# Network Management is a Mess

Paul Francis
Cornell University
May 24, 2005

Networks are managed by a hodge-podge of tools operated by Gurus. Networks are growing in size and complexity, including the networks in our homes, and the Guru model can't keep pace. SNMP, the one generalized network management tool standardized by IETF (as an afterthought to the basic architecture) can't even run until IP itself is up and running. While SNMP may have been an appropriate architecture when it was Developed (1989: pre-firewall, pre-tunnel, pre-NAT, pre-DHCP, pre-all-that-stuff-that-makes-IP-hard-to-bring-up), it has long-since been inadequate.

One broad approach to network management is to try to auto-discover, auto-configure, and auto-debug from the edges. While useful, this approach can ultimately go only so far. Fundamentally, it relies on the underlying network allowing discovery to take place. This has always been a risky proposition (routers, for instance, get in the way), and it can only get worse as managers increasingly batten down their networks. Further, the edge approach only allows discovery of elements that want to be discovered. In this approach, the network is dumb, and cannot itself recognize when, for instance, a rogue component has attached itself to the network.

We believe that, rather than try to attack the problem from the edges, the network (and we mean 'network' broadly and deeply here...routers and hosts of all kinds, up and down the protocol stack) must fundamentally be imbued with basic management capabilities. This thought was inspired by recent research at CMU and AT&T (and other places) called 4D that proposes an exciting new fundamental architecture for network management. 4D, which stands for "decision, dissemination, discovery, and data," provides a low-level topology discovery and routing substrate that runs immediately above the link layer. This substrate is used to discover physical topology and allow communications between network components and a network controller called the "Decision Element". Because 4D runs right above the link layer, it has no dependencies on lower-level configuration and so can auto-discover and configure the network from the ground up.

In their initial proof-of-concept experiment, the 4D researchers used 4D to dynamically configure the IP FIBs (in lieu of traditional routing protocols) and filters of routers. In one sense, what they did is to use the basic low-level communications channel created by 4D to allow what amounts to remote command-line configuration on the router (enable_iface, set_FIB_Entry, set_Filter_Entry). The idea is that the Decision Element is given a set of high-level directives by the network operators (i.e. a reachability matrix), and automatically produces the correct FIB and filter configurations. This notion could be extended to all aspects of router and host configuration (LAN, VLAN, EAP, RADIUS, MPLS, PPP, IPv4, IPv6, IP mcast, GRE, IPsec, L2TP, MIP, DNS, DHCP, etc.). While we believe this would be a useful step in the right direction, ultimately it would result in an overly complex and error-prone Decision Element. For instance, discrepancies between what the Decision Element thinks a given configuration command does and what it actually does would result in a misunderstanding of how the network is operating and would lead to errors.

A better way to extend 4D is to expand its basic notion of topology discovery into the logical structure of the network---a "Deep-4D". In other words, in addition to having 4D elements operate above the link-layer and communicate with peer 4D elements over the physical link, they could operate within every protocol driver. Each 4D element could then communicate with neighbor 4D elements in drivers above and below them in the same network box, and with peer drivers in other network boxes. If a driver is "plumbed" to another driver in the same box, this would be discovered and reported to the Decision Element. Likewise, if a tunnel is created between peer drivers in different boxes, this would also be discovered and reported to the Decision Element. The Decision Element could then decide whether to allow such connections, and if so configure appropriate filters and other settings. Perhaps more to the point, the Decision Element could discover which drivers in a network box could potentially be plumbed and tunneled, and establish the plumbing and tunnels itself.

While of course different drivers at different layers have different capabilities, we believe that it should be possible to identify a basic setof components and functions that all drivers potentially have. Namely, unidirectional connectors above, below, and across, filters associate with each such connector, and internal switches that bridge the connectors. Further, these drivers and connectors could have associated with them a small set of simple primitives: show potential, show actual, create, delete, and test. This set of primivites would allow a Decision Element to understand and configure the complete physical and logical topology of the network, while minimizing the number of gory details that it has to understand. This understanding could be extended at least up to the sockets interface, thus essentially allowing the Decision Element to know what applications can communicate, and allowing the network manager to describe the desired network operation in terms of high-level (application-level) reachability. Indeed it might be possible to extend this understanding into applications themselves, thus accounting for communications channels created by applications like email, instant messaging, and BitTorrent.

This vision of network management is extremely disruptive, ultimately requiring changes in most software components of every networking stack, and likely into applications themselves. We believe, however, that such disruption is necessary and long overdue. The TCP/IP architecture has never had a cohesive and comprehensive network management structure, and we are increasingly paying the price for this with networks that are costly to manage and unreliable. It's hard to see how to extricate ourselves from this mess without the level of disruption suggested by a Deep-4D network management architecture.