Evolving Toward a Self-Managing Network

Jennifer Rexford
Princeton University

Why is network management so difficult? I think the complexity of the individual network elements is a big part of the problem, and a big barrier to fundamental change. Today's routers implement numerous routing protocols (e.g., BGP, OSPF, IS-IS, EIGRP, and RIP) and data-plane mechanisms (e.g., class-based queuing, RED, and access control lists), with inumerable configurable parameters that have a profound influence on the behavior of the network. Despite years of research work on protocols and mechanisms, we have relatively few meaningful guidelines for (i) selecting and composing these features to build a network and (ii) setting the tunable parameters to maximize the performance, reliability, and security of a running network. Rather than managing "the network," operators actually manage individual network elements in low-level, device-specific configuration languages.

Clearly, we need to raise the level of abstraction to design and manage at the *network* level. But, how should we approach this problem? On the surface, building the abstraction on top of today's devices is an appealing approach. We can design a high-level language for specifying network goals and objectives, and then generate network designs and the low-level configuration commands that should be applied to the individual devices. However, the intrinsic complexity of today's protocols and mechanisms makes this approach extremely difficult (if not impossible) to succeed in practice. An alternative approach is to design the network elements, and their protocols and mechanisms, with network-level management in mind. The "design for manageability" approach starts by asking what kind of network-level abstractions we want, and then designs the device-level interfaces and protocols to support them.

Though conceptually appealing, the "design for manageability" philosophy runs head-first against the important practical problem of how to achieve any significant deployment in practice. I think that breaking the stalemate requires us to think creatively about evolutionary approaches to revolutionary change. The key is the sometimes tedious notion of "incremental deployability." To be incrementally deployable, a solution must have two key ingredients:

- **Backwards compatibility:** To effect change, we need to find ways to move forward without requiring a "flag day" where the legacy equipment and protocols are replaced. For example, approaches that retain the message formats of our existing protocols can allow substantive change while still accommodating legacy equipment. Ethernet is a great example of this principle -- significant change has occurred in the past decade or so while (and arguably *because of*) remaining compatible with the legacy specification of the message format.

- **Incentive compatibility**: Each step along the way to the target end state needs to offer substantive benefits to the early adopters, and additional incentives for the remainder of the players to join the party. The need for incentive compatibility is so strong that it might dictate the choice of steps and, in some cases,even the target end state itself. A sound treatment of incentive compatibility requires accurate models of the cost of deployment and the benefits at each stage of adoption. The construction of the steps along the way is arguably a research problem in its own right.

As an example, the Internet's interdomain routing system is widely viewed fraught with difficult management problems. The Border Gateway Protocol (BGP) is hard to configure, slow to converge, prone to serious anomalies (e.g., persistent oscillation, forwarding loops, and black holes), vulernable to malicious attack, difficult to troubleshoot, and overly sensitive to small topology changes. Building meaningful abstractions on top of such a system is fundamentally hard. The research community has made progress in creating static-analysis tools for detecting configuration errors, checking if a collection of routing policies are vulnerable to routing anomalies, and predicting the effects of configuration changes on the flow of traffic. Other research has created tools for analyzing measurement feeds of BGP update messages to detect and diagnose routing problems. These contributions have significantly improved our understanding of BGP and our ability to "work around" some of its limitations. However, raising the level of abstraction for BGP has remained elusive. Moreover, having a "flag day" to replace BGP with a new protocol is not viable in practice. We cannot simply "reboot the Internet."

Yet, BGP has one key feature that makes real change tantalizinglypossible: any system that sends BGP messages to a router in the appropriate format can tell the router what routes to use. This enables us to change everything about interdomain routing, while still speaking to the legacy routers in terms they can understand. In our work on the Routing Control Platform (RCP) [1], we exploit this observation and propose a three-step evolution to a new interdomain routing architecture:

1. Path selection in a single domain: In the first phase, the RCP has internal BGP (iBGP) sessions with the operational routers in a single Autonomous System (AS). This requires just a small configuration change on the routers (to exchange iBGP messages with the RCP, rather than one another), and enables the RCP to make customized routing decisions for each destination prefix on behalf of each router. This phase enables more flexible traffic engineering and network maintenance, and allows the AS to avoid routing anomalies such as protocol oscillation, forwarding loops, and black holes by explicitly enforcing correctness constraints. These capabilities provide a powerful incentive for an AS to deploy the RCP even if other ASes have not.

2. Flexible routing policy in a single domain: In the second phase, the RCP has external BGP (eBGP) sessions directly with the border routers in neighboring ASes. This requires the neighboring domains to make a small change to the configuration of the eBGP sessions on their border routers to exchange BGP messages with the RCP. As a result, the RCP has complete control over the

sending and receiving of BGP messages, as well as the policies for path selection and export. The operational routers no longer have any BGP configuration state, except for the iBGP sessions to the RCP. This phase enables the use of new policy specification languages, intelligent route-flap damping, minimization of the number of routing-table entries on the routers, and many other applications.

3. Redefinition of interdomain routing: In the third phase, ASes coordinate interdomain routing directly through their RCPs. This requires the participating ASes to run an interdomain routing protocol between their RCPs, while still communicating with legacy routers via iBGP. Although the RCPs could conceivably run a policy-based, path-vector protocol like BGP, they need not. For example, a new routing protocol could attach prices to advertised routes or explicitly support inter-AS negotiation. RCPs could also base their routing decisions on measured end-to-end performance, as proposed in work on overlay networks, and even make the performance statistics available to end-host overlays through appropriate interfaces. The RCP could also be used to deploy an interdomain routing protocol with better security properties than BGP.

Each step offers strong deployment incentives by simplifying network management and enabling new services, while remaining backwards compatible with the installed base of routers. (In addition, experiments with our prototype implementation [2] show that the RCP is feasible, from a systems perspective; the RCP can be made fast and reliable enough to make routing decisions for a backbone network with hundreds of routers.) If the RCP approach is successful, future routers could be built with much less control software, and with new dissemination protocols for communicating with the RCP (rather than continuing to use iBGP for this purpose).

Stepping back from the specific example of BGP and the idea of the RCP, I think that making significant progress in improving network management requires changes in the division of labor between the network devices and the management systems. Making these changes actually happen requires us to grapple with backwards compatibility with the legacy equipment (e.g., by finding ways to use the existing protocols and message formats to coax the devices) and to identify compelling incentives for incremental deployment (e.g., by solving real problems and enabling new applications for the early adopters).

[1] Nick Feamster, Hari Balakrishnan, Jennifer Rexford, Aman Shaikh,and Jacobus van der Merwe, **The case for separating routing from routers**," *Proc. ACM SIGCOMM workshop on Future Directions in Network Architecture*, August 2004. http://www.cs.princeton.edu/~jrex/papers/rcp.pdf

[2] Matthew Caesar, Donald Caldwell, Nick Feamster, Jennifer Rexford, Aman Shaikh, and Jacobus van der Merwe, **Design and implementation of a Routing Control Platform**, *Proc. Networked Systems Design and Implementation*, May 2005. http://www.cs.princeton.edu/~jrex/papers/rcp-nsdi.pdf