MULTI-LEVEL NETWORK RESILIENCE: TRAFFIC ANALYSIS, ANOMALY DETECTION AND SIMULATION

Angelos Marnerides[1], Cyriac James[2], Alberto Schaeffer-Filho[3], Saad Yunus Sait[4], Andreas Mauthe[5] and Hema Murthy[6]

[1, 3, 5]*School of Computing and Communications, Lancaster University, United Kingdom*
E-mail: [1]a.marnerides@lancaster.ac.uk, [3]asf@comp.lancs.ac.uk, [5]andreas@comp.lancs.ac.uk
[2,4,6]*Department of Computer Science and Engineering, Indian Institute of Technology Madras, India*
E-mail: [2]cyriac@lantana.tenet.res.in, [4]saad@lantana.tenet.res.in, [6]hema@lantana.tenet.res.in

Abstract
Traffic analysis and anomaly detection have been extensively used to characterize network utilization as well as to identify abnormal network traffic such as malicious attacks. However, so far, techniques for traffic analysis and anomaly detection have been carried out independently, relying on mechanisms and algorithms either in edge or in core networks alone. In this paper we propose the notion of multi-level network resilience, in order to provide a more robust traffic analysis and anomaly detection architecture, combining mechanisms and algorithms operating in a coordinated fashion both in the edge and in the core networks. This work is motivated by the potential complementarities between the research being developed at IIT Madras and Lancaster University. In this paper we describe the current work being developed at IIT Madras and Lancaster on traffic analysis and anomaly detection, and outline the principles of a multi-level resilience architecture.

Keywords:
Traffic Analysis, Core and Edge Networks, Network Resilience, Anomaly Detection

1. INTRODUCTION

Traffic analysis and anomaly detection are extensively used to understand and characterize network traffic behaviour, as well as to identify abnormal operational conditions such as malicious attacks. However, techniques for traffic analysis and anomaly detection are typically carried out independently in different parts of the network, either in the edge or in the core networks alone. In fact, different traffic characteristics and anomalies can normally be better observed in a specific part of the network, although they affect the network as a whole. In this paper, we advocate that a comprehensive architecture for network resilience must combine traffic analysis and anomaly detection mechanisms and algorithms deployed both in the core and edge networks, thus introducing the notion of *multi-level network resilience*.

This work is motivated by the potential complementarities between the research being developed at IIT Madras and Lancaster University. On the one hand, researchers at IIT Madras have concentrated on traffic analysis and anomaly detection at the edge of the network, monitoring and measuring Internet traffic at web server proxies. In addition to prediction of network attacks based on SYN flooding, overall traffic and individual user's traffic behaviour is also monitored, in order to classify, predict and control user bandwidth using ARIMA timeseries models of network usage. On the other hand, Lancaster University has focused on the classification of attack-related traffic in core networks, in terms of volume-based detection of anomalies. The applicability of energy Time-Frequency distributions alongside the use of Renyi information has been utilized for distinguishing malicious traffic from the rest of application-layer protocols as seen on the transport layer of a core network.

Both Lancaster and IIT Madras are looking at similar machine-learning algorithms for anomaly detection, traffic shaping and prediction. IIT Madras' approach is interesting since there are computational resources available at the edge of the network to perform efficiently algorithms for anomaly detection. However, considering anomaly detection only at the end node is limiting since it only observes the effects on particular end-systems at the edge of the network but does not consider the effects on the network itself. Particularly with multiple, coordinated attacks the network might be the actual target of the attacker rather than any specific services. Therefore, detection should be done as early as possible, alongside the network, to increase the evidence and to detect anomalies in a more coordinated manner. This is in line with recent trends in network-wide and backbone anomaly detection research.

Hence the proposed joint work on multi-level resilience is timely and current. There is a good potential for collaboration in investigating the effects, results and benefits of both approaches combined. This papers presents an initial investigation on an integrated anomaly detection infrastructure, combining metrics and mechanisms from both edge and core networks.

This paper is organised as follows: Section 2 describes our independent work in traffic analysis and anomaly detection, both in the core and in the edge networks, presenting the mechanisms and the algorithms that we intend to integrate in order to realise the notion of multi-level network resilience. Section 3 outlines the simulation platform that will permit the evaluation of multi-level network resilience strategies. Section 4 briefly describes a case study scenario that is used to illustrate the ideas presented in this paper. Finally, Section 5 presents our concluding remarks.

2. CORE NETWORKS AND EDGE NETWORKS OBSERVATIONS

2.1 TRAFFIC ANALYSIS IN CORE NETWORKS

Network backbone link traffic exhibits dynamic characteristics which are classified as highly non-stationary due to their non-constant instantaneous frequency and group delay on the Time-Frequency (TF) plane [7]. Under the intention of characterizing these dynamics we found as most appropriate to exploit the capabilities provided by energy distributions and we

have particularly employed formulations belonging in the Cohen class of distributions [8]. In particular, we assessed the task of application-layer classification with the utilization of meaningful, discriminative metrics extracted from the Choi-Williams distribution [9]. These metrics were used as the numerical features for our tree-based classification as being provided within the treefit() and classregtree() MATLAB utility functions.

Our analysis was based upon the independent evaluation and observation of the transport protocol's (i.e. TCP, UDP) volume-wise behaviour (i.e. counts of packets and bytes per unidirectional flow) with respect to their utilization from the application protocols/clients. The operational network trace used was an hour long sample captured at the Samplepoint-B backbone link of the WIDE network and provided by the Mawi working group [10].

2.1.1. Cohen-Based Energy Distribution and the Choi-Williams Formulation:

One of the main problems occurring in TF energy representations is that many functions developed under the "energy" concept generating such distributions include complex and negative values. Even though authors in [8] support the incentive of employing a TF energy distribution function by having a signal under its analytical form they argue that the resulting distributions cannot be considered as proper probabilities for real valued signals. Due to the aforementioned drawback, Cohen in [11] complementing the work by Wigner formulated an initial generic method to generate a range of functions which can satisfy the marginal properties which enabled the acquisition of positive distributions.

If we let our signal be *s(t)* and its analytical form be *s(u)*, the representation of the Cohen class of distributions is formulated as:

$$M(t,\omega)=\frac{1}{4\pi^2}\int\int\int_{-\infty}^{+\infty} e^{-j\theta t-j\tau\omega+j\theta u}\varphi(\theta,\tau)\cdot s^{*}(u-\frac{1}{2}\tau)s(u+\frac{1}{2}\tau)dud\tau d\theta \quad , \tag{1}$$

where $\varphi(\theta,\tau)$, is a two-dimensional kernel function determining the specific distribution on a given signal and should satisfy:

$$\varphi(\theta,0)=\varphi(\tau,0)=1 \tag{2}$$

Under their intention of attenuating cross and self terms initiated by a signal on the TF plane Choi and Williams in devised a Cohen-based distribution which intuitively (and in many cases in practice) satisfies up to a high level of accuracy the description of a signal's energy concentration on the TF plane. The initial approach by Choi and Williams was to consider a signal *s(t)* constructed by a number of components:

$$s(t)=\sum_{a=1}^{N} s_a(t) \tag{3}$$

By substituting the above definition in the general Cohen class of distributions equation (i.e. formula 1) we obtain a resulting distribution which is composed by the sum of self and cross terms as naturally evolved by frequency scaling on the signal itself alongside the cross-scaling triggered by the individual signal components.

$$M(t,\omega)=\sum_{a=1}^{N} M_{aa}(t,\omega)+\sum_{\substack{a,i=1\\ i\neq a}}^{N} M_{ai}(t,\omega) \tag{4}$$

The sum of cross terms $M_{ai}(t,\omega)$ always under the general Cohen equation (formula 1) is represented as:

$$M_{ai}(t,\omega)=\frac{1}{4\pi^2}\iiint e^{-j\theta t-j\tau\omega+j\theta u}\varphi(\theta,\tau)\cdot s_a*(u-\frac{1}{2}\tau)s_i(u+\frac{1}{2}\tau)dud\tau d\theta \tag{5}$$

In parallel with their intention of exploring distributions under the notions of self and cross terms, Choi and Williams realized that interference terms would be greatly reduced with a choice of a particular kernel function for $\varphi(\theta,\tau)$. This kernel function would minimize the cross terms and at the same time retain the desirable properties of the self terms. By investigating the local autocorrelation function as well as using the ambiguity concept their finalized kernel method was derived as:

$$\varphi(\theta,\tau)=e^{-\theta^2\tau^2/\varepsilon}, \tag{6}$$

where ε is a constant and is the core variable for controlling the suppression of the signal's cross terms and frequency resolution. Using formulas 1, 4, 5, 6 and after per-θ integration the resulting Choi-Williams distribution is:

$$CW(t,\omega)=\frac{1}{4\pi^{3/2}}\iint\frac{1}{\sqrt{\tau^2/\varepsilon}}e^{-\left[(u-\tau)^2/(4\tau^2/\varepsilon)\right]-j\tau\omega}s*\left(u-\frac{1}{2}\tau\right)\cdot s\left(u+\frac{1}{2}\tau\right)dud\tau \tag{7}$$

The Choi-Williams (CW) distribution was the mean for mapping the measurements obtained for the packet/byte time series as seen from each transport protocol. In parallel they constituted towards the generation of the Renyi information which it was used as the basic discriminative feature for classifying our application-signals.

2.1.2. Renyi Information:

According to generic information-theoretical principles, the extraction and measurement of information on the TF plane can be easily achieved with the utilization of the (always positive) spectrogram and the well known Shannon entropy. Nevertheless, in virtue of the resolution trade-off [12] and bias of the spectrogram it is quite disadvantageous to employ it on any of the Cohen-based TF distributions. In parallel, due to the negative values present in some of these distributions it is also quite inefficient to apply the Shannon's entropy for gaining valuable information with respect with the complexity of a given signal on the TF plane.

However, information-theoretic and signal processing literature [14], suggest the 3rd order Renyi entropy (in some cases also called as dimension) as one of the most accurate methods for interpreting a signal's complexity. The notion of complexity is mapped as the number of elementary components composing the main signal and is measured in bits. In our elementary independent components in our signal are corresponding to large deviations from the average packet/byte count. Therefore, a large positive-valued Renyi information value in bits shows that our signal has high complexity and it is composed by a number of components which significantly differ numerically from the average byte/pkts count on the TF plane.

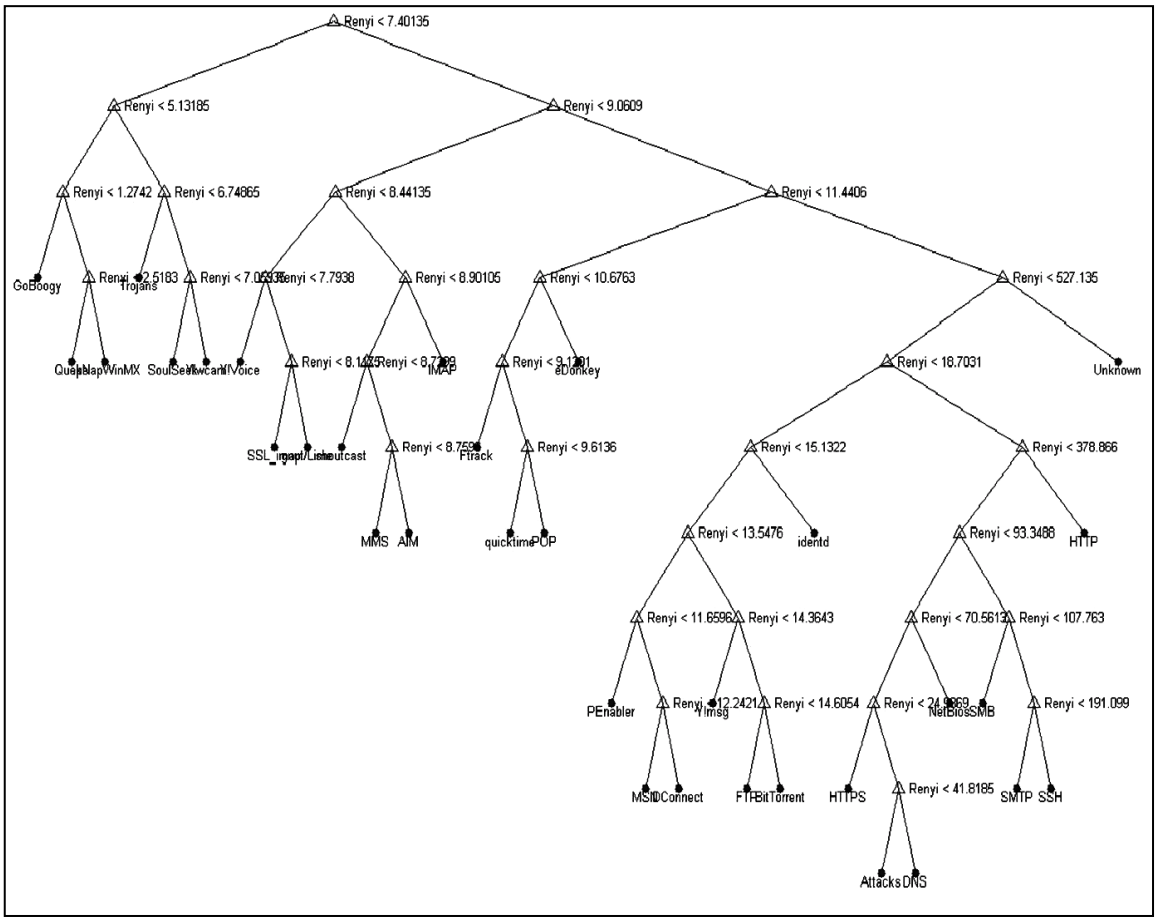


Fig.1. Tree-based application classification under the CW distribution for WIDE on TCP-packets

The generalized Renyi information in contrast with the Shannon entropy information is a formulation that accepts negative values within the under analysis energy TF distribution and is defined as:

$$R_M^a = \frac{1}{1-a}\log_2\left\{\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} M^a(t,\omega)dtd\omega\right\} \quad a \geq 0,\ a \neq 1. \tag{8}$$

As shown in formula 8, the generalized Renyi information is dependent upon the tuning variable *a* that determines the order of the Renyi entropy. In the case of 1st order Renyi entropy, we recover the Shannon entropy and the Kullback-Leibler divergence and as mentioned earlier it is inefficient to use it particularly on any Cohen-based TF distribution. Furthermore, the case of $a = 2$ is ruled out since according to [1]a.[11] the term $\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} M^2(t,\omega)dtd\omega = 1$ resulting to $R_M^2 = 0$ for any type of signal. Thus, for the specific case of Cohen-based TF distributions it suggested for $a \geq 3$ and by employing it on formula 8 we obtain:

$$R_M^3 = -\frac{1}{2}\log_2\left\{\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} M^3(t,\omega)dtd\omega\right\}. \tag{9}$$

Formula 9 enabled our experimentation to determine distinct complexity measurements for particular Internet applications and alongside the TF moments we were able to construct a feature set for each of the observational datasets and use them within our tree-based categorization. The evaluation presented within this document describes mainly the packet-based application-classification.

2.1.3. Classifying Applications on WIDE Based Upon the TCP Packet-Based Utilization:

The hour long trace from WIDE was broken in to 4 equal bins (i.e. WIDE-I, II, III and IV) of 13.75 minutes each.

Using the behavioral and port-based extraction scheme provided by the work done in BLINC [1]a.[12] we initially extracted the counts of packets associated with TCP and then associated each application with its packet-based utilization. A subsequent process was to compute the CW distribution for each distinct application and further estimate the Renyi information provided by each. Specifically we used the sample of WIDE-I as a training set and the remaining samples as the testing sets in our classification.

As Fig.1 illustrates our simple tree-based classification provided ranks for every application present within the WIDE trace and assigned complexity ranks to each.

Apparently, from a bytes and packets perspective, the most complex signal which is structured by intensive flows is the traffic triggered by unknown applications (i.e. R<527.135). Even though this particular traffic category didn't exhibit the largest amount of unidirectional flows it contained unknown applications that were frequently observed throughout the whole

WIDE trace and most of them consumed a considerable portion of TCP from a packets perspective. Following the lead of unknown traffic and second in rank was as anticipated the HTTP protocol. As being the dependency for the most commonly used applications (e.g. WWW), HTTP had volume-wise the biggest amount of flows but its transmissions on average were not largely intensive from a packets and bytes perspective. We argue that one of the main reasons of being ranked as highly complex in this case is due to its enormously big sample size in comparison with the rest of the extracted application protocols. Apart from that, there were obviously transmissions associated with certain applications (e.g. Apache HTTP servers) that demonstrated extremely high TCP utilization from a packets perspective. However, we can generally conclude that HTTP-based applications throughout the WIDE trace exhibited individually a higher number of packets per flow rather than being flows with fewer packets determined by big byte sizes.

SMB resides as well in the high valued Renyi branches (i.e. R<107.763) indicating its frequent appearance in the WIDE trace overall. Its placement in this level of complexity indicates that SMB-related applications are volume-wise consuming in both TCP packets and bytes. In addition, as similarly observed in the Keio networks, this high SMB utilization is supportive evidence in the argument that the majority of users in the WIDE network were running MS-Windows. Leaving aside SMB, we also noticed high packet-wise utilization from SMTP-based mail as well as with SSH. Based upon the byte-based rankings, SMTP was expected to be categorized as highly complex and intensive but on the other it was intriguing the classification achieved for SSH. Via this packet-based categorization we could get an insight related with SSH-related transmissions and summarize that SSH unidirectional flows are much more intensive from a packets perspective. Subsequent intensive application protocols in the branches where R<70.5613 were expected protocols involved with network operations (i.e. NetBIOS), DNS, attack-related traffic particularly on ports 135 and 1025, as well as secured HTTP connections (i.e. HTTPS).

Possessing a noteworthy amount of TCP packet-wise traffic was a TCP utility protocol that of Ident (R<15.1322) which was followed by intensive chat applications such as Yahoo! Messenger and MSN. Furthermore, file transfers through FTP alongside P2P distribution platforms like Bittorrent, PeerEnabler and DirectConnect were noticed to use a significant amount of TCP packets in each of their flows. Going to lower branches we see rankings of medium complexities (i.e. R<11.4406) to be assigned in a variety of applications dealing with P2P (eDonkey, FastTrack), live streaming with QuickTime and POP-based mail. Relatively having slightly less volume and complexity-wise significance were applications in the even lower branch (i.e. R<9.0609) indicating their occasional utilization such as IMAP-based mail, chat through AOL's Messenger (i.e. AIM), encrypted IMAP `mail, P2P traffic via Gnutella and Limewire as well as internet live streaming through Shoutcast and MMS. Of lesser importance with respect to their Renyi estimates (i.e. R<7.40135) were traditionally UDP-based applications such as P2P distribution platforms such as OpenNapster, GoBoogy and SoulSeek and Yahoo! Messenger's video utilities. Similarly with the byte-based analysis, Trojans initiated in various TCP destination ports indicated an insignificant traffic impact with low packet complexity. In fact, the majority of identified Trojans and malicious flows carrying worms were associated with well known vulnerable TCP ports (e.g. 16, 27, 30, 68). Specifically the Trojans identified were those of Skunk, Backboor.Trojan and Assassin [1]a.[14].

2.2 TRAFFIC ANALYSIS IN EDGE NETWORKS

2.2.1. Detection of DoS Attacks:

Time series models are used for modeling and prediction of network traffic. Most techniques assume stationarity and predictability of the given series. In this paper, an attempt is made to analyse network traffic at an edge router in the context of TCP SYN based DoS (Denial of Service) attack, using linear time series models. The network feature studied in this paper is called the *Half-Open Count* discussed in Section 5.1. Earlier work [24] on this feature modeled it using Auto-Regressive (AR) technique, for detection of TCP SYN based DoS (Denial of Service) attack at the victim server. But, some of the drawbacks of this approach are: stationarity and predictability of the time series data are not ensured, but rather assumed, as in the case of other related work [4] – [6]. Further, model coefficients are recomputed for every observation window of size 6 samples (with sampling interval of 10s), which slides over the time series one sample at a time. The recomputed coefficients are then used for prediction of the attack. This perhaps takes care of the non-stationarity in the data, but re-computation of coefficients for every window of such small size may not be a good approach in terms of practical implementation.

Internet being an extremely dynamic and constantly evolving system, it is unlikely to find any feature that will be invariant across all networks at all times. Hence the solution to traffic modeling lies in performing suitable transformations on the *half-open time series*. In particular, two transformations are discussed in this paper, namely differencing and averaging. The effect of these on the original series are studied and compared. Further, from the analysis, predictability of the time series is found to have strong correlation with Hurst exponent, Auto-correlation and Smoothness of the series.

Data Set Used

Network trace was collected over a period of three months from July 15th 2010 to September 30th 2010, from the TENET gateway of IIT Madras, using TCPDUMP tool [25]. The gateway connects TENET network to the Internet. The bandwidth of the link connecting the gateway to the service provider is currently 4Mbps. From this trace, the difference in number of incoming (traffic entering from outside network) SYN packets and number of outgoing (traffic leaving to outside network) SYN-ACK packets, called half-open count, are extracted at equally spaced time intervals of 10s. For the analysis, three data sets are created, each consisting of 5 days (i.e 24 hrs x 5 days) data, from Monday to Friday. These data sets are checked for any existing TCP SYN DoS attack, by verifying whether all the connection requests are valid. This is done to demonstrate the effectiveness of our approach in the context of TCP SYN based DoS attack detection, discussed in Section 5.7. Since the detection algorithm works on the number of half-open connections, it will scale for different network sizes.

Transformations of Time Series

Three different time series of half-open feature are compared in this paper: (a) original, (b) difference and (c) average. To generate the difference series, absolute value of the difference for consecutive samples is computed. This is also called first differencing [26], [27]. A moving average function of the original time series using a sliding window of size 30 samples, with a one-sample shift, is computed, to form the average time series.

Fig.1 (a) – (c) show the original, difference and average time series for the data set-1 for Monday and Thursday respectively. Similar trend is observed for other days. Couple of observations are worth mentioning: First, the short time variations (spikes) can be seen in the original and difference series, but are absent in the average series. Such frequent spikes hints that any threshold based techniques, like the ones discussed in [28], [29], to detect DoS attacks, may not work for these two cases. This is because it is highly unlikely that a model can be built which can predict frequent spikes accurately. Second, long term variations or trend present in the original series are removed in the difference series but retained in the average series.

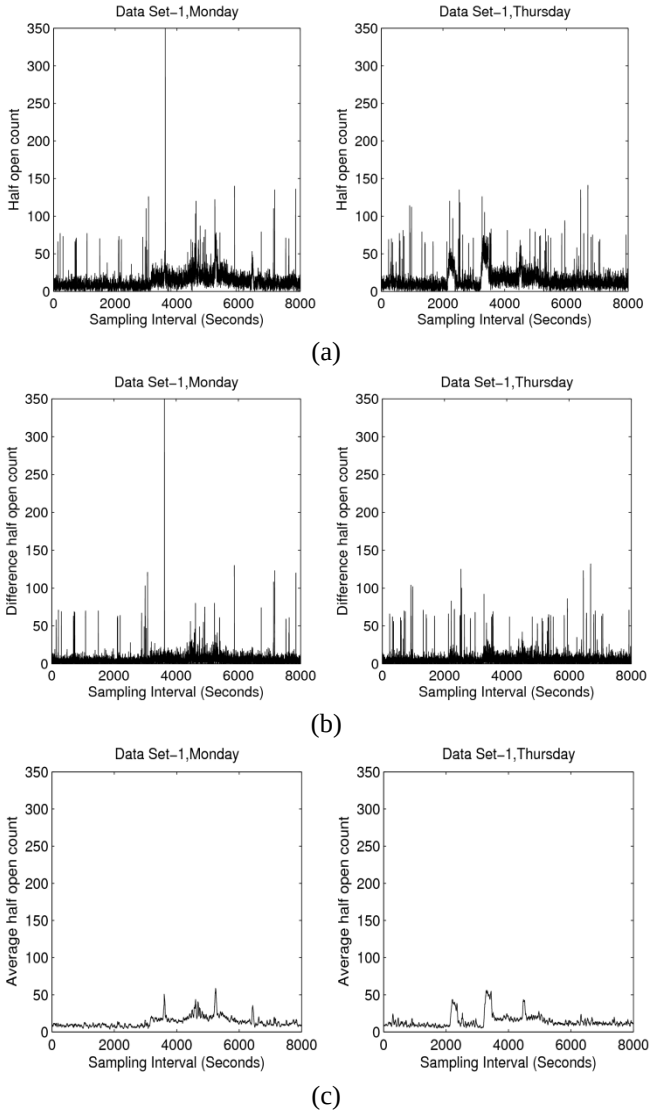


Fig.2. (a) Original (b) Difference (c) Average time series for data set 1

Stationarity Test

A process is said to be wide-sense stationary if the mean and autocorrelation of the process are invariant over time. This is also called weak stationarity [26], [27]. The intuitive idea of checking stationarity is to ensure whether the model parameters estimated are time invariant. Table.1, Table.2 and Table.3 show the mean for the original, difference and average series respectively. It can be observed that the means vary across three data sets, for all the three time series. But the means within each data set, for all the three series, does not show much variation.

Table.1. Mean of Original Series

Day	**I**	**II**	**III**
Mon	13.41	7.89	8.16
Tue	11.43	8.15	6.7
Wed	14.09	8.41	4.99
Thur	14.37	8.44	4.71
Fri	13.39	8.26	6.01
Avg.	13.34	8.23	6.11

Table.2. Mean of Difference Series

Day	**I**	**II**	**III**
Mon	5.54	5.04	4.84
Tue	5.01	5.09	4.21
Wed	5.64	5.18	3.78
Thur	5.74	5.04	3.67
Fri	5.37	5.27	3.99
Avg.	5.46	5.12	4.1

Table.3. Mean of Average Series

Day	**I**	**II**	**III**
Mon	13.43	7.89	8.16
Tue	11.43	8.15	6.71
Wed	14.1	8.41	4.99
Thur	14.37	8.44	4.71
Fri	13.39	8.26	6.01
Avg.	13.34	8.23	6.11

Fig.3.(a)-(c) show ACF graphs for data set-2, plotted for original, difference and average series respectively, which doesn't show much variation across different days for each series. Similar observations are seen for the other two data sets also. This indicates that, though all three series are non-stationary across data sets, it appears that all are quasi-stationary with window size of 5 days or perhaps a week.

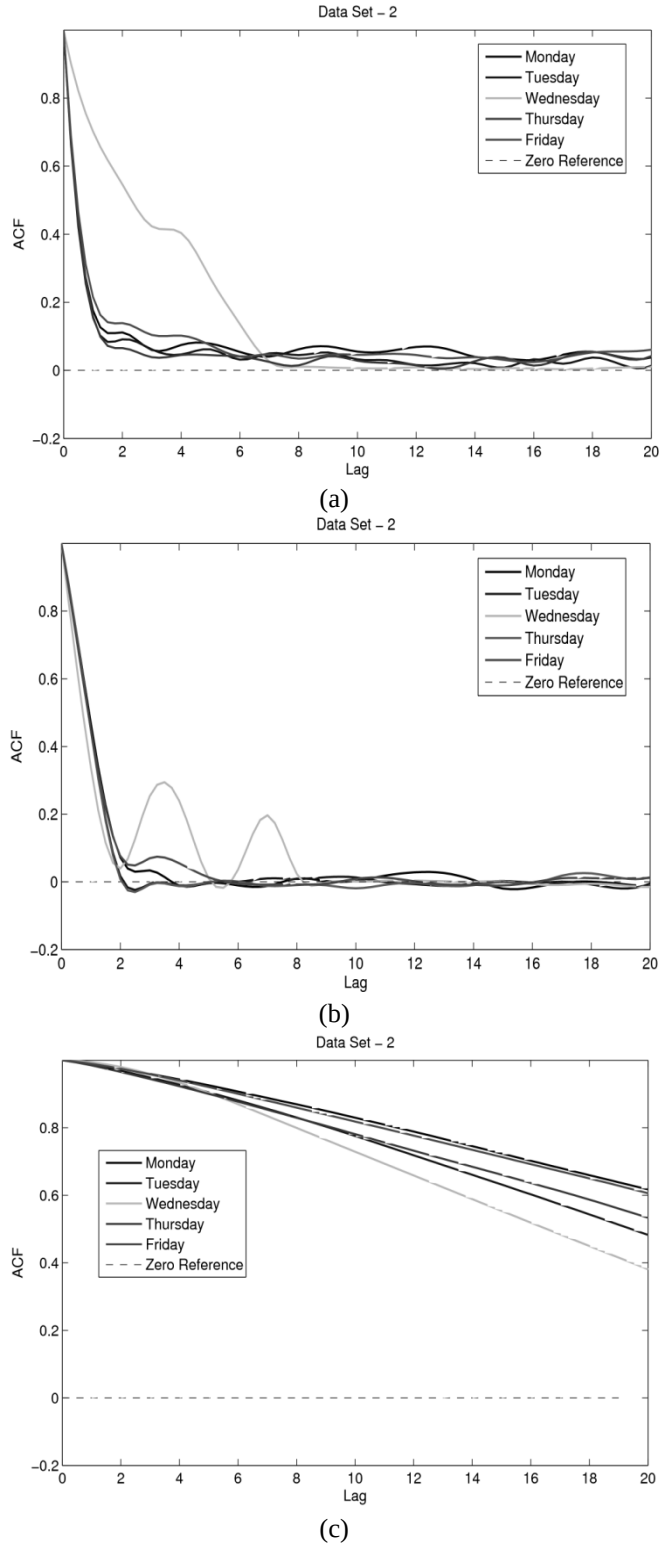


Fig.3. ACF for (a) Original (b) Difference (c) Average series for data set 2

Smoothness Factor

Matthew Roughan et al in their work on modeling backbone traffic [30] have quantified the smoothness of the time series in terms of relative variance (variance divided by the mean). If the relative variance is low, series is said to be smooth or less bursty. Table.4 shows the smoothness computed for all the three transformations. Average series is found to be smooth compared to the other two series.

Table.4. Smoothness

Data Set	Original Series	Difference Series	Average Series
I	8.41	12.26	3.55
II	14.72	19.38	2.15
III	10.61	17.14	2.42

Hurst Exponent Estimation

Hurst exponent (H) [31], [32] is a measure of the burstiness of the time series. Time series can be classified based on the H value as (1) $H = 0.5$, for white Gaussian noise, (2) $0 < H < 0.5$, is a mean reverting and less bursty series and (3) $0.5 < H < 1$, is a bursty and trend reinforcing series. None of the Hurst estimators give correct value and often give conflicting results as indicated in [31], [32], [33]. For the analysis described in this paper, a software package called SELFIS [34] is used for estimating Hurst exponent. Out of the several estimators that were tried out, apart from the rescaled range estimator (r/s method), other estimators show conflicting results and sometimes Hurst values outside the range, 0 to 1. Hence, Hurst estimation based on r/s method is used along with smoothening factor discussed in Section 5.4, to arrive at a meaningful conclusion. Details of r/s estimation are discussed in [33]. Table.5 shows the Hurst exponent values for the original, difference and average series. It can be observed that the Hurst values for the original and difference series are higher than the average series. This shows that the average series is less bursty and smoother than the other two series. Also, there is no significant difference between Hurst exponent of original and difference series.

Table.5. Hurst Values

Data Set	Original Series	Difference Series	Average Series
I	0.71	0.62	0.44
II	0.59	0.56	0.42
III	0.62	0.57	0.42

Modeling and Prediction

In order to study the predictability of the series, each data set is considered stationary since the mean and auto- correlation are relatively constant within each data set as discussed in Section 5.3. Auto-Regressive model discussed in [26], [27], is used to build the model using traffic on 1st day and predict the 5th day's traffic. It is observed experimentally that model order of 2 is a good value for prediction. Note that, for each transformation, three models need to be built, one for each data set. Estimation of AR model coefficients are done by Yule-Walker method discussed in [26], [27]. Average relative prediction error is shown in Table.6. Relative error is computed by normalising prediction error with the actual value for every sample value predicted.

Fig.4 shows the actual and predicted time series for all the three transformations of data set-1. It shows larger prediction error for the original and difference series compared to the average series. Similar behaviour is observed for other two data

sets as well. It can be concluded from Table.6 and Fig.4 that the average series, which has retained the trend and removed the short-term variations or spikes from the original series, is more predictable than the other two. Also, it may be noted that the relative error for the difference series is more than the original series, which means that differencing has made the original series less predictable.

Table.6. Average Relative Error

Data Set	Original Series	Difference Series	Average Series
I	0.46	0.90	0.01
II	0.78	0.85	0.02
III	0.77	0.87	0.03

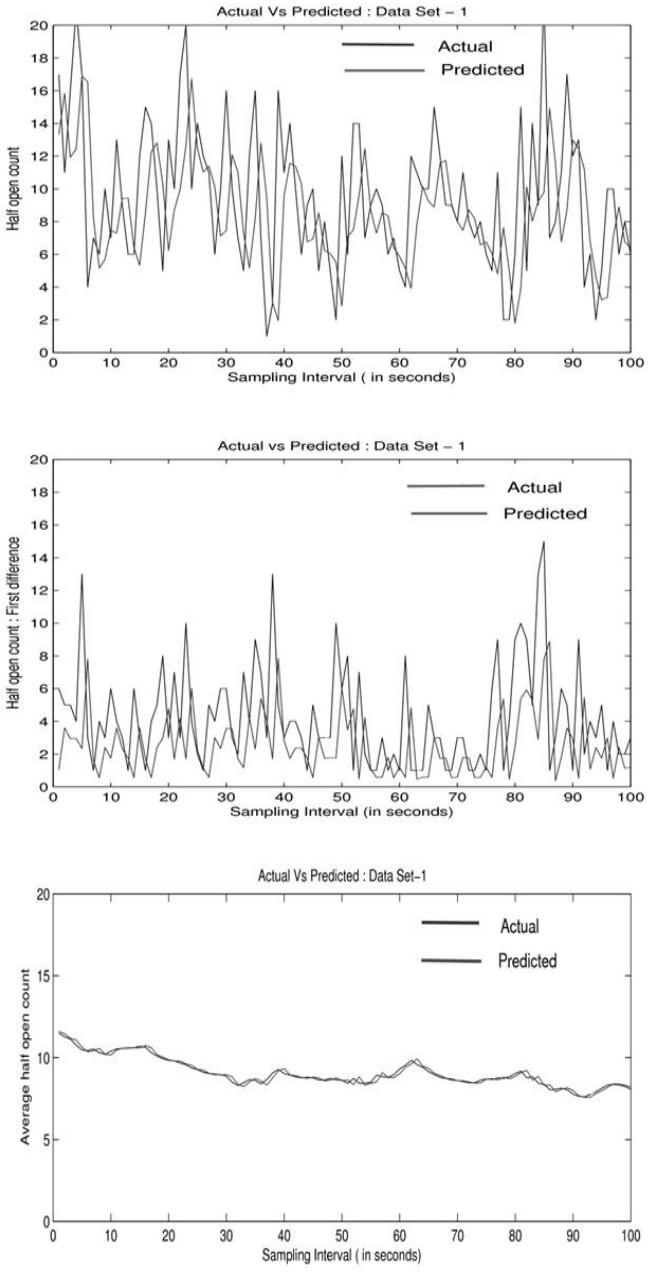


Fig.4. Actual vs Predicted for Original, Difference and Average Series for data set 1

Detection of Attack

A trace driven low rate TCP SYN based DoS attack is simulated by generating SYN packets with spoofed unreachable IP addresses to a victim server. Rate of the attack is varied uniformly from 10 to 20 SYN/second. Duration of the attack is 5 minutes. Traffic trace at the victim server is collected and is mixed with normal traffic of 5th day of each data set to generate the attack traffic. The experiment is repeated for 10 times. Model is built on the average time series and tested to predict the attack traffic. The relatively large prediction error during an attack can be utilised for detecting the attack by fixing a threshold on the prediction error. Fig.5 shows the probability of wrongly detecting an attack called False Positive (FP) and the probability of not detecting an attack called False Negative (FN) for various threshold values. It is found that for a threshold value as 0.4, there is 0% FN and 3% FP. The worst-case detection delay is 5 minutes taking into account the window size of 30 samples for building the average series and sampling interval of 10s.

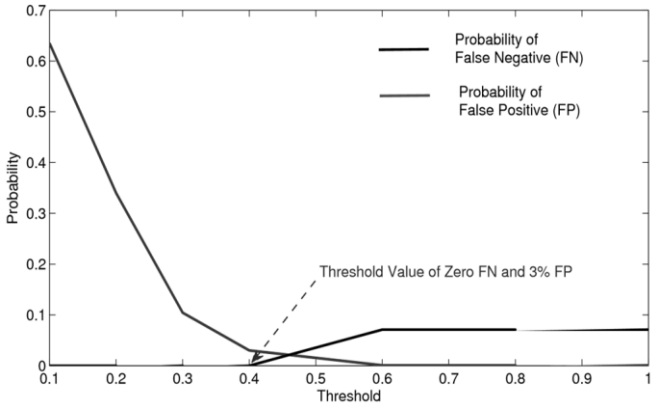


Fig.5. False Positive Vs False Negative

Discussion

Most time series analysis of network data assumes stationarity and predictability of the given series. But the analysis has shown that such assumptions may not be valid at all times. Also, various transformations on the time series are studied and compared. It is found that appropriate transformations on the series can lead to linear models and good prediction. Predictability of a series is found to have increased with slowly decaying ACF, low Hurst exponent value and low relative variance. Detection of low rate TCP SYN DoS attack is demonstrated with 3% false positive for detecting all the attacks. Since the solution is applied at the edge router of the victim server and is based on the number of half-open connections, it is scalable to Distributed DoS attack as well.

2.2.2. Prediction and Control of Bandwidth:

A frequent and commonly occurring phenomenon in any network is the excessive use of bandwidth by a select group of users. Analysis of IITM proxy server logs shows that (Fig.6) upper band users who constitute less than 10% of users consume about 40% of the traffic, roughly following the Pareto distribution. It is commonly observed that even though the LAN has a dedicated high-speed link to the ISP, during peak hours, poor response times are observed by Internet users. Abnormal use may be treated as a special case of an anomaly. We therefore

propose the framework shown in Fig.7 for detection and control of excessive bandwidth usage. We first monitor user activity, and based on this we generate models for predicting bandwidth usage for each user. The training of the models may be done once a month, while prediction may be performed on a daily basis. Then, we predict bandwidth usage, which may be used to categorize each user as belonging to one of three categories – high, middle or low bandwidth users. A suitable control algorithm may be implemented in order to control abnormal usage.

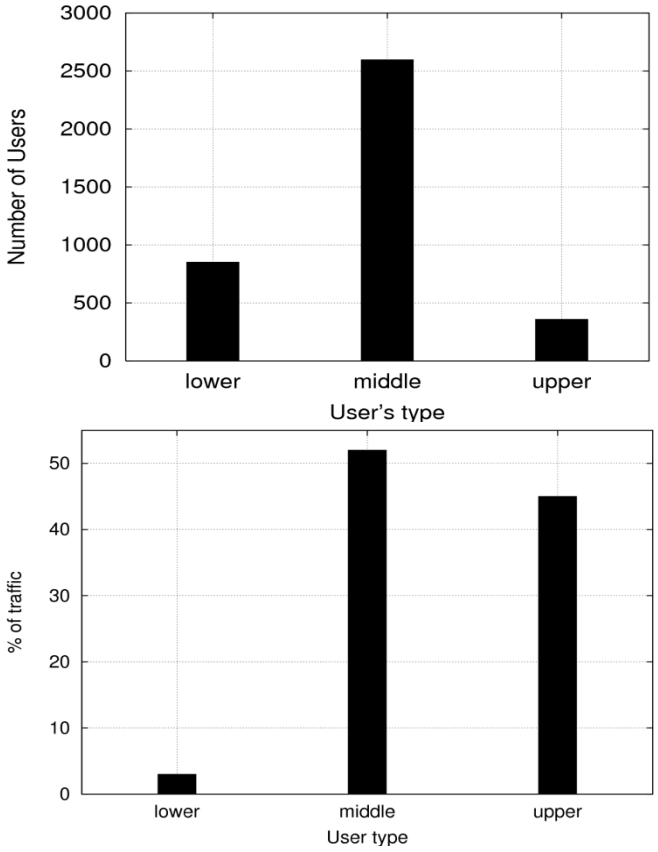


Fig.6. Distribution of users (top) and traffic generated by each category (bottom)

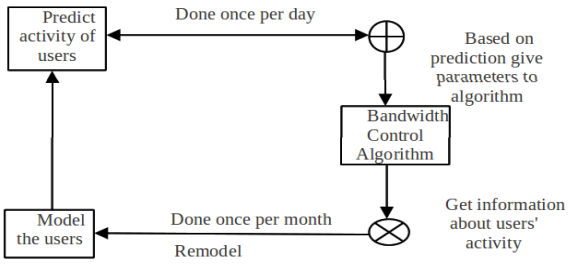


Fig.7. Framework for Bandwidth Prediction and Control

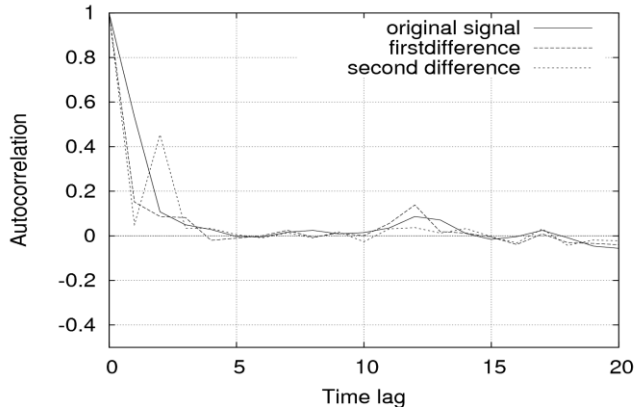
Time series models have to offer many benefits for predicting values at discrete intervals of time. Firstly, they handle well sudden (erroneous) spikes in the data and prevent them from affecting the forecast. Secondly, they automatically adjust to the current trend in the values – which may be caused by changes in the environment – for example – the unavailability of a link to the ISP, which may in turn affect bandwidth usage. Thus, we have chosen time series models for modeling bandwidth usage.

Predicting Bandwidth Usage

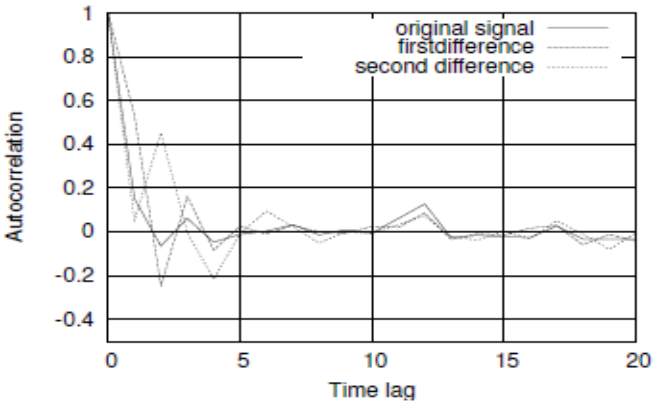
A plot of the autocorrelation and partial autocorrelation functions (Fig.8(a) and (b)) for the original series as well as the first and second difference show that the plots tail off quickly, suggesting that the data is stationary, and so time series models may be used.

All our experiments were carried out using Squid Proxy Server logs for the month of January and February 2009. January data was used for training – i.e. computing model coefficients, and February data was used during the testing phase to compute the error between the forecast and actual values. The features we have considered are number of HTTP requests and number of bytes received for a particular user. Each sample data point in our case consists of the value of a feature (e.g. number of HTTP requests) accumulated over a period of two hours. We performed the following experiments:

1. *Static Coefficients*: Coefficients computed during the training phase were kept constant right through the testing phase.
2. *Dynamic Coefficients*: Coefficients were re-computed each time for forecasting the next value in the testing phase.
3. *Common Model for each Category*: Common model coefficients were computed for each category (e.g. high band users) and forecasting performed using this common model for all users.



(a)



(b)

Fig.8(a) Auto-correlation function and (b) Partial Auto-correlation function

These experiments were carried out using AR, ARI, ARMA and ARIMA time series models for the same data set. The average relative error was computed as the error in prediction normalized by the actual data value. This was used to compare the predictive capability of the different models.

The results indicate that there is no substantial difference between the performance of the static coefficients model and dynamic coefficients model. However, the static coefficients model performed better than the common model. The results for static coefficients across different models have been shown in Table.7. Best forecasting is obtained for ARI and ARIMA models.

Table.7. Comparison of Different Time Series Models

Model	Average relative error for no of bytes	Average relative error for no of requests
AR	2.06	0.86
ARI	0.52	0.51
ARMA	1.6	0.88
ARIMA	0.49	0.49

Entropy-Based Scheme for Classification

Entropy also seems to be a promising alternative means of analysing user access patterns. We may define for example the normalized entropy as

$$H = \frac{-\sum_{u \in U} P(u) \cdot \log(P(u))}{\log(|U|)}$$

where U is the set of URLs accessed by a user. If the user typically has few frequently accessed sites, his normalized entropy value would be low, whereas if he typically accesses a large number of URLs as in the case of a high band user, his normalized entropy value would be high. Fig.9 shows the distribution of users for different values of normalized entropy.

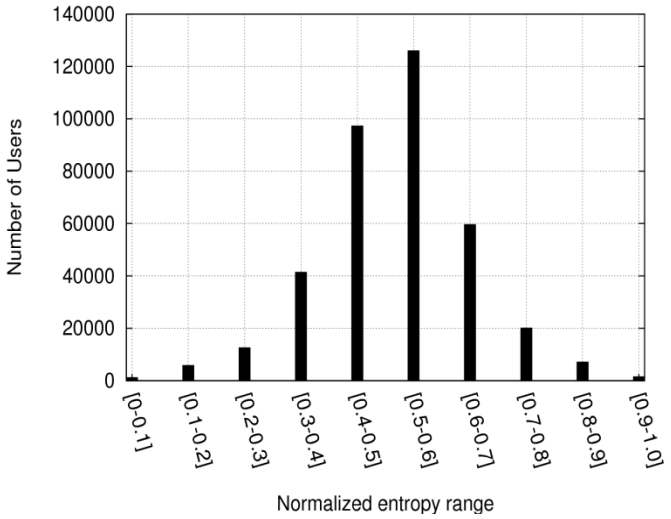


Fig.9. Distribution of Users for Different Normalized Entropy Ranges

We would like to propose as a future direction the use of information-theoretic criteria to design a full-fledged classification scheme.

3. SIMULATION PLATFORM

It is difficult to evaluate complex multi-level resilience strategies that involve the interplay between a number of detection and remediation mechanisms both at the core and edge networks, which must be activated on demand according to events observed in the network (as opposed to hardcoded protocols). In order to evaluate such resilience strategies we have previously proposed the notion of a policy-driven resilience simulator [1], based on the integration of a network simulator and a policy management framework. The toolset allows the use of policies to specify the required management actions to be performed, according to conditions observed during run-time in the simulation. The use of policies for the specification of resilience strategies was previously described in [2]. For example, the policy shown Fig.10 can be used to reconfigure a rate-limiting component based on the occurrence of a high risk event (raised, for example, by an anomaly detection component) and additional contextual information, e.g. the current utilisation on a specific link.

```
on AnomalyDetectorMO . highRisk ( link , src , dst )
if (LinkMonitorMO. getUtilisation () >= 75%)
        do RateLimiterMO . limit ( link , 60%)
```

Fig.10. Management policy for reconfiguring a rate limiter component based on a high-risk event

One of the direct benefits of integrating a network simulator with a policy framework is that we can understand how real policies dynamically affect the operation of resilience mechanisms running within the simulation environment, and then evaluate resilience strategies before they are deployed in the network infrastructure. This permits the evaluation of complex resilience strategies without the need of a real testbed deployment of mechanisms, which typically involves high costs of hardware and effort. The next sections briefly outline the design of the simulation platform and give details about our prototype implementation.

3.1 DESIGN ISSUES

The resilience simulation platform is primarily based on the integration between a network simulator and a policy management framework. In [16], a number of techniques for integrating a network simulator environment and external third party applications were compared: ***(a) socket connection*** relies on proxies within the simulation that maintain connections to third party applications, without incurring major changes to the third party application. This technique, however, may suffer from CPU scheduling problems because simulations typically run faster than the integrated third-party application; ***(b) source code integration*** can be straightforward for simple applications, but it is difficult to be implemented for larger applications because of build dependencies that must be resolved. Furthermore, threads in the third party application can still suffer from CPU scheduling issues and cause problems such as access violations; finally, ***(c) shared libraries*** is similar to source code integration, but is based on the integration between the simulator and the binary code of the third party application. This avoids the problems related to the building process, however, it still suffers from the threading and timing problems.

The integration technique used in our prototype implementation is based on proxies, similar to the *socket connection* method, but using RMI/RPC objects instead. Typically, this technique can be used if data from lower layer protocols is not exchanged with the external application [16]. For the resilience simulation platform, it is expected that CPU scheduling and synchronisation issues can be minimised because packet-level information is not exchanged with the policy framework. Instead, exchanges are limited to selected control events and the corresponding management commands alone.

We have considered the use of the most popular network simulators, including NS-2 [18], NS-3 [19], OMNeT++ [20], SSFNet [21] and OPNET [22]. The choice of a network simulator was driven by a number of requirements, including *(i)* the ability to extend and instrument the simulation tool, *(ii)* the availability of a large number of network models, *(iii)* the scalability and performance of the simulator, and *(iv)* the ability of the simulator to model different types of networks. After an initial evaluation we discarded OPNET as it is a commercial tool and the source code for its simulation kernel is not publicly available, and NS-2 due to recurring reports of its poor scalability. We considered NS-3, OMNeT++ and SSFNet equally suitable for our requirements, but due to our previous experience with SSFNet and familiarity with its API the initial implementation described in [1] is based on this simulator. Recently, we have ported this prototype to an OMNeT++ implementation, since OMNeT++ is considered one of the most popular simulators for research in communication networks.

3.2 PROTOTYPE IMPLEMENTATION

The prototype is based on the integration between the network simulator and the Ponder2 policy framework [23]. Ponder2 implements a policy execution framework that supports the enforcement of *event-condition-action* (ECA) and *authorisation* policies. Policies are written in terms of managed objects programmed in Java, which are stored in a local domain service. Based on our previous investigations [17], Ponder2 was considered to be more extensible and with better infrastructure support when compared to other policy frameworks.

Resilience mechanisms are represented by instrumented components in the simulation environment. They provide callback functions for management operations, and run alongside standard simulated objects. These instrumented components are implemented as OMNeT++ modules, and at the moment most are extensions of the standard *Router* module (we plan to instrument additional modules at the upper layers of the protocol stack as part of our future work).

FlowExporterMO and *IntrusionDetectionMO* are positioned above the network layer, and receive duplicate packets. *RateLimiterMO* is placed in-line between the network and physical layers. Finally, *LinkMonitorMO* was implemented by modifying an existing channel type, allowing us to place it at any position within the network topology.

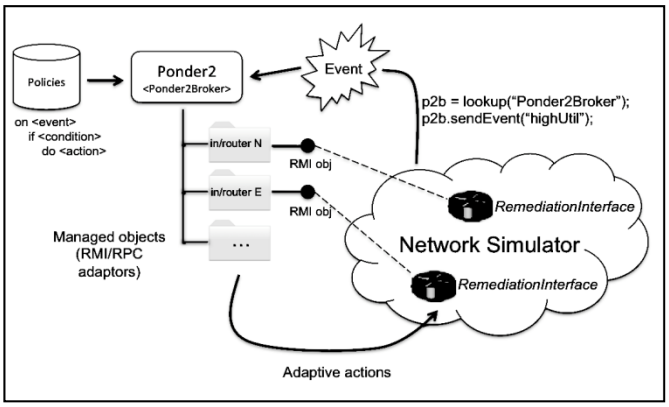


Fig.11. Integration between a network simulator and Ponder2

Such components export their callback functions through management interfaces, which are made accessible by the policy decision point (PDP). Communication between simulated objects and the external policy framework is implemented via adapter objects, which abstract invocation details using the XMLRPC protocol. An *event broker* resolves and maps event notifications from inside the simulation (e.g., anomaly detections, link utilization, etc) to the policy framework, which according to a dynamic set of policies invokes adaptive actions in the instrumented components running within the simulation (Fig.11).

This platform permits the implementation of detection and remediation components both in core or edge networks. It can be used to explicitly model the interactions between these mechanisms and observe how they can dynamically impact the operation of the network. As part of our future work we intend to build a library of instrumented resilience components implementing detection and remediation mechanisms at both the core and the edge portions of the simulated network.

4. MULTI-LEVEL RESILIENCE STRATEGIES

4.1 MULTI-LEVEL WORM DETECTION

Ultimately, multi-level resilience architecture would involve explicit interactions between edge and core mechanisms. We anticipate, for example, the exchanges of *"hints"* between edge and core detection mechanisms to assist in the more effectively gathering of evidence for particular types of anomalies. These hints may, for example, cause the modification of the set of management policies enforced by the components operating in a specific segment of the network. Likewise, we envisage also the collaboration between remediation mechanisms at different levels to mitigate an anomaly.

In order to demonstrate the feasibility of a multi-level resilience strategy we propose a case study based on a worm propagation scenario. In this scenario, a collaborative core/edge methodology would employ cross-layering early detection and remediation of worm propagation before its full effect is achieved. Computer worms can quickly propagate in the Internet due to their self-replication capability, and severely disrupt the operation of the network in particular due to the increased network traffic. Therefore it is needed to detect worm propagations as early as possible in order to allow sufficient

warning time for reaction before the whole network is compromised. Examples of particularly disruptive worms include *Code Red* [3], *Blaster* [4], *Sasser* [5] and *Slammer* [6].

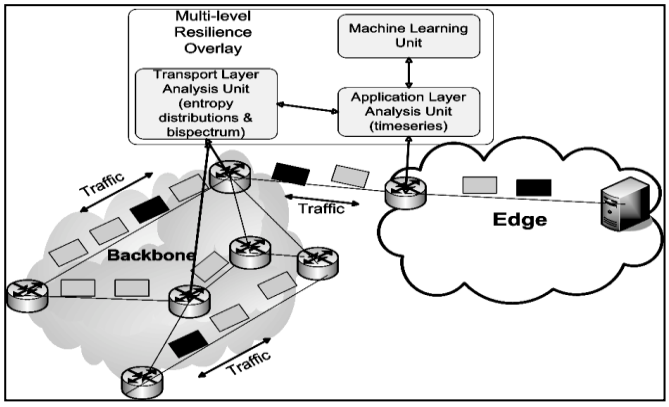


Fig.12. Multi-level architecture for worm detection

Worms typically spread through the exploitation of vulnerabilities in the operating systems. Typically, a worm presents simple attack behaviours, and all computers infected by this worm will send malicious traffic with statistically similar behaviour. It differs from other challenges to network operation, such as a DDoS, in the sense that a DDoS has only one or a small number of targets, whereas a worm has no specific target in the network.

Our proposed multi-level architecture for addressing worm detection includes both transport layer mechanisms operating at the core network as well as application layer mechanisms in the edge network. On the one hand, the transport layer analysis will use entropy distributions (of src/dst ports, src/dst IP addresses, payload, or byte and packet sizes) as input to the Bispectrum analysis. On the other hand, application layer analysis will perform time series analysis on application-layer protocols and use machine learning techniques for storing worm behaviour. The conceptual multi-level architecture for the worm scenario is depicted in Fig.12. This architecture relies on the cooperation between transport and application layer mechanisms to share and correlate information for identifying and predicting worm propagation. We intend to refine this architecture and simulate the interactions between core and edge mechanisms as part of our future work.

5. CONCLUSIONS

In this paper, we proposed the notion of multi-level network resilience. This work is motivated by the potential complementarities between the research being developed at IIT Madras and Lancaster University, in order to provide a collaborative traffic analysis and anomaly detection, combining mechanisms and algorithms deployed both in the core and edge networks. We described in this paper the current work developed at IIT Madras and Lancaster on traffic analysis and anomaly detection, which will form the basis of a multi-level resilience architecture. We also outlined the simulation platform we intend to use to evaluate our joint work, as well as a worm propagation case study scenario that illustrates how mechanisms from the two realms can be combined.

As part of our future work, we are going to refine the ideas presented in this paper and, based on the case study scenario, propose a multi-level architecture using a combination of traffic classification and detection techniques operating both at the core and edge networks. The simulation platform will allow us to evaluate how network-, transport- and application-layer mechanisms can exchange information and operate together.

Moreover, as part of the work on the multi-level resilience architecture we also intend to develop a joint data-set analysis and validation of anomaly detection methods and time series models. Both IIT Madras and Lancaster use time series models on different datasets, the former focusing on application and packet data from the network edge and the latter focusing on network flows at the network core. Although the approaches are sound, promising and have established similar ground truths, it is still necessary to validate these approaches on common data sets.

ACKNOWLEDGEMENTS

This work has been supported by the EPSRC funded India-UK Advance Technology Centre in Next Generation Networking.

REFERENCES

[1] Schaeffer-Filho, P. Smith, and A. Mauthe, "Policy-driven network simulation: a resilience case study", *in Proceedings of the 26th ACM Symposium on Applied Computing (SAC 2011)*, ACM. Taichung, Taiwan.

[2] P. Smith, A. Schaeffer-Filho, A. Ali, M. Schöller, N. Kheir, A. Mauthe, D. Hutchison, "Strategies for Network Resilience: Capitalising on Policies", *in: Proceedings of the 4th International Conference on Autonomous Infrastructure, Management and Security (AIMS 2010)*, ser. LNCS. Zurich, Switzerland. Springer, pp.118-122.

[3] eEye Digital Security, .ida March 2011, "Code Red" Worm. Available online at: http://www.eeye.com/Resources/Security-Center/Research/Security-Advisories/AL20010717. Access March 2011.

[4] eEye Digital Security, Blaster Worm. Available online at: http://www.eeye.com/Resources/Security-Center/Research/Security-Advisories/AL20030811. Access March 2011.

[5] eEye Digital Security, Sasser Worm. Available online at: http://www.eeye.com/Resources/Security-Center/Research/Security-Advisories/AD20040501. Access March 2011.

[6] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the Slammer worm", *IEEE Security and Privacy Magazine*, Vol. 1, No. 4, pp. 33-39, 2003.

[7] A. K. Marnerides, "Traffic Deomposition and Characterization", *In Proceeding of Multi-Service Networks (MSN'10)*, Cosener's House, Abingdon, Oxford, UK, 2010.

[8] L. Cohen, "Time-Frequency Distributions – A Review", *Proceedings of the IEEE*, Vol. 77, No. 7, pp.941-981, 1989.

[9] H. Choi, W.J.Williams, "Improved Time-Frequency Representation of MultiComponent Signals Using Exponential Kernels", *in IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 37, No. 6, 1999.
[10] The MAWI working group : http://mawi.wide.ad.jp/mawi/
[11] L. Cohen, T. E. Posch, "Positive Time-Frequency Distribution Functions", *in IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol ASSP-33, No. 1, 1985.
[12] R. Baraniuk, P. Flandrin and O. Michel, "Information and Complexity on the Time-Frequency Plane", ICASSP, Vol.6, *IEEE International Conference on Acoustics, Speech and Signal Processing*, 1994.
[13] T. Karagiannis, K. Papagiannaki, M. Faloutsos, "BLINC: Multilevel Classification in the Dark", *in ACM SIGCOMM* 2005, Philadelphia, Pensylvania.
[14] F. Auger, P. Flandrin, P. Concalves, O. Lemoine, "Time-Frequency Toolbox for use with MATLAB".
[15] SpeedNet Internet Ports Guide: http://www.speedguide.net/ports_sg.php.
[16] C. P. Mayer and T. Gamer, "Integrating real world applications into OMNeT++", Telematics Technical Report TM-2008-2, Institute of Telematics, Universita¨t Karlsruhe, 2008.
[17] A. Schaeffer-Filho, "Supporting Management Interaction and Composition of Self-Managed Cells", PhD thesis, Imperial College London, 2009.
[18] NS-2 Website. The Network Simulator - NS-2. http://www.isi.edu/nsnam/ns/. Accessed March 2010.
[19] NS-3 Website. The NS-3 network simulator. http://www.nsnam.org/. Accessed March 2010.
[20] OMNeT++ Website. OMNeT++. http://www.omnetpp.org/. Accessed March 2010.
[21] SSFNet Website. Modeling the Global Internet. http://www.ssfnet.org/. Accessed March 2010.
[22] OPNET Website. OPNET Modeler Accelerating Network R&D (Network Simulation). http:// www.opnet.com/solutions/network rd/modeler.html. Accessed March 2010.
[23] K. Twidle, E. Lupu, N. Dulay, and M. Sloman, "Ponder2 - a policy environment for autonomous pervasive systems", *IEEE Workshop on Policies for Distributed Systems and Networks*, pp.245–246, 2008.
[24] D. M. Divakaran, H. A. Murthy, and T. A. Gonsalves, "Detection of SYN flooding attacks using linear prediction analysis", *ICON'06*, Vol. 1, pp. 1 – 6.
[25] "Tcpdump," http://www.tcpdump.org/.
[26] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, "*Time Series Analysis:Forecasting and Control*", Pearson Education, 1994.
[27] G. Kirchgassner and J. Wolters, "*Introduction to Modern Time SeriesAnalysis*", Springer, 2007.
[28] H. Wang, D. Zhang, and K. G. Shin, "Detecting syn flooding attacks," *in Proceedings of the IEEE INFOCOM*, 2002.
[29] H. Wang, D. Zhang, and K. Shin, "Syn-dog: Sniffing syn floodingsources," *in ICDCS*, 2002.
[30] M. Roughan and J. Gottlieb, "Large Scale Measurement and Modelingof Backbone Interent Traffic," *in Internet Performance and Control of Network Systems*, 2002.
[31] T. Karagiannis, M. Faloutsos, and R. H. Riedi, "Long-range dependence: Now you see it, now you dont!" *in GLOBECOM*, 2002.
[32] T. Karagiannis, M. Molle, and M. Faloutsos, "Long-range dependence ten years of internet traffic modeling," *in IEEE Internet Computing*, Vol. 8, pp. 57–64, 2004.
[33] B. Qian and K. Rasheed, "Hurst Exponent And Financial MarketPredictability," *in IASTED conference on Financial Engineering and Applications*, 2004.
[34] "University of california riverside," www.cs.ucr.edu/~tkarag.