Lightweight Hierarchical Clustering of Network Packets Using (p, n)-grams

Abdulrahman Hijazi*, Hajime Inoue†,
Ashraf Matrawy*, P. C. van Oorschot*, and Anil Somayaji*

SCS Technical Report TR-09-03

November 2008

Abstract

The complexity of current Internet applications makes understanding network traffic a challenging task. By providing larger-scale aggregates for analysis, unsupervised clustering approaches can greatly aid in the identification of new applications, attacks, and other changes in network usage patterns. In this paper we introduce ADHIC, a new algorithm that clusters similar network traffic together without prior knowledge of protocol structures. Packet similarity is determined through comparisons of (p, n)-grams (substrings within packets at distinguishing offsets). ADHIC is notable in that it 1) assumes no prior knowledge of packet structure, 2) produces a hierarchical decomposition of network traffic, and 3) has the potential to cluster packets at wire speeds. We find that ADHIC appropriately segregates well-known protocols, clusters together traffic of the same protocol running on multiple ports, and segregates traffic from applications, such as p2p, that do not use standard ports. Potential applications for ADHIC include network performance analysis, real-time alerts of flash crowds or worm activity, and dynamic DoS-resistant bandwidth management. We also introduce NetADHICT, our implementation of ADHIC.

Index Terms

computer networks, network security, quality of service, traffic shaping, protocol characterization, peer-to-peer, lightwight clustering, divisive hierarchical clustering, unsupervised learning, ADHIC, NetADHICT.

I. Introduction

THE behavior of modern computer networks is fundamentally complex: many users, many uses, numerous protocols, massive operating systems, complex applications, and a wide variety of connected devices. As a result, understanding the behavior of even the simplest local area network can be challenging. Such analysis, however, for debugging, network management, and security purposes is increasingly important. Common analysis strategies have been to classify traffic using predetermined classifiers, based on ports and IP addresses or also using protocol dissectors [4]. While such approaches can help one understand a given set of packets or flows, they are much less helpful when the problem is to understand the overall structure of network traffic.

To better capture this structure, we believe we need an analysis technique with several unusual qualities. First, rather than analyze traffic with respect to pre-defined categories that may not best describe a given network, we want a technique for clustering traffic into semantically meaningful classes that are inferred from observed traffic.

*A. Hijazi, A. Matrawy, P. C. van Oorschot, and A. Somayaji are with the Carleton Computer Security Lab, Computer Science Department, Carleton University, Ottawa, ON, K1S 5B6, Canada (Email: {ahijazi, amatrawy, paulv, soma}@ccsl.carleton.ca).

†H. Inoue is with ATC-NY Corporation, Ithaca, NY, 14850, USA (Email: hinoue@ccsl.carleton.ca).

Manuscript created November 2008.

The authors would like to thank G. Scott Knight with the Department of Electrical and Computer Engineering at the Royal Military College of Canada for testing ADHIC at the RMC environment, and running and analyzing ADHIC against a rich dataset of the College. The authors would also like to thank Evan Hughes for his assistance with our early (p, n)-gram analysis tools and Dana Jansens for work on NetADHICT.

This work was supported by Canada's National Science and Engineering Council (NSERC) through the Canada Research Chair's program (PVO) and Discovery grants (AM, PVO, & AS), Ontario Graduate Scholarships (AH), Ontario Graduate Scholarships in Science and Technology (AH), Public Safety and Emergency Preparedness Canada Scholarships (AH), the Communications Security Establishment, and MITACS.

Because patterns of network behavior can manifest in packet headers (traffic from a given host) or in packet payloads (a flash crowd visiting a given URL), we would like to build clusters using data from anywhere in a packet. To facilitate online traffic monitoring, however, there must be an efficient means to assign new packets to clusters. Network traffic has a naturally hierarchical structure (e.g., HTTP $\subseteq$ TCP $\subseteq$ IP), thus we should build a hierarchy of clusters; however, to maximize simplicity and potential coverage, we want our clustering technique to have no built-in knowledge of standard packet structure.

To meet these requirements, we have developed a new unsupervised clustering algorithm, ADHIC (Approximate Divisive HIerarchical Clustering) [8], [9]. ADHIC creates semantically equivalent clusters without manually labeled training data or complex statistical features. Instead, it represents clusters using fixed-offset, fixed-length strings, a pattern we refer to as (p, n)-grams [20], where p is the offset, and n is the length. Most (p, n)-grams are rare and correspond to fragments of payload data. High frequency (p, n)-grams, however, correspond to the structural features of network packets. By choosing (p, n)-grams of appropriate frequency, we can represent the structure of network traffic without any other assumptions regarding the contents of packet headers or payloads.

More specifically, ADHIC clusters packets using an iteratively derived (p, n)-gram decision tree, where each (p, n)-gram is chosen on the basis of its frequency in observed traffic. Because the presence of (p, n)-grams can be efficiently measured, and because the frequencies of common (p, n)-gram can be estimated using small samples, ADHIC can continually monitor traffic *and* adjust its clustering performance online and at high speed.

To evaluate ADHIC, we implemented a packet analysis tool called NetADHICT [10], [11] (pronounced "net addict"). In experiments on three different networks of varying size and usage patterns, we have found that ADHIC can quickly capture the overall structure of traffic in a way that reflects the relative popularity of different uses of network bandwidth. While much of the inferred structure corresponds to typical divisions of network traffic (TCP vs. UDP, web vs. non-web traffic), these divisions are arrived at using (p, n)-grams that generally are most meaningful within a given environment. Because our clustering is often accomplished without direct reference to ports, the use of non-standard ports for protocols has little effect on clustering performance. Applications that do not have standard ports, such as the BitTorrent P2P file sharing protocol [3], [24], can also be clustered appropriately without requiring any protocol-specific information. Similarly, encrypted traffic is also often clustered appropriately because we are able to appropriately segregate other traffic. These results show that ADHIC holds promise as a powerful yet lightweight method for discovering new patterns in the structure of network traffic.

To summarize, in this paper, we present 1) (p, n)-grams as a useful mean to gauge clustering; 2) ADHIC, a novel algorithm in network traffic clustering; and 3) NetADHICT, our implementation of ADHIC. We, finally, 4) describe our results on various datasets, taken from three distinct environments, which show that the clusters produced by ADHIC are semantically meaningful, even when confronted with P2P traffic and/or denied header information. This paper is an expanded version of an earlier conference paper [9]; here, present more detailed results including data from two additional networks.

The rest of this paper proceeds as follows. Section II explains the key design decisions underlying ADHIC, and Section III describes the ADHIC algorithm. Section IV presents NetADHICT and describes our experimental setup. Sections V, VI, VII, and VIII present more detail on ADHIC's performance through an explication of a few decision trees, overall classification behavior, and performance in the face of peer-to-peer traffic and traffic with out considering headers using data gathered from a small lab. Section IX presents results on ADHIC's performance on other networks. We review related work in network traffic analysis in Section X. Section XI places our work in context and discusses limitations and future work, and we conclude with Section XII.

II. Capturing Packet Structure

Our main goal with this work has been to devise a technique for clustering network packets into semantically meaningful classes without using any domain-specific knowledge. The method had to be unsupervised because we wanted it to find the structure already present, rather than finding the structure we expected to find. Also, it had to do so efficiently and online so that changes in network traffic may be tracked as they occur. To achieve these goals, we chose to create a divisive hierarchical clusterer. Divisive clusterers work in a top-down fashion: they assume all data first belong to one cluster, and then this cluster is iteratively divided into smaller clusters to capture finer-grained details [12]. Divisive clustering was a good fit for our goals because we wanted to capture large scale patterns rather than the fine-grained details of network behavior. Further, we chose a hierarchical clustering

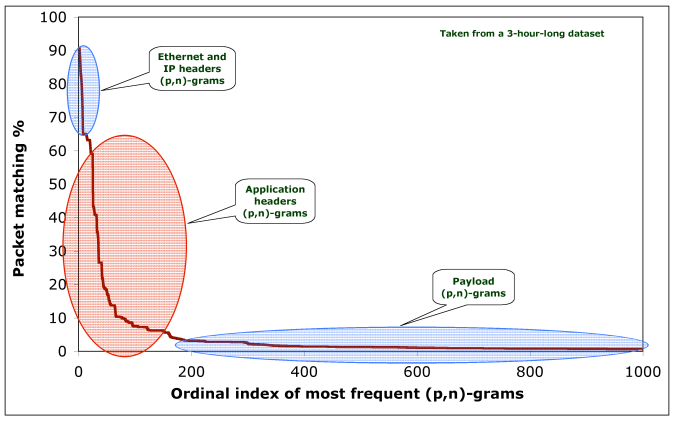

Fig. 1. (p, n)-gram frequency distribution in network traffic: this graph shows the percentage of packet matching of the most frequent (p, n)-grams taken from a sample 3-hour maturation window in the CCSL January trace (See Section IV). Note the graph three regions and the corresponding (p, n)-grams distribution.

approach because Internet traffic has an encapsulated structure, with HTTP 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.

Existing approaches to divisive hierarchical clustering, however, were not suitable to our task because they typically employ an entropy minimization calculation which is $O(n^2)$ in the number of clustered items [5], something that is too expensive for a high-speed implementation. What we wanted instead was a "sub-linear" algorithm: it should only need to look at a small portion of a packet before deciding in what cluster it belongs to. Consider, for example, how high-speed routers can quickly classify incoming packets through looking at only a subset of bytes in a packet header (the destination IP address in particular). We thus decided 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 is an n-byte string offset p-bytes from the start of a packet. For example, in an IP packet encapsulated inside of an ethernet packet, the destination IP address is 4-byte string offset 30 bytes. A wide variety of similarities in packet headers can be represented using one or more (p, n)-grams; further, payload similarities can also be represented so long as a given pattern is present at a consistent offset. In addition, packets containing identical data will share a variety of (p, n)-grams. Of course many patterns, such as the presence of a string anywhere in a packet, cannot be efficiently represented using (p, n)-grams. Our hypothesis was that there was sufficient richness in the (p, n)-gram representation that we could still capture the high-level structure of network traffic.

A key problem that we faced was how to choose the right (p, n)-grams. If we need to perform complex packet analysis to find them, the performance advantages of (p, n)-grams would be lost. However, in our experiments we found that there are a significant number of high and moderate frequency (p, n)-grams (see Figure 1), the frequency of which appears to follow a power-law similar to Zipf's law [28]. Note the three different regions in the graph. Very few (p, n)-grams have very high frequency (higher than 70%). For the most part, those are found in the packets' Ethernet or IP headers. Majority of the (p, n)-grams, however, are very infrequent (less than 5%) and are mostly found in the packets' payload portion. The region in the middle represents (p, n)-grams that have relatively high frequency (5% - 70%) and are mainly found in the packets' application and transport headers. These (p, n)-grams are most often structural ones (See Section V). Thus, so long as we use relatively high frequency (p, n)-grams, we will be most likely capturing packet structure—without any assumptions regarding packet structure.

We chose two bytes ($n = 2$) for the length of (p, n)-grams (See Section IV). Long (p, n)-grams provide a large amount of context, providing more semantically meaningful splits, but long (p, n)-grams are not found frequently. Shorter (p, n)-grams are easier to find in large quantities, but may not be as meaningful. We considered (p, n)-gram lengths of 1, 2, 3, and 4. Our initial algorithm for (p, n)-gram analysis worked well with $n = 4$ [20]; for ADHIC, however, we found that $n = 2$ produced the best results on our datasets.

Thus, to capture the structure of network traffic, we designed an online divisive hierarchical clustering algorithm that segregates packets using high-frequency (p, n)-grams. The details of our algorithm are described below.

```
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)
    }
```

Fig. 2. Pseudocode for the ADHIC adjustment algorithm. `cache` holds a sample of recently observed packets.

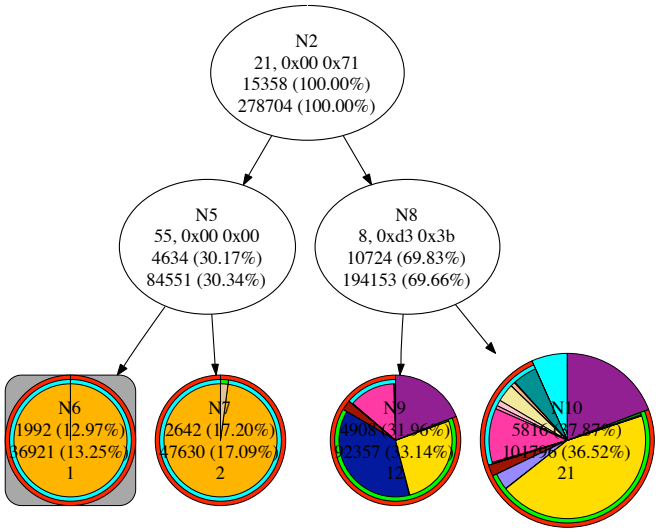

Fig. 3. An example decision tree. All nodes have a node identifier (e.g. N2) and traffic statistics (packet counts and percentage) for the *update period* (last 10 min) and *maturation window* (last 3 hours); see Section IV. Internal nodes (i.e. N2, N5, and N8) are each associated with a (p, n)-gram (e.g. 21, 0x00 0x71). Left branches indicate matches, while right branches indicate the absence of a given (p, n)-gram. The circle size of a leaf node (cluster) represents the number of packets seen over the last update period. The numbers at the bottom of each leaf node tells how many different protocols seen in the cluster. The leaf node with a rounded grey box means 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. Each protocol type within a cluster is represented by a slice, where the size of a slice represents the number of packets. Slices with one color represent Ethernet protocols; two colors (slice + one stripe) are IP protocol types including TCP and UDP; three colors (slice + two stripes) are specific TCP and UDP protocols. See Table I for protocol-color matchings. Note that the colors come from our reference classifier, not ADHIC.

III. Approximate Divisive Hierarchical Clustering

ADHIC works by recursively subdividing traffic into binary classes, with each subdivision being defined by the presence or absence of a given (p, n)-gram. ADHIC stops dividing classes when the resulting traffic is below some configurable threshold in volume or is too similar or too dissimilar.

ADHIC produces a binary decision tree (Figure 3, for instance, shows a simple one captured an early stage) that consists of internal decision nodes and leaf nodes which are the final (*terminal*) clusters. Traffic that matches a classification rule within the node is directed to the left, or *true* subtree. The rest of the traffic is directed to the right subtree. Because rightmost subtrees have not matched any 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*.

Traffic within each terminal cluster can be viewed as the result of a boolean equation constructed by following the path from the root node to the leaf. Left subtrees are combined with **and** and right subtrees **and not**. If a packet contains the same n bytes at byte offset p as specified by the (p, n)-gram associated with the node, it is said to *match*. We use the following notation to refer to specific (p, n)-grams. A (p, n)-gram substring of length two ($n = 2$), consisting of the bytes 0x00 and 0x08 at offset 43 is denoted (43, 0x00 0x08).

The tree is updated through two operators: **split** and **delete**. Splitting is attempted when a leaf cluster matches too much traffic. To split, we search for a (p, n)-gram that matches approximately half of the packets of the cluster; the matching percentage range, around 50%, we refer to as the *similarity spread*. In our experiments, we set the similarity spread parameter to 20% and split threshold to 2% (see Table II). Thus, a leaf cluster will split if it matches more than 2% of the packets, and a (p, n)-gram is found such that it exists in 40% to 60% of the packets in that cluster. These statistics are measured over a period called the *maturation window*. Nodes which have been

TABLE I
PROTOCOL CLASSIFICATION AND CONTENT STATISTICS FOR THE FOUR CCSL NETWORK TRACES. ONLY PROTOCOLS WITH PERCENTAGE $\geq 0.05\%$ ARE SHOWN (BEST VIEWED IN COLOR)

Protocol	August 13-19, 2004	December 10-16, 2005	January 20-26, 2006	April 03-09, 2006
IPv4	9461744 (100.00%)	13014116 (85.73%)	11910975 (83.29%)	8684678 (87.00%)
TCP	4445742 (46.99%)	10141461 (66.81%)	7747114 (54.17%)	6107644 (61.18%)
TCP Unknown	543377 (5.74%)	87450 (0.58%)	861184 (6.02%)	44388 (0.44%)
IPP	481 (0.01%)	400141 (2.64%)	568669 (3.98%)	486936 (4.88%)
IMAPS	51629 (0.55%)	58799 (0.39%)	118224 (0.83%)	166291 (1.67%)
HTTPS	18192 (0.19%)	50889 (0.34%)	17492 (0.12%)	123983 (1.24%)
SSH	755329 (7.98%)	372847 (2.46%)	497408 (3.48%)	245513 (2.46%)
MS Streaming/RTSP	121034 (1.28%)	69326 (0.46%)	69768 (0.49%)	97814 (0.98%)
TCP No Payload	2359999 (24.94%)	7442843 (49.03%)	4891984 (34.21%)	4069094 (40.76%)
FTP	136476 (1.44%)	3292 (0.02%)	26310 (0.18%)	5298 (0.05%)
HTTP	431479 (4.56%)	1585943 (10.45%)	630874 (4.41%)	797035 (7.98%)
Others	27746 (0.29%)	69931 (0.46%)	65201 (0.46%)	71292 (0.71%)
UDP	4730279 (49.99%)	2526432 (16.64%)	3864788 (27.02%)	2288802 (22.93%)
UDP Unknown	2493908 (26.36%)	5951 (0.04%)	8618 (0.06%)	3346 (0.03%)
DNS	21968 (0.23%)	91051 (0.60%)	60339 (0.42%)	67311 (0.67%)
CUPS	336827 (3.56%)	314578 (2.07%)	296348 (2.07%)	128278 (1.28%)
RTP	0 (0.00%)	176322 (1.16%)	1587449 (11.10%)	248642 (2.49%)
NBDGM	67167 (0.71%)	80742 (0.53%)	73665 (0.52%)	62802 (0.63%)
Ganglia	0 (0.00%)	402664 (2.65%)	233585 (1.63%)	219859 (2.20%)
NBNS	55779 (0.59%)	54753 (0.36%)	187518 (1.31%)	177278 (1.78%)
RIPv1	410864 (4.34%)	41948 (0.28%)	41940 (0.29%)	41538 (0.42%)
HSRP	1307515 (13.82%)	1306755 (8.61%)	1306607 (9.14%)	1293451 (12.96%)
Others	36251 (0.38%)	51668 (0.34%)	68719 (0.48%)	46297 (0.46%)
ICMP	4620 (0.05%)	82944 (0.55%)	36000 (0.25%)	28430 (0.28%)
EIGRP	280532 (2.96%)	261475 (1.72%)	261672 (1.83%)	258756 (2.59%)
Others	571 (0.01%)	1804 (0.01%)	1401 (0.01%)	1046 (0.01%)
ARP	N/A	1779217 (11.72%)	1990922 (13.92%)	877582 (8.79%)
ETHER (old)	N/A	385789 (2.54%)	399206 (2.79%)	420344 (4.21%)
Others	N/A	508 (0.00%)	117 (0.00%)	225 (0.00%)
Total no. of Packets	**9461744**	**15179630**	**14301220**	**9982829**
Total Size in MB	6421	3415	3510	2089
Average Packet Size in B	711	236	257	219

modified within a maturation window of the current time are locked and cannot be split or deleted.

Deletion occurs when a subtree has not matched a minimum threshold of traffic percentage during the most recent maturation window. The subtree's parent node is also deleted. The parent node's other tree, the one not deleted, becomes the direct child of the parent node's parent. See Figure 2 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. The similarity spread, maturation window, update period, and the thresholds for splitting and merging are all configurable parameters (Table II shows the values we 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 be deleted. Individual (p, n)-grams within nodes would then be deleted if they did not match a packet within the maturity period. This variant more closely approximates the operator regime used in our previous linear classifier [20]. We found, however, that the quality of the clusters produced by ADHIC with either operator regime was similar. ADHIC produces much better clusters than our linear classifier, regardless of the set of operators used.

Figure 3, a sample decision tree produced at an early stage by our ADHIC algorithm, shows traffic split into two clusters by the (p, n)-gram (21, 0x00 0x71). The first byte in this (p, n)-gram is part of the "fragment offset" field, while the second byte is "time-to-live" (TTL). The two clusters were then split into four terminal (leaf) clusters. The traffic that matched (21, 0x00 0x71) is matched against the (p, n)-gram (55, 0x00 0x00). Note, in this example, that the two left terminal clusters are dominated by packets from one protocol type: RTP (Real-time Transport Protocol). They all shared a special TTL value (0x71) that made them easily distinguishable from the rest of the

packets.

The space of (p, n)-grams is very large and the total number of (p, n)-grams that appear in a network trace, though much smaller than the total possible, is also large. For a fixed n, the number of (p, n)-grams in a single packet is the length of the packet minus n plus 1; thus, the number of (non-unique) (p, n)-grams for each trace is approximately the number of bytes in the trace. The space of (p, n)-grams is dependent on the length of each packet and the length of the (p, n)-gram. For example, by assuming a packet length of 1500 bytes, there are approximately 384,000 (p, n)-grams of length 1 $(256 * 1500)$, 100 million of length 2, and 6.5 trillion for length 4. For this work we set $n = 2$; for a further discussion of the settings of this and other parameters, see Section IV.

By viewing packets as individual (p, n)-grams, the algorithm treats packets as high dimensional vectors—the number of dimensions is the packet's length. Note these dimensions are not independent. The effective information provided by the (p, n)-gram vector is reduced by its overlapping nature; also, the presence of one (p, n)-gram often affects the probability of other, non-overlapping (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 [5]. 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-1} choices for each split (where m is the number of packets in the original cluster).

ADHIC uses individual (p, n)-grams as a proxy for a more expensive entropy calculation. As we mentioned earlier, the (p, n)-gram frequency distribution in network traffic appears to conform to a power law analogous to Zipf's law [28]. Because power-law distributions decay 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.

If we assume a normal distribution of the traffic packets, the sample size required is estimated using the well known function $n = \frac{1}{B^2}$ where B is the error rate. For a 5% error rate, then, the sample size should be 400 packets. Network traffic does not follow a normal distribution [23], however; thus, we must sample more. In our experiments, we chose the conservative sampling rate of 20% (see Table II). However, we found that, for our lab data, only 3% of packets are needed to achieve an error rate of less than 5% for (p, n)-gram frequencies.

IV. Experimental Setup

We evaluated ADHIC's performance at network analysis through a series of experiments on various real network traces using NetADHICT [11], our prototype implementation of ADHIC. NetADHICT is licensed under the GNU General Public Licence (GPL), version 2 or later, and available from the Carleton Computer Security Laboratory (CCSL) website [10].

We initially tested ADHIC against datasets gathered from our CCSL network. These captures contained raw, full Ethernet packets. The CCSL is a production laboratory with over 15 machines, two network printers and about a dozen regular users. The lab provides web, webmail, IMAPS, DNS, SSH, SMTP, and CUPS services to external hosts. The four datasets we used for our primary analysis are described in Table I. Each of them consists of incoming network traffic captured throughout a course of seven consecutive days. More recently, we have also tested ADHIC on two other networks; the results from this work are presented in Section IX.

To better understand the behavior of ADHIC, we constructed an independent "*reference classifier*." It is a port-based tuple classifier, primarily relying on IP protocol and port information, but also monitoring features such as Ethernet packet type. NetADHICT uses the reference classifier to label ADHIC's output trees by protocol. Although ADHIC is not designed to classify protocols, the protocol labeling produced by the reference classifier allows us to quickly compare ADHIC with port-based traffic classification, a standard, lightweight approach for understanding network behavior.

Table I provides packet counts statistics of the main protocols constituting traffic observed in the four tested CCSL traces, as measured by our reference classifier. 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 including TCP and UDP. Protocols with three color levels are specific TCP and UDP protocols.

All the ADHIC experiments referenced herein were run with a set of fixed parameter values (see Table II), determined by exploring several sets of alternative values using the CCSL traces. Our evaluation of ADHIC's

TABLE II
ADHIC PARAMETERS USED IN MOST OF OUR EXPERIMENTS

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%

performance relied on analysing the decision trees it produces after every update period along with the updated statistics. In our testing environment, ADHIC is not highly sensitive to most of these parameter values and tends to produce qualitatively similar trees under many settings; thus, we have chosen these 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. The one significant exception to this, however, is the value of n, for which we tested the values 1–4 and settled on a value of 2 (see Section III).

On an Apple Mac Pro with 1GB of main memory and 2.66 GHz "Woodcrest" cores, the single threaded current implementation of ADHIC (with logging minimized) is able to cluster packet data at about 250Mbps. 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.

V. AN ADHIC DECISION TREE

To illustrate the effectiveness of ADHIC, we next briefly describe an example decision tree and the types of clusters it produces. As explained in the previous section, the cluster labels come from our reference classifier, not ADHIC; ADHIC only produces the (p, n)-grams for each node.

Figure 4 shows a decision tree produced after four days of execution from the CCSL January trace. We have also added an annotated, symbolic version that is easier to explain in the same figure. Each triangle in the annotated graph represents one or more terminal cluster nodes that contain same protocol 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 rules. 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.

The first (starting from left to right) cluster on the annotated tree graph (denoted by a triangle) at N11 primarily separates ARP (Address Resolution Protocol) from other traffic. The N14 subtree is a large mix of protocols, where the clusters demonstrate the effectiveness of the technique for non-TCP protocols, showing singular clusters for DTP (Dynamic Trunking Protocol), EIGRP (Enhanced Interior Gateway Routing Protocol), IGMP (Internet Group Management Protocol), and Ganglia (a cluster monitoring application). All the other clusters are singular (except for a default cluster) and contain TCP control packets featuring zero-length payload. Similarly, the subtree with root N17 shows that the algorithm was successful in segregating much of the HTTP traffic and separating CUPS (Common UNIX Printing System) packets in two singular clusters. The right most cluster in this subtree functions as a default cluster and is evenly divided between low bandwidth UDP protocols and HTTP.

N41 subtree entirely segregates Cisco HSRP (Hot Standby Router Protocol) traffic. The subtree further divides HSRP evenly into 6 clusters according to their other special (p, n)-grams matching. The packets in each HSRP cluster are exact copies, with each type of HSRP packet segregated to a different cluster.

The N158 subtree shows interesting clusters for POP (Post Office Protocol), IMAPS (secure Internet Message Access Protocol), HTTP, and IPP (Internet Printing Protocol). These protocols are what people mainly use when they first arrive in the morning: they check email, websites, and print papers. All of the IPP data along with their related zero-length-payload TCP control packets are grouped together at the N566 subtree. On the other hand, IMAPS packets along with their related TCP control packets were automatically grouped together in two clusters, namely: N546, and N569. Moreover, POP packets resided in one singular cluster at N653. Two default clusters

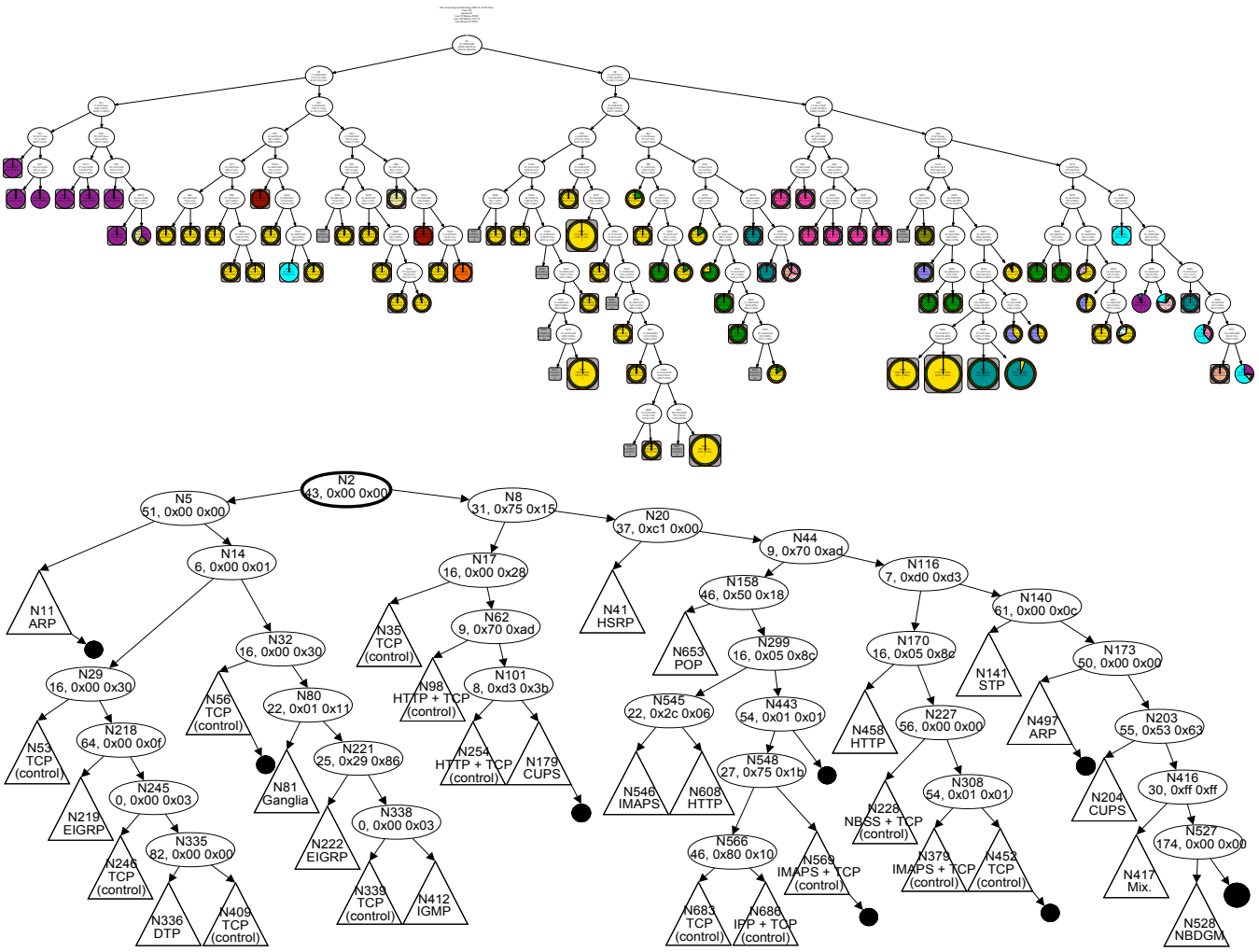

Fig. 4. (Best viewed in color and electronically to allow enlargement) A decision tree produced by ADHIC (top) and a simplified version (bottom), from a snapshot taken the morning of January 24, 2006 from the January CCSL network trace. The original tree contains 89 terminal clusters (leaves) and 88 internal nodes. Terminal clusters are represented by pie charts (color key provided in Table I). Singular clusters are presented on a gray box. In the simplified tree, ovals represent internal nodes, triangles represent subtrees, filled circles represent default clusters.

appear in this subtree (i.e. right hand side of N443 and N569) but they are both dominated by TCP control packets and IMAPS and their related TCP control packets respectively.

The remaining clusters on the right contain six singular clusters: HTTP, TCP control packets, STP (Spanning Tree Protocol), IPP, and NBDGM (NetBIOS Datagrams). Note that NBDGM packets did not match any of the top tree (p, n)-grams, but were finally grouped together at N528 separately from all other packets that went to the tree's global default cluster. The remaining three clusters are a mix of non-IP and UDP protocols such as ARP, and RIPv4 (Routing Information Protocol).

A. *Special* (p, n)*-grams*

It is worth noting that the tree in Figure 4 features an unusual root node (p, n)-gram because a significant portion of that traffic (about 35% by packet count) is HTTP traffic running on a non-standard port. The root node, (43, 0x00 0x00), was chosen by ADHIC because it matched padding on those non-standard HTTP packets that have zero-length payload (mostly TCP control packets). We find that ADHIC often uses features such as padding to word lengths to cluster packets.

In addition, ADHIC often finds common special patterns within packet payloads and uses them to distinguish between different protocols. For example, most ARP packets in the N11 subtree are segregated through (51, 0x00

0x00), which is part of the Ethernet trailer. The other ARP packets end up in two far right clusters (nodes N497 and global default cluster). Another example is (82, 0x00 0x00) which groups all DTP packets (both STP and DTP are labeled as "Ethernet (old)" in the tree and in Table I) with zeros padding in their trailers at N336. Shorter DTP packets do not have this padding and are less well separated in other nodes. A third example is (61, 0x00 0x0c), which appears in the STP packets' frame check sequence, and clusters matching STP packets in N141. Other STP packets, with a different frame check sequence are gathered in the tree's global default cluster.

B. 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 and are used to cluster packets. For example, in one of the experiments, IPP packets were segregated by (27, 0x75 0x1b), part of the source IP address. Other times (p, n)-grams are discovered deep within the payload. An example of this is (174, 0x00 0x00), as in Figure 4, which uniquely identifies all NetBIOS-DGM 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 packet structure. Another example is the (301, 0x00 0x00) (p, n)-gram, which effectively segregates 75% of RIPv1 traffic in the August tree (see Figure 8). This (p, n)-gram is part of the zero-padding within the RIPv1 "IP address" field.

Sometimes header and payload (p, n)-grams are used to cluster the same protocol in different contexts. For example, in Figure 4, all HSRP packets are 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. 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.

Sometimes both header and payload (p, n)-grams are used to segregate the same protocol into different clusters in the same tree. For example, in Figure 4, EIGRP packets were all grouped at N219 and N222, having separated at N14. Note that while N219 and N222 are split using different (p, n)-grams, packets in both clusters in fact share these two (p, n)-grams (along with others).

So why was EIGRP traffic separated at N14? That node splits traffic based on a source MAC address pattern, however, the EIGRP packets come from multicast traffic produced by two different machines.

C. Encrypted and unstructured packets

Multimedia traffic and encrypted TCP traffic are usually either segregated by header (p, n)-grams or are shunted away towards the default subtree. For example, in Figure 4 IMAPS packets were neatly separated from others through header (p, n)-grams such as (22, 0x2c 0x06) in N546 ("time-to-live" and "protocol ID") and (54, 0x01 0x01) in N569 (NOP, NOP in "options"). ADHIC clusters encrypted packets together because they are dissimilar to all the other structured traffic: they do not match any of the (p, n)-grams near subtree roots.

In many occasions, ADHIC also separates different types of encrypted traffic using header (p, n)-grams; these (p, n)-grams, however, are not necessarily part of the IP address or port fields. For example, in Figure 8, although SSH and IMAPS packets share a common path in this tree, they are separated at the end by (54, 0x01 0x01) which is in the "options" field in SSH and the "content type" and "version" fields in IMAPS.

VI. ADHIC vs. the Reference Classifier

Most past work on traffic analysis and classification relies on prior knowledge of network protocols (See Section X). In Section V, we described a typical decision tree produced by ADHIC and explained how the tree's terminal clusters are represented by colored pie charts produced by the port-based reference classifier. Singular clusters (denoted in the tree by rounded grey boxes) are those clusters that the reference classifier reports as containing packets of only one protocol. Those clusters are semantically meaningful. However, that 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.

In this section we give quantitative results comparing ADHIC's clusters with the reference classifier. Such a comparison is the closest metric available translating to "semantically meaningful". Our results show that ADHIC

TABLE III
CLASSIFICATION-LIKE CLUSTERING: GAUGING ADHIC'S CLUSTERING PERFORMANCE USING THE CONVENTIONAL CLASSIFICATION VIEW OF THE REFERENCE PORT-BASED CLASSIFIER

Dataset	"Classification-like" clustering using **default** ADHIC settings				
Aug 13-19	All Protocols	TCP	UDP	Other-IP	Non-IP
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
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
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
Median	64.08%	98.41%	94.53%	98.12%	69.95%
Std-Dev	17.49%	13.15%	13.59%	24.09%	15.52%

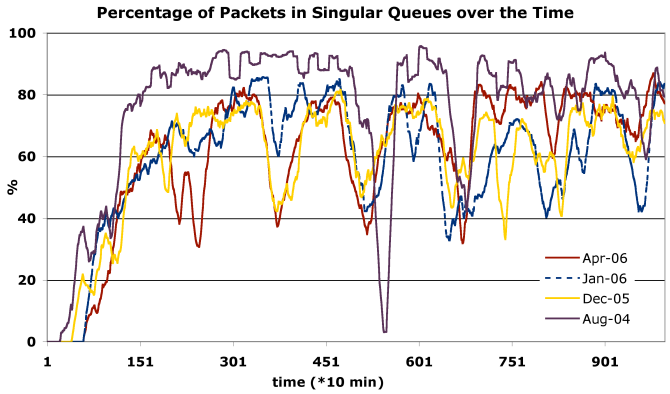

Fig. 5. The percentage of packets in singular clusters at each update period for the four CCSL network traces.

regularly finds and clusters together semantically meaningful packets. With this regard, we found that ADHIC might sometimes be better than the reference classifier itself.

Figure 5, shows ADHIC performance by reporting how close 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. Although ADHIC was not meant to work as a classifier in the first place, 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 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):

$$percentage = \frac{n_s * 100}{n_t}$$

Table III, on the other hand, shows the median and 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, forth, 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.

Note that ADHIC makes relatively better performance with TCP and UDP packets compared to the other-IP and the non-IP packets. This is because very low-volume protocols (including non-IP and other-IP) have relatively very few packets that are sometimes grouped with packets of other protocols in the same cluster; hence, not counted in singular clusters. Nevertheless, the structural similarities between packets within each of the four groups are still strong enough that ADHIC can usually recognize and separate them.

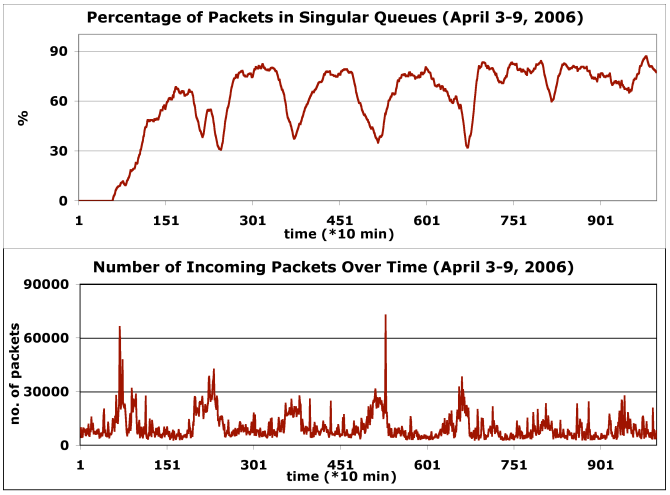

Fig. 6. The percentage of packets in singular clusters at each update period for the CCSL April dataset (top), in contrast with the number of packets seen in the trace for the same time period (bottom). Note how at each spike's peak (bottom), the reference classifier shows a sudden degradation of ADHIC's performance (top).

The results in Table III exclude the first day of operation because ADHIC requires 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 5.

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 our headers-based classifier, but with any headers-based classifier.

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 is very clear in the decision tree visualization. Note how spikes are classified as non-singular and lower the performance because it is high volume. Figure 6, shows this effect clearly on the CCSL April dataset.

As a side note, we use median rather than average as a representative of effectiveness. Consider, for example, Figure 5, 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 III. While Table III and Figure 5 are presently the best metrics for evaluating ADHIC, they under-represent its effectiveness because of the defects inherent in all traditional classifiers.

VII. Clustering P2P traffic

In the past few years, high bandwidth P2P applications have started to disguise their traffic to avoid traffic shaping mechanisms at the Internet service provider (ISP) level. These applications may even deliberately use other protocols' port numbers to disguise their traffic [13]. As a result, traffic shaping tools that use specific packet header fields to characterize network traffic will fail to distinguish, for example, between HTTP and P2P traffic that

uses port 80. The popularity of these high bandwidth applications may have a great impact on the overall network performance if they cannot be discriminated from others.

We have performed several experiments with ADHIC against P2P traffic and observed promising results. These experiments used the conventional BitTorrent [3] client software to download relatively large files (over 500MB) to several of 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 just examples of what the experiments included. While some P2P captured traffic featured unique source port numbers, many others were running on constantly changing port numbers. Traffic pertaining to these experiments was then individually merged with some of the four CCSL datasets tested earlier (see Section IV). In all the experiments, ADHIC was able to segregate P2P traffic from all others and cluster it in a small number of leaves.

Figure 7 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. ADHIC managed to segregate and cluster them in special clusters, and 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. Such 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.

VIII. 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 not UDP or TCP. We then tested ADHIC against the four CCSL traces again, collected statistics, and examined the decision tree at the same snapshot in time.

Similar to Table III, Table IV 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.

We expected ADHIC would perform badly when not allowed to use most of the header information because it usually uses both on 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 8). 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 III and IV, 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 sequence in which ADHIC chooses its (p, n)-grams. From a pool of many (p, n)-gram candidates, 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.

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

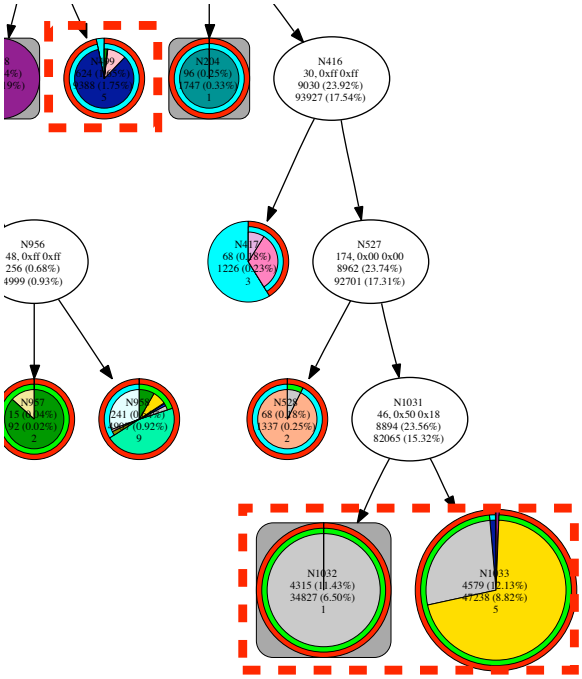

Fig. 7. CCSL January tree snapshot with the presence of P2P traffic (best viewed electronically). All BitTorrent traffic went to the highlighted clusters: N499, N1032, and N1033. The dark blue cluster (N499) represents the BitTorrent's UDP packets, while the gray and yellow clusters (i.e. N1032 and N1033) represent the TCP data and TCP control packets respectively.

TABLE IV
PERCENTAGE OF PACKET STATISTICS AFTER IGNORING HEADER INFORMATION

Dataset	"Classification-like" clustering using **no header** information				
Aug 13-19	All Protocols	TCP	UDP	Other-IP	Non-IP
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
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
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
Median	73.24%	97.49%	91.72%	98.85%	56.85%
Std-Dev	18.40%	10.87%	11.67%	7.49%	21.65%

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 impossible to distinguish between types of encrypted packets—there is no structural information available.

IX. OTHER NETWORKS

To test whether ADHIC's performance on our lab network carried over into other environments, we conducted two experiments on other networks. The first was a three week observation of the uplink of a private small-size sales company in Maryland, USA. The dataset is over 8GB in size, and is mostly populated by web, email, and other application-specific types of traffic. The network consists of over a dozen windows-based network machines including two file servers and a VoIP phone system. The second was an hour-long observation of the uplink of the Royal Military College (RMC) in Kingston, Ontario, Canada. The size of this dataset is about 12GB in size, where the college network has over 1000 users. While ADHIC performed qualitatively similarly in these environments as

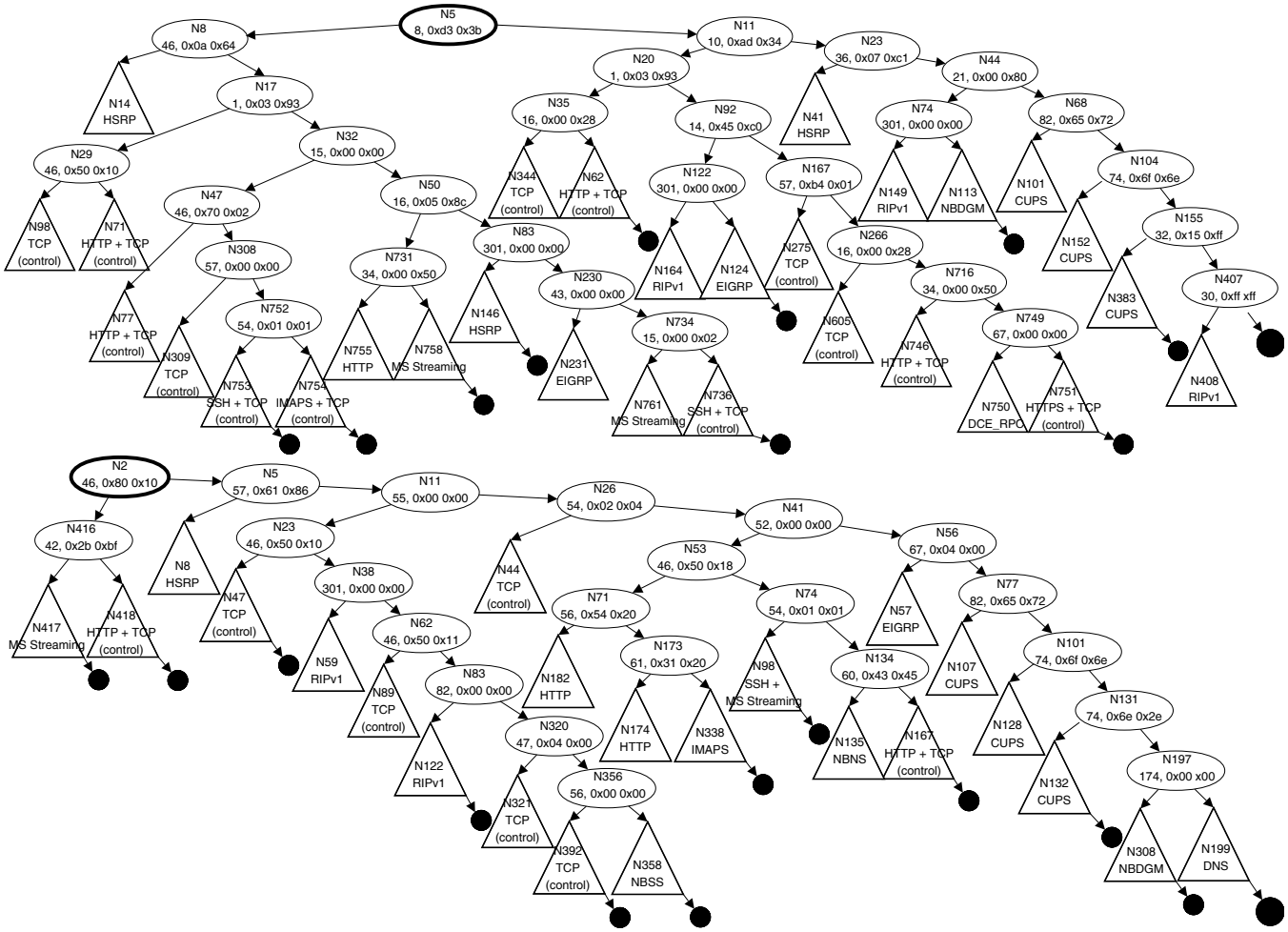

Fig. 8. (Best viewed electronically to allow enlargement) Annotated decision trees produced by ADHIC using default parameters (top) and without looking at the packets' header portion (bottom). Both trees were produced from the same snapshot taken from the CCSL August trace.

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

Figure 9 shows the Maryland network produced a more fine protocol clustering (a deeper tree) than the CCSL trees, due in part to the longer observation time (three weeks versus one week). Several protocols were clustered in this network that were not available in the CCSL runs. For example, AIM (AOL Instant Messenger), RTP (Real-time Transport Protocol), SIP (Session Initiation Protocol), and DNS (Domain Name System) 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 10 shows a very high-level overview of a decision tree taken from the RMC network. Here, ADHIC has chosen (5, 0xXX 0xXX) 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 most probably part of a special zero padding field that is usually found in the Ethernet trailer of the TCP no-payload control packets. The two other identical (p, n)-grams (20, 0x00 0x00) (which represent the IP flags and fragment

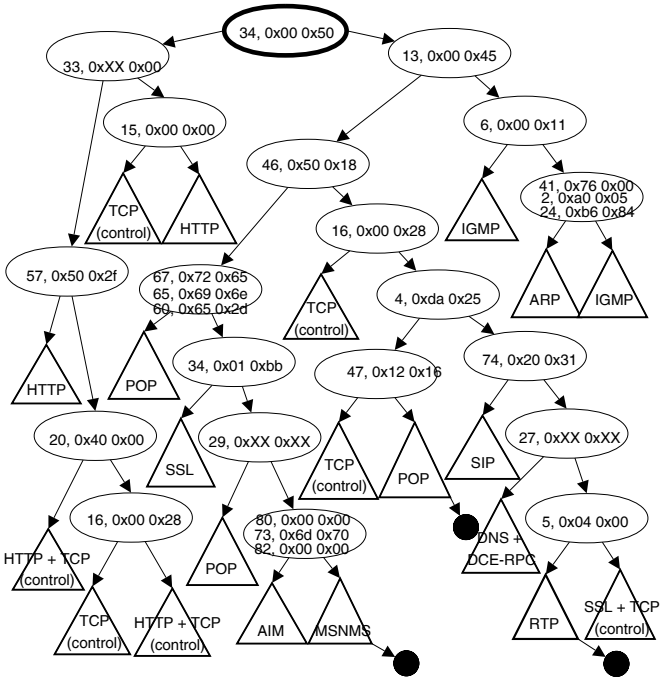

Fig. 9. A simplified decision tree produced by ADHIC from a snapshot taken from the Maryland experiment.

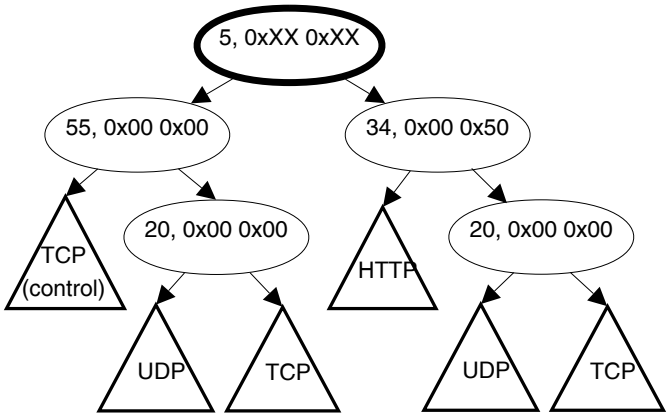

Fig. 10. A high-level simplified decision tree produced by ADHIC from a snapshot taken from the RMC experiment.

offset fields) are acting as discriminators between UDP (matching) and TCP (non-matching) packets on both sides of the tree.

Based on these observations of a small business network and a university network, ADHIC appears to be able to appropriately cluster network traffic in very different network environments.

X. Related Work

ADHIC relates to past work both as a technique for traffic analysis and as a new network-domain specific machine learning algorithm. We begin by discussing the general area of traffic analysis and then focus on analysis using machine learning.

Substantial research has explored the question of how to characterize and model Internet traffic. While it does not in general follow a Poisson model [23], network traffic typically is quite bursty and has been noted to have self-similar properties [18] as have IP addresses [16].

Security concerns have prompted many to look at adaptive methods for analyzing network traffic. The general problem of intrusion detection has inspired several approaches (see [7] for a discussion of several early systems), as has that of extracting the signatures of propagating worms [17], [26], [15]. Rather than address the problem of

identifying malicious or unusual packets, however, we are focused on the problem of clustering packets to learn the structure of network traffic.

Network administrators have standard tools for analyzing individual traffic flows [4] and monitoring traffic volumes [22]; commercial tools, such as QRadar [1], allow traffic volumes to be monitored on a per-protocol basis. The problem of extracting other patterns (such as new protocols, denial-of-service attacks, or flash crowds) is still a research problem. Traditional machine learning approaches have been applied to specific packets characteristics such as standard 5-tuples, packet lengths, and inter-arrival times to classify traffic into predetermined classes. Various algorithms have been used, including Bayesian classification [27] and expectation maximization [21]. While most clustering work only uses header information, Bernaille et al. [2] studied the feasibility of application-level clustering using the size and direction of the first few packets of a TCP connection. Clustering has also been applied to the problem of identifying quality of service classes [25].

Others have developed techniques for classifying traffic without the use of port numbers of payloads. Karagiannis et al. [14] observe host behavior at the social level (host interaction), functional level (popularity, role), and transport layer information. They report classifying 80%-90% of traffic with 95% accuracy. Ma et al. [19] classify traffic by building statistical models of messages exchanged in a protocol. They also do not rely on port number to identify applications.

Estan et al. [6] developed Autofocus, a tool for multidimensional clustering of Internet traffic. Autofocus creates hierarchical clusters; however, its clusters consist of aggregates at the IP packet 5-tuple level.

Our approach differs from all these methods in that we do not rely on any previous knowledge of packet contents, nor do we designate any particular fields as fields of interest in the learning process. We do not attempt to generate a classifier from the clusterer. We do not need pre-labeled input data, flow reconstruction (as required by Ma et al. [19]), packet reassembly, or packet normalization. Moreover, most traditional machine learning cluster algorithms assume offline analysis that involves a quadratic or greater number of comparisons [5]. Our clustering algorithm, however, uses only a small number of packets (through packet sampling) during each monitoring period to construct a cluster decision tree. Because of this, ADHIC is able to assign packets to clusters and adapt the cluster decision tree to changing traffic patterns simultaneously. Finally, our clustering algorithm is dissimilar to other machine learning algorithms in common use (see Section III).

XI. Discussion

As stated before, a primary goal in developing ADHIC was to develop an algorithm for capturing the high-level semantic structure of network traffic without using domain-specific information. Our results suggest that the clusters discovered by ADHIC have a close correlation with semantic classes of interest to network administrators, researchers, and security officers. We find these results both promising and remarkable: ADHIC is both very simple and very effective. They also suggest many important avenues for future work.

For example, ADHIC inherently requires structure within packets to operate well. We have shown that ADHIC can often segregate encrypted and obfuscated packets, but this is done by recognizing other structured protocols and then assigning the remaining traffic to default clusters. Will this behavior persist in larger, more complex environments? With the evidence from the RMC experiment, we suspect it will, so long as the distribution of (p, n)-grams continues to follow a similar power law as traffic in our experiments: in this case, there will be plenty of high-frequency, structural (p, n)-grams for ADHIC to extract. Although in larger more complex networks, we expect to see more of the packets that don't match anything (and are of several encrypted types, or evasive types), these packets are usually segregated from each other using header fields before they all get clumped together. While likely, such characteristics need to be verified through further experiments on enterprise-class networks.

We would also like to develop a better measure of "semantically meaningful" clusters. To this point, we have verified the quality of ADHIC's clusters in our lab through the use of our reference classifier and standard network analysis tools such as WireShark [4]. ADHIC, 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 [5], that will "upper bound" the structure extraction ability of ADHIC.

We also plan to improve the runtime performance of ADHIC by testing various parameter settings and algorithmic variants. Further, we can accelerate ADHIC by sampling fewer packets. In the experiments reported here we sampled 20% of all packets because we were interested in minimizing error and maximizing repeatability. As we described

in Section III, on our network only 3% of packets are needed to achieve an error rate of less than 5% for (p, n)-gram frequencies.

Ultimately, ADHIC is a clustering technique, analysis with which can complement other analysis 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 approach such as ADHIC holds the promise of revealing new patterns of behavior before they become significant problems, as well as mitigating those problems when they do occur [20]. Thus, we believe further work on lightweight approaches to extracting patterns in network behavior is a rich area for future research.

XII. Conclusion

ADHIC, a clustering algorithm based on divisive hierarchical clustering using (p, n)-grams, effectively and efficiently clusters network traffic without specific knowledge of protocol structures.

ADHIC segregates most protocols into distinct clusters, even with encrypted traffic or traffic that purposely disguises itself, as with the BitTorrent P2P protocol. Even when more than one type of encrypted traffic are seen together, they usually get segregated in different clusters through their header-field (p, n)-grams. While further testing is needed, we believe the approach demonstrated by ADHIC is a promising one for analyzing and managing network traffic behavior.

References

[1] Qradar. http://www.q1labs.com, 2008.

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

[3] B. Cohen. Bittorrent protocol specification. http://www.bittorrent.org, 2008.

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

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

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

[7] J. Frank. Artificial intelligence and intrusion detection: Current and future directions. In *Proceedings of the 17th National Computer Security Conference*, oct 1994.

[8] 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.

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

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

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

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

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

[14] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel traffic classification in the dark. In *ACM SIGMETRICS*, 2005.

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

[16] E. Kohler, J. Li, V. Paxson, and S. Shenker. On the structure of addresses in ip traffic. In *Proceedings of ACM Internet Measurement Workshop*, 2002.

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

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

[19] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker. Unexpected means of protocol inference. In *ACM IMC*, 2006.

[20] A. Matrawy, P.C. 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.

[21] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow clustering using machine learning techniques. In *Proceedings of Passive and Active Measurement Workshop*, 2004.

[22] Tobias Oetiker. Multi router traffic grapher. http://www.mrtg.com, 2008.

[23] V. Paxson and S. Floyd. Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.

[24] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips. The Bittorrent P2P file-sharing system: Measurements and analysis. In *Proceedings of 4th Int'l Workshop on Peer-to-Peer Systems (IPTPS)*, Feb 2005.

[25] 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.

[26] 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.
[27] S. Zander, T.T.T. Nguyen, and G. Armitage. Automated traffic classification and application identification using machine learning. In *Proceedings of IEEE LCN*, 2005.
[28] G. K. Zipf. *The Psychobiology of Language*. Houghton-Mifflin, 1935.