MASTER'S THESIS

Change Detection Algorithms for Information Assurance of Computer Networks

by Alvaro A. Cardenas
Advisor: John S. Baras

MS 2003-3

ABSTRACT

Title of Thesis:	CHANGE DETECTION ALGORITHMS FOR INFORMATION ASSURANCE OF COMPUTER NETWORKS
Degree candidate:	Alvaro A. Cardenas
Degree and year:	Master of Science, 2002
Thesis directed by:	Professor John S. Baras Department of Electrical Engineering

As our reliance on computer networks grows, the need for better and more accurate intrusion detection systems to identify and contain attackers becomes a fundamental research topic.

In this thesis we will focus on the detection of three attack scenarios: spreading of active worms throught the Internet, distributed denial of service attacks and routing attacks to wireless ad hoc networks. For the first two attacks we will determine anomalous changes in the network flow. For the third attack, we provide an abstract representation of a highly mobile ad hoc network in order to establish a baseline for detecting abnormalities generated by intrusions changing the behavior of the routing protocol. We consider these problems in the framework of sequential

change detection theory as we want to detect the appearance of an attack early in its development.

CHANGE DETECTION ALGORITHMS FOR INFORMATION ASSURANCE OF COMPUTER NETWORKS

by

Alvaro A. Cardenas

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2002

DEDICATION

To my grandmother Delfina.

ACKNOWLEDGMENTS

I am grateful to my advisor, Professor John S. Baras for his support. I would also like to thank Dr. Eyad Abed and Dr. William S. Levine for agreeing to serve on my committee and review the thesis. I am in particular indebted to Dr. Vahid Ramezani, Svetlana Radosavac, Karthikeyan Chandrasekar, Tao Jiang, Marco Alzate and all my colleagues in the SEIL lab and the CSHCN for their suggestions and support.

I am grateful for the support of my research work and graduate studies through the following contracts and grants (all with the University of Maryland College Park): ARO DAAD190110494, ONR N000149710501EE and a graduate school fellowship

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Intrusion Detection

The number of computer attacks increases steadily per year. At the time of this writing the Internet Security Systems' baseline assessment is that a new, unprotected computer with an Internet connection will be compromised within one day or sooner [1]. Intrusion prevention techniques such as encryption and authentication are usually the first line of defense. In practice however, most of the times there are weaknesses in the implementation of intrusion prevention and intrusion detection becomes a second line of defense.

1.2 Change Detection Algorithms

In detection theory if we observe a given stochastic process for a fixed interval of time, the typical objective is to minimize the error probabilities (probability of false alarm and probabiliti of missed detection). Optimal detection schemes have been found, which produce tests that compare the likelihood ratio between the given hypotheses with a threshold dependent on the error probabilities.

In classical sequential detection methods the goal is to find optimal solutions to a hypothesis testing problem while accounting for trade-offs between the probability of error and the decision time. This problem is typically formulated as a Markov stopping time and the usual test is a sequential probability ratio test (SPRT). A more specific problem within sequential detection that we will also consider is the quickest change detection, where the trade-off is between the delay of detection and the false alarm rate.

Most of the presentation of these problems and formulations follows [2].

1.2.1 Sequential detection

We will assume that we are observing a discrete time stochastic process $(X_k)_{k\geq 1}$. We say that a test $\zeta = (g, T)$ is a sequential probability ratio test (SPRT) for testing between simple hypotheses $H_0 = \{\theta : \theta = \theta_0\}$ and $H_1 = \{\theta : \theta = \theta_1\}$ if we sequentially observe data and if for each time k we make one of the following decisions

- accept H_0 when $Z_k \leq a$
- accept H_1 when $Z_k \geq h$
- continue to observe and to test when $a < Z_k < h$.

In the above,

$$Z_k = \log \frac{f_{\theta_1}(X_1^k)}{f_{\theta_0}(X_1^k)} \tag{1.1}$$

is the log likelihood ratio function of the two hypotheses. $X_1^k = \{X_1, ..., X_k\}$, and a and h are positive thresholds such that $-\infty < a < h < \infty$. Next, let us define

$$g(X_1^T) = \begin{cases} 1, \text{ when } Z_T \geq h \\ 0, \text{ when } Z_T \leq a \end{cases} \tag{1.2}$$

where T is the exit time

$$T = T_{a,h} = min\{n \geq 1 : (Z_n \geq h) \cup (Z_n \leq a)\} \tag{1.3}$$

Wald's sequential probability ratio test (SPRT) is optimal for binary hypotheses testing (M=2) between two independent distributions in the sense that it simultaneously minimizes both expectations of the sample size among all tests for which the probabilities of error do not exceed predefined values.

1.2.2 Optimal change detection algorithms

Despite the abundance of techniques addressing the change detection problem, optimum schemes can mostly be found for the case where the data are independent and identically distributed (i.i.d.) and the distributions are completely known before and after the change time k_0 [3].

The cumulative sum (CUSUM) and the Shiryaev-Roberts statistics are the two most commonly used algorithms for change detection problems.

1.2.2.1 CUSUM algorithm

For the i.i.d. case, the CUSUM algorithm minimizes (non asymptotic optimality) the following objective function

$$F(T) = \sup_{k_0 \geq 1} \; ess \, \sup E_{k_0}[(T - k_0 + 1)^+ | x_1, ..., x_{k-1}] \tag{1.4}$$

(where $(x)^+$denotes $\max\{0, x\}$) among all stopping times subject to the false alarm constraint $E_\infty[T] \geq 1/F_a$ for some given $F_a \geq 0$. The algorithm is given by

$$z_i = \log \frac{f_{\theta_1}(x_i)}{f_{\theta_2}(x_i)} \tag{1.5}$$

$$Z_k = \sum_{i=1}^{k} z_i \tag{1.6}$$

$$m_k = \min_{1 \leq j \leq k} Z_j \tag{1.7}$$

and the stopping time is:

$$T = \inf_k \{k : Z_k - m_k \geq h\} \tag{1.8}$$

for a given threshold h satisfying the false alarm constraint.

It can be shown that $Z_k - m_k$ satisfies the recursion $Z_k - m_k = (Z_{k-1} - m_{k-1} + z_k)^+$ with $Z_0 - m_0 = 0$.

1.2.2.2 Shiryaev-Roberts statistic

For the i.i.d. case, the Shiryaev-Roberts statistic minimizes (non asymptotic optimality) the objective function

$$F(T) = \sup_{k \geq 1} E_k[T - k_0 | T \geq k_0] \tag{1.9}$$

subject also to a given false alarm rate. The algorithm is computed by the recursion

$$R_n = (1 + R_{n-1})e^{z_i} \text{ with } R_0 = 0 \tag{1.10}$$

The stopping time is given by

$$T = \inf_n \{n : \log(R_n) \geq h\} \tag{1.11}$$

1.2.2.3 Algorithm with a prior of the change time

There is a final optimal algorithm for i.i.d. observations proposed by Shiryaev that minimizes the objective function

$$F(T) = E\{1_{\{T<k_0\}} + c(T - k_0)^+\} \tag{1.12}$$

where c represents the cost of each "extra" observation after the change time k_0 and $1_{()}$is the indicator function. The algorithm assumes a geometric prior for the change time. The optimal stopping rule consists in stopping the first time the posterior probability of the change time exceeds some constant threshold.

1.2.3 Nonparametric CUSUM and Gishik-Rubin-Shiryaev Statistics

Most change detection algorithms applied to network traffic use nonparametric statistics as it is very complicated to know or model the pre-change and post-change distributions of an observation of the network flow at all times by parametric families.

The non-parametric version of the CUSUM assumes a random process with some regularity conditions. The form of the process is

$$x_k = a + \xi_k 1_{(k<k_0)} + (b - \psi_k) 1_{(k \geq k_0)} \tag{1.13}$$

where $E[\xi_k] = E[\psi_k] = 0$, $b + a > 0$, $a < 0$ and $1_{()}$is the indicator function.

The statistic for such a sequence is

$$S_k = (S_{k-1} + x_k)^+,\ S_0 = 0 \tag{1.14}$$

and the corresponding decision rule is

$$T = \inf_n \{n : S_n > h\} \tag{1.15}$$

for a given threshold h selected based on the false alarm rate.

The nonparametric version equivalent to the Shiryaev-Roberts statistic is the Gishik-Rubin-Shiryaev statistic [4].

1.2.4 GLR for composite hypothesis and nuisance parameters

The case of composite hypotheses is more useful and important from a practical point of view. Unfortunately, this case is much more complicated and there are fewer available theoretical results than in the case of simple hypotheses.

Two possible solutions for the case of composite hypotheses are the generalized likelihood ratio (GLR) and the average over the prior distribution of the composite and nuisance parameters.

The precise optimal properties of the GLR test in the general case are unknown, but for many special cases, the GLR test is optimal and therefore it will be our preferred choice.

To present the algorithm consider the case where the parameter θ_1 after the change is unknown and before the change θ_0 is known. The log-likelihood ratio for observations from time j up to time k is

$$Z_j^k(\theta_1) = \log \frac{f_{\theta_1}(x_j, ..., x_k)}{f_{\theta_0}(x_j, ..., x_k)}, \tag{1.16}$$

so the ratio is a function of two unknown independent parameters: the change time and the value of the parameter after the change. The standard statistical approach is to use the maximum likelihood estimates of these two parameters, and thus the double maximization:

$$g_k = \max_{1<j<k} \sup_{\theta_1} Z_j^k(\theta_1) \tag{1.17}$$

1.3 Network definitions and algorithms

The network topology will play an important role in the chapter on worm detection so we will include some basic definitions.

Various classes of graphs (networks) have recently attracted attention in relation to various dynamic properties of the Internet and other complex networks.

One of the first network models studied is the *Erdós − Rényi random graph* model, where one starts with N nodes and connects every pair of nodes with probability p, ending up with a graph with approximately $pN(N-1)/2$ edges distributed randomly, and with the typical *distance* (hop count) between two nodes scaling as $\log(N)$. Another important parameter in the description of a network is the node *degree*, i.e. the number of edges (links) emanating from a node. The important discriminant is the degree distribution:

$$Pr[\text{a randomly selected node has k edges}]{:=}P(k)$$

In a random graph the majority of the nodes have approximately the same degree, close to the average degree of the network k_{av}, and the degree distribution is Poisson. On the other hand, in most real large networks such as the router graph on the Internet and the WWW graph, the degree distribution deviates significantly from a Poisson distribution. It has a power law tail $P(k) \approx k^{-\gamma}$. Such networks are often called *scale-free networks.*

A key parameter in the analysis and discrimination between graphs is the *clustering coefficient* of a node i, characterized by the number of edges E_i between all neighbors k_i over all possible edges among them $k_i(k_i - 1)/2$.

A good overview of these topics including the basic algorithms to generate different kind of networks is presented in [5].

Chapter 2

Worm Detection

2.1 Description of the Attack

Worms are programs that self-propagate across a network by exploiting security flaws in widely-used services offered by vulnerable computers. In order to locate the vulnerable computers, the worm probes different computer addresses at the specific port number of the service it is looking for. By exploiting the security flaw in the service, the worm usually can execute arbitrary code with elevated privileges, allowing it to copy and execute itself in the compromised machine. In order to reproduce, the worm scans for new vulnerable machines from each new compromised computer.

There are other types of worms that we will not consider: for example email worms (viruses) such as Melissa, which require some sort of user action to abet their propagation. As such they tend to propagate more slowly and are easier to detect.

Worms are popular attacks because no other mechanism allows for the rapid and widespread distribution of malicious code, with virtually no way to trace the attacker.

We will focus on the description of some worms to clarify the concepts:

2.1.1 Examples of self-propagating code in the Internet

The second half of 2001 witnessed the appearance of three major worms with a high infection success.

- Code Red I (v2) was originally detected in July 19, 2001. The same day it was reported that more than 250,000 machines were infected in just 9 hours. This worm exploits a security flaw in Microsoft IIS web servers (the .ida vulnerability.) Once it infects a host, it spreads by launching 99 threads which generate uniformly random IP addresses. Each of these 99 threads probed the destination computers to determine if the service was accessible. After the 3 TCP way handshake, it uploads all the code, waits for the acknowledgment and moves on to infect other computers.

- Code Red II was originally detected in August 2001. It exploited the same vulnerability as Code Red I. If the language installed in the system was Chinese, it ran 600 threads and spread for 48 hours, otherwise it ran 300 threads and spread for 24 hours. The scanning of each thread was different from that of Code Red I. It attempted to infect computers with addresses close to the infected machine. With probability 4/8 it probed an address in the same class A network, with probability 3/8 it probed the same class B network and with probability 1/8 it probed an address in the whole Internet. This worm died by design by rebooting its host machines after October 2001.

- Nimda was detected on September 2001. It has been called a hybrid worm because it tries to spread by different means such as probing web servers, mass emailing, and by scanning for the back-doors left behind by Code Red II.

Even though Code Red I and Nimda are still very popular, there are new worms being released in the Internet very often. Here we mention two recent examples to show the prevalence of this threat to date:

- The SQL Spida worm follows the footsteps of Nimda and Code Red. It was detected in May 2002. Once a vulnerable SQL server is found, the worm will infect that target and begin scanning for new targets. The scanner bundled with the worm is multi-threaded. It searches for uniformly random IP addresses over the Internet.

- Slapper was detected in September 2002. It spread rapidly throughout the Internet, affecting over 20,000 vulnerable servers within 72 hours. It exploits an OpenSSL buffer overflow vulnerability in Apache web servers. The worm attacks by sending an initial HTTP request to port 80 (HTTP) and examining the Server header response. Only if the target server is running certain

Linux distributions it will proceed to connect to port 443 (HTTPs) to send the exploit code to the SSL service that is listening on the remote computer.

2.2 Why is it important to detect worms early in their development?

In the latest Internet Risk Impact Summary report published by ISS at the time of this writing [1], the prevalence and increase of computer attacks using worms was emphasized. The top three categories of computer attacks are directly related to worms and other self-propagating hybrid threats which exploit multiple vulnerabilities across desktops and servers. In addition, the worm attacks of Code Red and Nimda were selected as the top worst security attacks in the last five years [6].

We would like to detect a worm as soon as possible in order to minimize the number of compromised hosts. A case example was that the quick discovery and prompt action by System Administrators prohibited Slapper from spreading further and prevented its damage [1]. Some highly contagious worms can also induce side effects such as BGP routing instabilities when they reach their peak.

Currently, however, detection usually relies on informal email discussion on a few key mailing lists. This process takes hours at a minimum, which is very slow for the rapidly-propagating worms.

Furthermore in [7] it is stated that the spread of the theoretical flash or Warhol worms will be so fast that no human-driven communication will suffice for adequate identification of an outbreak before nearly complete infection is achieved. It is therefore proposed to sponsor research in automated mechanisms for detecting worms based on their traffic patterns. The appearance of such a worm was selected as the greatest security threat in the Internet [6].

2.3 Detection algorithms

Although the spread of a worm increases traffic over a network, the worm itself is small (Code Red was 4KB), and it only takes 40 bytes for a TCP SYN packet to determine if a service is accessible, so detection cannot rely on bandwidth statistics. Furthermore, detection based on side effects such as the use of host unreachable messages and connection attempts to routers as a way of detecting worms will be less reliable while the worm is getting off the ground if it uses a hit-list scanning [7].

We will focus on the fact that the self propagating code will try to use specific vulnerabilities that can be identified with certain port numbers. So in the rest of this chapter we will assume that the traffic monitoring variable is the connection attempts (probes) to a given TCP/UDP port number(s). We will also assume most of the time a probability distribution on the traffic observations. Such probability can be of the form Pr(Number of probes to a given port number | A service is provided in such a port). So in our framework we assume that there is a baseline of connections to the given monitored port in all sensors (computers) of the network. The observations can be made at different participating ISPs enforcing policies for blocking self-propagating code once it is detected.

We will explore the effect of aggregation from distributed sensors. This approach is motivated by the current infrastructure of distributed Intrusion Detection Systems such as myNetwatchman, Dshield and Symantec's DeepSight Threat Management System. Another motivation is presented in [7] as the authors propose to foster the deployment of a widespread set of sensors for worm detection (possibly in the Internet backbone.)

2.3.1 Detection of a change in the mean

Clearly the simplest approach to change detection is to detect a change in the mean. We will denote by $\{X_{l,k}\}$ the observed stochastic process at sensor $l \in \{1, ..., L\}$ at time $k \in \mathbb{Z}^+$ and by X_k we will denote the aggregate observation at time k, i.e.

$$X_k = \sum_{l=1}^{L} X_{l,k} \tag{2.1}$$

2.3.1.1 Change detection in distributed sensor systems

In this first approach we will assume that each sequence $\{X_{l,k}\}$ is i.i.d. with pdf $f_l^{(0)}$ before and $f_l^{(1)}$ after the change. Furthermore, in traditional distributed detection it is assumed that the observation sequences $\{X_{1,k}\}, ..., \{X_{L,k}\}$ are mutually independent s.t. $f^{(i)}(x_{1,k}, ..., x_{L,k}) = \prod_{l=1}^{L} f_l^{(i)}(x_{1,k})$ for $i \in \{0, 1\}$ in order to keep the problem tractable. In our case we can assume the independenve condition among sensors holds when the worm scans randomly the whole network monitored by the sensors i.e. the worm has no scanning preference between sensors.

Based on the information available at time k a message $V_{l,k}$ depending on $X_{l,k}$ is sent to the fusion center. We want to find a stopping time $T \in \mathbb{Z}^+$ minimizing

$$\sup_{1 \le k_o < \infty} E_{k_o}\{(T - k_o)^m | T \ge k_o\} \tag{2.2}$$

for all m, (where k_0 is the unknown change time) subject to a false alarm rate $1/E_\infty[T] < F_a$. It was shown in [8] that the optimal message (in the asymptotic case when $F_a \to 0$) $V_{l,k}$ is completely determined by threshold comparisons of the likelihood ratio at the sensor l: $\beta_{V_l} < f_l^{(1)}(X_{l,k})/f_l^{(0)}(X_{l,k}) \le \beta_{V_l+1}$ (note the independence on V on k). It is of importance in traditional detection over distributed sensors to minimize the transmission bandwidth, so $V_{l,k}$ is usually a quantized observation based on the partition on $f_l^{(1)}(X_{l,k})/f_l^{(0)}(X_{l,k})$ by the thresholds β_{V_l}. In the limit case, the sensor will not send anything (0) if the likelihood ratio is below a certain threshold, and will send an alarm (1) if it is greater than a threshold β. In our case since bandwidth is not a limiting factor we will send the likelihood (or log-likelihood) evaluation to the fusion center.

The message $V_{l,k}$ will have an induced distribution $g_l^{(i)}$ when the observation $X_{l,k}$ is distributed as $f_l^{(i)}$. The test at the fusion center using the Shiryaev-Roberts detection is thus

$$Z(k) = \sum_{l=1}^{L} \log \frac{g_l^{(1)}(V_{l,k})}{g_l^{(0)}(V_{l,k})} \tag{2.3}$$

$$R(k) = (1 + R(k-1))e^{Z(k)},\ R(0) = 0 \tag{2.4}$$

$$T(F_a) = \inf\{k \geq 1 : R(k) \geq 1/F_a\} \tag{2.5}$$

We will assume that $f_l^{(0)}(x_{l,k}) = e^{(-\lambda_{0,l})}\frac{\lambda_{0,l}^{x_{l,k}}}{x_{l,k}!}$ and $f_l^{(1)}(x_{l,k}) = e^{(-\lambda_{1,l})}\frac{\lambda_{1,l}^{x_{l,k}}}{x_{l,k}!}$ with $\lambda_{1,l} > \lambda_{0,l}$ for all sensors. The likelihood ratio at each sensor is then $e^{\lambda_{0,l}-\lambda_{1,l}}\left(\frac{\lambda_{1,l}}{\lambda_{0,l}}\right)^{x_{l,k}}$. It is then clear that the thresholds comparisons of the likelihood are reduced to thresholds comparisons on $x_{l,k}$. For example in the case of a binary message alphabet $\{0,1\}$,

$$V_{l,k} = \begin{cases} 1, \text{ if } x_{l,k} > \frac{\log\beta + \lambda_{1,l} - \lambda_{0,l}}{\log\lambda_{1,l} - \log\lambda_{0,l}} \\ 0, \text{ if } x_{l,k} \leq \frac{\log\beta + \lambda_{1,l} - \lambda_{0,l}}{\log\lambda_{1,l} - \log\lambda_{0,l}} \end{cases} \tag{2.6}$$

In the asymptotic case, in which we do not care about bandwidth constraints and are allowed to use an arbitrarily large message alphabet, $g_l^{(i)} \to f_l^{(i)}$.

2.3.1.2 CUSUM of aggregated traffic

For this approach we send $x_{l,k}$ directly to the fusion center, without computing the local likelihood. Again, we will assume i.i.d. distributions at the sensors with $f_l^{(i)}(x_{l,k}) = e^{(-\lambda_{i,l})}\frac{\lambda_{i,l}^{x_{l,k}}}{x_{l,k}!}$. It is known that the aggregate X_k will be distributed as

$$f^{(i)}(x_k) = e^{-\lambda_i}\frac{\lambda_i^{x_k}}{x_k!} \tag{2.7}$$

where $\lambda_i = \sum_{l=1}^{L}\lambda_{i,l}$ $(i \in \{0,1\})$. The log-likelihood ratio at the fusion center needed to apply the CUSUM algorithm presented in the introduction is thus $z_k(x_k) = (\lambda_0 - \lambda_1) + x_k \log\left(\frac{\lambda_1}{\lambda_0}\right)$.

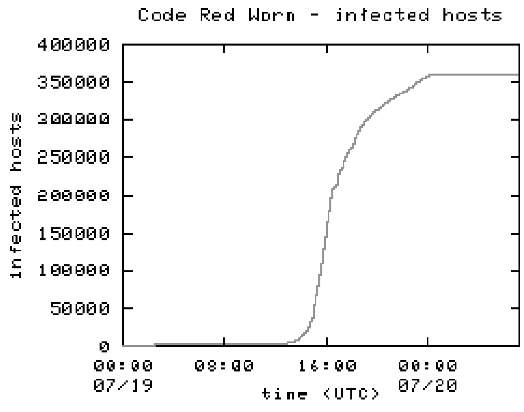

Figure 2.1: Code Red I Infection (source CAIDA)

2.3.2 Detection of an exponential signal in noise

The i.i.d. assumption of the observations after the change is not valid because each infected host will try in general to scan the same number of hosts in a given interval of time, and as more and more hosts become infected $x_{l,k}$ will increase with k. In particular we know from simple population dynamic models that a worm scanning uniformly randomly the whole network will follow a logistic growth [7].

In figure 2.1 we see the number of infected hosts by Cod Red I during its first day. In figure 2.2 we see the growth in number of probes by the worm w32.Leave.

Let η be the population of infected hosts. Let r be the intrinsic growth rate (the growth rate when $\eta(t)$ is small) and let a be a given positive constant. Then

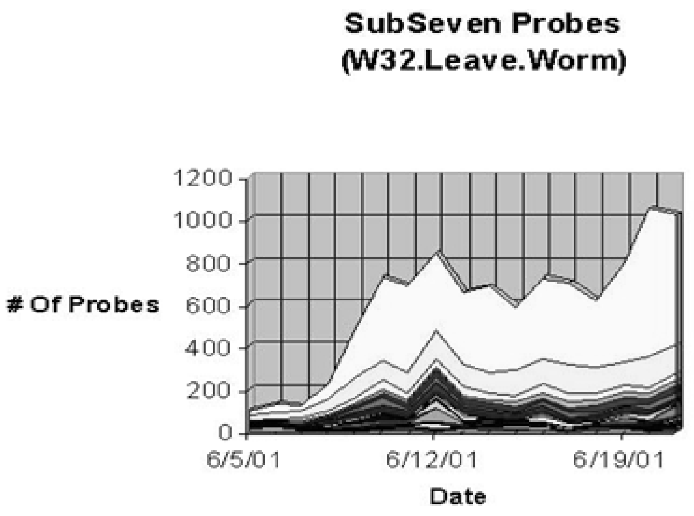

Figure 2.2: Number of probes due to the w32.Leave worm

the logistic growth satisfies the nonlinear ordinary differential equation

$$\frac{d\eta}{dt} = (r - a\eta)\eta \tag{2.8}$$

with solution

$$\eta(t) = \frac{N_0 B}{N_0 + (B - N_0)e^{-rt}} \tag{2.9}$$

where $B = r/a$ and N_0 is the population at time 0. Since we are interested in detecting a worm as soon as possible we look at the behavior of $\eta(t)$ when it is small, i.e. we consider the exponential growth

$$\eta(t) = N_0 e^{rt} \tag{2.10}$$

The equivalent discrete time recursion obtained from difference equations is

$$\eta_k^d = N_0 m^k \tag{2.11}$$

(d stands for "discrete") where m is the discretized growth rate when η^d is small and N_0 is the number of hosts compromised at $k = 0$. The discretized version of the continuous time equation (2.10) is

$$\eta(k\Delta t) := \eta_k = N_0 e^{rk} \tag{2.12}$$

We can see that η_k matches η_k^d when $r = \log(m)$. We can use η_k and η_k^d interchangeably although for the cases in which we need integer values, η_k^d will be easier to use.

For the detection problem we will assume that the values of N_0 and r (or m) are unknown. Moreover we will consider that the normal traffic $W_{l,k}$ at each sensor l, is distributed under a (probably unknown) distribution $f_l^{(0)}(w_{l,1}, ..., w_{l,k})$ for $k \in \mathbb{N}$ giving a normal traffic aggregate

$$W_k = \sum_{l=1}^{L} W_{l,k} \tag{2.13}$$

distributed as $f^{(0)}(w_1, ..., w_k)$.

Our main assumption in this section is that the number of probes seen at the sensors will be proportional to the number of infected hosts $\alpha\eta_k^d$. The usual change detection hypothesis testing problem for the aggregate traffic (2.1) would be as follows:

$$H_0 : x_k = w_k \qquad \text{when } 1 \leq k \leq M$$

$$H_1 : \begin{array}{l} x_k = w_k \qquad \text{when } 1 \leq k < k_0 \\ x_k = \alpha\eta_k^d + w_k \text{ when } k_0 \leq k \leq M \end{array}$$

However, we want k to restart to 1 whenever H_0 is accepted, so we use a sequential hypothesis testing where the change time k_0 is implicitly given by the time in which the sequential test restarted and H_1 was accepted.

$$H_0 : x_k = w_k \qquad \text{when } 1 \leq k \leq M$$

$$H_1 : \; x_k = \alpha\eta_k^d + w_k \text{ when } 1 \leq k \leq M$$

In order to detect a signal in noise we must compute the generalized likelihood ratio in the time window [1,...,M] and compare it to the threshold h. To accommodate different growth rates, the intervals Δt of the time steps k can increase or decrease.

2.3.2.1 Exponential signal detection in noise

The detection of the signal $\alpha\eta_k^d$ in noise w_k is achieved with the test:

$$\frac{\sup_{\alpha,N_0,m} f^{(0)}(x_k - \alpha\eta_k^d)}{f^{(0)}(x_k)} \underset{H_0}{\overset{H_1}{\gtrless}} h \tag{2.14}$$

Note that the parameters α and N_0 can be combined into a single parameter A_0 for the optimization problem.

We will start by assuming that the normal traffic $W_{l,k}$ is i.i.d. with distribution $f_l(w_{l,k}) = e^{(-\lambda_l)}\frac{\lambda_l^{w_{l,k}}}{w_{l,k}!}$. So under H_0, X_k is then distributed as

$$f^{(0)}(x_k) = e^{-\lambda}\frac{\lambda^{x_k}}{x_k!} \tag{2.15}$$

where $\lambda = \sum_{l=1}^{L} \lambda_l$.

The discrete noise distribution presents the problem that the optimization algorithm in (2.14) has to restrict $\alpha\eta_k^d$ to integer values. Note that in this case $\forall(A_0,\ m), \forall(k,j)\ \left(x_k - \eta_k^d(A_0,m)\right)$ is independent of $\left(x_j - \eta_j^d(A_0,m)\right)$. So we can compute the likelihood ratio for each time step k. The likelihood ratio function for time step k is then

$$\max_{A_0,m} \frac{\lambda_0^{x_k}\lambda_0^{-A_0m^k} x_k!}{\lambda_0^{x_k}(x_k - A_0m^k)!} = \max_{A_0,m} \lambda_0^{-A_0m^k} x_k(x_k-1)\cdots(x_k - A_0m^k+1). \tag{2.16}$$

For the whole time window [1,...,M] the log-likelihood ratio is then

$$\max_{A_0,m} \sum_{k=1}^{M} \left(-A_0m^k \log(\lambda_0) + \sum_{j=0}^{A_0m^k-1} \log(x_k - j) \right) \tag{2.17}$$

The maximization is subject to the constraints $x_k - A_0m^k \geq 0$, $m > 1$ and $A_0m^k \in \mathbb{Z}^+$. This equation cannot be solved assuming continuous variables and rounding up or down the optimum values found. We must therefore rely on combinatorial optimization.

2.3.2.2 Exponential change in the mean

In this subsection we will assume the same noise model as in subsection 2.3.2.1. We will assume that the mean of $f^{(0)}$ varies with respect to k via an exponential growth model. The advantage of this approach is that we can use standard optimization algorithms over $\mathbb{R}^k$ for the GLR method, as the mean can take any real value. The disadvantage of this approach is that it is not truly a problem of signal detection in noise. Furthermore for certain discrete random variables such as those with a Poisson distribution, if the mean changes the variance also changes.

Under the first approach to H_1: the exponential growth in the mean, X_k will be distributed as

$$f^{(1)}(x_k; A_0, r) = \max_{A_0, r} e^{(-\lambda - \eta_k(A_o, r))} \frac{(\lambda + \eta_k(A_0, r))^{x_k}}{x_k!} \tag{2.18}$$

where the dependence on the unknown nuisance parameters A_0 and r is stated explicitly. Following the GLR algorithm given in the introduction we could compute

$$g_M = \max_{1 \le k \le M} \log \frac{f^{(1)}(x_k; A_0, r)}{f^{(0)}(x_k)}$$

However since the signal we are trying to detect is exponential we prefer to avoid the maximization step over k and use a batch approach with a sliding window of size M. Note that given fixed $(A_0,\ m)$, for all $(k, j) k \neq j\ \ (x_k)$ is independent of (x_j), so we can compute the likelihood ratio for each time step k. The likelihood ratio at time step k is then

$$\max_{A_0, r} \frac{e^{-(\lambda_0 + A_0 e^{rk})} (\lambda_0 + A_0 e^{rk})^{x_k}}{e^{-\lambda_0} \lambda_0^{x_k}} = \max_{A_0 r} e^{-A_0 e^{rk}} \left(1 + \frac{A_0 e^{rk}}{\lambda_0}\right)^{x_k} \tag{2.19}$$

For a window [1,...,M] the log-likelihood ratio is:

$$\max_{A_0, r} \sum_{k=1}^{M} \left(-A_0 e^{rk} + x_k \log\left(1 + \frac{A_0 e^{rk}}{\lambda_0}\right)\right) \tag{2.20}$$

The maximization is subject to the constraints $A > c_1$ and $r > c_2$, where c_1 and c_2 are two given positive constants.

2.3.2.3 Nonparametric regression detection

So far we have always been assuming a parametric distribution $f^{(0)}(w_1, ..., w_k)$ for the normal traffic. This assumption is valid for a wide number of ports as the traffic seen can be regular. However in some cases the real distribution can be quite difficult to obtain. For example the number of probes seen to port 80 (http) or port 21 (ftp) for computers providing those services can exhibit long range dependence and multifractal behavior.

In order to deal with some of the more complicated problems, in particular those where no clear mean of $f^{(0)}(w_1, ..., w_k)$ can be established, we propose a nonparametric change detection algorithm.

The idea is to do a linear regression on $log(x_i)$. This regression will produce two parameters, a slope c_k and the error err_k from the estimated regression of $x_{k-M+1}, ..., x_k$. From this we compute the statistic $z_k = c_k/err_k$. We use a sliding window over k to compute the statistic. Then we apply the non-parametric version of CUSUM or the Girshik-Rubin-Shiryaev algorithms to z_k.

2.4 Simulation

2.4.1 Worm simulation

We created different scale-free network topologies with a delay of one unit step per edge for simulating the communication among different nodes. One of our main

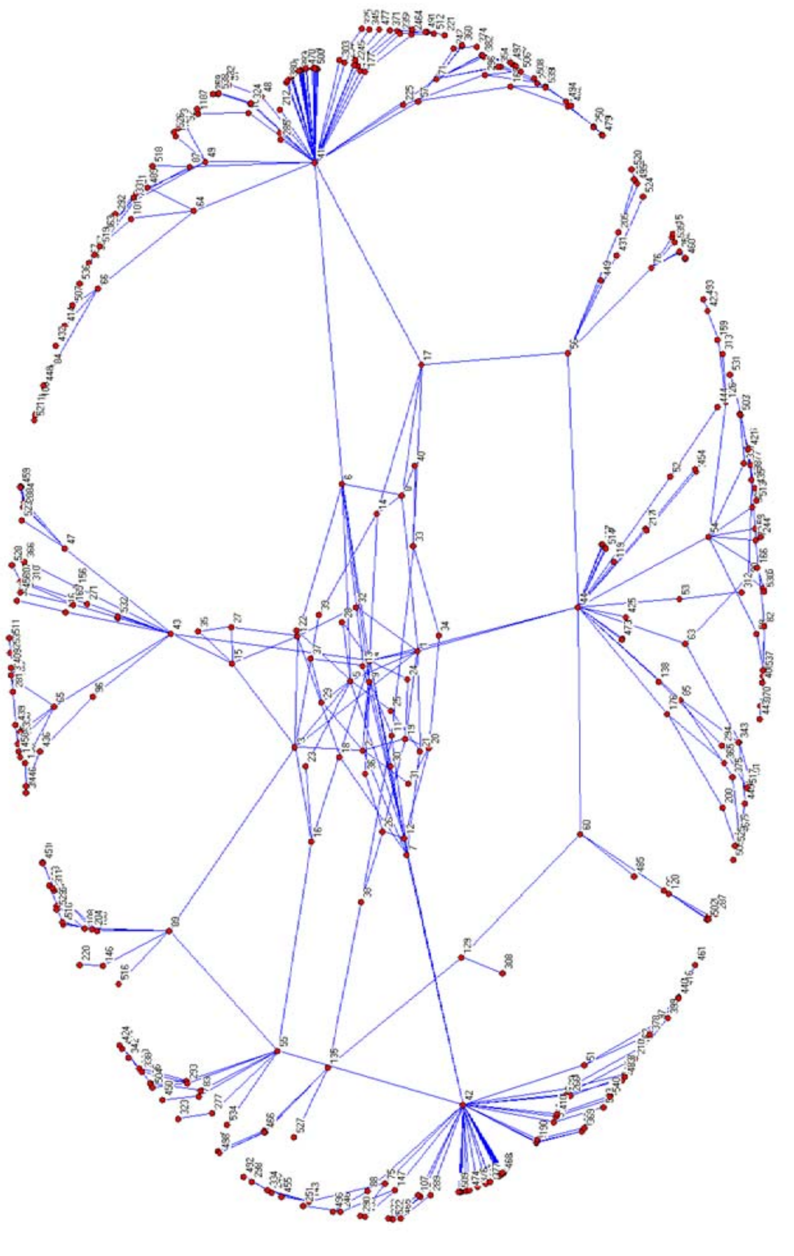

Figure 2.3: One of the Simulated Networks with 540nodes

algorithms was a modification to the $Albert - Barabási$ (AB) algorithm [5]. In this setup we build a cluster of nodes using the (AB) algorithm of size $L_{cluster}$ where each new node is included with degree $d_{cluster}$. Afterward we add L_{AS} nodes ($L_{AS} = 4$ in our simulations) with two edges connecting them to random nodes in the cluster. From this point forward we continue to include nodes of degree 1 selecting an edge from any node except from the node in the clusters, so this growth simulates the creation of 4 autonomous systems (AS) connected through a high degree cluster. At the end the L_{AS} nodes (selected among all nodes created after the first $L_{cluster} + L_{AS}$ nodes) with the highest degree are connected to the high degree cluster to reduce the bottlenecks from nodes in the subnetworks to reach the cluster. A typical result of this algorithm is shown in figure 2.3.

The simulations can be placed in the framework of the distributed IDS presented such as myNetwatchman and Symantec's Deep Sight Threat Management system. Each node represents the IDS of a given subnetwork and the worm scans different computers in different subnetworks until it finds a vulnerable machine. However the simulation environment we present is more closely related to the widespread of sensors in the Internet backbone [7], because each node acts as a router and will see transit probes (e.g. a leaf router). Under this framework the worm spreads by using hit-list scanning because all the initial probes find a vulnerable machine. Therefore no attack probe is wasted scanning immune machines.

For the normal flow model, each node tries to send a normal probe every T units of time to a uniformly randomly selected node in the network. Each node sends the probes independent of any other node. T was assumed to be a random variable either Exponential (with time delay equal to the average round trip time of the network) or Pareto distributed (with parameters a=1.5 and b=1). At the change time there is only one abnormal node. The node tries to contact another node at random to reproduce the worm. The malicious node then waits four round trip times to start searching for another vulnerable node. This waiting time was selected as a model of the three way handshaking plus the small upload of the worm found in real networks. In figure 2.4, the first subfigure shows the time in which each particular node got infected. The second subfigure shows the number of infected hosts. The infection starts at time 500 and by time 750 more than 85% of the nodes are infected.

2.4.2 Application of the detection algorithms

- By "fusion" we refer to the algorithm presented in section 2.3.1.1. The mean $\lambda_{0,l}$ of the assumed Poisson distribution $g_l^{(0)}$ was estimated during the first 200 time steps for all l. The mean of the anomalous Poisson distribution was set arbitrarily to $\lambda_{1,l} = 1.2\lambda_{0,l}$.

- By "aggregate cusum" we refer to the algorithm of section 2.3.1.2. The mean λ_0 of the assumed Poisson distribution $f^{(0)}$ was again estimated for the aggregate traffic during the first 200 time steps. The anomalous Poisson distribution was also set to $\lambda_1 = 1.2\lambda_0$.

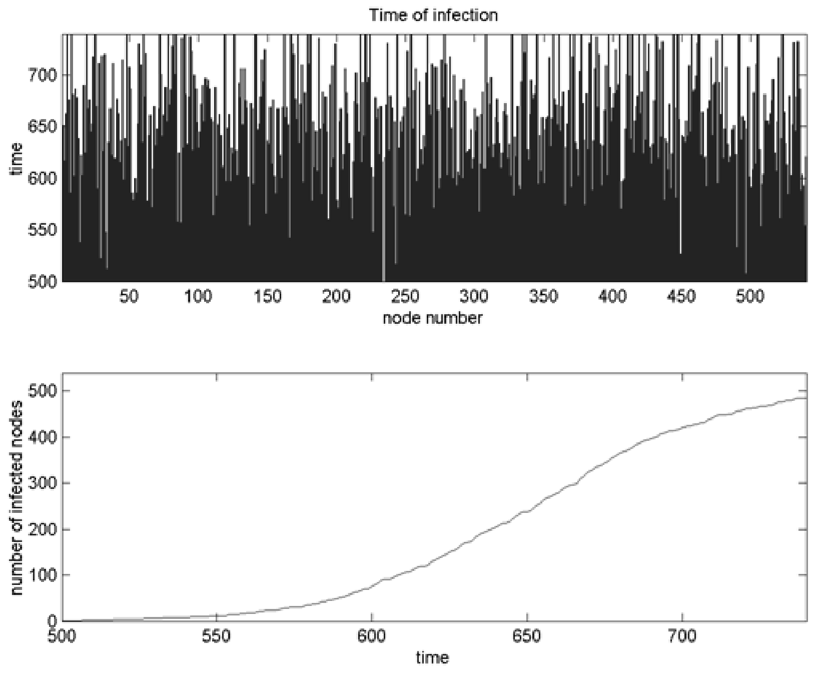

Figure 2.4: Infection time

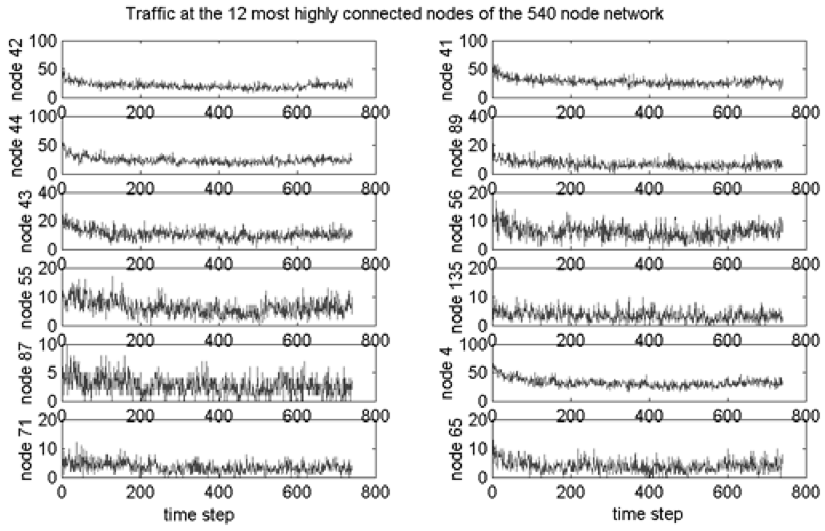

Figure 2.5: Number of probes at the 12 highest degree nodes.

- By "exponential in noise" we refer to the algorithm of section 2.3.2.1. We used combinatorial optimization for different time windows as the optimization function in equation (2.17) cannot be maximized using continuous variables and rounding up at the end. The integer restriction gives us a problem, as m^k for $m \in \{2, 3, ...\}$ and $k \in \mathbb{N}$ grows extremely fast. Our solution was to select two windows. In the first window of length 5 the median of $X_{k-4}, X_{k-3}, X_{k-2}, X_{k-1}, X_k$ is obtained as $\hat{X}_k$. The second window of length 4 is taken among $\hat{X}_{k-3}, \hat{X}_{k-2}, \hat{X}_{k-1}, \hat{X}_k$. It is in this window that the optimization in equation (2.17) takes place.

- By "exponential change in mean" we refer to the algorithm of section 2.3.2.2. A variant of this algorithm was obtained by setting A_0 fixed and maximizing over r. The nonparametric CUSUM algorithm was applied to the optimal estimate $\hat{r}$. This algorithm will be called "r nonparam cusum". Proving the concavity of the function we want to maximize (equation (2.20)) is very difficult. We instead computed the Hessian and checked if it is negative definite at the solution of the gradient. For the simple case of fixed A_0 the gradient of the function is given by $\frac{\partial}{\partial r} \sum_{k=1}^{M} \left(-A_0 e^{rk} + x_k \log \left(1 + \frac{A_0 e^{rk}}{\lambda_0} \right) \right) = \sum_{k=1}^{M} \left(-k A_0 e^{rk} + x_k \frac{k A_0 e^{rk}}{A_0 e^{rk} + \lambda_0} \right)$, so the optimization relies on solving a nonlinear function. The Hessian is $\sum_{k=1}^{M} \left(-k^2 A_0 e^{rk} + x_k \frac{\lambda_0 k^2 A_0 e^{rk}}{\left(A_0 e^{rk} + \lambda_0 \right)^2} \right)$. When we optimize also with respect to A_0 the problem is reduced to solving a set of nonlinear equations, but we used mostly optimization methods without the gradient computations.

- By "regression" we refer to the algorithm presented in section 2.3.2.3. A variant of this algorithm is called "robust regression", and was obtained by using the iterated weighted least squares estimate of the slope. This robust approach was selected as a way to cope with outliers without including directly the error (in the least square fit) in the change statistic. The slope c_k^{robust} obtained at each time window $[k-M+1, ..., k]$ was used as the stochastic process for the nonparametric CUSUM algorithm.

2.4.3 Detection results

The following detection delay plots are the worst case detection delay as all change detection statistics are reinitialized to zero at the time of infection. Furthermore all detection delays were obtained by averaging 10 runs.

Our first performance metric was the detection delay *vs* the number of sensors under an exponential waiting time for the normal traffic. The sensors were picked in order starting from the highest degree nodes. Surprisingly, for our simulated network in figure 2.3, the detection delay does not seem to decrease monotonically as the number of sensors increases, or even with different network sizes, except for the regression statistics (figure 2.7.)

According to our network construction algorithm we have four subnetworks and each is connected on average by two gateways to the highly connected cluster

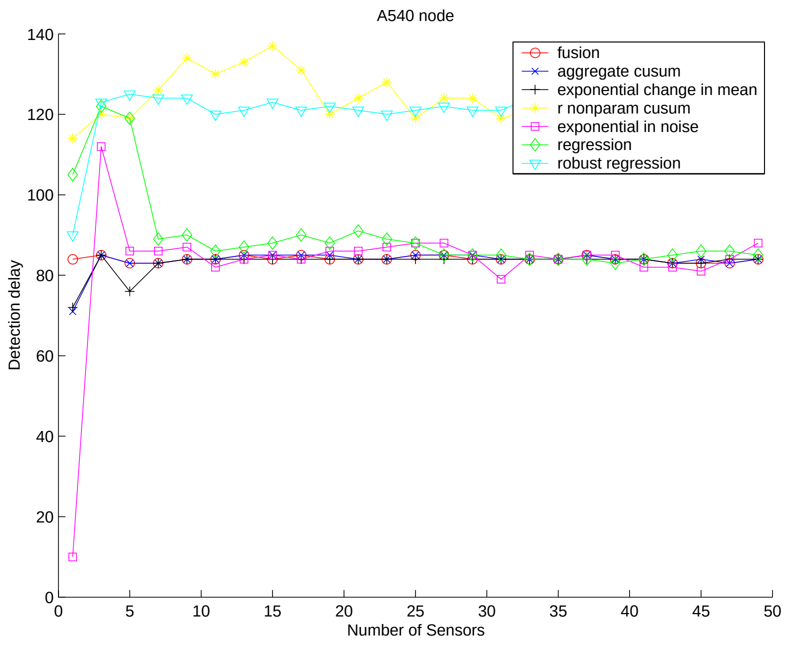

Figure 2.6: Detection delay *vs* high degree node aggregation, 540 nodes.

and thus any node within that subnetwork trying to reach a node outside will send packets through this gateways so they tend to contain most of the traffic. To test if the result was a consequence of our four subnetwork constructions we generated a scale-free network based on the original (AB) algorithm. During the construction each new node is attached to the network with m edges. For the figure 2.8 we chose $m = 2$.

The results seem to suggest that in a general scale-free network a very small set of the highly connected nodes is enough to detect any global variation on the network flow, and that increasing the number of sensors does not on average decrease -or increase- the detection delay, in fact on average it remains constant.

However if the sensors are picked at random then aggregation helps according

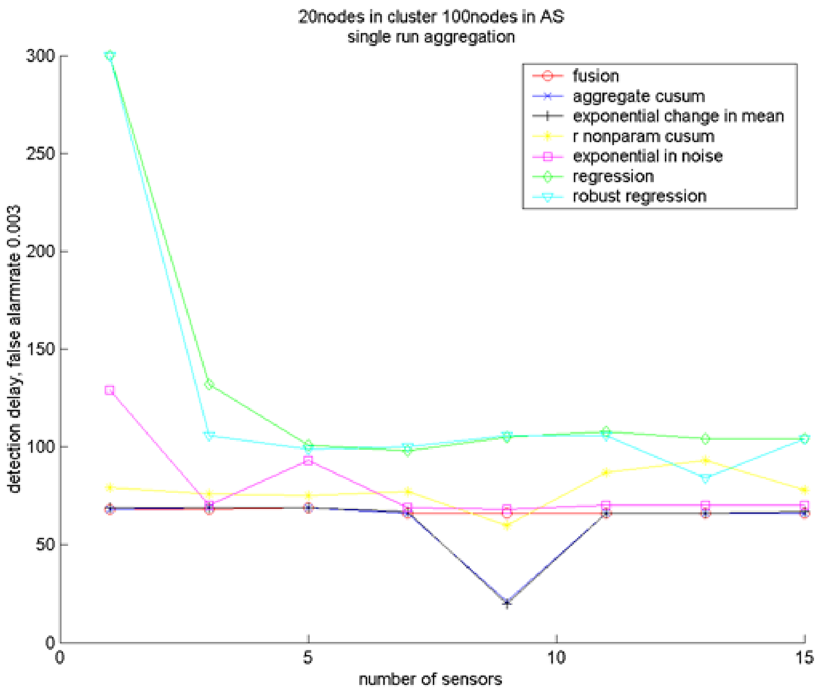

Figure 2.7: Detection delay *vs* high degree node aggregation, 120 nodes

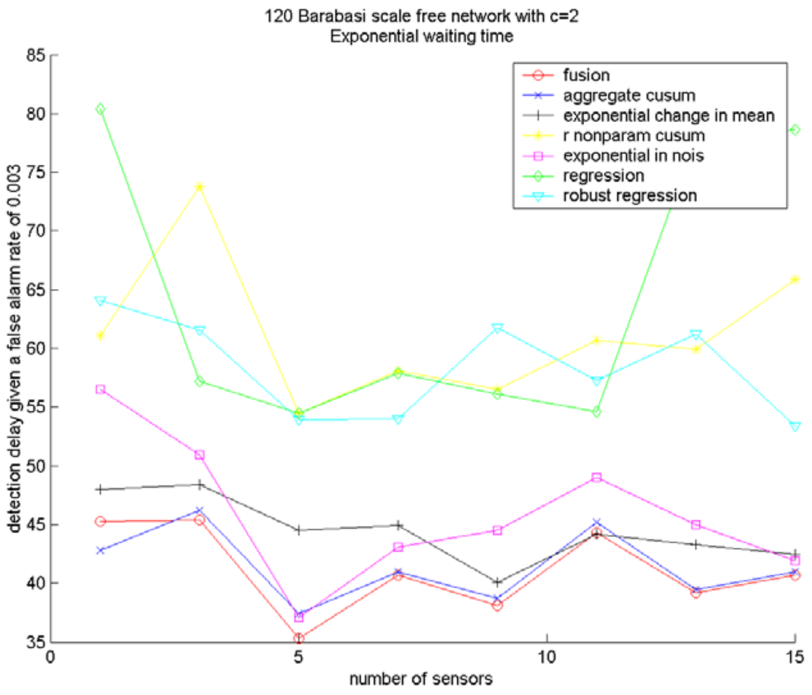

Figure 2.8: Detection delay *vs* high degree node aggregation for (AB) network

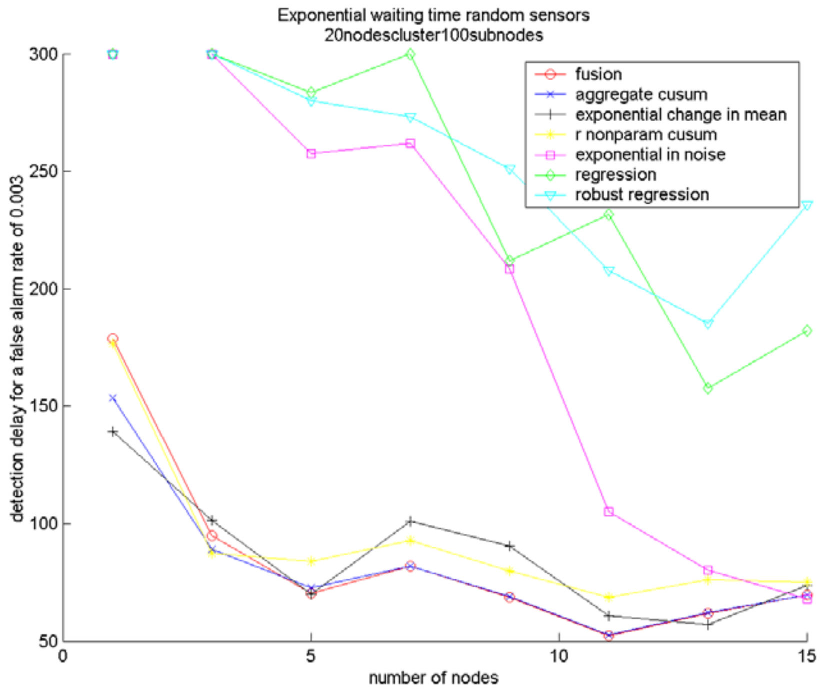

Figure 2.9: Detection delay *vs* node aggregation selecting the nodes at random

to figure 2.9.

Furthermore if we select the highest degree nodes in a random network (random graph), the aggregation also helps. See figure 2.10.

Partial answers can be found by the degree distribution in figure 2.11 of our original 540 node network. As should be expected very few nodes have a high degree that captures most of the traffic in the network. The high degree nodes for the original 540 node network are given in table 2.1. This is consistent with the heavy tailed degree distribution of the (AB) model network in figure 2.12. The degree distribution of a random network follows a Poisson distribution and thus the number of high degree nodes is larger (figure 2.13.)

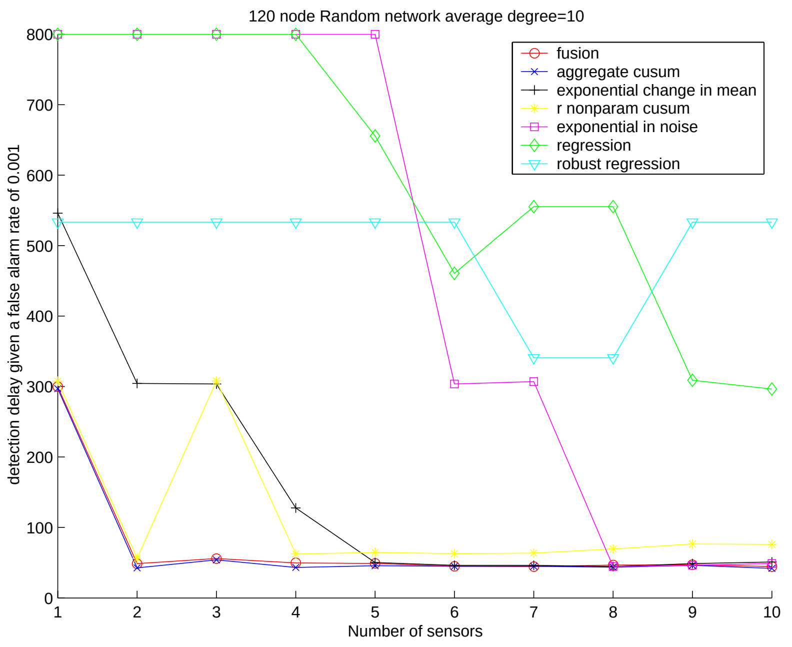

Figure 2.10: Detection delay *vs* high degree node aggregation in a random network.

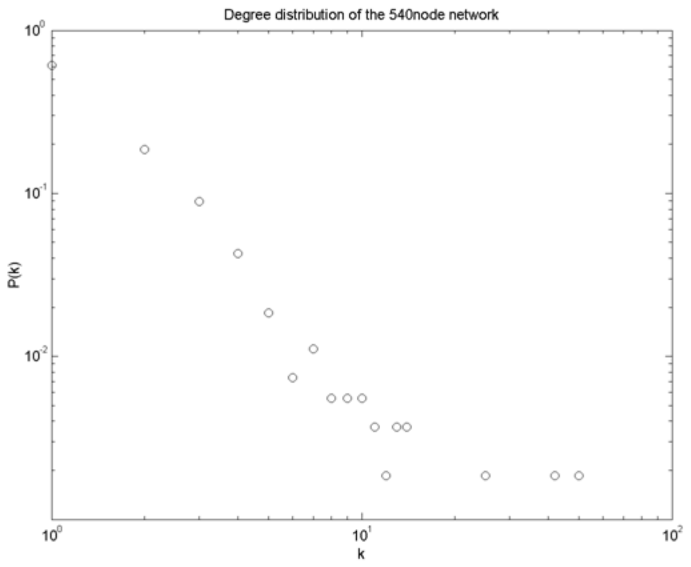

Figure 2.11: Degree distribution of the 540 node network

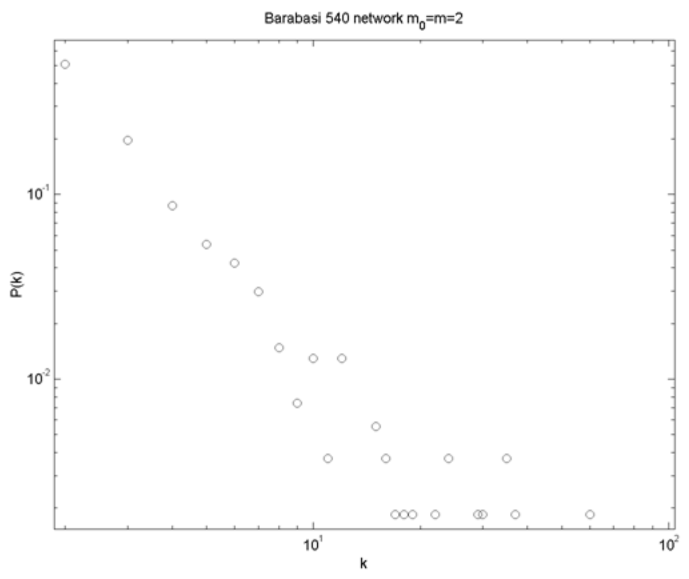

Figure 2.12: Degree distribution of a 540 node network created with the (AB) algorithm

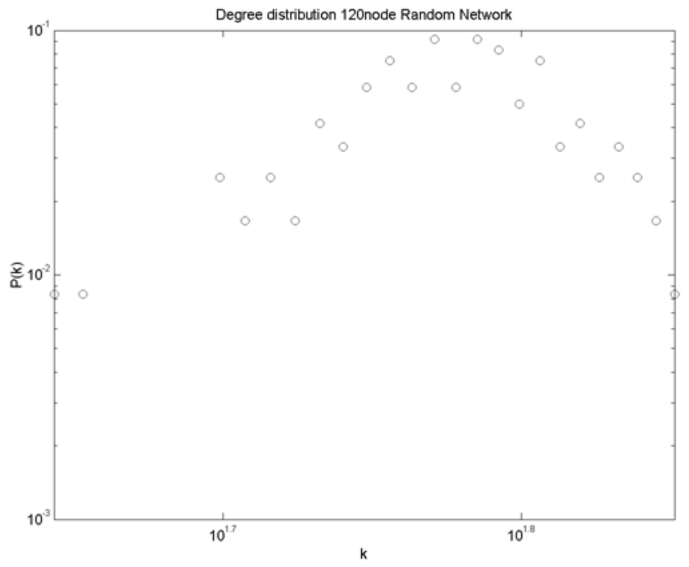

Figure 2.13: Degree distribution of a random network

Node	42	41	44	89	43	56	55	135	87	4
Degree	50	42	25	14	14	13	13	12	11	11

Table 2.1: High degree nodes for the network in figure 2.3

So far most of the statistics have performed similarly except for "regression", "robust regression" and the "r nonparam cusum". The "exp in noise" statistic also suffers when the nodes aggregated are selected at random (figure 2.9) and in a random network (figure 2.10.) This result is due to the constrained optimization as discussed before. When we aggregate random sensors in a scale-free network, or when we monitor any node in a random network, the aggregated overall traffic is smaller and thus it is very difficult to fit a fast exponentially increasing function (equation (2.17)) in small intervals.

To test the effectiveness of the nonparametric statistics we then simulated the waiting time of the normal traffic with the Pareto distribution (see figures 2.14 and 2.15).

The detection delay *vs* the average time for false alarm can be seen in figures 2.16, 2.17 and 2.18.

2.5 Conclusions

In scale-free networks a very small set of the highly connected nodes is sufficient for detection and aggregation only improves the performance of the nonparametric

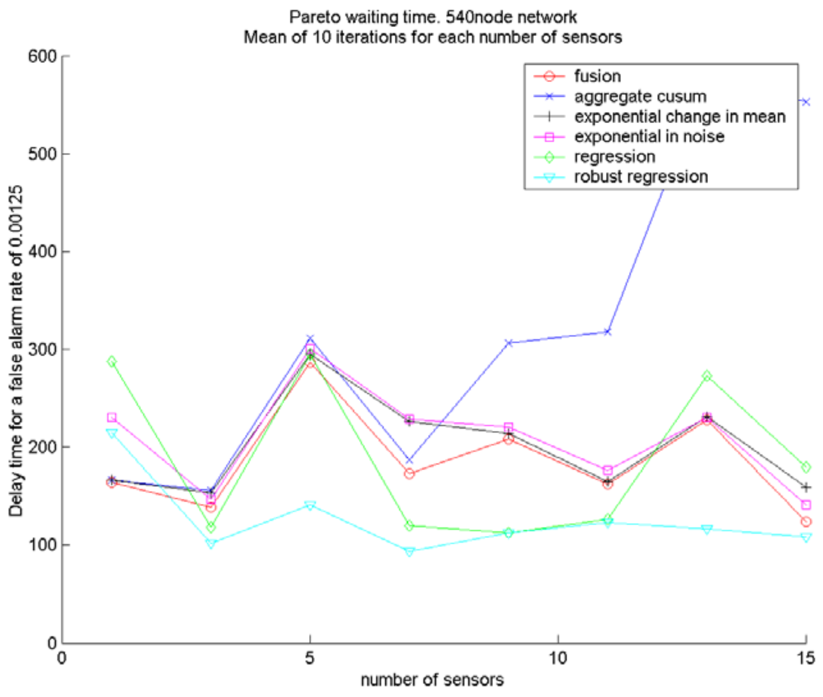

Figure 2.14: Detection delay under Pareto traffic 540 node network

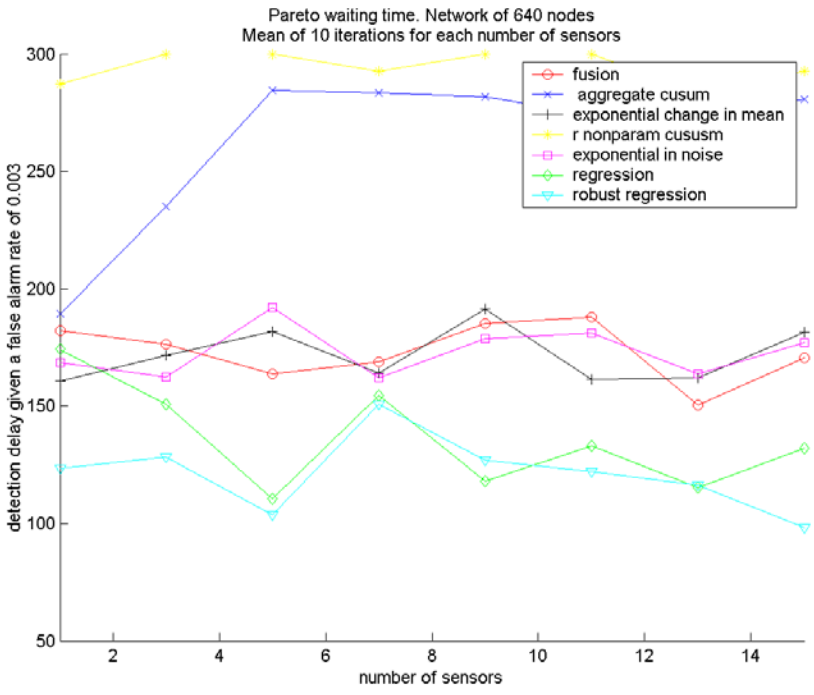

Figure 2.15: Detection delay under Pareto traffic 640 node network

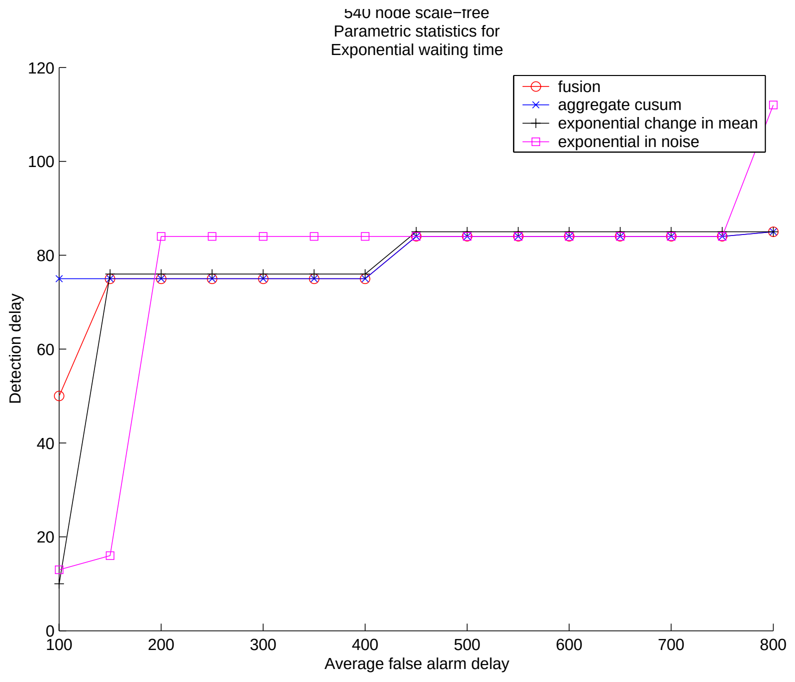

Figure 2.16: Detection delay *vs* false alarm rate 540 node network.

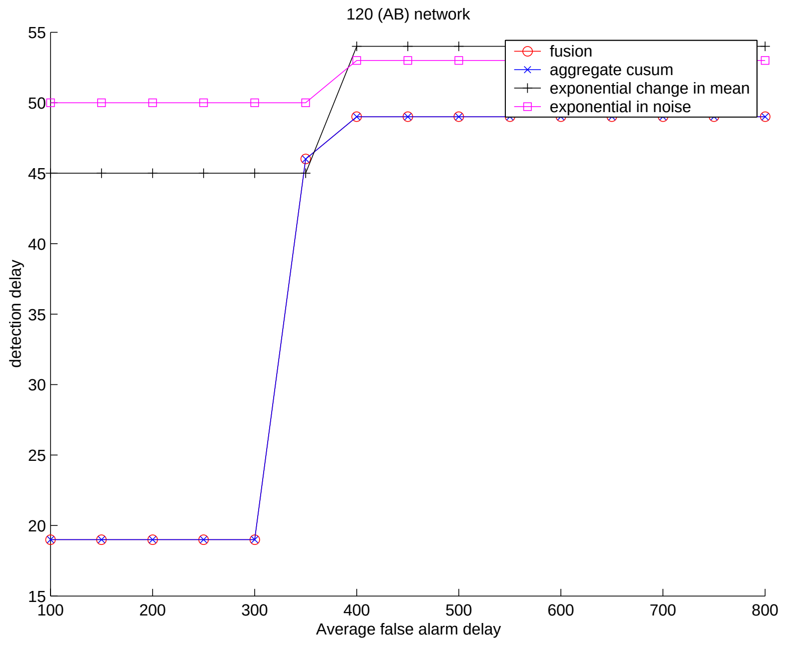

Figure 2.17: Detection delay *vs* false alarm rate (AB) network

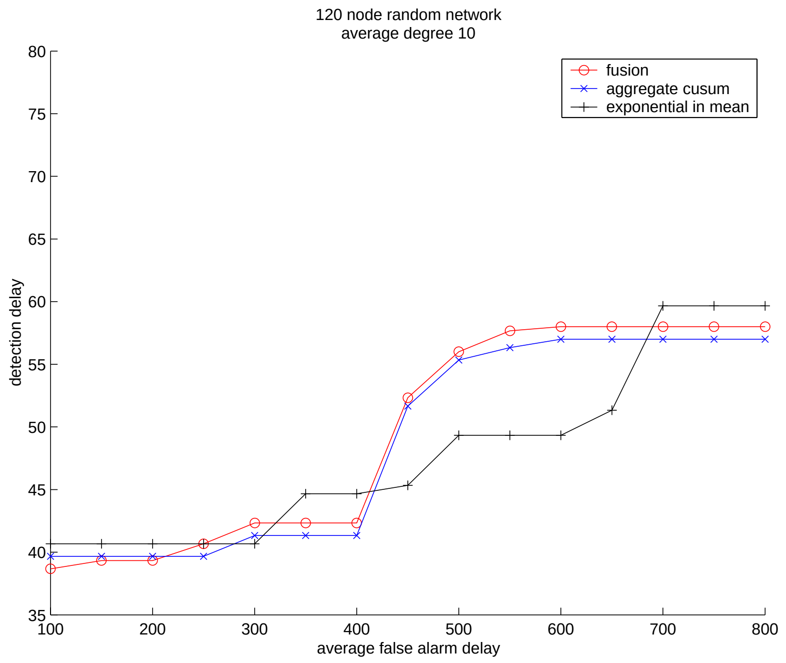

Figure 2.18: Detection delay *vs* false alarm rate for a random network

statistics. If we select sensors at random or if we monitor a random network then aggregation is very important for detection. Most of the parametric statistics perform comparably under a wide variety of conditions. The best performance on average is always obtained by the statistics measuring a change in the mean, mainly "fusion", "aggregate CUSUM" and "exponential change in mean". When the traffic deviates significantly from the assumed traffic distribution, the best performance is produced by the nonparametric statistics "regression" and "robust regression".

The large variability in the aggregation can be accounted in part to the small spikes that cross the threshold before the real statistic grows (lucky alarms). For

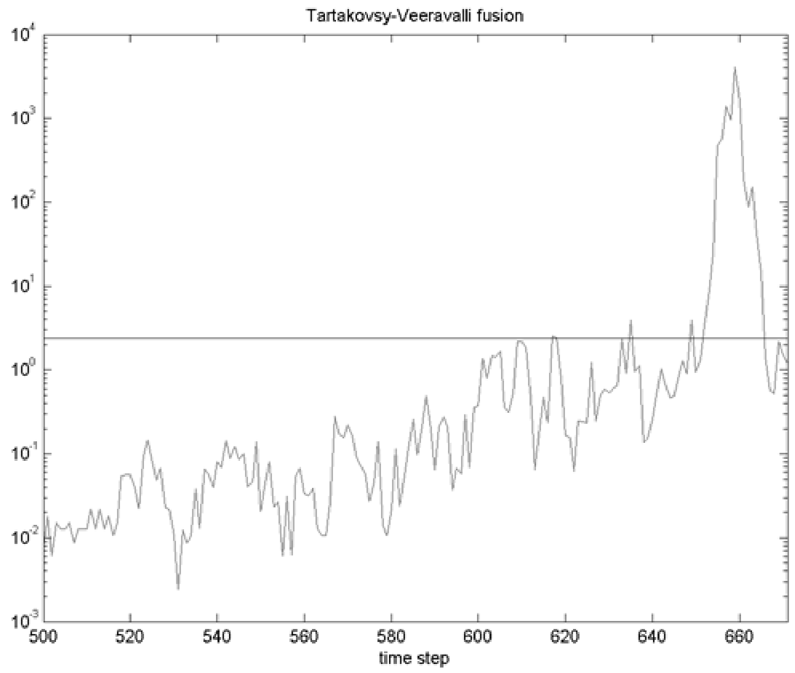

Figure 2.19: Fusion statistic under Pareto traffic

example in figure 2.19 the alarm was set at time 610, even when the real growth in the statistic was after time 650.

Chapter 3

Detection of Distributed Denial of Service Attacks

3.1 Description of the attack

A denial of service attack (DoS) can be defined as an attack designed to disrupt or completely deny legitimate users' access to networks, servers, services or other resources. The most common DoS attack involves sending a large number of packets to a destination causing excessive amounts of network endpoint bandwidth to be consumed and (or) cpu processing rate at the destination.

In a distributed denial of service (DDoS) attack typically an attacker compromises a set of Internet hosts (using manual or semiautomated methods like a worm)

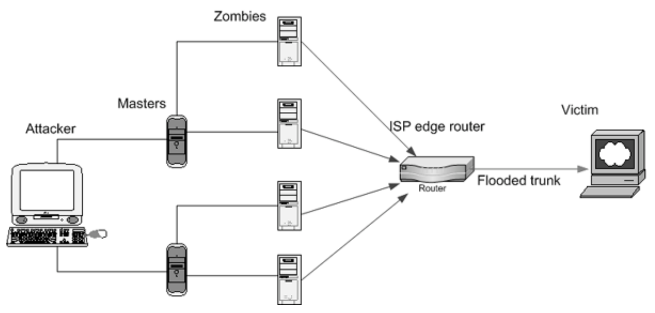

Figure 3.1: The attacker creates a zombie army to flood a target.

and installs a small attack daemon on each host, producing a group of ”zombies” (figure 3.1). This daemon typically contains both the code for producing a variety of attacks and some basic communications infrastructure to allow for remote control. Using variants of this basic architecture an attacker can focus a coordinated attack from thousands of zombies installed across different autonomous systems (AS) boundaries, onto a single site causing excessive amounts of endpoint and possibly transit network bandwidth to be consumed [9].

3.1.1 Persistence and characterization of DDoS attacks

In [10] an estimate of 12,805 attacks was provided over the course of three weeks in February 2001. Most of the attacks were short, 50% were less than 10 minutes and 80% were less than 30 minutes. However the tail of the distribution was long: 1% of the attacks were greater than 10 hours. Although most of the attacks are targeted against home machines, suggesting that minor DDoS are used to settle personal vendettas, there is a small but significant fraction of attacks directed against network infrastructure. 2-3% of attacks target domain name servers (DNS) and 1-3%, target routers (with a disproportionately large number of packets.)

According to Alan Paller, director of the SANS Institute, DDoS attacks are not going to be solved because we get some new hardware in the system, but by re-engineering the whole Internet.

3.1.2 Recent examples of DDoS attacks

One of the more complex and elaborate DDoS attacks was the attack on October 22, 2002 to the 13 main DNS root servers. The attack flooded with ICMP messages the 13 DNS root servers around the world with more than 10 times the normal amount of data. Seven of the servers were affected enough to have periods of "zero-reachability". It took about an hour for security specialists to enact defensive measures and restore service. The attack failed to disrupt service because the data on the 13 key servers is replicated tens of thousands of times by ISPs and other computers around the world. Only an estimated 6% of the requests reached the root DNS servers. These attacks were easy to defend against because they relied on a simple ICMP flood which is not typically seen in very high volumes. Instead of sending a flood of packets that all use the same protocol, attackers usually disguise a DDoS attack as normal traffic. In such an attack, nothing about the protocols used or the packets sent would appear unusual, but the volume of traffic would be enough to overwhelm the targeted server. Even more pernicious would be attacks that target the routing infrastructure (as opposed to the DNS infrastructure) of the Internet according to security engineers at Arbor Networks.

3.1.3 Related work

Several vendors offer early-detection and response tools for dealing with DoS attacks. The main focus of such technologies is to quickly give IT managers the information needed to filter out malicious traffic while letting in legitimate users.

Most of these products work by comparing live network traffic against some previously defined baseline and by alerting users if there is a significant divergence from that baseline. The alerts might be based on a comparison of byte or packet rates, traffic that's directed at specific resources, traffic that originates from specific IP addresses or spikes in network traffic.

The change detection approach to detect denial of service attacks has also been addressed in [11, 12] where the detection is done at the victim's network. We will try to present new algorithms for the detection in the upstream “transit” network.

3.2 Why is it important to quickly detect routers participating in a denial of service attack?

Almost all DDoS attacks involve multiple networks and attack sources, many of which have spoofed IP addresses to make detection even harder. So completely choking off the offending traffic requires network administrators to call upstream service providers, alerting them of the attack and having them shut down the traffic. That process has to be repeated all the way back to every attack source. So although DDoS are easily identified at the victim's site, it is natural to extend the quickest detection problem to the transit network.

There are various techniques and ideas for mitigation of denial of service attacks that require the identification of the routers participating (involuntarily) in the attack. Most of these techniques consume a significant amount of router resources so it is advisable to use them only when needed. Some examples that are provided in some Cisco Routers are TCP Intercept and Committed Access Rate. Some related work is also presented in [13, 14] where the key step in the proposed denial of service mitigation algorithm consists in identifying the routers forwarding the malicious traffic.

Network administrative policy influences how the nodes are monitored and what to do once we start correlating alarms. Does it make sense to monitor all routers? Or does this monitoring consume too many resources? Are the nodes monitored computing the statistics necessary to detect a local change by themselves? Or are they sending all the information (packets per link) to compute the statistics at the network-operating center (a fusion center). A reasonable assumption for transit networks carrying a lot of traffic which cannot be analyzed at line rate, is that routers do not keep the number of packets to a specific destination, as this might be too expensive during operation. Thus we are interested only in passively monitoring the network.

3.3 Detection Algorithms

Our main goal is to attempt a first approach to the problem of detecting when a distributed denial of service attack is taking place in one sub-network of a transit (core) network comprised only of routers.

Parametric and non-parametric change detection algorithms are applied to the problem of detecting changes in the direction of traffic flow. The directionality of the change in a network flow is assumed to have an objective or target. We are interested in the quickest detection problem when the attack is distributed and coordinated from several nodes against a targeted one.

We use a directionality framework, which gives us a way to compute the severity and directionality of the change. The severity represents a composite M-ary sequential hypothesis test that is easily solved under Gaussian assumptions.

We finally introduce a heuristic distributed change detection mechanism for correlating the alarms in a subset of monitored nodes. Given an alarm as a pair (direction and severity) we correlate the severity of the alarms with alarms from other nodes in the same direction. And we show that this is equivalent to another M-ary sequential detection problem among the routers.

3.3.1 Problem formulation

We take a new approach for identifying Distributed Denial of Service attacks by a set of nodes in a transit network. The basic idea is that at each highly connected node, the data tends to aggregate from the distributed sources toward the destination, giving a sense of directionality to the attack. This directionality idea will provide us a framework to design change detection algorithms that are going to be less sensitive to changes in the average intensity of the overall traffic and will focus on differentiating the different random fluctuations of the network traffic versus fluctuations where there is a clear change in the direction of the flow at a given node. We are considering packets in a very broad and general way, but clearly our approach can be extended to monitor certain specific packet types given the protocol. For example we might be interested in measuring only TCP SYN-ACK response packets for identifying a reflected distributed denial of service attack, or ICMP packets for identifying ping floods.

Let's assume we are monitoring node d in figure 3.2. Let $X_k^{d,m}$ denote the stochastic process representing the total number of packets sent by d through the link (d,m) (an ordered pair) at time step k, where $m \in \mathcal{N}(d)$ denotes a neighbor of d, and $\mathcal{N}(d)$ the set of neighbors of d. Let X_k^d denote the vector with the elements $X_k^{d,m}$ and let

$$\theta_0^d := \begin{bmatrix} E_0[X_k^{d,a}] \\ E_0[X_k^{d,b}] \\ E_0[X_k^{d,c}] \end{bmatrix} \tag{3.1}$$

We will be interested in changes of the form:

$$\theta_0^d + \nu \Upsilon_m \tag{3.2}$$

where ν is a non-negative scalar and Υ_m (in the case of three observed links $|\mathcal{N}(d)| = 3$) is one of the usual basis vectors of the three dimensional Euclidean

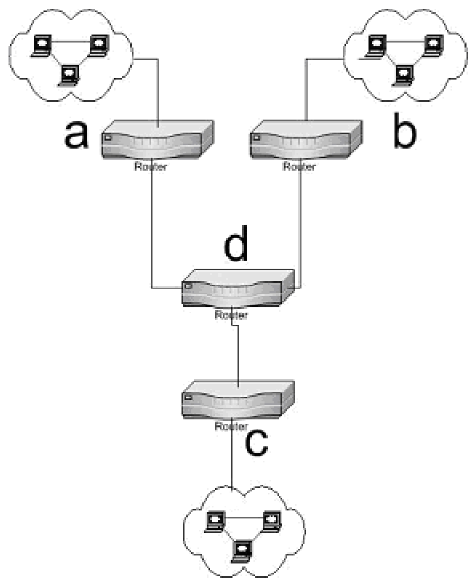

Figure 3.2: A transit network composed of nodes a, b, c and d.

space. Namely:

$$\Upsilon_a = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \Upsilon_b = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \Upsilon_c = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.3}$$

So in figure 3.2, if node d suddenly starts a broadcast, there will be a change in the mean of all processes. However we are not interested in such a change. Instead if there are attackers in the subnetworks attached to b and c, and they target a host in the network attached to a by flooding it, there will be a change in the direction Υ_a. Testing directions should help us in discriminating unwanted false alarms due to random fluctuations of the flows.

To formalize our ideas we consider the framework discussed in [2] of change detection in a known direction but unknown magnitude of the change. Our problem is a little bit different in that we are considering an M-ary sequential hypothesis testing problem and that we will not allow changes with negative or zero values for ν, i.e. we impose the restriction $\nu \geq 0$.

Thus the resulting change detection problem is:

$$\theta^d(k) = \begin{cases} \theta_0^d \quad \text{when } k < t_{change} \\ \theta_0^d + \nu\Upsilon_a \quad \text{or} \\ \theta_0^d + \nu\Upsilon_b \quad \text{or} \\ \theta_0^d + \nu\Upsilon_c \text{ when } k \geq t_{change} \end{cases} \tag{3.4}$$

where t_{change} is an unknown time step when the change occurs.

Since we have an unknown parameter ν we follow the GLR approach from the introduction. We run in parallel a GLR for each possible direction Υ_m versus the null hypothesis θ_0^d :

$$g_k^{d,m} = \max_{1 \leq j \leq k} \log \frac{\sup_{\nu^{d,m} \geq 0} \prod_{i=1}^k f_{\theta_0^d + \nu^{d,m}\Upsilon_m}(X_k^d)}{\prod_{i=j}^k f_{\theta_0^d}(X_k^d)} \tag{3.5}$$

Only the test $g_k^{d,m}$ that reaches its given threshold is stopped. The threshold $h^{d,m}$ for each of the parallel tests is selected given a fixed false alarm rate probability.

Equation (3.5) has a nice closed form solution when the distributions are assumed to be Gaussian $\mathfrak{N}(\theta_0^d, \Sigma_d)$. In this case we get the constrained optimization problem:

$$\min_{\nu \geq 0} \sum_{i=j}^{k} \frac{1}{2} \left(X_i^d - \theta_0^d - \nu\Upsilon_m\right)^T \Sigma_d^{-1} \left(X_i^d - \theta_0^d - \nu\Upsilon_m\right) \tag{3.6}$$

If the restriction is inactive then we obtain

$$\hat{v}_k^{d,m}(j) = \frac{\Upsilon_m^T \Sigma_d^{-1} \left(\frac{1}{k-j+1} \sum_{i=j}^k X_i - \theta_0^d\right)}{\Upsilon_m^T \Sigma_d^{-1} \Upsilon_m} \tag{3.7}$$

If the restriction is active then $\hat{v}_k^{d,m} = 0$.

The intuitive idea of this approach is that you are projecting the difference between all available sample means and the mean θ_0^d (or historic mean estimate for practical purposes) into each of the possible directions Υ_m, and selecting the time step of this projection that maximizes the likelihood of the alternate hypothesis assuming $\theta_0^d + \hat{v}_k^{d,m}(j)\Upsilon_m$, i.e. when we think the sample mean started moving in the Υ_m direction. The uncorrelated covariance (in our case) is just a weighting parameter for being cautious about declaring a change in the mean too soon if the process is observed to have large fluctuations around the mean.

We tested the robustness of this approach even when the distribution was not Gaussian, as long as we are computing the mean and covariance in a window of time, much bigger than our false alarm delay, to keep mean and covariance up to date and only detect abrupt changes. Recall that in the worm detection chapter we computed the mean only once.

Following most change detection approaches applied to network traffic [11, 12], we also use the nonparametric CUSUM test for each link independently: $S_k^{d,m} = \max\{0, S_k^{d,m} - X_k^{d,m} - N_k^{d,m} - c_k^{d,m}\}$, where $c_k^{d,m}$ is a positive deterministic sequence chosen experimentally to minimize the average detection delay and $N_k^{d,m}$ is the historical estimate of the mean $E_0[X_k^{d,m}]$.

3.3.2 Sensor Fusion

So far we have been focusing on detecting a change in a single node. One of the main advantages in having several nodes under monitoring is that we can perform an aggregation of the statistics between the different nodes in order to decrease our detection delay given a fixed false alarm rate probability. In particular if we are monitoring nodes far away from the destination, most of the local statistics will not yield an alarm and the attack might be unnoticed.

The alarm aggregation can be performed by several methods. Here we propose a simple algorithm that will only require the knowledge of the routing tables for the nodes being monitored.

We want a mechanism to aggregate the different statistics at each monitored node. Clearly the aggregation mechanism cannot be multiplicative, because if we are monitoring a node physically unable to detect the attack (a node that is not in the routing path of any of the attackers and the victim) the low value of the computed statistic of this node will adversely affect any small information that any other node might have in relation to the attack. On the other hand the computed statistics for all nodes can vary to different scales of magnitude yielding a biased addition.

To cope with this latter problem we compute the normalized statistic $\varphi_k^{d,m} := \frac{g_k^{d,m}}{h^{d,m}}$. If none of our monitored nodes has raised an alarm, the number of monitored nodes will be bounded by $\sum_d \varphi_k^{d,m}$. This can be in turned interpreted as a new upper bound for a collective threshold which can be selected given a false alarm rate probability. Note the similarity of this approach with the fusion detection from the chapter on worms.

Selecting which statistics to correlate (add) is the key issue. In keeping with our directionality framework we will correlate only the statistics relating two or more nodes to a common node. This is the reason we need the routing table information of our monitored nodes.

The algorithm is as follows:

```
Given two nodes d and e;
For each link $d \rightarrow m$
    For each link $e \rightarrow n$
        If there is a node f reachable through $d \rightarrow m$ and $e \rightarrow n$
            then correlate their normalized statistic;
```

In the following section the application of the aggregation will be described, and we will show also the relation to another suboptimal M-ary sequential detection problem where M is the number of all possible link combinations of the two (or more) routers that reach the same destination. We will apply this formulation to the case of two nodes, but it extends recursively when we are monitoring three or more nodes.

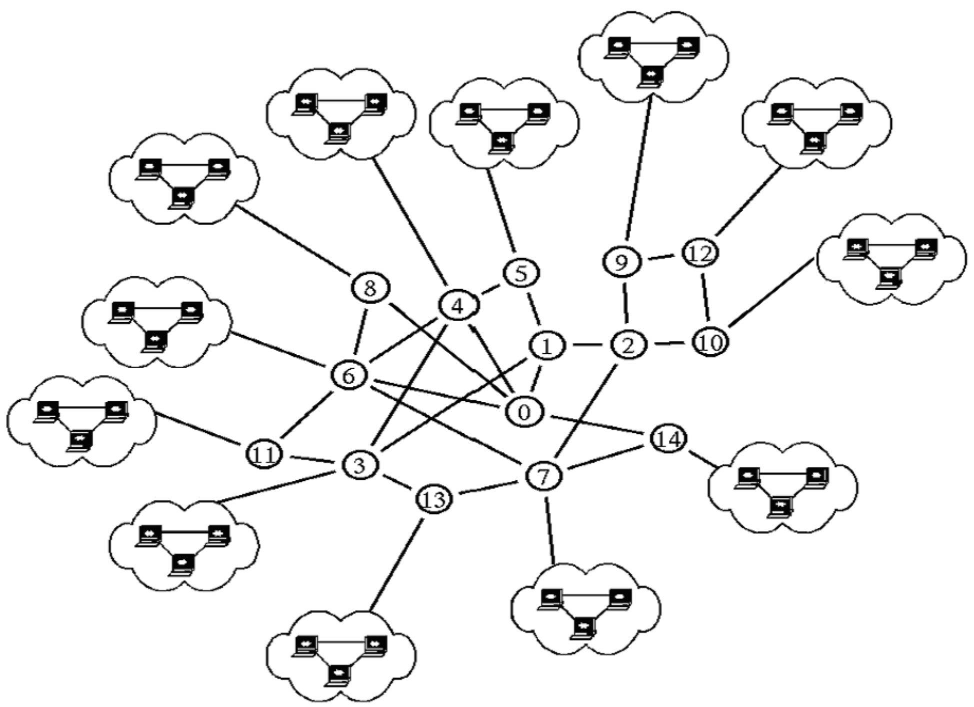

Figure 3.3: The transit network

3.4 Simulation Results and evaluation

For our experimental results we used the network simulation software ns2. We created a transit (scale-free) network topology given in figure 3.3 where the transit network consists of 15 routers numbered from 0 to 14. Each cloud represents a subnetwork whose internal architecture we are not interested in. It consists of 15 transit nodes performing only routing between each subnetwork. There are 12 subnetworks with 65 hosts each.

During the normal operation of the network each of these 780 hosts selects randomly a host in another sub network, and establishes an On-Off source connection with Pareto distributed times. The routing protocol selects a route with the least number of hops toward a given destination. The attack is simulated with a given number of compromised nodes in different subnetworks. During the attack, each of these nodes will start a constant bit rate connection toward a specific node. The rate of the attackers was varied to test the detection algorithms with different percentage of attack packets circulating over the transit network at a given time. We considered 7 attackers. One in each of the subnetworks connected to nodes 3, 4, 5, 8, 9, 11 and 13. The victim is in the network connected to node 14.

Some typical link usage characteristics can be seen in figure 3.4, where node 6 uses node 0 to reach 14. An attack occupying 2.5 percent of the transit network traffic begins at k=350.

We first tested the performance of the statistics $S_k^{d,m}$ and $g_k^{d,m}$ at individual nodes. With the network under normal operation we experimentally obtained the thresholds for each statistic for a given false alarm rate of 0.003.

We selected the nodes 0, 1, 2 and 7 to test the detection delay of the statistics independently of each other. The results can be seen in figure 3.5, showing the average detection delay per node when we monitor individually nodes 0,1,2 and 7. A detection delay of 1 is the same as the average delay for a false alarm. The average delay of detection is computed between the four nodes for different percentages of the amount of traffic the attack generates over the transit network. Node 0 had the smallest detection delay in all cases, as it is a node where most

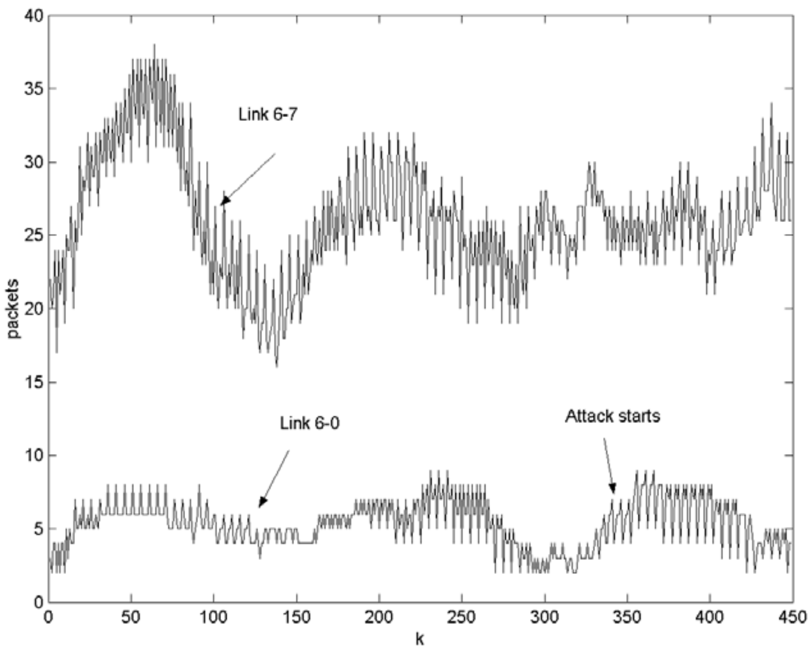

Figure 3.4: Link usage

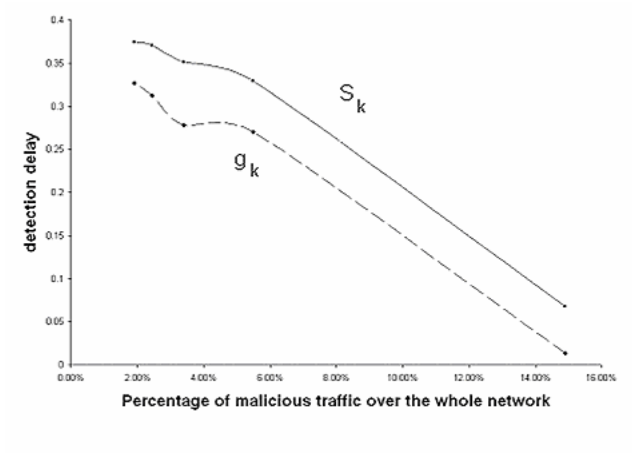

Figure 3.5: Average detection delay

of the traffic toward node 14 gets agglomerated, giving thus a large change in the mean. Node 2 performed poorly because it only routes toward node 14 the attacks packets generated by the attacker in the subnetwork of node 9 and thus sees very little traffic. $g_k^{d,m}$ performs marginally better than $S_k^{d,m}$.

The aggregation mechanism can be applied to decrease the detection delay when we are monitoring more than one node. In previous results we mentioned the poor performance of nodes routing very few packets toward the destination. Lets consider the case of two nodes far (in the number of hops) from the victim. We will pick nodes 6 and 3 for the aggregation. Node 6 routes only the malicious traffic from the subnetwork attached to node 11 and node 3 only route malicious traffic from its subnetwork. Furthermore if we consider an attack reaching less than 2% of the transit network traffic it will be very difficult to detect any abnormal change without a global integrated view of the statistics at different nodes.

For testing our aggregation algorithm we will start at k=1 an attack reaching 1.5% of all traffic at the transit network. Again the attack is toward a host in the subnetwork of node 14.

The normalized statistics are computed for each link in the nodes. The solid dark curve in figure 3.6 is the normalized statistic for the link (3,1), the only link that the attack uses. The dotted curves are the normalized statistics for the links (3,13), (3,4), (3,11) and the link to its subnetwork. Figure 3.7 shows the normalized statistics for node 6. The solid dark curve is the normalized statistic for the link (6,0). The circles identify the statistic from node 6 to its subnetwork.

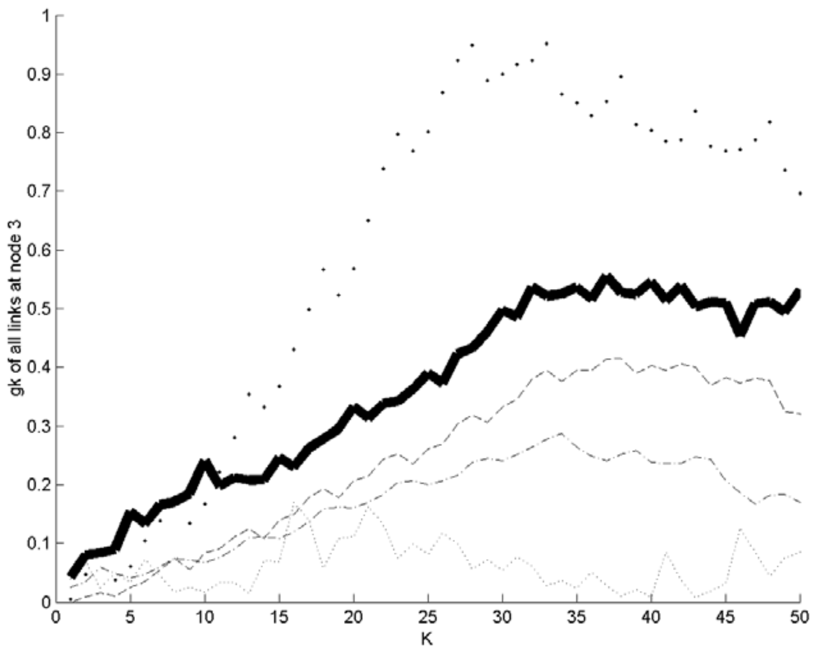

Figure 3.6: All normalized statistics for node 3.

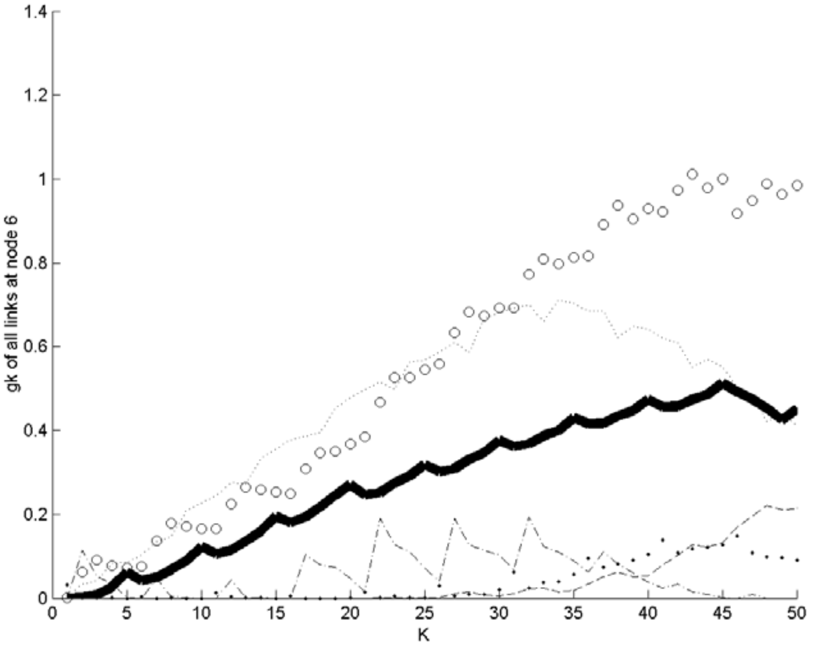

Figure 3.7: All normalized statistics for node 6.

This statistic raises a local false alarm at k=42. In both cases we can see that the statistic closest to reaching the threshold, is not the one by the links being used in the attack.

The routing tables required for the aggregation algorithm are given in Tables 1 and 2.

Link	Routing to nodes
(6,7)	7,13,2,10,12,9
(6,0)	0,14,1
(6,4)	4,3,5
(6,11)	11,3
(6,8)	8
(6,subnetwork)	

Table 3.1: Routing table for node 6

By simple inspection of the routing tables we see that we need to correlate the link (6,0) with (3,1) because nodes 6 and 3 use them (respectively) to reach nodes 0, 1 and 14. Similarly, the link (6,11) must be correlated with (3,11), link (6,4) with (3,4), link (6,7) with (3,13), (6,7) with (3,1) and (6,8) with (3,11).

If we denote as H_i the hypothesis when node i or its subnetwork are under attack, then we have the following hypothesis testing problem created by the aggregation mechanism

Link	Routing to nodes
(3,1)	1,0,2,14,10,9,12
(3,13)	7,13
(3,4)	4,5
(3,11)	11,6,8
(3,subnetwork)	

Table 3.2: Routing table for node 3

$$\begin{aligned}
1_{(H_0\vee\ H_1\vee H_{14})} &= \varphi^{6,0}+\varphi^{3,1} > h_{0\vee 1\vee 14}\\
1_{(H_{11})} &= \varphi^{6,11}+\varphi^{3,11} > h_{11}\\
1_{(H_4\vee H_5\ )} &= \varphi^{6,4}+\varphi^{3,4} > h_{4\vee 5}\\
1_{(H_{13}\vee H_7)} &= \varphi^{6,7}+\varphi^{3,13} > h_{13\vee 7}\\
1_{(H_2\vee H_{10}\vee H_{12}\vee H_9)} &= \varphi^{6,7}+\varphi^{3,1} > h_{2\vee 10\vee 12\vee 9}\\
1_{(H_8)} &= \varphi^{6,8}+\varphi^{3,11} > h_8
\end{aligned}$$

where $1_{(\,)}$ is the indicator function of the Hypotheses. If we have fixed routes in the network, the thresholds $h_{i\vee\ldots\vee j}$ can be computed to reach a given false alarm rate.

The results of our aggregation algorithm between all allowable normalized statistics are shown in figure 3.8. The dark solid curve is the correlated statistic of the links (6,0) and (3,1). The other dotted curves represent the correlated

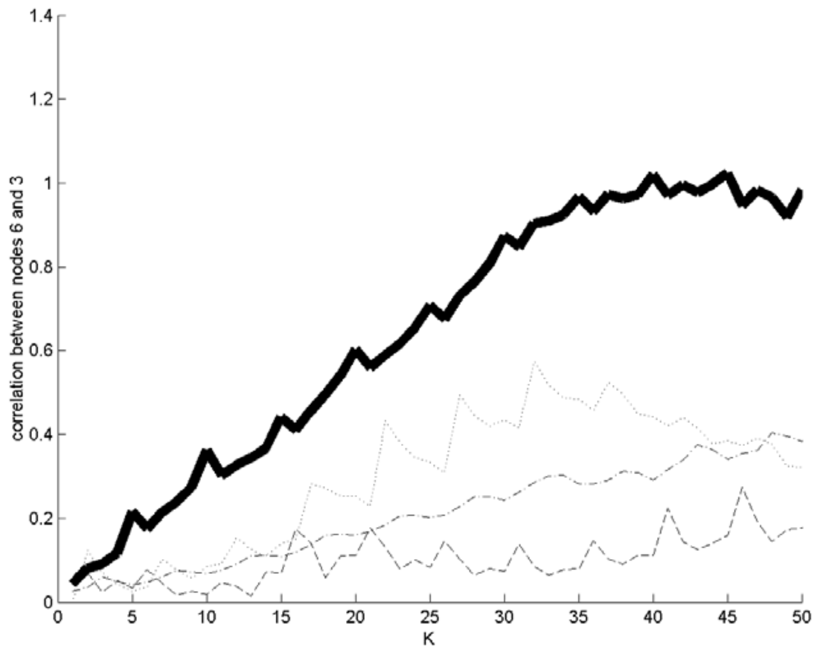

Figure 3.8: Correlated statistics.

statistics of the remaining allowable links. It is clear that the aggregation of the normalized statistics at nodes 6 and 3 gives a better resolution of the attack than the statistics of 6 and 3 alone while discriminating the uncorrelated random fluctuations of the traffic intensity that cause most of the false alarms.

Not only can we achieve the detection of the attack (depending on the new aggregation threshold), but also we can diminish the impact of the false alarm originating at node 6. Another important conclusion is that without the need to extract or store header information from the packets transmitted through the network, we are able to infer (from the intersection of the two routing tables for the highest correlated statistic of the links (6,0) and (3,1)) the only three possible targets: Namely the nodes 0, 1 and 14, i.e. hypothesis $H_0 \vee H_1 \vee H_{14}$.

A simple marginal constant reduction in the delay detection such as that obtained with $g_k^{d,m}$ vs. $S_k^{d,m}$ can provide significant help when we correlate the statistics because constant gains will get multiplied by the number of nodes participating in the detection reducing exponentially the detection delay.

3.5 Conclusions and further work

Note that one main difference between $S_k^{d,m}$and $g_k^{d,m}$ is that for testing a change in the link m, $S_k^{d,m}$does not use information from the other links of node d whereas $g_k^{d,m}$ does. A possible use of the information contained in all the links from d is to compute a statistic that measures the changes in the normalized traffic seen through a link (d, m):

$$\rho_k^{d,m} := \frac{X_k^{d,m}}{\sum_{m \in \mathcal{N}(d)} X_k^{d,m}}$$

This approach has the added advantage that a positive change in the mean of $\rho_k^{d,m}$ tends to yield a negative change in the mean on any other neighbor n of d as $\rho_k^{d,m}$and $\rho_k^{d,n}$ for $n \neq m$ are negatively correlated because

$$\sum_{m \in \mathcal{N}(d)} \rho_k^{d,m} = 1$$

. Experimental validation shows that the process $\rho_k^{d,m}$ has less variations than its normalized counterpart and will be more amenable to a mean computation and the usage of the CUSUM nonparametric statistic.

A way to complement the use of the direction concept in the formulation of the statistic is by testing each possible direction against a change in all directions (Υ_d)) for detecting when the overall traffic of the network has increased or decreased, i.e. computing

$$g_k^{d,m} = \max_{1 \le j \le k} \log \frac{\sup_{v \ge c_1} \prod_{i=j}^{k} f_{\theta_0^d + v\Upsilon_m}(X_k^d)}{\sup_{\lambda} \prod_{i=j}^{k} f_{\theta_0^d + \lambda\Upsilon_d}(X_k^d)}$$

where λ is a scalar not necessarily greater than a positive constant c_1 unlike ν.

Optimal solutions to the problem of sequential testing of more than two hypotheses are, in general, intractable. Hence research in sequential multi-hypotheses testing has been directed toward the study of practical, suboptimal sequential tests and the evaluation of their asymptotic performance. Simple generalizations of the Wald's binary SPRT, such as a parallel implementation of a set of simple SPRT's, may be far from optimum in multi-hypotheses problems [15].

In [15] a proof of asymptotic optimality as the decision risks (or error probabilities) go to zero for two nontrivial extensions of the SPRT are presented. The tests are also asymptotically optimal to any positive moment of the stopping time distribution, and the results are extended to continuous-time statistical models that may include correlated (non-i.i.d.) and non-homogeneous observation processes. This generalization justifies the use of these tests in a variety of applications.

Our M-ary change detection procedure used in this chapter is thus suboptimal. We could improve results by following the optimal detection procedures.

Chapter 4

Detection of routing attacks in MANETS

4.1 Attacks to the routing protocols

Mobile -wireless- ad hoc networks (MANETS) are particularly vulnerable to attacks to the routing protocols. Unlike fixed networks, the routers usually do not reside in physically protected places and can fall under the control of an attacker more easily. Such an attacker can then send incorrect routing information. Furthermore messages can be eavesdropped and faked messages can be injected into the network without the need to compromise nodes.

General attacks are misrouting, false message propagation, packet dropping, packet generation with faked source address, corruption on packet contents and denial-of-service [16, 17]. In [18] they consider up to 40% misbehaving nodes which agree to forward packets but fail to do so. Solutions are based on identifying these nodes and avoiding them. Detection is made by listening promiscuously to the next node's transmissions. If the next node does not forward the packet, then it is misbehaving.

One of the attacks exploiting the wireless medium is the wormhole attack. In this problem an attacker records packets or bits at one location in the network and tunnels them to another location by means of a single long-range directional wireless link or through a direct wired link to a colluding attacker. At the end of the tunnel certain packets are retransmitted into the network [19]. The wormhole attack can be devastating to a routing protocol. For example in DSDV if each routing advertising sent by node A were tunneled to node B (and any data packet is not retransmitted) and vice versa, then A and B would believe that they are neighbors. If they were not within wireless transmission range, they would be unable to communicate. Furthermore, if the best existing route from A to B were at least $2n+2$ hops long, then any node within n hops of A would be unable to communicate with B and vice versa.

4.1.1 Related work

To our knowledge there is only one publication on intrusion detection for mobile ad hoc networks [20]. Besides describing the overall architecture of the IDS, their approach consists in training two classification algorithms for the simulation of attacks falsifying routing information and dropping packets. The classification algorithms used were support vector machines (SVMs) and RIPPER (a rule induction algorithm). Features such as the percentage of changes in a routing table in a given amount of time, position of the nodes obtained from GPS devices, and

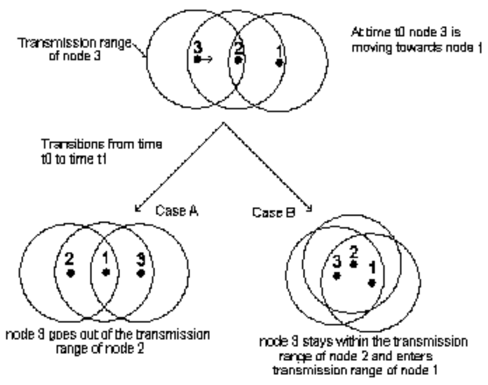
Transmission range of node 3
At time t0 node 3 is moving towards node 1
Transitions from time t0 to time t1
Case A
Case B
node 3 goes out of the transmission range of node 2
node 3 stays within the transmission range of node 2 and enters transmission range of node 1

Figure 4.1: Allowed state transitions in an ad hoc network

percentage of traffic change were considered. There is no deductive modeling of the intrusions; although rules are induced by RIPPER but their performance is not as good as the SVM decisions, which do not provide any insight on the classification of the attacks.

4.2 Problem formulation

Our approach is mostly an academic exercise for building a model capturing the dynamics of a highly mobile ad hoc network. The basic idea is that an attacker will change the routing information in such a way that our perceived mobility of the nodes will differ from our previous experience.

We want to learn the allowable state transitions (which depend in our sampling interval.) In figure 4.1 we have a simple scenario in which node 3 moves toward node 1 and we have 2 possible transitions. It can be the case that based on the

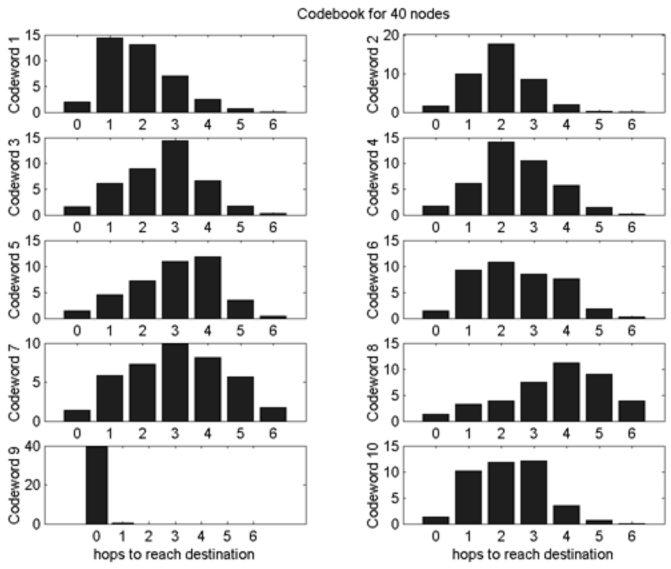

Figure 4.2: Codebook of size 10 of the hop count distribution

mobility pattern for the network, only transition to state B is allowed (e.g. node 3 can not move fast relative to 1.) For establishing a baseline of the allowable state transitions we build a discrete hidden Markov model (HMM) with parameters (π, A, B) for modeling the temporal relations of various statistics encoding the underlying mobility of the nodes. The states of the HMM can be viewed as abstractions of different spatial configurations of the mobile nodes. We will pick the observation variable to be the hop count distribution at a given node (figure 4.2.) For simplicity we will assume a proactive distance vector routing protocol such as DSDV in order to have all hop counts at any time.

If we have N nodes (actually $N+1$ but each node see only N other nodes), the hop count distribution seen at any node at the time step k can be considered as a vector in $\{0, ..., N\}^D$: $X_k = [X_k^0, ..., X_k^{D-1}]'$ ($X_k^i \in \{0, ..., N\}$ where D is a limit we impose in the maximum number of hops we will consider, i.e.

$$\begin{aligned}
X_k^0 &= \text{number of nodes 0 hops away (disconnected and itself)} \\
X_k^1 &= \text{number of nodes 1 hop away} \\
&\vdots \\
X_k^{D-2} &= \text{number of nodes } D-2 \text{ hops away} \\
X_k^{D-1} &= \text{number of nodes } D-1 \text{ or more hops away} \qquad (4.1)
\end{aligned}$$

In order to consider a discrete HMM we need a way to deal with the high-dimensional observation vectors X_k. The number of all possible observations is $(N+1)^{D-1}$ as we note that we are working with a hyperplane in $(N+1)^D$ because $\sum_{i=0}^{D-1} X_k^i = N+1$. The most natural approach is to quantize (compress) X_k to a set of M codewords. The codebook Ξ for vector quantization can be learned by using the classic LBG algorithm, in which for a given fixed rate R, one tries to find the codebook that minimizes the given distortion function. We will consider a quadratic distortion. Figure (4.2) shows a typical codebook of size 10 obtained for the hop count distribution under the mobility model discussed in the simulations section.

In order to continue in the change detection setup we follow a CUSUM procedure applicable to the case of dependent observations x_j with distributions f_{θ_1}

and f_{θ_0} under hypotheses H_1 and H_0 respectively[21]:

$$S_n = \left\{ S_{n-1} + \log\left(\frac{f_{\theta_1}(x_n|x_{n-1},...,x_k)}{f_{\theta_0}(x_n|x_{n-1},...,x_k)}\right)\right\}^+ \tag{4.2}$$

where x_k is the first sample after the last reset, i.e., $S_{k-1} = 0$. It is clear that this algorithm is only a reformulation of the SPRT algorithm for the log-likelihood ratio: $\log\left(\frac{f_{\theta_1}(x_n|x_{n-1},...,x_k)}{f_{\theta_0}(x_n|x_{n-1},...,x_k)}\right)$ with the lower threshold selected at 0. The upper threshold h will be selected given a false alarm rate.

In the field of intrusion detection most statistic models are built for anomaly detection. In the case of observations in a finite alphabet (the codebook in our case) the alternate hypothesis can be considered as the uniform distribution, i.e. $\forall \hat{x} \in \Xi\,(P(\hat{x}) = 1/M)$. This is a way of not assuming anything about the attack and therefore it is particularly suited for detecting the attacks we do not know of. So under the assumption that we have an HMM model $(\pi_{\theta_0}, A_{\theta_0}, B_{\theta_0})$ for the normal operation of the network, equation (4.2) can be reduced to

$$S_n = (S_{n-1} + \log(1/M) - \log f_{\theta_0}(\hat{x}_k, ..., \hat{x}_n) + \log f_{\theta_0}(\hat{x}_k, ..., \hat{x}_{n-1}))^+ \tag{4.3}$$

where $\forall k, \hat{x}_k \in \Xi$.

4.3 Simulation results

The simulations take place in a 1000x1000m^2 region with 40 nodes moving at speeds of $5m/s$ and with a communication range of $250m$. For the mobility of the nodes we used the standard random waypoint algorithm in which the nodes select

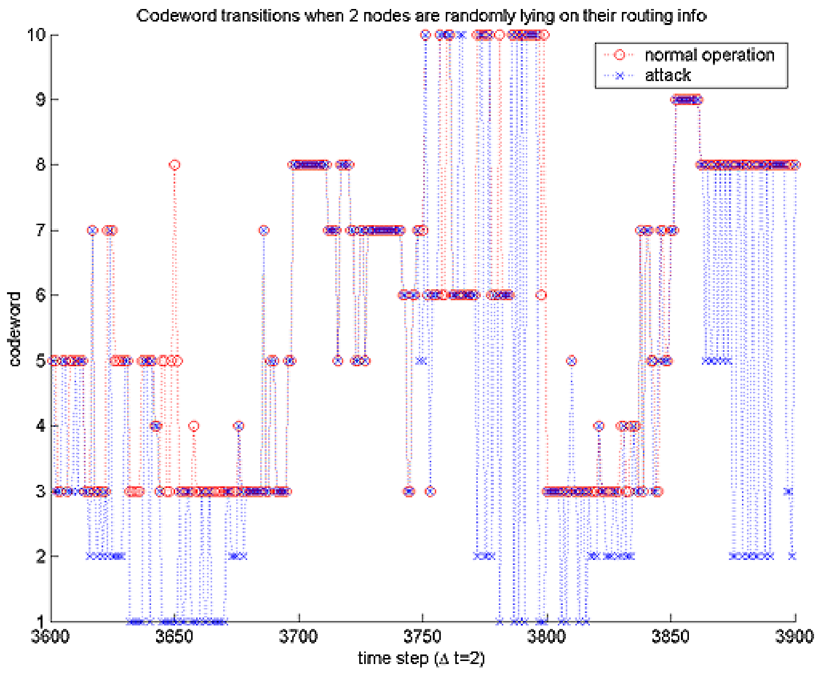

Figure 4.3: Codeword transitions with two nodes sending bad routing information at random

with a uniform distribution a destination point from the region and after reaching their target they remain there for a given amount of time ($5s$ in our simulation) before selecting again another destination. We will be assuming that we monitor one node in which the hop count distribution is sampled every two seconds. A total simulated time of four hours was selected. The HMM is trained with the first 3600 samples. In the following 1800 samples the nodes continue to behave normally and in the final 1800 samples the attack takes place.

A second mobility model considered (avw mobility) is one in a 52x42m^2 U-shaped building where there are 47 offices and 47 nodes. Each node spends most of the time in its home office and travels with a speed of $1m/s$ among different offices selected at random. Each node has a communication range of $11m$. A total simulated time of one hour was selected and the time step between measurements was 2 seconds.

The prior is initialized as a uniform distribution. The number of states is selected to be equal to the number of observations and both matrices A_{θ_0} and B_{θ_0} are initialized with higher values in their respective diagonal component than the rest of the values.

The first attack we simulated was an artificial attack to take advantage of the HMM. Under this scenario two nodes start sending bad routing information randomly at any given time about their distance to any other node. The spurious pattern of the attack caused by the random attack produces more codewords transitions as seen in figure (4.3). As seen in figure (4.4), the attack was easily detected

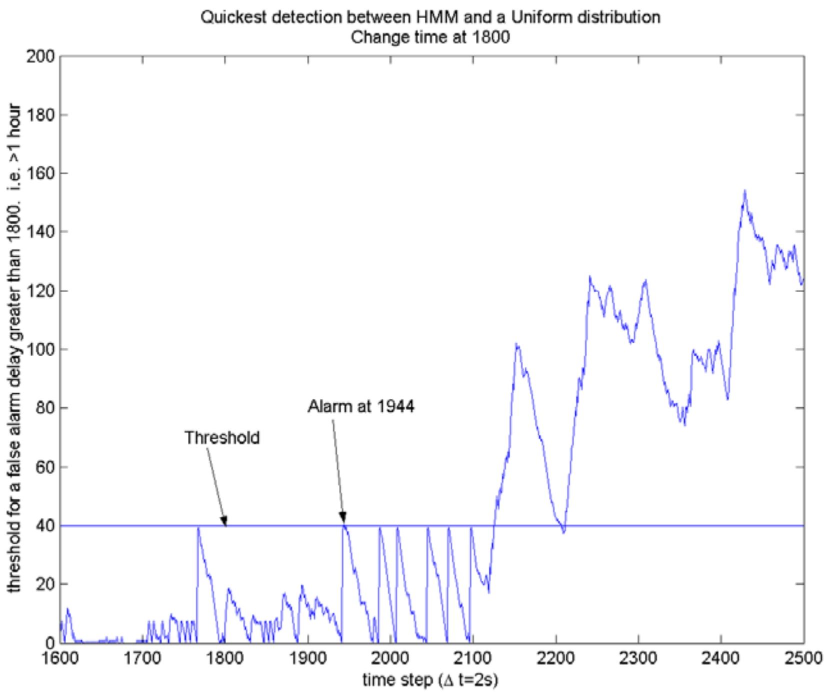

Figure 4.4: Change detection statistic for attack 1

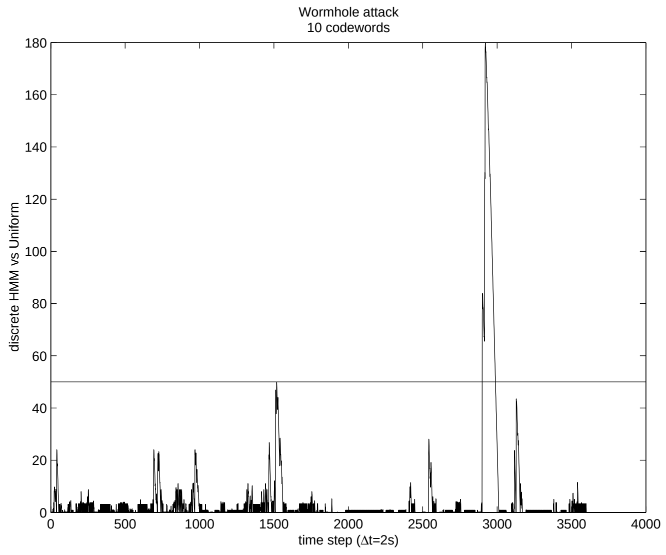

Figure 4.5: Change detection statistic for a wormhole attack

using equation (4.4).

The second scenario considers a wormhole attack, where the endpoints of the wormhole are located at x, y coordinates $(1000/3, 500)m$ and $(2000/3, 500)m$ for the $1000x1000m^2$ plane. This attack was also detected using equation (4.4). However, we can see in figure 4.5 that the statistic is not robust to the change.

A way to improve the performance is by including information on the attack. We trained another HMM $(\pi_{\theta_1}, A_{\theta_1}, B_{\theta_1})$ for the attack mode. The resulting performance of this new change detection statistic between two HMMs can be seen in figure 4.6.

The performance of the change detection statistics to the wormhole attack is similar to an attack in which one node is sending bad routing information constantly. However in this case, a batch detection procedure by testing the likelihood

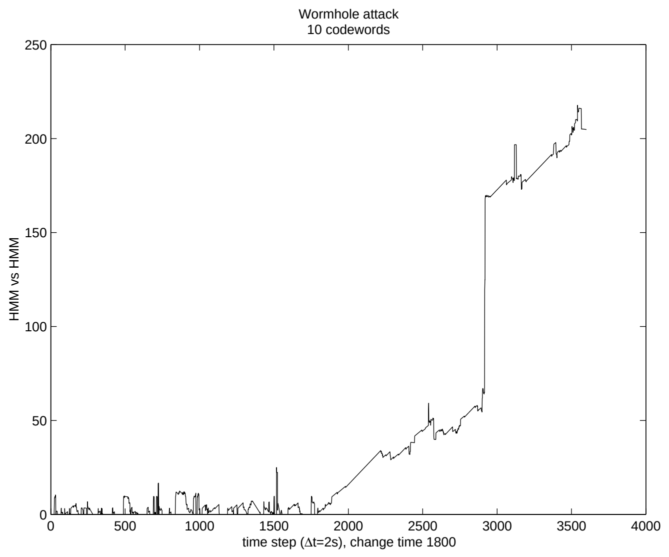

Figure 4.6: Change detection statistic when we have an HMM model of the attack

of the state transitions for a given window of time seems to perform better when we do not have an HMM of the attack. In figure 4.7 we can see that the state transitions of a normal sequence have in average a higher log-likelihood value than the states transitions from the attack. Figure 4.8 is the ROC curve.

Another way to deal with the problem is to build a more complex HMM. A natural generalization to avoid the dependence of the detection on the trained codebook is to consider directly the continuous hop count distribution vector $X_k \in \mathbb{R}^D$. To deal with this continuous vector we trained an HMM with a mixture of two Gaussians with two hidden states. With continuous observations we also need a fixed interval in $\mathbb{R}^D$ for defining the uniform probability distribution. However we decided to estimate the mean of the likelihood $f_{\theta_0}^{GaussianHMM}(x_n|x_{n-1},...,x_k)$ for any new time step x_n, to replace the distribution $\log 1/M$. The resulting

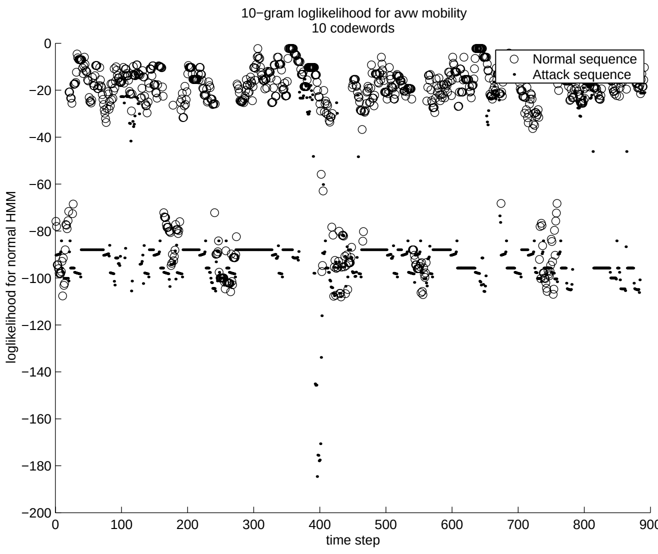

Figure 4.7: Log-likelihood of state transitions

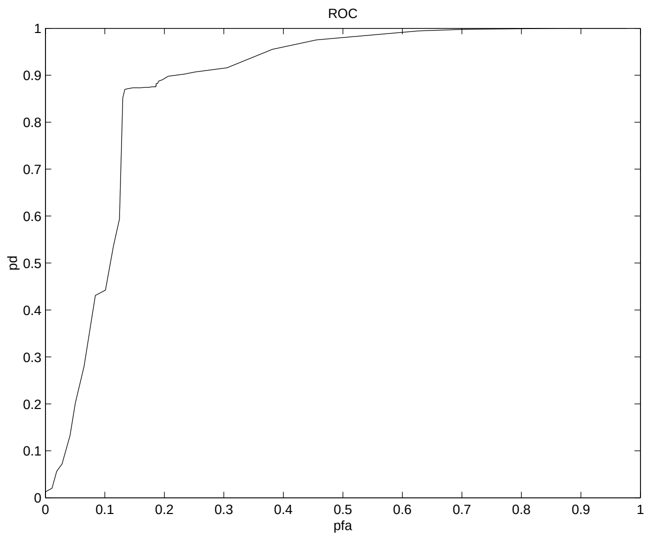

Figure 4.8: ROC curve for batch detection

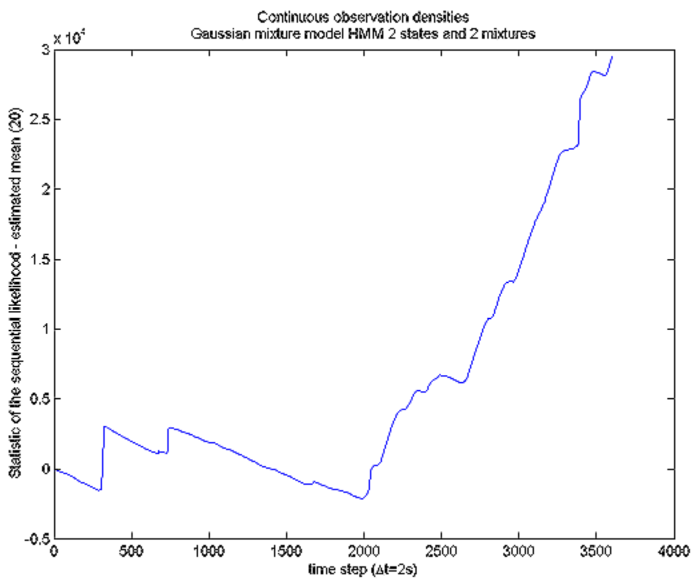

Figure 4.9: Change detection statistic using a Gaussian mixture output HMM

performance of the statistic is shown in figure (4.9).

4.4 Conclusions and further work

Although the attacks introduced can be detected by very different and easy means, the principle of detecting an unknown attack to the routing protocol with different characteristics was obtained. In particular, attack 1 produces a change in the variance of the hop count distribution, and attack 2 produces a change in the mean of the hop count distribution. Both attacks were detected by simply testing on the likelihood of our learned model.

The drawbacks of any anomaly detection approach are that we always need to assume that we can train our model in an attack free environment. Issues we have to study further is the amount of time required for training our normal model, and the dependence on the mobility pattern.

Another of the main limitations is that it flags as an alarm any deviation from normal conditions and determining a normal baseline for ad hoc networks might be very difficult in practice.

Chapter 5

Conclusions

In this thesis we have presented new approaches to detect intrusions using a statistical framework.

The detection of self-propagating code was discovered to depend on the network connectivity. For traffic satisfying the parametric assumptions the statistics "fusion" "aggregate cusum" and "exponential change in mean" perform better than the others. When the traffic has a long range dependence, the nonparametric statistics "regression" and "robust regression" perform better. However this depends highly on the burst spikes of the normal traffic and their performance will degrade as the traffic becomes more bursty.

In the detection of denial of service attacks we introduced a new directionality framework to improve the detection in transit networks from statistics using only a change in one link. In real networks it is very likely that the sensors will be placed at the peering routers because these routers will be close to the target. So to apply our framework for detecting the transit routers we can use a change detection statistic with a prior in the change time. The change time will be assumed to be the time when the alarm is set from one of the peering routers.

For ad hoc networks we presented a new formulation for representing the mobility of the nodes. In practice the mobility might have very large variations and therefore a model robust to these normal changes must be studied.

BIBLIOGRAPHY

[1] Internet Security Systems Inc. Internet risk impact summary for june 25, 2002 - september 27, 2002.

[2] M. Basseville and I.V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[3] George V. Moustakides. Quickest detection of abrupt changes for a class of random process. *IEEE Transactions on Information Theory*, 44(5), September 1998.

[4] B.E. Brodsky and B.S. Darkhovsky. *Nonparametric Methods in Change-Point Problems*. Kluwer Academic Publishers, 1993.

[5] R. Albert and A.L. Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, pages 47–97, January 2002.

[6] Ed Skoudis. Infosec's worst nightmares. In *Information Security Magazine*, pages 38–49. TruSecure Corp., November 2002.

[7] Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to 0wn the internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium (Security '02)*, 2002.

[8] Alexander G. Tartakovsky and Venugopal V. Veeravalli. Change-point detection in multichannel and distributed systems with applications. In N. Mukhopadhyay, editor, *Applications of Sequential Methodologies*. Marcel Dekker Inc, New York, 2002.

[9] K. J. Houle and G. M. Weaver. Trends in denial of service attack technology. Technical report, CERT, October 2001. v1.0.

[10] David Moore, Geoffrey M. Voelker, and Stefan Savage. Inferring internet denial-of-service activity. In *Proceedings of the 2001 USENIX Security Symposium*, 2001.

[11] R. B. Blazek, H. Kim, Boris Rozovskii, and A. Tartakovsky. A novel approach to detection of denial-of-service attacks via adaptive sequential and batch-sequential change-point detection methods. *IEEE Systems, Man and Cybernetics Information Assurance Workshop*, June 2001.

[12] Haining Wang, Danlu Zhang, and Kang G. Shin. Detecting syn flooding attacks. In *Proceedings of INFOCOM 2002*, New York City, New York, June 2002.

[13] John Ioannidis and Steven M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *Proceedings of Network and Distributed System Security Symposium, Catamaran Resort Hotel San Diego, California 6-8 February 2002*, 1775 Wiehle Ave., Suite 102, Reston, VA 20190, February 2002. The Internet Society.

[14] S. Bohacek. Optimal filtering for denail of service mitigation. In *IEEE Conference on Decision and Control*, 2002.

[15] Vladimir P. Dragalin, Alexander G. Tartakovsky, and Venugopal V. Veeravalli. Multihypothesis sequential probability ratio tests-part i:asymptotic optimality. *IEEE Transactions on Infromation Theory*, 45(7), November 1999.

[16] Yih-Chun Hu, David B. Johnson, and Adrian Perrig. Sead: Secure efficient distance vector routing in mobile wireless ad hoc networks. In *Fourth IEEE Workshop on Mobile Computing Systems and Applications* (WMCSA '02), pages 3–13, 2002.

[17] S. Buchegger and J. Le Boudec. Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks.

[18] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 255–265, 2000.

[19] Y.-C. Hu, A. Perring, and D. B. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. Technical Report TR01-384, Rice University, December 2001. Revised September 2002.

[20] Y. Zhang, W. Lee, and Y. Huang. Intrusion detection techniques for mobile wireless networks. In *ACM/Kluwer Mobile Netowrks and Applications (MONET)*, 2002.

[21] Biao Chen and Peter Willet. Detection of hidden markov model transient signals. *IEEE Transactions on Aerospace and Electronics Systems*, 36(4):1253–1268, October 2000.