identification, Data gravitation, Machine learning.**

## I. Introduction

Traffic identification, a fundamental technique for network management and security, has attracted wide interest in the past decade. Existing traffic identification approaches commonly fall into three categories: port based identification, deep payload inspection, and machine learning based identification. As traditional approaches, the former two have fatal drawbacks when confronting modern Internet traffic, e.g. dynamic port allocation and encrypted traffic. Machine learning based traffic identification, with abilities of identifying Internet traffic in an intelligent way, has been considered as the most promising approach. However, most existing machine learning based traffic identification techniques extract features from a whole Internet flow [1], [2], [3]. Moore et al. [4] developed a widely method for Internet traffic feature extraction in 2005. They extract 248 statistical features, such as RTT, maximum, minimum and average packet size, based on a 24-hour full-duplex traffic trace. Unfortunately, in real circumstances, it makes little sense to recognize Internet traffics when they have ended, unless there is evidence to demonstrate that some traffics have repeated patterns. Hence, it is more desirable to identify Internet traffics accurately in their early stage so that we can apply subsequent management and security policies. Therefore, some researchers have turned to find effective models which are able to identify Internet traffics at their early stage, which makes early stage identification a hot topic in traffic identification [5].

Even though the possibility of early stage traffic identification has been demonstrated by Qu et al. in [6], how to accurately identify traffics **using only a few early stage packets** still remains a challenging problem. Bernaille et al. presented a famous early stage traffic identification technique in 2006 [7]. They use the size of the first few data packets of each TCP flow as the features, and by applying K-means clustering technique, they got high identification rates for 10 types of application traffics. Este et al. demonstrated in 2009 [8] that early stage packets of an Internet flow carry enough information for traffic classification. They analyzed round trip time (RTT), packet size, inter-arrival time (IAT) and packet direction of early stage packets and found that packet size is the most effective feature for early stage classifications. Huang et al. investigated the early stage application characteristics and used them for effective classification [9]. Recently, they extracted early stage traffic features by analyzing the negotiation behaviors of different applications. They use packet size (PS) and inter packet time (IPT) of the first 10 packets for some classifiers, while for other classifiers, they use average and standard deviation values of PS and IPT of the early packets. They applied these features in machine learning based classifiers and demonstrated high accuracy on traffic identification [10]. Hullár et al. proposed an automatic machine learning based method that consumes limited computational and memory resources for P2P traffic identification at early stage [11]. Dainotti et al. constructed high effective hybrid classifiers and used a hybrid feature extraction method for early stage traffic classification [12]. Nguyen et al. used statistical features derived from sub-flows for timely identification of VoIP traffics [13]. They extended the concept of early stage to "timely", since a sub-flow refers to a small number of most recent packets taken at any point in a flow's lifetime. Rizzi et al. proposed an efficient neuro-fuzzy system for early stage traffic identification [14].

In [15], [16] we presented a new classification model named the data gravitation based classification (DGC) model. DGC simulates the Newton's law of universal gravitation. It refers to a data instance in the data space as a data "particle", and considers the type of "gravitation" between any two data particles in the computation. This gravitation is directly proportional to the product of the "masses" of two data particles and is inversely proportional to the square of the distance between the data particles. By comparing the gravitation from different data classes in the training set, DGC can effectively classify a testing data instance in a simple manner.

In this paper, we set out to create an effective early stage traffic identification scheme by applying our DGC model. Three network traffic data sets, including two open data sets, are selected for the empirical study. The payload sizes of the first six packets of each flow, along with some simple derived features such as min/max payload size and standard deviation, are extracted as the early stage features. The total accuracy and Cohen's kappa coefficient, are used to evaluate the performances of the proposed scheme. We compare our approach with eight classical supervised learning algorithms in trace-based experiments, and results give an encouraging pattern: comparing with the other methods, our proposal is able to improve 0.1 to 74 percent with regard to kappa, and 0.02 to 12.64 percents with regard to accuracy.

The rests of the paper are organized as follows: Section II introduces the fundamental DGC theory. We introduce the characteristics of the selected data sets and features in Section III and IV, respectively. The experimental settings including the comparing methods and the performance measurements are given in Section V. The details of experimental results and analysis are given in Section VI, and we also give some in-depth discussion in this section. Finally, the paper is concluded in Section VII.

## II. A BRIEF OVERVIEW ON THE DGC MODEL

### *A. Classification using data gravitation*

DGC [15] is a classification model that simulates the Newton's law of universal gravitation. The basic idea of DGC is described as follows: For a class-unknown data instance, i.e. a testing instance, DGC firstly computes the gravitation from each class in the training set, and then compares the gravitation of different classes. The testing instance is classed into the class that exhibits the largest gravitation.

DGC is supported by a set of theories that includes a number of concepts and lemmas. All of the concepts and lemmas are designed to simulate the gravitational actions between two particles in the real physical world. In the following, we briefly present some concepts and lemmas that are necessary to understand our DGC-based traffic identification scheme. For more details on DGC, please refer to our previous work [15].

**Concept (Data Particle):** Data particle simulates the concept of physical particle in universe. A number of instances with a certain relationship form a data particle, where the relationship refers to the geometrical neighborhood of these instances. A data particle with only one instance is called atomic data particle. Data particle has two basic properties: data mass and centroid. Data mass is the number of instances in the data particle, and data centroid is the geometrical center of these instances.

Following the concepts of data mass and centroid, a data particle can be denoted using a pair expression $< m, \mathbf{x} >$, where $m$ is its mass, and $\mathbf{x}$ is its centroid. When taking the class property (feature $y$) into account, the data particle could be described as a triple expression $< m, \mathbf{x}, y >$. It should be noted that instances from different classes cannot be put into a single particle. Moreover, a testing instance is an atomic particle because its class property is unknown.

**Concept (Data Gravitation):** Data gravitation is a simulation of physical gravitation in data spaces. Any pair of data particles in a data space have a kind of interacting force, namely data gravitation, and the gravitation is the direct ratio to the product of data mass of the two data particles, and reverse ratio to the square of distance between them, which can be computed by equation:

$$F = \frac{m_1 m_2}{r^2} \tag{1}$$

where $F$: Gravitation between two data particles. $m_1$: Data mass of data particle 1. $m_2$: Data mass of data particle 2. $r$: The Euclidean distance between the two data particle in data space. It can be seen that such a form of data gravitation computation is the same as Newton's law of universal gravitation. However, data gravitation is a kind of scalar without direction, which make it different from physical forces.

For a testing instance (an atomic particle) $P$, the gravitation from the particles of a single class $c$ can be superposed, and this is the superposition principle:

**Lemma (Superposition Principle):** Suppose $p_1, p_2, ..., p_m$ are $m$ data particles in a data space, and they belong to the same data class $c$. The gravitations they act on the testing data particle $P$ are $F_1, F_2, ..., F_m$, and then the composition of gravitation is:

$$F(c) = \sum_{i=1}^{m} F_i \tag{2}$$

On the other hand, the gravitation from different classes can be compared, which forms the the simple classification principle of the DGC model.

**Lemma (Classification Principle):** Suppose $c_1, c_2$ are two data classes in a training data set. For a given testing instance $P$, the gravitation that data particles in $c_1$ acts on $P$ is $F(c_1)$, and $F(c_2)$ is the gravitation that data particles in $c_2$ acts on $P$. If $F(c_1) > F(c_2)$, then the degree $P$ belongs to $c_1$ is stronger than that to $c_2$.

Figure 1 shows the principle of classification with a binary data set.

For a multi-class data set with $k$ classes, if $l_1, l_2, ..., l_k$ are the numbers of data particles, respectively, then for a testing instance $P$, the gravitation that data class $c_i$ act on it is given by:

$$F(c_i) = \sum_{j=1}^{l_i} \frac{m_{ij}}{|\mathbf{x}_{ij} - \mathbf{x}|^2} \tag{3}$$

where $m_{ij}$ is the data mass of data particle $j$ in data class $i$, $\mathbf{x}_{ij}$ is its centroid, and $\mathbf{x}$ is the centroid of $P$.

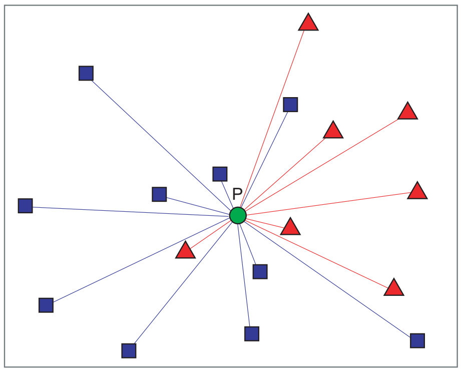

Fig. 1. Classification based on data gravitation. The strength of gravitation determine which class a testing instance belongs to. The blue squares denote data particles in class $c_1$, the red triangles denote data particles in class $c_2$.

If $F(c_{i'}) = max\{F(c_1), F(c_2), ..., F(c_k)\}$, then according to lemma 2, the test data element belongs to data class $c_{i'}$.

## B. Weighting features

DGC uses weight to measure the importance of a feature (input). Suppose there are $n$ features in target problem feature set, and every feature has a weight value, so all feature weights form a $n$-expression: $< w_1, w_2, ..., w_n >$, We mark it as the feature weight vector $\mathbf{w}$. As the features have been weighted, the distance between two data particles in the data space is not their Euclidean distance in the data space, but the weighted distance:

$$r' = \sqrt{\sum_{i=1}^{n} w_i(x_{1i} - x_{2i})^2} \tag{4}$$

The calculation of the data gravitation is also changed as:

$$F = \frac{m_1 m_2}{r'^2} \tag{5}$$

## C. Feature weight optimization using PSO

We use particle swarm optimization (PSO) [17], [18] to optimize the feature weights. PSO has been proven as an effective random optimization algorithm in many studies. Comparing with genetic algorithm (GA), PSO has no crossover and mutation operations, each particle moves according its velocity and the global sharing information. In GA, any pair of individuals (chromosomes) may exchange their information by the crossover operation. For PSO, all particles only receive the information from the global best particle and the best particle of the current population [19]. In many applications, PSO is proved to converge in fewer generations [20], or have better computational efficiency [21] than GA. Yet, GA also has advantages in comparing with PSO, and further comparison between the algorithms is beyond the scope of the paper.

PSO conducts searches by using a population of particles that correspond to individuals in an evolutionary algorithm. A population of particles is randomly generated initially. Each particle represents a potential solution and has a position that is represented by position vector $\overline{\mathbf{x}}_i$. A swarm of particles moves through the problem space, and the velocity of each particle is represented by velocity vector $\overline{\mathbf{v}}_i$. At each time step, a function $f_i$ representing a quality measure is calculated by using $\overline{\mathbf{x}}_i$ as the input. Each particle monitors its own best position, which is associated with the best fitness achieved by a particle in vector $\overline{\mathbf{p}}_i$. Furthermore, the best position among all particles obtained in the population is tracked as $\overline{\mathbf{p}}_g$. The best position signifies the best global solution obtained so far. In addition to the global version, another PSO version monitors the best position among all the topological neighbors of a particle (for more information on neighborhood topology, refer to [22] and [23]).

At each time step $t$, by using the individual best position, $\overline{\mathbf{p}}_i(t)$, and the global best position, $\overline{\mathbf{p}}_g(t)$, a new velocity for particle $i$ is updated by

$$\overline{\mathbf{v}}_i(t+1) = \overline{\mathbf{v}}_i(t) + c_1\phi_1(\overline{\mathbf{p}}_i(t) - \overline{\mathbf{x}}_i(t)) + c_2\phi_2(\overline{\mathbf{p}}_g(t) - \overline{\mathbf{x}}_i(t)) \tag{6}$$

where $c_1$ and $c_2$ are positive constant and $\phi_1$ and $\phi_2$ are uniformly distributed random numbers in [0,1]. The term $\overline{\mathbf{v}}_i$ is limited to the range of $\pm\mathbf{v}_{max}$. If the velocity violates this limit, it is set to its proper limit. Changing velocity this way enables the particle $i$ to search around its individual best position, $\overline{\mathbf{p}}_i$, and global best position, $\overline{\mathbf{p}}_g$. Based on the updated velocities, each particle changes its position according to the following equation:

$$\overline{\mathbf{x}}_i(t+1) = \overline{\mathbf{x}}_i(t) + \overline{\mathbf{v}}_i(t+1) \tag{7}$$

Based on Equations (6) and (7), the population of particles tend to cluster together with each particle moving in a random direction. This may result in premature convergence in many problems. An effective method to avoid premature convergence is to update the velocity by the following formula[22]:

$$\overline{\mathbf{v}}_i(t+1) = \chi(\omega\overline{\mathbf{v}}_i(t) + c_1\phi_1(\overline{\mathbf{p}}_i(t) - \overline{\mathbf{x}}_i(t)) + c_2\phi_2(\overline{\mathbf{p}}_g(t) - \overline{\mathbf{x}}_i(t))) \tag{8}$$

where two new parameters, $\chi$ and $\omega$, are also real numbers. The parameter $\chi$ controls the magnitude of $\overline{\mathbf{v}}$, whereas the inertia weight $\omega$ weights the magnitude of the old velocity $\overline{\mathbf{v}}_i(t)$ in the calculation of the new velocity $\overline{\mathbf{v}}_i(t+1)$.

In DGC model, we use $n$ real values range from 0 to 1 to form a real value sequence, and the sequence is a PSO particle. If the population size is $s$, then we generate $s$ such sequences which make up the population for each PSO iteration. Here, $n$ is the feature number of the target problem. Therefore, a single particle is an alternative solution of the feature weights $< w_1, w_2, ..., w_n >$ defined in section II-B. And the fitness evaluating method will be illustrated in the next subsection in detail. By this mean, PSO algorithm is used to optimize the feature weights of the DGC model.

## III. Data sets

We select two sets of open network traffic traces, and a set of traces collected in our campus network for our study. The characteristics of the selected traces are depicted in Table I.

TABLE I
CHARACTERISTICS OF THE SELECTED NETWORK TRAFFIC TRACES

| Auckland II traces | | | UNIBS traces | | | UJN traces | | |
|---|---|---|---|---|---|---|---|---|
| Type | # of instances | Bytes | Type | # of instances | Bytes | Type | # of instances | Bytes |
| ftp | 251 | 136241 | bittorrent | 3571 | 6393487 | Web Browser | 11890 | 58025350 |
| ftp-data | 463 | 5260804 | edonkey | 379 | 241587 | Chat | 11478 | 60212804 |
| http | 23721 | 139421961 | http | 25729 | 107342346 | Cloud Disk | 1563 | 109552924 |
| imap | 193 | 86455 | imap | 327 | 860226 | Live Update | 2169 | 28759962 |
| pop3 | 498 | 98699 | pop3 | 2473 | 4292419 | Stream Media | 810 | 785556 |
| smtp | 2602 | 1230528 | skype | 801 | 805453 | Mail | 803 | 2092862 |
| ssh | 237 | 149502 | smtp | 120 | 43566 | P2P | 326 | 2521089 |
| telnet | 37 | 21171 | ssh | 23 | 39456 | Other | 1408 | 3635558 |

### A. *Auckland II traffic traces*

Auckland II is a collection of long GPS-synchronized traces taken using a pair of DAG 2 cards at the University of Auckland which is available at [24]. There are 85 trace files which were captured from November 1999 to July 2000. Most traces were targeted at 24 hour runs, but hardware failures have resulted in most traces being significantly shorter. We selected two trace files captured at Feb 14 2000 (20000214-185536-0.pcap and 20000214-185536-1.pcap) for our study.

Since the application payloads were not recorded in Auckland II, DPI tools are invalid to obtain ground truths. The only way to pick out the original application type is using port numbers. In this study, we only accounted TCP case since TCP is the predominant transport layer protocol. Each flow is thus assigned to the class identified by the server port. We selected eight main types from Auckland II traces and filtered mouse flows with no more than six non-zero packets.

### B. *UNIBS traffic traces*

UNIBS is another opening traffic traces developed by Prof. F. Gringoli and his research team, available at [25]. They developed a useful system namely GT [26] to application ground truths of captured Internet traffics. The traces were collected on the edge router of the campus network of the University of Brescia on three consecutive working days (Sept 30, Oct 1 and Oct 2 2009). They are composed of traffic generated by a set of twenty workstations running the GT client daemon. Traffics were collected by running Tcpdump [27] on the Faculty's router. We again use TCP flows in this data set for our study. By using GT, UNIBS traces recorded the application information of each captured flow. We can get the application ground truths by both TCP port numbers and GT records. We also chose eight main types in UNIBS for our study. Different from Auckland II traces, there are two popular P2P applications in this data set, bittorrent and edonkey, recorded by GT. Skype is also selected as an import Internet application. Flows with no more than six non-zero payload packets are also filtered.

### C. *UJN traffic traces*

The third data set is collected in a laboratory network of University of Jinan using Traffic Labeler (TL) [28]. TL system captures all user socket calls and their corresponding application process information in the user mode on a Windows host, and sends the information to an intermediate NDIS driver in the kernel mode. The intermediate driver writes application type information on the TOS field of the IP packets which match the 5-tuple. By this mean, each IP packet sent from the Windows host carries their application information. Therefore, traffic samples collected on the network have been labeled with the accurate application information and can be used for training effective traffic classification models. We deployed 10 TL instances on Windows user hosts in the laboratory network of Provincial Key Laboratory for Network Based Intelligent Computing. A mirror port of the uplink port of the switch was set, and a data collector was deployed at the mirror port. The deployed TL instances ran at work hours every day. The data collecting process lasted two days in May 2013. Again, flows with no more than six non-zero payload packets are also filtered.

## IV. FEATURES

- **Packet size:** Packet size is proven to be the most effective original packet level feature in early stage of traffics [8]. We use the packet sizes of the first six packets as the original early stage traffic features in this study, since we have proven that the first six packets are most effective for early stage feature extraction [29], [30]. And all statistical features are computed based on the packet sizes. We use the abbreviation $ps_i$ for the packet size of the $i$th packet.
- **Average:** The average is also known as the arithmetical mean, which is an extensively used statistic. This feature is calculated as follows:

$$avg = \sum_{i=1}^{n} ps_i \tag{9}$$

  And we use the abbreviation $avg$ for this feature.
- **Standard deviation:** The standard deviation shows how much variation or dispersion from the average exists. And the feature is defined as:

$$stdev = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(ps_i - avg)^2} \tag{10}$$

  where $n$ is the number of packets, i. e. 6 in this study. And we use the abbreviation $stdev$ for this feature.

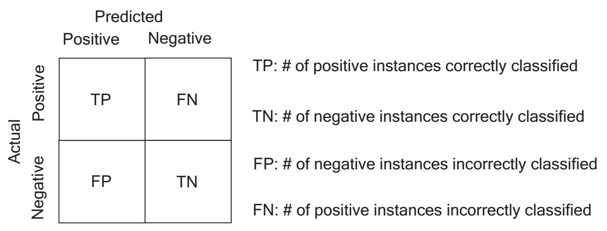
Predicted
Positive
Negative
Actual
Positive
Negative
TP
FN
FP
TN
TP: # of positive instances correctly classified
TN: # of negative instances correctly classified
FP: # of negative instances incorrectly classified
FN: # of positive instances incorrectly classified

Fig. 2. Confusion Matrix

- **Maximum and minimum:** The maximum and minimum payload size are also applied in the study, and we use the abbreviations of $max$ and $min$ respectively.

## V. Experimental settings

### A. Compared methods

We execute our identification experiments using eight well-known machine learning classifiers. We use Weka data mining software [31] as our experiment tool. All classifiers are run in Weka and all processed data sets are formatted into the Weka data file with the extension name of "arff". The classifiers we selected fall into five categories according to Weka:

- **Bayes:**Bayes classifiers are based on Bayes theorem, which is widely applied in many engineering areas. In this study, we choose Averaged 2-Dependence Estimators (A2DE) and Bayesian network (BayesNet) as Bayes classifiers.
- **Lazy:** A lazy learning algorithm just stores the training instances in the training phase, the real classification process is executed in the testing phase. k-Nearest Neighbor (KNN) is the most representative one, and we choose it for this type.
- **Rule:** As the name suggests, a rule based classifier extracts rules using a specific policy, e. g. probability and decision trees, and uses the rules to classify testing data. PART is selected for this category in this study.
- **Trees:** This refers to decision trees. A decision tree divides the target feature space hierarchically. Each division produces a node on the decision trees. A classification procedure is a procedure that goes from the root node to a specific leaf node on the tree. In this study, C4.5 and NBTree are selected for this category.
- **Functions:** Weka refers all classifiers based on specific functions to this category. We choose Logistic and SimpleLogistic for this category.

### B. Performance measures

The confusion matrix is the basis in measuring a classification task, wherein rows denote the actual class of the instances and the columns denote the predicted class. Fig. 2 shows a typical confusion matrix of a binary classification. We conduct many types of measures based on the confusion matrix to evaluate classifier performance. In this study, the following measures are used:

- **Accuracy:** Classification accuracy (Acc) is defined as the total proportion of all correctly classified instances:

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{11}$$

- **TPR:** True positive rate (TPR) is the most effective measure when we focus on the classification performance of a single class (positive). The measure is defined as the ratio between the number of the correctly classified positive instances and the total number of positive instances:

$$TPR = \frac{TP}{TP + FN} \tag{12}$$

- **Cohen's kappa coefficient:** Cohen's kappa coefficient [32] was originally used to measure the inter-rater agreement in medial diagnoses. By considering the change between the evaluations of two raters, Cohen's kappa is able to measure whether the agreement between the two raters occurs internally or by change. It is generally thought to be a more robust measure than simple percent agreement calculation. As a medical diagnosing procedure is essentially a classification task, Cohen's kappa is widely applied in many other type of classification tasks. For a classification task with $k$ classes, Cohen's kappa is defined as the following equation with regarding to the confusion matrix:

$$\kappa = \frac{n\sum_{i=1}^{k} n_{ii} - \sum_{i=1}^{k} n_{i.}n_{.i}}{n^2 - \sum_{i=1}^{k} n_{i.}n_{.i}} \tag{13}$$

Where $n$ the number of the instances, $n_{ii}$s are the numbers at the diagonal of the confusion matrix, $n_{i.}$ and $n_{.i}$ are the sums of the $i$th row and column, respectively. $\kappa$ owns a value range between -1 and 1. Typically, there are three flag values for $\kappa$: a value of -1 implies the total disagreement between the classifier predicted results and the actual results; a zero value indicates a random classification results; a value of 1 means the total agreement between the two groups of results.

### C. K-folder cross-validation

K-folder cross-validation is a popular and effective experimental validation procedure for classifying tasks. We use the standard 10-folder cross-validation in the experiments. Each traffic data set is uniformly divided into ten subsets, i.e., instances of each traffic type are uniformly distributed in all subsets. The training and testing processes of each data set are conducted ten times. For the $i$th time, the $i$th subset is used as the testing set, and the other four subsets are used as the training set.

## VI. Results and analysis

In this section, the detailed experimental results, including the total Acc and kappa results and the TPR of each class, are firstly given in subsection VI-A. In subsection VI-B, we present some analysis and discussions based on the experimental results.

TABLE II
OVERALL RESULTS OF THE THREE DATA SETS

| | | Auckland II | UNIBS | UJN |
|---|---|---|---|---|
| A2DE | Acc | 0.9937 | 0.9938 | 0.9702 |
| | kappa | 0.9771 | 0.9818 | 0.8951 |
| BayesNet | Acc | 0.9808 | 0.9829 | 0.9401 |
| | kappa | 0.9322 | 0.9508 | 0.8043 |
| Logisitc | Acc | 0.9287 | 0.9483 | 0.8440 |
| | kappa | 0.7448 | 0.8496 | 0.1569 |
| SimpleLogistic | Acc | 0.9297 | 0.9502 | 0.8444 |
| | kappa | 0.7457 | 0.8561 | 0.1560 |
| KNN | Acc | 0.9941 | 0.9930 | 0.9579 |
| | kappa | 0.9786 | 0.9794 | 0.8568 |
| PART | Acc | 0.9930 | 0.9931 | 0.9659 |
| | kappa | 0.9748 | 0.9797 | 0.8803 |
| C4.5 | Acc | 0.9916 | 0.9924 | 0.9635 |
| | kappa | 0.9694 | 0.9777 | 0.8708 |
| NBTree | Acc | 0.9929 | 0.9926 | 0.9651 |
| | kappa | 0.9741 | 0.9783 | 0.8766 |
| DGC | Acc | **0.9952** | **0.9945** | **0.9704** |
| | kappa | **0.9825** | **0.9839** | **0.8964** |

## A. *Results comparison*

The overall results of the three selected data sets are shown in Fig. II, wherein both of the Acc and kappa values of each algorithm are given. We marked the highest values in bold face to stress the best performed algorithm. A first look at the table suggests the performance advantages of DGC in comparing with the other algorithms: it hit all the highest values. The highest Acc values suggest the highest identification rates of DGC on all of the three selected traffic data sets, and the high kappa values show the high agreement between the predicted results of DGC and the actual results. It should be noticed that most of the compared algorithms are able to get fairly high identification rates on the three data sets: most of the Acc values of the Auckland II data set are higher than 0.98, and for the UNIBS and UJN data sets, most of the values are higher than 0.98 and 0.94, respectively. A fact behind the results is the effectiveness of the early stage features of the Internet traffic. The results imply again that identifying Internet traffic at the early stage using payload sizes and the derived features is reasonable.

For the purpose of further analysis, we provide the detailed TPR results of each class, which are shown in Fig. 3 to 5. The bars of most classes, especially for http (WebBrowser in the UJN data set) class, are higher than 0.9. It suggests that the selected algorithms can get high identification rates for most type of traffic classes. Http, with its predominant instance numbers, got extremely high identification rates. However, some types output relatively low identification rates: telnet of the Auckland II data set, edonkey and skype of the UNIBS data set, chat and other of the UJN data set. This reveals an important issue in traffic identifications—imbalanced identification. It can be seen from Table I that the instance number distributions are class imbalanced, i.e. the http instances are far more than that of the other classes in the three data sets. The experimental results suggest that some types of traffic instances are not easy to be picked up from the numerous major class instances. For example, the TPR values of skype in the UNIBS data set, all of which lower than 0.6, suggest that it is hard for a classifier to recognize such a traffic instance. The reason is partially relied on the instance number data in Table I: there are only 801 skype instances in the UNIBS data set. However, class imbalance is not totally responsible for the low TPR values of skype: ssh in the same data set, with only 23 instances, output high TPR values. From this view, the kappa metric is more important than the Acc metric. For example, despite of the relatively high Acc value of Logistic for the UJN data set, its TPR values of the most minor classes are very low, leading to the extremely low kappa value.

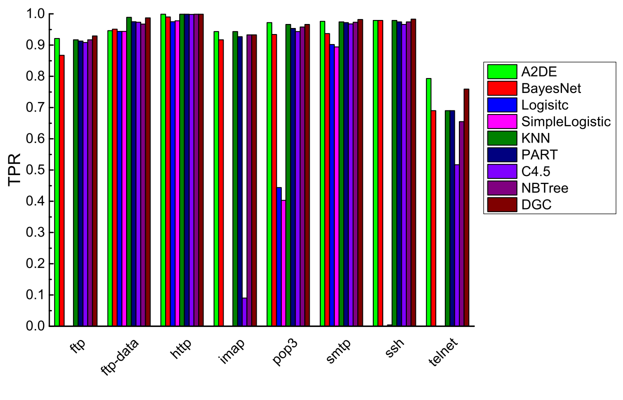

Fig. 3. TPR results of the Auckland II data set

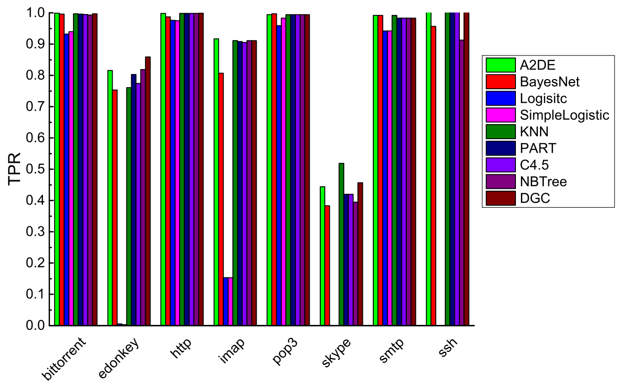

Fig. 4. TPR results of the UNIBS data set

DGC got the highest or the second highest TPR values for most cases. Although the differences between the TPR values of DGC and some other algorithms are not significant, the overall Acc and kappa results show the high performances of DGC.

## B. *Summaries and discussions*

Our experimental results show some interesting patterns, which can be summarized as follows:

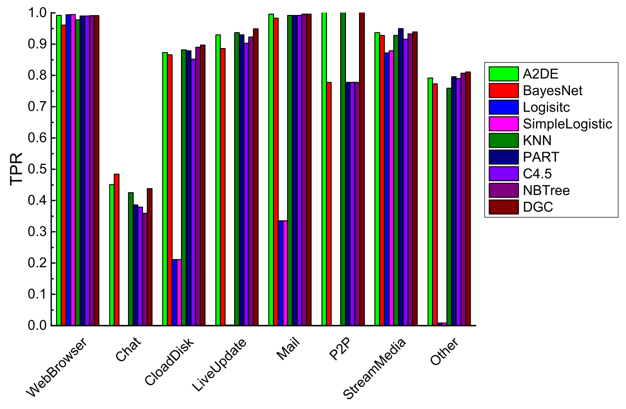

Fig. 5. TPR results of the UJN data set

- First of all, DGC shows high performances in early stage traffic identification. As can be seen, DGC gets all the highest accuracy and kappa values for all of the three data sets. The high accuracy values mean that DGC is able to achieve high total identification rates, and the high kappa values say that DGC is able to get good trade-off among different traffic types, especially for imbalanced traffic type distributions.
- The empirical study proved effectiveness of early stage traffic features. Only the first six packets of each flow are used to extract the features, and most algorithms is able to get relatively high identification accuracies by such features.
- For a early stage traffic identification method, performance evaluation using Cohen's kappa coefficient is as important as the total identification rate, as kappa is a measurement to evaluate the tade-off abilities of a method. In our experiments, some classifiers show high performances for the accuracy measurement, but they do not perform well for the kappa measurement. e. g. Logistic gets a considerably high accuracy value for the UJN data set. However, its kappa value of this data set is very low.

## VII. Conclusions

In this paper, we try to build an effective early stage traffic identification approach using data gravitation based classification (DGC) model. Three traffic data sets include two opening data sets are selected for the experimental evaluations. We extract features using the first six packets of each flow, which occurred at the early stage. Eight classical classification algorithms are applied for experimental comparisons. According to the experimental results, we conclude that DGC is effective for early stage traffic identification. As can be seen from the experimental results that DGC outperforms the other algorithms for most experiment cases. Furthermore, DGC does not only get high total identification rates, but also show good trade-off among different traffic types. And this is very important for traffic identification, especially for the cases with an imbalanced data distribution.

## Acknowledgment

This research was partially supported by the National Natural Science Foundation of China under grant No. 61472164, No. 61402475, No. 61373054, No. 61173078, No. 61203105, and No. 61173079, the Provincial Natural Science Foundation of Shandong under grant No. ZR2012FM010, No. ZR2011FZ001, and No.ZR2012FQ016.

## References

[1] A. Este, F. Gringoli, and L. Salgarelli, Support Vector Machines for TCP traffic classification, Computer Networks, 53(2009), pp. 2476-2490.
[2] W. Li, and A. W. Moore, A Machine Learning Approach for Efficient Traffic Classification, in Proceedings of IEEE MASCOTS'07, pp. 310-317, 2007.
[3] A. W. Moore, and D. Zuev, Internet Traffic Classification Using Bayesian Analysis Techniques, in ACM SIGMETRICS'05, pp. 50-60. 2005.
[4] A. W. Moore, D. Zuev, and M. Crogan, Discriminators for use in flow-based classification, Intel Research Tech. Rep. 2005.
[5] A. Dainotti, A. Pescapé, K. C. Claffy, Issues and future directions in traffic classification, IEEE Network, 26(1)(2012), pp. 35-40,
[6] B. Qu, Z. Zhang, L. Guo, et al., On accuracy of early traffic classification. in: Proc. IEEE 7th International Conference on Networking, Architecture and Storage (NAS), IEEE press, pp. 348-354, 2012.
[7] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, Traffic Classification On The Fly, in ACM SIGCOMM'06, pp. 23-26, 2006.
[8] A. Este, F. Gringoli, and L. Salgarelli, On the Stability of the Information Carried by Traffic Flow Features at the Packet Level, in ACM SIGCOMM'09, pp. 13-18, 2009.
[9] N. Huang, G. Jai, H. Chao, Early identifying application traffic with application characteristics, in: Proc. IEEE Int. Conference on Communications (ICC'08), Beijing, China, pp. 5788-5792, 2008.
[10] N. Huang, G. Jai, H. Chao, et al. Application traffic classification at the early stage by characterizing application rounds, Information Sciences, 232(20)(2013), pp. 130-142.
[11] B. Hullár, S. Laki, A. Gyorgy, Early identification of peer-to-peer traffic, in: Proc. 2011 IEEE International Conference on Communications (ICC), IEEE, pp. 1-6, 2011.
[12] A. Dainotti, A. Pescapé, C. Sansone, Early classification of network traffic through multi-classification, Lecture Notes on Computer Science, 6613(2011), pp. 122-135.
[13] T. T. T. Nguyen, G. Armitage, P. Branch, et al., Timely and continuous machine-learning-based classification for interactive IP traffic. IEEE/ACM Transactions on Networking (TON), 20(6)(2012), pp. 1880-1894,
[14] A. Rizzi, S. Colabrese, A. Baiocchi, Low complexity, high performance neuro-fuzzy system for Internet traffic flows early classification. in: Proc. 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC), IEEE press, pp. 77-82, 2013.
[15] L. Peng, B. Yang, Y. Chen, and A. Abraham, Data gravitation based classification, Information Sciences, 179(2009), pp. 809-819.
[16] L. Peng, H. Zhang, B. Yang, et al., A New Approach for Imbalanced Data Classification Based on Data Gravitation, Information Sciences, 288(20)(2014), pp. 347-373.
[17] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in Proceedings of IEEE International Conference on Neural Networks, pp. 1942-1948, 1995.
[18] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, Y. Nakanishi, A particle swarm optimization for reactive power and voltage control considering voltage security assessment, IIEEE TRANSACTIONS ON Power Syst, 15(2000), pp. 1232-1239.
[19] R. C. Eberhart, Y. Shi, Comparison between genetic algorithms and particle swarm optimization, Evolutionary Programming VII. Springer Berlin Heidelberg, pp. 611-616, 1998.
[20] S. Panda, N. P. Padhy, Comparison of particle swarm optimization and genetic algorithm for FACTS-based controller design, Applied soft computing, 8(4)(2008), pp. 1418-1427.

[21] R. Hassan, B. Cohanim, O. Weck, et al, A comparison of particle swarm optimization and the genetic algorithm, In Proceedings of the 1st AIAA multidisciplinary design optimization specialist conference, 2005.

[22] J. Kennedy, SmallWorlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance, in Proceedings of the 1999 Congress of Evolutionary Computation, 3, pp. 1931-1938. 1999.

[23] T. Krink, J. Vesterstrom, J. Riget. Particle Swarm Optimization with S-patial Particle Extension, in Proceedings of the Congress on Evolutionary Computation, 2002.

[24] Waikato Internet Traffic Storage (WITS). http://www.wand.net.nz/wits.

[25] UNIBS: Data sharing. http://www.ing.unibs.it/ntw/tools/traces/.

[26] F. Gringoli, L. Salgarelli, M. Dusi, et al. Gt: picking up the truth from the ground for internet traffic, ACM SIGCOMM Computer Communication Review, 39(5)(2009), pp. 12-18.

[27] Tcpdump/Libpcap. http://www.tcpdump.org.

[28] L. Peng, H. Zhang, B. Yang, et al., Traffic Labeller: Collecting Internet Traffic Samples with Accurate Application Information, China Communications, 11(1)(2014), pp. 82-91.

[29] L. Peng, B. Yang, Y. Chen, Effective Packet Number for Early Stage Internet Traffic Identification, Neurocomputing, 156(25)(2015), pp. 252-267.

[30] L. Peng, B. Yang, Y. Chen, Effectiveness of Statistical Features for Early Stage Internet Traffic Identification, International Journal of Parallel Programming, 44(1)(2016), pp. 181-197.

[31] Weka 3: Data Mining Software in Java, http://www.cs.waikato.ac.nz/ml/weka/.

[32] A. Ben-David, Comparison of classification accuracy using Cohens Weighted Kappa, Expert Systems with Applications, 32(2)(2008), pp. 825-832.