Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Science of Computer Programming 60 (2006) 1–26

Science of Computer Programming

www.elsevier.com/locate/scico

# Probabilistic anomaly detection in distributed computer networks

Mark Burgess

*Oslo University College, Cort Adelers gate 30, 0254 Oslo, Norway*

Received 6 July 2004; received in revised form 3 April 2005; accepted 8 June 2005
Available online 27 July 2005

**Abstract**

Distributed host-based anomaly detection has not yet proven practical due to the excessive computational overhead during training and detection. This paper considers an efficient algorithm for detecting resource anomalies in event streams with either Poisson or long tailed arrival processes. A form of distributed, lazy evaluation is presented, which uses a model for human–computer interaction based on two-dimensional time and a geometrically declining memory to yield orders of magnitude improvements in memory requirements. A three-tiered probabilistic method of classifying anomalous behaviour is discussed. This leads to a computationally and memory economic means of finding probable faults amongst the symptoms of network and system behaviour.

*Keywords:* Machine learning; Anomaly detection; Data-mining

## 1. Introduction

Computer anomaly detection is about discerning regular and irregular patterns of behaviour, in the variables that characterize computer systems. The detection of anomalies in computer systems has often been pursued as the unambiguous goal of searching for potential breaches of security; it often goes hand in hand with Network Intrusion Detection, in which content analyses of data are performed in real time with the aim of revealing

*E-mail address:* Mark.Burgess@iu.hio.no.

0167-6423/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2005.06.001

suspicious activity [18,42,25,29,34]. However, this is only one application for anomaly detection; computers can also be approached as self-regulating systems that respond to changes in their environment in order to stabilize their own behaviour. In that case, anomaly detection becomes an integral part of the system's regulatory process. Previously, the cost of performing such an analysis on every host has been prohibitive, but this paper will suggest a way of overcoming this difficulty.

Anomaly detectors apply machine learning and analysis to see whether any long term trends can be found in data. One such approach was suggested in the early 1990s and has recently been revived [30,21]. Automated self-regulation in host management has also been discussed in Refs. [7,9,8], as well as adaptive behaviour [51] and network intrusion detection [46,29]. Other authors have likened such mechanisms to immune systems, striking the analogy between computers and other collective systems in sociology and biology [33,24,8].

The ultimate aim of anomaly detection systems is to have adaptive behaviour that responds in 'real time', so that problematical events can be countered as quickly as possible. However, normal behaviour can only be determined by learning about past events: trends take time to learn and analyse. This paradox can only be resolved by modelling future behaviour, on the basis of a statistical idealization of the past and an observation of the present (like weather forecasting). Even then, a timely response requires a rapid processing of observations. The computational burden of real-time anomaly detection can be considerable. One would thus like to spread the burden as far as possible around the network to minimize the load at any particular place.

This paper is motivated by two goals: to develop an efficient method of anomaly detection that avoids bottlenecks, and implements 'lazy evaluation' to avoid unnecessary computational burden; and to develop a language for expressing one's *policy* about what constitutes an anomalous occurrence, relative to what has already been learned about the signal in the past. We shall make some progress towards both of these goals. The paper is organized as follows:

(1) We begin with a brief summary of the idea of host-based anomaly detection, its aims and motivations in relation to the future challenges of mobile and pervasive computing.
(2) Existing techniques for mapping out empirical data characteristics are summarized and appropriate statistical measures for discussing normality are identified.
(3) The notion of policy is then introduced, to account for the arbitrary aspects of data analysis, such as threshold values and the representation of corroborating environmental information that is not represented in the learning abilities of the nodes.
(4) On the basis of the known characteristics of host data, a pseudo-periodic parametrization of time series is developed, which partitions the arrival process into weekly units. Some comments are made about data distributions and the implications for machine learning.
(5) A description of the limited span, unsupervised learning algorithm, with predictable 'forgetting power', is presented.
(6) Finally, a multi-stage classification of data is proposed, where a response is instigated only if a probabilistic detector signals a *probably significant* event (lazy evaluation).

## 2. Host-based observation of anomalies

In contemporary network design, traffic congestion is avoided by packet switching. i.e. by isolating traffic to ‘parallel’ branches of a network spanning tree that is absolutely necessary for delivering data to their destinations. Computers, or *nodes*, occupy points at the leaves of these branches and therefore experience an individual (subjective) view of the traffic.

Each computer in a network has a different experience of the environmental bath of requests and replies that commit its resources. Because the concept of an anomaly is a subjective one (what is unusual for one node is a common occurrence for another), one might imagine that nowhere in the network is better equipped to reveal anomalies than the nodes at which they finally arrive. That is the viewpoint we shall adopt here, justified by a previous study of the same data as are discussed in this paper [4].

Traditionally, anomaly detection has been centralized to trunk limbs of a network, in the belief that one can only see the big picture if one is in possession of as much of the data as possible at a single place. Some approaches even attempt to combine the streams at different observation points into a single stream [31]. There are two problems with this strategy: it places the entire burden of analysis at a single location, and it does not gain access to anomalies that occur on non-network variables (e.g. disk usage), since these are never transmitted across the network.

The utility of centralized analysis cannot be completely dismissed, but it can be shown to have a limited value [4], and its clear disadvantage is the bottlenecking of traffic that is contrary to modern network design; placing the burden of computation at a single location, meaning that additional computing facilities are required. Studies at Oslo University College find that correlations between nodes are generally too noisy to be useful, unless there is an obvious functional relationship between hosts by design. The conclusion is that there is little to be gained by sharing resource data between hosts [4], since the only clear results are already known in the logistic map of services for the network. Hence there is no pressing need for centralization or serialization of traffic.

Another compelling reason for abandoning the idea of serialization of the data stream is that computers will soon be ubiquitous and devices will be transmitting and receiving data without any regard for a centralized authority, over unguided media. In such a world, the strategy of trying to centralize anomaly detection, at a single gateway, is flawed. A detection scheme in which each host node is responsible for itself and no others reflects the true distributed governance of the network and embodies the move from monolithic centralized control to the more ‘free market economy’ approach.

The present work is based on the idea of computer immunology [55,8], in which one considers every computer to be an independent organism in a network ecology. The cfengine project places the individual computer rather than the network centre stage, in the belief that soon a majority of nodes will not be aligned with any centralized authority [7,8]. Serialization of traffic is abandoned and one combines the analysis of network variables with the analysis of internal host variables, using the natural filtration of data by packet switching.

Arriving events have to be classified and counted in order to identify their statistical significance. They have internal attributes, such as names, addresses, values, with semantic interpretations. These internal attributes contain information that can be used to identify what is meant by an anomaly, by placing events in a specific context and category. An anomaly engine is therefore a ‘prism’ and ‘spectrometer’, or a decision tree that expands an event arrival/renewal process [27] into a spectrum of distinct attributes. By looking at these attributes, with policy criteria that are appropriate for each, and then reassembling the information into a consistent picture, we perform something analogous to a ‘medical scan’ of the incoming event, which allows us to determine its significance to the system.

All scientific observations are facilitated in the context of a model, and one must therefore formulate such a model for classifying the observations as they arrive, in order to derive their meaning. The separation of scales is a crucial aspect of any model [13]; the distinction between trend and event plays a special role for anomaly detection [12]. Measurements of time series autocorrelations show that significant variations are only observed in trends over times of the order of greater than 20 min in human-driven activity [15]. Since an anomaly response time can be up to 30 min in most systems, whether they depend on humans or automation, there is no point in labelling data much more extensively than this, even though many hundreds, even thousands, of individual events can occur per minute. The trends which we must learn in the data do not change as quickly as the microscopic details of the data stream and do not need to be sampled more often than a typical rate of change.

The philosophy in this work thus diverges from the strategy of examining every event exhaustively. A change-event only acquires importance if can be successfully decoupled from a known trend. We thus use a compromise between autocorrelation of numerical event scales and macroscopic level correlations, and split time into granules of five minutes. The data collector measures signals for a whole granule before deciding how it should respond to each coarse grained event.

One ends up with a decision based on the following spectrum of attributes:

- The significance of the arrival time (the granule label).
- The significance of the arrival rate (number per granule), relative to a trend (average number per granule).
- The probable uncertainty of in the assumed trend (a specified number of standard deviations above or below the average trend).
- Entropy content of the distribution of symbol content within granules (described below).
- The symbolic content of specific attributes themselves, collected over a granule.

The memory required to implement this characterization is quite small: it comprises the space required to store each measurement granule, plus the space required to remember the significant attributes within the measured granule, for a finite number of granule labels.

The remainder of the paper considers how to rationally optimally compare incoming granule observations to a memory of what is statistically normal, using an economical method.

## 3. Entropy: Symbolic uncertainty

Interpreting the meaning in a data stream is central to the problem of defining when an event is an anomaly. ‘Anomalousness’ is a subjective judgement, made within the context of past experience, and can be codified into a 'policy' about what is sufficiently anomalous to warrant a response. We can use numerical estimates of significance (statistics), but a complete analysis must also take into account the symbolic attributes of the arrivals too.

In order to define a practical policy for what anomalies are, one must have a straightforward classification of criteria. This is a challenge for the symbolic content, which can comprise many different data types, but we can use a simple information theoretical measure as a first approximation.

In random event processes we learn distributions of values for observations, which have characteristic shapes, and hence characteristic uncertainties. A sharp distribution about a given value means low uncertainty; a broad or flat distribution signifies a highly uncertain value. In either case, one must take the uncertainty in data into account before drawing conclusions about it. This means that one requires a measure of that uncertainty, in order to acknowledge it and use it.

The simplest gauges of a distribution are its *moments*, of which the variance (second moment) is the most well known. The square root of this (or standard deviation $\sigma$) is a simple scale of the uncertainty. However, except in the case of Gaussian distributed data, there is no clear relationship between the uncertainty and $\sigma$. In digital distributions, i.e. histograms with few classes, the standard deviation is inaccurate and clumsy.

A very convenient measure of uncertainty, for any type of data, is the Shannon entropy [52,13,9]. In the theory of information, the Shannon entropy is a numerical characterization of a value distribution $p_i = n_i / \sum_i n_i$, based on frequency counts $n_i$ of types $i$:

$$S = -\sum_{i=1}^{C} p_i \log p_i, \tag{1}$$

where $1 \leq i \leq C$ runs over the distinguishable classes of observation, $n_i$ is the number of events of type $i$ and $p_i$ is the normalized 'probability' of measuring an event of type $i$ in a similar signal.

The value of the entropy embodies the learning that has occurred about the random process and provides a convenient scalar measure for making policy about a random process. The entropy has a minimum value of zero, when all the observations are in a single class, and a maximum value of $S = \log \sum_i n_i$, when each class is occupied equally. This provides an adaptive, relative scale that can be applied to any interval of observation.

**Example 1.** Consider the entropy of a particular attribute: the origin IP address of a data stream: in a network data stream, packets come from different IP addresses. In the following example, we see that traffic has arrived from five different IP addresses (interpreted as different symbol classes), but predominantly from the first address. It forms a sharply focused distribution, as cfengine identifies:

```
Frequency: 157.158.24.40    |****************************+ (47/53)
Frequency: 80.203.17.11     |*  (1/53)
Frequency: 66.196.72.28     |*  (1/53)
Frequency: 80.202.77.107    |**  (2/53)
Frequency: 80.213.238.106   |**  (2/53)
-
Scaled entropy of addresses = 12.7 %
(Entropy = 0 for single source, 100 for flatly distributed source)
```

This is a low entropy distribution, because most of the symbols (in this case IP addresses) are of a single type. The alphabet for this comparison is learned in situ, and is the set of five Internet Protocol addresses listed above.

If we measure entropy as a percentage of the maximum attainable value, then it can itself be classified into high, middle and low, using arbitrary thresholds. This may then be used as a filter in policy description. If we need a more specific characterization, there is always the standard deviation (the square root of the second moment of the distribution).

## 4. Lazy attribute extraction

'False positives' or 'ghost anomalies' are events where current algorithms find problems that are bogus; they are the familiar lament of anomaly detection designers. The dilemma is to know when an anomaly is 'false' and when an anomaly is uninteresting. However, false and uninteresting are two rather different criteria. To call an anomaly false is to assume that we have pre-decided a policy for what is truly an anomalous event and what is not. To call an anomaly interesting is to suggest either that a feature of the data is not only abnormal but highly unusual or that it is usual but not according to a recognizable pattern. Unfortunately, both of these criteria are matters of opinion rather than absolute measuring sticks. What is missing from most network anomaly detectors is an ability to express policy decisions about what is desirable and undesirable information.

In the present work, it is assumed that false anomalies occur for two main reasons:

- Because one attempts to classify data inappropriately (without a model).
- Because the policy for distinguishing anomalies is overconstrained.

The latter is often a by-product of the security applications of anomaly detection: one is easily duped into overt 'cold war' paranoia that leads to an arms race of sensitivity: the desire to scrutinize every event.

Looking, as many have, to biological detection in the immune system [33,24,8] for inspiration, one finds an excellent yet imperfect system that is cheap to operate. Cost is important: an immune system that was so expensive that it has to kill us to keep us alive would be of little use. A host must continue with its primary function while detecting and responding to anomalies.

The biological immune system is a multi-tiered reactor with many levels of detection and a short memory. Organisms tolerate small amounts of harmful material, but mobilize countermeasures once they begin to do damage [38]. The key method by which the immune system avoids responding to false positives is the use of *co-stimulation*. A confirmation signal is required (like a dual key system) to set off an immune response. The system is lazy, in that it need not look for the confirmation signal unless the probability of an anomaly is already high.

For a computer detection scheme, we can use the same approach. First we look for a probably anomaly by comparing observation to learned experience. If the event looks probable, we can consider the evidence determined from a supporting semantic model. This reduces the amount of processing involved in detecting anomalous behaviour to an absolute minimum, by using 'lazy evaluation':

(1) The system learns the normal state of activity on a host.
(2) New events are considered anomalous if reliable data can place them at some sufficient number of standard deviations above the expected value at a given time of week.
(3) The meaning of ‘sufficient’ must be defined as a matter of policy, in a given context. It is subjective.
(4) If an event is found anomalous, it is dissected through our ‘prism’ in terms of its informational entropy and symbolic content.

The policy referred to here can be used to describe which anomalies are *interesting*, and specify when to respond.

**Definition 1** (*Anomaly Identification Policy*). An anomaly identification policy is a specification of predicates and thresholds, for observable attributes in a stream of observations, that is used to identify anomalies. It maps the set of observable attributes into the Boolean set {True,False}.

The strategy used here is to base anomaly policy on what has been learned about past behaviour. For this, we measure the significance of events relative to known trends and variations, and then use the symbolic content of the events, if statistical anomalies are found, to determine how we should respond to them. This breakdown is important, because it emphasizes the need for a *policy* for describing the importance of events in a local environment. A policy codifies information that is not available by direct observation of the host state (information that would require evolutionary timescales to incorporate in biological systems) and is therefore an important supplement to the regulatory system.

**Example 2.** For example, suppose one observes anomalous amounts of World Wide Web traffic often come from a single IP address source. Given no further information, one might dismiss a lot of traffic from a single source as a scan by an Internet search engine. However, search engines generally scan from a number of IP addresses in parallel. The *IP address entropy* of a friendly search engine scan is relatively high. So one might not be worried about high entropy, high traffic combinations.

By examining the IP addresses contributing to a granule, and trying to resolve them, however, one sees that low entropy sources are sometimes associated with unregistered IP addresses (those which are not in the Domain Name Service or DNS). Such addresses are often highjacked or spoofed addresses and make one immediately suspicious of the source. Hence one can use this information and now codify a policy of responding to low entropy statistical anomalies from unregistered IP addresses. Policy is therefore a specification of acceptable attributes, here: arrival rate, address entropy and address resolvability.

Fig. 1 shows an example of how one can easily split the example of a multifaceted network event into separate attributes that can be evaluated. The incoming packet is first examined to see whether it is an IP (Internet Protocol) packet. If so, it has an address, a port number (except for ICMP) and a ‘layer 3’ encapsulation type (TCP, UDP etc). The different kinds of events can be counted to learn their statistical significance (we call these counting variables) and the remaining symbolic information (Internet

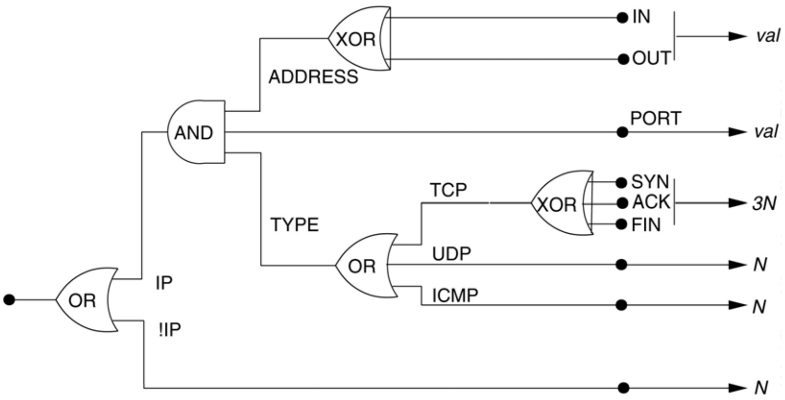

Fig. 1. An example network prism that splits an incoming event into generic categories. The signal enters at the left hand side and is classified as it passes to the right. This can be viewed as a reversed sequence of logic gates. One ends up with six frequency variables of magnitude $N$ that count arrivals and two symbolic values.

addresses and port numbers) can be stored temporarily while the current sample is being analysed. A sample is a coarse grained ensemble of events, collected over a five minute interval.

We now have two questions: how are data learned and how are events identified as statistically significant?

## 5. Arrival–renewal processes and self-similarity

A question that has been raised in recent years is that of the type of arrival or renewal process experienced by the end nodes in the network. This is often relevant for network analyses in which one attempts to model anomalies by looking at inter-arrival times of events, i.e. especially where one attempts to invoke memory of the recent past to track persistent events like connections.

Traditionally, arrival processes have been assumed to be memoryless Poisson processes and analyses have used time correlations [32,42,43] to gauge likelihood of anomaly, but measurements of network traffic and indeed computer behaviour in general show that the arrival processes of normal computer operations sometimes have long tails and exhibit power law behaviour [36].

If a renewal process is *stable*, one has the chance of characterizing it with a time series model. Two kinds of stable distributions are known for the inter-arrival times of renewal processes: Poisson distributions and generalized stable Lévy processes [49] (power laws). Aside from these, all other distributions are unstable under convolution [13].

Unfortunately, traffic volumes are rarely high enough to see stable distributions at leaf nodes. This has consequences for the analysis the time series: statistical quantifiers sometimes diverge or become ill-defined. In particular, correlations over absolute time have little value.

The two inter-arrival time distributions of interest are the Poisson and the power law:

$$N(\Delta t, n = 1) = (\lambda \Delta t)^1 \mathrm{e}^{-\lambda \Delta t} = \sum_{m=0}^{\infty} \frac{(-1)^{-1}(\lambda \Delta t)^{m+1}}{m!} \tag{2}$$

$$N(\Delta t) \qquad = \mu(\Delta t)^{-\alpha}. \tag{3}$$

In general, both $\lambda$ and $\mu$ can be functions of time, in the presence of trends. We shall make use of this below.

We can roughly gauge the type of process by means of an approximate measure of its degree of self-similarity, called the Hurst exponent $H$. This is a scaling exponent for the time series over an range of average granule sizes. In other words, one assumes a general scaling law for a time series observation $q(t)$ made at time $t$:

$$s^{-H} q(st) = q(t). \tag{4}$$

One then applies this to locally averaged functions:

$$s^{-H} \langle q(st) \rangle = \langle q(t) \rangle, \tag{5}$$

where $\langle \cdot \rangle$ is defined in Eq. (11). The exponent $H$ can be estimated for real data by noting that, over an interval $\Delta t$,

$$\langle \max(q(t)) - \min(q(t)) \rangle_{s\Delta t} = s^H \langle \max(q(t)) - \min(q(t)) \rangle_{\Delta t}, \tag{6}$$

i.e.,

$$H = \frac{\log \left( \frac{\langle \max - \min \rangle_{s\Delta t}}{\langle \max - \min \rangle_{\Delta t}} \right)}{\log(s)}. \tag{7}$$

The values above unity signify probable power laws, with bursty behaviour. The data used in this paper fall into two main groupings. Some data for these are summarized in Table 1. The results show a wide variety of behaviours in the signal, as measured over many months, some of which would tend to indicate self-similar behaviour. One therefore expects to have problems with the naive analysis of time correlations in these data. What is clear from comparing the actual graphs of data with their Hurst exponents is that the type of arrival process (characterized by $H$) is in no way correlated with the ability to separate signal from noise in this periodic parametrization discussed in this paper (characterized by small error bars, e.g. in Fig. 3).

Another approach is required. In fact, we can avoid troubles associated with inter-arrival time tails entirely, below, by means of a simple transformation that integrates long or short inter-arrival time tails completely, by projecting them into a periodic time topology. This will leave us with a completely normalizable framework, with no ambiguities.

It was shown in Ref. [10] that the transformation, to be described, may be used to represent the data in arrival processes, even with noise. Indeed, it is implicit in the Fourier theorem that any function projected into a periodic topology can be represented using a spectral (count per interval) representation.

Table 1
Approximate Hurst exponent ranges for different variables show that the data exhibit a variety of scaling behaviours once projected into a periodic framework. Some show long tail indications while others have only Gaussian jitter

| $q(\tau)$ | $H(q)$ |
|---|---|
| Users and processes | $0.6 \pm 0.07$ |
| Network connections (various) | $1.0 - 2.1 \pm 0.1$ |

## 6. Two-dimensional cyclic time parametrization

The solution to the arrival time issue is to make time itself finite and cyclic, using what is known about the processes in a computer system [16].

By making our model of time periodic we solve two problems: we are able replace the idea of time correlation with a trend analysis based on counting, i.e. a traditional frequency analysis, and we achieve a significant simplification of the machine-learning and analysis algorithms.

The basic observation that makes resource anomaly detection simpler and more efficient than traditional time series analysis [5] is that there is an inhomogeneous pattern to human resource usage, and this is reflected in computer resource usage. This can be used to remodel and simplify the data. It yields, in effect, a simple and automatic supervised classification of the machine-learning process.

The approximate weekly periodicity observed in computer resources [16], allows one to parametrize time in topological slices of period $P = 1$ week, using the relation

$$t = nP + \tau, \tag{8}$$

$n = 0, 1, 2, \ldots$. In this parametrization, time assumes a cylindrical form, labelled by two interleaved coordinates $(\tau, n)$, both of which are discrete in practice [11]. We then sum or average over the $n$ variable, leaving a single angular variable $\tau$, after a renormalization of the distribution.

This transformation is indisputably justified in any process that is periodic in time, by Fourier theory; however, it may also be applied to functions that are only pseudo-periodic or close to periodic on average. See Ref. [10] for a detailed discussion of this model. The key observation is that the arrival processes have periodic inhomogeneities $\lambda(\tau)$ and $\mu(\tau)$.

Applying this transformation to the problematical inter-arrival time distributions in Eq. (3), we see that the Poisson law may be written in terms of a periodic, dimensionless quantity $\tau/P$, by making the substitution Eq. (8):

$$\begin{aligned} N(\Delta\tau/P, n) &= \sum_{m=0}^{\infty}\sum_{n=1}^{\infty}(-1)^m \frac{\lambda(\tau)^{m+n}}{n!m!}(nP + \Delta\tau)^{n+m} \\ &\propto f(\tau) \sum_{\alpha=-\infty}^{\infty} \zeta_\alpha(\Delta\tau/P) \end{aligned} \tag{9}$$

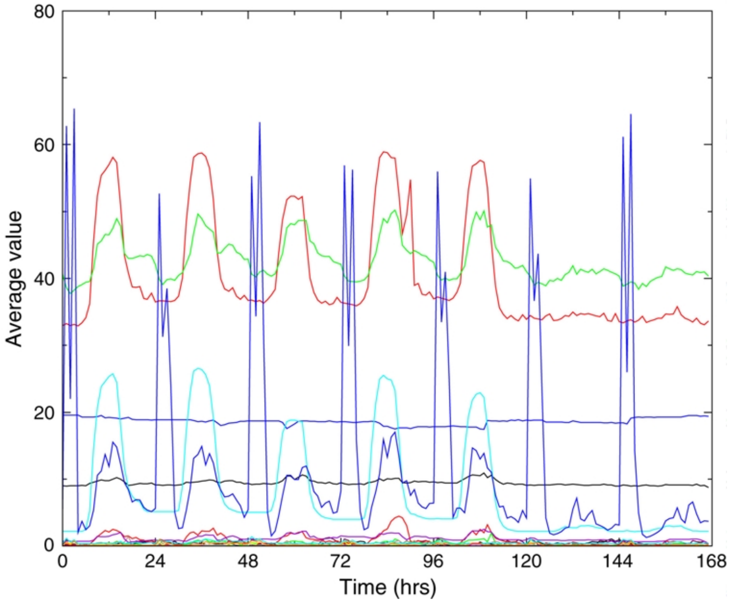

Fig. 2. A weekly periodogram of some resource variable expectations. These values are scaled and smoothed, measured by cfengine, and uncertainties have been suppressed. The lines represent the effective average thresholds for normal behaviour. Note how each line has its own characteristic 'shape' or pattern of usage, which the system learns by empirical measurement.

and the power law is simply

$$N(\Delta\tau/P) = \mu(\tau)\sum_{n=0}^{\infty}(nP+\Delta\tau)^{-\alpha} \propto f'(\tau)\zeta_\alpha(\Delta\tau/P), \tag{10}$$

where $f$, $f'$ are unknown, but periodic functions. Both of these results are expressed in terms of generalized forms known as zeta functions, multiplied by periodic amplitude functions. In this form, the Poisson law is seen to be simply the generalization of the power law, formed by superimposing many signals with different characteristic decay rates.

What is significant is that the resulting function of $\tau/P$ is well behaved for large times, over the limited interval $0 \leq \tau \leq P$; indeed, the large $n$ behaviour plays a less and less significant role that can be normalized away in the sums. The short $\tau$ behaviour is still singular for each power law component of positive $\alpha$, but this is easily eliminated by coarse graining, which prevents $\delta\tau$ from ever being zero. These transformations are the standard tricks of renormalization statistics [39] (see Section 8). To summarize, by restricting attention to fixed-size granules $\delta\tau$, taken one at a time, we can sum out and renormalize away the effects of the arrival process, leaving a periodic trend and an unknown amount of scatter.

As one would expect from the Fourier theory, a superposition of many such signals will lead to a strong trend if there are periodic variations in the data. Any other signals will appear as fluctuations that will either appear as noise or as anomalies, depending on their relative magnitude. In Fig. 4, on the other hand, there is no convincing periodicity to be

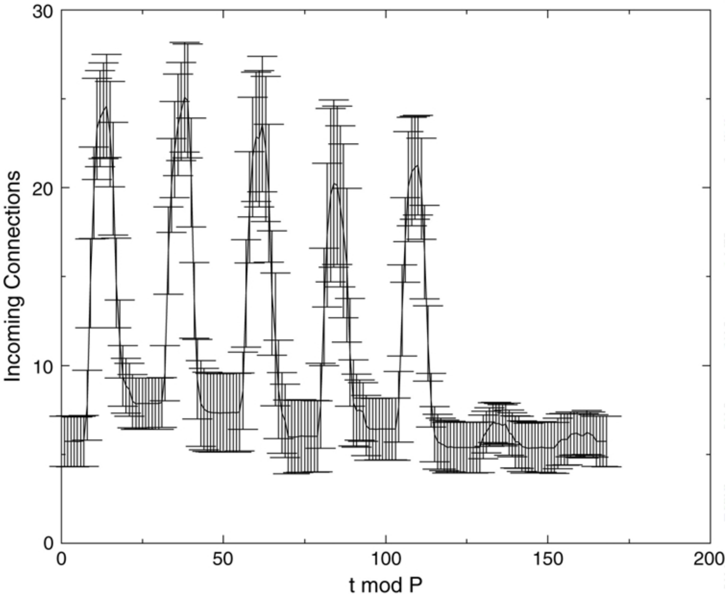

Fig. 3. Measured time trace of NETBIOS name lookups, averaged over 19.4 weeks. This basic pattern has been measured several times, starting with no data, and has remained stable for almost two years. It has a clear signal. Uncertainties characterized by error bars represent the standard deviations $\sigma_{\langle\langle P\rangle\rangle}(t \bmod P)$.

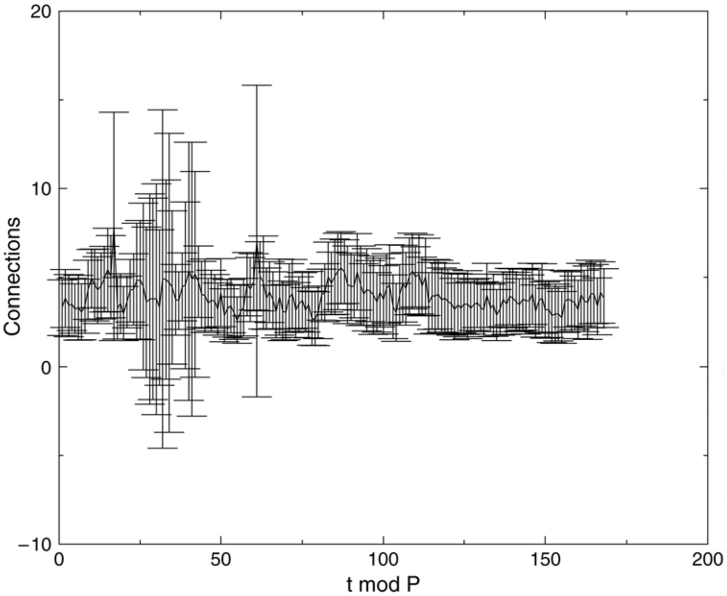

Fig. 4. Measured time trace averaged over 19.4 weeks of ftp connections. Here there is no discernible signal in the pattern of variations. The level of noise represented by the error bars is greater than the variation in the signal.

seen in the pattern of the data series, which begs the questions of whether this method of insisting on periodicity is ‘appropriate’. However, the contention here is that this is not the way to look at it. The empirical studies indicate that weekly periodicity is by far the most important structural trend in human computer data [16]. If we can model the periodic parts of a signal that are attributable to weekly behaviour, then it will be possible to extract those parts and get them under control. Everything that is left, relative to this trend, is either noise that can be renormalized away, or an anomaly that can be identified more easily. Empirical studies provide compelling evidence for this.

We shall therefore *assume* that it is appropriate to project into periodic time, since the lack of periodicity is simply caused by a lack of a human interaction and hence a lack of signal, meaning no pattern of normalcy and nothing would be gained by allowing time to extend indefinitely. In pedestrian terms, if we cannot separate significant variations from noise, in this parametrization, we cannot distinguish anomalies either, in this model; thus no harm is done by making the assumption.

The periodic parametrization of time means that one can average (over $n$) the values at each point $\tau$, leading to a mean and standard deviation of observations at each corresponding time of the week. Both the mean and standard deviations are thus functions of $\tau$, and the latter plays the role of a scale for fluctuations at $\tau$, which can be used to grade their significance. The cylindrical parametrization also enables one to invoke a compression algorithm on the data, so that one never needs to record more data points than exist within a single period. It thus becomes a far less resource intensive proposition to monitor system normalcy. This compression is not possible in a linear time series approach, using splines etc. [30].

Test data are taken to be a number of universal and easily measurable characters (see Fig. 2):

- Number of users.
- Numbers of processes.
- Unix ‘load averages’.
- Average utilization of the system (load average).
- Number of incoming/outgoing connections to a variety of well known services.
- Numerical characteristics of incoming and outgoing network packets.

These variables have been examined earlier and their behaviour is explained in [9,16]. Other variables might be studied in the future. A further advantage of this model is that the learned average behaviour can be stored indefinitely by using a simple database format, covering only a single working week in granules of five minutes, as we now show in the next section.

## 7. Separation of scales

In a dynamical, stochastic system, there are two basic kinds of change: non-equilibrium change (slow, progressive, trend variation that occurs on a timescale that is long compared to measurement) and fluctuations (occurring on a timescale that is fast compared to the measuring process). If the system is approximately stable, i.e. close to a steady state, then the combination of these can be used to characterize the recent history of the system. Fluctuations can be measured as a time series and analysed [30] in order to provide the necessary information, and averaged out into *granules* or sampling intervals. During a sampling interval, data are collected, the mean and variance of the sample are found and these values are stored for the labelled interval. The sampling interval is chosen arbitrarily on the basis of the typical autocorrelation length of the data being observed [16].

Time series data can consume a lot of space. However, a considerable compression of the data can be achieved, and several orders of magnitude of computation time can be spared by separating the weekly data trends from the arrival of random events and by updating belief-estimates of only those trends, iteratively, rather than using an offline analysis based on a complete journal of the past (see Section 11).

This can yield a good approximation to an appropriate sliding window, time series data sample [16]. One obvious approach, for such a method, is to use a convergent geometric series in order to define an average which degrades the importance of data over time: in other words, a series which forgets old data at a predictable rate. After a certain interval, the oldest memories in the data contribute only an insignificant fraction to the actual values.

## 8. Computing expectations with memory loss

Our aim, then, is to benefit from the fact that we do not have to store the entire history of the system in order to infer its normal behaviour in the present. Rather, we can develop a Markov-style model in which the system not only learns but also forgets at a predictable rate.

Following the maintenance theorem of Ref. [12], we define the normal behaviour of a system as its *expected behaviour*. The standard deviation of the data *values* is a convenient scale with which to measure anomalies, and we now ignore the time-like nature of the arrival process for events.

For a body of data, consisting of $N$ data points $\{q_1, \ldots, q_N\}$, one conventionally defines averages and standard deviations as follows:

$$\begin{aligned}
\langle q \rangle &= \frac{1}{N} \sum_{i=1}^{N} q_i \\
\langle q | Q \rangle &= \frac{1}{N} \sum_{i=1}^{N} q_i Q_i \\
\sigma &= \sqrt{\frac{1}{N} \sum_{i=1}^{n} (q_i - \langle q \rangle)^2} \\
&= \sqrt{\langle q^2 \rangle - \langle q \rangle^2} \\
&= \sqrt{\langle \delta q | \delta q \rangle} \\
&= \sqrt{\langle \delta q^2 \rangle}. \qquad (11)
\end{aligned}$$

This notation will help us to see that alternative definitions, with tailor-made properties, can unambiguously replace the expressions $\langle\cdot\rangle$ for averages. The use of these measures as characteristic scales in no way implies a model based on Gaussian distributions.

To update the memory of averages and variances, an algorithm is required, satisfying the following properties:

- It should approximate an offline sliding window time series analysis that forgets old data at a predictable rate [16].
- It should present a minimal load to the system concerned.
- It must have a predictable error or uncertainty margin.

These goals can be accomplished straightforwardly as follows. We replace the usual expectation function with a new one with the desired properties, in such a way that derived quantities bear the same functional relationships as with the usual definitions:

$$\langle q\rangle \rightarrow \langle\langle q\rangle\rangle \tag{12}$$

which gradually forgets old data in a controlled manner. Similarly, we replace the standard deviation (or second moment of the data distribution) with

$$\sigma(\langle q\rangle) \rightarrow \sigma(\langle\langle q\rangle\rangle), \tag{13}$$

where

$$\sigma(\langle\langle q\rangle\rangle) \equiv \sqrt{\left[\langle\langle q^2\rangle\rangle_N - \langle\langle q\rangle\rangle_N^2\right]} = \sqrt{\langle\langle \delta q^2\rangle\rangle_N}. \tag{14}$$

The new expectation function is defined iteratively, as follows:

**Definition 2** (*Iterative Expectation Function*). Let $q$ be an observation, and $\langle\langle q_i\rangle\rangle$ be the $i$th estimator of the average, with geometric fall-off; then $\langle\langle q_i\rangle\rangle$ may be defined by the recurrence relation

$$\begin{aligned} \langle\langle q\rangle\rangle_{i+1} &= (q \mid \langle\langle q\rangle\rangle_i) \\ \langle\langle q\rangle\rangle_0 &= 0, \end{aligned} \tag{15}$$

where

$$(q_1|q_2) = \frac{w\,q_1 + \overline{w}\,q_2}{w+\overline{w}}. \tag{16}$$

and $w, \overline{w}$ are constants.

Significantly, the number of data is now unspecified (we denote this by $i \rightarrow \infty$) meaning that this algorithm does not depend specifically on the arbitrary number of data samples $N$. Instead it depends on the ratio $w/\overline{w}$ which is a forgetfulness parameter.

We note that, as new data points are measured after $N$ samples, $\langle q\rangle$ changes only by $q/N$ while $\langle\langle q\rangle\rangle_N$ changes by a fixed fraction $wq/(w+\overline{w})$ that is independent of $N$. Thus as the number of samples becomes large over time, the $\langle\cdot\rangle$ measure ceases to learn anything about the current state, as $q/N \rightarrow 0$, but $\langle\langle\cdot\rangle\rangle$ continues to refresh its knowledge of the recent past.

The repeated iteration of the expression for the finite-memory average leads to a geometric progression in the parameter $\lambda = \overline{w}/(w + \overline{w})$:

$$\langle\langle q\rangle\rangle_N \equiv (q_1|(q_2|\ldots(q_r|(\ldots|q_N)))) = \frac{w}{w+\overline{w}}\,q_1 + \frac{\overline{w}w}{(w+\overline{w})^2}\,q_2 + \cdots + \frac{w\,\overline{w}^{r-1}}{(w+\overline{w})^r}\,q_r + \cdots \frac{\overline{w}^n}{(w+\overline{w})^n}\,q_N. \tag{17}$$

This has easily predictable properties. Thus, on each iteration, the importance of previous contributions is degraded by $\lambda$. If we require a fixed window of size $N$ iterations, then $\lambda$ can be chosen in such a way that, after $N$ iterations, the initial estimate $q_N$ is so demoted as to be insignificant, at the level of accuracy required. For instance, an order of magnitude drop within $N$ steps means that $\lambda \sim |10^{-N}|$.

The learning procedure proposed here is somewhat reminiscent of a Bayesian probability flow [44,57], but it differs conceptually. A Bayesian algorithm assumes that each new datum can tell us the truth or falsity of a number of hypotheses. In our case, we have only single hypothesis: the normal state of the system, with a potentially unlimited amount of input. We do not expect this procedure to converge towards a static ‘true’ value as we might in a Bayesian hypothesis. Rather we want to implement a certain hysteresis in the normality function.

We now need to store the following triplets in a fixed-size database:

$$\{\tau, \langle\langle q\rangle\rangle(\tau), \sigma^2(\langle\langle q\rangle\rangle, \tau)\}. \tag{18}$$

We also use the $\delta$ symbol to represent the current deviation from average of a pseudo-periodic variable $q(t)$:

$$\delta q(t) \equiv q(t) - \langle\langle q\rangle\rangle_t. \tag{19}$$

To satisfy the requirements of a decaying window average, with determined sensitivity $\alpha \sim 1/N$, we require:

(1) $\frac{w}{w+\overline{w}} \sim \alpha$, or $w \sim \overline{w}/N$.

(2) $\left(\frac{\overline{w}}{w+\overline{w}}\right)^N \ll \frac{1}{N}$, or $\overline{w}N \ll w$.

Consider the ansatz $w = 1 - r$, $\overline{w} = r$, and the accuracy $\alpha$. We wish to solve

$$r^N = \alpha \tag{20}$$

for $N$. With $r = 0.6$, $\alpha = 0.01$, we have $N = 5.5$. Thus, if we consider the weekly update over five weeks (a month), then the importance of month old data will have fallen to one hundredth. This is a little too quick, since a month of fairly constant data is required to find a stable average. Taking $r = 0.7$, $\alpha = 0.01$ gives $N = 13$. On the basis of experience with offline analysis and field testing, this is a reasonable arbitrary value to choose.

## 9. Pseudo-periodic expectation

The recent behaviour of a computer can be summarized by $n$th-order Markov processes, during periods of change, and by hidden Markov models during steady state behaviour, but one still requires a parametrization for data points. Such models must be formulated on a periodic background [10], owing to the importance of periodic behaviour of users. The precise algorithm for averaging and local coarse graining is somewhat subtle, and involves naturally orthogonal time dimensions which are extracted from the coding of the database. This is discussed here using an ergodic principle: a bi-dimensional smoothing is implemented, allowing twice the support normally possible for the average, given a number of data points. This provides good security against "false positive" anomalies and other noise.

Consider a pseudo-periodic function, with pseudo-period $P$,

$$\begin{aligned} q(t) &= \sum_{n=0}^{\infty} q(nP+\tau) \qquad\qquad (0 \le \tau < P) \\ &\equiv \sum_{n=0}^{\infty} \chi_n(\tau). \end{aligned} \tag{21}$$

This defines a set of periodic functions $\chi_n(\tau)$ with periodic coordinate $0 \le \tau < P$. The time coordinate $\tau$ lives on the circular dimension. In practice, it is measured in $p$ discrete time intervals $\tau = \{\tau_1, \tau_2, \ldots, \tau - p\}$. In this decomposition, time is a two-dimensional quantity. There are thus two kinds of average which can be computed: averages over corresponding times in different periods (topological averages $\langle\chi(\tau)\rangle_T$) and averages of neighbouring times in a single period (local averages $\langle\chi(\tau)\rangle_P$). For clarity, both traditional averages and iterative averages will be defined explicitly. Using traditional formulae, one defines the two types of mean value by

$$\begin{aligned} \langle\chi\rangle_T(\tau) &\equiv \frac{1}{T}\sum_{n=l}^{l+T}\chi_n(\tau) \\ \langle\chi\rangle_P(n) &\equiv \frac{1}{P}\sum_{\ell=\tau}^{\tau+P}\chi_n(\ell) \end{aligned} \tag{22}$$

where $l$ and $\tau$ are arbitrary start values and $P, T$ are integer intervals for the averages, in the two time-like directions. Within each interval that defines an average, there is a corresponding definition of the variation and standard deviation, at a point $\tau$:

$$\begin{aligned} \sigma_T(\tau) &\equiv \sqrt{\frac{1}{T}\sum_{n=l}^{n=l+T}(\chi_n(\tau) - \langle\chi\rangle_T(\tau))^2} = \sqrt{\langle\delta\chi_T|\delta\chi_T\rangle_T} \\ \sigma_P(n) &\equiv \sqrt{\frac{1}{P}\sum_{\ell=\tau}^{\ell=\tau+P}(\chi_n(\ell) - \langle\chi\rangle_P(\ell))^2} = \sqrt{\langle\delta\chi_P|\delta\chi_P\rangle_P}. \end{aligned} \tag{23}$$

Limited memory versions of these may also be defined, straightforwardly, from the preceding section by replacing $\langle\delta q|\delta q\rangle$ with $\langle\langle\delta q^2\rangle\rangle$ from Eq. (15):

$$\begin{aligned} \langle\chi\rangle_P &\to \langle\langle\chi\rangle\rangle_P \\ \langle\chi\rangle_T &\to \langle\langle\chi\rangle\rangle_T. \end{aligned} \tag{24}$$

Similarly, the deviations are given by

$$\begin{aligned}\sigma_{\langle\!\langle T\rangle\!\rangle}(\tau) &\equiv \sqrt{\langle\!\langle(\delta_{\langle\!\langle T\rangle\!\rangle}\chi)^2\rangle\!\rangle_T}\\ \sigma_{\langle\!\langle P\rangle\!\rangle}(n) &\equiv \sqrt{\langle\!\langle(\delta_{\langle\!\langle P\rangle\!\rangle}\chi)^2\rangle\!\rangle_P}\end{aligned} \tag{25}$$

where, for any measure $X$, we have defined

$$(\delta_{\langle\!\langle P\rangle\!\rangle}X) \equiv X - \langle\!\langle X\rangle\!\rangle_P \tag{26}$$

$$(\delta_{\langle\!\langle T\rangle\!\rangle}X) \equiv X - \langle\!\langle X\rangle\!\rangle_T. \tag{27}$$

Here one simply replaces the evenly weighted sum over the entire history with an iteratively weighted sum that falls off with geometric degradation.

A major advantage of this formulation is that one only needs to retain and update two values per variable, the mean and the variance, in order to obtain all the information, not $2N$ data, for history size $N$.

## 10. Cross-check calibration: Annealing potential anomalies

We now have a stable characterization of the time series that makes optimum use of the known structure of the data. In a two-dimensional time series, one has two independent vectors for change that must be considered in generating a normal surface potential for comparison.

So far, the discussion has focused on a single periodicity in the time series data; however we must also acknowledge the existence of sub-patterns within a single period. These patterns are not clear harmonics of the period, so they cannot be eliminated by redefinition of the period itself. Rather, they lead to apparent short term variations that, together with noise, can lead to apparent anomalies that are false.

It comes as no surprise to learn that the major sub-pattern is a daily one, once again driven by the daily 24-hour rhythm of activity, but it is not immediately clear why it is not the fundamental period of the system. The weekly pattern can be reproduced with very low levels of noise, because the variations over many weeks of the weekly pattern are small. The daily pattern has much higher levels of uncertainty, since not all days are equivalent: weekends typically show very low activity and artificially increase the uncertainty in the expected signal. The difference between a weekend day and the variation in any day of the week over several weeks is significant; hence the working week yields the cleanest periodicity, at least in the data that have been collected in the present investigations [16].

One might perhaps expect that, in a clearly periodic signal, minor sub-patterns in observations would average out leaving a clear and smooth trend. This would render the problem of false anomalies insignificant; however, the favoured sensitivity of the new expectation function to recent events can also lead to artificial uncertainty. Random fluctuations at closely neighbouring times of day can also lead to apparent variations in the expectation function that are not statistically significant. We therefore define a procedure of smoothing or annealing the anomalies by computing a *local average* as the smoothed vicinity of the current period. A traditional expectation expression for this would be

$$\langle \chi \rangle_L(\tau) \equiv \frac{1}{L} \sum_{\ell=\tau-L/2}^{\tau+L/2} \langle\langle \chi \rangle\rangle_T(\tau), \tag{28}$$

and, in limited memory form, one has

$$\langle\langle \chi \rangle\rangle_L(\tau) \equiv \langle\langle\, \langle\langle \chi \rangle\rangle_T(\tau)\, \rangle\rangle_L, \tag{29}$$

and

$$\delta_{\langle\langle L \rangle\rangle} \chi(\tau) \equiv \langle\langle \chi \rangle\rangle_T - \langle\langle \chi \rangle\rangle_L, \tag{30}$$

with corresponding measures for the standard deviations. Using these averages and deviation criteria, we have a two-dimensional criterion for normalcy, which serves as a control at two different timescales. One thus defines normal behaviour as

$$\{\delta_{\langle\langle L \rangle\rangle} \chi(\tau), \delta_P \chi(n)\} < \{2\sigma_{\langle\langle L \rangle\rangle}(\tau), 2\sigma_{\langle\langle P \rangle\rangle}(n)\}. \tag{31}$$

These may be simply expressed in geometrical, dimensionless form:

$$\Delta(\tau, n) = \sqrt{\left(\frac{\delta_{\langle\langle L \rangle\rangle} \chi(\tau)}{\sigma_{\langle\langle L \rangle\rangle}(\tau)}\right)^2 + \left(\frac{\delta_{\langle\langle P \rangle\rangle} \chi(n)}{\sigma_{\langle\langle P \rangle\rangle}(n)}\right)^2}, \tag{32}$$

and we may classify the deviations accordingly into concentric, elliptical regions:

$$\Delta(\tau, n) < \begin{cases} \sqrt{2} \\ 2\sqrt{2} \\ 3\sqrt{2} \end{cases}, \tag{33}$$

for all $\tau$, $n$, which indicate the severity of the deviation, in this parametrization. This is the form used by cfengine's environment engine [14]. Put simply, cfengine smudges gradients to make them smoother, as well as periodicities; hence, the smoothing algorithm is fully two dimensional.

Fig. 5 shows detailed, high resolution averages for process count data on a given host, over an interval of two months. At this level of detail one sees a jagged curve with error bars which vary considerably in magnitude. Fig. 6 shows an image created using the iterative algorithm, on the same data, at the same resolution. The main trends of the curve are still visible. Error bars have been suppressed to avoid clutter (see Fig. 7 for distribution characteristics).

## 11. Co-stimulation: Judging the calibrated sensitivity

The greatest problem for anomaly detection is in dealing with periods of low activity, i.e. a lack of recent past experience. In a period of low activity, every event is "abnormal". Cold spots pull the averages down and oversensitize the detection of random events. In such a case, only a policy decision can renormalize the threshold level to avoid the detection of certain events.

Our anomaly measurement scale is the standard deviation, but a single standard deviation is often not even resolvable for a lightly used host, i.e. it is less than the discrete counting scale of the events; the appearance of a single new event might trigger a standard

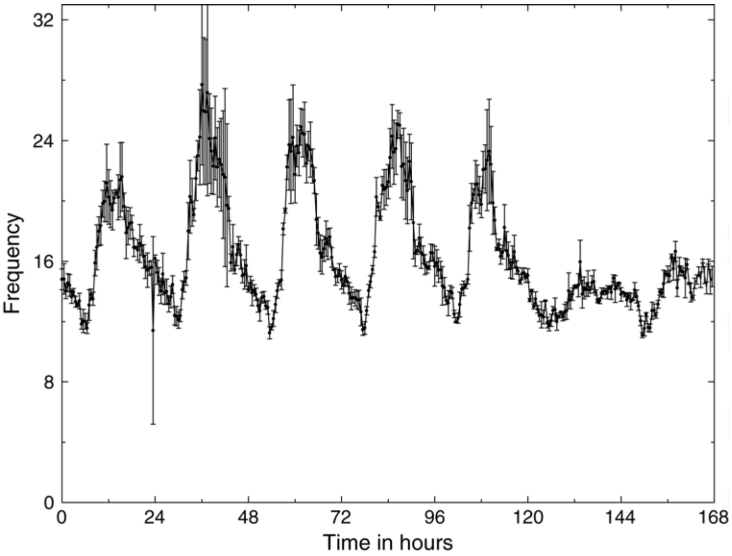

Fig. 5. A high resolution plot of process count behaviour, using the time series sliding window from Ref. [16]. This may be taken as a reference point. The data size is 21114875 bytes.

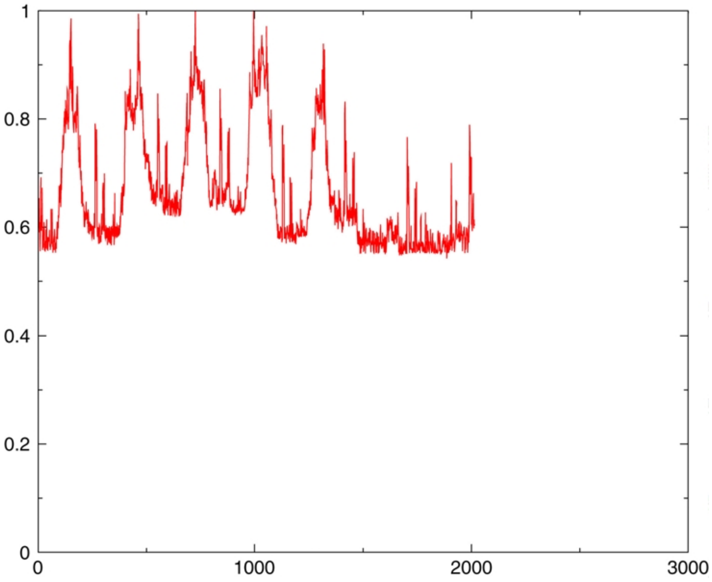

Fig. 6. A high resolution plot using the iterative algorithm using the same process count data as in Fig. 5. The data size is 1531904 bytes (13.8 times smaller).

deviation from the norm. For more heavily loaded hosts, with persistent loading, more reliable measures of normality can be obtained.

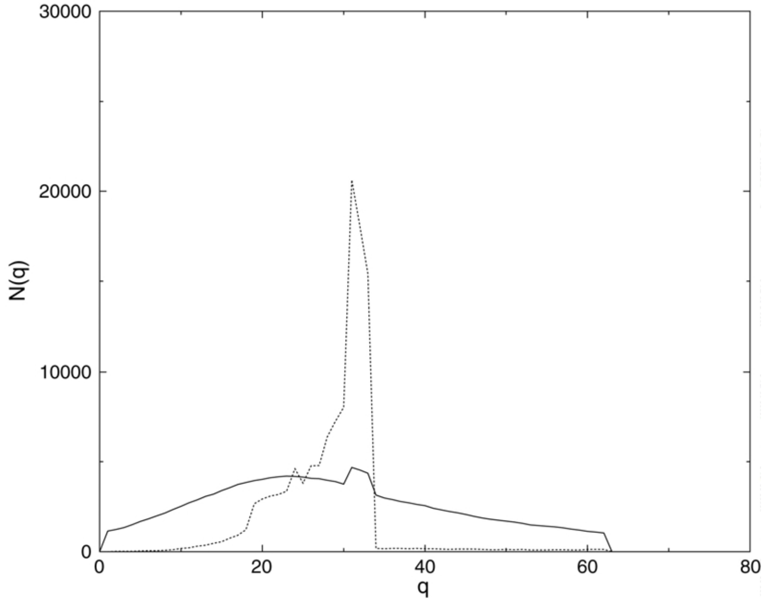

Fig. 7. The scaled scatter distributions, relative to the periodic trend, for HTTP traffic to and from a host. The solid line shows arriving connections, and the broken line shows outgoing connections. The peaks represent normal or expected values. The scatter of two values about the normal is quite different, and far from Gaussian in one case, and thus our idea of anomaly must also be different. The shapes of some of these curves can be calculated using the periodic model; see Ref. [10] for details.

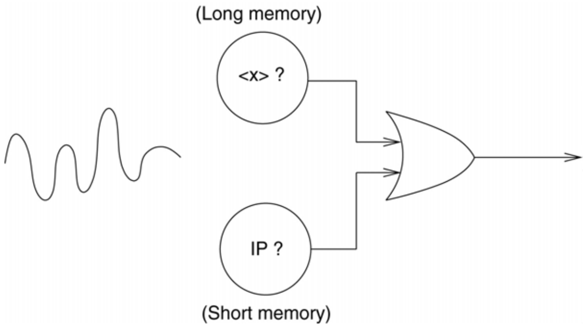

Fig. 8. A strategy of co-stimulation is used to sequentially filter information. First, long term (low grade) memory decides whether an event seems statistically significant and assesses the likelihood of danger. If it is significant, short term (high grade) memory is used to recognize the source of the anomaly.

How can one avoid a deluge of ‘false positives’ in anomaly detection? We must invoke policy to further classify events as interesting or uninteresting, using the information content of the events. As part of the policy, we can combine the symbolic and numerical classifications of events in a kind of ‘co-stimulation’. In other words, one can use qualifiers (see Fig. 8) to decide when to respond to the classified anomaly.

The scheme presented in this work has been implemented and tested as part of the cfengine project [14]. cfengine is a distributed system administration agent, based loosely on the idea of a computer immune system. Anomaly detection is used to identify unusual behaviour that can diagnose problems of configuration and perhaps security. cfengine already has a declarative language that is based on the *classification of observations*. It serves as a useful test-bed for anomaly policies.

cfengine adds descriptive predicates for every true classification of its observations. For example, to generate a simple alert, one could write:

```
alerts:

 entropy_smtp_in_low & anomaly_hosts.smtp_in_high_dev2::

   "LOW ENTROPY smtp anomaly on $(host)/$(env_time)"

   ShowState(incoming.smtp)
```

This would generate an alert if incoming e-mail (SMTP connections) exceeded two standard deviations above normal, at any given time of day, *and* if the traffic was predominantly from a single source (low entropy). Such an event could be a candidate for being a ‘spam’ or junk-mail attack, for instance. The choice of how many standard deviations above expectation is a matter for policy, but it can be guided by the learned shapes of the data, such as those in Fig. 7.

With this policy, a few anomalies per day is reasonable to expect from a moderately loaded host. At this level, anomaly warnings can maintain the attention of a human operator. Less significant anomalies can be handled in silence, by attaching programmed responses to the conditions that arise.

## 12. Some related work

The model used in this work to evaluate anomalies is unlike that of any previous systems known to the author. Nevertheless, it is fitting to place it in the context of other work. The literature on anomaly detection is vast and spans a variety of approaches and architectures, too numerous to chronicle. Only a few milestones are therefore mentioned below.

It is conventional, in the professional literature, to distinguish anomaly detectors from rule-based pattern matchers [46,48]. An anomaly detector uses the idea of unsupervised learning [22,23], whereas an intrusion detection system is based on a supervised expert database of already identified patterns [41,6].

Most anomaly detectors attempt to learn temporal sequences, i.e. patterns or shapes that can be seen in a multivariate data stream. Early work on this for resource anomalies was carried out in [30], using time series analysis and spline identification. Since then, more attention has been given to architectures for data collection, rather than for data analysis—even now, e.g. [28]. This seems to reflect a philosophy that more data will lead to a greater chance of finding anomalies, which is the apogee of the approach considered here. Today, packet analysis using the Snort [48] software is popular owing to its ready availability.

An important landmark in intrusion detection history was the SRI International (formerly Stanford Research Institute) Emerald project [32,45]. The aim of Emerald was to devise a layered approach to real-time anomaly detection in service traffic that was distributed and reintegrated centrally. Emerald was more of a collection system than an approach to anomaly decipherment. As such, the approach used is not necessarily at odds with that presented here, but it does not make apparent use of any compression based on observations of the data's structure. Its handling of statistics is somewhat unclear in the literature, but it was noted that it does not attempt to reduce false positives [40].

A more interesting technique has been used by Forrest et al., applying a probabilistic detection method, which is also inspired by immunology. The approach is fundamentally different to the work here: the authors employ stochastic searching of symbolic information streams, most notably digital streams of system codes. Notable references are [54,20,19], which inspired an application of this paper's methods as a supplementary enhancement in Ref. [3]. No comparison of this method with the present one is possible in the present context. See Ref. [3] for more details.

Statistical approaches have been employed in various Refs. [50,47,26], but these are not similar as periodic trend compression seems to have been used.

The work that comes closest to the present approach is the Bayesian learning algorithm in Ref. [35]. This author stops short of finding a way of using significant trends in the data to eliminate noise. In Bayesian learning, a system updates its belief about what to expect [22,56].

The signal analysis approach in Ref. [2] takes an approach that is distantly related to the strategy, used in this paper, of using periodic functions to approximate the reconstituted time series. It applies wavelet analysis to an unstructured time series; this approach has become popular in the analysis of self-similar time series. However, the analysis is an offline technique and did not yield clear advantages over cheaper methods.

The need for a policy for resolving subjective ambiguities in computer systems has been explored in a variety of access-security related contexts [53,17], but this is not a concept that has been discussed for pattern recognition. For intrusion and anomaly systems, policy usually amounts to defining lists of regular expressions to match symbolic traffic payloads. Although not all symbolic languages are regular, any finite symbolic language is regular [37], and all sequences are finite in practice. The computational simplicity of using regular expressions makes this approach the overwhelming approach of choice.

Policy is normally only applied to Intrusion Detection Systems and firewalls, rather than anomaly detection systems; see for example Ref. [1]. Approaches that attempt to characterize and utilize the shape of statistical distributions, other than implicitly with a Gaussian model, are unknown to the present author.

## 13. Conclusions

The lazy host-based method of anomaly detection, used by cfengine, employs a two-dimensional time slice approach and a policy identification language. The resources required to store learned data are several orders of magnitude less than with traditional sliding window methods, due to an iterative learning scheme based on geometric series.

It weights events on a sliding scale of importance so that recent events are geometrically more important than old events.

The periodic counting parametrization avoids problems associated with long tailed time-correlation divergences. It is unnecessary to assume Gaussian or Poisson statistics in their value and time distributions. The result is a probabilistic method, for detecting anomalous behaviour, that uses a policy-specified deviation from statistical expectation as a first sign of danger, and only then allows symbolic content to confirm or revoke the sign (co-stimulation).

The final aim of anomaly research is to have a turn-key, plug and play solution to the problem of detection, into which users insert their policy requirements and where the machine does the rest. This requires a language for expressing anomalies in intuitive but *quantitative* terms. The model that is developed here allows policy to be expressed in terms of two statistical quantifiers:

- Number of arrival deviations above or below average, at any moment.
- The sharpness or bluntness of internal information within an observation.

Put in more information theoretical terms, this is nothing more than:

- The statistical uncertainty of the environment.
- The symbolic uncertainty of the event.

With just these two parameters, the cfengine project has shown that one can elegantly classify and filter anomalies without supervision, and with only a pre-specification of policy. Thus, we are able to say when an event is anomalous with surprising economy of expression and resources.

What it is still missing, in the technology of system regulation, is the determination of meaning in an anomaly, in the sense of how to mount a response to it. This suggests finding an efficient characterization of the *non-statistical*, symbolic attributes of events, summarized in a *response policy* that attaches countermeasures to the anomalies. This is a separate and challenging problem which must be the subject of future work.

## References

[1] E. Al-Shaer, H. Hamed, Firewall policy advisor for anomaly detection and rule editing, in: Proc. IEEE/IFIP 8th Int. Symp. Integrated Network Management, IM 2003, March 2003, pp. 17–30.

[2] P. Barford, J. Kline, D. Plonka, A. Ron, A signal analysis of network traffic anomalies, 2002.

[3] K. Begnum, M. Burgess, A scaled, immunological approach to anomaly countermeasures, in: Proceedings of the VIII IFIP/IEEE IM Conference on Network Management, 2003.

[4] K. Begnum, M. Burgess, Principle components and importance ranking of distributed anomalies, Machine Learning Journal 58 (2005) 217–230.

[5] G. Box, G. Jenkins, G. Reinsel, Time Series Analysis, Prentice Hall, New Jersey, 1994.

[6] H. Bunke, J. Csirik, Parametric string edit distance and its application to pattern recognition, IEEE Transactions on Systems, Man and Cybernetics 25 (1995) 202.

[7] M. Burgess, A site configuration engine, in: Computing Systems, vol. 8, MIT Press, Cambridge, MA, 1995, p. 309.

[8] M. Burgess, Computer immunology, in: Proceedings of the Twelfth Systems Administration Conference (LISA XII), USENIX Association, Berkeley, CA, 1998, p. 283.

[9] M. Burgess, Automated system administration with feedback regulation, Software Practice and Experience 28 (1998) 1519.

[10] M. Burgess, The kinematics of distributed computer transactions, International Journal of Modern Physics, C 12 (2000) 759–789.

[11] M. Burgess, Two dimensional time-series for anomaly detection and regulation in adaptive systems, in: IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management, DSOM 2002, 2002, p. 169.

[12] M. Burgess, On the theory of system administration, Science of Computer Programming 49 (2003) 1.

[13] M. Burgess, Analytical Network and System Administration—Managing Human–Computer Systems, J. Wiley & Sons, Chichester, 2004.

[14] M. Burgess, Cfengine www site, http://www.iu.hio.no/cfengine, 1993.

[15] M. Burgess, G. Canright, K. Engø, A graph theoretical model of computer security: from file access to social engineering, International Journal of Information Security 3 (2004) 70–85.

[16] M. Burgess, H. Haugerud, T. Reitan, S. Straumsnes, Measuring host normality, ACM Transactions on Computing Systems 20 (2001) 125–160.

[17] N. Damianou, N. Dulay, E.C. Lupu, M. Sloman, Ponder: a language for specifying security and management policies for distributed systems, Imperial College Research Report DoC 2000/1, 2000.

[18] D. Denning, An intrusion detection model, IEEE Transactions on Software Engineering 13 (1987) 222.

[19] P. D'haeseleer, An immunological approach to change detection: Theoretical results, in: 9th IEEE Computer Security Foundations Workshop, 1996.

[20] P. D'haeseleer, S. Forrest, P. Helman, An immunological approach to change detection: algorithms, analysis, and implications, in: Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy, 1996.

[21] Y. Diao, J.L. Hellerstein, S. Parekh, Optimizing quality of service using fuzzy control, in: IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management, DSOM 2002, 2002, p. 42.

[22] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, Wiley Interscience, New York, 2001.

[23] R. Durbin, S. Eddy, A. Krigh, G. Mitcheson, Biological Sequence Analysis, Cambridge University Press, Cambridge, 1998.

[24] S. Forrest, S. Hofmeyr, A. Somayaji, Communications of the ACM 40 (1997) 88.

[25] S. Forrest, S.A. Hofmeyr, A. Somayaji, T.A. Longstaff, Proceedings of 1996 IEEE Symposium on Computer Security and Privacy, 1996.

[26] J.A. Freeman, D.M. Skapura, Neural Networks: Algorithms, Applications and Programming Techniques, Addison Wesley, Reading, 1991.

[27] G.R. Grimmett, D.R. Stirzaker, Probability and Random Processes, 3rd edition, Oxford Scientific Publications, Oxford, 2001.

[28] S.-H. Han, M.-S. Kim, H.-T. Ju, J.W.-K. Hong, The architecture of ng-mon: A passive network monitoring system for high-speed ip networks, in: IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management, DSOM 2002, 2002, p. 16.

[29] S.A. Hofmeyr, A. Somayaji, S. Forrest, Intrusion detection using sequences of system calls, Journal of Computer Security 6 (1998) 151–180.

[30] P. Hoogenboom, J. Lepreau, Computer system performance problem detection using time series models, in: Proceedings of the USENIX Technical Conference, USENIX Association, Berkeley, CA, 1993, p. 15.

[31] IETF, Intrusion detection exchange format, http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-14.txt.

[32] H.S. Javitz, A. Valdes, The SRI IDES Statistical Anomaly Detector, in: Proceedings of the IEEE Symposium on Security and Privacy, May 1991, IEEE Press, 1991.

[33] J.O. Kephart, A biologically inspired immune system for computers, in: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, MIT Press, Cambridge, MA, 1994, p. 130.

[34] C. Kruegel, G. Vigna, Anomaly Detection of Web-based Attacks, in: Proceedings of the 10th ACM Conference on Computer and Communication Security, CCS'03, October 2003, ACM Press, 2003, pp. 251–261.

[35] T. Lane, Machine learning techniques for the computer security, Ph.D. Thesis, Purdue University, 2000.

[36] W.E. Leland, M. Taqqu, W. Willinger, D. Wilson, On the self-similar nature of ethernet traffic, IEEE/ACM Transactions on Networking (1994) 1–15.

[37] H. Lewis, C. Papadimitriou, Elements of the Theory of Computation, 2nd edition, Prentice Hall, New York, 1997.

[38] P. Matzinger, Tolerance, danger and the extended family, Annual Review Immunology 12 (1994) 991.

[39] W.D. McComb, Renormalization Methods: A Guide for Beginners, Oxford University Press, 2003.

[40] P.G. Neumann, P.A. Porras, Experience with EMERALD to date, 2000, pp. 73–80.

[41] B.J. Oommen, R.L. Kashyap, A formal theory for optimal and information theoretic syntactic pattern recognition, Patter Recognition 31 (1998) 1159.

[42] V. Paxson, Bro: A system for detecting network intruders in real time, in: Proceedings of the 7th Security Symposium, USENIX Association, Berkeley, CA, 1998.

[43] V. Paxson, S. Floyd, Wide area traffic: the failure of Poisson modelling, IEEE/ACM Transactions on Networking 3 (3) (1995) 226.

[44] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgen Kaufmann, San Francisco, 1988.

[45] P.A. Porras, P.G. Neumann, EMERALD: Event monitoring enabling responses to anomalous live disturbances, in: Proc. 20th NIST-NCSC National Information Systems Security Conference, 1997, pp. 353–365.

[46] M.J. Ranum et al., Implementing a generalized tool for network monitoring, in: Proceedings of the Eleventh Systems Administration Conference (LISA XI), USENIX Association, Berkeley, CA, 1997, p. 1.

[47] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, Cambridge, 1996.

[48] M. Roesch, Snort, Intrusion Detection System, http://www.snort.org.

[49] K.I. Sato, Levy Processes and Infinitely Divisible Distributions, in: Cambridge Studies in Advanced Mathematics, Cambridge, 1999.

[50] R. Sekar, T. Bowen, M. Segal, On preventing intrusions by process behaviour monitoring, in: Proceedings of the Workshop on Intrusion Detection and Network Monitoring, USENIX, 1999.

[51] M.I. Seltzer, C. Small, Self-monitoring and self-adapting operating systems, in: Proceedings of the Sixth Workshop on Hot Topics in Operating Systems, IEEE Computer Society Press, Cape Cod, MA, USA, 1997.

[52] C.E. Shannon, W. Weaver, The Mathematical Theory of Communication, University of Illinois Press, Urbana, 1949.

[53] M. Sloman, Policy driven management for distributed systems, Journal of Network and Systems Management 2 (1994) 333.

[54] A. Somayaji, S. Forrest, Automated response using system-call delays, in: Proceedings of the 9th USENIX Security Symposium, 2000, p. 185.

[55] A. Somayaji, S. Hofmeyr, S. Forrest, Principles of a computer immune system, in: New Security Paradigms Workshop, September 1997, ACM, 1997, pp. 75–82.

[56] M. Steinder, A. Sethi, Distributed fault localization in hierarchically routed networks, in: IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management, DSOM 2002, 2002, p. 195.

[57] M. Steinder, A. Sethi, A survey of fault localization techniques in computer networks, Science of Computer Programming 53 (2003) 165.