Network Traffic Characterization Using (p, n)-grams Packet Representation

By

Abdulrahman Hijazi

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

Carleton University
Ottawa, Ontario, Canada

Abstract

With the ever increasing advances in network protocols and traffic complexity, new challenges are emerging in traffic characterization and management. In this thesis, we propose a new approach that can complement existing ones with a simple high-level understanding of network traffic. Our approach uses (p, n)-grams representation to analyze network traffic, where a (p, n)-gram is an n-byte string starting at offset p.

We argue that the (p, n)-grams representation combines the efficiency of using specific packet fields (e.g. ports) with the generalized pattern matching of n-grams, without the complexity and overhead of full packet pattern matching. We also show that using (p, n)-grams allows for traffic analysis at all packet parts (payload content, header port/flow, and other header behavior fields), without mixing between similar patterns that may accidentally exist at different fields within packets.

As a proof of concept, we develop a (p, n)-gram-based lightweight unsupervised clustering algorithm (ADHIC) that makes no prior assumptions about the involved protocols. We show that ADHIC can automatically cluster network traffic using a binary decision tree into equivalence classes that closely approximate standard measures of network traffic. We also show that ADHIC can be used to monitor network traffic through observing the dynamic updates to the clustering tree. Those incremental updates highlight the temporal changes in network traffic that are not easily detected using standard network analysis methods.

We then research the characteristics and distributions of (p, n)-grams in network packets, and how they can be utilized for traffic analysis. In particular, we argue that (p, n)-grams have automatic fingerprinting capability where a simple frequency

analysis of network packets can capture structural (p, n)-grams based on their relative high frequencies. These (p, n)-grams represent protocol and sub-protocol structures and cross-protocol patterns.

We observe that (p, n)-grams follow a power-law-like distribution where the structural ones constitute the rapidly-dropping-off curve before the long tail. We argue that this special distribution adds to the efficiency of (p, n)-grams-based traffic analysis as it describes structural (p, n)-grams as 1) a small set of (p, n)-grams that 2) can be easily distinguished from the long list. Our observation relies on a thorough empirical analysis using independent network traffic traces. In addition, we create an entropy-based conceptual model that explains this distribution behavior in the context of the hierarchy of network protocols and statistics of Internet traffic.

Acknowledgment

Dedicated to my dearest father, Abdullah Hijazi, dearest mother, Ameerah Deyab, and dearest wife, Nesrin Sarmini for their endless support and courage throughout this difficult journey. My father has been always the source of passion and enthusiasm to pursue this long path. My mother gave me the ideal example of dedication and determination. My wife made everything possible to provide the best working environment while raising our five little children.

I would like to extend my sincerest thanks and appreciation to my advisor, Anil Somayaji for all the great guidance, advice, and help he gave me throughout my time in the PhD program. Anil was an advisor and a friend at the same time who helped me facilitate all the difficulties and obstacles I went through during my program.

I would also like to specially thank Hajime Inoue who I worked with on the ADHIC algorithm and was the one who coded the entire implementation of ADHIC (i.e., NetADHICT). It was also a great honor and privilege to work with Paul van Oorschot and Ashraf Matrawy on the first publications we had on ADHIC.

Special thanks to Scott Knight and Bilal Hejazi who helped us experimenting ADHIC with real datasets from their organizations (RMC and MD respectively), and then getting the results after proper anonymization to the data.

Thank you to Gunes Kayacik, Mohammad Mannan, and Gehana Booth for reading through the entire thesis and providing insightful feedback.

Another thank you to my PhD proposal committee, Scott Knight, Evangelos Kranakis, Ashraf Matrawy, and Liam Peyton, and my PhD examination board, Carey Williamson, Marc St-Hilaire, Robert Biddle, and Carlisle Adams for all their useful

comments and great insights.

Finally, I would like to remember OGS, OGS-ST, PSEPC, NSERC, and MITACS for their generous awards and fundings.

Contents

List of Tables

List of Figures

1 Introduction

Traffic of computer networks is fundamentally hard to understand. Enterprises add numerous database, file service, network management, and proprietary protocols as part of custom applications. Even the smallest home networks today connect to thousands of remote hosts in order to access email, instant messaging, voice-over-IP, peer-to-peer file sharing, streaming media, and social media.

To maintain their networks, administrators must make sense of this cacophony even as remote hosts shift, undocumented protocols evolve, and usage patterns constantly change. Today network administrators use a variety of tools to help them understand and manage this chaos. Current network analysis technology, however, is not sufficient for this task.

Standard network monitoring tools commonly capture traffic patterns using two approaches. One is using specific header fields such as the 5-tuples in the packet headers (i.e., source IP, source port, destination IP, destination port, and protocol ID) which are used to identify flows and sessions. The key advantage of using these fields specifically is that they can be observed efficiently while also providing good insights into traffic patterns.

The other is regular expression-encoded signatures that are used to identify patterns associated with specific protocols, attacks, or other chosen communications. The key advantage to signatures is that they allow for very specific types of traffic to be extracted. They are much more expensive to process than 5-tuples, however, because the entire packet contents—not just the headers—must be searched for each signature.

While both are powerful on their own, these representations limit our ability to learn about network traffic. If we are looking for specific signatures, then we will only find patterns matching those signatures. If we are looking at 5-tuples, then our findings are limited to patterns that reveal themselves within 5-tuples. The key question of this thesis is: are there other ways of representing network traffic that combine the specificity of signatures with the efficiency of 5-tuples? To begin to address this question, we start with packet similarities.

1.1 Packet Similarities

In a typical network traffic, packets of the same protocol type possess field similarities and byte repetitions. These can be found at any portion of the network packets including both the header and payload fields. For certain protocol types, however, packet similarities are mostly found in packet headers (e.g., encrypted protocols).

Examples of patterns in the header fields include protocol ID (TCP, UDP, ICMP), port number, Quality-of-Service (QoS) flags, and special values at common header fields, such as: time to live (TTL), checksums, and options. Examples of payload patterns, on the other hand, include strings like "GET" and "POST" in the payloads of the HTTP GET and POST request packets respectively. Other examples are the URI field in the CUPS protocol and the certificate information exchanged by the SSL protocol's communicating parties. Figure 1.1 presents the general structure of the HTTP GET-request packet, including the "GET" subsequence pattern.

Note that some patterns, such as the letters "GET", will have a higher frequency than others (such as the text of the subsequent URL). Moreover these patterns will tend to occur more often in certain parts of the packet (say, the beginning of the TCP payload) than others. We refer to such high-frequency patterns as being "*structural*" or "*semantic*" patterns within network packets as they tend to be associated with protocol or program-level characteristics of communication. In contrast, lower frequency patterns are associated with more varied phenomena such as user communication.

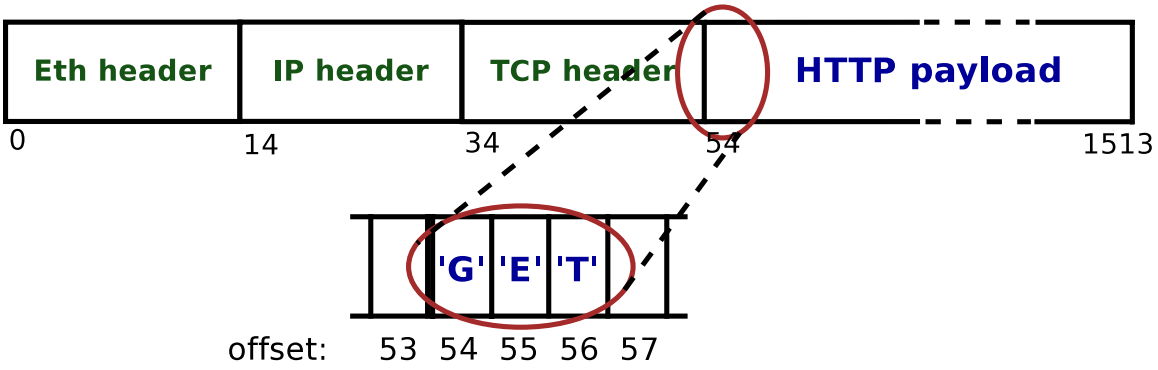


Figure 1.1. Simplified structure of an HTTP GET-request packet.

Here we ask the question, how can we identify and characterize structural patterns in a general way, such that packets matching those patterns can be efficiently identified? As we will show, the key lies in a new representation of network traffic patterns, (p, n)-grams.

1.2 (p, n)-grams

A well-known alternative to regular expression-based signatures for representing patterns in network packets is n-grams, where an n-gram is a string of n consecutive bytes within any part of the raw packet. Viewing network traffic using this representation is not limited by protocols or flows, and does not require prior knowledge of existing patterns. Comparing usages of n-grams and 5-tuples in representing network traffic, however, shows two different emphases. While using n-grams emphasizes pattern contents, using 5-tuples emphasizes pattern locations.

n-grams are commonly used in machine learning statistical methods to analyze network packets and file types (Section 2.2.1). Examples include analyzing binary contents of files [100], and detecting anomalous file segments [172]. On the other hand, PAYL [185], and Anagram [186] use n-grams to analyze network packets with machine learning algorithms for intrusion and anomaly detection respectively.

One key disadvantage of n-grams, though, is that like signatures, they require the

entire contents of a packet to be scanned in order to determine a match. Another is that they cannot capture certain kinds of structure in network packets. For example, a 4-gram representing an IP address could match either the source or destination address in the IP packet header.

We can address both of these limitations by adding location information to n-grams. A simple way to do this is to add an offset p, thus giving us (p, n)-grams. Just like n-grams, (p, n)-grams are not limited to patterns in packet headers and do not require prior knowledge of existing patterns. (p, n)-grams, however, can be matched more efficiently and can capture structural patterns (such as the difference between source and destination IP addresses) that n-grams cannot capture. On the other hand, the 5-tuples can be represented by a set of five (p, n)-grams.

Note that Matrawy et al. [113] were the first to propose (p, n)-grams for traffic shaping of network packets. Their work, however, was mainly focused on mitigating DOS attacks through diversity-based network management using (p, n)-grams.

We define structural (p, n)-grams as those with a relatively high frequency in observed network traffic. In this thesis we argue that structural (p, n)-grams capture the high-level organization and semantics of traffic and that (p, n)-grams can be used to do lightweight hierarchical clustering of traffic and to fingerprint network protocols and sub-protocols.

1.3 Hypotheses

Here are our hypotheses about (p, n)-grams that we address in the following chapters:

First, we hypothesize that we can find representative structural (p, n)-grams efficiently using hierarchical clustering, specifically through an algorithm we co-invented, Approximate Divisive HIerarchical Clustering (ADHIC). (Chapters 3, 4, and 5)

Second, we hypothesize that (p, n)-grams with relatively high frequency constitute a *small subset* of the total possible set of (p, n)-grams in network traffic. We also conjecture that (p, n)-gram frequencies in network traffic follow a power-law-like behavior

similar to that of Zipf's law [203]. (p, n)-grams with relatively high frequency represent the short rapidly-dropping-off portion of the distribution line without the long tail. Our hypothesis is that this characteristic gives (p, n)-gram-based approaches an efficiency advantage in terms of the required space and time complexities to capture and process structural patterns (Chapters 6 and 7).

Third, we hypothesize that these relatively frequent (p, n)-grams capture high-level structures of network traffic including network protocols, high-volume communication flows, and frequently communicating hosts; thus, we call them structural (p, n)-grams. Our hypothesis is that structural (p, n)-grams can form a "fingerprint" of network protocols that may be used to identify them in a fashion similar to that of hand-crafted regular expression signatures (Chapter 6 and 8).

1.4 Related Publications

Some parts of this thesis have been peer-reviewed and published. The following list shows these publications in chronological order:

1. Hajime Inoue, Dana Jansens, *Abdulrahman Hijazi*, Anil Somayaji, "NetADHICT: A Tool for Understanding Network Traffic," USENIX: 21st Large Installation System Administration Conference (LISA'07), Dallas, USA, Nov 2007.

2. *Abdulrahman Hijazi*, Hajime Inoue, Anil Somayaji, "Lightweight Unsupervised Hierarchical Network Traffic Clustering," NIPS: Workshop on Machine Learning in Adversarial Environments for Computer Security (NIPS'07 workshop), Whistler, Canada, Dec 2007.

3. *Abdulrahman Hijazi*, Hajime Inoue, Ashraf Matrawy, P. C. van Oorschot, Anil Somayaji, "Discovering Packet Structure through Lightweight Hierarchical Clustering," Proceedings of the IEEE International Conference on Communications (ICC'08), Beijing, China, May 2008.

4. Carson Brown, Alex Cowperthwaite, *Abdulrahman Hijazi*, Anil Somayaji, "Analysis of the 1999 DARPA/Lincoln Laboratory IDS Evaluation Data with NetADHICT," Proceedings of the IEEE Second Symposium on Computational Intelligence for Security and Defense Applications (CISDA'09), Ottawa, Canada, Jul 2009.

While the parts of this thesis related to ADHIC have been published (Chapters 3, 4, and 5), the other parts have not as of yet. Those include the parts on empirical power-law characterization of network traffic (Chapter 7), protocol fingerprinting (Chapter 8), and the conceptual model of traffic (Chapter 9).

1.5 Independent and Collaborative Work

Basically this thesis is my independent work except for some collaborative work presented in Chapters 3 and 4. In these two chapters, my collaborative work (mainly with Hajime Inoue) was in defining the clustering application requirements, and working on the design of ADHIC: Approximate Divisive HIerarchical Clustering [65, 80].

On the other hand, planning, conducting all experiments and working on the empirical analysis [63, 64] and conceptual insights, were all my own independent work. Note, however, that Hajime Inoue was the one who did the software implementation of ADHIC: NetADHICT [80].

Finally, Carson Brown and Alex Cowperthwaite have helped in preparing the DARPA dataset for experimentation with ADHIC; however, all the analysis and conclusion work presented in Chapter 5 were my own [20].

1.6 Thesis Organization

This section presents a high-level overview of the arguments and contributions for the thesis chapters, and provides brief discussion of their main components.

Chapter 2 discusses the related work in network traffic characterization. The chapter focuses on the statistical analysis and machine learning algorithms proposed to fingerprint, cluster, and classify network traffic including identifying peer-to-peer (P2P) and encrypted traffic.

Chapter 3 presents the design of ADHIC (Approximate Divisive HIerarchical Clustering). ADHIC is an unsupervised packet clustering algorithm that works without *a priori* knowledge about protocol structures. The output of ADHIC is a dynamic tree graph that gives a decomposition of the inspected traffic and its changes over time.

Chapter 4 provides empirical analysis on ADHIC's clustering performance of normal traffic using data from three independent networks.

Chapter 5 examines the ability of ADHIC to monitor abnormal and evasive network traffic and that of synthetically generated network traffic. It also examines how well ADHIC can segregate protocols without looking at the packet header fields.

Chapter 6 analyzes the characteristics of (p, n)-grams that enable algorithms such as ADHIC. It introduces in more detail hypotheses about network traffic that are evaluated in later chapters. It also introduces a novel use of Shannon entropy as a metric to measure the similarity level of fields in network packets and applies this measure to multiple network captures.

Chapter 7 empirically tests the frequency distributions of (p, n)-grams in network traffic, and how their frequency analysis can be used to discover structural patterns in network packets.

Chapter 8 discusses the pattern-matching capabilities of (p, n)-grams, with emphasis on their representation richness to capture structural patterns at any portion within network packets. The chapter also discusses how these patterns can be used as fingerprints to distinguish between different protocols and sub-protocols.

Chapter 9 builds an abstract conceptual model to explain our empirical findings of the (p, n)-gram frequency distributions in the context of the current design and implementation of Internet protocols and statistics of network traffic. It serves as a formal approximation to validate that our results are not dataset dependent.

Finally, *Chapter 10* concludes with a summary of our contributions and their limitations, and discusses our plan for future work.

2 Background and Related Work

With the increasing advances in designs and types of network traffic protocols, a substantial number of researchers have explored the question of how to characterize Internet traffic. Surveying the literature for existing techniques in this domain shows various approaches with different criteria.

For example, the characterization goal might be focused on traffic fingerprinting, identification, classification, clustering, reverse engineering, or others. On the other hand, the scope could cover all Internet traffic, or specific protocols, and the analysis technique might vary from direct field check to machine learning or other statistical or heuristic analysis techniques. In addition, the analysis environment might be real-time or off-line, and the scale can vary from traffic on one host to an enterprise or ISP-level. Finally, different approaches analyze different parts of the packets. For example, there are techniques that check the packet header fields only, and others that analyze the payload portion as well.

In taking what part of network packets is analyzed as a use case, the literature shows four common ways to look at packets [36]: 1) check their port numbers, 2) analyze their flows (e.g., 5-tuples), 3) analyze their behaviors (e.g., inter-arrival time, packet size, and other packet header fields), and 4) analyze packet payloads (i.e., deep-packet inspection). It is also common to use a combination of these four as a hybrid approach in the same traffic characterization technique.

Our research is mainly about using (p, n)-grams within network packets (header and payload) to cluster network traffic. Therefore, this chapter discusses the related work with some emphasis on machine learning clustering algorithms. The chapter also

briefly discusses specific related topics such as analyzing encrypted traffic and cites examples of recent work done in protocol fingerprinting, inference and identification. The chapter also includes a section on using the n-gram and (p, n)-gram representations to analyze network traffic. Finally, the chapter concludes with a section that puts our work in the context of the related work.

2.1 Analyzing Traffic Using Ports and Flows

Checking packet port numbers is a well known analysis approach to classify network traffic [107, 137]. Specifically, port numbers have been commonly used to identify network traffic types and applications. This technique worked well with legacy protocols that used to comply with the fixed traditional port numbers assigned by IANA [72]. For example, the CoralReef [32] tool, developed by CAIDA (Cooperative Association for Internet Data Analysis) [23], uses a module (AppPorts) to convert application port numbers to application and protocol names. Another example is MRTG [120], which is a standard network system administration tool that shows traffic utilization in terms of port usage, with ports being used as a proxy for identifying network protocols and uses.

However, the advances in network protocol designs and implementations have rendered this approach inaccurate. The problem arises in more than one way. On the one hand, there is a wide usage of common ports for arbitrary applications in order to circumvent firewalls, as well as the use of dynamic port allocation for evasive applications such as peer-to-peer (P2P) file sharing [89]. On the other hand, there are new applications that don't have IANA registered ports. This is, of course, in addition to the use of port translation [36].

Another way to analyze network traffic is to check flow properties of the network packets. This is mainly to statistically analyze the flow fields in the packet header (5 tuples: source IP, source port, destination IP, destination port, and IP protocol) to characterize network traffic. FlowScan [137] is one such software package that

extracts flow information from IP routers.

In addition to techniques that examine individual flows, aggregates of flows have been explored and analyzed. Estan et al. [52] introduced a traffic characterization technique that relies on clustering traffic according to resource usage and consumption patterns, based on their 5-tuple flow information. They use their prototype system (AutoFocus [52]) to accomplish this task and generate traffic reports. Mahajan et al. [108, 109] studied aggregates of flows in the context of preventing flash crowds and denial-of-service (DoS) attacks. They proposed using a pushback mechanism where routers cooperate in controlling aggregate traffic.

Another example is NetFlow [28] which is a proprietary network protocol developed by Cisco to collect IP traffic information on systems that run Cisco Internetwork Operating System (IOS). NetFlow provides several IP services including network performance and security monitoring. Basically, it captures flow information[1] of network traffic and analyzes it at monitoring hosts.

Looking at the packet flow information is very fast. However, unless some sort of clustering of flow records is efficiently used, this flow-based approach has the drawback of emphasizing individual flows, which usually results in excessive detail and less high-level understanding of the traffic [52].

2.2 Analyzing Traffic Using Payloads (Deep-Packet Inspection)

Analyzing packet payloads is a common traffic characterization technique that is usually referred to as deep packet inspection. This technique relies mainly on looking for special signatures in the packet payloads, or doing more complicated syntactical matching.

Worm detection is one of the early applications where payload-based traffic analy-

[1]Note that Cisco usually defines a network flow by a 7-tuple entity where the SNMP ingress interface and IP type of service are added to the traditional components of the 5-tuple flow (i.e., source and destination IP address, source and destination port number, and IP protocol).

sis has been commonly used. For example, EarlyBird [162, 163] automatically detects unknown worms based on a common content sequence of exploits that comes along with a range of unique sources spreading the infection and destinations being targeted in the attack.

HoneyComb [95] uses a honeypot system to capture network traffic and apply pattern matching techniques and protocol conformance checks to automatically generate attack signatures for intrusion detection systems (IDS). Both EarlyBird and Autograph systems employ Rabin fingerprints to index counters of content substrings. Alternately, Autograph [93] generates its signatures specifically for worms that propagate using TCP transport by analyzing the prevalence of portions of flow payloads.

There are also very useful payload-based tools that rely on *a priori* knowledge of network protocols and their structures to analyze network traffic. For example, Wireshark [31] applies deep packet inspection to identify network protocols. Its current database can recognize hundreds of protocols, while new protocols get added over time. Wireshark displays captured packets graphically to the user, and gives in-depth details about their contents and structures.

In addition, Sandvine's Network Data Analytics [153] is meant for fast analysis of huge aggregate data from Tier 1 or Tier 2 networks. It offers a comprehensive coverage of many of the available protocols and combines accuracy and efficiency. However, the tool is very knowledge intensive, which limits its functionality to the already known and analyzed network traffic protocols.

In spite of its powerful characterization capabilities, one common problem with payload analysis is its limitations when inspecting encrypted traffic [60, 90, 91]. In addition, deep packet inspection usually introduces privacy concerns due to the automated analysis of transmitted user data. Another problem with payload-based characterization of network traffic is that it is usually computationally expensive, and requires continuous maintenance of identifying rules [36].

2.2.1 Using n-grams Representation To Analyze Network Traffic

One of the common network traffic analysis approaches to discover high-level patterns is to use n-grams in machine learning statistical methods to analyze network packets and file types.

For example, Li et al. [100] proposed an n-gram-based machine learning method to identify file types through analyzing their binary contents. Normalized n-gram distributions are used to represent all files of a certain type. They are then used to detect unknown file types or check if the content of a file matches the type indicated in its header. Stolfo et al. [172] then extended this work for detection of suspicious anomalous file segments using n-grams.

Wang et al. [187], on the other hand, used n-grams to analyze network packets and build a machine learning intrusion detection system called PAYL. During the training phase, PAYL profiles application payloads of network packets using frequency distribution of n-grams and their standard deviation. PAYL does that for packets with the same length, destination address and port. A size of $(n = 1)$ is used to simplify the system's computations. A distance function (Mahalanobis, a standard distance metric in statistics) is used at the production phase to measure the distance between existing profiles and the inspected new data using a predetermined threshold. Experiments showed successful results with low false positives. The work on PAYL was further extended in order to detect zero-day worms and generate their signatures [185].

Anagram [186] is another payload-based machine-learning algorithm (created by the same group) that uses n-grams to analyze network traffic for anomaly detection. While PAYL computes a profile of byte n-grams (i.e., $n = 1$) frequency distribution, Anagram models high-order n-grams (i.e., $n \geq 2$) to capture consecutive byte information. Relying on the n-grams' special frequency distributions within packet payloads, both PAYL and Anagram were able to gauge similarity between packets that share the same application, length, host, and port.

In recent research, Sheu et al. [158] proposed a two-tier enhanced hierarchical multipattern matching (EHMA) algorithm for packet inspection in a network intrusion

detection system (NIDS). Their algorithm assumes that most packets are innocent ones, and looks for their common grams. It then uses these grams to narrow down the search for bad packets. Their matching process concentrates patterns into a small on-chip table for performance efficiency. Their simulation results show relatively good performance compared to other similar algorithms in [157], [181], and [105].

Also, Wei et al. [106] proposed an unsupervised botnet detection system that divides traffic into known applications and then clusters traffic on each application community, using n-grams extracted from the network flows. The purpose of their work is mainly to find anomalous behaviours. Both [106] and [158] have found n-grams with (n=1) to be good enough to achieve their results.

Matrawy et al. [113] were the first to propose (p, n)-grams for traffic shaping of network packets. (p, n)-grams in network packets could be thought of as a special case of n-grams with a sliding window (or simply n-grams with offsets p). However, using n-grams for packet matching over time requires maintaining a sliding-window state for each n-gram used at each time. The added offset p is what maintains this information for each n-gram.

Their algorithm (i.e., [113]) constructs 50 queues and associates 20 (p, n)-grams with each queue, for a total of 1,000 (p, n)-grams. (p, n)-grams are selected based on their matching frequency in an earlier day. Each packet is checked sequentially against the set of (p, n)-grams at each queue. The packet gets forwarded to the queue where the match occurs or otherwise to a default queue. Each queue is served with an equal share of the network bandwidth. The goal of this system is to mitigate the effect of DOS attacks through grouping similar packets together in one queue with a limited share of the bandwidth. That is, if the queues manage to do a proper classification, and one of the queues starts to receive a worm or flash crowd traffic, it will be exclusively affected. The undesired traffic would only consume a limited size of the bandwidth that is originally assigned to each queue.

2.3 Analyzing Traffic Using Behavior Information and Other Header Fields

Analyzing network packets through their behaviors usually relies on behavioral and flow information all located in the packet header fields. That is, the approach looks for special host communication patterns to differentiate between packets of various applications. In this approach, packet sizes, directions, and inter-arrival times, are examples of packet behavior characteristics that can be correlated together [136]. Correlation usually goes across different flows, but could also go across different sessions. For example, Bernaille et al. [13] use the sizes of the first six packets in a session as the protocol signature.

Paxson and Floyd [135] studied packet arrivals in network traffic, and showed that in the case of wide area network (WAN) traffic, a Poisson distribution may only apply to packet arrivals of certain traffic types like TCP file transfer and remote login. In addition, Leland et al. [97, 98] researched the statistical self-similarity properties of Ethernet traffic.

Karagiannis et al. [91] classified traffic by examining host behavior. Their strategy was to observe host behavior at the social level (host interaction), functional level (popularity, role), and transport level. They reported classifying 80%-90% of traffic with 95% accuracy. Also, Bernaille et al. [14] studied the feasibility of early clustering of applications by observing the size and direction of the first few packets of a TCP connection.

Another example of packet behavior analysis is that of Kumpulainen et al. [96], in which they used a multi-layer clustering algorithm to monitor traffic patterns and characterize servers and devices generating the traffic. They collected behavioral information from packet headers (e.g., sending and receiving sequences, TCP and UDP connections, etc.) of each address within one-hour time frames. They used that then to create informative variables that describe the traffic.

Borders et al. [17] use their Web Tap tool to differentiate between legitimate HTTP traffic and other protocols that use covert channels (through HTTP tunnels)

to communicate with web servers behind the firewalls. Filtering with Web Tap uses traffic behavior parameters such as request regularity, bandwidth usage, inter-request delay time, and transaction size. Similarly, Pack et al. [133] use packet features such as packet size, change of packet size, and packet interleaving time to construct behavior profiles of application network sessions and detect HTTP tunneling activities.

One limitation of behavior-based traffic analysis is that it usually overlooks some of the key protocol-specific information that is within the packet contents, and can be used to identify or fingerprint the original protocol type. This information can usually be derived from the protocol-specific fields within the headers and payloads of packets [107].

2.3.1 Characterizing Encrypted Traffic

While payload-based traffic analysis techniques assume differences in packet contents for different traffic or protocol types, behavior-based techniques assume differences in behaviors. Although the two approaches usually complement each another in traffic analysis, the strength of each approach relies on which differences are manifested in the protocol implementations. This, however, may vary depending on the protocols being analyzed.

A common example where behavior-based traffic analysis usually gives more accurate identification is when dealing with disguised protocols [90, 91]. That is, P2P applications usually use encrypted or compressed payloads, and may disguise their ports (e.g. using the HTTP port 80) to escape traffic shaping; however, their packets usually feature a special communication behavior that can be distinguished from the normal behavior of HTTP traffic [75].

For instance, Wright et al. [194, 195] observe that different application protocols have different behaviors of packet exchange between the communicating parties. Studying and profiling these different behaviors on unencrypted traffic, and then matching the behavior of encrypted traffic to one of them allows a relatively accurate guessing of the encrypted traffic protocol type and ID. In addition, Gebski et al. [56]

rely on the size, timing and direction of network packets to profile encrypted traffic using a graph-comparison approach.

Three more examples of characterizing encrypted protocol identification are those of Wright et al. [193], Sun et al. [174], and Liberatore et al. [102]. Wright et al. [193] use the length behavior of encrypted Voice-over-IP (VoIP) packets in a session to predict the language used in a phone conversation. This was found possible when Variable Bit Rate (VBR) coders were used for encoding. On the other hand, Sun et al. [174] observe that visiting different pages on the web results in different number and size of object downloads. Studying and reporting traffic signatures of thousands of web pages and then checking the encrypted web traffic against these signatures can help identifying the visited web pages. Similarly, Liberatore et al. [102] have worked on identifying the sources of encrypted HTTP connections using similarity checks to a library of known profiles.

2.4 Analyzing Traffic Using Machine Learning and Statistical Analysis

Machine learning algorithms are commonly used to analyze network traffic. Although machine learning analysis can be done using any of the packet parts including headers and payloads [130, 36], it has shown promising results even with traffic that may preclude payload analysis (e.g., encrypted and obfuscated traffic).

There are three common types of machine learning algorithms: supervised, unsupervised, and semi-supervised (or constrained clustering). Approaches with supervised learning start with building a model using training and labeled data and then using this model to classify subsequent traffic [119, 101, 190, 51]. Unsupervised learning approaches, on the other hand, use clustering algorithms to cluster flows together based on their similar characteristics without labeled data [114, 50, 199, 200]. Finally, semi-supervised or constrained clustering combines a clustering algorithm and some set of rules like a must-link and can't-link constraint to increase clustering

accuracy [188, 192, 159].

In all the above three learning types, however, proposed solutions vary between those based on traffic flows, payloads, behavior, and/or a combination of them. For example, Wang et al. [188] proposed a semi-supervised clustering that focuses on packet flow correlation information. On the other hand, Szab et al. [175] proposed another semi-supervised clustering algorithm focusing on connectivity patterns (behaviors) such as packet sizes and arrival times. Dehghani et al. [39], and Wong et al. [192], proposed a hybrid supervised algorithm based on payload contents as well as packet behavior statistical features such as size and inter-arrival time. While the work by [39] worked well with HTTP and FTP applications, the work by [192] targeted the BitTorrent traffic detection.

It is important to note the variety of clustering algorithm techniques used in the literature. Among the common ones are hierarchical clustering, decision trees, K-Means algorithms, statistical distributions, genetic algorithms, Bayesian networks, association rules and self-organization maps, SVM, and neural networks [125, 202, 8, 159].

Clustering traffic into application classes using machine learning was also studied by Zander et al. [200] where they used an approach based on Bayesian classification. McGregor et al. [114] clustered application traffic using the Expectation Maximization algorithm. Also, Roughan et al. [148] clustered traffic into different QoS classes.

Other applications of machine learning algorithms have been in the area of traffic fingerprinting and classification. For example, Haffner et al. [60] proposed a machine learning algorithm that automates construction of application signatures. Moore et al. [118], on the other hand, suggested an iterative classification algorithm that operates on flows. Moreover, Sen et al. [155] and Choi et al. [26] have proposed an algorithm to inspect available documentation and packet-level traces, and a content-aware application traffic measurement, respectively.

2.4.1 Protocol Fingerprinting

In protocol fingerprinting, content and/or behavior of network packets are analyzed to identify specific features of network protocol implementation [160]. This can be on the syntax level or semantic level, and can go on the headers and/or payload level of network packets. Examples of syntax level fingerprinting include the work of Beverly [15], in which he shows TCP/IP traffic header fingerprinting using probabilistic learning can identify a host's OS. Moreover, fingerprints of worm attacks are commonly made through substring signatures within the packet payloads. A substring, such as: "X-Kazaa-*" is an example of a common string that may identify the Jazz P2P protocol [107].

Website fingerprinting is an example of identifying semantic information even when encrypted tunnels are used. Gong et al. [57] were able to fingerprint a website with 80% accuracy using round-trip time (RTT) calculations from a virtual machine that tries to simulate the network conditions on the user's home network. Cai et al. [22], on the other hand, used packet bahavior information such as timing, direction, and size to fingerprint websites with accuracy between 50% and 90%.

Fingerprinting network packets can be created manually prior to traffic analysis such as those hand-crafted strings used in Snort [147], and Bro [134]. Alternatively, they can be automatically generated using signature extracting algorithms such as those used in EarlyBird [162, 163], HoneyComb [95], Netbait [27], and Autograph [93], or using machine learning algorithms such as that used in [21].

The manual approach assumes *a priori* knowledge about existing protocols, and can't help with zero-day protocols or applications. The automatic approach, however, allows systems to extract signatures while analyzing the byte representation of network packets in order to achieve automatic pattern inference and/or generation of attack signatures. For example, Dusi et al. [46] presented a statistical fingerprinting method based on behavior header information at the network layer (e.g. packet sizes, and inter-arrival times) that can detect application-layer tunnels and enforce network-boundary security policies.

On the other hand, Zhang et al. [201] proposed another statistical fingerprinting method that can detect stealthy P2P Botnets. Their method fingerprints the Command-and-Control communication patterns and use that to distinguish between hosts of legitimate P2P networks and P2P bots. Similarly, Tegeler et al. [179] proposed BotFinder, an algorithm that can automatically build multi-faceted models for Command-and-Control traffic of different malware families.

Moreover, in recent research work, Shu et al. [160] proposed a formal model for fingerprinting based on a Finite-State Machine (FSM) that can specify complex protocols like the TCP congestion control and SSL handshaking subprotocol.

2.4.2 Protocol Inference and Identification

The work on protocol inference and identification extends from or overlaps with the work on protocol fingerprinting and classification. For example, deep packet inspection can be used to fingerprint protocols and identify them based on their special matching. In behavior analysis, traffic protocols can be inferred based on their packet behaviors. Therefore, it is common to find the related research focusing on specific protocol identification such as encrypted traffic (See Section 2.3.1), Bit torrent, and other P2P traffic [202], Skype [125], etc.

It is common to find that pattern matching is performed through finite automata (FA). Antonello et al. [5] have observed that several consecutive transitions in FA lead to the same destination state. Thus, they proposed a range compressed deterministic finite automaton (RCDFA) that aims to decrease space requirements when used to perform pattern matching.

Interesting protocol inference and identification research was introduced by Ma et al. [107]. They proposed an unsupervised protocol inference framework that relies on common flow contents (packet strings), not just flow information. Basically, they classify traffic by building statistical models of messages exchanged in a protocol without relying on port numbers to identify applications. Their approach assumes that flows from the same application/protocol possess content similarities that can

distinguish them from others. Thus, it looks for common flow content and uses that to identify traffic that employs the same application/protocol. Using three classification techniques (product distributions, Markov processes, and common substring graphs), they were able to capture statistics and structures of messages exchanged in a protocol, and use that to group protocols without relying on ports or other *a priori* knowledge about protocol structures.

Now that we briefly discussed the main areas of related work, it is time to put our work in context.

2.5 Our Work in Context

In this research, our goal is to use another representation of traffic that can extend the advantages of using n-grams to efficiently find common patterns in network packets. Therefore, based on the previous related-work discussion, our (p, n)-gram research may be relatively closer in implementation to the n-gram based traffic analysis algorithms.

When compared with other signature-based or field-specific traffic analysis techniques, n-gram-based traffic analysis helps in better extracting interesting patterns within network packets as it assumes no *apriori* knowledge about where in the packet they are located. However, it does not give a semantic meaning on what these strings may belong to within the protocol packets (e.g., header vs payload). Moreover, n-gram-based algorithms compare each string to the whole packet for pattern matching. a process that at best requires linear time.

Our research, however, augments the capabilities of n-grams by adding some semantic meaning to each n-gram to help in the analysis part. In addition, instead of looking at the whole packet for each pattern matching, we follow the approach for which high-speed routers are optimized in order to achieve better scalability: matching a sequence of bytes at specific packet locations (using offset p).

With respect to the machine learning algorithm used, we use ADHIC (Approxi-

mate Divisive HIerarchical Clustering), a simple unsupervised machine learning clustering algorithm that uses a dynamic binary decision tree to divide and cluster network traffic and create a decomposition of the inspected traffic in real-time fashion.

On the other hand, considering the different ways to analyze packet parts discussed above (port-based, flow-based, behavior-based, and payload-based), our (p, n)-gram-based research offers a simple and efficient traffic characterization technique that achieves combined advantages as it looks at the entire packets with all fields with equal attention. For instance, (p, n)-grams may represent port numbers, flow data (5-tuples), payload data, as well as packet behavior information (e.g., packet length, time-to-live (TTL), options, etc.).

Our work is similar to that of Ma et al.'s [107] in that it uses an unsupervised header/payload machine learning approach to group protocols without looking at specific packet fields or assuming *apriori* knowledge about network protocols. They both suggest unexpected (non-traditional) means to infer network protocols. For example, we observe that using (p, n)-grams in characterizing network traffic can discover payload patterns within protocols and sub-protocols that can go cross-flow in network packets. However, in addition to the algorithm differences, our work is different from Ma et al. in that it represents network packets using (p, n)-grams, where each (p, n)-gram consists of a 2-byte string and an offset, as opposed to the classical representation using just strings (with fixed or variant sizes).

In summary, with the goal of augmenting other traffic analysis algorithms to achieve high-level traffic analysis, our work is different from all n-gram-based approaches discussed earlier in three ways: objective, methodology, and efficiency. From the objective perspective, we want to build a blind packet-structure system that can relate packets through their semantic structure similarities as well as content similarities without *apriori* knowledge of the traffic protocols (Chapters 3, 4, and 5).

From the methodology point of view, we attach a semantic meaning to each n-gram through adding an offset p. This increases the domain space and gives traffic analysis another dimension. It also allows us to efficiently capture specific protocol

structures that can be used to fingerprint different network protocols (Chapters 6 and 8). Finally, from the efficiency perspective, packet matching with (p, n)-grams requires comparing the (p, n)-gram only with the packet's n-byte sequence at offset p, as opposed to comparing with the whole packet as in the n-gram case. A common efficiency similarity though between (p, n)-grams and n-grams is their power-law-like distribution behavior that allows the relatively frequent ones to be easily distinguished from the rest (Chapters 6 and 7).

In comparison to the DOS mitigation work by Matrawy et al. [113] using (p, n)-grams, our work researches the (p, n)-grams' ability to fingerprint network protocols efficiently and use that to cluster and monitor network traffic. Our research is based on empirical analysis and conceptual models to support our hypotheses of (p, n)-grams and their characteristics in network traffic (Section 1.3). In particular, they support our hypotheses of the (p, n)-grams ability to capture protocol structures for fingerprinting purposes, and their special power-law-like distribution behavior that allows them to be efficiently distinguished from other (p, n)-grams found in the traffic.

3 Introducing ADHIC

This chapter and the following one present collaborative work I did with Hajime Inoue to develop a clustering algorithm that can cluster network traffic efficiently and effectively. We defined the clustering application requirements and designed ADHIC (Approximate Divisive HIerarchical Clustering algorithm) [65, 80]. The empirical analysis and theoretical insights in later chapters, however, are my own work [64, 63].

In this chapter, we introduce our clustering algorithm ADHIC [63, 65, 64], and discuss how it works. We also reference its current prototype implementation NetADHICT (Network Approximate Divisive HIerarchical Clustering Tool) [80]. NetADHICT is licensed under the GNU General Public Licence (GPL), and is available from the Carleton Computer Security Laboratory (CCSL) website [79]. Chapter 4 presents an empirical analysis of ADHIC's performance on real network traffic. Our empirical results show the effectiveness of ADHIC to cluster and classify network traffic for network management and security purposes.

3.1 Rationale Behind ADHIC

In a nutshell, ADHIC is a binary-tree-based clustering algorithm that continuously divides monitored traffic based on packet matchings with automatically calculated (p, n)-grams. ADHIC analyzes monitored packets to find (p, n)-grams with high frequency and then applies a divisive hierarchical clustering algorithm (a standard machine learning method) to build a dynamically changing binary tree whose leaf nodes

constitute clusters of semantically similar packets.

The objective of this research is to find a technique that can efficiently cluster network packets into semantically meaningful classes without using any domain-specific knowledge. Our approach to achieve this is to find the packet design structures, for fingerprinting purposes (See Section 2.4.1), that are already present, even if we do not know them, rather than the structures that we expect to find. We, therefore, target an *unsupervised* algorithm that can efficiently cluster network traffic, while promptly adapting to the recurrent changes in network traffic.

Our work is inspired by earlier work in our research group (see Section 2.2.1) on mitigating distributed denial-of-service (DoS) attacks using (p, n)-grams [113]. Initially, ADHIC was to be a more scalable version of this algorithm. As we will show, though, ADHIC is particularly suited to semantic clustering of network packets.

Our approach is different from the other network traffic clustering and classifying methods mentioned in the related work chapter (i.e., Chapter 2) in three ways. First, ADHIC does not rely on any previous knowledge of packet contents, nor does it assume particular byte ranges as fields of interest in the learning process. Second, although ADHIC is an unsupervised clustering algorithm, the clusters are semantically equivalent without requiring pre-labelling. Third, ADHIC works on raw network packets and does not require flow reconstruction, packet reassembly, or packet normalization to perform its traffic clustering, making it inherently more lightweight than methods that do require such preprocessing (see Section 2.2).

We choose a *hierarchical* clustering approach since Internet traffic has an encapsulated structure. For example, HTTP is encapsulated in a TCP session, whose packets are encapsulated in IP and, typically, Ethernet packets. Such a structure is best represented with a hierarchy rather than with a simple collection of clusters.

We also choose to create a *divisive* hierarchical clustering algorithm that works in a top-down fashion. Divisive clustering is a good fit for our goals as we want to capture large scale patterns rather than the fine-grained details of network behavior. Our divisive clustering algorithm assumes that all data belongs to one cluster at

the beginning. It then iteratively divides the cluster into smaller ones to capture finer-grained details [88, 62].

Existing approaches to divisive hierarchical clustering typically employ an entropy minimization calculation, which is $O(n^2)$ in the number of clustered items [44]. This is, however, too expensive for a high-speed implementation. Therefore, we target a *linear* algorithm ($O(n)$) in the number of inspected packets, and *sub-linear* in the number of Bytes within inspected packets (i.e., an algorithm that only needs to look at a small portion of the packet before making a clustering decision).

Our insight here comes from how high-speed routers are designed to quickly classify incoming packets. High-speed routers look at a subset of bytes within the packet (i.e., destination IP address) to determine where the packet should go. Our choice, therefore, has been to base our divisive hierarchical clusterer on a generalization of what high-speed routers are optimized to observe: (p, n)-grams. A (p, n)-gram, in our research, corresponds to a byte-sequence of size n byte(s) and offset p in network packets, where p starts at the beginning of the Ethernet frame. (p, n)-grams can represent common patterns within packets so long as they appear at fixed offsets within the packets. Patterns such as the presence of a string anywhere in a packet, however, cannot be efficiently represented using the offset-based (p, n)-grams.

3.2 How ADHIC Works

ADHIC recursively divides traffic into binary classes, with each subdivision being defined by the presence or absence of a given (p, n)-gram. It stops dividing classes when the resulting traffic is below some configurable threshold in volume. ADHIC also produces a binary decision tree that consists of 1) internal nodes, where each is populated with a decision (p, n)-gram, and 2) leaf nodes, which constitute the final (*terminal*) clusters.

ADHIC is a single-pass algorithm that looks at each packet only once while clustering network traffic. While ADHIC needs to inspect all packets for (p, n)-gram

matchings, it accelerates the clustering process by sampling fewer packets for (p, n)-gram frequency analysis and tree generation. In most of our experiments, we sampled only 20% of all packets for the tree generation part because we were interested in minimizing error and maximizing repeatability. Section 3.3.2 shows that, for the datasets we worked with in our network, only 3% of packets are needed for the tree generation process to achieve an error rate of less than 5% for (p, n)-gram frequencies. ADHIC utilizes this *approximate* divisive hierarchical clustering approach to assign packets to clusters and adapt the cluster decision tree to changing traffic patterns simultaneously.

3.2.1 Introducing ADHIC Trees

Figure 3.1 shows a simple ADHIC binary decision tree captured at an early stage (a typical mature tree has slightly over 100 total nodes). In this tree, all nodes have a node identifier (e.g., N2) and two entries of traffic statistics of packet counts and percentage. Those two statistics entries represent the number of packets encountered in two time periods (windows), namely: *update period* (currently set to last 10 min), and *maturation window* (currently set to last 3 hours). We introduce these time periods in Section 3.2.3.

Each of the internal nodes (i.e., N2, N5, and N8) is associated with a (p, n)-gram (e.g., 21, 0x00 0x71 for N2). Left branches in the tree indicate matches, while right branches indicate the absence of a given (p, n)-gram. The circle size of a leaf node (cluster) varies depending on the number of packets seen in each node over the last update period. The numbers at the bottom of each leaf node tells the number of different protocols represented by packets in the cluster.

Each protocol type within a cluster is represented by a slice, where the size of a slice reflects the relative volume or number of packets. Slices with one color represent Ethernet protocols (e.g., ARP); two colors (slice + one stripe) are IP protocol types (e.g., EIGRP); three colors (slice + two stripes) are specific TCP and UDP protocols (e.g., HTTP). See Table 4.1 for protocol-color matchings.

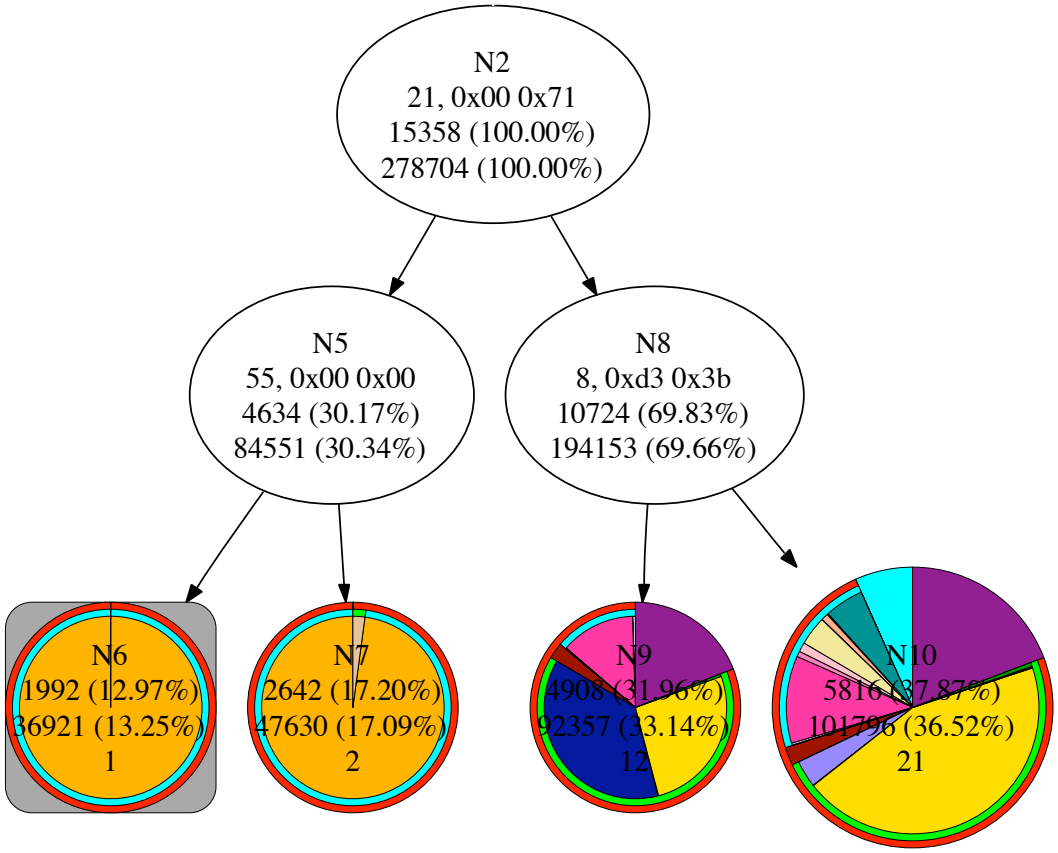


Figure 3.1. (Best viewed in color) An example ADHIC decision tree. Terminal clusters are represented by pie charts (color key provided in Table 4.1). Singular clusters are presented on a gray box.

It is important to note that the colors come from a "reference classifier" that scans packets in the tree nodes (clusters) after they get generated by ADHIC. The reference classifier is a port-based classifier that we use in addition to ADHIC to compare its output with the classical classification (i.e., through port numbers) of packets. We discuss more about the reference classifier in Section 4.2.

3.2.2 Traffic Clustering within the Tree

Traffic that matches a classification rule within the node (i.e., a decision (p, n)-gram) is said to *match* and is directed to the left, or *true* subtree. The rest of the traffic is

directed to the right, or *false* subtree. Because rightmost subtrees have not matched any of the classification rules within their subtrees, we sometimes refer to these as *default* clusters. The rightmost cluster of the entire tree is the *global default cluster*. See Figure 3.2 for a pseudocode representation of the ADHIC matching algorithm.

```
for each packet:
    start at root node
    while node <> leaf_node
        if(node.png in packet)
            node = node.left
        else
            node = node.right
    assign packet to leaf_node
```

Figure 3.2. Pseudocode for the ADHIC matching algorithm.

Traffic within each terminal cluster can be viewed as packets that were filtered by a boolean equation constructed through a path from the root node to the leaf. Left subtrees are combined with **and**, whereas, right subtrees are combined with **and not**. On the other hand, leaf nodes with rounded grey boxes mean that all packets within that cluster belong to the same protocol type (thus, number 1 is at the bottom). We call these special nodes *singular* clusters as they contain packets of one protocol only.

For example, Figure 3.1 shows traffic split into two subtrees through the following decision (p, n)-gram: (21, 0x00 0x71) (Table C.1 presents a basic preview of the common IP TCP/UDP packet structure). The two subtrees were further split into four terminal (leaf) clusters, where traffic that matches (21, 0x00 0x71) is then matched against (55, 0x00 0x00).

In this example, the two left terminal clusters are dominated by packets from a proprietary MP3-Stream protocol (similar to the Real-time Transport Protocol -RTP). Packets in these two clusters feature a special "fragment offset" and "time-to-live" value, namely: 0x00 at offset 21, and 0x71 at offset 22 respectively. ADHIC automatically computes the corresponding (p, n)-gram of these two bytes (i.e., (21, 0x00 0x71)) through its relatively high frequency. It then uses this (p, n)-gram as a fingerprint to segregate the MP3-Stream packets and cluster them in the left hand

side of the tree.

3.2.3 Basic Tree Operations

ADHIC's trees first start with one root node, and consistently get updated, over time, through two operators: **split** and **delete**. Splitting is attempted when a leaf cluster matches too much (more than a *split threshold*) traffic during the most recent *maturation window*. Nodes which have been modified within a maturation window of the current time are locked and cannot be split or deleted.

To split, we search for a (p, n)-gram that matches approximately half (50%) of the packets in the cluster. We refer to the matching percentage range as the *similarity spread*. In our experiments, we set the similarity spread parameter to 20%, split threshold to 2%, and maturation window to 3 hours (see Table 3.1). Thus, using the current settings of ADHIC, a leaf cluster will split if it matches more than 2% of the packets in the past 3 hours, and a (p, n)-gram is found such that it exists in 40% to 60% of the packets in that cluster. Note that ADHIC chooses the first (p, n)-gram it finds within the similarity spread and assigns that to the new decision node it creates.

```
for each node:
    if(node.traffic_perc < min)
        parent.delete(node)
    else if(node is_leaf && node.traffic_perc > max)
    {
        png = find_png(node, cache)
        if(png.found()) node.split(png)
    }
```

Figure 3.3. Pseudocode for the ADHIC adjustment algorithm. `cache` holds a sample of recently observed packets.

Deletion occurs when a subtree has not matched a minimum threshold of traffic percentage (currently set to 0%) during the most recent maturation window. The subtree's parent node is also deleted. The parent node's other subtree, the one not deleted, becomes the direct child of the parent node's parent. See Figure 3.3 for a

pseudocode representation of the ADHIC adjustment algorithm.

For performance reasons, splitting and deletion do not occur continuously; instead, they are restricted to *update period* intervals of several minutes (currently set to 10 minutes). The similarity spread, maturation window, update period, and the thresholds for splitting and merging are all configurable parameters. Proper configuration of these parameters may depend on more than one criterion, such as volume of network traffic to be analyzed, number of different protocols involved, change frequency in application activities, desired zooming-level of traffic monitoring, etc. Table 3.1 shows the values we used in most of our experiments. Those parameters were set based on experimenting with multiple value options.

Parameter	Value	Parameter	Value
(p, n)-gram length	2		
maturation window	3 hours	update period	10 minutes
delete threshold	0%	split threshold	2%
sampling rate	20%	similarity spread	20%

Table 3.1. ADHIC parameters used in most of our experiments

We considered complementing the **and** and **not** operators of ADHIC with an **or** operator implemented through multiple internal nodes with multiple (p, n)-grams. Internal nodes would acquire multiple (p, n)-grams by being merged with other nodes rather than being deleted. Individual (p, n)-grams within nodes would then be deleted if they did not match a packet within the maturity period. We found, however, that the quality of the clusters produced by ADHIC with either operator regime was similar.

By viewing packets as individual (p, n)-grams, the algorithm treats packets as high dimensional vectors, where the number of dimensions is the packet's length $-n + 1$. Note, however, that these dimensions are not independent. The effective information provided by the (p, n)-gram vector is reduced by its overlapping nature; also, due to the often observed non-random nature of packet contents, the presence of one (p, n)-gram often affects the probability of other, non-over-lapping (p, n)-grams.

ADHIC's splitting method is far more efficient than that used in most divisive

hierarchical clustering methods. Most choose a split that minimizes entropy in the generated groups [44]. Entropy minimization is a natural choice because it ensures that similar items are grouped together. Unfortunately, entropy minimization is a slow calculation as it requires a separate computation for each of the $2^m - 2$ choices for each split (where m is the number of packets in the original cluster).

3.3 ADHIC Performance

Throughout this research, we test performance of ADHIC through its NetADHICT implementation. We run most of our experiments with NetADHICT on an Apple Mac Pro with 1 GB of main memory and 2.66 GHz "Woodcrest" cores. Using this hardware, the single-threaded NetADHICT (with logging minimized) is able to cluster packet data at about 250 Mbps.

While its current speed is more than sufficient for a research prototype, NetADHICT currently is not fast enough to monitor high-speed links. The lightweight nature of our algorithm, however, should permit much higher-speed implementations. Such work is a topic for future research.

The following two subsections discuss the lightweight nature and performance characteristics of ADHIC and its current implementation (NetADHICT) in light of two aspects: (p, n)-gram representation, and packet sampling.

3.3.1 (p, n)-gram Representation

Existing approaches to divisive hierarchical clustering typically employ an entropy minimization calculation, which is $O(n^2)$ in the number of clustered items [44]. This is, however, too expensive for a high-speed implementation.

In comparison, our (p, n)-gram based approach applies a *sub-linear* (in the number of packet Bytes) algorithm that only needs to look at a small portion of the packet before making a clustering decision. This is also to be compared with common n-gram

based algorithms that compare each string to the whole packet for pattern matching; a process that requires linear time.

On the other hand, when it comes to the process of selecting (p, n)-grams for clustering, we find that the rapidly-dropping-off distribution of (p, n)-grams with a power-law-like behavior (see Chapter 7) gives an additional *space* efficiency advantage. In essence, this distribution behavior implies that the structural (p, n)-grams are easily distinguishable from the rest in the long tail due to their unique high frequency. It also implies that only a small set of structural (p, n)-grams is required for the traffic characterization applications.

In addition, our current implementation of ADHIC (i.e., NetADHICT) also applies a specific update-periods policy to do tree splitting and deletion (see Section 3.2.3) in order to reduce this overhead while keeping the dynamic tree sensitively adaptive to new traffic changes. NetADHICT also has a max tree height policy that limits the number of checks each packet has to go through before it reaches its final cluster.

3.3.2 Packet Sampling

Our choice of using (p, n)-grams for traffic characterization allows ADHIC to use only a small number of packets, through packet sampling, in order to construct its clustering decision tree. ADHIC utilizes this *approximate* divisive hierarchical clustering approach to assign packets to clusters and adapt the cluster decision tree to changing traffic patterns simultaneously.

In essence, ADHIC uses unbiased packet sampling while searching for (p, n)-grams within the similarity spread matching percentage range. That is, ADHIC uses individual (p, n)-grams as a proxy for a more expensive entropy calculation. If we assume a normal distribution of the traffic packets, then the sample size (m) required to have an error rate of (B) or less, is estimated using the well known simple formula [94]:

$$m = \frac{1}{B^2} \tag{3.1}$$

That is, for a 5% error rate or less, the sample size should be 400 packets. Network traffic, however, does not follow a normal distribution [135], and thus, we must sample more. In all our experiments referenced in this thesis, we choose a conservative *sampling rate* of 20% (see Table 3.1).

In our implementation of ADHIC, we use a pseudo random number generator function to do the sampling. That is, for each inspected packet, we generate a random number "r" between 0 and 1 and compare it with 0.2. If $r \leq 0.2$, we sample the packet for (p, n)-gram frequency analysis, otherwise, we don't.

It is important to note though that we found the proper sampling rate to be network dependent. For most of our tested datasets, a sampling rate of only 3% of the inspected packets is enough to achieve an error rate of less than 5% for (p, n)-gram frequency distributions. More on sampling is discussed in Chapter 7.

4 Clustering Network Traffic Using ADHIC

Chapter 3 discusses our network traffic clustering algorithm ADHIC and explains how it works on a high level. This chapter tests ADHIC's functionality and performance using experiments with four datasets (described in Section 4.1). The primary purpose of these tests is to analyze the accuracy and efficiency of ADHIC in clustering network traffic and to confirm its consistent behavior in different network environments, thus reducing the probability that our conclusions might be network dependent.

The chapter first describes the experimental setup we used throughout this research, and introduces our port-based reference classifier which we use to evaluate ADHIC performance (Section 4.2). The chapter then describes how ADHIC works on our main dataset (Section 4.3), and examines its performance in contrast to the reference classifier (Section 4.4). It finally examines ADHIC's performance on other networks we had access to (Section 4.5).

The next chapter (Chapter 5), on the other hand, tests ADHIC's performance on our main dataset without looking at the packets' header portion (Section 5.1). It also discusses how ADHIC can be further used to classify network traffic for network management and security purposes. That is, it examines the ability of ADHIC to detect evasive traffic using its (p, n)-grams approach.

4.1 Experimental Setup

This section presents the experimental setup that we used throughout this research. In particular, it describes the main network traffic datasets and traces that we used to test our developed applications (described in Chapters 4 and 5), as well as to verify (p, n)-gram characteristics in network traffic (described in Chapters 7 and 8).

4.1.1 Datasets Description

There are four independent datasets that we used throughout this research to verify our results and test our applications, namely: CCSL, MD, RMC, and LL. While the last one represents a synthetic dataset created at Lincoln Labs, the first three represent full captures of network traffic from three independent networks. This section describes the first two (i.e., CCSL, MD) as they constitute the datasets mostly used in this Chapter. On the other hand, RMC and LL are later described where they are exclusively used in Chapters 4 and 5, respectively.

CCSL and MD are both full-capture original datasets from which we extracted traces of various sizes. CCSL (Carleton Computer Security Lab) represents traffic captures from our research lab at Carleton University, Ottawa; whereas, MD (Maryland) represents traffic captures from a private sales company in Maryland.

The CCSL dataset belongs to a graduate student laboratory network with over 15 machines, two network printers and about a dozen regular users. The network provides common services to external hosts, such as a web server, web mail, email server, domain name, secure shell, and network printing services.

In particular, the dataset represents a network traffic capture of all incoming packets to the CCSL Lab where the destination MAC address is either a broadcast (ff:ff:ff:ff:ff:ff), a multicast (e.g., HSRP and EIGRP router protocols), or the specific CCSL's firewall MAC address. This consists of several months of traffic capture, of which we focus on four one-week long traces that correspond to the following periods: Aug. 13-19, 2004, Dec. 10-16, 2005, Jan. 20-26, 2006, and Apr. 3-9, 2006. In

contrast to the other three weeks, the August trace contains IP traffic only, where non-IP (e.g., ARP protocol [6]) packets were all excluded from the dataset.

On the other hand, the MD dataset represents a two-month long traffic capture from a private small-size sales company in Maryland. The network is comprised of over a dozen windows-based machines including two file servers and a Voice-over-IP (VoIP) phone system. The dataset is over 8 GB in size, and is mostly populated by web (HTTP and HTTPs), email (POP), and media streaming (RTP) traffic. The dataset consists of the company's incoming traffic captured during two months, starting from Oct 30, 2007 until Dec 26, 2007.

Table 4.1 provides a hierarchical composition of each of the four traces of the CCSL dataset, along with the first week trace of the MD dataset. It also provides protocol statistics for each protocol (Appendix B.2 provides a list of protocol names, references and acronyms). Note the hierarchical labeling of protocols. Protocols with one color level constitute Ethernet protocols including IPv4. Protocols with two color levels are IP protocol types. Protocols with three color levels are specific TCP and UDP protocols.

Depending on the purpose of the experiments, these datasets are used completely and/or partially (using random traces of sub-captures) to verify the (p, n)-gram characteristics in network traffic (further discussed in Chapter 7) and to test the performance of the developed applications.

It is important to note that during the experiments reported here we sampled 20% of all packets. As we described in Section 3.3.2, on our network only 3% of packets are needed to achieve an error rate of less than 5% for (p, n)-gram frequencies; however, we sampled at 20% because we were interested in minimizing error and maximizing repeatability.

We found that finding proper datasets for this type of research is not easy. In essence, most of the experiments done for verification purposes require datasets with full packet captures of real (as opposed to synthetic) network traffic. In these experiments, payloads are as important as headers, and both should be treated by the tools

Protocol	CCSL '04 Aug 13-19	CCSL '05 Dec 10-16	CCSL '06 Jan 20-26	CCSL '06 Apr 03-09	MD '07 Nov 01-07
IPv4	**100.00 %**	**82.34 %**	**79.00 %**	**86.66 %**	**88.34 %**
TCP	**51.24 %**	**48.24 %**	**50.71 %**	**53.73 %**	**72.66 %**
TCP Unknown	**10.34 %**	**7.53 %**	**1.01 %**	0.62 %	0.08 %
MS WBT/MS RDP	0.00 %	0.08 %	0.00 %	0.14 %	0.00 %
IPP	0.01 %	**4.98 %**	**4.64 %**	**6.88 %**	0.00 %
IMAPS	**1.01 %**	**1.03 %**	0.68 %	**2.35 %**	0.00 %
HTTPS	0.36 %	0.13 %	0.59 %	**1.75 %**	**14.81 %**
SSH	**14.75 %**	**4.32 %**	**4.15 %**	**3.37 %**	0.00 %
MS Streaming/RTSP	**2.36 %**	0.17 %	0.80 %	**1.38 %**	0.00 %
MSNMS	0.03 %	0.01 %	0.06 %	0.05 %	0.26 %
XMPP	0.00 %	0.01 %	0.01 %	0.02 %	0.02 %
TCP Sophos	0.00 %	0.07 %	0.24 %	0.08 %	0.00 %
TCP No Payload	**13.34 %**	**25.06 %**	**22.53 %**	**26.69 %**	**14.10 %**
RTSP	0.00 %	0.16 %	0.09 %	0.03 %	0.00 %
TELNET	0.12 %	0.00 %	0.00 %	0.00 %	0.00 %
FTP	**2.66 %**	0.23 %	0.04 %	0.07 %	0.00 %
SMTP	0.03 %	0.15 %	0.27 %	0.37 %	0.03 %
IMAP	0.00 %	0.00 %	0.03 %	0.01 %	0.00 %
CVS	0.00 %	0.00 %	0.00 %	0.16 %	0.00 %
POP	0.00 %	0.00 %	0.02 %	0.07 %	**7.64 %**
HTTP	**6.22 %**	**4.28 %**	**15.56 %**	**9.67 %**	**34.98 %**
AIM	0.00 %	0.00 %	0.00 %	0.00 %	0.73 %
UDP	**43.26 %**	**31.52 %**	**24.37 %**	**28.93 %**	**14.97 %**
UDP Unknown	0.01 %	0.05 %	0.05 %	0.01 %	0.02 %
DNS	0.43 %	0.52 %	**1.05 %**	0.95 %	0.80 %
CUPS	**6.58 %**	**2.58 %**	**3.65 %**	**1.81 %**	0.00 %
WHO	0.13 %	0.06 %	0.08 %	0.09 %	0.00 %
MP3-Stream	0.00 %	**13.90 %**	**2.04 %**	**3.51 %**	0.00 %
NBDGM	**1.31 %**	0.64 %	0.93 %	0.88 %	0.00 %
DCE_RPC	0.11 %	0.20 %	0.24 %	0.22 %	0.11 %
SIP	0.00 %	0.00 %	0.00 %	0.00 %	**1.80 %**
NBNS	**1.07 %**	**1.62 %**	0.61 %	**2.49 %**	0.00 %
RIPv1	**8.03 %**	0.37 %	0.49 %	0.59 %	0.00 %
HSRP	**25.55 %**	**11.39 %**	**15.15 %**	**18.28 %**	0.00 %
DHCP	0.01 %	0.16 %	0.03 %	0.02 %	0.01 %
NTP	0.01 %	0.04 %	0.06 %	0.07 %	0.10 %
RTP	0.00 %	0.00 %	0.00 %	0.00 %	**12.11 %**
ICMP	0.02 %	0.30 %	0.88 %	0.35 %	0.01 %
EIGRP	**5.48 %**	**2.28 %**	**3.03 %**	**3.66 %**	0.00 %
ARP	N/A	**17.28 %**	**20.58 %**	**12.29 %**	**11.66 %**
STP and DTP	N/A	0.38 %	0.42 %	**1.05 %**	0.00 %
Total no. of Packets	**5,117,600**	**11,422,323**	**8,622,721**	**7,075,868**	**7,275,137**
Total Size in GB	**2.7**	**2.8**	**3.1**	**1.8**	**1.6**

Table 4.1. (Best viewed in color and electronically to allow enlargement) Protocol statistics for the 1-week-long CCSL and MD network traces. Only protocols with percentages $\geq 0.02\%$ are shown, and with percentages $\geq 0.1\%$ are highlighted.

and algorithms without modifications (except for IP-address anonymization).

Therefore, although we *conjecture* that our results do not reveal private information, we had to face the very strict privacy policies governing these types of datasets. This explains the small number of datasets that we managed to experiment with. As for the datasets described in this section, we had proper permission from the users as well as sufficient knowledge of the network setup and running applications.

4.2 The Reference Classifier

Evaluating network traffic clustering algorithms can be looked at from different perspectives. However, regardless of how this is done, there needs to be a reference that we compare to. One of the main goals of our clustering algorithm is to cluster traffic blindly into semantically meaningful clusters. Thinking of a common reference that allows us to understand what ADHIC is doing and gives us quantitative results measuring accuracy of ADHIC led us to use a classical port-based classifier as a reference.

We understand that using a port-based classifier has its own limitations, especially with traffic that does not use its default port number. We, however, found that using this reference classifier would give us the closest metric translating to "semantically meaningful". We thus introduce the "*reference classifier*" as an independent port-based classifier that we use to better understand the behavior of ADHIC. Note that although ADHIC was not meant to work as a classifier in the first place, we measure ADHIC clustering performance using the conventional classification view of the reference classifier. We call this feature "*classification-like*" clustering.

Our reference classifier primarily relies on IP protocol and port information, but also monitors features such as Ethernet packet type. NetADHICT uses the reference classifier to label ADHIC's output trees by protocol types. The protocol labelling produced by the reference classifier allows us to quickly compare ADHIC with port-based traffic classification, a standard lightweight approach for understanding network

behavior.

We explore the differences between the reference classifier and ADHIC in Section 4.4. We also show how ADHIC can go beyond port-based classification, with the ability to cluster without headers (Section 5.1) and to cluster evasive traffic (Section 5.2).

Finally, in addition to the reference classifier, we have also verified the quality of ADHIC's clusters in our lab through the use of standard network analysis tools such as Wireshark [31].

4.2.1 Parameter Settings

Our experiments with ADHIC mostly used a set of parameter values (see Table 3.1) that were determined by exploring several options. Our evaluation of ADHIC relies on analyzing the decision trees produced after every update period along with the updated statistics.

In our testing environment, we found that ADHIC is not highly sensitive to most of these parameter values and it tends to produce qualitatively similar trees under many settings. We, therefore, have chosen the parameter values as a reasonable trade-off between accuracy and performance. For example, slightly better results were obtained with a two hour maturation window; analysis runtimes are much faster, however, with a three hour maturation window.

We also found that the optimal sizes of the maturation and update windows are better set with respect to the observed traffic volume. While shorter window sizes achieve better clustering accuracy, they require more processing and thus degrade the overall performance. For example, we found that a three-hour maturation windo, and a ten-minute update window give a reasonable tradeoff between speed and accuracy with the CCSL traces. On the other hand, we found an optimal setting of a one-hour maturation window and a one-minute update window when testing ADHIC with the dataset of the enterprise RMC network traffic (see Section 4.5).

One significant exception to parameter sensitivity, however, is the size of n, for

which we tested the size values between 1 and 4 and settled on a value of 2. Our choice is primarily based on the observation that although long (p, n)-grams provide a large amount of context (thus, giving more semantically meaningful splits), they are not found frequently. On the other hand, shorter (p, n)-grams are easier to find in large quantities, but they may not be as meaningful. We found that setting $n = 2$ with ADHIC produces the best tradeoff results on our datasets. Sections 7.2.2 and 7.2.3 examine the issue of the appropriate choice of n in more detail.

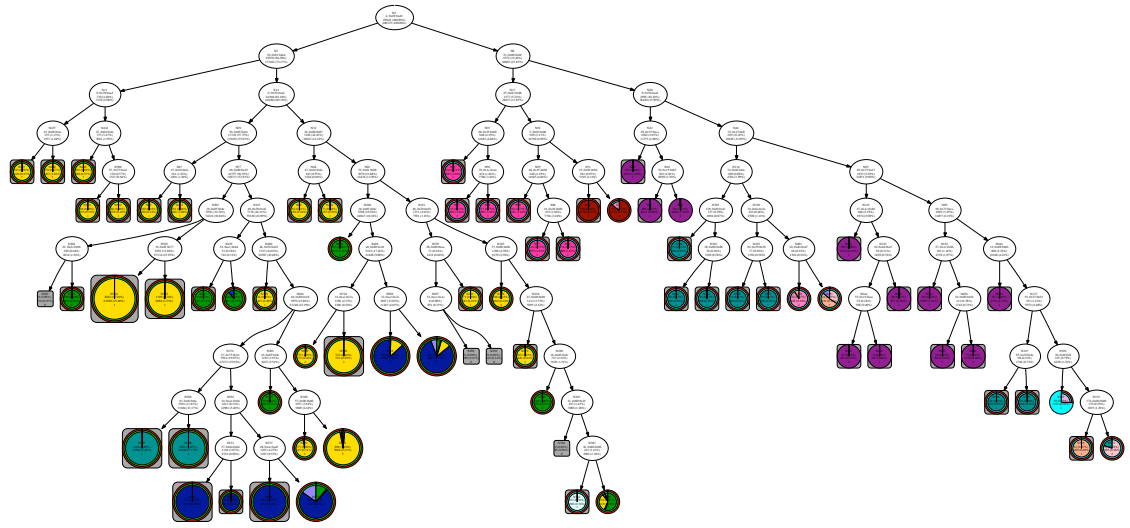
4.3 An ADHIC Decision Tree

This section describes an example decision tree and the types of clusters it produces. As explained in Section 3.2.2, the cluster labels in the ADHIC output trees come from the reference classifier, not ADHIC; ADHIC only generates the tree and produces the (p, n)-grams for each node.

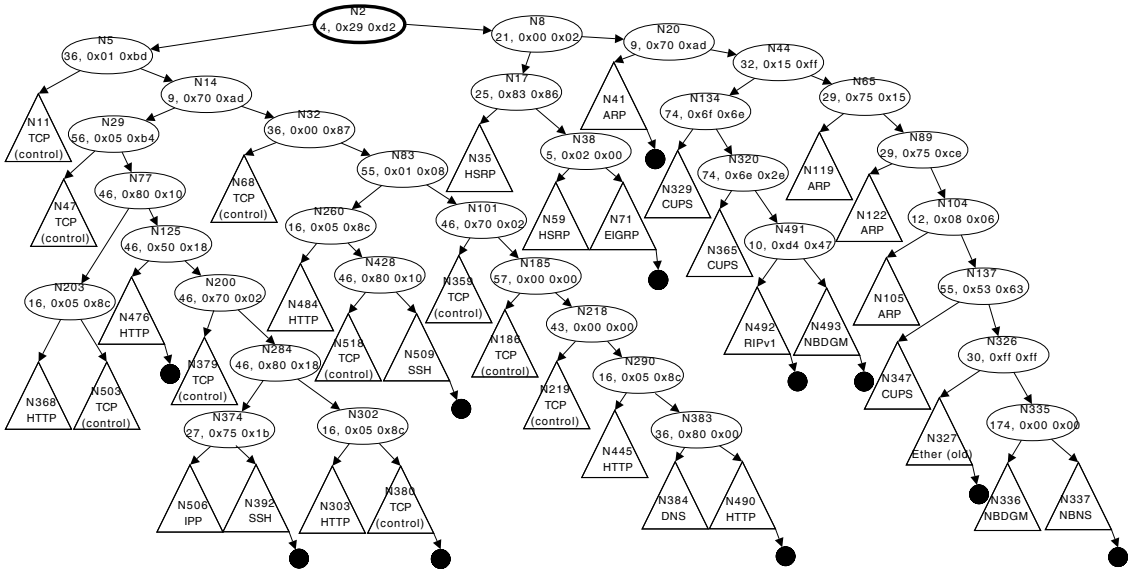
Figure 4.1 (a) shows an original decision tree produced in the morning of January 24, 2006, after four days of execution from the CCSL January trace. We have also added an annotated, symbolic version (Figure 4.1 (b)) that is easier to explain in the same figure. Each triangle in the annotated graph represents one or more terminal cluster nodes that contain the same protocol as in the original tree.

The black circles in the annotated graph are called *default* clusters, and constitute the rightmost child of subtrees. Default clusters are the product of several non-matching (p, n)-grams. Thus, packets in default clusters are "everything that is not something else". In the original tree graph, rounded gray boxes denote *singular* clusters. We define singular clusters as those that the reference classifier reports as clustering packets of one protocol type only.

In this particular tree, the (p, n)-gram at the root node (i.e., (4, 0x29 0xd2) at node N2) in the annotated tree belongs to the destination MAC address field. Therefore, the root node in this tree acts like a distinguisher that splits the tree into two halves based on their destination MAC address. While the left half belongs to packets



(a) A decision tree produced by ADHIC



(b) A simplified annotated tree

Figure 4.1. (Best viewed in color and electronically to allow enlargement) An original decision tree produced by ADHIC (4.1(a)) and a simplified annotated version (4.1(b)). The original tree contains 70 terminal clusters (leaves), of which 4 are empty, and 48 are singular. In the simplified tree, triangles represent subtrees, and filled circles represent default clusters.

that were exactly destined to our CCSL's firewall, the right half represents those that were broadcasted or multicasted. This is why the right half of the tree shows protocols such as CUPS (Common UNIX Printing System [34]), HSRP (Hot Standby Router Protocol [67]), EIGRP (Enhanced Interior Gateway Routing Protocol [48]), ARP (Address Resolution Protocol [6]), NBDGM (NetBIOS Datagrams [126]), NBNS (NetBIOS Name Service [127]), RIPv1 (Routing Information Protocol [146]), and some other old Ethernet protocols, such as STP (Spanning Tree Protocol [173]) and DTP (Dynamic Trunking Protocol [43]) (we labeled both as "Ethernet (old)" in the tree and in Table 4.1).

Another observation in that tree is that (p, n)-gram offsets at the internal nodes vary between the Ethernet header, IP header, TCP/UDP header, and payload. For example, the payload (p, n)-grams at nodes N134 (i.e., 74, 0x61 0x6e), N320 (i.e., 74, 0x6e 0x2e), and N137 (i.e., 55, 0x53 0x63) exclusively segregate the CUPS packets. Those common (p, n)-grams are part of the printer descriptions usually spelled out in the CUPS packets.

Another payload (p, n)-gram example is the one at node N335 (i.e., 174, 0x00 0x00) which exclusively segregates NBDGM packets before they go to the global default cluster. This (p, n)-gram represents one of the option fields in the NBDGM's Server Message Block Protocol. On the other hand, (p, n)-grams like those at node N17 (i.e., 25, 0x83 0x86) and node N491 (i.e., 10, 0xd4 0x47) are from the header fields of the HSRP and RIPv1 protocols respectively.

The left half of the tree has more of the TCP and UDP common user protocols, such as the HTTP, SSH (Secure Shell [171]), IPP (Internet Printing Protocol [82]), and DNS (Domain Name System [41]) protocols. Note that in this tree portion, there are clusters for TCP (control) packets. Those are TCP control packets that feature zero-length payloads, such as SYN, FIN, RST, or ACK. We find that these packets are often clustered together, away from their corresponding data-containing packets in the ADHIC trees.

4.3.1 ADHIC Training Time

ADHIC output trees dynamically change over time, adapting themselves to the new traffic being captured. Frequency of changes depend on various factors, but are mainly due to the traffic type and volume, and the ADHIC parameters set before operation (e.g., update period and maturation window). In running our experiments with the parameter values set in Table 3.1, we observed that after one day of parsing the CCSL 1-week long dataset, the output trees start to saturate to a general high level shape. This high level shape remains similar for the rest of the week, with changes in the lower level of the tree depending on the types of traffic changes (i.e., a new network spike).

We also observed that the training time required to bring trees into maturity varies depending on the application types, their change frequency, and their operation times. Our results suggest that ADHIC requires some time to develop enough clusters to effectively segregate protocols when there is a common set of network applications running in the system. In essence, ADHIC requires a training time that covers all traffic activities seen in the network during the day and night times before the general tree shape saturates. This, however, was different with the MD dataset, where ADHIC only took approximately twelve hours to reach a saturated high level tree structure.

4.3.2 Header vs. payload (p, n)-grams

Because ADHIC is not biased in what part of the packet it examines, both header and payload (p, n)-grams can be used to cluster packets. For example, in one of the experiments, IPP packets were segregated by (27, 0x75 0x1b) which is part of the source IP address. Other times (p, n)-grams are discovered deep within the payload. An example of a payload (p, n)-gram is (174, 0x00 0x00) at node N335 in Figure 4.1, which uniquely identifies all NBDGM packets and segregates them at N528 just prior to the global default cluster. This (p, n)-gram refers to the "reserved" and "parameter count" fields within the NBDGM packet structure.

A second example of a payload (p, n)-gram is the (301, 0x00 0x00) (p, n)-gram, which effectively segregates 75% of RIPv1 traffic in another tree for the August dataset. This (p, n)-gram is part of the zero-padding within the RIPv1 "IP address" field. A third example is (61, 0x00 0x0c), which appears in the STP packets' frame check sequence, and usually is found to cluster matching STP packets together. Moreover, both ARP and DTP packets are sometimes found segregated through their trailer patterns. For example, (51, 0x00 0x00) and (82, 0x00 0x00) are common (p, n)-grams found in the trailers of the ARP and DTP packets respectively.

It is important to note that we often find the same protocol getting clustered using different (p, n)-grams in different contexts. For example, we sometimes find HSRP packets segregated by (37, 0xc1 0x00) which uniquely identifies the last byte of the destination port and first byte of the UDP length. In other trees, however, both (48, 0x35 0x00) from the payload and (5, 0x02 0x00) from the header are used to segregate them. The payload (p, n)-gram represents the "hold time" and "priority" field while the header (p, n)-gram indicates the last and first bytes of the destination and source MAC addresses, respectively.

Moreover, we sometimes find that both header and payload (p, n)-grams are used to segregate the same protocol into different clusters in the same tree. For example, in one of the trees, we find EIGRP packets were grouped using the payload (p, n)-gram (64, 0x00 0x0f) (which represents the "hold time" parameter) at one node, and were grouped using (25, 0x29 0x86) at another node in the same tree.

4.3.3 Encrypted packets

ADHIC uses (p, n)-grams from both the packet headers and payloads to cluster network traffic. Packets with encrypted payloads are usually clustered together by ADHIC simply because they are dissimilar to all the other structured traffic. On many occasions, however, ADHIC also separates different types of encrypted traffic using header (p, n)-grams which are neither part of the IP-address or port fields.

For example, IMAPS packets are sometimes separated from others through header

(p, n)-grams such as (22, 0x2c 0x06) (representing the "time-to-live" and "protocol ID" fields) and (54, 0x01 0x01) (representing NOP, NOP in the "options" fields). In another tree example, we find that SSH and IMAPS packets share a common path in the tree until they get separated at the terminal clusters by (54, 0x01 0x01) which matches the "options" field in SSH and the "content type" and "version" fields in IMAPS.

4.4 ADHIC vs. the Reference Classifier

Terminal clusters in ADHIC trees are represented by colored pie charts produced by the port-based reference classifier. Singular clusters (denoted in the tree by rounded grey boxes), however, are those clusters that the reference classifier reports as containing packets of only one protocol. Those clusters are semantically meaningful, in that the clusters represent traffic belonging to a single protocol. However, this does not mean that clusters that are not singular are not semantically meaningful. Protocol classification is simply one aspect of meaning that ADHIC can discover.

By comparing ADHIC's clusters with the reference classifier described in Section 4.2, our results show that ADHIC regularly finds and clusters together semantically meaningful packets. Moreover, we sometimes find that ADHIC might be doing better than the reference classifier itself, especially when packets don't use the right port (e.g. P2P using port 80).

Figure 4.2 shows ADHIC performance by reporting how closely ADHIC clustering was acting like a conventional port-based protocol classifier. The y-axis represents the percentage of packets that were clustered in singular clusters at each 10 minute update period.

Again, ADHIC was not meant to work as a classifier in the first place; however, this figure shows ADHIC's clustering performance using the conventional classification view of the reference classifier. We call this feature "*classification-like*" clustering, and we measure it at each update period by calculating the percentage of packets residing

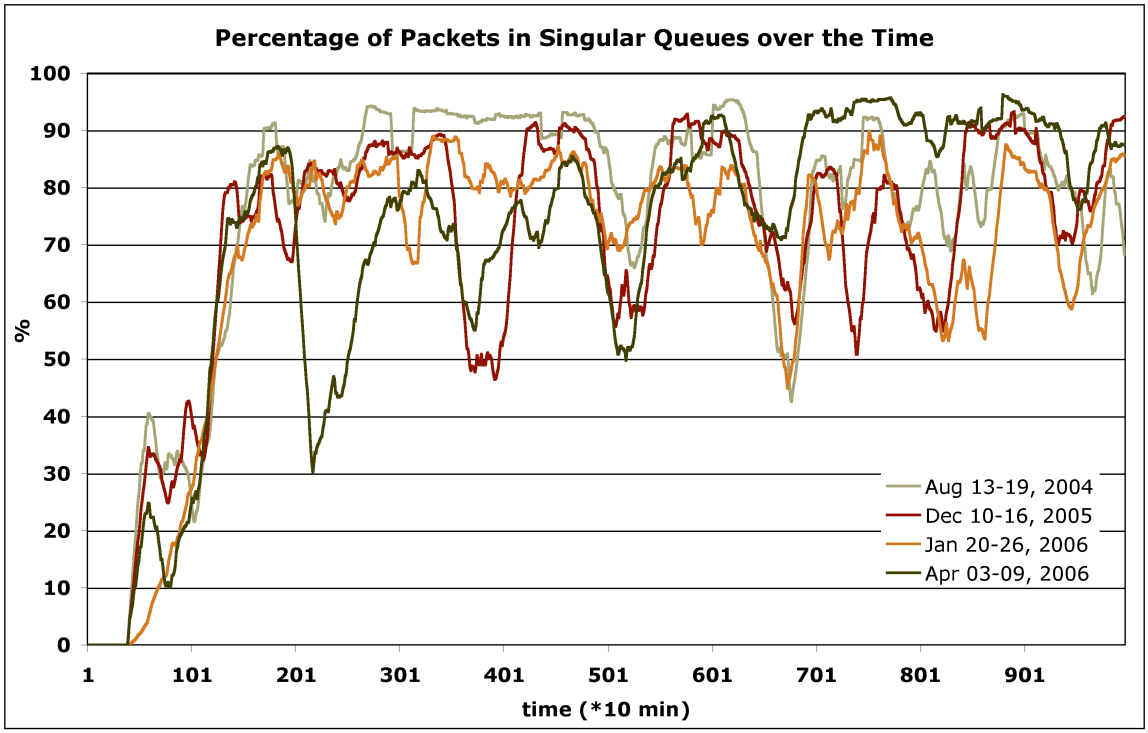


Figure 4.2. Percentage of packets in singular clusters at each update period for the four CCSL datasets.

in singular clusters (n_s) (i.e., clusters with only one protocol type) with respect to the total number of packets seen during that update period (n_t):

$$percentage(\%) = \frac{n_s * 100}{n_t} \tag{4.1}$$

Table 4.2, on the other hand, shows the median and standard deviation (std-dev) calculated over the whole examined time period of the datasets (7 days). While the first column shows the results considering all types of protocols, the second column shows the percentages of the TCP packets only residing in singular clusters with respect to the total number of TCP packets seen in the dataset. The third, fourth, and fifth columns show the results when considering UDP packets, other IP packets (i.e., IP packets that are neither TCP nor UDP), and non-IP packets, respectively.

Dataset	"Classification-like" clustering using **default** settings				
Aug 13-19	All Protocols	TCP	UDP	Other-IP	Non-IP
Average	71.73%	84.21%	91.31%	42.22%	N/A
Median	78.03%	89.76%	94.11%	46.10%	N/A
Std-Dev	18.74%	16.70%	13.71%	12.41%	N/A
Dec 10-16	All Protocols	TCP	UDP	Other-IP	Non-IP
Average	76.43%	84.72%	88.68%	43.98%	79.29%
Median	81.17%	89.26%	91.88%	47.79%	79.87%
Std-Dev	14.87%	13.61%	11.16%	15.37%	9.38%
Jan 20-26	All Protocols	TCP	UDP	Other-IP	Non-IP
Average	72.92%	86.37%	85.79%	84.03%	79.51%
Median	77.57%	93.48%	88.17%	95.94%	79.80%
Std-Dev	16.43%	18.02%	10.84%	27.10%	8.52%
Apr 3-9	All Protocols	TCP	UDP	Other-IP	Non-IP
Average	61.24%	93.57%	90.04%	83.87%	67.35%
Median	64.08%	98.41%	94.53%	98.12%	69.95%
Std-Dev	17.49%	13.15%	13.59%	24.09%	15.52%

Table 4.2. Classification-like clustering: Gauging ADHIC's clustering performance using the conventional classification view of the reference port-based classifier. Median and standard deviation are computed using update period statistics.

ADHIC has relatively better performance with TCP and UDP packets compared to the other-IP and non-IP packets. This is because the other-IP and non-IP clusters may get a few of the TCP or UDP encrypted or streaming packets routed to them, as they match the same (p, n)-grams in the tree path. That is, those clusters are mainly populated by single protocols, but they also have a few packets from other protocols; thus, they are no more counted as singular. Nevertheless, the structural similarities between packets within each of the four groups are still evident enough that ADHIC can usually recognize and use them to separate different protocols from each other.

The results in Table 4.2 exclude the first day of operation because ADHIC requires some time to develop enough clusters to effectively segregate protocols. This period could be thought of as an unbiased short training period, and can be clearly seen in the first 144 update periods in Figure 4.2. The one-day period covers all traffic

activities seen in the network during the day and night times.

Note also that the median percentage of packets in singular clusters is lower when considering all protocols. For example, the median number of packets clustered in singular clusters mostly varies between 64% and 81% compared to the 88% and above for the TCP and UDP classes. There are several reasons for this.

First, the reference classifier is, on occasion, simply wrong. In several instances, particularly in the CCSL August trace, oddly configured application-layer protocols were mixed with flows of the same protocol. For example, we found that ADHIC clusters together same-protocol traffic running on more than one non-standard port number (e.g., HTTP traffic running on other than port 80); however, just because they do not share the same port number, the reference classifier would not consider their cluster as singular. This is a problem not just with this port-based reference classifier, but with any headers-based classifier.

Second, sometimes the reference classifier is not flexible enough. The reference classifier differs from other network header-based classifiers in that it treats TCP control packets (e.g., SYN, FIN, ACK, etc.) as a separate category from the other same-protocol packets with data. This category is added in the reference classifier because ADHIC often segregates control packets as they constitute a semantically meaningful group. However, sometimes ADHIC will simply group all packets of a TCP flow together, lumping control packets in with data packets. Thus the reference classifier does not classify these as singular clusters. This is another reason Table 4.2 under-reports the effectiveness of ADHIC.

Finally, the adaptive behavior of ADHIC is also partly responsible for the difference. ADHIC does not split the clusters containing network bursts as they appear; instead, it assumes that the bursts are transient and assigns the bursts to existing leaf nodes. Only if the burst lasts several update periods does ADHIC attempt to segregate the burst. This change in behavior becomes very clear by comparing the visualization of consecutive output decision trees of ADHIC.

Figure 4.3 shows this effect clearly on the CCSL April dataset. The sine wave

in the graph shows the day/night time throughout the week on the x-axis. It is shifted to show 12:00pm at the highest sine wave peaks, and 12:00am at the lowest peaks. It is clear how at each spike's peak (bottom), the reference classifier shows a sudden degradation of ADHIC's performance (top). Note how spikes are classified as non-singular and lower the performance because it is high volume.

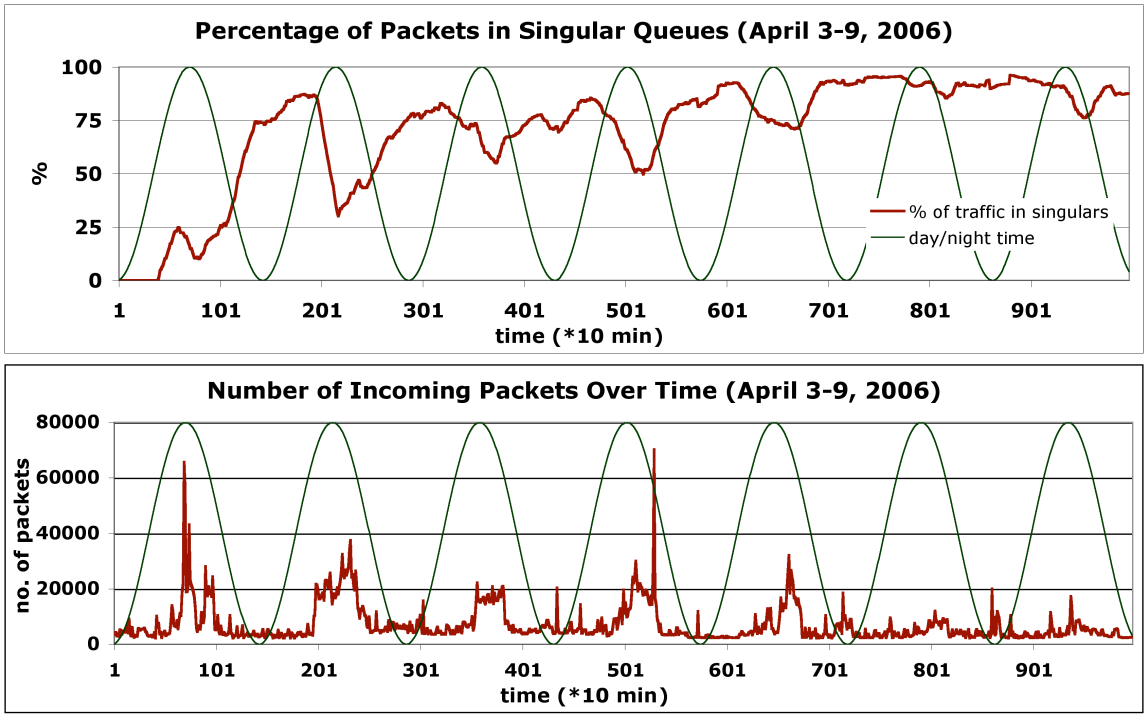


Figure 4.3. Percentage of packets in singular clusters at each update period for the CCSL April dataset (top), in contrast with the number of packets seen in the trace for the same time period (bottom).

On a side note, we use median rather than average as a representative of effectiveness. Consider, for example, Figure 4.2, which shows ADHIC clustering performance on all the four CCSL traces. A spike occurs in the CCSL August trace close to update period number 550, which dramatically reduces the percentage of packets at singular clusters. When the traffic spike subsides, the singular cluster packet percentage recovers. This proper function of ADHIC decreases the apparent performance when compared with the reference classifier.

These reasons explain the apparently "low" statistics in Table 4.2. While Table 4.2 and Figure 4.2 are presently the best metrics for evaluating ADHIC, they under-represent its effectiveness because of the defects inherent in the traditional classifiers.

4.5 Testing ADHIC with Other Networks

In this section, we test ADHIC against two independent full-capture datasets: MD and RMC. As introduced in Section 4.1.1, the MD dataset represents two months of captures from a private small-size sales company in Maryland, USA. The network is comprised of over a dozen windows-based machines with about ten users. The machines also include two file servers, and a Voice-over-IP (VoIP) phone system. The dataset is over 8 GB in size and is mostly populated by web (i.e., HTTP and HTTPs), email (POP), and media streaming (RTP) traffic. The dataset consists of incoming traffic to the company captured during a two-month period from Oct 30, 2007 until Dec 26, 2007. Table 4.1 gives protocol classification and content statistics for the first week of this dataset.

The RMC dataset, on the other hand, represents a one-hour capture from the uplink of the Royal Military College (RMC) in Kingston, Ontario, Canada. The size of this dataset is about 12 GB, where the college network has over 1000 users. For this dataset, we were not able to inspect packets manually to generate our detailed results. Therefore, we do not use this dataset in other experiments. However, we had access to the ADHIC output trees which gave us fair evidence that ADHIC would still be useful in clustering traffic into semantically equivalent classes even if it is run against enterprise network datasets.

While ADHIC performed qualitatively similarly in these environments as it did in the CCSL lab, the trees generated by ADHIC capture a number of interesting structural features of these network environments.

Figure 4.4 shows that the MD network produced a more fine protocol clustering (a deeper tree) than the CCSL trees, due in part to the longer observation time. This

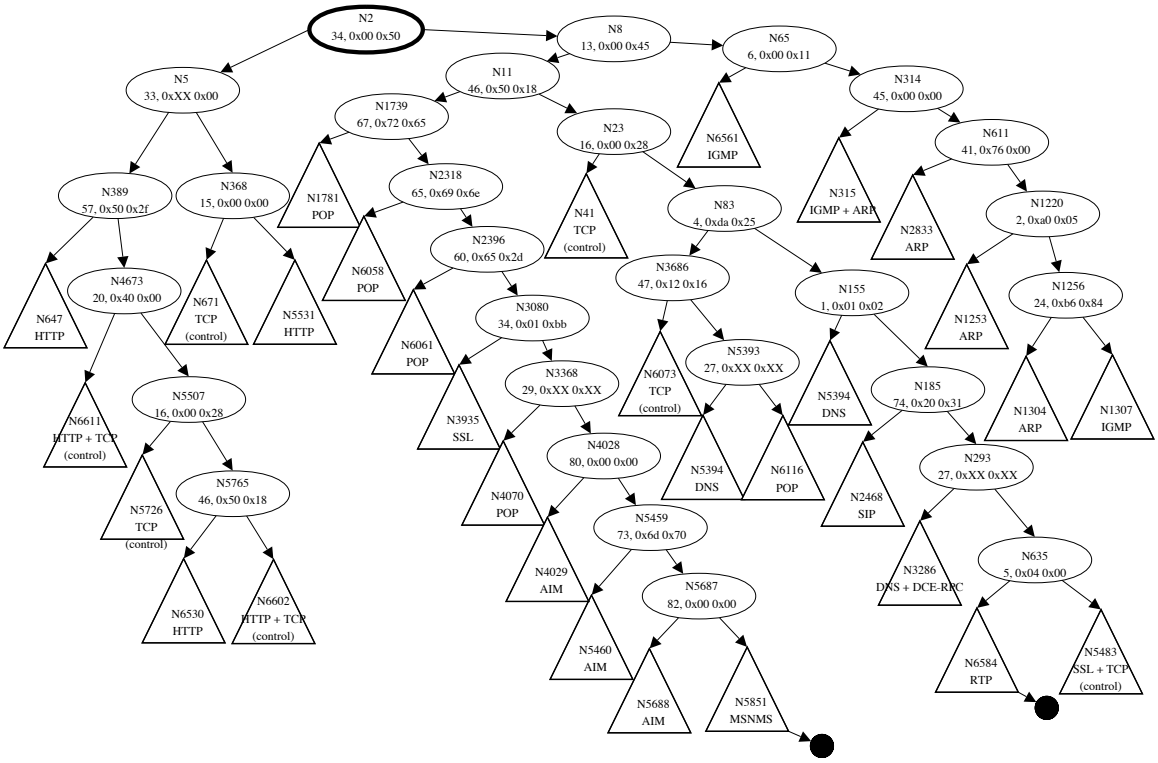


Figure 4.4. (Best viewed electronically to allow enlargement) Annotated decision tree produced by ADHIC from a snapshot taken from the Maryland experiment.

MD annotated tree snapshot in Figure 4.4 was taken after processing three weeks of dataset as opposed to one week in the case of the CCSL snapshot in Figure 4.1.

Several protocols were clustered in this network that were not available in the CCSL runs. For example, AIM (AOL Instant Messenger [2]), RTP (Real-time Transport Protocol [150]), SIP (Session Initiation Protocol [165]), POP (Post Office Protocol [138]), and DNS (Domain Name System [41]) are all appropriately clustered here. Note that several protocols were classified using payload (p, n)-grams. POP clusters were branched at offsets like 60, 65, and 67 which represent the response description field in the POP packets. AIM clusters were identified at offsets like 80 and 82 which are part of the AIM buddylist service field. SIP packets were clustered together using a payload (p, n)-gram at offset 74.

Figure 4.5 shows a very high level overview of a decision tree taken from the RMC

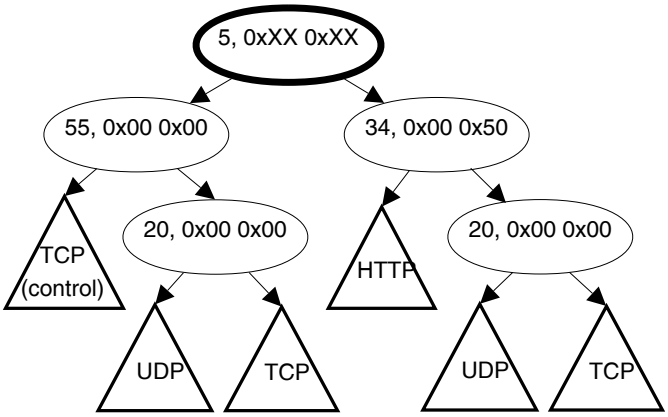


Figure 4.5. A high level simplified decision tree produced by ADHIC from a snapshot taken from the RMC experiment.

network. Here, ADHIC has chosen (5, 0xXX 0xXX)[1] to do the first tree split and form the root (p, n)-gram node. This (p, n)-gram belongs to the last byte of the destination MAC address field and the first byte of the source MAC address field. Therefore, all packets going on the left hand side of the tree belong to traffic destined for a specific router. The right hand side is comprised of all the other packets including traffic broadcasted and not specifically destined to this router.

A significant amount of HTTP traffic is segregated on the right half of the tree using the header (p, n)-gram (34, 0x00 0x50), which represents the source port 80.The payload (p, n)-gram (55, 0x00 0x00), on the other hand, is part of the "urgent pointer" field which is usually set to zeros if the URG is not set. The two other identical (p, n)-grams (20, 0x00 0x00) (which represent IP flags and fragment offset fields) are acting as discriminators between UDP (matching) and TCP (non-matching) packets on both sides of the tree.

Unfortunately, our sample was sufficiently short (just one hour long) that we could not see how ADHIC may proceed if it were to continue with more packets. However, our results with the RMC dataset re-assure us about ADHIC's main strategy of generating its binary decision tree. In particular, it starts from a high level segregation

[1]Due to the strict privacy policy agreement we made with RMC, we had to anonymize all MAC and IP addresses.

in the root node (using a destination MAC address (p, n)-gram), to a lower level where it segregates between TCP and UDP. Based on our experience with the MD and CCSL datasets, we expect that the next split level in the tree will be more fine grained and will be based on protocol types, followed by flow types.

5 Monitoring Abnormal Traffic Using ADHIC

This chapter examines the ability to detect evasive and/or abnormal traffic using the (p, n)-grams approach. It first examines how well ADHIC can segregate protocols even if header information becomes useless (Section 5.1). The chapter shows that despite the packet's missing header portion, ADHIC can still find useful (p, n)-grams in the payload that will cluster traffic similar to the results produced with full packet information.

The chapter then examines the ability of ADHIC to detect P2P traffic even if it is obfuscated to run under port 80 (Section 5.2). The chapter shows that ADHIC rarely uses ports to cluster traffic; rather, it clusters based on common patterns. Thus, it segregates the bulk of the P2P traffic by not finding the same patterns it finds with other traffic.

Finally, the chapter shows how ADHIC can detect certain abnormal behaviors in simulated traffic using the (p, n)-grams approach (Section 5.3). More experiments with the synthetic DARPA old datasets are discussed. Basically, the IDEVAL dataset is normally used to evaluate network intrusion detection performance; however, many researchers have observed that the synthetic background traffic in this dataset deviates from normal background traffic in a few key ways, specifically regularities, predictability, traffic distribution and lack of crud. We hypothesized that these deviations would cause ADHIC to generate abnormal traffic trees. As we now explain, this was indeed the case.

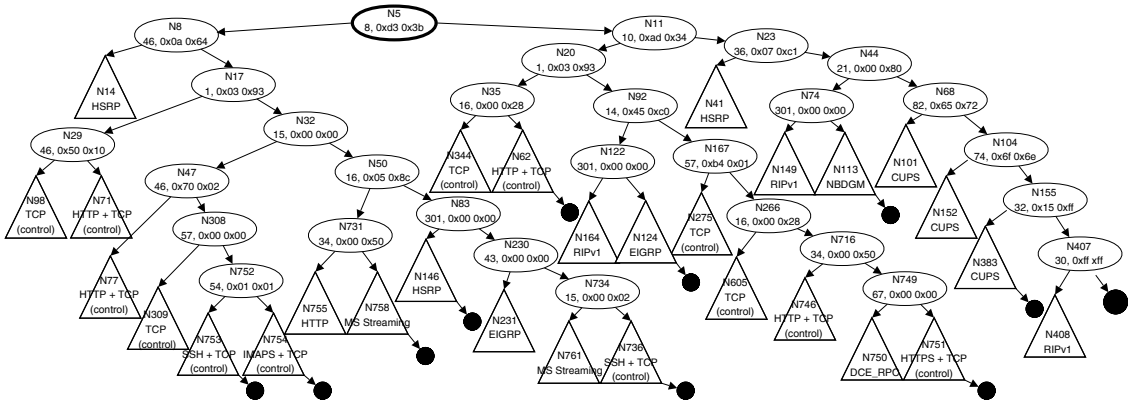
5.1 Clustering without header information

We continue our investigation of evasive traffic by examining how well ADHIC can segregate protocols even if header information becomes useless. We configured NetADHICT to ignore the first 38 bytes of each packet. This excludes the 14 bytes of Ethernet header, the IP header, and port information for both TCP and UDP. As a side effect, it also excludes payload information for packets that are neither UDP nor TCP. We then tested ADHIC against the four CCSL traces again, collected statistics, and examined the decision tree at the same snapshot in time.

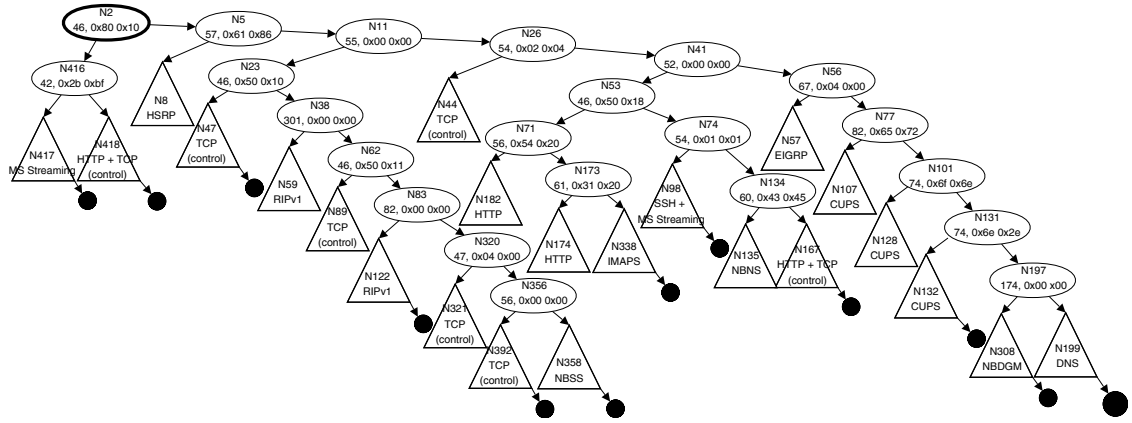
Dataset	"Classification-like" clustering using **no header** information				
Aug 13-19	All Protocols	TCP	UDP	Other-IP	Non-IP
Average	82.06%	94.97%	94.25%	89.14%	N/A
Median	91.31%	97.49%	95.90%	92.20%	N/A
Std-Dev	20.77%	8.18%	9.41%	15.15%	N/A
Dec 10-16	All Protocols	TCP	UDP	Other-IP	Non-IP
Average	65.85%	86.89%	88.34%	89.81%	66.76%
Median	70.46%	91.72%	91.81%	97.38%	68.19%
Std-Dev	17.05%	14.99%	11.63%	17.70%	10.82%
Jan 20-26	All Protocols	TCP	UDP	Other-IP	Non-IP
Average	65.67%	91.65%	86.59%	92.94%	67.94%
Median	67.40%	97.92%	89.06%	95.98%	68.08%
Std-Dev	17.65%	15.58%	13.16%	11.76%	12.50%
Apr 3-9	All Protocols	TCP	UDP	Other-IP	Non-IP
Average	68.03%	93.18%	87.64%	97.10%	56.35%
Median	73.24%	97.49%	91.72%	98.85%	56.85%
Std-Dev	18.40%	10.87%	11.67%	7.49%	21.65%

Table 5.1. Percentage of packet statistics after ignoring header information

Similar to Table 4.2, Table 5.1 shows the new statistics for the four CCSL network traces when Ethernet header, IP header, and both source and destination port numbers are skipped during the generation of (p, n)-grams. Once again, the table reports how much (in terms of packets percentage) ADHIC clustering was performing like a conventional protocol classifier—one that *did* have access to packet headers.



(a) ADHIC tree with default paramters



(b) ADHIC tree without looking at the headers

Figure 5.1. (Best viewed electronically to allow enlargement) Annotated decision trees produced by ADHIC using default parameters (5.1(a)) and without looking at the packets' header portion (5.1(b)). Both trees were produced from the same snapshot taken from the CCSL August trace.

We expected ADHIC would perform badly when not allowed to use most of the header information because it usually uses both the header and payload in building the decision tree. However, it pleasantly surprised us by generating trees that were qualitatively similar to the trees produced using all the packet information (see Figure 5.1). ADHIC's ability to cluster is occasionally degraded, especially with TCP-based streams, but it still finds a large amount of structure within many protocols.

By comparing the results in Tables 4.2 and 5.1, one can see how ADHIC sometimes performs better when no header information is given during (p, n)-gram generation. This is mainly due to the random sequence in which ADHIC chooses its (p, n)-grams. From a pool of many (p, n)-gram candidates, the payload (p, n)-gram that ADHIC picks (because it happens to be the first one it finds to match the similarity spread -see Chapter 3) might work better with packets seen later in the traffic, resulting in better trees and improved segregation results.

The largest difference is in ADHIC's inability to segregate encrypted traffic when headers are restricted. In the CCSL August trace, all encrypted traffic is routed to a single cluster. This contrasts with the earlier experiments that allowed ADHIC to examine header information, in which each encrypted protocol was routed to a distinct cluster. This is because, without header information, it becomes almost impossible to distinguish between types of encrypted packets—there is no structural information available.

5.2 Clustering P2P traffic

We have performed several experiments with ADHIC against P2P traffic. Some of these experiments used the BitTorrent [16] client application to download relatively large files (over 500 MB) to our lab machines. BitTorrent is perhaps among the most evasive of the popular P2P protocols. Linux binaries, free public compressed movies, and live video streaming are examples of what the experiments included. Traffic pertaining to these experiments was then individually merged with the CCSL Jan

dataset for testing.

While some P2P captured traffic featured unique source port numbers, many others were running on constantly changing port numbers. Based on our experience with ADHIC and no header traffic, our hypothesis was that ADHIC would be able to segregate P2P traffic into clusters separate from other traffic even without consistent port information. As we will show, our results were consistent with this hypothesis.

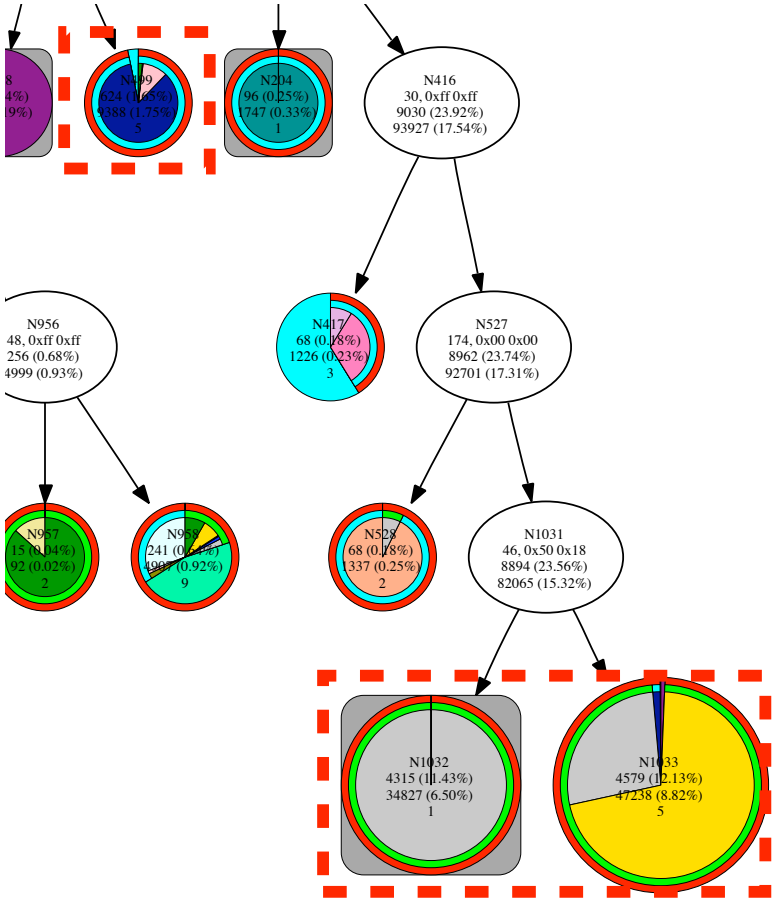


Figure 5.2. CCSL January tree snapshot with the presence of P2P traffic (best viewed electronically). All BitTorrent traffic went to the highlighted clusters: N499, N1032, and N1033.The dark blue cluster (N499) represents the BitTorrent's UDP packets, while the gray and yellow clusters (i.e., N1032 and N1033) represent the TCP data and TCP control packets respectively.

Figure 5.2 shows an example of how BitTorrent traffic was clustered together using

ADHIC after it was merged with the CCSL January trace. In particular, one cluster (N499) managed to segregate most of the UDP tracker related data packets through (50, 0x00 0x00)—a (p, n)-gram that is not in the IP-header portion; all the other related TCP packets (whether data or control packets) got routed to the tree's global default cluster at N1033 and its adjacent cluster at N1032, as they did not match any of the (p, n)-grams higher in the tree.

Note that the reference classifier recognized UDP tracker cluster and labeled it with its special dark blue color while it could not recognize the TCP data packets (unknown port numbers) and labeled them as unknown with the grey color. Due to the huge amount of P2P traffic, further splitting of the default cluster occurs later in the trace; however, the BitTorrent traffic was always segregated on its own or in the global default cluster along with a few other unusual packets.

One question we had was whether BitTorrent would be clustered differently if it were run over a standard port. To test this, we obfuscated all BitTorrent packets by changing their port number to 80 and re-ran our experiment. We found that each packet was clustered exactly as before in the tree, however, the reference classifier wrongly labeled the packets as if they were HTTP.

This performance can be explained by two observations. One is that ADHIC rarely uses ports to cluster traffic. But more significantly, ADHIC was able to segregate the bulk of the BitTorrent traffic not by characterizing it directly, but by characterizing other network traffic as having patterns that were absent in the BitTorrent traffic. Thus, so long as most well-behaved traffic can be appropriately clustered, evasive protocols can be identified simply by their lack of structural resemblance to other traffic.

5.3 Synthetic Background Traffic: DARPA Dataset

The Lincoln Laboratory Intrusion Detection Evaluation (IDEVAL) datasets [103] (referenced hereafter as LL datasets) are entirely synthetic datasets that have been

primarily used to evaluate intrusion detection and other network security systems. These datasets are useful because they contain no proprietary or confidential information for any real users. The LL datasets have been brought to the attention of the network security and machine learning communities, and have been used in a number of well-cited papers [111, 139, 187, 197].

However, the LL datasets were criticized many times for having a number of artifacts that make them less useful for evaluation [115]. For example, the normal traffic is too uniform: the machines behave in a too similar manner, and there is a distinct lack of malformed background traffic, or "crud". Mahoney et al. [110] also reported several other inconsistencies with real traffic captures, notably regularities regarding TCP SYN packets and severe predictability in source addresses and packet header fields such as the time-to-live. Because of these features, attacks in the LL datasets are much easier to detect than in regular network traffic.

In this section, we test ADHIC's performance on the LL dataset. The goal is to check whether ADHIC can detect these known abnormalities in this LL synthetic traffic, as well as finding other abnormal behavior that distinguishes synthetic traffic from real traffic. Obviously, our work of analyzing the LL dataset with NetADHICT has been prefaced with the past research identified in [115] and [110]. It is with this known artificiality in hand that we are inspecting the results found from using NetADHICT, noting discrepancies from past research with real datasets.

Note that any conventional anomaly or pattern analysis tools may be able to detect what was found by examining the trees created by NetADHICT, however, NetADHICT reveals many representations all at once. Our analysis in this section demonstrates how we have arrived at some interesting observations for the LL dataset. Before we describe our testing of the LL dataset with ADHIC, let's show the traffic distribution of LL dataset compared to our CCSL and MD real traffic datasets.

5.3.1 Traffic Distribution of LL Dataset

In 1998, and again in 1999, the Lincoln Laboratory at MIT, under contract from DARPA, developed a series of datasets in order to test the correctness and robustness of existing Intrusion Detection Systems (IDS) [104]. These datasets were created by using host computers connected together with a traffic generator to model a small US Air Force base of limited personnel, connected to the Internet. Network traffic and host audit information was recorded during the experiments. Three weeks of training data and two weeks of test data were released, as well as a list of all attacks included in these synthetic datasets.

We use three weeks of the LL dataset training data captured from the sniffer on the inside of the network. Each week is comprised of data for Monday through Friday from approximately 8am to 6am the following day. Some traces were cut short due to system crashes during the data capture [103].

Table 5.2 compares the protocol composition statistics for three dataset traces: 1-week MD trace (Oct 31-Nov6, 2007); CCSL trace (Apr3-Apr10, 2006); and LL trace (Mar8-Mar13, 1999). We note that our traffic capture also contains attacks, just as the second week of the LL dataset contains labelled attacks. However, our network is significantly smaller than the LL virtual machines, although its network topology is similar, and it generates traffic at a similar scale.

5.3.2 Testing the LL Dataset with ADHIC

We performed two rounds of testing with ADHIC. The first involved running ADHIC on each of the $\leq$ 22 hour traces. ADHIC was run with its standard parameters: 10 minute update periods and an 18 update period maturation window (180 minutes). This resulted in trees with a maximum depth of 5 or 6, and 18 to 26 terminal clusters.

For the second test, we merged the datasets together to form three week-long traces. The timestamps in the traces were shifted to remove the two hour empty period between each trace (from approximately 6am to 8am) and then merged into a

Protocol	MD 2007 Oct 31-Nov 6	CCSL 2006 Apr 3-Apr 10	LL 1999 Mar 8-Mar 13
IPv4	**88.34 %**	**86.66 %**	**98.96 %**
TCP	**72.66 %**	**53.73 %**	**89.68 %**
TCP Unknown	0.08 %	0.62 %	0.05 %
MS WBT/MS RDP	0.00 %	0.14 %	0.00 %
IPP	0.00 %	**6.88 %**	0.00 %
IMAPS	0.00 %	**2.35 %**	0.00 %
HTTPS	**14.81 %**	**1.75 %**	0.00 %
SSH	0.00 %	**3.37 %**	**4.73 %**
MS Streaming/RTSP	0.00 %	**1.38 %**	0.00 %
MSNMS	0.26 %	0.05 %	0.00 %
XMPP	0.02 %	0.02 %	0.00 %
TCP Sophos	0.00 %	0.08 %	0.00 %
TCP No Payload	**14.10 %**	**26.69 %**	**42.22 %**
RTSP	0.00 %	0.03 %	0.00 %
NBSS	0.00 %	0.00 %	0.03 %
IRC	0.00 %	0.00 %	0.04 %
TELNET	0.00 %	0.00 %	**26.54 %**
FTP	0.00 %	0.07 %	**1.48 %**
SMTP	0.03 %	0.37 %	**3.45 %**
CVS	0.00 %	0.16 %	0.00 %
POP	**7. 64 %**	0.07 %	0.03 %
HTTP	**34.98 %**	**9.67 %**	**11.11 %**
AIM	0.73 %	0.00 %	0.00 %
UDP	**14.97 %**	**28.93 %**	**8.95 %**
UDP Unknown	0.02 %	0.01 %	0.06 %
DNS	0.80 %	0.95 %	**7.25 %**
CUPS	0.00 %	**1.81 %**	0.00 %
WHO	0.00 %	0.09 %	0.00 %
MP3-Stream	0.00 %	**3.51 %**	0.00 %
NBDGM	0.00 %	0.88 %	0.02 %
DCE_RPC	0.11 %	0.22 %	0.01 %
SIP	**1.80 %**	0.00 %	0.00 %
NBNS	0.00 %	**2.49 %**	0.17 %
RIPv1	0.00 %	0.59 %	0.54 %
HSRP	0.00 %	**18.28 %**	0.00 %
DHCP	0.01 %	0.02 %	0.00 %
SNMP	0.00 %	0.00 %	0.05 %
NTP	0.10 %	0.07 %	0.84 %
RTP	**12.11 %**	0.00 %	0.00 %
ICMP	0.01 %	0.35 %	0.33 %
EIGRP	0.00 %	**3.66 %**	0.00 %
IGMP	0.69 %	0.00 %	0.00 %
ARP	**11.66 %**	**12.29 %**	0.42 %
ETHER (old)	0.00 %	**1.05 %**	0.09 %
Total no of Packets	**1521232**	**7075868**	**7275137**
Avg Packt Size in B	**746.6**	**253.37**	**205.75**
Total size in GB	**1.1**	**1.8**	**1.6**

Table 5.2. Protocol classification and content statistics for the second week of the LL dataset compared to two 1-week datasets from the CCSL and MD datasets. Only protocols with percentage $\geq 0.02\%$ are shown, and with percentage $\geq 0.1\%$ are highlighted.

single large trace file. ADHIC was again run with standard parameters. This second run provided a longer term view of the evolution of the tree, providing more time for the tree's structure to stabilize. The trees for the week long traces were much larger: they had a maximum depth of 10 or 11 and contained 60 to 75 terminal clusters.

We divide our analysis of the LL dataset into two main parts. First, we examine the temporal distribution of traffic as shown by the evolution of ADHIC's trees (Section 5.3.3). We then examine anomalies in the distributions of (p, n)-grams in Section 5.3.4. We finally discuss the summary of our analysis in Section 5.3.5.

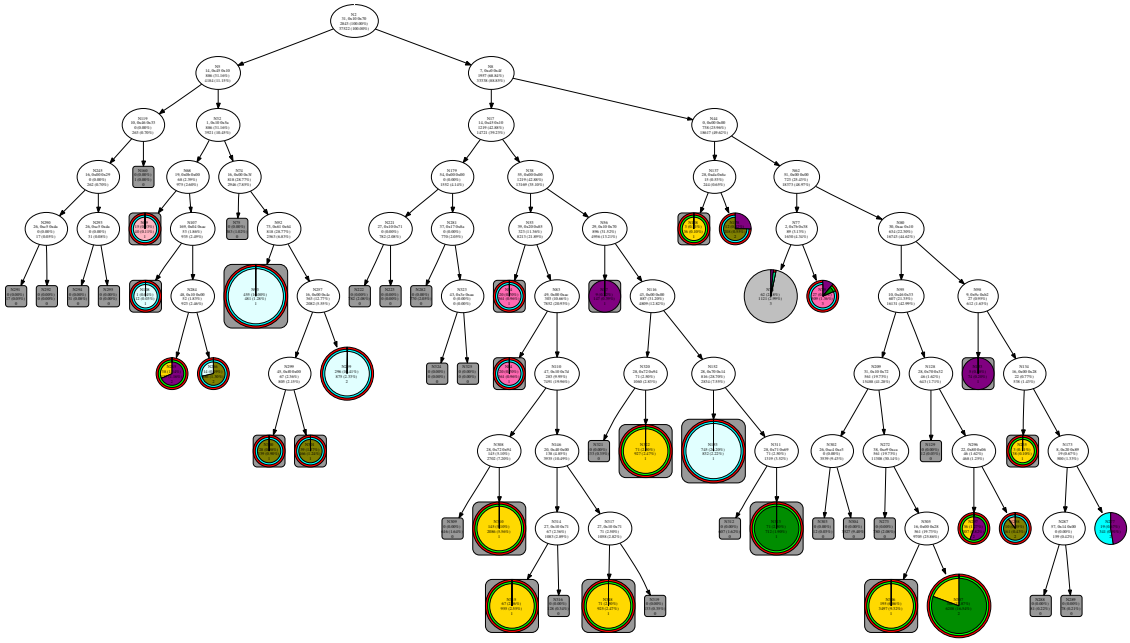
5.3.3 Temporal Distribution of Traffic

Network traffic is made up of many bursts: connections are established, used and then closed and each one is different in some way. Most of these connections share similar features and underlying consistency. ADHIC works by extracting this similarity, using (p, n)-grams, and clustering like traffic together. The LL data appears to be missing some of this consistency. We make this observation because we see a "strobe"-like effect in regards to the trees ADHIC creates.

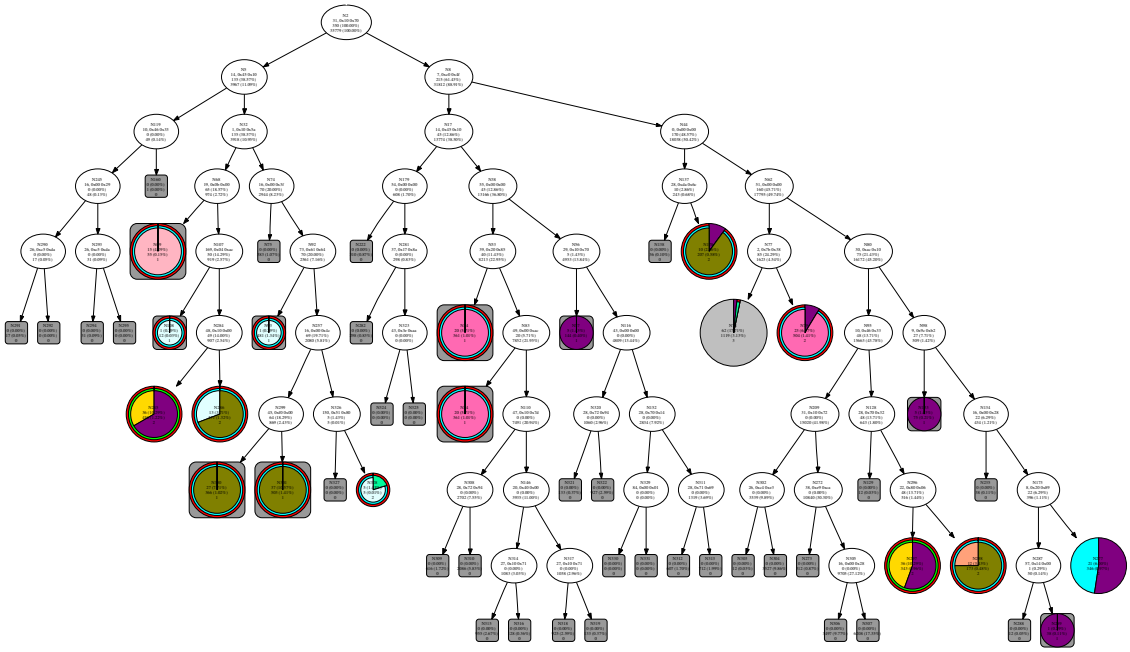
Consider Figures 5.3(a) and 5.3(b), which show two consecutive "snapshots" of the ADHIC tree from the second week of the LL dataset, ten minutes (one update period) apart. Note how the trees are almost completely different. Many clusters are created from a particular burst of traffic, then left empty when the burst ceases. New bursts cause new nodes to be created, and they then also quickly disappear.

Such large changes in tree structure over a short period of time is something we never see in regular network traffic. Some clusters may grow or shrink; overall, though, the structure of the trees remains consistent. The ever-changing trees shown here from the LL dataset lead us to the conclusion that the traffic does not resemble normal traffic.

We further characterize the bursty nature of the LL datasets in Figure 5.4. Here we have compared them to the CCSL April dataset. Note the LL graphs contain data which has been modified to close the two-hour gap between traces; the overlaid sine

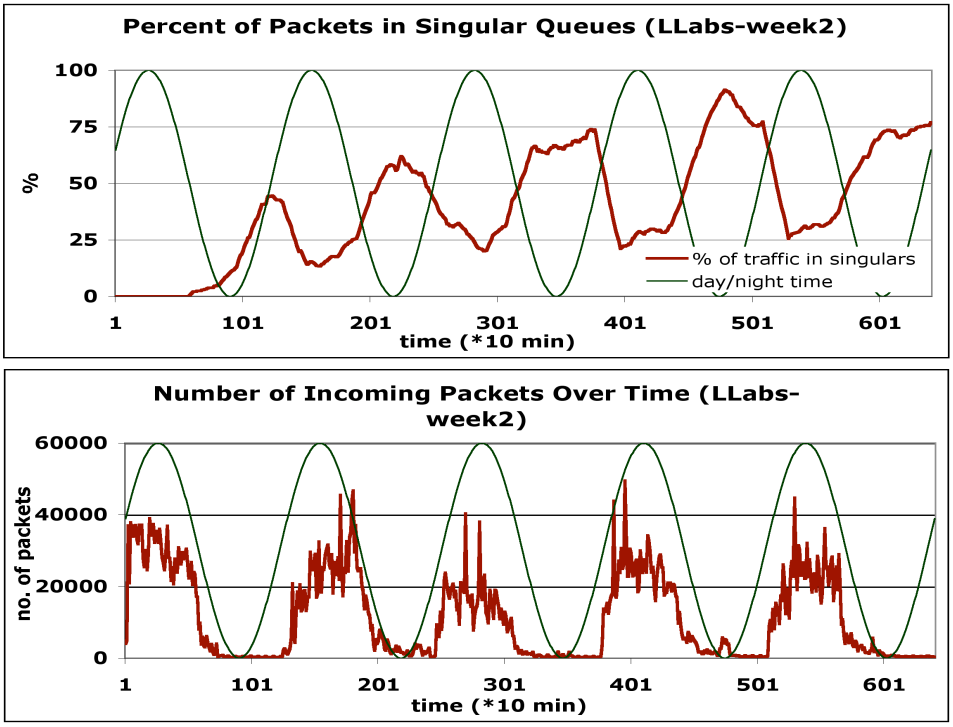


(a) LL, week 2 dataset at 240^{th} update period

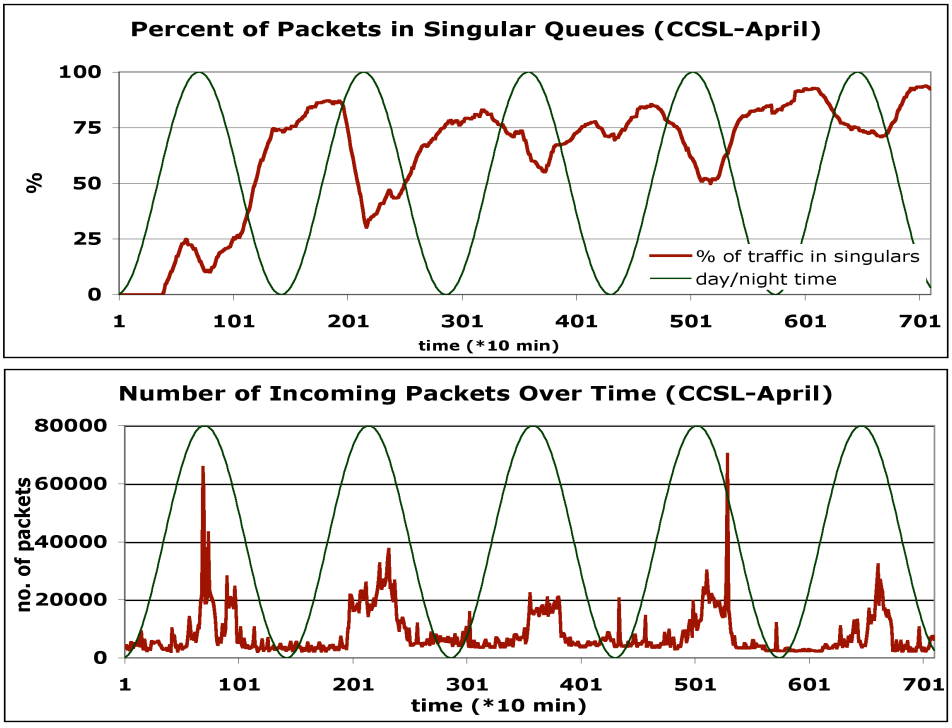


(b) LL, week 2 dataset at 241^{st} update period

Figure 5.3. (Best viewed in color and electronically to allow enlargement) The synthetic LL data lacks consistency, which causes erratic, strobing trees, with clusters appearing and disappearing. Note the large number of empty clusters (small gray squares) in Figure 5.3(b).



(a) The LL dataset



(b) CCSL dataset

Figure 5.4. Temporal analysis of packet distribution over one week periods in the LL dataset and our lab (CCSL).

wave has been adjusted to account for this. The top of the crest of the wave in both figures denotes noon, and the bottom of the valley denotes midnight.

Surges of traffic are more pronounced in the LL dataset, fitting much more closely to the passage of daytime (see Figure 5.4(a)). The nighttime hours contain far less traffic than our lab captures, providing at times only a few hundred packets over ten-minute intervals—something remarkable for a network with thousands of machines. In contrast, our lab (Figure 5.4(b)), with many fewer (but real) machines, has a steady baseline of thousands of packets in the same sized intervals.

5.3.4 Distributions of (p, n)-grams

The breakdown of traffic in the LL dataset compared to our lab's capture in Table 5.2 does not show any irregularities. However, if we look at the traffic at shorter time periods (10 minutes), we can see that some protocols are over-populating the traffic. This is exemplified by Figure 5.5 with the large amount of DNS traffic over a 10 minute time period.

The graph in Figure 5.5 shows a single cluster—before any splits have occurred—dominated by approximately 85% DNS traffic. This is the first 40 minutes of the second week of the dataset. Little explanation is available for such a large amount of DNS traffic effectively flooding the network. Observing LL dataset's output trees makes it easy to detect such a behavior.

Figure 5.6 looks at offsets of the 1000 most frequent (p, n)-grams in three periods of the CCSL and LL datasets. While the percentages of (p, n)-grams throughout the three different periods of CCSL show consistency between day and night, the percentages of the LL datasets do not. Moreover, the consistency difference is also visible when examining the 10-minute period against the 3-hour period it is part of. The discrepancy with the LL dataset can be clearly seen among the day (3-hour and 10-minute) and night (3-hour) time periods with payload (i.e., $p > 53$) and TCP header (i.e., $37 > p > 54$) (p, n)-grams. Note that this is not the case with the very consistent real traffic also in the figure.

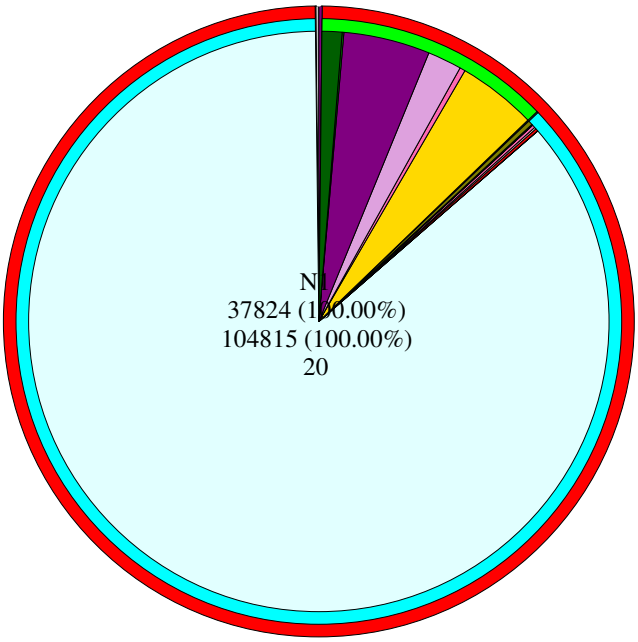


Figure 5.5. Example of high volumes of DNS traffic in the LL dataset. DNS is illustrated as the large, light blue wedge.

5.3.5 Summary

Similar to the "crud" discussed by Mahoney et al. and McHugh, ADHIC leaves a portion of the analyzed traffic unclassified, found in the furthest right leaf of the tree. In our analysis of the LL datasets, we have noticed a lower amount of this unclassified traffic. Most traffic is successfully classified through the ADHICT port-based reference classifier; however, particular protocols are unknown. Compared to the traces from our lab, there is a much lower quantity of unclassified traffic. While the lack of unclassified traffic does point to a lack of "crud," it is also potentially due to the greater variety of network protocols currently in use today, when compared to the 1999 simulation.

To summarize, ADHIC quickly revealed a number of unusual traffic patterns in the LL dataset, illustrating shortcomings in its simulation of normal network traffic. Some of these patterns, such as the unusually uniform distribution of packets [110], have been previously noted. Other observations (in particular, the extreme temporal

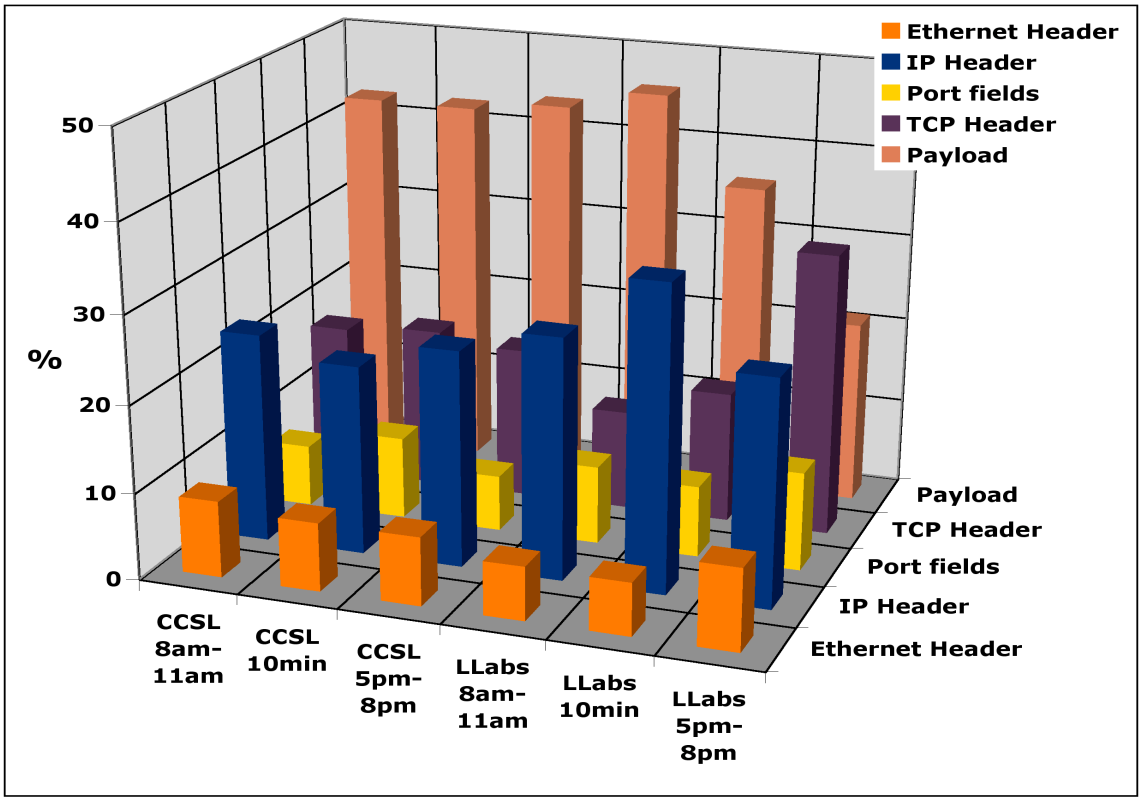


Figure 5.6. Comparison of the CCSL and LL traffic captures over three periods (two 3-hour periods and one 10-minute period), separated by packet type. The x-axis represents the two 3-hour datasets (morning and evening) of both datasets along with the last 10 minutes of the 3-hour morning time period of each. The y-axis describes at what packet offset p those (p, n)-grams are found (Ethernet header, IP header, port fields, TCP header and payload). The z-axis (height) of the graph denotes the percentage of (p, n)-grams contained within each packet offset range.

variation) we believe are novel.

What is notable with this analysis is the ease with which we could identify the unusual properties of the datasets. The temporal variation manifests as a remarkably dynamic tree that "strobes" in a way that virtually never happens with traces gathered from production networks. A modest amount of subsequent analysis then revealed the other characteristics, such as the lack of "crud," identified by past researchers.

The LL dataset was designed to provide a high-level view of network data, one that reveals large-scale patterns that may or may not follow the bounds of IP addresses and ports. While such functionality is potentially valuable when monitoring production networks, here we show that it is also a potentially valuable tool for the researcher, one that complements standard packet aggregate counts and manual packet and flow-level inspection. While there are many patterns that it does not readily capture (such as flow counts), we believe the LL dataset's ability to unify high-level and low-level network traffic views make it a powerful addition to the network researcher's toolbox.

The problem of creating network datasets for research purposes is a difficult one. Synthetic and anonymized datasets are essential resources; however, artifacts in them can lead to conclusions that do not hold on production networks. We believe lightweight clustering strategies such as those employed on the LL dataset hold the potential for proactively identifying data artifacts in network data—captured, synthetic, and anonymized—so they may be factored into experimental design. Such work should increase the quality of research results and reduce the need for later critiques.

6 (p, n)-gram Characteristics in Network Traffic

The previous chapters have shown evidence of how ADHIC can segregate different kinds of network traffic without having any explicit models of traffic. In this chapter we begin exploring the characteristics of (p, n)-grams that enable applications like ADHIC, both for the purpose of better understanding under what circumstances we can expect ADHIC to be an effective clustering algorithm, and to provide a foundation for other potential uses for (p, n)-grams.

The first part of this chapter explores the characteristics of (p, n)-grams that we have observed in our formal and informal experiments. Most of the key observations are explored in later chapters; others, however, are more a reflection of our own experiences and should be seen as potentially needing further validation. The overall framework, though, is important to understand in order to place the later chapters in the appropriate context.

We then introduce a novel use of Shannon entropy as a metric of content similarity between network packets. In Chapter 8 we use this measure to empirically calculate entropy models of individual network protocols with different design structures. This shows the mapping between design structures of individual protocols and the corresponding content similarities and (p, n)-gram distributions. Then in Chapter 9 we utilize this entropy definition in building a conceptual model that generalizes and explains the (p, n)-gram characteristic distributions observed in network traffic.

6.1 (p, n)-gram Characteristics

As introduced in Section 1.3, there are two main characteristics of (p, n)-grams that give them the ability to be used for traffic characterization applications. The first characteristic is their ability to capture semantically-relevant structural differences between packets of different protocols. This counts for their fingerprinting functionality, and it is composed of two parts:

a) Frequent (p, n)-grams calculated in a network traffic reflect semantic design structures of the corresponding network protocols. We call these frequent (p, n)-grams "*structural*" (p, n)-grams (briefly introduced in Section 1.2).

b) Protocols with different design structures feature different frequency and offset distributions of structural (p, n)-grams. This allows structural (p, n)-grams to uniquely fingerprint structured protocols in network traffic.

The second characteristic of (p, n)-grams is their rapidly-dropping-off frequency distribution behavior. This characteristic accounts for their fingerprinting efficiency as it assures a representation of network protocols with a *small* number of structural (p, n)-grams. Consequently, protocol design structures can be represented by a small set of (p, n)-grams in the corresponding network traffic.

Both of the characteristics of (p, n)-grams are inherited from the encapsulated IP packet structure design. Moreover, finding structural (p, n)-grams through their relative high frequency can be done automatically without *a priori* knowledge about the involved protocol specifications.

We start with discussing the efficiency characteristic first, and then we discuss the two parts of the functionality characteristic. We finally discuss how these (p, n)-gram characteristics can be leveraged to meet the traffic characterization application requirements.

6.1.1 Rapidly-Dropping-Off Frequency Distribution

Figure 6.1 shows the general common "frequency distribution" behavior we observe of (p, n)-grams in network traffic. This distribution represents the relationship between *frequency* and *rank* (ordinal index) of (p, n)-grams in network traffic. Frequency distribution is directly related to the impact of *content similarities* found between network packets.

Typically, the frequency distribution of (p, n)-grams shows a rapidly-dropping-off curve with a "*power-law-like*" shape similar to that of Zipf's law [203], where the first few (p, n)-grams have very high packet matching frequencies, whereas the remaining majority have low frequencies.

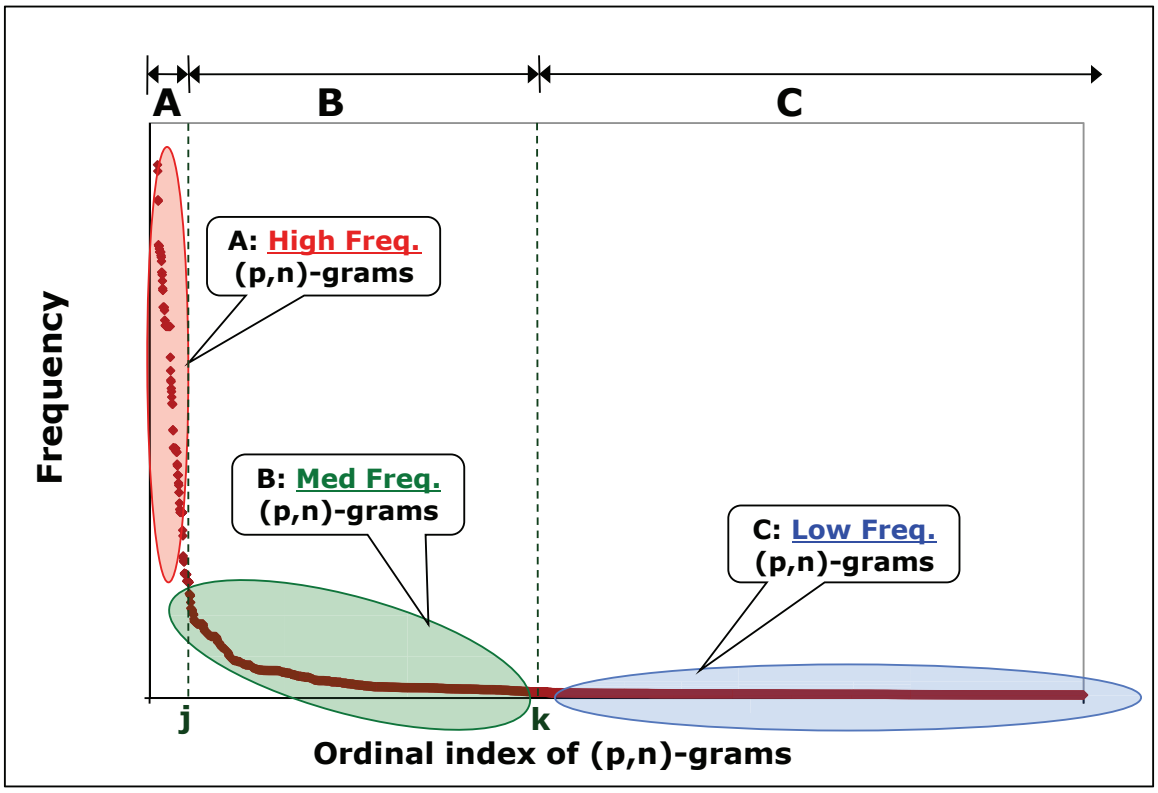


Figure 6.1. (Best viewed in color) Frequency distribution of (p, n)-grams on a normal graph scale. Three different frequency-level regions of (p, n)-grams can be recognized in the graph, namely: Regions **A**, **B**, and **C**.

The scatter graph shows three regions of distinguishable (p, n)-gram frequency

levels, namely: Region **A**, **B**, and **C**. Note that, for clarity purposes, the very long tail that extends Region **C** to the right only partially appears on the graph.

Region **A** is a very small one that covers the period [1,j], and contains the first portion of the curve. This portion declines very fast and contains the most frequent (p, n)-grams in the inspected network traffic. Contrarily, Region **C** is the longest one covering the period [k,m] where m is the total number of distinct (p, n)-grams calculated in the network trace.

The portion of Region **C** displayed on the graph constitutes the beginning of a long tail of a power-law-like curve, where the total number m may go to several millions[1].

Extending the x-axis in Figure 6.1 to include all calculated (p, n)-grams in Region **C** shows a long straight line (long-tail) that moves slowly down approaching 1 on the y-coordinate. This long tail of Region **C** contains (p, n)-grams with the lowest percentage of packet matching frequency.

Finally, Region **B** covers the period [j,k], and contains the rounded part of the curve that connects Regions **A** and **C**. Region **B** contains (p, n)-grams with medium packet matching percentages.

Note that calculating (p, n)-grams for different network traces might give different values of j and k on the x-coordinate, depending on the size of the examined trace and the nature of network protocols it is composed of. Our experiments, however, show that similar period sizes on the x-coordinate are usually observed when inspecting different trace sizes of the same network traffic (Chapter 7 provides statistical details).

Re-plotting Figure 6.1 on a log-log scale shows a power-law-like behavior that we *conjecture* to follow "*Zipf's law*" (See Appendix A.2). That is, a straight power regression line with a slope (power exponent) close to a negative unity. Figure 6.2 shows this behavior on a log-log-scale scatter graph.

The empirical analysis presented in Chapter 7 uses multiple traces from two independent datasets to provide statistical evidence of this general frequency distribution

[1]This number can go up to the size of the (p, n)-gram domain space, that is: $(packetSize - n + 1) \times 2^{8n}$. Therefore, if we use $n = 2$, and assume the common maximum Ethernet packet size of 1,500, we get a domain space of: $(1,500 - 2 + 1) \times 2^{8*2} = 98,238,464$ (p, n)-grams.

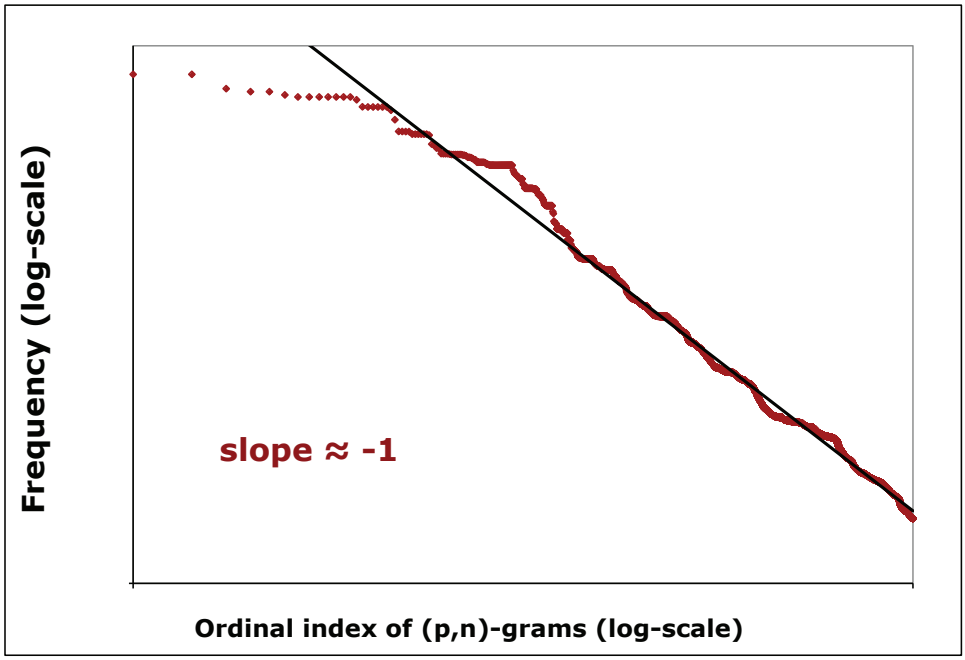


Figure 6.2. Frequency distribution of (p, n)-grams on a log-log scale. Notice how the distribution behavior is close to a straight line with a slope of negative unity and a good fit around the regression line.

behavior. We describe this behavior as a rapidly-dropping-off frequency distribution that appears as a straight line on a log-log scale graph with a slope close to -1 and slightly varying from trace to trace. This behavior is what we refer to as "*power-law-like*" distribution.

But the question remains: how can this rapidly-dropping-off distribution behavior of (p, n)-grams be seen as a good fit for the traffic characterization problem? In essence, our research shows that (p, n)-grams that can be used to characterize and fingerprint network traffic appear in the first two regions of Figure 6.1: Regions **A** and **B**. This finding is further discussed in Section 6.1.2, and later in Chapter 8.

Therefore, the rapidly-dropping-off frequency distribution behavior assures that only a small set of distinguishable (p, n)-grams is required to fingerprint network traffic. This gives space efficiency and threshold setting advantage for (p, n)-gram-based traffic characterization applications. Section 6.1.3 discusses this issue in more detail.

Finally, it is worth noting that the power-law-like behavior shown in Figure 6.2 reflects Regions **A** and **B** and a portion of Region **C**. Extending the graph to include the long-tail (p, n)-grams in Region **C** usually shows deviation from the power-law-like distribution with more flattened slopes that are not very close to unity.

6.1.2 Capturing Differences in Protocol Structural Designs

The other main characteristic of (p, n)-grams in network traffic is their ability to capture structural design differences between packets of different protocols. In principle, this characteristic accounts for the fingerprinting functionality of (p, n)-grams, and it is due to the semantic meanings of frequent (p, n)-grams. We discuss this characteristic in two steps. The first step discusses the semantic meanings of frequent (p, n)-grams, while the second one shows how different protocols feature different distribution behaviors of (p, n)-grams.

Semantic Meaning of Frequent (p, n)-grams

One of the advantages that using the offset p gives to (p, n)-grams over n-grams is the additional semantic meaning within network packets. That is, knowing the specific field in which an n-gram appears in a packet gives another dimension to its substring content. Consider for example how finding the two (p, n)-grams (54, 0x47 0x45) and (55, 0x45 0x54) in a packet implies that these two bytes are consecutive. Moreover, knowing that they are part of the application header field (offsets 54 and 55) may further suggest that they belong to an HTTP GET request packet (the hexadecimals 0x47, 0x45, and 0x54 represent 'G', 'E', and 'T' respectively).

Another advantage of p is the additional domain space that it adds to (p, n)-grams. This gives (p, n)-grams more richness while describing network packets. That is, the number of all possible n-gram values in a packet is equal to 2^{8n}, whereas, the number of all possible (p, n)-grams is equal to $(packetSize - n + 1) \times 2^{8n}$.

Re-plotting the three regions in Figure 6.1, with offsets of (p, n)-grams on the

y-axis, gives us their "offset distribution" in network traffic. Figure 6.3 shows the general behavior of this distribution, which gives the relationship between *offset* and *rank* (ordinal index) of (p, n)-grams. Offset distribution is directly related to the impact of *structural designs* in network packets[2].

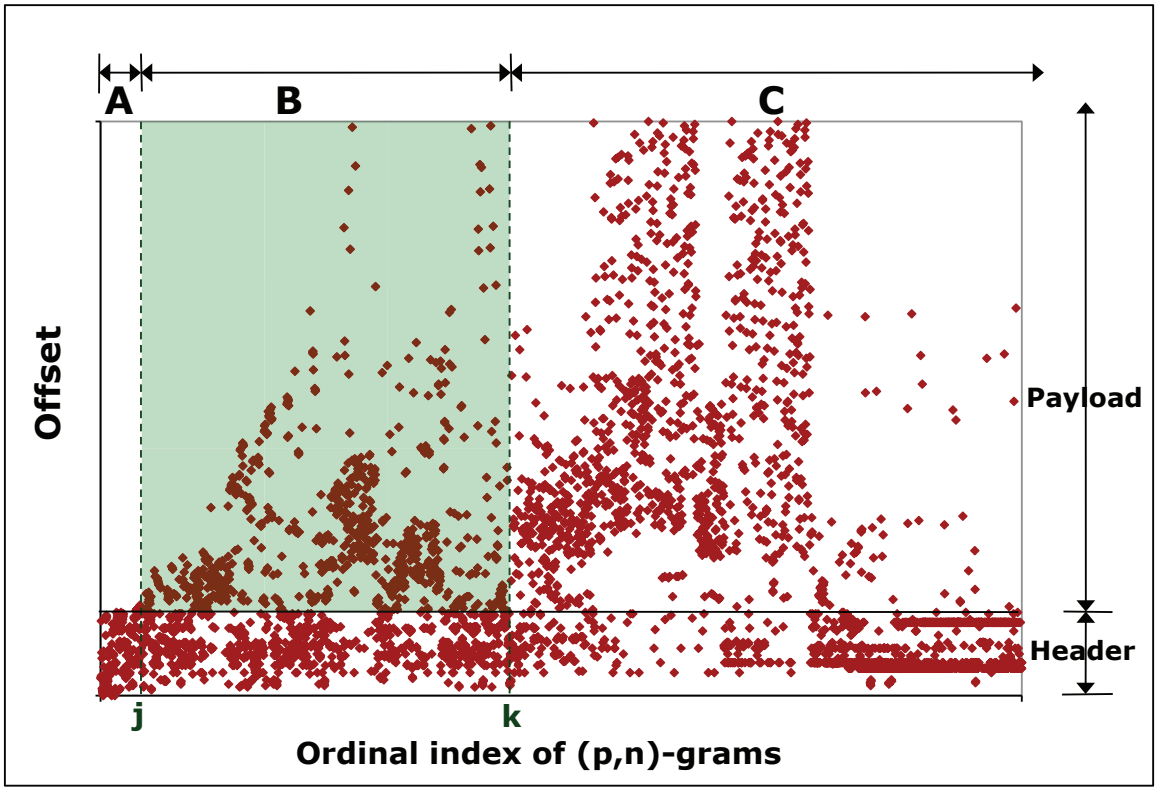


Figure 6.3. Offset distribution (OD) of (p, n)-grams. The shaded area represents frequent payload (p, n)-grams that are application-dependent.

In principle, (p, n)-grams in Regions **A** and **B**, in Figure 6.3, reflect semantic design structures in the corresponding network protocols. These (p, n)-grams either represent application-specific packet structures or represent packet header fields revealing common information about network topology and traffic behavior. Again, we call these (p, n)-grams "*structural*" (p, n)-grams based on their semantic meaning.

[2]Note that offsets are counted starting from the beginning of the packet's Data-link header (e.g., Ethernet header) with offset 0 being the first byte. Appendix B.2 provides a brief overview of the IP packet structure.

Focusing more on (p, n)-grams in Region **A** (i.e., period [1,j]), shows that they are all located in the packets' header fields. Those (p, n)-grams mainly represent common network information (e.g. IP and MAC addresses, ports, etc.) or traffic behavior parameters (e.g. QoS parameters, total length, TimeToLive, etc.). In typical IP network traffic, many of these packet header fields have similar structures and common values across different protocols. Therefore, (p, n)-grams in this region may not always be the best to distinguish network protocols from each other.

Region **B** (i.e., period [j,k]), is where frequent payload (p, n)-grams start to appear in addition to the other frequent header (p, n)-grams. The payload (p, n)-grams (in the shaded area) are mostly application-dependent or protocol-specific (p, n)-grams representing protocol structural fields. This representation allows them to be used to distinguish between different protocols. An example of what payload (p, n)-grams might be pointing to is the sequence "ipp://", which is located in the payload URI field within the CUPS packets.

Semantic (p, n)-grams, in Regions **A** and **B**, can be simply found through their relatively high frequency in network traffic. Our ADHIC clustering algorithm (discussed in Chapter 3) relies on automatically finding these structural (p, n)-grams in the two regions, and using them as discriminators to classify network traffic. Note that the small size of both regions explains why protocol structural designs in network traffic can be represented by a small set of frequent (p, n)-grams.

On the other hand, (p, n)-grams in the third region **C** (i.e., period [k,m]) represents the majority *infrequent* (p, n)-grams that either belong to unstructured payload contents or to header fields with infrequent contents such as checksum. Those (p, n)-grams are very infrequent and, thus, can't be used to represent common protocol patterns.

It is important to note that frequent (p, n)-grams in both regions, **A** and **B**, reflect content similarities between network packets. The similarity level differs from offset to offset depending on the packet field. For example, higher content similarity is usually found between packets at the header's "IP version" field than at any another

deep offset within the packets' payload. Section 6.2 introduces a novel use of Shannon entropy [156] as a metric to measure content similarity between network packets at fixed offsets. We use this metric to explain content similarities between network packets and their impact on (p, n)-gram distributions.

Different Protocols have Different Distribution Behaviors

Different network protocols have different levels of content similarity between their packets. Our experiments show that calculating frequency and offset distributions of (p, n)-grams generally gives slightly different behaviors depending on the nature of the inspected network traffic. Behavior differences are mainly due to the types and volumes of protocols constituting the inspected network traffic.

In principle, (p, n)-grams represent common patterns in the various packet fields of network traffic. Therefore, structural differences between network packets cause the representational behavior of (p, n)-grams to be protocol dependent. Obviously, this has its direct impact on their frequency and offset distributions. Our experiments show that distribution differences between network traces appear when their traffic consists of different protocol types and/or volumes. These differences are more evident when the distributions are calculated for single-protocol traces with different protocol types (e.g., HTTP vs. DNS).

To visualize these differences, Figure 6.4 shows the (p, n)-gram frequency and offset distributions of two single-protocol traces, where each trace represents a different protocol. The differences between the two protocols are evident in terms of their frequency distribution slopes (0.82 vs. 1.69) and in terms of their offset distribution scatter graphs (i.e., patterns and areas of concentration). Chapter 8 provides empirical examples that show how different single-protocol traffic produce different frequency and offset distribution behaviors depending on their protocol types and application modes of operation.

Our experiments suggest that the frequency and offset distributions of (p, n)-grams for any network trace are mainly influenced by the following parameters:

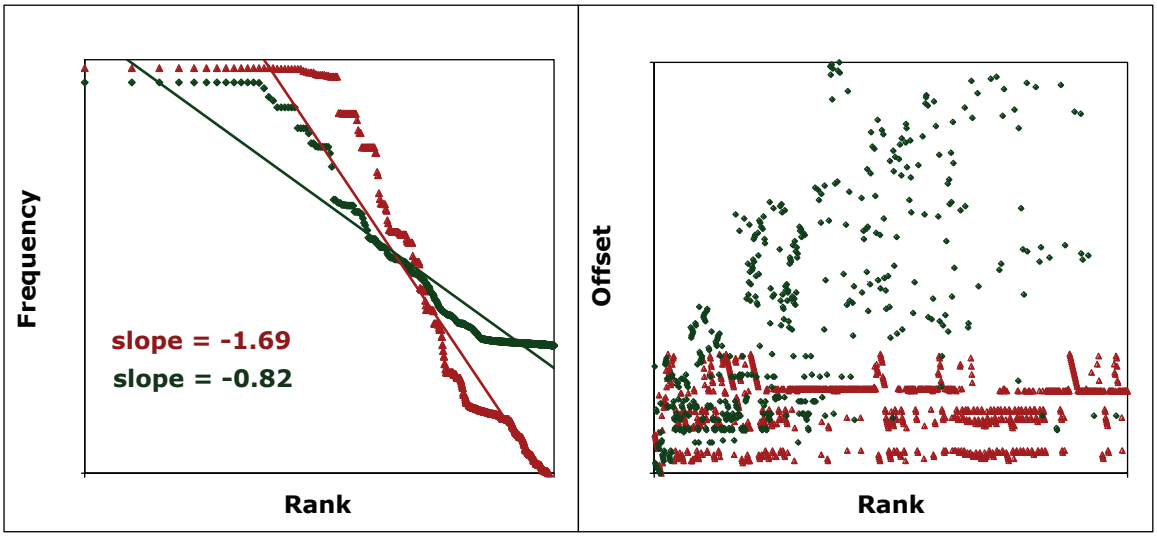


Figure 6.4. Protocol-dependent (ARP in red triangles and HTTP in green diamonds) (p, n)-grams frequency and offset distributions.

1) *Network topology*: Topology of the inspected network directly impacts (p, n)-grams that represent network-mapped fields, such as IP and MAC addresses.

2) *Protocol types and volumes*: Depending on their specific packet design structures, individual protocols constituting the network trace have their impact on the overall distributions of (p, n)-grams. Moreover, protocols with relatively high volumes have more chance to affect the overall (p, n)-grams distributions.

3) *Mode of operation*: Different operation modes result in different (p, n)-gram distribution behaviors for the same protocol. For example, using the HTTP protocol to surf text-based Websites generates a different distribution behavior of (p, n)-grams than that generated by using the HTTP protocol to download binary files. Similarly, both distributions are different than that generated by using the HTTP protocol for HTTP tunneling (i.e., using an HTTP packet as a wrapper to encapsulate packets of other protocols [17]).

Monitoring the impact of these parameters on (p, n)-grams distribution is what we rely on to build our network protocol classification and security applications (see Section 6.1.3). For example, ADHIC can differentiate between TCP and UDP, and

between IPP and HTTP protocols based on their differences in (p, n)-gram distribution behaviors. By the same technique, ADHIC can even differentiate between HTTP Web surfing traffic and HTTP-like P2P traffic. These classification and security monitoring functionalities of ADHIC were discussed in Chapters 4 and 5.

6.1.3 Mapping (p, n)-gram Characteristics with Applications

One of the main research objectives we discussed in Chapter 1 has been to find network packet features that can be calculated without *a priori* knowledge of the involved protocols and can be used to efficiently characterize network traffic. In this section, we briefly discuss how the main (p, n)-gram characteristics presented so far can be leveraged to meet these application requirements.

First, the semantic meanings of frequent (p, n)-grams give them adequate representation of the different network protocol packets. This representation coupled with the ability to capture protocol structural differences are the two (p, n)-grams' functionality characteristics that we use to fingerprint network protocols and distinguish between different protocols in network traffic. Not only can (p, n)-grams reflect the type and size of the main running protocols, but they can also reflect their mode of operation. We further employ this functionality to implement our traffic clustering and security monitoring applications.

Second, using (p, n)-grams in pattern matching gives a *time* efficiency advantage over the regular n-gram pattern matching technique. That is, (p, n)-grams require only a sublinear time in the size of the packet for packet matching as opposed to the regular linear time required in looking at every byte in a packet using n-gram pattern matching. In addition, the rapidly-dropping-off distribution with a power-law-like behavior gives an additional *space* efficiency advantage of (p, n)-grams. This distribution behavior implies that the structural (p, n)-grams are easily distinguishable from the rest in the long tail due to their unique high frequency. It also implies that only a small set of structural (p, n)-grams are required for the traffic characterization applications.

Third, (p, n)-gram characteristics are naturally inherited from the hierarchical and encapsulated IP packet design. This gives (p, n)-grams an applicability advantage to be used to characterize network traffic without *a priori* knowledge of the specific protocol packet structures.

Thus, we leverage (p, n)-grams and their characteristic distributions to build a traffic characterization framework of three applications. In the first application, traffic clustering (discussed in Chapters 3 and 4), we use the ability to automatically find structural (p, n)-grams in the complex network traffic to build an effective traffic clustering system, where structural (p, n)-grams can be used as a proximity measure of semantic similarity between network packets. This clustering application allows traffic to be classified into equivalence classes that closely approximate standard measures of network traffic.

In the second application (traffic monitoring, discussed in Chapter 5), we watch over time for temporal changes to the (p, n)-gram distribution behavior. This allows us to signal instances of deviation from the expected normal behavior of the network traffic. For example, ADHIC can use sudden changes in the (p, n)-grams distribution behavior to indicate abnormal behavior, such as, a single-protocol surge (e.g., worm, flash crowd, etc.) dominating the network traffic.

In the third application, protocol fingerprinting (will be discussed in Chapter 8), we rely on the differences found between network protocols in order to fingerprint their different types. Those differences are mainly in terms of 1) the sets of representative structural (p, n)-grams, 2) their frequency distributions, and 3) their offset distributions, to fingerprint the different protocol and traffic types.

Finally, the implied deep packet inspection is a common privacy concern when dealing with content-based network traffic characterization techniques. However, in this (p, n)-gram-based approach, structural (p, n)-grams are only calculated through their high frequency, and they only constitute short packet strings that represent protocol structures. We, therefore, *conjecture* that inferring protocol structures using (p, n)-grams would not reveal private information or raise privacy concerns except for

some highly frequent network information, such as server IP-addresses. We discuss this further in Chapter 10.

6.2 Entropy as a Metric to Measure Content Similarity

This section introduces a novel use of Shannon entropy [156] to describe content similarity in network traffic. The main purpose of this is to define an abstract model that allows us to generalize and conceptually explain the two main characteristics of (p, n)-grams, namely: 1) their rapidly-dropping-off frequency distribution, and 2) their ability to capture protocol design structures.

In particular, we introduce using Shannon entropy as a metric to measure the level of content similarity at fixed offsets in network packets. In addition to presenting our empirical results, Chapters 8 and 9 utilize this entropy definition in building a conceptual model that explains the two characteristics of (p, n)-grams.

6.2.1 Entropy Model Definition

Shannon entropy is commonly used to measure randomness in an event. When applied to an n-byte field (data source), Shannon entropy can be defined to give the number of bits required to encode data based on its content value repetitions. For example, an entropy of 6 bits calculated on a 1-byte field means that all the different values seen in that field can be encoded, on average, using 6 bits only (i.e., with the remaining 2 bits being redundant). This definition of entropy gives an indication of how repeated (and hence similar) the values that appear at the corresponding field are.

We use this entropy definition to express variances in packet contents at fixed fields. That is, for all the inspected packets in a sample, we check the different values found at a specific field and their repetitions. The higher the entropy, the higher the level of content variances or dissimilarity in that field (i.e., less repetition). Using this definition, we observe that protocols with very similar packet contents, such as

broadcast and multicast protocols, feature low entropy at most of their packet fields. This is to be compared with encrypted protocols whose packet entropy is high at most of their packet's payload fields.

Our application of entropy definition on network packets may apply to any 1-byte-long field at a fixed offset p {p: 1, 2, ..., packet-size - 1} in the network packet (i.e., (p, n)-grams with $n = 1$). In particular, we define a random variable X, for each possible byte value at offset p, with $2^8 = 256$ outcomes $\{x_i$: 0x00, 0x01, ..., 0xFF$\}$, and then, Shannon entropy is defined as:

$$H(X) = -\sum_{i=1}^{2^8} pr(x_i) * \log_2(pr(x_i)) \tag{6.1}$$

Where:

a) $pr(x_i)$ is the probability that X is in the state x_i, and

b) $pr(x_i) * \log_2(pr(x_i))$ is defined as 0 if $pr(x_i) = 0$.

In this definition, $H(X)$ is bounded by two values: $0 \leq H(X) \leq 8$. That is, if we have a sample size of 2^8 (p, n)-grams, where:

1. all (p, n)-grams are *identical*, then $pr(x_i) = 1$, and $\log_2(pr(x_i)) = 0$. Thus, $H(X) = -\sum_{i=1}^{2^8} 1 * 0 = 0$ *bits*.
2. all (p, n)-grams are *different*, then $pr(x_i) = \frac{1}{2^8}$, and $\log_2(pr(x_i)) = -8$. Thus, $H(X) = -\sum_{i=1}^{2^8} \frac{1}{2^8} * (-8) = -\frac{1}{2^8} * 2^8 * (-8) = 8$ *bits*.

Therefore, an entropy of 0 at offset p means that the same byte value appears in all packets at that offset, whereas an entropy of k $(0 \leq k \leq 8)$ means that there is an average number of 2^k distinct byte values that appear in the packets at the same offset.

Since entropy is expressed in a logarithmic scale, a linear difference between two entropy levels (e.g., 6 and 7) implies an exponential difference in the corresponding

(p, n)-grams similarity level (i.e., a total of 2^6 vs. 2^7 distinct (p, n)-grams in the same field).

6.2.2 Applying Entropy Model to Network Traffic

The purpose of defining Shannon entropy on network packets is to use it as a metric to measure content similarities between network packets at fixed offsets. Figure 6.5 shows this novel use of Shannon entropy on a scatter graph with entropy calculated on a random network traffic trace. Points in the graph represent entropies calculated at each packet offset (i.e., from offset 0 to offset 1499).

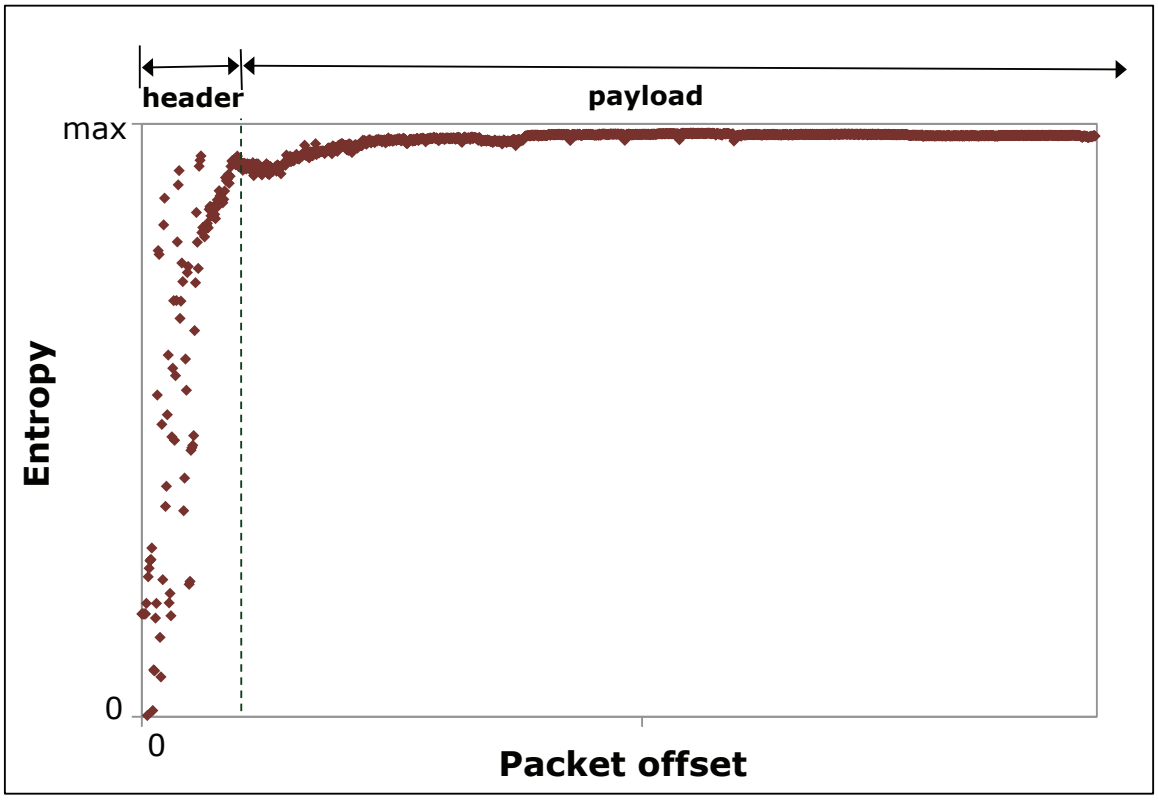


Figure 6.5. Shannon Entropy calculated at each 1-byte-long packet offset. Note that the maximum entropy on the y-axis is equal to 8 bits, whereas the maximum packet offset for Ethernet packets is equal to $1500 - 1 = 1499$.

A closer look at the graph, taking into consideration the two packet portions

(header and payload), shows that packet fields in the header portion possess lower entropies than those in the payload. This can be simply explained through the types of contents that exist in each portion. In essence, packet header fields usually contain network parameters with common values, such as IP addresses, protocol ID, TTL, etc. This is in comparison with the payload fields that usually contain infrequent data. Note, however, that due to the protocol design structures within the payload portion, some of the payload fields possess lower entropies than others.

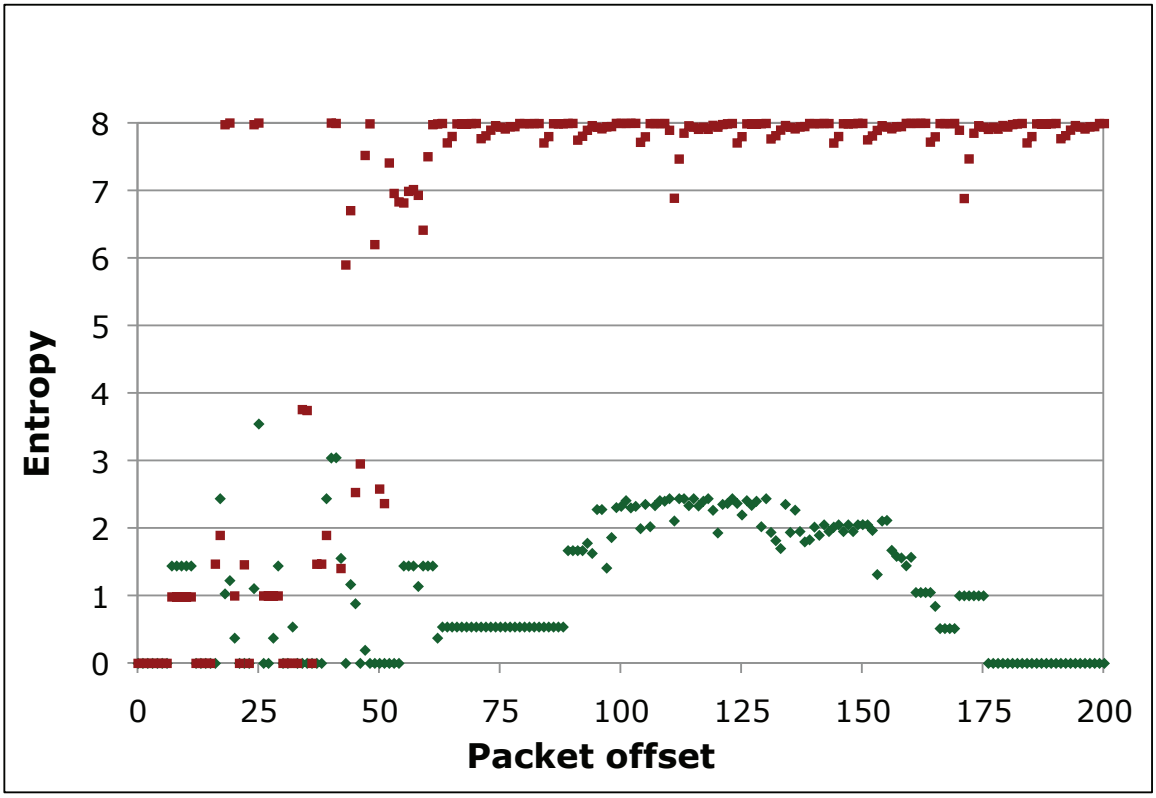


Figure 6.6. Shannon Entropy calculated for every 1-byte offset for two different protocols (CUPS in green diamonds, and MP3 streaming in red squares). Notice how each scatter graph represents the structural designs in the corresponding protocol.

Therefore, when the entropy graph is calculated for packets of one protocol type, the structural designs of the protocol become more evident in the payload portion. Figure 6.6 shows a scatter graph with entropies calculated on two traces, each representing one protocol type. The first graph (green diamonds) represents a protocol

(CUPS) that features high content similarity at both the header and payload portions of the packet. This indicates that there are more structured fields in the payload portion than just data streaming fields.

On the other hand, the second graph (red squares) represents a data streaming protocol (MP3) where the majority of the payload portion features high entropy, except for a few fields with relatively lower entropy. Those fields with relatively lower entropies represent protocol design structures within the packet payload. Chapter 8 discusses this in more detail and gives examples of empirically calculated entropy for several protocol types.

Chapters 7 and 8 utilize this entropy definition to build a conceptual model that we use to generalize and explain the two main characteristics of (p, n)-grams in network traffic, along with their functionality and efficiency features for traffic characterization applications. The model specifically uses statistics of Internet traffic as a test-case to generalize the empirically observed characteristic distribution behaviors of (p, n)-grams in the context of the current design and implementation of IP protocols.

In particular, Chapter 7 shows that packet fields with low entropy levels are mainly found in the packets' header portion as well as the short structural fields of the packets' payload portion. On the contrary, packet fields with high entropy levels are mainly found in the packets' long payload portion. Comparing the size of the two types in an average-size Internet packet shows that low entropy fields constitute a much smaller portion of the total packet size than the other ones.

On the other hand, Chapter 8 shows that when entropy is calculated on different protocols, the structural design differences between protocols appear in the shape of fields (at various offsets, with different sizes) featuring relatively low entropies. Therefore, calculating frequent (p, n)-grams will capture those relatively low entropy fields which at the same time represent their protocol types.

One of the advantages of using our application of Shannon entropy with network traffic is its ability to find design structures in the inspected packets without any knowledge about their protocol specifications. This could be helpful in the process

of reverse engineering proprietary protocols. We propose exploring this feature as a topic of future research in Chapter 10.

7 Frequency Distributions of (p, n)-grams

This chapter and the following one use empirical analysis to show how a *small* set of frequent (p, n)-grams can be calculated to capture protocol design structures and uniquely fingerprint individual protocols in network traffic. In particular, this chapter provides statistical evidence for the rapidly-dropping-off distribution behavior of (p, n)-grams. This specific distribution behavior implies that the interesting frequent (p, n)-grams (required to fingerprint the protocols' high-level structural designs) only constitute a *small* set of the total domain of (p, n)-grams. This characteristic accounts for the efficiency advantage in using (p, n)-grams to characterize network protocols.

The chapter first describes our experimental procedure and rationale. It then presents our empirical analysis and results supporting the power-law-like distribution behavior of (p, n)-grams. More experiments are also presented to test their distribution behavior when using different sizes of n ($1 \leq n \leq 16$), and different lengths of network traces. Chapter 9 provides a conceptual model that explains and generalizes our empirical results of the (p, n)-grams' frequency distribution behavior.

7.1 Experiments Procedure and Rationale

Experiments here follow a general procedure to analyze and validate (p, n)-gram frequency distribution behaviors in network traffic. The following steps describe this procedure for all inspected traces of network traffic:

Step 1: Calculate (p, n)-gram frequencies: The first step is to calculate packet match-

ing frequencies of all distinct (p, n)-grams in the inspected network trace.

Different sizes of n ($n = 1, 2, ..., 16$) are tried in Section 7.2.2 in order to compare their behaviors and choose the proper size of n. Based on our results, we choose to use a default size of ($n = 2$). This choice is based on our observation that frequent (p, n)-grams with $n \geq 3$ usually represent long patterns of the same protocol in network packets, whereas frequent (p, n)-grams with $n = 1$ are more likely to represent patterns (short or long) of more than one protocol at the same time. (p, n)-grams with $n = 2$, on the other hand, combine these two representation advantages, which make them a better choice for our traffic characterization applications. We further discuss our choice of the default size of n in Section 7.2.3.

Step 2: Calculate the distribution's model: This step graphs frequency on the y-axis versus rank (ordinal index) on the x-axis, on a log-log scale, using the frequency data obtained in step 1. It then calculates the slope of the regression line (i.e., model or power-law exponent α) using the following conventional slope formula:

$$\text{slope} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = -(\alpha) \tag{7.1}$$

where $\bar{x}$ and $\bar{y}$ are the means of the x and y values respectively.

Note that we calculate the frequency distribution behavior of the first 1,000 most frequent (p, n)-grams only. We choose this specific number because we observe that the distribution behavior of (p, n)-grams may follow more than one regime as we consider (p, n)-grams from the third concentration Region (i.e., Region **C**, introduced in Section 6.1).

This is analogous to the frequency distribution behavior of natural language words which deviates from Zipf's law after considering their long tail (i.e., rank $\geq$ 5,000), as discussed in Section A.2. Therefore, in most of our experiments, we test the first 1,000

most frequent (p, n)-grams only, in order to 1) achieve consistency, and 2) exclude (p, n)-grams of Region **C** from the frequency distribution computation[1].

Step 3: Validate the model's goodness of fit: The third step informally validates the calculated model through calculating its goodness of fit using R^2 (the *coefficient of determination*) [42]. R^2 measures the strength of the relationship between frequency on the y-axis and rank on the x-axis. More specifically, it represents the proportion of common variation in the two variables y and x through the following formula:

$$R^2 = \left(\frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt[2]{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \right)^2 \tag{7.2}$$

R^2 is commonly used in the literature to measure the goodness of fit of regression lines in power-law distributions [30, 7, 33]. The range of R^2 goes between 0.00 and 1.00, where a value of 1.0 means a perfect fit, and a value of 0.0 means no fit. Multiplying R^2 by 100 gives the percentage of variance in common between the two variables [66]. For example, a value of $R^2 = 0.90$ in a (p, n)-grams' frequency-vs-rank graph is interpreted as: 90% of the variability in the (p, n)-gram's frequency can be attributed to or explained by the variance in the (p, n)-gram's ordinal index.

Using R^2 as the only parameter to deduce goodness of fit may not be always accurate [30]. However, it is good enough for the purpose of this research where the main focus is to verify the rapidly-dropping-off distribution behavior of (p, n)-grams rather than to determine the exact distribution behavior model.

Step 4: Confirm our findings: We confirm our findings of the calculated (p, n)-gram distributions through testing two conditions, namely: trace independence, and scale invariance. For trace independence, we try two independent datasets from

[1]Our tests with all the traces extracted from the two datasets show that the average size for Region **A** in these datasets is close to one hundred (p, n)-grams.

different network environments (CCSL and MD). For each dataset, we try many traces from random dates, and during selected periods of time that represent different network behaviors and modes of operation. On the other hand, for scale invariance, we try traces with different time sizes (1-sec to 1-week) in order to ensure that the calculated frequency distribution behavior scales with the trace size.

It is important to note, however, that in this research, we don't assert a firm compliance of (p, n)-grams frequency distribution to Zipf's law. Instead, we assert a rapidly-dropping-off frequency distribution of (p, n)-grams and *conjecture* that it follows a power-law-like behavior similar to that of Zipf's law. Although we provide statistical evidence to support our conjecture, we believe that assuring an accurate Zipf's law behavior requires experiments with multiple enterprise datasets, and needs error parameters that are more sensitive than R^2 (such as the ones suggested by Clauset et al. [29, 30]). Within the scope of this research, assuring a rapidly-dropping-off distribution behavior of (p, n)-grams is sufficient to ensure the required applications' efficiency.

7.2 Rapidly Dropping Off Distribution Behavior

Section 4.1.1 describes the datasets we used to analyze (p, n)-gram frequency distributions in network traffic. This section presents the experiments that we did to test, validate, and confirm the distribution behavior of (p, n)-grams in network traffic using multiple traces from the CCSL and MD datasets at various random dates and operation modes. The experiments also study the distribution behavior using different sizes of n, and different trace lengths.

7.2.1 Empirical Analysis

Table 7.1 shows the power-law models calculated for several 3-hour traces from the CCSL dataset using a default size of ($n = 2$). These network traces were randomly selected to cover four different time periods: two in the morning (4am-7am and 8am-11am), one in the afternoon (1pm-4pm), and one in the evening (5pm-8pm). In addition, Table 7.2 shows the power-law model calculated for four randomly selected 3-hour traces from the MD dataset.

Our choice of the four specific CCSL times is based on their representation of the different working activity types we have in the CCSL lab. In the early morning (4am-7am), no users are expected to be in the lab. Therefore, network traffic captures at that time are usually dominated by automated routine network-related packets, such as ARP, HSRP, and EIGRP. Appendix B.2 provides full names and references for these network protocol acronyms.

During the second morning period (8am-11am) and the first afternoon period (1pm-4pm), on the other hand, most of the students come to the lab and start using their machines to run applications, execute shell scripts, check their emails, surf the Internet, print documents, etc. Protocols such as HTTP, SSL, SSH, and CUPS are commonly found during these time periods. Finally, during the evening period (5pm-8pm), some students may play media applications, and download media files. Protocols, such as RTP and others are examples of protocols commonly found during this period.

Both tables show the calculated model or power exponent α, which represents the slope of the regression line in a log-log-scale graph multiplied by -1. They also show its goodness of fit measure R^2 (coefficient of determination) which ranges from 0.0 to 1.0, where 1.0 means a perfect fit. In addition, the tables show the percentages of the TCP, UDP, non-IP, and other IP protocols in the corresponding network trace[2] as well as the total size, and the average packet length for each trace.

As discussed in Section 7.1, R^2 measures the strength of correlation between two

[2]Note that the Aug traces are the only traces that do not include non-IP protocol packets.

	Aug 13 Fri. 4-7am	**Dec 11 Sun. 4-7am**	**Jan 20 Fri. 4-7am**	**Apr 8 Sat. 4-7am**
α	**0.72**	**1.01**	**1.21**	**1.10**
R^2	0.81	0.94	0.91	0.93
TCP (%)	5.00 %	38.90 %	18.35 %	16.05 %
UDP (%)	84.46 %	36.96 %	46.06 %	50.62 %
Other IP (%)	10.55 %	6.16 %	4.89 %	8.28 %
Non-IP (%)	**0.00 %**	17.98 %	**30.71 %**	25.04 %
avg pack size (B)	147.44	94.27	80.52	95.42
trace size (MB)	7.4	9.4	9.5	6.1
	Aug 19 Thu. 8-11am	**Dec 15 Thu. 8-11am**	**Jan 24 Tue. 8-11am**	**Apr 5 Wed. 8-11am**
α	**0.73**	**1.03**	**1.05**	**1.07**
R^2	0.87	0.95	0.97	0.94
TCP (%)	38.07 %	42.50 %	77.44 %	41.68 %
UDP (%)	55.11 %	31.42 %	11.87 %	35.93 %
Other IP (%)	6.82 %	5.04 %	1.61 %	4.44 %
Non-IP (%)	**0.00 %**	21.04 %	9.08 %	17.95 %
avg pack size (B)	217.79	326.27	180.99	213.4
trace size (MB)	17	39	59	24
	Aug 16 Mon. 1-4pm	**Dec 13 Tue. 1-4pm**	**Jan 26 Thu. 1-4pm**	**Apr 6 Thu. 1-4pm**
α	**0.71**	**1.16**	**1.13**	**1.08**
R^2	0.91	0.95	0.98	0.98
TCP (%)	51.32 %	61.72 %	63.78 %	70.69 %
UDP (%)	43.39 %	16.80 %	27.65 %	22.70 %
Other IP (%)	5.29 %	3.06 %	1.52 %	1.65 %
Non-IP (%)	**0.00 %**	18.42 %	7.05 %	4.96 %
avg pack size (B)	428.41	587.37	238.49	352.86
trace size (MB)	41	124	96	119
	Aug 16 Mon. 5-8pm	**Dec 12 Mon. 5-8pm**	**Jan 22 Sun. 5-8pm**	**Apr 5 Wed. 5-8pm**
α	**0.72**	**1.14**	**1.29**	**1.07**
R^2	0.85	0.96	0.94	0.94
TCP (%)	29.81 %	48.77 %	45.71 %	37.73 %
UDP (%)	62.26 %	28.12 %	22.02 %	41.80 %
Other IP (%)	7.93 %	3.05 %	3.33 %	6.27 %
Non-IP (%)	**0.00 %**	20.06 %	**28.94 %**	14.20 %
avg pack size (B)	189.42	184.43	73.15	262.13
trace size (MB)	13	37	13	24

Table 7.1. Observing the (p, n)-grams power-law-like distribution behavior in the CCSL dataset. This table gives the power exponent α calculated for randomly selected 3-hour traces from the CCSL dataset.

	Nov 1 Thu. 12-3pm	**Nov 5 Mon. 4-7pm**	**Nov 7 Wed. 2-5pm**	**Nov 13 Tue. 12-3pm**
α	**0.89**	**1.07**	**1.21**	**0.95**
R^2	0.88	0.94	0.96	0.91
TCP (%)	71.41%	95.77%	84.00%	96.23%
UDP (%)	25.12%	0.98%	12.65%	1.93%
Other IP (%)	0.20%	0.20%	0.20%	0.38%
Non-IP (%)	3.27%	3.05%	3.14%	1.46%
avg pack size (B)	823.36	1213.76	952.65	1001.39
trace size (MB)	96	134	115	58

Table 7.2. Observing the (p, n)-grams power-law-like distribution behavior in the MD dataset. This table gives the power exponent α calculated for different 3-hour traces from the MD datasets.

variables (i.e., frequency and ordinal index of (p, n)-grams in our case). That is, multiplying R^2 by 100 represents the percent of variance in common [66]. For example, a value of $R^2 = 0.94$ in this table, means that 94% of (p, n)-gram frequencies are directly attributable to their ordinal indexes according to the computed regression line (with slope of $-\alpha$) and vice versa.

Focusing on the values of α in both tables shows that they are mainly close to 1.0, but vary slightly from one trace to another. Similarly, the values of R^2 are mostly ≥ 0.91, which indicate that the model has a good level of fit. The consistent value of the slope (α), its goodness of fit (R^2), and its slight variation from trace to trace, all support our *conjecture* of the power-law-like behavior (introduced in Section A.2) of (p, n)-grams frequency distribution in network traffic.

Our observations suggest that the larger and more diverse the network trace, the higher the goodness of fit (i.e., R^2) we get for the model. This is similar to the case of the word frequencies in natural languages that we present in Section 10.3. The larger and more comprehensive the natural language corpora, the more precise the Zipf's law behavior of the words frequency distribution.

Another look at the two tables shows how different traces consist of different volumes of protocol types. Further analyzing their impacts shows that the differences observed in the values of α are due to the differences between the types and volumes

of the protocols constituting each trace. More specifically, we found that although the values of α are close to 1 (in most of the traces), they may go either slightly lower or slightly higher, depending on the dominant protocols in the trace, their percentages, structural designs and modes of operation.

This explains the impact of the specific time period during which the network trace was captured. Basically, the value of the power exponent α, as well as the goodness of fit, depend on the temporally running applications and their percentages in the trace. For example, the relatively low value of α in the CCSL Aug traces is mainly due to the missing non-IP (e.g., ARP) packets whose value of α is usually relatively high compared to the other IP protocols. On the contrary, the high percentage of ARP packets in the Jan 20 (4-7am), and Jan 22 (5-8pm) traces contributes to their relatively high value of α.

Chapter 8 studies the behaviors of (p, n)-gram frequency distributions for individual protocols (i.e., protocol-specific network traces). It shows that, for each protocol, the distribution relies on the special design structures found in the protocol's corresponding packets. It also observes a rapidly-dropping-off frequency distribution of (p, n)-grams with a power-law-like behavior for most of the IP protocols that were tested. The main exception to this common behavior was for the broadcast and multicast protocols, such as EIGRP and HSRP. Those protocols feature similar or identical contents for their entire packet bodies.

For all the mixed-protocol traces we tested in Tables 7.1 and 7.2, multicast and broadcast protocols (i.e., those that don't follow a power-law-like distribution behavior) constitute a very limited traffic volume compared to the rest of the running protocols. This limits their impact on the overall distribution behavior of (p, n)-grams, especially as we only consider the first 1,000 frequent (p, n)-grams in our experiments.

Two questions may arise at this point of the discussion, namely: 1) do we get the same (p, n)-gram frequency distribution behavior when using different sizes of n? 2) Does the power-law-like distribution behavior of (p, n)-grams scale with traffic length? We discuss the answers to these questions and others in the following subsections.

7.2.2 Different Sizes of n

We conducted several experiments to test the (p, n)-gram frequency distribution behaviors with different sizes of n. Figure 7.1 and Table 7.3 summarize a subset of our empirical results with values of n between 1 and 16. Our results suggest that smaller values of n (i.e., $n \leq 6$) show a closer compliance with the power-law-like behavior than larger values.

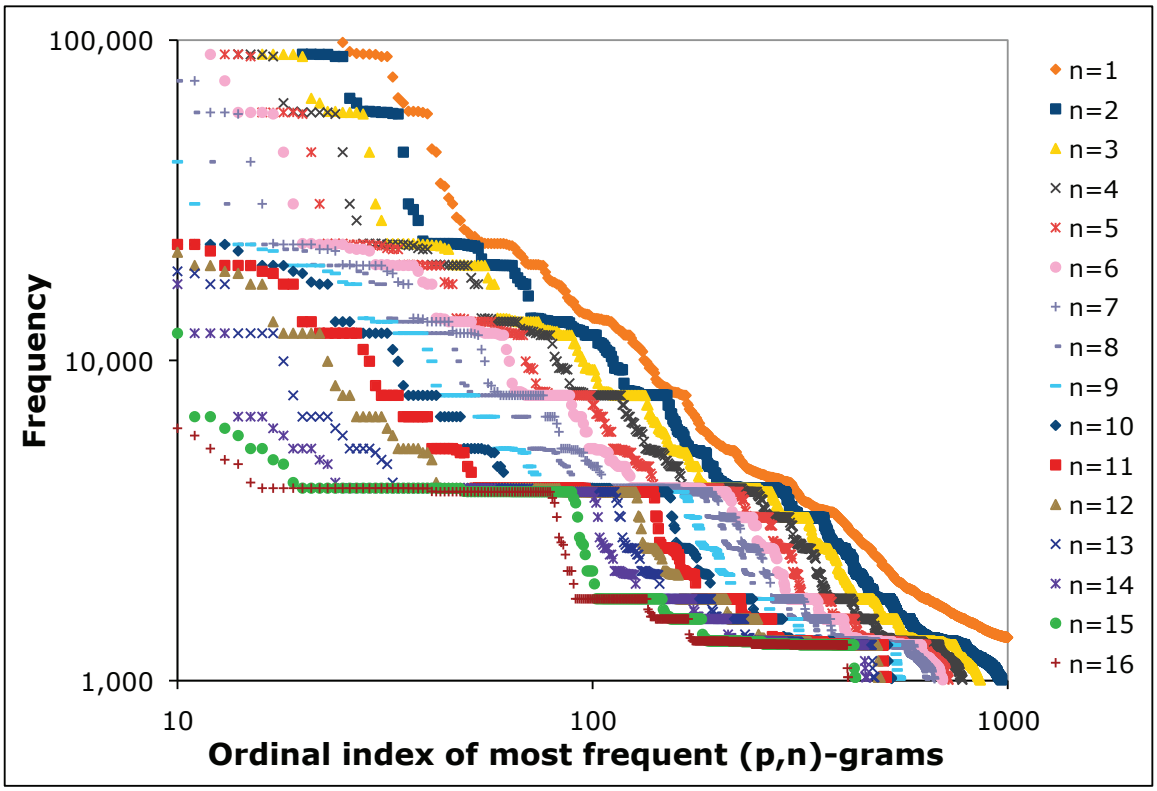


Figure 7.1. (Best viewed in color) (p, n)-gram frequency distributions with different sizes of n (1-hour CCSL trace). Note that as the size of n gets larger, the line becomes more flattened and less smooth (i.e., consisting of connected short line segments instead of connecting points).

In other words, as n increases, the slope starts to deviate from unity, and the goodness of fit decreases. This can be more clearly recognized by looking at Figure 7.1. Note that as n increases, the line becomes 1) more flattened (resulting in lower values of α), and 2) less smooth (resulting in lower values of R^2).

	n=1	**n=2**	**n=3**	**n=4**	**n=5**	**n=6**	**n=7**	**n=8**
α	1.034	1.047	1.035	1.039	1.015	0.983	0.947	0.908
R^2	0.977	0.981	0.983	0.982	0.982	0.982	0.982	0.979

	n=9	**n=10**	**n=11**	**n=12**	**n=13**	**n=14**	**n=15**	**n=16**
α	0.888	0.846	0.807	0.765	0.720	0.671	0.629	0.595
R^2	0.976	0.972	0.968	0.962	0.955	0.947	0.938	0.936

Table 7.3. Power exponent α behaviors with different sizes of n (1-hour CCSL trace).

The more flattened-line behavior (i.e., $\alpha < 1$) is due to the additional matching constraint that is added by increasing the size of the matching substring. Thus, the larger the n, the less frequent the (p, n)-grams become. Table 7.4 shows this behavior on the first 10 frequent (p, n)-grams, using three different sizes of n (2, 3, and 4).

The first column of each table (labeled **%**) represents the (p, n)-gram's matching frequency percentage with respect to the total number of packets in the inspected network trace. Note, for example, how the first (p, n)-gram with $n = 2$ has a matching frequency of 72.15%, as opposed to 48.43% and 32.04% in the case of the first (p, n)-gram with $n = 3$ and $n = 4$, respectively.

$(p, \mathbf{2})$-grams				$(p, \mathbf{3})$-grams					$(p, \mathbf{4})$-grams					
%	p	1^{st}	2^{nd}	**%**	p	1^{st}	2^{nd}	3^{rd}	**%**	p	1^{st}	2^{nd}	3^{rd}	4^{th}
72.15	0	45	00	**48.43**	0	45	00	00	**32.04**	39	00	00	00	00
71.85	38	00	00	**46.52**	16	86	75	e1	**31.98**	40	00	00	00	00
70.27	16	86	75	**29.43**	39	00	00	00	**31.91**	41	00	00	00	00
57.05	6	40	00	**29.43**	40	00	00	00	**27.55**	38	00	00	01	01
53.46	1	00	00	**29.37**	42	00	00	00	**27.48**	40	01	01	08	0a
51.10	17	75	e1	**29.36**	41	00	00	00	**27.48**	39	00	01	01	08
40.71	12	86	75	**24.86**	38	00	00	00	**27.29**	38	00	00	00	00
36.63	28	00	00	**24.75**	38	00	00	01	**25.28**	28	00	00	00	00
33.13	40	00	00	**24.73**	39	00	01	01	**20.23**	16	86	75	e1	3a
32.22	39	00	00	**24.66**	41	01	08	0a	**17.21**	34	00	00	00	00

Table 7.4. List of the first 10 most frequent (p, n)-grams and their matching frequencies using three different sizes of n (2, 3, and 4). This sample was calculated from a 4-week CCSL network trace. Note that for all (p, n)-grams, the offsets (p) are reported in decimal, while the actual bytes (1^{st}, 2^{nd}, 3^{rd}, and 4^{th}) are reported in hexadecimal.

On the other hand, the less smooth line behavior (with lower goodness of fit of the linear regression line) is because frequent (p, n)-grams with larger values of n usually represent the same long patterns in the inspected traffic (further discussed

in Section 7.2.3). (p, n)-grams matching and overlapping substrings of these long patterns would feature similar frequencies. Thus, they show as disconnected straight horizontal lines on the frequency distribution graph instead of a diagonal line of connecting points as is the case with smaller values of n.

7.2.3 Our Default Size of n

Based on the discussion in Section 7.2.2, applications that use (p, n)-grams to distinguish between different traffic types need to carefully set their size of n. The size of n has an impact on the pattern lengths that (p, n)-grams can recognize. Larger values of n are usually more suitable to recognize long patterns, however, they obviously can't represent short patterns whose sizes are shorter than n.

Moreover, we observe that frequent (p, n)-grams with $n \geq 3$ usually represent long patterns of the same protocol in network packets, whereas frequent (p, n)-grams with $n = 1$ are more likely to represent patterns (short or long) of more than one protocol at the same time. This means that, from a functionality point of view, there might be more than one good size of n for an application. Therefore, if efficiency is not a concern, this may suggest using (p, n)-grams with different sizes of n in the same application, where each size has its own functionality advantage.

On the other hand, the size of n has an impact on efficiency. Consider for example the impact on the overall (p, n)-grams *sample space*. The size of (p, n)-grams' *sample space* in a network trace is equal to the number of *calculated* distinct (p, n)-grams in the trace[3]. Dealing with a significantly larger size of (p, n)-grams sample space may cause an efficiency degradation to the system due to the number of (p, n)-grams to be considered during the process of calculating (p, n)-gram frequencies.

Table 7.5 shows the domain space size (assuming a maximum Ethernet packet size of 1,500) and the actual sample space size computed for (p, n)-grams with different

[3]This is less than or equal to the (p, n)-grams *domain space*, which represents the total number of *possible* distinct (p, n)-grams that can be represented by n bytes. Domain space can be simply calculated using $(packetSize - n + 1) \times 2^{8n}$.

sizes of n, in a short 1-hour CCSL network trace. Note that while domain space grows exponentially with larger sizes of n, the actual sample space starts with a relatively small value at $n = 1$ (360,875), a larger value at $n = 2$ (29,447,300), and then grows slowly after $n = 3$ (39,846,657). This suggests that using (p, n)-grams with size $n = 1$ gives the most efficient performance.

	n=1	**n=2**	**n=3**	**n=4**	**n=5**	**n=6**
Sample space	360,875	29,447,300	39,846,657	41,416,971	42,491,928	43,429,463
Domain space	383,744	98,304,000	2.514e+10	6.438e+12	1.648e+15	4.219e+17

	n=7	**n=8**	**n=9**	**n=10**	**n=11**	**n=12**
Sample space	43,993,236	44,503,647	45,003,856	45,489,899	45,900,143	46,233,942
Domain space	1.080e+20	2.765e+22	7.078e+24	1.812e+27	4.639e+29	1.187e+32

	n=13	**n=14**	**n=15**	**n=16**		
Sample space	46,557,734	46,867,934	47,175,689	47,405,809		
Domain space	3.040e+34	7.783e+36	1.992e+39	5.100e+41		

Table 7.5. Sample space and domain space of (p, n)-grams with different sizes of n (1-hour CCSL trace).

Taking both points (functionality and efficiency) into consideration, we tried our applications with different sizes of n, and found that values between 1 and 4 give the most accurate traffic characterization results. Out of the four values, we found that using (p, n)-grams with $n = 2$ gives a good tradeoff between functionality and efficiency in most of our experiments. That is, they are relatively more efficient than (p, n)-grams with $(n \geq 3)$ and at the same time they combine the two representation advantages of (p, n)-grams that exist with larger and smaller values of n. This is what makes (p, n)-grams with $n = 2$ a better choice for our traffic characterization applications.

7.2.4 Different Trace Lengths

An interesting question that we wanted to explore with (p, n)-gram distributions was whether their power-law-like behavior scales with the length of the network traffic or

not. We tested this scalability question using various traces from the CCSL and MD datasets. Table 7.6 and Figure 7.2 show an example of how using different lengths of network traces may impact their (p, n)-gram frequency distributions.

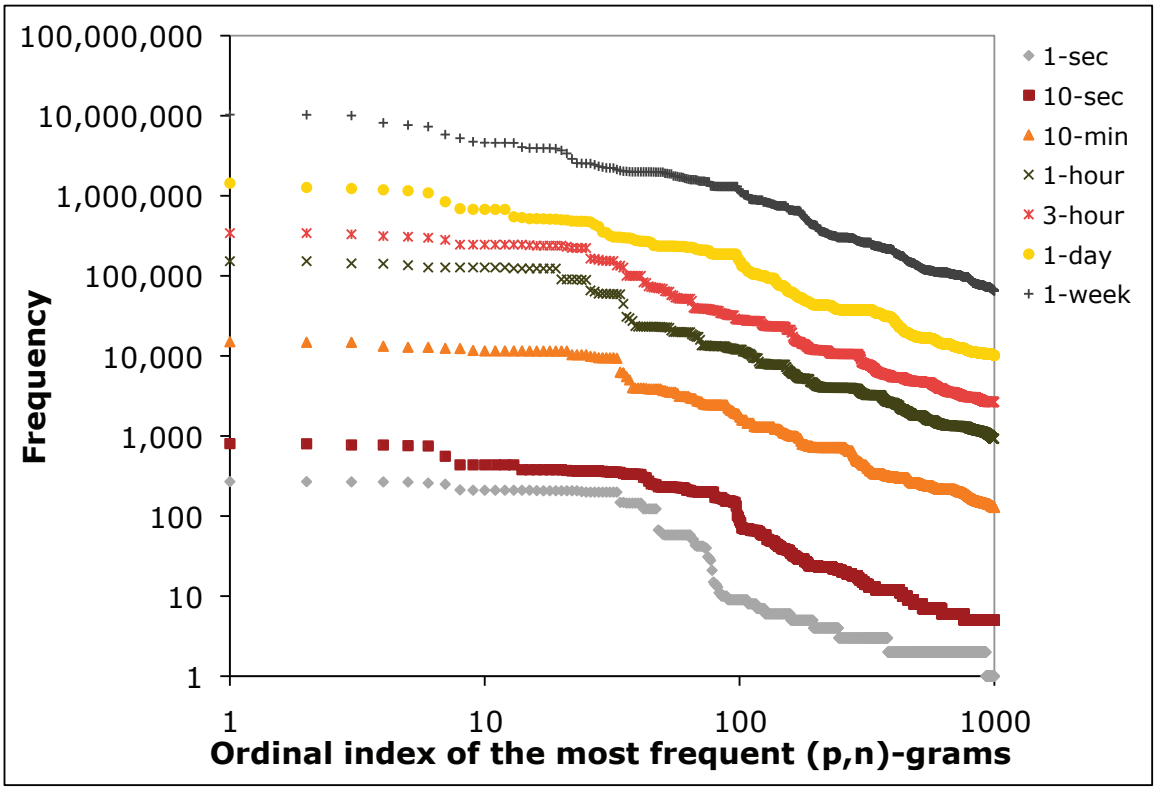


Figure 7.2. (Best viewed in color) (p, n)-grams frequency distribution with different capturing time periods, when $n = 2$ (CCSL network traces).

	10-min trace		1-hour trace		3-hour trace		1-day trace		1-week trace	
	α	R^2	α	R^2	α	R^2	α	R^2	α	R^2
n=1	1.07	0.97	1.03	0.98	1.07	0.98	n/a	n/a	n/a	n/a
n=2	1.05	0.97	1.05	0.98	1.05	0.98	1.02	0.97	1.04	0.97
n=3	1.02	0.98	1.03	0.98	1.00	0.98	0.97	0.97	0.98	0.96
n=4	0.99	0.98	1.04	0.98	0.97	0.98	0.93	0.96	0.94	0.98

Table 7.6. Power exponent α behaviors with different capturing time periods (10-min, 1-hour, 3-hour, 1-day, and 1-week sample traces)

Basically, (p, n)-gram distributions look similar across the different trace lengths.

That is, almost all tested lengths of network traces (1-week, 1-day, 3-hour, 1-hour, and 10-minute) give similar behaviors of (p, n)-gram frequency distributions. We call this behavioral characteristic *scale invariance*, and describe it as a behavior that does not change if the length of the system is multiplied by a common factor.

We find, however, that with relatively short traces (e.g., 1-sec trace in Figure 7.2) the distribution may deviate from the common behavior, due to the effect of spikes in the captured network traffic. That is, short traces are usually dominated by one or a few protocol-specific spikes, which impact the (p, n)-grams distribution accordingly. In other words, the frequency distribution takes the same behavior as the dominating protocol in the spike.

This explains why the 1-sec trace doesn't show similar behavior to the longer ones in Figure 7.2. A closer look at this trace shows that it was captured during an SSH spike, and that it is mostly populated with SSH packets. This gives it an SSH-similar (p, n)-grams distribution behavior that is slightly different from the commonly found one with mixed-protocol traces. Chapter 8 further explains this special case.

7.2.5 Packet Sampling

As discussed earlier in Chapter 6, the (p, n)-gram frequency distribution in network traffic follows a rapidly-dropping-off distribution with a power-law-like behavior. Because this distribution decays very quickly, very few (p, n)-grams are frequent enough to be candidates for splitting. Thus, it becomes feasible to estimate (p, n)-gram frequencies using packet samples, further increasing the efficiency of the algorithm. Section 3.3.2 discusses how we implement packet sampling in ADHIC.

Figure 7.3 plots the (p, n)-gram frequency distributions for a 3-hour dataset using different sampling rates (50%, 20%, 10%, 5%, 3%, 2%, and 1%). Note that (p, n)-gram frequency distributions are not noticeably affected by packet sampling. That is, calculating (p, n)-gram frequencies while performing packet sampling can still resemble the general composition of the (p, n)-grams frequency distribution behavior of the underlying network.

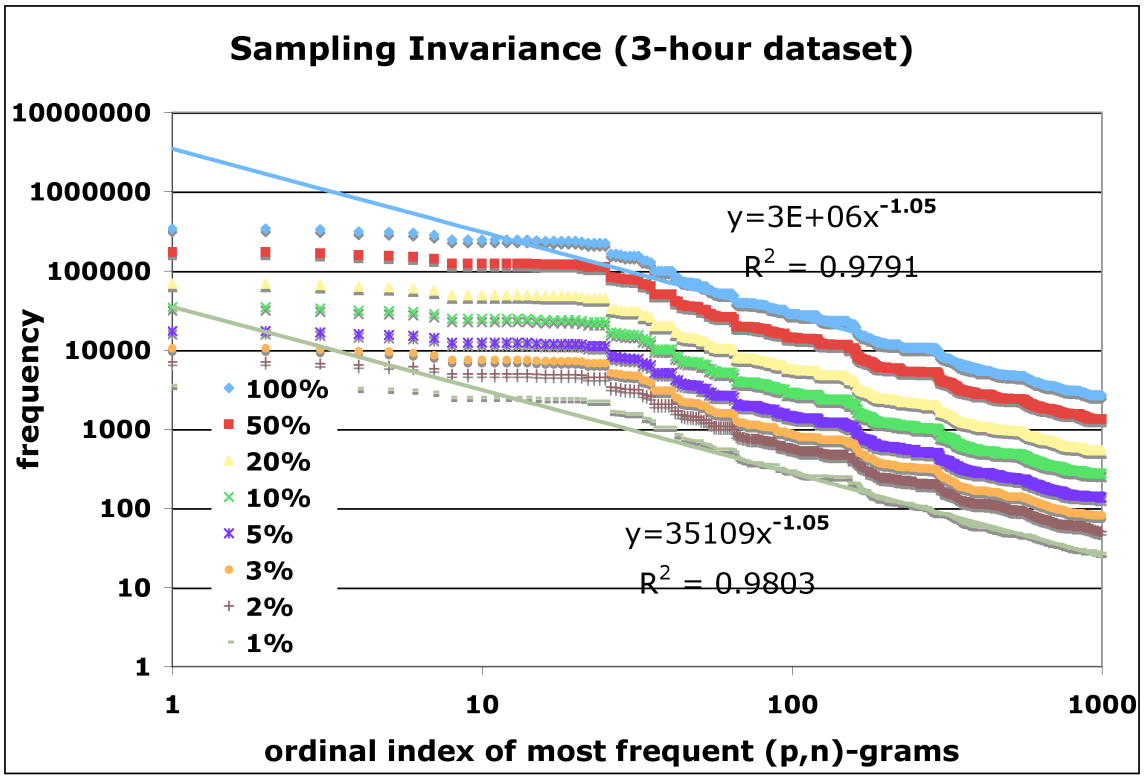


Figure 7.3. (p, n)-gram frequency distributions feature sampling invariance. Note how the (p, n)-gram frequency distribution does not seem to be affected by the rate at which packets are sampled.

8 Pattern Capturing Using (p, n)-grams

This chapter discusses the pattern capturing characteristic of (p, n)-grams, which gives (p, n)-grams the desired functionality to be used to fingerprint individual structured protocols, as long as their corresponding protocols differ in their design structures. It starts by discussing the semantic meanings of frequent (p, n)-grams in network traffic, which are crucial to their pattern capturing characteristic. It also shows how we utilize our entropy definition on network packets (introduced in Section 6.2) to calculate entropy models of individual network protocols with different design structures. Protocol entropy models give a visualization aid to map design structures of individual protocols and the corresponding content similarities and (p, n)-gram distributions.

The chapter then uses traces of individual protocols to calculate their (p, n)-gram distribution behaviors. It shows that differences between protocol structural designs are reflected in the corresponding (p, n)-gram frequency and offset distribution behaviors. Our methodology is to use structural (p, n)-grams to fingerprint network protocols by leveraging their ability to capture differences between network protocols in their design structures. Chapters 4 and 5 already implemented this fingerprinting methodology in our traffic clustering and security monitoring applications.

Our traces of individual protocols were extracted from the CCSL and MD datasets (introduced in Section 4.1). Therefore, recalculating (p, n)-gram characteristics and distribution behaviors using these complementary traces also serves as a "*cross-validation*" of the concluded (p, n)-gram distribution behaviors presented in Chapter 7.

8.1 Semantic Meanings of Frequent (p, n)-grams

As discussed in Section 6.1.2, offset p adds semantic meanings to (p, n)-grams that give (p, n)-grams adequate representation of the different network packet types. These semantic meanings can be visualized using graphs of (p, n)-grams offset distribution. For example, Figure 8.1 plots the (p, n)-grams offset distribution in Region **A** and part of Region **B** (both regions are defined in Section 6.1.2), for three different sizes of n ($n = 1$, 2, and 3) using a random 3-hour trace from the CCSL dataset.

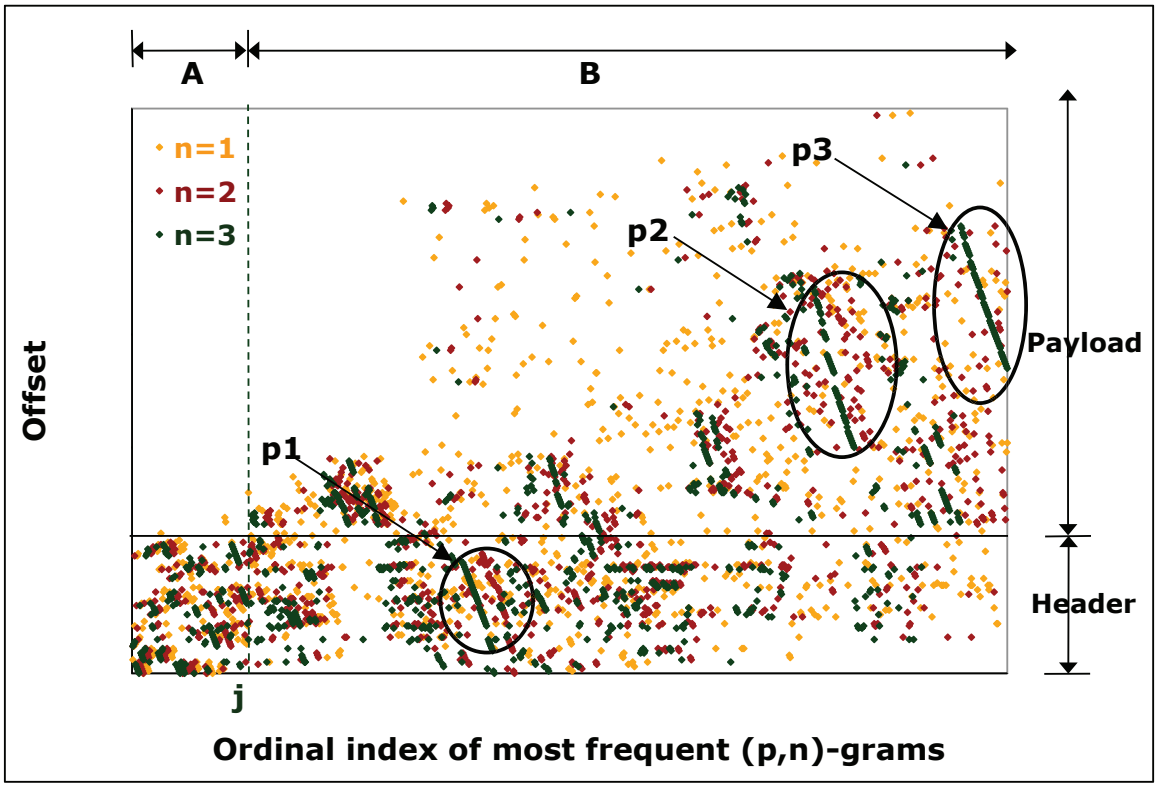


Figure 8.1. (Best viewed in color and electronically to allow enlargement) (p, n)-gram offset distribution graph showing protocol patterns in network traffic with three different sizes of n ($n = 1, 2, 3$). Circles in the graph (**p1**, **p2**, and **p3**) point to examples of common patterns in some of the component protocols.

The scatter graph in this figure visualizes the byte-sequence patterns representing protocol design structures in network packets. It also shows that Region **A** has a heavy concentration of header (p, n)-grams whereas the displayed portion of Region

B has a mixture of header and payload (p, n)-grams. It is common for these patterns to appear as continuous or fragmented diagonal lines on the scatter graph, where the length of the diagonal line is proportional to the pattern length in the corresponding network packets.

For example, the first circle **p1** in Figure 8.1 indicates a common pattern in the Ethernet STP (Spanning Tree Protocol) packets. The lines in the graph correspond to a byte-sequence between offsets 17 and 51 in the payload portion of the STP packets. Moreover, the second circle **p2** indicates another common pattern in some TCP IPP (Internet Printing Protocol) packets. This pattern covers an offset range between 100 and 170, and represents the printer's details negotiated between participating systems. It includes sequences like "attributes-natural-language" in the "operation-attributes name" field of the IPP request packets. On the other hand, the third circle **p3** indicates a pattern in some TCP SSHv2 (Secure Shell Version 2) packets that appears in the offset range between 135 and 198. This pattern represents the encryption algorithm being negotiated between communicating parties. It includes sequences like "diffie-hellman-group1-sha1" in the "key-algorithms string" field of the SSHv2 Key Exchange Initialization packets.

Patterns captured by structural (p, n)-grams may reflect protocol types, design structures, as well as modes of operation. This reflected semantic meaning of frequent (p, n)-grams along with their ability to capture protocol differences in design structures are the two keys for the (p, n)-grams' ability to fingerprint network protocols. Capturing design structures is what we discuss in the following sections.

A careful look at Figure 8.1 shows that the continuous lines representing protocol patterns are more visible when $n = 3$. (p, n)-grams with $n = 1$, on the contrary, mostly do not show these lines and are rather scattered in the chart. As discussed in Section 7.2.3, common (p, n)-grams with $n = 1$ usually belong to more than one pattern in different protocol packets. That is, they are more likely to represent more than one protocol or session at the same time.

(p, n)-grams with $n = 2$, on the other hand, may either represent specific protocol

patterns in the network packets, or be part of more than one protocol pattern within the various network packets. Therefore, for visualization purposes, a size of $n \geq 3$ may give a more clear representation of the patterns. However, for traffic characterization purposes, we usually use a size of $n = 2$ as it can also accommodate shorter patterns or multiple-protocol patterns that may not be captured otherwise.

8.1.1 ADHIC without header (p, n)-grams

As discussed in Section 5.1, we examined how well ADHIC can segregate protocols even if header information becomes useless (we configured NetADHICT to ignore the first 38 bytes of each packet). We found that ADHIC sometimes performs better when no header information is given during (p, n)-gram generation. This is due to the way ADHIC chooses its (p, n)-grams, and the richness of frequent payload (or protocol-specific) (p, n)-grams in the captured traffic. ADHIC may pick a payload (p, n)-gram earlier in the tree that works better with packets seen later in the traffic, resulting in better trees and improved segregation results.

Figure 8.2 gives a closer look at the most frequent (p, n)-grams when calculated with and without the headers. Note the left shifting of the points when we ignore more of the headers. This shift is due to the exclusion of more frequent header (p, n)-grams. The more header (p, n)-grams we exclude, the more payload (p, n)-grams come into the 1000 most frequent (p, n)-grams list.

8.2 Protocol-Dependent Entropy Models

In Section 6.2, we define Shannon entropy on network packets in order to use it as a metric to measure their content similarities. When applied to network packets, Shannon entropy represents the average number of bits that can encode data within a field. Figure 6.5 shows a typical entropy model that we get for network traces from the CCSL and MD datasets.

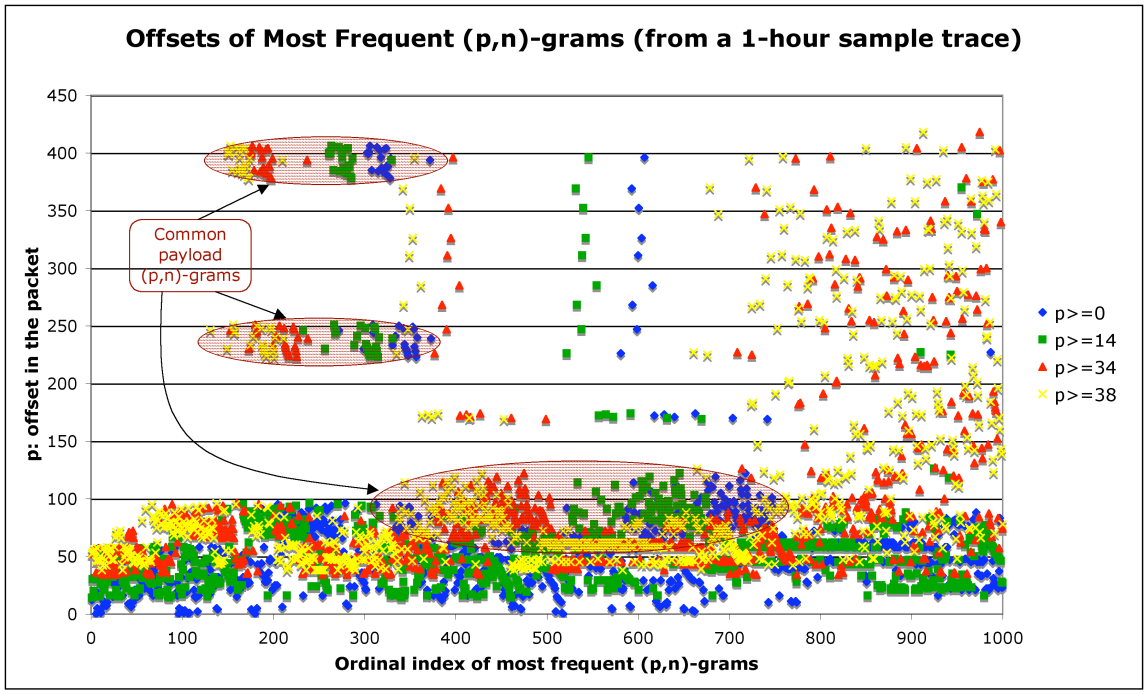


Figure 8.2. Most frequent (p, n)-grams calculated on the whole packets, without Ethernet headers ($p \geq 14$), without IP headers ($p \geq 34$), and without ports ($p \geq 38$). Even when all headers are ignored, it is easy to find many common payload (p, n)-grams.

Each point in the entropy model graph represents the average number of bits that are required to encode 1-byte of data at the corresponding packet offset. Therefore, the lower the entropy level at an offset, the higher the content similarities between packets at that offset. For example, if all the packets in a trace feature one of two value options (say a_1 and a_2) at an offset p_1, with equal probability, then only one bit (i.e., $log_2(2) = 1$) is required, on average, to encode data at this offset. That is, a bit value of "0" may represent a_1, and a value of "1" may represent a_2, or vice versa. Applying Equation 6.1 in this case, gives an entropy of 1 at offset p_1 on the graph.

A quick look at the typical entropy values in Figure 6.5 shows that most of the header offsets feature relatively low entropy levels, as opposed to high entropy levels in the payload portion. This general entropy model is what we usually get when inspecting original traffic captures from the CCSL and MD datasets. However, as

different protocols possess different specifications and design structures, we hypothesize that calculating entropy models exclusively for individual protocols gives different model behaviors. Similarly, (p, n)-grams frequency and offset distribution behaviors of individual protocols are going to be different.

We verify these behavior differences between individual protocols in this chapter. First, we calculate entropy models for traces of individual protocols, and then we discuss the impact of these entropy model differences on their corresponding distribution behaviors of (p, n)-grams (Section 8.3). Differences in entropy models between network protocols explain the (p, n)-grams' ability to fingerprint individual network protocols when they differ in their design structures.

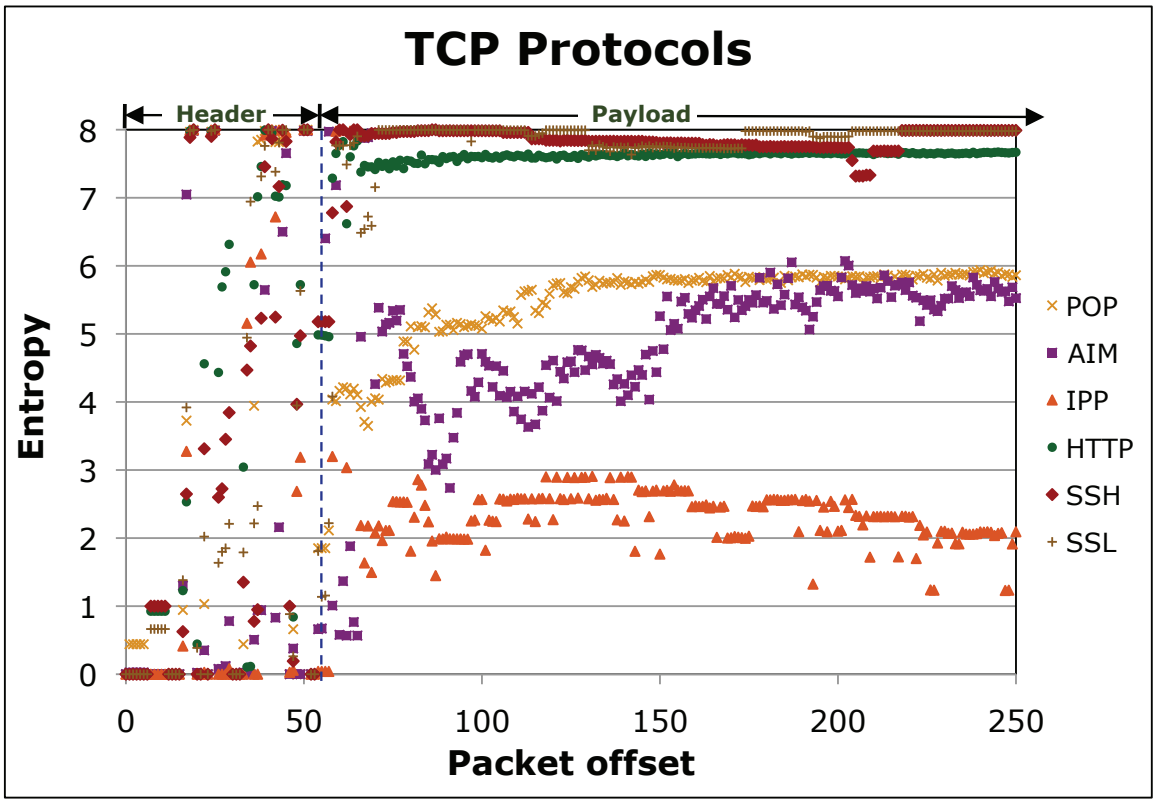


Figure 8.3. (Best viewed in color and electronically to allow enlargement) Shannon entropy models calculated for individual TCP protocols, namely: POP, AIM, IPP, HTTP, SSH, and SSL.

Our experiments here use special traces that were manually extracted from the

CCSL and MD datasets. Each trace exclusively represents one network protocol type. Figures 8.3, 8.4, and 8.5 show entropy model graphs calculated for individual network protocols. The graphs represent six TCP protocols (POP, AIM, IPP, HTTP, SSH, and SSL), six UDP protocols (HSRP, DNS, CUPS, SIP, RTP, and MP3-Streaming), and an Ethernet protocol (ARP) and other IP protocols (EIGRP and ICMP) respectively.

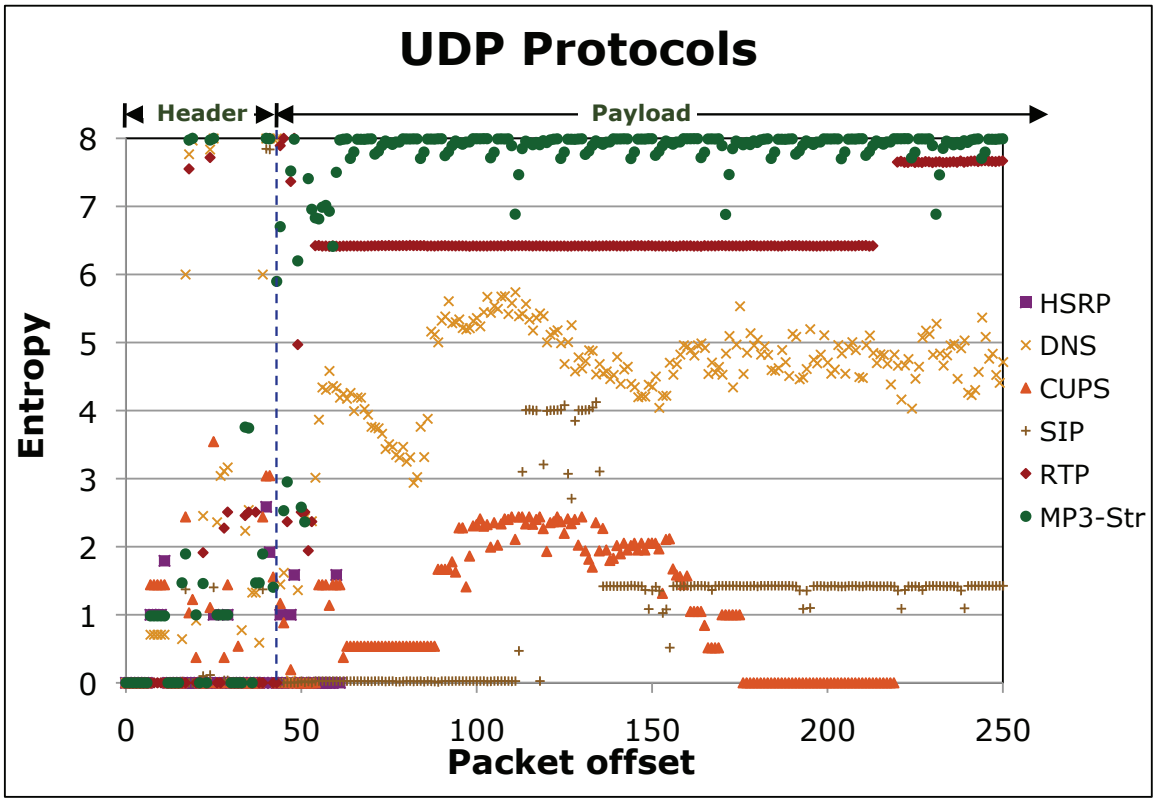


Figure 8.4. (Best viewed in color and electronically to allow enlargement) Shannon entropy models calculated for some individual UDP protocols, namely: HSRP, DNS, CUPS, SIP, RTP, and MP3-Streaming.

It is obvious from the three figures that individual protocols feature different entropy models. Packet offsets with relatively low entropy levels correspond to special patterns or structural packet fields in which contents are highly similar. The entropy model graphs visualize the differences between individual protocols in terms of their special packet specifications and content types. Take for example, the field differences (e.g., offsets and lengths) between packets of Web protocols (e.g., HTTP), multicast

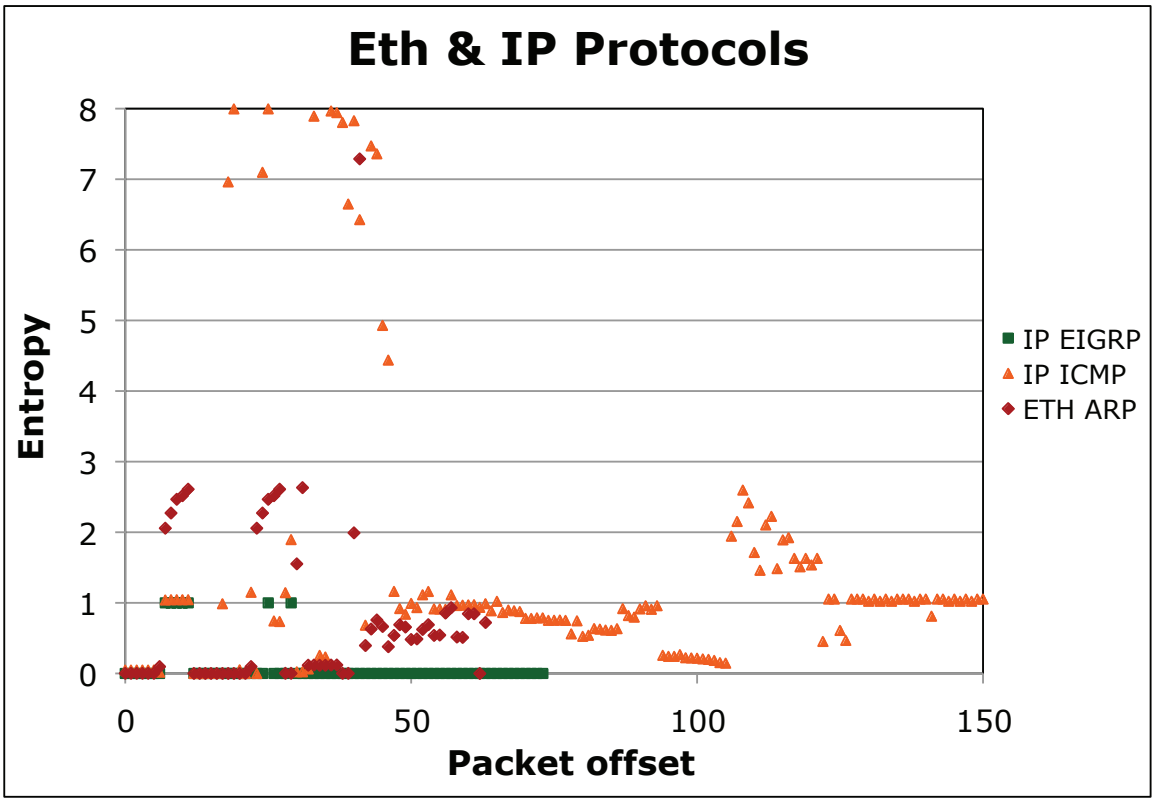


Figure 8.5. (Best viewed in color and electronically to allow enlargement) Shannon entropy models calculated for the EIGRP, ICMP and ARP protocols.

protocols (e.g., EIGRP), streaming protocols (e.g., RTP), and encrypted protocols (e.g., SSH).

Entropy models calculated from traces of individual network protocols are directly related to the frequency and offset distribution behaviors of their (p, n)-grams. Consider, for example, multicast protocols, such as HSRP and EIGRP protocols, which feature low entropy values at the majority of their packet offsets (including both the header and payload portions). Experimenting with a trace of 1,293,451 HSRP packets and another trace of 258,756 EIGRP packets gave a total number of only 102 distinct (p, n)-grams for HSRP, and 83 distinct (p, n)-grams for EIGRP[1]. Frequent

[1]This further explains why (p, n)-grams of multicast protocols do not follow the rapidly dropping off distribution behavior featured by most of the other TCP/IP protocols (as introduced in Section 7.2.1).

(p, n)-grams within the payload low entropy fields are mainly what we use to uniquely fingerprint individual protocols. We further discuss this feature in Section 8.3.

On the other hand, encrypted protocols (e.g., SSL and SSH protocols) and streaming protocols (e.g., MP3-Streaming) feature high entropy values in most of their packet fields. High entropy values imply a high number of distinct (p, n)-grams at each field, which may impact their ability to be used for unique protocol fingerprinting. However, note in the graphs, that in spite of the common high entropy values, there are some packet fields that feature relatively low entropy values (e.g., several fields in the MP3-Streaming protocol, and fields around offset 150 and offset 200 in the SSL and SSH protocols respectively). These fields represent payload patterns and design structures in each protocol, whose frequent (p, n)-grams can be leveraged for protocol fingerprinting.

8.3 Capturing Design Structures in Individual Protocols

This section studies (p, n)-gram distribution behaviors of individual protocols using traces of single-protocol network traffic. The purpose of this study is to test the ability of structural (p, n)-grams to capture differences in design structures between individual protocols. More specifically, the section measures their capturing ability by the impact of the specific design structures of network protocols over their corresponding (p, n)-grams offset and frequency distribution behaviors.

8.3.1 Offset Distribution Behaviors

(p, n)-gram offset distribution graphs for traffic of individual protocols show interesting pattern shapes of each protocol. Figure 8.6 shows examples of these distributions calculated for eight different protocols within a 1-hour random CCSL trace. The protocols are ARP, CUPS, DNS, HTTP, ICMP, SMTP, SSH, and SSL (Appendix B.2, provides a list with protocol names, acronyms, and references).

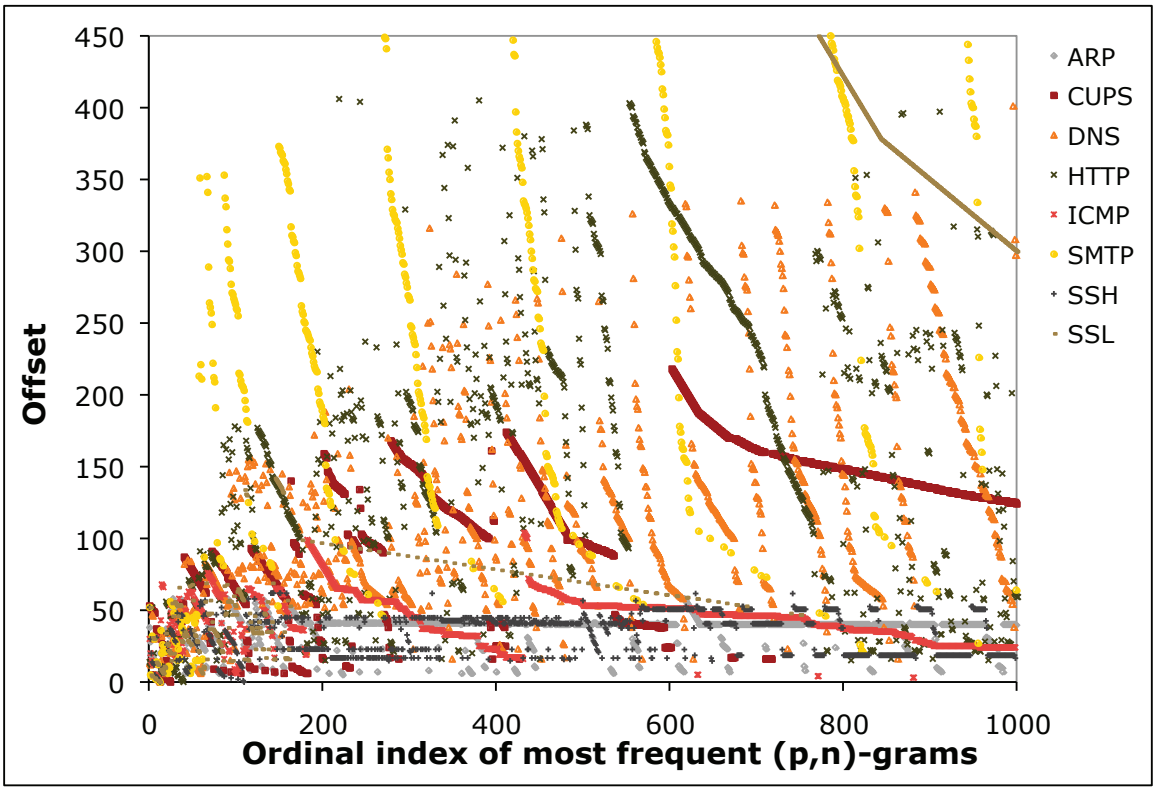


Figure 8.6. (Best viewed in color and electronically to allow enlargement) Offset distribution of the most frequent 1,000 (p, n)-grams for individual protocols, using a 1-hour random CCSL trace.

Basically, lines in the scatter graph represent special patterns in the packets of each protocol. These patterns constitute content similarities between same-protocol packets, which reflect their protocols' special design structures. Plotting the (p, n)-grams offset distribution scatter graph for each protocol visualizes these patterns as continuous or fragmented lines, where the length of each line reflects the pattern's length and varies from one point to a wide offset range.

Some protocols in Figure 8.6 feature horizontal pattern lines while others feature diagonal pattern lines. Different line shapes have different interpretations. For example, the diagonal lines usually represent common patterns that span consecutive bytes within packets. The SSL diagonal line is one good example which represents a common byte sequence describing the certificate information exchanged by the SSL

communicating parties. Another example is the SMTP diagonal lines which represent a special-value padding that spans a packet offset range between 100 and 700. The horizontal lines, on the other hand, mostly represent more than one pattern that appear at the same offset (i.e., different 1-byte value options) with similar frequency. The horizontal lines of the ARP and ICMP protocols are good examples of this behavior.

Figures 8.7 and 8.8 show more focused scatter graphs of the (p, n)-grams offset distributions for individual protocols. The graphs represent four single-protocol traces extracted from 1-week-long CCSL and MD traces. The two figures compare the distributions between TCP and UDP protocols, that is, TCP IPP and TCP MSNMS versus UDP CUPS and UDP SIP.

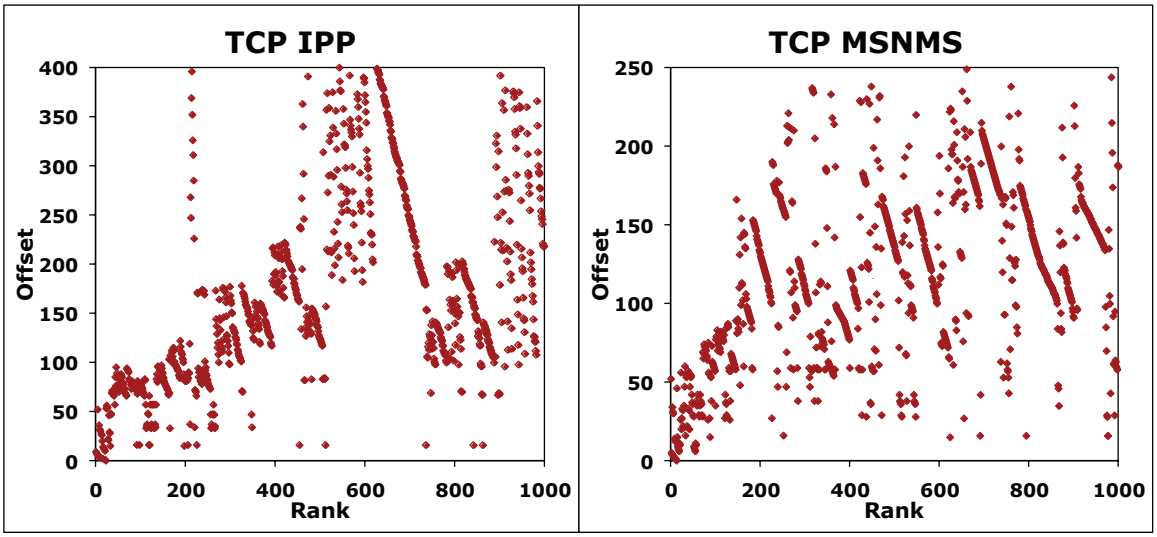


Figure 8.7. (p, n)-gram patterns in two TCP protocols, namely: TCP IPP, and TCP MSNMS.

Each point in the graphs represents a structural (p, n)-gram indicating a special design structure in the corresponding protocol. These points may also be part of lines representing long patterns in their protocols. For example, some of the lines in the IPP protocol's graph represent packet patterns that correspond to the IPP "Printing Operation" attributes, such as charset, language, and printer URI.

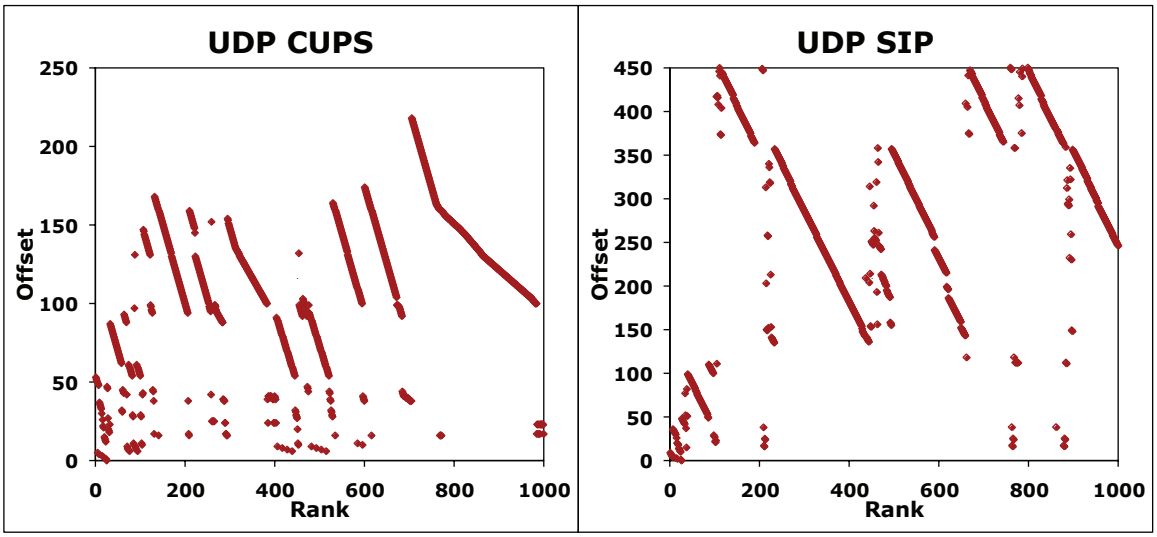


Figure 8.8. (p, n)-gram patterns in two UDP protocols, namely: UDP CUPS, and UDP SIP.

Similarly, some of the lines in the CUPS protocol's graph map to parameters of the CUPS "Browsing Protocol", such as printer URI, location, make and model, etc. Lines in the SIP and MSNMS graphs, on the other hand, represent patterns of the SIP "Message Headers" attributes, such as frequently dialled phone numbers, and host addresses, and patterns of the MSNMS "MSN Messenger Service" parameters, such as the language preference and content type, respectively.

Although the pattern lines in these scatter graphs exist in both types, they are more visible in the UDP protocol graphs than in the TCP ones. Differences between TCP and UDP protocols are commonly observed especially when comparing UDP control protocols with TCP streaming protocols.

In addition, Figures 8.9 and 8.10 compare low and high entropy protocols. That is, they compare two multicast protocols IP EIGRP and UDP HSRP, and two encrypted protocols TCP SSL and TCP SSH. For the EIGRP and HSRP multicast protocols, both the header and payload portions of the packet feature (p, n)-grams with high frequency.

On the other hand, the majority of the high frequency (p, n)-grams of the SSH and

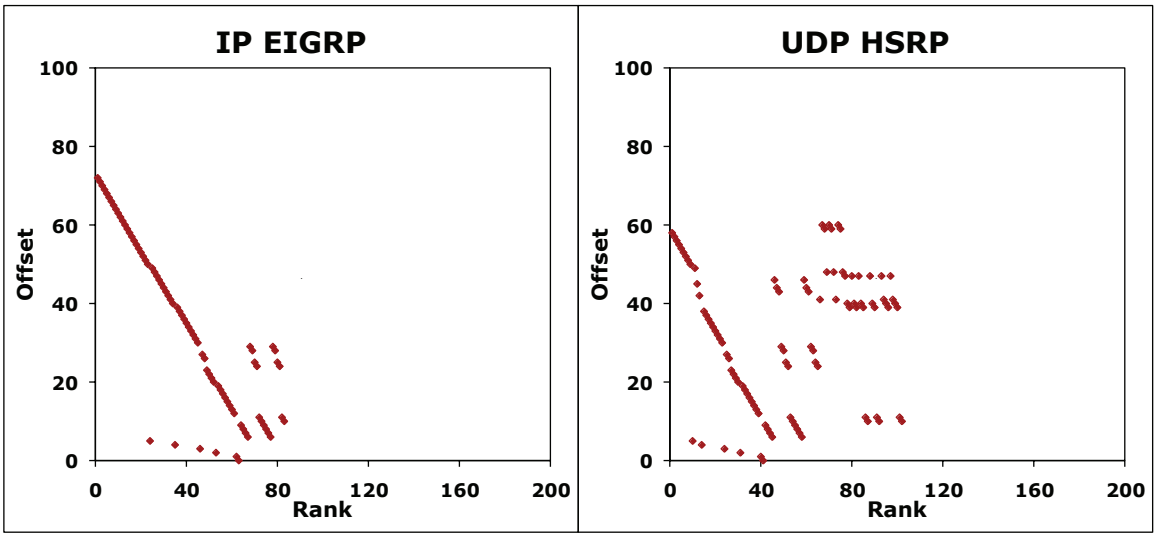


Figure 8.9. (p, n)-gram patterns in two low entropy protocols, namely: IP EIGRP, and UDP HSRP.

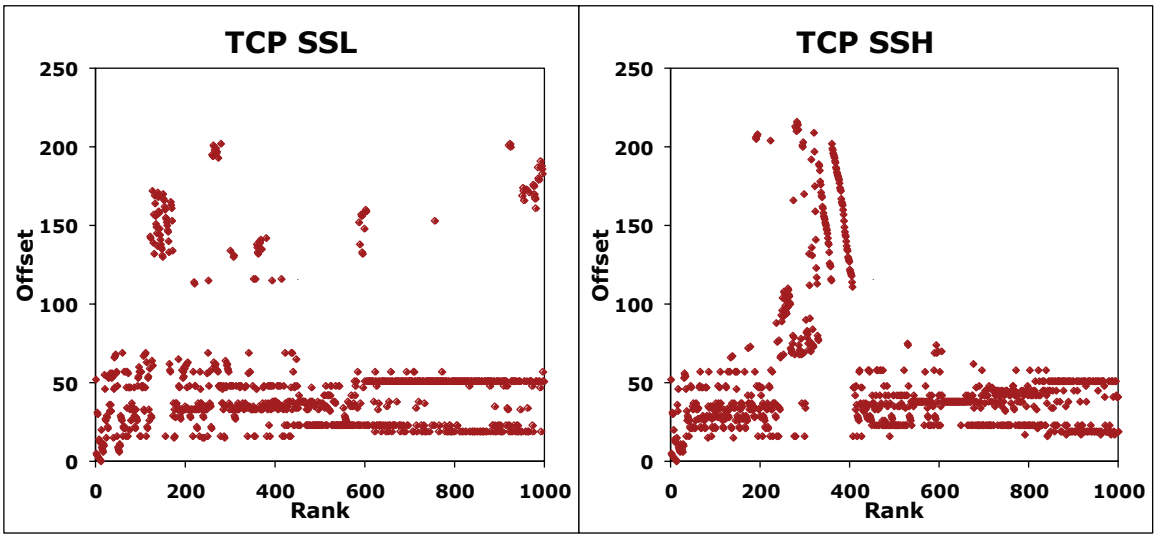


Figure 8.10. (p, n)-gram patterns in two high entropy protocols, namely: TCP SSL, and TCP SSH.

SSL encrypted protocols are in the packet header portion. However, there are still some common patterns in their packet payload portion (fields between offsets 100 and 200 for TCP SSL and TCP SSH). Some of these patterns represent the encryption algorithms negotiated between the communicating parties.

8.3.2 Frequency Distribution Behaviors

This section tests the impact of protocol design structures over (p, n)-gram frequency distributions of the corresponding protocols. Table 8.1 summarizes our empirical results after testing the (p, n)-gram frequency distribution behaviors for 22 traces of different individual protocols. In addition to the values of α and R^2, the table provides entries for average packet size, number of packets in the trace, and number of distinct (p, n)-gram instances calculated in the trace. The single-protocol traces used in these experiments were all extracted from 1-week-long traces from the CCSL and MD datasets (Appendix B.2 provides protocol names and references for all used protocol acronyms).

The examined single-protocol traces include nine traces of individual TCP protocols (i.e., HTTP, IPP, SMTP, SSH, SSL, MSMMS, MSNMS, POP, and AIM), nine UDP protocols (HSRP, NBNS, MP3-Streaming, CUPS, DNS, SIP, RIP, RTP, NBDGM), two traces of other IP protocols (i.e., ICMP, and EIGRP), and a trace of a non-IP protocol (i.e., ARP). This is in addition to a trace of header-only TCP packets that usually constitute a high percentage (about 40%) of the total number of packets in any network traffic [4, 164, 191]. In addition to the results provided by Table 8.1, Figure 8.11 plots the (p, n)-gram frequency distribution behaviors of some of these protocols on a scatter graph.

A careful look at both the figure and table shows that the majority of the distributions feature a general rapidly dropping off behavior of (p, n)-grams. The multicast or broadcast router protocols HSRP and EIGRP are two exceptions though (note their α and R^2 values). These two protocols feature low entropy levels, where content similarities span their entire packets. This implies that all (p, n)-grams at all offsets are very frequent; a feature that can be further observed in the very low number of distinct (p, n)-grams in both traces (i.e., 83 distinct (p, n)-grams in 258,756 EIGRP packets; and 102 distinct (p, n)-grams in 1,293,451 HSRP packets).

Despite the general rapidly dropping off distribution behavior, we observe that each protocol has its own specific behavior type. That is, for some protocols, α is close

	Non-IP ARP	**IP EIGRP**	**IP ICMP**	**TCP headers**	**TCP HTTP**	**TCP IPP**
α	**1.69**	**0.17**	**1.43**	**1.22**	**0.82**	**0.75**
R^2	0.93	0.27	0.94	0.91	0.88	0.87
Avg pack size	60	74	81.63	64.15	1255.2	131.28
# packets	869,565	258,756	23,963	1,888,325	684,335	486,936
# (p, n)-grams	22,790	83	170,744	1,200,566	90,671,438	1,918,481
	TCP SMTP	**TCP SSH**	**TCP SSL**	**TCP MSMMS**	**TCP MSNMS**	**TCP POP**
α	**0.72**	**1.20**	**1.29**	**1.47**	**0.81**	**0.92**
R^2	0.78	0.97	0.96	0.93	0.97	0.88
Avg pack size	1,101.89	213.61	405.83	1,116.98	179.47	346.49
# packets	25,982	238,123	275,465	97,881	3,758	116,252
# (p, n)-grams	6,259,473	25,057,270	47,326,238	57,211,409	202,919	5,332,559
	TCP AIM	**UDP HSRP**	**UDP NBNS**	**UDP MP3-Str**	**UDP CUPS**	**UDP DNS**
α	**0.77**	**0.59**	**1.51**	**1.29**	**1.35**	**0.91**
R^2	0.93	0.58	0.90	0.94	0.79	0.96
Avg pack size	323.34	62	92.14	383.71	160.5	167.66
# packets	11,079	1,293,451	176,379	248,642	128,278	66,945
# (p, n)-grams	802,373	102	241,075	29,834,449	21,778	761,937
	UDP SIP	**UDP RIP**	**UDP RTP**	**UDP NBDGM**		
α	**0.60**	**1.80**	**0.76**	**1.18**		
R^2	0.59	0.69	0.88	0.85		
Avg pack size	469.43	125.99	214	246.52		
# packets	27,395	41,538	184,270	62,493		
# (p, n)-grams	42,551	35,611	3,638,410	170,719		

Table 8.1. Power-law slope calculated for different protocols. AIM, POP, RTP, and SIP protocols were extracted from the MD Nov 1-week trace. All other protocols were extracted from the CCSL Apr 1-week trace.

to unity (e.g., POP, DNS, and SSH), whereas for others, it is not (e.g., ARP, HSRP, and RIP). Even for those that are close to unity, they are still slightly different. These behavior differences are due to the different design structures in network protocols which impact their corresponding (p, n)-gram distribution behaviors.

In other words, frequent (p, n)-grams in network traffic belong to packet fields that feature relatively low entropy levels. These fields are mainly either common header fields representing network information (e.g., MAC addresses, protocol ID, padding, etc.), or payload fields representing specific protocol design structures. Therefore,

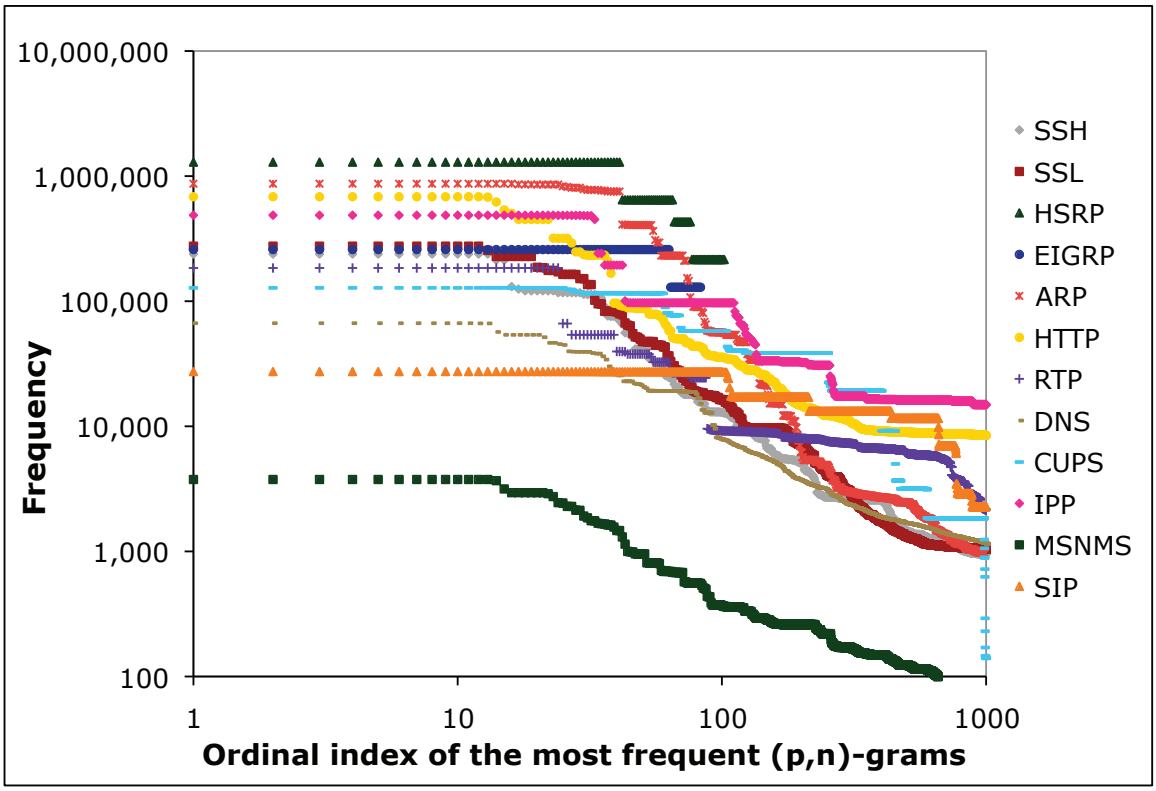


Figure 8.11. (Best viewed in color) (p, n)-gram frequency distributions for traces of individual protocol traffic. The majority of them follow a rapidly dropping off behavior, but each protocol features a specific behavior model.

changes in the protocol type and/or network topology impact the overall (p, n)-gram distributions.

For example, ARP is an Ethernet protocol whose main purpose is to map between IP addresses and their hardware MAC addresses. This type of task is best suited to short packets with very structured payloads. HTTP, on the other hand, is a TCP protocol that is mainly used to transfer Web contents, including text and media, between Internet systems. This task requires part of the packet payloads to be structured, whereas the other part needs the longest allowed Ethernet packet size in order to carry the desired contents.

Moreover, as packets of the two protocols transfer different types of contents, HTTP packets are relatively dissimilar (with higher entropy) compared to the ARP

packets. For example, in one of the experiments we performed with two single-protocol short traces of ARP and HTTP, we found 5,054 distinct (p, n)-grams in 58,725 ARP packets, as opposed to 2,427,821 distinct (p, n)-grams in 12,378 HTTP packets.

8.3.3 Discussion

It is important to note that the patterns and distribution behaviors of individual protocols discussed above are consistent under the same network topology and mode of operation (e.g., network setup, application purpose, etc., discussed in Section 6.1.2). While network topology impacts header (p, n)-grams that represent network-mapped fields, such as IP and MAC addresses, mode of operation impacts the payload (p, n)-grams used to transfer data.

These consistent protocol-dependent distribution behaviors of (p, n)-grams are what we leverage to fingerprint the different protocols in network traffic. We show how we implement this fingerprinting methodology for traffic clustering and monitoring applications in Chapters 3, 4, and 5.

Another careful look at Table 8.1 shows that for some protocols (e.g., EIGRP and HSRP), the goodness of fit of the linear regression line is relatively low (compare the results summarized in Tables 7.1 and 7.2). This further confirms that not all single-protocol traces follow power-law-like (p, n)-gram frequency distributions.

Nevertheless, the power-law-like distribution behavior observed in mixed-protocol network traffic is a natural result of two factors, namely: 1) a power-law-like distribution is the behavior featured by the majority of the individual protocols; and 2) protocols that don't follow a power-law-like distribution constitute a low percentage of the total traffic (e.g., EIGRP and HSRP in our case) which reduces their impact on the overall distribution behavior of the network traffic.

This explains what we observed in Section 7.2.1 that the values of α in mixed-protocol traces are similar but vary from trace to trace. In essence, the specific α values depend on the component protocols and their volumes within the trace.

9 Conceptual Model

The purpose of this chapter is to support our statistical evidence of the observed (p, n)-gram frequency distribution behaviors (Chapters 6, 7, and 8) in network traffic through an abstract conceptual model. That is, we build a conceptual model to explain and generalize our empirical results in the context of the current design and implementation of Internet protocols. The model serves as a formal approximation to validate our empirical observations, and ensure that they are not dataset dependent.

In particular, the model first supports the rapidly dropping off frequency distribution of (p, n)-grams in network traffic using recent Internet traffic statistics along with our definition of Shannon entropy on network packets (introduced in Section 6.2). The model then explains the power-law behavior using features of the common underlying topology of networks, as well as the common usage of Internet applications and protocols. It makes an analogy with other network models that follow power-law distributions, and uses the *rich-get-richer* rule to describe how the current implementation of Internet protocols impacts the different levels of richness in network packet fields.

9.1 Rapidly Dropping Off Frequency Distribution

The model uses recent Internet traffic statistics to compare the size of low entropy fields to the size of high entropy fields in an average-size Internet packet. Low entropy fields are those fields in the packet's header or payload that contain structural (p, n)-

grams representing protocol-specific design structures.

The model shows that the size of low entropy packet fields within an "average-size" Internet packet is much smaller than that of high entropy fields. More specifically, it shows that low entropy packet fields constitute almost 9% of the overall packet size as compared to 91% for the high entropy packet fields.

The process of comparing high and low entropy fields in Internet packets goes through two steps:

Step 1: Identify the different types of packet contents in network packets using recent statistics of Internet traffic. The different content types are distinguished by their corresponding entropy levels using our definition of Shannon entropy for network packets (introduced in Section 6.2). This step classifies packet fields into two types: high entropy fields and low entropy fields.

Step 2: Calculate an approximate average size of Internet packets using recent enterprise Internet traffic statistics. The average packet size is expressed in terms of its two components: low entropy and high entropy fields. This is then used to compare the size of the two types of packet fields.

The two steps are further explained in the following subsections.

9.1.1 Step 1: Identify the Main Different Types of Packet Contents

The purpose of this step is to identify the main types of packet contents, with respect to their entropy levels, considering the network protocols commonly found in Internet traffic. The step relies on some recent Internet statistics that give an approximate estimate of the types and volumes of these protocols. This approximation is then used to identify the types of packet contents in Internet traffic.

According to a 2008/2009 Internet study by IPOQUE [81] (a European provider of Internet traffic management solutions), P2P traffic generates the highest traffic volume in all their monitored regions (these include Europe, Africa, South America,

and Middle East) ranging from 43% to 70%. This is followed by Web usage traffic ranging from 16% to 34%, and then media streaming traffic ranging from 4% to 10% [154]. These statistics give a general approximate estimate of traffic types and volumes in Internet traffic. The actual protocol types and volumes, however, may vary for different networks[1].

Taking into account the above statistics of Internet protocols, we classify the types of their packet fields, with respect to their contents' entropy levels, into two types: low entropy fields, and high entropy fields. We use our Shannon entropy definition for network packets (introduced in Section 6.2) to further discuss the fields of these two content types, as follows:

1. **High entropy fields:** Contents of this type have a large domain of value options. The two common examples of high entropy contents are binary data and text data. Examples of binary data include encrypted data, compressed data, and binary streaming data, whereas examples of text data are Web surfing payloads, Internet text streaming, and text emails. Both of these types are mainly found in the packets' *payload* portion.

   Byte values of binary data can take any of the possible binary combinations. This gives (p, n)-grams of binary data a distribution close to the uniform distribution, where each (p, n)-gram has the same probability $(pr(x_i) = \frac{1}{2^8})$. Calculating entropy's upper bound for a 1-byte binary data gives:

$$H(X) = -\sum_{i=1}^{2^8} pr(x_i) * \log_2(pr(x_i)) = -\sum_{i=1}^{2^8} \frac{1}{2^8} * \log_2\left(\frac{1}{2^8}\right) = \log_2(2^8) = \textbf{8 bits}$$

   This means that if we consider a 1-byte field with binary data, we expect a maximum number of distinct (p, n)-grams of $2^8 = 256$ (p, n)-grams.

   On the other hand, values of text data are usually represented in one of the

[1] CCSL traces, for example, contain relatively limited P2P traffic as they were captured at a university research lab. However, the protocols making the highest volume in the CCSL traces are the Web usage protocols (e.g., HTTP and HTTPS), followed by the streaming protocols (e.g., RTP and RTSP). Table 4.1 provides a list of the protocols and their percentages.

character encoding schemes, such as, ASCII, UTF-8, etc. To simplify our entropy calculations, we assume text streaming with an ASCII representation, where each text character has an equal probability to appear.

The full ASCII table has a range that goes between 0x00 and 0xFF, and covers three types of codes: control characters (between 0x00 and 0x1F), text characters (between 0x20 and 0x7F; that is, a total of 96 text characters), and extended codes (between 0x80 and 0xFF).

This means that in a common text sequence a byte can take any of the 96 values. Thus, by assuming that all characters have the same probability, we get: $pr(x_i) = \frac{1}{96}$, provided that x_i consists of one character, and $pr(x_i) = 0$, otherwise (i.e., control characters and extended codes). Therefore, an entropy upper bound for any 1-byte text sequence is equal to:

$$\begin{aligned} H(X) &= -\sum_{i=1}^{2^8} pr(x_i) * \log_2(pr(x_i)) \\ &= -\sum_{i=1}^{2^8-96} 0 * 0 - \sum_{i=1}^{96} \frac{1}{96} * \log_2\left(\frac{1}{96}\right) = \log_2(96) \approx \textbf{6.58 bits} \end{aligned}$$

A high level of entropy within a packet field results in low frequency levels of its (p, n)-grams. (p, n)-grams with low frequency mainly constitute the (horizontal) long tail of Region **C** in the frequency distribution graph discussed in Section 6.1.1.

2. **Low entropy fields:** Contents of this type either take a fixed value or vary between a limited number of value options within network packets. Fields of this type are usually found in the packets' *header portion* or as part of protocol-specific structural fields in the packets' *payload portion*.

   Examples of this content type in the packets' header portion include ETHER type field (e.g., 0x08 0x00 in a pure IP network trace), and common packet header fields, such as total length, receive window size, protocol id, padding, etc.

Even actively changing fields, such as ports, MAC addresses, and IP addresses sometimes correspond to limited-option domains within individual networks. A common exception to this behavior is the checksum field which is usually found within the header portion but features high entropy content.

On the other hand, examples in the packets' payload portion include protocol-specific structural fields, such as URI field in CUPS packets, key_algorithms string in SSHv2 protocol, request version in the HTTP protocol, etc. These fields, however, are usually very short compared to the rest of the payload.

The entropy of (p, n)-grams in these fields depends on the number of value options and their probabilities. This number may vary from field to field, but it is usually limited by a small domain size rendering its entropy's upper bound level relatively low compared to those of binary contents.

In order to calculate entropy's lower bound for this type, we assume that all (p, n)-grams belong to a fixed value z, where $pr(x_i)$ equals 1 when $x_i = z$, and 0 otherwise. This gives:

$$\begin{aligned} H(X) &= -\sum_{i=1}^{2^8} pr(x_i) * \log_2(pr(x_i)) \\ &= -\sum_{i=1}^{2^8-1} 0 * 0 \; - \; \sum_{i=1}^{1} 1 * \log_2(1) = \log_2(1) = \mathbf{0 \text{ bits}} \end{aligned}$$

This means that if we consider a 1-byte field with fixed-value data, we expect a low entropy behavior where the minimum number of distinct (p, n)-grams that can be found is $2^0 = 1$ (p, n)-gram. This explains why the majority of (p, n)-grams in these fields feature high frequency.

A low level of entropy within a packet field results in high frequency levels of its (p, n)-grams. (p, n)-grams with high frequency mainly constitute Regions **A** and **B** in the frequency distribution graph discussed in Section 6.1.1.

9.1.2 Step 2: Compare the Sizes of Low and High Entropy Fields

This step makes an approximate comparison between the size of low entropy fields and high entropy fields in an average-size Internet packet. This comparison is one of the arguments we use later to support and explain the rapidly dropping off frequency distribution behavior of (p, n)-grams.

As was discussed in the first step, most of the packet header fields feature contents with low entropy levels, whereas most of the payload fields feature high entropy contents. On the other hand, both low entropy payload fields (i.e., fields representing protocol design structures) and high entropy header fields (e.g., checksum) are usually very short compared to the size of the other fields in the header and payload, respectively.

We simplify our calculations by ignoring those short fields at both packet portions, and assuming that header fields are low entropy in general, whereas payload fields are high entropy. This simplification allows us to make our comparison based on the size-ratio between header and payload fields in an average-size Internet packet. Taking this into account, this step starts by calculating an approximate average size of Internet packets, and then uses that to express the ratio between the two packet portions: header and payload.

Recent observations of network traffic show that packet sizes feature a bimodal distribution with two distinct modes [4, 164, 191]. While almost 50% of the packets feature the maximum allowable data size, another 40% are much shorter and are header-only packets. The sizes of the other 10% of the packets have random distribution, and are mainly dependent on the nature of the running applications [191].

The maximum length of an IP-datagram allowed by the Ethernet is defined by the Maximum Transmission Unit (MTU) parameter and is usually bounded by 1,500 bytes in most Ethernet LANs [144]. However, old implementations of TCP [145] use a maximum segment size of 576 bytes. According to a test study by Agilent [177], about 11.5% of the tested Internet traffic was packets with a maximum size of 576 bytes, whereas about 10% was packets with a maximum size of 1,500 bytes [178].

We use these statistics as an approximation to simplify our calculations. Thus, if we denote the size of the packet's headers portion by S_h and the size of the payload portion by S_p, we may describe four common types of packet sizes in Internet traffic, as follows:

1. *Header-only* packets: Those packets constitute about **40%** of the total number of packets, and consist of headers only, where the majority of them have a total size of S_h = 14 (Ethernet header) + 40 bytes (IP-datagram) = **54 bytes**.

2. *Full-size* packets (with a maximum IP-datagram size of 1,500 bytes): Those packets constitute about **10%** of the total number of packets, and consist of headers and payloads, with a total size of $S_h + S_p$ = 14 (Ethernet header) + 1,500 bytes (IP-datagram) = **1,514 bytes**.

3. *Full-size* packets (with a maximum IP-datagram size of 576 bytes): Those packets constitute about **11.5%** of the total number of packets, and consist of headers and payloads, with a total size of $S_h + S_p$ = 14 bytes (Ethernet header) + 576 bytes (IP-datagram) = **590 bytes**.

4. *Full-size* packets (with variable IP-datagram maximum sizes): Those packets constitute 50% - (11.5% + 10%) = **28.5%** of the total number of packets, and consist of headers and payloads with variable sizes. To simplify our calculations, we assume that these packet types will have an average IP-datagram size between 576 and 1,500. This gives a total packet size of 14 bytes (Ethernet header) + $\frac{(576+1,500)}{2}$ bytes (IP-datagram) = **1,052 bytes**.

5. *Other* packets: Those packets constitute **10%** of the total number of packets, and consist of headers and payloads with random sizes. Again, to simplify our calculations, we assume that the random sizes packets will have an average IP-datagram size of $\frac{1,500}{2}$. This gives a total packet size of $S_h + S_p$ = 14 bytes (Ethernet header) +$\frac{1,500}{2}$ bytes (IP-datagram) = **764 bytes**.

Using the above observations, statistics, and approximations, we now calculate an approximate average size of Internet packets as follows:

54 bytes * 40% + 1,514 bytes * 10% + 590 bytes * 11.5% + 1,052 bytes * 28.5% + 764 bytes * 10% $\approx$ **617.07 bytes**.

This means that in an average Internet packet size, the payload portion, which mainly features medium to high entropy (p, n)-grams, constitutes a high percentage of about $\frac{617.07-54}{617.07} \approx 91\%$ of the total packet size as opposed to a low percentage of 9% for the header portion, which mainly features low entropy (p, n)-grams.

Even if we assume the shortest possible sizes for the above last two types of packet sizes (i.e., 590 bytes for type 4 and 54 bytes for type 5), we get:
54 bytes * 40% + 1,514 bytes * 10% + 590 bytes * 11.5% + 590 bytes * 28.5% + 54 bytes * 10% = 414.4 bytes.

This is still a high percentage of $\approx 87\%$ for the payload portion compared to 13% for the header portion.

Finally, the average packet size, in practice, may differ from network to network depending on the specific types of network protocols (e.g., P2P, media streaming, Web traffic, etc.) dominating the network traffic and their volumes. However, even our experiments with the CCSL January 2006 dataset (Table 4.1), which features a low volume of P2P and media streaming traffic, shows an average packet size of about 364 bytes. This gives a percentage ratio of $\frac{364-54}{364} \approx 85\%$ for the payload portion as opposed to 15% for the header.

9.1.3 Conclusion

Our discussion so far has shown that the size of low entropy packet fields within an average-size network packet is much smaller than that of the high entropy packet fields. This implies that frequent (p, n)-grams (i.e., (p, n)-grams within the low entropy packet fields) are relatively few compared to infrequent ones.

In addition to their relatively small size in network packets, low entropy fields feature many fewer distinct (p, n)-grams than high entropy fields (further discussed in Section 6.2.1). That is, entropy is expressed in a logarithmic scale, and thus, a linear difference between two entropy levels implies an exponential difference in the

corresponding (p, n)-grams similarity level.

These two arguments constitute our basis to conceptually conclude the rapidly dropping off frequency distribution behavior of (p, n)-grams in Internet traffic. Again, this behavior is directly related to the inherited structure and layering design of the IP network packets, where IP packets feature encapsulated structures.

9.2 Power-Law Behavior

Our empirical analysis in Section 7.2 suggests that the distribution exhibited by (p, n)-grams in network traffic follows a power-law behavior. The purpose of this section is to conceptually explain this behavior by making an analogy with other network models that follow power-law distributions. For example, Barabasi et al. [10] explain the power-law distribution behavior in the scale-free networks model. They state that the two main features that make scale-free networks follow a power-law distribution are 1) new nodes get continuously added to the system, and 2) the system follows the "rich-get-richer" rule.

Barabasi et al. state that as a new vertex joins the system, it will connect to existing vertices with a probability that is proportional to their degrees at the time of joining. That is, the higher the degree of a vertex in the network, the higher the chance that a new node, joining the network, will connect to it and increase its degree. Similarly, Adamic et al. [1] explain the power-law distribution in the number of links a site receives. They correlate the number of links a site already has with the number of links a site receives as new sites join the Internet.

In the case of (p, n)-grams, on the other hand, the frequency distribution behavior is merely a reflection of the underlying network topology and the involved Internet protocols and applications. This can be explained by considering a system that inspects packets of network traffic for (p, n)-gram frequencies. In this system, different frequency levels of (p, n)-grams can be envisioned as different levels of richness, where a rich (p, n)-gram means a frequent (p, n)-gram coming from a low entropy field with

a high level of content repetition. This system shows the following two features:

1. *Addition of new packets*: New network packets continuously get added to the system as long as there are network activities. Each packet adds to the system

   a) Header (p, n)-grams that mainly reflect the specific network setup, topology, and parameters.

   b) Payload structural (p, n)-grams that mainly reflect the running protocols and applications within the network.

   c) Payload non-structural (p, n)-grams that mainly reflect the current data transfer for each running application.

2. *Rich-get-richer rule*: As a new packet arrives to the system, its extracted (p, n)-grams will add to the frequencies of the existing ones with a probability that is proportional to their current frequencies. This is because

   a) Common (p, n)-grams in the headers mainly represent information about active network systems (switch, server, network printer, etc.) and their parameters. Therefore, the more active the system, the more traffic it handles, and thus the more frequent those common header (p, n)-grams become.

   b) Common structural (p, n)-grams in the payloads mainly represent structural designs in the packets of active protocols or applications. Again, the more active the application, the more relevant packets are in the traffic, and thus, the more frequent those structural (p, n)-grams become.

   c) Payload non-structural (p, n)-grams mainly represent session-specific data transfers which usually differ in each packet. Therefore, (p, n)-grams from new network packets are not likely to add to the frequencies of existing ones.

As described in Section 7.2.1, there are few protocol types (e.g., broadcast and multicast protocols) that do not seem to feature the rich-get-richer phenomenon nor to follow a rapidly dropping off distribution behavior. Examples include the EIGRP

and HSRP multicasting protocols that produce almost identical packets repeatedly in the network. However, when the (p, n)-grams frequency distribution is calculated for the entire Internet traffic of a network system, the impact of these types of protocols on the overall distribution is very limited due to their relatively small volume.

10 Concluding Remarks

Our dissertation contributes to the on-going research on network traffic characterization and management. It gives a new perspective to the high-level understanding of the complex traffic through using the (p, n)-grams representation. This representation complements existing approaches with a simple yet meaningful analysis of network traffic. In this chapter, we summarize our contributions, highlight key limitations of our work, and propose some research ideas for future work.

10.1 Contributions

The research started with studying content similarities between network packets and the patterns that they may create. Our goal has been to find a way that allows for establishing a quick high-level understanding of traffic contents when inspecting an unknown network trace. Our approach was to find a representation that can 1) capture those patterns 2) efficiently enough to allow real-time traffic analysis, and 3) without making assumptions about what they look like or where in the packet they can be found while at the same time 4) give some intuition about their semantics.

Researching existing content-based analysis approaches with these requirements in mind brought us to some of the gaps that the (p, n)-grams representation can fill. Using (p, n)-grams has an efficiency advantage similar to that of using specific packet fields (such as ports and flow field) because it doesn't require looking at the entire packet to detect a pattern. At the same time, however, it has the packet-wide gen-

eralized pattern matching advantage of n-grams. (p, n)-gram-based analysis provides extra semantic meanings to the captured patterns that can distinguish between patterns in headers and payloads, and does not go through the complexity and overhead of full packet pattern matching.

This thesis is the first to research the (p, n)-grams characteristic distributions in network traffic and how that can be used to fill in some of the gaps in the current approaches of traffic analysis. Summarizing our achievements in this research work highlights the following four main contributions.

10.1.1 ADHIC for Traffic Clustering

Our first contribution was to develop ADHIC as a light-weight unsupervised traffic clustering algorithm. This work addresses our first hypothesis (Section 1.3), and shows how ADHIC can automatically discover structural patterns within network packets based on the frequency levels of their corresponding (p, n)-grams.

What makes ADHIC special is that it captures structural patterns at protocol, sub-protocol, and cross-protocol levels without assuming *a priori* knowledge of network protocols. In addition, ADHIC uses those patterns to efficiently cluster network traffic into semantically meaningful equivalence classes that closely approximate standard measures of network traffic even if packet ports were maliciously altered or obfuscated. Examples include separating IP from non-IP, TCP from UDP, email from web traffic, etc. ADHIC's hierarchical decomposition of traffic also shows semantically-based divisions within protocols such as web traffic on non-standard ports and high traffic URLs, or encrypted packets negotiating the same encryption algorithm.

Much of the structure that ADHIC typically finds would also be found through traditional analysis techniques. However, because ADHIC looks at traffic with no pre-existing biases (i.e., through frequency distributions), it also clusters using unconventional measures. For example, (p, n)-grams corresponding to special-value padding, Ethernet frame addresses, and payload contents can all be found in ADHIC decision trees. It is also common for ADHIC to cluster *control* packets with zero-length pay-

loads, such as SYN, FIN, RST, or ACK, together, away from *data* packets. Using these unconventional features allows ADHIC to be consistent in its classification of network traffic even if header data is omitted.

10.1.2 ADHIC for Traffic Monitoring

Our second contribution was to design and test ADHIC for traffic monitoring purposes. We design ADHIC to update its binary decision trees by the beginning of a pre-configured time window in order to adapt to the temporal changes in network traffic. ADHIC, in return, consistently produces new graphs of the binary tree reflecting its dynamic changes over time.

Monitoring ADHIC's graphs allows for interesting incidents to be detected, such as high bandwidth consumers (on an application, host, or network basis), repetitive network transmissions, temporal changes in network traffic, and even patterns that are related to packet sizes. In addition, the dynamically changing graphs allows to monitor network traffic for evasive protocols and unexpected behaviors in protocol types and volumes. For example, ADHIC allowed us to identify an abnormal growth or shrinkage in traffic volumes and types (e.g., P2P flash crowd) and focus the attention on a limited number of clusters. This was despite the fact that the P2P packets were obfuscated to appear as HTTP packets running over port 80.

In summary, ADHIC allows network administrators and researchers to have a different view of network traffic. This has its advantage in promptly and conveniently alerting administrators to abnormal or malicious traffic activities. Additional potential applications of ADHIC include network performance analysis, real-time alerts of flash crowds or worm activities, and dynamic DoS-resistant bandwidth management.

10.1.3 Characteristic distributions of (p, n)-grams

Our third contribution was to research the characteristic distributions of (p, n)-grams in network traffic. This work addresses our second hypothesis (Section 1.3) and shows

that (p, n)-gram frequencies in network traffic follow a power-law-like behavior where (p, n)-grams with relatively high frequency represent the short rapidly dropping off portion of the distribution curve before the long tail. These (p, n)-grams constitute the common structural patterns in network traffic which are a small subset of the total set of (p, n)-grams in the long tail.

Our conclusion of a power-law-like behavior is based on extensive empirical analysis along with a conceptual model that we build to validate our observations and ensure they are not dataset dependent. On the one hand, our empirical analysis used various traces taken from two independent network datasets to provide statistical evidence of the characteristic distributions in network traffic. The conceptual model, on the other hand, modeled (p, n)-gram variances in the different packet fields using Shannon entropy, and used that to explain and generalize the main characteristics of (p, n)-grams in the context of IP-protocol design and implementation.

The power-law-like distribution behavior of (p, n)-grams has special functional meanings and applications. In essence, it means that 1) structural patterns do exist, and that 2) they constitute a *small* subset that 3) can be easily distinguished from the other (p, n)-grams. This demonstrates that structural (p, n)-grams can be *efficiently* calculated through observing their special frequency levels in network packets *without* requiring any previous knowledge about the participating protocols or their packet structures.

Finally, since the high frequencies of structured (p, n)-grams are measured relative to all others, some efficient packet sampling can be used during traffic inspection without negatively impacting the overall frequency analysis. These efficiency advantages come in addition to the fast sub-linear pattern matching with (p, n)-grams (compared to the linear complexity of matching with n-grams).

10.1.4 Fingerprinting with (p, n)-grams

Our fourth contribution was to research the ability of (p, n)-grams to fingerprint network traffic. This work addresses our third hypothesis (Section 1.3) and shows

that structural (p, n)-grams can form a "fingerprint" of network protocols that may be used to identify them in a fashion similar to that of hand-crafted regular expression signatures.

In this capacity, our research demonstrated the ability of (p, n)-grams to capture high-level structural patterns in network traffic irrespective of flows. We show that those structural patterns are not location or type restricted and may pertain to different categories such as protocols, sub-protocols, high-volume communication flows, and frequently communicating hosts.

Again, our study used both empirical analysis and a conceptual model to test and explain the pattern-capturing and fingerprinting capabilities of (p, n)-grams in network traffic. Our empirical analysis tested various traces from two independent network datasets in order to provide statistical evidence of the (p, n)-grams' semantic representation of protocol and sub-protocol structures. We also used our entropy-based model to build entropy models for different TCP and UDP protocols. Those protocol entropy models give a visualization aid to map design structures of individual protocols to the corresponding content similarities and (p, n)-gram distributions.

The key advantages of using (p, n)-grams for fingerprinting are that 1) it can be done without assuming *a priori* knowledge about the inspected traffic and existing protocols, and that 2) it allows for unexpected (non-traditional) means to infer network protocols. For example, we observe that using (p, n)-grams in characterizing network traffic can discover payload patterns within protocols and sub-protocols that can go cross-flow in network packets.

We conclude that the special fingerprinting and characteristic distributions of (p, n)-grams are what enable applications like ADHIC to do efficient clustering and monitoring with a combination of classical and unexpected means of classification. Understanding (p, n)-grams characteristics helps in identifying what may improve ADHIC's effective clustering algorithm, and also provides a foundation for other potential uses for (p, n)-grams.

10.2 Limitations

A primary goal in researching the use of (p, n)-grams in network traffic analysis was to efficiently capture the high level semantic structure of network traffic without using domain-specific information. Both our empirical results and conceptual models suggest that the captured protocol structures have a close correlation with the semantics that are of interest to network administrators, researchers, and security officers. We find these results both promising and remarkable: (p, n)-grams representation is both simple and effective. However, these findings still come with limitations.

First, (p, n)-grams representation inherently requires structure within network packets to operate well. That is, the more encrypted packets in the inspected traffic, the fewer payload structural (p, n)-grams that can be captured. For example, although we have shown that ADHIC can often segregate encrypted and obfuscated packets, this is mainly done by recognizing other structured protocols and then assigning the remaining traffic to default clusters. Our evidence from the RMC experiments suggests that this behavior holds in larger and more complex environments.

However, the question remains as to whether this behavior will persist with the trend we see in newly evolving protocols (i.e., more encryption, compression, obfuscation, and P2P style traffic). We suspect that these encrypted packets would still contain unencrypted header fields (e.g., flags, checksums, options, paddings, etc.) as we see with some of the common encrypted traffic (e.g., SSL and SSH). These header fields may produce some identical (p, n)-grams per protocol within the same flow session. This, however, needs to be verified in future experimentation.

Second, a potential disadvantage of using (p, n)-grams may rise in the case of pattern jitters. This is where the same pattern appears at different offsets in similar packets. However, our experiments and empirical data suggest that this problem may not be noticeable as the afflicted (p, n)-grams are usually few compared to the other semantic ones in the same packets.

Third, a common problem that faces using deep packet inspection (which (p, n)-grams uses) in traffic analysis is their violation of privacy policies. We, however, *con-*

jecture that the current implementation of (p, n)-grams analysis does not raise major privacy concerns. This is because 1) (p, n)-grams usually represent short sequences of bytes scattered in the whole packet bodies and because 2) structural (p, n)-grams are solely calculated and found through their frequency distributions. This means that it is most likely that private data and user PII (Personally Identifiable Information) will not be captured as they are presumably not common in the traffic. If, however, they turn out to be common enough to be replicated in almost 5% of the traffic or more, then this may represent an area that is worth investigation.

Fourth, an ADHIC-specific limitation we have in our design of the splitting trees is that it requires a minimum volume size of each traffic type (i.e., relative to the overall traffic) in order to be clustered independently. This brings a limitation in catching stealthy attacks or protocols with relatively small volumes, when ADHIC is being considered for security monitoring purposes. We suspect that this problem can be partly addressed by considering shorter maturation window sizes. This allows frequent (p, n)-grams that only appear in a short period of time to be captured. Note, however, that this will make the analysis part more costly. It will also make ADHIC less immune to the impact of the commonly occurring network traffic spikes that don't represent a security concern.

10.3 Future work

Ultimately, our research highlights using (p, n)-grams-based network traffic analysis to complement other existing approaches and strategies. There are fundamental limitations to any approach to understanding network behavior that does not incorporate protocol-level knowledge. Knowledge-based approaches, however, will always lag the latest applications or malicious software. A generic (p, n)-grams-based approach holds the promise of revealing new patterns of behavior before they become significant problems, as well as mitigating those problems when they do occur. Thus, with the research results of (p, n)-grams characteristics we believe that further exploring

other (p, n)-grams-based approaches to extracting patterns in network behavior is a rich area for future research.

With respect to clustering with (p, n)-grams, we would like to develop a better measure of "semantically meaningful" clusters. To this point, we have verified the quality of our clusters through the use of our reference classifier and standard network analysis tools. (p, n)-grams analysis, however, finds significant patterns that these tools miss. We hope to develop additional measures, ones potentially based upon entropy minimization or other standard machine learning measures [44], that will "upper bound" the structure extraction ability of (p, n)-grams-based clustering.

Moreover, there are other ADHIC-specific algorithm enhancements and configuration settings that we would like to try for further accuracy and performance improvements. These include 1) optimizing the (p, n)-gram selection process for better entropy, 2) using multi (p, n)-grams at decision nodes, 3) using specially-seeded trees to study network behaviors, 4) experimenting with clustering based on the packet header fields only to test performance with encrypted traffic, and 5) adding a protocol identification capability to ADHIC through profiling traffic at cluster nodes.

Finally, as discussed in Chapter 2, Matrawy et al. [113] proposed using (p, n)-grams for DOS mitigation. Our work in this thesis, however, lays the conceptual and empirical foundation for using (p, n)-grams for this type of "diversity-based traffic management". A next step in this research could be to study the feasibility of mitigating such DOS damage through an adaptive bandwidth allocation scheme that we add to our clustering algorithms. This could be done by allocating equal bandwidth shares on a per-set cluster basis so that any one use of the network will be prevented from excluding other users and uses.

Appendices

A. Using Frequency Analysis in Natural Language Processing

Texts of natural languages possess a number of characteristics that can be used in the process of language identification and text categorization. In their survey, Sibun et al. [161] have listed some of these characteristics, including unique accented letters, special sequences of letters, common words, and frequent n-grams.

Several algorithms were proposed to address the language identification problem using one or more of these text-based characteristics [24, 45, 141, 61, 18]. Grothe et al. [58] made a comparative study between two common approaches for modelling natural languages based on frequency analysis, namely: word-based and n-gram-based. While the n-gram-based approach relies on frequencies of common n-grams, the word-based approach may either rely on word frequencies, or on identifying special short words that are language specific.

With their ability to efficiently capture specific language semantics, n-grams have been successfully used in the areas of natural language identification, text categorization, and subject classification [78, 12]. Our research takes advantage of the common research similarities between using n-grams for natural language processing and using (p, n)-grams for network traffic analysis.

When applied to network traffic, however, n-grams can't capture network protocol semantics. To compensate for that, our research uses (p, n)-grams instead. Offset p in (p, n)-grams substitutes the missing built-in semantic meaning that n-grams feature in natural languages.

This section presents how frequency analysis of words and n-grams is used in the

process of natural language processing. It discusses the n-grams' functionality and efficiency features in natural language processing as a template to our (p, n)-gram-based approach.

A.1 Advantages of using Frequency Analysis

Using words' frequency analysis in natural language processing comes with two main characteristics. The first characteristic is their ability to capture specific language semantics. That is, the most common words in a document can identify the language and subject types of a document [24]. For instance, given any article, the set of the most frequent words is highly correlated with the article's language type (e.g., "of" and "the" for English vs. "de" and "la" for French). Moreover, the set of the second most frequent words is more correlated with the article's subject (e.g., "atom" and "molecule" for chemistry vs. "cell" and "plasma" for biology).

The second characteristic of using word's frequency analysis in natural language processing is their usage efficiency. That is, the same words that can represent the language and subject of a document are very few compared to the rest of the words. This characteristic is better described by Zipf's law [203], in which George Zipf [149] found a special power-law relationship between frequencies of English words and their ranks. Zipf's law states that the frequency of any word in a corpora is inversely proportional to its rank:

$$f_r = f_1 * r^{-1} \tag{A.1}$$

That is, the most frequent word in an English corpora appears twice as often as the second most frequent word, and thrice as often as the third most frequent word, etc.

$$f_1 = 2 * f_2 = 3 * f_3 = ... = r * f_r = (r + 1) * f_{r+1} \tag{A.2}$$

where: r = rank, and f_n = frequency of the r^{th} most frequent word in the corpora.

Zipf's law implies that there are few words that are very common, whereas majority of the words are infrequent. In other words, in a given article, there are very few words that are 1) very common, 2) easily distinguishable from the rest, and 3) they represent the language and subject types of the article. Zipf's law is further discussed in Section A.2.

The two characteristics of words in natural languages allow for language identification and text categorization applications. These characteristics explain their effective functionality, and their efficiency in terms of reducing the required space and computation complexities.

Since n-grams constitute inflection forms or morpheme components of the full words in natural languages, the same characteristics of words extend to n-grams [24]. This gives n-gram-based applications the same functionality and efficiency advantages in the process of natural language identification and text categorization.

A.2 Language Identification and Text Categorization using n-grams

The process of identifying an unknown document's language, using n-gram or word frequencies, typically goes through the following steps [24]: First, all possible natural languages are profiled and modelled using their most frequent words or n-grams. Second, the unknown document is profiled using its most frequent words or n-grams. Third, the unknown document's profile is checked against all the previously calculated language profiles using a similarity distance function. Thus, the language profile with shortest distance determines the language type of the unknown document.

Two parameters are to be set before the identification process: 1) the similarity distance function, and 2) the number of n-grams or words to be considered in profiling [11]. Several distance functions were proposed to measure similarities [58], including: ranking order [24], relative entropy [161], Bayesian decision rule [45], vector space model [37, 141], and Monte Carlo sampling [140]. Depending on the distance

function used, the number of n-grams needed to be considered for high performance profiling varies from one function to another. For example, 400 n-grams work well using the ranking order function.

In spite of their similar functionality and efficiency features, many research studies have found n-gram-based approaches to be more advantageous for language identification than word-based ones [37]. This finding can be explained by more than one reason. First, misspelling in a long word affects the entire word, but may only impact a small number of shorter n-grams. Misspelling errors may come from various reasons, such as: erroneous data entry, and scanning and OCR (Optical Character Recognition) problems. Second, n-grams provides more flexibility while dealing with stream text, due to their short fixed size [112]. Third, unlike whole words, n-grams achieve automatic word stemming results when considering different words that share the same root (e.g., 'work', 'working', 'worked', 'works', etc. share the same n-gram root: 'work') [24].

Cavnar et al. [24] introduced an accurate, yet efficient, n-gram-based approach for language identification and subject classification. Their approach uses a simple similarity distance function that is based on the ranking order of most frequent n-grams. In spite of its simple and fast implementation, the approach achieves an accuracy level of 99.8% of text characterization and language identification when applied on Usenet newsgroup articles in different languages and different subjects.

As a special case, Vega et al. [152] used weighted n-grams with size $n = 3$ to check if the document is written in a specific language (Indonesian in this case). This strategy is useful when the inspected languages don't have enough vocabulary differences. For example, Malay and Indonesian share almost 80% of their vocabularies.

Damashek [37] used n-grams for spelling and error corrections, text compression, and text search and retrieval. His research demonstrates how n-grams are useful in categorizing text in a non-restricted multilingual environment. Damashek introduced Acquaintance; an approach that uses n-grams along with another vector space technique. Acquaintance gives a similarity measure that can work with a large collection

of documents in a non-restricted range of topics without requiring *a priori* knowledge of the document's content or language.

Martins et al. [112] used n-grams to identify the language used in Internet web pages. Online text might differ from that in document collections in more than one way. For example, text in web pages usually contains more spelling errors, and may feature multiple languages in the same page. Moreover, hyperlinks are usually displayed as part of the text in the online documents. In their experiments, Martins et al. achieved accurate results by using a heuristic-based n-gram algorithm along with some proper similarity measures.

A similar work was done by Baykan et al. [11] who tried to identify the language of Internet web pages using their URL addresses only. They used n-grams with size $n = 3$ along with other methods. Due to the short size of URL addresses, better results were achieved when custom-made features were used, like: country code, number of hyphens, and dictionary with city names. Those extra features were most useful when non-English web pages use URLs with English-looking words.

B. Power-Law Distributions

A power-law frequency distribution describes a distribution where there are *few frequent* incidents, and *many* infrequent ones. Power-law distributions are found in many phenomena in physics and economics [143], where they appear ubiquitously in various fields [99]. An example is the distribution of city populations in a country. There are few major cities in any country compared to many small towns. Other examples include people income, earthquake levels [129], and company sizes in a country [7].

Simply put, a power law is a polynomial relationship between two entities x and y, such that:

$$y = P(x) = cx^{-\alpha} \tag{B.3}$$

where: c is a constant, and α is called the power exponent. Taking the log on both sides gives:

$$log(y) = log(cx^{-\alpha}) = log(c) + log(x^{-\alpha}) = c' + (-\alpha)log(x) \tag{B.4}$$

Drawing the power-law function (B.4) on a log-log scale gives a straight line like in Figure B.1, where -α is the slope and c' is the intercept.

Power-law distributions also exist in computer and network related systems. Huberman et al. [70] found that power-law distributions apply to the number of pages found in a Website. In computer networks, Barabasi et al. [10] found that vertex connectivities (degrees) follow a power-law distribution. Albert et al. [3] generalized this power-law model to the World Wide Web, where vertices represent documents

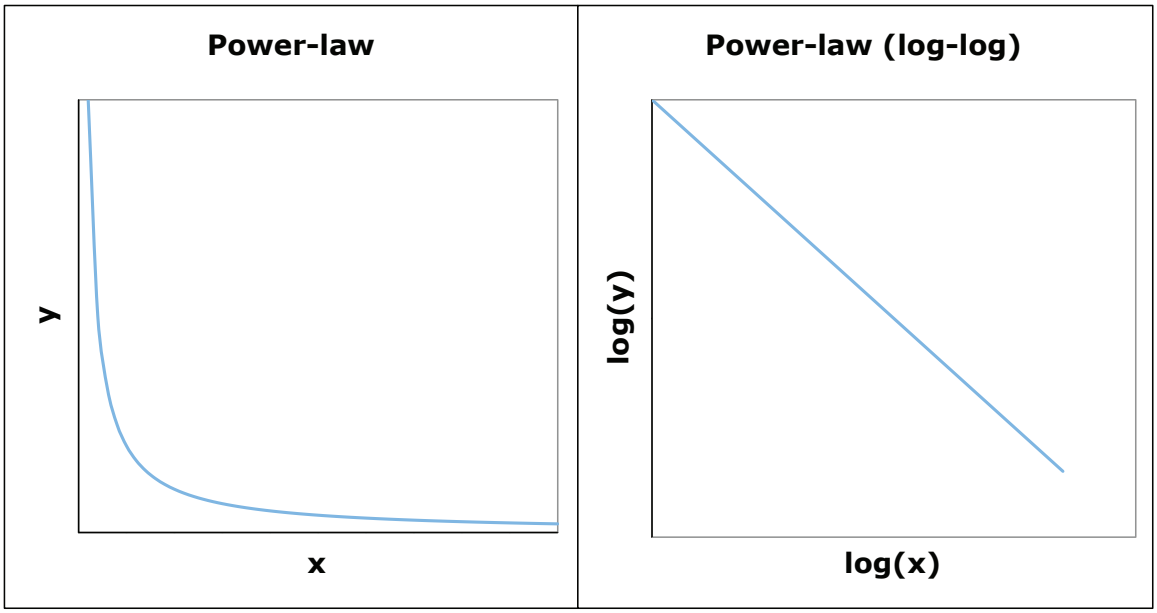


Figure B.1. Power law on a normal scale (left) and a log-log scale (right). Note the power law's straight line on the log-log graph, where the slope equals the negative of the power exponent α (e.g., in this graph, $\alpha = 1$, and $slope = -1$).

and edges represent hyperlinks.

Power laws have other interesting behavioral features. Their statistical relationships do not change at different measurement scales. This is usually referenced as "*scale-free*" distribution [129]. In addition, power laws feature smooth curves that usually has its impact on the system operational expectations and frequency thresholds [24]. These features and others of power laws have brought a special interest among researchers to further study power laws, their types and applications [25].

B.1 Zipf's Law

Power-law relationships take different forms depending on the exponent's value. Zipf's law [203] is a special form where the exponent is equal to unity (i.e., 1). Zipf's law describes the frequency distribution of words in texts of natural languages. It states that the frequency f of the r-th word (ordered by their frequency rank) in an English

corpus is inversely proportional to its rank r, thus:

$$f = cx^{-\alpha} \tag{B.5}$$

where c is a constant, and α is close to unity [30, 129].

To give a real-life example of this relationship, we experimented with the frequency list [92] extracted by Adam Kilgarriff from the British National Corpus [49]. This list contains about 1,000,000 distinct words taken from a corpus of about 100,000,000 words. Our test shows that the most frequent word reported in the corpus was "the" with frequency of 6,187,267.

The second and third most frequent words were "of", and "and" with frequencies of 2,941,444, and 2,682,863 respectively. Notice that the frequency of occurrence of "the" is approximately twice as often as the frequency of "of" and three times as often as the frequency of "and". In this experiment, it is evident how the very high frequent words (e.g., "of", "the", "is", "if", etc.) are few in number, whereas the infrequent ones (e.g., "parachute", "optimum", "navigating", etc.) constitute the majority of the English language words.

We also did the same experiment with a French corpus. We used the frequency list [184] extracted by Jean Veronis from the Monde Diplomatique 1987-1997 [53]. This list contains over 150,000 distinct words taken from a corpus of 11,139,376 words. Our experiment shows the same Zipf's law distribution behavior.

Figure B.2 shows the frequency/rank graph for these English and French corpora. Note how the most frequent 1,000 words adhere to Zipf's law for both corpora with a slope very close to unity (-0.999 for English corpus, and -1.016 for French) along with a good model fitness calculated by the *coefficient of determination* R^2 [42], where R^2 takes a value between 0.00 and 1.00, with 1.0 indicating a perfect fit. We further discuss R^2 and how to interpret it in Section 7.1.

Zipf's law was also found to apply to other languages (e.g., Chinese language) [59], and to n-grams as well [24]. In addition, Quan et al. [59] reported that when Zipf's law is tested on huge language corpora, the slope behavior starts to deviate from

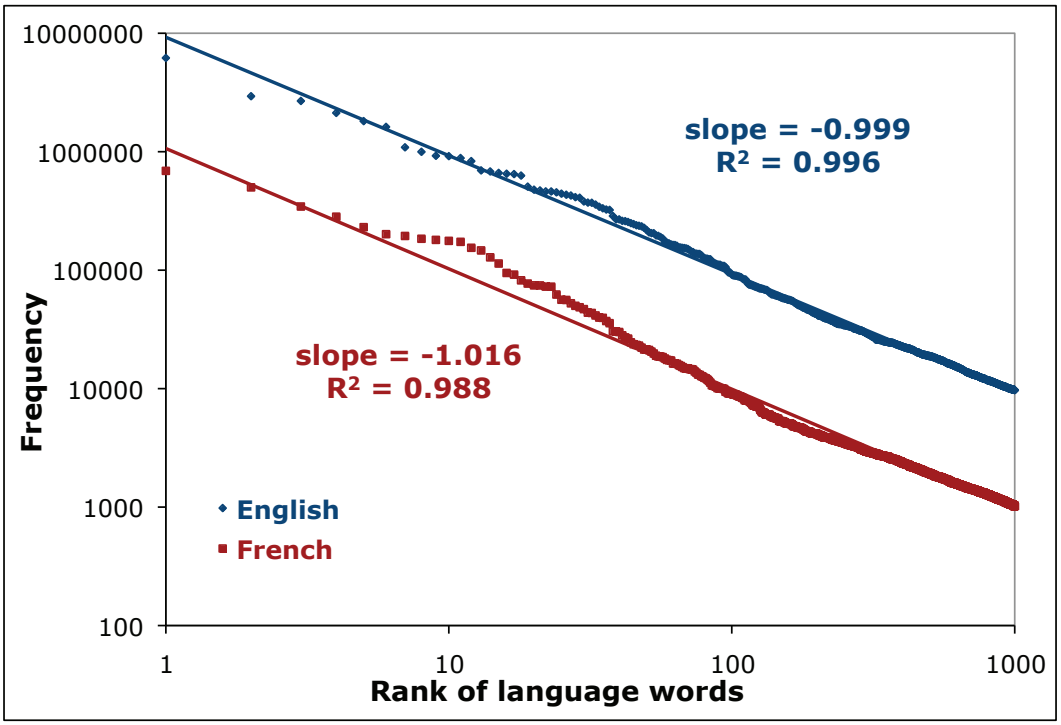


Figure B.2. Zipf's law for the English and French corpora (first 1,000 entries).

unity (i.e., $\alpha = 1$) at high word ranks. In particular, Zipf's law was found to best apply to the first 5,000 English and French words and first 1,000 Chinese words.

A power-law distribution is said to precisely follow Zipf's law when the power exponent is strictly equal to 1. The term "*Zipf-like*" distribution, on the other hand, describes a power-law distribution where the value of the power exponent is close to 1, but varies from trace to trace [19]. Examples of Zipf-like distributions include those observed in Web sites' page hits [71], and Web caching [19].

B.2 Power-Laws: From Observations to Applications

What makes a system follow a power-law distribution? Although there is no definite answer, there might be more than one way to explain the basis of this distribution behavior. One of the common explanations is the "*rich-get-richer*" rule [47], also known as "*preferential attachment*" [10]. For instance, in the power-law city population model, the bigger the city, the higher the chance that more people will join the

city (e.g., new comers, newly born babies, etc.).

In the scale free network problem, the rich-get-richer rule can be observed during the continuous expansion of the network. That is, newly added vertices are usually attached to others that are already well-connected in the system. In other words, the probability that an old vertex gets connected with a newly added one is directly proportional to the old vertex's degree.

Understanding the power-law behavior of a system potentially has useful applications. Mitzenmacher [117] emphasized that research on power law has to move from observation, modelling and interpretation to validation and application. For example, relying on the findings of Barabasi et al. [10] about the power law distribution of vertex degrees in a system, Balthrop et al. [9] suggested an effective way to stop the spread of computer viruses through targeting highest degree vertices for immunization.

C. IP Packet Structure

ETHER		
0 – 5:		6 bytes: Destination MAC Address
6 – 11:		6 bytes: Source MAC Address
12 – 13:		2 bytes: Type (usually: 0x08,0x00, i.e., IP)
IP		
14:		1 byte: IP version (4 bits: e.g., 0x4: IPv4) + Length (4 bits: e.g., 0x5: 20 bytes)
15:		1 byte: Type of Service
16 – 17:		2 bytes: Total Length
18 – 19:		2 bytes: Identification (aid in assembling the fragments of a datagram)
20 – 21:		2 bytes: Flags (3 bits) + Fragment Offset (13 bits)
22:		1 byte: TimeToLive
23:		1 byte: Protocol (ICMP:0x01, IGMP:0x02, TCP: 0x06, UDP: 0x11)
24 – 25:		2 bytes: Header Checksum
26 – 29:		4 bytes: Source IP Address
30 – 33:		4 bytes: Destination IP Address
34 – 36:		3 bytes: Options (optional)
37:		1 byte: Padding (optional)
TCP (*1st column if "options" and "padding" were not used in the IP header; 2nd column otherwise*)		
34 – 35:	38 – 39:	2 bytes: Source Port
36 – 37:	40 – 41:	2 bytes: Destination Port
38 – 41:	42 – 45:	4 bytes: Sequence Number
42 – 45:	46 – 49:	4 bytes: Acknowledgement Number
46:	50:	1 byte: Header Length (4 bits e.g., 0x8: 8 words) + Reserved (4 bits: set to 0)
47:	51:	1 byte: Flags (CWR, ECE, URG, ACK, PSH, RST, SYN, FIN)
48 – 49:	52 – 53:	2 bytes: Receive Window Size (e.g., 0x01F5 for 501)
50 – 51:	54 – 55:	2 bytes: Checksum
52 – 53:	56 – 57:	2 bytes: Urgent Pointer (usually set to 0s if URG is not set)
54 – 73:	58 – 77:	20 bytes: Options (optional)
74 – ...:	78 – ...:	**Data** (this field contains the Application header, if any)
UDP (*1st column if the "options" and "padding" were not used in the IP header; 2nd column otherwise*)		
34 – 35:	38 – 39:	2 bytes: Source Port
36 – 37:	40 – 41:	2 bytes: Destination Port
38 – 39:	42 – 43:	2 bytes: Length
40 – 41:	44 – 45:	2 bytes: Checksum
42 – ...:	46 – ...:	**Data** (this field contains the Application header, if any)

Table C.1. IP Packet Structure

D. Protocol References

Acronym	Protocol Name	Reference
IPv4	Internet Protocol version 4	[84]
TCP	Transfer Control Protocol	[176]
MS WBT/RDP	Microsoft Remote Display Protocol	[122]
IPP	Internet Printing Protocol	[82]
IMAP	Internet Message Access Protocol	[76]
IMAPS	IMAP over TLS	[77]
HTTP	Hypertext Transfer Protocol	[68]
HTTPS	HTTP over TLS	[69]
SSH	Secure Shell	[171]
RTSP	Real Time Streaming Protocol	[151]
MYSQL	MYSQL Protocol	[124]
SMB	Server Message Block	[166]
MSNMS	Microsoft Network Messenger Service	[121]
XMPP	Extensible Messaging and Presence Protocol	[169]
TCP Sophos	*Anti-virus application packets*	[198]
URD	URL Rendezvous Directory for SSM	[183]
TCP No Payload	*TCP (headers only) control packets*	
NBSS	NetBIOS Session Service	[128]
Bit Torrent	Bit Torrent Protocol	[16]
IRC	Internet Relay Chat Protocol	[87]
NNTP	Network News Transfer Protocol	[131]
TELNET	TELNET Protocol	[180]
FTP	File Transfer Protocol	[54]
SMTP	Simple Mail Transfer Protocol	[167]
CVS	Concurrent Versions System	[35]
POP	Post Office Protocol	[138]
AIM	AOL Instant Messenger	[2]
UDP	User Datagram Protocol	[182]
DNS	Domain Name Service	[41]
CUPS	Common UNIX Printing System	[34]
IPSec	Internet Protocol Security Protocol	[83]
WHO	Messages Produced by the Unix WHO Command	[189]
XDMCP	X Display Manager Control Protocol	[196]
RTP	Real-time Transport Protocol	[150]
MS SQL	Microsoft SQL Protocol	[123]
NBDGM	NetBIOS Datagram Service	[126]
DCE_RPC	Distributed Computing Environment/Remote Procedure Calls	[38]
Bit Torrent	Bit Torrent Protocol	[16]
MDNS	Multicast Domain Name Service	[116]
Ganglia	*Distributed Monitoring System*	[55]
NBNS	NetBIOS Name Service	[127]
RIPv1	Routing Information Protocol	[146]
HSRP	Hot Standby Router Protocol	[67]
DHCP	Dynamic Host Configuration Protocol	[40]
SNMP	Single Network Management Protocol	[168]
NTP	Network Time Protocol	[132]
SRVLOC	Service Location Protocol	[170]
SIP	Session Initiation Protocol	[165]
ICMP	Internet Control Message Protocol	[73]
IGMP	Internet Group Management Protocol	[74]
EIGRP	Enhanced Interior Gateway Routing Protocol	[48]
ARP	Address Resolution Protocol	[6]
RARP	Reverse Address Resolution Protocol	[142]
IPX	Internet Packet Exchange	[86]
IPv6	Internet Protocol version 6	[85]
STP	*Spanning Tree Protocol*	[173]
DTP	*Dynamic Trunking Protocol*	[43]

Table D.1. Protocol References

References

[1] L. Adamic, B. Huberman, A. L. Barabasi, R. Albert, H. Jeong, and G. Bianconi. Power-Law Distribution of the World Wide Web. *Science*, 287(5461):2115a–, 2000.

[2] AIM. Aol instant messenger. http://dashboard.aim.com/aim.

[3] R. Albert, H. Jeong, and A. Barabasi. Internet: Diameter of the world-wide web. *Nature*, 401:130–131, 9 September 1999.

[4] H. Anderson. *Fixed Broadband Wireless System Design*, page 338. Wiley, 2003.

[5] R. Antonello, S. Fernandes, D. Sadok, J. Kelner, and G. Szabo. Deterministic finite automaton for scalable traffic identification: The power of compressing by range. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 155–162, 2012.

[6] ARP. Address resolution protocol. http://www.rfc-editor.org/rfc/rfc826.txt.

[7] R. Axtell. Zipf distribution of u.s. firm sizes. *Science Magazine*, 293:1818–1820, 7 September 2001.

[8] C. Bacquet, A. Zincir-Heywood, and M. Heywood. Genetic optimization and hierarchical clustering applied to encrypted traffic identification. In *Computational Intelligence in Cyber Security (CICS), 2011 IEEE Symposium on*, pages 194–201, 2011.

[9] J. Balthrop, S. Forrest, M. E. J. Newman, and M. Williamson. Technological networks and the spread of computer viruses. *Science Magazine*, 304:527–529, 23 April 2004.

[10] A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science Magazine*, 286:509–512, 15 October 1999.

[11] E. Baykan, M. Henzinger, and I. Weber. Web page language identification based on urls. In *Proceedings of the VLDB Endowment*, Auckland, New Zealand, 2008.

[12] K. R. Beesley. Language identifier: A computer program for automatic natural-language identification on on-line text. In *Proceedings of the 29th Annual Conference of the American Translators Association*, pages 47–54, Seatle, Washington, 1998.

[13] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian. Traffic classification on the fly. *SIGCOMM Comput. Commun. Rev.*, 36(2):23–26, 2006.

[14] L. Bernaille, R. Teixeira, and K. Salamatian. Early application identification. In *Proceedings of CONEXT*, 2006.

[15] R. Beverly. A robust classifier for passive tcp/ip fingerprinting. In *Passive and Active Network Measurement*, volume 3015 of *Lecture Notes in Computer Science*, pages 158–167. Springer Berlin Heidelberg, 2004.

[16] bittorrent.org. Bittorrent protocol specification. http://www.bittorrent.org.

[17] K. Borders and A. Prakash. Web tap: Detecting covert web traffic. In *In Proceedings of the 11th ACM Conference on Computer and Communication Security*, pages 110–120, 2004.

[18] G. Botha, V. Zimu, and E. Barnard. Text-based language identification for the south african languages. In *Proceedings of the17th Annual Symposium of the Pattern Recognition Association of South Africa*, 2007.

[19] L. Breslau, P. Cue, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *In INFOCOM*, pages 126–134, 1999.

[20] C. Brown, A. Cowperthwaite, and A. Hijazi. Analysis of the 1999 darpa/lincoln laboratory ids evaluation data with netadhict. In *Proceedings of the IEEE Second Symposium on Computational Intelligence for Security and Defense Applications*, CISDA '09, Ottawa, Canada, July 2009.

[21] J. Caballero, S. Venkataraman, P. Poosankam, M. Kang, D. Song, and A. Blum. Fig: Automatic fingerprint generation. In *Proc. 14th Ann. Network and Distributed System Security Symp. (NDSS)*, 2007.

[22] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson. Touching from a distance: website fingerprinting attacks and defenses. In *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12, pages 605–616, New York, NY, USA, 2012. ACM.

[23] CAIDA. The cooperative association for internet data analysis. http://www.caida.org.

[24] W. Cavnar and J. Trenkle. N-gram-based text categorization. In *Proceedings of the 1994 Symposium on Document Analysis and Info Retrieval (SDAIR)*, pages 161–175, Las Vegas, NV, USA, 1994.

[25] N. Chater and G. Brown. Scale-invariance as a unifying psychological principle. *Elsevier Science*, 69(3):B17–B24, 1999.

[26] T. Choi, C. Kim, S. Yoon, J. Park, B. Lee, H. Kim, and H. Chung. Content-aware internet application traffic measurement and analysis. In *Proceedings of IEEE/IFIP NOMS*, April 2004.

[27] B. Chun, J. Lee, H. Weatherspoon, and B. N. Chun. Netbait: a distributed worm detection service. Technical report, Intel Research, 2002.

[28] Cisco. Cisco ios netflow. www.cisco.com/web/go/netflow.

[29] A. Clauset, C. Shalizi, and M. Newman. Power-law distributions in empirical data. http://www.santafe.edu/~aaronc/powerlaws/.

[30] A. Clauset, C. Shalizi, and M. Newman. Power-law distributions in empirical data. *E-print: arXiv:0706.1062v1*, 7 June 2007.

[31] G. Combs et al. Wireshark. http://www.wireshark.org, 2007.

[32] CoralReef. Traffic analysis tool by caida. http://www.caida.org/tools/measurement/coralreef.

[33] C. Cunha, A. Bestavros, and M. Crovella. Characteristics of www client-based traces. Technical report, Boston University, 1995.

[34] CUPS. Common unix printing system. http://www.cups.org/documentation.php.

[35] CVS. Concurrent versions system. http://www.nongnu.org/cvs/.

[36] A. Dainotti, A. Pescape, and K. Claffy. Issues and future directions in traffic classification. *Network, IEEE*, 26(1):35–40, 2012.

[37] M. Damashek. Gauging similarity with n-grams: Language independent categorization of text. *Science Magazine*, 267:843–848, 10 February 1995.

[38] DCE-RPC. Distributed computing environment - remote procedure calls. http://www.samba-tng.org/docs/tng-arch/tng-arch05.html.

[39] F. Dehghani, N. Movahhedinia, M. Khayyambashi, and S. Kianian. Real-time traffic classification based on statistical and payload content features. In *Intelligent Systems and Applications (ISA), 2010 2nd International Workshop on*, pages 1–4, 2010.

[40] DHCP. Dynamic host configuration protocol. http://www.ietf.org/rfc/rfc2131.txt.

[41] DNS. Domain name service. http://www.ietf.org/rfc/rfc1035.txt.

[42] N. Draper and H. Smith. *Applied Regression Analysis*, page 245. Wiley-Interscience, 1998.

[43] DTP. Dynamic trunking protocol. http://www.cisco.com.

[44] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification, 2nd ed.*, chapter Unsupervised Learning and Clustering. Wiley, 2001.

[45] T. Dunning. Statistical identification of language. Technical report, New Mexico State University, 1994.

[46] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli. Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting. *Computer Networks*, 53(1):81 – 97, 2009.

[47] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press, 2010.

[48] EIGRP. Enhanced interior gateway routing protocol. http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/en_igrp.htm.

[49] EngCorpus. British national corpus. http://www.natcorp.ox.ac.uk/.

[50] J. Erman, M. Arlitt, and A. Mahanti. Traffic classification using clustering algorithms. In *Proceedings of ACM SIGCOMM MineNet Workshop*, September 2006.

[51] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson. Offline/realtime traffic calssification using semi-supervised learning. In *In IFIP Perfermance*, October 2007.

[52] C. Estan, S. Savage, and G. Varghese. Automatically inferring patterns of resource consumption in network traffic. In *Proceedings of ACM SIGCOMM*, 2003.

[53] FreCorpus. Monde diplomatique 1987-1997. http://www-a2k.is.tokushima-u.ac.jp/member/kita/NLP/lex.html.

[54] FTP. File transfer protocol. http://www.ietf.org/rfc/rfc0959.txt.

[55] Ganglia. Distributed monitoring system. http://ganglia.sourceforge.net/.

[56] M. Gebski, A. Penev, and R. K. Wong. Protocol identification of encrypted network traffic. In *IEEE / WIC / ACM International Conference on Web Intelligence (WI 2006)*, Hong Kong, China, 2006.

[57] X. Gong, N. Kiyavash, and N. Borisov. Fingerprinting websites using remote traffic analysis. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 684–686, New York, NY, USA, 2010. ACM.

[58] L. Grothe, E. D. Luca, and A. Nurnberger. A comparative study on language identification methods. In *Proceedings of teh 6th International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, 2008.

[59] L. Q. Ha, P. Hanna, J. Ming, and F. Smith. Extending zipfi£¡s law to n-grams for large english and chinese corpora. In *Proceedings of International Conference Cognitive Modeling in Linguistics*, Sofia, Bulgaria, 2007.

[60] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. Acas: automated construction of application signatures. In *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, MineNet '05, pages 197–202, New York, NY, USA, 2005. ACM.

[61] J. Hakkinen and J. Tian. n-gram and decision tree based language identification for written words. In *Proceedings of workshop on Automatic Speech Recognition and Understanding (ASRU '01)*, 2001.

[62] T. Hastie, R. Tibshirani, and J. Friedman. *Hierarchical Clustering*, pages 520–528. Springer, 2009.

[63] A. Hijazi, H. Inoue, A. Matrawy, P. van Oorschot, and A. Somayaji. Towards understanding network traffic through whole packet analysis. Technical Report TR-07-06, Carleton University, 2007.

[64] A. Hijazi, H. Inoue, A. Matrawy, P. van Oorschot, and A. Somayaji. Discovering packet structure through lightweight hierarchical clustering. In *Proceedings of IEEE International Conference on Communications (ICC'08)*, Beijing, Chiina, 2008.

[65] A. Hijazi, H. Inoue, and A. Somayaji. Lightweight unsupervised hierarchical network traffic clustering. In *Workshop on Machine Learning in Adversarial Environments for Computer Security (NIPS'07)*, Whistler, BC, Canada, 2007.

[66] T. Hill and P. Lewicki. Statistics methods and applications. http://www.statsoft.com/textbook/, StatSoft, Tulsa, OK, 2007.

[67] HSRP. Hot standby router protocol. http://www.ietf.org/rfc/rfc2281.txt.

[68] HTTP. Hypertext transfer protocol. http://www.ietf.org/rfc/rfc2616.txt.

[69] HTTPS. Http over tls. http://www.ietf.org/rfc/rfc2818.txt.

[70] B. Huberman and L. Adamic. Internet: Growth dynamics of the world-wide web. *Nature*, 401:131–132, 9 September 1999.

[71] B. Huberman, P. Pirolli, J. Pitkow, and R. Lukose. Strong regularities in world wide web surfing. *Science Magazine*, 280:95–95, 3 April 1998.

[72] IANA. Internet assigned numbers authority. http://www.iana.org.

[73] ICMP. Internet control message protocol. http://www.ietf.org/rfc/rfc792.txt.

[74] ICMP. Internet group management protocol. http://www.ietf.org/rfc/rfc1112.txt.

[75] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese. Network monitoring using traffic dispersion graphs. In *ACM Internet Measurement Conference (IMC'07)*, 2007.

[76] IMAP. Internet message access protocol. http://www.ietf.org/rfc/rfc2060.txt.

[77] IMAPS. Imap over tls. http://tools.ietf.org/html/rfc2595.

[78] N. Ingle. A language identification table. *The Incorporated Linguist*, 15(4):98–101, 1976.

[79] H. Inoue, A. Hijazi, and D. Jansens. Netadhict. http://www.ccsl.carleton.ca/software.

[80] H. Inoue, D. Jansens, A. Hijazi, and A. Somayaji. NetADHICT: A tool for understanding network traffic. In *Proceedings of the USENIX 21st Large Installation System Administration Conference (LISA'07)*, Dallas, TX, USA, November 2007.

[81] IPOQUE. Ipoque. http://www.ipoque.com/.

[82] IPP. Internet printing protocol. http://www.ietf.org/rfc/rfc2911.txt.

[83] IPSec. Internet protocol security protocol. http://rfc.net/rfc2401.html.

[84] IPv4. Internet protocol version 4. http://www.ietf.org/rfc/rfc0791.txt.

[85] IPv6. Internet protocol version 6. http://www.ietf.org/rfc/rfc2373.txt.

[86] IPX. Internet packet exchange. http://www.apps.ietf.org/rfc/rfc1132.html.

[87] IRC. Internet relay chat protocol. http://www.ietf.org/rfc/rfc1459.txt?number=1459.

[88] A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[89] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos. Is p2p dying or just hiding? In I. C. S. Press, editor, *Proceedings of IEEE GLOBECOM*, Dallas, Texas, November 2004.

[90] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy. Transport layer identification of p2p traffic. In *ACM Internet Measurement Conference (IMC'04)*, Taormina, Sicily, Italy, 2004.

[91] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: multilevel traffic classification in the dark. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '05, pages 229–240, New York, NY, USA, 2005. ACM.

[92] A. Kilgarriff. Frequency list of the british national corpus. http://www.kilgarriff.co.uk/bnc-readme.html.

[93] H. Kim and B. Karp. Autograph: Toward Automated, Distributed Worm Signature Detection. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[94] G. Klass. *Just Plain Data Analysis: Finding*. Rowman and Littlefield Publishers, 2008.

[95] C. Kreibich and J. Crowcroft. Honeycomb - Creating Intrusion Detection Signatures Using Honeypots. In *Proceedings of HOTNETS-II*, 2003.

[96] P. Kumpulainen, K. HÃďtÃűnen, O. Knuuti, and T. Alapaholuoma. Internet traffic clustering using packet header information. In *Proceedings of the 14th Joint International IMEKO TC1+TC7+TC13 Symposium*, 2011.

[97] W. Leland, M. Taqq, W. Willinger, and D. Wilson. On the self-similar nature of Ethernet traffic. In *ACM SIGCOMM*, pages 183–193, 1993.

[98] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.

[99] W. Li. Zipf's law everywhere. *Glottometrics*, 5:14–21, 2003.

[100] W. Li, K. Wang, S. Stolfo, and B. Herzog. Fileprints: identifying file types by n-gram analysis. In *6th IEEE Information Assurance Workshop*, West Point, NY, 2005.

[101] Z. Li, R. Yuan, and X. Guan. Accurate classification of the internet traffic based on the svm method. In *Proceedings of IEEE International Conference on Communications (ICC'07)*, 2007.

[102] M. Liberatore and B. N. Levine. Inferring the source of encrypted http connections. In *Proceedings of the 13th ACM conference on Computer and communications security*, Alexandria, VA, 2006.

[103] Lincoln Laboratory, MIT. DARPA intrusion detection data sets, 2008. http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html.

[104] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das. The 1999 darpa off-line intrusion detection evaluation. *Computer Networks*, 34(4):579 – 595, 2000. Recent Advances in Intrusion Detection Systems.

[105] R.-T. Liu, N.-F. Huang, C.-N. Kao, and C.-H. Chen. A fast pattern matching algorithm for network processor-based intrusion detection system. In *Performance, Computing, and Communications, 2004 IEEE International Conference on*, pages 271–275, 2004.

[106] W. Lu, G. Rammidi, and A. A. Ghorbani. Clustering botnet communication traffic based on n-gram feature selection. *Comput. Commun.*, 34(3):502–514, Mar. 2011.

[107] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker. Unexpected means of protocol inference. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pages 313–326, New York, NY, USA, 2006. ACM.

[108] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling High Bandwidth Aggregates in the Network. *In ACM SIGCOMM Computer Communications Review*, July 2002.

[109] R. Mahajan, S. Floyd, and D. Wetherall. Controlling High Bandwidth flows at the congested router. *In Proceedings of the International Conference on Network Protocols (ICNP'01)*, 2001.

[110] M. V. Mahoney and P. K. Chan. An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. In *Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection*, pages 220–237. Springer-Verlag, 2003.

[111] M. V. Mahoney and P. K. Chan. Learning rules for anomaly detection of hostile network traffic. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, pages 601–, Washington, DC, USA, 2003. IEEE Computer Society.

[112] B. Martins and M. J. Silva. Language identification in web pages. In *Proceedings of the 2005 ACM symposium on Applied computing*, Santa Fe, New Mexico, 2005.

[113] A. Matrawy, P. van Oorschot, and A. Somayaji. Mitigating network denial-of-service through diversity-based traffic management. In *Applied Cryptography and Network Security (ACNS'05)*. Springer, 2005.

[114] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow clustering using machine learning techniques. In *In PAM*, 2004.

[115] J. McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Trans. Inf. Syst. Secur.*, 3(4):262–294, 2000.

[116] MDNS. Multicast domain name service. http://www.multicastdns.org/.

[117] M. Mitzenmacher. Editorial: The future of power law research. *Internet Mathematics*, 2(4):525–534, 2006.

[118] A. Moore and K. Papagiannaki. Toward the accurate identification of network applications. In C. Dovrolis, editor, *Passive and Active Network Measurement*, volume 3431 of *Lecture Notes in Computer Science*, pages 41–54. Springer Berlin Heidelberg, 2005.

[119] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *Proceedings of ACM SIGMETRICS*, 2005.

[120] MRTG. Multi router traffic grapher (mrtg). http://oss.oetiker.ch/mrtg/.

[121] MSNMS. Microsoft network messenger service. http://messenger.msn.com.

[122] MSRDP. Microsoft remote display protocol. http://support.microsoft.com/default.aspx?scid=kb;EN-US;q186607.

[123] MSSQL. Microsoft sql protocol. http://www.microsoft.com/sql/default.mspx.

[124] MySQL. Mysql protocol. http://www.redferni.uklinux.net/mysql/MySQL-Protocol.html.

[125] Z. Nascimento, D. Sadok, and S. Fernandes. A hybrid model for network traffic identification based on association rules and self-organizing maps (som). In *ICNS 2013, The Ninth International Conference on Networking and Services*, 2013.

[126] NBDGM. Netbios datagram service. http://rfc.net/rfc1001.html.

[127] NBNS. Netbios name service. http://rfc.net/rfc1001.html.

[128] NBSS. Netbios session service. http://www.keyfocus.net/kfsensor/help/AdminGuide/adm_NBT.php.

[129] M. Newman. Power laws, pareto distribution and zipf's law. *Contemporary Physics*, 46(5):323–351, 2005.

[130] T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. *Communications Surveys Tutorials, IEEE*, 10(4):56 –76, quarter 2008.

[131] NNTP. Network news transfer protocol. http://www.faqs.org/rfcs/rfc977.html.

[132] NTP. Network time protocol. http://www.faqs.org/rfcs/rfc1305.html.

[133] D. Pack, W. Streilein, S. Webster, and R. Cunningham. Detecting http tunneling activities. In *in 2002 IEEE, Workshop on Information Assurance,. 2002. United States Military Academy, West Point, NY: IEEE*, 2002.

[134] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23):2435–2463, 1999.

[135] V. Paxson and S. Floyd. Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.

[136] P. Piskac and J. Novotny. Using of time characteristics in data flow for traffic classification. In I. Chrisment, A. Couch, R. Badonnel, and M. Waldburger, editors, *Managing the Dynamics of Networks and Services*, volume 6734 of *Lecture Notes in Computer Science*, pages 173–176. Springer Berlin Heidelberg, 2011.

[137] D. Plonka. A network traffic flow reporting and visualization tool. In *Proceedings of USENIX Large Installation System Administration Conference (LISA'00)*, 2000.

[138] POP. Post office protocol. http://www.ietf.org/rfc/rfc1939.txt.

[139] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering. In *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*, pages 5–8, 2001.

[140] A. Poutsma. Applying monte carlo techniques to language identification. In *In Proceedings of Computational Linguistics in the Netherlands (CLIN*, pages 179–189. Rodopi, 2001.

[141] J. Prager. Linguini: Language identification for multilingual documents. In *Proceedings of The 32nd Annual Hawaii International Conference on System Sciences*, 1999.

[142] RARP. Reverse address resolution protocol. http://www.ietf.org/rfc/rfc903.txt.

[143] W. Reed. The pareto, zipf and other power laws. *Economics Letters*, 74:15–19, 2001.

[144] RFC. A standard for the transmission of ip datagrams over e. http://tools.ietf.org/html/rfc879.

[145] RFC. The tcp maximum segment size and related topics. http://tools.ietf.org/html/rfc879.

[146] RIPv1. Routing information protocol. http://www.faqs.org/rfcs/rfc1058.html.

[147] M. Roesch et al. Snort-lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX conference on System administration*, pages 229–238. Seattle, Washington, 1999.

[148] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 135 – 148, 2004.

[149] R. Rousseau. George kingsley zipf: life, ideas, his law and informetrics. *Glottometrics*, 3:11–18, 2002.

[150] RTP. Real-time transport protocol. http://www.ietf.org/rfc/rfc3550.txt.

[151] RTSP. Real time streaming protocol. http://www.rtsp.org/.

[152] V. V. S. and S. Bressan. Continuous-learning weighted-trigram approach for indonesian language distinction: A preliminary study. In *In Proceedings of 19th International Conference on Computer Processing of Oriental Languages*, 2001.

[153] Sandvine. Sandvine's network data analytics. http://www.sandvine.com/products/network_data_analytics.asp.

[154] H. Schulze and K. Mochalski. Internet study 2008/2009. Technical report, ipoque, 2009.

[155] S. Sen, O. Spatscheck, and D. Want. Accurate, scalable in-network identification of p2p traffic using application signatures. In *Proceedings of the 13th International World Wide Web (WWW) Conference*, May 2004.

[156] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, pages 27: 379–423, 623–656, 1948.

[157] T.-F. Sheu, N.-F. Huang, and H.-P. Lee. A novel hierarchical matching algorithm for intrusion detection systems. In *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, volume 3, pages 5 pp.–, 2005.

[158] T.-F. Sheu, N.-F. Huang, and H.-P. Lee. In-depth packet inspection using a hierarchical pattern matching algorithm. *Dependable and Secure Computing, IEEE Transactions on*, 7(2):175–188, 2010.

[159] A. Shrivastav and A. Tiwari. Network traffic classification using semi-supervised approach. In *Machine Learning and Computing (ICMLC), 2010 Second International Conference on*, pages 345–349, 2010.

[160] G. Shu and D. Lee. A formal methodology for network protocol fingerprinting. *Parallel and Distributed Systems, IEEE Transactions on*, 22(11):1813–1825, 2011.

[161] P. Sibun and J. C. Reynar. Language identification: Examining the issues. In *Proceedings of the 5th Annual Symposium on Document Analysis and Info Retrieval (SDAIR)*, 1996.

[162] S. Singh, C. Estan, G. Varghese, and S. Savage. The EarlyBird System for Real-time Detection of Unknown Worms. Technical report - cs2003-0761, UCSD, 2003.

[163] S. Singh, C. Estan, G. Varghese, and S. Savage. Automated Worm Fingerprinting. In *Proceedings of 6th USENIX Symposium on Operating Systems Design and Implementation (OSDI'04)*, December 2004.

[164] R. Sinha, C. Papadopoulos, and J. Heidemann. Internet packet size distributions: Some observations. Technical Report ISI-TR-2007-643, USC/Information Sciences Institute, May 2007. Orignally released October 2005 as web page `http://netweb.usc.edu/~rsinha/pkt-sizes/`.

[165] SIP. Session initiation protocol. `http://www.ietf.org/rfc/rfc3261.txt`.

[166] SMB. Server message block. `http://samba.anu.edu.au/cifs/docs/what-is-smb.html`.

[167] SMTP. Simple mail transfer protocol. http://www.faqs.org/rfcs/rfc821.html.

[168] SNMP. Single network management protocol. http://www.faqs.org/rfcs/rfc1157.html.

[169] Sophos. Anti-virus application packets. http://www.sophos.com/.

[170] SRVLOC. Service location protocol. http://tools.ietf.org/html/rfc2608.

[171] SSH. Secure shell. http://www.ietf.org/rfc/rfc4252.txt.

[172] S. Stolfo, K. Wang, and W. Li. Towards stealthy malware detection. *Advances in information security*, 27:231–249, 2007.

[173] STP. Spanning tree potocol. http://www.rfc-editor.org/rfc/rfc4318.txt.

[174] Q. Sun, D. Simon, Y. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu. Statistical identification of encrypted web browsing traffic. In *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, California, USA, 2002.

[175] G. Szabó, J. Szüle, Z. Turányi, and G. Pongrácz. Multi-level machine learning traffic classification system. In *ICN 2012, The Eleventh International Conference on Networks*, pages 69–77, 2012.

[176] TCP. Transfer control protocol. http://www.apps.ietf.org/rfc/rfc793.html.

[177] A. Technologies. A measurement company. http://www.agilent.ca/.

[178] A. Technologies. Mixed packet size throughput. Technical report, Agilent Technologies, 2001. Released as white-paper on the web. PDF file: 1MxdPktSzThroughput.pdf.

[179] F. Tegeler, X. Fu, G. Vigna, and C. Kruegel. Botfinder: finding bots in network traffic without deep packet inspection. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, CoNEXT '12, pages 349–360, New York, NY, USA, 2012. ACM.

[180] TELNET. Telnet protocol. http://www.ietf.org/rfc/rfc0854.txt.

[181] N. Tuck, T. Sherwood, B. Calder, and G. Varghese. Deterministic memory-efficient string matching algorithms for intrusion detection. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2628–2639 vol.4, 2004.

[182] UDP. User datagram protocol. http://www.ietf.org/rfc/rfc0768.txt.

[183] URD. Url rendezvous directory for ssm. http://newsroom.cisco.com/dlls/fspnisapi5992.html.

[184] J. Veronis. French word frequency list. http://www.up.univ-mrs.fr/~veronis/data/DiploFreq.ZIP.

[185] K. Wang, G. Cretu, and S. Stolfo. Anomalous payload-based worm detection and signature generation. In *Proceedings of the Eighth International Symposium on Recent Advances in Intrusion Detection (RAID'05)*, 2005.

[186] K. Wang, J. Parekh, and S. Stolfo. Anagram: A content anomaly detector resistant to mimicry attack. In *Proceedings of the ninth International Symposium on Recent Advances in Intrusion Detection (RAID'06)*, 2006.

[187] K. Wang and S. Stolfo. Anomalous payload-based network intrusion detection. In *In Proceedings of the Seventh International Symposium on Recent Advance in Intrusion Detection (RAID'04)*, 2004.

[188] Y. Wang, Y. Xiang, J. Zhang, and S. Yu. Internet traffic clustering with constraints. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International*, pages 619–624, 2012.

[189] WHO. Messages produced by the unix who command. http://en.wikipedia.org/wiki/Who_(Unix).

[190] N. Williams, S. Zander, and G. Armitage. A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification. *ACM SIGCOMM Computer Communications Review*, October 2006.

[191] C. Williamson. Internet traffic measurement. *IEEE Internet Computing*, 5(6):70–74, November 2001.

[192] R. S. Wong, T.-S. Moh, and M. Moh. Efficient semi-supervised learning bittorrent traffic detection - an extended summary. In *Proceedings of the 13th international conference on Distributed Computing and Networking*, ICDCN'12, pages 540–543, Berlin, Heidelberg, 2012. Springer-Verlag.

[193] C. Wright, L. Ballard, F. Monrose, and G. Masson. Language identification of encrypted voip traffic: Alejandra y roberto or alice and bob? In *Proceedings of the 16th Annual USENIX Security Symposium*, Boston, MA, 2007.

[194] C. Wright, F. Monrose, and G. Masson. On inferring application protocol behaviors in encrypted network traffic. *Journal of Machine Learning Research*, pages 6: 2745–2769, 2006.

[195] C. Wright, F. Monrose, and G. Masson. Using visual motifs to classify encrypted traffic. In *Proceedings of the 3rd international workshop on Visualization for computer security (VizSEC'06*, New York, NY, USA, 2006.

[196] XDMCP. X display manager control protocol. http://www.xfree86.org/current/xdmcp.pdf.

[197] K. Xinidis, I. Charitakis, S. Antonatos, K. Anagnostakis, and E. Markatos. An active splitter architecture for intrusion detection and prevention. *Dependable and Secure Computing, IEEE Transactions on*, 3(1):31–44, Jan.-March 2006.

[198] XMPP. Extensible messaging and presence protocol. http://www.xmpp.org/specs/.

[199] S. Zander, T. Nguyen, and G. Armitage. Self-learning ip traffic classification based on statistical flow characteristics. In *In PAM*, 2004.

[200] S. Zander, T. Nguyen, and G. Armitage. Automated traffic classification and application identification using machine learning. In *Proceedings of IEEE LCN*, 2005.

[201] J. Zhang, R. Perdisci, W. Lee, U. Sarfraz, and X. Luo. Detecting stealthy p2p botnets using statistical traffic fingerprints. In *Dependable Systems Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*, pages 121–132, 2011.

[202] M. Zhang, H. Zhang, B. Zhang, and G. Lu. Encrypted traffic classification based on an improved clustering algorithm. In *Trustworthy Computing and Services*, pages 124–131. Springer, 2013.

[203] G. K. Zipf. *The Psychobiology of Language*. Houghton-Mifflin, 1935.