Establishing Discernible Flow Characteristics for Accurate, Real-Time Network Protocol Identification

R.G.Goss and R.A.Botha

Institute for ICT Advancement and School of ICT
Nelson Mandela Metropolitan University, Port Elizabeth, South Africa
e-mail: ryan@goss.co.za; reinhardta.botha@nmmu.ac.za

Abstract

As the number of business applications, games and other real-time applications adopting peer-to-peer communication approaches increase, so too does the requirement for accurate identification of these protocols on the network. This movement is attributed mostly to increased scalability in application deployment, reducing the overall cost of network resources. Without the identification of critical applications, networks run the risk of becoming overwhelmed by lower-priority, less important protocols, reducing resources available to higher priority applications. Deep packet inspection and statistical characteristic profiling have been used to identify various application flows, however there is an apparent disconnect between the two. Whilst both produce fairly accurate results, this paper aims to increase the accuracy of these systems by marrying the two into a single classifier using artificial neural networks.

Whilst many traffic profiling systems examine the full network flow post-termination, this paper proposes a methodology for utilizing the unique characteristics of network traffic flows which distinguish various applications at the beginning of the flow, in real-time, allowing early identification and thus effective control of a flow within the first few packet exchanges.

Keywords

Network Traffic Flows, Deep Packet Inspection, Traffic Classification, Statistical Profiling, Flow Discrimination, Artificial Neural Network, Sigmoidal Logistic Regression

1. Introduction

The growth of network communications has always been hampered by the finite resources available, restricted by a number of elements, the most prominent being bandwidth availability. These limitations are often imposed by the availability of physical wiring and/or the cost of provisioning the services through a 3rd party, such as the national telecommunications company. The available bandwidth, therefore, needs to be managed, providing optimal use of available resources to satisfy customer/user demand. Application protocols which comprise network traffic need to be identified in order to be effectively managed and prioritised. The identification and categorization of network traffic flows aids network administrators in quickly diagnosing problems, network capacity planning tasks and identifying misuse of the provisioned resources (Wright et al., 2006).

In times past, network administrators extracted Open System Interconnection (OSI) layer 4 port and protocol information from packet headers to identify and classify traffic. This method is not suitable in today's next generation networks as the ports and protocols used for a particular flow are chosen by the two communicating hosts. Hosts can therefore negotiate alternative layer 4 information, bypassing various filters and restrictions (Alshammari & Zincir-Heywood, 2008). The inefficiency of port-based classification methodologies in successfully matching these evasive protocols forced researchers to consider a number of alternatives.

Deep Packet Inspection (DPI) serves as one such alternative, aiming to solve these problems by tracking each flow's OSI layer 7 protocol information, applying predefined signatures and searching the packet's payload for string matches. This method of traffic identification can be extremely accurate, as long as the underlying payload is not encrypted or otherwise opaque (Alshammari & Zincir-Heywood, 2008). DPI is often applied to the first few packets of a connection, since the layer 7 protocol information is exchanged at the very beginning of the connection.

Whilst DPI has proven its worth as a flow classification methodology, it falls short when inspecting opaque application protocols such as encrypted traffic flows. It has been argued that much information can be inferred about a traffic flow by observing various statistical characteristics of each flow.

This paper is proposes a set of discriminators which, when used in conjunction with one another, provide enough granularity to accurately identify an application protocol in real-time, within the first few packet exchanges of newly established flows. The premise of this paper is that by the marriage of statistical and DPI discriminators, a suitable granularity of flow information can be inferred within the first 4 payload-carrying packets of host-to-host data exchange. This process is demonstrated through experimentation, discriminating a group of applications from one another.

2. Related Work

A significant amount of research has been conducted in the identification of the underlying application protocol at the early stage of a network flow's existence, through statistical analysis and DPI. For example, Bernaille et al. (2006) describes a statistical method of grouping Transport Control Protocol (TCP) flows which exhibit similar behaviour, using K-Means, Gaussian mixture model, and spectral on Hidden Markov Model techniques. According to Bernaille et al. (2006), packet size information obtained during the first few packet exchanges of a flow serve as a good metric for identifying the underlying application protocol. Gargiulo et al. (2009) asserts that significant accuracy can be achieved in identifying a flow's application protocol by examining the direction of the first four packets along with the payload sizes of each. Moore & Papagiannaki (2005) argue that in some cases, the accuracy achieved by observing the first payload-bearing packet is enough to identify the protocol, whilst in others up to 1Kbyte of payload is required before a decision is made. Various statistics including minimum, average and maximum packet lengths derived from a flow are described by Alshammari et al. (2007) as suitable discriminators for identifying a flow's underlying application protocol.

The ideas set forth by this paper are built on the observations of these works, verified through experimentation.

3. Methodology

The authors conducted a number of experiments in order to determine a base set of discriminators, which would be granular enough to distinguish application protocols from one another. Each experiment built on the discriminator set used by its predecessor, creating a more complex, granular classifier. The results of these experiments are presented at the end of this paper, providing a strong set of discriminators for use in the early identification process.

The objective of the experiments was to define a set of discriminators which, when applied to early packet data, would provide enough granularity to uniquely identify each application protocol, rather than simply group them by their operational characteristics as described by Zuev & Moore (2005).

4. Experimentation

A literature study was undertaken to determine popular discriminators for consideration in the final flow identification engine. The selected discriminators served as input vectors for use in training and testing a number of neural networks, each for a unique protocol.

Discriminators that rely on capturing the complete flow were stripped from the list of potentials as this hinders early, real-time detection. The list of literature reviewed and a summarized output of the common discriminators proposed are outlined in Table 1.

	Protocol	DST Port	SRC Port	Payload Stats	PKT Order Dir	PKT IAT	DPI
Li, Z. et al. (2007)	x	x	x	x			
Auld, T. (2007)				x			
Bernaille, L. et al. (2006)				x	x	x	
Este, A. et al. (2008)				x	x		
Gargiulo, F. et al (2009)		x		x	x		
Moore, A. W. et al (2005)		x	x	x	x	x	
Moore, A. et al (2005)						x	x
McGregor, A. et al (2004)				x		x	
Huang, K. et al (2008)							x
Alshammari, R. et al. (2007)				x		x	

Table 1: Popular Discriminators in Flow Detection

According to Table 1, certain flow attributes were more popular than others. These discriminators, including the Destination Port, Source Port, Payload Statistics, Packet

Order Direction, Packet Inter-Arrival Time and Deep Packet Inspection were considered for the experiments of this paper. Of these discriminators, those which bore reference to packet header information (DST and SRC ports) were removed, preventing the effect of client or server port manipulation. Packet Inter-Arrival Time (PKT IAT) was also removed from consideration, due to its dependence on the latency of the communications medium, traffic shaping mechanisms and congestion at any given time.

Experimental data was collected using a 60 minute snapshot of network traffic flows recorded on a Local Area Network (LAN) segment at a large corporate establishment.

At the end of the recording process, each flow was manually identified and marked by the authors for verification purposes. In accordance with Bernaille, L. et al (2006) and Gargiulo, F. (2009), only Transport Control Protocol (TCP) flows were recorded as part of the snapshot flow set. Furthermore, flows where the initial synchronize (SYN) packet was not observed were excluded as the discriminators rely on the first few packet exchanges of each flow. Table 2 provides a break down of the remaining, manually identified flows.

Protocol	Recorded Flows
HTTP	4452
HTTPS	3147
SSH	392
FTP	102
POP3	62
SMTP	17

Table 2: Observed Network Traffic Flows

Based on the direction of the initial SYN packet, both the client and the server host can be identified and thus the direction of packet flow. Packets with no payload were assumed to be normal TCP control packets, which provide the reliable transport platform for the encapsulated application protocol operations.

Artificial Neural Networks (ANN) were used as classifiers in the experiments, therefore the inputs needed to meet basic ANN requirements, falling within the bounds of 0.0 and 1.0. The authors accomplished this by the application of a sigmoidal logistic regression function, producing a probability value between 0 and 1.

The application flows captured were used to train a variety of Feed-forward, Back Propagation Artificial Neural Networks. A neural network was created and trained for each of the six application protocols identified in table 2. A subset of flow discriminators were chosen for each of the experiments, incorporating the discriminators used by their predecessor. For each experiment, the training set values were passed through each neural network and the accuracy of the results noted.

For consistency, each neural network consisted of a number of inputs which matched the number of discriminators for that experiment's particular feature subset. Directly connected to each of the input neurons was a second, hidden layer whose neuron count matched that of the input layer. Finally, a single neuron, connected to each of the neurons in the hidden layer, represented the output layer. The output neuron was responsible for producing a decimal value between 0 and 1, indicating the certainty that the inputs provided matched the protocol the neural network was trained to identify.

The extracted discriminators from the flow snapshot were fed to each neural network in turn over a period of 1000 iterations. The flows which the network should match were marked with a "1" as an expected output, whilst those the network should not match a "0". For example, the neural network created to match the Hypertext Transfer Protocol (HTTP) would have all HTTP flow inputs marked with a 1, whilst the remaining 5 input sets marked with a 0.

After a successful training session, the inputs and their respective outputs were once more passed through each neural network in order to determine accuracy. The actual output for each discriminator input set was compared to the expected output and the variance noted. A separate score was tallied for matching both the expected 0 (correctly identifying a protocol other than the one the network was trained to identify) and expected 1 (correctly identifying a protocol the network was trained to identify). A high degree of accuracy is required in order to demonstrate a clear discrimination between the various protocols across each of the 6 trained neural networks.

5. Experiment 1: Directional Discrimination

The first discriminator set tested included the directions for each of the first 4 payload-bearing packet exchanges. Client to server packets were marked with a 0, whilst server to client packets with a 1. The observed patterns for each protocol and the number of times each were detected are shown in Table 3.

	FTP	**HTTP**	**HTTPS**	**SSH**	**POP3**	**SMTP**
0100	4					
0101	98				62	17
0110				392		
1000		2105	1428			
1001		254	287			
1010		1856	1166			
1011		21	242			
1100		216	12			

Table 3: Payload-carrying Packet Directionality

Table 4 denotes the results observed, post training process, when passing each network's training set through it and comparing the actual result to the expected result. The probability for each expected case was determined and the average percentile achieved noted. A well trained classifier is expected to reach an average probability score in excess of 90%, indicating its level of understanding of the data within the training set.

	FTP	HTTP	HTTPS	SSH	POP3	SMTP
EXPECT 0	99.42%	99.34%	99.14%	99.97%	99.59%	99.86%
EXPECT 1	50.13%	0.93%	1.37%	98.73%	35.97%	7.29%

Table 4: Results of Experiment 1

Table 5 details the results observed when a single flow of each application protocol was passed through each of the neural networks, gauging their effectiveness in recognizing them.

Protocol	Ordered Classifier Probability Results
FTP	FTP (48.28%) POP3 (35.97%) SMTP (7.29%) SSH (1.23%) HTTPS (0.43%) HTTP (0.01%)
HTTP	HTTP (1.33%) HTTPS (0.70%) SMTP (0.01) POP3 (0.00%) FTP (0.00%) SSH (0.00%)
HTTPS	HTTP (1.33%) HTTPS (0.70%) SMTP (0.01%) POP3 (0.00%) FTP (0.00%) SSH (0.00%)
SSH	SSH (99.74%) HTTPS (0.20%) SMTP (0.12%) FTP (0.11%) POP3 (0.08%) HTTP (0.01%)
POP3	FTP (48.28%) POP3 (35.97%) SMTP (7.29%) SSH (1.23%) HTTPS (0.43%) HTTP (0.01%)
SMTP	FTP (48.28%) POP3 (35.97%) SMTP (7.29%) SSH (1.23%) HTTPS (0.43%) HTTP (0.01%)

Table 5: Experiment 1 - Classifier Accuracy in Identification

6. Discussion

Experiment 1 illustrates that using even a limited number of discriminators; a fair degree of certainty can be inferred upon the flow's underlying application protocol. FTP, SSH, POP3 and SMTP protocols exhibited similar characteristics in that the first packet originated from the server to the client, whilst the HTTP and HTTPS flows first payload-carrying packet was from the client to the server. Table 3 shows the similarities between HTTP and HTTPS in the direction of packet flows – something expected as HTTPS is essentially an encrypted HTTP stream. These similarities in directionality explain why the training process resulted in the HTTPS and HTTP neural networks providing low degrees of accuracy. POP3, SMTP and FTP also exhibit directionality similarities and thus suffer from inaccurate distinction at this level. Table 5 shows that the FTP classifier incorrectly scored higher than the SMTP and POP3 protocols for their own protocol test. Furthermore, in the case of SMTP, both the FTP and POP3 classifiers exhibited a higher probability than the SMTP protocol itself. SSH is the only protocol that does not share directionality discriminators, hence the high degree of accuracy in matching both SSH and "other" protocols (99.97% and 98.73% respectively) with so few discriminators. The grouping of multiple protocols into distinct classes using directional discriminators was also noted by Bernaille, L. et al (2006), Este et al. (2008) and Gargiulo, F. et al (2009).

7. Experiment 2: Direction and Packet size Statistics

The second experiment built on the first by adding the average, minimum and maximum payload sizes for the first 4 payload-carrying packets to the discriminator set. The inputs thus grew from 4 in the first experiment, to 7 in the second.

After the training process had completed, the following results were observed.

	FTP	HTTP	HTTPS	SSH	POP3	SMTP
EXPECT 0	99.52%	99.85%	99.87%	99.98%	99.89%	99.86%
EXPECT 1	60.61%	33.50%	18.21%	98.99%	10.02%	7.86%

Table 6: Results of Experiment 2

Table 7 details the results observed when a single flow of each application protocol was passed through each of the neural networks, gauging their effectiveness in recognizing them.

Protocol	Ordered Classifier Probability Results
FTP	FTP (46.12%) POP3 (10.71%) SMTP (7.64%) HTTP (0.55%) SSH (0.38%) HTTPS (0.10%)
HTTP	HTTP (42.47%) HTTPS (0.03%) SMTP (0.00%) POP3 (0.00%) FTP (0.00%) SSH (0.00%)
HTTPS	HTTPS (39.43%) SSH (0.00%) HTTP (0.00%) SMTP (0.00%) POP3 (0.00%) FTP (0.00%)
SSH	SSH (99.91%) SMTP (0.09%) HTTPS (0.06%) POP3 (0.01%) FTP (0.00%) HTTP (0.00%)
POP3	FTP (46.24%) POP3 (9.94%) SMTP (7.72%) SSH (0.43%) HTTP (0.30%) HTTPS (0.12%)
SMTP	FTP (46.09%) POP3 (8.85%) SMTP (7.86%) SSH (0.48%) HTTP (0.26%) HTTPS (0.12%)

Table 7: Experiment 2 - Classifier Accuracy in Identification

8. Discussion

Experiment 2 increased the granularity of the flow information by including additional discriminators. Flows which exhibit small payload exchanges, including real-time applications such as SSH, will over the course of the flow exhibit a much lower average payload size than that of a bulk transfer application such as HTTP. Experiment 2 tested if this was plausible within the first 4 packet exchanges. The results of experiment 2 clearly indicate an increase in accuracy for all but one protocol, POP3. Although HTTP and HTTPS bare resemblance in packet directionality patterns, packet size variances between the two protocols as early as the first 4 packets appeared evident. This increase in identification ability is indicative of strengthening pattern identification within the protocol as the packet size statistic discriminators are added to the discriminator set. Table 7 indicates that using packet size statistics, HTTP is now more discernable from HTTPS and thus the probability score for HTTPS correctly placed the HTTPS classifier as the identifier for the HTTPS protocol. No change was noted for the probability positioning for the SMTP and POP3 classifiers.

9. Experiment 3: Direction, Packet Size and Payload

The final experiment conducted was an attempt to combine the statistical classification process with that of DPI. DPI is most commonly implemented using regular expressions, attempting to match patterns within strings. The motivation for this 3rd experiment was the assumption that application protocols require a certain amount of initialization before any data exchange could take place and that this initialization will take place within the first few payload-carrying packets of a flow. For this reason, the first 3 bytes of payload from the first 2 payload-carrying packets were extracted and converted to their ASCII integer values. These 6 new values were then included in the discriminator set from experiment 2, bringing the total neural network inputs to 13.A set of 6 new neural networks were created and the new training sets applied.

The subsequent results were reported as follows:

	FTP	HTTP	HTTPS	SSH	POP3	SMTP
EXPECT 0	99.99%	99.94%	99.94%	99.99%	100.00%	99.99%
EXPECT 1	98.17%	99.88%	99.80%	99.88%	98.86%	96.79%

Table 8: Results of Experiment 3

The results observed when passing a single instance of each of the 6 application flows to each neural network are recorded in table 9.

Protocol	Ordered Classifier Probability Results
FTP	FTP (98.07%) SMTP (0.087%) SSH (0.50%) POP3 (0.10%) HTTP (0.00%) HTTPS (0.00%)
HTTP	HTTP (99.93%) HTTPS (0.06%) SMTP (0.00%) POP3 (0.00%) FTP (0.00%) SSH (0.00%)
HTTPS	HTTPS (99.95%) HTTP (0.14%) SSH (0.00%) SMTP (0.00%) POP3 (0.00%) FTP (0.00%)
SSH	SSH (99.92%) FTP (0.02%) HTTP (0.00%) SMTP (0.00%) HTTPS (0.00%) POP3 (0.00%)
POP3	POP3 (99.06%) SSH (0.84%) SMTP (0.41%) HTTP (0.07%) HTTPS (0.00%) FTP (0.00%)
SMTP	SMTP (96.92%) FTP (2.36%) POP3 (1.00%) SSH (0.53%) HTTP (0.00%) HTTPS (0.00%)

Table 9: Experiment 3 – Classifier Accuracy in Identification

10. Discussion

Experiment 3 provided the most promising results as DPI discriminators were added to the directional and payload size discriminator set. The results recorded indicate the best understanding by each network across the board for all protocols. The increased granularity and uniqueness in application protocol initialization exchanges in the first few bytes of communication between hosts is to thank for this. Although HTTP and HTTPS share similarities in directional and payload size discriminators, the underlying payload of HTTP is plain-text, whilst HTTPS is encrypted. HTTPS will therefore use SSL control bytes during the first few packets of the data exchange, whilst HTTP will begin with HTTP request arguments. Further, whilst POP3 and SMTP exhibited similarities in their directional and payload size statistics, the underlying payload was very different. SMTP servers respond to newly established sockets by sending a “220”, followed by a server message. POP3 servers will respond to new clients with “+OK” followed by a short string. This difference in early application protocol exchange clearly distinguishes the one protocol from the other. Table 9 indicates exact matching for each of the selective flow tests, with the closest second matching classifier reporting a variance of 94.56% to the nearest rival in the case of SMTP. The authors believe this may be accredited to the lack of flows captured for the SMTP protocol when compared with the others.

11. Conclusion

The observed results conclude that the marriage of statistical and DPI discriminators form the best possible chance for accurately discerning one protocol from another at the early stages of a network traffic flow. Furthermore, increasing the number of discriminators to the neural networkincreased the granularity of the application protocol, creating a better possibility for pattern identification between protocols and thus higher degrees of accuracy in identification.

Increasing the granularity of flow information over the course of 3 experiments produced an increasing accuracy in identifying each protocol. Furthermore, similar protocols exhibited a decline in their probability scores, indicating increased understanding of the protocol each were trained to identify.

This paper therefore proves that it is possible to identify and classify application protocol flows in real-time, at the beginning of the flow, using discernible flow characteristics such as packet directionality, packet size statistics and DPI. This paper further concludes that it is possible to apply this technique to encrypted and plain-text flows alike and that due to the removal of dependence on packet header information, it is possible to detect these protocols on any port or protocol.

12. References

C.V. Wright, F. Monrose& G. M. Masson (2006), “On Inferring Application Protocol Behaviors in Encrypted Network Traffic”, *Journal of Machine Learning Research 7,* 2006, pp. 2745-2769

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

Williams, N., Zander, S. &Armitage, G., (2006), “A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification”, *ACM SIGCOMM Computer Communication Review 7,* Volume 36, Number 5, October 2006

Li, Z., Yuan, R. & Guan, X. (2007), “Traffic Classification – Towards Accurate Real Time Network Applications”, *In HCI (4),* pp. 67-76, 2007

Auld, T., Moore, A.W. & Gull, S.F. (2007), “Bayesian Neural Networks for Internet Traffic Classification”, *IEEE Trans. On Neural Networks*, 18 (1) pp. 223-239, January 2007.

Bernaille, L., Teixeira, R. &Salamatian, K. (2006), “Early Application Identification”, *Proceedings of the 2006 ACM CoNEXT conference*, 2006

Este, A., Gargiulo, F., Gringoli, F., Salgarelli, L. &Sansone, C. (2008), “Pattern Recognition Approaches for Classifying IP Flows”, *Structural, Syntactic and Statistical Pattern Recognition – Lecture Notes in Computer Science*, Volume 5342, pp. 885 – 895, 2008

Gargiulo, F., Kuncheva, L.I., &Sansone, C. “Network Protocol Verification by a Classifier Selection Ensemble”.*In Proceedings of MCS.*Pp314-323, 2009.

Moore, A. W. & Zuev, D., “Discriminators for use in flow-based classification”, *Technical report, Intel Research*, Cambridge, 2005.

Moore, A. & Papagiannaki, K. (2005), “Toward the Accurate Identification of Network Applications”, *Lecture Notes in Computer Science,* pp 41-54, vol 3431, 2005.

McGregor, A., Hall, M., Brunskill J., &Lorier P., "Flow Clustering Using Machine Learning Techniques", *Passive and Active Measurement Conference (PAM)*, Apr 2004.

Huang, K. & Zhang, D., "A Byte-Filtered String Matching Algorithm for Fast Deep Packet Inspection," *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for* , pp.2073-2078, 18-21 Nov. 2008

Alshammari, R. &Zincir-Heywood, A.N., "A flow based approach for SSH traffic detection," *Systems, Man and Cybernetics, 2007.ISIC. IEEE International Conference on* , pp.296-301, 7-10 Oct. 2007

Goss, R & Botha, R, “Traffic Flow Management in Next Generation Service Provider Networks - Are We There Yet?”,*Information Security South Africa Conference,2011.*

Wright, C. V., Monrose, F. & Masson, G. M., “On Inferring Application Protocol Behaviors in Encrypted Network Traffic”, *Journal of Machine Learning Research 7,* pp. 2745-2769, 2006

Zuev, D. & Moore, A. W., “Traffic Classification using a statistical approach”, *Passive and Active Network Measurement, Lecture Notes in Computer Science*, 2005, Volume 3431/2005, pp 321-324, 2005