Applied Soft Computing 9 (2009) 685–694

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Online hybrid traffic classifier for Peer-to-Peer systems based on network processors

Zhenxiang Chen [a], Bo Yang [a,*], Yuehui Chen [a], Ajith Abraham [a,b], Crina Grosan [c], Lizhi Peng [a]

[a] School of Information science and Engineering, University of Jinan, 106 Jiwei Road, 250022 Jinan, PR China
[b] Centre for Quantifiable Quality of Service in Communication Systems, Norwegian University of Science and Technology, Trondheim, Norway
[c] Babes-Bolyai University, Cluj Napoca, Romania

ARTICLE INFO

Article history:
Received 25 January 2007
Received in revised form 25 September 2008
Accepted 28 September 2008
Available online 17 October 2008

Keywords:
Peer-to-Peer (P2P)
Network processors (NPs)
Traffic classification
Flexible Neural Tree (FNT)
Hybrid
Adaptation
Online learning

ABSTRACT

It is estimated that 70% or more of broadband bandwidth is consumed by transmitting music, games, video and other content through Peer-to-Peer (P2P) clients. In order to detect, identify, and manage P2P traffic, some port, payload and transport layer feature based methods were proposed. Most of them were applied to offline traffic classification mainly due to the performance reason. In this paper, a network processors (NPs) based online hybrid traffic classifier is proposed. The designed hardware classifier is able to classify P2P traffic based on the static characteristic namely on line speed, and the Flexible Neural Tree(FNT) based software classifier helps learning and selecting P2P traffic attributes from the statistical characteristics of the P2P traffic. Experiment results illustrate that the hybrid classifier performs well for online classification of P2P traffic from gigabit network. The proposed framework also depicts good expansion capabilities to add new P2P features and to adapt to new P2P applications online.

1. Introduction

Since the emergence of Peer-to-Peer (P2P) applications in the late-1990s, P2P file-sharing has relentlessly grown to represent a formidable component of Internet traffic. Recently, it is very common that P2P applications also reserve a big part of operator's total bandwidth [1]. It is often considered as a threat to the businesses of Internet operators, Internet Service Providers (ISP) and LAN operators. The bandwidth costs to the upstream ISPs and inter exchange carriers are creating growing financial pressure. In practice, this means that less bandwidth can be used for other network usage like web browsing, e-mail and other critical applications. Some research studies illustrate that there are many known and unknown potential P2P traffic, which consumes almost 70–80% bandwidth but still cannot be well controlled and managed. The financial benefits and fairly use of network resources are undeniable motives for controlling P2P communications and managing P2P hosts. Despite this discomfiture, reliable profiling of P2P hosts and traffic remains elusive.

Some of the currently proposed P2P traffic classification methods, such as mapping of used IP-addresses or monitoring port numbers, are not reliable. Packet payload capture and analysis poses a set of insurmountable methodological land mines, which includes legal, privacy, technical, logistics, security and encryption, financial obstacles and growing number of poorly documented P2P protocols. A transport layer identification method attempts to identify P2P traffic only based on transport layer and other special P2P features but does not make use of the payload information. Recently, some machine learning based methods were used to identify network applications. Almost all the proposed methods, however, are only suitable for offline traffic classification due to the performance reasons. In reality, the offline classification is not helpful for detecting or managing online traffic.

In this scenario, in order to achieve flexibility and high performance, the most promising solution is represented by the adoption of network processors (NPs) [2]. NPs are emerging platforms that offer very high packet processing capabilities (e.g. for gigabit networks) and combines the programmability of general-purpose processors with high performance typical of hardware-based solutions. This paper proposes a NPs-based online hybrid traffic classifier to identify active P2P traffic. The NPs-based

* Corresponding author.
E-mail addresses: czx@ujn.edu.cn (Z. Chen), yangbo@ujn.edu.cn (B. Yang), yhchen@ujn.edu.cn (Y. Chen), ajith.abraham@ieee.org (A. Abraham), plz@ujn.edu.cn (L. Peng).

1568-4946/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2008.09.010

hardware classifier classifies traffic by static characteristics and the Flexible Neural Tree (FNT) [3] based software classifier helps classifying traffic by the statistical characteristics. Experiment results reveal that the hybrid classifier has high performance and is very competent for gigabit network traffic online classification, besides having some other good features.

The remaining part of the paper is organized as follows. Section 2 reviews related work in P2P traffic classification. Section 3 presents the proposed P2P features and researches the normalization method for the statistical features. The design of hardware and software united hybrid classifier is described in Section 4. Section 5 describes the experiments, results and related evaluation of the new method. Section 6 concludes our work and gives the future work.

2. Related research literature

A quick literature review reveals that some network packet [23,25] and traffic identification related research and classifiers [24] have been proposed. In order to learn more about traffic information, some P2P traffic identification research emphasized detailed features of a small subset of P2P protocols and/or networks [4–6], often motivated by the dominance of that protocol in a particular provider's infrastructure or during a specific time period. Some other systems used information such as known port number, flow level analysis [1], payload information (such as signatures) [8], topological feature and transport layer features [7].

Known transport-layer port numbers used to be an accurate and efficient way for traffic classification. The problem, however, is that recently P2P applications use arbitrary ports and even well known application ports (e.g. 80, 21) to disguise their traffic in order to go through firewalls or avoid legal persecution, thus making it even harder to identify.

Then, the payload information identification method was proposed. Most protocols contain a protocol specific string in the payload that can be used for identification. These strings are often public information and can also be determined by examining a number of network traffic traces. Subhabrata et al. [8] presented an analysis of a number of P2P protocols and their signatures. Their classifier could accurately identify traffic by reading the payload and using feature matching to discover protocol signatures. Matthew et al. [9] also used application signatures for classification but incorporated heuristics for categorizing unknown applications with respect to a known class of applications. Payload-based classifiers illustrate good results but have three major limitations: (a) they cannot be used if payload information is not available because of legal, privacy and encryption reason; (b) they cannot, in general, identify unknown classes of traffic; (c) examining the payload to classify traffic in real time is impractical due to its high overhead, especially if there is a need for high network utilization.

In order to overcome the limitations of port and payload based classification schemes [7,10], Thomas et al. [7] focus on identifying P2P traffic by the source and destination connection features. Two heuristics were used, which in most cases are sufficient to work well. The first heuristic identifies source-destination IP pairs that use both TCP and UDP transport protocols. Concurrent use of both TCP and UDP is usually an indication that the traffic is P2P. The second heuristic is based on monitoring connection features of {IP, port} pairs. BLINC [10] explores a number of heuristics (Social level, Function level and Application level) for classification at transport layer.

Recently, Machine learning methods were also used to identify game traffic [11] and network applications [12]. Cluster analysis is one of the most prominent methods for identifying classes amongst a group of objects, and has been used as a tool in many fields such as biology, finance, and computer science. Recent work by McGregor et al. [13] and Zander et al. [14] show that cluster analysis has the ability to group Internet traffic using only transport layer characteristics. Mahanti and co-workers [26], Quin et al. [27], used semi-supervised learning methods to classify the network traffic and application.

Soft computing method such as genetic algorithm was used to network traffic patterns classification [30]. Chandramathi et al. [31] utilized fuzzy approach to estimate cell loss probability and Support Vector Machine (SVM) method was used to video streams classification by Awada et al. [33]. Especially, other online hybrid approaches were used to system identification [35],network traffic classification [29,36] and data mining of data streams [32,34],which promote the online network traffic classification and controlling research greatly.

In contrast, the proposed approach is based on three principles:

(1) The target is to design a hybrid P2P traffic classifier, which includes most of the proposed effective methods to improve classification accuracy;
(2) The proposed classifier is designed based on network processors, which can help distinguishing P2P traffic online from high speed gigabit network stream;
(3) The software classifier can learn P2P characters from hardware identified P2P traffic, which can help to find new P2P applications intelligently and automatically.

3. The P2P traffic features

3.1. Static P2P traffic features

P2P traffic has some port-based and payload-based characteristics, which is called static features in this article. It proves to have some defects for classifying P2P traffic. The analysis experience shows that part of P2P clients still has obvious static characteristics, which is helpful for accurate P2P traffic identification, because none of the non-P2P application will disguise as P2P traffic.

3.1.1. Port-based feature

Port matching is very simple in practice and there are some P2P clients running on well-known ports. Some of the well-known P2P ports [1,7,15] are listed in Table 1.

3.1.2. Payload-based feature

To address the afore-mentioned drawbacks of port-based classification, several payload-based analysis techniques have been proposed [5,7,11]. In this approach, packet payloads are analyzed to determine whether they contain characteristic signatures of known applications. Studies show that these

Table 1
Port numbers of known P2P applications.

P2P application	Port numbers	Protocol
Limewire	6346/6347	TCP/UDP
Morpheus	6346/6347	TCP/UDP
BearShare default	6346	TCP/UDP
Edonkey	4662	TCP/UDP
EMule	4662(TCP)/4672(UDP)	TCP/UDP
Bittorrent	6881–6889	TCP/UDP
WinMx	6699(TCP)/6257(UDP)	TCP/UDP
eDonkey2000	4661–4665	TCP/UDP
Fastidentify	1214	TCP/UDP
Gnutella	6346–6347	TCP/UDP
MP2P	41170	TCP/UDP
Direct connect	411–412	TCP/UDP

approaches still can work well for current Internet except being encrypted. But for recent new Kazaa version (v1.5 or higher), a peer may send an encrypted short message before it sends back above response. Note that both messages include a field called X-Kazaa-SupernodeIP. This field specifies the IP address of the supernode to which the peer is connected including the TCP/UDP supernode service port. This information could be used to identify signaling using flow records for all communication. Using the special HTTP headers found in the Kazaa downloading, we find two steps can identify Kazaa stream:

(1) The string following the TCP/IP head is one of following: 'GET', and 'HTTP'.
(2) There must be a field with string: X-Kazaa.

Some other payload features of P2P protocols such as Gnutella, eDonkey, DirectConnect and BitTorrent were analyzed in Ref. [8], others are analyzed in the proposed classifier system.

3.2. Statistic attributes selection and normalization

The proposed software classifier is a machine learning based method. So, a set of effective P2P token features should be selected to support it. Three proposed features and one new found feature were selected to create the feature set in the developed hybrid classifier prototype system (see Section 5.1).

3.2.1. Transport layer protocol feature

Traffic which source-destination IP pairs concurrently use both TCP and UDP during a special time can be expected as potential P2P traffic. Six out of the nine analysed P2P protocols in Ref. [7] use both TCP and UDP as layer-4 transport protocols. These protocols include eDonkey, Fastidentify, WinMx, Gnutella, Emule, Bittorrent and Direct Connect, some of them are listed in Table 1. Generally, controlling traffic, queries and query-replies by UDP, and actual data transfers use TCP. It is found that besides P2P protocols, only a few applications use both TCP and UDP:DNS, NETBIOS, IRC, gaming, streaming, FTP and SQLserver, which collectively typically put in a port numbers set, which signed as $M = (135, 137, 139, 445, 53, 3531, 21, 1433, 1434)$. The newly found applications with the same feature can be added in this set when applied to real environment. Fig. 1 shows the transport layer connection feature of both TCP and UDP.

According to this analysis, when TCP and UDP both have been used for a fixed IP (host) pairs during a given time t, the traffic between the host pairs may be seen as potential P2P traffic. During a given time t, for a fixed IP, the number of connected IPs that use both TCP and UDP is signed as N_{Pro}, the number of connected destination IPs with port that listed in set M is signed as $N_{Special}$, and the number of all connected IPs is signed as N_{all}. The normalized transport layer protocol feature can be described as

$$f_{Pro} = \frac{N_{pro} - N_{Special}}{N_{pro}} \qquad (1)$$

Larger value of f_{Pro} means higher potential of P2P traffic between monitored IP pairs.

3.2.2. IP and port pair feature

In some P2P networks, each host chooses an arbitrary port to connect to a different host. A connection feature where the number of distinct connected IPs is equal to the number of distinct connected ports usually is an evidence of P2P traffic [7]. In the decentralized and hybrid P2P network, each peer selects a temporary source port and connect to the advertised listening port of peer A. For the advertised destination {IP, port} pair of host A, the number of distinct IPs connected to it will be equal to the number of distinct ports used to connect to it. The probability that two distinct hosts pick the same random source port at the same time is extremely low.

As in the P2P case, each host connects to a pre-specified {IP, port} pair, e.g., the IP addresses of a web server W and port 80. In summary, web traffic will have a higher ratio than P2P traffic of the number of distinct ports versus number of distinct IPs connected to the {IP, port} pair {W, 80}. So this can also be seen as a potential P2P feature. The 4-tuple feature graph is illustrated in Fig. 2. During a given time t, the number of connected IPs was defined as N_{IP} and the number of connected different ports is defined as N_{port}, then the normalized IP and port pair feature is calculated by

$$f_{IP-Port} = \frac{N_{IP} - N_{Port}}{N_{IP}} \qquad (2)$$

The smaller $f_{IP-Port}$ value is, the more potential of the P2P host it is and the related traffic can be seen as potential P2P traffic.

3.2.3. UDP port connection feature

Gong [15] found that unique traffic behavior to UDP connection feature exists with P2P applications. This can be used to process network traffic and find out, which hosts are running P2P applications in a decentralized network structure. Today almost all P2P applications using a decentralized structure have a built-in module to fulfill their interaction work, because there are many control purpose packets that requires to be sent out to many destinations. A great deal of the modern P2P networks and protocols select UDP as the carrying protocol. An investigation of

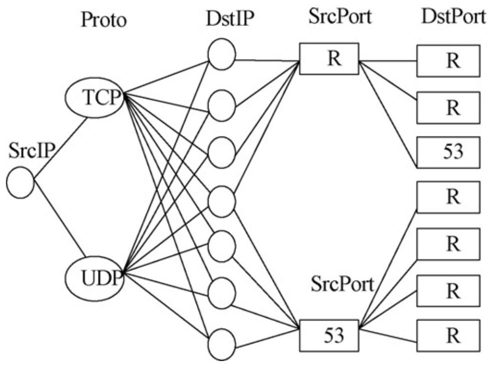


Fig. 1. The transportal layer connection feature of both TCP and UDP connected (IP, port) pair. R is an arbitrary port number and 53 is one of the special non-P2P application port, which also connect the (IP, Port) pair with both TCP and UDP protocol.

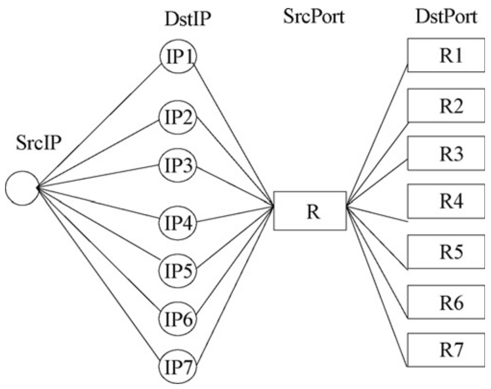


Fig. 2. The 4-tuple feature of IP and port connection. R is an arbitrary port number and R1,. . .,R7 mean different arbitrary port numbers.

some other decentralized P2P applications, such as BearShare, Skype, Kazaa, EMule, Limewire, Shareaza, Xolox, MLDonkey, Gnucleus, Sancho, and Morpheus leads to a similar conclusion.

All these applications have the same connection feature: for a period of time t, from one single IP, fixed UDP port connected to fixed or random UDP ports of y destinations IPs. They use one or more UDP ports to communicate with many outside hosts during their lifetime. Research experience shows that when t equals 5 min, y equals three, as administrators scanning for a P2P application, the satisfying result could be obtained. This 4-tuple feature graph can is depicted in Fig. 3. According to this analysis, we assume that during a given time t, for a fixed IP with m UDP ports who connected to the fix or random UDP ports of N_i or more IPs, then the normalized UDP port connection feature is obtained from

$$f_{U\ port} = \frac{m}{\sum_{i=1}^{m} N_i} \tag{3}$$

As shown in (3), the smaller $f_{U\ port}$ value is, just saying that the larger of each UDP port average connected IP number is, the more potential P2P traffic can be seen related to the host.

3.2.4. DNS query log feature

Run effectively without support of DNS service is one of the primary characters of P2P application [16]. The structured and unstructured model based P2P application, such as Skype, Kazaa, EMule, Bittorent and Papstry, Chord, based P2P applications can get the neighbours' information from the center server, the super node or the DHT, but never need locate the destination by DNS service. So the DNS server log record can be checked as P2P evidence by obtaining six months of statistical data, during a given time of 5 min, for a fixed IP (host), if it connected more than 5 IPs but without none of the connected IP's DNS query log. It may be seen as a potential P2P host. This feature is especially true in a LAN environment with self-governing DNS server. During a given time, the connected IPs number with query destination log is signed as N_{log} and all connected IP number is signed as N_{all}. Eq. (4) shows the normalized DNS query log feature of a P2P host.

$$f_{DNS} = \frac{N_{log}}{N_{all}} \tag{4}$$

If $N_{all} > 5$ and f_{DNS} equal 0, the fixed IP can be seen as potential P2P host and if $f_{DNS} = 1$ then it may be not a P2P host. The small f_{DNS} means high P2P host potentiality. The host related traffic can be seen as potential P2P traffic. The feature details are also illustrated in Fig. 4. There are other P2P features such as symmetry

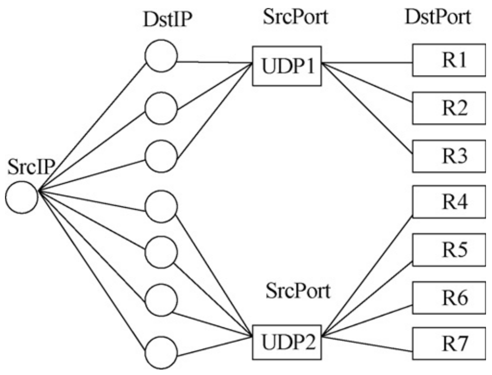


Fig. 3. The UDP port connection feature in a given time. R1,...,R7 mean different arbitrary port numbers.

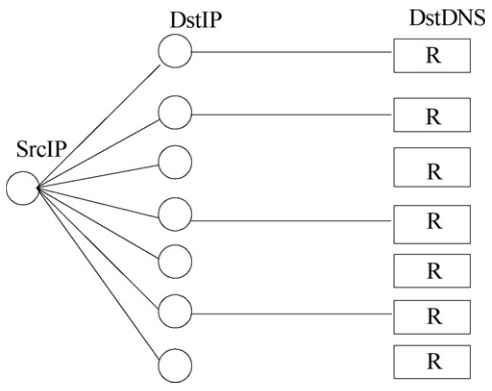


Fig. 4. The DNS query log record feature. R means the connected destination IP's DNS querying log record. The connection line between DstIP and DstDNS means that a log record of the DstIP can be found.

feature, network diameter, flow time and other new found features that could be added to the features set in our future research.

4. The design of hybrid classifier

In case, where performance is key point of online classification, it is possible to combine high-speed hardware classification with low cost and increased flexibility of software classification. To do so, a hybrid classifier that contains a hard classing stage for standard static cases and a software stage for exceptions is designed in this paper. In essence, a hybrid design uses hardware as a filter. If it recognizes a P2P flow packet, the hardware classifier diverts the packet into P2P traffic. Unrecognized packets are passed as unclassified traffic and related features can be statistically collected for the software classifier at the same time.

4.1. NP-based hardware classifier

A network processor is a special purpose, programmable hardware device that combines the low cost and flexibility of a RISC processor with the speed and scalability of custom silicon. Network processors are building blocks used to construct network system [17]. NP has the following characters:

(1) Relatively low cost
(2) Straightforward hardware interface
(3) Ability to access memory
(4) Programmability
(5) Ability to scale to high data rates
(6) Ability to scale to high packet rates

IXP2400 [18] is selected as the hardware platform. The Intel IXP2400 network processor is a member of the Intel's second-generation network processors family. It is designed to perform a wide range of functionalities, including multi-service switches, routers, broadband access devices and wireless infrastructure systems. The IXP2400 is an evolution of the first-generation Intel IXP1200 and it is a fully programmable network processor that implements a high-performance parallel processing architecture on a single chip suitable for processing complex algorithms, detailed packet inspection, traffic management and forwarding at the wire speed.

The Intel IXP2400 architecture combines a high-performance Intel XScale core with eight 32-bit independent microengines MEv2 (connected in two clusters of four) that cumulatively provide more than 5.4 giga-operations per second. The Intel XScale core is a general-purpose 32-bits RISC processor (ARM

Version 5 Architecture compliant) used to initialize and manage the NP, to handle exceptions, and to perform slow path processing and other control plane tasks. For more details about Intel IXP2400 NP the reader may refer to [18,19].

Since the Intel IXP2400 NP is hosted by an external hardware circuit, in order to accurately design a packet classifier, acquisition of the sizes of the available external memories is very important. We adopt the Radisys ENP-2611 [20] hardware circuit, equipped with 8 MB SRAM and 256 MB DRAM.

For example, in the case of classifying Edonkey traffic by static rules, three classification rules are to be defined:

(1) The 2-octet type field in frame contains 0800_{16}
(2) The 1-octet type field in the IP datagram contains 6
(3) The 2-octet destination port field in the TCP segment contains 4662 in software classification model, the conditional statement in the pseudocode about the classification can be written as:

```
(1) If((frame type==0x0800)&&(IP type==6)&&(TCP port==4662))
(2) declare the packet that match the classification;
(3) Else
(4) declare the packet that does not match the classification;
```

In order to further optimize the classification accuracy, one approach uses parallel hardware to avoid testing header fields sequentially. As shown in Fig. 5, when online traffic arrives the classifier, the hardware moves a packet header across a wide data path from memory to a dedicated register. When packet is moved to hardware register, the classifier extracts pertinent fields, concatenates the fields into a multi-octet value, and compares the resulting values with a constant. The classifier concatenates the fields into a five-octet value. And then compares the value to the dotted hexadecimal constant $08.00.06.12.36_{16}$. The value of the constant is derived directly from the classification rules. The hexadecimal constants 08.00 and 06 are specified explicitly; the 12.36, in the fourth and fifth octets, is the 2-octet hexadecimal equivalent of the decimal values 4662.

Once the header has been moved, hardware extracts specific octets and passes them to a Micro Engine (ME) based comparator circuit. If it finds a match between header values and the predefined constant, the comparator recognize it as a P2P traffic; otherwise, the comparator identify it as an exception.

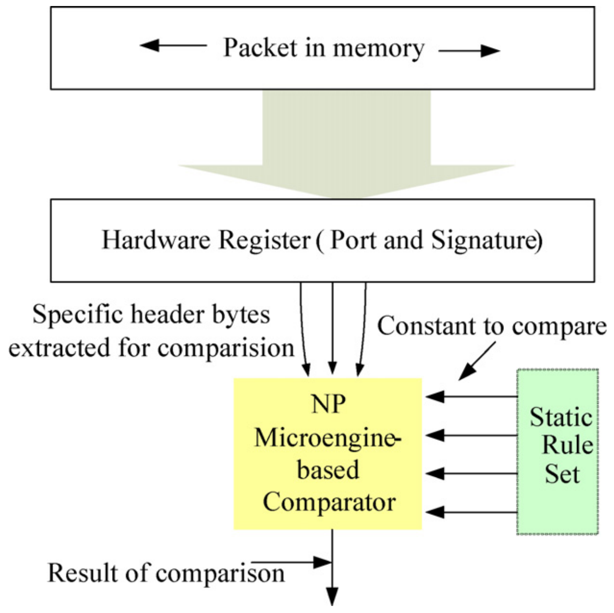

Packet in memory
Hardware Register (Port and Signature)
Specific header bytes extracted for comparision
Constant to compare
NP Microengine-based Comparator
Static Rule Set
Result of comparison

Fig. 5. Hardware classifier architecture for static classification.

4.2. The FNT based software classifier

4.2.1. The FNT model

For the dynamic statistical feature of P2P traffic, machine learning technology is a potential useful solution for effective identification. Based on payload independent statistical features, such as flow size features, connection features, protocol features. a formalized P2P feature set was created. Owning to the characters of simple, fast,high identify rate and good ability of selecting features, the FNT (Flexible Neural Tree) model [3,21,22] is selected as the machine learning tool. Based on the constructed feature set and selected machine leaning method, A software classifier is constructed.

The Flexible Neuron Instructor and FNT model are described as follows: the function set F and terminal instruction set T used for generating a FNT model S is described as

$$S = F \cup T = \{+_2, +_3, \ldots, +N\} \cup \{x_1, \ldots, x_n\} \tag{5}$$

where $+_i (i = 2, 3, \ldots, N)$ denote non-leaf nodes' instructions and taking i arguments. $x_1, x_2, \ldots, x_n$ are leaf nodes' instructions and taking no other arguments. The output of a non-leaf node is calculated as a flexible neuron model. From this point of view, the instruction $+_i$ is also called a flexible neuron operator with i inputs.

In the creation process of neural tree, if a non-terminal instruction, i.e., $+_i (i = 2, 3, 4, \ldots, N)$ is selected, i real values are randomly generated and used for representing the connection strength between node +I and its children. In addition, two adjustable parameters a_i and b_i are randomly created as flexible activation function parameters. Some examples of flexible activation functions are shown in Table 2. For developing the FNT classifier, the flexible activation function used is

$$f(a_i, b_i, x) = e^{-((x-a_i)/b_i)^2} \tag{6}$$

The output of a flexible neuron $+_n$ is calculated as follows. The total excitation of $+_n$ is

$$net_n = \sum_{i=1}^{n} w_i x_j \tag{7}$$

where $x_j (j = 1, 2, \cdots, n)$ are the inputs to node $+_n$. The output of the node $+_n$ is

$$out_n = f(a_n, b_n, net_n) = e^{-((net_n - a_n)/b_n)^2} \tag{8}$$

A fitness function maps FNT to scalar, real-valued fitness values that reflect the FNT's performances on a given task. Firstly, the fitness functions should be seen as error measures, i.e., MSE or RMSE. A secondary non-user-defined objective for which algorithm always optimizes FNTs is the size of FNT usually measured by number of nodes. Among FNTs having equal fitness values smaller FNTs are always preferred. In this research, the fitness function used for the Probabilistic Incremental Program Evolution

Table 2
The number of samples of each class.

Class	Training	Testing
HTTP	772	772
HTTPS	428	428
FTP	182	182
POP3	278	278
SMTP	247	247
Others	361	361

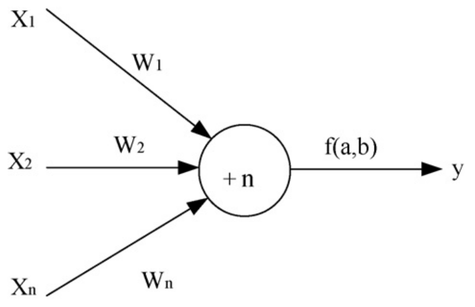


Fig. 6. A flexible neuron operator.

(PIPE) and Simulated Annealing (SA) is given by Mean Square Error (MSE).

A typical flexible neuron operator and a neural tree model are illustrated in Figs. 6 and 7. The overall output of flexible neural tree is computed from left to right by depth-first method, recursively.

For optimal design of FNT, a tree structural evolutionary algorithm, Probabilistic Incremental Program Evolution and Particle Swarm Optimization algorithms (PSO) are employed. Reader may refer to [3,22] for more technical details.

4.2.2. The optimization of FNT model

The optimization of FNT including the tree-structure and parameter optimization. Finding an optimal or near-optimal neural tree is formulated as a product of evolution. A number of neural tree variation operators are developed as follows: mutation five different mutation operators were employed to generate offspring from the parents. These mutation operators are as follows:

(1) Changing one terminal node: randomly select one terminal node in the neural tree and replace it with another terminal node.
(2) Changing all the terminal nodes: select each and every terminal node in the neural tree and replace it with another terminal node.
(3) Growing: select a random leaf in hidden layer of the neural tree and replace it with a newly generated subtree.
(4) Pruning: randomly select a function node in the neural tree and replace it with a terminal node.
(5) Pruning the redundant terminals: if a node has more than 2 terminals, the redundant terminals should be deleted.

Crossover select two neural trees randomly and select one nonterminal node in the hidden layer for each neural tree randomly, and then swap the selected subtree. The crossover operator is implemented with a pre-defined a probability 0.3 in this study. Selection evolutionary programming (EP) style tournament selection was applied to select the parents for the next generation [28]. Pairwise comparison is conducted for the union of one parents and one offsprings. For each individual, q opponents are chosen uniformly at random from all the parents and offspring. For each comparison, if the individual's fitness is no smaller than the opponent's, it receives a selection. Select 1 individuals out of parents and offsprings, that have most wins to form the next generation. This is repeated for each generation until a predefined number of generations or when the best structure is found. Parameter optimization by MA parameter optimization is achieved by the MA algorithm as described in Section 2. In this stage, the architecture of FNT model is fixed, and it is the best tree developed during the end of run of the structure search. The parameters (weights and flexible activation function parameters) encoded in the best tree formulate an individual. The GA algorithm works as follows: (a) initial population is generated randomly. The learning parameters crossover and mutation probabilities in MA should be assigned in advance; (b) the objective function value is calculated for each individual; (c) implementation of the local search, selection, crossover and mutation operators; (d) if maximum number of generations is reached or no better parameter vector is found for a significantly long time (100 steps), then stop, otherwise go to step (b).

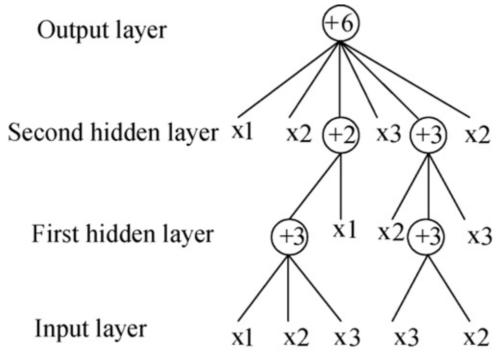


Fig. 7. A typical representation of the FNT with function instruction set $F = +_2; +_3; +_4; +_5; +_6$, and terminal instruction set $T = x_1; x_2; x_3$.

4.2.3. Feature/input selection with FNT

It is often a difficult task to select variables (features) for the classification problem, especially when the feature space is large. A fully connected NN classifier usually cannot do this. In the perspective of FNT framework, the nature of model construction procedure allows the FNT to identify important input features in building an P2P classifier that is computationally efficient and effective. The mechanisms of input selection in the FNT constructing procedure are as follows: (1) initially the input variables are selected to formulate the FNT model with same probabilities; (2) the variables which have more contribution to the objective function will be enhanced and have high opportunity to survive at next generation by an evolutionary procedure; (3) the evolutionary operators, i.e., crossover and mutation, provide a input selection method by which the FNT should select appropriate variables automatically.

4.2.4. FNT based traffic classification

For soft classification evaluation, the network traffic classification is selected for the test. The source of test data is the public domain available packet trace called Auckland IV. The Auckland IV trace contains only TCP/IP headers of the traffic going through the University of Auckland's link to the Internet. We used a subset of the Auckland IV trace from 20 February 2001 at 21:01:22 to 21 February 2001 at 02:00:00. There are 9,575,122 packets in this trace.

The public domain available Auckland IV traces include no payload information. Thus, to determine the connections "true" classifications port numbers are used. For this trace, we believe that a port-based classification will be largely accurate, as this archived trace predates the widespread use of dynamic port numbers. The classes considered for the Auckland IV data sets are HTTP, HTTPS, FTP (control), SMTP, POP3 and THE-OTHERS. Since the flows number of P2P applications in this trace was not enough to form the data set, we did not consider it.

For evaluation within a reasonable amount of time, the data set we used was a random subset from the transformed statistical flow information data set. We extracted samples randomly from data sets and formed two new data sets: one for training and another for testing. There are 2268 samples in training data set and 2268 samples in testing data set. This subset provided sufficient connection samples to build our model. There are six classes in the data set: HTTP, HTTPS, FTP (control), POP3, SMTP and THE-OTHERS. Table 2 illustrates the sample distribution over these classes in both data sets:

The statistical flow characteristics considered include: number of packets in each direction, mean packets length in each direction, variance of packets length in each direction and duration. Our decision to use these seven characteristics was based primarily on the previous work done by Zander et al. [14].

4.2.5. Result evaluation

In order to evaluate the different experiments, two criteria of Training Accuracy (TA), Accuracy of Classed Samples (ACS) are defined as follows:

$$TA = \frac{Number^{correct}_{training}}{Number^{total}_{training}} \tag{9}$$

where $Number^{total}_{training}$ is defined as the number of total samples in training data set. $Number^{right}_{training}$ is defined as the number of right classified samples in training data set.

$$ACS = \frac{Number^{correct}_{test}}{Number^{classified}_{test}} \tag{10}$$

where $Number^{classified}_{test}$ is defined as the number of classified samples in test data set.

Performance of the proposed method after eight independent run is depicted in Table 3. The Min and Max is defined as the least and maximum accuracy and *M* is the mean accuracy of each class. As evident from Table 3, for both training and test data sets, the accuracy of FNT classification accuracy is much higher than that of a direct neural network approach. Experiment results combined with other related works [21,22] illustrate that the FNT model is suitable for network traffic classification.

Table 3
Accuracy information of different network traffic class.

Class	Standard	Neural network		FNT	
		TA	ACS	TA	ACS
HTTP	MIN	77.84%	77.09%	81.83%	82.69%
	MAX	79.03%	79.52%	84.99%	85.43%
	M	78.46%	77.90%	**83.12%**	**84.19%**
HTTPS	MIN	81.31%	78.22%	83.42%	81.34%
	MAX	82.98%	83.11%	85.78%	86.22%
	M	82.10%	81.26%	**84.39%**	**84.31%**
FTP	MIN	93.12%	91.40%	95.41%	95.23%
	MAX	96.98%	96.47%	97.48%	97.30%
	M	93.45%	93.51%	**96.73%**	**96.16%**
PoP3	MIN	94.18%	94.00%	95.33%	94.80%
	MAX	96.16%	96.83%	96.82%	97.56%
	M	94.96%	96.00%	**96.17%**	**96.28%**
SMTP	MIN	96.38%	96.08%	97.06%	97.15%
	MAX	96.95%	97.31%	97.88%	97.79%
	M	96.71%	96.75%	**97.23%**	**97.35%**
Others	MIN	84.67%	83.88%	84.82%	84.78%
	MAX	85.20%	85.24%	86.34%	86.11%
	M	84.86%	84.40%	**85.13%**	**85.16%**

4.3. The combined hybrid classifier

After designing the hardware and software classifier, the combined hybrid classifier can work effectively as illustrated in Fig. 8. When the online traffic flows into the classifier, part of the traffic with static features can be identified by the hardware firstly (as described in Section 4). At the same time, it collects the designed statistical features of the identified traffic(as described in Section 3), which can be used as online training data set for the ML-based software classifier.

Part of hardware classifier ignored traffic will be classified by the software classifier. The DNS log can be obtained from the sever in real-time. The unrecognized packets are grouped into flows based on IP addresses, TCP or UDP ports and the flow characteristics (features) are computed. The flow data used for training each class must be representative for the particular network application. Especially for supervised learning algorithms, the flow data needs to be labeled with class labels corresponding to the network applications prior to training. The previously captured traffic traces, which may be from a special network (Fig. 9) or merged from the hardware classifier results can greatly satisfy this requirement. The role of the expert manager is a person who has the ability to find out the static rule from the software classified traffic. With the help of an expert manager, it will returns the identifier rule to the NP based hardware to filter the traffic after the software classifier identified the traffic correctly. It should be noted that at the beginning, small fraction of the traffic cannot be

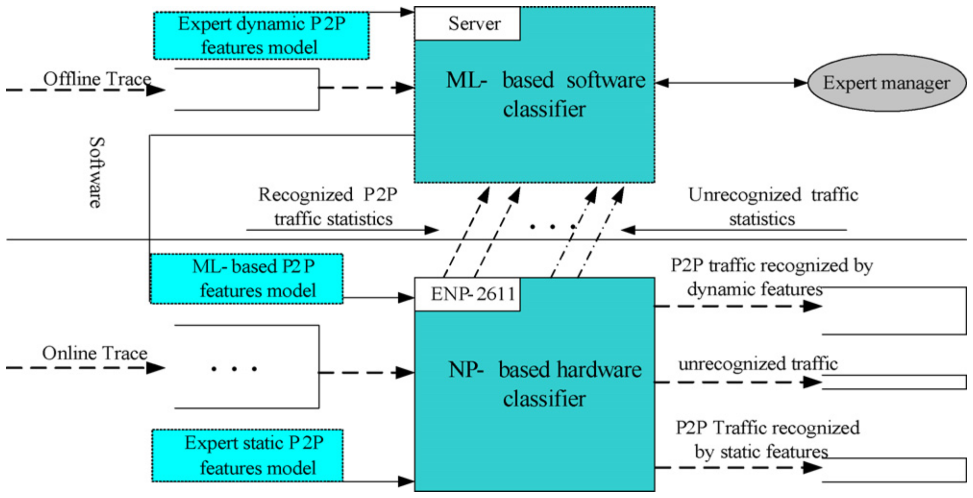
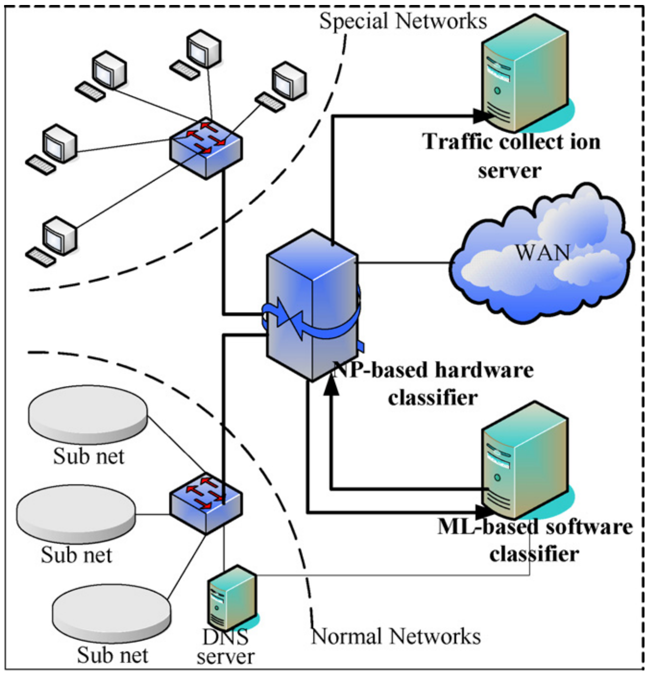

Expert dynamic P2P features model
Server
Offline Trace
ML- based software classifier
Expert manager
Software
Recognized P2P traffic statistics
Unrecognized traffic statistics
ML- based P2P features model
ENP-2611
P2P traffic recognized by dynamic features
Online Trace
NP- based hardware classifier
unrecognized traffic
Expert static P2P features model
P2P Traffic recognized by static features

Fig. 8. Hybrid classifier architecture.



Special Networks
Traffic collect ion server
WAN
NP-based hardware classifier
ML-based software classifier
Sub net
Sub net
Sub net
DNS server
Normal Networks

Fig. 9. The developed hybrid classifier prototype system in the Network Research Laboratory of University of Jinan.

handled using this method, especially for new P2P traffic. Once the classifier has been trained new flows can be identified based on their statistical attributes. The expert manager can monitor the hybrid classifier according to the requirements and can add, delete or modify the classification knowledge to ensure the classification accuracy, reduce the classification FP (False Negative) and enhance the TP (True Positive).

5. Experiments and evaluation

5.1. Experiment environment

Experiments were designed to evaluate our methodology (Fig. 9). The architecture includes five special hosts only running given P2P clients and some other normal hosts without running any P2P clients in the Network research laboratory of University of Jinan. The special networks only run the given P2P clients: BitTorrent(BT), kazaa, Edonkey, PPlive and skype. The normal networks work in the normal way without running any P2P clients but performing normal application, such as web browsing, ftp, e-mail and playing games. Traffic from special network is default seen as P2P traffic and traffic from normal network is default seen as non-P2P traffic. The proposed classifier is put at the gateway of the campus network. At the same time, the synchronous DNS query log is transported to the software classifier. In order to validate the experiment results, the classified traffic were stored in traffic collection server.

5.2. Data set analysis

In order to get the real traffic data, A hybrid classifier prototype is developed in the Network Research Laboratory of University of Jinan (Fig. 9). The experiments were arranged in three stages during three days (3 h in each day). The collected data (D1–D3) are described in Table 4. At the first stage, the static features of Edonkey and Bittorent was input to the hardware classifier and the software classifier without any statistical features. In this stage,

Table 4
The collected traffic data set (D1 on 25 November 2006,D2 on 26 November 2006, D3 on 27 November 2006).

Data	P2P	P2P type	During (h)	D1 (MB)	D2 (MB)	D3 (MB)
D20	100	Kazaa	3	21.21	41.31	88.34
D21	100	Edonkey	3	401.15	511.36	438.56
D22	100	BiTorent	3	320.77	430.18	302.66
D23	100	PPlive	3	55.38	110.20	37.72
D24	100	Skype	3	21.31	36.38	18.56
D25	0	Non-P2P	3	310.19	390.27	789.11

there was also a DNS log including 302 records got from the DNS server. Based on the features defined in Section 3.2, the identified P2P data was normalized to four elements records (f_{pro}, $f_{IP-Port}$, $f_{U\,port}$, f_{DNS}) in each 5 min.

Then the software classifier trains itself by the normalized data set in first stage. The selected data records of (f_{pro}, $f_{IP-Port}$, $f_{U\,port}$, f_{DNS}) were used as the inputs (x_0, x_1, x_2, x_3) of FNT. After the optimization of the tree structure by PIPE algorithms and optimization of tree weights by PSO algorithms, FNT performance is illustrated in Fig. 10. The output y from the FNT is a real number between 0 and 1. For a given IP, if y is a number between 0 and 0.5, then it may be seen as potential P2P host and related traffic may be potential P2P traffic. The smaller y is, the higher potential P2P host it is. Otherwise, it may be seen as non-P2P host when y lies between 0.5 and 1, lager y means higher potential of non-P2P host and the related traffic can be seen as potential non-P2P traffic.

In the 3 h of the second stage, the received DNS log records number is 356. After this stage, the identified P2P and non-P2P data are all normalized to four elements records in each five minutes. Then the train feature set include known P2P traffic and known non-P2P traffic features were obtained. The software classifier retrains itself from the normalized data set in the second stage. In the third stage, the received DNS log records number is 388.

5.3. Result evaluation

The True Positive, False Negative and total accuracy standard were used to evaluate the experiment result. When calculating the total accuracy, it is the summation of the hardware classifier

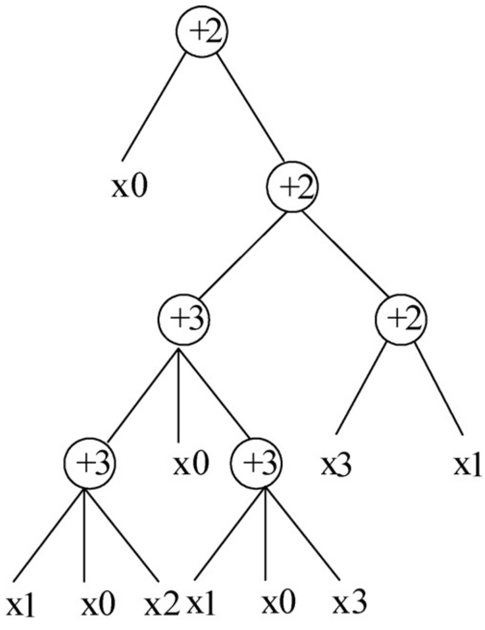


Fig. 10. The optimized FNT tree.

Table 5
Test results of the first stage.

Stage	Traffic	TP (%)	FP (%)	Total accuracy (%)
First	Non-P2P	98.15	1.61	69.71
First	P2P	98.89	29.63	69.71
Second	Non-P2P	90.67	6.61	93.37
Second	P2P	94.39	5.33	93.37
Third	Non-P2P	94.67	5.61	95.67
Third	P2P	96.49	4.31	95.67

Table 6
Comparision of the hybrid method. Port and payload based method means the related static information. Transport-L is based on transport layer information and ML is the machine leaning method.

Method	Port	Payload	Transport-L	ML	Hybrid
Application signature	No	Yes	No	No	Yes
Private info	No	Yes	No	No	Yes
Need port info	Yes	No	Some	Yes	Yes
Accuracy	Low	High	High	High	High
Identify encrypted	Some	No	Yes	Yes	Yes
Identify unknown P2P	No	No	Yes	Yes	Yes
Expansibility	No	No	Middle	High	High
Online learning ability	No	No	No	No	Yes

accuracy and the software classifier accuracy to a tested traffic flow. The classified result is depicted in Table 5. At the first stage, P2P identification and FP is high (29.63%) and the total accuracy is very low (69.71%). But at the second stage, because the software learned the statistical knowledge of identified P2P traffic from first stage, the total accuracy increased to 93.37%, and the P2P identification and FP decreased to 5.33%. After retraining, the total accuracy in third stage increased to 95.67%, TP up from 94.39% to 96.49%, and the P2P identification and FP also decreased from 5.61% to 4.31%. As evident from the experiment results, the hybrid classifier can learn to identify P2P traffic with high accuracy but only need small set of static features at the beginning. Most importantly, the proposed method can identify the hosts that run unknown P2P clients and the unknown P2P traffic, such as Maze (one of the most popular P2P application in CERNET) in our further tests.

As experiment experience, the given time space can greatly affect different normalized feature value, which is an important factor for the software identification rate. It needs adjustment and retest for deciding the suitable time space for the entire feature. FP dependent factor should be further investigated in future work. In Table 6 illustrate the performance of the hybrid classifier and other methods. The proposed model has the ability to learn online and retrains itself by the newly obtained data sets at the same time. Specially, the easy extendability is one of the most important highlights of the proposed method.

6. Conclusions

P2P application is becoming more and more prominent. In the past, some methods were used to identify the P2P traffic, but most of them were tested and evaluate by offline traffic. In reality, offline identification is not helpful for online controlling and managing mainly due to the performance reason. These facts lead to the adoption of high performance and flexible network components. In this paper, a network processors based online hybrid traffic classifier was designed. The hardware classifier can classify P2P traffic by the static characteristics of online speed, and the Flexible Neural Tree based software classifier could help learning and selection of P2P traffic attributes from the statistical characteristics. The selected attributes can be further returned to NPs and be handled with high performance.

Experiment results clearly illustrate that the hybrid classifier can be competent for classifying P2P traffic from gigabit network stream online. It also shows good extension ability to add new P2P features, to learn and find new P2P applications. As evident from the experiments, the proposed approach seems to be very promising. For further research, we try to improve the intelligence with the help of some clustering algorithms.

Acknowledgments

This research was supported by the National Natural Science Foundation of China Grant 60873089, National Natural Science Foundation of China Grant 60773109 and the Natural Science Foundation of Shandong Province of China Grant Z2006G03. Ajith Abraham is supported by the Centre for Quantifiable Quality of Service in Communication Systems, Centre of Excellence, appointed by The Research Council of Norway, and funded by the Research Council, NTNU and UNINETT.

References

[1] S. Sen, J. Wang, Analyzing peer-to-peer traffic across large networks, IEEE/ACM Transactions on Networking (2004) 219–232.
[2] E.C. Douglas, Network System Design Using Network Processor, Prentice-Hall, 2003, pp. 115–126.
[3] Y. Chen, B. Yang, J. Dong, Nonlinear systems modelling via optimal design of neural trees, International Journal of Neural systems 14 (2004) 125–138.
[4] M. Izal, G. Urvoy-Keller, E.W. Biersack, P.A. Felber, A. Hamra, L. Garc'es-Erice, Dissecting BitTorrent: five months in a Torrent's lifetime, in: Proceeding of PAM'04, 2004, 267–277.
[5] P. Karbhari, M. Ammar, A. Dhamdhere, H. Raj, G. Riley, E. Zegura, Bootstrapping in Gnutella: a measurement study, in: Proceeding of PAM'04, 2004, 189–201.
[6] K. Tutschku, A Measurement-based traffic profile of the eDonkey filesharing service, in: Proceeding of PAM'04, 2004, 137–149.
[7] K. Thomas, B. Andre, F. Michalis, K. Claffy, Transport layer identification of p2p traffic, in: Proceedings of IMC'04, 2004, pp. 121–134.
[8] S. Subhabrata, S. Oliver, 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, pp. 512–521.
[9] R. Matthew, S. Subhabrata, S. Oliver, D. Nick, Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification, in: Proceedings IMC'04, 2004, pp. 135–148.
[10] K. Thomas, P. Konstantina, F. Michalis, Blinc: multilevel traffic classification in the dark, ACM SIGCOMM 35 (2005) 229–240.
[11] W. Nigel, Z. Sebastian, A. Grenville, Evaluating Machine Learning Methods for Online Game Traffic Identification, (CAIA) Technical Report 060410C, 2006.
[12] W. Nigel, Z. Sebastian, A. Grenville, Evaluating Machine Learning Algorithms for Automated Network Application Identification, (CAIA) Technical Report 060410B, 2006.
[13] A. McGregor, M. Hall, P. Lorier, J. Brunskill, Flow clustering using machine learning techniques, in: Proceedings of PAM'04, 2004, pp. 19–20.
[14] S. Zander, T. Nguyen, G. Armitage, Automated traffic classification and application identification using machine learning, in: Proceedings of LCN'05, 2005, pp. 5–17.
[15] http://www.securityfocus.com/infocus/1843/3, Access time: May 15, 2006.
[16] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim, A survey and comparison of peer-to-peer overlay network schemes, Journal of IEEE Communications Survey and Tutorial 7 (2005) 72–93.
[17] K. Keutzer, S. Malik, R. Newton, J. Rabaey, A. Sangiovanni-Vincentelli, System level design: orthogonalization of concerns and platform-based design, IEEE Transactions on Computer-Aided Design of Circuits and Systems 19 (2000).
[18] Intel Corporation:Intel IXP2400 Network Processor. http://www.intel.com/design/network/prodbrf/27905302.pdf, access time: October 20, 2006.
[19] J.J. Erik, R.K. Aaron, IXP2400/2800 Programming, Intel Press, 2003, p. 12.
[20] RadiSys Corporation: ENP-2611 Data Sheet. http://www.radisys.com, Access time: November 20, 2006.
[21] Y. Chen, B. Yang, J. Dong, A. Abraham, Time-series forecasting using flexible neural tree model, Information Science 174 (2005) 219–235.
[22] Y. Chen, B. Yang, A. Abraham, Feature selection and classification using flexible neural tree, Neurocomputing 70 (2006) 305–313.
[23] S. Shieh, F. Lee, Y. Lin, Accelerating network security services with fast packet classification, Computer Communications 27 (2004) 1637–1646.

[24] Y. Wang, S. Ye, Y. Tseng, A fair scheduling algorithm with traffic classification for wireless networks, Computer Communications 28 (2005) 1225–1239.
[25] D. Pao, Y.K. Li, P. Zhou, Efficient packet classification using TCAMs, Computer Networks 50 (2006) 3523–3535.
[26] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, C. Williamson, Offline/realtime traffic classification using semi-supervised learning, Performance Evaluation 64 (9–12) (2007) 1194–1213.
[27] F. Qian, G. Hu, X. Yao, Semi-supervised internet network traffic classification using a Gaussian mixture model, AEU - International Journal of Electronics and Communications, 62 (2008) 557–564.
[28] K. Chellapilla, Evolving computer programs without subtree crossover, IEEE Transactions on Evolutionary Computation 1 (1997) 209–216.
[29] L.E. Rocha-Mier, L. Sheremetov, I. Batyrshin, Intelligent agents for real time data mining in telecommunications networks, in: V. Gorodetsky, et al. (Eds.), Lecture Notes in Computer Science, vol. 4476, Springer-Verlag, 2007, pp. 138–152.
[30] D. Montana, T. Hussain, Adaptive reconfiguration of data networks using genetic algorithms, Applied Soft Computing Journal 4 (2004) 433–444.
[31] S. Chandramathi, S. Shanmugavel, Estimation of cell loss probability for self-similar traffic in ATM networks—a fuzzy approach, Applied Soft Computing 3 (2003) 71–83.
[32] L. Cohena, G. Avrahamia, M. Lasta, A. Kandelb, Soft computing for dynamic data mining info-fuzzy algorithms for mining dynamic data streams, Applied Soft Computing 8 (2008) 1283–1294.
[33] M. Awada, Y. Motaia, Dynamic classification for video stream using support vector machine, Applied Soft Computing 8 (2008) 1314–1325.
[34] L. Cohena, G. Avrahami-Bakisha, M. Lasta, A. Kandelb, O. Kipersztokc, Real-time data mining of non-stationary data streams from sensor networks, Information Fusion 9 (2008) 344–353.
[35] S. Purwar, I.N. Kar, A.N. Jha, On-line system identification of complex systems using Chebyshev neural networks, Applied Soft Computing 7 (2007) 364–372.
[36] G. Zhang, G. Xie, J. Yang, Y. Min, Z. Zhou, X. Duan, Accurate online traffic classification with multi-phases identification methodology, in: Proceedings of CCNC'08, 2008, pp. 141–146.