The Cutting EDGE of IP Router Configuration

Don Caldwell, Anna Gilbert, Joel Gottlieb,
Albert Greenberg, Gisli Hjalmtysson, and Jennifer Rexford

AT&T Labs–Research; Florham Park, NJ
{dfwc,agilbert,joel,albert,gisli,jrex}@research.att.com

Abstract— **Human error in configuring routers undermines attempts to provide reliable, predictable end-to-end performance on IP networks. Manual configuration, while expensive and error-prone, is the dominant mode of operation, especially for large enterprise networks. These networks often lack the basic building blocks—an accurate equipment inventory, a debugged initial configuration, and a specification of local configuration policies—to support the holy grail of automation. We argue that *migrating an existing network* to automated configuration is a rich and challenging research problem rooted in data analysis and in the modeling of network protocols and operational practices. We propose a novel, *bottom-up* approach that proceeds in three phases: (i) analysis of configuration data to summarize the existing network state and uncover configuration problems; (ii) data mining to identify the network's local configuration policies and violations of these policies; and ultimately (iii) boot-strapping of a database to drive future configuration changes. The first stage reduces the number of errors, the second normalizes the local policies, and the third prevents new errors and reduces the manpower needed to configure the network. We describe the architecture of our EDGE tool for steps (i) and (ii), and present some examples from our experiences applying the tool to several large enterprise networks.**

I. Introduction

Predictable, reliable network performance depends on the correct and consistent configuration of the routers. Configuring the routers is one of the most important and complex aspects of running a large network. Manual configuration, while time-consuming and error-prone, is extremely common, and may be responsible for many of the outages and anomalies that occur in practice. Automated provisioning improves on manual configuration by generating the router configuration commands from a set of rules applied to a database that models the network and its services. Although these kinds of systems are relatively common in large service provider networks, comparable automation in enterprise networks remains elusive, despite the fact that many of enterprise networks are

- quite large, sometimes spanning multiple continents,
- managed in a decentralized fashion, due to mergers and acquisitions or geographic boundaries,
- running a mix of network protocols (IP, IPX, Appletalk) and routing protocols (RIP, EIGRP, OSPF, BGP), and
- operating under strict reliability requirements (e.g., for carrying retail, financial, or medical data).

Without a provisioning database and well-codified rules, network administrators often continue the risky practice of manual configuration or embark on a time-consuming redesign of their networks with automation in mind. Instead, we argue that a detailed analysis of the existing router configuration data can drive the migration of the network toward automated provisioning.

A. Router Configuration is Hard

Several factors conspire to make configuring IP routers extremely challenging:

High degree of configurability: Routers implement a vast array of protocols and mechanisms, with many tunable options. Network administrators must configure the routing protocols, interfaces, access control lists, QoS mechanisms, and services such as SNMP, NTP, and NAT. For example, modern routers implement numerous routing protocols such as RIP, EIGRP, OSPF, IS-IS, BGP, MPLS, and multicast protocols, each with dozens of configurable options (e.g., timer values, link weights, route filters, route injection between routing protocols, and session establishment). As another example, interfaces have many parameters at the IP level (e.g., address, mask length, and packet filter) and Medium Access Control level (e.g., encapsulation, framing, and error checking).

Complex, low-level configuration languages: The configuration languages designed by router vendors are essentially a collection of thousands of "assembly language" commands that each provide a small piece of functionality. An operational router in a large IP network may have several thousands of lines of commands. Some work has considered vendor-neutral constructs for certain aspects of router configuration, such as routing policy [1]. In addition, some router vendors offer Web-based interfaces for specifying and applying configuration commands. However, generic data models and high-level languages for router configuration do not exist and are likely to remain elusive for some time (see the charter for the new IETF Network Configuration working group [2]).

Rapid changes in router features: IP protocols and equipment continue to evolve, and administrators periodically enable new functionality in their networks. Over the past few years, IETF activity has led to several completely new functions, such as MPLS routing, Random Early Detection (RED), and differentiated services. Existing protocols such as BGP have seen a number of extensions to add new configurable options (e.g., soft-reconfiguration and extended communities). Router vendors have periodic releases of their operating systems. Any data models, languages, or configuration tools would need sufficient extensibility to accommodate this rapid change.

B. Manual Configuration is Bad

Despite the many challenges, moving beyond manual configuration is crucial because:

Manual configuration is error-prone: Configuration mistakes can cause network outages, degradation in performance, and security vulnerabilities. For example, installing the wrong packet filter, filtering valid routes, advertising an incorrect block of IP addresses, or assigning the same IP address to multiple pieces of equipment, can lead to reachability problems. As another example, a link cannot carry traffic if the two ends belong to different OSPF areas or apply a different CRC technique. A mistake in specifying or installing a packet filter may leave a network open to denial-of-service attacks. A recent study estimates that half of network outages stem from human configuration error [3]; similar results have been found in studies of Internet services [4]. Analysis focusing specifically on BGP routing suggests that configuration errors are responsible for many network anomalies [5].

Manual configuration is expensive: Manual configuration leads to costly mistakes and expensive delays in enabling service for new users. Network administrators may be forced to limit or delay upgrades and introductions of new features, protocols, and services. In addition, manual configuration requires hiring and training a potentially large number of skilled engineers. In practice, network administrators often take lengthy certification courses to acquire the necessary skills to configure the routers, as well as on-the-job training to learn the local networking policies of their organization. These engineers may need additional training in how to diagnose and fix configuration problems. The investment in engineers trained on a particular router product can make it extremely difficult to incorporate products from other vendors.

C. Moving Beyond Manual Configuration

The problems inherent in manual router configuration have recently attracted a great deal of commercial attention. Existing products and services focus on:

Reducing the complexity of manual configuration: Several kinds of products help administrators in their manual configuration tasks. GoldWireTech [6] supports authentication of engineers connecting to the routers, archiving of past versions of configuration files, logging of human keystrokes, and scheduled pushes of configuration commands. Other tools like WANDL's IPAT [7] and OpNet's NetDoctor [8] generate reports of the network topology and configuration errors. Individual router vendors provide Web-based interfaces for configuring the routers to reduce the likelihood of typographical errors, without moving the network toward greater automation.

Automating for new networks/services: Products such as Orchestream [9] support automated configuration of new services, such as Virtual Private Networks. These products tend to "start from scratch" in configuring the new service, and they often have proprietary internal databases that are difficult to integrate with other parts of the provisioning process (such as ordering, billing, network inventory, and work-flow management).

Automating the tuning of specific parameters: Other tools focus on specific operational tasks, such as traffic engineering or mitigation of Denial-of-Service (DoS) attacks. For example, Cariden's MATE [10] and OpNet's SP Guru [11] products support tuning OSPF costs or MPLS Label Switched Paths to the prevailing traffic, and ArborNetwork's PeakFlow DoS [12] product detects DoS attacks and generates filters to block the offending traffic. While very useful for specific tasks, these tools focus on a small portion of the configuration state.

Although these kinds of tools are useful, we argue that the *complete* configuration of the network should be driven from an automated provisioning system. New networks can, and should, be designed with this approach in mind. However, network administrators rarely have the luxury of starting from scratch. Instead, we argue that migration of existing networks is an important part of the configuration problem. The next section discusses the target design of a provisioning system and describes our three-phased approach to boot-strapping a manually-configured network. Our approach is deeply rooted in analysis of the configuration state of the existing network. In Section III, we describe the software architecture of our EDGE (Enablement and Debugging of Growing Enterprises) tool for analyzing the configuration data, and present examples based on our experiences with several large enterprise networks. The paper concludes in Section IV with a discussion of directions for future work.

II. Automating Router Configuration

In the automated provisioning systems common in service provider networks, all configuration changes flow through a database using explicit local rules. However, for many existing enterprise networks, the routers' configuration state *is* the "database"[1]. We believe that a detailed analysis of this configuration state should play an important role in the move toward automation.

A. Target Automated Provisioning System

In large service provider networks, automated provisioning typically follows the high-level approach in Figure 1. Each configuration task, such as adding a new router or a new customer, depends on input data collected using a technical questionnaire (TQ). For example, adding a new customer connection might require information about the customer's name, address blocks, media type and speed, and the desired service. The TQ details depend on the specific task, and vary from network to network. In addition, other identifiers such as interface names and access-control list (ACL) numbers may need to be assigned. The chosen identifiers may depend on past assignments; for example, a new customer may be assigned the next available slot in the router and the next available ACL identifier for a packet filter.

The TQ data and the identifiers are fed into a database

[1] Yet, this "database" has no mechanism for normalization. In a real database, a policy (e.g., for packet filtering) might be defined once and then used again and again. Under manual configuration, the policies are instantiated separately on each router, and tend to diverge as the network evolves, whether or not this is the administrator's intent.

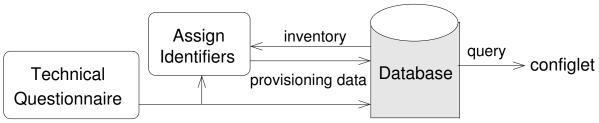

Fig. 1. End-to-end view of router provisioning process

that stores key information about the network equipment and the existing configuration. Rather than storing the raw configuration commands, the database stores the information necessary to *generate* the sequence of commands—a *configlet*—to apply to the router. Like the TQ, the database schema would vary across networks, depending on the key parameters underlying the services they provide. The configuration commands are generated by applying rules based on the contents of the database[2]. For example, a Serial interface may be configured to apply a particular framing technique. In practice, the rules can be much more complicated. For example, the IP addresses owned by a BGP-speaking customer may drive the configuration of both the packet filter on the interface and the route filter on the BGP session. An example of an automated provisioning system for BGP is presented in [13].

B. Bootstrapping an Existing Network

Despite the conceptual appeal of Figure 1, many enterprise networks don't have explicit TQs, provisioning rules, or an underlying database. In some cases, an informal TQ exists and the rules are codified in English text; however, these documents are often out-of-date or incomplete, and administrators do not always adhere to them. Administrators may have a list of the equipment in the network but not necessarily a complete record of how the routers are configured. In essence, *the network itself is the database*. This observation suggests a *bottom-up* approach to enabling automation. We argue that network administrators should start with a snapshot of the configuration state of the network—a single configuration file for each active router—and proceed in three stages:

1. **Generic analysis:** Analysis within and across configuration files enables us to reverse-engineer the router topology and summarize the status of the network. As part of this process, we identify configuration mistakes that the network administrators can fix. This offers immediate value in terms of boot-strapping a simple inventory database and improving the configuration of the network. Several recent tools provide this kind of service [7, 8, 14].
2. **Data mining:** Further analysis seeks to identify distinct patterns in the configuration that reflect the local design choices of the network administrators. The output of this process is a set of inferred rules and the exceptions to these rules. For example, if 99 of 100 routers have the finger daemon disabled, the inferred rule would be "the finger daemon should be disabled" and the exception would be the one non-compliant router. This facilitates interaction with the engineers to formalize and fine-tune these rules, and fix the violations. This offers immediate value by alerting network administrators to notable inconsistencies within and across routers.
3. **Database design:** At the end of the second stage, the network has a correct configuration that is consistent with the complete, fully-codified set of rules. In essence, the network *looks* as if it has been automatically configured. The TQ and database schema can be designed, and the database can be populated from the state of the existing network. Going forward, the automated system can drive future configuration changes.

Ultimately, these steps require human intervention to fix configuration mistakes, vet and fine-tune the rules, and design the TQ and database. Still, we can expedite the discovery and design with suitable tools for rapid and detailed analysis of the configuration data. The next section describes our EDGE prototype for the first two stages.

III. EDGE Analysis of Configuration Files

A detailed analysis of router configuration data can provide a wealth of information about the state of the network. This section presents a brief overview of the software architecture of our EDGE tool, and examples of the kinds of analysis the tool performs to aid network administrators in building an inventory database, identifying configuration mistakes, visualizing the network design, and reverse-engineering the local configuration policies.

A. EDGE Software Architecture

The EDGE software runs daily on tens of thousands of configuration files from dozens of large enterprise networks. In designing the software, we focused on scalability (to support a large number of networks and configuration files) and extensibility (to support the rapid addition of reports on different technologies used in enterprise networks). The tool starts with a snapshot of the configuration file for each router in the network and produces a collection of Web reports, as illustrated in Figure 2:

Configuration data feed: EDGE operates on a snapshot of the configuration state from Cisco routers, or other routers that emulate the Cisco IOS (Internet Operating System) command set. Dumping a router's configuration file involves applying the `show running-config` command at the Telnet interface or issuing a `get` via the SNMP interface. In many cases, the network administrator has a backup server storing these files. EDGE checks the configuration files into a CVS repository to store only the differences. This substantially reduces the overhead of storing the data and also facilitates our analysis of the changes to the network over time.

Parsing of configuration files: The parser incorporates knowledge of Cisco IOS syntax, such as command *modes* that define the various sections of the configuration files (e.g., `interface` and `router bgp`). The output of the parser is a Perl hash table that can be easily read by subsequent parts of the software. Rather than understanding the syntax of *every* command, the parser focuses on

[2] In practice, the configlet might be generated in two steps, starting with some vendor neutral form followed by a mapping into the command syntax necessary for a particular router.

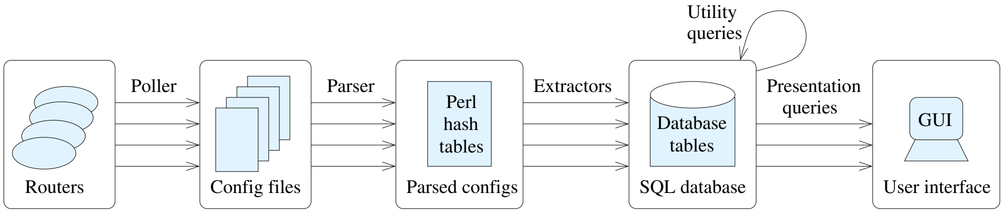

Fig. 2. Flow of configuration data from the routers through the database to the Web-based reports

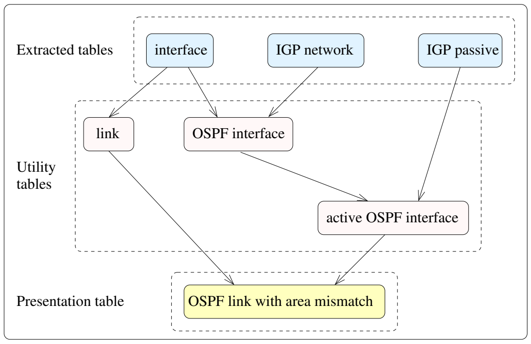

Fig. 3. Example OSPF report based on joins of multiple tables

commands that occur frequently in practice and figure in the analysis in subsequent parts of the software. Configuration commands that the parser does not understand are annotated as unparsed features that can be analyzed by other software as needed. The parser can be (and has been) extended over time to parse additional commands without requiring changes to the rest of the software.

Extraction into database tables: Simple extraction routines pull information from the Perl hash representation into database tables. These tables capture important information underlying the queries that analyze the configuration data. For example, the *interface* table stores basic, layer-3 information about each interface such as router name, interface name, IP address, mask length, description (comment field), and status (shutdown or active). As another example, the *IGP network* table lists all subnets participating in each routing protocol instance on a router; each row includes a router name, routing protocol (e.g., OSPF, EIGRP, or RIP), the area or Autonomous System, the subnet address, and the subnet mask. Rather than trying to model the entire router configuration state, these tables are designed as needed to support the queries.

Joining the database tables: Specific features of EDGE are supported as utility queries on the database. Multiple queries may be necessary to generate a single report. For example, consider the problem of identifying links with end points configured in different OSPF areas. As illustrated in Figure 3, this requires referencing the *IGP network* table that identifies the subnets participating in OSPF on each router. By joining this table with the *interface* table, we identify which interfaces on each router have addresses that fall in these subnets. Some interfaces may be configured as *passive* and thereby excluded from OSPF. The *interface* table is also used to group interfaces into *link*s based on their network addresses. By combining the *active OSPF interface* and *link* tables, we can check whether the remote end of an interface (i) exists, (ii) participates in OSPF, and (iii) has the same OSPF area. Despite the need for multiple joins, each step is a simple SQL query.

Generating Web reports: The final stage presents tabular reports and network diagrams on the Web for EDGE users. Users navigate presentation queries by topic (e.g., inventory, routing, etc.) and select reports to display. A presentation query is an XML file that includes the necessary SQL statements to generate the tables. The Web page also displays the output of the visualization software [15] to show different views of the network (e.g., network topology, BGP sessions, OSPF routers/links, etc.). The EDGE Web site serves as a sort of "portal" for the network administrators to track the status of their networks and identify configuration errors.

B. Building a Network Inventory Database

Analysis of the configuration files provides information about the equipment, topology, address allocations, and the specific commands applied to the routers.

Equipment: For each router, the configuration file specifies the interfaces by slot position and media type. We use this information to populate a simple inventory database that stores basic information about each router (loopback address, operating system version, slot allocation) and interface (media type, IP address, network address).

Commands: Having a list of the commands and their usage frequency is useful for knowing which router features are important to the successful operation of the network. The list can also drive testing in the lab to ensure that the commands work correctly before upgrading the production routers with new versions of the operating system. In addition, we report commands used very infrequently as a possible sign of inconsistencies in configuring the routers.

Addresses: Managing the allocation of IP addresses is a challenging part of operating a large network. We use the configuration data to summarize the addresses and subnets that are assigned to equipment (router loopback and other interfaces), connections to neighboring domains (packet filters, route filters, and static routes), and network services (NTP, SNMP, measurement collectors).

Topology: The network topology consists of routers and layer-3 links. A link consists of two or more interfaces in the same subnet [16]. For example, the prefix 10.1.2.116/30 consists of two interfaces with addresses

10.1.2.117 and 10.1.2.118, as well as the network and broadcast addresses. Looking across configuration files enables us to join the interfaces with the same network address. In other cases, the link may be an ATM or Frame Relay PVC that is specified in the configuration file.

C. Generating Configuration Errors and Warnings

Combining information from different parts of the configuration files can uncover mistakes and poor practices.

Referential integrity: Many commands refer to items defined by some other command. For example, an access control list (ACL) defined in one place may be referenced elsewhere to instantiate a packet or route filter. By maintaining a list of definitions and references by type, we report an error when an undefined item is used and a warning when a definition is not used. We construct and extend our list of commands by analyzing existing configuration files and inspecting vendor Web pages.

Duplicate IP addresses: Having multiple active interfaces with the same IP address can cause significant disruptions in communication. In addition to reporting duplicate addresses, we identify cases where interfaces on the same link have different network addresses (i.e., mask lengths), since inadvertent inconsistencies can lead to reachability problems that are difficult to diagnose.

Routing protocol integrity: Routing protocol configuration combines information within and across configuration files. We perform local checks such as identifying the interfaces participating in each protocol and ensuring that routing attributes (such as OSPF cost) are only assigned to participating interfaces. Our global checks verify that all end-points of a link participate in the same protocol and match in key attributes (such as OSPF area); otherwise, the link does not actually participate in routing.

Violation of best common practices: Additional checks can generate warnings when the configuration does not follow “best common practices.” For example, we check that two routers with an internal BGP session communicate using their loopback addresses, rather than specific interfaces, for robustness to link failures. We also generate warnings when the configuration relies on default parameters, since the administrator may have unintentionally neglected to configure a value. For example, router vendors typically assign interfaces a default OSPF cost (e.g., inversely proportional to link capacity), but depending on the default may be unwise.

D. Visualizing the Network Design

EDGE creates accurate network diagrams, which provide immediate insight into not only the layer-3 topology but also the routing architecture and misconfigurations.

Topology layout: Automatic layout tools are essential to effective visualization. We adapt undirected graph layout techniques to visualize the network topology, where vertices or nodes in the graph represent routers and edges are the layer-3 links. We employ layout techniques that impose a model where each edge is a spring of the desired length and the desired Euclidean distance between two vertices is their graph-theoretic distance [17]. The layout algorithm then tries to find the placement of the vertices and edges that minimizes the potential energy of the system, which places closely related vertices near each other.

Network design: Routers and links participate in various protocols and have certain attributes, such as OSPF area or BGP AS number. Our layout method can place all routers in the same OSPF area in the same region, with, for example, the backbone area (area 0) at the top or center of the picture. For more complex networks with multiple routing protocols, the layout incorporates the role played by each protocol, and the boundary points between the protocols. For example, OSPF or EIGRP may be used as an intradomain routing protocol in several regions, with BGP sessions providing connectivity between the regions. We use a variety of visual cues to convey network structure. For example, many enterprise networks have a hub-and-spoke topology, with one or more hubs; options for coloring the nodes can be used to distinguish the network core (hubs) from the edge (spoke).

Graph compression: Large enterprise networks can have hundreds of routers and thousands of links. We apply three types of graph compression to reduce complexity while still capturing the key features of the network design. First, with vertex clustering, routers with the same network or physical attribute (e.g., OSPF area, BGP AS, or geographic location) are represented by a single vertex. Second, rather than representing shared media (such as Ethernets or FDDI rings) as cliques in the graph, we cluster these edges and replace them with single edges to a separate distinguished vertex (such as an Ethernet hub). This form of compression is just one criterion for edge clustering. Third, we use edge collapse or vertex removal to define the inherent structure of our network. This allows us to remove low-degree vertices (typically access routers) that are linked to high-degree vertices (typically hub or backbone routers) while retaining the basic network structure.

E. Extracting the Network’s Configuration Policies

Data mining can be used to identify patterns (and exceptions to the patterns) in the configuration files. This analysis is a first step in codifying a network’s policies.

Graph mining: The representation of the network as a graph (where nodes and edges have numerous attributes, such as link capacity and OSPF weight/area) can drive various kinds of clustering analysis. Nodes and links can be compared in terms of their similarity. For example, the analysis of a dual hub-and-spoke network might show that nearly all spoke nodes connect to two hubs, drawing attention to an isolated router that does not have a backup connection to the second hub. Analysis of the OSPF area attribute might show that nearly all links belong to area 0, except for one isolated link that is assigned to area 1. Analysis of the OSPF link cost might suggest that links to the primary hub have cost 100 while links to the backup hub have cost 200.

Feature mining: A more detailed analysis of the configuration commands can help in identifying the *templates* that should drive certain aspects of router provisioning. For example, the vast majority of the routers might have the finger daemon disabled, suggesting that this is part of a base configuration for initializing the routers. Similarly, all frame-relay interfaces might be configured with a particular kind of encapsulation or framing. These rules can be relatively complex and depend on the presence or absence of other commands. For example, a network might assign an OSPF cost to all backbone interfaces that are not administratively shutdown. Identifying these kinds of patterns require an effective way to generate and evaluate various candidate rules against the configuration data.

Common definitions: Often, the same definitions of ACLs, routing policies, and other variables appear on multiple routers. For example, BGP sessions to upstream providers may have a small set of basic routing policies. EDGE extracts and compares the definitions across files. The definitions may be equivalent (representing a base part of the router configuration) or have subtle inconsistencies. For example, the same name (e.g., import policy PROVIDER_IN) may be defined in one way on 99 routers and slightly differently on another router; the one exception may be a configuration error. Similarly, two names corresponding to the same definition (e.g., import policy PROVIDER_OUT and PROV_OUT) may be represent an opportunity to normalize the configurations.

Provisioning use cases: Analyzing a network's configuration across time provides a way to identify specific provisioning tasks that need to be included in the automated provisioning system. For example, adding a new BGP neighbor may proceed in several distinct steps: (i) adding an edge link, (ii) associating a BGP session with that link, and (iii) specifying routing policies for the session, with explicit testing after each phase. When these practices are applied repeatedly, they leave a clear signature in the configuration data. In addition to aiding the design of the provisioning system, making these tasks (and the individual steps) explicit is important for subjecting the processes to critique and, ultimately, to streamlining.

IV. Conclusion

Manual configuration of IP routers is expensive, time-consuming, and error-prone. Automated provisioning is the holy grail but does not address the very real problem of getting from here to there. In this paper, we propose a *bottom-up* approach based on a detailed analysis of the router configuration state of the network. Our EDGE tool follows this approach and provides valuable information to the administrators of several large enterprise networks. Still, challenging research problems remain:

- **Network modeling:** Router configuration spans an immense variety of complex network protocols and mechanisms. Designing models that accurately represent the low-level configuration state of each of these technologies would be extremely useful, both for the data analysis and for the eventual automation of router configuration.
- **Best common practices:** Developing a set of best common practices (BCPs) for each technology (and the interaction between technologies) would be invaluable to network administrators. This deeper understanding of BCPs could drive queries that generate warnings when these practices are violated and give administrators a sense of the goodness of their network design.
- **Data mining:** Uncovering the local rules for configuring the routers depends on applying effective data-mining techniques to the relatively flat models of the configuration state. New analysis techniques that operate on graph representations, sets of related database tables, or the raw configuration files would all be quite valuable.
- **Data modeling:** The third stage in our bottom-up approach involves designing the Technical Questionnaire and the provisioning database. These tasks would benefit greatly from innovation in automating portions of the design process. This stage will probably always require human intervention but any techniques for expediting the design process would greatly simplify the problem.

Although the networking community has created a variety of flexible protocols and mechanisms for routers, relatively little attention has focused on helping network administrators manage the resulting complexity. We believe there is substantial scope for research in this area.

References

[1] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, and M. Terpstra, "Routing Policy Specification Language (RPSL)," RFC 2622, IETF, August 1999.

[2] Network Configuration Working Group. http://www.ietf.org/html.charters/netconf-charter.html.

[3] Z. Kerravala, "Configuration management delivers business resiliency." The Yankee Group, November 2002.

[4] D. Oppenheimer, A. Ganapathi, and D. A. Patterson, "Why do Internet services fail, and what can be done about it?," in *Proc. USENIX Symposium on Internet Technologies and Systems*, 2003.

[5] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP misconfiguration," in *Proc. ACM SIGCOMM*, August 2002.

[6] Gold Wire Tech. http://www.goldwiretech.com/.

[7] WANDL IP Analysis Tools. http://www.wandl.com/html/ipat/IPAT_new.cfm.

[8] OPNET NetDoctor. http://www.opnet.com/products/modules/netdoctor.html.

[9] Orchestream Service Activator. http://www.metasolv.com/MSLV/CDA/General/ProdSrvs_FocusDetail/1,2543,17%,00.asp.

[10] Cariden MATE Framework. http://www.cariden.com/products/.

[11] OpNet SP Guru. http://www.opnet.com/products/spguru/home.html.

[12] Arbor Networks Peakflow. http://www.arbornetworks.com/products_sp.php.

[13] J. Gottlieb, A. Greenberg, J. Rexford, and J. Wang, "Automated provisioning of BGP customers," *IEEE Network Magazine*, vol. 17, November/December 2003.

[14] A. Feldmann and J. Rexford, "IP network configuration for intradomain traffic engineering," *IEEE Network Magazine*, pp. 46–57, September/October 2001.

[15] E. R. Gansner and S. C. North, "An open graph visualization system and its applications to software engineering," *Software–Practice and Experience*, vol. 00, no. S1, pp. 1–5, 1999.

[16] F. Baker, "Requirements for IP Version 4 Routers," RFC 1812, IETF, June 1995.

[17] T. Kamada and S. Kawai, "An algorithm for drawing general undirected graphs," *Information Processing Letters*, vol. 31, pp. 7–15, April 1989.