

Towards the On-line Identification of Peer-to-peer Flow Patterns

Antonio Nogueira, Paulo Salvador, Andre Couto, Rui Valadas
 University of Aveiro/Institute of Telecommunications
 Campus de Santiago, 3810-193 Aveiro, Portugal
 Email: {nogueira, salvador, andrecouto, rv}@ua.pt

Abstract—The number and variety of IP applications have hugely increased in the last few years. Among them, peer-to-peer (P2P) file-sharing applications have become very popular: more users are continuously joining such systems and more objects are being made available, seducing even more users to join. An accurate mapping of traffic to applications is important for a wide range of network management tasks. Besides, traditional mapping approaches have become increasingly inaccurate because many applications use non-default or ephemeral port numbers, use well-known port numbers associated with other applications, change application signatures or use traffic encryption. This paper proposes a framework to identify Internet applications that can be mainly used in situations where existing identification frameworks are not efficient or can not be used at all. The core block of the identification tool is based on neural networks and is able to identify different flow patterns generated by various Internet applications. Neural network based identification relies on a previous identification of the different IP applications that can be obtained offline using any reliable method. In this way, the paper also presents a module to process IP traffic flows and identify the underlying applications using payload analysis techniques. The identification results obtained from this tool are used in the training phase of the neural network identification framework. The accuracy of the identification framework was evaluated by performing a set of intensive tests and the results obtained show that, when conveniently trained, neural networks constitute a valuable tool to identify Internet applications while being, at the same time, immune to the most important disadvantages presented by other identification methods.

Index Terms—Port matching, protocol analysis, semantic and syntactic analysis, neural networks, P2P applications, traffic flows.

I. INTRODUCTION

Since 2000, peer-to-peer (P2P) networking introduced a major shift in the application and traffic mix of the Internet and established itself as the main driver of increasing traffic volume [1], [2]. The P2P paradigm proved to be much more efficient than client-server communication especially for fast distribution of large amounts of data, since bottlenecks at servers with sporadic popularity are avoided by distributing requested data and the available access capacity over a global community of recipients. While file sharing may decrease in future because of unresolved problems with copyright infringement, many legal services are expected to be launched via peer-to-peer

overlay networks: video streaming, online gaming, radio and TV over P2P are some of the stronger candidates.

The high requirements of some P2P applications can have a deep influence on network operational issues: these applications consume vast amounts of network resources and can prevent mission critical applications from accessing the network. These applications can also be a security flaw to systems and networks, since hackers can explore them to access and attack large installed bases.

Since P2P applications can originate logistic, security and legal troubles for network administrators, having the ability to correctly identify them can be crucial for several network management and measurement tasks, including traffic engineering, service differentiation, performance/failure monitoring, and security. For example, once a service provider is able to associate traffic to its corresponding IP application it can use this information to group traffic with different or similar (depending on the most advantageous situation) statistical characteristics in order to optimize the bandwidth occupancy of the links. Besides, being able to accurately identify Internet applications, in a general sense, can also be used to detect security attacks, like worms, zombies, botnets, among others, and consequently trigger the appropriate defense and/or repair actions.

The identification of IP applications has been traditionally based on different techniques, like port-based analysis, statistical analysis or payload inspection, each one having its own advantages and limitations. Section II will briefly describe these approaches, pointing out their main advantages and disadvantages. This paper proposes a framework to identify Internet applications that can be used mainly in those situations where existing frameworks are not efficient or can not be used at all (for example, under strict confidentiality requirements): the approach is based on Neural Networks (NNs) and the identification of each application relies on its characteristic traffic pattern. First, the NN model is trained using a set of known traffic values associated with each application - these values can be obtained from measured traffic using conventional application mapping approaches or from known traffic generated in a controlled environment. After this training phase, the trained NN model can identify applications based on new traffic values that are presented as inputs. One of the most important properties of these approaches is its ability to work on real time: the training phase is

computationally demanding but, once the identification models are conveniently trained, they can be used on an online identification tool since the computational requirements of the test phase are very easy to fulfill.

In this approach, the ability to identify an IP application relies on the intrinsic traffic characteristics or profile associated to that application. For example, from Figure 1 we can see that typically a File Sharing application has a very high bandwidth usage pattern characterized by a large variability, which can be attributed to the large number of TCP sessions that are opened/closed; the HTTP service is characterized by a set of non-periodic high bandwidth utilization peaks, with very short durations, that result from the user clicks; on the contrary, the Streaming service is characterized by a constant medium bandwidth usage.

Even considering traffic mixtures/aggregates, which is in fact the real traffic of an operating network scenario (Figure 2), it is still possible to identify differentiating patterns: on a mixture of 80% of File Sharing and 20% of HTTP traffic, there is a high bandwidth usage and large variability for medium bandwidth values, typical characteristics of this level of File Sharing traffic, and non-periodic very short duration peaks, which is a typical behavior of the HTTP service; for a mixture of 20% of File Sharing and 80% of HTTP traffic, it is easy to identify medium bandwidth usage and large variability for smaller bandwidth values, resulting from the presence of 20% of File Sharing traffic, and non-periodic very short duration peaks, which are due to the typical behavior of the HTTP service; for a mixture of 10% of File Sharing, 70% of HTTP traffic and 20% of Streaming traffic, we can identify a medium bandwidth consumption, resulting from the File Sharing service, small variability for smaller bandwidth values, resulting from the presence of Streaming and File Sharing traffic, variability around a specific bandwidth value suggesting the presence of File Sharing and non-periodic very short duration peaks, which are due to the typical behavior of the HTTP service.

The accuracy of the proposed neural network framework will be tested using P2P traffic flows but, since it is a general approach, it can be used to identify any Internet application having a characteristic traffic pattern. The results obtained show that the framework is able to identify various P2P applications, being at the same time immune to the most important drawbacks presented by traditional detection techniques. Once the model is conveniently trained, it can be used to identify Internet applications for new traffic values that are presented as inputs without looking at the packet contents. The computational requirements of the training phase can be significant, but this is an off-line phase - the on-line simulation phase can be performed almost instantaneously.

This paper is an extended version of the work published in [3]: now, we include an extended related work section, a more detailed description of the different modules of the identification framework and a more detailed explanation of the neural network model used in the identification

framework.

The paper is organized as follows: section II briefly describes some related work on this field; section III presents the details of proposed identification framework, including the description of the signature-based pre-identification module and the detailed analysis of the neural network model that constitutes the main framework building block; section IV gives an overview of the traffic traces that are used to test the efficiency of the proposed framework; section V presents and discusses the results obtained and finally, section VI presents the main conclusions and some topics for future research.

II. RELATED WORK

Traditionally, the identification of IP applications has been based on different techniques. Port-based identification was first suggested by [4], [5] and is the most basic and straightforward method to detect applications and users based on network traffic. It is based on the simple concept that many applications have default ports on which they function. When these applications are run, they use these ports to communicate with the outside. To perform port based analysis, administrators just need to observe network traffic and check whether there are connection records using these ports. If a match is found, it may indicate a particular application activity. Port matching is very simple in practice, but its limitations are obvious. Most applications allow users to change the default port numbers by manually selecting whatever port(s) they like. Additionally, many newer applications are more inclined to use random ports, thus making ports unpredictable. Besides, since the closure of Napster more and more P2P applications begin to masquerade their function ports within well-known application ports [6]. In [7], authors have shown that this technique achieves an accuracy no better than 50 to 70% using the official IANA1 list.

Another identification approach is payload or protocol analysis: in this case, the traffic that passes through the network is monitored and the data payload of the packets is inspected according to some previously defined application signatures. This approach has been shown to work very well for Internet traffic including P2P applications [6]–[11]. However, this technique also has some drawbacks: first, payload analysis poses privacy and security concerns; second, the technique typically requires increased processing and storage capacity [5], [12]–[14]; third, it is unable to cope with encrypted transmissions and, finally, this approach only identifies traffic for which signatures are available and are unable to classify previously unknown traffic.

Syntactic and semantic analysis of the data flow avoids some of the disadvantages of port-based analysis and protocol analysis. This approach can perform protocol recognition regardless of any encapsulation and is able to extract data specific to each protocol, involving stateful reconstruction of session and application information from packet content [15]. This technique provides very

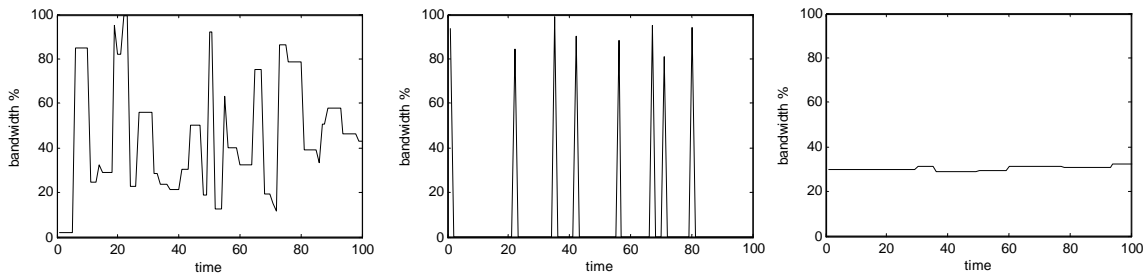


Figure 1. Bandwidth utilization (in percentage) per service: (left) File sharing; (center) HTTP; (right) Streaming.

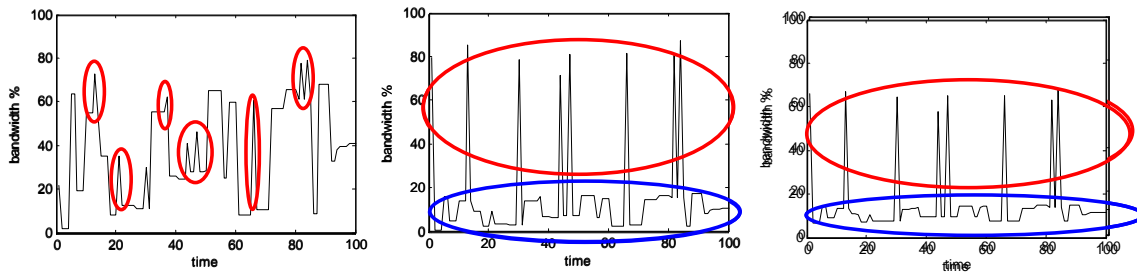


Figure 2. Bandwidth utilization (in percentage) for a mixture of: (left) 80% of File sharing and 20% of HTTP; (center) 20% of File sharing and 80% of HTTP; (right) 80% of File sharing and 20% of HTTP.

accurate and reliable application identification, but imposes significant complexity and processing load on the traffic identification device. It must be kept up-to-date with extensive knowledge of application semantics and network-level syntax, and must be powerful enough to perform concurrent analysis of a potentially large number of flows.

Karagiannis *et al.* [6] proposed a new algorithm which is based on the behavior characteristic of the transport layer: by using little information of transport layer packets, this method can accurately identify 99% of the P2P traffic, but the algorithm can only be used offline. In [16] a new identification method is proposed, which relies on patterns of host behavior at the transport layer. Instead of studying TCP (or UDP) flows individually, this scheme pays attention to all the flows generated by specific hosts and can accurately associate each host with the services it provides or uses (application server, web client, etc). However, this method has to gather information from several flows of each host before it can decide on the host role, which makes it very time-consuming.

Several methods have been also proposed to classify traffic based on summarized flow information such as duration, number of packets and mean inter-arrival time [17], [18], but they are all off-line algorithms and are not sufficiently mature. In [19], the authors use some fundamental characteristics of P2P protocols to identify the P2P applications, such as the huge network diameter and the presence of many hosts acting as both servers and clients. It utilizes only the transport layer header of every packet, and can identify unknown P2P protocols. However, it is time-consuming too. In [20], [21] authors proposed a technique that relied on the observation of

the first five packets of a TCP connection to identify the application. In [22] an algorithm is proposed to identify P2P traffic based on machine learning techniques: by investigating the ratio between the upload and download traffic volume of several P2P applications, a characteristic library is constructed; then, the unknown network traffic can be recognized online using this library. In references [23], [24], authors also use machine learning techniques for traffic classification and [25] proposes a traffic classifier using supervised machine learning based on a Bayesian trained neural network. On reference [26] a back propagation neural network model is used to distinguish between P2P and non-P2P applications; finally, in [27] neural networks were successfully used to identify several Internet applications, although none of them was of the P2P type.

Although several identification techniques have been proposed in the last years, some of them present important drawbacks that inhibit them to be applied under specific operational scenarios: some approaches are not reliable when applied to some specific types of IP applications, like P2P for example; others cannot be used on real time, due to their high processing requirements; most of the approaches are not able to work under very stringent confidentiality requirements, because they rely on packet inspection. The identification methodology that is proposed is general, being able to identify any IP application that has a characteristic traffic pattern; it can work under confidentiality restrictions, since the identification relies on patterns of general traffic statistics and not on any analysis of the packet contents; besides, it can be applied on real time because the identification phase is almost instantaneous (the only computation burden is located on

the training phase, which is inherently an off-line phase).

III. IDENTIFICATION FRAMEWORK BASED ON NEURAL NETWORKS

Neural Networks (NNs) have been successfully used in several different fields due to their advantageous properties like parallel processing of information, capacity to handle non-linearity and quick adaptability to system dynamics: pattern recognition in the presence of noise and non-linearity [28]–[30], classification of Internet users [31], [32], intrusion detection [33], [34], among other applications. Neural Networks can also be used to identify Internet applications, and particularly P2P applications, based on their characteristic traffic patterns. The NN model is trained using a set of traffic values associated with completely known/identified applications (this pre-identification can be based on conventional application mapping approaches or can result from known traffic that is generated in a totally controlled environment; then, the trained NN model is used to identify applications corresponding to new traffic values that are presented as inputs. This approach is not affected by any confidentiality requirements since it does not look at the packet contents and can be used on real time: in fact, the training phase is computationally demanding but, once the identification models are conveniently trained, they can be used online since the computational requirements of the test phase are very low.

The functioning principles of the proposed framework to identify Internet applications is depicted in Figure 3. The central element of the identification tool is the neural network model: first, the NN is trained (that is, its parameters are conveniently adjusted) using a set of known traffic values associated with each application; after the training phase, the trained NN model can be used to identify IP applications based on new traffic values that are presented as inputs. Obviously, the NN training relies on a pre-classification of the various IP applications that is based on offline measurements that were previously made and stored. The pre-classification can rely on conventional application mapping approaches or can derive from known traffic generated and measured in a controlled environment. So, the training phase, although it can be computationally demanding, is an offline phase.

The online classification phase relies on on-line measurements that are continuously made on the network infrastructure and are also stored to become the essential historic data for further NN training. The result of the on-line classification phase, that is, the correct identification of the different applications, must finally be validated. This process involves human intervention and must also take into account the pre-classification of the different applications. The cycle that constitutes this framework is a continuously evolving structure, so the different blocks of the flowchart are continuously updated in such a way that reflects the best possible classification of Internet applications.

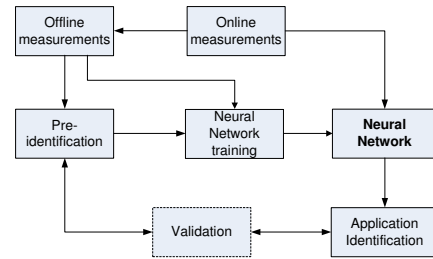


Figure 3. Framework based on NNs to identify IP applications.

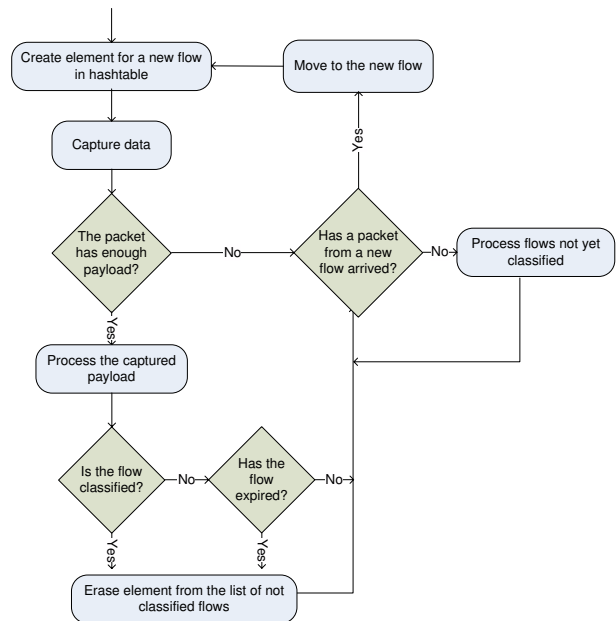


Figure 4. Methodology used for flow processing.

A. Pre-identification

This module must be able to capture IP traffic, identify traffic flows, calculate statistical information about each flow and identify the IP protocol or application that originated that flow. Each IP traffic flow is typically identified by the value of the five-tuple (Source IP Address, Source Port, Destination IP Address, Destination Port, transport layer Protocol). In order to test the accuracy of the proposed framework, we have used a pre-identification methodology based on a database of distinctive strings that was constructed off-line.

The methodology used for flow processing is depicted in Figure 4. Flow identifiers and statistical parameters are saved in memory by using a hash table; each time a new packet is captured, a search is made in the table in order to determine to which flow it belongs. If no match is found, which means that the packet belongs to a new flow, a new element is created including all the relevant information about the flow.

During the capture process, once sufficient packets with payload are captured the application will process them, that is, it will compare the captured payload data with the strings contained in the rules database. If a match is found, the flow is classified and tagged. No more processing will

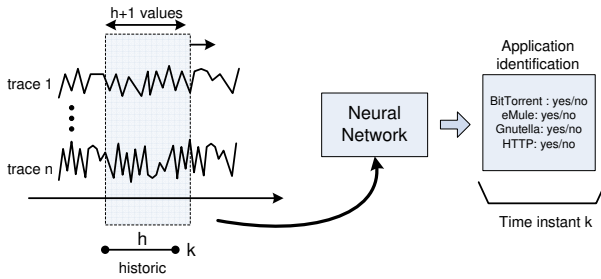


Figure 5. Capturing the temporal correlation of the traffic values.

be made to the packets of this flow, except for its statistics, which is then removed from the list of "not yet classified" flows. When the flow is not classified, the application will not erase it until the pre-defined limit of captured payload is reached, until all packets are processed or until a pre-determined period of time has elapsed. Then, the application waits for the arrival of a new flow. If there is a new flow and sufficient packets with payload have been captured, the new flow is processed. Otherwise, the flows included in the list of "not yet classified" are processed. In parallel, the application is waiting for a new flow. This process continues until the capture finishes and the payloads for all flows have been read and processed.

The identification of the protocol or application that generated each traffic flow relies on a set of rules that were written in a very simple format and include the distinctive strings of each protocol, an offset value that indicates where to start searching for these strings and the message that will be printed when identification is achieved. The rules also allow for a search in a selected range of the saved payload, allowing for a more robust identification of P2P traffic and clients that were used to generate this traffic. The accuracy of the pre-identification module was tested using completely known/identified traces and for all applications the identification was totally correct.

B. Neural Network model

The main objective of this study is to identify different P2P Internet applications based on their generated download and upload traffic, as represented by the number of download/upload bytes per sampling interval and the number of download/upload packets per sampling interval. In order to incorporate some history in this identification process (that can take into account correlations that possibly exist between the traffic distribution per application in adjacent time periods), the current and the last h values of the download/upload traffic will be used to estimate the current values of traffic per application (Figure 5). This problem will be solved using a back propagation NN model.

In general, Neural Networks (NNs) include several layers of *neurons* or processing units: the input layer that receives inputs from the outside, one or more hidden layers that receive inputs only from other processing units, and an output layer that receives the outputs of a previous layer of processing units. Each input value of

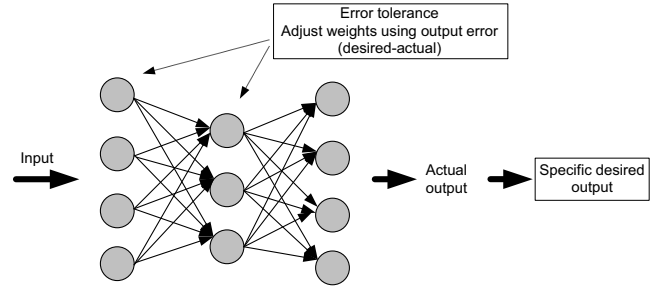


Figure 6. Back propagation neural networks.

a processing unit (that corresponds to a single element of the network input or to each output of the previous layer) is multiplied by a weight and the summation of all these values together with a scalar bias (specific to each neuron) are applied to a transfer function (previously defined for each layer), producing the output value of the neuron. There are three major connection topologies that define how data flows between the input, hidden, and output processing units: feed-forward, limited recurrent, and fully recurrent networks [35], [36]. Feed-forward networks are appropriate for solving problems where all information can be presented to the neural network at once.

The combination of topology, learning paradigm and learning algorithm defines a NN model. Back propagation is an appropriate learning algorithm for training multilayer feed-forward networks for vector classification, modeling and time-series forecasting [37]. It is a general purpose learning algorithm, that is powerful but also expensive in terms of computational requirements for training. A back propagation NN uses a feed-forward topology, supervised learning, and the back propagation learning algorithm. A back propagation network with a single hidden layer of processing elements can model any continuous function to any degree of accuracy (given enough processing elements in the hidden layer) [37].

The basic back propagation algorithm consists of three steps (Fig. 6). The input vector is presented to the input layer of the network. These inputs are propagated through the network until they reach the output units. This forward pass produces the actual or predicted output vector. Because back propagation is a supervised learning algorithm, the desired outputs are given as part of the training set. The actual network outputs are subtracted from the desired outputs and an error signal is produced. This error signal is then the basis for the back propagation step, whereby the errors are passed back through the neural network by computing the contribution of each hidden processing unit and deriving the corresponding adjustment needed to produce the correct output. The connection weights are then adjusted and the NN has just learned from an experience.

Two major learning parameters are used to control the training process of a back propagation network: the learning rate is used to specify whether the neural network is going to make major adjustments after each learning

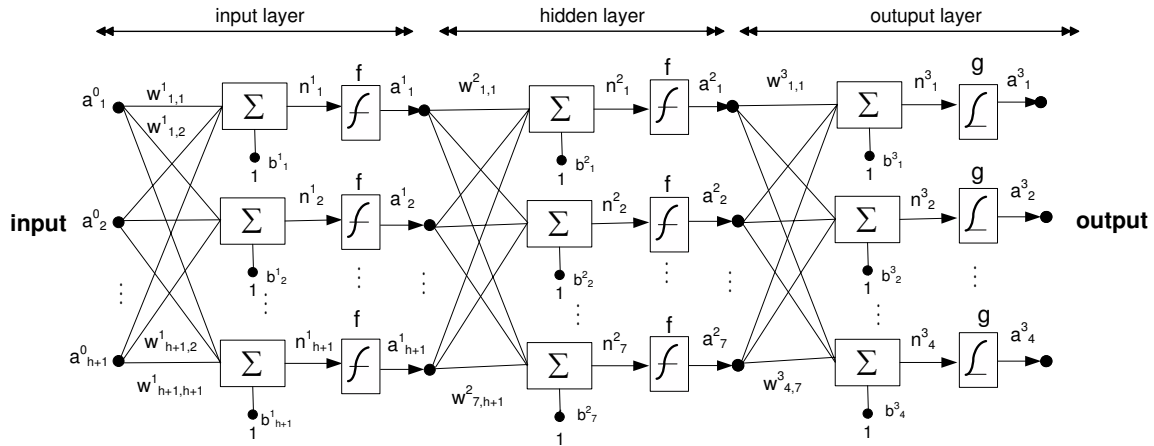


Figure 7. Architecture of the neural network used to identify Internet applications.

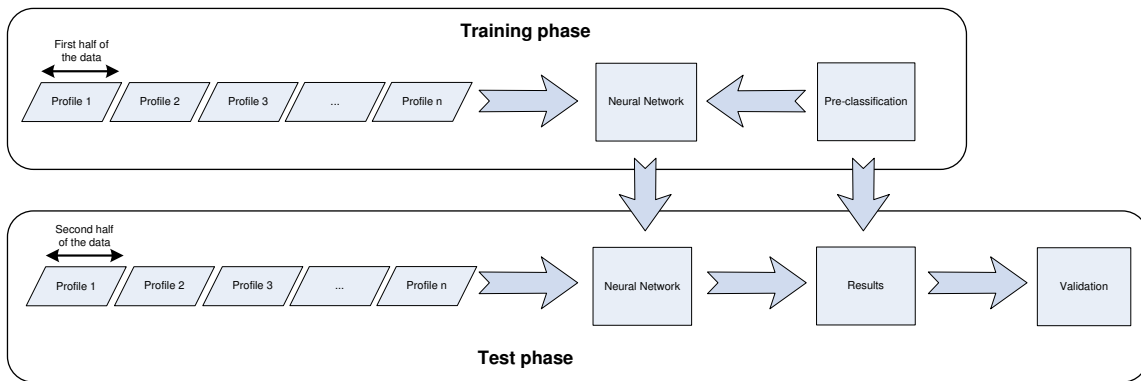


Figure 8. Phases of the neural network application.

trial or if it is only going to make minor adjustments; the momentum is used to control possible oscillations in the weights, which could be caused by alternately signed error signals. These two parameters are the ones that usually produce the most impact on the NN training time and performance.

For the dimension of our problem a conventional feedforward back propagation network with three layers seems to be appropriate. The input layer will have $h + 1$ neurons, corresponding to the dimensionality of the input vectors: each input vector is composed by the current and the last h values of the aggregate traffic. We have tried different values of h , concluding that the performance gains obtained did not increase substantially from the ones given by a seven neuron input layer NN ($h = 6$). Thus, we have considered a NN model having 7 neurons at the input layer. The output layer will have 4 neurons, that is, one neuron per each one of the 4 Internet applications considered in this study. The number of nodes in the hidden layer is empirically selected such that the performance function (the mean square error, in this case) is minimized. Different NNs with variable number of neurons in the hidden layer were considered and their performance was calculated, leading to a choice of 7 hidden nodes. The final structure of the proposed NN

model is presented in Figure 7.

In the proposed NN model, scalar $w^i_{n,j}$ represents the weight value corresponding to layer i , $i = 1, 2, 3$, that is multiplied by input n of neuron j , where n and j have different ranges depending on the network layer. Scalar b^i_j represents the bias associated with neuron j of layer i . For the input and hidden layers, a log-sigmoid transfer function (represented by \mathbf{f} in Figure 7) is used, generating outputs between 0 and 1 as the neurons input goes from negative to positive infinity. For the output layer, a linear transfer function (represented by \mathbf{g} in Figure 7) is used in order to generate any output value. It is known that multiple layers of neurons with nonlinear transfer functions allow the network to learn nonlinear and linear relationships between input and output vectors [35], [36]. Automated Bayesian regularization is used to improve generalization of the neural network, in order to avoid overfitting [38]. Using the above notation, the j^{th} neuron in layer i produces an output a^i_j given by: $a^i_j = F(\sum_n (w^i_{n,j} a^{i-1}_n) + b^i_j)$, $n = 1, 2, \dots, N$, $j = 1, 2, \dots, J$, where N and J are the number of inputs and size of layer i , respectively, and F represents the transfer function of layer i (it can be \mathbf{f} or \mathbf{g}).

As depicted in Figure 8, the application of a NN to solve a particular problem involves two phases: a

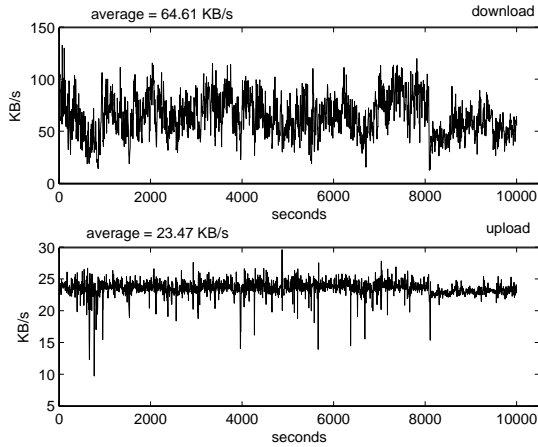


Figure 9. BitTorrent download and upload traffic (in bytes).

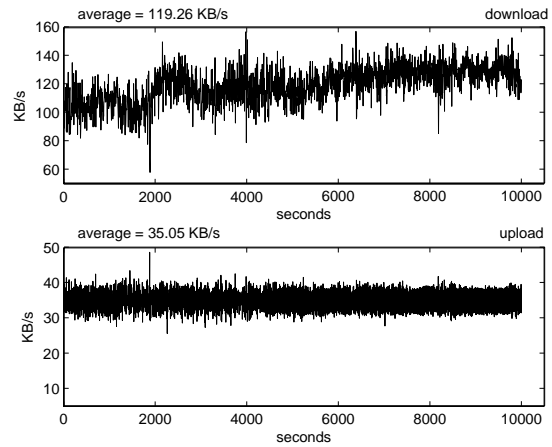


Figure 11. Gnutella download and upload traffic (in bytes).

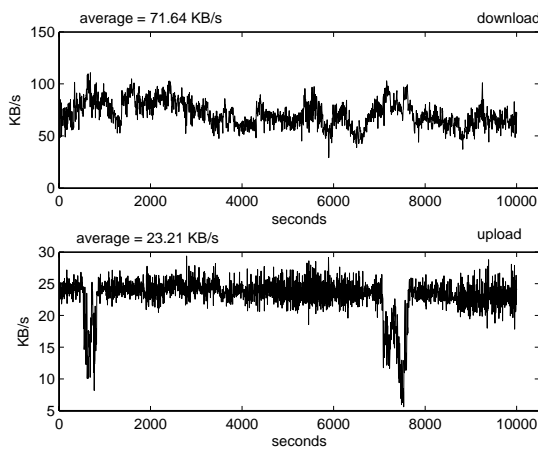


Figure 10. Emule download and upload traffic (in bytes).

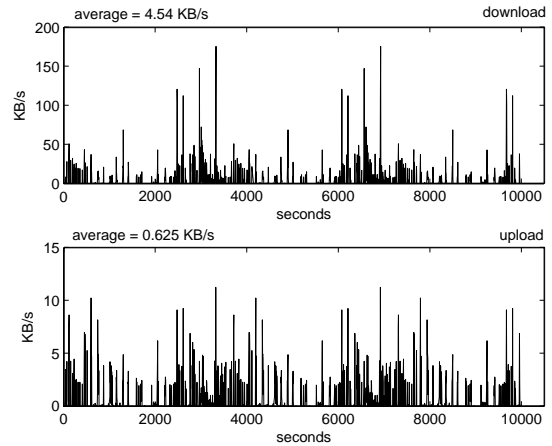


Figure 12. HTTP download and upload traffic (in bytes).

training phase and a test phase. In the training phase, a training set is presented as input to the NN which iteratively adjusts network weights and biases in order to produce an output that matches, within a certain degree of accuracy, a previously known result (named target set). This phase leads to a trained neural network model whose parameters were conveniently adjusted according to the identification requirements. In the test phase, the second half of the application profiles are inputted to the trained neural network model, producing the corresponding identification results. By comparing these results with the pre-identification values, the model can be conveniently validated.

There are two learning paradigms (supervised or non-supervised learning) and several learning algorithms that can be applied, depending essentially on the type of problem to be solved. The network was trained incrementally, that is, network weights and biases were updated each time an input was presented to the network, and the training method used was the Levenberg-Marquardt algorithm, combined automated with Bayesian regularization. Basically, this constitutes a modification of the Levenberg-Marquardt training algorithm to produce net-

works that generalize well, thus reducing the difficulty of determining the optimum network architecture.

In our identification problem the original data set was divided in two subsets of equal size: the first subset was used to train the NN - it became the training set - and the second one was used to test the trained NN - the test set.

IV. OVERVIEW OF THE TRAFFIC TRACES

Our analysis resorts to data traces measured on an ADSL access link characterized by the following maximum allowed transfer rates: 4000 Kbit/s in the downstream direction and 256 Kbit/s in the upstream direction. The traces were measured on September 5 and 6, 2007, and each trace corresponds to the exclusive usage of the access link, during a 5 hour period, by one of the following applications: eMule 0.47c, Azureus 2.3.0, Limewire 4.12.11 and Internet Explorer. The first three applications were chosen since they are freely available at the Internet and correspond to three of the most popular/used P2P networks: eMule, BitTorrent and Gnutella, respectively. The fourth application was chosen in order to have an HTTP trace that could work as a comparative term between P2P traffic and traffic corresponding to Internet web browsing,

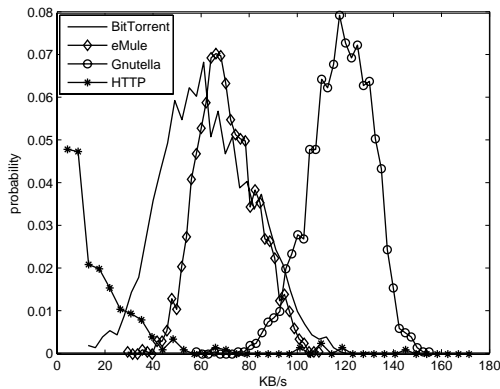


Figure 13. PDF of the download traffic (in bytes).

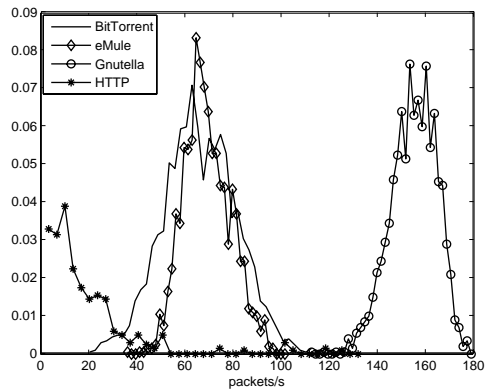


Figure 15. PDF of the download traffic (in packets).

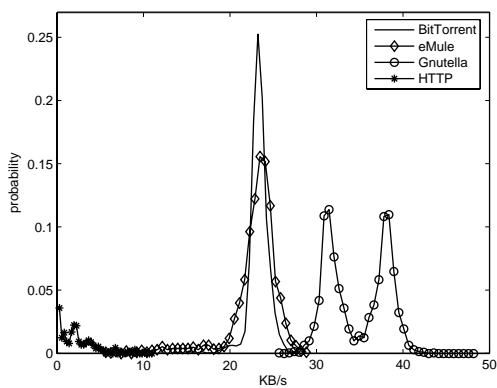


Figure 14. PDF of the upload traffic (in bytes).

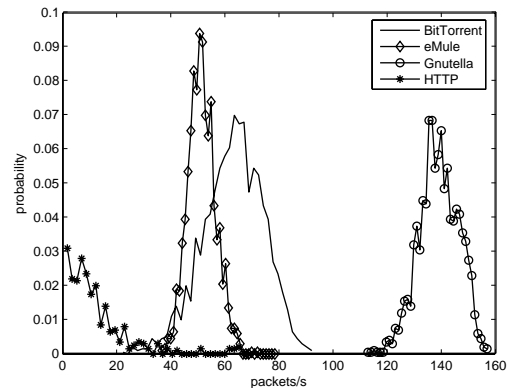


Figure 16. PDF of the upload traffic (in packets).

which was frequently considered in the past years as “normal” Internet usage. For the measurement scenarios involving P2P applications the same files were transferred and a similar number of peers was considered in all situations. The HTTP measurement scenario consisted on accesses to several web sites in such a way that tried to emulate the normal behavior of an Internet user. Each measurement session registered the number of bytes and packets in both directions; no packet drops were reported. The traffic analyzer was a 1.2 GHz AMD Athlon PC having 1.5 Gbytes of RAM and running WinDump.

Figures 9, 10, 11 and 12 represent the download and upload traffic (in bytes/s) corresponding to the BitTorrent, Emule, Gnutella and HTTP applications, respectively. The corresponding Probability Density Functions (PDFs) are represented in figures 13 and 14 for download and upload traffic, respectively. Figures 15 and 16 represent the PDFs of the download and upload traffic, respectively, when they are expressed in packets/s. For these cases, the average traffic rates are the following: 67.54/63.98 for BitTorrent download/upload, 70.08/52.04 for eMule download/upload, 156.16/139.71 for Gnutella download/upload and 5.00/3.27 for HTTP download/upload.

Looking at these graphs, some remarks can be immediately pointed out: (i) the download traffic rates are higher than the upload rates; (ii) despite the involvement

of a similar number of peers and the transfer of the same contents by the different P2P applications, the traffic rates differ from one P2P application to another; (iii) the Limewire application exhibits the highest download and upload rates; (iv) the Azureus and eMule applications generate the most variable traffic rates; (v) the PDFs corresponding to the different applications are not coincident - they are usually quite well separated, with the exception of the eMule and BitTorrent rates’ PDFs that are not very far from each other. These facts will have obvious effects on the identification results obtained by applying neural networks.

Going into a more detailed analysis of the traffic plots, we can see that the download traffic, when expressed in bytes per second, has more variability than the upload traffic, for all P2P applications. When traffic is expressed in packets per second, the variability is also higher for download but the difference is not so significant. These facts can be explained having in mind some headlines about the way these P2P applications work: the average download traffic is higher than the average upload traffic, for all measurement scenarios; the number of peers involved in downloading is higher than the number of peers involved in uploading; there is a higher diversity of packet lengths on the download traffic, since these operations generally include large data packets and small

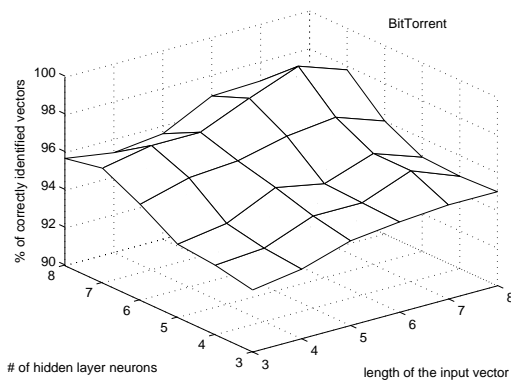


Figure 17. Performance of BitTorrent traffic identification.

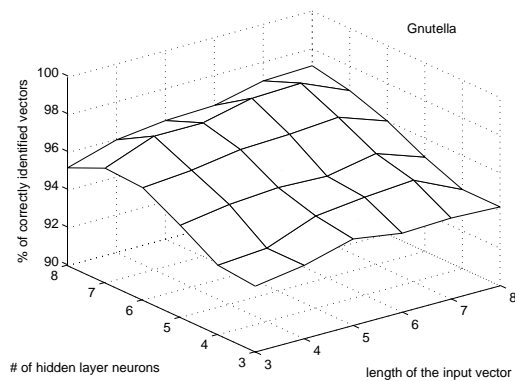


Figure 19. Performance of Gnutella traffic identification.

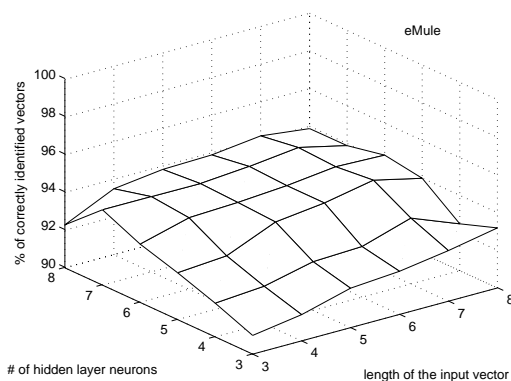


Figure 18. Performance of eMule traffic identification.

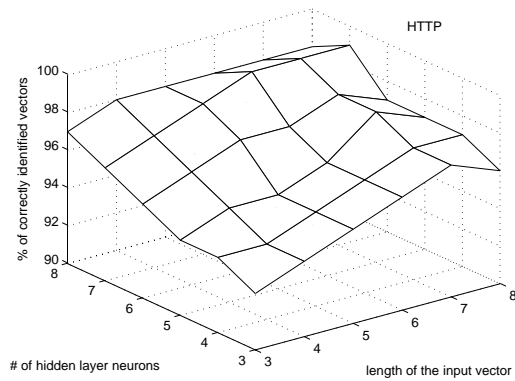


Figure 20. Performance of HTTP traffic identification.

acknowledgment packets.

V. RESULTS

The traffic prediction results given by the NN model are shown in Figures 17 to 20, where each figure corresponds to a specific application. Each plots presents the identification results, expressed as the percentage of correctly identified input vectors, as a function of the number of hidden layer nodes and the length of the input vector (that represents the “quantity” of historic data that must be inputted to the neural network model for a correct identification). As can be seen from the graphs, the best identification results are obtained for a NN model having seven neurons in the first layer, that corresponds to an historic length of 6 values, and seven neurons in the hidden layer. So, these were the selected values that lead to the previously presented NN model. Regarding the results, we can see that they are quite good for all applications: the highest identification values are achieved for HTTP but even for the three selected P2P applications the correct identification percentages are all above 90%.

Table I presents the detailed values of the identification performance, that is, the percentage of correctly identified input vectors for the different numbers of hidden layer nodes and input vector lengths.

From these results we can say that the neural network model can be efficiently used to identify the characteristic traffic patterns of HTTP and P2P Internet applications (more precisely, the three P2P applications that were selected). The traffic patterns associated to these applications are sufficiently distinct to be accurately identified by the NN model: of course, it is possible to face applications that have very similar or almost indistinguishable traffic patterns and, in those cases, this framework will not be efficient. The proposed approach is not conditioned by the most important drawbacks presented by other identification methodologies, mainly high computational processing requirements and confidentiality issues. Once conveniently trained, this methodology is able to detect on real time applications corresponding to new incoming traffic, since the only computationally demanding phase is the offline training phase. Since this approach does not perform any packet inspection, traffic encryption and confidentiality are not problems that can limit its utilization possibilities.

VI. CONCLUSIONS AND FURTHER RESEARCH

As the number and diversity of IP applications increases, it becomes more and more important to accurately map Internet traffic to their corresponding applications. Network management and measurement tasks

# of hidden layer neurons	length of the input vector				
	5	6	7	8	9
3	93.7/91.4/93.9/94.1	94.5/92.1/94.6/95.3	94.8/92.3/94.2/96.2	95.0/92.7/94.3/97.1	94.9/93.2/94.2/96.0
4	93.9/91.9/93.9/94.2	94.9/92.6/94.9/95.1	94.9/92.7/95.2/96.4	95.9/93.5/95.4/97.3	94.5/92.5/94.2/97.0
5	94.3/92.7/95.3/95.2	95.5/93.8/95.5/95.0	95.0/94.1/95.4/96.3	95.8/94.6/95.9/98.1	95.0/93.9/95.0/97.8
6	95.8/93.8/96.2/96.3	95.9/93.9/96.6/97.1	96.6/94.1/96.7/97.3	96.9/94.4/96.9/98.4	96.0/94.3/96.0/97.6
7	96.6/94.0/97.1/97.3	96.6/94.0/97.2/98.1	97.7/94.1/97.7/99.0	98.7/94.5/97.8/99.2	97.8/93.9/96.7/99.1
8	95.3/93.5/95.9/98.0	95.6/93.8/96.3/98.1	96.7/93.9/96.4/98.2	97.0/94.2/97.0/98.3	96.6/93.9/97.1/98.1

TABLE I.

PERCENTAGE OF CORRECTLY IDENTIFIED INPUT VECTORS, PER APPLICATION, AS A FUNCTION OF THE NUMBER OF HIDDEN LAYER NODES AND THE LENGTH OF THE INPUT VECTOR (EACH COMPOSED VALUE A/B/C/D CORRESPONDS TO THE BITTORRENT/EMULE/GNUTELLA/HTTP ORDER).

like traffic engineering, service differentiation, performance/failure monitoring, and security can greatly benefit from this mapping ability. This paper proposed a new framework, based on neural networks, that was able to identify flow patterns generated by HTTP and P2P Internet applications while, at the same time, avoid the most important disadvantages presented by existing identification methods. This is a preliminary work that will necessarily lead to further research: other Internet applications can be included in this study and different neural network types and architectures can be used to solve the identification problem; besides, the identification methodology can be applied to mixtures of applications, that is, the neural network framework can be used to identify aggregated traffic belonging to more than one application.

REFERENCES

[1] K. Gummedi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles 2003*, 2003.

[2] T. Karagiannis, A. Broido, N. Brownlee, K. C. Claffy, and M. Faloutsos, "Is P2P dying or just hiding?" in *IEEE Global Telecommunications Conference 2004*, 2004.

[3] A. Couto, A. Nogueira, P. Salvador, and R. Valadas, "Identification of peer-to-peer applications' flow patterns," in *Proceedings of the 4th Euro-NGI Conference on Next Generation Internet Networks*, 2008.

[4] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, pp. 219–232, 2004.

[5] D. Moore, K. Keys, R. Koga, E. Lagache, and K. C. Claffy, "The coralreef software suite as a tool for system and network administrators," in *Proceedings of the 15th USENIX conference on Systems Administration (LISA'01)*, 2001, pp. 133–144.

[6] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy, "Transport layer identification of P2P traffic," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, 2004, pp. 121–134.

[7] A. W. Moore and D. Papagiannaki, "Toward the accurate identification of network applications," in *Proceedings of the 6th Passive Active Measurements Workshop*, vol. 3431, 2005, p. 41.

[8] M. Roesch, "Snort: Lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX Conference on Systems Administration (LISA'99)*, 1999, pp. 229–238.

[9] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer Networks*, no. 31.

[10] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, "Acas: Automated construction of application signatures," in *Proceedings of the SIGCOMM05 Workshops*, 2005.

[11] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of P2P traffic using application signatures," in *Proceedings of the 13th International Conference on World Wide Web*, 2004.

[12] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood, "Deep packet inspection using parallel bloom filters?" *IEEE/Micro*, vol. 24, no. 1.

[13] T. Kocak and I. Kaya, "Low-power bloom filter architecture for deep packet inspection," *IEEE/Communications Letters*, vol. 10, no. 3.

[14] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: a survey," *Internet Mathematics*, vol. 1, no. 4.

[15] C. I. Documentation, "Network-based application recognition and distributed network-based application recognition," 2006. [Online]. Available: <http://www.cisco.com>

[16] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: multilevel traffic classification in the dark," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2005.

[17] 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 ACM SIGCOMM Internet Measurement Conference*, 2004, pp. 135–148.

[18] A. Moore and D. Zuev, "Internet traffic classification using bayesian analysis," in *Proceedings of International Conference on Measurement and Modeling of Computer Systems*, 2005, pp. 50–60.

[19] F. Constantinou and P. Mavrommatis, "Identifying known and unknown peer-to-peer traffic,," in *Proceedings of Fifth IEEE International Symposium on Network Computing and Applications*, 2006, pp. 93–102.

[20] L. Bernaille, R. Teixeira, and I. Akodkenou, "Traffic classification on the fly," *Computer Communication Review*, vol. 36, no. 2, pp. 239–26, 2006.

[21] L. Bernaille and R. Teixeira, "Early recognition of encrypted applications," in *In Proceedings of the 8th Passive and Active Measurement Conference (PAM 2007)*, 2007.

[22] H. Liu, W. Feng, Y. Huang, and X. Li, "A peer-to-peer traffic identification method using machine learning," in *Proceedings of the International Conference on Networking, Architecture and Storage*, 2007.

[23] R. Y. Z. Li and X. Guan, "Accurate classification of the internet traffic based on the SVM method," in *In*

Proceedings of the 42th IEEE International Conference on Communications (ICC 2007), 2007.

- [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.
- [25] 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.
- [26] F. Shen, C. Pan, and X. Ren, "Research of P2P traffic identification based on BP neural network," in *Proceedings of the Third international Conference on international information Hiding and Multimedia Signal Processing (IHH-MSP 2007)*, 2007.
- [27] A. Ali and R. Tervo, "Traffic identification using artificial neural network," in *Canadian Conference on Electrical and Computer Engineering*, vol. 1, pp. 667–672.
- [28] B. D. Ripley, *Pattern recognition and neural networks*. Cambridge University Press, Inc., 1996.
- [29] C. Looney, *Pattern recognition using neural networks: theory and algorithms for engineers and scientists*. New York, NY, USA: Oxford University Press, Inc., 1997.
- [30] C. G. Looney, *Pattern recognition using neural networks: theory and algorithms for engineers and scientists*. New York, NY, USA: Oxford University Press, Inc., 1997.
- [31] A. Nogueira, M. R. de Oliveira, P. Salvador, R. Valadas, and A. Pacheco, "Using neural networks to classify internet users," in *First Advanced International Conference on Telecommunications*, July 2005.
- [32] A. Nogueira, R. Oliveira, P. Salvador, R. Valadas, and A. Pacheco, "Classification of internet users using discriminant analysis and neural networks," in *First Conference on Traffic Engineering for the Next Generation Internet*, April 2005.
- [33] J. Ryan, M.-J. Lin, and R. Miikkulainen, "Intrusion detection with neural networks," *Advances in Neural Information Processing Systems*, MIT Press, vol. 10, 1998.
- [34] S. Mukkamala and A. H. Sung, "Identifying key features for intrusion detection using neural networks," in *ICCC '02: Proceedings of the 15th international conference on Computer communication*. Washington, DC, USA: International Council for Computer Communication, 2002, pp. 1132–1138.
- [35] L. Fausett, *Fundamentals of Neural Networks*. Prentice Hall, 1994.
- [36] K. Gurney, *An Introduction to Neural Networks*. UCL Press, 1997.
- [37] H. Demuth and M. Beale, *Neural Network Toolbox Users Guide*. The MathWorks, Inc., 1998.
- [38] F. Foresee and M. Hagan, "Gauss-newton approximation to bayesian regularization," in *Proceedings of the 1997 International Joint Conference on Neural Networks*, 1997, pp. 1930–1935.

Antonio Nogueira was born in Mozambique, on March 12, 1969. He graduated in Electronics and Telecommunications Engineering in 1992, received his MSc. degree in Electronics and Telecommunications Engineering in 1997 and the PhD degree in Electrical Engineering in 2005, all from University of Aveiro, Portugal. In 1995, he joined Institute of Telecommunications as a researcher. He is currently an Assistant Professor at University of Aveiro. His main research areas of interest are traffic modeling, traffic engineering, network modeling and planning and network security.

Paulo Salvador was born in Coimbra, Portugal, on February 17, 1975. He graduated in Electronics and Telecommunications Engineering in 1998 and received the PhD degree in Electrical Engineering in 2005, both from University of Aveiro, Portugal. In 1998, he joined Institute of Telecommunications, Aveiro, Portugal, as a researcher. He is currently an Assistant Professor at University of Aveiro. His main research areas of interest are traffic modeling, traffic engineering, network planning and Internet security.

Luis Andre Couto was born in Portugal, on January 6, 1983. He graduated in Electronics and Telecommunications Engineering in 2007, receiving his MSc. degree in Electronics and Telecommunications Engineering in the same year, both from University of Aveiro, Portugal. In 2007, he joined Institute of Telecommunications as a researcher, being currently assigned to the IST WIP (An All-Wireless Mobile Network Architecture) project. His main research areas of interest are wireless mesh networks, mobility and traffic management.

Rui Valadas graduated in Electrical and Computers Engineering, in 1986, from the Instituto Superior Tecnico, Lisbon, Portugal, and received the Ph.D. degree from the University of Aveiro, Aveiro, Portugal, in 1996. He joined the University of Aveiro in 1986, where he is now an Associate Professor of the Department of Electronics and Telecommunications. He is also leader of the Networks and Multimedia Communications Group at the Institute of Telecommunications, Aveiro, Portugal. His main research interests are in the area of traffic engineering.