

Scaling behaviour of peer configuration management in logically ad hoc networks

Mark Burgess IEEE Member
Geoffrey Canright

Abstract— Current interest in *ad hoc* and *peer-to-peer* networking technologies prompts a re-examination of models for configuration management, within these frameworks. In the future, network management methods may have to scale to millions of nodes within a single organization, with complex social constraints. In this paper, we discuss whether it is possible to manage the configuration of large numbers of network devices using well-known and not-so-well-known configuration models, and we discuss how the special characteristics of ad hoc and peer-to-peer networks are reflected in this problem.

Keywords— **Configuration management, ad hoc networks, peer to peer.**

I. INTRODUCTION

Policy based configuration management deals with the site-wide specification of device or host behaviour in a network. The configuration of a host determines its efficiency, its correctness in performing a function and its security in risky situations and is usually specified as a file or a database of file attributes and process characteristics. Policy includes not only permissions and usage patterns but also software customizations and the answers to any other questions that are of an arbitrary nature, i.e. those that cannot be fixed by general principles like optimization or self-consistency.

Various approaches have been devised for configuration management over the last thirty years. For instance, the IETF model of configuration management revolves traditionally around the Simple Network Management Protocol (SNMP)[12]. This is read/write state based protocol for altering values in a management information database (MIB), and is used by a number of commercial software products. The ‘Telecommunications Management Network’ or TMN[20] is an alternative scheme designed for telecommunications networks and has a strong relationship with the OSI management model. These systems use an abstraction based on the concept of ‘managed objects’. An different approach is used by systems like cfengine[4] and PIKT[21], which use descriptive languages to describe the attributes of many objects at the same time, and agents to enforce the rules.

Since traditionally many configuration management schemes are based on monitoring and instruction, steered from a central location, the emphasis in management has been on developing communications protocols. The ability to send or receive messages is crucial to configuration management of network devices and hosts, even for man-

agement schemes that are not reliant on a network for their operation. Every configuration scheme has to compare an existing state with a template state, and therefore must essentially link these two with a channel for observation and instruction. Indeed, maintaining the configuration of hosts over time has many features in common with the problem of information transmission over a noisy channel[5]. The probability P_c that a receiver will be correctly configured at time t is proportional to the probability of error-free transmission from source to receiver.

Today, distributed systems are geographically global and are linked, both conceptually and physically by a network infra-structure; but that infra-structure varies enormously, from the fixed line to the wireless or ad hoc link. Passing messages from one part of a system to another is subject to a plethora of uncertainties. SNMP uses an unreliable transport protocol UDP for communication and any configuration scheme that relies on the availability of a resource or component at a specific moment has only a limited chance of being carried out. Hosts and devices themselves can be unavailable due to power failures, physical breakages, absence of dependencies and so on. There is thus an ad hoc element to network connectivity even in an ostensibly permanent infra-structure. The additional complication of mobile services, with partial or intermittent connectivity adds to this problem.

II. LOGICAL NETWORKS

Physical infrastructure of a network has dominated research for a long time, but it is only one aspect of configuration management. The multi-layered structure of network services, usually classified by the OSI model, means that many protocol layers have to work in concert to guarantee communication. Logical as well as physical layers can fail to perform. Logical layers are implemented by ‘host services’, while the physical transport is usually referred to as ‘network services’. Both of these should be included in a model that purports to deal with site configuration management. Finally, one need not be reliant on host to host communication at all at a given moment. One must allow for caching of policy and distributed action, so that the availability of a management (maintenance) system need not be directly related to the OSI layers of communication, but can also take place over internal buses or hidden pathways not normally considered as explicit communications channels.

To address these issue we choose to view all communications channels in this apper as being unreliable or prob-

M. Burgess is at Oslo University College, Norway. E-mail: Mark.Burgess@iu.hio.no .

G. Canright is at Telenor Research, Fornebu, Oslo, Norway. E-mail: Geoffrey.Canright@telenor.com

abilistic networks. We can then identify a framework that is general enough to make algebraic comparisons between different kinds of network, using the notion of ‘ad hoc’ connectivity.

An ‘ad hoc’ network (AHN) is usually defined to be a networked collection of mobile hosts, each of which has the possibility to route information. The union of those hosts forms an arbitrary graph that changes with time. The nodes are free to move randomly thus the network topology may change rapidly and unpredictably. Ad hoc networks are important in a mobile computing environment, where hosts are partially or intermittently connected to other hosts, but they can also be used to discuss fixed infrastructure service networks with partial reliability since a fixed location host is clearly just a special case of a mobile host. We therefore generalize the term ad hoc network in this paper.

The ability to include mobility in our discussion is very interesting. While there has been some discussion of decentralized network management using mobile agents[23], the problem of mobile nodes (and so strongly time-varying topology) has received little attention. However, we posit that ad-hoc networks provide a useful framework for discussing the problems surrounding configuration management in all network types, both fixed and mobile, because logical networks can be ad hoc even if their infrastructure is permanent. Note finally that we are not referring to the notion of ‘ad hoc management’[19], which concerns randomly motivated and scheduled checks of the hosts.

The plan for our paper is as follows. We begin by outlining how reliability can be discussed in terms of ad hoc connectivity in order to take advantage of its known scaling properties. Then noting how peer to peer communication implies decentralized policy, we estimate the required flow of configuration information as a function of the number of hosts, for a number of management models, in order to determine their scalability.

III. AVAILABILITY OF PEERS IN A NETWORK

We begin with an abstract idea: that the correct configuration of a system over time depends on the system having regular maintenance checks. This is the essence of the maintenance theorem in ref. [6] which notes that human-computer systems are stochastic in nature and subject to random errors in configuration. Such errors occur due to human interventions, automatic updates, software failures and even low probability bugs.

In order to effect maintenance, a host must be able to read its policy specification. The probability that a host will be correctly configured is thus related to reliability of its ‘communication’ with a policy source. In some cases, policy will be cached locally but updates will still require hosts to communicate with trusted sources located elsewhere on the network. As long as this involves networked communication, we need to characterize the reliability of the network in order to discuss configuration management. As a simplest case, we assume that the reliability of each node and each link is independent of all others, so that

the probabilities of availability are all independent random variables. In general this is not true, since some hosts/nodes depend on others for crucial services (e.g. the domain name service (DNS)), but this should suffice to gauge orders of magnitude.

Definition 1: A set of nodes or hosts is defined by a vector of probabilities $\vec{h}^T = (p_1, p_2, \dots, p_N)$, where p_i ($i = 1 \dots N$) is the probability that node i is available. If the probabilities are 1, the hosts are said to be reliable, otherwise they are partially reliable.

The nodes themselves may have any geographical location, and may be connected by any means. The connectivity between the nodes is represented by a matrix.

Definition 2: A network is defined by its adjacency matrix. By convention, the adjacency matrix of a network or graph is a symmetric matrix with zero leading diagonal.

$$A(1) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (1)$$

Zeros denote no connectivity, while a 1 means a connection. The notation $A(1)$ distinguishes this (instantaneous) matrix, whose entries are binary-valued, from the time-averaged matrix discussed below. Owing to access and routing controls, this matrix need not be symmetrical in practice, but we shall not address that issue here.

The properties of networks can be discussed in detail, using the adjacency matrix representation (see for instance, ref. [22]). It is not our intention to go into excessive detail here, but rather to distill a way of estimating the properties of networks. For this, we choose to look at the average properties of the networks.

We define a simple measure of the availability of a service, transmitted within a closed network, by an invariant scalar value χ :

Definition 3: The *connectivity*, χ , of a network \mathcal{N} , is the probability (averaged over all pairs of nodes) that a message can be passed directly between any two nodes. χ may be written as

$$\chi = \frac{1}{N(N-1)} \vec{h}^T A \vec{h}. \quad (2)$$

χ has a maximum value of 1, when every node is connected to every other, and a minimum value of zero when all nodes are disconnected.

For a fixed topology and time-independent node availabilities, χ is a constant characterizing the network. In general χ is time-dependent; one then obtains a static figure for the network by taking the long-time average.

$$\langle \chi \rangle = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n \chi(t_i). \quad (3)$$

The utility of this measure is that it enables us to gauge and compare different network configurations on equal terms. It is also the vehicle by which we can map the problem of unreliable hosts in a fixed network onto a corresponding problem of reliable hosts in an ad hoc network.

IV. AD HOC NETWORKS

Ad hoc networks are networks whose adjacency matrices are subject to a strong, apparently random time variation. If we look at the average adjacency matrices, over time, then we can represent the probability of connectivity, between pairs of hosts in the network, as an adjacency matrix of probabilities. An ad hoc network can be represented as a random, square matrix, in which the rows and columns label the nodes in a time-varying graph, representing the network. In a partially reliable network, one has only a finite probability for successful communication, thus it too can be represented in this form.

In a general ad hoc network, with a fixed number of nodes, the links are not independent variables. They are constrained both by the physical geography in which the nodes move (only nodes that are close enough are candidates for linking), and by interference effects among the set of nodes near to a given node. Any given node thus may or may not establish a working link with a near node, depending on interference from other near nodes. For our purposes here, these dependencies are not important; the important property of the ad-hoc net is the intermittency of the links, due to the nodes' mobility.

Definition 4: An ad hoc network is represented by a symmetric matrix of probabilities for adjacency. Thus the time average of the adjacency matrix (for, e.g., four nodes) may be written as

$$\langle A \rangle = \begin{pmatrix} 0 & p_{12} & p_{13} & p_{14} \\ p_{21} & 0 & p_{23} & p_{24} \\ p_{31} & p_{32} & 0 & p_{34} \\ p_{41} & p_{42} & p_{43} & 0 \end{pmatrix} \quad (4)$$

An ad hoc network is therefore a partially reliable network. To motivate our discussion further, we note that:

Theorem 1: A fixed network of partially-reliable nodes, h_i , can be described as an ad hoc network of reliable nodes, on average.

Proof: This is easily seen from the definition of the connectivity, using a matrix component form:

$$\begin{aligned} N(N-1)\langle \chi \rangle &= \sum_{ij} h_i(p_i) \langle A_{ij}(1) \rangle h_j(p_j) \\ &= \sum_{ij} h_i(1) \langle A_{ij}(p_i p_j) \rangle h_j(1). \end{aligned} \quad (5)$$

The proof demonstrates the fact that one can move the probabilities (uncertainties) for availability from the host vectors to the connectivity matrix; for example

$$\begin{aligned} &\begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}^T \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}^T \begin{pmatrix} 0 & p_1 p_2 & p_1 p_3 \\ p_2 p_1 & 0 & p_2 p_3 \\ p_3 p_1 & p_3 p_2 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}. \end{aligned}$$

Thus an array of hosts with reliability probabilities p_i , is equivalent to an array of reliable hosts in an unreliable network, where the probability of communication between them is the product of probabilities (assumed independent) from the reliability vector.

Superposed onto the routing problem is another problem of conceptual dependence. One is not merely dependent on connectivity to provide a route for messages, but one depends on trusted sources of information. Thus the arrows from source to receiver are not merely bytes exchanged but authorized policy instructions. We shall consider this issue below.

V. PEER TO PEER

The emergence of network file sharing applications such as Napster, Gnutella, has focused attention on an architecture known as peer-to-peer, whose aim is to provide worldwide access to information via a highly de-centralized network of 'peers'. Although, technically, peers use 'client-server' sockets to communicate, both functions are supported by every node thus one does not distinguish the logical status of hosts into clients and servers. An important challenge to providing a fully distributed information sharing system is the design of scalable algorithmic solutions. Algorithms such as those for routing and searching peer-to-peer networks are typically implemented in a form of an application-level protocol.

Definition 5: A peer to peer network service is one in which each node, at its own option, participates in or abstains from exchanging data with other nodes, over a communications channel.

Peer to peer has a deeper significance than ad hoc file sharing. It is about the demotion of a central authority, in response to the political wishes of those participating in the network. This is an issue directly analogous to that of determining policies used for configuration management. In large organizations, i.e. large networks, we see a frequent dichotomy of interest:

- At the high level, one has specialized individuals who can paint policy in broad strokes, dealing with global issues such as software versions, common security issues, organizational resource management, and so on. Such issues can be made by software producers, system managers and network managers.
- At the local level, users are more specialized and have particular needs, which large scale managers cannot address. Centralized control is therefore only a partial strategy for success. It must be supplemented by local know-how, in response to local environmental issues. Managers at the level of centralized control have no knowledge of the needs of specialized groups, such as the physics department of a university, or the research department of a company. In terms of configuration policy, what is needed is the ability to accept the advice of higher authorities, but to disregard it where it fails to meet the needs of the local environment. This kind of authority delegation is not catered for by SNMP-like models. Policy based management attempts to rectify some of these issues[14].

What we find then is that there is another kind of networking going on: a social network, superimposed onto the technological one. The needs of small clusters of users override the broader strokes painted by wide area management. This is the need for a scaled approach to system management[6].

VI. CONFIGURATION MANAGEMENT IN AD HOC NETWORKS

Configuration management deals with the problem of establishing and maintaining a policy conformant configuration on workstations and other hosts distributed around a network. Policy is usually a set of rules and specifications about the software and resources of each host, defined perhaps by a central authority and disseminated to the individual hosts either on demand, or by common update.

Configuration management relies on two main things: i) the availability of trusted resources to each networked host, including a policy P , and ii) the consistency of the configuration specified by that policy. In an unpredictable environment one has potentially several problems: Critical dependencies, including the policy itself, can become unavailable or out of date; trust relationships are less certain if hosts cannot verify one another's' identity, location or integrity. Thus security and verifiable control, within specified time limits, are at stake.

Even in a fixed infrastructure network, with only partial connectivity, the availability of the resources is open to uncertainty. This means that the ability to correctly disseminate policy configuration is open to uncertainty. The framework of ad hoc networks thus encompasses a number of issues and offers a basis for discussing configuration strategies in general. In recent times, there has been a move towards self-configuring networks. Discovery protocols like JINI have to deal with the ad hoc nature of networks, and the protocols themselves will need to take the uncertainties in topology into account. Today, most protocols assume a fixed infra-structure.

One question that has been posed in this connection is whether a peer to peer strategy, for disseminating configuration policy, could provide a way of spreading information quickly about the network. If that were the case, then the temporary unavailability of a node to a central resource would not necessarily imply its isolation from fresh, critical data. This kind of data distribution has been discussed before[13] in connection with the scalability of software distribution. On the down side, peer to peer reliance is clearly an open invitation to engage in denial of service activity.

VII. PREDICTABILITY AND SCALING

As networks grow, some configuration strategies do not scale well. They continue to be used, however, by force of habit. We are interested in examining the scaling properties of different configuration management schemes, especially in the context of network models that look to the future of configuration management.

We consider a number of cases, in order of decreasing centralization, or increasing delegation. One could imagine

that this ordering also corresponds, roughly, to decreasing level of certainty about the network. However this interpretation may be misleading, since even centralized control schemes (at the start of our list) are also prone to noise, and local or even catastrophic system-wide failure. The cases that we consider are presented in Table 1 below. Our basic 'constitutive' assumption is that there is a simple linear relationship between the probability of successful configuration and the rate(s) of communication with the policy- and enforcement-source(s). We look only at the coarsest averages over time, in order to determine the long-term behaviours of the models. We consider a change of configuration ("charge") ΔQ to be proportional to an average rate of information flow (current) I , over a time Δt ; that is $\Delta Q = I\Delta t$. This equation is valid when I represents the time-averaged flow over the interval. Since we are interested in the limiting behaviour for long times, this is sufficient for our needs.

Now we apply this simple picture to configuration management for dynamic networks. We take the point of view of a 'typical' or 'average' host. It generates error in its configuration at the (average) rate I_{err} , and receives corrections at the rate I_{repair} . Hence the rate of increase of error for the average node is:

$$I_{\text{fail}} = (I_{\text{err}} - I_{\text{repair}})\theta(I_{\text{err}} - I_{\text{repair}}). \quad (6)$$

The Heaviside step-function is defined by $\theta(x) = 1$ if $x > 0$ and $\theta(x) = 0$ if $x \leq 0$, and signifies the fact that, if the repair rate exceeds the error rate, then (on average, over long times) nothing remains outstanding and there is no net rise in configuration error. Thus this averaged quantity is never negative.

If random errors and changes to configuration occur at a rate I_{err} and the configuration agent is unavailable to correct them, then $I_{\text{fail}} = I_{\text{err}}$. If this holds during a time Δt , the configuration falls behind by an amount:

$$\frac{\text{Bytes missing}}{(\Delta Q)} = \frac{\text{bytes/sec}}{(I_{\text{err}})} \times \frac{\text{seconds unavailable}}{(\Delta t)} .$$

In the following we will use p to denote the average (over time, and over all nodes) probability that configuration management information flow (repair current) is not available to a node. This unavailability may come from either link or node unreliability. We can lump all the unreliability into the links (see above) and so write $p = (1 - \langle A_{ij} \rangle)$, where $\langle A_{ij} \rangle$ denotes both time and node-pair average. Each node then can only receive repair current during the fraction $(1 - p)$ of the total elapsed time.

The repair current is generated by two possible sources in our models: i) a remote source, and ii) a local source. In each case, the policy can be transmitted and/or enforced at a maximum rate given by the channel capacity of the source. We shall denote the channel capacities by C_R and C_L for remote and local sources for clarity, but we assume that $C_R \sim C_L$, since source and target machines are often comparable, if not identical. If the communication by network acts as a throttle on these rates, then one can further

assume that $C_R < C_L$. In any case, the weakest link determines the effective channel capacity. Note that in the case of a confluence of traffic, as in the star models below, the channel capacity will have to be shared by the incoming branches. We now have a criterion for eventual failure of a configuration strategy. If $I_{\text{fail}} = \frac{\Delta Q}{\Delta t} > 0$, the average configuration error will grow monotonically for all time, and the system will eventually fail in continuous operation. Our strategy is then to look at the scaling behaviour of I_{fail} as the number of nodes N grows large.

A. Star model

The starting point for our models is the traditional (idealized) model of host configuration is based on the idea of remote management (e.g. using SNMP). Here there is a central manager who decides and implements policy from a single location, sending instructions that read and write necessary state. In this first case, all of the network connections and hosts are considered to be completely reliable. The manager must monitor the whole network, using bi-directional communication. This leads to an $N : 1$ ratio of clients to manager (see fig 1). This first model is an

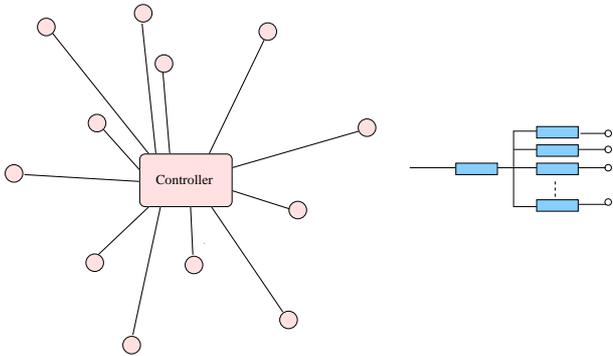


Fig. 1. Model 1: the star network. A central manager maintains bi-directional communication with all clients. The links are perfectly reliable, and all enforcement responsibility lies with the central controller.

idealized case in which there is no unreliability in any component of the system. It serves as a point of reference.

The topology on the left hand side of fig 1 is equivalent to the component topology on the right hand side. We can assume a flow conservation of messages on average, since any dropped packets can be absorbed into the probabilities for success that we attribute to the adjacency matrix. Thus the currents must obey Kirchoff's law:

$$I_{\text{controller}} = I_1 + I_2 + \dots + I_N. \quad (7)$$

The controller current cannot exceed its capacity, which we denote by C_S . We assume that the controller puts out repair current at its full capacity (since the Heaviside function corrects for lower demand), and that all nodes are average nodes. This gives that

$$I_{\text{repair}} = \frac{C_S}{N}. \quad (8)$$

The total current is limited only by the bottleneck of queued messages at the controller, thus the throughput per

node is only $1/N$ of the total capacity. We can now write down the failure rate in a straightforward manner:

$$I_{\text{fail}} = \left(I_{\text{err}} - \frac{C_S}{N} \right) \theta \left(I_{\text{err}} - \frac{C_S}{N} \right). \quad (9)$$

In other words, the failure rate is zero until the total rate of errors exceeds the capacity of the channels, at which point it scales inversely with N .

As the number of hosts becomes very large, as in a pervasive environment, $N \rightarrow \infty$, $I_{\text{fail}} \rightarrow I_{\text{err}}$ —i.e., the controller contributes a vanishing repair current per node and no errors are corrected. At this point, the system is out of control. The system fails however at a finite $N = N_{\text{thresh}} = C_S/I_{\text{err}}$. This highlights the clear disadvantage of centralized control, namely the bottleneck in communication with the controller.

B. Star model in intermittently connected environment

The previous model was an idealization, and was mainly of interest for its simplicity. Realistic centralized management must take into account the unreliability of the environment.

In an environment with partially reliable links, a remote communication model bears the risk of not reaching every host. If hosts hear policy, they must accept and comply, if not, they fall behind in the schedule of configuration. Monitoring in distributed systems has been discussed in ref. [1].

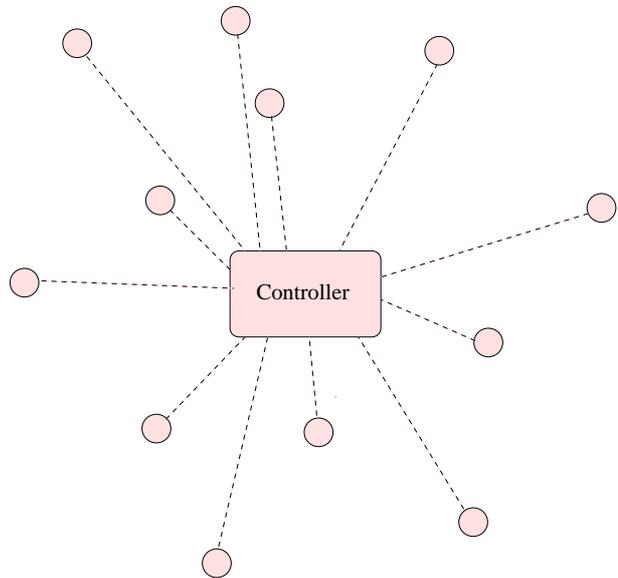


Fig. 2. Model 2: a star model, with built-in unreliability. Enforcement is central as in Model 1.

The capacity of the central manager C_S is now shared between the average number of hosts ($\langle N \rangle$) that is available, thus

$$I_{\text{repair}} = \frac{C_S}{N \langle A_{ij} \rangle} \equiv \frac{C_S}{\langle N \rangle}. \quad (10)$$

This repair current can reach the host, and serve to decrease its policy error ΔQ , during the fraction of time

TABLE I

Comparison of models from the viewpoint of the different dimensions: policy dissemination, enforcement, freedom of choice, whether hosts can exchange chosen policy ideas with peers and how political control flows. A ‘push’ model implies a forcible control policy, whereas ‘pull’ signifies the possibility to choose. Model 3 lies between these two, in having the possibility but not the inclination to choose.

Model	Application Topology	Enforcement	Policy Freedom	Policy Exchange	Control Structure
1	Star	Transmitted	No	No	Radial push
2	Star	Transmitted	No	No	Radial push
3	Mesh	Local	No	No	Radial pull
4	Mesh	Local	Yes	No	Radial pull
5	Mesh	Local	Yes	Yes	Hierarchical pull
6	Mesh	Local	Yes	Yes	P2P pull

$(1 - p)$ that the typical host is reachable. Hence we look at the net deficit ΔQ accrued over one “cycle” of time Δt , with no repair current for $p\Delta t$, and a maximal current $C_S/\langle N \rangle$ for a time $(1 - p)\Delta t$. This deficit is then

$$\Delta Q(\Delta t) = I_{\text{err}}p\Delta t + \left(I_{\text{err}} - \frac{C_S}{\langle N \rangle} \right) (1 - p)\Delta t \quad (11)$$

(here it is implicit that a negative ΔQ will be set to zero). Thus, the average failure rate is

$$I_{\text{fail}} = I_{\text{err}}p + \left(I_{\text{err}} - \frac{C_S}{\langle N \rangle} \right) (1 - p) = I_{\text{err}} - \frac{C_S}{N} \quad (12)$$

(Again there is an implicit θ function to keep the long-time average failure current positive.) This result is the same as for Model 1, the completely reliable star. This is because we assumed the controller was clever enough to find (with negligible overhead) those hosts that are available at any given time, and so to only attempt to communicate with them.

This model then fails (perhaps surprisingly), on average, at the same threshold value for N as does Model 1. If the hunt for available nodes places a non-negligible burden on the controller capacity, then it fails at a lower threshold. For large N , we have:

$$I_{\text{fail}} \rightarrow I_{\text{err}}. \quad (13)$$

Note that we now have an interesting conclusion. It can actually pay, for some N to have certain hosts down or unavailable, since this will increase the throughput to the central manager and delay the inevitable failure.

C. Mesh topology with centralized policy and local enforcement

The serialization of tasks in the previous models forces configuration ‘requests’ to queue up on the central controller. Rather than enforcing policy by issuing every instruction from the central source, it makes sense to download and cache a summary of the policy to each host and empower the host itself to enforce it.

There is still a centrally determined policy for every host, but now each host carries the responsibility of configuring itself. There are thus two issues: i) the update of the policy

and ii) the enforcement of the policy. A pull model for updating policy is advantageous here, because every host then has the option to obtain updates at a time convenient to itself, avoiding confluence contentions; moreover, if it fails to obtain the update, it can retry until it succeeds. We ask policy to contain a self-referential rule for updating itself.

The distinction made here between communication and enforcement is important, because it implies distinct types of failure, and two distinct failure metrics: i) distance of the locally understood policy from the latest version, and ii) distance of host configuration from the ideal policy configuration. In other words: i) communication failure, and ii) enforcement failure.

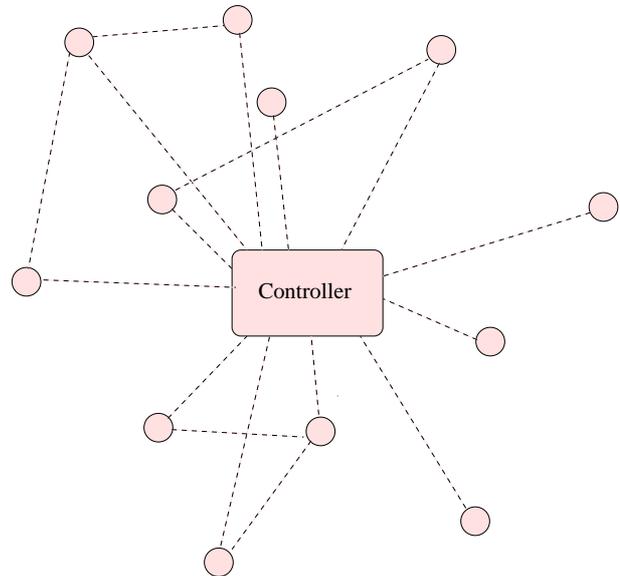


Fig. 3. Model 3. Mesh topology. Nodes can learn the centrally-mandated policy from other nodes as well as from the controller. Since the mesh topology does not assure direct connection to the controller, each node is responsible for its own policy enforcement.

The host no longer has to share any bandwidth with its peers, unless it is updating its copy of the policy, and perhaps not even then, since policy is enforced locally and updates can be scheduled to avoid contention.

Let I_{update} be the rate at which policy must be updated.

This current is usually quite small compared to I_{err} , and was neglected in the previous models. Based on the two failure mechanisms present here, we break up the failure current into two pieces: $I_{\text{fail}} = I_{\text{fail}}(i) + I_{\text{fail}}(ii)$. The former term is

$$I_{\text{fail}}(i) = (I_{\text{err}} - C_L)\theta(I_{\text{err}} - C_L); \quad (14)$$

this term is independent of N and may be made zero by design. $I_{\text{fail}}(ii)$ is still determined by the ability of the controller to convey policy information to the hosts. However, the load on the controller is much smaller since $I_{\text{update}} \ll I_{\text{err}}$. Also, the topology is a mesh topology. In this case the nodes can cooperate in diffusing policy updates, via flooding¹.

The worst case—in which the hosts compete for bandwidth, and do not use flooding over the mesh—is that, for large N , $I_{\text{fail}} \rightarrow I_{\text{update}}$. This is a great improvement over the two previous models, since $I_{\text{update}} \ll I_{\text{err}}$. However note that this can be further improved upon by allowing flooding of updates: the authorized policy instruction can be available from any number of redundant sources, even though the copies originate from a central location. In this case, the model truly scales without limit, i.e. $I_{\text{fail}} = 0$.

There is one caveat to this encouraging result. If the (meshed) network of hosts is truly an ad-hoc network of mobile nodes, employing wireless links, then connections are not feasible beyond a given physical range r . In other words, there are no long-range links: no links whose range can grow with the size of the network. As a result of this, if the AHN grows large (at fixed node density), the path length (in hops) between any node and the controller scales as a constant times \sqrt{N} . This growth in path length limits the effective throughput capacity between node and controller, in a way analogous to the internode capacity. The latter scales as $1/\sqrt{N}$ [15], [16]. Hence, for sufficiently large N , the controller and AHN will fail collectively to convey updates to the net. This failure will occur at a threshold value defined by

$$I_{\text{fail}}(ii) = I_{\text{update}} - \frac{C_S}{c\sqrt{N_{\text{thresh}}}} = 0, \quad (15)$$

where c is a constant. The maximal network size N_{thresh} is in this case proportional to $\left(\frac{C_S}{I_{\text{update}}}\right)^2$ —still considerably larger than for Models 1 and 2.

An alternative to flooding is to simply rely on the central server for updates, as in Models 1 and 2. In this case, the update term $I_{\text{fail}}(ii)$ is the same as for those models, with I_{err} replaced by I_{update} . This is worse than the flooding method above.

D. Mesh topology with partial host autonomy and local enforcement

As a variation on the previous model, we can begin to take seriously the idea of distance from a political centre.

¹Note, flooding in the low-level sense of a datagram multicast is not necessarily required, but the effective dissemination of the policy around the network is an application layer flood.

In this model, hosts can choose not to receive policy from a central authority, if it conflicts with local interests.

For example, our mobile, ad hoc future is likely to involve many personal devices, governed by their owners rather than configured according to any standard. These devices will roam, like tourists, through regions of foreign policy where they must either agree or conform to local rules or defy them. In such an environment, personal freedoms come into play and users will demand the freedom to decide over their own property. No absolute control will be acceptable.

Communication thus takes the role of conveying ‘suggestions’ from a local central authority, in the form of the latest version of its policy. For instance, a central authority might insist on the use of a particular vendor’s proprietary product in order to qualify for local service provision. This might be unacceptable to the individual client who would prefer to defer its access to the service until a more acceptable solution is available. In another scenario the central authority might suggest a new version of widely-used software, but the local authority might delay the upgrade due to compatibility problems with local hardware.

To implement this kind of freedom, local enforcement is now a necessity. Each node is empowered to hold to its chosen policy P_i . Thus communication and enforcement use formally distinct channels (as with Model 3); the difference is that each node has its own target policy P_i which it must enforce.

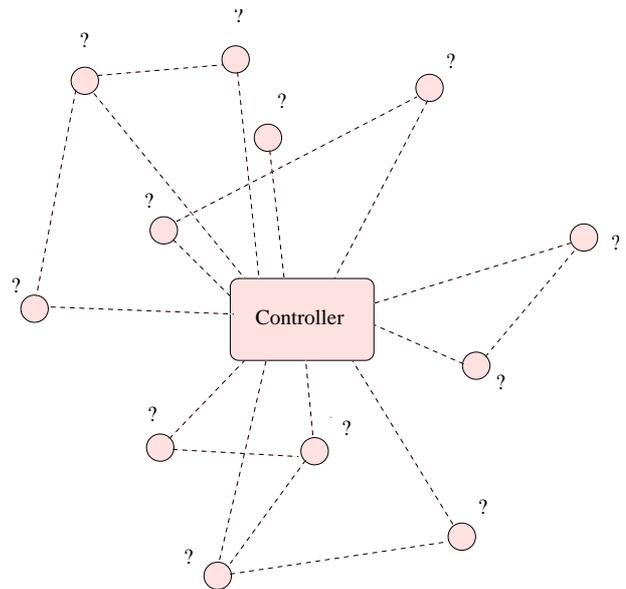


Fig. 4. Model 4. As in Model 3, except the hosts can choose to disregard or replace aspects of policy at their option. Question marks indicate a freedom of hosts to choose.

The communications and enforcement challenges faced by Model 4 are the same (in terms of scaling properties) as for Model 3: i.e. I_{fail} is the same as that in Model 3. Hence this model can in principle work to arbitrarily large N . Model 4 is the model used by cfengine[4], [8]. The largest current clusters sharing a common policy are known to be of order 10^4 hosts, but this could soon be of

order 10^6 , with the proliferation of mobile and embedded devices.

E. Mesh, with partial autonomy and hierarchical coalition

An embellishment of Model 4 is to allow delegation of responsibility and overriding of central politics in a hierarchy of management. Local groups of hosts may now form policy coalitions, that serve to their advantage. A policy coalition is a region of the network that agrees on policy. Such groups of hosts might belong to one department of an organization, or to a project team, or even to a group of friends in a mobile network. Once groups form, it is natural to allow sub-groups and thence a generalized hierarchy of policy refinement through specialized social groups.

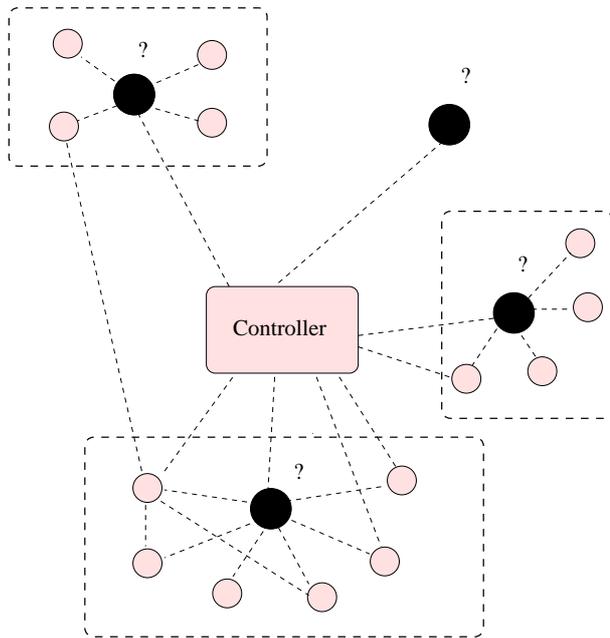


Fig. 5. Model 5. Communication over a mesh topology, with policy choice made hierarchically. Sub-controllers (dark nodes) edit policy as received from the central controller, and pass the result to members of the local group (as indicated by dashed boxes). Question marks indicate the freedom of the controllers to edit policy from above.

If policies are public knowledge then the scaling argument of Model 3 still applies since any host could cache any policy; but now a complete policy must be assembled potentially from several sources. One can thus imagine using this model to distribute policy so as to avoid contention in bottlenecks, since load is automatically spread over multiple servers. In effect, by delegating local policy (and keeping a minimal central policy) the central source is protected from maximal loading.

Specifically, if there are S sub-controllers (and a single-layer hierarchy), then the effective update capacity is multiplied by S . Hence the threshold N_{thresh} is multiplied (with respect to that for Model 3) by the same factor. This model can be implemented using cfengine, with some additional scripting.

F. Mesh, with partial autonomy and inter-peer policy exchange

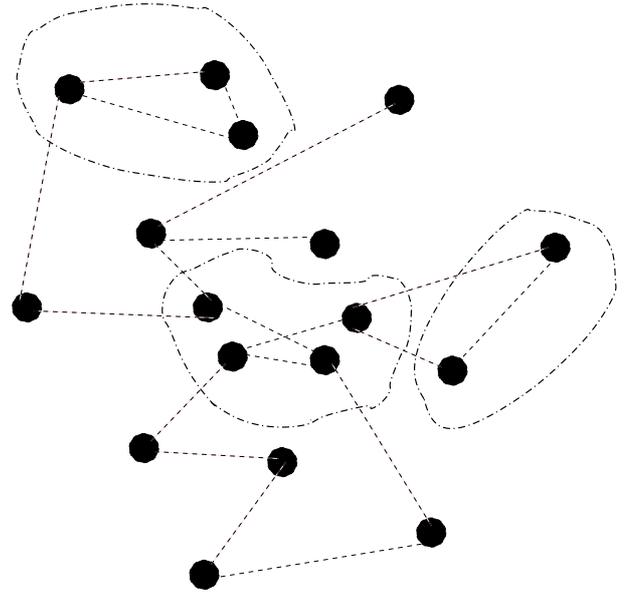


Fig. 6. Model 6. Free exchange of policies in a peer-to-peer fashion; all nodes have choice (dark). Nodes can form spontaneous, transient coalitions, as indicated by the dashed cells. All nodes can choose; question marks are suppressed.

The final step in increasing autonomy is the free exchange of information between arbitrary hosts. Hosts can now offer one another information, policy or source materials in accordance with an appropriate trust model. In doing so, impromptu coalitions and collaborations wax and wane, driven by both human interests and possibly machine learning. Such coalitions can be stable if they have sufficient numbers to withstand outside influence or competing influences. All we require is the existence of a stable equilibrium in graphical terms (for an explanation see ref. [7]).

A peer-to-peer policy mechanism of this type invites trepidation from those who have invested traditionally in control mechanisms, but this need not be a concern. The peer equilibration model is really no more than a distributed genetic algorithm. With appropriate constraints it could always be made to lead to sensible convergent behaviour. Such constraints can be built into policy or can be implemented as standards that constrain allowable policies. The difficulty in this ‘free market economy’ view of policy exchange is that the very freedoms that peer to peer preserves make it impractical to be able to guarantee a standard for policy transmission. Each device has a potentially different ‘master’ determining its policy. Without special care, communication between devices can lead to spreading of policies throughout a network and therefore a loss of the illusion of ‘control’. There is a strong possibility that the policy network will then self-organize rather than behaving in a manner that can be considered ‘managed’. In such a situation, it becomes critical to know what the stable policies for local behaviour are: there would be no

sense in configuring a policy that would immediately fall apart.

A possible way to reduce this uncertainty is to employ a communications protocol based on only voluntary acceptance. A test model of this kind of inter-peer exchange has been implemented in cfengine[4] using a transport protocol called Voluntary RPC[11]. In this scheme hosts can safely exchange services and information without committing themselves to any reciprocal behaviour. All collaboration is entirely on a voluntary basis. This implementation is both a proof of principle and a practical tool for collaboration in environments such as Web Hotels and Internet Service Providers, where ‘ad hoc clients’ come and go over relatively long periods of time. We are also testing large scale configuration management using virtual machines based on User Mode Linux[3], using cfengine as a management agent.

One example of a collaborative peer network that has led to positive results is the Open Source Community. The lesson of Open Source Software is that it leads to a rapid evolution. A similar rapid evolution of policy could also be the result from such exchanges. Probably policies would need to be weighted according to an appropriate fitness landscape. They could include things like shared security fixes, best practices, code revisions, new software, and so on. Until this exchange nears a suitable stationary point, policy updates could be much more rapid than for the previous models. This could potentially dominate configuration management behaviour.

The peer model has no centre. Hence it is, by design, scale-free: all significant interactions are local. Therefore, in principle, if the model can be made to work at small system size, then it will also work at any larger size. In practice, this model is subject to potentially large transients, even when it is on its way to stable, convergent behaviour. These transients would likely grow with the size of the network. Here we have confined ourselves to long-time behaviour for large N —hence we assume that the system can get beyond such transients, and so find the stable regime.

Finally we note that we have only assessed the quality of a given model according to its success in communicating and enforcing policy. When policy is centrally determined, this is an adequate measure of quality. However, for those cases in which nodes can choose policy, one would also like to evaluate goodness of the resulting choices. We do not address this important issue here. We note however that Model 6, of all the models presented here, has the greatest freedom to explore the space of possible policies. Hence an outstanding, and extremely nontrivial, question for this peer-to-peer model of configuration management is: can such a system find ‘better’ policies than centralized systems?

When policy is determined by ad hoc collaboration many questions arise. Do such ad hoc policies form stable regions or will the consensus in a system fall apart? Are such regions stable to ‘infection’ by mutant policies? How stable do they need to be? In ref. [6] it is argued that policies

should be based always on fixed points or equilibria of possible host strategies, since these are the regions that provide the necessary stability for hosts to perform a reliable function. Studies of destabilization of policies by successful mutant policies are a matter for game theoretical studies[2]. We aim to return to these in future work in more detail. For now, we only point out that there is a graph theoretical phenomenon to be analyzed when policy is allowed to communicate itself from host to host. In a real sense, the dynamics of policy become directly analogous to the dynamics of biological diversity: DNA spreading and viral mutation. Policies become indistinguishable from viruses and the likelihood of their dominance in a network ecology depends on the large scale structure of the network graph[10].

In recent work, Stadler et al, have considered how policy could be updated not at random but according to algorithmic patterns that form standing waves on bounded networks[17], [18]. Although this idea is mainly directed at routing networks with non-trivial topologies, there is a suggestion that this is a more deterministic way of flooding a network with a single policy species.

VIII. SUMMARY AND CONCLUSION

We have presented several models for configuration management on networks, extending ref. [9]. Our Models 3–6 depart from mainstream practice in various ways. The motivation for considering these models is the perception that highly centralized systems are not well adapted to networks that are too large, too heterogeneous, or too dynamic. Since current and future networks are taking on more and more of these three qualities, it is of interest to examine alternative models for configuration management.

We have held ourselves to a limited set of goals. The first of these is the definition of the models themselves. These models offer broad avenues for future research in configuration management; variants of one (or several) of them are likely to be important in future systems. An important dimension along which our models vary is the degree of centralization: of policy determination, of policy communication, and of policy enforcement. For the reasons given in the previous paragraph, it is of significant interest to explore possible forms of network management with varying degrees and types of de-centralization. We offer a small step in this direction here.

Our second goal has been to assess the scaling behaviour of these models with respect to two criteria: *communication* of the current policy to the hosts, and *enforcement* of the communicated policy. We have considered the various models’ ability to meet these criteria, as the number of hosts N in the network grows large. We find, not surprisingly, that the highly centralized systems suffer from a communications bottleneck that limits the size at which they can function effectively. De-centralizing one or both of the two functions gives much better scaling behaviour—to the point that all of the Models 3–6 can, in principle (with some qualifications), implement policy communication and enforcement for very large systems.

Of course, de-centralization brings with it new problems, not addressed by the centralized system: problems of trust, of the quality of chosen policies, and of convergence to a stable regime. These new problems offer attractive issues for further research, due both to their intrinsic interest, and to their relevance to the future implementation of de-centralized network systems.

REFERENCES

- [1] H. Abdu, H. Lutfiya, and M. Bauer. A model for adaptive monitoring configurations. *Proceedings of the VI IFIP/IEEE IM conference on network management*, page 371, 1999.
- [2] R. Axelrod. *The Complexity of Cooperation: Agent-based Models of Competition and Collaboration*. Princeton Studies in Complexity, Princeton, 1997.
- [3] K.M. Begnum, J. Sechrest, and M. Burgess. Applications of user mode linux for virtual machine studies. *Proceedings of SANE conference*, 2004.
- [4] M. Burgess. A site configuration engine. *Computing systems (MIT Press: Cambridge MA)*, 8:309, 1995.
- [5] M. Burgess. System administration as communication over a noisy channel. *Proceedings of the 3rd international system administration and networking conference (SANE2002)*, page 36, 2002.
- [6] M. Burgess. On the theory of system administration. *Science of Computer Programming*, 49:1, 2003.
- [7] M. Burgess. *Analytical Network and System Administration — Managing Human-Computer Systems*. J. Wiley & Sons, Chichester, 2004.
- [8] M. Burgess. Cfengine’s immunity model of evolving configuration management. *Science of Computer Programming*, 2004.
- [9] M. Burgess and G. Canright. Scalability of peer configuration management in partially reliable and ad hoc networks. *Proceedings of the VIII IFIP/IEEE IM conference on network management*, page 293, 2003.
- [10] M. Burgess, G. Canright, and K. Engø. Stable flow distributions on directed networks. *Paper in preparation*, 2004.
- [11] M. Burgess, S. Hagen, and F. Sandnes. Voluntary rpc: how distributed automation can be accomplished in a pervasive computing environment. *Paper in preparation*, 2003.
- [12] J. Case, M. Fedor, M. Schoffstall, and J. Davin. The simple network management protocol. *RFC1155*, STD 16, 1990.
- [13] A.L. Couch. Chaos out of order: a simple, scalable file distribution facility for intentionally heterogeneous networks. *Proceedings of the Eleventh Systems Administration Conference (LISA XI) (USENIX Association: Berkeley, CA)*, page 169, 1997.
- [14] N. Damianou, N. Dulay, E.C. Lupu, and M. Sloman. Ponder: a language for specifying security and management policies for distributed systems. *Imperial College Research Report DoC 2000/1*, 2000.
- [15] P. Gupta and P.R. Kumar. The capacity of wireless networks. *IEEE Trans. Info. Theory*, 46(2):388–404, 2000.
- [16] J. Li, C. Blake, D.S.J. DeCouto, H.I. Lee, and R. Morris. Capacity of ad hoc wireless networks. *Proc. 7th ACM Intl. Conf. on Mobile Computing and Networking*, pages 61–69, 2001.
- [17] K-S Lima and R. Stadler. A navigation pattern for scalable internet management. *Proceedings of the VII IFIP/IEEE IM conference on network management*, 2001.
- [18] K-S Lima and R. Stadler. Weaver: Realising a scalable management paradigm on commodity routers. *Proceedings of the VIII IFIP/IEEE IM conference on network management*, 2003.
- [19] J.P. Martin-Flatin. Push vs. pull in web-based network management. *Proceedings of the VI IFIP/IEEE IM conference on network management*, page 3, 1999.
- [20] M. Matsushita. Telecommunication management network. *NTT Review*, 3:117–122, 1991.
- [21] R. Osterlund. Pikt: Problem informant/killer tool. *Proceedings of the Fourteenth Systems Administration Conference (LISA XIV) (USENIX Association: Berkeley, CA)*, page 147, 2000.
- [22] D.B. West. *Introduction to Graph Theory (2nd Edition)*. (Prentice Hall, Upper Saddle River), 2001.
- [23] M. Zapf, K. Herrmann, K. Geihs, and J. Wolfgang. Decentralized snmp management with mobile agents. *Proceedings of the VI IFIP/IEEE IM conference on network management*, page 623, 1999.