Creating Models Of Internet Background Traffic Suitable For Use In Evaluating Network Intrusion Detection Systems

Song Luo

University of Central Florida

CREATING MODELS OF INTERNET BACKGROUND TRAFFIC SUITABLE FOR USE IN EVALUATING NETWORK INTRUSION DETECTION SYSTEMS

by

SONG LUO
B.E. North China Institute of Electrical Power, 1995
M.S. University of Central Florida, 2002

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the School of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Major Advisor: Gerald A. Marin

Fall Term
2005

ABSTRACT

This dissertation addresses Internet background traffic generation and network intrusion detection. It is organized in two parts. Part one introduces a method to model realistic Internet background traffic and demonstrates how the models are used both in a simulation environment and in a lab environment. Part two introduces two different NID (Network Intrusion Detection) techniques and evaluates them using the modeled background traffic.

To demonstrate the approach we modeled five major application layer protocols: HTTP, FTP, SSH, SMTP and POP3. The model of each protocol includes an empirical probability distribution plus estimates of application-specific parameters. Due to the complexity of the traffic, hybrid distributions (called mixture distributions) were sometimes required. The traffic models are demonstrated in two environments: NS-2 (a simulator) and HONEST (a lab environment). The simulation results are compared against the original captured data sets. Users of HONEST have the option of adding network attacks to the background.

The dissertation also introduces two new template-based techniques for network intrusion detection. One is based on a template of autocorrelations of the investigated traffic, while the other uses a template of correlation integrals. Detection experiments have been performed on real traffic and attacks; the results show that the two techniques can achieve high detection probability and low false alarm in certain instances.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Gerald A. Marin for his guidance, encouragement and support. I am very fortunate to have had the chance to work with him during my lengthy Ph.D. studies.

I thank my family, my parents and my wife Lin, for their endless support of my doctoral studies. No matter what challenges I have to face, I know they are behind me.

My research also benefited from interactions with many others. I would especially like to thank Dr. William Allen and Dr. David Nickerson for many illuminating discussions and much helpful advice.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: OVERVIEW AND MOTIVATION

Although network traffic engineers have long worked to model and predict traffic flow, the need to test Intrusion Detection Systems (IDS) in realistic environments makes modeling and simulating today's Internet traffic more critical. However, simulating real Internet traffic is never an easy task. The heterogeneity of the physical infrastructure, the mix of network and application layer protocols, the rapid rate of change, and the problems of scale combine to create a daunting modeling challenge.

Realistic simulation (and laboratory emulation) of network traffic is especially critical when developing and testing an IDS. As argued in [88], one cannot simply replay existing real traffic traces to test an IDS, because real traffic may contain attacks and other abnormalities that obscure IDS performance. A better approach is to provide completely clean or attack-free synthetic background traffic and integrate it with prepared attacks (known or unknown to the IDS). The synthetic traffic can be used as a part of test bed for IDS evaluation - either in a laboratory environment or a simulation environment.

Intrusion detection systems are either based on analyses of attack signatures or, for previously unseen attacks, on some type of anomaly detection. Often anomaly detection depends on statistical properties of the network traffic such as arrival rates, traffic volumes, source and destination distributions, port distributions, and latency changes. Thus, for anomaly detection it is particularly important that traffic be modeled accurately with regard to such characteristics. For example, previous work has shown that real Internet traffic is self-similar, and the intrusion detection technique developed in [88] uses loss of self-similarity as an indication that an attack may be present. To test such detection techniques we must create modeled traffic that includes subtle distributional characteristics. This is an important goal of the present work.

Of course Internet traffic simulation is used in many other fields, such as developing new protocols, network congestion control, and traffic engineering, and much more. The modeling and traffic generation work described here is applicable to these and other fields; however, in this dissertation the focus is on how to provide a realistic test bed for IDS testing and evaluation.

As the world relies more on electronic information and interconnected networks, the security risks to people and to businesses increase. Evidence shows that the recent increase in focus on computer security from government, research and industry has not stemmed the accelerating trend of computer security incidents. Before 1998, there were fewer than 5000 computer security incidents each year; in 2003, this number jumped to 137,529, a significant 68% increase over 2002's number of 82,094 [14]. It is thought that the situation became worse in 2004 because of widespread use of automated attacking tools.

The above statistics suggests that providing security protection to computer systems is a critical task, especially in an inter-networking environment. The research into intrusion detection techniques, in particular, has been substantial in recent years, but today's IDSs are still experiencing fatal problems including high false-alarm rates, configuration difficulty, and bandwidth limitations.

Li [54] proposed a theoretically robust approach for detecting network intrusions (DoS attacks) with a high detection probability and a low false-alarm rate. The approach calculates the autocorrelation of the examined traffic, and the difference between the autocorrelation and a predefined traffic template is used to judge if there is an attack in the traffic. Allen [86] proposed an intrusion detection technique based on comparing fractal dimensions of the examined traffic to a traffic template.

Both of the above two techniques require a reliable template of “normal” traffic. How to estimate such a template is left as an open question in both Li and Allen’s work. Therefore, in this thesis, we present techniques for building useful traffic templates that are based, in turn, on the modeling techniques also developed here.

CHAPTER 2: LITERATURE REVIEW

2.1 Early work on Telecommunication and Network Simulation

Molina [46] reports that back in 1898, G. T. Blood of the AT&T Co. had done simulation experiments, and found a close agreement between the terms of a binomial expansion and the results of the observations on the distribution of busy telephone calls. That was probably the earliest work on simulating traffic in a telecommunication system.

Communications between servers and users (or terminals) have been studied since the rise of time-sharing systems in the 1960's. E. Fuchs and P.E. Jackson [26] developed a data stream model from a study of multi-access computer communications [64]. The model was basically composed of two kinds of segments: the user burst segment and the computer burst segment. During a user burst segment, the user is sending characters to the computer; during a computer burst segment, the user is receiving characters from the computer. This early work assumed half-duplex operation, but with minor modification, applies to full-duplex operation. Within a given burst segment, there are periods of line activity and of line inactivity. They are called think time or idle time, depending on whether the previous segment is a computer burst segment or a user burst segment. Independent random variables are introduced to describe the elements in the model, such as the number of burst segments per call, think time, user inter-burst time, etc. Fuchs and Jackson inferred distributions of each random variable and tested them using the goodness-of-fit method.

In the following years, computer communication and computer network traffic were modeled based on the concept of burst segments. And it was widely considered appropriate to model packet traffic using pure Poisson or Poisson-related processes, such as Poisson-batch or Markov-Modulated Poisson processes [33], packet-train models [71], fluid flow models [17], etc. However, more and more evidence has been found that for both local area and wide area network traffic the distribution of packet interarrivals clearly differs from the exponential [70], [32], [63]. Leland [92] argues convincingly that LAN traffic is much better modeled using statistically self-similar processes, which have remarkably different theoretical properties than Poisson processes. For self-similar traffic, there is no natural length for a burst, and traffic bursts appear on a wide range of time scales. Furthermore, Paxson in [85] pointed out that for wide area traffic Poisson processes are valid only for modeling the arrival of user sessions, such as TELNET connections and FTP control connections, and that they fail as accurate models for other WAN arrival processes.

2.2 Introduction to Self-semiliarity

According to Beran [39], a stochastic process can be called self-similar (with Hurst parameter H) if the rescaled process, with an appropriate rescaling factor, is equal in distribution to the original process. Leland et al. [92] studied the self-similar behavior in Ethernet traffic and described several methods for determining the Hurst parameter.

Let X={X_t: t=0, 1, 2, …} be a wide-sense stationary stochastic process with autocorrelation function r(k), k≥0. Let {$X_k^{(m)}$: k=1,2,3,…} represent a family of aggregate processes produced by summing the original time series, X, over adjacent, non-overlapping blocks of size m. In particular, for integer m≥1, $X^{(m)}$ = {$X_k^{(m)}$, k=1,2,3,…} where $X_k^{(m)} = \frac{1}{m}\sum_{i=km-(m-1)}^{km} X_i$. Generally stated, if the distribution of each of the aggregate processes $X^{(m)}$, m>1 is approximately the same as that of the original process, X=$X^{(1)}$, then X is self-similar.

Self-similar processes are classified into 2 models: One is an exactly self-similar model and the other an asymptotically self-similar model [39]. A process X is called exactly second-order self-similar with parameter H $\in$ (0.5, 1), if its autocorrelation function is

$$r(k) = \frac{1}{2}[(k+1)^{2H} - 2k^{2H} + (k-1)^{2H}]\}, k \in I. \qquad (2\text{-}1)$$

A process is called asymptotically second-order self-similar with parameter H$\in$(0.5, 1), if its autocorrelation function is with the form

$$r(k) \sim ck^{2H-2}, k \to \infty, \qquad (2\text{-}1)$$

where c>0 is a constant.

In actual applications, an exactly self-similar model is too narrow to model real traffic, thus in most cases, an asymptotically self-similar model is used.

Asymptotic self-similarity is indicated by several properties that can be used to test for its presence [92]:

- The variance of the sample mean decreases more slowly than the reciprocal of the increase in sample size m, thus $\mathrm{var}(X^{(m)}) \sim cm^{-\beta}$, as $m \rightarrow \infty$, where c is some positive constant and $0<\beta<1$.
- The hyperbolic rate of decay of the autocorrelations, proportional to $k^{-\beta}$, indicates that $\sum_k r(k)=\infty$; by definition the process is long-range dependent.
- The spectral density, $f(\lambda)=\sum^{\infty}_{k=-\infty} r(k)e^{-ik\lambda}$, of the process obeys a power law near the origin, $f(\lambda)\sim I(\lambda)=c\lambda^{-\gamma}$ as $\lambda \rightarrow 0$, where $0<\gamma<1$, $\gamma=1-\beta$ and c is some positive constant.

Hlavacs al et. [34] summarized methods to test self-similarity and the Hurst parameter. The methods include Variance-Time plot, R/S plot, Periodogram, Whittle estimator, correlogram plot and Wavelet estimator.

Self-similarity has a strong influence on the resulting traffic. Traffic with self-similarity usually has features including long-range dependence, heavy-tailed distributions with infinite variance, and traffic bursts.

2.3 Related Work on Network Traffic Modeling & Generation

After the self-similarity or long-range dependence feature of network traffic was revealed, many solutions and techniques have been proposed to better model and simulate network traffic for traffic generation purposes. Here we introduce some of the important research done in this area.

2.3.1 Tcplib

Danzig [61] describes a library, Tcplib, to help generate realistic TCP/IP network traffic. Tcplib models 5 different types of Internet application traffic (FTP, SMTP, NNTP, TELNET and RLOGIN) based on Internet traffic traces collected from UC Berkeley, University of Southern California, and Bell Communication Research. By using routines provided by the library, users can generate network traffic that is statistically similar with the real traffic traces collected.

Several limitations exist in Tcplib. First, it needs a better model of conversation arrival rates. Second, it lacks several application-specific details. For example, the interarrival time of FTP control packets and the distribution of number of request-response handshakes that occur during SMTP and NNTP conversations were not modeled. Also, because this work preceded the growth of the web, Tcplib does not include a model of HTTP traffic, which is critical for today's network traffic simulation.

2.3.2 Paxon's Models

Perhaps the most frequently referenced models of TCP connection distributions are those from Paxson [85], [83]. Paxson and his colleagues examined 3 million TCP connections from a number of wide-area traffic traces and a variety of sources. Some analytical models were derived to describe the random variables associated with TELNET, NNTP, SMTP and FTP connections. In [83], it was found that, except for user-initiated TCP session arrivals, other TCP connection arrivals are not Poisson. The following table summarizes the models used in his work to describe the traffic characteristics of different services.

Table 1: Summary of Analytical Models from Paxson's Paper

Protocol	Variables	Model
TELNET	Session arrivals	Poisson
	Originator bytes	Log2-extrem
	Responder bytes	Log2-normal, 80-100%
	Connection size(in packets)	Log2-normal
	Packet interarrivals	Empirical, from Tcplib
SMTP	Session arrivals	Poisson
	Originator bytes	Log2-normal
FTP	Session arrivals	Poisson
	Connection bytes	Log2-normal
	FTP-data spacing	Log-normal/log-logistic
	Session bytes	Log2-normal
NNTP	Originator bytes	Log2-normal

Notably missing from Table 1 (and from the referenced work) is any reference to HTTP.

2.3.3 WWW Traffic Model

Because web traffic has dominated the Internet since the middle of the 1990's, modeling this traffic has become critical to any simulation of real Internet traffic. We present several major efforts made in this area.

2.3.3.1 Empirical Model

Mah in [8] constructed an empirical model of HTTP traffic. The model consists of a number of probability distributions determined by analysis of TCP traces of actual HTTP conversations. The analysis shows that HTTP requests exhibit a bimodal distribution, and that sizes of HTTP replies have a heavy-tailed distribution (to be discussed more fully). Some other components are also discussed, such as document size (number of files per document) and user think time. Although the measurement of this work is basically consistent with measurements of other studies, this empirical model does not deal with the case of persistent-connection HTTP, where new measurement and analysis methodologies may be required. The self-similar feature discussed in [52] [45] is not adequately handled. Finally, the empirical distributions instead of closed-form analytical expressions inhibit the model's applicability in a variety of environments.

Table 2: Random variables of page oriented HTTP model

Random Variable	**Distribution**
Session Interarrival Time	Exponential
Pages per session	Lognormal
Time between pages	Gamma
Page size	Pareto
Packet size	Multimodal
Packet interarrivals time	Exponential

Another early empirical measurement of HTTP traffic was done by Paul Barford and Mark Crovella in [62]. Today's web traffic generators are usually based on results from one or both of these pioneering measurement projects.

2.3.3.2 Page Oriented Model

Reyes presented a structural, multilevel model for WWW traffic in [6]. The model is based on HTTP traces and is designed to simulate application-level traffic. Reyes analyzed the HTTP traffic in three levels: session level, page level, and packet level. On each level, several random variables are derived to describe the traffic. Table 2 summarizes those random variables and his findings about their distributions.

2.3.3.3 New features of today's HTTP traffic

By examining over 500GB of TCP/IP protocol header traces in [28], Smith et al. draw a number of interesting conclusions about contemporary Web traffic's evolving nature. This study confirms that persistent HTTP connections play a major role in transferring web objects.

Although only 15% of HTTP connections are persistent, they represent 40-50% of all the web objects requested, and their use results in a 50% reduction in the total number of TCP connections required to deliver web content. Other interesting observations include: 65% of all web pages are constructed entirely by responses from a single server; close to 70% the consecutive top-level page references go to an IP address that is different from the address used for the previous top-level page reference, possibly representing the impact of large organizations managing their web sites with a "server farm" for load balancing; an increase in the number of embedded objects per web page but a decrease in frequently occurring sizes, possibly due to the pervasive use of "banner adds" and icons to decorate pages.

Casilari [25] studied HTTP traces by alternatively enabling the two versions of HTTP protocols (HTTP 1.0 and HTTP 1.1), and found that although HTTP 1.1 prefers persistent connections to retrieve multiple objects from remote server for one page, HTTP 1.1 traffic exhibits the same statistical properties as HTTP 1.0. In this sense, the heavy-tailed nature of the size of HTTP connections is proved to be an invariant. Casilari also reported that longer traces should be collected to analyze the impact of other improvement introduced by HTTP 1.1.

2.3.3.4 Recent HTTP traffic modeling

Based on Riedi's [74] work, Balamash [2] applied multi-fractal analysis in modeling WWW traffic on the server side. The presented model captures the essential characteristics of WWW traffic, including the temporal and spatial localities as well as the popularity profile. The main advantage of this model is that it simultaneously captures the marginal distribution and ACF (Autocorrelation Function) of the traffic.

Li [53] presented a form of autocorrelation function for asymptotically self-similar processes and verified that this form satisfactorily fits the real traffic data on investigated Ethernet. His work made the effort to solve the problem that, although we know the general shape of the autocorrelation function of an asymptotical self-similar process, we still don't know its closed form with fixed finite lags. In [55], Li used his general form of autocorrelation function to fit real WWW traffic, and found that, with minor modification, the function can describe the real data to a satisfactory degree.

2.3.4 Self-Similar Traffic Models

Self-similarity has been widely observed for modern network traffic. To simulate today's network traffic, we cannot ignore self-similarity and related heavy-tailed properties. Actually, much research has been done to reproduce self-similarity in simulated networking traffic, and a number of relevant models have been proposed. A survey of models of self-similar stochastic processes is given in [34].

This section briefly introduces the major models used to simulate self-similar network traffic.

2.3.4.1 Fractional Brownian Motion (FBM)

Define a zero mean Gaussian Process $B_H(t)$ with Hurst parameter H as follows:

(1) $E[B_H(t)]=0$

(2) $B_H(0)=0$

(3) $B_H(t+\delta) - B_H(t)$ is normally distributed $N(0, \sigma|\delta|^H)$

(4) $B_H(t)$ has independent increments

(5) $E[B_H(t)\ B_H(s)]= \sigma^2/2(|t|^{2H}+|s|^{2H}-|t-s|^{2H})$

$B_H(t)$ is exactly self-similar, perfectly determined by H.

FBM can be used to model the sum or integral of self-similar traffic (as observed in network buffers, file sizes of audio/video streams, etc). Its increments/derivatives yield self-similar fractional Gaussian Noise.

2.3.4.2 Fractional Gaussian Noise (FGN)

The increments of FBM are known as Fractional Gaussian Noise (FGN) [10] and form a stationary process $G_H(t)$ with the following properties:

(1) $G_H(t) = (B_H(t+\delta) - B_H(t))/ \delta$

(2) $G_H(t)$ is normally distributed $N(0, \sigma|\delta|^{H-1})$

(3) $E[G_H(t+\tau)\, G_H(t)] = \sigma^2 H(2H-1)|\,\tau|^{2H-2}$ for $\tau >> \delta$

Discrete time FGN also has the following autocorrelation function ([39]):

$$\rho_X(k) = \frac{1}{2}(|k+1|^{2H} - 2|k|^{2H} + |k-1|^{2H}), k \geq 1. \tag{2-3}$$

In [84] an algorithm is given to create estimated discrete-time FGN. The algorithm first generates an estimate of the power series $f(\lambda, H)$ of the desired traffic stream at the discrete frequencies $\lambda_j = 2\pi j/n$, $j=1,\ldots, n/2$. Here, only the Hurst parameter H is necessary. After some transformations, a sequence of n complex numbers is obtained, which is transformed back via the inverse Fourier transformation to obtain a sequence $\{x_k\}^n_{k=1}$. In [84], an instance of the algorithm is given explicitly and programmed in the statistics language S.

FGN is exactly second-order self-similar traffic and is a good candidate for modeling the traffic of Ethernet, ATM, VBR coded video, Web Telnet and FTP instances.

2.3.4.3 ARFIMA

Fractional ARIMA models (ARFIMA or FARIMA) ([92], [39]) are built on classical ARIMA models. $\{X_n\}^{\infty}_{n=0}$ is called an ARFIMA(p,d,q) process, if $\{\Delta^d X_n\}^{\infty}_{n=0}$ is an ARMA(p,q) process for some non-integer d>0. Let B be the Backshift-operator $B(X_n)=X_{n-1}$ and Δ^d can be represented by

$$\Delta^d = (1-B)^d = \sum_{u=0}^{\infty} \pi_u B^u \tag{2-4}$$

with $\pi_0=0$ and

$$\pi_u = \frac{\Gamma(u-d)}{\Gamma(u+1)\Gamma(-d)} = \prod_{k=1}^{\infty} \frac{k-1-d}{k}, u = 1,2,... \tag{2-5}$$

ARFIMA processes are asymptotically self-similar, with Hurst parameter H=d+0.5, 0<d<0.5. For large lags, the correlations of an ARFIMA(p,d,q) process are similar to those of an ARFIMA(0,d,0) with the same d.

ARFIMA models are similar to FGN, yet they are very flexible due to the natural correspondence to ARIMA(p,d,q) models and to their higher number of parameters.

2.3.4.4 Wavelets

The above described stochastic models try to capture short- and long-term dependencies as observed in VBR video or Ethernet traffic. Wavelets provide a means of transforming the original self-similar process into a new process with much less self-similar behavior. For this new process, simpler models can be applied. Traffic is then generated first in the wavelet domain, and then transformed back into the time domain by applying the inverse wavelet transformation ([65], [58]).

Wavelets are capable of capturing both short-range and long-range dependencies. They are thus well suited for modeling Ethernet, ATM, VBR, Telnet and Web traffic.

2.3.4.5 On/Off Processes

A large number of superimposed heavy-tailed On/Off processes ([51]) can yield self-similar traffic as well. An On/Off process is either in the On or Off state. One can construct a time series by observing the number of On-Processes at any time point. If On-times and Off-times are drawn from a heavy-tailed distribution like the Pareto distribution with parameters α_1 and α_2, then the observed stochastic process is self-similar fractional Gaussian noise with $H=(3-\min(\alpha_1, \alpha_2))$. On/Off processes can be used to create network traffic at the packet level, or streams of requests at a higher level.

2.3.4.6 Self-Similar Markov Modulated

In [77] self-similarity is simulated by using a Markov modulated discrete-time, discrete-state process. The proposed modulating Markov chain depends only on 3 parameters.

2.3.4.7 Multifractal Traffic

In [4], the multifractal nature of WAN traffic is demonstrated. In contrast to monofractal (self-similar) traffic, where the local scaling behavior is constant, multifractal traffic takes into account the changing local scaling behavior over time. This local scaling behavior is measured as the rate at which the number of bytes/packets observed in the interval $[t_0, t_0+\delta t]$ tends to zero as $\delta t \rightarrow 0$. In [4] this local scaling behavior is calculated by using wavelet transforms. The multifractal property is then motivated by the cascading nature of WAN traffic (each trace consists of sessions, each session consists of traffic requests, each traffic request consists of TCP connections, each TCP connection consists of IP packets, …).

CHAPTER 3: MODELING INTERNET BACKGROUND TRAFFIC USING MIXTURE DISTRIBUTIONS

Ideally, to evaluate an Intrusion Detection System (IDS) one needs realistic traffic that contains only known attacks. (The nature and location of the attacks should be known by testers; attacks should not all be known by IDS developers.) Furthermore, the traffic needs to be "live" so that one can better judge the system's ability to protect targeted test-bed systems. In addition, the non-attack traffic ideally should reflect the characteristics of typical traffic of the defended network because today's tools generally need to be tuned to a particular environment in order to mitigate false alarms.

In pursuit of such an environment, we have been analyzing the characteristics of the CS LAN traffic at UCF to identify the major contributors to that traffic, to model the corresponding statistical distributions, and to measure key distribution parameters. The intent is to model a majority of the traffic threads present in the original LAN traffic and either a) emulate the modeled sources of traffic while targeting them towards servers that are present in the lab environment, or b) simulate the software and targeted servers using the NS-2 simulator.

3.1 LAN Traffic Analysis

The traffic traces we captured from the Computer Science department at UCF are TCP headers from millions of Ethernet frames during a two-day period in February 2003. The CS department uses Network File System (NFS) and an auxiliary server to reduce the load on its main email and file server. Thus, all email, FTP, and web traffic that is addressed to the main server is automatically redirected to port 2049 of the NFS server. Once we understood this, we eliminated the duplicate traffic on port 2049 from further consideration. The CS LAN also supports significant printer sharing with the network printer service at port 9100. We also eliminated this traffic prior to modeling. Table 3 gives an overview of the remaining traffic statistics from the two days.

Table 4 lists the 10 most heavily loaded machines (bytes received) on the CS LAN along with the number of bytes received and the percent of total bytes. This type of information is important if one is trying to reproduce a particular organization's traffic patterns or is concerned with detecting anomalies based on existing destination patterns. (It is interesting to note that only 10 machines account for more than 90% of the CS LAN traffic by destination.) In later chapters, we report our discoveries on the key features of CS LAN traffic and the results obtained by building analytical distribution models for FTP, HTTP, SMTP, POP3 and SSH protocols.

Table 3: TCP connections on Day 1 and Day 2

Day 1 - Feb. 05			Day 2 - Feb. 07		
PKTS	32,070,078		PKTS	20,162,983	
	IP	32,070,078		IP	20,162,983
	ICMP	29,197		ICMP	26,694
	UDP	1,740,446		UDP	1,631,138
	TCP	30,300,214		TCP	18,504,809
BYTES	19,857,363,053		BYTES	13,295,981,557	
	ICMP	2,994,825		ICMP	2,658,906
	UDP	772,239,405		UDP	657,411,561
	TCP	19,082,115,557		TCP	12,635,890,568
TCPCONN	212,716		TCPCONN	170,550	
	HTTP	92,034		HTTP	78,581
	SMTP	5,520		SMTP	5,214
	FTPDATA	2,236		FTPDATA	3,817
	TELNET	39		TELNET	37
	FINGER	8		FINGER	10
	FTP	435		FTP	466
	POP3	13,586		POP3	12,018
	TIME	2		TIME	2
	SSH	833		SSH	690
	IRC	0		IRC	0
	IDENT	315		IDENT	246

Table 4: The top 10 destined IP addresses in the CS LAN

IP Address	Number of Bytes	Proportion
132.170.108.140	802,439,489	26.30%
132.170.108.162	657,245,632	21.54%
132.170.108.1	512,159,566	16.79%
132.170.108.39	325,281,391	10.66%
132.170.107.219	258,709,111	8.48%
132.170.108.2	187,601,525	6.15%
132.170.107.188	65,045,048	2.13%
132.170.107.125	37,364,870	1.22%
132.170.107.132	36,940,338	1.21%
132.170.108.183	31,208,124	1.02%

3.2 Heavy-tailed Feature and Usage of Hybrid Distributions

In the usual way, we denote the cumulative distribution function (cdf) of a random variable X as

$$F_X(x) = P[X \le x] \tag{3-1}$$

and its associated probability density function (pdf) as

$$f_X(x) = F_X^{'}(x) \tag{3-2}$$

when this derivative exists. The distribution of a random variable X is said to be heavy-tailed if

$$1 - F_X(x) = P[X > x] \sim x^{-\alpha}, \text{ as } x \to \infty,\ 0 < \alpha < 2. \tag{3-3}$$

Heavy-tailed distributions have a number of properties that are qualitatively different from distributions more commonly used, such as Poisson, normal or exponential distributions [52]. As parameter α decreases, an arbitrarily large portion of the probability mass may be present in the tail of the distribution. In other words, a random variable that follows a heavy-tailed distribution can give rise to extremely large values with non-negligible probability.

To assess the presence of heavy tails in our data, we can employ log-log complementary distribution (LLCD) plots. These are plots of the complementary cumulative distribution $\overline{F}(x) = 1 - F(x) = P[X > x]$ on log-log axes. Plotted this way, heavy-tailed distributions have the property that

$$\frac{d \log \overline{F}(x)}{d \log x} = -\alpha \text{ , x>}\theta \tag{3-4}$$

for some real threshold θ and the shape parameter α>0.

To check for the presence of a heavy tail, we form the LLCD plot, and look for approximate linear behavior over a significant range in the tail.

An LLCD plot is also used to estimate the parameters θ and α. Once the existence of linear behavior is confirmed, parameter α can be estimated by a linear regression. The value of θ is the estimated point on X-axis beyond which the plot appears to be linear.

The most commonly used heavy-tailed distribution is the Pareto distribution with pdf given by

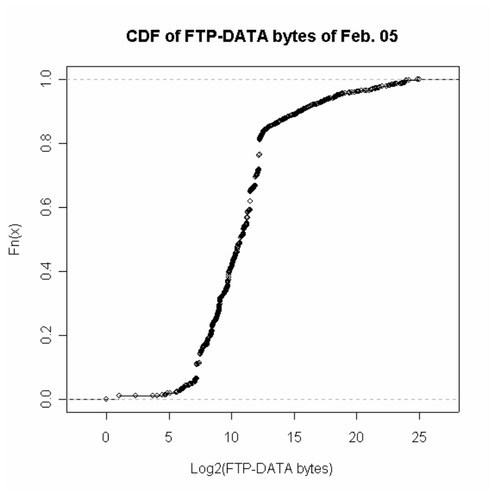
$$f(x) = \alpha k^{\alpha} x^{-\alpha-1} \text{, α>0, k>0.} \tag{3-5}$$

The corresponding cdf is

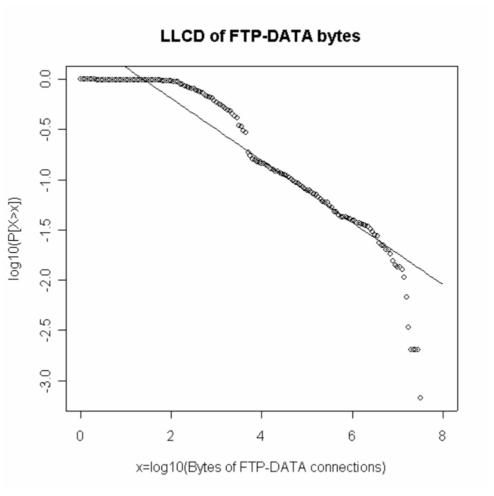
$$F(x) = P[X \le x] = 1 - (k/x)^{\alpha} . \tag{3-6}$$

Note that the parameter α can be measured by the slope of straight line behavior in LLCD plot; the parameter k can be estimated by estimating the points at which the data begin to show heavy-tailed behavior.

Having a heavy tail is a well-known feature of network traffic [92] [93], and we have indeed seen it in the CS data. For instance, the random number of bytes transmitted during an FTP-DATA connection is obviously heavy-tailed. The straight line behavior in its LLCD plot also confirms the existence of its heavy-tailed feature (Figure 1).



Cumulative Distribution Function plot of FTP-DATA bytes from trace of Feb. 05, 2003. It has an apparent upper heavy tail.



LLCD plot of FTP-DATA bytes from trace of Feb. 05, 2003. The straight line indicates existence of heavy tail.

Figure 1: CDF and LLCD plot of FTP-DATA bytes

We found more evidence of heavy-tailed behavior in other random variables, and we also recognize that it is not sufficient to model these random variables by using only one heavy-tailed distribution. Our analysis and experiments show that a mixed, or hybrid, approach often gives better performance. For example, for the FTP-DATA bytes, a hybrid model composed of Exponential and Pareto distributions fits very well (see Figure 2). Chi-square testing supports a finding of no significant difference between distribution of real trace data and the hybrid model. In the following sections, we will describe our work on modeling random variables for different Internet protocol traffic, and demonstrate in detail our approach of using mixed distributions.

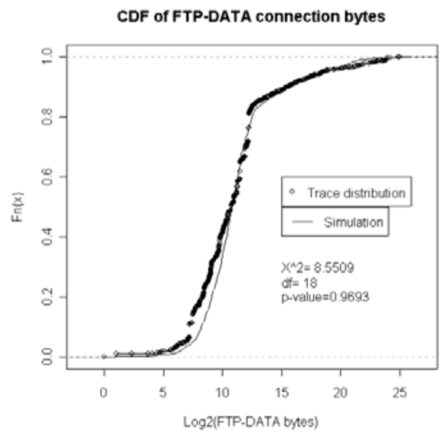


Figure 2: CDF of FTP-DATA bytes and its model

3.3 Modeling FTP traffic

Each FTP session includes an FTP control connection and either zero, one, or multiple FTP-DATA connections in "active" or "passive" mode, as described in [83]. In this section we are interested in modeling distributions of the following random variables:

A_{FTP}:	FTP session arrivals;
N_{FDC}:	The number of FTP-DATA connections per session;
B_{FDC}:	The number of bytes transferred during a single FTP-DATA connection;
I_{FDC}:	Idle-time between adjacent FTP-DATA connections.

One difficulty in identifying passive FTP-DATA traffic is that the associated port number is variable. Unlike transmission in active mode, which always uses the port 20 on the server, passive FTP transmissions might use any number above 1024 as the port number for the client or the server. The following 8 rules to distinguish valid FTP-DATA connections from the synthetic background traffic:

- Connections on port 20 are valid FTP-DATA connections
- Connections with no actual data payload are rejected
- That both sides of a connection send data is not allowed
- If a connection is a passive FTP-DATA, port numbers of both sides must be above 1024

- If a connection is a passive FTP-DATA, its parent session's client must initiate the FTP-DATA connection
- The time span of an FTP-DATA connection must be completely covered by its parent FTP session
- An FTP session's child data connections are not overlapped in time
- The port numbers of FTP-DATA connections spawned by an FTP session should be increasing, when the FTP-DATA connections are ordered by creation time.

3.3.1 FTP Session Arrival (A_{FTP})

According to Table 1, FTP session arrival is a typical user-initiated process and should be Poisson. Figure 4 depicts both the empirical cdf for FTP session arrivals and the predicted cdf for Poisson arrivals first for Day 1 (Feb. 05) and then for Day 2 (Feb. 07). Both plots confirm the prediction. We use time intervals of 5 minutes to count FTP session arrivals.

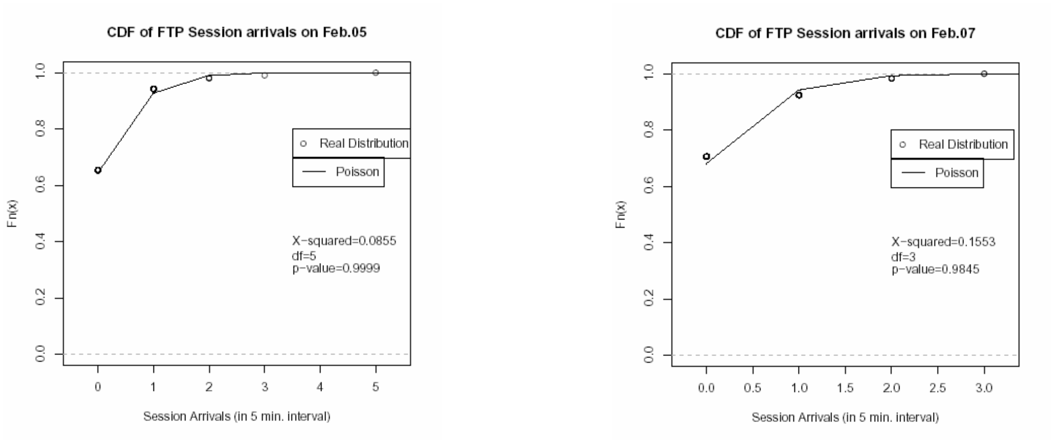
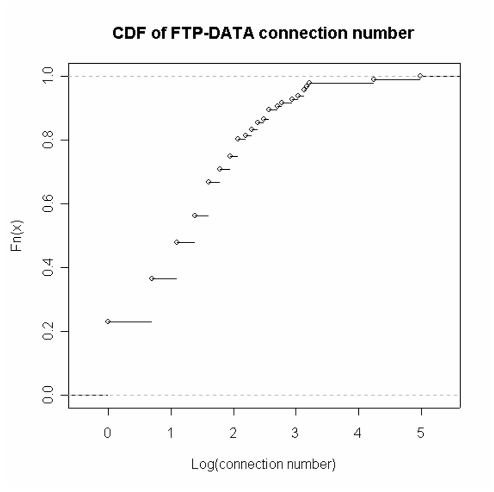


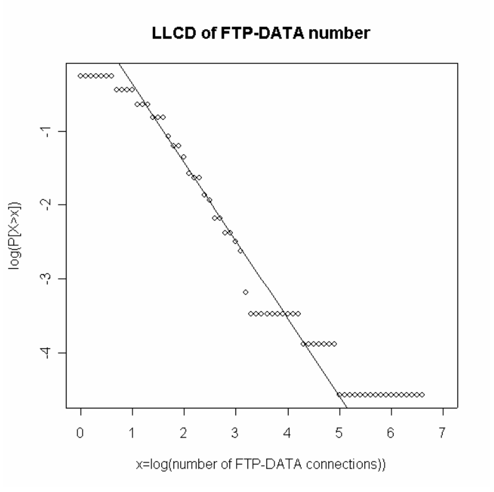
Figure 3: CDFs of A_{FTP} and the predition

3.3.2 Number of FTP-DATA Connection (N_{FDC})

The random variable N_{FDC} counts the number of FTP-DATA connections that are spawned by a single FTP session. The reason we model N_{FDC} instead of FTP session bytes (number of bytes transmitted during a session) is that, it is much easier to control session termination using the number of its FTP-DATA connections than using the total bytes transferred. Figure 4 shows that the distribution of N_{FDC} seems to be heavy-tailed, while the LLCD plot confirms our guess, and indicates that the threshold θ is near 1. The straight line in the LLCD plot has a slope of -1.0595, which suggests 1.0595 as the value of the α parameter of the corresponding Pareto model. The straight line begins at point x=1, therefore 3 (e^1~3) could be a good estimation for parameter k of the Pareto model.



CDF plot of N_{FDC} suggests a heavy tail may exist in its disdistribution



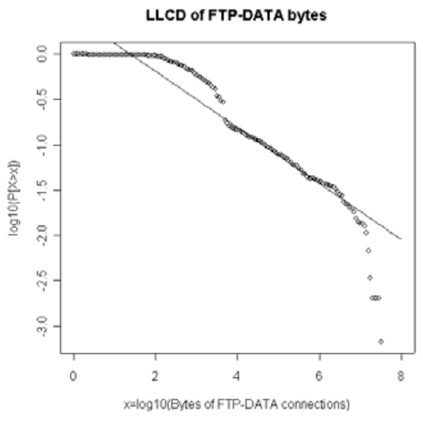
LLCD plot of N_{FDC} confirms the existence of heavy tail. The straight line in the picture also gives parameter estimations: α=1.0595, k=3

Figure 4: CDF and LLCD plot of N_{FDC}

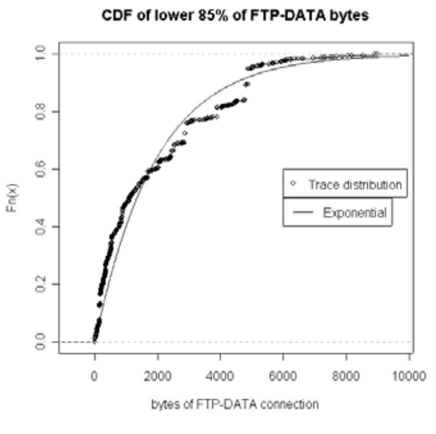
3.3.3 Bytes per FTP-DATA Connection (B_{FDC})

The random variable B_{FDC} represents the bytes transferred during a single FTP-DATA connection. Although an “active” FTP-DATA connection always connects to port 20 of FTP server, a “passive” FTP-DATA connection could be established on any server port number. We developed software to identify both “active” and “passive” FTP connections, and to extract them from synthetic traffic traces. Thus, there is a guarantee that our samples include both kinds of connections.

Extensive investigation indicates that the distribution of B_{FDC} does not match any known classic model. In this one the distribution does have a heavy tail, which is seen clearly in the LLCD plot, Figure 5.



B_{FDC} has heavy-tailed feature only on a range of its value domain



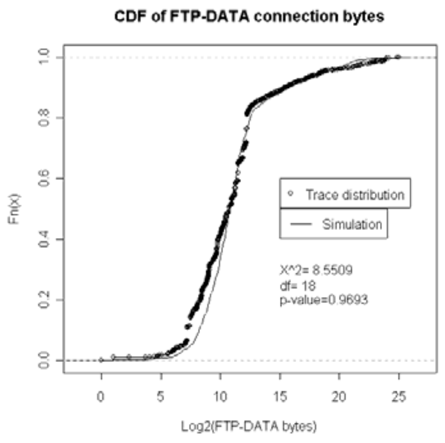
The lower 85% samples of B_{FDC} matche closely to Exponential distribution

Figure 5: B_{FDC} plots

From Figure 5, one can see the straight-line behavior begins roughly at x=4. The range corresponds to bytes number greater than 10000, which accounts for about 15% of all sample values. A good model for this part might be a Pareto distribution with parameter k=10000 and α=0.3090 (the negative slope of the straight line).

Now let's consider the lower 85% samples (or samples with a value less than 10000). CDF plot of this part suggests an Exponential distribution with the rate parameter 0.00052. The CDF plot of Figure 5 includes the fitted Exponential plot.

Having determined distribution models for the upper and lower part of the hybrid distribution sample, we combine these to build the final model for the random variable B_{FDC}. Figure 6 depicts the Chi-square test results.



The comparison between the sample distribution and the mixture model of B_{FDC}

Figure 6: The final model of B_{FDC}

3.3.4 Idle Time between FTP-DATA Connections (I_{FDC})

Basically, people use FTP in two manners. The first one is manual, in which the user types a command in an FTP client shell, waits for the server response, and then types the next command. In this way, the idle times between two consecutive data transfers are usually more than 1 second. Alternatively, people may use automatic FTP client software tools, such WS-FTP and SmartFTP. In these tools, users select all desired files before the actual transferring, and download/upload them in a batch mode. The file transferring is controlled by the software and in an automatic manner. The idle times between two continuous files are normally much less than 1 second.

Based on this observation, we divide our collected FTP-DATA idle times in two groups. One group contains only the times equal to or more than 1 second. We call it the manual group. The other group contains only idle times less than 1 second, and it is called the automated group.

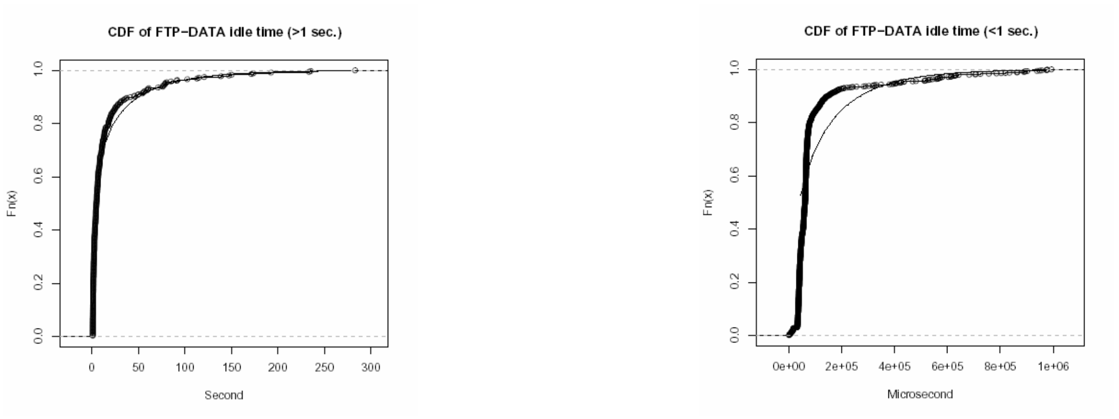


Figure 7: CDFs of I_{FDC}

Figure 7 depicts the cdf of I_{FDC} from both groups, compared with the expected Gamma distribution, for Day 1.

3.4 Modeling HTTP traffic

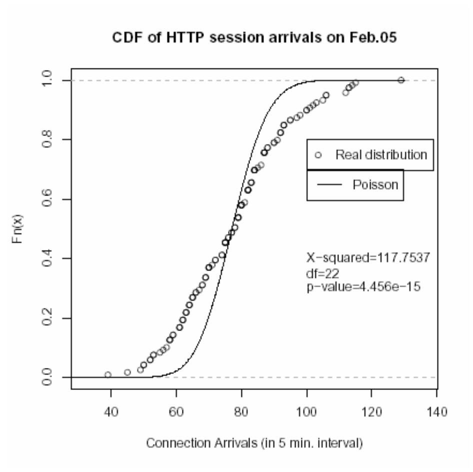
The currently used HTTP protocols include version 1.0 and version 1.1. In HTTP 1.0, if a requested page contains images or other special objects, separate HTTP connections will be initiated to get these objects, even though they reside on the server that hosts the page. In HTTP 1.1, only one TCP connection is needed to retrieve contents of a web page, provided that the contents are stored in the same server. In [25], analysis has been preformed on HTTP traffic generated separately by these two versions of HTTP. Results demonstrate that distributions of the two versions were not significantly different. Based on this conclusion, we do not distinguish HTTP 1.0 traffic from HTTP 1.1 traffic in our experiments and analysis.

It follows that no matter how large a page and how many connections it has, it is always appropriate to model only one TCP connection for this page. Thus we don't need to keep details of multiple connections, and the pages size (in byte) is calculated as the sum of the sizes of all connections associated with it. When we analyze and simulate the HTTP traffic, we need only include session level and page level. In this paper, 4 random variables are HTTP session arrivals, number of pages browsed during an HTTP session (session size), bytes transferred for one page (page size) and page spacing (idle time between pages).

We use the same structure to model HTTP traffic as that in [1] except that we do not need packet level. Another difference is how we determine both session and page generation times. When we come up with a new connection from a user, two time periods are measured. One is the connection spacing, which is the time between the start of new connection and the end of last connection from the same user. If this time exceeds 30 min., or 1800s, we consider the new TCP connection as the start of a new coming HTTP session. Another measurement is the difference of start times of two consecutive TCP connections. In [6], 30 seconds was the threshold to distinguish two pages. Thirty seconds may be appropriate for wireless networks (the work of [6] was done on analysis of wireless traffic), but for wired networks, it is too large. Our experiments suggest that 1 second is a good threshold for wired network.

3.4.1 HTTP Session Arrival (A_{HS})

One HTTP session is defined as continuous web browsing activities by a user. During a HTTP session, the user may open one or multiple web pages. These pages are separated by various lengths of thinking time. HTTP session arrival is supposed to satisfy Poisson distribution [8] [25], but the fit is greatly influenced by the size of the time interval in which we count HTTP session arrivals. Figure 8 depicts the cdf of HTTP session arrivals from the two day's traffic on 5-minute interval, while Figure 9 depicts cdf on 1-minute interval.



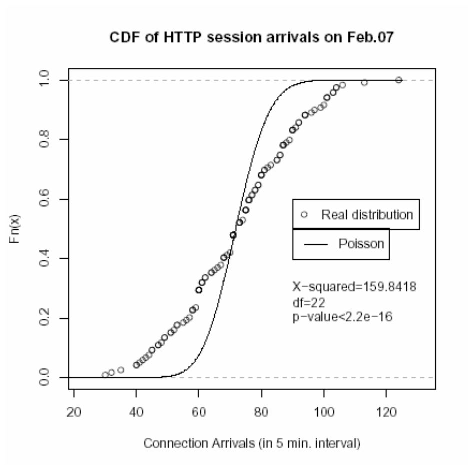
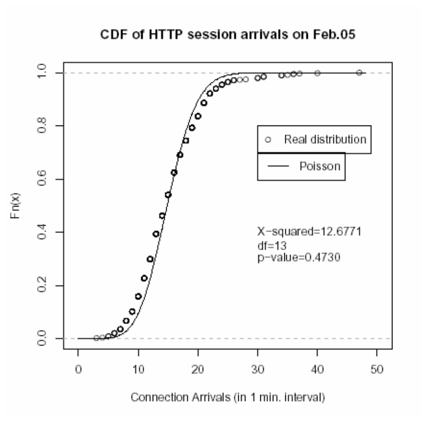


Figure 8: CDFs of HTTP session arrivals



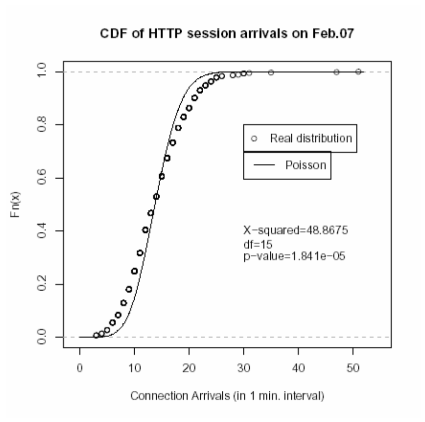


Figure 9: CDFs of A_{HS}

It is clear that the distribution with 1-minute interval matches Poisson much better. The implication is that when simulating HTTP session arrivals using Poisson process, it is critical to choose proper time intervals to achieve expected results.

3.4.2 Number of HTTP Pages per Session (N_{HPS})

Random variable N_{HPS} counts the number of web pages browsed in one user session. For the CS data sets, of both Feb. 05 and Feb. 07, N_{HPS} shows obvious heavy-tailed feature. In Figure 10, LLCD plot of this random variable for Feb. 05 has a linear pattern demonstrated by strait line Y=-0.08408-1.26188X. Thus, a Pareto distribution with α=1.26 and k=1 is appropriate.

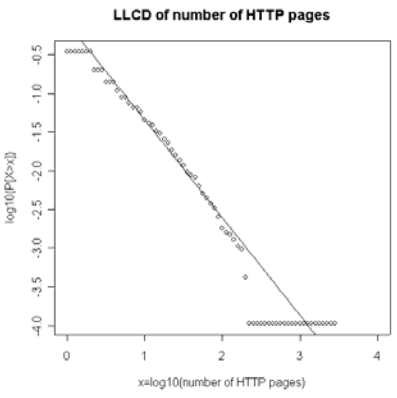


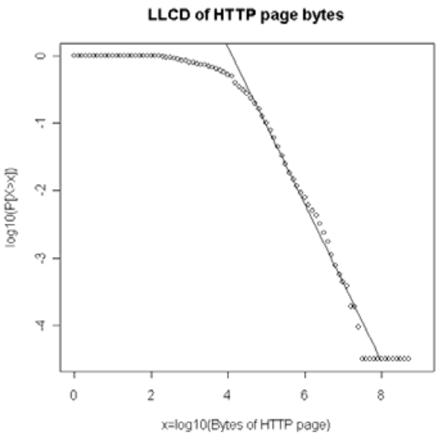
Figure 10: LLCD of random variable N_{HPS}

3.4.3 Bytes per HTTP Page (B_{HP})

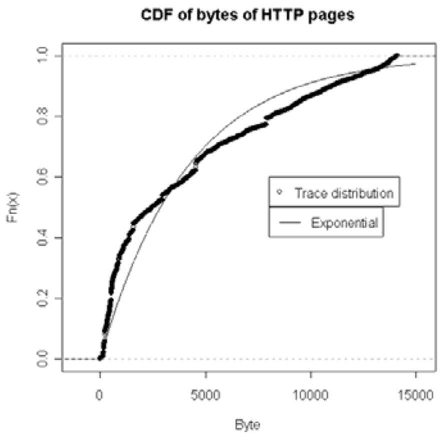
Random variable of B_{HP} measures the number of byte transferred by one web page. It includes bytes of HTML text, embedded pictures and all other objects in the page. Our analysis reveals that B_{HP} also has heavy tail in its distribution plot, see Figure 11.

LLCD shows the straight line behavior happens for byte number ranged from $10^{4.25}$ to $10^{7.5}$, with slope –1.164. This suggests a Pareto distribution (α=1.164, k=$10^{4.25}$) for http page bytes more than $10^{4.25}$ bytes (upper 36% on distribution plot).

For the lower 64% of distribution part, HTTP page byte number is more like an Exponential, with parameter rate 0.0002419939. The CDF plot depicts the trace distribution and the fitted Exponential model.



LLCD of B_{HP} shows the random variable has heavy tail in its upper 36% part of values



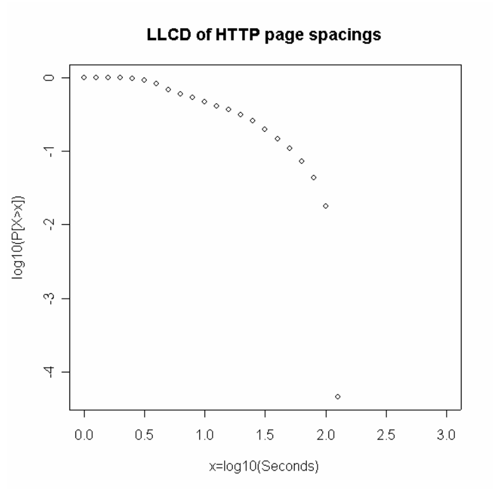
lower 64% part of values of B_{HP} has a distribution more like Exponential

Figure 11: LLCD and CDF of the lower part of B_{HP}

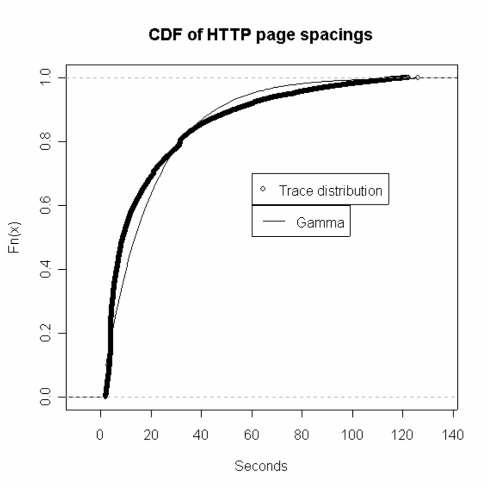
3.4.4 Time between HTTP Pages (T_{HP})

T_{HP}, which is related to user thinking time, is the time between two sequential “clicks”. It is the time between the beginnings of two consecutive web pages of a single user session.

The LLCD plot in this case shows that T_{HP} does not have a heavy-tailed distribution. We found the distribution is actually very close to a Gamma model (Figure 12). From the CS data, the shape and rate parameters of the Gamma model are 0.9936 and 0.0504, respectively.



LLCD of T_{HP} shows there's no heavv-tailed feature in its distribution



Distribution of T_{HP} can be closely fitted bv a Gamma model

Figure 12: LLCD and CDF of T_{HP}

3.5 Modeling SMTP Traffic

3.5.1 Arrivals of SMTP Connections (A_{SC})

There are no classic models found to describe SMTP traffic, except that documents from DARPA experiments [72] mentioned that a Poison process was used to simulate SMTP connection arrivals. Our analysis of the CS traffic confirms that SMTP session arrival is a Poisson distribution. Again however, the distribution with 1-minute intervals shows better match, compared with 5-minute intervals. See Figure 13.

3.5.2 Bytes per SMTP Connection (B_{SC})

Our experiments show even for the SMTP connection with one of the simplest email sent (i.e. the email contains no message in the body, no attachment, but only sender and receiver's addresses and a very short one-word subject), it has a data payload slightly more than 500 bytes in its TCP packets. We conclude that for each valid SMTP connection, it must have at least 500 bytes. We discard those SMTP connections with payloads less than 500 bytes, assuming they are generated by scans.

LLCD analysis indicates B_{SC} also has heavy-tailed feature, and a Pareto model with parameters α=0.8454 and k=1250 might be a good fit. Figure 14 plots the comparison between distributions of real traces and the simulation by Pareto model.

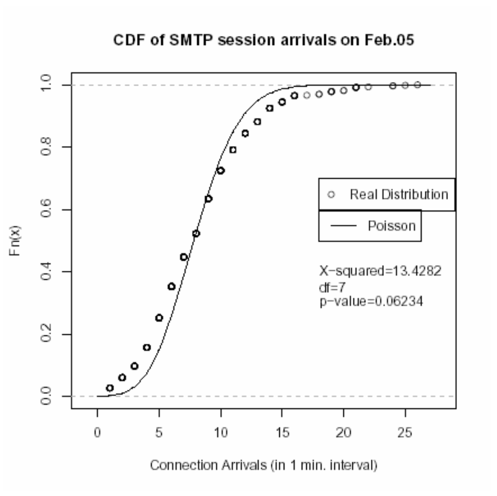


Figure 13: CDF of A_{SC}

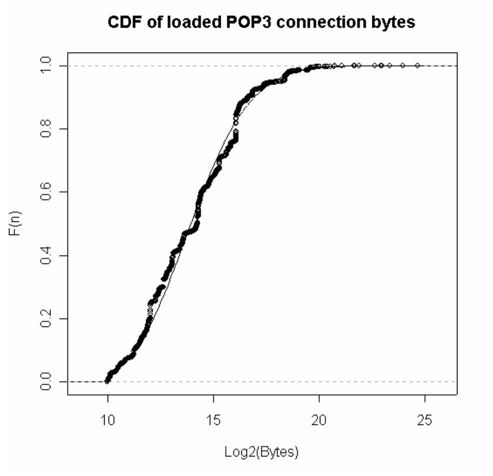


Figure 14: B_{LPC} matches a Gamma distribution

3.6 Modeling POP3 Traffic

3.6.1 Arrivals of POP3 Connections (A_{PC})

The arrival of POP3 connections is still a Poisson process. Again, this conclusion is based on observations of small time intervals, in this case, 1 minute. See Figure 15.

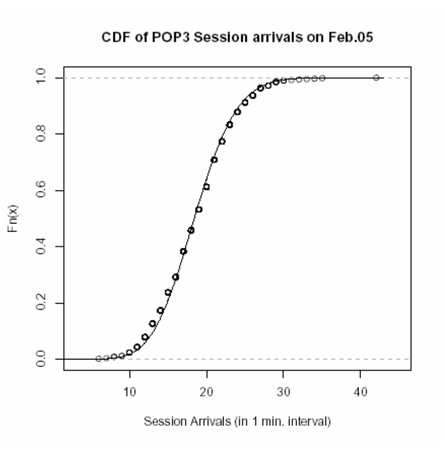


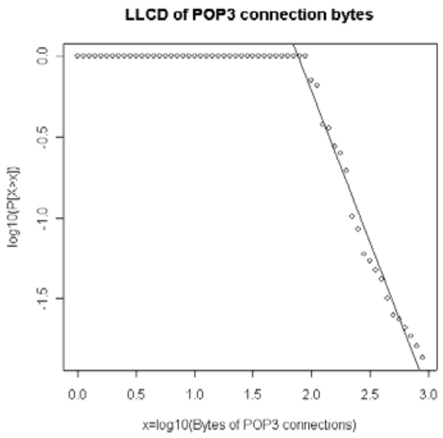
Figure 15: The arrival of POP3 connection

3.6.2 Bytes per POP3 Connection (B_{LPC} and B_{UPC})

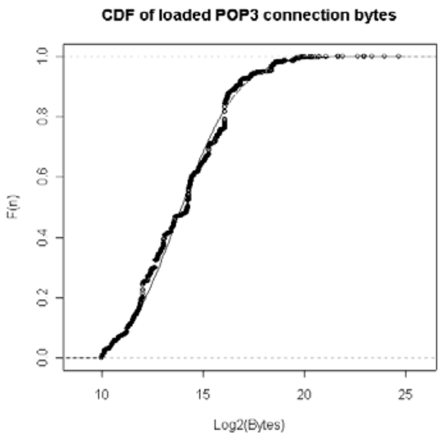
Experiments on POP3 connections show that, POP3 client programs issue several commands, such as LIST and UIDL. After a connection with a server is established and authorization is passed, the client will try to get information about maildrop and messages on the server. The server responds to every command from the client; however, the length of the response depends on how many email messages (for the particular user account) exist on the server.

Experiments also show that a successful POP3 conversation with a remote server that has an empty maildrop has a sum about 90 bytes of data payload. On the other hand, a POP3 connection that actually receives emails from the server will have an average data payload of 1000 bytes. Here we classify captured POP3 connections into 3 categories: “Invalid” connections are POP3 connections with payload less than 90 bytes, which cannot possibly complete the simplest conversation; “Unloaded” connections are those with payload between 90 and 1000 bytes; “Loaded” connections as those with payload more than 1000 bytes. “Unloaded” POP3 connections log into a server successfully, and query the maildrop, but do not download any email. “Loaded” connections include all actions of the “Unloaded” connections, and they retrieve at least one email from the server. It is possible that some connections have more than 1000 bytes of payload but retrieve nothing; they should be classified as “Unloaded”. In practice, we cannot distinguish them from “Loaded” connections based only on TCP header information. It is appropriate and more feasible to think them as “Loaded” connections.

We use random variables B_{LPC} and B_{UPC} to represent the number of bytes per “Loaded” connection and per “Unloaded” connection, respectively. As indicated by the LLCD plot, the B_{UPC} distribution is heavy-tailed . The B_{LPC} distribution is apparently Gamma. See Figure 16.



B_{UPC} is heavy tailed.



B_{LPC} is Gamma.

Figure 16: Plots of B_{UPC} and B_{LPC}

Statistics from Feb. 05 traffic reveal that 43.17% of total POP3 connections are "unloaded", while the rest are "loaded". Simulation based on this percentage returns a sample distribution which is very similar to that of CS traces, as shown in Figure 17.

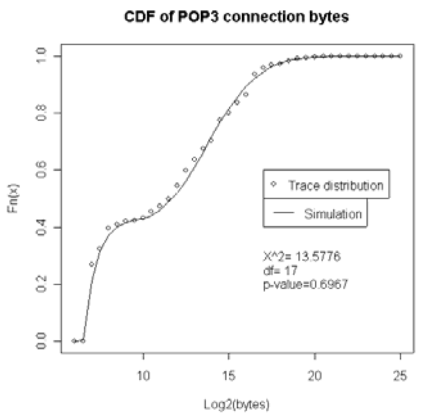


Figure 17: CDF of POP3 connection bytes

3.7 Modeling SSH Traffic

Generally, the traffic between an SSH client and server is similar to TELNET traffic, and we can use the random variables developed for TELNET to model SSH traffic. (See Table 1.)

3.7.1 Arrivals of SSH Connection (A_{SSH})

Contrary to the other modeled user-initiated TCP connections, The SSH arrival distribution is apparently Gamma (Figure 18).

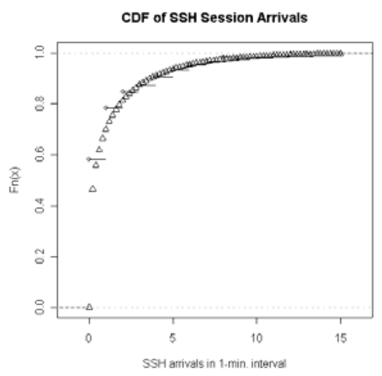


Figure 18: CDF of SSH connection arrival

3.7.2 Number of Originator Packets per SSH Connection (N_{SSH})

This random variable counts the total number of packets sent by a client (or originator) during a single SSH connection. The information exchanged in a SSH connection follows a fixed pattern: each time an SSH client sends a packet (request) to a server, the server will give a response of one or more packets to the client. Usually, the server does not send packets to client unless it receives a request. This protocol is simple to include in simulations and implies that control can be based on modeling carefully the number of packets that originate at the client.

Based on the analysis of the CS traffic traces, the number of SSH packets originating from a client fits a log2-normal distribution. (See Figure 19.)

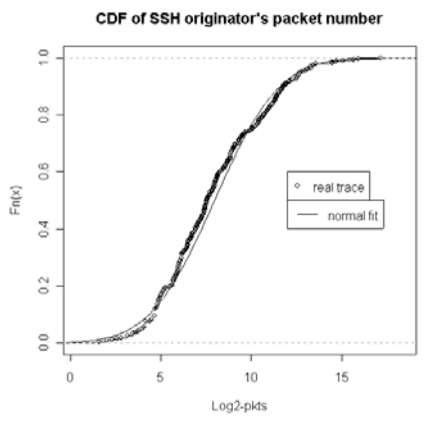


Figure 19: A_{SSH} is approximately lor2-normal.

3.7.3 Time Between Adjacent SSH Originator Packet (T_{SSH})

T_{SSH} is the time between two adjacent client packets in an SSH session. It controls the frequency of SSH traffic on the network. We found it difficult to find an appropriate distribution to model this random variable, even though the hybrid method lost its power when attempting SSH originator's packet spacing. However, we eventually discovered that the most distortion happens with small values T_{SSH}, less than 1 millisecond. Larger values of T_{SSH} satisfy a log-normal distribution (Figure 20). We decide to use this log-normal distribution as the model for T_{SSH}, because differences less than 1 millisecond will not cause an observable effect on simulation (when statistics are calculated on a time scale of seconds or minutes).

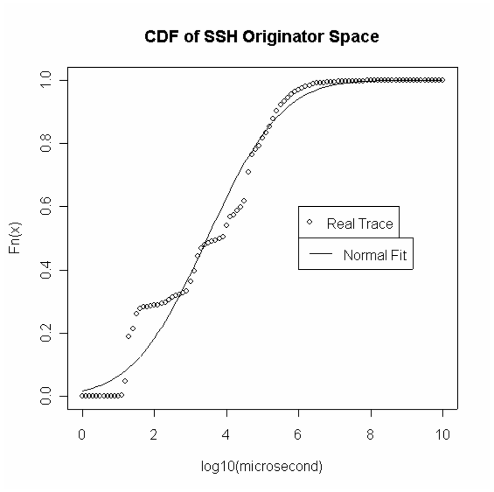


Figure 20: Log10-normal model of T_{SSH}

3.7.4 Number of Bytes per SSH Response (B_{SSH})

As described in 3.7.2, in a typical SSH conversation, the server responds to each client packet. The response can be only one packet or as many as thousands of packets. Unlike client packets, which are normally very small, server packets vary greatly in the number of bytes. Thus, we take the total number of bytes of a response, instead of number of packets, as the measure of SSH server's response.

After removing extremely large outliers (larger than 8000 bytes), we found that B_{SSH} has a close match to heavy-tailed Pareto distribution. The LLCD plot also confirms the finding. Figure 21 depicts that B_{SSH} of Feb. 05 traffic can be modeled by a Pareto distribution with parameters α=1.079 and k=32.

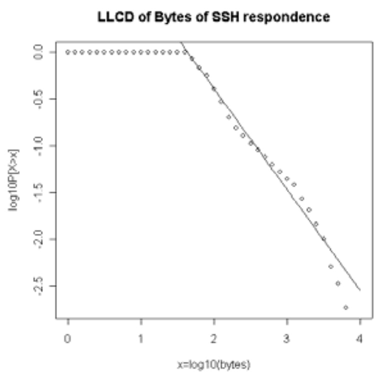


Figure 21: B_{SSH} is heavy tailed

CHAPTER 4: SIMULATING SYNTHETIC INTERNET BACKGROUND TRAFFIC IN NS-2

4.1 Why NS-2

Simulating Internet traffic is a tough job not only because of the challenges associated with building appropriate distribution models for all the random variables we need, but also because it is difficult to construct a simulation environment. Such a simulation environment must meet criteria given in the following paragraphs.

First, the simulation environment must be highly efficient. In our approach, we analyzed hourly traffic traces captured over a number of days. Thus, we were simulating many millions of packets over many hours of network time. Furthermore, it was generally necessary to repeat simulation runs often in response to changes and in an effort to obtain greater statistical confidence.

Second, the simulation environment must be highly flexible with regard to its structure. The target networks may range from a computer group in a small company to a large university's campus network, and even to the Internet. The ideal simulation environment supports adding, removing and changing network components in an easy manner.

Third, the simulation environment must model networking protocols realistically. Many simulations run at the application level, and their realism depends greatly on how well the simulation environment captures the important effects of the underlying transmission and routing protocols.

Finally, a good simulation environment must support trace collection on different levels, including the packet level. Only adequate trace analysis can compare simulation results to simulated traffic on multiple levels - both for validation and for analytical results.

NS-2 [36] is a discrete event simulator targeted at network research. NS-2 supports many kinds of network devices (end node, switch, router, link…) and various communication protocols (IP, TCP, UDP…). Because it is open-source, users can write or modify existing code for their own devices or protocols and introduce them to the NS-2 system. NS-2 users can easily construct their experiments even when designing complex networks. The complexity of the simulation is only limited by the computer's power. NS-2 provides authentic support for TCP, which means that in NS-2 TCP runs realistically with virtually all options supported. NS-2 also allows users to perform simulations at the packet level as well as the application level. No matter the simulation level, NS-2 can capture and store all passing packets at points specified by the user. Because of these characteristics plus its widespread acceptance in the research community, we used NS-2 for our simulation experiments.

4.2 Experiment Design

The purpose of our simulation experiment is to generate network traffic that is statistically similar to the Internet background traffic of a real organization. In our approach, the work begins with the analysis of traffic traces from an organization of interest.

The work reported here uses captured UCF traffic headers. We first built distribution models and estimated their parameters for all the required random variables. This step is described in sections 3.3 to 3.4. Then we ran a set of programs that implemented all the models and produced simulated values for each of the modeled random variables. For the FTP protocol, we got simulated values for FTP session arrival time, number of FTP-DATA connections for each of simulated session, number of bytes transmitted by FTP-DATA connections, and idle time between two sequential FTP-DATA connections. With all of these values in hand, we are able to run the NS-2 simulation, which takes inputs of simulated values and creates application-level connections. NS-2 completes all details of simulated connections including establishing TCP connections, dividing information into packets, handling congestion control, and so on. We set up a monitor point at the main link, through which all traffic packets between clients and servers must pass. This monitor captures all packets and stores them in a file. By reading the file and retrieving all the simulated packets, we are able to compare the statistics of simulation traffic with real traffic traces at the packet level.

Table 5 gives random variables and their actual parameters for the five major protocols we used in simulating the CS traffic. Figure 22 illustrates the simulation procedure. Figure 23 depicts the network structure we used in NS-2. In this structure, all client computers lay inside an organization and service requests are all sent to servers outside (over the Internet). A main link with large throughput connects the inside router and outside routers, representing an Internet access link. All client computers in the organization connect to the inside router via links with a small delay time, representing high speed LAN connections. All servers are connected to the outside router also by high speed connections. We set up a monitor at the access link between the organization and the Internet.

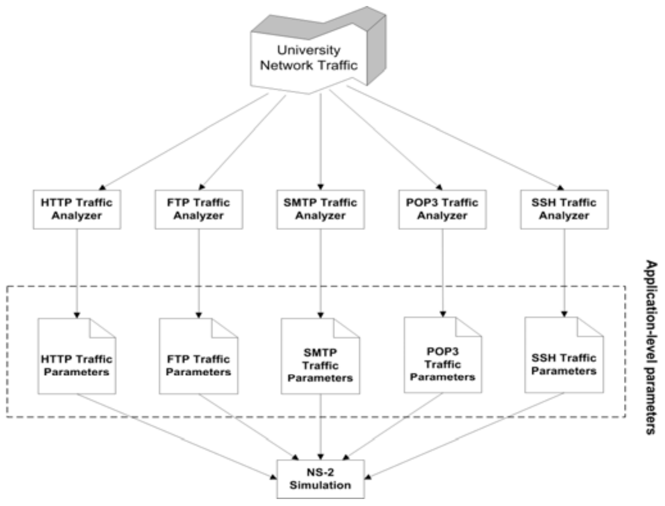


Figure 22: Simulation Procedure

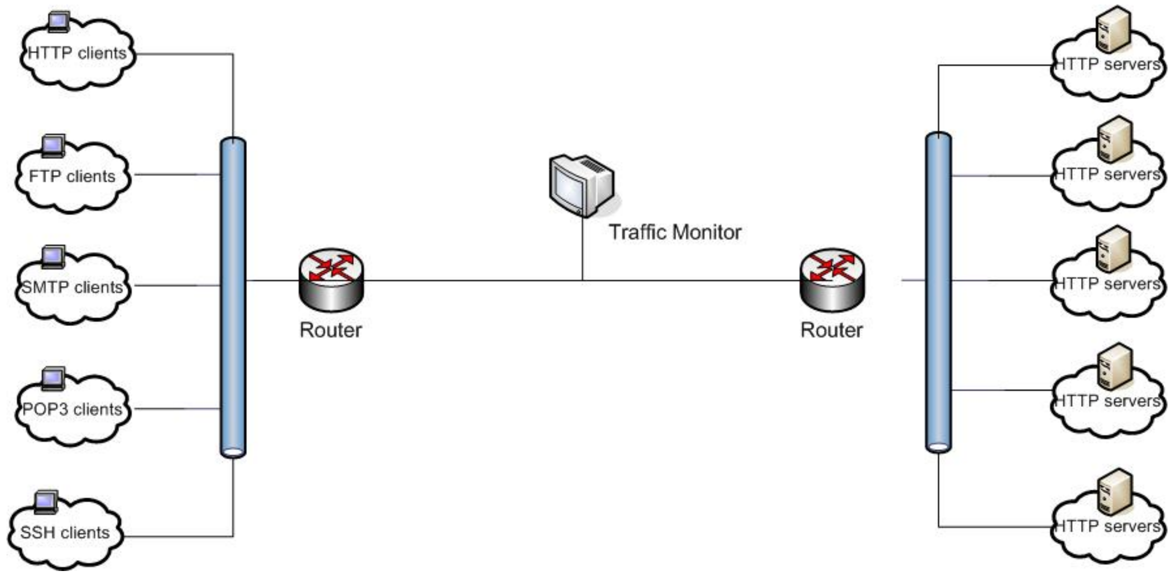


Figure 23: NS-2 simulating network structure

The CS traffic traces were captured from 8am to 6pm on Feb. 05, 2003. We simulated the entire 10 hours of collected traffic on the NS-2 platform. The running time of the simulation program in the Debian Linux on a computer with Pentium 4 2.0 GHz CPU and 1GB memory is about 40 minutes.

Table 5: Summary of random variables used in simulation

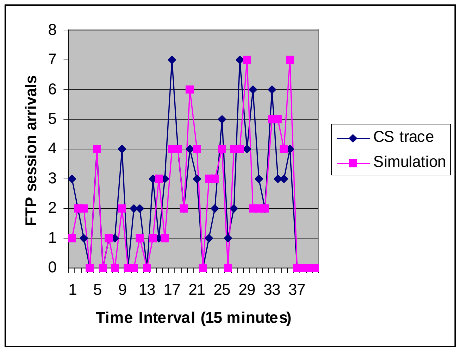
Protocol	Variable Name	Distribution	Parameters
HTTP	Session Arrival	Poisson	λ varies every 1 minute
	Bytes per page	Pareto (upper 36%)	α=1.164, k=$10^{4.25}$
		Exponential (lower 64%)	rate=0.0002419939
	Number of Pages per Session	Pareto	α=1.26, k=1
	Idle Time between 2 pages	Gamma	shape=0.9936, rate=0.0504
FTP	Session Arrival	Poisson	λ varies every 5 minute
	Number of FTP-DATA per session	Pareto	α=1.0595, k=3
	Bytes per FTP-DATA	Pareto (upper 15%)	α=1.15, k=10000
		Exponential (lower 85%)	rate=0.00052
	Idle time between 2 FTP-DATAs	Gamma (>1 sec.)	shape=0.227, scale=73.962
		Gamma (< 1 sec.)	shape=202.04, scale=0.079
SMTP	Connection Arrival	Poisson	
	Bytes per Connection	Pareto	α=0.8454, k=1250
POP3	Connection Arrival	Poisson	
	Bytes per Connection	Gamma (loaded)	shape=40.6029, rate=2.8903
		Pareto (unloaded)	shape=1.873, k=90
SSH	Connection Arrival	Gamma	Shape=0.2784 Rate=0.2260
	Number of Originator Packet per connection	Log2-normal	μ=7.9613, σ=2.8304
	Time between 2 Originator Packets	Log10-normal	μ=3.4624, σ=1.6275
	Bytes of Response	Pareto	α=1.079, k=32

4.3 Results

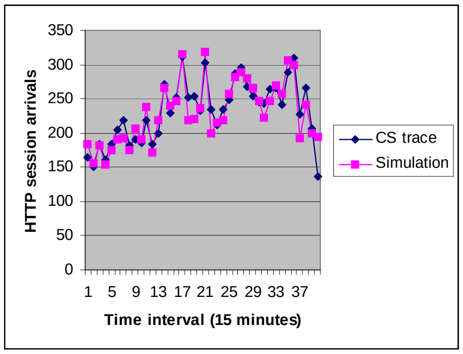
Here we compare the simulation results with the original CS traffic traces. We give the comparison of session arrivals for all five protocols. Although we obtained satisfactory results for other random variables, we only show comparison plots for session arrivals due to space limitation. Several other measurements at the packet level are also compared. The first packet-level measurement is packet arrivals per second. Autocorrelation, self-similarity, correlation integral and fractal dimension are then calculated and compared. Note that the latter 4 measurements are calculated based on the packet arrival measurements.

4.3.1 Session Arrivals

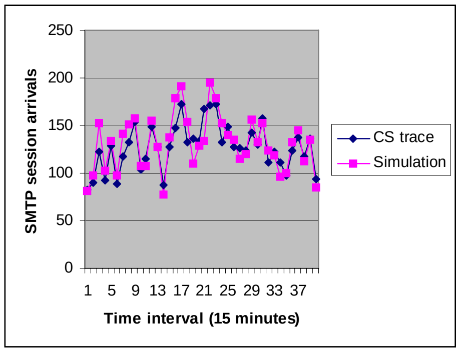
The simulated session arrivals of the 5 protocols are depicted and compared to the CS traffic trace in Figure 24. The simulation results compare favorably to real data.



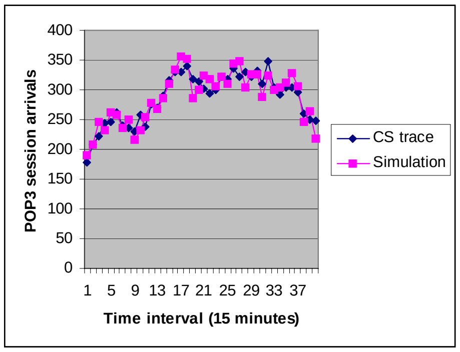
FTP session arrival



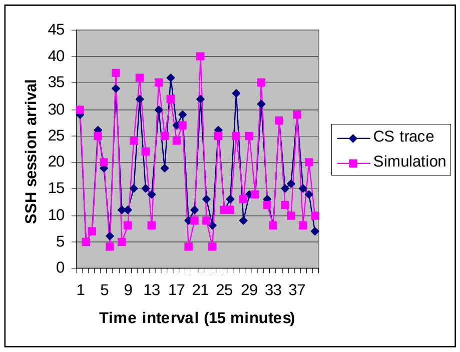
HTTP session arrival



SMTP session arrival



POP3 session arrival



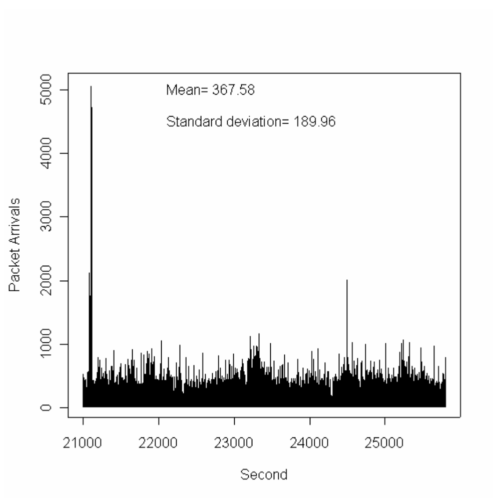
SSH session arrival

Figure 24: Session arrivals of 5 protocols

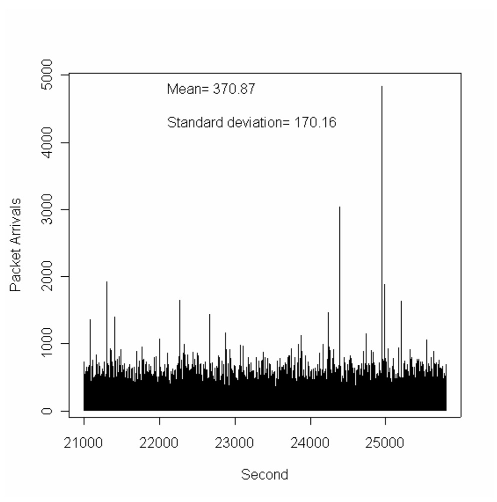
4.3.2 Packet Arrival

From section 4.3.2 to 4.3.5, we discuss several packet-level measurements obtained from the simulation. To make the comparisons we used the time period in the CS traces (from 1:50pm to 3:10pm) when packet arrivals were most intense. Figure 25 compares the packet arrivals of the CS traces with those of the simulation. The means and standard deviations compare favorably.

The maximum burst of both traces is around 5000 bytes/sec., while the simulation has a larger number of small bursts.



Packet arrival of CS traffic traces



Packet arrival of NS-2 simulation traces

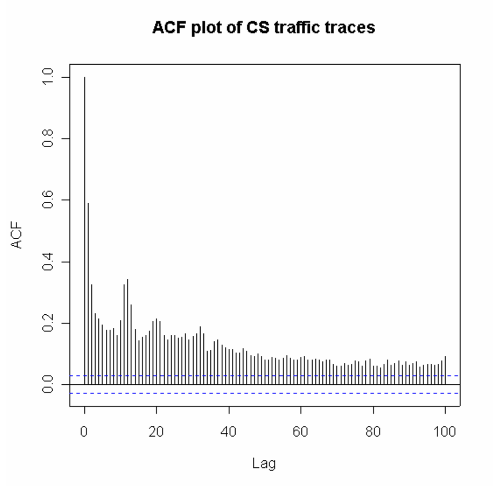
Figure 25: Comparison of packet arrivals

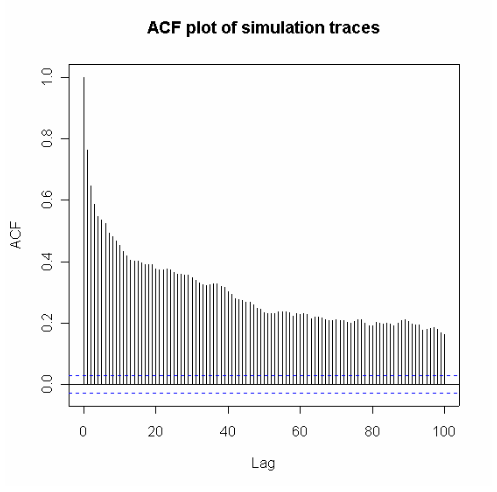
Network traffic has the feature of long-range dependence, which can be demonstrated by the Autocorrelation function (ACF). Figure 26 gives ACF plot for CS traces and the simulation. The ACFs show that, as expected, both represent long-range dependence.

4.3.3 Self-similarity

As noted previously, self-similarity is an important characteristic of network traffic. Here we compare the data and simulation results with respect to this important characteristic.

We collected TCP packet counts for every second from both UCF and simulation traces. We tested their self-similarity using log-log spectra density plots near the origin, as showed in Figure 27. This method of testing self-similarity is described in [52] [93].

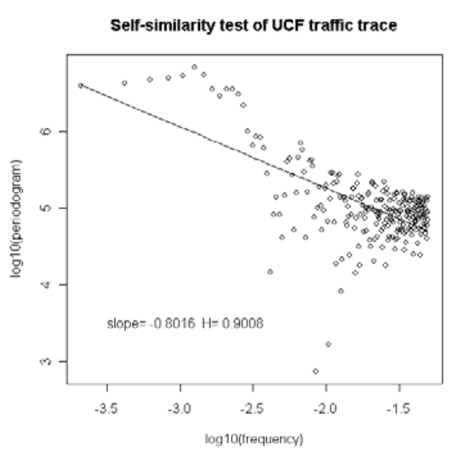


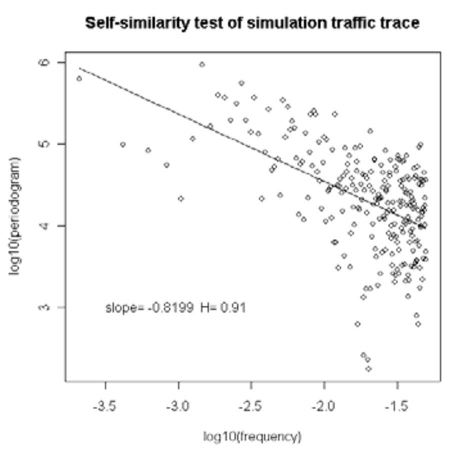


ACF plot of CS traffic traces

ACF plot of simulation traces

Figure 26: ACF plots of CS and simulation traffic





Log-log specta density plot of UCF trace

Log-log spectra density plot of simulation trace

Figure 27: LLCD plots of CS and simulation traffic

The straight lines in figure 33 are linear models of the data. The degree of self-similarity of the data is obtained from the slope of the straight line. The well-known Hurst parameter is found as H = (1-slope)/2 [52]. Repeated simulations yield similar results: the Hurst parameters of real traffic traces and simulation traces have comparable values.

4.3.4 Correlation Integral and Fractal Dimension

That network traffic also exhibits "fractal-like' behavior that has been observed and studied by Leland [92], Willinger and Paxson [94]. Grassberger and Procaccia [66] give one method to calculate a time series' fractal dimension through correlation integrals. Let x_i be the number of packet arrivals at the ith second, we use following formula to calculate correlation integrals:

The difference between the number of packets arriving at the ith time step and the jth time step is the "distance" $|x_i - x_j|$. C(r) takes values in the range [0,1].

$$C(r) = \lim_{N\to\infty} \frac{1}{N^2} \sum_{i,j=1,i\neq j}^{N} \theta(r - |x_i - x_j| / D) \tag{4-1}$$

where $|x_i - x_j|$ is the Euclidean norm and θ is the Heaviside function

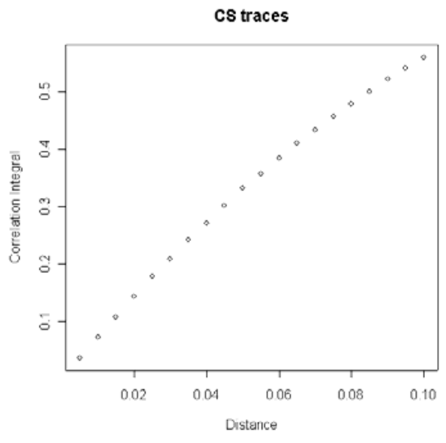
$$\theta(x) = \begin{cases} 0, x < 0 \\ 1, x > 0 \end{cases}$$

Grassberger and Procaccia also showed in [66] that for small r, the correlation integral C(r) grows according to a power law:

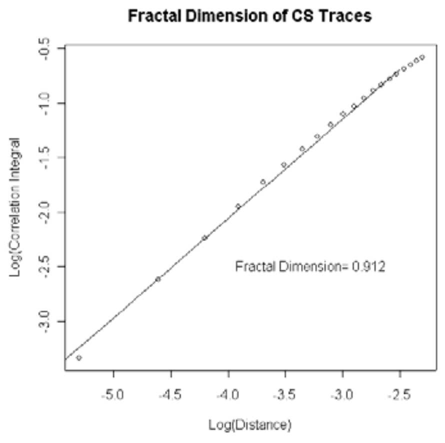
$$C(r) \sim r^{v}, \tag{4-2}$$

and v is the estimation of the correlation fractal dimension, and it can be determined by plotting C(r) vs. r on a log-log plot.

Figure 28 and Figure 29 display correlation integral plots on normal and log-log axis, and they also gives the estimated values of fractal dimension. Figure 28 shows the plots for CS traces, and Figure 29 shows those for simulation traces.



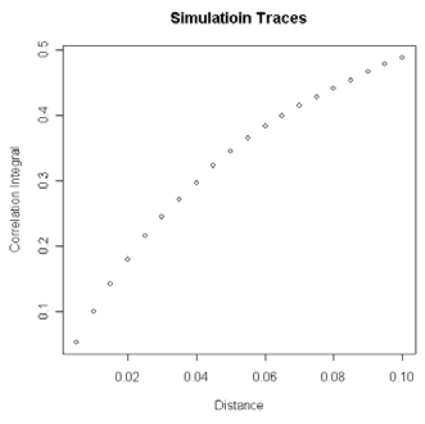
Correlation integrals



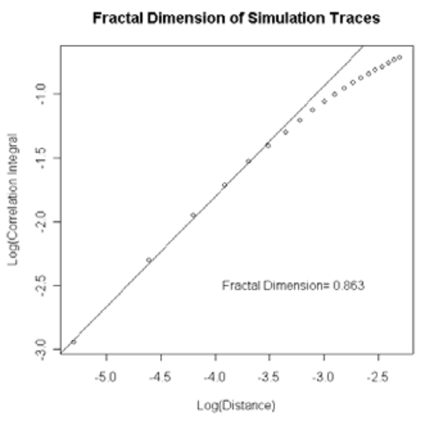
Correlation Integrals on log-log axis and the estimated fractal dimension

Figure 28: Correlation integrals and fractal dimension of CS traces

The plots show that the simulation has similar correlation integrals and close values of fractal dimension to the simulated traffic.



Correlation integrals



Correlation integrals and estimated fractal dimension

Figure 29: Correlation integrals and fractal dimension of simulation

CHAPTER 5: GENERATING NETWORK TRAFFIC USING HONEST

Although using NS-2 for traffic simulation has its own advantages, it never replaces the need for a real laboratory networking environment. Live traffic with real content payload in each packet will provide the complexity of the real world, upon which an IDS system can be reliably tested.

The Hands-On Networking Environment for Security Testing (HONEST) is designed to run traffic simulation experiments in a laboratory network environment in order to test Intrusion Detection Systems. The system is comprised of a few independent computers- four at a minimum- as well as other networking components. In order to run properly, the system needs one master system, one database server, and two (or more) machines that serve as *matrices*. A matrix is one physical machine that can simulate multiple logical machines. HONEST includes following components: 1) at least one Server Matrix machine; 2) at least one Client Matrix machine; 3) a data network hub where clean generated traffic as well as attack data are sent from clients to servers; 4) a management network switch that allows the test to be set up and run from one master system without interfering with the test data; 5) a database server for use with the model-based background traffic generation; and 6) the MAGNA attack management and generation tool. The GUI resides on the master system and controls each function of the process, which is divided into two concrete sections: set-up and execution. The system layout is depicted in Figure 30. Most programming work was done by Michael A. Cooper at FIT [48].

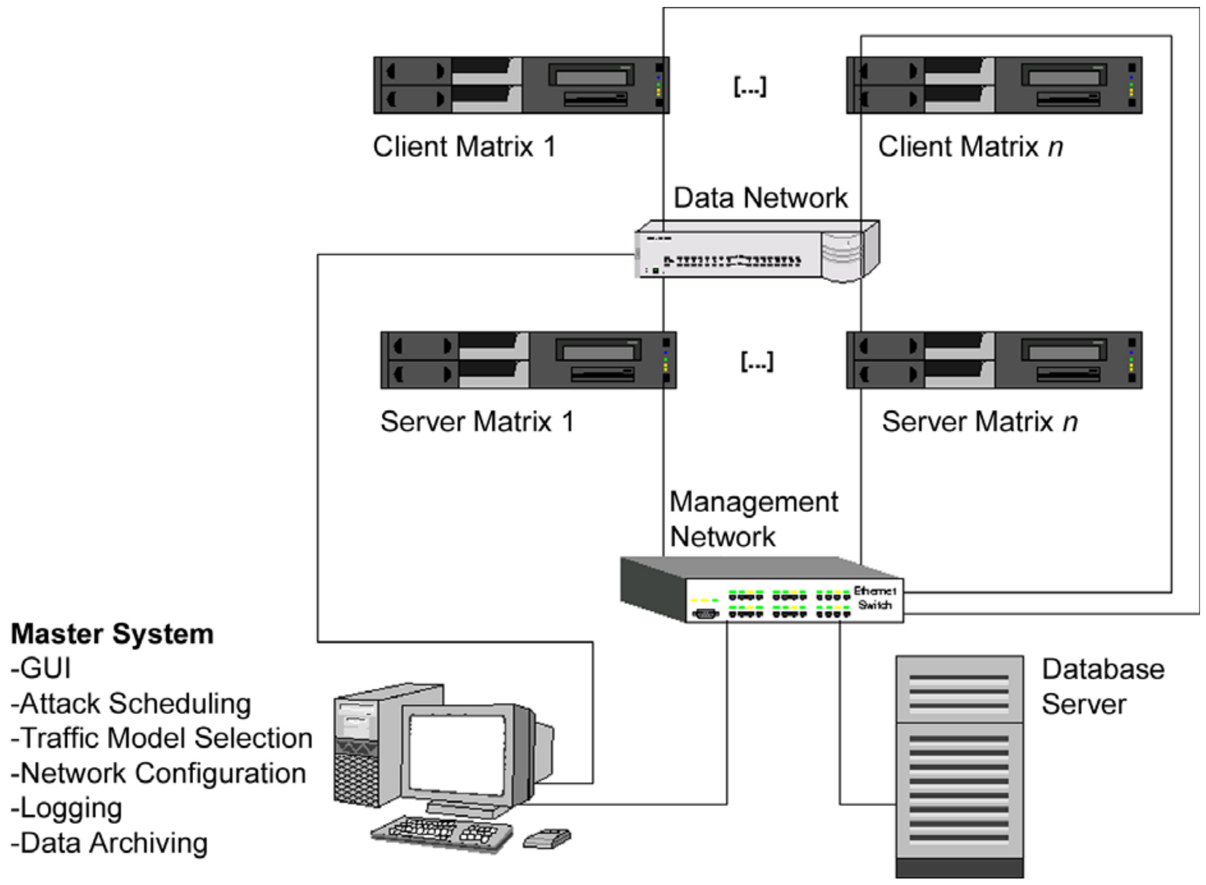


Figure 30: HONEST System Layout

HONEST has several advanced features. First, it is a distributed simulation environment. The configurations of servers and clients are sent to networked computers, which accordingly play the roles (server or client) assigned by the master system. Second, HONEST integrates background traffic generation with attack generation. Besides the models of background traffic, users can also choose attack models through MAGNA, a system used to generate realistic network attacks. Third, HONEST is a scalable simulator. Users can configure the traffic intensity of each protocol for computers in the network, and thus, have the control to adjust the simulation to make it more like a specific network. Users can also define how many virtual machines there are in a matrix, depending on the size of the network to be simulated. At last, HONEST provide a convenient GUI interface, through which users can input, review and modify the simulation configuration, as well as watch the log information.

5.1 Experimental Design

This section describes the experimental settings used in HONEST to create the dataset that was compared to the original traffic model. With these settings, a user will be able to duplicate the experiment as it was run for this work using an appropriately configured HONEST system.

For this work, HONEST ran with a total of five machines including a database server used as the Server Matrix for the experiment. Three Client Matrices were used to produce traffic and the Master System was used for control and data collection. This configuration allows the many facets of the HONEST system to be showcased, albeit in a limited capacity.

Each of the Client Matrices simulates ten clients, and the one Server Matrix simulates five servers. The Server Matrix supports multiple IP addresses and connections on behalf of clients. The data network hub allows all the produced traffic (comprised of background traffic from four protocols plus the scheduled attacks) to be collected and archived using the master system. The management network switch allows fast and simple communication between the master system and each of the matrices, as well as between each of the matrices and the database server. All of this communication is left out of the archived dataset; appropriate log entries are made for events such as file transfers and program starts and stops.

The scripts that are generated from skeleton script files are also archived with the network capture, which aids in tracking down errors if any occur and also serves as an archive of the experimental execution. Combined with the network capture dataset and the experiment file, a complete record of the settings of the specific execution, and of the results can be reviewed.

5.2 Experiment Settings

5.2.1 Traffic Model

The first step in the setup process requires the traffic model to be selected so that the protocols being modeled are available to later steps in the process. Here, the definition of model is not the same as previous chapters, which refers to a statistical distribution. In this chapter, a model used by HONEST refers to some choices that have been already been made and parameters defined to represent a particular traffic data set.

The model that is selected for this experiment is the UCF model that was described in the previous chapters. This model includes four protocols: HTTP, FTP, SSH, and SMTP. Note that the POP3 model is missing because of the lack of appropriate simulation tools for this protocol. This experiment used the default model timing of 36,000 seconds (10 hours), but the user may wish to run smaller experiments for debugging purposes. Running longer experiments is not supported because the estimated parameters vary with time and runtime may not exceed the length of time specified.

5.2.2 Network Parameters

The next step is configuring the network settings. This experiment used one Server Matrix and three Client Matrices. The Server Matrix was a "unix" type machine, and simulated five servers with IPs in the 192.168.0.0/16 range. There were three Client Matrices, each simulated ten client IPs. The IP address ranges of these three Client Matrices were 192.168.1.0/24, 192.168.2.0/24, and 192.168.3.0/24.

Once the network information is entered, the scale factors and percentages for each server must be configured by protocol. This experiment used settings that may be typical in a real networking environment, such as only one server for SMTP traffic and SSH traffic and a mix of HTTP and FTP traffic. Because this experiment had one Server Matrix hosting five simulated servers, there were a few servers available to divide up the active protocols. Each of the scale factors was left at 1 in order to produce the same number of arrivals that are specified in the model. Increasing this number increases the simulated "users" and therefore, the amount of traffic generated; conversely, decreasing this number will reduce the amount of traffic generated. The settings for each server are shown in Table 6: Experimental settings for "Total Traffic per Protocol.

Table 6: Experimental settings for "Total Traffic per Protocol"

	Server0	**Server1**	**Server2**	**Server3**	**Server4**
HTTP	60	10	20	0	10
FTP	75	0	25	0	0
SSH	0	0	75	0	25
SMTP	0	0	0	100	0

After each of the servers is assigned to host a percentage of a protocol's total traffic, each server's traffic must be divided among the clients. For this experiment, server traffic streams were divided evenly among the clients. Because there are three Client Matrices, each simulating ten clients, there are a total of thirty clients, which makes an even distribution 3% of each protocol per client, for each server. Each of the five servers is configured similarly.

5.3 Experiment Results

This section details the results of the experiments that were performed with the HONEST system. The results compare two different experimental settings with the original captured traffic. A first dataset was determined to be unsatisfactory, and a second dataset (described in the previous section) was run from the redesigned experiment in an effort to produce more realistic traffic. The original traffic was captured from the UCF Computer Science LAN on February 5th, 2003. The first dataset was produced using only three matrix machines, with a total of four client IPs and two server IPs. The second dataset used the experimental settings from the previous section, with a total of 30 client IPs and five server IPs. The results are first presented for the overall traffic based on the second dataset, and are then broken down by each protocol.

5.3.1 Overall Traffic Comparisons

The most obvious difference between the simulation datasets and the original traffic capture is that the simulations have more TCP packets, but less TCP connections. Since there are fewer connections, the simulations have a much higher number of packets per connection. Table 7 shows the number of packets and connections for each of the datasets, while Figure 31 and Figure 32 show comparative graphs of packet arrivals and connection arrivals, respectively.

Table 7: Packet and connection number statistics

	Packets	Connections	Pkts/Conn.
UCF_data	8541517	100072	85.35
Simulation_#1	10955542	42608	257.12
Simulation_#2	14306800	39300	364.04

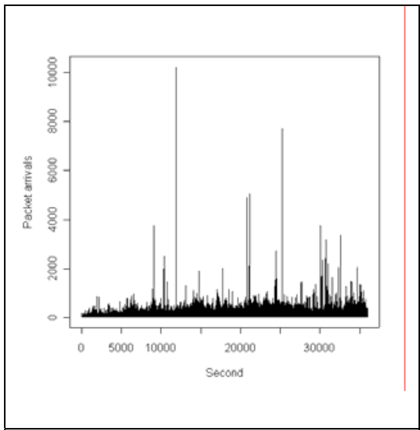
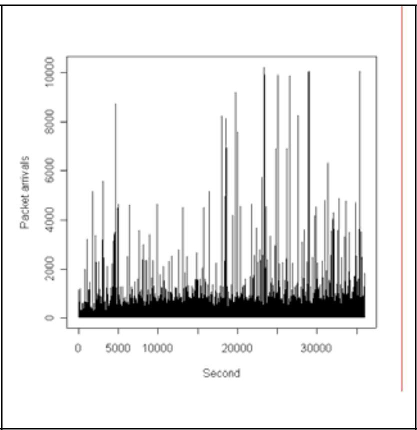
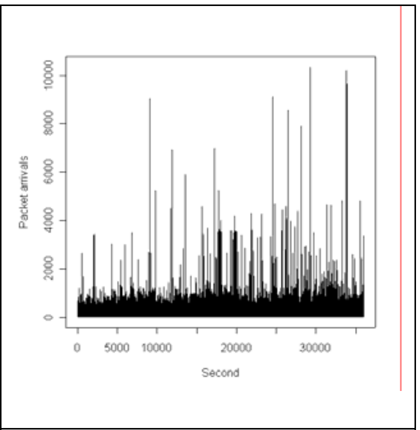
		
Packet arrivals of the UCF traffic	Packet arrivals of simulation #1	Packet arrivals of simulation #1

Figure 31: Packet Arrivals

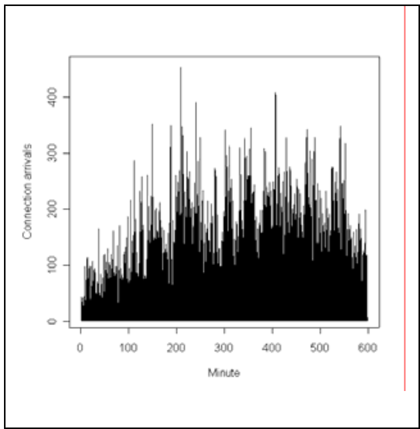
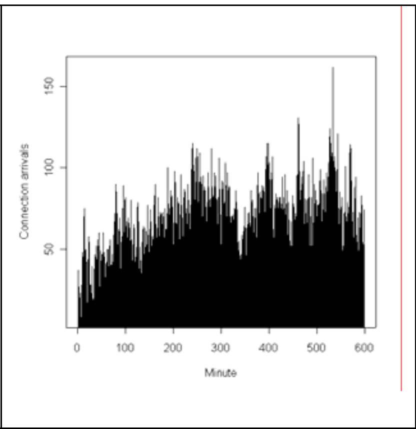
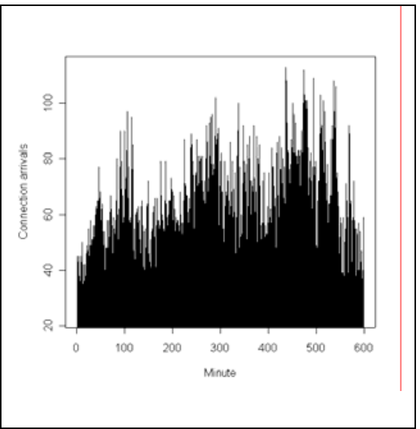
		
Connection arrivals of the UCF traffic	Connection arrivals of simulation #1	Connection arrivals of simulation #2

Figure 32: Connection Arrivals

The packet percentages of each protocol that make up the complete traffic dataset are shown in Table 8. These results show that with the exception of SMTP, the overall percentages of each of the protocols is different when compared to the original UCF capture. For HTTP, the dominant protocol in the UCF dataset, packets dropped from 51% to 23% in simulation #1 and, 13% in simulation #2. In the UCF capture, SSH only accounts for 36% of the traffic, while in simulations #1 and #2 it accounts for 59% and 66%, respectively. Similarly, FTP increases from 10% in the UCF capture, to 15% in simulation #1 and 19% in simulation #2. The SMTP traffic decreases slightly from 3% in the UCF capture to each of the simulations, which each have SMTP percentages of 2%.

Table 8: Packet percentages per protocol

	HTTP	**SSH**	**SMTP**	**FTP**	**SUM**
UCF_data	51%	36%	3%	10%	100%
simulation_#1	23%	59%	2%	15%	100%
simulation_#2	13%	66%	2%	19%	100%

While the packet percentages changed dramatically between each of the datasets, the number of connections did not. Table 9 details the connection numbers of each of the 3 datasets. HTTP decreases from 90% in the UCF capture to 84% and 83% in simulations #1 and #2. SMTP, increases from 7% to 11% and 12% in the simulations, while SSH and FTP increase by 1%. Based upon the information in each of these tables we can see that HTTP traffic was decreased and that SSH traffic was increased between the UCF traffic capture and the two simulations. The cause for this change needs to be determined, and may lie in further analysis of each of the protocols individually.

Table 9: Connection percentages per protocol

	HTTP	SSH	SMTP	FTP	SUM
UCF_data	90%	1%	7%	2%	100%
simulation_#1	84%	2%	11%	3%	100%
simulation_#2	83%	2%	12%	3%	100%

5.3.2 Autocorrelation

As shown in Figure 33, we produced autocorrelation function (ACF) plots on the three datasets. The UCF traffic capture shows more long-range-dependence of the packet arrivals when compared to the two simulation datasets. Of the two simulations, simulation #2 has much more obvious long-range-dependence on the packet arrivals than simulation #1. Because simulation #2 used more client and server IPs, we infer that long-range-dependence in network traffic can be better simulated by using more IP addresses to increase the number of available clients and servers, or use more physical machines because the TCP stack and queue management on each physical machine influences the packet transmission, thus having an effect on the autocorrelations of the packet arrivals.

Autocorrelation of connection arrivals is surprisingly better. As shown in Figure 34, both simulations show long-range-dependence for connection arrivals as does the UCF captured traffic. Since connection arrivals are on the application layer, they are not affected by TCP stack and queue management.

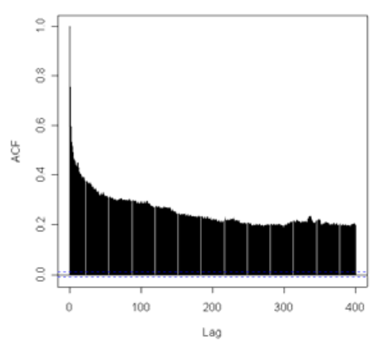
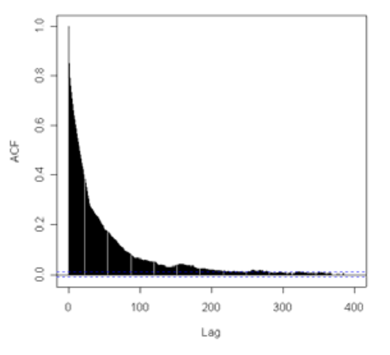
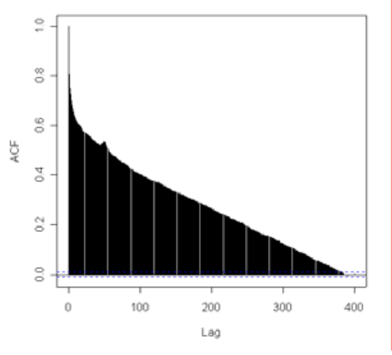
		
ACF of packet arrivals of the UCF traffic	ACF of packet arrivals of simulation #1	ACF of packet arrivals of simulation #2

Figure 33: Autocorrelation of packet arrivals

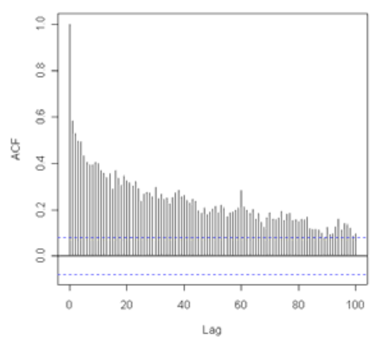
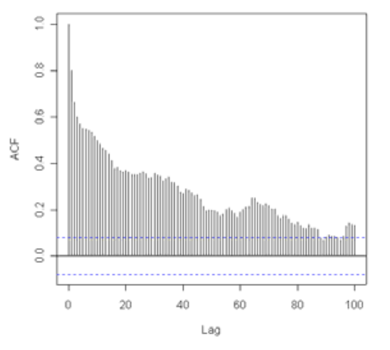
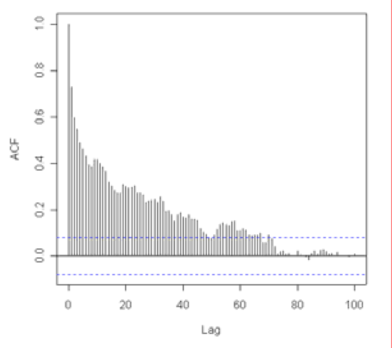
		
ACF of connection arrivals of the UCF traffic	ACF of connection arrivals of simulation #1	ACF of connection arrivals of simulation #2

Figure 34: Autocorrelation of connection arrivals

5.3.3 Self-similarity

In order to determine self-similarity of each of the datasets, the Hurst parameter was estimated for the UCF traffic and the two simulations. Self-similar sequences have a Hurst parameter H between 0.5 and 1.0. As shown in Figure 35, the Hurst parameter for the UCF traffic capture was 0.9243- confirming self-similarity. However, each of the simulations has Hurst parameters above 1.0. Simulation #1 has a Hurst parameter of 1.0482; and simulation #2 has a value of 1.049. While these values are close to 1.0, they are greater and we can therefore conclude that the two simulation datasets lost self-similarity. Because self-similarity depends upon long-range-dependence, we believe that the loss of self-similarity is a natural consequence of the loss of long-range-dependence as shown in section 5.3.2. Figure 33 in section 5.3.2 indicates that long-range-dependence improves when the number of client and server IPs increases. A reasonable explanation is that the TCP queuing will affect the traffic's long-range-dependence. When traffic from many sources has to use a single TCP queue, the queuing management will restrain bursts, and thus damage the long-range-dependence and self-similarity. To solve this problem, one can add more computers into the lab, and simulate as many IP addresses as necessary. But due to the limitation of our lab environment, using many computers is currently inapplicable.

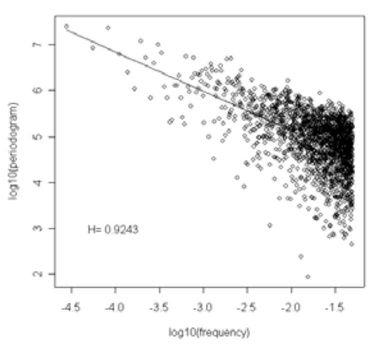
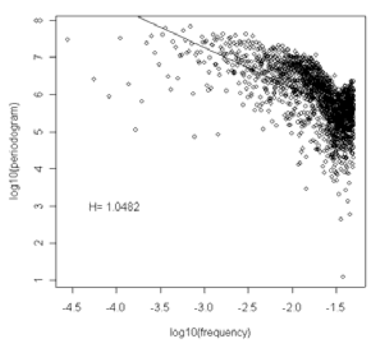
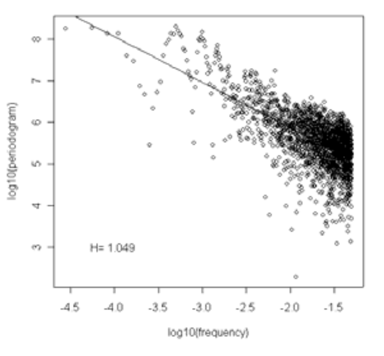
		
Self-similarity of packet arrivals of the UCF traffic	Self-similarity of packet arrivals of simulation #1	Self-similarity of packet arrivals of simulation #2

Figure 35: Self-similarity

5.3.4 HTTP Protocol Comparisons

In the UCF traffic capture, one HTTP page may contain more than one TCP connection, since a page may contain images and other elements that are loaded separately. In our simulations, each page is represented by a single connection. As seen in Table 10, both simulations generate a similar number of page views when compared with the traffic capture. However, on average they generate fewer packets per page, which explains why HTTP accounts for less of the total traffic in the simulations than the original UCF traffic capture.

In Figure 36 we show the HTTP packet arrivals for all three datasets. The two simulations do not match the original traffic capture, and have larger extremes as high as 10,000; the largest value in the original UCF traffic is around 2000. Another aspect of the HTTP traffic of the simulations is that the packet arrivals display a layered pattern. This is because the HTTP files that are retrieved are mock files, created for the simulation with sizes of powers of two.

Table 10: HTTP packet and page statistics

	Packets	**Pages**	**Pkts/Page**
UCF_data	4365316	31630	138.01
Simulation_#1	2561310	35966	71.21
Simulation_#2	1879462	32629	57.6

Figure 37 shows the page arrivals of the HTTP traffic for each of the datasets. Here the simulations appear similar to that of the original UCF traffic capture. This means that the algorithms generating session and page arrivals are correct, and may only need to be updated to include better and more diverse file selection.

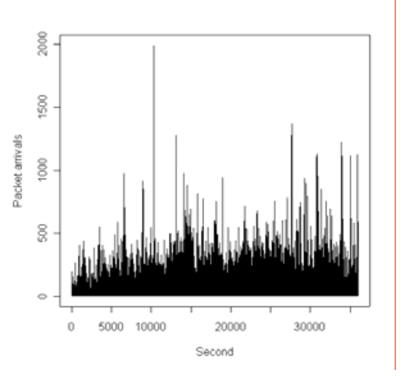
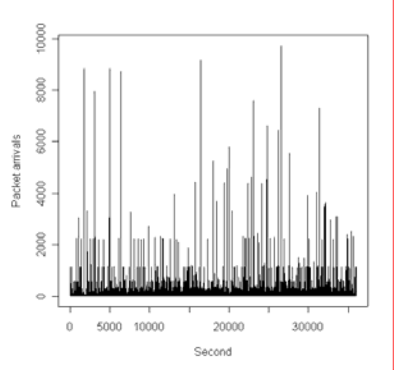
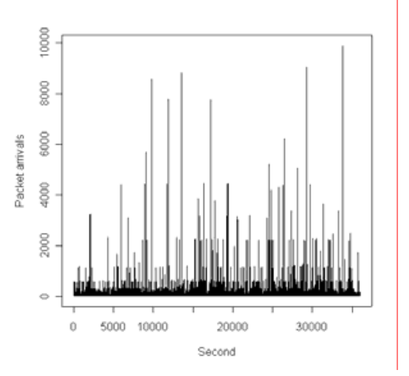
		
HTTP packet arrivals of the UCF traffic	HTTP packet arrivals of simulation #1	HTTP packet arrivals of simulation #2

Figure 36: HTTP packet arrivals

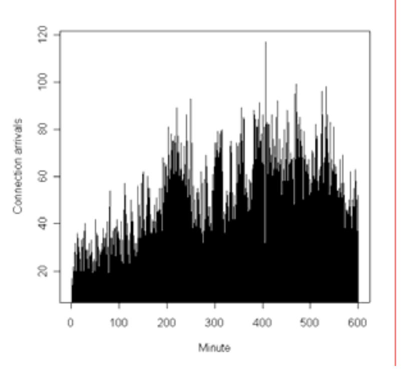
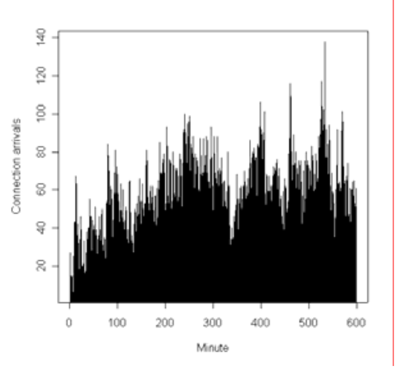
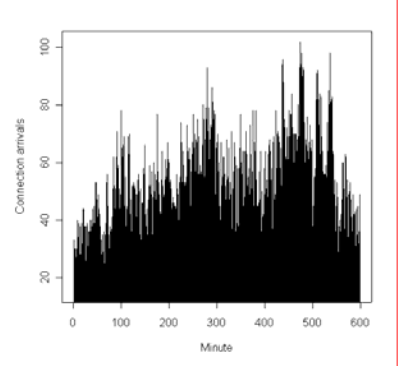
		
HTTP page arrivals of the UCF traffic	HTTP page arrivals of simulation #1	HTTP page arrivals of simulation #2

Figure 37: HTTP page arrivals

At this time, the application layer random variables cannot be compared for HTTP because the relatively low number of client IPs in both of the simulations, when compared to the UCF traffic capture, make it difficult for the analyzing programs to identify individual sessions that are repeatedly launched from the same client IP.

5.3.5 SSH Protocol Comparisons

The actual number of SSH connections are similar to the original UCF traffic capture. The average packets per connection (Table 11), however, are doubled for simulation #1, and tripled for simulation #2! This explains why SSH is seen as a much larger percentage in the simulations than in the UCF dataset as described in section 5.3.1, which can be observed in Figure 38. Since the session arrivals are similar (Figure 39), we suspect that it is the connection size that increases the SSH traffic so dramatically.

Table 11: SSH packet and connection statistics

	Packets	**Connections**	**Pkts/Conn.**
UCF_data	3084105	739	4173.35
Simulation_#1	6451523	776	8313.82
Simulation_#2	9391463	904	10388.79

For SSH traffic generation, there are two random variables that determine the connection size: the number of originator's packets and the responder's bytes size. These variables determine how much data is sent from the client to the server, and then back from the server to the client, respectively. Figure 40 and Figure 41 show the cumulative distribution function (CDF) plots of these variables.

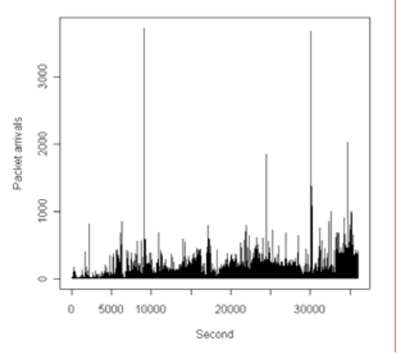
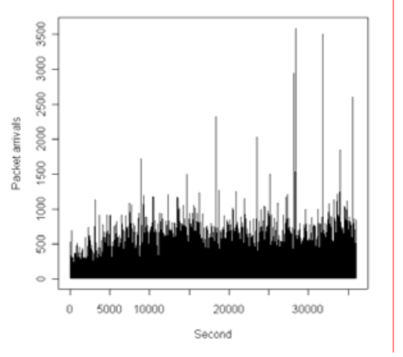
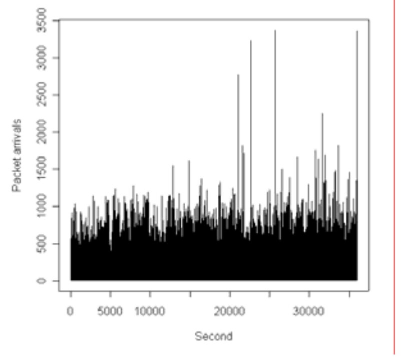
		
SSH packet arrivals of the UCF traffic	SSH packet arrivals of simulation #1	SSH packet arrivals of simulation #2

Figure 38: SSH packet arrivals

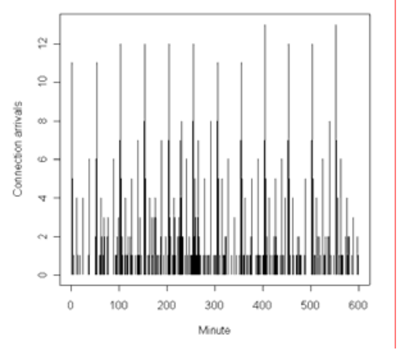
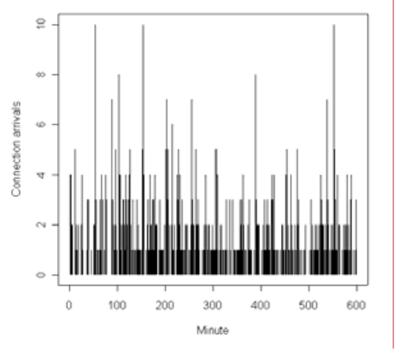
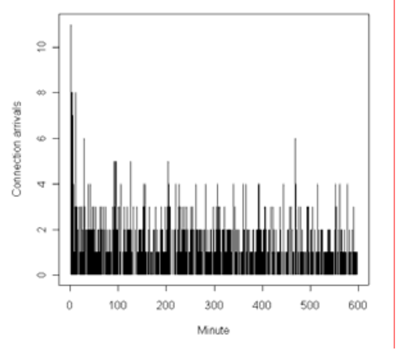
		
SSH connection arrivals of the UCF traffic	SSH connection arrivals of simulation #1	SSH connection arrivals of simulation #2

Figure 39: SSH connection arrivals

Figure 40 shows that the number of originator's packets for both simulations generally fit the UCF traffic capture. However, Figure 41 shows that the responder's bytes for both of the simulations shift to big values, and the distributions change to Normal-like while the UCF traffic capture is Exponential-like. For the SSH protocol it appears that either the model data used or that the traffic generation scripts have errors, and both need to be reviewed.

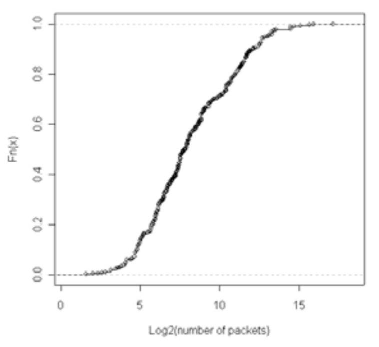
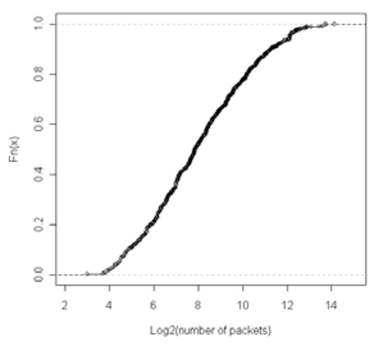
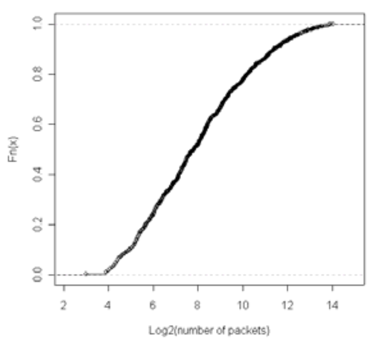
		
CDF of SSH connection size of the UCF traffic	CDF of SSH connection size of simulation #1	CDF of SSH connection size of simulation #2

Figure 40: SSH connection size CDF

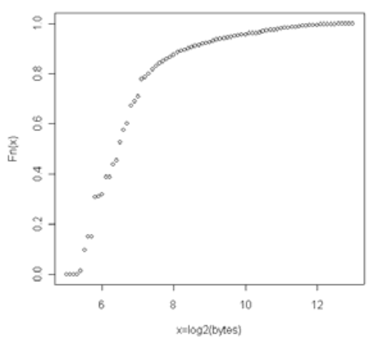
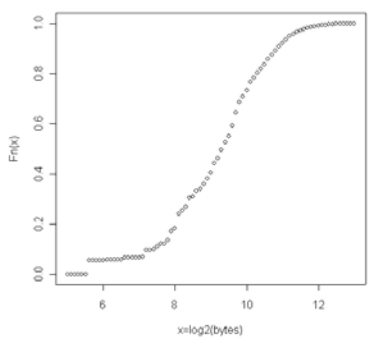
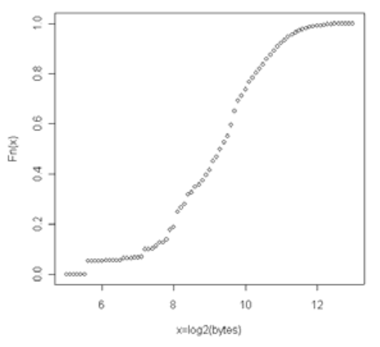
		
CDF of SSH responder's bytes of the UCF CS traffic	CDF of SSH responder's bytes of simulation #1	CDF of SSH responder's bytes of simulation #2

Figure 41: SSH responder's bytes CDF

5.3.6 SMTP Protocol Comparisons

In Table 12, we see that the simulation generated for SMTP has fewer connections, but the packets per connection are greater so total number of packets is similar. We believe that the differences between the real and simulated number of connections and packets per connection are in an acceptable range. However, there are a few seconds with noticeably large packet arrivals in the UCF traffic capture that are missing from the simulations. When these few bursts are removed, the simulation plots (Figure 42 and Figure 43) look similar to the original UCF dataset.

Table 12: SMTP packet and connection statistics

	Packets	**Connections**	**Pkts/Conn.**
UCF_data	294951	6715	43.92
Simulation_#1	245719	4568	53.79
Simulation_#2	301241	4703	64.05

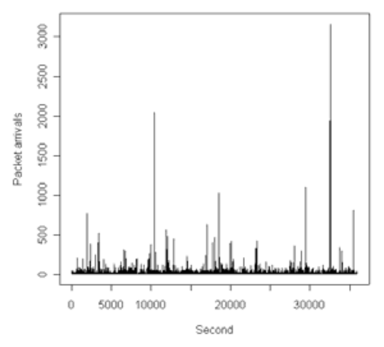
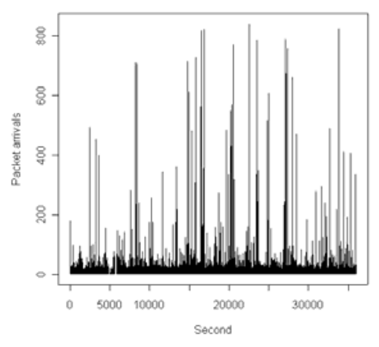
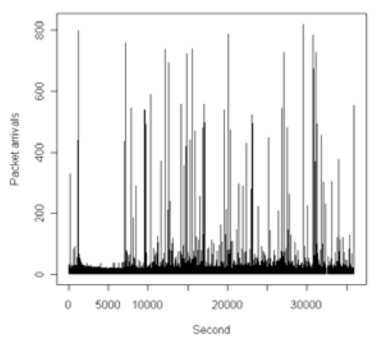
		
SMTP packet arrivals of the UCF traffic	SMTP packet arrivals of simulation #1	SMTP packet arrivals of simulation #2

Figure 42: SMTP packet arrivals

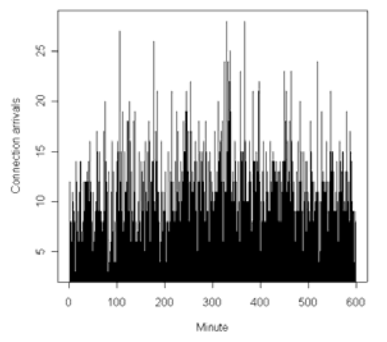
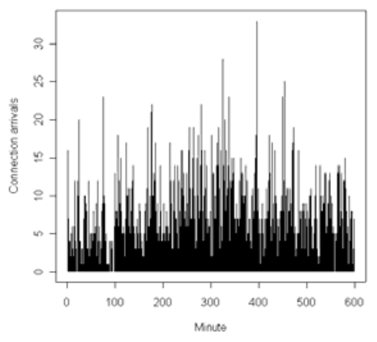
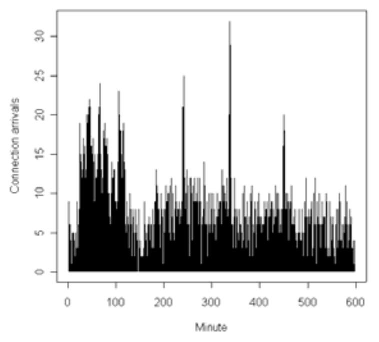
		
SMTP connection arrivals of the UCF traffic	SMTP connection arrivals of simulation #1	SMTP connection arrivals of simulation #2

Figure 43: SMTP connection arrivals

5.3.7 FTP Protocol Comparisons

For the FTP protocol, there are fewer connections and a much higher number of packets per connection. Simulation #1 had three times as many packets per connection as the UCF traffic capture, and simulation #2 had six times as many packets per connection! The simulations have much larger mean bytes for the FTP-DATA connections than the UCF dataset does, which might again be related to the file set used for transfers- HTTP and FTP used the same files with sizes as powers of two. The simulations may have transferred more of the larger sized files than those seen in the UCF traffic capture.

Table 13: FTP packet and connection statistics

	Packets	Connections	Pkts/Conn.
UCF_data	829710	1919	432.37
Simulation_#1	1697206	1302	1303.54
Simulation_#2	2734631	1052	2599.46

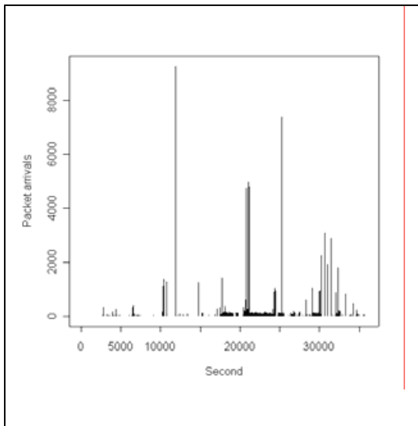
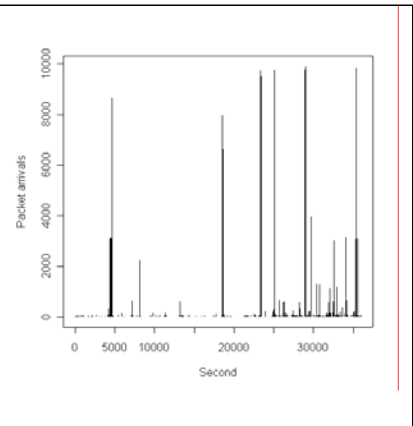
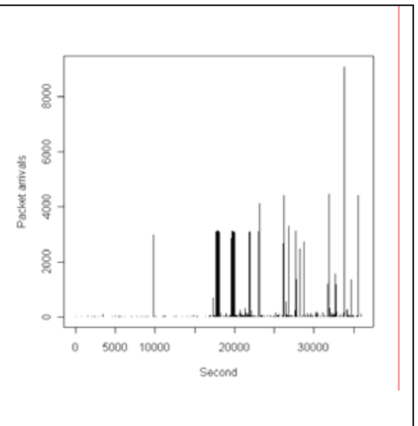
		
FTP packet arrivals of the UCF traffic	FTP packet arrivals of simulation #1	FTP packet arrivals of simulation #2

Figure 44: FTP packet arrivals

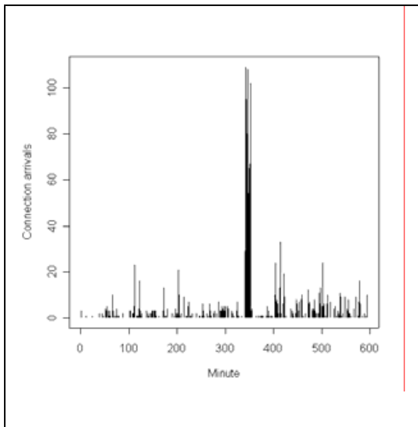
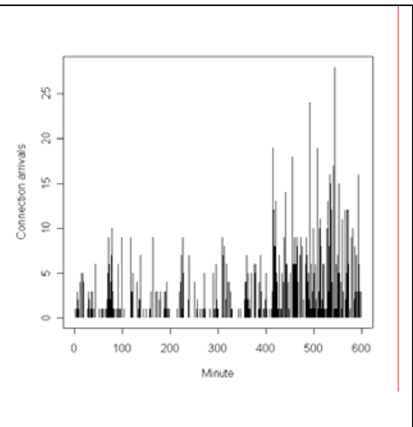
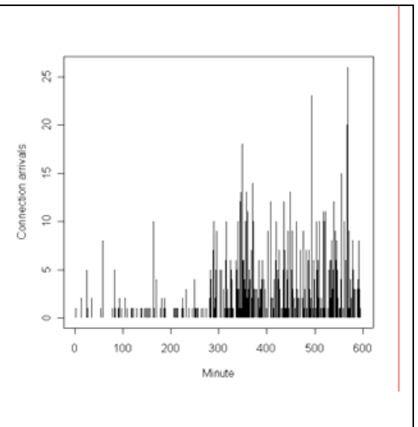
		
FTP connection arrivals of the UCF traffic	FTP connection arrivals of simulation #1	FTP connection arrivals of simulation #2

Figure 45: FTP connection arrivals

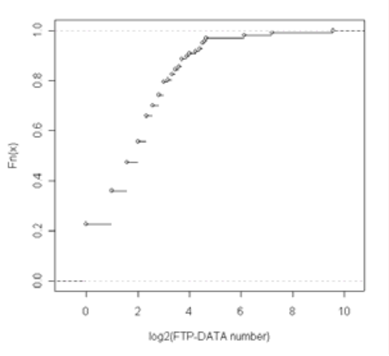
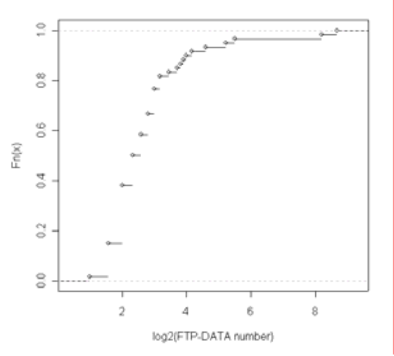
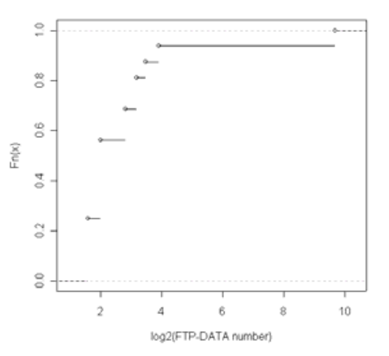
		
CDF of FTP-DATA number of the UCF traffic	CDF of FTP-DATA number of simulation #2	CDF of FTP-DATA number of simulation #2

Figure 46: CDF of FTP-DATA numbers

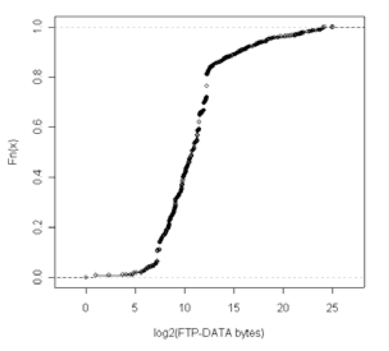
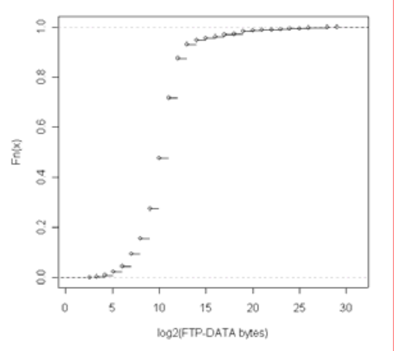
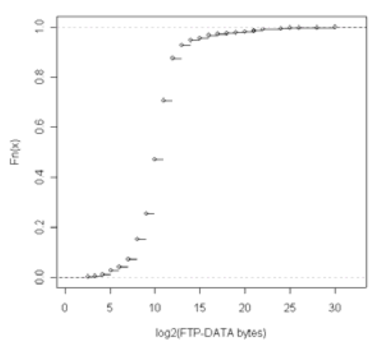
		
CDF of FTP-DATA bytes of the UCF CS traffic	CDF of FTP-DATA bytes of simulation #1	CDF of FTP-DATA bytes of simulation #2

Figure 47: CDF of FTP-DATA bytes

5.4 Conclusions

In our experiments, the UCF data set was simulated by HONEST with two different network settings, and the results were compared against the original traffic traces. The results are not satisfactory when we examine some key features of the traffic traces, such as autocorrelation structure and self-similarity. The reasons are complex. Many problems can be explained by the limitations inherent in HONEST.

For example, because HONEST is still in its preliminary stage, the number of physical computers and network interfaces that were used is low. Although a single computer can simulate multiple IP addresses, there is an actual maximum limit due to available computing power. In this situation, simulating traffic of a relatively large network, such as UCF CS LAN, is not actually suitable. Another example is the deployment of target files on servers. Some protocols, including HTTP, FTP, and SSH, need target files deployed in advance on the server's file system, so that the clients can generate network traffic by retrieving these files from servers. In our experiments, target files are created with a number of fixed sizes for the simplicity of deployment. But for CS LAN data, the file size has a much wider variety. The consequence is that, for simulation, the distribution of connection size may obviously deviate from the measurement.

With the above exceptions our experiments proved that HONEST is a useful and working tool allowing a network simulation to run in a real laboratory environment. More to the point, the experiments also proved that our protocol models are able to generate traffic that reasonably matches the measured CS background (within the limitations of HONEST).

CHAPTER 6: INTRODUCTION TO NETWORK INTRUSION DETECTION

In the following section, we look back the history of computer security and the development of Intrusion Detection Systems. The early and current state of research on Intrusion Detection will be reviewed, and the relevant IDS products will be discussed.

6.1 Network Security Becomes a Big Challenge

The CERT Coordination Center of Software Engineering Institute at Carnegie Mellon University has been collecting computer intrusion incident data from 1988. Since then, computer systems and the Internet have changed substantially. More networks are interconnected, new applications and protocols are implemented, the number of Internet users and the amount of information on the Internet is exploding. These changes result in an exponentially increased number of reported intrusion incidents. Figure 48 shows the number of incidents reported to CERT from 1988 to 2003 [14].

The rapid increase of intrusion incidents is greatly due to widespread use of automated and high-speed attacking tools. Attackers usually use scanning tools as a first step in launching an attack. Previously, vulnerabilities were discovered after the completion of a long-range scan (at the port or host level). Now, advanced attacking tools can exploit vulnerabilities as part of the scanning activities, which speeds up the attack. Also, attacking tools are become more automated. In the past, most attacking tools required a human to manipulate and initiate a new attack cycle. Today, many attacking tools can initiate new cycles themselves, without human intervention.

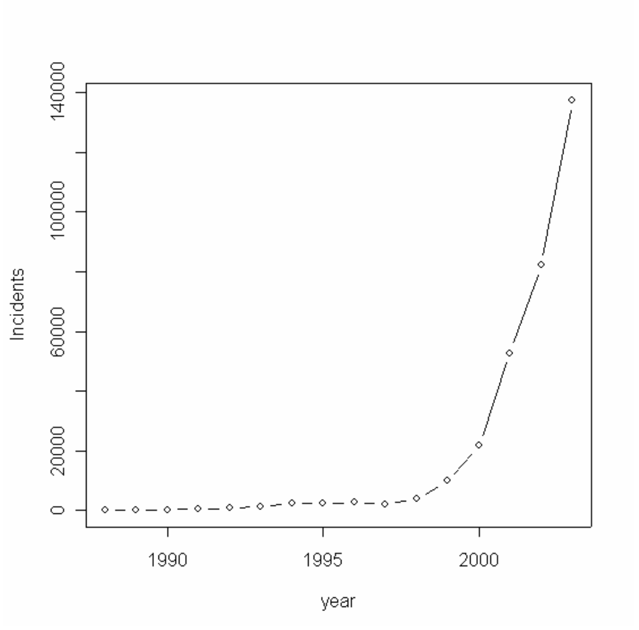


Figure 48: Incidents reported from 1988 to 2003

Now, attackers can distribute their automated smart attacking tools all over the world, and are able to launch a global attack quickly. This is a real nightmare for every computer system/security administrator. On July 19, 2001, a worm "Code Red" [13] and its variants infected more than 250,000 systems in just 9 hours. The "Code Red" worms propagated to other systems automatically and waited for the attacking date. On the specified date (July 19, 2001), worms distributed all over the world launched attacks simultaneously and caused a traffic flood to specific targets.

Attacking tools are also becoming more advanced. In [5], 3 advanced features of the modern attacking tools are summarized. First, current attackers are good at hiding the nature of attacking tools. Security experts have to spend more time to analyze attacking tools and understand the rapidly developing threats. A prevention and cure solution usually cannot be obtained immediately after a threat has been discovered. Second, today's attacking tools commonly have dynamic behaviors, which make it more difficult to detect them. They can vary their patterns and behaviors based on random selection, a predefined decision path, or through direct intruder management. Third, many of today's attacking tools have a feature of modularity, which means they can change themselves by replacing or upgrading a portion of the tool. In an extreme case, this causes polymorphic tools that self-evolve to be different in each instance.

With the Internet growing rapidly and with easy access to advanced attack tools, the Internet security situation is worsening. In 2004, because attacks against Internet-connected systems have become so commonplace, the counts of the number of the incidents reported lost relevance with regard to assessing the scope and impact of attacks. Thus, CERT/CC ceased to publish the number of incidents reported, and chose to develop and report other more meaningful metrics, such as "E-Crime Watch Survey".

"2004 E-Crime Watch Survey" is a project conducted by CSO magazine, United States Secret Service and CERT/CC. The survey shows a significant number of organizations reporting an increase in electronic crimes (e-crimes) and network, system or data intrusions. Forty-three percent (43%) of respondents report an increase in e-crimes and intrusions versus the previous year and 70% report at least one e-crime or intrusion was committed against their organization. Respondents say that e-crimes cost their organizations approximately $666 million in 2003.

The job of securing computer systems enclosed by the Internet has become a true challenge for everyone who wants to use the resources and functions provided by the Internet.

6.2 The Early and Current State of Network Intrusion Detection

Because traditional firewall techniques cannot provide complete protection against attacks, an Intrusion Detection System (IDS), as a necessary component of defense-in-depth, attempts to detect (and possibly counteract) unauthorized activities in a computer network or on a host computer. The goal of an IDS is simply to identify all attacks present without generating false alarms [96]. The false alarm constraint is critical because a low false alarm rate can still obscure valid alerts when applied to high data volumes.

6.2.1 Early Work on Computer Security and Intrusion Detection

The possible earliest work about computer security was done by Saltzer and Schroeder in 1974 [40]. They described a number of important design principles and protection mechanisms that are necessary for the protection of computer system resources and data. They introduced standard terminology, provided a detailed look at the fundamental access control features, and surveyed the research projects and working systems available at that time.

In [38], Anderson proposed a scheme to classify various threats and attacks on computer systems into 3 main categories:

- an *external penetration* by an unauthorized person

- an *internal penetration* by an individual who is authorized to use certain parts of the systems, but who is attempting to use resources that he is specifically not authorized to use

- the *misuse* of resources by an individual that is authorized to use them, but not in the way or for the purpose that he/she is using them

Among these three kinds of threats, the “misuse” type by one having supervisor or system-programming privileges is the most difficult to detect. These kind of intruders can easily modify or erase the traces he leaves, and to keep his intrusion under cover. As a solution, [38] suggested that an independent auditing on the major components of the system should be carried on by an external unit, which is not under the intruder’s control.

Anderson also noticed that system auditing may produce a large amount of information that would be impossible to analyzed manually. To solve this problem, he suggested the security analysis should rely on knowing the “normal” usage pattern of each legitimate user. It would be possible that the security system could filter out only those normal patterns, and block any “abnormal” usage. The normal patterns can be built on a number of measures, including the user’s logon patterns, file access statistics, and I/O instances. Although Anderson’s work was limited and did not give details on how to detect abnormal patterns and intrusions, it was still a meaningful guideline for the future research on computer security and intrusion detection.

Two formal security models, the Bell-LaPadula model and the Denning's model, were developed in the 1970's and the 1980's, and have been serving as the basis for most intrusion detection research and products.

The Bell-LaPadula model was presented in a report by Bell and LaPadula [18] that was a result of a series of research projects commissioned by the Electronic Systems Division of the U.S. Air Force. The Bell-LaPadula model is the first formal model of computer security. In the Bell-LaPadula model, data or resources are defined as objects, programs or users as subjects. The model defines a set of restrictions on access to objects (data or resources) at different security levels. Subjects are able to access objects only if their security level meets a set of conditions that implement the security level. The Bell-LaPadula model was used in the document "Trusted Computer System Evaluation Criteria [60]" as the standard of computer information security, published by the Department of Defence, 1985.

Another famous model was proposed by Dorothy Denning [19]. Denning's model was based on the idea that intrusions can be detected by recognizing abnormal patterns of behavior in a computer system's audit records. This model was proposed to specifically address the design of a real-time intrusion detection system using the techniques of expert systems.

In Denning's model, profiles are created for each legitimate user that describes the user's normal behavior pattern. The profile is built from the system's audit records on the actions of subjects upon objects. A number of metrics and statistics models can be used to create a meaningful profile of a user's past behavior. Denning also provided a detailed description of the profile's structure. When a system is running, generated audit records are compared to existing profiles. If there are any records that do not match a known profile, they are considered to be anomalous, and an "anomaly record" is created in a security log. If needed, further action can be applied, such as production of summary reports, profile updates or alerts to system administers.

During the 1980's and the early 1990's, some research projects and products of Intrusion (or Misuse) Detection systems were developed. Haystack [78] is a host-based audit trail analyzer for use on a multi-user system. It uses both signature detection and a simple statistical anomaly detector to find events of interests to security officers. It mainly focuses on misuse by internal users. Wisdom & Sense [37] is one of the first ID systems to determine behavior criteria automatically. It has two main components: the "wisdom" component and the "Sense" component. The "wisdom" component performs an analysis on audit files that record past activities, and produces a set of rules that describes the normal behavior pattern of the system and each user. Once the behavior patter is determined, the "Sense" component (usually an expert system), will check the recent audit log, and produce a score which indicates the likelihood that the user's activities are anomalous enough to indicate a misuse. The Information Security Officer's Assistant (ISOA) [43] combines the real-time analysis of the user and host activity with a more detailed analysis of an entire session, usually preformed when the session ends. With the help of information provided by analyzing the audit records of complete sessions, ISOA is able to describe a user's behavior pattern more accurately. The Intrusion Detection Expert System (IDES) [35] is also one of the first ID projects. The IDES is based directly on Denning's model [19], and developed with the idea that a user's behavior does not change much over time and a model of that behavior can be used to detect abnormal behaviors, either caused by users who attempt to misuse resources or by another individual who attempts to masquerade as the user. The statistical analysis component of IDES develops a profile for 'normal' activities for each user. The profile is based on a deep statistical analysis of the user's past audit records. The IDES will give a score to each new audit record based on profiles, and the score value indicates the degree of abnormality.

6.2.2 The Current State of Intrusion Detection

Since the middle of the 1990's, with the rapid growth of networking and the Internet, computer systems and users have to face threats and attacks more from the external world: attacks from other systems carried by the Internet. ID systems also have been developed into new generation with advanced abilities. These new ID systems usually have multiple sensors, can collect data from a distributed system or a network environment, and detect intrusions using both signature-based and anomaly-based algorithms. More important, these systems focus more on detecting intrusions originated from outside the local network, not only misuses by insiders.

6.2.2.1 Techniques

Even today, many ID systems are still using Denning's model as the base. Profile of a user or system's normal behavior is built not only from system audit records, but also from network packets, or any other observable trail of system activities. Bai and Kobaysashi [96] overviewed current techniques of intrusion detection and divided them into two categories: *Anomaly Intrusion Detection* and *Misuse Intrusion Detection*.

6.2.2.1.1 Anomaly Intrusion Detection

Anomaly Intrusion Detection uses deviation from the established normal usage patterns to identify intrusions. Anomaly detection techniques assume that all intrusive activities are necessarily anomalous. Because of this, they generally alert administrators and do not immediately act to block traffic. Several major approaches to the anomaly intrusion detection systems are described below.

- Statistical Approach

ID systems implementing this approach create profiles of normal usage or behavior pattern before detection. Then the anomaly detector starts collecting data while the system continues running. The anomaly detector constantly generates the variance of the present profile from the original one. If the variance is statistically significant enough, an intrusion alarm will be sent to security officer.

The main advantage to statistical systems is that a statistically driven ID system can learn and thus be more sensitive than expert systems. However, this learning ability can be leveraged by intruders. Intruders can train the system by gradually changing their behavior until an intrusive activity cannot be detected.

- Predictive Pattern Generation

This approach of intrusion detection tries to predict future events based on the events that have already occurred.

The main advantages of this method are: First, rule based sequential patterns can detect anomalous activities that are difficult with traditional methods. Second, ID systems of this type are highly adaptive to changes, because bad patterns are continuously removed. Third, it is easier to detect users who try to train the system during its learning period.

- Artificial Neural Network (ANN)

The idea of this approach is to train the neural network to predict a user's next action or command. The network is trained on a set of representative user commands. After the training phase, the network tries to match actual commands with the user profile which is already present in the net. Any incorrectly predicted events actually measure the deviation from the established profile.

Advantages of using ANN include strong anti-noise ability with input data; their success does not depend on any statistical assumption about the nature of the underlying data; they are easier to modify for new users. The disadvantages of ANN include: a small window will result in false positives while a large window will result in irrelevant data and increase the chance of false negatives; the net topology is only determined after considerable trial and error; the intruder can train the net during its learning phase.

- Data Mining Based Intrusion Detection

As computer systems become more and resulting audit data become more volumnious, analytical approaches may be overwhelmed. Data mining based intrusion detection tries to use data mining techniques to discover consistent and useful patterns of system and traffic behavior and use these to compute classifiers that can recognize anomalies and known intrusions [91]. Data mining approaches have great potential to help alleviate the problem of automatically detecting anomalous patterns in a large amount of audit data.

The data mining approach also has its disadvantages and limitations. First, such systems have a higher rate of false positives than traditional signature based systems. Second, these systems have lower efficiency during the training phase and the evaluation phase. At last, data mining based systems require large amount of training data and are more complicated than existing systems.

- Data Fusion Based Intrusion Detection

Multi-sensor data fusion, or distributed sensing, is used to combine multiple and diverse sensors and sources in order to make more accurate inferences about events, activities and situation.

Input into data fusion based IDS consists of sensor data, commands and priori data from established database. For example, the system input would be data from numerous distributed packet sniffers, system log file, SNMP traps, user profile database etc. The outputs of data fusion based IDS can be the estimate of the identity of an intruder, the intruder's activities, and an evaluation of the attack behaviors, and so on. Tim Bass discussed how to apply multi-sensor and data fusion technology to an IDS [80].

6.2.2.1.2 Misuse Intrusion Detection

Misuse intrusion detection uses patterns of known attacks or weak spots of the system to match and identify attacks. Ideally, not only an attack itself, but also its variations can be detected by misuse intrusion detection. Anomaly detection systems try to detect the complement of bad behavior. Misuse detection systems try to recognize known bad behavior.

In most cases, it is not enough to have only anomaly detection systems, because they are ineffective to detect attacks from the inside. On the other hand, misuse detection systems cannot detect new and unknown attacks. Having both of them is a reasonable solution for many current IDSs.

Several major approaches to misuse intrusion detection are described below.

- Expert System

Expert systems are modeled in such a way as to separate the rule matching phase from the action phase. The matching is operated according to audit trail events. Expert system is formulated by security experts, who define rules to be used in the detection component. However, because the system greatly depends on security experts and need to be tuned manually, it is insensitive to new attacks.

- Model Based Intrusion Detection

Model Based Intrusion Detection assumes that certain scenarios can be inferred by certain other observable activities. If the system monitors these activities, it is possible that an intrusion attempt can be predicted by observing activities that infer the intrusion scenario.

This kind of system can predict the attacker's next step based on the intrusion model. These predictions can be used to verify an intrusion hypothesis, to take preventive measures, or to determine what data to look for next. However, there are some restrictions in such systems: (1) the patterns for intrusion scenario must be easily recognized. (2) The patterns must always occur in the behavior being looked for. (3) The patterns must not be associated with any other normal behavior.

- State Transition Based Intrusion Detection

A State Transition Based Intrusion Detection system is described as a state transition diagram. The system is in a valid state when it starts and changes from state to state while the system is running, if a given condition is satisfied. The final state of the transition diagram is the compromised state. With the technique based on state transition, it is possible to detect some unknown attacks. However, because attacks are only described by a sequence of activities, the system is unable to detect complicated attacks which cannot be described by a transition diagram.

- Pattern Matching Based Intrusion Detection

This approach encodes known intrusion signatures as patterns that are matched against the data. Systems implementing this technique are usually efficient because of a simple design. The main problem with this model is that it can only detect known attacks.

6.2.2.2 Products

In this section, we discuss several representative IDS products developed with current techniques. Many current IDSs use both a anomaly detection technique and a misuse detection technique.

NIDES

The Next-Generation Intrusion Detection Expert System (NIDES) ([81], [16]) was developed at SRI as the successor to IDES. NIDES collects data from a number of hosts and performs an analysis in a central location. Like IDES, it uses two different detection techniques, statistical analysis and rule-based (expert system) analysis. To simplify processing and archiving data, each host's native audit records are converted into a standard NIDES audit format before transmission to the analyzer. NIDES can operate in either a real-time or batch mode. The batch option allows for a more detailed analysis and can be selected from the user interface.

Experiments have shown that NIDES can detect anomalies that IDES could not and that its performance, even when monitoring a large number of hosts, is improved. The original design of

NIDES did not include a network monitoring component, but the option of adding that feature was considered by the designers.

EMERALD

The system called Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD) ([68], [69]) is the successor to the IDES and NIDES systems, but it represents a different approach to the problem of detecting intrusions in a large network. Instead of designing a complete ID system, EMERALD intends to provide a framework within which modules can be added to solve specific problems.

The EMERALD framework includes a three-level design which can scale to any number of network nodes and hosts.

- **Service analysis level** – local, distributed EMERALD monitors collect and analyze data and communicate with peers
- **Domain-wide level** – supports separately-administered domains within an enterprise network and coordinates information from the EMERALD monitors at the local level to detect wide-spread intrusions or coordinated attacks
- **Enterprise-wide level** – gathers information from the domain level, looking for system-wide anomalous activities

The EMERALD monitor is not a specific piece of software; it is a conceptual object that can be implemented in many ways to support specific platforms or services. Each EMERALD monitor can be customized to a particular host platform or a network and solve a particular problem. Each EMERALD monitor must provide certain essential features and services, including an interface used to interact with the target being monitored; a profiler engine which performs anomaly detection using the audit information available to the monitor (host audit logs, network traffic or data from lower-level monitors); a signature engine that performs rule-based detection on filtered audit data; a universal resolver that correlates the results of both analyzer engines and communicates with other monitors.

The universal resolver is the most important analysis component of the EMERALD system. It uses a rule-based expert system to process reports and alerts from the local profiler and signature engine, and from other monitors in the system, to detect intrusions or misuse. It is also responsible for determining which information should be passed on to other monitors. A limited number of preprogrammed countermeasures are available for responding to attacks and the resolver determines if they are appropriate.

NSM

The Network Security Monitor (NSM) ([47], [11]) was developed by the University of California at Davis to detect intrusions by directly monitoring network traffic instead of analyzing host audit logs. Thus, NSM can monitor a heterogeneous system without having to directly support a variety of operating systems or platforms. However, this means that NSM is limited to detecting only remote intrusions or misuses; it is unable to determine if an intrusion is attempted from the host's console or if malicious code is executing on a host.

NSM uses a layered model, called the Interconnected Computing Environment Model (ICEM) to determine the overall state of the networked system. Packets received from the network are processed through an ICEM layer, which correlates related information and fills in data structures (referred to as *vectors*) that represent the behavior of a connection between hosts or the overall network behavior of each individual host. At the highest layer, the set of all the host and connection vectors represents the state of the entire system and is used by an expert system to perform intrusion detection.

The expert system has access to profiles of network activity that it uses to determine if an intrusion or misuse has occurred. These profiles include a list of the expected set of connection paths between hosts, lists of the connection types (telnet, ftp, mail, etc.) that should be found in normal network traffic, security levels assigned to each host and signatures of past known attacks. Based on the profile information and the current state of connections, a suspicious level is calculated and reported to security personnel.

Bro

The Bro IDS [82] uses passive network monitoring to detect intrusions in a local network. It operates in real time and, because it is a network monitor and not a host monitor, it neither places a performance burden on hosts nor requires installation on or modification of hosts. Furthermore, it is capable of monitoring a heterogeneous network from one UNIX workstation. Bro’s designer considered the possibility of attacks on the network monitor and provided some solutions to this problem. Bro’s design goals were:

- the system should be able to monitor high-speed networks with a large volume of traffic
- the packet capture process should not drop packets
- the system should provide real-time notification of intrusions
- a separation of the mechanism from the security policy
- the system should be extensible
- security policies should be simple and clear to reduce mistakes in implementation
- it is expected that the monitor will be attacked

Each packet is processed through three layers: packet capture and filtering, event detection and interpretation of the security policy for a given event. This layering supports several of the design goals listed above. The first three goals are aided by the separation of packet capture from processing. By handling event detection and policy interpretation in two separate layers, goals 4 and 6 are supported. The Bro language, used to create detection rules and the specific action that should be taken when each rule is triggered, fulfill the needs of the fifth goal. The layered approach also makes Bro more robust and several extra features, such as a *watchdog timer* which determine if Bro has stopped responding to input, make it more resistant to attack.

CSM

The Cooperating Security Managers (CSM) [31] provides a distributed, peer-to-peer approach to network intrusion detection. Each host in the system runs a copy of the CSM software and communicates with its peer hosts to both detect and respond to intrusions. When a user remotely accesses one host from another, the CSM on each endpoint of the connection will share information about the user, supporting user tracing and preventing masquerading or spoofing. System security operators can request a trace of user activities that will show when and where a user connected to each host in the system. Although the prototype of CSM was developed on UNIX hosts, it was designed to be platform and OS independent. The designers of CSM focused mainly on the cooperative aspects of the system and less on local intrusion detection or the operator's user interface. The cooperative approach is intended to allow a more scalable system and to discover intrusions that independent host monitors could not detect.

GrIDS

Its designers refer to GrIDS [79] as a "Graph-based Intrusion Detection System" for large networks. In this case, the term graph-based refers not to a graphical, i.e. visual technique, but to the use of graph theory to represent activity on hosts and the network traffic between them. By analyzing the resulting graphs, GrIDS can detect security violations. This technique is particularly suited to attacks or misuse activities that are propagated throughout a network or involve a large number of hosts. GrIDS is organized as a hierarchical system. The lower levels detect local intrusions and report them to higher levels where large-scale intrusions can be detected by correlating information from several local sources. GrIDS uses rule-based detection and can determine whether to take specific actions at each level or to pass the result from that level up to the next level for further processing.

The FrIDS system includes several components: a software manger, a graph-building engine, a module builder and data sources. The software manger oversees the system's state and communicates with its distributed modules. The data sources are actually host-based modules that monitor the activities of each host or local network and report to the other components. The graph-building engine constructs an *activity graph* which includes a subset of the hosts in the network. Each node in the graph represents a host and the edges represent the specific connections between those hosts that are involved in the intrusion. The information used to construct a lower-level graph is received from the data sources in its local area and the graph-builder's results are reported to the next level of the hierarchy. This continues up to the highest level where an activity graph is constructed that shows the overall extent of detected intrusions.

STAT and NetSTAT

STAT (the State Transition Analysis Tool) [44] was designed to be one detection component within a host-based ID system. It was expected that the ID system would include a profile-based anomaly detector to complement the rule-based state transition features provided by STAT. audit record would be collected by the system and converted into a format that STAT could process and STAT would deliver its results to the security officer's interface so that action could be taken.

NetSTAT [30] is a distributed, network-based intrusion detection system that monitors network traffic for attacks and abnormal behavior. NetSTAT has access to a *network fact base* that includes information about the network topology and available services. An *attack scenario* database contains state transition information about known network attacks. That information is represented as a series of transitions between known states that are triggered by *signature attack*, i.e., known events. When a signature action is detected, NetSTAT triggers the appropriate state transitions. If a series of signature actions that lead to a known compromise are detected, the set of state transitions associated with that compromise will each have been visited in turn and the end state will have been reached, indicating that the attack occurred. Thus, just as a specific sequence of rules in a expert system can be used to indicate the complex series of events that lead to an intrusion, the state transition described in the scenario database can model complex attacks and intrusions.

SHADOW

SHADOW and Snort, which will be introduced later, are two open-source ID systems. These ID products are developed with contributions from the whole open-source community and are widely supported by many third-party plug-ins to enhance their performance or functionality. Also, because of large number of users, new attacks can be reported promptly and corresponding rules can be added to the system in a short time.

The SHADOW system was built by the staff of the Naval Surface Warfare Center, based upon their many years of experience as intrusion analysts [76]. SHADOW [95] is an open source collection of scripts and programs that supports intrusion detection and analysis at low cost. The minimum SHADOW system is composed of a *sensor*, usually located outside of the firewall where is exposed to attack traffic, and an *analyzer* which should be inside the firewall. Data collected by the sensor is transmitted to the analyzer where it is filtered by a set of scripts that search for intrusive or suspicious behavior. The results can be monitored via a web browser.

Snort

Snort [57] is a packet filtering and inspection program that can be use either as one component in a more complex ID system or as a stand-alone, lightweight IDS. Snort is freeware and open source community provides strong support for it in many ways. There are more Snort detection rules for known attacks than for any of the commercial ID systems [76] and features like SQL database support, web-based interfaces, data-mining software and professional technical support are all widely available.

Snort uses a rule-based detection engine to detect known intrusions or attacks. Rules are written in a specific, simple notation and output can be to the screen, to log files, to a database or to another IDS component via syslog or email messages. In addition to examining packet header fields for anomalies or suspicious activities, Snort can look into the packet's payload to find data that indicates hostile activities such as buffer overflow, CGI script exploits or macro virus code. Snort's packet capture engine is fast enough to avoid dropped packets, even in busy networks.

Snort can be used as the packet logging component for a more sophisticated IDS, such as the SHADOW system, where Snort finds known attacks and intrusions using its detection rules and passes the captured packets to SHADOW for further investigation. If SHADOW finds an unknown intrusion, the information that SHADOW found can be used to create a new rule for Snort and added to its detection engine. The combination of two different detection techniques, similar to the configuration of many of the research ID system described above, makes for an effective and efficient intrusion detection system which can be based entirely on open source software [3].

6.3 Limitations of Current Intrusion Detection Systems

6.3.1 The False Alarm Problem

A false alarm happens when the ID system reports an intrusion alarm but actually there is no attack. It becomes a problem when the false alarm rate is too high that the security officer is overwhelmed by alarming information, and the true intrusion activities are not noticed.

In 2002, Network World conducted a test on seven leading commercial IDS systems plus the open source system Snort, with the purpose of reviewing major IDS products and testing their reliability and performance [24]. The tests were run on Opus One, a relative small ISP in Arizona that provides service for about 50 small to medium-sized businesses. The Opus One backbone bandwidth is about 12 Mbps. The tested ID systems were not stressed. However, the testing result was not encouraging: Several IDSs crashed repeatedly under the burden of the false alarms they churned out; when real attacks came along, some products didn't catch them and others buried reports so deep in false alarms that they were easy to miss; overly complex interfaces made tuning out false alarms a challenge.

In 2003, Network World did a similar test [42] and found that the false alarm was still a big problem for IDSs. Although tested IDSs were improved on the way they manage and present the alarm output, high volume of alarm messages made system sluggish and ill-behaved. In some cases, alarms produced by IDS used up the hard disk and the whole system were unusable.

6.3.2 Problems with High-speed Network and Real-time Response

Network World also performed another test on IDSs in 2002 focused on the performance on Gigabit Ethernet [12]. The first step was to run 28 well-known attacks against each product on a wire that had no other traffic running on it, then compare the detection result with that obtained under Gigabit traffic (970Mbps). It turned out the most products downgraded their performance by averagely 50%. When under Gigabit network traffic, they only detect half the same attacks as they could under light traffic.

A relevant problem is the response time for on-line real-time IDS. Unlike off-line IDS, a real-time IDS needs to recognize and response to attacks immediately, especially when the attacks are against the IDS itself. When the volume of attacks is very high, real-time IDSs may not be able to perform the matching for all rules, or they cannot response in the pre-defined time. A scheme is proposed in [90] that IDS monitors the event rate. When the event rate is below a threshold, the IDS runs at a *quiet* mode and all rules are utilized to cover all attacks. If the event rate increases beyond the threshold, the IDS only uses rules which are deemed as critical. This scheme can protect IDS from overwhelmed under heavy attacks and guarantee it responses in real time. However, real-time IDSs under this scheme could possibly pass some attacks into the defense line.

6.3.3 Problems with Training Data

As we have seen from previous sections, many anomaly-based IDS need a 'normal' data to train the system. The training data can be audit records, network traffic or any other trail of user's activities. No matter what it is, the training data is generally required to be attack-free. To let an IDS work in a real environment, the best way is to train it using the real data, for example, real traffic packets. However, when using the real traffic packets, IDS administrator is facing a big problem: even though all known attacks can be removed from the traffic data, there could be unknown attacks in it. As a result, the trained IDS cannot detect them in the future.

6.3.4 Problems with Management Interface

IDS is a complex system and usually requires more time than security administrators have available to devote to their proper use [7]. IDS administrators have to deal with multiple data input sources, manage detection rules, monitor high volume of alarms and determine proper action etc. Results of a survey into the state of security management in a variety of companies and institutes clearly show that the state of network security management is poor [41]. This is partly because of lack of good tools or management interfaces to administrators. Testing experiments of [24] and [12] also confirm that the bad user interface decreases usability of ID systems.

CHAPTER 7: BUILDING AUTOCORRELATION TEMPLATES TO DETECT DOS ATTACKS

In this chapter, we first introduce a theory of network intrusion detection that is based on the analysis of the correlation structure of network traffic. The theory is designed to achieve high detection probability and low false alarm rate. A successful implementation of this theory relies on an accurate template of the autocorrelation of "normal" traffic, and the paper that introduced this theory [54] did not treat the problem of obtaining such a template. Here we offer an approach for constructing and updating templates based on data collected in sliding traffic windows.

7.1 Denial-of-Service Attacks

The term *denial-of-Service* (DoS) is used to describe network attacks where legitimate users are denied services or access to information resources [20]. Allen [86] divides DoS attacks into two categories: DoS-SE and DoS-TE.

In DoS-SE (DoS system exploit) attacks, attackers create one or a few specially-crafted packets that exploit some weakness in the target's operating system, network protocol stack, or software application(s). These packets cause the executing code to enter an unanticipated state or to overwrite critical data, and the result is software failure. In August 2001, a DoS-SE attack "Code Red" was reported to CERT/CC [15], [13]. The "Code Red" worm can crash Microsoft IIS 4.0 server by sending a crafted URL if the server is configured to use URL redirection.

DoS-TE (DoS traffic exploit) attacks take a different approach. DoS-TE attacks generate a flood of packets that overwhelm the target's ability to process them. This type of attack may rely on brute force to disrupt the target's services, or the attack may leverage known target vulnerabilities. In the latter case an attack may still be classified as a traffic exploit if it relies on the volume of the traffic to achieve its end. One example of a DoS-TE is the denial-of-service attack on domain name servers that occurred in April 2000 [21]. In that attack, intruders sent a large number of UDP-based DNS requests to name server(s) using a spoofed source IP address. Any name server response was sent back to the spoofed IP address as the destination. In this scenario, the spoofed IP address was overwhelmed by flood of responses forwarded by one or more name servers, and represented the victim of the denial of service attack. The name server is an intermediate party in the attack. Other DoS-TE attacks include *apache2*, *smurf*, *UDP storm* and *Neptune*.

In addition to attacks discussed above, network scanning is another common type of intrusion and may be considered to be a DoS attack. Typically a scan gathers information about the network's configuration or probes for the existence of host vulnerabilities. A scan can be stealthy, meaning that it leverages relatively few packets or spreads its packets over a long time period to avoid notice. On the other hand, "brute-force" scans generate large numbers of packets during relatively short periods intending to occupy the target so that it cannot perform its intended functions.

7.2 Decision Analysis of Network-based IDSs - The Theory

While network intrusion has become a real challenge to security community, network-based ID systems are suffering from high false alarm rates and from other problems [24] [42] [12]. In theory, whatever the approach, each intrusion detection technique can be characterized by curves showing probability of detection, *pd*, versus probability of false alarm, *pfa*. A good IDS should achieve a high *pd* and a low *pdf*.

7.2.1 Basic Idea

In a paper by Li, Jia, and Zhao [54], the author offers a mathematical approach for intrusion detection that yields high probability of detection and low probability of false alarm. In this section, we give an introduction to their theoretical approach.

Let the time series $x(t_i)$ represent the attack-free portion of an organization's background traffic; specifically let $x(t_i)$ give the number of bytes arriving at time t_i. Let $y(t_i)$ represent total bytes arriving, then $y(t_i)$ can be represented by

$$y(t_i) = x(t_i) + n(t_i) \ , \qquad (7\text{-}1)$$

where $n(t_i)$ represents bytes arriving from any attack traffic that may be present. Based on the assumption that the $x(t_i)$ and $n(t_i)$ are statistically independent, the following formula holds:

$$r_y(k) = r_x(k) + r_n(k), k \in I \ , \tag{7-2}$$

where $r_y(k)$, $r_x(k)$ and $r_n(k)$ are autocorrelations of $y(t_i)$, $x(t_i)$ and $n(t_i)$ respectively. The work described here is based on detecting differences among these autocorrelations; in particular, r_x serves as the template of normal traffic. The distance

$$\xi = \|r_n\| = \|r_x - r_y\| \tag{7-3}$$

is used as the measure of deviation between the normal and abnormal traffic. A threshold V is determined so that, if the distance is larger than V, the IDS will determine that there is a DoS intrusion; otherwise, the IDS will treat the traffic as benign.

Figure 49 depicts the basic architecture of the system.

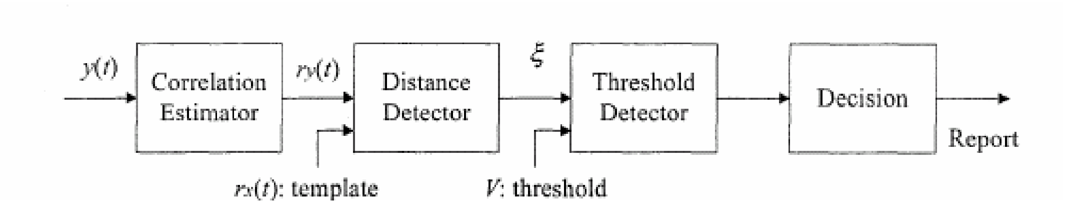


Figure 49: Diagram of IDS architecture

7.2.2 Traffic Template

Obviously, successful intrusion detections rely on the availability of a useful template of normal traffic $r_x(t)$. In fact, the effectiveness of the theory depends on the accuracy of such a template. As noted, however, the original research did not address the template estimation problem.

The self-similarity feature of network traffic has been widely studied ([92][85][52][45] et. al.) and accepted as elemental for network traffic. As noted in [39], x is called asymptotically second-order self-similar with Hurst parameter H between 0.5 and 1, if its autocorrelation, in the discrete case, satisfies

$$r_k \sim ck^{-\alpha}(k \to \infty), \alpha \in (0,1)\,, \tag{7-4}$$

where c>0 is a constant, and α=2-2H.

However, equation (7-4) is only a qualitative expression of autocorrelation function of a self-similar process, and it is too rough for use in pattern matching. Li and his colleagues give an exact autocorrelation function for self-similar network traffic in [53], and apply it in network traffic simulation [55] [56]. The autocorrelation function given in [53] is

$$r(k) = (|k|^{\alpha} + 1)^{2H-2}, \alpha \in (0,1], H \in (0.5,1) \tag{7-5}$$

Li uses expression (7-5) to obtain autocorrelations of normal and abnormal traffic, and then calculates distance between these two.

7.2.3 Distance Measures and Probability Distributions

The theory needs two distance measures to achieve intrusion detection. They are

$$\xi = \left\| r_x(k) - r_y(k) \right\| \text{ and} \tag{7-6}$$

$$\zeta = \left\| r_x(k) - r_{xl}(k) \right\| \tag{7-7}$$

In (7-7), r_{xl} stands for the autocorrelation obtained in the absence of attack traffic and not used as a template. Thus, the random variable ξ measures the distance between the normal traffic template and the abnormal traffic; while the random variable ζ measures the distance between the normal traffic template and attack-free traffic. Given a detection threshold V if ξ>V, IDS detects an intrusion successfully; if ζ >V, IDS gives a false alarm.

In [54], it is suggested that the distance be calculated using the Itakura-Saito norm [50], i.e.

$$\xi = \left\| r_x - r_y \right\| = \sum_k \left| \frac{r_y}{r_x} - \log\frac{r_y}{y_x} - 1 \right|. \tag{7-8}$$

The theory assumes that in detection, enough samples of $y(t_i)$ are arranged that ξ is approximately Gaussian, and we have

$$\xi \sim N(\mu_\xi, \sigma_\xi^2) = \frac{1}{\sqrt{2\pi}\sigma_\xi} e^{-\frac{(\xi-\mu_\xi)^2}{2\sigma_\xi^2}}, \tag{7-9}$$

where μ_ξ and σ_ξ^2 are the mean and variance of the random variable ξ.

Let

$$\Phi(t) = \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} dt . \tag{7-10}$$

Then detection probability P_d can be expressed by

$$P(V < \xi < \infty) = \int_{\frac{V-\mu_\xi}{\sigma_\xi}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} dt = 1 - \Phi\left[(V - \mu_\xi)/\sigma_\xi\right] = P_d . \tag{7-11}$$

Similarly, the probability of false alarm P_f can be expressed as

$$P(V < \zeta < \infty) = 1 - \Phi\left[(V - \mu_\zeta)/\sigma_\zeta\right] = P_f . \tag{7-12}$$

Note that μ_ζ and σ_ζ^2 are the mean and variance of the random variable ζ.

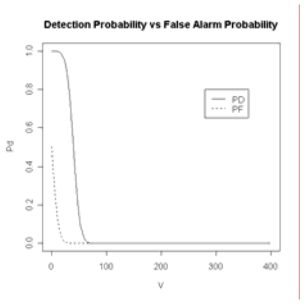
In Li's paper, the probability of a miss (failing to recognize a real attack) is also presented. Because the miss probability P_m=1-P_d, it is sufficient to discuss only P_d here.

The goal is that in the absence of attacks there should be no significant difference between the normal template traffic and total traffic received. Thus, μ_ζ=0 in most cases. In addition, the variances of ξ and ζ can be standardized such that σ_ξ=σ_ζ=σ. Then P_d and P_f can be given by

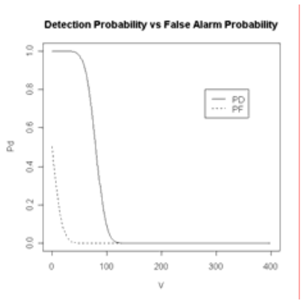
$$P_d = 1 - \Phi\lfloor (V - \mu_\xi)/\sigma \rfloor, \tag{7-13}$$

$$P_f = 1 - \Phi(V/\sigma). \tag{7-14}$$

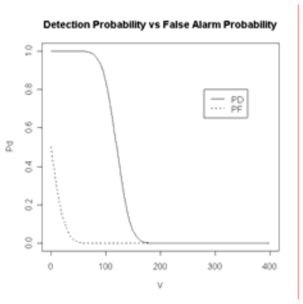
Figure 50 depicts curves of P_d vs P_f given ξ of three different normal distributions.



ξ~N(40,10)



ξ~N(80,15)



ξ~N(120,20)

Figure 50: Plots of P_d vs P_f

7.2.4 Selecting the Threshold

Ideally, we desire an IDS with P_d as high as possible and P_f as low as possible, i.e. $P_d \rightarrow 1$ and $P_f \rightarrow 0$ at the same time.

If we let d be the lower bound of P_d, then

$$P_d = 1 - \Phi(\frac{V - \mu_\xi}{\sigma}) < d. \tag{7-15}$$

Because

$$1-\Phi(\frac{V-\mu_\xi}{\sigma})=\Phi(\frac{\mu_\xi-V}{\sigma}),$$

(7-15) can be written as

$$\Phi(\frac{\mu_\xi-V}{\sigma})<d, \qquad (7\text{-}16)$$

and we have

$$0<V<\mu_\xi-\sigma\cdot\Phi^{-1}(d). \qquad (7\text{-}17)$$

Theoretically, we can take maximum value of d=1. If the precision of the normal distribution expressed by Φ is determined (for example, 4), in this case, we have

$$0<V\le\mu_\xi-4\sigma. \qquad (7\text{-}18)$$

We can also use the same method to investigate the relationship between P_f and the threshold V. Let f be the upper bound of P_f, we have

$$P_f=1-\Phi(V/\sigma)<f. \qquad (7\text{-}19)$$

Because

$$1-\Phi(\frac{V}{\sigma})=\Phi(\frac{-V}{\sigma}), \qquad (7\text{-}20)$$

we get

$$V \geq -\sigma\Phi^{-1}(f). \qquad (7\text{-}21)$$

If we set f=0 and we detect based on 4 standard deviations, we have

$$V \geq 4\sigma. \qquad (7\text{-}22)$$

Concluding the above, to let $P_d \geq d$ and $P_f \leq f$, the threshold v must satisfy the following condition:

$$V \in [-\sigma\Phi^{-1}(f), \mu_\xi - \sigma\Phi^{-1}(d)]. \qquad (7\text{-}23)$$

In the extreme case, when d=1 and f=0 and the precision of the normal distribution is 4, we have

$$V \in [4\sigma, \mu_\xi - 4\sigma], \mu_\xi - 4\sigma > 0. \qquad (7\text{-}24)$$

Note that the constraint of (7-24) is equal to

$$\mu_{\xi} > 8\sigma . \qquad (7\text{-}25)$$

In other words, to apply this detection theory and obtain high probability of detection and low probability of false alarm, two requirements must be met: (1) the mean of random variable ξ (distance between normal and abnormal traffic) must be at least 8 times its standard deviation; (2) the selected threshold V must be some value from the interval defined by (7-24).

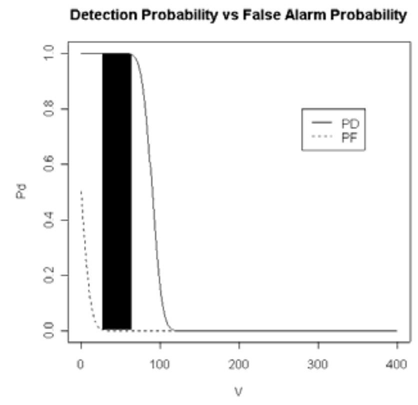


Figure 51: Detection area with ξ~N(90,10)

Figure 51 gives an example of selecting the threshold V with ξ~N(90,10). According to (7-24), extremely good results (high P_d and low P_f) can be achieved by selecting V from [72] [18]. But Figure 51 shows that useful results can be obtained by choosing values of V from the dark area. The dark area is called detection area in [54].

7.3 Building the Autocorrelation Template

Li et al. have provided a theoretical proof for the above method. They conducted experiments by assuming normal and abnormal traffic had particular autocorrelation functions, but the method has not been evaluated against real traffic data. Once we decided to apply Li's theoretical work and pursue a practical IDS, we found several major problems which hinder its implementation on real traffic: (1) The x values of the time series $\{x_i\}$ used by Li are the byte counts for each time increment, and the theory depends on the self-similar nature of $\{x_i\}$. However, when we tried to demonstrate self-similarity based on byte counts, it yielded undependable results; (2) the theory requires a template of traffic autocorrelation, but does not provide an approach for obtaining it; (3) the detection theory is constructed on the assumption that the norm $\xi = \left\| r_x(k) - r_y(k) \right\|$ will be normally distributed. But based on our analysis of the UCF CS traffic traces and several other data sets, we don't observe normal distribution for the norm ξ.

Among these difficulties, the most critical one is how to achieve a good template of the autocorrelation data. In this section, we demonstrate a method to construct the autocorrelation template using time series forecasting. The experimental results of this method show that in most cases, it forecasts correctly the autocorrelation structure for a period which contains a future period.

7.3.1 Method

We replace the original byte sequence with a packet arrival sequence $x(t_i)$, and calculate the autocorrelations of a sliding window on the arrival sequence. For each position of the sliding window, we calculate the autocorrelation functions using a suitable maximum lag value "max." The sliding window will run a finite number of steps to provide data for time series analysis. A time series is constructed by including ACF values corresponding to the same lag as computed across different steps. Hence, there are "max" different time series constructed. Then we build an ARIMA model for each time series to predict the ACF values in the future; each time series predicts a particular lag in a future window. The method is illustrated in Figure 52.

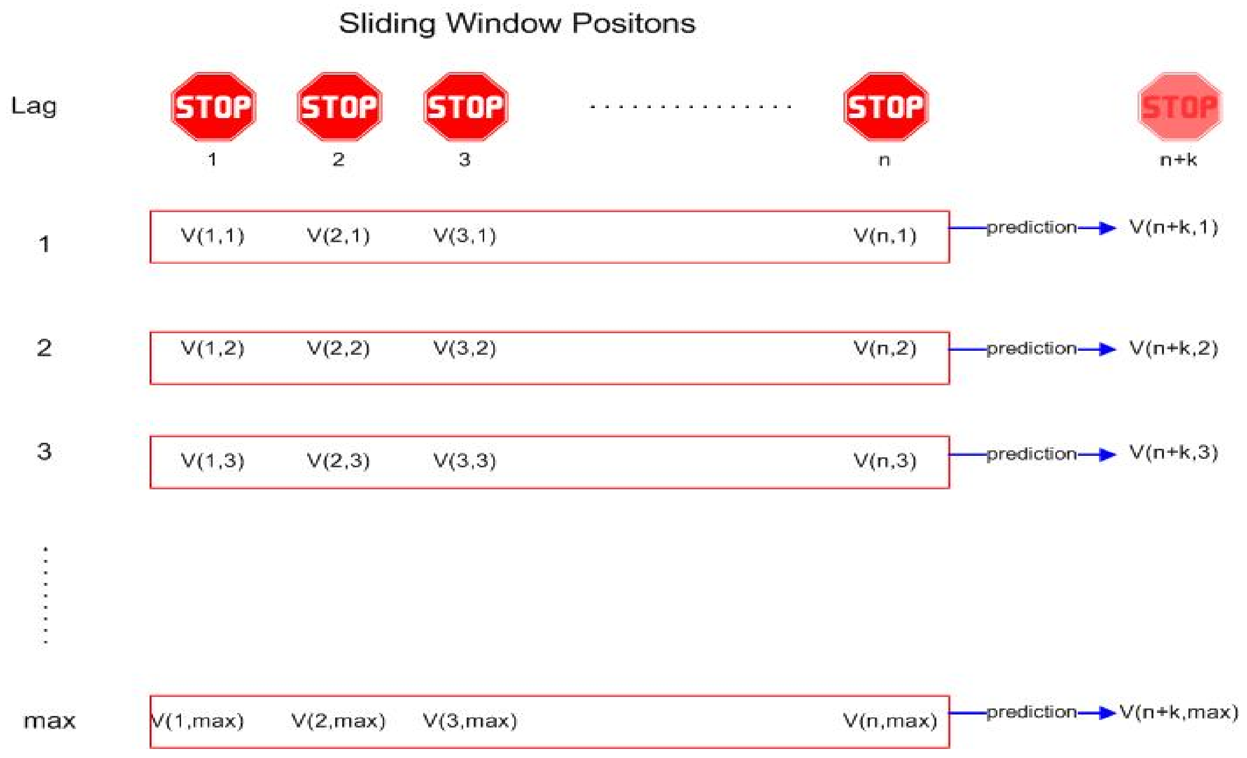

Sliding Window Positons
Lag
STOP 1
STOP 2
STOP 3
STOP n
STOP n+k
1
V(1,1) V(2,1) V(3,1) V(n,1) prediction V(n+k,1)
2
V(1,2) V(2,2) V(3,2) V(n,2) prediction V(n+k,2)
3
V(1,3) V(2,3) V(3,3) V(n,3) prediction V(n+k,3)
max
V(1,max) V(2,max) V(3,max) V(n,max) prediction V(n+k,max)

Figure 52: Building the template of autocorrelation function

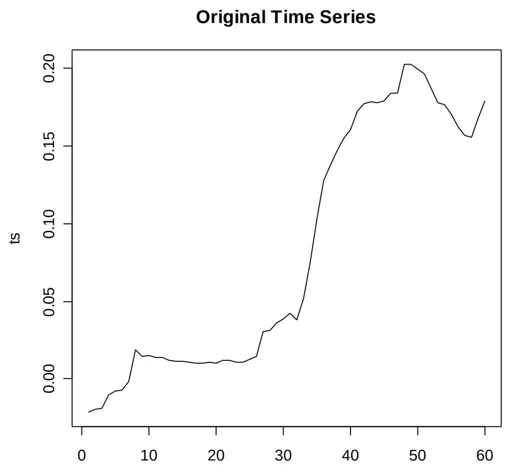
Using the above approach we can predict ACF values k steps ahead. The accuracy of the predictions is affected by several selections: the size of the sliding window; the size of the window's sliding step; the number of stops used for training; and the value of k, the number of steps we wan to predict ahead.

7.3.2 Experiment and Results

We captured all TCP headers in 5 days from 8am to 8pm at the Olin Engineering building of FIT. The sliding window size was set to one hour, or 3600 seconds. The step size was 1 minute, or 60 seconds. We used the first 60 steps for training, which means there are 60 elements in each time series. We set k=5; thus, the ARIMA model from each lag time series predicts 5 steps, or 5 minutes, ahead.

In building an ARIMA model one assumes that the time series of interest is stationary. Our program uses a time series linear model to judge if there is a trend. When a trend is present, the program will perform a differencing operation upon the time series, and attempt to eliminate the trend. After the first differencing, the program fits the differenced data to a linear model and tries to find and eliminate any existing trend. This procedure repeats until no significant trend exists.

Figure 53 depicts a time series with a trend and the resulting stationary process after two differencing operations on the original data. Our investigation showed that there usually was no seasonal component in the traffic time series; thus, it was not necessary to remove a seasonal component.



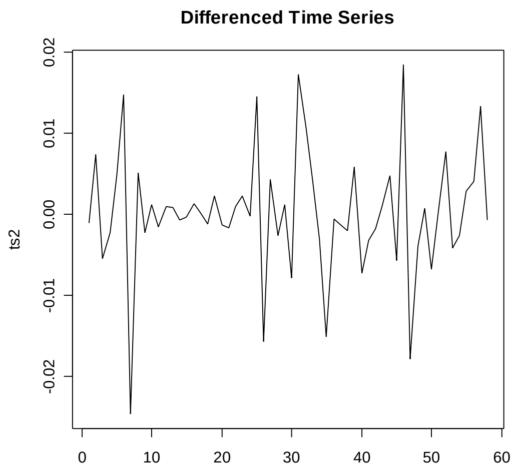


Figure 53: Obtain stationary process by differencing

After the stationarity of a particular time series is assured, the next step is to choose a proper ARIMA model for the time series. Choosing the best ARIMA model is a complex task. Here we follow a simple but efficient rule for building an AR model for each time series [67]. An AR(p) model is usually appropriate for a time series if its partial ACF function becomes zero after lag p. Our program calculates the PACF functions for every stationary time series and finds a proper p to build the AR model. Figure 54 shows the PACF plot for the above differenced time series. The PACF value of lag 1 in the plot lies outside the 95% confidence interval, thus, an AR(1) model should be proper for the time series, and is used to forecast the future values.

Figure 55 gives the prediction result for the time period from second 2041 to second 5460 of day 1. The black line represents the observed ACF structure; the red line represents the predicted ACF structure.

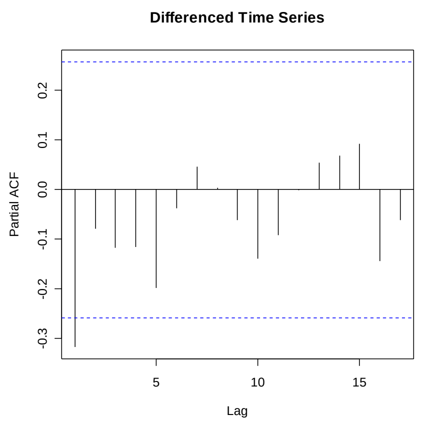


Figure 54: Partial autocorrelation plot

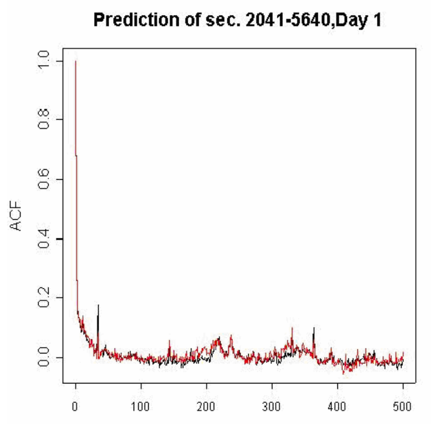


Figure 55: Predicted autocorrelation structure

Among the predictions of all 5 days, most predictions were a good match to the observed sample ACF. A small fraction of the results do not have a satisfactory visual match. According to our investigation, we found they are associated with significant changes in the traffic pattern. Because we only have the information from TCP headers (and not segment data), it is difficult to investigate the traffic data and determine whether these changes are due to attacks or due to particular valid applications. At any rate these changes, or the disagreement between the sample and the prediction, are definitely worth a security administrator's attention.

7.4 Summary

Implementing Li's theory of intrusion detection involves many practical issues, such as how to define and calculate norms, how to construct the key random variable which satisfies a normal distribution, and how to provide both normal and attack-free traffic. We have obtained a solution for the most challenging problem – that of finding a template of the background traffic.

This chapter developed an approach to solve the problem of building the autocorrelation template. The experimental results show that this approach is effective, in the sense that it produces matching predictions of autocorrelation structures at most times. In the instances that there is not a match between prediction and measurement, the differences are associated with significant changes in the traffic pattern. While more experimentation is needed, the approach for building an autocorrelation template appears quite promising.

CHAPTER 8: DETECTING DoS ATTACKS USING THE CORRELATION INTEGRAL

8.1 What is a Correlation Integral

A *Correlation integral* is defined by Grassberger and Procaccia [66] as a tool to calculate fractal dimension.

The concept of fractal dimension was formally brought forward by Mandelbrot [9]. Mandelbrot extended the Euclidean concept of dimension (i.e. the number of parameters needed to describe an object's shape or location in space) to include the dimension of fractal curves which usually do not have an integer number of dimensions.

Grassberger and Procaccia defined the *information fractal dimension*, and use correlation integral as a tool to calculate the fractal dimension. The correlation integral measures spatial correlation and it is based on the correlation between pairs of points in a fractal curve. The correlation integral is formally defined as:

$$C(r) = \lim_{N\to\infty} \frac{1}{N^2}\{the \quad number \quad of \quad pairs \quad (i,j) \quad whose \quad dis\tan ce \quad |X_i - X_j| < r\}$$

For a single-variable time series {x_i, i=1,…,N, x∈R}, the correlation integral is defined as:

$$C(r) = \lim_{N\to\infty} \frac{1}{N^2} \sum_{i,j=1,j\neq j}^{N} \theta(r - |x_i - x_j|)$$

where $|x_i - x_j|$ is the Euclidean norm and θ is the Heaviside function

$$\theta(x) = \begin{cases} 0, x < 0 \\ 1, x > 0 \end{cases}$$

C(r) measures the probability that the distance between any pair of points in the time series is less than r.

Grassberger and Procaccia also showed that for small r, C(r) grows according to a power law:

$$C(r) \sim r^{v}$$

The *correlation exponent*, v, is a measure of the local structure of the fractal curve and is an estimate of the correlation fractal dimension.

8.2 Related Work

Leland [92] and Willinger [94] discussed Fractal dimension of network traffic. In their work, the relationship between fractals and self-similarity were discussed, but the application of the correlation integral or fractal dimension of traffic was not involved.

Ayedemir et al. [49] showed that the correlation fractal dimension could be used to compare synthetic traffic to real traffic so that one could determine how accurately the synthetic traffic was generated. The researchers used a modified definition of correlation integral to derive local structure of short traffic sequences and to measure the traffic's fractal dimensions. Based on stochastic analysis, they claimed that the correlation dimensions of independent traffic sequences can be compared to determine if their behaviors are similar. The authors did not discuss applications to network intrusion detection.

Allen [86] proposed an intrusion detection technique by comparing fractal dimensions between normal network traffic and the traffic being attacked. In his method, traffic fractal dimensions are calculated for fixed-sized time segments both for investigated traffic and a traffic template. The traffic template should possess characteristics of typical normal traffic, or traffic not being attacked. When the fractal dimension measurements of investigated traffic segments are obtained, they are compared to the measurements of the template using an F-test. DoS attacks are indicated by a low p-value.

Allen's method relies on the availability of a good traffic template; obtaining such a template was left as an open problem. Again, this work uses changes in fractal dimension for detection alerts.

8.2.1 Correlation Integrals of Network Traffic

In this section, we introduce an intrusion detection approach using the correlation integral, and we provide a method for estimating an appropriate template for attack-free traffic. The template is derived from real traffic traces, and in a general form. We also report the preliminary experimental result of applying this technique in detection of DoS-TE attacks.

As discussed in section 4.1, a correlation integral C(r) measures the probability that the distance between any pair of points in the time series is less than r. If we consider a network traffic sequence as a time series $\{x_i$=packet arrivals in the i^{th} time unit$\}$, then C(r) of $\{x_i\}$ is greatly influenced by any relatively large values (bursty arrivals) in the sequence. According to the definition, two sequences may have same correlation integral measures, even though they have significantly different total packet sums or average packet numbers per time unit. The factor that really matters is the distances between pair of points. The network traffic's self-similarity and bursty nature make it reasonable to assume that correlation integral might be the right building block for a new detection method.

In the following section, we use the UCF CS traffic traces to demonstrate the procedure.

8.2.2 Constructing a Traffic Template

The UCF traffic dataset includes Internet traffic traces captured over 10 hours (from 8am to 6pm). To get attack-free background traffic, we removed obvious attacks by running an Intrusion Detection program (Snort) on the traces and filtered out all packets except TCP packets that did not match Snorts malicious signatures. Then packets of 5 major Internet protocols (FTP, HTTP, SMTP, POP3, SSH) were extracted from the traffic traces. The extracted traces were sent to the NS-2 simulator introduced in Chapter 4, and the traffic traces produced by the simulation were used to construct the traffic template. In this was we guaranteed that the template is built from attack-free traffic having similar correlation integral structure to that of the UCF CS traffic. We counted the packet arrivals each second in the simulation traffic, and obtained a time series $\{x_i, i=1,\ldots,36000, x\in I\}$. x_i is the number of packets arrival in the i^th^ second.

We use a slightly modified definition correlation integrals for the simulation traffic.

$$C^{'}(r) = \lim_{N\to\infty} \frac{1}{N^2} \sum_{i,j=1,j\neq j}^{N} \theta(r - |x_i - x_j| / D)$$

The distance $|x_i - x_j|$ is defined as the difference in the number of packets between the ith and the jth second. D is the maximum distance found between any two x_i, x_j values. We introduce D into the formula so that r only takes values in the range [0,1].

Because the time series $\{x_i\}$ has 36000 elements (36000 seconds), it is computationally infeasible to calculate correlation integrals based on the entire dataset. We divide the dataset into 10 non-overlapping subsets, and each set corresponds to 1000 continuous seconds of time. We calculate C'(r) for each subset with r=0.005,…,0.1. r increases with a step of 0.005. Thus, for all 10-hour traffic traces, we get 36 vectors of correlation integrals:

$$V_1 = \{C_1^{'}(0.005), C_1^{'}(0.01), ..., C_1^{'}(0.1)\}$$

$$V_2 = \{C_2^{'}(0.005), C_2^{'}(0.01), ..., C_2^{'}(0.1)\}$$

…

$$V_{36} = \{C_{36}^{'}(0.005), C_{36}^{'}(0.01), ..., C_{36}^{'}(0.1)\}$$

The traffic template V_t is a vector similar to the above, but its element at a particular position is the mean of elements at the same position of all 36 measured vectors.

$$V_t = \{C_t^{'}(0.005), C_t^{'}(0.01), ..., C_t^{'}(0.1)\}$$

$$C_t^{'}(r) = \frac{1}{36}\sum_{i=1}^{36} C_i^{'}(r)$$

Figure 56 depicts the template of CS dataset and its 99% confidence interval.

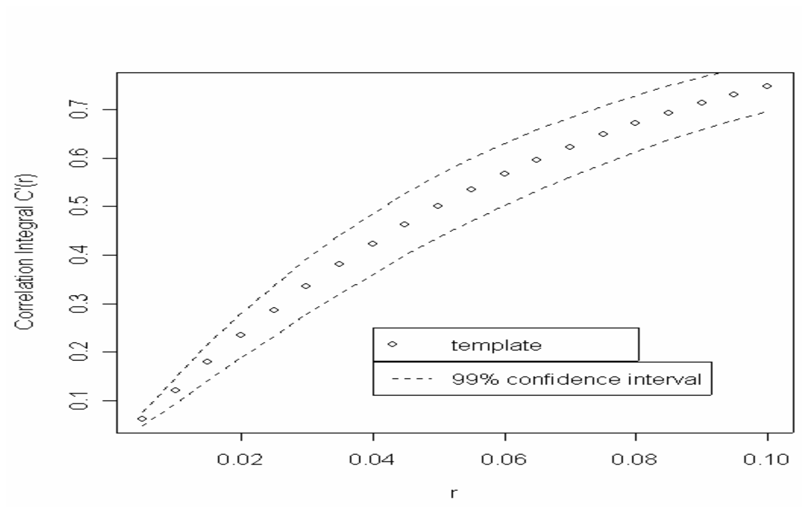
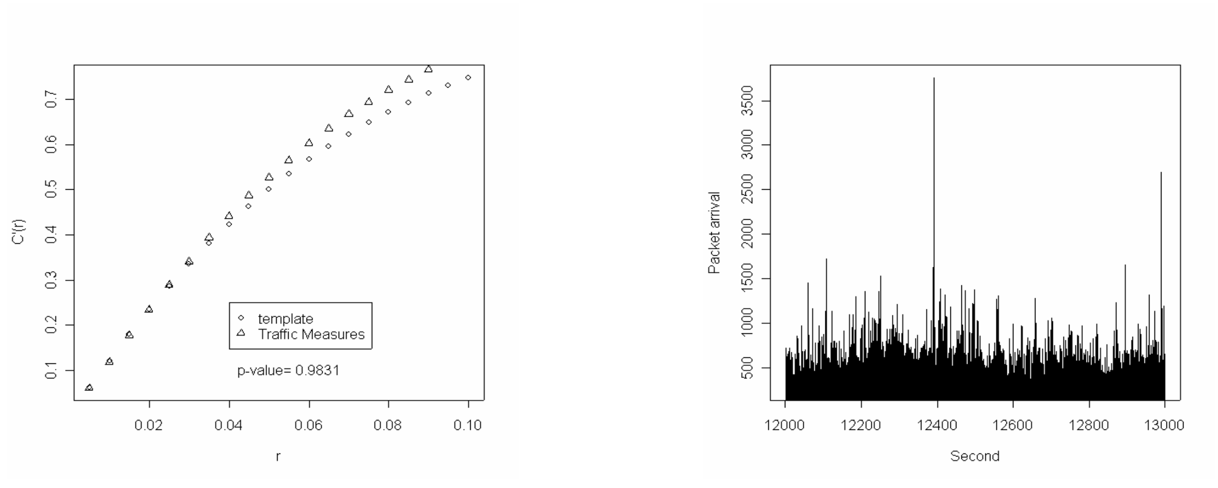


Figure 56: Template of CS traffic dataset

According to the definition of C’(r), the correlation integral is sensitive to extremely large bursts. The template in our discussion reflects the average burstiness of the traffic. If the network is not going through a very high frequency of packet arrivals from DoS attacks or other suspicious activities, the correlation integral vector should have no statistically significant difference with the template vector. The p-value from a Kolmogorov-Smirnov test on the similarlity between the template and the correlation integral vector of investigated traffic determines whether any difference is significant.

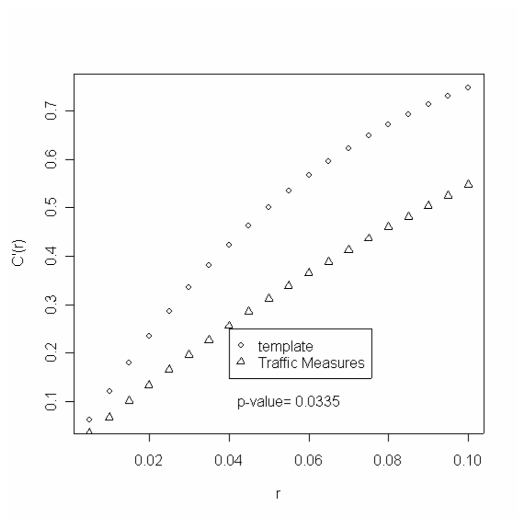
Investigation of the 36 CS traffic subsets shows that the majority of them have a correlation integral vector with similar shape to the template, and the p-values of Kolmorov-Smirnov tests are large enough to indicate no significant difference. Figure 57 gives an example of subset which has a high p-value; Figure 62 and 63 give two examples where low p-values are obtained.



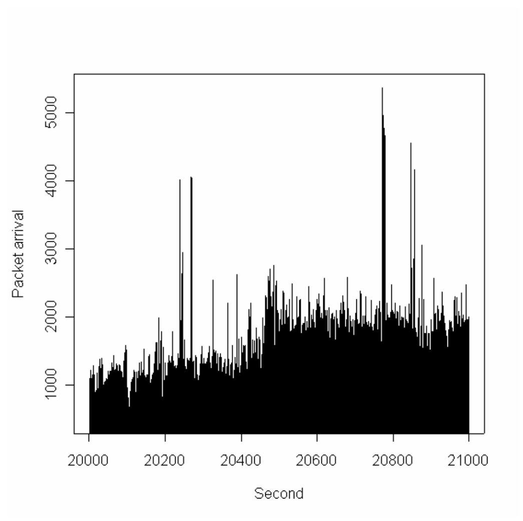
Correlation integral vector fits to the template well

Plot of actual packet arrival

Figure 57: An example of good match to the template

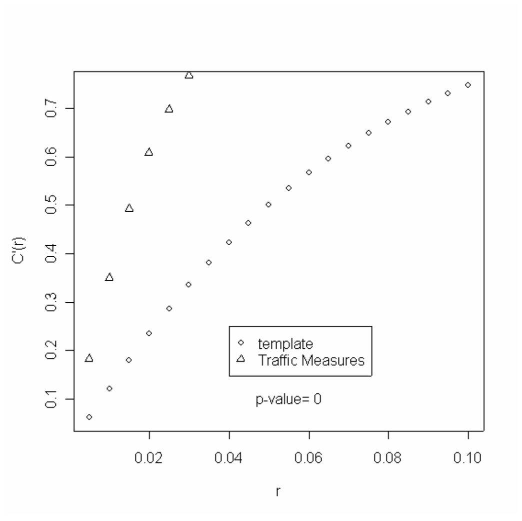


Correlation integral vector fits with the template poorly

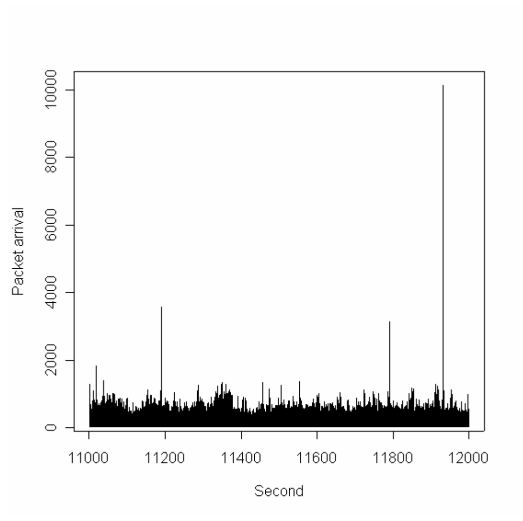


Plot of actual packet arrival

Figure 58: An example of poor match to the template



Correlation integral vector fits with the template poorly



Plot of actual packet arrival

Figure 59: Another example of poor match to the template

Figure 57 shows the typical normal traffic pattern of the CS network. Mostly, the packet arrival rates are under 1000/sec. Some traffic bursts occur, and one burst has an arrival rate larger than 3500/sec. The correlation vector of the traffic does not show significant difference (p-value=0.9831) with the template - despite the bursts.

In Figure 58, the network is under heavy usage. The average arrival rate first increases steadily to more than 1000/sec, and then to about 2000/sec. Some bursts have arrival rates more than 5000/sec. The correlation integral vector of this traffic obviously deviates from the template.

Figure 59 demonstrates another case in which unmatched vectors and a low p-value result. In this case, the traffic is normal except for a few seconds having an extremely high packet arrival rate of around 10000/sec. Due to the correlation integral's sensitivity to extremely large values, the vector of the traffic shows an obviously different pattern compared with the template. The Kolmogorov-Smirnov test returns a p-value of 0.

Table 14 summarizes p-values of Kolmogorov-Smirnov tests between the template and correlation integral vectors of 36 CS traffic subsets. The first column displays the beginning and ending second of the traffic, the second column gives the p-value.

Table 15 gives the percentage of vectors whose p-values are above some specific thresholds.

Table 14: P-values of K-S tests between traffic segments and the template

Time duration(sec.)	P-value	Time duration (sec.)	P-value
1-1000	0.001115802	18001-19000	1
1001-2000	0.8319696	19001-20000	0.8319696
2001-3000	0.335591	20001-21000	0.0335417
3001-4000	0.174533	21001-22000	1
4001-5000	0.08105771	22001-23000	0.9999924
5001-6000	0.571336	23001-24000	0.9999924
6001-7000	0.03354166	24001-25000	0.0122986
7001-8000	0.571336	25001-26000	0.0039673
8001-9000	0.9999924	26001-27000	0.9831369
9001-10000	0.9999924	27001-28000	0.9831369
10001-11000	0.571336	28001-29000	0.9999924
11001-12000	9.55E-06	29001-30000	0.0335417
12001-13000	0.9831369	30001-31000	0.571336
13001-14000	0.9999924	31001-32000	0.9999924
14001-15000	0.01229861	32001-33000	0.9831369
15001-16000	0.9831369	33001-34000	0.8319696
16001-17000	0.9831369	34001-35000	0.9999924
17001-18000	0.9831369	35001-36000	0.174533

Table 15:Kolmogorov-Smirnov Test Results

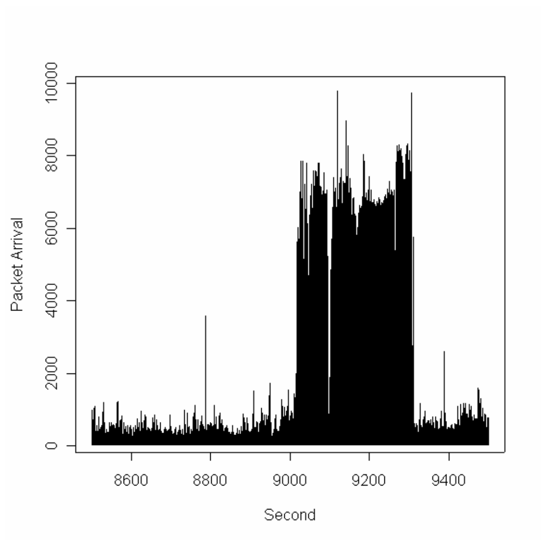
p-value	Counts (out of 36)
≥0.8	20
≥0.6	20
≥0.4	24
≥0.2	25
≥0.1	27

8.2.3 Detecting DoS-TE attacks

The correlation integral vector's sensitivity to extremely high packet arrival rates and sustained large amount of traffic makes it an ideal tool to detect those DoS-TE attacks that cause traffic floods. One can detect the DoS-TE attacks by comparing the correlation integral vectors of suspicious traffic to the template and perform various kinds of statistical tests. An ID system judges if there's any attack by reading the p-values of tests. If the p-value drops below a threshold, a warning might be sent and further analysis or action can be taken.

We demonstrate this intrusion detection technique by using real background and attack traffic. The background traffic is from the UCF CS dataset we used in previous sections. The attack traffic is taken from traces captured at FIT (Florida Institute of Technology) campus network on

Oct. 30, 2004. Analysis indicates that heavy scan activities are present in FIT traffic traces. We identified the scan packets by running Snort on the dataset, and extracted all those scan packets into a separate trace file. We used 300 seconds of heavy scan packets from FIT, and merged them into 1000 seconds of background traffic from UCF (background traffic is taken from the second 8501 to 9500; scan begins at the second 9011), as if they happened in the same network. The combined traffic is plotted in Figure 60.



Attack traffic begins at second 9001, ends at second 9300

Figure 60: Background + attack traffic.

We use a sliding time window for intrusion detection. The window is a time series of packet arrivals for 1000 consecutive seconds. To begin with the head of the window is positioned at second 9001, and the window moves to seconds of higher values with one-second steps. Each time the window arrives a new position, its correlation integral vector V is calculated and compared to the template V_t.

The sliding window is arranged to begin at the second 9001, 10 seconds before attack traffic, and stop running at the second 9040. We make this arrangement in order to demonstrate the changes of p-values before and after the beginning of the attack. Figure 61 depicts p-value versus the head position of sliding window.

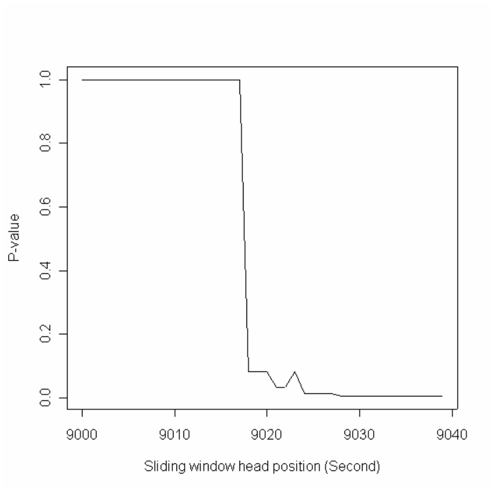


Figure 61: P-values of sliding window

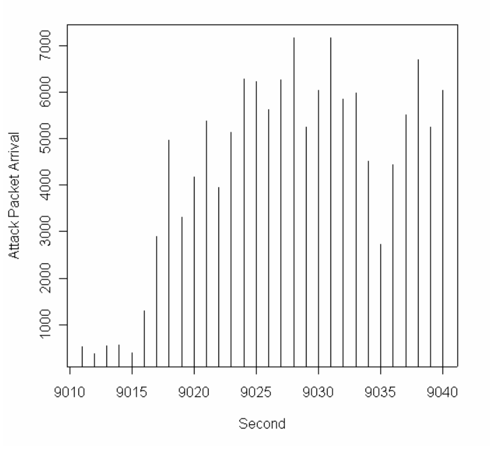


Figure 62: Attack packets during second 9011- 9040

Although the attack begins at second 9011, the p-values between second 9011 and 9018 are still high. It is because that there are not enough attacking packets during this period to damage the traffic's overall burstiness. But once the attacking packet arrival rate reaches 5000 at the second 9018, the p-value drops immediately at the second 9019, and keeps low while attack continues (Figure 62).

It is notable that we use 1 second as the step size of the sliding window. Increase of the step size will reduce the needed computation, but also increase the attack response time. Computation can also be reduced by using smaller windows.

We also performed detection experiments on the 19 DoS attacks included in the 1999 DARPA/Lincoln Lab Intrusion Detection Evaluation [73]. All of these are examples of TE attacks as defined in 7.1. We selected a period of "normal" traffic (i.e., having high p-values in the previous K-S tests) from the UCF CS background data. Because the average packet arrivals of the DARPA/Lincoln Lab experiment were much lower than that of the UCF CS data, the attack intensities (packets/second) were rescaled so that the peak attack intensity to average background intensity matched to that of original.

We used a sliding window of 1,000 seconds and step size of one second. At each position of the sliding window, we computed the correlation integral vector and compared with the vector obtained from the template (as discussed). A sharp drop of p-value below 0.1 (from the K-S test) indicated the detection of a DoS attack. Table 16 lists the 19 attacks and the number of seconds to detection.

Table 16: Time to Dectection of DoS attacks Used in the DARPA/IDS Evaluation

Attack Name	Detection Time(sec.)	Attack Name	Detection Time (sec.)
Fri4.mailbomb	9	Thu4.mailbomb	1
Fri4.smurf	2	Thu4.satan	2
Fri5.back	40	Tue5.back	4
Fri5.neptune	2	Tue5.neptune	2
Mon4.smurf	1	Wed4.mailboumb	1
Mon5.apache-1	2	Wed4.satan	1
Mon5.apache-2	2	Wed4.smurf	2
Mon5.neptune	2	Wed5.apache	1
Mon5.udpstorm	1	Wed5.back-1	1
Wed5.back-2	2		

The results show that the technique successfully detected 18 attacks within 10 seconds after the attacks began. The longest time to detection is 40 seconds. In real events, a short detection time may allow security administrators to act before an attack causes irreparable damage.

8.3 Summary and Future Work

This chapter demonstrates a correlation integral methodology that is able to track changes in traffic structure and it presents an approach for estimating the required template. The methodology is sensitive to extremely high packet arrival rates and sustained traffic volumes. These features can be used to detect DoS attacks. Computation time and sensitivity can be adjusted by selecting different sizes of sliding window and its moving step. More work will be done to determine sensitivity to stealthier attacks.

Currently, the template is built using means of correlation integrals of multiple traffic segments. Other more efficient computing method may exist, such as using weighted means. One can also change the constructional method to improve the template. For example, overlapped, instead of non-overlapped, traffic segments may generate more useful templates under some conditions.

CHAPTER 9: CONCLUSION AND FUTURE WORK

9.1 Traffic Modeling and Simulation

In this dissertation, we introduced an approach for modeling Internet background traffic using mixture distributions. The approach is proposed because traditional distributions are ineffective for a number of random variables that are essential in modeling Internet traffic. Our approach has been demonstrated for five major Internet protocols (HTTP, FTP, SMTP, POP3 and SSH) based on data collected from the CS LAN at UCF. Based on these results, an NS-2 simulation environment and a real simulation lab have been built and used for simulating the Internet background traffic and generating traffic traces. The experimental results are discussed and compared with the original traffic. The simulation results show that our models can produce realistic network traffic, especially in the NS-2 simulation environment, with regard to the following:

- The random variables of the simulation traffic have similar distributions with the real traffic
- The packet arrival patterns of the simulation compare favorably with actual arrivals
- Both simulation results and real traffic demonstrate long-range dependence
- The simulation results and real traffic compare with regard to self-similarity
- The correlation integral results are comparable for both
- The fractal dimension results from both are comparable.

One limitation of this approach is that the model is protocol-specific. One must build different models for different Internet applications, and must usually re-estimate parameters for models when applying the method in a new network environment. One possible area of future work is to classify the Internet traffic into major different categories in the hope that many protocols will fit with a single pattern. Thus, the traffic of different applications/protocols can be classified into the same classification, as long as they have the same communication pattern at the packet level. Then the mixture-distributions approach can be applied on each traffic class.

We also found that, although a close degree of self-similarity was achieved with the simulation, the simulated traffic does not have extremely high packet arrivals that characterize the real traffic. The simulation traffic looks less bursty than the real traffic. The reason may lie with the Pareto distribution that we used to model the heavy-tailed behavior of random variables. Another distribution with more obvious heavy-tailed features might be suitable to produce extremely high bursts.

Finally, we note that, although we do not report the results here, we have made similar comparisons with data collected at Florida Institute for Technology (FIT) and obtained similar results.

The ultimate goal of this project is to provide a test-bed for IDS testing and evaluation. Thus, besides the generated background traffic, one needs to model and generate network attacks, and combine them into the background to produce a synthetic traffic to test an IDS.

MAGNA [87] [89] (Modeling and Generating Network Attacks) is a system developed at UCF and FIT to generate realistic network attacks. MAGNA models attacks by their characteristics and behavior. The term characteristic refers to the contents of each unique packet included in the attack; the term behavior describes an attack's interactions with the target. The goal of MAGNA is to generate attacks that are realistic in their behavior, not just exact duplicates of sample attacks. A number of attacks have been modeled and successfully recreated by MAGNA, and experimental results show that the generated attack packets match the original packets. [89]

With successful simulations of Internet background traffic and network attacks, one can build a realistic network environment, in which the real capabilities of an IDS can be tested.

9.2 Intrusion Detection

The detection technique based on the template of traffic autocorrelation has demonstrated the potential to achieve high detection probabilities and low false-alarm rates. The key challenge with this technique is to build a template of the “normal” traffic autocorrelation. Li al et. [54] proposed the overall theory, but no actual application was demonstrated. The theory also requires attack-free traffic, which is used to compute the false-alarm rate; however, Li’s paper offers no suggestions for obtaining such attack-free traffic.

We focus on enhancing Li’s detection technique by mitigating the implementing problems in practice. Particularly, we answered the question of how to build the autocorrelation template and how to provide attack-free traffic.

Another detection technique introduced in this dissertation is based on the correlation integral. The correlation integral is closely related to the idea of fractal dimension. Allen [86] suggests a detection technique based on a fractal dimension technique and demonstrates its ability to detect a wide range of attacks. Our experiments show that the correlation integral technique is more reliable. With the help of the background traffic simulation, we can now use a more realistic traffic simulation for template building and intrusion detection.

Although this technique is successful on the UCF CS traffic trace and several other data sets, more experiments on different data are needed to validate it. There are also a number of ideas for constructing the template that we hope to pursue further.

LIST OF REFERENCES

[1] 2004 E-Crime Watch Survey. http://www.cert.org/about/ecrime.html.

[2] A. Balamash, and M. Krunz, “Appplication of multifractals in the characterization of WWW traffic”, IEEE International Conference on Communications 2002. Vol. 4.

[3] A. DelVecchio, “Building network intrusion detection systems using open source software”, http://www.sans.org, 2001

[4] A. Feldmann, A.C. Gilbert, and W. willinger, “Data networks as cascades: Investigating the multifractal nature of internet wan traffic”, Computer Communication Review, vol. 28, no. 4, pp. 42-55, 1998

[5] A. Householder, K. Houle, and C. Dougherty, “Computer attack trends challenge Internet security”**,** Computer, Vol.35, Iss.4, Apr 2002

[6] A. Reyes-Lecuona, et al, ``A page-oriented WWW traffic model for wireless system simulations", 16th International Telegraphic Congress, Vol. 2, pp 1271-1280, Edinburgh, June, 1999.

[7] A. T. Zhou, J. Bluestein and N. Zincir-Heywood, “Improving intrusion detection systems through heuristic evaluation”, CCECE 2004, May 2004

[8] B. A. Mah, “An empirical model of HTTP network traffic”, In Proceedings of IEEE INFOCOM, pages 592-600, 1997.

[9] B. B. Mandelbrot, The Fractal Geometry of Nature, W. H. Freeman, 1983

[10] B. B. Mandelbrot and J.W. Van Ness, “Fractional Brownian motions, fractional noises and applications” SIAM Review, vol. 10, no. 4, pp. 422-437, 1968

[11] B. Mukherjee, T. Heberlein, and K. Levitt, “Network, intrusion detection”, IEEE Network, May/June 1994

[12] B. Yocom, R. Birdsall and D. Poletti-Metzel, “Gigabit intrusion-detection systems”, Network World, Dec. 2002

[13] CERT/CC Advisory CA-2001-23, http://www.cert.org/advisories/CA-2001-23.html

[14] CERT/CC statistics, http://www.cert.org/stats/cert_stats.html

[15] "Code Red" Worm Crashes IIS 4.0 Servers with URL Redirection Enabled, CERT Incident Note IN-2001-10.

[16] D. Anderson, T. Frivold, and A. Valdes, “Next-generation Intrusion Detection Expert System (NIDES) a summary”, Technical report SRI-CSL-95-07, SRI International, May 1995.

[17] D. Anick, D. Mitra, and M. M. Sondhi, “Stochastic theory of a data-handling system with multiple sources”, Bell System Technical Journal. 61(8):1871-1894, 1982.

[18] D. Bell and L. LaPadula, “Secure computer system: unified exposition and multics interpretation”, Technical report MTR-2997, Mitre Corporation, 1976.

[19] D. E. Denning, “An intrusion detection model”, IEEE Transactions on Software Engineering, 13(2), Feb. 1987

[20] D. E. Denning, “Information Warfare and Security”, Addison-Wesley, 1999.

[21] Denial of Service Attacks using Nameservers, CERT Incident Note IN-2000-04.

[22] D. Heyman and T.V. Lakshman, “Source models for VBR broadcast video traffic”, IEEE/ACM Transactions on Networking, 4(1):37-46, Feburary 1996.

[23] D.L. Jagerman, B. Melamed, and W. Willinger, “Stochastic modeling of traffic processes”, in Frontiers in Queuing: Models, Methods and Problems, J.H. Dshalalow, Ed. 1996, CRC Press.

[24] D. Newman, J. Snyder, and R. Thayer. “Crying wolf: False alarm hide attacks”, http://www.nwfusion.com/techinsider/2002/0624security1.html, 2002

[25] E. Casilari, F. J. Gonzalez, and F. Sandoval, “Modeling of HTTP Traffic”, IEEE Communication Letters, vol. 5, No. 6, June 2001

[26] E. Fuchs and P.E. Jackson, “Estimates of distributions of random variables for certain computer communications traffic model”, Communication ACM, vol. 13, no. 12, pp:752-757, Dec. 1970.

[27] E. W. Knightly and H. Zhang, “D-bibd: An accurate traffic model for providing QoS guarantees to VBR traffic”, IEEE/ACM Transactions on Networking, 5(2):219-231, April 1997

[28] F. Donelson Smith, F. Hernandez-Campos, Kevin Jeffay, and David Ott., “What TCP/IP protocol headers can tell us about the web”, In Proc. Of SIGMETRICS/Performance, pages 245-256, 2001.

[29] F. Hernández-Campos, A. B. Nobel, F. D. Smith, and K. Jeffay, “Statistical Clustering of Internet Communication Patterns”, In Proceedings of the 35th Symposium on the Interface of Computing Science and Statistics, Computing Science and Statistics, Vol. 35, July 2003.

[30] G. Vigna and R. Kemmerer, “NetSTAT: a network-based intrusion detection system”. In Proceedings, 14th Computer Security Conference, 1998

[31] G. White, E. Fisch, and U. Pooch, "Cooperating Security Managers: A peer-based intrusion detection system", IEEE Network, Jan/Feb 1996

[32] H. Fowler and W. Leland, "Local area network traffic characteristics, with implication for broadband network congestion management", IEEE Journal on Selected Areas in Communications. 9(7):1139-1149, Sept. 1991.

[33] H. Heffes and D. M. Lucantoni, "A Markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance", IEEE Journal on Selected Areas in Communication, 4(6):856-868, 1986.

[34] H. Hlavacs, G. Kotsis, and C. Steinkellner, "Traffic source modeling", Technical Report TR-99101, Institute of Applied computer Science and Information Systems, University of Vienna, 1999

[35] H. Javitz and A. Valdes, "The SRI IDES statistical anomaly detector", In Proceedings, IEEE Symposium on Research in Security and Privacy, May 1991.

[36] HTTP://www.isi.edu/nsnam/ns/

[37] H. Vaccaro and G. Liepins, "Detection of anomalous computer session activity", In Proceeding, IEEE Symposium on Research in Security and Privacy, 1989.

[38] J. Anderson. "Computer security threat monitoring and surveillance", Technical report, J. P. Anderson Co., 1980

[39] J. Beran, "Statistics for Long-Memory Processes", chapman & Hall, New York, 1994

[40] J. H. Saltzer and M. D. Schroeder. "The protection of information in computer systems", Communications of the ACM, 17(7), July 1974

[41] J. Nielsen, "Heuristic evaluation", http://www.useit.com, 2003.

[42] J. Snyder, "False positives remains a major problem", http://www.nwfusion.com/reviews/2003/1013idsalert.html, 2003

[43] J. Winkler and W. Page, "Intrusion and anomaly detection in trusted systems", In Proceedings, 5th IEEE computer Security Applications Conference, Dec. 1989.

[44] K. Ilgun, R. Kemmerer and P. Porras, "State Transition Analysis: a rule-based intrusion detection approach". IEEE Transaction on Software Engineering, 21(3), Mar. 1995

[45] K. Park, G. Kim, and M. Crovella, "On the relationship between file sizes, transport protocols, and self-similar network traffic", Technical Report 1996-016, Boston University, 30, 1996.

[46] L. E. Schrage, and L. W. Miller, "The queue M/G/1 with the shortest processing remaining time discipline." Operations Research, 14:670 – 684, 1966.

[47] L. Heberlein, G. Dias, K. Levitt, et al. "A Network Security Monitor", In Proceedings, IEEE Symposium on Research in Security and Privacy, 1990

[48] M. A. Cooper, "HONEST: Hands-on Networking Environment for Security Testing", Master Thesis, FIT 2005

[49] M. Ayedimir, L. Bottomley, et al., "Two tools for network traffic analysis", Computer Networks, 36, 2001

[50] M. Basseville, "Distance measure for signal processing and pattern recognition", Signal Processing, vol. 18, 1989.

[51] M.E. Crovella and A. Bestavros, “Explaining world wide web traffic self-similarity”, Technical Report TR-95-015, Computer Science Department, Boston University, Oct. 1995.

[52] M. E. Crovella and A. Bestavros, “Self-similarity in World Wide Web traffic: evidence and possible causes”, IEEE/ACM Transactions on Networking, 5(6):835-846, 1997.

[53] M. Li, W. Jia, and W. Zhao, “A whole correlation structure of asymptotically self-similar traffic in communication networks”. Proceedings of the First International Conference on Web Information Systems Engineering, Vol. 1, 19-21 June 2000.

[54] M. Li, W. Jia and W. Zhao, “Decision analysis of network based intrusion detection systems for Denial-of-Service attacks”, In Proceedings, IEEE Conferences on Info-tch and Info-net, 2001.

[55] M. Li, W. Jia, and W. Zhao "Modeling WWW-traffic Data by Autocorrelations", Proceedings of the 7th International Conference on Distributed Multimedia Systems (DMS 2001), Taipei, Taiwan, 26-28 September 2001, pp 338-343.

[56] M. Li, W. Jia and W. Zhao, “Simulation of long-range dependent traffic and a simulator of Tcp arrival traffic”, Int. J. of Interconnection Networks, Sep. 2001.

[57] M. Roesch, “Snort – lightweight intrusion detection for networks”, In Proceedings, 13th USENIX System Administration Conference, Nov. 1999

[58] M. Stoksik, R. Lane, and D. Nguyen, “Accurate synthesis of fractional Brownian motion using wavelets”, Electronics Letters, vol. 30, no. 2, pp. 383-384, 1994

[59] M. W. Garrett and W. Willinger, “Analysis, modeling and generation of self-similar VBR video traffic”, SIGCOMM, pages 269-280, 1994

[60] National Computer Security Center. “Trusted computer system evaluation criteria”. Technical report DOD 5200.28-STD, Department of Defence, 1985.

[61] P. B. Danzig and S. Jamin, “Tcplib: A library of TCP/IP traffic characteristics”, USC Networking and Distributed Systems Laboratory TR CS-SYS-91-01, October, 1991.

[62] P. Barford and M. Crovella, “Generating representative web workload for network and server performance evaluation”, In proc. Of the ACM SIGMETRICS, pages 151-160, 1998.

[63] P. Danzig, S. Jamin, R. Caceres, D. Mitzel and D. Estrin, “An empirical workload model for driving wide-area TCP/IP network simulations”, Internetworking: Research and Experience 3:1-26, Mar., 1992.

[64] P. E. Jackson and C. D. Stubbs, “A study of multi-access computer communications”, Proc. AFIPS 1969 SJCC, vol. 34, AFIPS Press, Montvale, N. J., pp. 491-504.

[65] P. Flandrin, “Wavelet analysis and synthesis of fractional Brownian motion”, IEEE Transactions on Information Theory, vol. 38, no. 2, pp. 910-917, 1992

[66] P. Grassberger and I. Procaccia, “Characterization of strange attractors”, Physical Review Letters, 50(5), 1983.

[67] P. J. Brockwell and R. A. Davis, “Introduction to Time Series and Forecasting”, Springer 2002.

[68] P. Neumann and P. Poras, “Experience with EMERALD to date”, In proceedings, USENIX Workshop on Intrusion Detection and Network Monitoring, April 1999

[69] P. Porras and P. Neumann, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances", In Proceedings, 20th National Information Systems Security Conference, Oct. 1997

[70] R. Gusella, "A measurement study of diskless workstation traffic on an Ethernet", IEEE Trans. on Communications. 38(9):1557-1568, Sept., 1990.

[71] R. Jain and S. A. Routhier, "Packet trains: Measurements and a new model for computer network traffic", IEEE Journal on Selected Areas in Communication, 4(6):986-995, 1986.

[72] R. K. Cunningham, R. P. Lippmann, D. J. Fried, S. L. Garfinkle, I. Graf, K. R. Kendall, S. E. Webster, D. Wyschogrod, M. A. Zissman, "Evaluating Intrusion Detection Systems without Attacking your Friends: The 1998 DARPA Intrusion Detection Evaluation", SANS 1999.

[73] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, K. Das. "The 1999 DARPA Off-Line Intrusion Detection Evaluation". Computer Networks: The International Journal of Computer and Telecommunications Networking. Volume 34, Issue 4, 2000.

[74] R. Riedi, M. Crouse, V. Ribeiro, and R. Baraniuk, "A multifractal wavelet model with application to network traffic", IEEE Transaction on Information Theory, 45(3):992-1018, April, 1999.

[75] S. Floyd, V. Paxon, "Difficulties in Simulating the Internet", IEEE/ACM Transaction on Networking, Volume 9, Issue 4, pp. 392 - 403, 2001.

[76] S. Northcutt and J. Novak, "Network Intrusion Detection: an Analyst's Handbook", New Riders Publishing, second edition, 2001.

[77] S. Robert and J.Y. Le Boudec, “New models for pseudo self-similar traffic”, Performance Evaluation, vol. 30, pp. 57-68, 1997

[78] S. Smaha, “Haystack: an intrusion detection system”, In Proceedings, IEEE 4th Aerospace Computer Security Applications Conference, Dec. 1998.

[79] S. Staniford-Chen, S. Cheung, R. Crawford, et al. “GrIDS – a Graph-based Intrusion Detection System for large networks”. In Proceedings, 19th National Computer Security conference, Oct. 1996.

[80] T. Bass, “Intrusion detection systems and multisensor data fusion”, *Communications of the ACM*, April 2000/vol. 43, No. 4, pp99-105

[81] T. Lunt, “Detecting intruders on computer systems”, In Proceedings, Conference on Auditing and Computer Technology, 1993

[82] V. Paxon, “Bro: A system for detecting network intruders in real-time”, Computer Networks, 31, Dec. 1999.

[83] V. Paxson, “Empirically derived analytic models of wide-area TCP connections”, Networking, IEEE/ACM Transactions on , Volume: 2 Issue: 4 , Aug. 1994, Page(s): 316 –336

[84] V. Paxson, “Fast approximation of self-similar network traffic”, Technical Report LBL-36750/UC-405, Lawrence Berkeley National Laboratory, Apr. 1995.

[85] V. Paxson, and S. Floyd, “Wide area traffic: the failure of Poisson modeling”, Networking, IEEE/ACM Transactions on , Volume: 3 Issue: 3 , June 1995, Page(s): 226 –244

[86] W. H. Allen, “Analysis, detection, and modeling of attacks in computer communication networks”, Ph.D. thesis, University of Central Florida, 2003

[87] W. H. Allen, G. Marin, “MAGNA: Modeling And Generating Network Attacks”, in Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN’04), Nov. 2004

[88] W. H. Allen and G. A. Marin, "On the Self-similarity of Synthetic Traffic for the Evaluation of Intrusion Detection Systems", in Proceedings of the IEEE Symposium on Applications of the Internet (SAINT ’03), January 2003

[89] W. H. Allen, S. Luo, G. Marin, “Modeling network traffic and attacks for security testing”, In theProceedings of the 2nd IASTED Conference on Computer and Communication Networks, Nov. 2004

[90] W. Lee, J.B.D. Cabrera, A. Thomas et al. “Performance adaption in real-time intrusion detection systems”. In Proceedings, 5th International Symposium on Recent Advance in Intrusion Detection (RAID 2003), Oct. 2002

[91] W. Lee and S. J. Stofo, “Data mining approaches for intrusion detection”, Proceedings of the 7th USENIX Security Symposium, 2000

[92] W. Leland, M. Taqqu, W. Willinger, and E. Wilson, “On the self-similar nature of Ethernet traffic (extended version)”, IEEE/ACM Trans. Networking, vol 2, pp. 1-15, Feb., 1994.

[93] W. Willinger, M. Taqqu, R. Sherman and D. Wilson, “Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level”, Proc. SIGCOMM ’95, pp. 100-113, 1995.

[94] W. Willinger and V. Paxson, "Where mathematics meets the Internet", Notices of the American Mathematical Society, 45(8), Sep. 1998

[95] W. Ralph, "SHADOW installation manual", http://www.nswc.navy.mil, 2001

[96] Y. Bai and H. Kobayashi, "Intrusion detection systems: technology and development", In Proceedings, IEEE 17th International Conference on Advanced Information Networking and Application,2003

[97] Y. Bai, H. Kobayahsi, "Intrusion Detection Systems: Technology and development", Procedings of 17th International Conference on Advanced Information Networking and Application, March 2003