View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Dartmouth Digital Commons (Dartmouth College)

Dartmouth College

Dartmouth Digital Commons

Dartmouth College Ph.D Dissertations

Theses and Dissertations

9-1-2011

Anomaly Detection in Network Streams Through a Distributional Lens

Chrisil Arackaparambil
Dartmouth College

Follow this and additional works at: https://digitalcommons.dartmouth.edu/dissertations

Part of the Computer Sciences Commons

Recommended Citation

Arackaparambil, Chrisil, "Anomaly Detection in Network Streams Through a Distributional Lens" (2011). *Dartmouth College Ph.D Dissertations*. 37.
https://digitalcommons.dartmouth.edu/dissertations/37

This Thesis (Ph.D.) is brought to you for free and open access by the Theses and Dissertations at Dartmouth Digital Commons. It has been accepted for inclusion in Dartmouth College Ph.D Dissertations by an authorized administrator of Dartmouth Digital Commons. For more information, please contact dartmouthdigitalcommons@groups.dartmouth.edu.

ANOMALY DETECTION IN NETWORK STREAMS THROUGH A DISTRIBUTIONAL LENS

Dartmouth Computer Science Technical Report TR2011-707

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

in

Computer Science

by

Chrisil Arackaparambil

DARTMOUTH COLLEGE

Hanover, New Hampshire

September 2011

Examining Committee:

Amit Chakrabarti, Chair

David Kotz

Sergey Bratus

Mark Crovella

Brian W. Pogue, Ph.D.
Dean of Graduate Studies

Abstract

Anomaly detection in computer networks yields valuable information on events relating to the components of a network, their states, the users in a network and their activities. This thesis provides a unified distribution-based methodology for online detection of anomalies in network traffic streams. The methodology is distribution-based in that it regards the traffic stream as a time series of distributions (histograms), and monitors metrics of distributions in the time series. The effectiveness of the methodology is demonstrated in three application scenarios. First, in 802.11 wireless traffic, we show the ability to detect certain classes of attacks using the methodology. Second, in information network update streams (specifically in Wikipedia) we show the ability to detect the activity of bots, flash events, and outages, as they occur. Third, in Voice over IP traffic streams, we show the ability to detect covert channels that exfiltrate confidential information out of the network. Our experiments show the high detection rate of the methodology when compared to other existing methods, while maintaining a low rate of false positives. Furthermore, we provide algorithmic results that enable efficient and scalable implementation of the above methodology, to accomodate the massive data rates observed in modern infomation streams on the Internet.

Through these applications, we present an extensive study of several aspects of the methodology. We analyze the behavior of metrics we consider, providing justification of our choice of those metrics, and how they can be used to diagnose anomalies. We provide insight into the choice of parameters, like window length and threshold, used in anomaly detection.

Acknowledgements

I am grateful to my advisor Amit Chakrabarti, for his patience, encouragement, and support in my explorations (in spite of the occasional failure) in various directions while this thesis took shape. I also thank him for his guidance, concern, and direct involvement in my research, especially during my early formative stages as a researcher. Sergey Bratus has played the role of my co-advisor for this thesis. Sergey is responsible for rekindling my excitement for hacker research, and for playing a significant role in the final shape of this thesis. I am thankful to him.

I am also grateful to the other members of my thesis committee, Dave Kotz and Mark Crovella. Dave has also supervised my thesis for the past two years, and his advice kept my thesis-work moving when there were roadblocks. Mark has made frank and thoughtful comments that have added a significant level of maturity to this thesis.

The work in this thesis would not have been possible without the contributions of my co-authors, and I thank each one of them—Joshua Brody, Alper Caglayan, Chris McDonald, Anna Shubina, Keren Tan, Bennet Vance, Guanhua Yan, Jihwang Yeo. Guanhua has especially been closely involved in two of my projects in the last year, and influenced them in minute detail.

I sincerely thank the Computer Science department and Dartmouth College for giving me the incredible opportunity to pursue the program here. The process has been a one of self-discovery, and exposed me to the culture and ethic of the community in the Upper Valley, which in turn has had a significant impression on me.

There are several persons and communities that I have had the pleasure of knowing and working with, who have all enriched my time here at Dartmouth. I thank the many friends throughout my time here at Dartmouth for Good Times. Greg Petrics and his family, in particular, introduced me to the mountains and the woods. John Joline'70 has been an incredible mentor to me in the same way.

I thank the administrative and technical staff in the department over the years, and the system administrators. They have been very kind in working with me and shared many a light moment.

I would like to thank my parents for their encouragement, support and their contribution to my development as an individual in so many inexpressible ways.

Thanks also to the sponsors who have funded the various projects in this thesis. The research was supported by NSF CAREER Award CCF-044827, NSF Grant EIA-98-02068, the U.S. Department of Homeland Security Grant Award Number 2006-CS-001-000001, and NSF Award #0916565.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis is concerned with detecting anomalous events in network traffic streams. An anomalous event is one that is unexpected in the usual course of traffic, when the traffic is observed through the view of some well-defined statistical metric. The survey article of Chandola et al. [CBK09] defines anomalies as “patterns that do not conform to a well-defined notion of normal behavior.” Anomalies are interesting to the observer, because usually they are the manifestation of abnormal activities, problems, or state of either the network, its configuration, or its users. Some classic examples of causes of anomalies (sometimes simply referred to as anomalies themselves) include denial of service attacks, fault or failure of network nodes, flash events, misconfiguration, etc.

Robust anomaly detection methods are critical for operators of networks. The daily job of a network operator is to monitor the network to ensure its reliable, continued availability. In the process, the operator needs to wade through massive amounts of information flowing through the network to identify relevant characteristics that would provide a clue about the state of the network. Anomaly detection methods like ours are intended to support the task of the network administrator. By monitoring the network traffic stream and presenting a concise summary view of the traffic, our methods help identify important events.

Our methods operate on network traffic streams. A traffic stream is a sequence of data items that appear over time. Modern networks produce streams that flow at massive rates. Examples of such streams include update streams in social networks, traffic through Internet backbone routers, web search queries, database updates, etc. Our anomaly detection methods allow online detection when given such streams as input, and are designed to enable efficient and scalable computation in order to produce their results.

A basic strategy for anomaly detection on streams is to slice the stream into windows of fixed time intervals, and use a metric to measure some property of the sub-stream in each window. The metric is then monitored over the sequence of windows for significant changes. Our methodology is unique in that it is a unified distribution-based methodology. A distribution here consists of the histogram of items in a window sub-stream. While certain prior work considered individual metrics of the distribution for monitoring, our work takes a holistic approach. We collectively consider several distribution-based metrics. Each of these metrics captures different aspects of the distribution in question, resulting in a higher detection rate. Importantly, the combined inferences from the different metrics allows for automated diagnosis of anomalies when they are found, by serving as signatures for different classes of anomalies.

Our methodology is presented and experimentally evaluated in three application scenarios.

1. In 802.11 wireless networks we show that several kinds of attacks can be detected by monitoring wireless channels, and applying our methodology on the observed streams of frames from these channels.

2. Next, using Wikipedia as an example, we show the anomaly detection capability of

our methodology in update streams generated in information and social networks. From our experiments on our Wikipedia dataset, we were able to identify three kinds of anomalies—the activity of bots, flash events, and outages.

3. Data exfiltration via covert channels is a significant threat to the confidentiality needs of organizations. In our last application, we employ our methodology for detection of covert channels in Voice over IP traffic, and show its efficacy there.

Through these applications we demonstrate the superior detection capability and low rate of false positives of our methodology.

Next, in order to handle the massive rates of streams observed in modern networks, we also provide efficient and scalable algorithmic results for a well-defined distributed data stream model.

The above-mentioned applications also allow us to study several aspects of our methodology. We study how the metrics we choose behave under various types of distributions. This analysis justifies the choice of the metrics we use. Also, the behavior of the different metrics, when considered together, helps in diagnosis of anomalies when they are detected. Our analysis sheds light on the process for such diagnosis, paving the way towards automated classifiers of anomalies. Another aspect of our methodology meriting study is the choice of the various parameters like the window length and the detection threshold. In feedback from the research community early on, the choice of such parameters was regarded as an important aspect of the methodology worthy of study. Our work in this thesis provides an insight into the choice of these parameters.

Our work represents a well-rounded study covering important aspects of the methodology thereby allowing easy applicability by a practitioner. The contributions of this thesis are summarized thus:

- A unified distribution-based methodology for online anomaly detection in network traffic streams.
- Experimental evaluation of the methodology demonstrating its high detection rate and low false positive rate in three application scenarios—wireless anomaly detection, monitoring update streams in information networks, and detection of covert channels in VoIP traffic.
- Algorithmic results enabling efficient and scalable application of the methodology in order to accomodate the massive data rates in modern networks.
- Extensive study of different aspects of the methodology—choice of metrics and their use in anomaly diagnosis, parameters of window length and detection threshold.

We now discuss our results in brief.

Wireless Anomaly Detection

Network traffic monitoring is an integral part of the work at any Network Operations Center (NOC). Such monitoring is essential to maintain the security of the network, and also for troubleshooting problems such as those due to misconfiguration. In wireless networks, hardware manufacturers provide some solutions to enable monitoring, e.g., Cisco CleanAir [Cle] monitors for radio frequency interference. The monitoring of the entropy of "features" of a network traffic stream has previously been shown to be useful in detecting network anomalies [LCD05, LSO$^+$06, NSA$^+$08, LX01, BTW$^+$06, FS03, WP05, XlZB05]. A feature is a property of a network packet or frame that can be computed without additional context. For instance a feature may simply be a field, like the source IP address, in an IP packet. A feature may also be an overall property of the packet/frame, such as the frame length of an IEEE 802.11 frame. The values of a feature over all observed packets in a network form a data stream. Data stream algorithms to estimate the entropy of the stream were given by Bhuvanagiri and Ganguly [BG06], Chakrabarti, Cormode, and McGregor [CCM07], and Harvey, Nelson and Onak [HNO08]. Lall, Sekar, Ogihara, Xu and Zhang [LSO$^+$06] employed data stream algorithms for the efficient and accurate estimation of entropy of network traffic features.

In our work, we consider two approaches for anomaly detection.

First, we consider the distributional metric of the number of distinct elements for detecting wireless anomalies. We show the effectiveness of this metric in detecting three kinds of wireless anomalies. These anomalies include network reconaissance attempts using fuzzing and fingerprinting techniques, and the introduction of fake Access Points. Our dataset for this experiment consisted of wireless traffic captured from a large public building, during which we used software tools to introduce the anomalies.

Second, we consider network features in pairs, and measure the way in which features in each pair depend on each other. Measuring such dependence relations addresses two problems that arise when each feature is considered in isolation and its entropy is monitored for changes. First, the rate of false alarms with entropy monitoring (as with many Intrusion Detection Systems) is generally considered to be a problem. Second, there is the concern that a sophisticated attacker could attempt to mask the effect of his attacks on variability by shaping his injected traffic to mimic the distribution of normal traffic in the network. This would prevent the detection of his attack by an entropy monitoring sensor. Monitoring the dependencies between network traffic features is a promising direction that addresses the above problems. A change in a dependence relationship is classified as an anomaly. Dependence relationships tend to persist more so than individual feature distributions in the course of normal traffic, despite varied activities of network users, so that the rate of false positives is reduced. Further, when monitoring dependence relationships, the job of the attacker trying to mask his attack becomes harder, since it is more difficult to maintain dependency relationships than to maintain the distribution of individual features independently.

To measure dependencies between features we make use of the statistics of conditional entropy, and use some of the recently developed sketching algorithms to monitor these statistics. These statistics indicate the average extent of the dependence of one feature on another. Our experiments with wireless network traffic show that these measures are useful in profiling the traffic for network events, and that their application is efficient and scalable. Full details on these results are provided in Chapter 3 (first appeared in [ABBS10, TMV$^+$11]).

Anomaly Detection in Information Networks

Wikipedia has become a standard source of reference online, and many people (some unknowingly) now trust this corpus of knowledge as an authority to fulfil their information requirements. In doing so they task the human contributors of Wikipedia with maintaining the accuracy of articles, a job that these contributors have been performing admirably. We study the problem of monitoring the Wikipedia corpus with the goal of *automated, online* anomaly detection.

We present Wiki-watchdog, an efficient *distribution-based* methodology that monitors distributions of revision activity for changes. We show that using our methods it is possible to detect the activity of bots, flash events, and outages, as they occur. Our methods are proposed to support the monitoring of the contributors. They are useful in speeding up anomaly detection, and identifying events that are hard to detect manually. We show the efficacy and the low false-positive rate of our methods by experiments on the revision history of Wikipedia. Our results show that distribution-based anomaly detection has a higher detection rate than traditional methods based on either volume or entropy alone. Unlike previous work on anomaly detection in information networks that worked with a static network graph, our methods consider the network *as it evolves* and monitors properties of the network for changes. Although our methodology is developed and evaluated on Wikipedia, we believe it is an effective generic anomaly detection framework in its own right. Full details on these results are in Chapter 4 (first appeared in [AY11]).

Detecting Covert Channels in VoIP Traffic

We study the parameters (knobs) of distribution-based anomaly detection methods, and how their tuning affects the quality of detection. Specifically, we analyze the popular entropy-based anomaly detection in detecting covert channels in Voice over IP (VoIP) traffic.

There has been little effort in prior research to rigorously analyze how the knobs of anomaly detection methodology should be tuned. Such analysis is, however, critical before such methods can be deployed by a practitioner. We develop a probabilistic model to explain the effects of the tuning of the knobs on the rate of false positives and false negatives. We then study the observations produced by our model analytically as well as empirically. Our analysis yields an understanding of the subtle effects that come into play when the knobs are tuned. We examine the knobs of window length and detection threshold. Our results serve as a recipe for the practitioner to follow to achieve a high rate of detection, while maintaining a low rate of false positives. Our recommendations are generally applicable to all distribution-based anomaly detection methods. We also show how the throughput of the covert channel (the magnitude of the anomaly) affects the rate of detection, thereby allowing a practitioner to be aware of the capabilities of the methodology. Full details on these results are in Chapter 5 (to appear in [AYBC12]).

Distributed Monitoring Algorithms

In this thesis we provide efficient algorithms for monitoring the metric of empirical entropy in the particular data stream model called *distributed functional monitoring*. In our distribution-based anomaly detection methodology we use the metrics of empirical entropy and the frequency moments. The frequency moments were studied in the functional monitoring model, and algorithms provided in the work of Cormode, Muthukirshnan, and Yi [CMY08]. In our work we present algorithms for monitoring empirical entropy in the same model. Together these results enable the efficient implementation of our distribution-based anomaly detection methodology.

The model of distributed functional monitoring was first proposed by Cormode, Muthukrishnan and Yi [CMY08] as being useful in the study of the communication complexity of monitoring functions of distributed data streams. In their model, each of k sites reads a stream of tokens and is in communication with a central coordinator, who wishes to continuously monitor some function f of σ, the union of the k streams. The goal is to minimize the number of bits communicated by a protocol that correctly monitors $f(\sigma)$, to within some small error. In previous work the focus was on the classical data stream problem of estimating the frequency moments of data streams. The pth frequency moment F_p is defined as $F_p = \sum_i f_i^p$, where f_i is the frequency of element i in the data stream. Algorithms were provided for solving these problems along with lower bounds showing that the algorithms were nearly optimal in their communication cost. The important point to be noted is that the frequency moments are monotone—as the stream grows in length, these functions never decrease. Non-monotonic functions are more challenging to monitor. We consider the monitoring of entropy, a non-monotone function, in the functional monitoring model. We provide an efficient algorithm for monitoring entropy and show that it is nearly optimal by proving a strong lower bound. More details on these results are in Chapter 6 (first appeared in [ABC09]).

Chapter 2

Background and Related Work

We first introduce distribution-based anomaly detection. Distribution-based anomaly detection methods typically take in as input a stream of data. For instance, in network traffic the stream of observed packets might be the input, while in information networks the input might be the stream of updates to the network. Since monitoring each packet that flows through the network is not possible, metrics measuring aggregate statistics of traffic streams are used to summarize the traffic instead.

Distributional metrics such as entropy are aggregate functions of the distribution (histogram) of traffic features (essentially frame fields) in any given time interval. Lakhina, Crovella and Diot [LCD05] and Nychis et al. [NSA+08] showed the usefulness of monitoring the entropy of network traffic for detecting anomalies such as those due to network intrusion, malfunction, or user behavior. In Chapter 3 we show that monitoring the relations between pairs of features, via the metric of conditional entropy, is also useful for anomaly detection in wireless traffic streams. Distribution-based anomaly detection methods by definition are not effective in the detection of anomalies that do not affect the distribution. For instance, buffer overflows or malformed inputs that have only one-time effects on the stream would not be detected using distribution-based detection. For detecting such cases, other methods would be required.

Choosing good metrics that reveal useful characteristics of the traffic is a hard problem. There is usually a tradeoff to be made between the amount of detail gleaned about the traffic and the amount of information that NOC personnel can comprehend in a timely manner. The entropy of a network feature is a distribution-based metric that has received a lot of research attention [LCD05, NSA$^+$08, LX01].

We now define some terms more formally. Usually, if the traffic stream consists of complex multidimensional values, then one dimension may be selected for consideration as a *feature*. In network traffic, the source IP address is an example of a feature. A feature may also sometimes be an overall property of network packets, like the length of a packet, which is computable from the packet without additional context. We can assume that the feature under consideration may take values from a *universe* $[n] = \{1, \ldots, n\}$, for certain value of n. Then we can consider the *distribution* (histogram) of the feature values in the stream. Distribution-based methods are a class of methods that monitor several aspects of such distributions for significant changes that indicate anomalous activity.

Since the distribution itself is a multidimensional value, a *metric* is usually used to capture some aspect of the distribution. Volume, entropy, frequency moments, and KL-divergence have all been used previously as metrics for comparing distributions in this manner [LCD05, DCRM10, BKPR02, YEG09]. Usually the stream is monitored by slicing it into consecutive *windows* of equal length, either in terms of time or number of elements, and then computing and comparing the metric values in each window. Since each metric captures only certain characteristics of the distribution, in the distribution-based anomaly detection methodology presented in this thesis, several metric and feature combinations are monitored in combination. The results from monitoring those combinations may then be combined using a classifier for a more complete strategy.

Distribution-based anomaly detection is useful to network operators because they are challenged by the high data rates observed in their networks, and are unable to manually comb through the stream for anomalies. Distribution-based methods offer a summarized view of the stream for easy comprehension of network state. Computation of the metric over high-rate streams is a challenge. Data stream algorithms offer computational efficiency in this regard, and have been applied previously to enable efficient monitoring [LSO$^+$06].

Data stream algorithms. Data stream algorithms research largely began with the seminal paper of Alon et al. [AMS99]. Since then the area has received a lot of research attention in line with the explosive growth of information streams on the Internet, e.g., update streams in social networks, traffic through Internet backbone routers, web search queries, or database updates. The core problem is to estimate useful stream statistics in an online fashion. The challenge is to deal with massive stream sizes, and perform memory efficient computation.

A data stream is defined as a sequence $\sigma = (a_1, a_2, \dots, a_m)$, where $a_i \in [n]$. Here, the stream has m items in it, and the items in the stream are drawn from a universe of size n. Then, the problem is that we would like memory-efficient compututation of some metric $f(\sigma)$. The metrics f that we consider depend only on the frequency distribution of the items in the stream. In particular, f is invariant under permutations to σ. Data stream algorithms research is concerned with the computation of metrics f in a memory efficient manner, without having to store either the entire stream or the entire frequency distribution in memory at any given time: consider that the length of the stream is massive (e.g., packets arriving at a backbone router), and its universe is also massive (e.g., the universe of web search queries). The metrics we consider in this thesis are presented next, along with a note of some of the algorithms available. The algorithms are probabilistic approximation algorithms; i.e., with high probability these algorithms produce values that are close to their target metric values.

Metrics. We now describe the metrics we use in this thesis, and some of their properties.

- Length (volume) of the stream, m: This is a simple metric that has been considered for anomaly detection in early research [BKPR02]. This metric is also useful for normalizing the values of the other metrics.

- Entropy, $H(\sigma)$: This metric has received a lot of attention in the context of network traffic monitoring for anomalies. Let f_i denote the count (or frequency) of item $i \in [n]$ in stream σ. If X denotes the random variable corresponding to the empirical distribution of σ so that the induced probablilty of i,

  $$p_i := \Pr[X = i] = f_i/m, \quad \forall i \in [n],$$

  then the entropy of σ is given by $H(\sigma) = -\sum_{i \in [n]} p_i \log p_i$. Sometimes we denote the entropy as being a function of the random variable X instead. That is, we write it as $H(X)$.

  Entropy is an information-theoretic metric that captures certain aspects of the shape of the distribution. The entropy of a distribution is a maximum of $\log m$ when the frequencies of elements are uniform (but its value depends on the number of elements in the support of the distribution). The introduction of elements of high frequency into the distribution causes the entropy to decrease, while an increase in the number of elements in the distribution usually results in an increase in entropy. Many anomalies exhibit such effects, as shown by prior research, and also by experiments in this thesis.

- Conditional entropy, $H(X|Y)$: Frequently, streams observed are multidimensional. Consider stream σ_{XY} with two features X and Y, so that each stream item consists of a value for feature X (from a universe, say $[n_1]$), and another value for feature Y (from a universe, say $[n_2]$). Let σ_X and σ_Y denote the individual data streams corresponding to the two features of the observed network traffic stream. Let X and Y respectively also denote the corresponding random variables, and let $Z = (X, Y)$ be the random variable corresponding to the joint distributions of X and Y (note that the two variables X and Y may not be independent, because the underlying experiment is that of sampling from the multidimensional stream σ_{XY}). Then the conditional entropy of X given Y is

$$H(X|Y) = H(Z) - H(Y) = H(X, Y) - H(Y),$$

  where $H(X, Y)$ is computed using the joint probabilities $p_{ij} := \Pr[X = i, Y = j], \forall i \in [n_1], \forall j \in [n_2]$.

  The conditional entropy indicates the extent to which the value of the random variable X can be predicted in the stream, when given the value of random variable Y. The quantity $H(Z) = H(X, Y)$ is known as the joint entropy of random variables X and Y. Figure 2.1 shows the relationship between these entropies. The quantity $I(X; Y)$ in the figure is the statistic of mutual information between X and Y and is defined as $I(X;Y) = H(X,Y) - H(X) - H(Y)$. For an excellent treatise on information theory and more information on these statistics we refer the reader to Cover and Thomas [CT91].

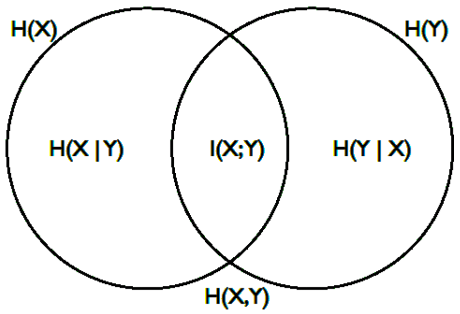


Figure 2.1: Relationship between various entropies. Intuitively, values of these entropies are given by the respective areas.

- Zeroth frequency moment, $F_0(\sigma)$: The value of this metric is the support size of the distribution, i.e., the number of elements i in the distribution with frequency $f_i > 0$. This metric is useful because it can identify anomalies where new elements appear in (or old elements disappear from) the distribution.

- Second frequency moment, $F_2(\sigma)$: The value of this metric is given by $F_2(\sigma) = \sum_{1 \le i \le n} f_i^2$. This metric enables us to identify when the distribution gets skewed towards a few elements in the support. This follows from the inequality $(a+b)^2 > a^2 + b^2$, whenever $a, b > 0$.

  The frequency moments F_0 and F_2 are generalized as $F_p = \sum_{1 \le i \le n} f_i^p$, for $p \ge 0$ (assuming $0^0 = 0$, for F_0). Note also that $F_1 = m$ (the volume).

Data stream algorithms to estimate the frequency moments and entropy are presented in the works of Alon et al. [AMS99] and Harvey et al. [HNO08] respectively.

Section 4.2 provides a more detailed insight into the behavior of the metrics we consider in this thesis.

Related Work

Information-theoretic statistics (like entropy) when applied to monitoring of network traffic have been shown to be very useful in detecting changes in its character, as first suggested in [LX01] and then developed in [LCD05] and other works (e.g., [BTW$^+$06, FS03, LSO$^+$06, NSA$^+$08, WP05, XlZB05]). While possessing the significant advantage of needing few modeling assumptions about what constitutes normal and abnormal traffic, information-theoretic methods incur the substantial computational cost of having to compute entropy and related statistics, which presented an obstacle to their practical adoption. Recent advances in streaming estimation algorithms for entropy [BG06, CCM07, HNO08, LSO$^+$06] pave the way to overcoming this obstacle. We further discuss data stream algorithms and their use in our setting in Chapter 3 and in Section 5.4.

Reducing the computation cost of entropy estimation opens the way not only for single-variable entropy-based anomaly detection, but also for analysis of pairwise feature dependencies, which can be an efficient anomaly detection tool. Nychis et al. [NSA$^+$08] commented that computing entropy of several features over time epochs, and tracking the correlation between the entropy of pairs of features over a window of time epochs might be a useful approach for anomaly detection. Our approach in Chapter 3 is different, because we do not monitor the correlation between entropy, but rather track the conditional entropy. This is a direct statistic of the concerned feature distributions, and not one of entropies of the feature distributions. Changes in the nature of network traffic are likely to manifest directly in the conditional entropy. The computation of statistics of pairs of features is computationally heavy and we also provide an approach for managing it in a distributed manner.

The survey by Chandola et al. [CBK09] presents a classification of anomaly detection methods by application and technique. Our techniques do not neatly fall into any specific class therein. Distribution-based metrics previously used individually for anomaly detection include volume [BKPR02], entropy [LX01] [LCD05], and KL-divergence [DCRM10]. In this thesis, we make use of the metrics of volume, entropy, conditional entropy, and the frequency moments. In Chapter 4 we consider several of these metrics together, and use their combined predictions to obtain a stronger detection capability.

Work related to information networks. Leskovec [Les08] studied time-evolving properties of networks, developed models governing that evolution, and also suggested anomaly detection in evolving networks of the kind we present in this thesis. There have also been other works concerned with various aspects of evolving networks. Almeida et al. [AMC07] study the evolution of the Wikipedia network with the goal of understanding behavior of users, article revisions, and processes behind them. Sun et al. [SFPY07] provide methods to detect community structures and their evolution in networks.

Prior work related to parameter selection. Work on selecting parameters for testing time series tends to be domain specific. In stock market forecasting [CG04], minor variations in parameter specifications have a big effect on the predictability performance, and it was noted that there is little explicit guidance from theory regarding the in-sample window used in the forecast model. In detecting shilling attacks on recommendation systems [ZCFM06], an optimal window size is derived when the number of attack profiles is known, and a heuristic to estimate the number of attack profiles and adaptively adjust the window size is proposed. In medical diagnosis [GH10], entropy analysis has been shown to be useful in detecting Alzheimer's disease with window length determined based on spectral analysis. In network anomaly detection, it has been shown [LX01] that a variable-length window is a more effective strategy for sendmail anomalies based on monitoring conditional entropy. In this thesis, we study the effects of parameter selection specifically in the setting of detection of covert channels in VoIP traffic using distribution-based methods. Due to the nature of our analyses, however, our recommendations tend to be generally applicable to all distribution-based anomaly detection methods.

Prior work related to covert channels. The survey of Giani et al. [GBC06] provides a taxonomy of covert channel designs. We consider only storage covert channels that use packet header fields for detection in this work. This class of channels is known to be practical for use, because of the high bandwidth provided by the channels. However, Huang et al. [HTY11] present a VoIP covert channel using steganography that claims to have a high capacity. Methods for detecting covert channels include those based on entropy [GW07] (but these are specifically only applicable to timing-based channels) and n-gram statistical analysis [CCG11].

Chapter 3

Wireless Anomaly Detection

Reliably detecting attack-related anomalies in network traffic is known to be a hard problem. There are many kinds of attacks that are likely to manifest themselves as traffic anomalies. Some examples of such attacks are: a denial of service (DoS) attack, interference or jamming by an attacker, “scanning” activity by reconnaissance tools or botnet nodes, and covert channels that make unintended use of (unused) features of network protocols to communicate data out of a network. Several approaches in recent research have focused on monitoring various statistics of the traffic stream [LCD05, KGKM05, XlZB05]. The property desired of such a statistic is that it should capture a characteristic of the traffic distribution that would reveal anomalies. The statistic value is monitored for deviations from “normal” values as ascertained by training with prior observed data and continuous adjustment by personnel at the Network Operations Center (NOC). Several features of the observed stream may be monitored together to improve reliability of detection.

In this respect, the use of the information-theoretic statistic of empirical Shannon entropy (or simply entropy) was pioneered by Lakhina, Crovella, and Diot [LCD05], and has also received a lot of attention in similar contexts [LSO$^+$06, NSA$^+$08, LX01, BTW$^+$06, FS03, WP05, XlZB05]. Computing entropy in the straightforward manner, by maintaining counters to keep track of the distribution histogram, is expensive with respect to the memory and computation required. Lall et al. [LSO$^+$06] showed that computing entropy either deterministically or precisely requires linear space, therefore in these cases no non-trivial savings can be achieved over naïve computation. Thus, one has to consider randomized approximation algorithms to have any hope of computing entropy efficiently. This problem of monitoring entropy efficiently has received widespread interest in the theory community, and several data streaming algorithms are now known [ABC09, BG06, CCM07, CDM06, GMV06, HNO08, LSO$^+$06] that use significantly less memory than straightforward computation. However, even the proposed streaming algorithms fail to keep up with ever-increasing channel bandwidths observed at a typical NOC of a large organization, especially considering that monitoring of several stream features is desirable for accurate detection.

Our contributions. We refer the reader to the abovementioned previous research for the strong justification of entropy's usefulness in network anomaly detection. In this chapter, we concentrate on making such entropy estimations on real traffic scalable and efficient, and extending them to conditional entropy and the number of distinct elements. In particular, we

- implement and evaluate a distributed sketching algorithm, and compare it with a previously studied streaming estimation algorithm;
- show that it computes pairwise-conditional entropy with reasonable accuracy;
- make a case study of conditional entropy successfully detecting an attack-related anomaly more efficiently than single-feature entropies; and

- demonstrate the use of the metric of the number of distinct elements in detecting several different wireless anomalies in a large scale production network.

We now discuss these contributions in more detail.

Sketching algorithms

In this work, we take advantage of a useful feature of some of the streaming algorithms mentioned previously. The algorithms of Bhuvanagiri and Ganguly [BG06] and Harvey et al. [HNO08] are in a class of algorithms called *sketching algorithms*. A sketching algorithm is a streaming algorithm that maintains a data structure called a *sketch*. A sketch can be combined with other sketch instances corresponding to other streams to obtain a "global" sketch for the union of the streams under consideration. As we demonstrate in the next section, this property is useful when trying to make our monitoring algorithms distributed, scalable and hence practically usable.

The case for conditional entropy

Even given algorithms for monitoring entropy efficiently and scalably, there are two challenges while deploying anomaly detection systems. First is the high rate of false positives. Monitoring entropy is subject to the activity of the users in a network, and it is difficult to have a reasonable model of its "normal" range even with continuous adjustment by dedicated personnel. For instance, at a university, when students in a sizable class are required to download a large software package, the high bandwidth usage pattern might well trigger a false alarm. Second, it is conceivable that an attacker would attempt to mask his effect on network traffic by trying to mimic the normal distribution of features in the packets he introduces or even train the intrusion detection system gradually to accept attack traffic as normal [Den87]. Both of these problems somewhat take away from the benefit of monitoring entropy of traffic distributions.

To overcome these obstacles, we propose monitoring the related statistic of *conditional entropy* (see Chapter 2 for a formal definition), which tracks the average dependence of one feature on another. The value of conditional entropy indicates the extent to which the first feature can be predicted given the value of the second feature, or, more precisely, how much uncertainty about the value of the second feature remains once the value of the first one is known. Monitoring this relationship between features is a promising direction in network anomaly detection. A change in the "predictability relationship" raises the possibility of suspicious activity. This dependence tends to persist in the course of normal traffic even through varied activities of network users, such as in the university example mentioned previously, so that the rate of false positives is minimized. Monitoring conditional entropy also has the added benefit in that it makes the job of masking by the attacker harder. Maintaining dependencies between features while at the same time carrying out an attack is harder than just maintaining the distribution of features independently. Also, the attacker would have a harder time (quadratically) guessing which pairs of features the defender has chosen to monitor.

Need for distributed, scalable algorithms. With pairs of distributions for conditional entropy, the amount of processing and memory required is effectively increased quadratically. This further motivates the requirement of a scalable, distributed monitoring scheme. In response to this challenge, we produced and evaluated the first implementation of a sketching algorithm that naturally lends itself to a distributed implementation, namely, the Hierarchical Sample Sketch algorithm of Bhuvanigiri and Ganguly [BG06]. We also compared its accuracy and performance with that of our implementation of the Lall et al. algorithm.

Evaluations with 802.11 link layer

We evaluate the strength of our approaches by experimenting with link-layer headers collected in realtime from an 802.11 network.

The 802.11a/b/g link layer is feature-rich and complex, and therefore allows a range of interesting attacks with corresponding statistical distribution anomalies. In particular, it includes large (2346–2358 bytes) management frames that can contain an entire ring 0 exploit in their L2 payload alone[1]. Besides the frame type and subtype fields, the link layer header may contain 1–4 MAC address fields, eight 1-bit flags (two of which affect interpretation of the address fields), and two 16-bit fields, frame sequence number and duration (the distribution of which has been shown [Cac] to identify wireless chipset–driver combination as a distinctive fingerprint). The earlier WEP (Wired Equivalent Privacy) encryption implementations could be leveraged to leak encryption key bits by responding to specially crafted injected frames, using for example the so-called KoreK attack, which we chose as an example to showcase our methods. We use this attack in our evaluations to demonstrate the effect on our entropy-based monitoring methods of the kind of anomalies it introduces in network streams.

To the best of our knowledge, our experiments with monitoring (conditional) entropy are the first application of streaming entropy estimation methods to 802.11 wireless headers.

The remainder of the chapter is organized as follows. In Section 3.1, we detail how

[1]Demonstrated by Ellch and Maynor at BlackHat 2006 for a vulnerability in MacBook's Wi-Fi kernel driver

conditional entropy can be monitored to analyze network traffic for anomalies. Then, in Section 3.2, we detail our evaluations of the proposed approach by testing our implementation with real data from a wireless network. Finally in Section 3.3 we demonstrate the applicability of the metric of the number of distinct elements, F_0, in anomaly detection.

3.1 Estimating and Monitoring Conditional Entropy

Before we describe our methodology for monitoring of conditional entropy, we need some definitions. We first define the kinds of error in the estimates produced by streaming algorithms.

Definition 1 (Error types)**.** If $\hat{Y}$ denotes the estimate produced by a randomized streaming algorithm computing a statistic with the exact value Y, then the error is said to be a *multiplicative ε-error* with probability δ if

$$\Pr[|\hat{Y} - Y| > \varepsilon \cdot Y] \leq \delta,$$

and it is said to be an *additive ε-error* with probability δ if

$$\Pr[|\hat{Y} - Y| > \varepsilon] \leq \delta,$$

where the probabilities are over the random bits used by the algorithm.

Thus for at least δ fraction of random bit sequences supplied to the algorithm, it will produce an accurate estimate depending on the parameter ε. The space and (update and estimation) time required by the algorithm typically depend on these parameters, increasing as ε and δ are made smaller. See Table 3.1 for actual space bounds of the algorithms we consider.

We now formalize the notion of sketching algorithms that we had described earlier.

Definition 2 (Sketching algorithm)**.** A *sketching algorithm* is a streaming algorithm that, on observing stream σ, maintains a data structure $\mathcal{D}(\sigma)$ called a sketch, with the following property. If $\mathcal{D}(\sigma_1)$ and $\mathcal{D}(\sigma_2)$ are the sketches corresponding to two independent instances of the algorithm observing streams σ_1 and σ_2 respectively, then given $\mathcal{D}(\sigma_1)$ and $\mathcal{D}(\sigma_2)$, it is possible to obtain $\mathcal{D}(\sigma_1 \cup \sigma_2)$ efficiently (time linear in the size of the sketches).

3.1.1 Estimating conditional entropy

As we noted in the introduction, monitoring conditional entropy has several benefits towards the goal of anomaly detection. In this section we describe how this monitoring is achieved. Although several streaming algorithms are available (see Table 3.1) for estimating entropy, the case for conditional entropy turns out to be different. Indyk and McGregor [IM08] showed that estimating conditional entropy with ε-multiplicative error requires $\Omega(m)$ space, so that no non-trivial savings are possible over naïve computation even with randomized, approximation algorithms. However, it is possible to obtain ε*-additive* error estimates of conditional entropy using streaming algorithms that estimate entropy multiplicatively, at the expense of some additional factor increase in space. To get an ε-additive

Algorithm	Space requirement
Naïve	$O(m \log n)$
Lall et al. [LSO+06]	$O((1/\varepsilon)^2 \log(1/\delta) \log m \log n)$
Bhuvanagiri & Ganguly [BG06]	$O((1/\varepsilon^3) \log(1/\delta) \log^5 m)$
Harvey et al. [HNO08]	$O((1/\varepsilon)^2 \log(1/\delta) \log m \log n \log(mn))$

Table 3.1: Comparison of theoretical space bounds of entropy estimation algorithms.

error approximation of conditional entropy $H(X|Y)$, one can run entropy estimation algorithms to get accurate estimates of $H(X, Y)$ and $H(Y)$, with multiplicative error parameter $\varepsilon' = \varepsilon/(2\log m)$. The estimate $\hat{H}(X|Y) = \hat{H}(X, Y) - \hat{H}(Y)$ is then returned. We have

$$\begin{aligned}
|\hat{H}(X|Y) - H(X|Y)| &= |(\hat{H}(X, Y) - H(X, Y)) - (\hat{H}(Y) - H(Y))| \\
&\le |\hat{H}(X, Y) - H(X, Y)| + |\hat{H}(Y) - H(Y)| \\
&\le \varepsilon H(X, Y)/(2\log m) + \varepsilon H(Y)/(2\log m) \\
&\le \varepsilon.
\end{aligned}$$

The last bound follows because both entropies are at most $\log m$. This shows that our error is indeed an ε-additive error.

To estimate the entropies $H(X, Y)$ and $H(X)$ we could use any of the algorithms listed in Table 3.1. In Section 3.2 we evaluate and compare implementations of both the Lall et al. algorithm and the Hierarchical Sample Sketching (HSS) algorithm of Bhuvanagiri and Ganguly, but we focus our attention on the latter since it is in addition a sketching algorithm. For the sake of completeness, we next give a short outline of this algorithm.

3.1.2 A brief overview of the HSS algorithm

At a basic level, the HSS algorithm maintains a set of counters organized hierarchically into levels. Each time an item is observed in the stream, the algorithm employs a clever sampling scheme that propagates the item to some of those levels, and a counter at each of the chosen levels is incremented. Pairwise-independent universal hash function families [CW77] are used at each level to decide which counter at this level is to be incremented. The final estimate is computed using the values of all counters grouped in a special manner.

Figure 3.1 gives an idea of the organization of counters in the HSS sketch and of how an item appearing in the stream updates the sketch data structure. For more specific details on the algorithm we refer the reader to Bhuvanagiri and Ganguly [BG06].

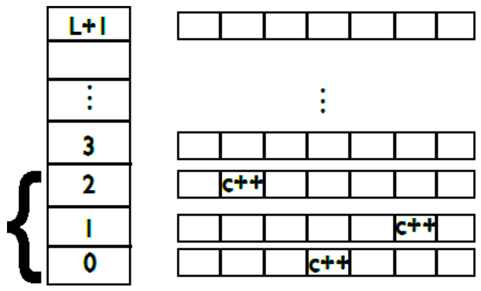


Figure 3.1: The HSS sketch. The levels a certain item is propagated to are marked by the brace. The corresponding counters incremented are marked "c++".

3.1.3 Distributed monitoring

As already mentioned, since the computational time available per item is extremely limited in high bandwidth channels, it is highly desirable to run a monitoring system in a distributed fashion. In fact, it is likely that this is the only practical way of building actual streaming-estimation based intrusion-detection systems.

We now describe how the *sketching property* of the HSS algorithm can be used towards this goal. It is easy to see why the data structure maintained by the algorithm (see Figure 3.1) is indeed a sketch. For if two such data structures corresponding to two streams are provided, the new structure obtained by simply adding corresponding counters in the two structures is the data structure corresponding to the union of the two streams under consideration. This is so because the set of counters incremented when a value appears in one stream is the same as the set of counters incremented when the same value appears in the other stream.

To use this property in a distributed monitoring scheme, we have a load balancer at the NOC that splits the observed stream of packets evenly over a set of sensors and forwards the packets to those sensors. Each sensor has the job of monitoring the (conditional) entropy of the forwarded stream. To get global estimates of the entropy, sensors forward their counter values (sketch) to an accumulator node that is able to use the sketching property to return entropy estimates. This scenario is shown in Figure 3.2. In an actual network, the accumulator may be co-located with the load balancer or with one of the sensors.

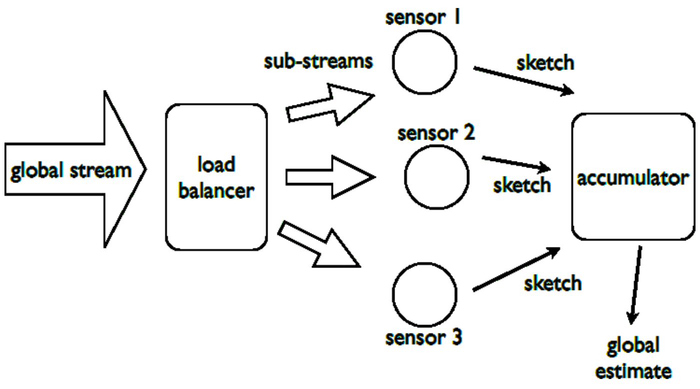


Figure 3.2: Distributed monitoring of (conditional) entropy using sketching.

3.2 Evaluations and Observations

In this section we present the results of our evaluations of the techniques mentioned in previous sections. We first describe the setup behind our evaluations.

Datasets. Our datasets consist of 802.11 headers collected by Air Monitors (AMs) distributed across the Computer Science department at Dartmouth College. For a technical description of our infrastructure see Sheng et al. [SCT^{+}08]. The AMs are sniffers that monitor several 802.11 channels for frames and forward the link-layer headers from the sniffed frames to a server. The headers arriving at the server are anonymized to protect MAC addresses by applying a pre-generated random one-to-one mapping so that distinct MACs never collide.

In our evaluations we use two traces collected in this manner, denoted further as Trace 1 and Trace 2. For both traces we used four AMs to collect data. Trace 1 was collected over 56 continuous hours of normal network activity. Trace 2 was collected over 41 hours, and apart from normal traffic this trace contained the data sniffed by an AM while we conducted the KoreK attack [inf09] on a setup in the same vicinity. We give more details on the attack in Section 3.2.2. Our infrastructure collected data in both traces at the average rate of about 130,000 frames every 10 minutes. We bin the data in each trace into disjoint windows of 100,000 frames each. On average, each window represents a time frame of about 7 minutes.

Distributions. In our evaluations we consider three features for entropy monitoring: *source MAC address* (last 2 bytes), *frame length*, and *duration/ID*. This gives a total of 6 combinations for computing the conditional entropy $H(X|Y)$ when X and Y are taken to be random variables representing different feature distributions.

Whenever we present values of (conditional) entropy, they are normalized by a factor of $1/\log m = 1/\log(100000)$ to allow for comparisons of these values between different pairs of features.

In all evaluations, we tested our algorithms on a server with four Intel x86_64 Xeon processors at 3.0 GHz, 4GB of RAM, running Linux kernel 2.6.22.14.

3.2.1 Performance of algorithms in conditional entropy estimation

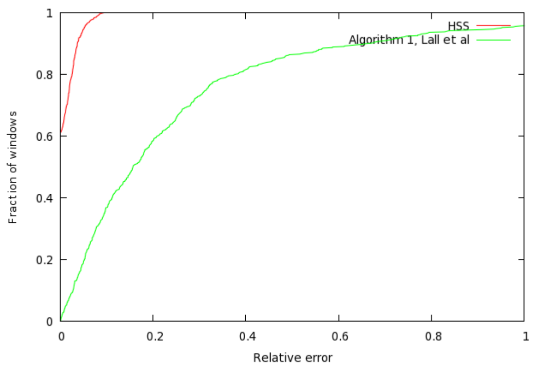
We now detail the performance in conditional entropy estimation of the two algorithms we consider: the HSS algorithm and Algorithm 1 of Lall et al. [LSO$^+$06] (henceforth referred to as LSO1). Our tests in this section were performed on Trace 1 and follow along the lines of the evaluations of Lakhina et al. [LSO$^+$06]. We evaluated the performance of the algorithms with respect to the accuracy of the algorithms and the memory used in the algorithms. We evaluate the accuracy by determining the relative error in computing conditional entropy over each window. The relative error in computing $H(X|Y)$ is $|H(X|Y) - \hat{H}(X|Y)|/H(X|Y)$, where $\hat{H}(X|Y)$ is the estimate returned by the algorithm. The memory usage of the algorithms is a function of the parameters ε and δ, since these determine the number of counters used in the algorithms. When comparing the relative errors between the two algorithms, we choose ε and δ in the two cases so that the number of counters is the same for both algorithms.

Before detailing the results of our performance evaluations it is useful to note what the values of conditional entropy look like for the different feature distributions over the windows in the trace. As we will see, the relative error in estimates produced by the algorithms depend on the magnitude of conditional entropy. This result is not surprising, since the allowed relative error grows as the value of estimates gets smaller, when the algorithm only guarantees bounds on additive error. Table 3.2 lists the mean values of conditional entropy over the windows, for each combination of features. We now show how the relative errors of the estimates vary with different features and different number of counters used in the algorithms. In Figures 3.3, 3.4, and 3.5 we plot the cumulative distribution function (CDF) of the relative error, showing in what fraction of windows the algorithms produce small relative error. Doing this for different numbers of counters shows what minimum

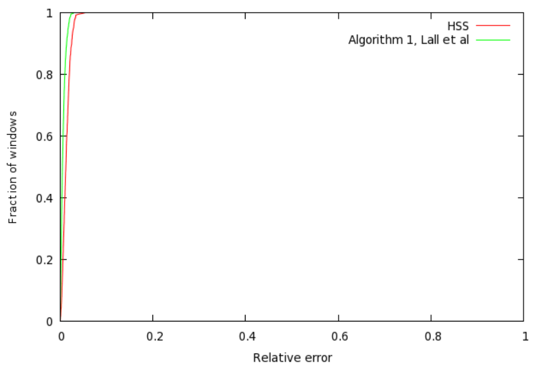
	Frame length	Duration/ID	Source MAC
Frame length	–	0.115	0.0105
Duration/ID	0.00351	–	0.00621
Source MAC	0.107	0.214	–

Table 3.2: Mean values of actual conditional entropy $H(X|Y)$. Rows represent variable X, and columns represent variable Y.

number of counters is necessary in each case. For the sake of brevity, we include only plots where the median relative error is at most 10%. *The first observation we can draw from these figures is that we can estimate conditional entropy accurately using the HSS algorithm, since in each case the relative error is at most 10% for a majority of windows and reaches a maximum of 20% in a small fraction of cases.* We also observe that the HSS algorithm does consistently better in producing accurate estimates as compared to LSO1. We attribute this to the fact that the error guarantees of LSO1 require assumption of certain bounds on the actual values of entropy. In our cases where the normalized conditional entropy is small, these assumptions may not hold, and indeed we observe that for the cases of Duration/ID given Source MAC, Frame length given Source MAC, and Duration/ID given Frame length, where the conditional entropy was at most 0.01 (see Table 3.2), the difference in accuracy between the two algorithms particularly stands out. In these cases, while the maximum relative error of the HSS algorithm is about 30%, the error due to LSO1 is $> 100\%$. Lastly, we observe that the number of counters required to compute estimates with reasonable error varies by the choice of features. To demonstrate this better we vary the number of counters used in estimating each feature pair and plot the mean relative error for the two algorithms. Figures 3.6, 3.7, and 3.8 show these plots. It is easily observed that the number of counters required to obtain the same level of accuracy is much higher in the three cases mentioned above where the true conditional entropy values are low. *Also the*

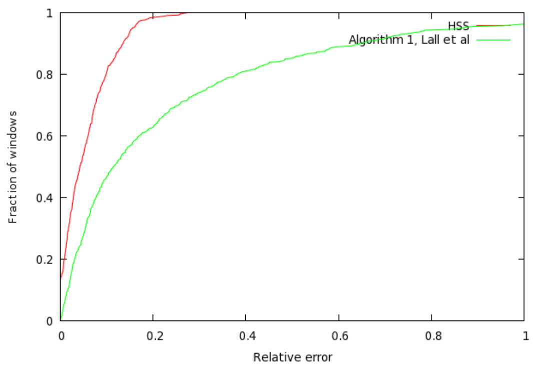


(a) $H(X|Y)$, 40,000 counters

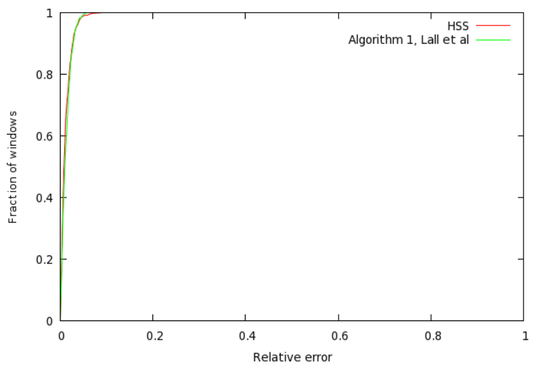


(b) $H(Y|X)$, 20,000 counters

Figure 3.3: Relative error of estimates, $X \equiv$ Duration/ID, $Y \equiv$ Src MAC



(a) $H(X|Y)$, 40,000 counters



(b) $H(Y|X)$, 25,000 counters

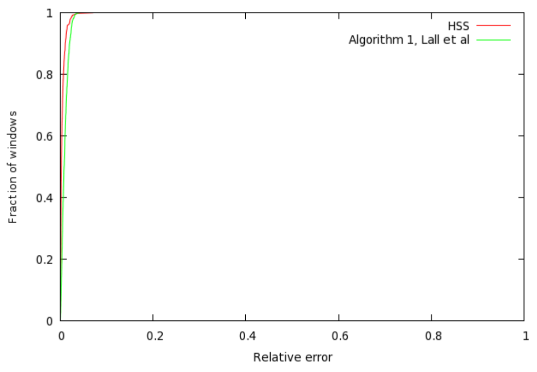
Figure 3.4: Relative error of estimates, $X \equiv$ Frame length, $Y \equiv$ Src MAC

HSS algorithm is able to achieve a mean relative error of 10% in almost all cases using 25,000 counters, while LSO1 is not able to achieve this kind of accuracy at all in the cases when conditional entropy is low.

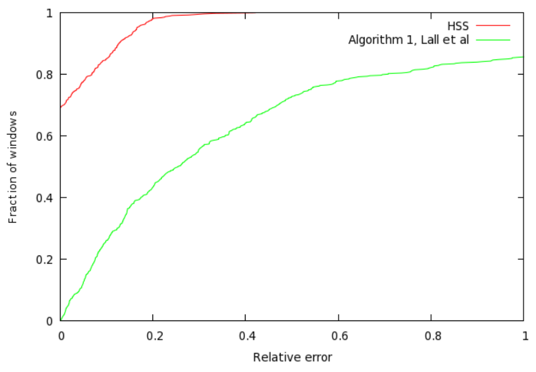
3.2.2 Detecting attacks with conditional entropy

We now look at how our methodology for detecting anomalies fares in the presence of a known attack.

For the purpose of demonstrating our methods, we chose the so-called KoreK attack

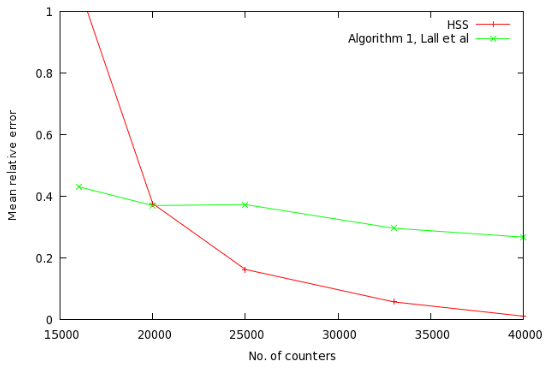


(a) $H(X|Y)$, 16,000 counters

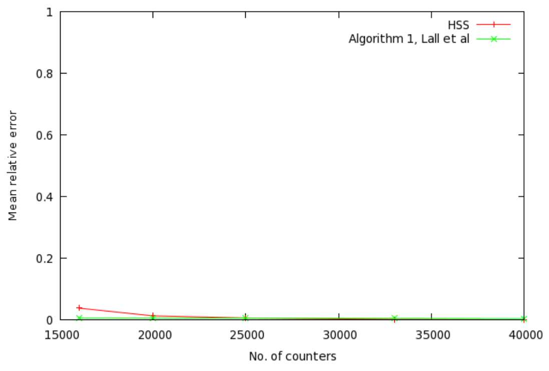


(b) $H(Y|X)$, 40,000 counters

Figure 3.5: Relative error of estimates, $X \equiv$ Frame length, $Y \equiv$ Duration/ID



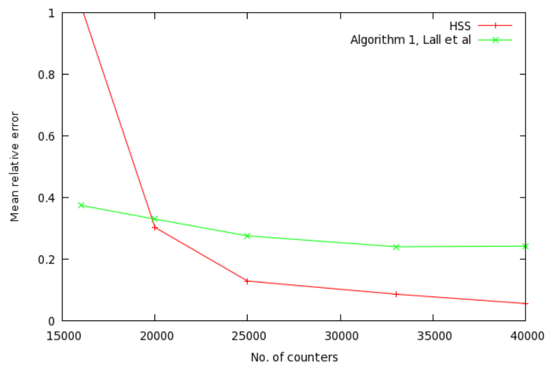
(a) $H(X|Y)$



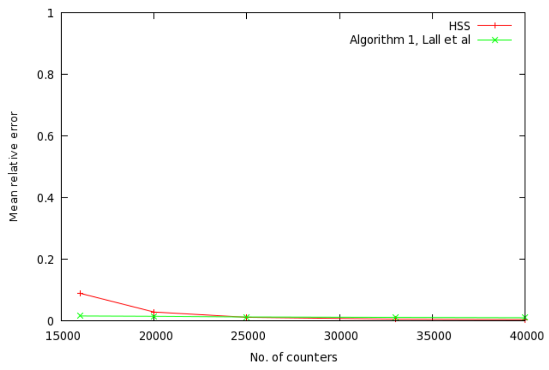
(b) $H(Y|X)$

Figure 3.6: Mean relative error with varying numbers of counters, $X \equiv$ Duration/ID, $Y \equiv$ Src MAC

on WEP. Although WEP should be considered antiquitated as a security measure, for our purposes this attack has a number of exemplary properties. In particular, it attacks a cryptographic system via a weakness in its particular implementation specific to a class of devices (a significant practical concern for 802.11 equipment, as successful attacks have shown); it relies on the attacker's capability to inject custom-built 802.11 frames including malformed frames; it does not recover the encryption key, but rather gives the attacker access to sensitive information about the network. Thus we regard this attack as a model for possible future attacks to come.

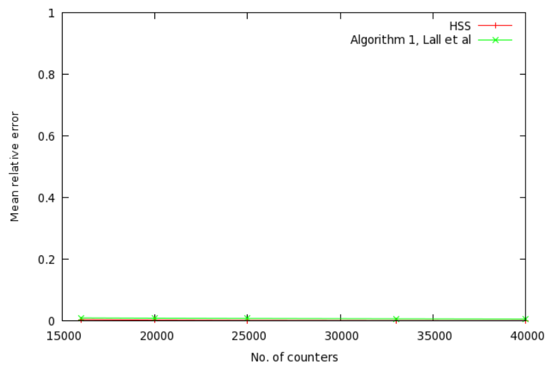


(a) $H(X|Y)$

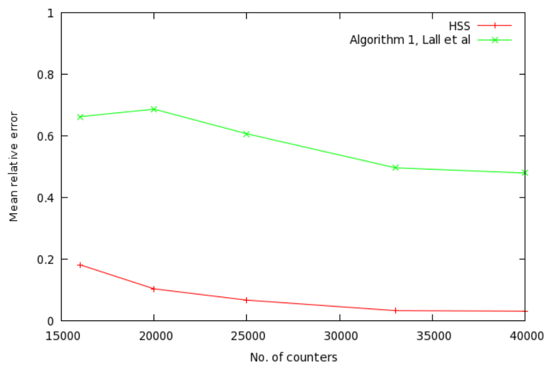


(b) $H(Y|X)$

Figure 3.7: Mean relative error with varying numbers of counters, $X \equiv$ Frame length, $Y \equiv$ Src MAC



(a) $H(X|Y)$



(b) $H(Y|X)$

Figure 3.8: Mean relative error with varying numbers of counters, $X \equiv$ Frame length, $Y \equiv$ Duration/ID

The KoreK attack. The KoreK attack attempts to decrypt a WEP data packet by guessing at the plaintext byte by byte, starting from the end of the packet [inf09] and making up to 256 guesses at each step. Attempting a guess, the attack produces a packet, which is a valid encrypted packet if the guess is correct, and transmits this packet onto the wireless network. If the packet is valid, the AP will resend it onto the network; if it is invalid, the AP will drop it; the attacker is thus able to discover the last plaintext byte. As soon as the packet is accepted and resent by the AP, and thus the last plaintext byte is found, the attack chops it off and continues working with the packet 1 byte shorter. Thus the KoreK attack generates different-sized clusters of packets of the same size, the size of the packets reduced by 1 byte from cluster to cluster. Since a number of header field values in the resulting stream end up being more uniformly distributed during the attack than in the absence of the attack, corresponding entropic measures show a spike when the KoreK attack is executed.

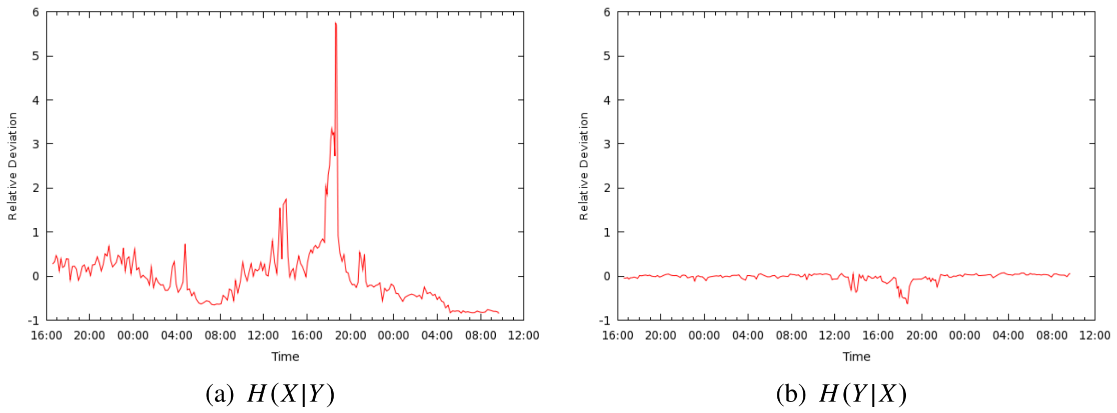
We used the KoreK functionality of the *aireplay-ng* tool [Kor] to decrypt packets on our encrypted wireless network.

We captured the resulting traffic in Trace 2 and we now present the results of running the HSS estimation algorithm on that trace. Figure 3.9 shows the relative deviation in the conditional entropies from the median value over windows of Trace 2 when the features of Frame length and Source MAC are considered. On the other hand, Figure 3.10 shows the relative deviation of the separate *entropies* of these features. We chose the median as the measure of central tendency to ignore the deviations due to the KoreK attack. The spikes due to the KoreK attack are visible by observation around the window at 18:30 hours on the second day in all cases. However, the conditional entropies are affected the most, as can be observed from the prominence of the spikes. This result is due to the fact that the KoreK attack causes large variability among lengths of frames that originate from a small set of MACs (i.e., those involved in the attack).

Table 3.3 shows that the attack disturbs the conditional entropy of Frame length given the Src MAC the most —570%— whereas the individual entropies are affected only modestly by up to 50%. *This confirms our hypothesis that conditional entropy, capturing the dependence between feature pairs, could be a more useful statistic in detecting anomalies.*

Feature(s)	Statistic median	For KoreK attack	Relative change
Frame length, given Src MAC	0.0192	0.129	+572%
Src MAC, given Frame length	0.211	0.0821	-61%
Src MAC	0.374	0.240	-36%
Frame length	0.187	0.287	+53%

Table 3.3: Relative change in (conditional) entropy in the KoreK attack

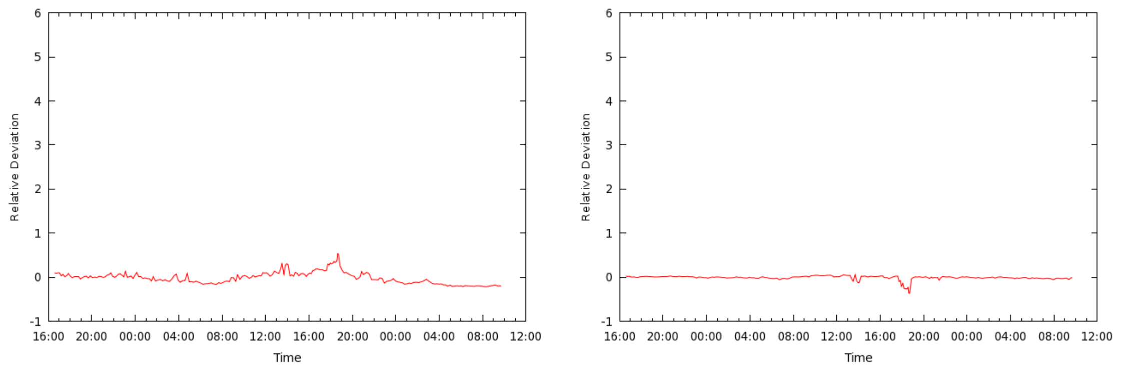


(a) $H(X|Y)$ (b) $H(Y|X)$

Figure 3.9: Relative deviation of conditional entropy estimates from median, $X \equiv$ Frame length, $Y \equiv$ Src MAC

3.3 Monitoring the Number of Distinct Elements for Anomaly Detection

We now present the results of our experiments with monitoring of the number of distinct elements, F_0, in the face of wireless anomalies. Our experiments here also have a wider scope and scale in terms of the size of the network we work with—we make use of the DIST (Dartmouth Internet Security Testbed) [BKT+09] infrastructure. This consists of about 200 wireless Air Monitors distributed across the Dartmouth College campus gathering 802.11 wireless headers from the observed traffic generated in the production campus wireless network. The headers are then stored on servers in a suitably anonymized form.



(a) $H(X)$ (b) $H(Y)$

Figure 3.10: Relative deviation of entropy estimates from median, $X \equiv$ Frame length, $Y \equiv$ Src MAC

Experiments

We present a summary of results of our experiments of distribution-based approaches to wireless traffic monitoring using the DIST infrastructure.

Method. As before, we partition the observed traffic into windows of equal time length, and monitor distributions of features (e.g., BSSID) in the windows using metrics. While we consider several metrics in our work that capture different aspects of the "shape" of the distribution, here we present results only using the number of distinct elements, F_0 (see Chapter 2 for a definition), in the feature stream. We found some of the anomalies also reflected while monitoring for entropy of some of the features. However, some of our metric-feature combinations did not show any visible effects. Furthermore, some metric-feature combinations exhibited the diurnal traffic cycles as the predominant effect, and other metric-feature combinations seemed to show anomalies, where we had not injected any. Understanding the reasons for these unknown anomalies was infeasible given only the Layer-2 headers. In the future, with possible permission to gather IP, TCP and UDP headers, it may be tractable to understand the causes of such embedded anomalies.

Dataset. To evaluate this method, we collected a 3-day dataset of wireless traffic collected by DIST between May 2 and May 4, 2010. For this experiment, we consider data from 52 AMs located in the Dartmouth College Library. During this period we injected four types of anomalous traffic using standard tools from the BackTrack Linux distribution [Bac] and Metasploit framework [Met].

- Our tool, *Baffle* [BCKP08], transmits frames with varying values in the flags field, to fingerprint the chipset, firmware, and driver combination of a device suspected of being a rogue AP.
- *Beacon frame fuzzer* injects beacon frames with malformed field values in an attack against wireless drivers and devices.
- *Probe Response frame fuzzer* injects malformed fields in probe response frames.
- *FakeAP* advertises fake access points using random SSIDs and BSSIDs.

We captured wireless traffic, including the anomalous traffic, and identified the expected anomalies with the frame features of BSSID, source and destination MACs, frame length, sequence number, and the flags and type fields.

Results. We can clearly identify the anomalies in the plots in Figure 3.11. As expected, Baffle results in several new flag combinations being observed (near 00/Mon), and the fuzzing and fake AP anomalies result in many new source MAC addresses (near 06/Sun).

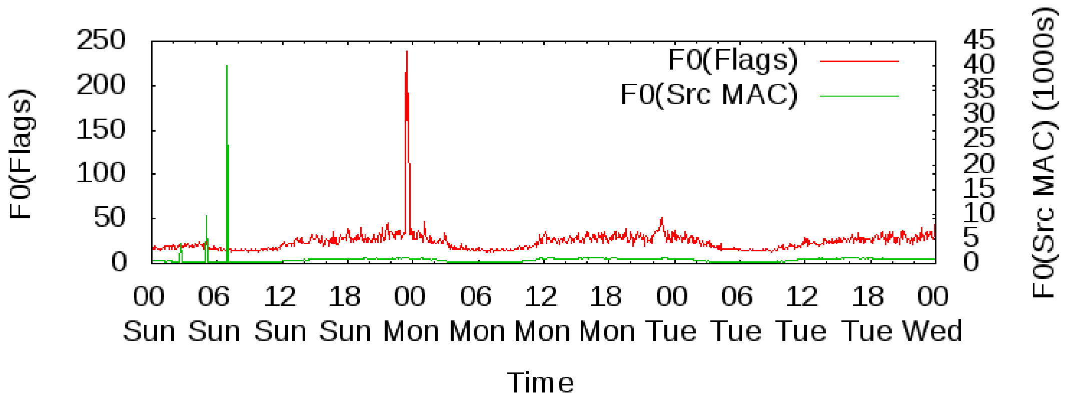


Figure 3.11: Timeseries of number of distinct flags, and distinct source MACs. [Best viewed in color.]

Chapter 4

Monitoring Information Networks

Wikipedia is a popular source of information content online. As an online encyclopedia, its content has been generated by its community of contributors that is open to anyone wishing to join. This novel methodology of open and collaborative information gathering and organization has led to some impressive results. As of August 2010, there were 3.4 million articles (we will also refer to them as pages) in the English language Wikipedia, and 16.7 million articles when considering all languages [Sta]. Figure 4.1 (the line for normal pages) shows the intense activity in page creation on Wikipedia since its inception. These articles span a wide and eclectic mix of topics ranging from the most popular (such as sports and celebrities) to the most obscure (e.g., List of Middle-earth inns, Permian Tetrapods). In our experience, more people have begun relying on Wikipedia as their first source of information, and are even trusting it to be accurate. As such, this corpus of information wields great power of influence, and many commercial and political and political entities are interested in leveraging this power to their advantage. Wikipedia pages commonly appear as the first result in many search queries.

Due to the open nature of Wikipedia, its contributors are additionally tasked with the job

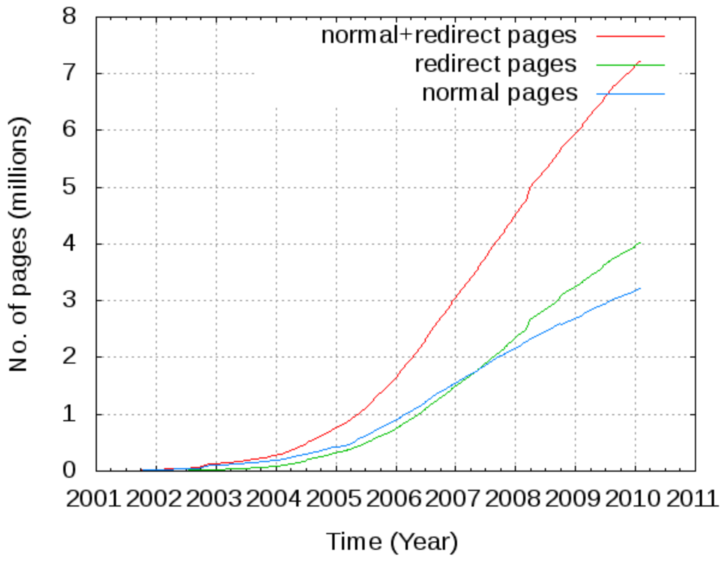


Figure 4.1: Growth of pages in the English language Wikipedia [Best viewed in color.]

of ensuring the relevance and accuracy of articles. Furthermore, the maintainers of the network are charged with ensuring the continued availability and integrity of the network, and providing interesting features to its readers using the knowledge corpus. While the contributors have been doing an admirable job of monitoring and correcting articles, an obvious open question is to design methods to automate some of those processes. In fact, there has been an entire workshop track recently [Wik10] devoted to developing machine-learning approaches for detecting vandalism attempts in Wikipedia. The focus of this workshop, however, was limited to classifying a given input text of an article revision as vandalism or not, after appropriate training of the classifier.

In this work, we present Wiki-watchdog, an efficient distribution-based methodology for anomaly detection in Wikipedia. It is intended to support the monitoring of the contributors and maintainers of the network. It is novel in that it is a holistic distribution-based methodology. That is, the goal of our methods is to monitor different aspects of distributions (or histograms) of the revision timeseries of Wikipedia for significant changes. Previous work on anomaly detection (e.g., [LCD05] [DCRM10] [BKPR02]) considered individual distribution based-metrics like volume, entropy, and KL-divergence to compare distributions (KL-divergence is not strictly a metric; we use term metric loosely). Each of these metrics captures a few aspects of the distribution(s) in question to varying degrees. Wiki-watchdog generalizes this approach and employs several efficient distribution-based metrics (and allows for more) that are collectively more powerful for anomaly detection. Each metric is efficient for monitoring massive volumes of timeseries data by applying appropriate data-stream algorithms available. Experiments on the timeseries we studied demonstrated that Wiki-watchdog has a higher detection rate than monitoring just entropy or volume of distributions.

The rate of false positives is generally considered to be a problem with anomaly detection methods. In our experiments, Wiki-watchdog exhibits a very low rate of false positives—over the course of five years, only two of the 64 anomalies flagged were false positives.

Unlike previous such work that consider (offline) anomaly detection in static network graphs (e.g., [AMF10]), our methods can be implemented efficiently online, and take into account properties of the network that evolve over time. By monitoring distributions of revisions/page and revisions/contributor as they change, we identify three kinds of anomalies: the activity of bots, flash events, and outages.

Wikipedia bots. Wikipedia provides a programming interface for the development of bots—programs automating certain tasks that are tedious to perform manually. For instance, a bot may pull demographic data from a government database and plug it into pages for towns and cities. Another example is the correction of certain frequent typographical errors. Detecting pronounced bot activity is important to the maintainers of the corpus as it can warn them in the case of rogue bots (one of the bots revealed by our analysis had been banned from Wikipedia [Lig] due to the nature of its activity) and of bots triggered by accident (many bot developers provide a means to disable the bot when this happens).

Flash events. Our methods revealed a number of flash events—events resulting in intense editing activity on a small set of pages, usually due to a particular newsworthy event. Some well-known flash events identified are Hurricane Katrina and the Virginia Tech Massacre, by the editing activity that ensued due to these events. Detection of such events is useful in early knowledge of news events as they develop, much like what is provided by the trending topics feature of Twitter. This kind of detection is also useful for identifying pages being abused, for example, due to edit wars [VWD04].

Outages. By our methodology, we were also able to detect instances of outages, either to all of Wikipedia, or to certain functionality within it. The capability of detecting different kinds of outages is important to the maintainers of Wikipedia because, in certain cases the outage is not immediately obvious. For instance, one of the outages discovered in our experiments only affected the functionality of bots.

Our dataset. To view the evolution of Wikipedia, and measure evolving properties for our experiments, we construct a dataset of timeseries of updates to Wikipedia (i.e., sorted by timestamp). Each update consists of the time of the update, the name of the updated page, the contributor responsible for the update, and the internal Wikipedia links that were modified. The nature and size of the dataset makes its construction a non-trivial process; we describe it briefly in Section 4.1.

Other applications. The methods we provide here can find applications in other centralized information and social networks as well, e.g., Facebook and Twitter. Our methods can be used for mining trends and events in streams generated in those networks. For example, analysis of the stream of addition and deletion of links to the network graph can provide insight into evolving trends of user centrality and connectivity. And applying our methods to streams of message postings can reveal properties of user relationships. The availability of data stream algorithms makes it feasible for the operators of these networks to implement our methods scalably in the face of the massive update streams produced.

The rest of the chapter is organized as follows. We describe our dataset in Section 4.1, and the distributions and metrics we use in Section 4.2. We present our anomaly detection methods in Section 4.3, and experimental results in Section 4.4.

4.1 Dataset Extraction

We now describe the dataset used in our experiments and the process we use to extract it. Our dataset is extracted from the English-language Wikipedia dump [Dum] dated January 30, 2010 made available by the Wikimedia Foundation. The dump is a 32GB compressed XML file that decompresses to 6TB. It contains the entire text of all 223 million revisions on the 3 million pages in Wikipedia, starting with the first revision on January 16, 2001. A revision (or update) is an action of a Wikipedia contributor to alter the contents of a page on Wikipedia. In our dataset we consider only pages in the Main "namespace" of Wikipedia. Other examples of namespaces include the User namespace containing pages about contributors, and the Talk namespace containing pages with user discussions about revisions to other pages. Revisions in the dump are grouped by page, but neither the page entries in the dump nor the revision entries within a page entry are sorted in any particular order. As our goal in this work is to study the properties of Wikipedia as they evolve in time, it is necessary to have a dataset supporting such analysis. Our dataset was extracted from the dump using a 28-node cluster to distribute computational load. It consists of a sequence of revisions sorted by timestamp. An example of a revision is:

```
1256478934 <t>Random_phase_approximation</t>
<c><username>Bodinio</username>
<add>[Keith_Brueckner]</add><del>[Brueckner]</del>
```

The Unix timestamp for this particular revision is 1256478934. The page on "Random phase approximation" was revised by the contributor "Bodinio". As a result, a link to the page "Brueckner" was deleted and a link to the page "Keith Brueckner" was added. In our dataset construction, we saved records of the changes to the links to enable future research.

4.2 Distributional Analysis

In this section we present the motivation for distributional analysis. We also present the distributions we consider for analysis, and the metrics we use to analyze those distributions.

Since the input to our anomaly detection methods is a stream of updates to Wikipedia, our methods must be able to monitor this stream efficiently to produce their results. Each update in the stream contains the name of the page updated, and the contributor responsible. Two natural distributions (histograms) that can be considered from this input stream are: (1) the distribution over the set of pages, of the number of revisions per page; and (2) the distribution over the set of contributors, of the number of revisions per contributor. Then, we can break down our timeline into a series of windows (say, each window being a day), consider the above distributions in each window, and track how they change over the windows.

4.2.1 Why Distributional Analysis?

In the context of Wikipedia (and social and information networks in general), distributional analysis has two advantages. First, the contributors of Wikipedia monitor only those pages they are interested in. Distributional analysis looks at Wikipedia as a whole, and provides a multidimensional view of revision activity from several different perspectives. This can reveal anomalies that cannot be observed by individual contributors. Second, it is not feasible for even the maintainers of the Wikipedia corpus to manually comb through the stream of updates looking for abnormal patterns. In this light, monitoring of distributional properties offers a good aggregate view of the stream of updates. As mentioned in previous chapters, entropy is one distributional metric that has previously been shown to be useful [LCD05] [NSA$^+$08] for monitoring network traffic for detecting DoS attacks, flash crowds, and port scanning. In this chapter, we show more generally that distributional analysis, using several distributional metrics to capture different aspects of the distribution, is a more effective method for anomaly detection. We employ the metrics of the volume F_1, the entropy H, the number of distinct elements F_0, and the second frequency moment F_2, for anomaly detection in Wikipedia. These metrics and their properties are described in detail in Chapter 2. Next, we compare the values of these metrics under different distributions.

4.2.2 Behavior of metrics

We now show the behavior of our metrics on distributions with different shapes. Each of the distributions in Figure 4.2 have the same value of F_1 (the stream length), to ensure that we only examine the effect of distribution shape, and avoid variation in volume. In our detection methodology we normalize the metrics (except F_1 itself) using F_1, to discount the effect of daily variation in F_1 on the metrics.

In Figure 4.2, Plot (a) shows the normal distribution for this analysis, against which we compare the other plots. Plots (b) and (c) show distributions that are more and less skewed respectively. Plots (d) and (e) show distributions that have a larger and a smaller support respectively. The remaining plots show distributions combining features from Plots (b) through (e). Each plot (except Plot (a)) also shows the values of the metrics and their relative changes (expressed as percentages) when compared to the normal distribution in Plot (a).

We can observe from the metric deviations in Plots (b) through (e) that entropy is able to capture the change in the skew and the support size of the distributions. We also see in those plots that F_2 is able to detect change in the distribution skew, and F_0 is able to detect change in the distribution support size. Now, looking at Plots (f) and (g) we see that although the distribution shapes are significantly different from that in Plot (a), entropy does not register the changes. On the other hand, both F_0 and F_2 are able to detect the changes. On comparing Plot (b) and Plot (e) to Plot (a) we find that although the entropy in the two distributions change by about the same amount, that deviation does not indicate *how* the distributions have changed. On the other hand, monitoring F_0 and F_2 tells us the changes in the two distributions are due to the increased skew and decreased support size of the two distributions respectively. To an extent, the same is the case when comparing Plot (c) and Plot (d) to Plot (a).

Again, the entropy in Plot (h) increases significantly due to the combined reinforcing effects of decreased skew and increased support size. But it is only observation of the values of F_0 and F_2 that reveal the reason behind the change in the distribution.

The implication of the above analysis is that it is not sufficient merely to monitor the entropy, and that monitoring other distributional metrics like F_0 and F_2 yields more knowledge of the shape of the distribution and any changes in it. This observation is useful for increasing the anomaly detection rate, as well as in diagnosis of anomalies when they are found.

4.3 The Wiki-watchdog Detection Algorithm

In this section, we describe our anomaly detection methodology. We divide the input update streams into consecutive windows W_1, W_2, . . ., each of some fixed time interval parameter T. In our experiments, we found that setting T to be 24 hours was most beneficial, since most of our anomalies tend to last from several hours to a few days. Within each interval W_i we compute our metrics on the sub-stream S_i corresponding to that interval. Thus, the metric values v_i over the intervals form a timeseries. As mentioned previously, we also normalize the value v_i of each metric to get $\hat{v}_i$ using the sub-stream length $m_i = F_1(S_i)$, to discount the effects of changing length on metric values. For the metrics H, F_0, F_2, we use the normalization factors $\log(m_i)$, m_i, and m_i^2 respectively for normalization. These terms are the maximum possible values of the respective metrics for given length m_i. After normalization, each metric has a value $\hat{v}_i$ in [0, 1], which we can then compare to its values from other time windows. We then monitor each such timeseries $\hat{v}_1, \hat{v}_2, \ldots$ for significant deviations, using an Exponential Weighted Moving Average (EWMA) based scheme that can be applied in an online fashion. The EWMA works as a filter to smooth out local fluctuations in the timeseries. This method helps in avoiding false positives/negatives due to noise in the timeseries. We compute the EWMA E of the concerned timeseries online, and compare each new value $\hat{v}_i$ of the timeseries against the EWMA value at the time. If the relative deviation $|\hat{v}_i - E|/E$ of the new value from the EWMA exceeds a given threshold parameter τ then we flag the new value as an anomaly.

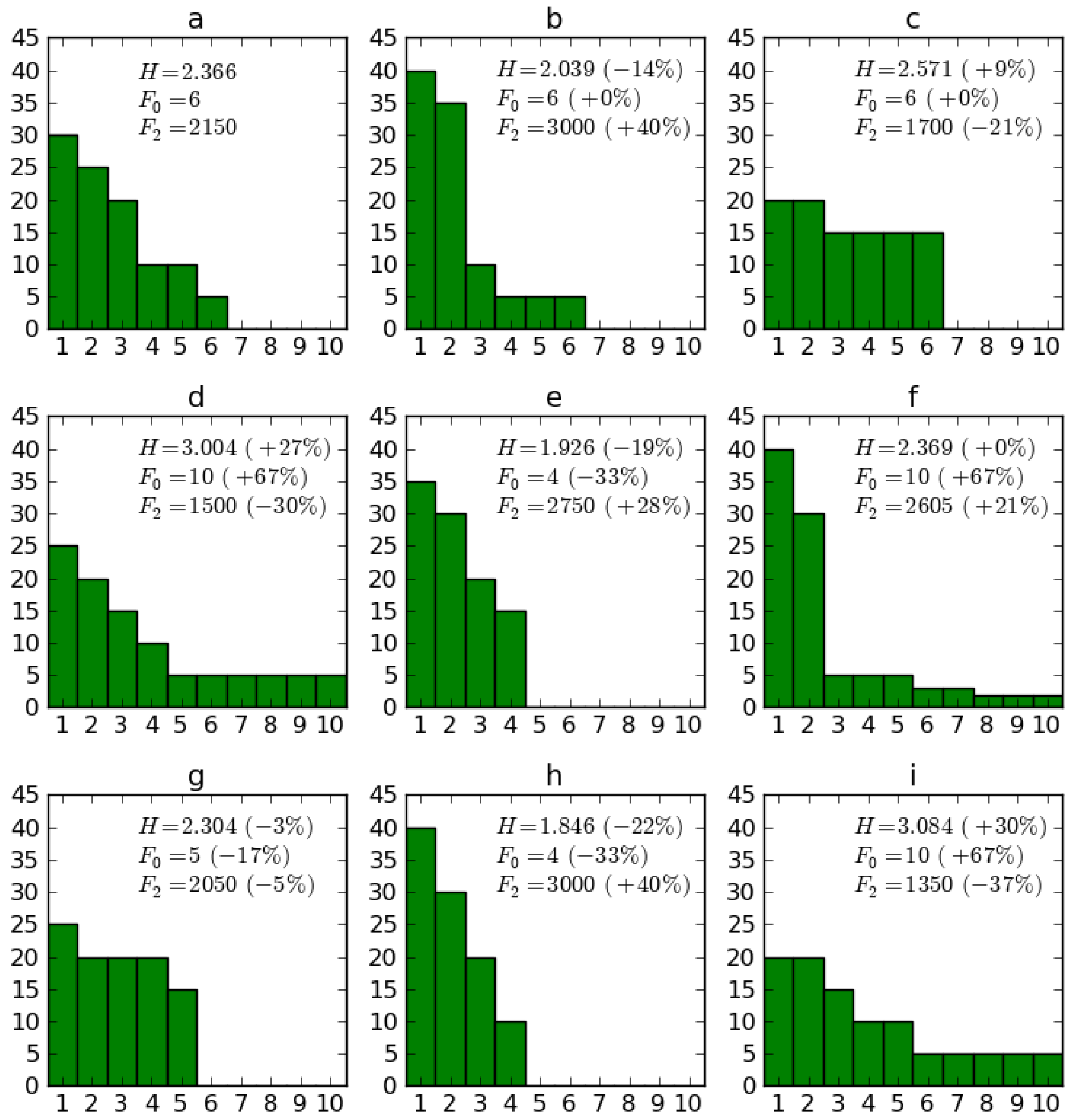


Figure 4.2: Behavior of metrics on various distributions. Y-axis is the histogram frequency.

Detection Algorithm. Input: stream S_i in window W_i

1. Obtain the metric value v_i using a data stream algorithm, and normalize to $\hat{v}_i$.
2. If $|\hat{v}_i - E|/E > \tau$, flag window W_i as an anomaly.
3. Update $E := E + \alpha(\hat{v}_i - E)$.

Parameter α is the EWMA multiplier. It determines how much weight is given to the historical values of the timeseries versus the newer values when computing the EWMA. The EWMA value E is initialized as the average of the metric values from the first few windows. In our experiments we used the values from the first week for initialization. When updating the EWMA, we exclude metric values from anomalous windows, and values from three following windows (unless the anomaly was determined to be a false positive after diagnosis). We do this to prevent the anomaly from destabilizing the EWMA. In a few cases we found that the anomaly spilled somewhat over the boundary of the day, and sometimes continued with a lower intensity over the following days. The metric values in the following windows, however, did not deviate sufficiently enough to be flagged an anomaly. So, to avoid such effects from compromising the stability of the EWMA we ignore those values.

Parameter choices. We now describe how the parameters α and τ were chosen in our experiments. Our first step was to look at the timeseries and determine visually which points appeared to be anomalies (prominent dips or spikes in the plot). This approach gave us a set of anomalies to work with, and we picked the two parameters to allow for this set of anomalies to be detected. A higher value of α gives more weight to the newly observed value in the EWMA. Given the noisy data we found that setting $\alpha = 0.3$ was a good compromise between detection capability and the smoothing effect of the EWMA. Figure 4.3 shows the effect of changing the two parameters on the number of anomalies reported by the detection method. For setting the threshold τ we used the the “elbow” in the anomaly detection curve as a reference (e.g., for entropy of the page distribution see Figure 4.3; we set the threshold to 0.014). The elbow is the point when reducing τ any further results in an exponential increase in the number of anomalies. It is the turning point between the threshold being set too high and too low. We chose a value of τ near the elbow, while also ensuring that our visually identified anomalies were captured.

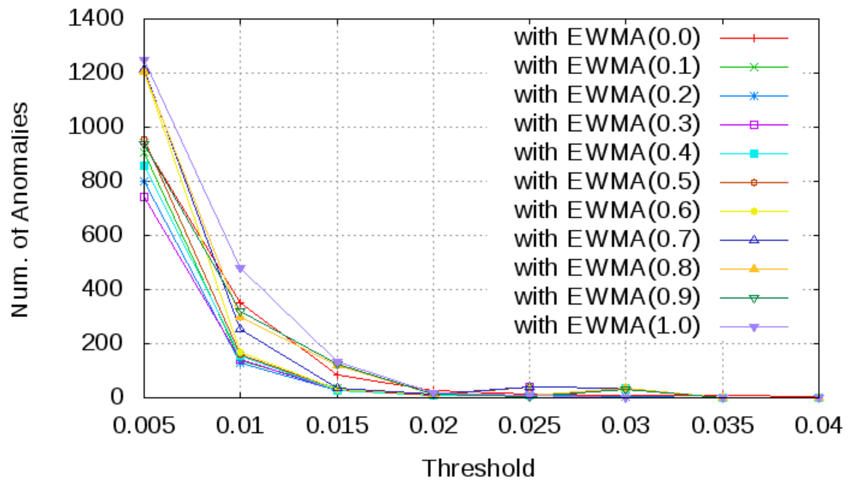


Figure 4.3: EWMA(α) anomaly curves for H(page dist.) [Best viewed in color.]

4.4 Experimental Analysis

In this section we present the results of our experimental analysis on the Wikipedia dataset described previously. Any anomaly detection scheme requires a stable system to start with, to ensure there is a baseline norm present to compare anomalies against. We found that in the early years of Wikipedia, contributor activity kept increasing before leveling off around 2005. Figure 4.4 shows the number of edits per week, and Figure 4.5 compares the outcome of the entropy metric of the page distribution, before and after 2005. From these figures we can see the instability of the system prior to 2005. Thus, we apply our detection method only on the timeseries starting 2005. Figures 4.6 and 4.7 show the detection method applied on the page distribution and the contributor distribution respectively. Within each figure there are plots for the monitoring of H, F_0, and F_2 over the corresponding distribution. Also, Figure 4.8 shows the monitoring of the F_1 metric. Note that the value of F_1 in a window is the same whether measured from the page distribution or the user distribution. Each plot shows the metric timeseries (red), EWMA timeseries (blue), and anomalies (green circles) according to our detection methodology. Tables 4.1 and 4.2 list the anomalies found and marked in the plots. A "+" in the table indicates that the anomaly was due to a spike in the timeseries curve (when the metric value deviates higher than the EWMA value), and a "-" in the table indicates a dip in the curve. The last column of the table lists results of our diagnosis of the anomalies found. To determine the causes of an anomaly we used two steps. First, we looked at what effects the anomaly had on the metrics on the distributions. The combination of these effects was usually able to provide a good indication of the properties we would find in the distributions on the day of the anomaly. Second, we extracted the histograms on the day of the anomaly and looked for the properties identified in the first step. Depending on these properties we were then able to find the contributor, page, or other reason for the anomaly. We can categorize the listed anomalies into three groups—bots, flash events, and outages. The table comments also indicate the category for each anomaly.

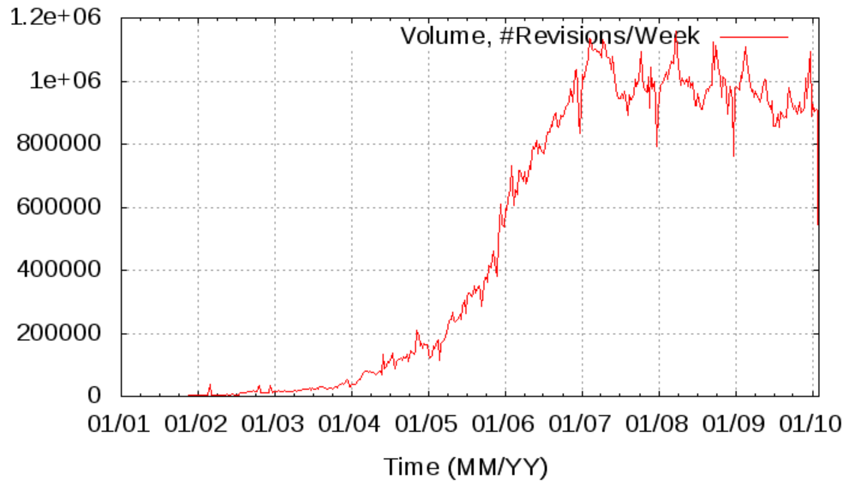


Figure 4.4: Volume of weekly revisions

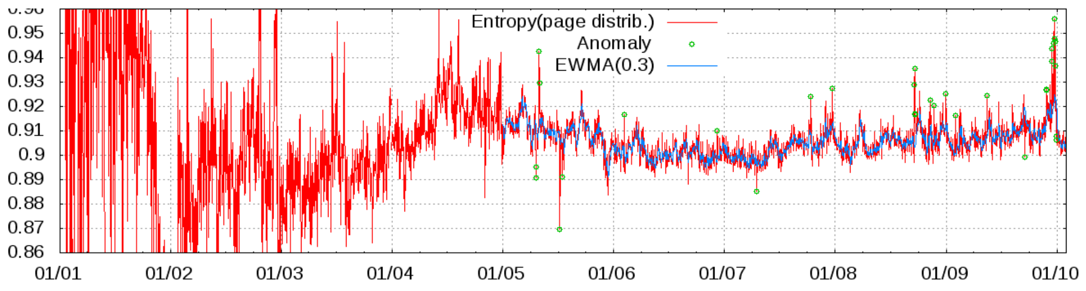


Figure 4.5: Timeseries of H(page dist.), when including data prior to 2005. Detection method only applied from 2005. [Best viewed in color.]

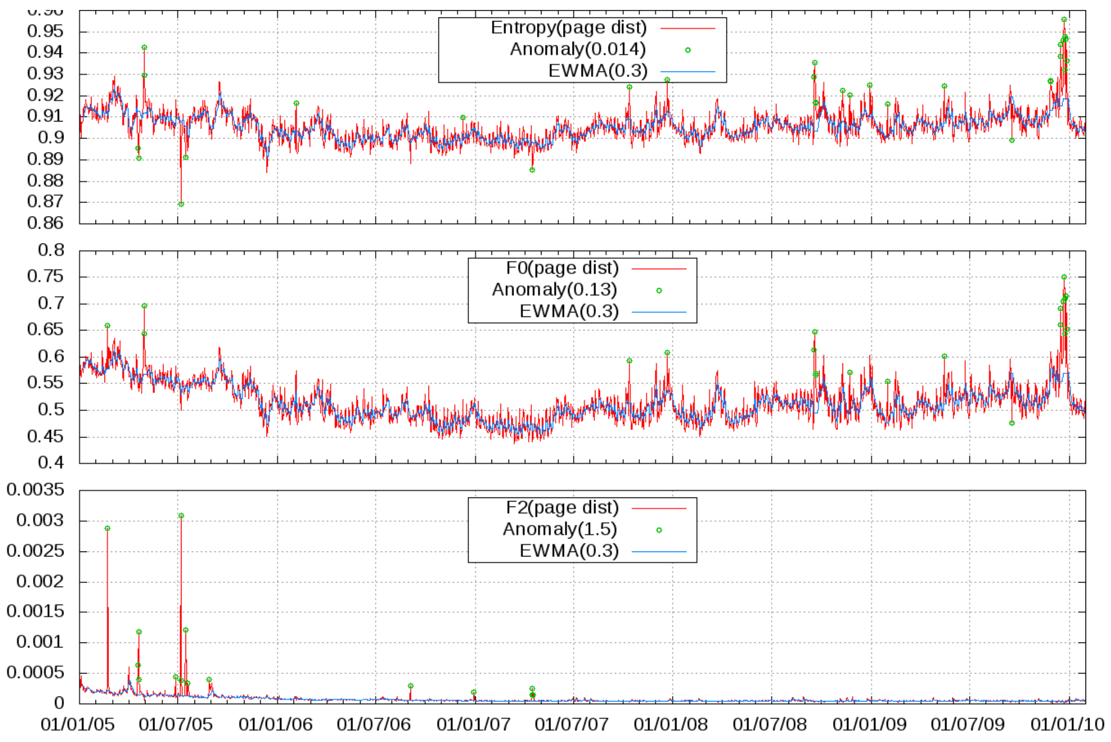


Figure 4.6: Anomaly detection with the page distribution timeseries. Y-axis is metric value. [Best viewed in color.]

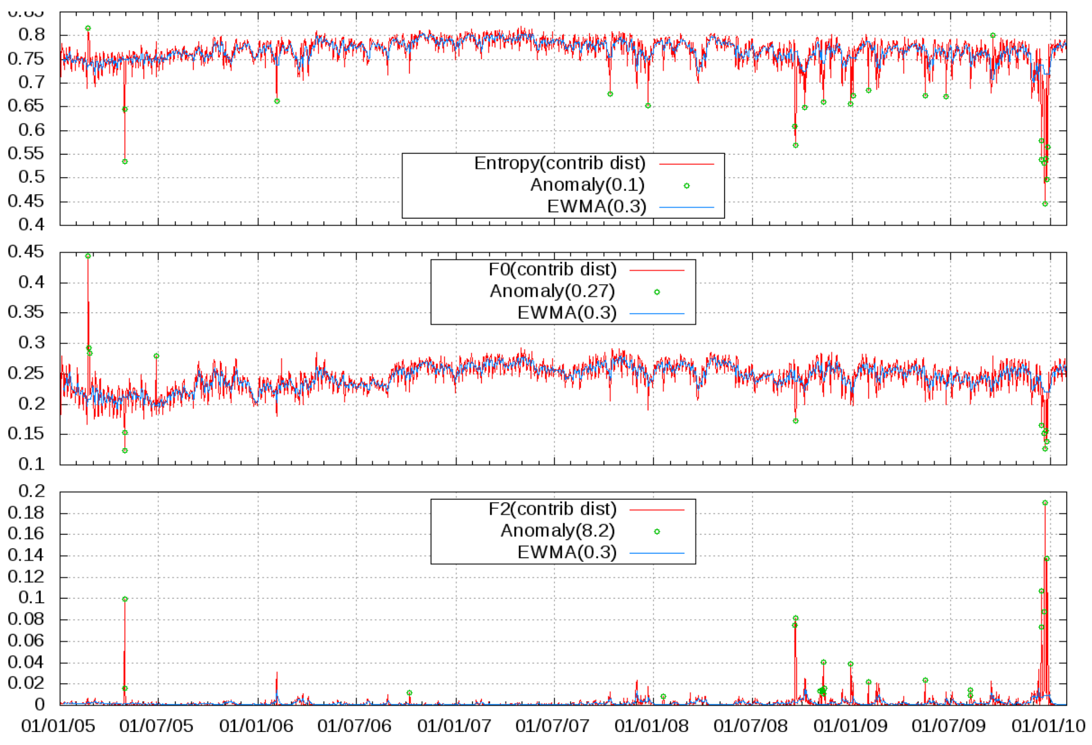


Figure 4.7: Anomaly detection with the contributor distribution timeseries. Y-axis is metric value. [Best viewed in color.]

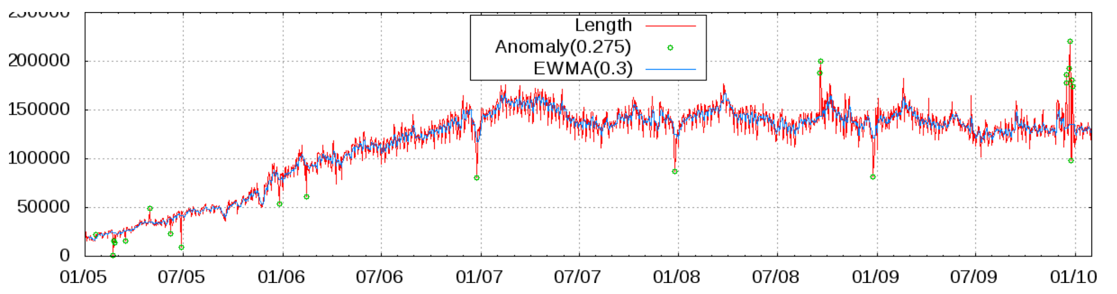


Figure 4.8: Anomaly detection with the F_1 (volume) metric timeseries. Y-axis is metric value. [Best viewed in color.]

	Page			Contrib.					
	H	F_0	F_2	H	F_0	F_2	F_1		Comments
Jan-21-05							+	*	False positive
Feb-22-05		+	+	+	+		-	[O]	Power failure; no editing
Feb-23-05					+		-		” ” ”
Feb-24-05					+		-		” ” ”
Mar-16-05							-	[O]	Database outage; editing disabled
Apr-19-05	-		+					[F]	Pope Benedict XVI elected
Apr-20-05	-		+						” ” ”
Apr-21-05			+						” ” ”
Apr-30-05	+	+		-	-	+	+	[B]	Flabot (updating interlanguage links)
May-01-05	+	+		-	-	+			” ” ”
Jun-07-05							-	[O]	Planned outage
Jun-27-05			+		+		-	[O]	MediaWiki update; editing blocked
Jul-07-05	-		+					[F]	London bombings
Jul-08-05			+						” ” ”
Jul-16-05	-		+					[F]	Harry Potter and the Half-blood Prince released
Jul-20-05			+					[F]	(US politics) John Roberts nominated to Supreme Court
Aug-29-05			+					[F]	Hurricane Katrina
Dec-25-05							-	[H]	Christmas
Feb-05-06	+			-				[B]	D6 (adding to Category:Living People)
Feb-13-06							-	[O]	Minor downtime
Sep-04-06			+					[F]	(Accident) Death of Steve Irwin
Oct-08-06						+		[B]	SmackBot
Dec-09-06	+							*	False positive
Dec-25-06							-	[H]	Christmas
Dec-30-06			+					[F]	Execution of Saddam Hussein
Apr-16-07			+					[F]	Virginia Tech Massacre
Apr-17-07	-		+						” ” ”
Apr-18-07			+						” ” ”
Oct-13-07	+	+		-				[B]	CapitolBot (Infobox, towns/cities)
Dec-22-07	+	+		-				[B]	DetroiterBot (infobox params, formatting, townships/counties)
Dec-25-07							-	[H]	Christmas
Jan-19-08						+		[B]	SmackBot

Table 4.1: Anomalies: [B]=Bot, [F]=Flash Event, [O]=Outage, [H]=Holiday

	Page			Contrib.					
	H	F_0	F_2	H	F_0	F_2	F_1		Comments
Sep-18-08	+	+		-		+	+	[B]	LightBot + Anomebot (demographics) + DinoBot2 (movie rating templates)
Sep-19-08	+	+		-	-	+	+		” ” ”
Sep-20-08	+	+							” ” ”
Sep-21-08	+	+							” ” ”
Oct-05-08				-				[B]	LightBot
Nov-03-08						+		[B]	LegoBot
Nov-07-08						+		[B]	LightBot (date audits)
Nov-08-08						+			” ” ”
Nov-09-08	+			-		+			” ” ”
Nov-10-08						+			” ” ”
Nov-12-08						+			” ” ”
Nov-22-08	+	+						[B]	YoBot (Category adding)
Dec-25-08							-	[H]	Christmas
Dec-30-08	+			-		+		[B]	LightBot (units/dates/other)
Jan-03-09				-				[B]	LightBot
Jan-31-09	+	+		-		+		[B]	Cydebot (moving categories)
May-16-09	+	+		-		+		[B]	D6 (formatting headline levels, fixing Unicode in templates)
Jun-24-09				-				[B]	BOTijo
Aug-07-09						+		[B]	Erik9bot
Aug-08-09						+			” ” ”
Sep-17-09	-	-		+				[O]	Bugs with MediaWiki update; bots stopped working
Nov-28-09	+							[B]	AnomieBot (editing IPA phonetic symbols) + Full-date unlinking bot
Nov-29-09	+								” ” ”
Dec-16-09	+	+		-		+	+	[B]	SmackBot (date maintenance etc.)
Dec-17-09	+	+		-	-	+	+		” ” ”
Dec-22-09	+	+		-	-	+	+	[B]	SmackBot (delink dates)
Dec-23-09	+	+		-	-	+	+		” ” ”
Dec-24-09	+	+		-	-				” ” ”
Dec-25-09	+	+					-	[H]	Christmas
Dec-27-09	+	+		-	-	+	+	[B]	SmackBot (delink dates)
Dec-28-09	+	+		-			+		” ” ”

Table 4.2: Anomalies: [B]=Bot, [F]=Flash Event, [O]=Outage, [H]=Holiday

Wikipedia Bots. Bots in Wikipedia perform their activity frequently, searching for appropriate pages to be updated. But sometimes, the activity of one or more bots is more pronounced, meaning that they update a large volume of pages in a relatively short span of time. Such activity reflects in a spike in both H and F_0 of the page distribution, as we would expect. On the other hand, for the contributor distribution we expect the bot activity to result in a dip in H, and a spike in F_2. For some of the bot anomalies, the effect is also observed as a spike in the volume metric. Tables 4.1 and 4.2 shows 33 anomalies due to 13 bots—Flabot, D6, CapitolBot, etc.

Flash Events. For flash events, the threshold τ serves as a knob for determining the level at which an event is declared as newsworthy. Flash events usually result in a dip in H or a spike in F_2 of the page distribution. We did not observe a significant effect on the contributor distribution.

Outages. The outage between February 22–24, 2005 was due to a power failure during which editing on the whole of Wikipedia was cut off. On June 27, 2005 access to Wikipedia was again blocked due to an upgrade to the MediaWiki software that Wikipedia runs on. These anomalies resulted in unpredictable effects on the page and contributor distributions, since there were only a few elements in the histograms (the elements before and after the anomaly from the included windows). This unstable behavior is an indicator of the outage. The dips in the volume metric are the most direct indicator of the anomalies. We also found another unique outage event through our analysis. On September 17, 2009 a bug in an update to the MediaWiki software caused the programming interface for bots to fail. Thus bots that were editing Wikipedia at the time ceased operation. This anomaly was tricky to diagnose. Because it is not a full-scale outage, it does not affect the volume metric.

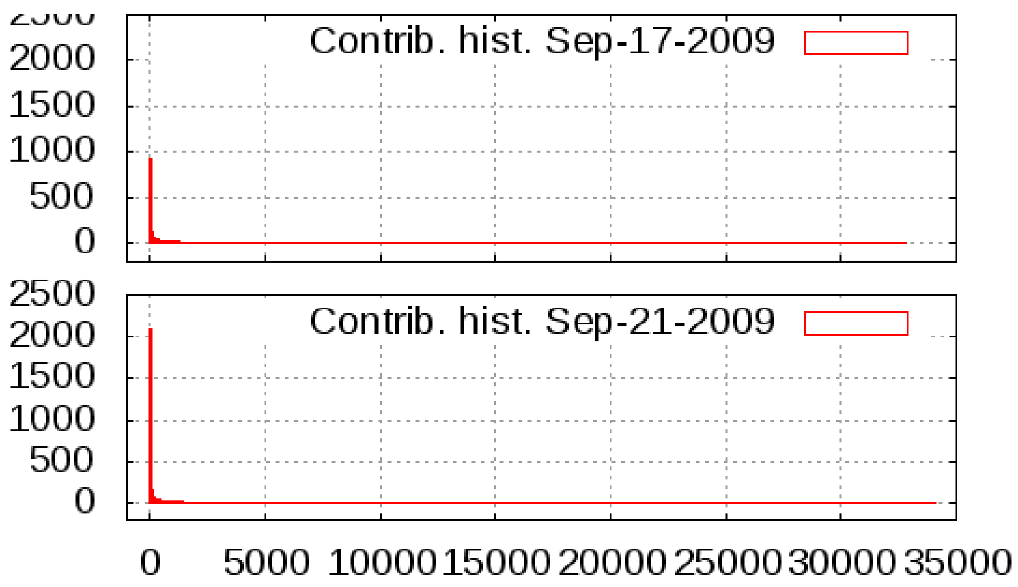


Figure 4.9: Sep-17-09 anomaly histogram compared with that a few days later.

The entropy of the contributor distribution spikes due to the anomaly. On examining the histogram (see Figure 4.9 for a comparison of the histogram on Sep-17 with the "normal" histogram a few days later), we found that although the support of the distribution decreased slightly (bots disappeared), the distribution became more uniform because of the absence of the bots (who were also the heavy hitters). This combines the effects from Plots (c) and (e) (much like Plot (g)) in Figure 4.2. For the page distribution, the entropy dips, but it does not seem to indicate why this occurs unless the other metrics and distribution are also considered. F_0 of the page distribution dips significantly. Initially, this was hard to justify; it was not clear why the number of pages edited was smaller. It is only the tail of the distribution with the single-edit pages (due to the bots) that disappeared, which is telling of the anomaly.

We were able to confirm the causes of the outages above by searching the archives of the Wikipedia Signpost [Sig] that provides news updates on Wikipedia-related events.

Other Anomalies. Besides the anomalies noted above, we were also able to identify the periodic annual lull in revision activity on Christmas day each year. This is clearly reflected in the volume metric (Figure 4.8).

False Positives. Our distribution-based methodology exhibits a low rate of false positives. Only two of the 64 anomalies flagged (i.e., 3%) over the course of five years were found to be false positives. The January 25, 2005 false positive can be attributed due to the instability and increasing volume of revisions at the start of our dataset. The second false positive on December 9, 2006 is the result of an unfortunate combination of a small increase in entropy with a small decrease in revision volume giving the impression of a large increase in entropy, due to the normalization.

An important observation to be drawn from Tables 4.1 and 4.2 is that each of the metrics is significant in our methodology. For each metric, the anomalies flagged are never consistently flagged by another metric.

Chapter 5

Detecting Covert Channels in VoIP Traffic

Exfiltration of data via covert channels is an important threat faced by organizations like the military that deal with classified or sensitive information. Voice over IP (VoIP) is increasingly being adopted for communication in the networks of these organizations due to the benefits of the technology. These organizations have, however, expressed concern over the data confidentiality risks posed by covert channels piggybacking on VoIP sessions. These concerns are well-founded, as we discuss in Section 5.1. Motivated by these concerns, we show the effectiveness of distribution-based anomaly detection methods for the detection of covert channels over VoIP. Intuitively, information exfiltrated over a covert channel *is* an anomaly for distributions associated with the abused carrier channel. While designed for VoIP, we believe our methodology is applicable to a much broader class of covert channels.

In our presentations and discussions with researchers on distribution-based anomaly detection methods, we have frequently had questions asked about the various parameters (we will refer to them as *knobs*) involved. We found that how those knobs should be tuned to achieve a good quality of detection is an open problem in general.

One of these knobs is the choice of the length of the window in which the distribution is measured. Distribution-based anomaly detection methods typically slice the observed stream into windows of equal length, either in terms of time or in terms of number of elements. The metric values of the distributions in successive windows is then monitored as a timeseries. Another knob is that of the detection threshold beyond which the metric value is flagged as an anomaly.

Many intrusion detection and prevention systems (IDS/IPS) offer such tunable knobs for user configuration. It is common knowledge among practitioners that large amounts of effort are required to tune these knobs for optimal performance. Optimal values tend to vary widely between systems and environments, for example, an educational institution would require different knob settings from a small enterprise, and both of these settings would be different from the settings required in a large corporation with several subsidiaries. Oftentimes, this is due to the existence of different baseline normals for distributions in these different environments. Despite the tuning of knobs being a complex and continuous task, a rigorous study of the effects involved has not as yet been undertaken. Such a study is also useful for estimating the effectiveness of effort invested in tuning.

In this work we construct a probabilistic model to observe the effects of the knobs, and then apply the model analytically and empirically by experiments on our VoIP testbed using the covert channels we developed. The primary contribution of this work is to show how the tuning of the knobs affect the quality of anomaly detection. To do the same, we analyze the results of a distribution-based method on a VoIP covert channel while varying the settings of the knobs. We design a simple VoIP covert channel to test the methodology. Our results serve to provide principles for tuning the knobs to practitioners deploying the methodology in real-world applications, and for systematically testing the limits of useful tunability. The latter is an important issue for many practitioners: whereas it is commonly assumed that tuning the parameters of an IDS/IPS can improve its performance on a given network, the limits of such improvements are not clear, and reaching them is something of an arcane art.

Covert channels give us a good handle to vary the magnitude of the anomaly by simply changing the throughput of the covert channel. This ability is perfect for studying how the magnitude of the anomaly affects detection rate. The Department of Defense Orange Book [DoD] of security guidelines set the direction here by prescribing a threshold bandwidth for covert channels that should not be allowed on a secure system. It is widely accepted that it is not possible to eliminate all forms of covert channels, particularly those of a low throughput. Instead, the practitioner should aim at eliminating covert channels of throughputs that are sufficiently high so as to pose a real threat. In this work we study how the detection rate (and the false-positive rate) vary as a function of the covert channel throughput, allowing the implementer to be aware of the capabilities and limitations of distribution-based methodology.

The contributions of this work are summarized thus:

1. A rigorous analysis of the effects of tuning the knobs of distribution-based anomaly detection on detection quality. Construction of a probabilistic model to observe such effects, and analytical and empirical applications of the model.

2. A recipe for a practictioner to follow, to tune the knobs optimally in his specific environment.

3. Our analysis yields an understanding of the subtle effects that come into play when knobs are tuned, and which have a resulting impact on detection quality.

4. Our work considers several different distributions and anomaly sizes (by varying the covert channel rate). It is thus generally applicable to all distribution-based anomaly detection methods, such as the entropy-based methods for anomaly detection in network traffic that have been considered in prior work.

5. Analysis of the tradeoff between the magnitude of the anomaly and the quality of detection, demonstrating anomalies of what sizes are feasible to detect robustly.

6. Design of a VoIP covert channel, and demonstration of the use of distribution-based anomaly detection to detect such a channel.

The rest of the chapter is organized as follows. We present related work in the next section. We discuss VoIP and describe our sample VoIP covert channel in Section 5.1. Section 5.2 presents our probabilistic model, and analytical application of the same. In Section 5.3 we present our empirical evaluations of the model.

5.1 VoIP Covert Channels

In this section we discuss our choice of the sample covert channel design. This choice is motivated by our perception of the relative prominence of both the protocol abuse tricks and the related network technologies to be used. A better motivation would be grounded in statistics of actual illicit use of covert channels in enterprise networks; unfortunately, such statistics are not widely available. Hence, we must base our choice on general architectural and practical considerations of network monitoring and administration.

5.1.1 Why VoIP?

VoIP is an ideal target for establishing resilient covert channels across an organization's network boundary by a malicious insider. VoIP functionality is critical to modern enterprise. VoIP-based phone and voice mail systems offer numerous business advantages (maintenance costs not least) over traditional "Plain Old Telephone System" private phone exchanges in organizations, and are replacing PBXs across the board. Thus, it is reasonable to expect VoIP functionality to be deeply integrated into the organization's network fabric—with multiple security exceptions required to support it.

Accordingly, it is a safe bet that the organizations' network security policies and controls such as firewalls and intrusion prevention systems (IPS) must accommodate VoIP protocols, possibly in multiple implementation flavors. However, any single VoIP flavor itself involves multiple interrelated protocols with complex endpoint state that is hard to model on in-line network appliances that are necessarily limited in the amount of context they can effectively mediate.

In particular, VoIP protocols are notoriously complex and demanding on firewall rules—for example, VoIP signaling and media use different protocols such as SIP and RTP, which require dedicated open port ranges to operate, and are hard to narrow down. Moreover, Network Address Translation (NAT), arguably the most effective way to isolate internal IP LANs, is well-known to cause problems for SIP-based media session setup, and therefore requires workarounds which in turn raise network configuration complexity and undermine firewall isolation of internal systems. Furthermore, VoIP media protocols such as RTP place considerable performance requirements on network elements and paths they cross. As a result, various forms of deep packet inspection and other in-line IPS functionality is impeded by *both* the latency and bandwidth requirements of these protocols to deliver good voice audio quality. The combination of these circumstances makes abusing VoIP protocols an effective way for a malicious insider to punch unnoticed through network boundary defences such as strict firewall rules and traffic auditing measures.

5.1.2 VoIP signaling and media protocol essentials

The basic design of VoIP involves separate protocols for signaling (e.g., SIP or Cisco's SCCP) to establish a voice medium session and for the actual transfer of encoded voice payload (e.g., using RTP). In particular, the signaling session facilitates the direct or proxied connection pathway between the VoIP endpoint devices such as handsets.

This separation is necessitated both by architectural concerns such as integration of the session creation with the organization's employee address directories and by performance considerations for established sessions (ideally, established directly between the communicating VoIP endpoints).

5.1.3 Basic VoIP convert channel design considerations

Having the firewalls' cooperation in passing the payloads outside the organization is only one step towards successful data exfiltration. The exfiltrated data must also be reasonably well-hidden from network auditors, *and* exfiltrated at a reasonable rate to reduce the time of the illicit sessions' duration.

The latter two goals of bandwidth vs. stealth are at cross-purposes. Steganographic channels (e.g., SteganRTP [Ste]) must reduce bandwidth to satisfy classic concepts of steganographic stealth, which assume that any given session is likely to be scrutinized for illicit content; this is far from the everyday reality of network monitoring, in which even obviously anomalous sessions slip through daily, due to large volumes of traffic and the costs of its processing.

Accordingly, in practice, the design of a covert channel need not be steganographic at the bandwidth's detriment. Indeed, most publicly available covert channel implementations do not pursue steganographic sophistication [SAN, JJT].

However, when designing a VoIP covert channel, one must account for the possibility of a human auditor getting to listen in on an arbitrary segment of the call, and modern network monitoring appliances provide such a capability to operators. Failing such a test or otherwise attracting operator attention with loggable errors would be imprudent for an insider attempting to exfiltrate information.

Accordingly, we formulate the following observations regarding VoIP covert channel design:

1. A covert channel must be resilient to network cross-site call paths common in production. This reduces the attractiveness of timing-based channels [GBC06] in favor of channels based on manipulation of protocol field values.

2. A covert channel should not create network error conditions other than those already commonly present.

3. A covert channel crossing a VoIP system should be inaudible to auditors, and to users making calls.

Together, these considerations motivate our choice of a simple covert channel.

5.1.4 Sample VoIP covert channel

Our sample covert channel *CCseq* uses the RTP protocol used by VoIP for the voice medium. RTP is either sent directly between the endpoints, or, in the presence of Network Address Translation (NAT), is proxied in a relatively lightweight manner to serve its real time delivery requirements. CCseq's design, while simple, is representative of popular non-steganographic covert channels.

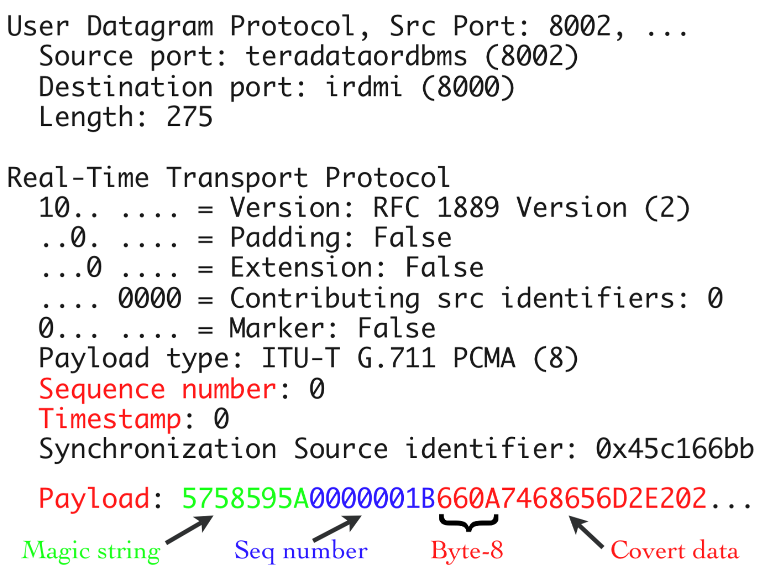
The CCseq "protocol" abuses the real-time nature of RTP voice streams to inject data chunks comparable in size to actual packet payloads, in place of the actual payloads. To keep these payloads from being interpreted as voice data (and creating loud audible noises and/or packet parsing errors), the RTP *sequence number* and/or *timestamp* in such packets is changed to appear outside of the current stream's window (see Figure 5.1). A number of softphones and handsets we tested silently discard such packets, assuming them to be delayed by the network (and delivered late, out of order).

The channel, despite its design naivete, is transparent to human listeners and commands much higher bandwidth than, say, SteganRTP, which is limited to just using the lower bit of each voice payload byte (under the simplest G.711 codec).

For the protocol's intended recipient on the remote end of the VoIP path (from the organization's border outward), the first 4 bytes of RTP payload contain a fixed magic string for easy identification, the next 4 bytes contain a sequence number to detect loss of injected packets, and the rest of the bytes are covert data (example in Figure 5.1).

5.2 Probabilistic Modeling of Detection Quality

In this section we present our probabilistic model to predict the detection rate and the false-positive rate in terms of the window length, covert channel rate, and the detection threshold. We then show analytical applications of the model.



```
User Datagram Protocol, Src Port: 8002, ...
  Source port: teradataordbms (8002)
  Destination port: irdmi (8000)
  Length: 275

Real-Time Transport Protocol
  10.. .... = Version: RFC 1889 Version (2)
  ..0. .... = Padding: False
  ...0 .... = Extension: False
  .... 0000 = Contributing src identifiers: 0
  0... .... = Marker: False
  Payload type: ITU-T G.711 PCMA (8)
  Sequence number: 0
  Timestamp: 0
  Synchronization Source identifier: 0x45c166bb

  Payload: 5758595A0000001B660A7468656D2E202...
```

Figure 5.1: Tshark output fragment illustrating a CCSeq tampered packet with RTP timestamp and sequence number set to 0. [Best viewed in color.]

Notation We call a packet (and the corresponding stream element) that is part of the covert channel as a *tainted packet* (*tainted element*), and a packet (element) that belongs to the normal VoIP stream as a *regular packet* (*regular element*). Let ℓ and t denote the number of tainted packets, and the interval (in terms of number of regular packets) between successive tainted packets, respectively. Further, let w denote the length of the window over which the detection method is applied, and τ denote the threshold used in the method to determine when the metric value has deviated far enough for the window to be flagged as an anomaly. We assume that the normal VoIP stream elements are produced by sampling from some underlying distribution $\mathcal{D}$ over the universe $[n]$ of possible values. For instance, when the RTP sequence numbers are considered as the stream for monitoring $n = 2^{32}$. We will denote a window W of elements sampled from the distribution $\mathcal{D}$ as $W \sim \mathcal{D}$. Let M denote the metric under consideration in the detection method, so that $M(W)$ denotes its value for the distribution induced by elements in window W. We will denote a window

W^C containing tainted elements among regular elements that are sampled from underlying distribution $\mathcal{D}^C$ as $W^C \sim \mathcal{D}^C$.

5.2.1 General Model

We first present our model in full generality, leaving $\mathcal{D}$ and M unspecified. Later on we demonstrate how the model can be applied analytically in this section, and experimentally in Section 5.3 with specific values of $\mathcal{D}$ and M.

Our goal is to model the effect of the knobs (the window length w, and the threshold τ) and the covert channel throughput (controlled by the variable t) on the rate of false positives and the rate of false negatives (which in turn determines the detection rate). First, we need to establish a baseline normal value for the metric against which the deviation is measured at the time of monitoring the stream. Our detection algorithm raises an alarm if it finds the deviation in any window is significant. The baseline is simply given by the expected value $\mathrm{E}[M] = \sum_W \Pr[W] \cdot M(W)$ of the metric in the absence of any covert channel, where the summation is over all possible windows W of length w. Our detection algorithm detects and anomaly if, in any window W_i, $1 \leq i \leq r$, we have $|M(W_i) - \mathrm{E}[M]| > \tau \cdot \mathrm{E}[M]$. We have the false-positive rate $x_w(\tau)$ and the false-negative rate $y_w(\tau)$ given by

$$x_w(\tau) = \Pr_{W \sim \mathcal{D}} \left[|M(W) - \mathrm{E}[M]| > \tau \cdot \mathrm{E}[M] \right], \text{ and} \tag{5.1}$$

$$y_w(\tau) = \Pr_{W_i^C \sim \mathcal{D}^C \forall i} \left[\left| M(W_i^C) - \mathrm{E}[M] \right| \leq \tau \cdot \mathrm{E}[M] \; \forall i \right], \tag{5.2}$$

where $r = \ell t / w$ is the number of windows required to accommodate the covert channel at the given rate. Note that, we consider tainted packets only while computing the false-negative rate, and not when computing the false-positive rate. In general, we expect the false positive rate to keep decreasing as the window length is increased, given the reduced variance of the sampled distribution, as the sample size increases.

Given the above model, the effect of w and τ on the false positive and false-negative rate can be observed by evaluating the probabilities $x_w(\tau)$ and $y_w(\tau)$. The effects can be visualized by plotting the ROC (Receiver Operating Characteristic) curves $R_w = [x_w(\tau), 1 - y_w(\tau)]$, for different fixed values of w, while varying τ.

The effect of the covert channel throughput (anomaly magnitude) can be observed by plotting the same ROC curves for different values of the throughput parameter t.

5.2.2 Applying the Model

We now demonstrate applications of our model to specific scenarios. For the metric M we consider the empirical entropy H since it is the most widely studied. There are two effects on distributions that frequently occur due to anomalies. First, the anomaly introduces elements, all of the same value in the stream. Second, the anomaly introduces elements each of which has a distinct value. For instance, in network traffic, a port scan results in the effect of the second case on the destination port distribution, and the effect of the first case on the source IP distribution. For more examples and justification, see [LCD05] (network traffic), and [AY11] (information networks). For brevity we model only the first case in the first analytical application below.

We apply analytical models for the following distributions.

All identical elements, with noise

In this case, we assume that the normal distribution $\mathcal{D}$ consists of all regular elements in the window being identical. However, we allow for the possibility of noise, with the elements introduced by noise being all distinct. We model this distribution as follows. Without loss of generality, each element x_i, $1 \le i \le w$ in the window is either 0 with probability $1 - p$, or is a distinct non-zero random element with probability p, for parameter $0 \le p \le 1$. The parameter p controls the noise in the stream. The number of noise elements k in the window follows the Binomial Distribution $\mathcal{B}(w, p)$. If we let H_k denote the value of entropy when there are k noisy elements in the window, then we have

$$\begin{aligned} H_k &= H(\underbrace{1, 1, \ldots, 1}_{k \text{ times}}, w - k) \\ &= k \cdot (1/w) \log(w) + (w - k)/w \log(w/(w - k)), \\ \mathrm{E}[M] &= \mathrm{E}[H] = \sum_{k=0}^{w} \binom{w}{k} p^k (1 - p)^{w-k} H_k. \end{aligned}$$

We can now compute $x_w(\tau)$ by simply finding all values of k for which $|H_k - \mathrm{E}[H]| > \tau \cdot \mathrm{E}[H]$, and summing up the corresponding probabilities $\binom{w}{k} p^k (1 - p)^{w-k}$.

In order to compute $y_w(\tau)$, we need to incorporate the effect of the covert channel. We can write $y_w(\tau) = (y'_w(\tau))^r$, where $y'_w(\tau) = \Pr_{W^C \sim \mathcal{D}^C} \left[|M(W^C) - \mathrm{E}[M]| \le \tau \cdot \mathrm{E}[M] \right]$, where r is the number of windows containing tainted packets (since windows are sampled independently). We consider the case when the covert channel introduces tainted elements that are all distinct and non-zero, and further that they are also distinct from noise elements, if any. If $w \le \ell t/2$, when we are guaranteed to have a complete window overlapping the covert channel. To see this consider the case when $w = \ell t$; it may happen that a window overlaps (and ends at) the first half of the covert channel (of length $\ell t/2$), and the following window overlaps (and begins at) the latter half of the covert channel. But when $w \le \ell t/2$ this cannot happen and we will have $\lfloor w/t \rfloor$ tainted elements in some window. On the other hand, if $w > \ell t/2$, then we will have a window with at least $\ell/2$ tainted elements. So if

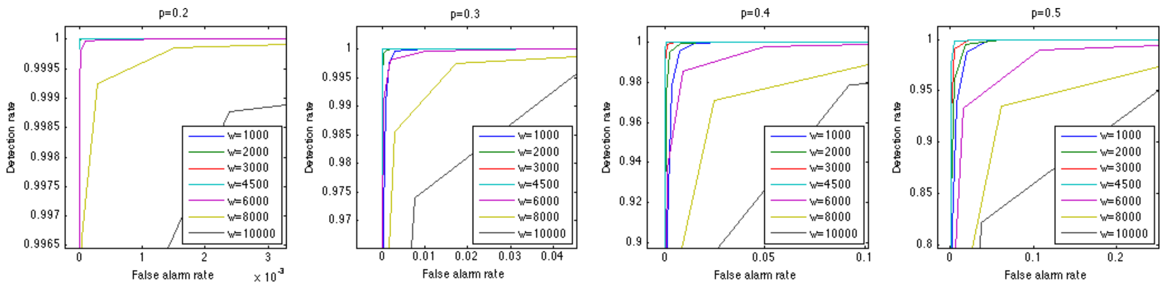


Figure 5.2: ROC curves computed analytically for the distribution in Case 1. [Best viewed in color.]

we let $c = \lfloor \min(w, \ell t/2)/t \rfloor$, then we get a histogram with k distinct noise elements and $\geq c$ distinct tainted elements (we assume there are exactly c tainted elements to get a lower bound on the detection rate). The number of noise elements now follows $\mathcal{B}(w - c, p)$. If we now let H_k^C denote the value of entropy in the presence of the covert channel, we have

$$\begin{aligned} H_k^C &= H(\underbrace{1, 1, \ldots, 1}_{k+c \text{ times}}, w - k - c) \\ &= (k + c) \cdot \frac{1}{w} \log(w) + \frac{w - k - c}{w} \log\left(\frac{w}{w - k - c}\right), \end{aligned}$$

We can now compute $y'_w(\tau)$ by simply finding all values of k for which $\left|H_k^C - \mathrm{E}[H]\right| > \tau \cdot \mathrm{E}[H]$, and summing up the corresponding probabilities $\binom{w-c}{k} p^k (1 - p)^{w-k-c}$.

Observations The ROC curves of the false positives and detection rates obtained in this manner are shown in Figure 5.2. The covert channel in the plots are introduced with parameters $\ell = 600$, and $t = 15$. There are four plots, one for each different value of p that we set. We observe that, other factors being equal, the ROC curves drift towards the 45-degree line (classifier quality degrades) as the noise probability increases. This result is primarily due to the increase in variance and hence the rate of false alarms as the noise levels increase. With a fixed value of p within a plot we see, however, that as the window length is increased, the ROC curves first approach the ideal classifier, until the window length is $w = \ell t/2 = 4500$. Thereafter, as the window length is increased further, the classifier quality deteriorates towards the 45-degree diagonal line. This suggests that the ideal window length for detection is $w = \ell t/2$. This may be clear as a consequence of the maximum value $\ell t/2$ possible for the parameter c. However, the reason is not completely clear merely by looking at the expressions for $x_w(\tau)$ and $y_w(\tau)$. A direct reason for the suggested setting of w is that as the window length is increased, beyond the length $\ell t/2$, the proportion of tainted packets in each window containing the covert channel keeps declining, resulting in a lower detection rate.

Multinomial distribution

In this general case we assume the normal distribution $\mathcal{D}$ consists of each element $i \in [n]$ being produced independently with some probability p_i in the stream ($\sum_{i=0}^{n-1} p_i = 1$). That is, the number of occurrences (frequency) m_i of any element i in a window of length w is distributed according to the multinomial distribution $\mathcal{M}(p_0, p_1, \ldots, p_{n-1}, w)$. Then for each possible frequency vector $\vec{m} = (m_0, m_1, \ldots, m_{n-1})$ possible in the window (i.e., for each such vector with $\sum_{i=0}^{n-1} m_i = w$), we have that

$$\Pr[\vec{m}] = w!/(m_0!m_1! \ldots m_{n-1}!)p_0^{m_0} p_1^{m_1} \ldots p_{n-1}^{m_{n-1}}.$$

The probabilities $p_i, 0 \leq i \leq n-1$ may be determined by observing a "training set" stream of regular elements. If we denote the entropy with frequency vector $\vec{m}$ as $H(\vec{m})$, then we have

$$\begin{aligned} H(\vec{m}) &= \sum_{i=0}^{n-1} m_i/w \log(w/m_i), \text{ and} \\ \mathrm{E}[H] &= \sum_{\vec{m} \text{ s.t. } \sum_i m_i = w} \Pr[\vec{m}] \cdot H(\vec{m}). \end{aligned}$$

We are now in a position to compute $x_w(\tau)$ from Equation 5.1 as before by summing up the relevant probabilities. We compute $y_w(\tau)$ similarly as before from Equation 5.2, only this time we use the Multinomial Distribution $\mathcal{M}(p_0^C, p_1^C, \ldots, p_{n-1}^C, w)$ instead that incorporates the covert channel. The probabilities p_i^C now may be obtained from a training set stream having the covert channel.

Computation. The difficulty of computing $x_w(\tau)$ and $y_w(\tau)$ is that it requires looping over the set of all possible vectors $\vec{m}$ such that $\sum_{i=0}^{n-1} m_i = w$. Generating such vectors is a non-trivial process, but more importantly as w increases, the number of such vectors grows at a rate that is exponential or greater. So, instead of computing $x_w(\tau)$ and $y_w(\tau)$ directly from the formulas, we instead perform a Monte-Carlo simulation to sample values from distribution $\mathcal{D}$, and use those values to compute first $\mathrm{E}[H]$ and then $x_w(\tau)$ and $y_w(\tau)$. With a large number of simulations we expect to get results that are close to the actual values of $x_w(\tau)$ and $y_w(\tau)$. We ran at least 1000 simulations in each case, and more as needed to ensure that the change in the values computed using the samples was not more than 0.1%.

Parameter ranges. We compute $x_w(\tau)$ and $y_w(\tau)$ in this manner for $t = 10$, 15, 20, 25, 30, 40, 50, and $\ell = 600$, as is the case in our experiments in the next section. We use window lengths from $w = 1000, 2000, 3000, 4500, 6000, 8000, 10000, 20000, 30000$. These values straddle the length of the covert channel ℓt for the values of t. We observed in previous models that the classification quality shows an inflection point at $w = \ell t/2$.

Features. We consider the following features that are affected by our covert channel described in Section 5.1:

- Byte 0 of RTP payload. In tainted packets this byte contains the fixed magic number, whereas in regular packets the byte is distributed according to the bytes generated by the G.711a voice encoding codec over our input voice stream. For the covert distribution, we chose the probabilities p_i^C with $p_k^C = 1$, and $p_i^C = 0$ for $i \neq k$, where k is the magic number. On the other hand for the regular feature probability distribution, we computed the probabilities p_i using our training set generated from a real VoIP call with about 350,000 RTP packets. This training set is described in detail Section 5.3. We found that the probability distribution p_i was close to a uniform distribution.

- Byte 7 of RTP payload. This byte contains a sequence number used internally by the covert channel. The sequence number is incremented sequentially for every new tainted packet. In regular packets, this feature contains G.711a voice encoded bytes. For the regular packet we chose the probabilities p_i from the training set (again we found it is close to normal). We set the covert channel probabilities $p_i^C = 1/n$ (i.e., uniformly distributed), since this is the best we can do given the multinomial distribution model.

- Byte 8 of RTP payload. In regular packets, this feature contains G.711a voice encoded bytes. For the regular packets we choose the probabilities p_i from the training set (again we find it is close to uniform). In the tainted packets this byte is distributed according to ASCII text. We used our entire text input, the text of *Alice's Adventures* *in Wonderland*, to compute the covert channel probabilities p_i. Note, however, that in sending the text via the covert channel, only specific bytes from the file at periodic offsets will appear in this byte position in the covert channel. The other bytes will occupy other bytes in the covert payload.

- RTP sequence number. This is the last byte of the RTP sequence number that is incremented sequentially for each new RTP packet. For regular packets, we set the probabilities $p_i = 1/n$ (i.e., uniformly distributed), since this is the best we can do given the multinomial distribution model. Our covert channel relies on setting the RTP sequence number to zero. So, for tainted packets, we set the probabilities p_i^C such that $p_0^C = 1$, and $p_i^C = 0$ for $i \neq 0$.

The ROC curves computed as a result of this analysis are shown in Figure 5.3. For brevity, we only include the plots for $t = 25, 30, 40, 50$.

Observations. As t increases (i.e., the covert channel throughput decreases) we observe that the ROC curves tend to be pulled toward the 45-degree diagonal. This indicates that performance degrades as a result of the decrease in detection rate due to the dilution of tainted packets by the regular packets in each window containing the covert channel. The effect of t on the false-alarm rate is a little harder to observe from the figure, but careful observation (for example, by looking at the end-points of the ROC curves) shows that the false-alarm rate remains in the same interval range for all values of t, indicating that the false-alarm rate was not affected by the change in t. And the reason for this effect is easy to see, because the false-positive rate is computed in the absence of anomalies.

Now, as w is increased, we observe consistently in the plots that the false-alarm rate decreases. The reason for this observation may be that as the number of samples in the

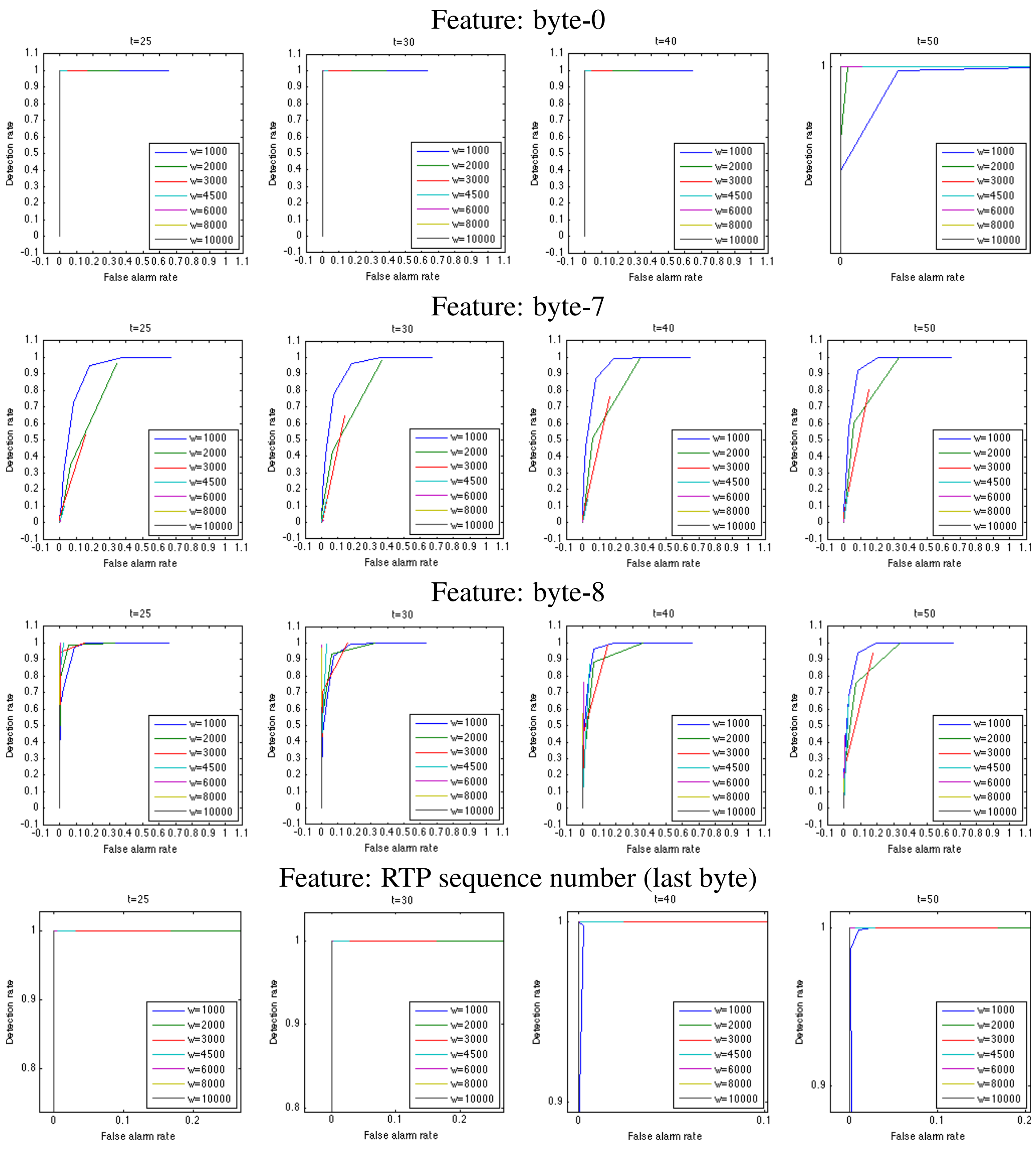


Figure 5.3: ROC curves computed analytically for Case 2 with various features. [Best viewed in color.]

window from the underlying distribution is increased, the variance decreases (analogous to the Law of Large Numbers). However, the effect of w on the detection rate is not consistent. As w increases beyond $\ell t/2$ we may expect the detection rate *per window* to decrease since the proportion of tainted elements decreases in a window (as also observed in Case 1). On the other hand, when $w < \ell t/2$ the detection rate per channel may decrease because of the smaller number of windows, when the window length is decreased. But although the proportion of tainted elements does not change in this case, different metrics react in different ways to the change in the window length and the detection rate may be determined by such metric behavior.

The combined effect of the detection rate and the false-alarm rate makes it hard to see the change in the detection rate alone by increasing w. To see the effect of detection rate separately, in Figure 5.4 we plot it against various values of the threshold τ (for the byte-0 feature with $t = 10$). We observe that for each window length, the detection rate fell from 1 to 0 as the threshold is increased. However, the fall was at a much later threshold for window lengths 1000, 2000 and 3000, and as the window length is increased we find that the fall occurs earlier. This observation indicates that when the window length was increased beyond 3000, the detection rate decreased. When $w < \ell t/2$, with the byte-7 and byte-8 features, we see the detection rate mostly decreasing with increase in w. But with byte-0 and the RTP sequence number the detection rate increased as w is increased (at least for $t = 50$). We attribute this inconsistent behavior to the way the metric behaves under different distribution combinations (i.e., tainted element distribution and regular element distribution).

For the byte-8 feature we find that at higher throughputs ($t = 25, 30$) the ROC curve tends away from the 45-degree diagonal as w is increased. When looking at the ROC curve

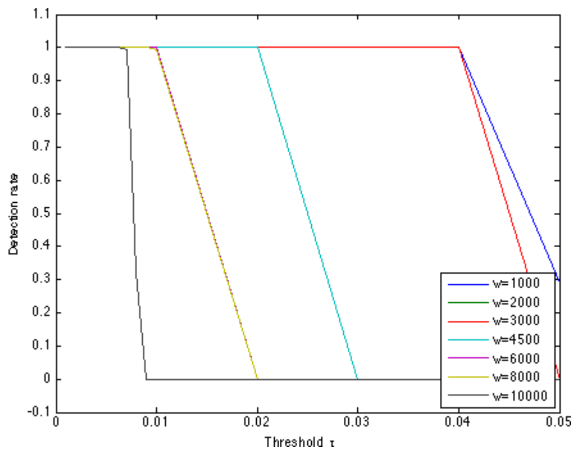


Figure 5.4: Detection rate for various windows. Feature: byte-0; t=10. [Best viewed in color.]

as a whole we observe that for our features, as the window length w is increased, the curve may drift away from the 45-degree diagonal or towards it depending on the feature and throughput.

Effect of Throughput. Our framework of analysis allows us also to see what effect the covert channel throughput would have on the detection quality. To see this effect, we must now fix a value of w and plot the ROC curves corresponding to different values of the parameter t, and then repeat the process for different values of w. For brevity, here, we present only the results for $w = 1000$, with the byte-8 feature. The practitioner should apply the same process for different values of w while modeling his setting. The results are shown in Figure 5.5. We observe, as expected, that as the rate (as a fraction of the total VoIP traffic) decreased from 10% to 2%, the ROC curve moved towards the 45-degree diagonal, indicating that the detection quality is getting poorer. These curves enable the practitioner to understand what rates of covert channel he has a hope of detecting within his acceptable false alarm rate.

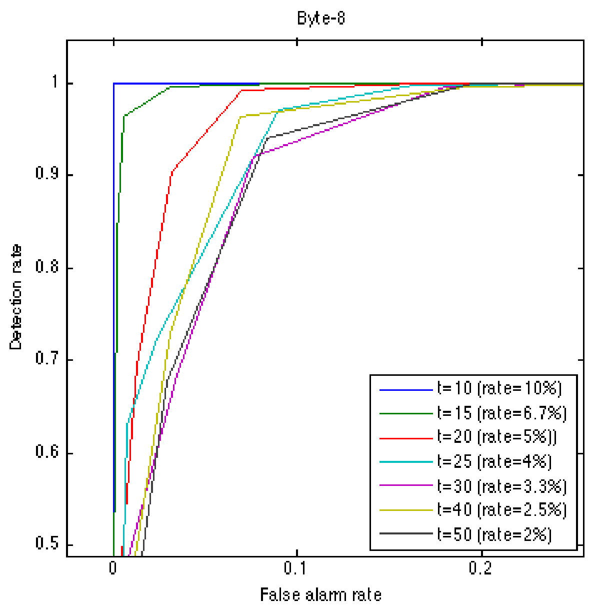


Figure 5.5: ROC curves for different throughput rates. Feature: byte-8; w=1000. [Best viewed in color.]

5.3 Experimental Analysis

In the previous section, our analytical models let us study some of the ways in which the knobs affect detection via our analytical models. In this section we look at the effects experimentally to understand the effects directly with actual calls in our testbed.

Dataset. First we describe our call setup and the datasets we use in our experiments. Our testbed consists of two client machines registered with an Asterisk SIP server. The clients are able to make calls to each other through their registration with the server, but the server is configured so that the RTP packets are exchanged directly between the clients without being routed through the server. Our dataset is generated by making calls in this manner. For the audio input in the call, we play a French language training audio program on both ends of the call. We save packet capture traces of calls, and our dataset consists of these traces. We save one trace consisting of approximately 350,000 RTP packets in each direction. This trace does not contain any tainted packets and serves to compute the expected normal value of entropy, and also to compute the false-alarm rate. We also capture traces while running our covert channel at different throughputs. The payload for our covert channel is the entire text of *Alice's Adventures in Wonderland* sent in 256 byte chunks per tainted packet. This results in a total of 602 tainted packets. For each value $t = 10, 15, 20, 25, 30, 40, 50$, we run a covert channel and save the call packet trace separately. These traces contain approximately just enough packets to include all tainted packets at the given covert channel throughput.

Experiments. We compute the entropy with window lengths $w = 1000$, 2000, 3000, 4500, 6000, 8000, 10000, 20000, 30000, in each of our data sets. In the data sets containing the covert channel we ensure that we do not consider any windows that do not contain any tainted packets at all. To increase our sample set size, in each trace, we slice the stream into windows, by first starting at different offsets $(1, 2, \ldots, 1000)$ from the start of the trace. We then compute the entropy in each window. Using these entropy values we are now able to compute the expected value $\mathrm{E}[H]$ and the false-alarm rate $x_w(\tau)$ from the trace not containing the covert channel, and then compute the detection rate $y_w(\tau)$ from the traces containing the covert channel.

Observations. The ROC curves from our experiments are shown in Figure 5.6. We have not included the plots for the RTP sequence feature, because we find that the false-alarm rate is always zero, leading to an empty figure. The reason for the zero false-alarm rate is that the distribution in each window is always the same—it consists of sequence numbers increasing consecutively. Note that in the analytical model in the previous section, the

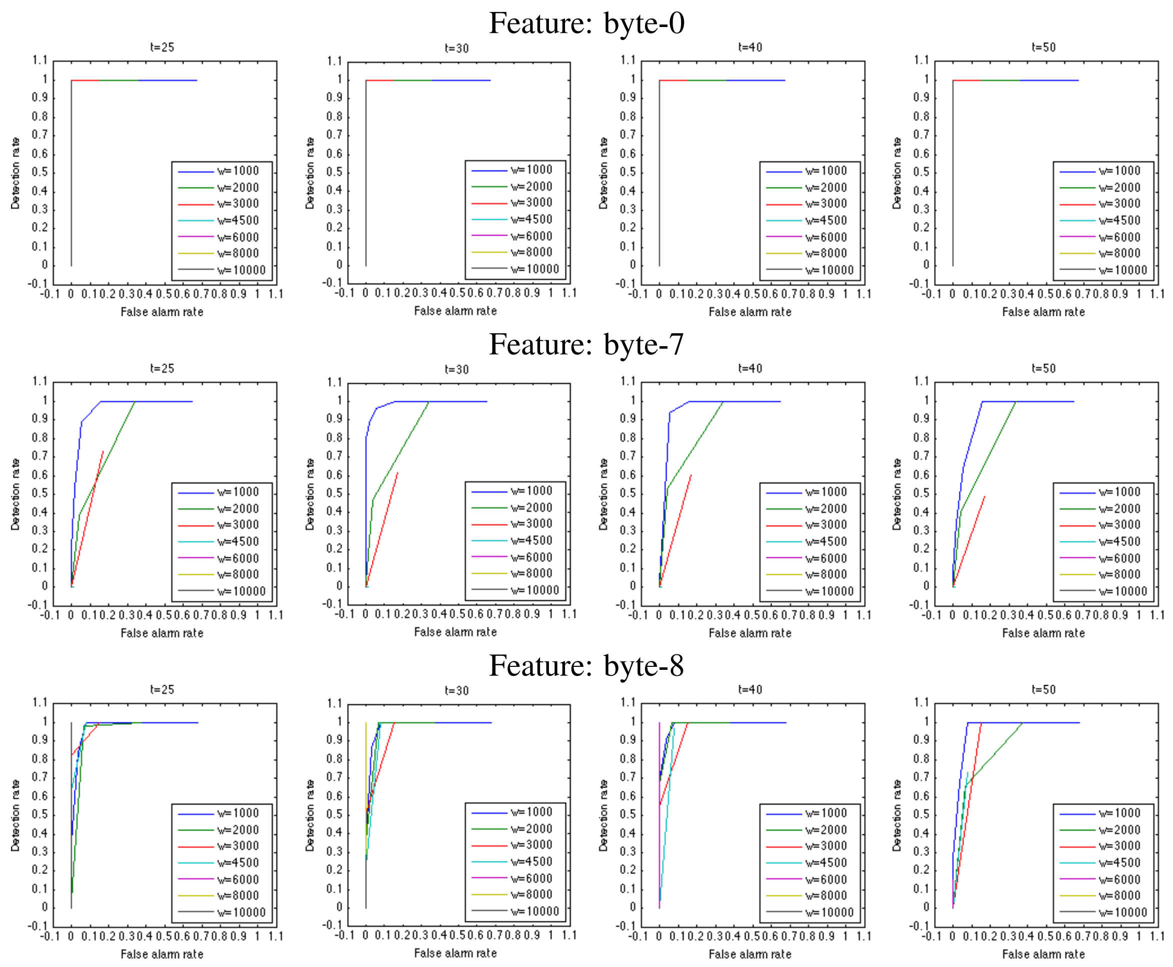


Figure 5.6: ROC curves computed experimentally with various features. [Best viewed in color.]

assumptions of the model—the underlying multinomial distribution—results in a non-zero false-positive rate in some cases. This demonstrates that the assumptions made by our analytical model may not always hold in practice. For other features our observations are similar to the analytical case. We found that increasing t tended to a poorer detection rate, resulting in a poorer overall detection quality even though the false-alarm rate was unaffected. For the effect of increase in w, we generally found that an increase in w led to a decrease in the detection rate, and a decrease in the false-alarm rate.

As in the analytical case, at higher throughputs for byte-8 we see the ROC curve move away from the 45-degree diagonal as w increases. But overall otherwise, we find, as in the analytical case, that as the window length is increased, the ROC curves move toward the 45-degree diagonal.

5.4 Recommendations

In this section we provide recommendations to a practitioner based on our results on how to tune the knobs for distribution-based anomaly detection methods in his specific setting. We also provide recommendations on how to efficiently apply the methods using sketching algorithms.

Our results show that,

- All other factors being equal, as the window length w is increased, the detection rate may increase or decrease, and the behavior may be different for $w < \ell t/2$ and for $w > \ell t/2$.

  The false-alarm rate, on the other hand, decreases monotonically as the window length is increased. This effect is attributed to the decrease in the variance as the number of samples in the window is increased. In the case when the distribution of regular elements is fixed in each window (like with the RTP sequence number), the false positive rate is always zero (even as the window length is increased).

- For a fixed window length, all other factors being equal, as the threshold is increased, both the false-alarm rate and the detection rate decrease. This effect is in agreement with our basic intuition.

- For a fixed window length, all other factors being equal, the detection rate increases with the increase in covert channel throughput. The covert channel rate has no bearing on the false-alarm rate, since the false-alarm rate is computed in the absence of anomalies.

Since the rate of change of the detection rate and false-alarm rate might be independent, an optimal tuning of the window length might depend on the covert channel throughput. Thus, it may be necessary to run the detection methods with several different window lengths, depending on the throughputs of the covert channels that one expects. Below we describe how detection methods with multiple window lengths can be employed efficiently. To tune the window length w and the threshold τ the practitioner must repeat our factorial-design-like analysis to compute the false-alarm rate and detection rate for several settings of the knobs and the covert channel throughput and compare them via ROC curves.

Efficient Monitoring with Sketching Algorithms. We now discuss how we can run distribution-based methods efficiently when our analysis suggests it is necessary to do so at different window lengths. As we mentioned in Chapter 2, data stream algorithms enable efficient online estimation of metrics of distributions when the input distribution arrives in a stream-like fashion. Suppose the stream σ consists of m elements $a_1, a_2, \ldots, a_m$ from the universe $[n]$. A data-stream algorithm estimates (to within some error bound) metric $M(\sigma)$ efficiently with respect to memory usage and computation. In our case, the stream σ is the window W for each window in network traffic. When the rate of traffic is massive, it is not feasible to compute $M(W)$ by first storing the entire window in memory. The problem is compounded when many features and metrics are to me monitored. Further, as we observe, monitoring metrics at different window lengths may be required. Data-stream algorithms, particularly sketching algorithms, address these problems. Recalling Section 3.1, these algorithms maintain a data structure $\mathcal{D}(W)$ called a *sketch*, from which the metric value $M(W)$ can be estimated. Sketches have the property that they can be combined efficiently: for two disjoint windows W_1 and W_2, given only their sketches $\mathcal{D}(W_1)$ and $\mathcal{D}(W_2)$, it is possible to combine them to obtain $D(W_1 \cup W_2)$. So in addition to estimating $M(W_1)$ and $M(W_2)$, sketches allow us to additionally estimate $M(W_1 \cup W_2)$ at almost no additional computational cost. The property extends to the case when sketches for multiple disjoint windows are provided. Sketching algorithms for computing entropy [HNO08] and the frequency moments [BGKS06] are available.

Now, when our analysis suggests that we run our distribution-based method with window lengths $w_1, w_2, \ldots, w_t$, we can run one instance of the distribution-based method with a window length of $w = \mathrm{GCD}(w_1, w_2, \ldots, w_t)$ sketching algorithms. Then, to estimate the metric value $M(W)$ in a window W of length w_i, we only need to combine the sketches from the w_i/w sub-windows, and estimate $M(W)$ from the combined sketch. Note, however, that the error bounds on the estimates are usually a function of the length of the stream, so that applying this method to combine a large number of windows may result in larger estimation errors. We refer the reader to the relevant algorithms [HNO08, BGKS06] for specific details on the error bounds.

Chapter 6

Distributed Monitoring Algorithms

Energy efficiency is a key issue in sensor network systems and communication, which typically uses power-hungry radio, is a vital resource whose usage needs to be minimized [EGHK99]. Several other distributed systems have a similar need for minimizing communication. This is the primary motivation for our present work, which is a natural successor to the recent work of Cormode, Muthukrishnan and Yi [CMY08], who introduced a clean formal model to study this issue. The formalization, known as *distributed functional monitoring*, involves a multi-party communication model consisting of *k sites* (the sensors, in a sensor network) and a single central *coordinator*. Each site asynchronously receives "readings" from its environment, formalized as a *data stream* consisting of *tokens* from a discrete universe. The union of these streams defines an overall input stream σ that the coordinator wishes to monitor continuously, using an appropriate protocol involving private two-way communication channels between the coordinator and each site. Specifically, the coordinator wants to continuously maintain approximate knowledge of some nonnegative real-valued function f of σ. (We assume that f is invariant under permutations of σ, which justifies our use of "union" above, rather than "concatenation.")

As is often the case in computer science, the essence of this problem is captured by a threshold version with Boolean outputs. Specifically, we have a threshold $\tau \in \mathbb{R}_+$ and an approximation parameter $\varepsilon \in \mathbb{R}_+$, and we require the coordinator to continuously maintain an output bit, which should be 0 whenever $f(\sigma) \leq \tau(1-\varepsilon)$ and 1 whenever $f(\sigma) \geq \tau$.[1] Following Cormode et al. [CMY08], we call this the $(k, f, \tau, \varepsilon)$ functional monitoring problem. As we shall see, this formulation of the problem combines aspects of streaming algorithms, sketching and communication complexity.

Motivation. Plenty of recent research has studied such continuous monitoring problems, for several special classes of functions f (see, e.g., [BO03, DGGR04, CMZ06, SSK07]). Applications have arisen not only in sensor networks, but also in more general network and database settings. However, most of this past work had not provided formal bounds on communication cost, an issue that was first addressed in detail in [CMY08], and that we continue to address here. Philosophically, the study of such monitoring problems is a vast generalization of Slepian-Wolf style distributed source coding [SW73] in much the same way that communication complexity is a vast generalization of basic source coding in information theory. Furthermore, while the problems and the model we consider here are strongly reminiscent of streaming algorithms, there are notable additional challenges: for instance, maintaining an approximate count of the total number of tokens received is a nontrivial problem in our setting, but is trivial in the streaming model. For further motivation and a more detailed discussion of prior research, we refer the reader to [CMY08] and the references therein.

[1]Clearly, a solution to the value monitoring problem solves this threshold version, and the value monitoring problem can be solved by running, in parallel, several copies of a solution to this threshold version with geometrically spaced thresholds.

Our Results and Comparison with Prior Work. Our work studies $(k, f, \tau, \varepsilon)$ functional monitoring for two natural classes of functions f: the frequency moments F_p, and the empirical Shannon entropy H and its generalization the Tsallis entropy. For an input stream σ of tokens from the universe $[n] := \{1, 2, \ldots, n\}$, let f_i denote the number of appearances of i in σ, where $i \in [n]$. As defined in Chapter 2, for $p \geq 0$, the pth frequency moment $F_p(\sigma)$ is $\sum_{i=1}^{n} f_i^p$. Note that p can be non-integral or zero: indeed, using the convention $0^0 = 0$ makes $F_0(\sigma)$ equal to the number of distinct tokens in σ. These functions F_p capture important statistical properties of the stream and have been studied heavily in the streaming algorithms literature [AMS99, Mut03]. The stream σ also implicitly defines a probability distribution over $[n]$, given by $\Pr[i] = f_i/m$, where m is the length of σ. For various applications, especially ones related to anomaly detection in networks that we have already seen in prior chapters, the Shannon entropy of this distribution, $H(\sigma)$— also called the empirical entropy of the stream — is a measure of interest[2]. Finally, we also consider the family of functions $T_\alpha(\sigma) = (1 - \sum_{i=1}^{n}(f_i/m)^\alpha)/(\alpha - 1)$, which are collectively known as Tsallis entropy [Tsa88] and which generalize Shannon entropy, as shown by considering the limit as $\alpha \to 1$.

We obtain three new classes of results. First, we prove new communication lower bounds for $(k, F_p, \tau, \varepsilon)$ functional monitoring, for various values of p. These either improve or are incomparable with previous lower bounds due to Cormode et al. [CMY08]; see Table 6.2 for a side-by-side comparison. For the rest of the chapter, we study the effect of *non-monotonicity* of f on the $(k, f, \tau, \varepsilon)$ problem: the bounds in [CMY08] crucially exploited the fact that the functions being monitored were monotone nondecreasing.

Our second class of results is on $f = F_p$ with *deletions* allowed: i.e., the stream

[2]Throughout this chapter we use "log" to denote logarithm to the base 2 and "ln" to denote natural logarithm.

can contain “negative tokens” that effectively delete earlier tokens. We give strong lower bounds showing that the resulting non-monotonicity drastically changes things: essentially, no good upper bounds are possible. We also give a lower bound for monitoring H, which is non-monotone even in the usual deletion-free setting.

Thirdly, we provide nontrivial monitoring protocols for H, and the related functions T_α. For this, we suitably extend recent sketching algorithms such as those due to Bhuvanagiri and Ganguly [BG06] and Harvey et al. [HNO08]. Besides providing easily usable and simple algorithms, our results here show that our lower bound for H is in the right ballpark.

In this chapter, we present only our bounds for entropy; the complete set of results are in the full paper.

	Problem	**Deterministic bounds**	**Randomized bounds**
Non-monotone	F_0, with deletions	$\Omega(\min\{m, mk/\varepsilon\tau\})$	$\Omega(\min\{m/k, m/\varepsilon\tau\})$
	F_p, $p > 0$, with deletions	$\Omega(\min\{m, mk/\tau^{1/p}\varepsilon\})$	$\Omega(\min\{m/k, m/\tau^{1/p}\varepsilon\})$
	H	$\min\{m, \Omega((k/\sqrt{\varepsilon})\log(\varepsilon m/k))\}$	$\widetilde{O}\left(\frac{k\log^3 m}{\varepsilon^3\tau^3}\right)$

Table 6.1: Summary of our results for non-monotone functions.

	Problem	**Previous Results**	**Our Results**
Monotone	F_1, deterministic	$O(k\log\frac{1}{\varepsilon})$, $\Omega(k\log\frac{1}{\varepsilon k})$	$\Omega\left(k\log\frac{k+\tau}{k+\varepsilon\tau}\right)$
	F_0, randomized	$\widetilde{O}(k/\varepsilon^2)$, $\Omega(k)$	$\Omega(1/\varepsilon)$, $\Omega(1/\varepsilon^2)$, if round-based
	F_p, $p > 1$, randomized	$\widetilde{O}(k^2/\varepsilon + (\sqrt{k}/\varepsilon)^3)$, $\Omega(k)$, for $p = 2$	$\Omega(1/\varepsilon)$, $\Omega(1/\varepsilon^2)$, if round-based

Table 6.2: Summary of our results for monotone functions and comparison with previous work [CMY08].

6.1 Preliminaries

We now define some notation and state some useful results from earlier work that we use at various points. We use $|\sigma|$ to denote the length of the stream σ and $\sigma_1 \circ \sigma_2$ to denote the concatenation: σ_1 followed by σ_2. We typically use $S_1, \ldots, S_k$ to denote the k sites, and C to denote the coordinator, in a $(k, f, \tau, \varepsilon)$ functional monitoring protocol. We tacitly assume that randomized protocols use a public coin and err with probability at most $1/3$ at each point of time. These assumptions do not lose generality, as shown by appropriate parallel repetition and the private-versus-public coin theorem of Newman [New91]. We always use m to denote the overall input length (i.e., number of tokens) seen by the protocol under consideration. We state our communication bounds in terms of m, k and ε, and sometimes τ.

We sometimes consider a restricted, yet natural, class of protocols that we call *round-based* protocols; the precise definition follows. Note that all nontrivial protocols in [CMY08] are round-based, which illustrates the naturalness of this notion.

Definition 3. A *round-based* protocol for $(k, f, \tau, \varepsilon)$ functional monitoring is one that proceeds in a series of rounds numbered $1, \ldots, r$. Each round has the following four stages. (1) Coordinator C sends messages to the sites S_i, based on the past communication history. (2) Each S_i read its tokens and sends messages to C from time to time, based on these tokens and the Stage 1 message from C to S_i. (3) At some point, based on the messages it receives, C decides to end the current round by sending a special, fixed, end-of-round message to each S_i. (4) Each S_i sends C a final message for the round, based on all its knowledge, and then resets itself, forgetting all previously read tokens and messages.

Some of our lower bounds use reductions from a fundamental problem in communication complexity: the "gap Hamming distance" problem, denoted GHD_c, where $c \in \mathbb{R}_+$

is a parameter. In this problem, Alice and Bob are given $x, y \in \{0, 1\}^n$ respectively and want to output 1 if $\Delta(x, y) \geq \frac{n}{2} + c\sqrt{n}$ and 0 if $\Delta(x, y) \leq \frac{n}{2} - c\sqrt{n}$; they don't care what happens if the input satisfies neither of these conditions. We shall need the following lower bounds on the randomized communication complexity $\mathrm{R}(\mathrm{GHD}_c)$, as well as the one-way randomized communication complexity (where the only communication is from Alice to Bob) $\mathrm{R}^{\rightarrow}(\mathrm{GHD}_c)$. Proofs of these bounds, as well as further background on the problem, can be found in Woodruff [Woo07].

Theorem 1. *Suppose $c > 0$ is a constant. Then $\mathrm{R}(\mathrm{GHD}_c) = \Omega(\sqrt{n})$ and $\mathrm{R}^{\rightarrow}(\mathrm{GHD}_c) = \Omega(n)$. Here, the Ω notation hides factors dependent upon c.*[3] □

It is conjectured that the general randomized bound is in fact as strong as the one-way version. This is not just a tantalizing conjecture about a basic communication problem. Settling it would have important consequences because, for instance, the gap Hamming distance problem is central to a number of results in streaming algorithms. As we shall soon see, it would also have consequences for our work here.

Conjecture 4. *For sufficiently small constants c, we have $\mathrm{R}(\mathrm{GHD}_c) = \Omega(n)$.*

6.2 Bounds for Entropy

We next give both upper and lower bounds for estimating empirical entropy. The upper bound represents the first nontrivial algorithm for functional monitoring of a non-monotone function. We abuse notation somewhat and let H denote both the binary entropy function and the empirical entropy of a stream.

[3]The versions of these bounds in [Woo07] restrict the range of c, but this turns out not to be necessary.

6.2.1 A Lower Bound

Theorem 2. *For any $\varepsilon < 1/2$, any deterministic algorithm that solves $(k, H, \tau, \varepsilon)$ functional monitoring must communicate* $\min\{m, \Omega((k/\sqrt{\varepsilon})\log(\varepsilon m/k))\}$ *bits.*

Proof. We again use an adversarial argument that proceeds in rounds. Each round, the adversary will force the protocol to send at least one bit. The result will follow by showing a lower bound on the number of rounds r that the adversary can create, using no more than m tokens. Let $\tau = 1$, and let z be the unique positive real such that $H(\frac{z}{2z+1}) = 1 - \varepsilon$. Note that this implies $H(\frac{z}{2z+1}) > 1/2 > H(1/10)$, whence $\frac{z}{2z+1} > 1/10$, hence $z > 1/8$. Furthermore, an estimation of H using calculus shows that $z = \Theta(1/\sqrt{\varepsilon})$. Fix a monitoring protocol $\mathcal{P}$. The adversary only uses tokens from $\{0, 1\}$, i.e., the stream will induce a two-point probability distribution.

The adversary starts with a "round 0" in which he sends nine 1s followed by a 0 to site S_1. Note that at the end of round 0, the entropy of the stream is $H(1/10) < 1/2$. For $i \in \{0, 1, \ldots, r\}$, let a_i denote the number of 0s and b_i the number of 1s in the stream at the end of round i. Then $a_0 = 1$ and $b_0 = 9$. For all $i > 0$, the adversary maintains the invariant that $b_i = \lceil a_i(z + 1)/z \rceil$. This ensures that at the end of round i, the empirical entropy of the stream is

$$H\left(\frac{a_i}{a_i + b_i}\right) \le H\left(\frac{a_i}{a_i(1 + (z + 1)/z)}\right) = H\left(\frac{z}{2z + 1}\right) = 1 - \varepsilon\,,$$

which requires the coordinator to output 0.

Consider the situation at the start of round i, where $i \ge 1$. If each player were to receive $\lceil (b_{i-1} - a_{i-1})/k \rceil$ 0-tokens in this round, then at some point the number of 0s in the stream would equal the number of 1s, which would make the empirical entropy equal to 1 and require the coordinator to change his output to 1. Therefore, there must exist a site S_{j_i}, $j_i \in [k]$, who would communicate upon receiving these many 0-tokens in round i. In actuality, the adversary does the following in round i: he sends these many 0s to S_{j_i}, followed by as many 1s as required to restore the invariant, i.e., to cause $b_i = \lceil a_i(z+1)/z \rceil$. Clearly, this strategy forces at least one bit of communication per round. It remains to bound r from below. Note that the adversary's invariant implies $b_i - a_i \le a_i/z + 1$ and $a_i + b_i \le a_i(2z+1)/z + 1 = a_i(2 + 1/z) + 1$. Therefore, we have

$$a_i \;=\; a_{i-1} + \left\lceil \frac{b_{i-1} - a_{i-1}}{k} \right\rceil \;\le\; a_{i-1} + \left\lceil \frac{1 + a_{i-1}/z}{k} \right\rceil \;\le\; a_{i-1}\left(1 + \frac{1}{zk}\right) + 2\,.$$

Setting $\alpha = (1 + 1/zk)$ and iterating this recurrence gives $a_r \le a_0\alpha^r + 2(\alpha^r - 1)/(\alpha - 1) = a_0\alpha^r + 2zk(\alpha^r - 1) = \alpha^r(a_0 + 2zk) - 2zk$. Using our upper bound on $a_i + b_i$, the above inequality, and the facts that $a_0 = 1$ and that $z > 1/8$, we obtain

$$\begin{aligned} a_r + b_r \;&\le\; \alpha^r\,(1 + 2zk)\left(2 + \frac{1}{z}\right) - 2zk\left(2 + \frac{1}{z}\right) + 1 \;\le\; (2 + 1/z)\,(1 + 2zk)\,\alpha^r \\ &\le\; (2 + 1/z)\,(1 + 2zk)\,e^{r/zk} \;\le\; 60zke^{r/zk}\,. \end{aligned}$$

Therefore, we can have $a_r + b_r \le m$, provided $r \le zk\ln(m/(60zk))$. Recalling that $z = \Theta(1/\sqrt{\varepsilon})$, we get a lower bound of $\Omega\left(\frac{k}{\sqrt{\varepsilon}}\log(\varepsilon m/k)\right)$. □

6.2.2 An Upper Bound

We now give a randomized upper bound for $(k, H, \tau, \varepsilon)$ functional monitoring. In the sequel, we shall give upper bounds for the Tsallis entropies T_α, which generalize Shannon entropy, H. We need several technical lemmas in this section.

At a high level, our algorithm treats the stream as a probability distribution, and monitors changes in this distribution, as measured by the L_1 distance. Recall that for probability distributions μ, ν on the set $[n]$, we have $\|\mu - \nu\|_1 = \sum_{i=1}^{n} |\mu(i) - \nu(i)|$. Our first two lemmas show that changes in Shannon entropy and in Tsallis entropy are bounded by changes in the probability distribution (in the L_1 sense).

Lemma 5. *Let A and B be collections of items from $[n]$, and μ and ν denote the distributions induced by A and $A \cup B$ respectively. Let $m = |A|$. If $|B| \leq m$, then,*

$$|H(A \cup B) - H(A)| \;\leq\; \|\nu - \mu\|_1 \log(2m)\,.$$

Lemma 6. *Let A, B, μ, ν and m be defined as in Lemma 5. Then, for all $\alpha > 1$,*

$$|T_\alpha(A \cup B) - T_\alpha(A)| \;\leq\; \|\nu - \mu\|_1 \cdot \min\left\{\log(2m), \frac{\alpha}{\alpha - 1}\right\}\,.$$

Finally, we show that if the number of newly arrived items is small, then the induced probability distribution does not change by much.

Lemma 7. *Let A, B, μ, ν and m be defined as in Lemma 5. Then if $|B| < \ell$, then $\|\nu - \mu\|_1 < 2\ell/m$.*

We are now ready to describe our algorithm. Within it, we use an entropy sketching algorithm as a black box: such an algorithm creates a small-space *sketch* of a collection of items A that allows one to compute a good approximation $\hat{H}(A)$ to $H(A)$. The next theorem, due to Harvey, Nelson and Onak [HNO08], provides such a sketch.

Theorem 3. *For $\varepsilon > 0$, one can compute an ε-additive-error sketch for empirical entropy, of size $\tilde{O}(\varepsilon^{-2} \log m \log n \log(mn))$. Here, the $\tilde{O}$ notation hides factors polynomial* *in* $\log\log m$ *and* $\log(1/\varepsilon)$.[4] $\square$

The Algorithm. We proceed in multiple rounds. Let σ_i be the overall input stream at the end of the ith round, and let m_i denote the total number of items in σ_i. In round 0, sites send items directly to the coordinator. The coordinator ends the round after seeing a constant $c_0 = 100$ number of items. At the end of the round, the coordinator uses the entropy sketch from Theorem 3 to get an estimate $\hat{H}(\sigma_0)$ of $H(\sigma_0)$ with an additive error of $\hat{\varepsilon} := \varepsilon\tau/4$.

For rounds $i > 0$, the coordinator runs an F_1 monitoring algorithm such as the one described in [CMY08], with an error of $1/2$ and a threshold τ_i depending on the previous estimate of entropy. The coordinator sets $\tau_i = \min\{m_{i-1}, m_{i-1}\lambda_i/(2\log(2m_{i-1}))\}$, where $\lambda_i = \tau(1-\varepsilon/4) - \hat{H}(\sigma_{i-1})$ if $\hat{H}(\sigma_{i-1}) < \tau(1-\varepsilon/2)$, and $\lambda_i = \hat{H}(\sigma_{i-1}) - \tau(1-3\varepsilon/4)$ otherwise.

Either way, when the F_1 monitoring algorithm outputs 1, the coordinator ends round i, fetches entropy sketches for round i from each site to get $\hat{H}(\sigma_i)$, and outputs 1 iff $\hat{H}(\sigma_i) \geq \tau(1-\varepsilon/2)$.

Theorem 4. *The above is a randomized algorithm for* $(k, H, \tau, \varepsilon)$ *functional monitoring that communicates* $\tilde{O}\left(\frac{k\log^3 m \log n \log(mn)}{\varepsilon^3\tau^3}\right)$ *bits.*

Proof. We first analyze the correctness. In round 0, it is trivial for the coordinator to output the correct answer. Now, for round $i > 0$, suppose the coordinator outputs 0 at the end of round $i-1$. Then, we must have $\hat{H}(\sigma_{i-1}) \leq \tau(1-\varepsilon/2)$, whence $H(\sigma_{i-1}) < \tau(1-\varepsilon/4)$ by the bound on the sketching error. By the correctness of the F_1 monitoring algorithm, we

[4]The storage requirements here differ slightly from the text of [HNO08]. Specifically, their $\tilde{O}(\varepsilon^{-2}\log m)$ lower bound is on the number of machine words of storage, and does not include additional storage to handle a pseudorandom generator. Including these terms in the sketch size demands additional terms of $O(\log(mn))$ and $O(\log n)$ respectively.

receive at most τ_i items during round i. Therefore by Lemmas 5 and 7, when going from σ_{i-1} to σ_i, the total entropy will be less than τ throughout round i. Hence, the coordinator is free to output zero through the end of round i. If the coordinator instead outputs 1 at the end of round $i-1$, we are guaranteed to remain above $\tau(1-\varepsilon)$ in a similar manner.

To bound the communication cost, we need to estimate both the number of rounds, and the number of bits exchanged in each round. It is easy to see that for each round i, $\lambda_i \geq \varepsilon\tau/4$. Suppose the stream ends during round $r+1$. Then, we have

$$\begin{aligned} m \;\geq\; m_r \;\geq\; m_{r-1} + \tau_r/2 \quad &\geq \quad m_{r-1}\left(1 + \min\left\{1/2, \frac{\tau\varepsilon}{16\log(2m_{r-1})}\right\}\right) \\ &\geq \quad m_{r-1}\left(1 + \min\left\{1/2, \frac{\tau\varepsilon}{16\log(2m)}\right\}\right). \end{aligned}$$

The second bound above follows from the guarantee of the F_1 monitoring algorithm. By iterating the above recurrence for m_r, we obtain $m \geq c_0 \min\left\{(1+1/2)^r, \left(1+\frac{\tau\varepsilon}{16\log(2m)}\right)^r\right\}$, which in turn implies

$$r \leq \log(m/c_0) \Big/ \min\left\{\log(3/2), \log\left(1 + \frac{\tau\varepsilon}{8\log m}\right)\right\} = O\left(\max\left\{\log m, \frac{\log^2 m}{\tau\varepsilon}\right\}\right),$$

where the final bound uses $\ln(1+x) \geq x/(x+1)$ for all $x > 0$. In each round, we use $O(k\log m)$ bits to send τ_i to the sites and $O(k)$ bits for the F_1 algorithm (since we set the error to be $1/2$). Each round, each site sends the coordinator a sketch update. Using the bound from Theorem 3 for the size of these sketches, we can bound the total communication by $\tilde{O}\left(\frac{\log^2 m}{\tau\varepsilon}\left(k + k\log m + \frac{k\log m\log n\log(mn)}{\varepsilon^2\tau^2}\right)\right) = \tilde{O}\left(\frac{k\log^3 m\log n\log(mn)}{\varepsilon^3\tau^3}\right)$, assuming that $\log m \geq \tau\varepsilon$. □

Our final result is an algorithm for monitoring Tsallis entropy. Lemma 6 bounds T_α in the same way that Lemma 5 bounds Shannon entropy. The rest of the algorithm and proof are the same except for some slight changes in parameters; we refer the reader to the full paper for the details.

Theorem 5. *There is a randomized algorithm for $(k, T_\alpha, \tau, \varepsilon)$ functional monitoring that communicates $\tilde{O}\left(\frac{k \log^3 m \log n \log(mn)}{\varepsilon^3 \tau^3}\right)$ bits.* □

6.2.3 Proofs of Technical Lemmas

The first two proofs use the multivariate Mean-Value Theorem, which we give here for completeness.

Fact 8 (Multivariate Mean-Value Theorem). *If $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable on the line segment between $\vec{a}$ and $\vec{b}$, then there exists a point $\vec{c}$ on that line segment such that $f(\vec{b}) - f(\vec{a}) = \nabla f(\vec{c}) \cdot (\vec{b} - \vec{a})$.*

Proof of Lemma 5. For $i \in [n]$, let $p_i = \nu(i), q_i = \mu(i)$, and $\delta_i = |p_i - q_i|$. Note that $|A| = m$, and $|A \cup B| \leq 2m$, so when p_i and q_i are positive, we must have $p_i \geq 1/2m$ and $q_i \geq 1/m$. Let $\vec{p} = (p_1, \ldots, p_n)$ and $\vec{q} = (q_1, \ldots, q_n)$ be the probability distribution vectors. Let U be the support of μ, and let $T = [n] \setminus U$. For $\vec{x} \in \mathbb{R}^n$, let $\vec{x}_U$ be the projection of $\vec{x}$ onto U. Finally, let $f(\vec{x}_U) = \sum_{i \in U} x_i \log x_i$, so that $H(A \cup B) = -\sum_{i \in T} p_i \log p_i - f(\vec{p}_U)$, and $|H(A \cup B) - H(A)| = |\sum_{i \in T}(p_i \log p_i - q_i \log q_i) + f(\vec{p}_U) - f(\vec{q}_U)| \leq \sum_{i \in T} |p_i \log p_i| + |f(\vec{p}_U) - f(\vec{q}_U)|$. We handle the cases $i \in U$ and $i \in T$ separately.

Consider $i \in U$. By straightforward differentiation, $\partial f(\vec{x})/\partial x_i = (1 + \ln x_i)/\ln 2$. By the Mean-Value Theorem, there exists $\vec{z} = (z_i)_{i \in U}$ such that $|f(\vec{p}_U) - f(\vec{q}_U)| = |\nabla f(\vec{z}) \cdot (\vec{p}_U - \vec{q}_U)|$. Clearly, each z_i lies between p_i and q_i, whence $z_i > 1/2m$. Furthermore,

$|(\vec{p}_U - \vec{q}_U)_i| = \delta_i$. Therefore,

$$|f(\vec{p}_U) - f(\vec{q}_U)| = \left|\sum_{i \in U} \frac{1 + \ln z_i}{\ln 2} \cdot \delta_i\right| \leq \sum_{i \in U} \left|\frac{1 + \ln z_i}{\ln 2}\right| \cdot \delta_i \leq \log(2m) \sum_{i \in U} \delta_i \, .$$

Consider $i \in T$. If $p_i > 0$, then $\delta_i = p_i \geq 1/2m$ and $|p_i \log p_i| = |\delta_i \log p_i| \leq \delta_i \log 2m$. If $p_i = 0$ then we trivially have $|p_i \log p_i| \leq \delta_i \log(2m)$. Putting both cases together, we see that $|H(A \cup B) - H(A)| \leq \sum_{i \in T} \delta_i \log(2m) + \sum_{i \in U} \delta_i \log(2m) = \log(2m)\|\nu - \mu\|_1$. □

Proof of Lemma 6. The lemma holds by combining two independent bounds. We borrow most of this notation from Lemma 5. The first bound uses the Mean-Value Theorem. Let $g(x) = x - x^\alpha$, and $f(\vec{x}_U) = (\sum_{i \in U} g(x_i))/(\alpha - 1)$. Also, note that, e.g., $T_\alpha(B) = \sum_{i=1}^n g(p_i)/(\alpha - 1)$.

As in the proof of Lemma 5, we need to be careful about items that aren't in the support. Consider $i \in T$. If $p_i = 0$ as well, then we clearly have $g(p_i) - g(q_i) = 0$. Alternatively, if $p_i > 0$, then $g(p_i) - g(q_i) = \frac{p_i - p_i^\alpha}{\alpha - 1} = \frac{\delta_i(1 - p_i^{\alpha-1})}{\alpha - 1}$. Using $1 - x < -\ln x$ for all $x > 0$, the above term becomes $g(p_i) - g(q_i) < \frac{\delta_i}{\alpha - 1} \ln(1/p_i^{\alpha-1}) = \frac{\delta_i}{\alpha-1}(\alpha - 1)\ln(1/p_i) \leq \delta_i \ln(2m) < \delta_i \log(2m)$ since $p_i > 1/2m$. In conclusion we have $g(p_i) - g(q_i) < \delta_i \log(2m)$ for all $i \in T$.

When $i \in U$, we use the Mean-Value Theorem. Suppose that $z_i > 1/2m$ for all $i \in U$. Then taking partial derivatives and using $1 - z < -\ln z$, we get

$$\frac{\partial f(\vec{z})}{\partial z_i} = \frac{1 - \alpha z_i^{\alpha-1}}{\alpha - 1} < \frac{-\ln(\alpha z_i^{\alpha-1})}{\alpha - 1} = \frac{(\alpha - 1)\ln(1/z_i) - \ln \alpha}{\alpha - 1} < \ln 2m - \frac{\ln \alpha}{\alpha - 1} \, .$$

By the Mean-Value Theorem, there exists $\vec{z}$ such that $|f(\vec{p}_U) - f(\vec{q}_U)| \leq |\nabla f(\vec{z}) \cdot (\vec{p}_U -$

$\vec{q}_U)|$. As in Lemma 5, we have $z_i > 1/2m$, $|(\vec{p}_U - \vec{q}_U)_i| = \delta_i$, and similarly

$$|f(\vec{p}_U - f(\vec{q}_U)| \;=\; \left|\sum_{i\in U}\delta_i(\log 2m - \ln\alpha/(\alpha-1))\right| \;\le\; \sum_{i\in U}\delta_i\log(2m)\,.$$

Putting these bounds together yields

$$\begin{aligned}|T_\alpha(A\cup B) - T_\alpha(B)| \;&\le\; \sum_{i\in T}|g(p_i) - g(q_i)| + |f(\vec{p}_U) - f(\vec{q}_U)| \\ &\le\; \sum_{i\in T}\delta_i\log(2m) + \sum_{i\in U}\delta_i\log(2m) \;=\; \|\nu-\mu\|_1\log(2m)\,.\end{aligned}$$

For the second bound, we analyze Tsallis entropy more directly. Let $g(x) = x^\alpha$, so for all $0 < x < 1$ and $\alpha > 1$, we have $g'(x) = \alpha x^{\alpha-1} < \alpha$. In particular, we must have $g(x) - g(x-\delta) < \alpha\delta$ for all $0 \le \delta < x < 1$. Without loss of generality, assume $p_i \le q_i$ (the other case will be similar, with only the roles of p_i, q_i reversed). Then, we can write q_i as $q_i = p_i + \delta_i$ for some $0 \le \delta_i < q_i$. Therefore, the change in Tsallis entropy is at most

$$\begin{aligned}|T_\alpha(A\cup B) - T_\alpha(A)| \;&=\; \left|\frac{1 - \sum_{i=1}^n p_i^\alpha - 1 + \sum_{i=1}^n q_i^\alpha}{\alpha-1}\right| \\ &=\; \frac{1}{\alpha-1}\left|\sum_{i=1}^n g(q_i) - g(q_i-\delta_i)\right| \;<\; \frac{\alpha}{\alpha-1}\left|\sum_{i=1}^n\delta_i\right| \;=\; \frac{\alpha}{\alpha-1}\|\nu-\mu\|_1\,. \quad \square\end{aligned}$$

Proof of Lemma 7. Let $t = |B|$, and write the elements of B as $B = \{b_1,\ldots,b_t\}$. For each $i \in [n]$, let f_i be the frequency of i in A, so that $\mu(i) = f_i/m$. For each $1 \le j \le t$, let $B_j = \{b_j\}$, and let ν_j be the probability distribution induced by $A \cup B_j$. We first consider

$\|\nu_j - \mu\|_1$. Without loss of generality, assume that $b_j = 1$. Then,

$$\begin{aligned}
\|\nu_j - \mu\|_1 &= \left|\frac{f_1+1}{m+1} - \frac{f_1}{m}\right| + \sum_{i>1}\left|\frac{f_i}{m} - \frac{f_i}{m+1}\right| \\
&= \frac{m-f_1}{m(m+1)} + \left(\sum_{i>1} f_i\right)\left(\frac{1}{m(m+1)}\right) \\
&= \frac{1-\mu(1)}{m+1} + \frac{1-\mu(1)}{m+1} \le \frac{2}{m+1}\,.
\end{aligned}$$

We now make a claim:

Claim. *Let A, B, C be collections of items from $[n]$. If μ, μ_B, μ_C, and μ_{BC} are the probability distributions induced by A, $A \cup B$, $A \cup C$, and $A \cup B \cup C$ respectively, then*

$$\|\mu_{BC} - \mu\|_1 \le \|\mu_B - \mu\|_1 + \|\mu_C - \mu\|_1\,.$$

Applying this claim repeatedly with our sets $A, B_1, \ldots, B_t$, we get

$$\|\nu - \mu\|_1 \;\le\; \sum_{j=1}^{t}\|\nu_j - \mu\|_1 \;\le\; \frac{2t}{m+1} \;<\; \frac{2\ell}{m}\,,$$

which proves the lemma.

Finally, we prove the claim. For $i \in [n]$. let a_i, b_i, c_i denote the frequency of i in A, B, C respectively. Also, let $x = |A|$, $y = |B|$, and $z = |C|$ denote the sizes of A, B, and C. We have the following probabilities for item i:

$$\mu(i) = \frac{a_i}{x} \quad , \quad \mu_B(i) = \frac{a_i + b_i}{x+y} \quad , \quad \mu_C(i) = \frac{a_i + c_i}{x+z} \quad , \quad \mu_{BC}(i) = \frac{a_i + b_i + c_i}{x+y+z}.$$

Then, we have

$$\begin{aligned}
|\mu_{BC}(i) - \mu(i)| &= \left|\frac{a_i + b_i + c_i}{x + y + z} - \frac{a_i}{x}\right| = \left|\frac{(b_i x - a_i y) + (c_i x - a_i z)}{x(x + y + z)}\right| \\
&\le \left|\frac{b_i x - a_i y}{x(x + y + z)}\right| + \left|\frac{c_i x - a_i z}{x(x + y + z)}\right| \\
&\le \left|\frac{b_i x - a_i y}{x(x + y)}\right| + \left|\frac{c_i x - a_i z}{x(x + z)}\right| \\
&= \left|\frac{a_i + b_i}{x + y} - \frac{a_i}{x}\right| + \left|\frac{a_i + c_i}{x + z} - \frac{a_i}{x}\right| \\
&= |\mu_B(i) - \mu(i)| + |\mu_C(i) - \mu(i)|.
\end{aligned}$$

The claim now follows directly, by summing over all $i \in [n]$. □

Chapter 7

Summary and Future Work

The reliable detection of anomalies in streams produced in many information systems is vital. In the case of anomaly detection in network traffic, several researchers have made a strong case for monitoring empirical entropy of network features. The computational challenges of keeping up with high-bandwidth network links are still significant, however, even after the recent theoretical advances in streaming algorithms. We show that robust estimation of entropy in network streams can be done in distributed fashion, thus dramaticially increasing its scalability in practical implementation.

We produced an implementation of the Bhuvanagiri-Ganguly HSS sketching algorithm [BG06], and evaluated its accuracy and memory requirements in estimating entropy using network data collected by our experimental wireless infrastructure at Dartmouth. With the extra scalability advantages of the distributed approach to entropy estimation, the use of pairwise *conditional* entropies for anomaly detection now comes within reach. We study the application of HSS to estimating conditional entropies of 802.11 link-layer features, and make a case study of the KoreK attack with both single-feature and pairwise-conditional entropies. We demonstrate both feasibility and superior sensitivity of this approach.

We observed some anomalies that were infeasible to explain given the limited information from the Layer-2 headers in our experiments with the DIST infrastructure. In the future, with permission to additionally collect IP, TCP, and UDP headers, it may be feasible to understand the reason underlying such anomalies as well.

We developed an efficient, online distribution-based anomaly detection methodology. Our evaluation on the Wikipedia dataset shows that it is possible to detect several kinds of anomalies with a detection rate that is higher than traditional methods, and a low false-positive rate.

We showed the complex effects the knobs of distribution-based methods can have on detection quality. We provide a analysis framework that practitioners can use to tune these knobs when implementing such methods in their own environments. We observe that in many scenarios it will be necessary to employ detection methods using several window lengths. In such cases, we provide an efficient approach using sketches to minimize memory and computation overhead. We also show how the practitioner can examine the capabilities and limitations of such methods in detecting low-magnitude anomalies (or low-throughput covert channels).

Future work. An important direction for future work on our methodology is to develop a classifier that combines inferences of the different metric-distribution combinations to further automate the diagnosis of anomalies.

Also, the distribution-based approach proposed by this thesis could be expanded further, by the inclusion of more metrics like KL-divergence, JS-divergence. A study of of what additional capabilities those metrics would provide and what distribution characteristics those metrics capture, along the lines of our study in Section 4.2 would be required. Further, the development of efficient data-stream algorithms to estimate these metrics would also be required.

In this thesis, we did not apply sliding-window-based methods to monitor distributions. Instead we rely on monitoring a timeseries of disjoint windows. The main reason for this is the unavailablilty of efficient sketching algorithms in sliding-window model. Development of these algorithms is an open problem.

Development of efficient data-stream algorithms to estimate graph properties like centrality and connectedness would be useful non-distribution-based anomaly detection metrics in graph-based networks like Wikipedia. The Wikipedia dataset we constructed supports research on such evolving graph properties.

When studying the tuning of parameters of distribution-based anomaly detection, a way to visualize the parameter space, and the effects on detection quality of moving in different directions in the parameter space would be invaluable.

The most exciting research direction for the future, in our opinion, is to apply the methodology to other networks, to find what anomalies lie therein.

Summary. In summary, the contributions of this thesis are:

- A unified distribution-based methodology for online anomaly detection in network traffic streams.
- Experimental evaluation of the methodology demonstrating its high detection rate and low false positive rate in three application scenarios—wireless anomaly detection, monitoring update streams in information networks, and detection of covert channels in VoIP traffic.
- Algorithmic results enabling efficient and scalable application of the methodology in

order to accomodate the massive data rates in modern networks.

- Extensive study of different aspects of the methodology—choice of metrics and their use in anomaly diagnosis, parameters of window length and detection threshold.

Bibliography

[ABBS10] C. Arackaparambil, S. Bratus, J. Brody, and A. Shubina. Distributed monitoring of conditional entropy for anomaly detection in streams. In *Proc. of IEEE Workshop on Scalable Stream Processing Systems*, 2010.

[ABC09] Chrisil Arackaparambil, Joshua Brody, and Amit Chakrabarti. Functional monitoring without monotonicity. In *ICALP '09: Proceedings of the 36th International Colloquium on Automata, Languages and Programming*, pages 95–106, Berlin, Heidelberg, 2009. Springer-Verlag.

[AMC07] R. Almeida, B. Mozafari, and J. Cho. On the evolution of wikipedia. In *ICWSM'07: Intl. AAAI Conference on Weblogs and Social Media*, 2007.

[AMF10] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *Advances in Knowledge Discovery and Data Mining, vol. 6119*, Springer LNCS. 2010.

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999. Preliminary version in *Proc. 28th Annu. ACM Symp. Theory Comput.*, pages 20–29, 1996.

[AY11] C. Arackaparambil and G. Yan. Wiki-watchdog: Anomaly detection in wikipedia through a distributional lens. In *To appear in Proc. of IEEE/ACM Web Intelligence*, 2011.

[AYBC12] Chrisil Arackaparambil, Guanhua Yan, Sergey Bratus, and Alper Caglayan. On tuning the knobs of distribution-based methods for detecting VoIP covert channels. *Hawaii International Conference on System Sciences*, 2012.

[Bac] Backtrack linux. `http://www.backtrack-linux.org/`.

[BCKP08] Sergey Bratus, Cory Cornelius, David Kotz, and Danie l Peebles. Active behavioral fingerprinting of wireless devices. In *Proceedings of the first ACM conference on Wireless network securi ty*, WiSec '08, pages 56–61, 2008.

[BG06] Lakshminath Bhuvanagiri and Sumit Ganguly. Estimating entropy over data streams. In *Proc. 14th Annual European Symposium on Algorithms*, pages 148–159, 2006.

[BGKS06] Lakshminath Bhuvanagiri, Sumit Ganguly, Deepanjan Kesh, and Chandan Saha. Simpler algorithm for estimating frequency moments of data streams. In *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 708–713, 2006.

[BKPR02] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *Proc. of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, 2002.

[BKT$^+$09] Sergey Bratus, David Kotz, Keren Tan, William Taylor, Anna Shubina, Bennet Vance, and Michael E. Locasto. Dartmouth Internet Security Testbed (DIST):

building a campus-wide wireless testbed. In *Proceedings of the Workshop on Cyber Security Experimentation and Test (CSET)*. USENIX Association, August 2009.

[BO03] Brian Babcock and Chris Olston. Distributed top-*k* monitoring. In *Proc. Annual ACM SIGMOD Conference*, pages 28–39, 2003.

[BTW$^+$06] Daniela Brauckhoff, Bernhard Tellenbach, Arno Wagner, Martin May, and Anukool Lakhina. Impact of packet sampling on anomaly detection metrics. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 159–164, New York, NY, USA, 2006. ACM.

[Cac] J. Cache, Fingerprinting 802.11 Implementations via Statistical Analysis of the Duration Field. `http://uninformed.org/?v=5&a=1`.

[CBK09] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41:15:1–15:58, July 2009.

[CCG11] V. Crespi, G. Cybenko, and A. Giani. Attacking and defending covert channels and behavioral models. *CoRR*, abs/1104.5071, 2011.

[CCM07] Amit Chakrabarti, Graham Cormode, and Andrew McGregor. A near-optimal algorithm for computing the entropy of a stream. In *Proc. 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 328–335, 2007.

[CDM06] Amit Chakrabarti, Khanh Do Ba, and S. Muthukrishnan. Estimating entropy and entropy norm on data streams. *Internet Mathematics*, 3(1):63–78, 2006. Preliminary version in *Proc. 23rd International Symposium on Theoretical Aspects of Computer Science*, LNCS 3884:196–205, 2006.

[CG04] Michael J. Cooper and Huseyin Gulen. Is Time-Series Based Predictability Evident in Real Time? *SSRN eLibrary*, 2004.

[Cle] `http://www.cisco.com/go/cleanair/`.

[CMY08] Graham Cormode, S. Muthukrishnan, and Ke Yi. Algorithms for distributed functional monitoring. In *Proc. 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1076–1085, 2008.

[CMZ06] Graham Cormode, S. Muthukrishnan, and Wei Zhuang. What's different: Distributed, continuous monitoring of duplicate-resilient aggregates on data streams. In *Proc. 22nd International Conference on Data Engineering*, page 57, 2006.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.

[CW77] J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions (extended abstract). In *STOC '77: Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 106–112, New York, NY, USA, 1977. ACM.

[DCRM10] A. D'Alconzo, A. Coluccia, and P. Romirer-Maierhofer. Distribution-based anomaly detection in 3g mobile networks: from theory to practice. *Int. J. Netw. Manag.*, 20:245–269, September 2010.

[Den87] Dorothy E. Denning. An intrusion-detection model. *IEEE Transactions On Software Engineering*, 13(2):222–232, 1987.

[DGGR04] Abhinandan Das, Sumit Ganguly, Minos N. Garofalakis, and Rajeev Rastogi. Distributed set expression cardinality estimation. In *Proc. 30th International Conference on Very Large Data Bases*, pages 312–323, 2004.

[DoD] Department of Defense. 1985. Trusted Computer System Evaluation Criteria. DoD 5200.28-STD.

[Dum] WikiDump. `http://en.wikipedia.org/wiki/Wikipedia:Database_download`.

[EGHK99] Deborah Estrin, Ramesh Govindan, John S. Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *MOBICOM*, pages 263–270, 1999.

[FS03] Laura Feinstein and Dan Schnackenberg. Statistical approaches to DDoS attack detection and response. In *In Proceedings of the DARPA Information Survivability Conference and Exposition*, pages 303–314, 2003.

[GBC06] A. Giani, V. Berk, and G. Cybenko. Data exfiltration and covert channels. In *Proc. of the SPIE Vol. 6201, Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense IV Orlando, Florida*, 2006.

[GH10] C. Gomez and R. Hornero. Entropy and complexity analyses in alzheimer's disease: An meg study. *Open Biomed Eng J*, 4:223–35, 2010.

[GMV06] Sudipto Guha, Andrew McGregor, and Suresh Venkatasubramanian. Streaming and sublinear approximation of entropy and information distances. In

Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms, 2006. to appear.

[GW07] S. Gianvecchio and H. Wang. Detecting covert timing channels: an entropy-based approach. In *Proc. of ACM CCS*, pages 307–316, 2007.

[HNO08] Nicholas J. A. Harvey, Jelani Nelson, and Krzysztof Onak. Sketching and streaming entropy via approximation theory. In *Proc. 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 489–498, 2008.

[HTY11] Yong Feng Huang, Shanyu Tang, and Jian Yuan. Steganography in inactive frames of VoIP streams encoded by source codec. *Information Forensics and Security, IEEE Transactions on*, 6(2):296 –306, june 2011.

[IM08] Piotr Indyk and Andrew McGregor. Declaring independence via the sketching of sketches. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 737–745, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[inf09] informIT. Byte-sized decryption of WEP with Chopchop. http://www.informit.com/guides/content.asp?g=security&seqNum=196, 2009.

[JJT] `http://www.jjtc.com/Security/stegtools.htm`.

[KGKM05] Vijay Karamcheti, Davi Geiger, Zvi Kedem, and S. Muthukrishnan. Detecting malicious network traffic using inverse distributions of packet contents. In *MineNet '05: Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, pages 165–170, New York, NY, USA, 2005. ACM.

[Kor] Korek chopchop. `http://www.aircrack-ng.org/doku.php?id=korek_chopchop`.

[LCD05] Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffic feature distributions. In *SIGCOMM*, pages 217–228, 2005.

[Les08] J. Leskovec. *Dynamics of Large Networks*. PhD thesis, CMU, 2008.

[Lig] Lightmouse banned. `http://en.wikipedia.org/wiki/Wikipedia:Requests_for_arbitration/Date_de%linking#Lightmouse_banned`.

[LSO+06] Ashwin Lall, Vyas Sekar, Mitsunori Ogihara, Jun Xu, and Hui Zhang. Data streaming algorithms for estimating entropy of network traffic. *SIGMETRICS Perform. Eval. Rev.*, 34(1):145–156, 2006.

[LX01] Wenke Lee and Dong Xiang. Information-theoretic measures for anomaly detection. In *IEEE Symposium on Security and Privacy*, pages 130–143, 2001.

[Met] The Metasploit project. `http://www.metasploit.com/`.

[Mut03] S. Muthukrishnan. Data streams: Algorithms and applications. In *Proc. 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, page 413, 2003.

[New91] Ilan Newman. Private vs. common random bits in communication complexity. *Information Processing Letters*, 39(2):67–71, 1991.

[NSA+08] George Nychis, Vyas Sekar, David G. Andersen, Hyong Kim, and Hui Zhang. An empirical evaluation of entropy-based traffic anomaly detection. In *IMC*

'08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement, pages 151–156, New York, NY, USA, 2008. ACM.

[SAN] `http://www.sans.org/security-resources/idfaq/covert_chan.php`.

[SCT$^+$08] Yong Sheng, Guanling Chen, Keren Tan, Udayan Deshpande, Bennet Vance, Hongda Yin, Chris McDonald, Tristan Henderson, David Kotz, Andrew Campbell, and Joshua Wright. MAP: A scalable monitoring system for dependable 802.11 wireless networks. *IEEE Wireless Communications, Special Issue on Dependability Issues with Ubiquitous Wireless Access*, 15(5):10–18, October 2008.

[SFPY07] J. Sun, C. Faloutsos, S. Papadimitriou, and P. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *Proc. ACM SIGKDD Intl. conference on Knowledge discovery and data mining*, KDD '07, 2007.

[Sig] Signpost. `http://en.wikipedia.org/wiki/Wikipedia:Wikipedia_Signpost`.

[SSK07] Izchak Sharfman, Assaf Schuster, and Daniel Keren. A geometric approach to monitoring threshold functions over distributed data streams. *ACM Trans. Database Syst.*, 32(4), 2007.

[Sta] Wikipedia Stats. `http://stats.wikimedia.org/EN/TablesArticlesTotal.htm`.

[Ste] `http://steganrtp.sourceforge.net/`.

[SW73] D. Slepian and J. K. Wolf. Noiseless coding of correlated information sources. *IEEE Trans. Inf. Theory*, 19(4):471–480, 1973.

[TMV+11] Keren Tan, Chris McDonald, Bennet Vance, Chrisil Arackaparambil, Jihwang Yeo, and David Kotz. From MAP to DIST: the evolution of a large-scale WLAN monitoring system. *In submission*, 2011.

[Tsa88] Constantino Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *J. Stat. Phys.*, 52:479–487, 1988.

[VWD04] F. Viégas, M. Wattenberg, and K. Dave. Studying cooperation and conflict between authors with history flow visualizations. In *Proc. of the SIGCHI conference on Human factors in computing systems*, CHI '04, 2004.

[Wik10] In *1st Intl. Competition on Wikipedia Vandalism Detection, part of 4th Intl. Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse PAN-10*, 2010.

[Woo07] David P. Woodruff. *Efficient and Private Distance Approximation in the Communication and Streaming Models*. PhD thesis, MIT, 2007.

[WP05] Amo Wagner and Bernhard Plattner. Entropy based worm and anomaly detection in fast IP networks. In *Proc. 14th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WET ICE)*, 2005.

[XlZB05] Kuai Xu, Zhi li Zhang, and Supratik Bhattacharyya. Profiling internet backbone traffic: Behavior models and applications. In *In ACM Sigcomm*, pages 169–180, 2005.

[YEG09] Guanhua Yan, Stephan Eidenbenz, and Emanuele Galli. SMS-Watchdog: Profiling social behaviors of SMS users for anomaly detection. In *Proceedings of the 12th Intl. Symposium on Recent Advances in Intrusion Detection*, 2009.

[ZCFM06] Sheng Zhang, Amit Chakrabarti, James Ford, and Fillia Makedon. Attack detection in time series for recommender systems. In *Proc. of ACM SIGKDD KDD'06*, pages 809–814, 2006.