MARK BURGESS

configuration management: models and myths

PART 4: THERE'S NO I/O WITHOUT U

Mark Burgess is professor of network and system administration at Oslo University College, Norway. He is the author of Cfengine and many books and research papers on system administration.

Mark.Burgess@iu.hio.no

TRADE AND COMMUNICATION HAVE been in partnership since symbiosis emerged from evolution's curiosity shop, as a trick for smuggling contraband into the sum of the parts. This partnership is central to management of systems. Communication pervades, from the question a user asks at the help desk, to the data received from a router, to traffic flow statistics, to the implementation of configuration operations performed by software—there is an exchange of messages about state, about intention, and about change. Computer management is also increasingly about trade. All this communication is not for whim or curiosity; it has a direct value to us in terms of time, money, or service. However you look at it, the world of networks is the world of commerce.

When two parties wag their tongues or dance their dances, they are both altered by the exchange. When more parties are involved, there is a cumulative effect that spreads out along dentritic paths, binding the squawking flock together and forming a *network*. The trails blazed by those conversational patterns fashion the resulting behavior of all parties in the network. This wave of influence is the essence of management, whether it spreads like a diplomatic envoy or like a forest fire.

In this piece, I hope to persuade you to reexamine your beliefs about centralized, authoritative management in computing (or elsewhere). I want to argue that, in our modern world (surfing its way on the rising wave of free-market economics), we need to rethink the tradition of hierarchical, centralized governance in guiding the behavior of systems. It's a familiar song: Delegation and decentralization are not only desirable but inevitable if we are to cope with the rate at which we have to (re)configure and repair systems in a vibrant and adaptive network, built on the economics foundation of trade and services.

From a Cat's Cradle, Like a Bat Out of Hell

This is the network age, an age in which webs of communication are accelerating our technological and economic development. What is a network?

We overload this most important word with a plethora of meanings. Even in the limited domain of computer science, its meaning is not clear. Sometimes we mean the cable that joins the computer to the wall; sometimes we mean the infrastructure that enables communication between computers, including the routing and the switching; sometimes we implicitly mean the protocols that are spoken over these channels of copper and air; and sometimes we mean the abstract collection of computers themselves that are connected by the infrastructure (a social network of interaction formed between human and computer).

A network is "simply" a device or construct that joins many things or places together. The first technological networks were roads and sewers, built many thousands of years ago. Even before that, humans formed tribes linking humans together in structures of about thirty. But we should not be fooled into thinking that networks are human creations. In nature, networks are everywhere: Crystals and molecules are held together by networks of interatomic bonds. The structures of these networks determine the large-scale properties of the substances they form. In biology it is not genes that generate the complexity of the living world, but rather the networks of interconnections formed from the proteins that the genes encode. Our bodies are ripe with networks, as Arab scholars discovered and drew in exquisite detail during the first millennium: networks for blood transport, nerve signals, immunity, etc. Biology is a testament to the successful cooperation of multiple communicating parts.

Face-Centered Squareness

But as humans, we are frightened by complexity, even as we embrace it. The structures we configure purposely into networks (i.e., their topologies) are simple-minded. As they grow beyond our designs, we fret like worried parents over the consequences of this growth and try to protect systems with firewalls and other barriers. Recall Alvin Toffler's comments about industrialization from the last episode of the series.

It is no coincidence that published maps of the Internet look like snowflakes or leaves (see the images famously generated by Bill Cheswick at AT&T). It is not that the Internet appears biological; the point is that these biostructures are themselves networks. This is what networks look like when they emerge in the natural world purely as a result of mutually beneficial interaction. When engineers *design* networks they look quite different—like stars and trees. Humans only build in this "organic" way when things are "out of control" (i.e., when they are no longer designed by an engineer), but they "grow" following an economic principle of development rather than a regulated one. Why not? If these structures are so successful in nature, why don't we build like this? Perhaps we are small-minded.

Humans love to build centralized and hierarchical structures, whether industries, governments, or armies. Possibly there is a social-anthropic reason for this (perhaps an expert, on reading this, will tell me the answer): There is evidence to show that people have evolved to work in groups of around thirty at the most. Once this size is exceeded, they tend to break up into subgroups that cluster around a new leader. Another reason might be cultural, though the structure of families in which a family clusters around a dominant male, as attested to by the unfailingly nauseating and predictable references to "my father" or "daddy, daddy" (seldom "my mother" or "my family") at every tearful moment in American TV and film.

But there is a fascinating phenomenon going on here. Even if the size of our attention is limited, there is nothing obviously programmed into us that says how these groups must be organized, or is there? Well, roll up! Be amazed by our human propensity (perhaps desire) to subordinate ourselves before an authority figure. We behave like a bureaucracy of sheep in uniform. I always think of the scene from Monty Python's *Life of Brian* in which Brian tells the crowd, "Don't listen to me—you have to think for yourselves," to which the crowd cheers in unison, "Yes, yes! We must think for ourselves!"

Why command hierarchies? There is no evidence to suppose that such a structure is better than another. It is somewhat "natural" perhaps, like a dividing river or a branching tree, but it is far from robust. The tree structure has a certain clarity to it, but also great fragility. A tree is all about not reproducing the same alternative twice. A branching tree is a clean, economical, and logical separation of concerns, but the same properties also make it a structure of minimum redundancy and therefore quite fragile and blooming with bottleneck inefficiency.

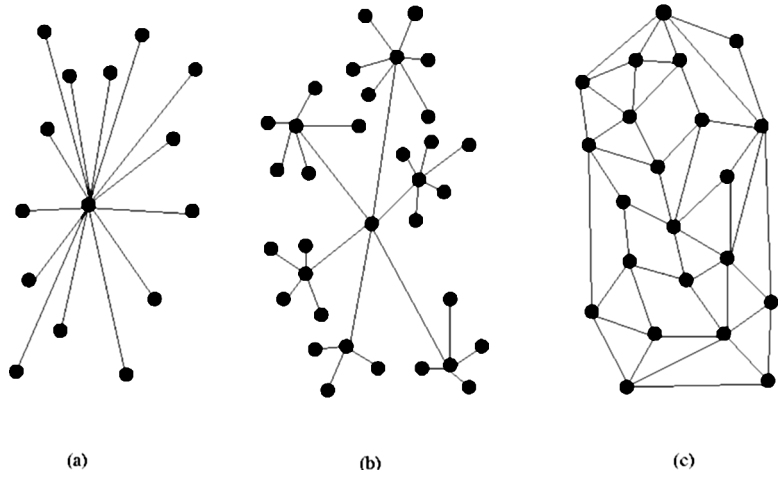

FIGURE 1. FROM CENTRALIZED STAR TOPOLOGY, TO HIERARCHICAL CENTRALIZATION, TO DECENTRALIZED MESH TOPOLOGY.

Create Like a God, Command Like a King, and Work Like a Slave

In terms of overhead, we can see why subordination (i.e., centralization) is appealing—if everyone follows a single master (Fig. 1a), then there are only $N - 1$ agreements instead of $N(N - 1)/2$ in a community of N agents. And yet public keys have shown us that we can form peer-to-peer collaborations with only N agreements and a little skill. What about consistency?

Every piece of knowledge must start from somewhere. That means there must be a source and a direction from which the information spreads. If there is only one source of information, then it must be consistent. Hence starlike topologies are perfect for local consistency. Q.E.D. If we move up a scale, then coordinating local communities according to a common policy can also be done from a single source; hence star hierarchies solve the problem (Fig. 1b). Tradition wins the day.

So, fine; centralization is sufficient, if we assume that the chief of the network can handle the burden—but is it necessary? The odds seem to be in

favor of centralization. But is this good engineering? Let's look at this dipolar list:

- Centralization (single source) versus delegation
- Top down versus bottom up
- Hierarchical versus peer-to-peer
- Data normalization versus data-mining

What if we look at survivable networks, such as biological organisms? Biological "devices" evolved only to survive in a changing environment. How? Through redundant distributed networking—the opposite of centralization. Even though some few of our major organs are singular (e.g., the brain and the heart) there is still redundancy built in: We are amazed by stories of how people who have suffered brain injuries learn, for the most part, to reroute their brain functions as a result of the phenomenal inbuilt redundancy. The single points of failure (spinal cord, heart, etc.) are still our greatest weaknesses, and these things limit our growth.

But surely all this redundancy and variation is much too expensive to maintain! I quote Toffler once again: "As technology becomes more sophisticated, the cost of introducing variations declines." Recall that the fear Toffler spoke of in industrialization was precisely that mass-production would lead to an inflexible lack of choice in a market—that you could have any color as long as it was black. Well, he also argued that this was nonsense once you have technology, because that is when you really can afford to make things cheaply.

So do we see any such technologies that diversify the playing field for configuration management? Indeed, we have various levels of automation tools, from SNMP-based Tivoli and Openview to policy-template configuration tools such as Cfengine, LCFG, Pikt, and the Web-based interfaces provided by a variety of operating systems. Although SNMP is now widely regarded as a failure (even by the IETF) for everything with the possible exception of monitoring, the template-based tools, albeit imperfectly, enable great variability in mass-produced environments. The largest Cfengine installations, for instance, run into the tens of thousands on a single site, some with large variations from laptops to supercomputers. Clearly variation management is no longer a real issue for technology. Let's see how it comes about.

Speak to Me in Many Voices, Make Them All Sound Like One

The first universal theory of symbolic communication was pioneered by Claude Shannon in the 1940s. He turned the idea of communication into a science and implicitly solved the issue of maintenance at the same time. His theory of communication over a noisy channel is one of the classics of electrical engineering (or information science; take your pick). It makes that important point that you cannot escape from the problem of signal noise in a system. All systems contain noise (uncontrolled variations), in some manner or form. If a message is communicated in a noisy environment it becomes unclear, rough and grainy; the chance of it being understood and obeyed is much smaller (as the pony said to the ventriloquist, "I'd love to talk to you, but I'm afraid I'm a little horse").

Symbolic (digital) communication was the basic technology that enabled electronic networking and computation. After this, the idea of queuing and packet switching brought us from the fast train-track communication of the telephone network to the automobile diversity of the Internet protocol.

Networks now relay communications between peers by encapsulating any kind of message with a single lingua franca of IP (more or less).

The messages might now all sound like a single language, but they carry more diversity than ever before. By deregulating the centralized structure of the telecoms and by deregulating the single source content using the open (emergent) standard of IP, diversity has grown into a commerce of communication with a tolerable level of variation. Remarkably, a plethora of standards has converged into one, just as kids who are left to dress without school uniforms converge on jeans and T-shirts and a small number of basic themes. The lesson here is that when you deregulate something, you might actually end up with greater uniformity than before, because people lose interest in fighting for supremacy—they become content to live and prosper in their own niches.

In previous issues, I talked about the structure of patterns in a configuration. A network too is a configuration, which can be laid out as a formal language. The hierarchical tree structure we are used to in a context-free language (e.g., XML) has no a priori superiority to that of a more random peer-to-peer structure. Hierarchies possess *relative computational simplicity*, meaning that they are cheap to parse or build by linear computation, but they are only marginally more expressive than regular grammars that correspond to peer relationships.

Why would we think that a context-free, military hierarchy was the best solution to management? Perhaps because the alternatives are currently too hard for us to fully understand. Grammars that use parentheses make it easy to put things in boxes, and this is a comfortable way of marking out territory and assigning responsibility. But, in practice, we do not even use deep hierarchies in the organization of network patterns, usually implementing only two levels: master and slave hosts. That depth of pattern grammar can easily be built as a regular language, but it is "faux," being more about limitation than structure. It requires only that each slave in the network promise to follow the instructions of a master, which in linguistic terms is just a prefix. This is perhaps a clue that what we really value is perceived *cheapness* rather than subordination. We just think that hierarchy must be cheaper than a less structured pattern, although the theory of languages says otherwise.

Selling Your Soul at the Crossroads

In the past few years, users and researchers alike have come to realize the economic limitations of hierarchical regulation. A hierarchy implies a set of bottlenecks and barriers, of permissions and subordinations, but users have been given the power to compute and, by George, they do! *Service-oriented computing* has arrived to stay. It is direct, it is valuable to individuals, and it is subordinate to no one.

Technology is no longer the plaything of governments and governing boards for strategic purposes; it lies in the hands of ordinary folk who simply want to trade. This desire to trade, to exchange information and services in mutually beneficial ways, is what has driven the Internet into a state of biological complexity. It is a new technological symbiosis that enables our society to move to a new level of cooperation that can handle groups of bigger than thirty.

But what of the cost of this ad hoc, symbiotic organization? Price is clearly a subjective point of view in this story, and the cost of management is somewhat dependent on your particular skills (as Toffler says, "As technol-

ogy becomes more sophisticated . . .”). Today these currencies for management are in flux, and centralization is being displaced by peer services, through the Web, and through file-sharing software. Could it be that *planned structure* could be supplanted by an organically (economically) grown *emergent structure*?

Well, this might all sound like a dream from some 1970s biological material, but don't reject this thought without seeing the wood in the trees: Just because a network was not designed does not mean that it is less functional than one that is. Just because behavior emerges from the cradle of economic self-interest (symbiosis) does not mean that it is less predictable than a military operation. All life and society emerged this way—and we do very nicely, thank you.

So, in case you thought I had forgotten about configuration management in this daydream about networking, let's tie the floating pieces in our Article-Area-Network together, seeing how we can have the best of both worlds: predictability and freedom along with personal safety and opportunistic self-interest.

Autonomous Meditation: The State of Standing Still

Shannon's model of the noisy channel applies equally to computers talking to themselves. Self-interest begins with the correct functioning of the individual. What easier way to maintain system state than to have it chant that state over and over again until it works harmoniously?

The passage of time brings many influences to bear on systems that we do not have any control over. Developers of computer systems have made a frequent error in viewing *configuration management* only as *change management* (as in a transaction system such as a database). It is a bit like believing that the weather is really an air conditioner with a nice neat knob to switch it on and off.

Self-maintenance is communication if you see a computer system as being in a constant state of meditation, at each moment repeating a mantra that we can call its state. We would like it to chant a message that agrees with our policy for its state. By repeating the message one then reinforces it. This metaphor describes the idea of autonomous configuration management.

From the previous articles, we think of the state of the computer as some string of configuration-operational characteristics that forms an alphabet. This can be coded in any imaginable way (e.g., suppose it is “ABHEKSYGHETFDH . . . ,” where A means something like “chmod 644 /etc/passwd,” etc.)

After a while, corruption of the state message owing to run-time interactions, meddlesome users, and network connections (i.e., noise) could lead to this state message being garbled, changing some of the symbols into others. Such a change must be corrected by reiterating the actual policy (e.g., “ABCD” -> “ABXD” -> “ABCD”). Just like the message over a noisy channel, we have to correct these errors.

The convergent operations we mentioned in the last article deal with this problem nicely. In operational language, a single operator is a unit of one kind of instigator of change, which we write:

$$O\ q = q'$$

to mean an operation applied to a state q leads to a transition to a new state q′. But rather than thinking about a transition from one state to a new state, think of this, rather, as error correction. A "convergent" operator is a message that tells any state to transform into a policy compliant state:

C (Any state) -> (Policy state)

The policy state is said to be a fixed point of the policy operator since once you get there you stay there. So, in terms of this language, all we need to do is to repeat the entire policy over and over again, like a never-ending mantra, with a separate operator for each independent kind of change:

$C_1C_2C_3 \ldots C_n$ (Any state) -> (Policy state)

Alva Couch called this the Maelstrom property in his LISA paper from 2001. The importance of this property is that a computer can simply chant its policy message to itself, applying strings of these operational messages, and this will ensure that it is always in the error-corrected, policy-compliant state. This is not science fiction. The approach was developed originally and used for Cfengine, in a limited form, and something like it is now being used in NETCONF and some other configuration management technologies designed to replace SNMP.

Though I Speak with the Tongues of Convergent Fixed-Point Operators

There is, of course, a danger to emphasizing the role of communication, language, or error correction too much. In the network management communities people often get stuck by confusing basic ideas with the technologies that communicate them. It would be no surprise to us that our leaders failed to govern the country if they held the view that management were the same as SNMP. Ideas generalize implementations; they should not be limited by them.

In the case of SNMP there was a conscious decision made by the IETF to avoid complexity in the protocol. This regulation, in turn, led to an explosion of complexity in the data structures (MIBs). You cannot suppress noise by trying to pretend it is not there. What fixed-point semantics offers us (at least in the cases where it has been possible to implement such a thing) is the guarantee that a message repeated will be easily implemented, rather than being a Mission Impossible.

What I am proposing here is that it can be sufficient to manage device internals individually, while allowing networks to trade freely. The result will not necessarily be "out of control" in a bad way; it will regulate itself *autonomically*.

Five Farthings, Say the Bells of St. Martin's

For managing the configurations of computers, networks are not essential. It is clearly possible to administer changes and repairs to a completely isolated, stand-alone computer, either manually or with the aid of automation, "one-on-one." However, the network opened the door to both collaboration and interconnection, therefore making it possible to manage systems remotely. This is only true, however, if such collaboration has an economic justification.

We can disconnect any computer from a network and take over the management of the device, and no one can stop us; hence the idea that computers are "controlled" from outside is only a convenient fiction. They are controlled insofar as they want to be. It is an act of *voluntary cooperation* to

allow oneself to be managed by an external authority. Just as we bow to authorities, network management researchers find this idea almost impossible to accept.

Messages of different types have different values for us. Just being associated with a service provider in a BGP peering agreement can be worth a lot of money—kudos to the peer that earns respect and hence the promise of future profit. Association is a social currency that is worth something—not necessarily money. We have to learn to recognize the different forms of currency in play in economic cooperation. The face of commerce is changing.

Why would anyone talk about trade if they could control everything and nail everything down. You only have to compare the state of dictatorships with democracies to answer that one. The authoritarian regime might argue, you need me for:

- Offloading and convenience
- Specialist knowledge
- Separation of concerns
- Mutual advantage

But all of these things can, in fact, be obtained from your neighbor and today these are used as the primary reasons for outsourcing to companies that are considered to be *subordinate,* not superordinate, authorities.

Don't we need a law-maker or some other kind of authority to govern? In contract law it is observed that the threat of litigation in an authoritarian regime is essentially insignificant in determining whether the terms of a contract are upheld or broken. The principal factor that determines whether or not people break the law is the potential loss in economic value to the participants in the contract. Hmmm. . . think about that one.

Haggling Over the Price

It is possible to describe policy patterns and structures graphically using a theory of *promises*. We have been developing this theory in Oslo for the past two years. It allows us to study these matters of economically motivated cooperation. Promise theory tells us that we do not have to abandon personal autonomy to have distributed cooperation.

The economics of system administration change. At the beginning of the 1990s, when I developed Cfengine, there was a particular need for competitive garbage collection in systems—simply to keep them alive. Disks had finite size and the memory of the system was limited. This shaped the functions that were built into the configuration engine. Today, this recycling task has (for the moment) been deemphasized, as we have wider margins in modern systems. Tools that are produced today place more emphasis on installation, as if our fossil reserves of storage were infinite, or they ignore the presence of unnecessary processes, as if the heat produced by these unnecessary computations will not cost us dearly in the long run (either from increased electricity bills or the melting of the polar ice caps). The only thing that will apparently convince us to be more careful today is our confused paranoia over "security" (whatever that means to us).

Today "over-provisioning" (over-provisionizationing) allows us to swagger richly through the data center, paying little attention to the waste. This too is purely a matter of economics. We currently have no incentive to try to improve, but the time will come again when this bountiful Cretaceous era of information diversity meets its Tertiary boundary, and our systems will

once again have to deal with loads that tax their resources to extinction. Limits will be reached, and we shall be operating once again on the edge where the balance of trade proves crucial to the survival of the species.

Please Sir, ISO Want to Buy Your BS-Enabled ITILity!

So what of the tail end of this tale? Services. The service paradigm has arrived to stay, in business, in commerce, and certainly in computing. This paradigm has all but wiped out the conventional notion of system administration in the eyes of network research and development and the telecom service providers. To them UNIX is an application service; Windows is something to be updated over a network.

Businesses are gearing up for service management. Standards of good practice for service delivery management were developed by the British government in the late 1990s and have grown into a ubiquitous standard of practice in business today called the Information Technology Infrastructure Library (ITIL). The manuals and documents for this standard are still sold at great expense, and a summary was published as British Standard BS15000 and now also as ISO20000. These documents mention configuration management, although they say nothing about how (or indeed if) it can be implemented. They merely recommend that a software engineering type of versioning be used for the management for all kinds of configuration data. This is not the same story I have been telling in these articles, but it is a higher-level aspect of it.

It is probably possible to describe any enterprise as a number of interacting services, trading with one another for mutual advantage. Once the advantage disappears, the enterprise falls apart. ITIL and its bureaucratic rival the NGOSS/eTOM (enhanced Telecom Operations Map) help managers to see some aspects of good governance, so that service providers will be able to document their organizations and associated feel-good behavior, but they speak only of quality of the process surrounding the service (a sort of managerial version of the meditation discussed here). They say nothing about practical implementation or of the technical challenges.

In this series I have tried to show that the matters of system configuration and policy-guided behavior have a technical basis that is sometimes ignored. The future of low-level configuration management (in the sense of computer governance) lies with automation, whereas the high-level behavior of interaction lies in commerce. There is much research to be done in this area. We must understand the science and the economics of this, not merely the tradition and the doctrine—and that science is of communication. Communication, in turn, requires language: There must be a language to express the changes, desires, and policies of our self-regulating systems. And only when all of these pieces of the puzzle are in place can we say that we have fully understood configuration management.