Chapter 6

Automating Network Protocol Identification

Ryan G. Goss and Geoff S. Nitschke

Contents

The proliferation of computer network users has, in recent years, placed a strain on network resources, such as bandwidth and number allocations. This issue is more apparent where connectivity is limited, such as in developing countries. The provisioning of services over these congested resources needs to be managed, ensuring a fair *quality of experience* (QoE) to consumers and producers alike. *Quality of service* (QoS) techniques used to manage such resources require constant revision, catering for new application protocols introduced to the network on a daily basis. This research proposes an efficient, autonomous method for distinguishing application protocols through the use of a *dynamic protocol classification system* (DPCS). Using this method, the burden of signature creation is reduced, while the accuracy achieved in application protocol identification increases.

6.1 Introduction

Computer networks have grown substantially in recent years, due in part to the increasing global reach of the Internet, network access speeds, and content availability. This growth continues to spur the development of numerous applications, using both client–server and *peer-to-peer* (P2P) communication architectures. Each application uses a specific application protocol, generating a number of flows in order to exchange data. In computer networks, a flow describes a sequence of packet exchanges between hosts, uniquely identified by its 5-tuple identifier: *source IP address*, *destination IP address*, *source port*, *destination port*, and *protocol identifier*. In order to identify the underlying application protocol of a flow, a unique signature is required.

Network administrators traditionally used packet header information, such as source and destination ports and protocol, to formulate signatures and classify flows on a network. As this information is configured by the two communicating hosts, alternative port and protocol information can be negotiated, effectively negating the effect of port-based filters and restrictions (Alshammari and Zincir-Heywood 2008). This issue is problematic, since an increasing number of application developers seek to evade classification. P2P application protocols exacerbate this problem by operating in a decentralized manner, dynamically selecting the ports and protocols on which they communicate (Auld et al. 2007). The inefficiencies associated with classic port-based classification forced industry and researchers alike to consider a number of alternatives, such as *deep packet inspection* (DPI) and statistical analysis.

DPI has become an essential tool for network engineers, enabling them to search both packet header and payload (content) for predefined application protocol signatures (Huang and Zhang 2008). These signature matches are often performed using regular expressions in software or through the use of specialized hardware, such as a *field-programmable gate array* (FPGA) (Huang and Zhang 2008). While DPI yields excellent results in identifying plaintext flows, encryption renders the content of packets opaque and thus the use of DPI inept. Statistical analysis addresses the problem of opaque, encrypted flows, by inferring the application (Gebski et al. 2006) or application class (Auld et al. 2007; Li et al. 2007; Moore and Papagiannaki 2005) of a flow by examining statistical information over a number of packet exchanges. The advantage of statistical analysis is that regardless of how applications attempt to disguise themselves, through encryption or port randomization, the characteristics exhibited over their packet exchanges remain intact.

The definition and provisioning of signatures is often a service provided by the vendors of traffic management devices, through the supply of updated signature packs. These signature packs need to be regularly updated in order to keep up with advances in application protocol development. According to Szabo et al. (2007), this is one of the most significant problems associated with signature-based systems, since each vendor is responsible for the creation of their own proprietary signature packs, compatible with their systems. Furthermore, the attributes, or discriminators, extracted to describe a network flow and the mechanism employed to distinctly identify them are often a closely guarded secret. For these reasons, accuracy and performance variances between vendor equipment are not uncommon. The process of creating and deploying signatures in this manner is suboptimal, as the development of new application protocols far exceed signature production by vendors. This results in a significant amount of network flows remaining unclassified or inaccurately classified until an update is released. This, in turn, results in network administrators being unable to manage the network resources at their disposal.

Given these issues, this research proposes a *dynamic protocol classifier* (DPC) method to address the two major issues pertinent in flow classification, *accuracy* and *automation*. Automation refers to methods that automate input from a user (manual annotation for new applications) or vendor

(signature packs written for new classifiers). Using the proposed DPC method, new application protocols are automatically discovered in a training data set using a select set of discriminators. These discovered protocols are subsequently used to train artificial neural network (ANN) classifiers, in order to identify future instances of the protocols. Accuracy refers to the ability of such an automated method to discriminate between different application protocols and noise in training and test data, as well as the ability for trained ANN classifiers to correctly identify future instances of the protocol.

Definition of "ANN" correct?

In order to increase the efficacy of clustered training data, the DPC method includes the *density-bases spatial clustering of applications with noise* (DBSCAN) method as its unsupervised learning component. DBSCAN is capable of forming arbitrarily shaped clusters to deal with noise in the data. We thus hypothesize that the DPC method will outperform (with statistical significance), for this case study, the accuracy achieved by a *hierarchical self-organizing map* (HSOM) (Goss and Nitschke 2013a) and *k*-means (Goss and Nitschke 2013b) method.

6.2 Related Work

Machine learning (ML) has been the subject of much research in classifying network traffic using flow statistics inferred from a network (Alshammari and Zincir-Heywood 2009; Auld et al. 2007; Bernaille et al. 2006). These statistical features, or discriminators, are calculated over multiple packet exchanges. ML classifiers are trained to associate particular discriminator sets, or patterns, with known traffic classes. The classifiers are then able to identify and differentiate future unclassified flows using previously learned rules (Nguyen and Armitage 2008). Hu and Shen (2012) compare the efficiency of several ML techniques, providing an overview of recent advances in this area. Hu and Shen (2012) present algorithms for the creation of specific feature sets and classification models, comparing the efficiency of each. The authors considered a number of classification methods, including *genetic algorithms* (GA) and Bayesian classification, as well as unsupervised clustering methods including *expectation maximum* (EM) clustering (Dempster et al. 1977), and *k*-means clustering (MacQueen et al. 1967).

Two key metrics for measuring the performance of a traffic classification system are accuracy and completeness (Szabo et al. 2007). As such, the discriminators used to train classifiers and those used to test existing classifiers are significant. The quality of the discriminators used to describe a flow is crucial to the performance of an ML method (Nguyen and Armitage 2008) and has thus been the topic of research, including that of Auld et al. (2007), Bernaille et al. (2006), Este et al. (2008), Gargiulo et al. (2009), Moore and Papagiannaki (2005), Moore et al. (2005), Mcgregor et al. (2004), Huang and Zhang (2008), Alshammari and Zincir-Heywood (2007), and Li et al. (2007). As the number of applications encrypting their communications rises (Nascimento et al. 2013), the dependency on DPI wanes. Instead, the focus has shifted toward discriminators that remain effective through plaintext and encrypted flows. For example, Mcgregor et al. (2004) examine plots of packet size against packet *interarrival times* (IATs) for a number of flows. The authors concluded that the results of these plots were indicative of the application type. In addition to these discriminators, the authors described the flows through the extraction of byte counts, connection duration, number of transitions between transaction and bulk transfer modes, and the amount of time spent in the idle state. These characteristics were used by an EM algorithm, which grouped them into a small number of clusters.

Bernaille et al. (2005) dispute the use of discriminators such as IAT, due to the influence of network load and *transport control protocol* (TCP) acknowledgements. Instead, the authors use the direction and size of each packet, recorded over a number of packet exchanges. Each flow record was transformed into a *hidden Markov model* (HMM). Thereafter, *spectral clustering* (Von Luxburg

2007) was used to find clusters in the likelihood space. The results of this work demonstrated a system capable of recognizing behavioral characteristics of a flow, with an accuracy performance of 90%, given observation of as little as the first several packets.

Li et al. (2007) demonstrate the classification of Internet traffic using a *support vector machine* (SVM) (Cortes and Vapnik 1995). The authors do so using 19 distinct discriminators, including the total number of packets in the flow, average packet size, duration of the flow, packet and byte ratios, and the average window size. Results showed a 99.41% degree of accuracy.

Alshammari and Zincir-Heywood (2009) focused on identifying the *Secure Shell* (SSH) and *Skype* application protocols. They do so by comparing five different ML techniques, including *AdaBoost* (Freund and Schapire 1995), SVM, naive Bayesian (Devroye 1996), RIPPER (Cohen 1995), and C4.5 (Quinlan 1993). Using discriminators such as various packet IAT measurements, forward and backward packet size information, and the duration of the flow, the authors were able to highlight C4.5 as the highest-performing algorithm, scoring approximately 97% in the best-case test scenario.

Goss and Botha (2012) demonstrate the utility of a generic discriminator set that is suitable for identifying distinct application protocols at an early stage of their existence. In this work, ANN classifiers were trained using manually annotated data sets to identify application protocols with a 99% degree of accuracy. Goss and Nitschke (2013a) extend this work, using an HSOM to automate the manual annotation process. Each cluster identified by the HSOM corresponds to a specific application protocol within the data set. Features of each cluster were subsequently used to train an ANN classifier to identify future instances of the protocol. Testing these classifiers resulted in a 98% degree of accuracy.

Goss and Nitschke (2013b) use *k*-means to cluster the recorded data sets. Dissimilar to *k*-means implementation described by Hu and Shen (2012), where the number of clusters, *k*, is preset, Goss and Nitschke (2013b) use an *evolutionary algorithm* (EA) (Eiben and Smith 2003) to approximate the best value of *k*. The EA uses the *silhouette cluster evaluation* method (Rousseeuw 1987), as part of the fitness function for evaluating each *k* value. Although a high degree of accuracy was demonstrated, the *k*-means algorithm is still not the most appropriate clustering method, due to its inability to find nonlinearly separable clusters (Jain 2010).

However, both the *k*-means and HSOM algorithms suffer from this problem and are thus susceptible to outlying points skewing the clustering results. The inability to detect and manage outliers renders the HSOM and *k*-means algorithms unsuitable for operation in noisy data sets, such as those describing live network traffic. The emergent spherical clusters indicative of *k*-means and HSOM clustering are highly susceptible to noise in the data. Furthermore, multiple passes on each datum are required by each of these algorithms as part of their clustering process, reducing the overall efficiency of these methods. Classifier efficiency is important to ensure completeness of the system and to ensure that the system is always up-to-date. Thus, an alternative solution is needed, where nonspherical clusters with the ability to counteract the effect of noise emerge.

6.3 Method

To address the problematic issues associated with the HSOM and *k*-means methods, we propose a DPC method for automating the identification of application protocols as they traverse the network. The DPC method also used an EA to automate the tuning of clustering parameters as well as to determine the optimal structure of each ANN classifier. The research objective was to demonstrate increased efficiency of the DPC method comparative to related methods applied to the same task. For data sets of fabricated network traffic, the accuracy and efficiency of the

DPC method were compared to the HSOM (Goss and Nitschke 2013a) and *k*-means (Goss and Nitschke 2013b) methods. For clustering data sets describing live network traffic, DBSCAN (Ester et al. 1996), was tested as an alternative to the HSOM and *k*-means methods.

Figure 6.1 delineates a method for automated network flow classification via automatically training ANN classifiers to identify application protocols traversing the network. ANNs were

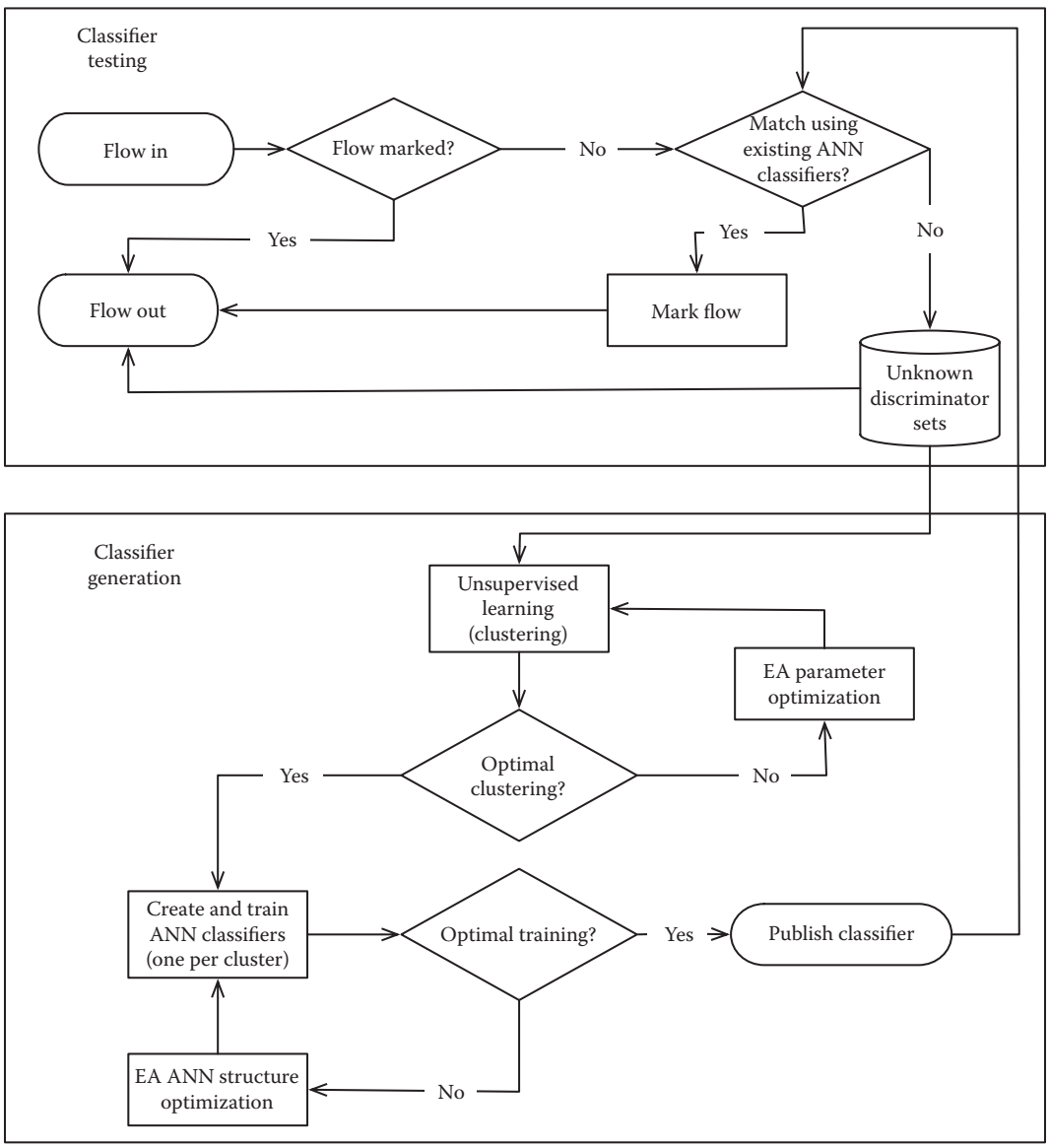

Figure 6.1 Bottom: An unsupervised clustering method is used to identify new protocols as clusters in the training data. An *artificial neural network* (ANN) classifier is trained to identify future instances of each cluster (application protocol). Top: Unknown protocols are broken down into their defining discriminator sets and placed in the training data, upon which the clustering method operates in subsequent iterations.

selected in favor of alternative supervised learning methods, such as decision trees, due to their ability to operate both as independent classifiers or in an ensemble. This allows a single classifier to be transported from one system to another without requiring all supporting classifiers to follow suit. The classifiers created by this method were subsequently used to test and mark each flow as it switched through a network. Critical to the method's success was the ability to describe a flow by its underlying application protocol. As each application communicates with a remote host using a unique pattern of byte exchanges, inferring the underlying application protocol of a flow is accomplished by examining certain statistical and payload characteristics. These characteristics, or discriminators, are common to both the training and testing data used by the method.

This research used a generic set of discriminators, capable of identifying new and previously observed application protocols, proposed and tested by Goss and Botha (2012). These discriminators were able to uniquely fingerprint the underlying application protocol of a flow early upon it entering the network. Early identification allows prompt prioritization of the flow and management of its resources as it transits the network. Resources refer to the amount of bandwidth (speed) and priority (order of preference when arriving at a router) allocated to a given application.

The discriminators included the *direction of the first four payload-bearing packets*, the *average size of these packets*, and the *numeric value of the first three bytes of payload in each direction*. The directionality of the flow was determined through observation of the *synchronization* (SYN) packet. A SYN packet is the first packet sent in a TCP communication when a new session is requested. The requirement for observation of a SYN packet therefore dictates that only flows utilizing the TCP protocol were considered.

It is envisaged that via modification of the base discriminator set, this method will also support user *datagram protocol* (UDP)–based flows. However, demonstration of this is outside the scope of this chapter.

After a predefined number of packet exchanges, the recorded discriminator sets for each flow were tested against existing classifiers. The order in which the classifiers were tested was based on previous results, with the most popular classifiers tested first. The popularity of each classifier was determined by the function $P(x) = f/t$ where x denotes the event of a successful match, f the number of previously successful matches, and t the number of trials. Evaluating a flow using the most probable classifiers reduces the number of tests required for classification, improving the overall efficiency of the system.

Discriminator sets belonging to unidentified flows were added to a database and the flow marked as *unknown*. These data were then clustered by the DBSCAN algorithm:

```
DBSCAN(D, eps, minPTS)
C = 0
for each unvisited point P in dataset D
mark P as visited
NeighborPts = regionQuery(P, eps)
if sizeof(NeighborPts) < minPTS
       mark P as NOISE
else
       C = next cluster
       expandCluster(P, NeighborPts, C, eps, minPTS)

expandCluster(P, NeighborPts, C, eps, minPTS)
add P to cluster C
for each point P' in NeighborPts
```

```
if P' is not visited
       mark P' as visited
       NeighborPts' = regionQuery(P', eps)
       if sizeof(NeighborPts') > = minPTS
       NeighborPts = NeighborPts joined with NeighborPts'
if P' is not yet member of any cluster
       add P' to cluster C

regionQuery(P, eps)
return all points within P's eps-neighborhood (including P)
```

DBSCAN requires two parameters for clustering, namely, the maximal distance between neighbors, *epsilon* (ε), and the minimum points per cluster (*minPTS*). The value for ε was restricted to $min(knn) < \varepsilon < max(knn)$, where *knn* was the distance of each datum to its nearest neighbor. The value of *minPTS* was in the range $D + 1 < minPTS < (|db|)/2$, where D denotes the number of dimensions of the input vector and *db* was the database of unknown discriminator sets.

A GA (Eiben and Smith 2003) tuned this clustering process, adjusting the values of ε and *minPTS* over a number of iterations. The GA's genotype was a bit-string that encoded the value of ε and *minPTS*. The initial genotype population consisted of randomly generated values. Using *silhouette cluster analysis* (Rousseeuw 1987), the GA's fitness function scored each genotype with a value in the range [–1, 1], according to how well DBSCAN clustered the training data with the given (encoded) ε and *minPTS* values. This score measured how tightly grouped the members of each cluster are and was thus a measure of the success of the clustering process. A score toward the value of 1 indicated more optimal clustering, while a lower score, the converse.

At each generation of the GA, *fitness proportionate selection* was used to select pairs of parents, where each pair of parents was recombined using one-point crossover (Eiben and Smith 2003) to produce one child genotype. This process was repeated until enough child genotypes had been produced to replace the previous population. *Elitism* was also applied such that the highest-scoring genotype was transferred to the next generation. *Bit-string mutation* was then applied to flip one bit of each child genotype with a 0.5 degree of probability (Table 6.1). The GA was run for 1000 generations (Table 6.1), after which time the fittest genotype was selected for the optimal ε and *minPTS*.

Each cluster within the optimized clustered data set describes a unique application protocol. As such, a classifier was required to identify future instances of the application. A separate ANN classifier was created for each cluster, with the data of the cluster used as the training set. Initially,

Table 6.1 EA Parameters

Parameter	*Initial Value*	*Start Range*	*End Range*
Generations	1000	–	–
Mutation Rate	0.5	–	–
Number of Genes	20	–	–
Crossover Point	Random	0	20
Epsilon (ε)	0.00002	0.00001	0.0001
minPTS	645	12	711

the number input nodes of each classifier ANN equals the magnitude of the discriminator set. The output of each classifier was a single node that returned the probability of a supplied input vector being a match.

The number of hidden layer nodes and the number of hidden layers was optimized by a GA. Each genotype was encoded as a bit-string, and the number of hidden layer nodes was set in the range [1, 5]. The fitness function was the classification accuracy of an ANN with n hidden nodes. The GA also employed fitness proportionate selection, generational replacement, one point crossover and bit-flip mutation in the same method as used to evolve the ε and *minPTS* parameters for the DBSCAN method.

6.4 Experiments

Three experiment sets were conducted to evaluate the classification of network traffic flows. The first experiment tested the capability of the unsupervised learning component (DBSCAN-based clustering) on a training data set. The second experiment set evaluated the EA optimization method that tunes DBSCAN clustering parameters and ANN topology of the ANN assigned to classify each cluster. The third experiment set verified the learnt classification behaviors of each of the ANN classifiers using a manually annotated data set recorded in a real-world network environment.

6.4.1 Experiment Set 1: Unsupervised Learning

Flow inspector software* was developed for recording discriminators from passing flows. This software observes and records information about traffic flows passing through the interfaces of the device on which it is deployed. The software made use of iptables and specifically the libipq module to pipe this information through a user-space daemon for analysis.

The software was configured to record 11 discriminators (determined by Goss and Botha 2012) for each flow, including the direction and average size of the first payload-bearing packets as well as the value of the first three bytes of payload in either direction. The software was deployed at a midsized corporate establishment, set to record flow information for a period of 1 h on a normal business day. At the outset, no predefined classifiers were made available, and therefore, all flows were sampled, with their discriminator sets appended to the training data set. In total, 1421 TCP flows were sampled and appended to the database for evaluation by DBSCAN.

A GA was applied to tune the DBSCAN clustering parameters ε and *minPTS* (Section 6.3), where the search space for ε was determined by plotting the distance to the nearest neighbor of each datum on a histogram (Figure 6.2). Figure 6.2 shows the average distance of each datum to its nearest neighbor, with the *knee* of the graph at an approximate value of 0.00005. The knee of the graph is indicative of the most likely value of ε, the point at which the distance between neighbors increases exponentially, due to the presence of outliers (noise). The search space for ε was therefore constrained to values between 0.00001 and 0.0001. Similarly, the search space for the *minPTS* parameter was configured such that $D + 1 <= minPTS <= (|db|)/2$, between 12 and 711. The GA parameters, their initial values, and the search space for each are listed in Table 6.1. This GA was executed for 1000 generations, where optimal ε and *minPTS* values of 0.00004 and 61, respectively, were found. The clustered data set consisted of six distinct clusters, with a silhouette

* Custom-designed by Ryan Goss. Details can be found at http://goo.gl/80BQE.

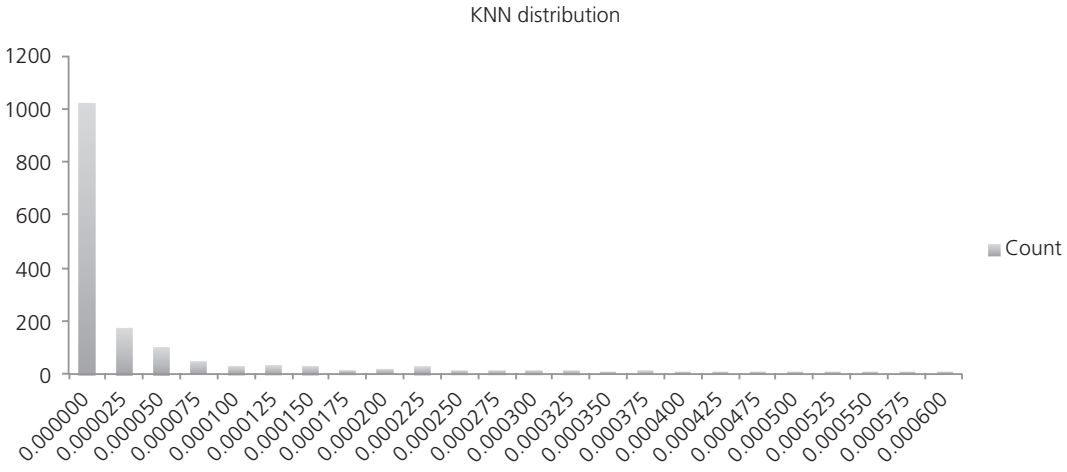

Figure 6.2 Histogram depicting nearest neighbors.

Table 6.2 Clustering Results

Cluster	*Datum*	*Silhouette*
1	127	0.993615
2	319	0.989984
3	101	0.997174
4	61	0.999032
5	175	0.996352
6	74	0.991109

value of 0.993615. The clusters, with their associated datum and silhouette values, are summarized in Table 6.2. The clusters were then saved for the second experiment set (Section 6.4.2).

6.4.2 Experiment Set 2: ANN Classifiers

Each of the six clusters identified by experiment set 1 (Section 6.4.1) were used as training sets by an ANN that attempted to classify the protocol represented by each cluster. The following pseudocode describes the ANN classification process, given the clustered data.

```
For each cluster c in [identified clusters]
1. Mark data linked to c with "1"
2. Mark data external to c with "0"
3. For i = 1 to EA_max_generations
       For each s genome in EA_population
              Create ANN a to identify c using structure s
              Train: 1000 iterations using marked datum
              If accuracy(a) > current_best_for_c
```

Table 6.3 EA Optimization Results

Cluster	*Hidden Layers*	*Neurons per Hidden Layer*	*Accuracy*
1	4	10	99.897%
2	4	11	99.983%
3	2	4	99.996%
4	2	11	99.992%
5	3	11	99.948%
6	4	11	99.970%

```
            current_best_for_c = a
            Fitness(s) = accuracy(a)
      Next s
Next i
4. Set c_ann = current_best_for_c
Next cluster
```

Each ANN was a fully connected feed-forward network with one hidden layer of log-sigmoidal nodes. The ANN has one output code (indicating the accuracy of classification within the range [0.0, 1.0]) that also uses a log-sigmoid activation function. The input layer consisted of 11 input neurons corresponding to the 11 discriminators used for classification (Section 6.4.1). A GA was used in order to determine an appropriate number of hidden layer nodes as well as number of hidden layers (Section 6.3) for each cluster in the clustered data.

The GA was run for each cluster, resulting in six different ANN topologies (Table 6.3). At the end of each GA run for each cluster, the ANN topology resulting in the highest score was retained as the classifier most suitable. Table 6.3 presents the ANN topology for each cluster classification that yielded the highest accuracy. These results were gained from training each ANN on each cluster of the clustered data set (Section 6.4.1) for 1000 iterations.

6.4.3 Experiment Set 3: Classifier Testing

The flow inspector software (Section 6.4.1) was once again deployed for the purpose of capturing discriminators in each passing TCP flow over a 15 min period. Concurrently, a separate instance of the flow inspector software was connected to a *wide area network* (WAN) of a home broadband user, recording TCP flows over a 1 h period. The flow inspector recorded for 1 h since the traffic volumes of the home user were significantly less than that of the corporate network. Each of the data recorded during this process were manually annotated by experts, identifying them by their underlying application protocol (for verification).

In order to adequately test the trained ANN classifiers, five randomly selected samples for each protocol were presented to the ANN classifiers trained in experiment set 2 (Section 6.4.2). The highest-scoring (classification accuracy) ANN classifier and the average output scores are presented in Table 6.4 for a given set of protocols. These results are presented with those attained by a heuristic annotation method (Goss and Botha 2012), an HSOM method (Goss and Nitschke 2013a), and a *k*-means method (Goss and Nitschke 2013b) for the same protocols. In Table 6.4, – indicates that the protocol was not tested, and DPC refers to the classification accuracy of the *DPC* method

Table 6.4 Classification Results on Test Data

Protocol	*Classifier Number*	*DPC*	*Heuristic*	*HSOM*	*K-means*
POP3	2	99.96%	99.06%	99.88%	–
SMTP	3	99.90%	99.92%	99.86%	99.94%
IMAP	–	–	–	99.77%	99.86%
HTTP	1	100.00%	99.93%	69.60%	99.88%
HTTPS	5	99.92%	99.95%	99.95%	99.89%
Soulseek	–	–	–	99.83%	–
BitTorrent	6	99.82%	–	99.70%	98.78%
PPTP	4	99.41%	–	–	–

If SMTP, IMAP, HTTP, and HTTPS are abbreviations, please define.

proposed in this chapter. Heuristic refers to the manual annotation method for protocol identification described by Goss and Botha (2012).

6.5 Discussion

Results indicate the degree of accuracy (close to% 100) for the evolved ANN classifiers. The topology of each of the six ANN classifiers was evolved specifically to maximize the classification accuracy of each with respect to clustered training data. Six ANN classifiers were evolved since there were six clusters (where each cluster corresponded to a detected application protocol) in the training data set. The high classification accuracy was verified for both training and test data. To properly test the capability of trained classifiers to accurately classify any given protocol, a set of application protocols were randomly selected from a given protocol set. Furthermore, fabricated *test* protocols were randomly generated from an existing discriminator set.

Trained ANN classifier accuracy is comparable to that yielded by an HSOM (Goss and Nitschke 2013a) and *k*-means (Goss and Nitschke 2013b) method for the same application protocol classification task. New flows were captured on corporate and a home ADSL network and were used as the test data for the trained ANN classifiers. This was done so the authors could ascertain how well the trained ANN classifiers are able to dynamically classify the most popular application protocols (such as HTTP and SMTP) from *live* network flows of corporate versus home networks. In order to ensure that the ANN classifiers were properly trained, that is, generalizing such that newly observed protocols were properly classified, only classifications yielding a 95% or higher degree of accuracy were considered. Classifications with a lower degree of accuracy were considered to have an unknown classification.

If "ADSL" is an abbreviation, please define at first mention.

Table 6.4 indicates that all classifiers yielded a degree of accuracy of 99% or higher when given the task of identifying eight given application protocols in live traffic flows. The DPC method yielded comparable classification results (no statistically significant difference) to the HSOM and *k*-means methods. The exception was that the DPC and *k*-means methods both significantly outperformed HSOM for accurately classifying the HTTP protocol. In the HSOM method, HTTP transactions using the HTTP 1.1 protocol (pipe-lining) spread the HTTP protocol over a number of clusters and, subsequently, a number of classifiers. Testing the discriminator sets manually

annotated HTTP by experts against each individual classifier subsequently resulted in the low score being produced. The *k*-means method was able to overcome this problem; however, the specifics of this are not mentioned in the work. In the DPC method, these HTTP 1.1 requests were, upon further investigation, discarded by DBSCAN as "noise" due to the lack of samples during the clustering process.

Another key difference between the DPC, HSOM, and *k*-means methods that affected overall classification accuracy in this comparative study was the efficacy of the clustering of training data. The capability of unsupervised learning (clustering) data to correctly cluster traffic flow data according to identified application protocols impacted the ANN backpropagation training process and, thus, ANN classification accuracy.

For this clustering process, both the *k*-means and HSOM methods required many iterations before clusters emerged in the final layer of the SOM (Goss and Nitschke 2013a). Both the HSOM and *k*-means methods worked with a data set of 1973 data points with 11 dimensions (discriminators). In the first layer of the HSOM, clustering occurred after 300 iterations. Each iteration equated to the evaluation of 1973 discriminators for each of the 400 neurons in the first layer. Clustering at the first layer of the HSOM resulted in each of 50 *best matching unit* (BMU) clusters being evaluated against 100 neurons in the second layer for an additional 100 iterations.

The final layer of the HSOM identified 14 clusters; however, only 11 data sets were considered and used in training ANN classifiers due to the lack of datum samples in three of the clusters. The *k*-means method (Goss and Nitschke 2013b) was similarly inefficient in that multiple iterations of the method were required for each datum within the traffic flow data set that was being clustered. Given randomly selected centroids, $k = 16$ clusters were derived, with a calculated silhouette value of 0.916821. For the DBSCAN method, a data set of 1421 data points with 11 dimensions (discriminators) was recorded. The DBSCAN method evaluates each data point only once for each run of the algorithm. In contrast to the HSOM and *k*-means methods, this resulted in increased efficiency when clustering data sets of recorded network traffic.

Also, the DBSCAN method did not produce clusters with an insufficient number of sample data points, as was the case for both the HSOM and *k*-means methods. In the case of both HSOM and *k*-means, clusters that contained an insufficient number of data points had to be manually removed. This meant that, dissimilar to the DPC method, the HSOM and *k*-means methods could not be fully automated. For example, in the HSOM clustering, three clusters emerged, which DBSCAN would have considered noise. This result is attributed to the capability of DBSCAN to form arbitrarily shaped clusters and its robustness toward outlier detection (noise), elements that skew both the HSOM and *k*-means methods.

Furthermore, the DPC method was advantageous in that the ε and *minPTS* parameters were evolved to values that worked well for the clustering of recorded traffic flow training data that were clustered by the DBSCAN component of DPC. The DPC method also had the advantage that the DBSCAN component was able to form complex shapes in its clustering process and thus cluster nonlinearly separable data.

6.6 Conclusion

This chapter presented the DPC method for a case study of automated application protocol identification on computer networks. This study extended previous work (Goss and Botha 2012; Goss and Nitschke 2013a,b) that showed the benefit of unsupervised clustering methods (HSOM and *k*-means) for dynamic application protocol identification on recorded traffic flows. Such

automation is beneficial since it alleviates for network administrators the delays experienced waiting for new signatures to be created by the vendor. Methods that automate the classification of new protocols must thus work with the data of live traffic flows and be able to dynamically ascertain the defining features of new protocols that traverse the network (Szabo et al. 2007).

This case study demonstrated that the DPC method was appropriate for dynamically identifying new protocols and generating signatures ad hoc. The clustering component (DBSCAN) of DPC was found to be more efficient and appropriate for the given protocol classification and identification task. *Efficient* refers to the lower number of evaluations per data point required for DBSCAN to cluster the training data. *Appropriate* refers to the capability of DBSCAN to more effectively (comparative to the HSOM and *k*-means methods) exclude noise from clustered data via forming nonspherical clusters, and to cluster nonlinearly separable data. Also, DBSCAN (more generally the DPC method) did not require the number of clusters to be specified *a priori*, as was necessitated by *k*-means.

The DPC method demonstrated an average classification accuracy of 99% given the task of dynamically identifying eight application protocols. The overall classification accuracy of the DPC method was comparable to that achieved by HSOM and *k*-means. Hence, the hypothesis that the DPC method would outperform (with statistical significance) the HSOM and *k*-means methods (Section 6.1) was refuted. However, these latter two methods are still to be tested for the PPTP, Soulseek, and POP3 protocols. While the average accuracy achieved was favorable (in excess of 99%), it is still to be determined whether such accuracy would persist with an increased sample set with additional protocols observed by the flow inspector software.

If "PPTP" and "POP3" are abbreviations, please define at first mention.

Future work will address the issue of clustering high-dimensional data, which has been elucidated as problematic for nonparametric density-based methods, such as DBSCAN (Jain 2010). Whereas this work used data sets defined by 11 discriminators (dimensions), future studies will compare the capability of *k*-means, HSOM, and the DPC method to work with significantly higher-dimensional data. Future work will also address extensions to the DBSCAN component of the DPC method, such as allowing for an ε value to be dynamically set for emerging clusters, after initially being set for the entire clustering process. Furthermore, the DBSCAN component will be compared with the OPTICS (Ankerst et al. 1999) and DeLiClu algorithms (Achtert et al. 2006), to investigate the possibility of overcoming DBSCAN's limitation in performing optimally within large density distributions.

If "OPTICS" is an abbreviation, please define at first mention.

References

Achtert, E., Böhm, C., & Kröger, P. (2006). DeLi-Clu: Boosting Robustness, Completeness, Usability, and Efficiency of Hierarchical Clustering by a Closest Pair Ranking. In *Advances in Knowledge Discovery and Data Mining* (pp. 119–128). Springer, Berlin, Heidelberg.

Alshammari, R., & Zincir-Heywood, A. (2007). A Flow Based Approach for SSH Traffic Detection. In *Proceedings of the IEEE International Conference on System, Man and Cybernetics* (pp. 296–301). IEEE Computer Society, Montreal, Que.

Alshammari, R., & Zincir-Heywood, A. (2008). Investigating Two Different Approaches for Encrypted Traffic Classification. In *Proceedings of the Sixth Annual Conference on Privacy, Security and Trust* (pp. 156–166). IEEE Computer Society.

Please provide editor(s), publisher name and location.

Alshammari, R., & Zincir-Heywood, A. (2009). Machine Learning Based Encrypted Traffic Classification: Identifying SSH and Skype. In *Computational Intelligence for Security and Defence Applications* (pp. 1–8).

Ankerst, M., Breunig, M., Kriegel, H., & Sander, J. (1999). OPTICS: Ordering Points to Identify the Clustering Structure. *ACM SIGMOD Record*, *28*(2), 49–60.

Please provide page number(s).
Auld, T., Moore, A., & Gull, S. (2007). Bayesian Neural Networks for Internet Traffic Classification. *IEEE Transactions on Neural Networks*, *18*(1).

Please provide publisher name and location if available.
Bernaille, L., Soule, A., Akodjenou, I., & Salamatian, K. (2005). Blind Application Recognition through Behavioral Classification. *CNRS LIP6, Technical Report*.

Bernaille, L., Teixeira, R., & Salamatian, K. (2006). Early Application Identification. In *Proceedings of the 2006 ACM CoNEXT Conference* (p. 6).

Cohen, W. (1995). Fast Effective Rule Induction. In *Proceedings of the Twelfth International Conference on Machine Learning* (Vol. 95, pp. 115–123).

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, *20*(3), 273–297.

Please provide publisher location.
Dempster, A., Laird, N., & Rubin, D. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 1–38. Please provide volume number.

Please provide publisher location.
Devroye, L. (1996). *A Probabilistic Theory of Pattern Recognition* (Vol. 31). Springer Verlag.

Eiben, A., & Smith, J. (2003). *Introduction to Evolutionary Computing*. Springer.

Este, A., Gargiulo, F., Gringoli, F., Salgarelli, L., & Sansone, C. (2008). Pattern Recognition Approaches for Classifying IP Flows. In *Proceedings of the 2008 Joint IAPR International Workshop on Structural, Syntactic and Statistical Pattern Recognition* (pp. 885–895). Springer-Verlag, Berlin, Heidelberg. doi: 10.1007/978-3-540-89689-0_92.

Ester, M., Kriegel, H., Sander, J., & Xu, X. (1996). A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Knowledge Discovery in Databases* (Vol. 96, pp. 226–231). Please provide editor(s), publisher name and location.

Please provide editor(s), publisher name and location.
Freund, Y., & Schapire, R. E. (1995). A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting. In *Computational Learning Theory* (pp. 23–37).

Gargiulo, F., Kuncheva, L., & Sansone, C. (2009). Network Protocol Verification by a Classifier Selection Ensemble. In *Multiple Classifier Systems* (pp. 314–323). Springer-Verlag, Berlin, Heidelberg.

Gebski, M., Penev, A., & Wong, R. (2006). Protocol Identification of Encrypted Network Traffic. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence* (pp. 957–960). IEEE Computer Society.

Goss, R., & Botha, R. (2012). Establishing Discernible Flow Characteristics for Accurate, Real-time Network Protocol Identification. In *Proceedings of the 2012 International Network Conference (INC2012)* (pp. 25–34). IEEE Computer Society, Port Elizabeth, South Africa.

Goss, R., & Nitschke, G. (2013a). Automated Network Application Classification: A Competitive Learning Approach. In *Proceedings of the IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2013)* (pp. 45–52). IEEE Press, Singapore.

Goss, R., & Nitschke, G. (2013b). Network Protocol Identification Ensemble with EA Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2013)* (pp. 1735–1736). ACM Press, Amsterdam, Netherlands.

Hu, B., & Shen, Y. (2012). Machine Learning Based Network Traffic Classification: A Survey. *Journal of Information and Computational Science*, *9*(11), 3161–3170.

Huang, K., & Zhang, D. (2008). A Byte-Filtered String Matching Algorithm for Fast Deep Packet Inspection. In *Proceedings of the Ninth International Conference for Young Computer Scientists* (pp. 2073–2078). IEEE Computer Society.

Jain, A. (2010). Data Clustering: 50 Years beyond K-means. *Pattern Recognition Letters*, *31*(8), 651–666. doi: 10.1016/j.patrec.2009.09.011.

Li, Z., Yuan, R., & Guan, X. (2007). Traffic Classification—Towards Accurate Real Time Network Applications. In *Proceedings of the Twelfth International Conference on Human-Computer Interaction: Applications and Services* (pp. 67–76). Springer-Verlag, Berlin, Heidelberg.

Please provide complete list of contributors.
MacQueen, J. et al. (1967). Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (Vol. 1, p. 14).

Please provide volume number.
Mcgregor, A., Hall, M., Lorier, P., & Brunskill, J. (2004). Flow Clustering Using Machine Learning Techniques. *Passive and Active Network Measurement*, 205–214.

Moore, A., & Papagiannaki, K. (2005). Toward the Accurate Identification of Network Applications. In *Passive and Active Network Measurement* (Vol. 3431, pp. 41–54). Springer. Please provide publisher location.

Please provide publisher location.
Moore, A., Zuev, D., Crogan, M., & Mary, Q. (2005). *Discriminators for Use in Flow-based Classification*. Queen Mary and Westfield College, Department of Computer Science.

Nascimento, Z., Sadok, D., & Fernandes, S. (2013). A Hybrid Model for Network Traffic Identification Based on Association Rules and Self-Organizing Maps (SOM). In *Proceedings of the Ninth International Conference on Networking and Services (ICNS2013)* (pp. 213–219).

Nguyen, T., & Armitage, G. (2008). A Survey of Techniques for Internet Traffic Classification using Machine Learning. *IEEE Communications Surveys & Tutorials*, *10*(4), 56–76.

Please provide volume and page number(s).

Quinlan, J. (1993). C4. 5: Programs for Machine Learning. *Morgan Kaufmann Series in Machine Learning*.

Rousseeuw, P. (1987). Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, *20*, 53–65.

Szabo, G., Szabo, I., & Orincsay, D. (2007, June). Accurate Traffic Classification. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks* (pp. 1–8).

Von Luxburg, U. (2007). A tutorial on Spectral Clustering. *Statistics and Computing*, 17(4), 395–416.