Networks and Embedded Knowledge: Challenges for Computer Ethics

Christopher N. Chapman, Ph.D.
Microsoft Corporation*

** The statements in this paper are the opinions of the author and do not necessarily reflect positions of the Microsoft Corporation.*

Paper presented at:
Philosophy in the Contemporary World Annual Meeting
July 2003
Santa Fe, New Mexico

Author contact:

(preferred)	*(other)*
P.O. Box 99554	1 Microsoft Way (cchap)
Seattle, WA 98199	Redmond, WA 98052
(206) 285-7324	(425) 705-2445
cchap@microsoft.com	

Biographical note

Christopher N. Chapman, Ph.D.
Microsoft Corporation

I am a usability engineer in the Windows Server development group at Microsoft Corporation. My role is to understand Windows network administrators and how they view, use, and understand computer systems, in order to build better products for their needs and the needs of their organizations and users.

Prior to joining Microsoft in 2000, I was a postdoctoral fellow at the University of Washington Medical Center where I worked on complex systems research applied to the psychophysiology of pain. I hold an MA and PhD in clinical psychology, and an MA in philosophy. In college and graduate school, I worked in a variety of technology positions, including computer consulting, support, and software development.

Networks and Embedded Knowledge: Challenges for Computer Ethics

Abstract

In this paper, I describe the traditional view of a computer network as a protected group of machines within a hardened perimeter, and show how such a view is inadequate as a description of current computer networks. We should instead think of networks as enmeshed systems that embody distributed knowledge. Several areas of computer ethics are affected by a shift to such a view, including theories of identity, accountability, and intellectual property. I give a case study of a recent Internet worm in light of the conceptual shift advocated here.

Introduction

The nature of our computer systems and the global computer network is constantly changing. Although many of the formal, theoretical characteristics of computer systems have been well-studied and known for decades, actual implementations of technology change rapidly. With changes in technology, there are simultaneous changes in the ways that people relate to technology, both as technology users and consumers and as technology professionals. As relationships between machines and people change, so do the relationships among people that are mediated by machines.

I focus here on a particular form of technology and the people who enter into direct relationships with it: network server technology. I outline this technology and then examine ways in which classical models of the technology are changing in the world around us. I argue that these changes mirror shifts in the ways that technology professionals conceive of and relate to computer systems and software.

Philosophical discourse about computer ethics often relies upon analysis of limited and increasingly outdated scenarios of software and computer use. Most discussions of computer ethics focus on end user systems, on the concepts and experiences relevant to individual users, and on software as predefined, packaged consumer products. In this paper, I focus instead on network models, with the belief that a better understanding of computer networks will enrich philosophical discourse about information technology. In particular, I highlight implications of modern changes in network systems for ethics of accountability, identity, and intellectual property.

Computer Networks and Boundaries

A "network" is the aggregation of any two entities that have a means to exchange information, along with the elements that comprise the connection between them. However, for the purposes of this paper, I shall use the word "network" as shorthand for a more specific, intuitive, and widespread notion: a "network" here refers to a collection of computers that are interconnected as a network proper, that fall under the professional management of some particular person or organization, and at least some of which have connections to a diversity of computers outside the organization. In other words, I will talk about organizational networks that have Internet connectivity. This includes the computer systems of virtually all businesses, governmental bodies, colleges and universities, and most other sizable organizations.[1]

The traditional view of such networks follows naturally from the intuitive description above: there is a collection of computers; those computers use specific electrical devices to connect to one another; there is software on them that mediates

[1] It leaves asides the following: networks that are not professionally administered, such as home networks; networks that have no connection to the outside, such as dedicated systems that operate factory production lines and the like; networks comprised entirely of devices that are not computers, such as electrical systems; and ad hoc collections of connected devices, such as a wireless ad hoc network. It is also possible for a network to have outside connectivity to networks other than the Internet (e.g., using dedicated links between business partners) but it is likely that such organizations also have systems on the Internet and thus fall under the scope here.

those connections; and there are some systems that facilitate the flow of information to and from the Internet.

In such a model, there is a rigid division between the local systems and outside systems. This is envisioned as a *perimeter*, with all the computers in the local organization inside the boundary, managed by the local network administrators. Outside the boundary is the Internet. Straddling the boundary are dedicated devices that manage the flow of information between inside and outside. In technical terms, there is a “local network” inside, a “wide area” network or “Internetwork” outside, and there are “gateways” that bridge the two.

The perimeter divides the world for security purposes: computers and users on the local network are trusted to access the private information on local computers, while machines and people outside the perimeter are viewed as hostile. Administrators speak of there being a “hardened wall” surrounding the local network; in contrast, the local network is “soft on the inside.” Enforcement of the division is performed by “firewalls.”

This perimeter-based conception of computer networks is grounded in the confluence of both political and historical factors: it maps naturally onto the organizational boundaries of traditional businesses and organizations, and it reflects the fact that computer systems were once largely independent of one another and only later became connected to the outside world. It also matches traditional categories of ethical, political, legal, and economic discourse: the local organization is responsible for its own systems; the boundary denotes “us” and “them”; there is, in principle, an adversarial relationship with everyone outside the perimeter; and the flow of information among entities is organized on the basis of a free resource (in terms of theoretic supply, not cost) with a mixture of contractual and laissez-faire relationships. In this model, people are free to associate or not and are ultimately accountable only for their own systems and their volunteered obligations to others.

However, this perimeter model is breaking down. There are five fundamental problems with the model. First, it is increasingly the case that organizations must support a variety of connections between people and machines that do not respect perimeter boundaries. For example, end user applications such as secure ecommerce and video conferencing, and server applications such as business partnerships, require live streams of data from point to point. Such point to point connections through a network are known as "tunnels", and networks are increasingly riddled with these holes. In general, network administrators cannot see exactly what is happening in these tunnels, and must rely upon assumptions about what programs are creating them and whether those programs are trustworthy.[2]

A second problem with the perimeter model is that it relies upon management of a network in terms of the computer addresses and network ports (virtual connection points used by computers) to which connections are permitted or denied. Unwanted or malicious software is able to spoof network addresses and take advantage of ports that are open for legitimate applications. This is similar to having identification checks at a national border – but using identification cards that can be readily forged.

[2] Although I disagree with some of her claims about public policy implications, Denning (1996; Denning & Baugh, 1999) presents excellent accounts of the basic issues with encrypted data transmission. Her analyses concern the contexts of public inspection and law enforcement, but they also apply to the activities of users as seen by network administrators.

A third problem is that apparently legitimate data may pass the perimeter inspection yet carry an unwanted component. This is the typical behavior of viruses that circulate as email attachments and word processing documents.

The fourth problem is that the perimeter model, in itself, neglects the fact that one of the largest threats comes not from computer intrusions but from an organization's own members. It is not the case that people or systems inside the boundary are necessarily trustworthy. One report found that 81% of corporate computer crimes are committed by employees (Brown, 1991); and problematic usage includes many things other than obvious crimes.

A fifth problem is that the virtual boundaries are themselves more imaginary than real; it is usually prohibitively difficult to deny all avenues of physical access to the "inside" of a perimeter-based system. Consider the case of a university or large corporation: there are typically network taps freely available at libraries, conference rooms, offices, and the like. In many cases, anyone can plug a machine into these and gain some level of access to the internal network. The growing use of wireless networks means that a physical connection or observable presence may be absent.[3]

An alternative to the perimeter model is one in which organizations assume that their network perimeters are porous, and focus instead on securing the information and systems that are important to them. This has been the approach for many years with virus detection software – it is most commonly used on end-point systems, not on network perimeter systems. Likewise, use of effective tools to manage end user and

[3] An interesting observation about wireless networking is that it is frequently regarded by organizations as a very serious security threat, despite the fact that traditional cabled access may be readily available in their environments. For example, I have heard university network administrators claim that wireless networking would be a security danger for their campuses – yet they have libraries and classrooms open to the public with freely accessible Ethernet taps.

server systems can help to ensure that dangerous applications are not running, that unnecessary computer services are turned off (such as a Web server on a desktop computer), and – with properly designed software – that data ingress and egress is to systems or people that are trusted (for example, using digital signatures).

When such management occurs throughout the network, we might speak of an *enmeshed* network, rather than a perimeter network. The defining characteristics of a system apropos the network would no longer be the boundary within which the system resides; rather, it would be the potential information flow to and from the individual system itself and the rules that permit or restrict that flow. It would be, in other words, an information-centric view, not an organization-centric view.

One philosophically interesting ramification of this shift is that it highlights the changing nature of identity for computer systems. The traditional, perimeter-based system was focused on three kinds of identity: the identity of a <u>user</u> as a member of an organization; the identity of a <u>machine</u> as a physical element in an organization's network (network address); and the identity of a <u>data stream</u> as reflected in the network port over which it travels. There were corresponding notions of accountability: users are accountable for what they do on their systems; the organization is accountable for the systems it maintains; and software providers are accountable for their applications maintaining integrity within the bounds of the ports and protocols they used

In an enmeshed network, identity is much more granular and dispersed. The identities above persist, but others are added. First, an end user machine is divided into components: the individual instantiation of the operating system is given an identity (a "machine account"); the user has an identity (user account); various elements of software have identities (software versions); each piece of software when running has an operational identity (service account); system components have various identities (such as serial numbers or electrical network addresses); discrete units of information (files and other data objects) have identities; and network connections themselves have identities (sessions). All of these are configured, changed, managed, logged, inspected, and otherwise subjected to rules that govern information flow.

This increased granularity in notion of identity can have both a salutary and detrimental effect on the concept of human identity. On the one hand, it clearly distinguishes between the people who use system and the components of the system; a user is not seen as equivalent (from the network point of view) to the machine that he or she is using. On the other hand, it reduces people to the same status as other elements of the network: users are entities with digital “identities” (i.e., codes) that have various properties and can be managed just like software or hardware.

One important implication is that users are increasingly unlikely to know about and be able to control the various identities that operate on their systems. A user’s system runs various pieces of software whose operation is entrusted to the vendor and to the network administrator. These services run under a variety of account names with varying system permissions that are unrelated to the account permissions granted to the user herself. Thus, the accountability of the user for the operation of her system is severely reduced.

A Changing Model for Software

For network administrators, the shift from a perimeter-based worldview to an enmeshed worldview has profound, if not immediately apparent, implications. On one account, little is changed: the users, computers, software, businesses purposes, machines, cabling, network equipment, and bandwidth may all be unchanged. However, the way in which these are *conceived,* and therefore managed, may be radically altered.[4]

We can think about this change as an alteration of the relationship of the network administrator to the various components that the network comprises. In the perimeter view, the network is a collection of objects that each serves a specific function in routing information flow, like plumbing: “these machines plug together here, and those go there, and we link them with this other device ….” In the enmeshed view, the network still has these pieces, but a more important element is the added *knowledge* embedded into the network. Instead of merely routing or blocking information, an enmeshed network distributes rules, categorical systems, and decision making throughout the network. Instead of being composed of interchangeable pieces, it is composed most essentially by the scheme that relates those pieces. The essential role of the system administrator is not to be a builder, but an architect.

Computer software must be conceived differently in these kinds of systems. A traditional view of software takes consumer products as analogues: software is a manufactured product that one buys and uses for some predetermined function or another. The revised view would be that network software is a platform that enables one to create and embed knowledge systems that link users, machines, and information sources.

[4] Thus, this is a shift not in the direct embodiment of technology, but rather in the hermeneutic relationship, the way in which the system is understood and the administrator’s role is conceived. Cf. Ihde (1991), chapter IV.

In considering specific software packages, this distinction becomes clear. A word processing package has a well-defined scope of operation along with core, immutable functionality, and it is clear what are its intended purposes and range of likely uses. By contrast, a server operating system is principally a platform that coordinates the operation of many other pieces of software, few or none of which are essential to it. A server operating system provides capabilities for creating and managing knowledge about systems and users – but does not in itself embed that knowledge or necessarily dictate its form or usage.

As this relationship to software changes, so do the relationships between software users and those who create the software. In the traditional, consumerist model where software has a very limited scope of operation, software can be engineered and produced just like any other product on the market. However, when software is a platform for embedding knowledge in dynamic systems, it is unlikely that the developers will be able to define the entire scope for the system. It becomes imperative to involve a large number of customers in close relationships, to engage them in the creation of a platform. Large software companies reflect both of these patterns: consumer software is developed largely in secret and then released with a marketing blitz, while network software is developed with continual input from hundreds of customers and is released in repeated, successive versions that permit customers to use and test it long before the product is officially complete.

Implications for Computer Ethics

I have already mentioned how this view of networks instantiates and alters views of identity. This poses a correlative challenge to notions of accountability. We have seen how the diverse identities within a system may undermine claims that an end user is accountable for the things running on his or her system.

It is more interesting to note that understanding networks as knowledge-based systems undermines claims that software manufacturers should be accountable for their products in a strict way. Nissenbaum (1997), for example, argues for a position of strict liability for computer software developers, but her analysis focuses on software systems that are developed, marketed, and operated as standalone systems. She notes the "problem of many hands" in terms of the development process, but neglects to notice that software is merely one part of sophisticated systems that are constructed by systems administrators. In fact, she tends to view systems as akin to persons: "[C]omputers perform tasks previously performed by humans in positions of responsibility. They calculate, decide, control, and remember." (ibid, p. 54). Such a view is unsustainable for systems that operate as enmeshed networks.

Does this eliminate responsibility for software manufacturers? No, of course not; it implies instead that responsibility is different than it would be in a consumerist, product-based model. Instead of insisting that software manufacturers should endeavor to create perfect systems – an impossibility for complex software – the focus must be on the relationship with customers. Given that software development cannot foresee every possible use and every possible attack, the responsibility is that developers act in accordance with reasonable standards of care and engage with customers to ensure continued care. In this way, accountability for complex software systems may be more akin to medical care than to consumer product design. The governing concepts are those of professional diligence and standards of care, not of perfection or foresight.

A recent case example can clarify some real-world implications of this shift. In January 2003, the so-called “SQL Slammer” worm affected systems that used versions of the Microsoft SQL Server software. This worm caused these systems to emit large numbers of copies of itself, directed to randomly addressed computers on the Internet. The volume of traffic slowed many Internet sites, and the worm rendered infected systems useless and unavailable for their usual operations. Much media attention focused on the fact that a patch had been available from Microsoft for many months that corrected this problem, and commentators implied that network administrators were remiss not to have installed the software patch.

We should note that this kind of analysis inherently relies upon outdated notions of software I outlined above: that software is a static, essential entity that is plugged together into a network; that perfect operation is possible; and that management of such a network consists in plugging the right pieces together. In fact, however, analysis suggests that the SQL Slammer worm was so prolific and verbose that even nearly perfect patching of all vulnerable systems would not have averted the problem.[5]

With an enmeshed view of the network, proper operation is not a property of any single system or deployed software package. It is important that systems work as well as possible, of course, but such operation should be reinforced by the knowledge embedded in the system. What does this mean in the case of worms like SQL Slammer? It means that the knowledge needed to prevent such occurrences should have several forms: vulnerable systems should be patched; unnecessary network ports to the Internet should be closed; unusual network traffic patterns should be detected and controlled automatically; and systems administrators should be available quickly to

[5] For an excellent analysis of SQL Slammer, see Graham (2003).

react to the system. The network is not a simple plan that is built once and then left to run; rather, it more like a living entity that has redundancy and is constantly monitoring itself and adjusting to the environment.[6] The challenge for software developers is twofold: to build better tools that allow management of these kinds of complex environments, and to shift to a relationship model (not a consumerist model) for engaging with customers in building and managing their systems.

The enmeshed network view affects notions of intellectual property as well. Clearly, in an enmeshed network such as I have described, there is intellectual value not only in the software and hardware elements of the network, but in the network architecture itself. In a hypothetical case: if a complex organization had all of its systems reset to "factory" state (without backup), while retaining all end user data, it would suffer a massive loss of function. It would be very difficult or impossible to restore the network to the prior state with precisely the same embedded knowledge about interconnections, rules, access permissions, and the like.

What is interesting about intellectual property in such a case is that it is unlike knowledge in many other human systems. For example, there is clearly knowledge embedded in an automobile and in humans who know how to drive them. However, in a computer system, the knowledge has a definite physical expression; it is not simply in the minds of the users and administrators. This suggests that analyses of computer

[6] This also implies something outside the scope of this paper: that the network enmeshes its operators (and even its users) in a system in which they must respond, but over which it may be increasingly difficult for them to exercise control. It would be interesting to see an analysis of enmeshed networks along the lines of Landon Winner's analysis of technology (1977).

intellectual property that focus exclusively on software packages[7] are neglecting potentially rich areas for exploration.

[7] Even “counter-culture” analyses of software intellectual property have an essentially consumerist, discrete package focus (cf. Stallman, 1985).

References

Brown, R. K. (1991). Security overview and threat (National Computer Security Educators Tutorial Track). Washington, DC: National Defense University. *Cited* in Kesar, S. and Rogerson, S. (1998), “Developing ethical practices to minimize computer misuse.” *Social Science Computer Review*, 16:3. Reprinted in Hester & Ford (2001), pp. 218-232.

Denning, D. E. (1996). “The future of cryptography.” Talk presented to the Joint Australian/OECD Conference on Security, Privacy, and Intellectual Property, 1996. Reprinted in Ludlow (2001), pp. 85-101.

Denning, D. E., and Baugh, W. E., Jr. (1999). “Hiding crimes in cyberspace.” Information, Communication, and Society, 2:3. Reprinted in Ludlow (2001), pp. 115-142.

Friedman, B. (1997). *Human Values and the Design of Computer Technology*. Stanford, CA: CSLI Publications.

Graham, R. (2003) “Advisory: SQL slammer.” Online at http://www.robertgraham.com/journal/030126-sqlslammer.html

Hester, D. M., and Ford, P. J., eds. (2001). *Computers and Ethics in the Cyberage*. Upper Saddle River, NJ: Prentice-Hall.

Ihde, D. (1991). *Instrumental Realism: The Interface between Philosophy of Science and Philosophy of Technology*. Bloomington: Indiana Univ. Press.

Ludlow, P., ed. (2001). *Crypto Anarchy, Cyberstates, and Pirate Utopias*. Cambridge, MA: MIT Press.

Nissenbaum, H. (1997) “Accountability in a computerized society.” In Friedman (1997), pp. 41-64.

Stallman, R. (1985). “The GNU Manifesto.” Free Software Foundation. Online at http://www.gnu.org/gnu/manifesto.html.

Winner, L. (1977). *Autonomous Technology: Technics-out-of-Control as a Theme in Political Thought*. Cambridge, MA: MIT Press.