Towards Automated Network Management:
Network Operations using Dynamic Views

Xu Chen   Z. Morley Mao   Jacobus van der Merwe
University of Michigan   AT&T Labs–Research
{*chenxu,zmao*}*@umich.edu*   *kobus@research.att.com*

ABSTRACT

We analyze data from a Tier-1 ISP that reflect the dynamic operational tasks performed in the ISP network to build a holistic view of configuration management operations. We observe that in addition to commands that lead to *persistent* configuration changes, virtually all management tasks also involve *status-checking* commands that do not change the configuration, but allow the operator to verify some router specific state. Based on this observation we model configuration modifications using automatically generated deterministic finite automata (DFA), where a state represents the configured behavior of an interface and an edge indicates the operations performed on the interface either to fulfill a specific task or to check the status of the router. The DFA model captures common configuration tasks derived by our analysis and their ordering and dependency. We argue that composing DFAs is a (small) step towards enabling operators to reason about the operational state of the network, as well as enabling tools for automated network management.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations

General Terms

Management, Measurement

Keywords

TACACS, Network management automation, DFA

1. INTRODUCTION

Managing IP networks is increasingly challenging due to diverse protocols, growing application requirements and primitive support from network devices. The state of the art in network configuration management tends to be template-driven and device-centric [1], with inadequate focus on network-wide objectives. These approaches usually focus exclusively on the configuration management aspects of network management, *i.e.,* the task of generating the persistent configuration files that dictate the behavior of the network elements making up the network. Indeed, much of our understanding of network management stems from analyses of such **static** configuration files obtained from operational networks [2, 3, 4].

We approach this problem by analyzing the TACACS (Terminal Access Controller Access-Control System) [5] logs obtained from a Tier-1 ISP. The network is configured to perform authentication and accounting through TACACS and as such **all** configuration commands executed on network elements in the ISP network are logged. We perform a bottom-up analysis to present a holistic view of the operational tasks performed in the ISP network over a four-month period. To our knowledge, it is the first time that such a dynamic view of network management has been reported.

In addition to the static view provided by snapshots of the network configuration files, we analyze the *dynamics* presented in the TACACS logs of how the configurations are modified across time. We perform detailed analysis of configuration commands by correlating commands using an interface-centric view. The starting point of our work is to extract the command sequences that are usually executed together in a fixed pattern to fulfill management tasks. From the data we also observe that in addition to commands that lead to *persistent* configuration changes, virtually all management tasks also involve *status-checking* commands that do not change the configuration, but allow the operator to verify some router-specific state.

Based on these observations, we model configuration modifications using automatically generated deterministic finite automata (DFA), where a state represents the configured behavior of an interface and an edge indicates the operations performed on the interface either to fulfill a specific task or to check the status of the network. The DFA model not only allows us to capture common management tasks, but also gives us the ordering and dependency information among those tasks. Containing the information about the temporal progression of network management under different network conditions, the DFA model provides a dynamic view of how large networks are managed today. Based on this understanding, better tools for automating network management can be built. We argue that composing DFAs is a better network management abstraction, which enables operators to reason about the operational state of the network.

2. METHODOLOGY

In this section we describe the data sources and the data processing required to perform our analysis of current operational practices in the ISP we studied.

2.1 Data Sources

We used three data sources in our study. The main data source is TACACS logs, containing the commands executed on the routers across the ISP network. TACACS is a remote authentication protocol used by network devices to communicate with authentication servers, and, in our case, to determine whether a user has access to a certain router and has sufficient privilege to execute a command. As such, the TACACS logs contain three types of records for login requests (authentication), privilege escalation (authorization) and commands executed (accounting) respectively.

The second data source we used is daily snapshot of the configuration of all the routers in the ISP network. TACACS data capture how the router configuration is modified across time. We combine the two data sources to form a continuous view of how the configuration of the network evolves over time. Configuration data follow the same syntax and structure as TACACS data and can be processed using the same parser.

Finally, we make use of data derived from configuration files [2, 6] to augment the first two data sources. Specifically, because we take an interface-centric view for our analysis, we use this data to indicate the *role* of an interface in the network, *e.g.,* backbone link versus customer link *etc.*

2.2 Data Pre-Processing

The data pre-processing steps involved in our study are detailed below. We are particularly interested in TACACS accounting entries, each of which contains several useful fields: `username`, `router-id`, `terminal`, `timestamp`, `task-id`, `command`, *etc.*

Ordering Commands Sequentially: The raw TACACS logs are in the form of large text files consisting of the records for all the routers saved by multiple TACACS servers. The `task-id` is a monotonically increasing counter inside each router, which can be used to uniquely identify an executed command. We first separate the commands executed on different routers, according to the `router-id` field. For a particular router, we sort the commands based on both the associated `timestamp` and the `task-id` fields, since `task-id` is initialized to zero when the router is reset to original factory default settings by command `reload`.

Login Session Extraction: There can be multiple operators logged into one router and each operator can have multiple simultaneous login sessions. We first extract out the commands which have the same `username` and `terminal` pair and then demarcate them according to special entries that flag the creation and termination of login sessions. Thus, we separate the commands executed across different login sessions on the same router.

Command Differentiation: Commands that inspect router status are commonly performed. We differentiate the commands that do not change the router's configuration or running status as *status-checking*, for example `show running-config`, as opposed to those *persistent* commands that actually modify the configuration or behavior of a router. The status-checking commands usually serve as the purpose of condition checking in the current network management practice to determine whether and how to proceed with the next configuration step, which we will discuss in detail later.

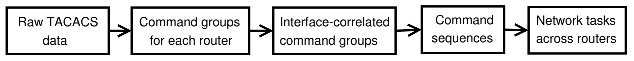

Raw TACACS data
Command groups for each router
Interface-correlated command groups
Command sequences
Network tasks across routers

Figure 1: The overall processing diagram

Command Parsing: We developed a parser for processing configuration commands. In the current implementation, we tailor our parser for Cisco router syntax[1]. We created regular expressions to match the commands in the data. Although we currently focus on Cisco syntax, our overall approach is also applicable to other configuration syntax. Our parser is capable of parsing over 98% of the persistent commands[2], and all status-checking commands observed can be parsed.

2.3 Command Group Generation

Different categories of commands, *e.g.,* related to access-list, interfaces, or BGP, are executed under different operational **contexts**. Once an operator logs into a router, she is under a normal context, under which only status-checking commands are allowed. Once `configure terminal` is executed, she switches to a configuration context, under which the configuration modifications are performed. To configure the interface `serial1/0:0`, she needs to execute `interface serial1/0:0` first, to switch to an interface configuration context. Each category has its own context-switch commands.

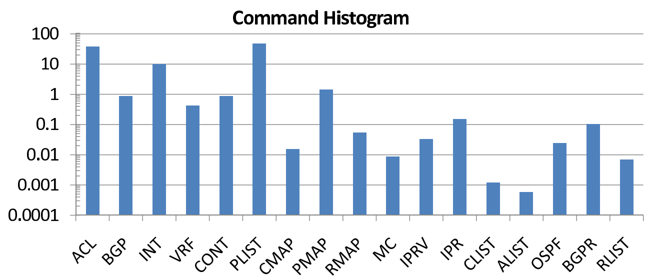
We define a **command group** to be a sequence of persistent commands that are consecutively executed under the same context, plus the proceeding context-switching commands. The context-switching commands of a command group not only specify its category (*e.g.,* interface, access-list), but also contain *variable names* (*e.g.,* the interface name), which together uniquely identify the configuration component that the command group modifies. Combining individual consecutive commands into command groups is a natural representation of the TACACS data, giving the information about the how different components of the router configuration are modified. The analysis results are shown in Section 3.1.

3. DATA ANALYSIS RESULTS

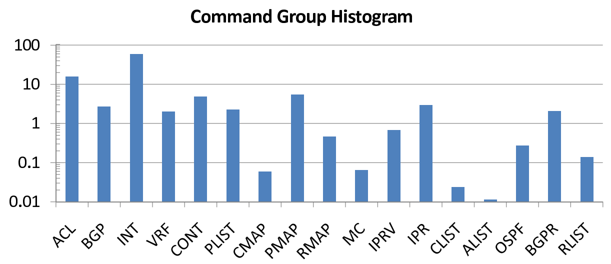
In this section, we first show our analysis results of the break-down of command groups that appeared in the four-month TACACS data and then describe the additional processing performed. Figure 1 depicts our processing steps. After command group generation, we associate each command group to appropriate interfaces, while preserving their temporal ordering. This gives us the ordered sequence of command groups executed for each interface, from which high-level network management tasks can be inferred.

[1]We also normalized all commands to accommodate different variations of the same command, *e.g.,* "int" is parsed to "interface".

[2]The remaining 2% of persistent commands are all hardware-specific commands related to specific interface types in the provider network and are not germane to the focus of our study.



(a) The breakdown of configuration commands



(b) The breakdown of configuration command groups

Figure 2: TACACS data analysis results

ACL	`access-list` group and `ip access-list` group which define ACLs
BGP	`router bgp` group, which define BGP sessions
INT	`interface` group, defining interface configurations
VRF	`ip vrf` group, which defines VRF profiles
CONT	`controller` group, which defines controller setups
PLIST	`ip prefix-list` group, which defines prefix-list that is used to filter routes
CMAP	`class-map` group, which defines a class of packets
PMAP	`policy-map` group, which defines the traffic-shaping policy of certain classes of packets
RMAP	`route-map` group, which defines how to manipulate certain routing messages
MC	`map-class` group, which encapsulates policy-maps and can be directly applied to interfaces
IPRV	`ip route vrf` group, defining VRF static routes
IPR	`ip route` group, defining static routes
CLIST	`ip community-list` group, which defines filters based on community values for route-maps
ALIST	`ip as-path access-list` group, which defines filters based on as-path for route-maps
OSPF	`router ospf` group, defining OSPF routing process
BGPR	`clear ip bgp` group, which resets BGP sessions
RLIST	`ip receive list` group, which uses an ACL to filter received packets of the router

Table 1: TACACS command group types

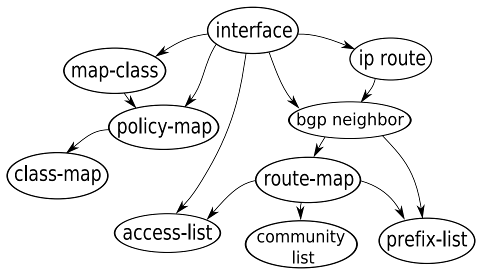


Figure 3: Correlation among command groups

3.1 Command Groups in the TACACS Data

Using our parser we processed four months of TACACS data containing the commands executed on all the core-backbone and majority of provider-edge routers within a Tier-1 ISP network. We extracted commands from different login sessions and generated command groups according to the change of context during execution. We excluded from this analysis a few command categories related to authentication configuration as well as low-level class-of-service commands. Additionally, we ignore 0.01% of the total number of TACACS commands for which we are unable to determine the execution context because of missing context-switching commands.

Figure 2 shows the histogram of the number of commands and the number of groups observed in the data respectively, broken down by the command group types shown in Table 1. The Y-axis is the log-scaled percentage value for their appearance. In Figure 2(a), we can see that the majority of the commands executed on routers are ACL (39%) and PLIST (47%), which correspond to access-lists that filter packets and prefix-lists that filter BGP routes. This is expected as access-lists and prefix-lists are the first line of defense for a network's data plane and control plane respectively. These command groups tend to have many entries, especially for a prefix-list that filters routes from a peering ISP, which can contain thousands of entries. Interface groups contribute to the third largest number of commands and around 60% of the total number of groups. As shown in Figure 2(b) the interface components are the most frequently modified in network management, as most of them are directly connecting to neighboring networks. The second and third largest number of groups come from access-list (16%) and policy-map (5.6%). The importance of policy-map also emerges, since it is used intensively to guarantee QoS.

3.2 Interface Correlation

The dependencies within router configurations have previously been studied by Feldman *et al.* [2] to check static configuration errors. In our work, we extend this dependency relationship to identify correlations among configuration command groups. We define command groups A and B to be **correlated**, if one command in A contains a variable name which is exactly group B's name. For example an interface group with command `ip access-group 123 in` is correlated with `access-list 123` command group. Figure 3 shows some possible correlations across different command groups. This graph is automatically generated by processing specification of commands in each category. The only exceptions are the arrows from interface and IP route to BGP neighbor group - this means the IP addresses used for BGP peering connections can be reached either by being directly connected to an interface or via static routes that point to an interface. We correlate BGP session setup to interface, based on the assumption that each customer link is used to connect to only one BGP neighbor.

We developed an algorithm to correlate command groups in TACACS data to related interfaces. This **interface correlation** is trivial for interface command groups. For a non-

Description	Num. of occurrences	Pctg of the total commands
ACL MOD	58825	36.6%
PREFIX MOD	2677	42.2%
STATIC INIT	10761	6.23%
VRF INIT	4156	1.3%
VRF ENABLE	3982	1.34%
BGP MOD	1795	4.77%
STATIC ENABLE	5605	3.30%
BGP ENABLE	464	0.36%
RECV MOD	638	0.55%
TOTAL	91447	96.7%

Table 2: Major command sequences

interface command group, we first try to correlate it to interface command groups executed on the same router within a two-day time window (from one day before to one day after) in the TACACS data. This is an adjustable heuristic taking advantage of the fact that closely executed command groups are very likely to correlate with each other. If this step fails, we then try to correlate the command group to non-interface command groups within the same time window, whose correlated interface can be determined by recursively calling the algorithm. If both steps fail, we take into consideration the command groups in recent configuration snapshot file. For an interesting pattern like an *ip prefix-list* group followed by a BGP session reset, our algorithm is able to identify the BGP session that uses the prefix-list and further the interface that is used to connect to the BGP neighbor with the help of configuration snapshot file. Note that one command group can be correlated to multiple interfaces. For instance, modifying an ACL that is used by two interfaces.

After correlating configuration command groups to interfaces, we have formed another level of abstraction of the TACACS data - *i.e.,* the configuration commands that are executed in order with respect to a single interface. Our interface-centric correlation is very effective: over 99% of the TACACS commands are successfully correlated to one or more interfaces. Our algorithm is not able to correlate a command group like policy-map which is defined but not referenced by an interface within the two-day time-window. This limitation accounts the 1% commands that are uncorrelated. We further analyzed the break-down of the commands executed on different edge interfaces - connecting to VPN, static-router and BGP neighbor. The notable differences are: 1) VPN customers are more likely to have policy-maps applied; 2) over 80% of the commands related to BGP neighbors are prefix-list modifications; 3) over 80% of the commands related to static-route customers are ACL modifications.

3.3 Command Sequence Extraction

Command groups that are correlated to one interface are usually executed together in fixed patterns, which we define to be a **command sequence**. It is likely that the operators have some automated ways of executing them together to fully or partially fulfill a single operational task. From the data we automatically generate frequently executed command sequences and then infer their purpose using domain knowledge.

Table 2 shows the result for sequence extraction. The major sequence "ACL MOD" is an event in which the ACLs

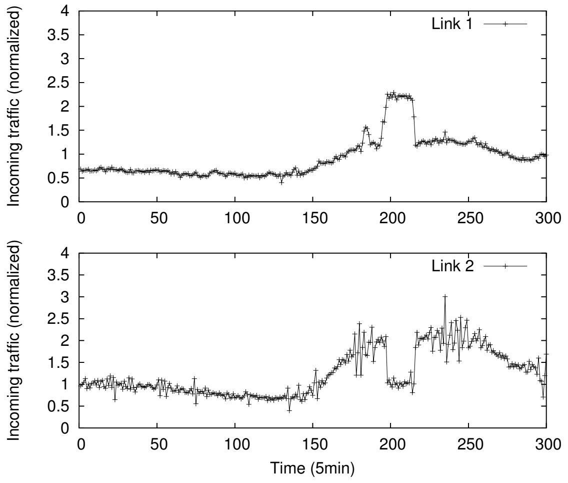


Figure 4: Correlated events across network

of the interface are modified, while "PREFIX MOD" means modifying the prefix-list used by a BGP session, followed by a BGP session soft reset. "BGP MOD" is the event in which the prefix-list of the BGP session and the ACL of the correlated interface are modified at the same time. "STATIC INIT" is a sequence of commands which initializes an interface (consisting of a controller group that allocates the sub-interface, an access-list group, and an interface group). The sequence of "VRF INIT" is similar to "STATIC INIT", but with additional MPLS VPN related setup. "STATIC ENABLE", "BGP ENABLE" and "VRF ENABLE" correspond to the sequences that finalize the configuration for an interface that connects to static-route customer, BGP neighbor or VPN customer respectively. There are four types of short but important command sequences not shown in the table - "SHUT DOWN" the interface (`shutdown`), "BRING UP" the interface (`no shutdown`), "REMOVE INT" by deleting the interface configuration and de-allocating the sub-interface if necessary, and "DIAG INT" for diagnosing problematic interfaces.

Command sequence is an interesting level of abstraction for understanding current network management. According to our domain knowledge, a command sequence can be translated to a specific task performed on a router related to a specific interface. For edge routers in particular, managing such an interface is conceptually equivalent to managing the customer site connected to the interface. Note that these nine command sequences cover almost 97% of the commands executed during our study period, representing the majority of the network tasks being performed.

3.4 Network-wide Event Correlation

Managing large networks sometimes requires configuring multiple routers simultaneously. For example, inter-domain traffic engineering usually needs to change the BGP routing policy of two or more BGP sessions between two ASes. Reflected in TACACS logs, BGP MOD sequences on those interfaces are likely to co-occur.

When extracting command sequences, we also know the interface being configured and the times when the configuration change happened. We did a very preliminary study by counting the number of appearances of two events happening together within five-minute time windows. The most frequently occurring pair is the BGP policy change of two

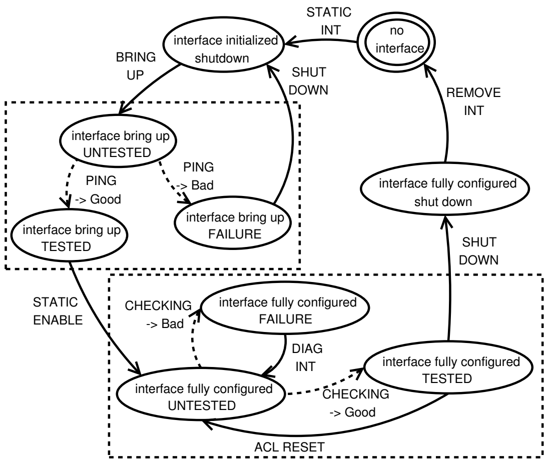

STATIC INT
no interface
interface initialized shutdown
BRING UP
SHUT DOWN
REMOVE INT
interface bring up UNTESTED
PING -> Good
PING -> Bad
interface bring up TESTED
interface bring up FAILURE
interface fully configured shut down
SHUT DOWN
STATIC ENABLE
CHECKING -> Bad
interface fully configured FAILURE
DIAG INT
interface fully configured TESTED
interface fully configured UNTESTED
CHECKING -> Good
ACL RESET

Figure 5: Sample DFA for static-routed interface

peering links connecting to one neighboring ISP. The traffic data near those events shown in Figure 4 clearly shows the combined effect of both command sequences.

4. MODELING NETWORK CONFIGURATION MODIFICATIONS USING DFAS

In this section, we propose a DFA model to create a unique view of network management. Our model is able to integrate the essential operational tasks, in terms of command sequences, as well as the *ordering* and *dependency* among them. We argue that this is a neat and compact way of modeling current network management practice and represents a step towards network management automation. We will briefly explain the generation of the DFA model and discuss its properties and usage.

4.1 DFA Model

DFA stands for deterministic finite automaton [7], which consists of a finite set of states and transitions that depend on input symbols at each step. We use DFA to model the status of an interface - how it is being configured and how it actually works. Figure 5 gives a simple example of our automatically generated DFA for an interface connecting to a static routed customer.

Given a router configuration at any time, we can find all the components that are correlated to a specific interface, using the same correlation algorithm described in Section 3.2. All these correlated configuration groups, which we denote to be **interface-correlated configuration** of an interface, exactly define how this interface *should* work.

We define a **configuration state** to be one possible interface-correlated configuration. We analyzed the router configuration snapshot data to identify those configuration states. Given that a snapshot file can be generated at any time, we do not consider an interface-correlated configuration in that file if the corresponding interface was being configured across the time when that particular snapshot file was taken, since the interface is likely to be in an intermediate state. After extracting configuration states, each consisting of several command groups, we canonicalize them by ignoring the variable names and parameters of commands to facilitate comparisons. We denote a canonicalized form of a configuration state to be a **state template**. Interestingly, we found that a very small number of state templates can capture all the configuration states for interfaces connecting to a particular type of customer, *e.g.,* static-route customer. In Figure 5, the lower-cased words within a state is the high-level description of that state template.

In our design, each edge of the DFA corresponds to either a persistent command sequence that changes the configuration or running-status, or a status-checking command sequence that helps determine the actual running-state of the interface. Persistent sequences and status-checking sequences are marked respectively as firm and dotted arrows in Figure 5.

Status-checking commands are very important in network management as they reveal the *actual* running-status of the router or an interface. Similar to persistent commands, status-checking commands are executed together in fixed patterns. We thus use similar techniques to generate status-checking command sequences. Note that status-checking commands can be executed periodically to inspect network running-status. We ignore those periodically executed commands by using a simple heuristic of ruling out commands that are executed roughly at the same time every day. Instead, we focus our study on the ones that are closely coupled with the execution of persistent command sequences.

An operator may proceed with different persistent commands, after observing different results of status-checking commands. In the example we have, after an interface is initialized (STATIC INIT) and brought up (BRING UP), the operator will continue to execute the finalization sequence (STATIC ENABLE), if the status-checking sequence reveals the interface is working properly, but shut down the interface (SHUTDOWN) instead if the interface is found to be not working properly. As we can see in Figure 5, the states within one dashed block have the same state template, while it is the status-checking commands that determine how the interface actually works. ACL MOD is a command sequence that changes the traffic filter of an interface. It does not change the canonicalized form of the configuration state, so the edge connects two states with the same state template as well.

We define three main types of running-status (marked in capitalized words): 1) UNTESTED, which means a configuration change has just been made, while the interface is pending some status-checking sequence; 2) TESTED, which means the result of the status-checking command sequence reveals that the interface is running properly; 3) FAILURE, which means the interface is not working properly. However, such information can only be obtained by interpreting status-checking commands' results, which are not logged by the TACACS data. There are still two ways that we can estimate these states: i) according to the subsequent configuration sequence - for different states with different running-status, the next-step configuration is likely to be different; ii) using our domain knowledge to estimate these possible states.

4.2 Discussion

The DFA model that we developed not only provides a way to visualize all the operations related to a particular interface, but also gives much more information about the temporal progression in network management. The current configlet-based approach [1, 8] does not provide the whole picture of network management. As we are able to derive directly from TACACS data, there exists interesting ordering and causal dependency in the execution of configuration commands.

The DFA model is currently inferred from the data analysis results; however, we can design similar DFAs to assist in network management automation. The persistent command sequences (outgoing edges) of each state define exactly what can be done given a particular state of the interface, guiding the next step in configuration. The condition checking can be automatically performed to immediately verify previous execution results and trigger the subsequent persistent configuration changes. By encoding possible network running-status into a group of states with the same configuration state, we can achieve more sophisticated network management automation using the DFA model. For example, we can define two states, "Interface fully-configured, has traffic going through during the last minute", "Interface fully-configured, no traffic going through during the last minute". We can then clearly specify that the command sequence which shuts down the link can only be executed under the latter state. This can prevent undesired traffic disruption and ensure that the network maintains a healthy running state. We leave designing such a system as future work.

5. RELATED WORK

Managing networks through router configuration is a challenging task and a significant contributor to lack of high network availability [9]. Many existing studies in this area of configuration management in IP networks focus on diagnosing network-wide misconfigurations from router configuration files [3, 10, 2, 4]. Our work is different from previous work in that we analyze data capturing continuous changes of the configuration of most routers in the network. Our approach provides more fine-grained analysis and reveals many interesting issues that are impossible to be discovered by studying static configuration snapshots.

Researchers have also worked on automatically generating configuration commands that can be directly applied to routers to ensure peering policies or fulfill network management tasks [6, 8, 11, 1]. Those generated configlets are highly related to the command sequences that we automatically extracted from the TACACS data. In our work, we develop the DFA model to capture *how* these configuration modifications are applied to the routers, identifying the essential steps and the dependencies among the steps. This contributes to understanding requirements for automated network management.

6. CONCLUSION

In this paper, we analyzed a TACACS data source consisting of all the commands executed on many routers within a Tier-1 ISP network. Starting from low-level raw data abstracted to high-level correlations, we developed a way to summarize the high-level network operations that are performed on the network. This is the first concrete study revealing the dynamic network management activities in real networks.

Today's network management is greatly eased by automatically generated configlets which can be translated from high-level policies and then applied to the routers directly. However, we found that configlets are usually applied to the routers through a sequence of carefully designed steps, which are usually causally dependent on each other or on the network's running-status. We developed a DFA model to characterize such dynamics in network management, which we believe is an important step towards automated network management.

7. REFERENCES

[1] W. Enck, P. McDaniel, S. Sen, P. Sebos, S. Spoerrel, A. Greenberg, S. Rao, and W. Aiello, "Configuration Management at a Massive Scale: System Design and Experience," in *Proceedings of Usenix Annual Technical Conference*, 2007.

[2] A. Feldmann, "Netdb: IP Network Configuration Debugger/Database," tech. rep., AT&T Research, July 1999.

[3] N. Feamster and H. Balakrishnan, "Detecting BGP Configuration Faults with Static Analysis," in *2nd Symp. on Networked Systems Design and Implementation (NSDI)*, (Boston, MA), May 2005.

[4] F. Le, S. Lee, T. Wong, H. S. Kim, and D. Newcomb, "Minerals: Using Data Mining to Detect Router," in *ACM Sigcomm Workshop on Mining Network Data (MineNet)*, September 2006.

[5] C. Finseth, "An Access Control Protocol, Sometimes Called TACACS." RFC 1492, July 1993.

[6] D. Caldwell, A. Gilbert, J. Gottlieb, A. Greenberg, G. Hjalmtysson, and J. Rexford, "The Cutting EDGE of IP Router Configuration," in *Proc. Workshop on Hot Topics in Networks (HotNets)*, 2003.

[7] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. July 2006.

[8] J. Gottlieb, A. Greenberg, J. Rexford, and J. Wang, "Automated provisioning of BGP customers," *IEEE Network Magazine*, 2003.

[9] D. Oppenheimer, "The Importance of Understanding Distributed System Configuration," in *Proceedings of CHI*, April 2003.

[10] G. Xie, X. Jibin, Z. David, A. Maltz, H. Zhang, A. Greenberg, G. Hjalmtysson, and J. Rexford, "On static reachability analysis of IP networks," in *Proc. IEEE INFOCOM*, 2005.

[11] H. Boehm, A. Feldmann, O. Maennel, C. Reiser, and R. Volk, "Network-Wide Inter-Domain Routing Policies: Design and Realization," in *draft*, April 2005.