Theoretical System Administration

Mark Burgess – Oslo University College

ABSTRACT

In order to develop system administration strategies which can best achieve organizations' goals, impartial methods of analysis need to be applied, based on the best information available about needs and user practices. This paper draws together several threads of earlier research to propose an analytical method for evaluating system administration policies, using statistical dynamics and the theory of games.

Introduction

System administration includes the planning, configuration and maintenance of computer systems. The discipline of system administration is traditionally founded on the anecdotal experiences of system managers [1, 2], but this can only be carried so far; formal (mathematical) analyses of system administration have only recently begun to enable more scientific studies to be carried out [3, 4]. A lack of formal methods makes it difficult to express objective truths about the field, avoiding marketing assertions and the vested interests of companies and individuals. The aim of this paper is to summarize a mathematical formulation of system administration, which can account for a basis of empirical evidence, and which provides an objective approach to study. This is central to the present discussion on developing system administration as a formal discipline.

In previous work by the author and collaborators, it has been shown how aspects of the average empirical behaviour of systems of computers and users can be modelled using fairly straightforward statistical ideas [5, 6, 7, 8]. This has allowed a coarse statistical model of computer systems to be built, which can be used as a backdrop for studies of system administration. Previous work by other authors has also attempted to look at computer ecosystems in terms of differential equations [9]. In the future additional mathematical models will, no doubt, be devised in order to study other issues.

One of the obstacles to formulating a complete theory of system administration is the complexity of interaction between humans and computers. There are many variables in a computer system, which are controlled at distributed locations. Computer systems are *complex* in the sense of having many embedded causal relationships and controlling parameters. Computer behaviour is strongly affected by human social behaviour, and this is often unpredictable. However, the central question in any scientific investigation is one of balance: can one formulate a quantitative theory of system administration, which is general enough to be widely applicable, but which is specific enough to admit analysis?

The outline of this paper is as follows. To begin the discussion some simplified assumptions about the aims of system administration are stated. Next, two types of quantitative model, describing a computer system interacting with users (possibly via a network), are described and the primitive operations which can be carried out within the scope of the models are identified. The two types of model are referred to as type I (passive) and type II (strategic). A method of quantifying the benefits and flaws of different strategies emerges from this discussion. Strategies for system administration and for user behaviour may then be formulated and arranged in a matrix allowing the task of administrating a computer system can be described in precise game theoretical terms. The primary goal of this work is to provide the recipe for performing this kind of analysis.

Basic Assumptions

Capturing such a complex pursuit as system administration in a few simple rules is presumptuous, but approximately possible if one focuses on core activities. In order to make progress one must agree on some specific aims for users and administrators. The purpose of defining the aims of the interested parties in a computer system, is to come up with a good enough abstraction for system management that specific issues may be addressed in quantifiable terms. In this paper, the word 'system' will be taken to mean any organized collection of computers interacting with a group of users. The assumptions used are these:

- The aim of the system administrator is to keep the system alive and running well so that users can perform a maximum amount of useful work.
- The aim of benign users is to produce useful work using the system. This consumes resources.
- The aim of malicious users is to maximize their control over system resources.

A possible quantitative definition of 'useful work' is the amount of user-data modified on the computer system. plus the information transmitted to or from remote locations. This can be refined for specific purposes. Time spent fighting for control of a damaged machine, or other users, for example, is not useful work for normal users.

This short list of aims does not encompass every eventuality, but it establishes a starting point. In addition to these points, it is necessary to provide a scheme of values about what is subjectively good or bad about the system. When are things going well and when are things going badly? This is done by specifying a *system policy* [10, 11].

> *A system policy is a specification of a system's configuration and its acceptable patterns of usage. A complete policy therefore affects the basic installation of the system and also the way it changes in time due to interaction with users.*

A system policy is the pillar of truth and measuring stick against which one determines whether system activity is acceptable or unacceptable. A sufficiently complete system policy can also include a complete configuration blueprint and thus determine whether the state of configuration is acceptable. The central theorem, which was found in [3] is:

> A sufficiently complete specification of a system policy leads to the notion of an ideal average state for the system. Over time, the ideal average state of the system degrades. The aim of system administration is to keep the system as close to its ideal state as possible.

The meaning of the theorem is that it identifies system administration as a *regulatory procedure*. This idea of regulatory action was originally introduced, using the term *convergence*, in connection with the system administration tool cfengine [5, 12, 13]. Cfengine is a program used to automate the regulation of host state, by making it converge towards its 'ideal state' with every execution of cfengine. For cfengine, the ideal state is achieved when every detail of a computer configuration appears to be correctly implemented and no changes to the system made by users contravene system policy. Thus 'state' refers to adherence to a policy. The cfengine model turns out to be a useful starting point for discussing system administration, since it offers a detailed and concrete idealization of system administration tasks in terms of sequences of primitive actions. Currently, cfengine does not have a complete picture of system state, at all levels, though part of the aim of this kind of work is to improve on that situation. However, by basing a study on this idea one also obtains, as a side effect, a theoretical evaluation of the model which can be used to improve cfengine's design in the future.

Policy, State, and Convergence

Without proving the central theorem in this paper, it is helpful to provide a brief explanation of how the ideal state is constructed, and why it is only possible to insist on an average description of idealness.

State is a snapshot of the condition of a system, which results from its current configuration and the history of all tasks which have consumed and released its resources over time. To picture state, it is helpful to think of a human analogy. In [6], the analogy with human health was drawn. Using another analogy, that of evolutionary fitness or adaptation for a purpose, host state can be envisioned on an arbitrary scale, which makes the ideal state that condition in which the system is best able to perform its tasks. As with humans, general fitness of a computer system is a combination of two parts: a part which is determined by inherited properties and a part which is the result of its interaction with the environment.

For humans, the state of fitness would be the sum of genetically determined attributes (roughly speaking, a policy for the operation of the organism) and current physical fitness (the attunement or degradation resulting from interaction with environment). For a computer system, there is a similar duality: state refers to a part which is the sum of all configuration and policy decisions (basic design quality) and a part which results from an interaction with users and network impulses (input/output).

Thus state separates into policy and environment. The state of a computer system $S(t)$ changes continuously with time, due mainly to the interaction with the environment, but also internally, as a general consequence of the second law of thermodynamics (a statistical inference which notes that the number of ways in which a system can be disordered is far greater than the number of ways it can be ordered (adhering to policy), thus any random change is statistically more likely to lead to disorder than to order). This is the principle of increase of entropy [14, 15].

The environment of a computer system can be thought of as an external batch of transactions (see Figure 1), i.e., input and output which appears and disappears as users interact with the system. Each transaction makes use of resources and has the possibility of affecting the state $S(t)$ of the system by an amount $\delta S(t)$. The number of transactions is generally large for periods of time over which one expects the host state to change significantly: transactions last usually milliseconds, whereas host behaviour is self-similar often over days and weeks [7].

From empirical studies [7, 8], one has a picture of a computer system as having an stable average condition over periods of time, but fluctuating considerably in response to specific transactions. In other words, over days and weeks, computer resources change, but over many weeks the pattern of usage has a mean value which shows a stable pattern. At any given time, the actual values of resource variables are different from the mean, but these differences average out to an average condition, or state. If this average state of the host is to be maintained near its ideal state then one hopes that the fluctuations from the mean

$\delta S(t)$ are small, i.e., the disturbance to the system resulting from user interactions results in only a small change to the actual state.

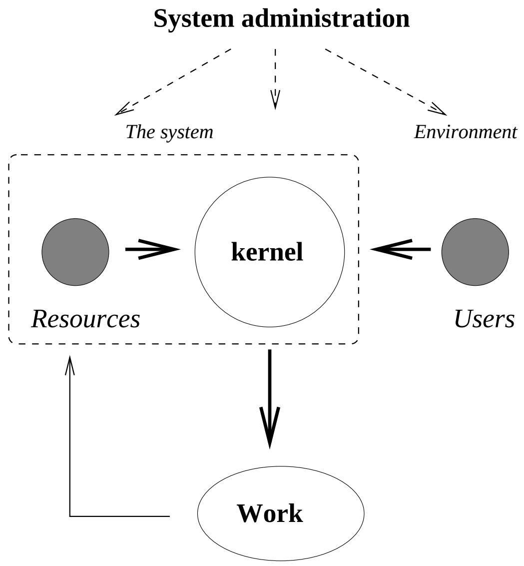

Figure 1: System administration is a regulative function over time.

There are thus four ideas of state which need to be considered: the ideal average state $\bar{S}^*(t)$, whose existence is implied by policy, the actual average state of the system $\bar{S}(t)$ which is the mean value of behaviour over weeks, the actual state of the system $S(t)$ and fluctuations from the average state $\delta S(t)$. All of these are functions of time. The latter three are related by a time-dependent relation:

$$S(t) = \bar{S}(t) + \delta S(t).$$

In order to speak of an average state, one has to say what average means. This has been defined precisely, using the theory of dynamical systems in [4] and turns out to be a regular arithmetical mean, calculated from a sliding window data sample, which advances over time. Because computer behaviour shows approximate periodic repetition, the average is defined in terms of co-cycles, over days and weeks: the two sociological influences which have profound implications for computer behaviour. Our empirical studies, at Oslo, have shown that fluctuations in the state must be averaged over a window of at least two or three weeks [7, 8] in order to see reliable stable behaviour under normal usage. This is the time scale over which users repeat their behaviour several times, within the framework of daily/weekly cycles.

Note that the ideal average state is only approximately constant (it changes slowly, at the same rate as the average changes), whereas the other states change with more rapidly time (on the time scale of individual interactions with the system). The average ideal state changes much more slowly than the actual state, precisely because it is averaged over coarser grains of time.

A notion of *idealness* can thus only be characterized for an average state because the system is constantly changing as users interact with it. Even the mean value is changing slowly with time. In physics of statistical systems, this is referred to as non-equilibrium behaviour. However, the fact that this decomposition is possible is important. It separates the effects of independent scales from one another. What happens in the short term is different to what happens on average, since one deviation might correct another, leaving no net problem. For example, if a user consumes a large amount of resources for a brief time (a temporary file, for instance) while performing useful work, this will only affect the actual state of the system for the duration of that task, provided the file is removed afterwards. A policy of file temporary file garbage collection can always remove such a file even if a user doesn't. The average state will therefore be relatively unaffected by short term changes. The meaning of the environmental ideal average state is therefore to define an interval of stability for interaction with environment. The system will always deviate from the so-called ideal state, but that need not be a problem as long as it does not deviate far from it for long periods of time.

There are two types of disturbance δS: those which (on average) preserve the state of the system, i.e., those which release as many resources as they consume, and those which consume resources without releasing them. The latter kind of disturbance is the most dangerous to the integrity of the system, since it can lead to runaway behaviour which sees the end of the system. This is the case in which it is necessary to introduce countermeasures to protect the state of the system. This is the purpose of computer immunology [6]. When large fluctuations are at hand, the system is in an intrinsically unstable state.

How can changes of state be characterized precisely? An obvious choice is to use the mathematical idea of a lattice, or discrete vector space. Although computer behaviour often has the appearance of a continuously varying load, the actual changes are all discrete in nature. Any interactive change in the system may be broken down into a sequence of discrete primitive operations. See Table 1 for the primitives used by cfengine [5, 12].

Any change to the configurable system, can be expressed in terms of these primitives. In addition to these, there are kernel variables which contain data that can be used to determine environmental state. Each independent primitive can be thought of as an axis in an n-dimensional lattice. Each change in the state of the system, of a given type, is a movement through the lattice in that direction. Moreover, since the averaging procedure for environment effectively divides up time into co-cyclic discrete units, (days and weeks) and scaled coarse-grained intervals, it is possible to draw the the state of the system on a lattice (see Figure 2). Mathematicians note: the lattice is only conformally distorted by changes in the averaging procedure; the structure is preserved.

Primitive type T^i	Comments/Examples
Create file object	Touch
Delete file object	Tidy garbage
Rename file object	Disable
Edit file	Configuration
Edit access control	Permissions
Request resource	Read/Mount
Copy file	Read/write
Process control	Start/stop
Process priority	Nice
Configure device	ifconfig/ioctl

Table 1: cfengine primitives.

In principle, there is a single point in the lattice which represents (at any given time) the most ideal state possible. In practice, one is only interested in keeping the system in a region, not too far from this practically unobtainable ideal point.

Changes to system policy must also be discrete strings of these primitives, since they have to be implemented using the primitives, and thus a change in policy is simply a translation of the ideal state through the lattice.

Suppose one places the ideal state arbitrarily at the origin of this lattice. The further the system deviates from this origin, the more precarious the state of the system. Eventually when the state strays a sufficient distance from the origin, the system will exhaust its resources and fail completely. In the intervening distance, the system is working in accordance with policy when it is close to the ideal state. Using the Euclidean distance as a Hamming distance, for change in the system, it is possible to see that the number of corrective actions for required to return the system to its ideal state grows only linearly, however the number of possible corrective procedures increases exponentially.

The number of equivalent paths $H(\vec{d})$ back to the ideal state is:

$$H(\vec{d}) = \frac{(\sum_{j=1}^{n} d_j)!}{\prod_{k=1}^{n}(d_k!)}$$

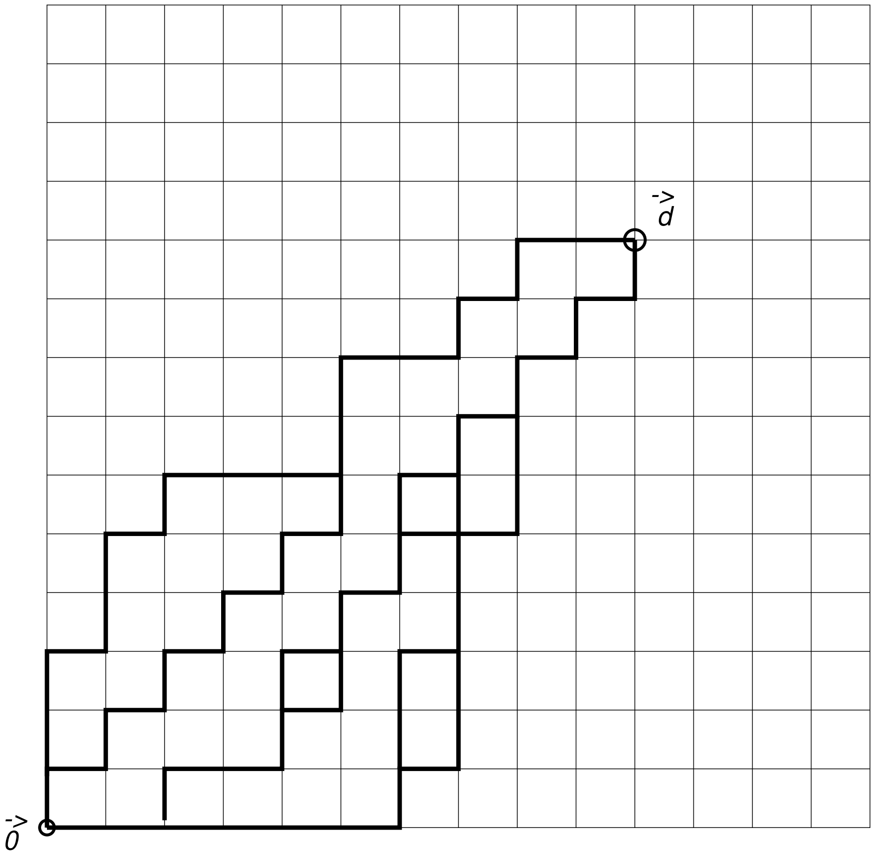

Figure 2: Deviations from the ideal state may be visualized as a random walk through a lattice of n-dimensions (here only two are shown). The number of paths of equal length by which one can return to the origin increases rapidly with the distance. For simplicity one may think of the axes as deviation due to policy and deviation due to usage.

This grows rapidly with the Euclidean distance $|\vec{d}|$:

$$|\vec{d}| \equiv d = \sqrt{\sum_{i=1}^{n} d_i^2}$$

The conclusion is that it is in the system's best interests to remain close to the ideal state at all times. If the remedy to a particular large deviation were unknown, the search for a remedy, in state space, would become extremely time-consuming as the magnitude of the problem increased. The expense or 'hopelessness' $H(\vec{d})$ measures this more than exponentially divergent problem. This hopeless search is the fate of any immune system without specific expert knowledge.

To summarize, every computer system, with a system policy, has an ideal state which is based on policy and environmental considerations. This ideal state fluctuates and degrades with time. The aim of system administration is to regulate the system to be close to this ideal state.

Modelling Computer Behaviour

An analysis of behaviour in a computer system, interacting with its environment of users and network impulses, requires two types of model, which may be referred to as passive (type I) and strategic (type II). The distinction refers to the level perceived intent behind the changes which take place.

In a type I description, the computer is viewed as being a mechanism coupled to a pseudo-periodic, random bath of impulses from an environment. One considers the effect of the this signal of impulses on the state $S(t)$. This is the view taken in [7, 8, 16, 4]. A type I model relates the behaviour of computers to that of other interacting, dynamical systems in mathematics and physics. This type of description is easily formulated and can be used to predict some of the average behaviour when the system is approximately stable. It works particularly well when large numbers of users, or transactions are involved, but not very well in situations of low usage. It is not good at predicting significant change, however, since the assumption of the method is that only gradual changes takes place over relatively long periods of time.

In a type II description, the computer system is viewed as the chequerboard for a game of competition between motivated individuals. This is the view taken in [3]. This type of analysis is designed to analyze the competitive processes which instigate significant change, at a more detailed level. It is good at determining the probability of success when using a set of strategies, and or finding the optimal strategies to solve a particular problem, but it is more difficult to apply than a type I description and relies on a knowledge or intuition of every relevant strategy which might be used by administrators and users alike.

Influences on the system can thus be classified as either random, stochastic or passive (type I), or as intentional, adversarial or strategic (type II), depending on the significance of the change. This distinction is partly artificial: all changes can clearly be traced back to the actions of humans at some level, but it is not always useful to do this. Not all users act in response to an important provocation, or with a specific aim in mind. It just happens that their actions lead to a general average degradation of the ideal state, no malice intended. Thus there is a part of the spectrum of changes which averages out to a kind of faceless background noise: the details of who did what are of no concern [7, 8].

From type I models, based on the empirical studies made at Oslo, we have found that computer systems behave remarkably like photonic gases in the limit of long times [16]. That is, the occurrence of events on computer systems mimics the behaviour of black body radiation in physics. The interpretation of any dynamical variable as a fluctuating, statistical quantity is made possible by considering the effect of infinitesimal perturbation to dynamical variables $q(t)$. One begins by defining averages and correlated products of the fields $q(t)$, with action $S[q]$. The action is a generating functional which determines the constraints on the behaviour of the dynamical variable $q(t)$ by a variational principle. For the simplest dynamical systems, one may write

$$S = \int dV_t \, \frac{1}{2} \, q(t) \hat{O} q(t).$$

The sum over all fluctuations of given latency may be written [4]:

$$\Gamma[< q(t) >] = - \ln \int_{\mathrm{TB}} d\mu \; e^{-S[<q>+\delta q]}.$$

The subscript TB refers stands for 'transaction bubbles' and refers to correlation graphs which are closed loops, i.e., complete transactions. This form is useful, since it is a self consistent form, which is derived on the assumption of linear statistical fluctuations an periodic time. It allows one to express self-consistent behaviour in terms of the measured variables alone. This quantity is essentially the free energy; it is a sum over all complete transactions in a fluctuating system and relies only on the assumed microscopic model which specifies available freedom and applied constraints. It can be calculated and compared to the fluctuation distributions measured for system variables.

Although the model is simple, the agreement with measured values is reasonable. The reason for this is a subtle but fascinating interaction between randomness and the order brought about by fixed daily and weekly rhythms. In fact, ensembles of events collected over weeks or months are insufficient in number to be perfectly described by these statistical methods, but the statistical model provides an idealized limit for computer behaviour, i.e., it provides a well defined envelope which approximates the system at scale of weeks. Moreover, it is so much simpler to understand than the actual behaviour of the system, that it has a valuable role to play in the discussion.

These studies have shown that computers behave like co-cyclic oscillators with periods of one day and one week (the rhythms imposed by the environment of users). Over periods of time which are long enough to gather enough data, these cyclic constraints reveal themselves as the shapes of distributions of events over time. They offer predictions about the statistical nature of the signal.

A type I model describes the average level of activity or *state* of the system which is related to the background noise. The second type of analysis which is required for a computer system is the analysis of non-cooperative user behaviour, i.e., analyzing which aspects of user behaviour affect the distribution of resources in the system. This analysis must be based on the *system policy*, since cooperation implies that the system is operating either within or outside the bounds of behaviour implied by the policy. Analysis must attempt to evaluate objectively the efficacy of different work patterns (strategies) employed by users in their interaction with the system.

A suitable framework for analyzing conflicts of interest, in a closed system, is the theory of games [17, 18]. Game theory is about introducing players, with goals and aims, into a scheme of rules and then analyzing how much a player can win, according to those restrictions. Each move in a game affords the player a characteristic value, often referred to as the 'payoff.' Game theory has been applied to warfare, to economics (commercial warfare) and many other situations. In this case, the game takes place on the n-dimensional board, spanned by the $\vec{d}$ vectors.

Resource management is a problem of economics, just as energy flows in physical systems are to do with the economics of energy. The difference in system administration is only that there is no a priori currency for describing the economics of system administration. It is necessary to invent one. In social and economical systems one has money as the book-keeping parameter for transactions. In physical systems, one has energy as the book-keeping parameter. These quantities count resources, in some well-defined sense.

There are several types or classifications of game. Some games are trivial: one-person games of chance, for example, are not analyzable in terms of strategies, since the actions of the player are irrelevant to the outcome. In a sense, these are related to the first kind of model referred to above. Some situations in system administration fit this scenario. More interesting, is the case in which the outcome of the game can be determined by a specific choice of strategy on the part of the players. The most basic model for such a game is that of a two-person zero-sum game, or a game in which there are two players, and where the losses of one player are the gains of the other.

One feature which distinguishes the analysis proposed here from pure game theory is that the value associated with different courses of action is not constant, but a function of time. The periodicities, discussed in the previous section must be taken into account as well as longer term changes, finite limits of system resources, non-linearities and so on. The implication of this is that the usefulness of a particular strategy varies according to when it is implemented.

The first kind of analysis assumes that the system has an average state and can therefore be used (at least in principle) to detect anomalous behaviour, e.g., behaviour which contravenes system policy. The second type of analysis looks at specific behavioural traits and attempts to evaluate their implications for the system state in more detail. Whether user behaviour lies within or outside the bounds of system policy is a matter of choice. Presumably one is interested in looking at all common behaviours, weighted by their likelihood in order to determine whether the system policy is effective enough. To make a type II theory realistic and tractable, one can imagine approximating the average background of the system activity using a type I model, and then studying specific strategies against this background. This leads to the notion of payoff, system currency in hybrid models.

Payoff in Type I and Type II Hybrids

Type I and type II models should not be should be thought of as completely separate issues: the best possible understanding of a computer system must involve both. Nonetheless it is primarily type II models which offer the chance to evaluate procedures and strategies of system administration. Type I models provide the background understanding of the resource behaviour, required to give substance to a type II model.

Equipped with a type I model for understanding the average interaction between user and system (which can be verified experimentally), one can construct a type II model in order to study a particular issue, against the average backdrop of type I activity. What is the outcome of introducing a new policy for governing a particular system resource, given what is known about how users generally interact with the system?

The determination of payoff, or the currency of the game is the central problem now. In order to find strategies which can keep the system close to its ideal state, one must assign a realistic value to strategies employed by users and system administrators. This is done by formulating a matrix (table) whose rows and columns specify the value or payoff associated with particular courses of action, for one of the players (see Figure 3). In the zero-sum approximation, it does not matter which player is chosen, since the losses of the one are the gains of the other. This is the only case to be considered here.

Courses of action available to each party, label the rows and columns. Rows are strategies and columns are counter-strategies, or vice versa. The values within the matrix are the values gained by one of the players, in units of the arbitrary currency of the game when a given row-strategy and column-strategy are chosen. These values are determined by policy and by information about how resources behave, acquired from type I models: they are a set of value judgements about what it important or unimportant in the system and to what degree.

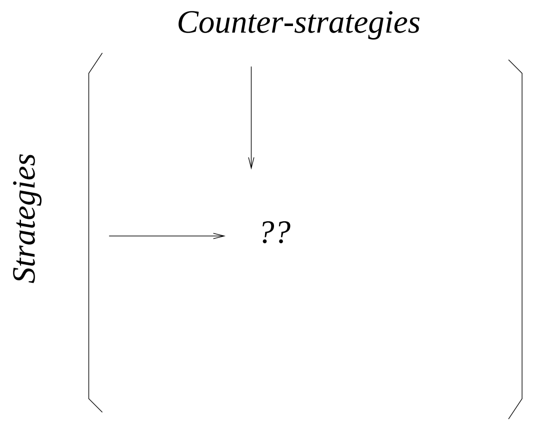

Figure 3: The payoff matrix is a table of strategies and counter strategies.

Once this 'payoff' matrix has been formulated, it contains information about the potential outcome of a game or scenario, using the strategies. This forms the basis for the theory of games [17, 18], whose methods and theorems make it possible to determine the optimal course or courses of action in order to maximize one's winnings. Obviously, any and all information which contributes to a judgement is useful, however one does not necessarily need a particularly detailed or accurate description to begin making simple value judgements about system behaviour. Even a simple quantification is useful, if it can distinguish between two possible courses of action.

How much can a user or an attacker hope to win? From our basic assumptions, the aim of a user is to maximize work produced or, in the worst case, maximize resources consumed. The system administrator, or embodiment of system policy, is not interested in winning the game for resources in the same way as users, but rather in confounding the game for users who gain too much control. The system administrator plays a similar role to that of a police force. In a vague sense, the administrator's jobs is to make sure that resources are distributed fairly, according to the policies laid down for the computer society (a Robin Hood role of altruistic government).

What is the currency of this evaluation? A definition is required in order to quantify the production of useful work by the system and its users. Clearly the term 'useful work' spans a wide variety of activities. Clearly work can increase and decrease (work can be lost through accidents), but this is not really germane to the problem at hand. The work generated by a user (physical and mental work and then computationally assisted results) is a function of the information input into the system by the user. Since the amount of computation resulting from a single input might be infinite, in practice, the function is an unknown.

In addition to the actual work produced by a user's strategy, other things might be deemed to be of value, such as privilege and status. In a community, wealth does not guarantee privilege or status unless that coincides with the politics of the community. Payoff can therefore be a complex issue to model. If one includes these ranking issues into calculations, one might allow for the possibility that a user plays the system rules in order to gain privileges for some later purpose. A user who accrues the goodwill of the system administrator, might eventually gain trust or even special privileges, such as extra disk space, access to restricted data etc. Such problems are of special interest in connection with security [19, 20].

For simplicity, the discussion of type II models in this paper refers only to games with two players. In a community, games are not necessarily two player zero sum engagements however. What is lost by one player is not necessarily gained by an obvious opponent. Moreover, the information available to different sides in a conflict can affect their modes of play. The so-called prisoner's dilemma, leads to the famous Nash equilibrium [21] which is a trade-off:

> A user of the system who pursues solely private interests, does not necessarily promote the best interest of the community as a whole.

Should users cooperate or fight to maximize their winnings? Users can sabotage their own self-interest by using up all the available resources on a finite system, gaining enemies or losing the goodwill of system police. Strategies which succeed in encouraging users to comply with guidelines can therefore be an effective way of ensuring a fair use of resources. The main reason for considering two person games here is the overriding simplicity of the two person game, compared to including more players. This should not be taken to imply that more complex models will not be important.

In a realistic situation one expects both parties in the two-person game to use a mixture of strategies. The number of possible strategies is huge and the scope for strategic contrivance is almost infinite. Strategies can be broken down into linear combinations of primitives just as any operation on the system can. What then is a strategy?

- An array of operations
- A schedule for the operations
- Rules for counter-moves or responses

In addition to simple strategies, there can be meta-strategies, or long-term goals. For instance, a nominal community strategy might be to:

- Maximize productivity or generation of work.
- Gain the largest feasible share of resources.

An attack strategy might be to

- Consume as many resources as possible.
- Destroy key resources.

Other strategies for attaining intermediate goals might include covert strategies such as *bluffing* (falsely naming files). Users can obey or ignore policy restrictions, use decoys, escalate or mitigate hostilities, attack/kill/delete a resource, retaliate. Defensive strategies might involve taking out an attacker, counter attacking, or evasion (concealment), exploitation, trickery, antagonization, incessant complaint (spam), revenge etc. Security and privilege, levels of access, integrity and trust must be woven into algebraic measures for the pay-off. One of the advantages of this formulation on system administration is that it places regular administration on the same footing as security issues. These were never separate issues and should not be considered as such, even in today's more security aware climate.

A means of expressing these devices must be formulated within a language which can be understood by system administrators, but which is primitive enough to enable the problem to be analyzed algebraically.

Example Games

The difficult part of a type II analysis is turning the high level concepts and aims listed above, into precise numerical values. To illustrate the procedure, consider an example of some importance, namely the filling of user disks. The need for forced garbage collection has been argued on several occasions [22, 5, 12], but the effectiveness of different strategies for avoiding disk may now be analyzed theoretically. This analysis is inspired by the user environment at Oslo University College, and the expressions derived here are designed to model this situation, not an arbitrary system.

The currency of this game must first be agreed upon. What value will be transferred from one player to the other in play? There are three relevant measurements to take into account: (i) the amount of resources consumed by the attacker (or freed by the defender); sociological rewards: (ii) 'goodwill' or (iii) 'privilege' which are conferred as a result of sticking to the policy rules. These latter rewards can most easily be combined into an effective variable 'satisfaction.' A 'satisfaction' measure is needed in order to set limits on individuals' rewards for cheating, or balance the situation in which the system administrator prevents users from using any resources at all. This is clearly not a defensible use of the system, thus the system defenses should be penalized for restricting users too much. The characteristic matrix now has two contributions,

$$\pi = \pi_r(\text{resources}) + \pi_s(\text{satisfaction}) .$$

It is convenient to define

$$\pi_r \equiv \pi(\text{resources}) = \frac{1}{2}\left(\frac{\text{Resources won}}{\text{Total resources}}\right) .$$

Satisfaction π_s is assigned arbitrarily on a scale from plus to minus one half, such that, Satisfaction π_s is assigned arbitrarily on a scale from plus to minus one half,

$$-\frac{1}{2} \le \pi_r \le +\frac{1}{2}$$
$$-\frac{1}{2} \le \pi_s \le +\frac{1}{2}$$
$$-1 \le \pi \le +1 .$$

The pay-off is related to the movements made through the lattice $\vec{d}$. The different strategies can now be regarded as duels, or games of timing; see Table 2. These elements of the characteristic matrix must now be filled, using a model and a policy. A general expression for the rate at which users produce files is approximated by:

$$r_u = \frac{n_b r_b + n_g r_g}{n_b + n_g} ,$$

where r_b is the rate at which bad users (i.e., problem users) produce files, and r_g is the rate for good users. The total number of users $n_u = n_b + n_g$. From experience, the ratio $\frac{n_b}{n_g}$ is about one percent. The rate can be expressed as a scaled number between zero and one, for convenience, so that $r_b = 1 - r_g$.

The payoff in terms of the consumption of resources by users, to the users themselves, can then be modelled as a gradually accumulation of files, in daily waves, which are a maximum around midday:

$$\pi_u = \frac{1}{2}\int_0^T dt\, \frac{r_u(\sin(2\pi t/24) + 1)}{R_{\text{tot}}} ,$$

where the factor of 24 is the human daily rhythm, measured in hours, and R_{tot} is the total amount of

Users/System	Ask to tidy	Tidy by date	Tidy above Threshold	Quotas
Tidy when asked	$\pi(1,1)$	$\pi(1,2)$	$\pi(1,3)$	$\pi(1,4)$
Never tidy	$\pi(2,1)$	$\pi(2,2)$	$\pi(2,3)$	$\pi(2,4)$
Conceal files	$\pi(3,1)$	$\pi(3,2)$	$\pi(3,3)$	$\pi(3,4)$
Change timestamps	$\pi(4,1)$	$\pi(4,2)$	$\pi(4,3)$	$\pi(4,4)$

Table 2: Games of timing.

resources to be consumed. Note that, by considering only good user or bad users, one has a corresponding expression for π_g and π_b, with r_u replaced by r_g or r_b respectively. An automatic garbage collection system (cfengine) results in a negative pay-off to users, i.e., a pay-off to the system administrator. This may be written

$$\pi_a = \frac{1}{2}\int_0^T dt\, \frac{r_a(\sin(2\pi t/T_p) + 1)}{R_{\rm tot}} ,$$

where T_p is the period of execution for the automatic system (in our case, cfengine). This is typically hourly or more often, so the frequency of the automatic cycle is some twenty times greater than that of the human cycle. The rate of resource-freeing r_a is also greater than r_u, since file deletion takes little time compared to file creation, and also an automated system will be faster than a human. The quota payoff yields a fixed allocation of resources, which are assumed to be distributed equally amongst users and thus each quota slice assumed to be unavailable to other users. The users are nonchalant, so $\pi_s = 0$ here, but the quota yields

$$\pi_q = +\,\frac{1}{2}\left(\frac{1}{n_b + n_g}\right) .$$

The matrix elements are expressed in terms of these.

$\pi(1,1)$: Here $\pi_s = -\,1/2$ since the system administrator is as satisfied as possible by the users' behaviour. π_r is the rate of file creation by good users π_g, i.e., only legal files are produced. Comparing the strategies, it is clear that $\pi(1,1) = \pi(1,2) = \pi(1,3)$.

$\pi(1,4)$: Here $\pi_s = 0$ reflecting the users' dissatisfaction with the quotas, but the system administrator is penalized for restricting the freedom of the users. With fixed quotas, users cannot generate large temporary files. π_q is the fixed quota payoff, a fair slice of the resources. Clearly $\pi(4,1) = \pi(4,2) = \pi(4,3)= \pi(4,4)$. The game has a fixed value if this strategy is adopted by system administrators. However, it does not mean that this is the best strategy, according to the rules of the game, since the system administrator loses points for restrictive practices, which are not in the best interest of the organization. This is yet to be determined.

$\pi(2,1)$: Here $\pi_s = 1/2$ since the system administrator is maximally dissatisfied with users' refusal to tidy their files. The pay-off for users is also maximal in taking control of resources, since the system administrator does nothing to prevent this, thus $\pi_r = \pi_u$. Examining the strategies, one finds that $\pi(2,1) = \pi(3,1) = \pi(3,2) = \pi(3,3) = \pi(4,1) = \pi(4,2)$.

$\pi(2,2)$: Here $\pi_s = 1/2$ since the system administrator is maximally dissatisfied with users' refusal to tidy their files. The pay-off for users is now mitigated by the action of the automatic system which works in competition, thus $\pi_r = \pi_u - \pi_a$. The automatic system is invalidated by user bluffing (file concealment).

$\pi(2,3)$: Here $\pi_s = 1/2$ since the system administrator is maximally dissatisfied with users' refusal to tidy their files. The pay-off for users is mitigated by the automatic system, but this does not activate until some threshold time is reached, i.e., until $t > t_0$. Since changing the date cannot conceal files from the automatic system, when they are tidied above threshold, we have $\pi(2,3) = \pi(4,3)$.

Thus, in summary, the characteristic matrix is given by Formula 1 where the step function is defined by,

$$\Theta(t_0 - t) = \begin{cases} 1 & (t \geq t_{0)} \\ 0 & (t < t_{0)} \end{cases} ,$$

and represents the time-delay in starting the automatic tidying system in the case of tidy-above-threshold. This was explained in more detail in [3].

It is possible to say several things about the relative sizes of these contributions. The automatic system works at least as fast as any human so, by design, in this simple model we have

$$\frac{1}{2} \geq |\pi_a| \geq |\pi_u| \geq |pi_g| \geq 0 ,$$

for all times. For short times $\pi_q > \pi_u$, but users can quickly fill their quota and overtake this. In a zero-sum game, the automatic system can never tidy garbage faster than users can create it, so the first inequality is always saturated. From the nature of the cumulative pay-offs, we can also say that

$$(\frac{1}{2} + \pi_u) \geq (\frac{1}{2} + \pi_u + \pi_a\Theta(t_0 - t)) \geq (\frac{1}{2} + \pi_u + \pi_a) ,$$

and

$$|\,\frac{1}{2} + \pi_u| \geq |\pi_g - \frac{1}{2}\,| .$$

Applying these results to a modest strategy of automatic tidying, of garbage, referring to Figure 4, one sees that the automatic system can always match users' moves. As drawn, the daily ripples of the automatic system are in phase with the users' activity. This is not realistic, since tidying would normally be done

$$\pi(u,s) = \begin{pmatrix} -1/2 + \pi_g(t) & -1/2 + \pi_g(t) & -1/2 + \pi_g(t) & \pi_q \\ 1/2 + \pi_u(t) & 1/2 + \pi_u(t) + \pi_a(t) & 1/2 + \pi_u(t) + \pi_a(t)\Theta(t_0 - t) & \pi_q \\ 1/2 + \pi_u(t) & 1/2 + \pi_u(t) & 1/2 + \pi_u(t) & \pi_q \\ 1/2 + \pi_u(t) & 1/2 + \pi_u(t) & 1/2 + \pi_u(t) + \pi_a(t)\Theta(t_0 - t) & \pi_q \end{pmatrix}$$

Formula 1: Characteristic matrix.

at night when user activity is low, however such details need not concern us in this illustrative example.

The policy created in setting up the rules of play for the game, penalizes the system administrator for employing strict quotas which restrict their activities. Even so, users do not gain much from this, because quotas are constant for all time. A quota is a severe handicap to users in the game, except for very short times before users reach their quota limits. Quotas could be considered cheating by the system administrator, since they determine the final outcome even before play commences. There is no longer an adaptive allocation of resources. Users cannot create temporary files which exceed these hard and fast quotas. An immunity-type model which allows fluctuations is a more resource efficient strategy in this respect, since it allows users to span all the available resources for short periods of time, without consuming them for ever.

According to the *minimax* theorem, proven by John Von Neumann, any two-person zero-sum game has a solution, either in terms of a pair of optimal *pure* strategies or as a pair of optimal *mixed* strategies [17, 18]. The solution is found as the balance between one player's attempt to maximize his pay-off and the other player's attempt to minimize the opponent's result. In general one can say of the pay-off matrix that

$$\max_{\downarrow}\min_{\rightarrow}\pi_{rc} \le \min_{\leftarrow}\max_{\uparrow}\pi_{rc} \ ,$$

where the arrows refer to the directions of increasing rows ($\downarrow$) and columns ($\rightarrow$). The left hand side is the least users can hope to win (or conversely the most that the system administrator can hope to keep) and the right is the most users can hope to win (or conversely the least the system admin can hope to keep). If we have Equation 2,

$$\max_{\downarrow}\min_{\rightarrow}\pi_{rc} = \min_{\leftarrow}\max_{\uparrow}\pi_{rc}$$

Equation 2: Equality in the payoff matrix.

it implies the existence of a pair of single, pure strategies (r^*, c^*) which are optimal for both players, regardless of what the other does. If the equality is not satisfied, then the minimax theorem tells us that there exist optimal mixtures of strategies, where each player selects at random from a number of pure strategies with a certain probability weight.

The situation for our time-dependent example matrix is different for small t and for large t. The distinction depends on whether users have had time to exceed fixed quotas or not; thus 'small t' refers to times when users are not impeded by the imposition of quotas. For small t, one has:

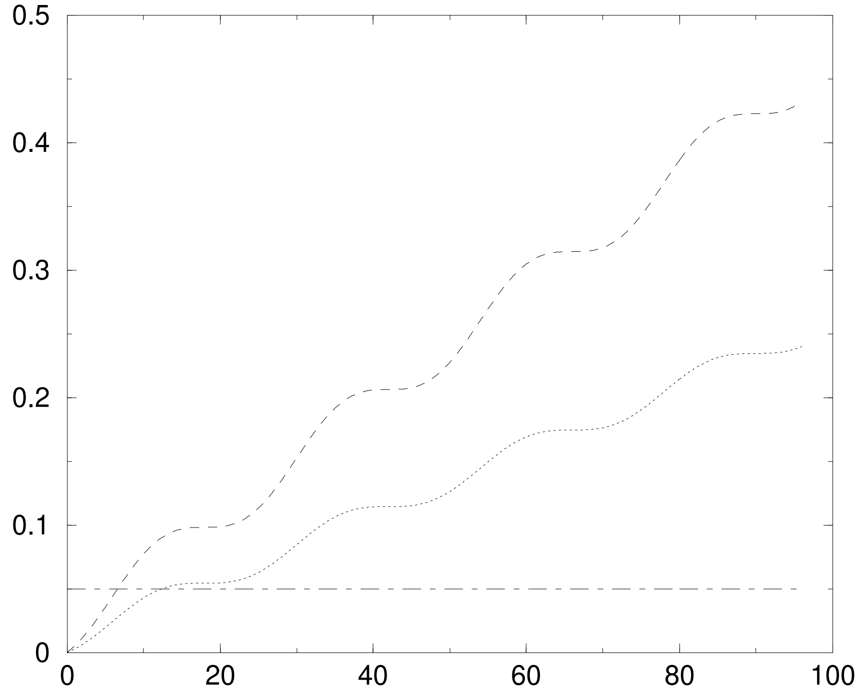

Figure 4: The absolute values of pay-off contributions as a function of time (in hours), For daily tidying $T_p = 24$. User numbers are set in the ratio $(n_g, n_b) = (99, 1)$, based on rough ratios from the author's College environment, i.e., one percent of users are considered mischievous. The filling rates are in the same ratio: $r_b/R_{\rm tot} = 0.99$, $r_g/R_{\rm tot} = 0.01, r_a/R_{\rm tot} = 0.1$. The flat dot-slashed line is $|\pi_q|$, the quota pay-off. The lower wavy line is the cumulative pay-off resulting from good users, while the upper line represents the pay-off from bad users. The upper line doubles as the magnitude of the pay-off $|\pi_a| \ge |\pi_u|$, if we apply the restriction that an automatic system can never win back more than users have already taken. Without this restriction, $|\pi_a|$ would be steeper.

$$
\max_{\downarrow} \min_{\rightarrow} \pi_{rc} = \max_{\downarrow} \begin{pmatrix} \pi_g - \frac{1}{2} \\ \frac{1}{2} + \pi_u + \pi_a \\ \frac{1}{2} + \pi_u \\ \frac{1}{2} + \pi_u + \pi_a \Theta(t_0 - t) \end{pmatrix} = \frac{1}{2} + \pi_u \ .
$$

The ordering of sizes in the above minimum vector is:

$$
\frac{1}{2} + \pi_u \geq \frac{1}{2} + \pi_u + \pi_a \Theta(t_0 - t) \geq \pi_u + \pi_a \Theta(t_0 - t) \geq \pi_g - \frac{1}{2}
$$

For the opponent's endeavours one has

$$
\min_{\rightarrow} \max_{\downarrow} \pi_{rc} = \min_{\rightarrow}(\frac{1}{2} + \pi_u, \frac{1}{2} + \pi_u, \frac{1}{2} + \pi_u, \pi_q) = \frac{1}{2} \pi_u \ .
$$

This indicates that the equality in Equation 2 is satisfied and there exists at least one pair of pure strategies which is optimal for both players. In this case, the pair is for users to conceal files, regardless of how the system administrator tidies files (the sysadm's strategies all contribute the same weight in Equation 2. Thus for small times, the users are always winning the game if one assumes that they are allowed to bluff by concealment. If the possibility of concealment or bluffing is removed (perhaps through an improved technology), then the next best strategy is for users to bluff by changing the date, assuming that the tidying looks at the date. In that case, the best system administrator strategy is to tidy indiscriminately at threshold.

For large times (when system resources are becoming or have become scarce), then the situation looks different. In this case one finds that

$$
\max_{\downarrow} \min_{\rightarrow} \pi_{(rc)} = \min_{\rightarrow} \max_{\downarrow} \pi_{rc} = \pi_q \ .
$$

In other words, the quota solution determines the outcome of the game for any user strategy. As already commented, this might be considered cheating or poor use of resources, at the very least. If one eliminates quotas from the game, then the results for small times hold also at large times.

This simple example of system administration as a strategic game between users and administrators is only an illustration of the principles involved in building a type I/II hybrid model. In spite of its simplicity, it is already clear that user bluffing and system quotas are strategies which are to be avoided in an efficient system. The value of 'goodwill' in curbing anti-social behaviour should not be underestimated. By following this basic plan, it should be possible to analyze more complex situations in future work.

Future Work

From the type I models studied at Oslo [4, 7], it appears that the most important characteristic of the average user behaviour is its periodicity: the average state of computers has a daily period and a weekly period; these trace the social cycles of users all around the world. It is possible, as more is learned, that more detailed characteristics will emerge which are general enough to be used in a type I/type II hybrid. The main promise of type I theories lies in the possibility of anomaly detection and self-analysis, leading to fault detection, intrusion detection and improvements in immune system technology at the user level (e.g., cfengine). However, it is also important to know, for strategic analysis, when the system is most loaded, must vulnerable and most available.

Only one example of a type II theory has been examined here. What other issues might be studied by a type II model? The possibilities include strategies such as: consolidation versus distribution in system planning (where should resources be located?); delegation vs centralization; choosing many simple tools or a few powerful ones [23] (cost of learning and support, functionality, likelihood of bugs, results, rate of evolution of task and tools); the effect of system work ethics on productivity in a business (does the business spend most of its time working against itself or its competitors?); is the best strategy one which leads to stability or perfection? Mission critical systems and high security systems are obvious candidates for analysis. Other resources uses: network share, processes, setting of permissions, placement of security etc. The possibilities are limited only by the imagination. The benefit of the type II model is in setting up a systematic method for making impartial judgements about strategies for system management and system regulation.

A common theme in all strategic studies, involving complex competitive behaviour, is the so-called Red Queen scenario. This is about working hard to maintain the status quo; it is a reference to a scene from Alice Through The Looking Glass:

> *"Well in our country," said Alice, still panting a little, "you'd generally get to somewhere else – if you ran very fast for a long time as we've been doing."*
> *"A slow sort of country!" said the Queen. "Now here, you see, it takes all the running you can do, to keep in the same place. If you want to get somewhere else, you must run twice as fast as that!"*

This is also referred to as an arms race. In a true dynamical system, nothing stands still. Adapting one's strategies to be optimal over time means a continual reappraisal of their efficacy. One has to be running all the time to keep up with the environment, continually adapting to change. The need for garbage collection is an example of this.

In choosing strategies which walk the line between compliance and conflict, within a framework of rules (the prisoner's dilemma), some studies have indicated that the best solutions are often cooperation at first, and then *tit for tat* after that, if cooperation does not work: i.e., try cooperation first to mitigate hostilities, and then send a message that one means business thereafter.

Sketching out this recipe for analyzing system administration policies reveals some potent ideas, which have a merit quite independently of their analytical value. The idea of using mixtures of strategies to most efficiently regulate a system, is so close to the ideas used in cfengine [5, 12] as to suggest that they could be adopted more even explicitly, and more dynamically. Rather than relying on batch operation, a policy engine like cfengine could be more dynamical in its responses to deviations from ideal state, and be able to set in motion a variety of parallel responses, which might extend its reach in dealing with more dynamical problems like network intrusions. It could also respond to more long-term trends in system usage and adapt its behaviour accordingly. Part of the motivation of this work was precisely to see what could be done to improve on cfengine [24]. Once refined, the approach in this paper will lead to improvements in cfengine, and improve the automation of host security.

Summary

The aim of this paper has been to propose a framework for analyzing models of system administration. Its main contention is that it is possible to see system administration as the effort to keep the system close to an ideal state, by introducing countermeasures in the face of competitive resource consumption. This is the formal basis which opens the way for objective analyses in the field.

With a mathematical approach, it becomes easier to see through personal opinions and vested interests when assumptions and methods are clearly and rigorously appraised. However, one can only distinguish between those possibilities which are taken into account. That means that every relevant strategy, or alternative, has to be considered. This is the limitation of game theory. It is not possible to determine strategies without the creative input of experts, and a clearly described policy.

Appealing only to a simple-minded analysis of disk filling, some straightforward conclusions are possible: the use of quotas is an inefficient way of counteracting the effects of selfish users, when the whole community's interests are taken into account. A quota strategy can never approach the same level of productivity as one which is based on competitive counterforce. The optimal strategies for garbage collection are rather found to lie in the realm of the immunity model [6, 15]. However, it is a sobering thought that a persistent user, who is able to bluff the immune system into disregarding it, (like a cancer) will always win against the resource battle. The need for new technologies which can see through bluffs will be an ever present reality in the future. With the ability of encryption and compression systems to obscure file contents, this is a contest which will not be easily won by system administrators.

Author Information

Mark Burgess is an associate professor of physics and computer science at Oslo University College, creator of cfengine and author of the book *Principles of Network and System Administration*. He may be reached at mark@iu.hio.no or http://www.iu.hio.no/~mark. Cfengine can be obtained from http://www.iu.hio.no/cfengine. Oslo University College's research pages for system administration are at http://www.iu.hio.no/SystemAdmin .

References

[1] R. Evard, "An Analysis of Unix System Configuration," *Proceedings of the 11th Systems Administration Conference (LISA)*, page 179, 1997.

[2] S. Traugott and J. Huddleston, "Bootstrapping an Infrastructure," *Proceedings of the 12th Systems Administration Conference (LISA)*, page 181, 1998.

[3] M. Burgess, "On the theory of system administration," Submitted to the *Journal of the ACM*, 2000.

[4] M. Burgess, "Information theory and the kinematics of distributed computing, submitted to *Physical Review E*, 2000.

[5] M. Burgess, "A site configuration engine," *Computing systems*, 8:309, 1995.

[6] M. Burgess, "Computer immunology," *Proceedings of the 12th Systems Administration Conference (LISA)*, page 283, Usenix Association, 1998.

[7] M. Burgess, H. Haugerud, and S. Straumsnes, "Measuring Host Normality, I," submitted to *Software Practice and Experience*, 1999.

[8] M. Burgess and Trond Reitan, "Measuring Host Normality, II," submitted to *Software Practice and Experience*, 1999.

[9] N. Glance, T. Hogg, and B. A. Huberman, "Computational Ecosystems in a Changing Environment," *International Journal of Modern Physics*, C**2**:735, 1991.

[10] E. D. Zwicky, S. Simmons, and R. Dalton, "Policy as a system administration tool," *Proceedings of the Fourth Systems Administration Conference (LISA)*, SAGE/USENIX, page 115, 1990.

[11] B. Howell and B. Satdeva, "We have met the enemy: An Informal Survey of Policy Practices in the Internetworked Community," *Proceedings of the fifth systems administration conference (LISA),* SAGE/USENIX, page 159, 1991.

[12] M. Burgess and R. Ralston, "Distributed resource administration using cfengine," *Software practice and experience*, 27:1083, 1997.

[13] M. Burgess, "Automated system administration with feedback regulation," *Software Practice and Experience*, 28:1519, 1998.

[14] F. Reif, *Fundamentals of Statistical Mechanics*, McGraw-Hill, Singapore, 1965.

[15] M. Burgess, *Principles of Network and System Administration*, J. Wiley & Sons, Chichester, 2000.

[16] M. Burgess, "Thermal, non-equilibrium phase space for networked computers," *Physical Review*, E62:(in press), 2000.

[17] J. V. Neumann and O. Morgenstern, *Theory of games and economic behaviour*. Princeton University Press, Princeton, 1944.

[18], M. Dresher, *The mathematics of games of strategy*, Dover, New York, 1961.

[19] V. Jones and D. Schrodel, "Balancing security and convenience," *Proceedings of the First Systems Administration Conference (LISA),* (SAGE/USENIX), page 5, 1987.

[20] I. S. Winkler and B. Dealy, "Information Security Technology? Don't Rely On It. A Case Study in Social Engineering," *Proceedings of the 5th USENIX Security Symposium*, page 1, 1995.

[21] J. F. Nash. *Essays on Game Theory*, Edward Elgar, Cheltenham, 1996.

[22] E. D. Zwicky, "Disk space management without quotas," *Proceedings of the third systems administration conference (LISA)*, (SAGE/USENIX), page 41, 1989.

[23] H. E. Harrison, "Maintaining a consistent software environment," *Proceedings of the First Systems Administration Conference (LISA),* (SAGE/USENIX), page 16, 1987.

[24] M. Burgess, "Evaluating cfengine's Immunity Model of System Maintenance, *Proceedings of USENIX/SANE 2000, Netherlands*, 2000.