Policy Units and Categories: Computer Networking Models for Simplifying Network Management

An Honors Thesis (CS 499)

by

Madeline Van Ness

Thesis Advisor

Dr. Xin Sun

Ball State University
Muncie, Indiana

May 2019

Expected Date of Graduation

May 2019

Abstract

This paper proposes new models for simplifying policy management in enterprise networks. The application of these models to five operational networks has demonstrated their capacity for abstractly modeling network policy structures and their potential for simplifying network management tasks, which would result in reduced opportunity for human error in network management. This is useful because human error is currently a significant cause of data breaches in enterprise networks, so a reduction in human errors would be greatly beneficial to network security. Specifically, we have created two new models for abstracting policies. The policy unit model divides a network into sections, called policy units, by grouping hosts together according to similarities in the existing policy rules that are applied them. This model allows policies to be viewed and analyzed at a higher level of abstraction, which would increase the efficiency of policy management (e.g., rule changes can be applied to an entire policy unit at once). The category model groups policy units together, giving the potential for even greater efficiency. Analysis of the application of these models to five enterprise networks exposed new insight in policy management in production networks, and showed that policy management in large networks could be greatly simplified through the use of these models. This indicates the potential benefit of integrating our models in future network management systems.

Acknowledgments

I would like to thank my research advisor, Dr. Xin Sun, for guiding me through the research processes involved in the research for and leading up to my thesis paper, for taking me to research conferences with him, and for all the support he has given me through my college career.

This thesis is based upon the research project led by Dr. Xin Sun, which is funded by the National Science Foundation (NSF) under Grant CNS-1660569, and also by Ontario Systems through the NSF Security and Software Engineering Research Center (S^2ERC). Any opinions, findings, conclusions, or recommendations expressed in this thesis are those of the author and do not necessarily reflect the view of NSF, Ontario Systems, or S^2ERC.

Process Analysis

The following research paper documents the creation and analysis of two new models, policy units and categories, that can be used to simplify the task of network administration. Being a paper about digital networking, there are some terms and concepts that the lay reader may be unfamiliar with, which I will briefly describe here. In a computer network, each computer, each server, and all other network-enabled devices (hosts) each have a designated address within the local network they are connected to, called an Internet Protocol (IP) address. This IP address is used to distinguish each device from the other devices in the network, so that digital messages can reach the intended recipients. (This system is often likened to the postal system, with IP addresses being digital equivalents to mailing addresses.) Messages that are sent from one host to another travel through various intermediary devices, such as routers, switches, and firewalls. Routers and switches check the destination IP address of the intended recipient of the digital message and send that message in the direction of its recipient, and routers and firewalls can be configured with rules as to which messages may be passed on, and which will be denied passage. (To continue the postal analogy, the routers and switches functions like post offices.) Networks are often divided into sections, called Virtual Local Area Networks (VLANs), according to policy distinctions (which messages are or are not allowed). This is particularly useful for grouping together hosts in a network that are physically separated from one another, so that they can be treated as a unit as though they were physically together in the network, with their digital messages flowing through the same switches and routers. This can be useful when, for example, a department in an office building has hosts scattered throughout the building, rather than being grouped together into a designated area of the building.

Network administrators are responsible for making sure that the network functions smoothly and securely, and a network's security can be boosted by setting restrictions on which hosts may communicate with which other hosts within and without the network. Some reasons for setting these restrictions might be to reserve a printer for usage only by a company's HR department, or to make sure that a university's grading servers may only be accessed by its faculty members, or to prevent network users from visiting malicious websites. These rules are implemented through Access Control Lists (ACLs), which are text files that consist of a list of individual rules for routers and firewalls to follow to decide whether to send a particular message to its designated recipient or to reject the message and not send it to its recipient. Each rule in an

ACL file (a policy, although some policies can be made of multiple rules) specifies a sender or group of senders and a recipient or group of recipients, and states whether to allow the sender(s) to send messages to the receiver(s). In practice, these ACL files can grow incredibly large, which can make maintenance of the rules difficult and prone to errors, which can in turn lead to vulnerabilities in a network's security.

The inspiration for the paper's research was a desire to help simplify the task of network management by providing an elevated level of abstraction through which to provide an automated, network-wide model of a network's ACL files in order to present them in a more compact and understandable format. The research process began with the acquisition of network configuration data and ACL files from several production networks, supplied by my research advisor Dr. Sun, alongside some previously-developed Perl scripts for deconstructing the network configuration into more easily usable parts, such as lists of the network's VLANs and their IP ranges. This previous research proposed the idea of the Policy Unit model my thesis discusses, and had developed scripts for forming policy units from a network's configuration. My research started with writing scripts to analyze these policy units in relation to the networks they came from, and in relation to each other, and I then visualized the data gathered from these analyses by making graphs.

After this analysis, Dr. Sun and I decided that it may be beneficial to look at not only policy units based around the hosts that were sending information (the kind analyzed in the previous research), but also policy units based around the hosts that were receiving information. In order to extract these two kinds of policy units from the network configuration files, I had to write my own set of scripts. These scripts were particularly challenging to write. Firstly, I needed to record an entire network's worth of configuration data into one multidimensional data structure so that individual configuration rules could be dynamically added as they were read in from the ACL file, and then later each part of the data structure could be compared to the rest of the parts in the structure for similarities from which to formulate the actual policy units. Next, implementing this multidimensional structure proved to be its own challenge because the language I used, Perl, does not support multi-dimensional data structures, so in order to contain an entire network's worth of configuration information in one data structure, I had to formulate a series of one-dimensional data structures that each referenced the data structure of the next level down (and because Perl data structures can only contain one type of data, I also had to create a

data-storage object for one of the data structures in order to be able to contain multiple types of data (integers and pointers) in a single data structure). After writing the scripts and executing them on the sample production networks, I analyzed these newly-generated policy units from each network just as the previous policy units had been analyzed, and made graphs to visualize the results.

With the policy unit model having been analyzed, discussions with Dr. Sun led to the proposal of a further level of abstraction for networks, the category, and I wrote scripts to generate categories from a network's policy units, analyze these categories in comparison to their networks' other attributes and to each other, and to generate graphs of the results. This model also required some creative thinking to determine the manner in which policy units would be grouped together into distinct categories; I ultimately developed an algorithm that would generate the smallest possible number of categories from the policy units in a network, to maximize the potential efficiency of the model.

1. Introduction

One major concern among enterprise networks is security, and if network management can't keep up with, and outpace, hackers, they run the risk of major data breaches, such as those of Yahoo, who lost the data of three billion users [1], [2], and Sony, who lost substantial quantities of employee information [3]. These breaches, and others like them, resulted in the loss of private data belonging to the companies and their users and employees, cost them time and money in researching the breaches and handling the technical, legal, and publicity aspects of the aftermath of the breach, and reduced users' trust in the security of their data with these companies. Another issue these networks face is unintended network downtime, which disrupts the userbase and prevents employees from being able to do their jobs; the greatest cause of this issue is human error in network management [4]. Across the board, issues resulting from faulty or insufficient network management are detrimental to all involved parties, so it is important to consider how we can improve network management techniques and tools to reduce the likelihood of such errors occurring.

This paper aims to try to help solve these issues in network management and security by exploring new abstractions and models for automated, systematic analysis of network security, and their potential value for integration into other research in this area to be applied to help

network administrators better do their jobs.

One of the proposed new models is the policy unit. Policy units are groups of network addresses that are subject to the same policy treatments from their network's security rules, such as access control rules and firewall rules, which can therefore be logically treated as a unit when considering changes to those security rules. Being able to handle network hosts as units in this manner could prove useful to network administrators by reducing the opportunities for human error when rules need to be changed. Presently, rule changes are applied to one host at a time [5], leaving plenty of room for errors of omission, where a host or two get skipped by the network administrator and therefore do not receive the rule change, and errors of inaccuracy, where the rule change gets mis-typed for one or more of the hosts. If a rule change only needs to be applied to one policy unit in order to affect all the desired hosts, such errors would be far less likely, and the network administrator's job would be much quicker and less tedious.

Another new model, which is a further abstraction from policy units, is the category. If one were to view each policy unit as a set of hosts that is produced from an intersection of two larger sets, then categories are those larger sets that intersect to produce all of the policy units of a network. In more practical terms, categories are broader groups of network addresses that might be handled with similar network policy rules; for example, one policy unit might cover faculty members within the physics department in a university network, and this policy unit would be produced by the intersection of the physics and faculty categories within that network. By applying this further abstraction of categories to the policy unit model of network administration, the modification of network policies could be further simplified: rather than having to apply the same change to each faculty policy unit within each department separately, that change could instead be applied once to the faculty category, which would further streamline and simplify the process of making such modifications to the network, beyond what could be achieved with just the policy unit model.

In our research, we have developed the concepts of policy units and categories, and have applied these models to real-world networks to analyze the applicability of the models, and the effects produced by their application. In order to validate our models and study their applicability, we have collected complete configuration snapshots from five operational enterprise networks. Each snapshot includes the configuration files pulled directly from all the networking devices in a network. The application of the models to our sample networks involved

the development of algorithms to extract and analyze existing network policies from the configuration files, and then fit and restructure the existing data into the framework of our models. We then examined the networks through the lens of our models to inspect the characteristics revealed, such as similarities and differences in distribution of policy units and categories across the networks, and trends in sizes of policy units and categories, and how well they align with network-level host grouping schemes that are currently in use (virtual local-area networks).

Our findings from our sample networks shoewd that policy units came in a wide range of sizes, but most were either large (upwards of 500 IPs) or small (ten or fewer IPs). Among the sample networks, the smallest observed policy units contained only one IP each, and the largest contained billions of IPs each, while most policy units contained between 100 and 100,000 IPs each. Additionally, it was found that the number of VLANs (Virtual Local Area Networks) in a network seemed to have little to no bearing on the number of policy units in that network. The number of IPs in a network also seemed to have only a minor influence on the number of policy units in that network; the size of a network and the complexity of that network therefore seem to be largely unrelated to one another. With the application of categories, we found that the number of categories in a network tended to be significantly smaller than the number of policy units in that network, especially in networks that had many policy units: one network in particular had more than 30 policy units, but fewer than 10 categories. However, it should be noted that networks with three or fewer policy units did not benefit from the application of the category model, as these networks had at least as many categories as policy units. This shows that networks with very simple policy layouts might not benefit from the addition of categories, despite the major benefits provided when applied to networks with complex policy layouts.

By applying the models of policy units and categories to networks, the task of network management can be simplified, which would reduce the chances for human error. The application of policy units would allow for individual hosts in a network to be grouped together, which would greatly reduce the number of steps needed to modify a network's policy rules, as opposed to applying a new rule to each individual host one by one, and the application of categories would allow policy units to be grouped together as well, which would further reduce the complexity of making changes to the network. This reduction in complexity would make the task of network management simpler and faster, which would lead to increased efficiency and

decreased human error in the management of the networks.

This work introduces and explores policy units and categories as models to be applied to networks, and analyzes the specific potential benefits of their application, but this is only the first step towards their introduction to the sphere of network management and security. Further efforts would go towards refining the models, perhaps further analyzing the effects of their applications to networks, and ultimately implementing them in a usable manner for usage by network administrators.

2. Related Work

Network management as a field tends to rely on software solutions for managing configurations, security, and performance, and to handle network faults, and while the software used for managing networks has been advancing along with network technology and usage requirements, it has been falling behind the curve [6], leading to increased reliance on manual, human operation over automated, digital operation. The task of network management can be made more difficult by the field's lack of comprehensive tools, leading network managers to have to rely on a set of software components that are specialized for various specific tasks, and are not necessarily compatible with one another [7], [8]. One proposed networking software solution divides networks into separately-managed domains [9]. The policy unit and category models proposed in this paper also break networks into smaller groups, but take a different approach towards doing so. As is common in many fields, however, not all software solutions are appropriate for all usages, as explored in [10]. Security is one of the most important issues in networking, as explored by [11], so there has been research into network management software tools that integrate network security features, such as is proposed in [12].

One research area in network administration that aims to fill in the gap between networking software and hardware is software defined networking (SDN), as introduced in [13], which is a system by which a network is controlled by a centralized software system, rather than the many, decentralized hardware components of the network. This centralization of network administration more easily facilitates consistency of the application of rules across the network, and increases the efficiency of making rule changes to the network. Different approaches to SDN can produce varying degrees of effectiveness and efficiency, and the differences between some of these approaches are explored by [14] and [15]. One such approach to an implementation of

SDN is to slice the topology of a network into dynamically-managed slices [16], and another approach is designed to allow the network administrator to manually group together parts of the network to fit their needs [17]. Our approach falls somewhere between these two approaches, in that networks are split up into groupings based on pre-existing rules, but these groupings are not overtly customizable. Different approaches to SDN offer differing levels of automation, customizability, and dynamic control, and our approach focusses on automation, with a moderate degree of dynamic control.

3. Policy Units and Categories: Concepts, Models, and Algorithms

This section introduces the policy unit and category models, explains their structures and purposes, and describes the algorithms used to produce them based on input data from existing network configurations. This section also goes over examples of how the introduced algorithms function, for a more thorough understanding of how the models and their algorithms function in practice.

3.1 Policy Units

The policy unit model is an abstraction by which individual IP addresses in a network can be grouped together and handled as a cohesive unit. More specifically, the IPs are bundled into these groups according to similarities in how they are already handled by their network's security rules (Access Control List (ACL) rule sheets, specifically). In this paper, we group IPs into policy units in two distinct ways:

- **Source Policy Units,** which are formed by grouping together IPs that may access the same set of other IPs, and
- **Destination Policy Units,** which are formed by grouping together IPs that may be accessed by the same set of other IPs.

3.1.1 Model

An illustration of the Policy Unit model applied to an example network is given below in Fig. 1: hosts A and B are computer science department faculty members, hosts C and D are computer science department printers, hosts E and F are computer science department students, host G is a biology department printer, hosts H and I are biology department students, hosts J and

K are biology department faculty members, and host L is a grading server. In this example, there are some accesibility rules in place. Faculty and students of a department may access their department's printers (A, B, E, and F may access C and D; and H, I, J, and K may access G). Faculty members from any department may also access the grading server (A, B, J, and K may access L).

Given these rules, we can group the hosts into *source*-based Policy Units (Source Policy Units): A and B (Computer Science Faculty), which may access C, D, and L (Computer Science Printers and the Grade Server); E and F (Computer Science Students), which may access C and D (Computer Science Printers); J and K (Biology Faculty), which may access G and L (Biology Printer and the Grade Server); H and I (Biology Students), which may access G (Biology Printer); and lastly, C, D, G, and L (printers and grading server), which may not access any of the other hosts.

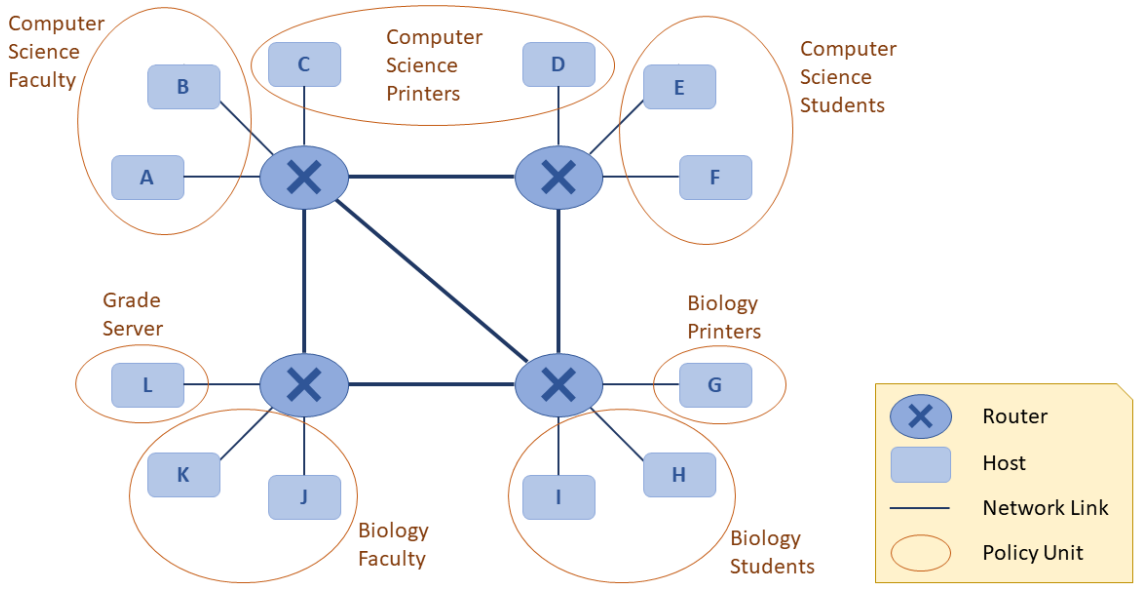


Figure 1, Policy Units in a Network

We can also use these accessibility rules to group the hosts into *destination*-based Policy Units (Destination Policy Units): C and D (Computer Science Printers), which may be accessed by A, B, E, and F (Computer Science Faculty and Students); G (Biology Printer), which may be accessed by H, I, J, and K (Biology Faculty and Students); L (Grade Server), which may be

accessed by A, B, J, and K (Computer Science and Biology Faculty); and lastly A, B, E, F, H, I, J, and K (Computer Science and Biology Faculty and Students), which may not be accessed by any of the other hosts.

3.1.2 Data Structure for Deriving Policy Units

The source Policy Unit model uses a nested data structure that has four levels, as follows (illustrated in Fig. 2):

- Level 1: The first (outermost) level structure is an array termed *Policy Array*. Each element of the array is a reference to a unique Policy Column, defined below.
- Level 2: The second level structure is a 3-tuple termed *Policy Column*. The first element of the tuple is the beginning IP of a source host IP range (a linear sequence of IPs that can assess, or cannot access, the same set of destination hosts). The second element is the ending IP of the same source host IP range. The third element is a reference to a unique Policy Body, which is defined below.
- Level 3: The third level structure is an array termed *Policy Body*. Each element of the array is a reference to a unique Policy Block (defined below). A Policy Body represents a segmentation of the entire IP space into blocks of IPs that can, and cannot, be accessed by the host IP range defined in the level 2 Policy Column that references it.
- Level 4: The fourth (innermost) level structure is a 3-tuple termed *Policy Block*. The first element of the tuple is the beginning IP of a destination host IP range (a linear sequence of IPs that can or cannot be accessed by the host IP range defined in Level 2). The second element is the ending IP of the same destination host IP range. The third element is a flag value indicating whether this destination host IP range is accessible or inaccessible by the source host IP range.

The data structure for destination policy units is nearly identical, with a few key differences: the Policy Column 3-tuples hold destination host IP ranges, and the Policy Body 3-tuples hold source host IP ranges. The significance of this difference is that the structure emphasizes which source hosts a group of destination hosts can be accessed by, instead of which destination hosts can be accessed by a group of source hosts (as in source policy units).

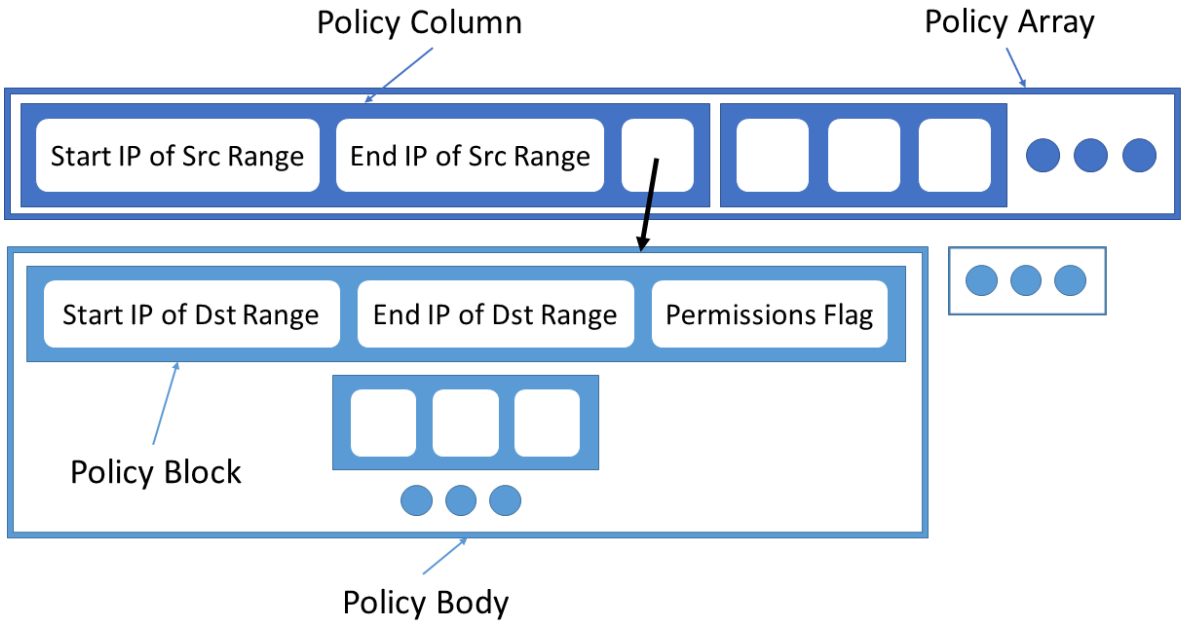


Figure 2, Source Policy Array structure: Overview of the layout of the Source Policy Array data structure, and the internal structure of the Policy Columns and the Policy Blocks

This data structure is initialized in an 'empty' state, in which the Policy Array contains one element, a reference to a Policy Column, which contains a start-of-range IP of 0 (0.0.0.0), an end-of-range IP of 4294967295 (255.255.255.255), and a reference to a Policy Body which contains one element, a reference to a Policy Block containing a start-of-range IP of 0 (0.0.0.0), an end-of-range IP of 4294967295 (255.255.255.255), and a flag of 0 (undefined accessibility; 1 is 'accessible,' and -1 is 'inaccessible').

3.1.3 Algorithms for Parsing Network Rules into Policy Units

Some automated scripts were written to facilitate the translation of network configuration files into the Policy Array data structure. These scripts extract both source and destination policy units from router ACLs of networks by recording each individual rule into the policy array data structure, and then producing policy unit definitions from this data structure. The scripts discover policy units by using the ACLs to determine which sets of IP addresses can access the same sets of other IP addresses (for defining source policy units), or which set of IP addresses can be

accessed by the same set of other IP addresses (for defining destination policy units).

After the data structure has been initialized, the network's ACL rules are processed one at a time, and their information is added to the data structure (see Algorithm 1). First, the source range of the rule is compared to the source ranges present in the data structure (Policy Array and Policy Columns), and if that specific range does not exist, the relevant existing sections of the data structure are split such that the specified range does exist (explanation of splitting is provided below). Once the source range exists in the data structure, the destination range of the rule is checked for an exact match in the Policy Bodies and Policy Blocks of the relevant Policy Column(s), and if it there is no match, existing ranges are split so that there is a match. Once the destination range exists in the data structure, the rule's accessibility statement is checked. If the rule is 'deny,' the flag of the relevant Policy Block(s) is set to -1 (inaccessible); if the rule is 'permit,' the current value of the flag(s) is checked, and if the flag has not already been set to -1 (inaccessible), then the flag is set to 1 (accessible).

Two input files are needed to execute this process: a file of network accessibility rules, which contains the network's ACLs, and a file of VLAN IP ranges, which lists the IP ranges of each VLAN. The contents of the VLAN ranges file are straightforward, with each line of the file listing the name of a VLAN, followed by that VLAN's starting and ending IP addresses (e.g. Vlan206 192.168.327.134 192.168.327.149). The contents of the file of accessibility rules is more complex. Each line contains a colon-separated list of items, starting with the source and destination VLANs, and ending with another colon-separated list within square brackets of accessibility rules dictating which IPs from the source VLAN may access which other IPs from the destination VLAN. Each of these accessibility rules are divided by spaces into sections, starting with a 'permit' or 'deny' statement, then an indicator of the type of accessibility it defines (e.g. 'ip,' 'tcp,' 'udp,' etc.), then a specification of the source IP(s) specified, and finally a specification of the destination IP(s) specified. If a single IP address is to be specified, it is given in the form of the word 'host', followed by the specific IP address (host 130.104.243.86). If a particular range of IP addresses is to be specified, it can be given in the form of an IP address and a mask (192.168.327.134 0.0.0.15). To specify all of the IPs within the source or destination VLAN, the word 'any' is used; if 'any' is used for the source IP range, it refers to all IPs within the rule's source VLAN, and if it is used for the destination IP range, it refers to all IPs within the rule's destination VLAN. An example of a line of rules is as follows:

`Vlan206:Vlan653:…:[]:[permit ip 192.168.473.232 0.0.0.7 any:permit ip 192.168.473.144 0.0.0.15 any:permit ip 192.168.473.208 0.0.0.7 any:deny ip any any:deny any any]`.

Algorithm 1: Fill Policy Array

Input: Network Accessibility Rules file (NAR), VLAN IP Ranges file (VIR)

Output: Policy Array data structure

```
FOR each line in NAR DO
    get source and destination VLAN IP ranges by referencing VIR
    split line into rules
    FOR each rule in line DO
        get permit/deny indicator of rule
        get source and destination ranges (if 'any', use relevant VLAN range)
        append the rule data to ruleline
    END FOR
    push ruleline to rulesAbrev array
END FOR
initialize PolicyArray
FOR each line in rulesAbrev DO
    FOR each rule in line DO
        get permission flag, src range start, src range end, dst range start, and dst
          range end from rule
        get PolicyArray indexes to match src range, splitting PolicyColumns as
          needed
        FOR each relevant PolicyColumn in PolicyArray DO
            get PolicyBody indexes to match dst range, splitting PolicyBlocks as
              needed
            FOR each relevant PolicyBlock in PolicyBody DO
                IF PolicyBlock's permission flag is not deny THEN
                    set PolicyBlock's flag to rule's permission flag
                END IF
            END FOR
        END FOR
    END FOR
END FOR
```

To split a range means that all of the existing ranges are scanned until the ranges that overlap the rule's range are found, and then those ranges are divided at the point of overlap, such that there are now two ranges where there was one, and each range covers a smaller span of IP addresses, with no overlapping IP coverage between the two. If the rule's range falls in the middle of a larger existing range (e.g. rule range of 20—30 overlaps existing range of 10—40), the existing range is split into three parts (e.g. new, split ranges of 10—19, 20—30, and 31—40) instead of two. When splitting ranges, the new ranges are kept in sequential order (e.g. 0—42, 43—97, 98—183, 184—185, 186—207, ...) so that it is easier to find which existing ranges overlap with new rules. In Policy Columns, the reference to the Column's corresponding Policy Body is cloned when splitting, such that each new Policy Column now has its own unique but identical Policy Body to reference. When Policy Blocks are split, the accessibility flag is kept identical for each of the two new ranges.

Once all of the ACL rules have been processed into the data structure in this manner, the data structure gets compacted down into strings, and policy units are determined, starting with Policy Bodies and Policy Blocks (see Algorithm 2). The script loops through each Policy Body to look at the flags of their Policy Blocks to determine if the range is accessible. If it is, the range is converted into a string of the form ‘startIP-endIP’ and stored in a holding array. Once all accessible IP ranges from the Policy Body have been added to the holding array, the holding array is joined into a string so that it is an ordered, comma-separated string of all accessible destination ranges in that particular Policy Body. Once this is done, the Policy Column’s reference to that Policy Body is changed to instead reference this string of accessible destination ranges. After all of the accessible destination ranges have been converted to strings in this manner, the script loops through the source ranges (Policy Array and Policy Columns) to see which ones have identical destination strings. Policy Columns with identical destination strings are collected together in string format (in the same manner as the destination ranges above) and removed from the Policy Array. When every source ranges with an identical destination range set to that of the first source range of the loop have been gathered, a policy unit has been found, so the source ranges are collapsed into a string, just as with the destination ranges, and the resulting pair of source range and destination range strings are recorded in a text file as a policy unit. This is repeated until all policy units have been found (which has been achieved when the Policy Array is empty).

Algorithm 2: Form Policy Units from filled Policy Array

Input: Policy Array data structure

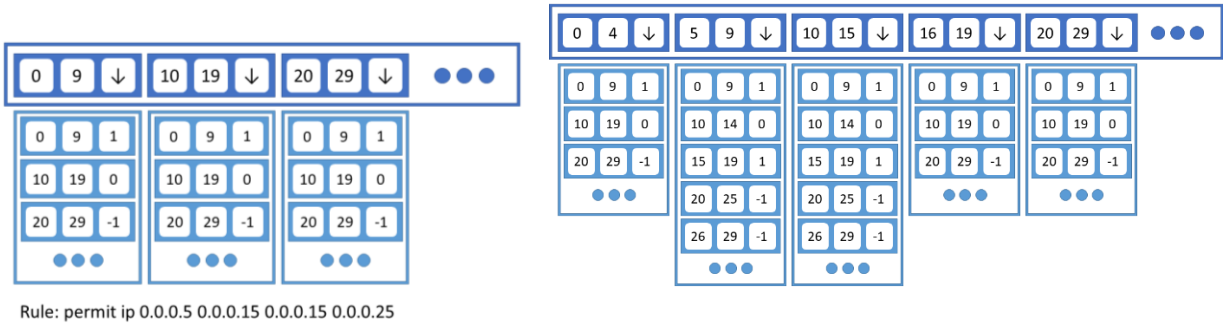
Output: Policy Units file

```
FOR each PolicyColumn in PolicyArray DO
    get PolicyBody pointer from PolicyColumn
    FOR each PolicyBlock in PolicyBody DO
        IF permission flag is 'accessible' THEN
            add range to dstRange string
        END IF
    END FOR
    replace Policy Column's Policy Body pointer with pointer to dstRange string
END FOR
WHILE PolicyArray is not empty DO
    skim out and remove from PolicyArray all PolicyColumns elements with
      dstRange strings matching that of the PolicyColumn at PolicyArray[0]
    gather the source ranges of each of these PolicyColumns into a srcRange
      string
    combine srcRange and dstRange strings to form a policy unit string
    write this policy unit string into Policy Units file
END WHILE
```

For destination policy units, the whole process is identical, excepting that destination ranges are stored in the Policy Columns, and source ranges are stored in the Policy Blocks.

3.1.4 Examples

In an example execution, The Policy Array may contain Policy Columns with source IP ranges of 0-9, 10-19, 20-29, and so forth, and each of these Policy Columns might point to Policy Bodies whose Policy blocks have destination ranges of 0-9, 10-19, 20-29, and so forth, with range 0-9 flagged as accessible, range 10-19 unflagged, range 20-29 flagged as inaccessible, and all other ranges being of arbitrary accessibility (see Fig. 3(a)).



(a) Policy Array before processing a rule (b) Policy Array after processing a rule

Figure 3, Policy Array structure

Let us say that the next ACL rule read in from the file of network accessibility rules is "permit ip 0.0.0.5 0.0.0.15 0.0.0.15 0.0.0.25", which means that that source IPs 5-15 may access destination IPs 15-25. When the script reads in this rule, it will look at the Policy Columns in the Policy Array to determine which ones will be affected by this rule, and it will determine that the first and second columns, source ranges 0-9 and 10-19, overlap the source IP ranges of the rule. However, they do not overlap exactly, so the Policy Columns must be split, so that only the intended source IPs will be affected by the modifications to the data structure that will indicate the accessibilities granted by the new rule. To accomplish this, the Policy Column of 0-9 will be split into two separate Policy Columns of 0-4 and 5-9, each with their own unique but identical Policy Bodies that are identical to the one that belonged to the Policy Column of 0-9, and likewise the Policy Column of 10-19 will be split into Policy Columns of 10-15 and 16-19. When the columns are split, they retain their places in the Policy Array such that they remain in sequential order relative to their IP ranges, for ease of future searching of Policy Columns in the Policy Array.

After this splitting is done, the Policy Array contains Policy Columns with source IP ranges of 0-4, 5-9, 10-15, 16-19, 20-29, and so forth, and the script proceeds to repeat this searching and splitting procedure on the Policy Bodies of the Policy Columns of 5-9 and 10-15. Both of the Policy Bodies are examined for the range of 15-25 and, as with the Policy Columns, the Policy Blocks within both Policy Bodies are split such that each contains Policy Blocks with destination IP ranges of 0-9, 10-14, 15-19, 20-25, 26-29, and so on, with the flag value of each split Policy Block being retained by each of its new fragments. After Splitting the Policy Blocks

of both Policy Bodies, the script checks the flag values of the Policy Blocks of 15-19 and 20-25. The 15-19 Policy Block in both Policy Bodies is unflagged, so the script sets the flags to ‘accessible’ to indicate the new accessibility data from the new rule. The 20-25 Policy Blocks are flagged as inaccessible, so the flags for these Blocks are left unchanged, because inaccessibility rules take precedence over accessibility rules (see Fig. 3(b)). Having altered all relevant flags, the script continues by reading in the next network accessibility rule, and keeps processing rules in this manner until all rules have been processed.

To continue this example execution with a demonstration of the process of condensing the Policy Array into Policy Units, we may assume that the rule processed above was the final rule to be processed, and proceed from that state of the Policy Array. We shall also assume, for the sake of simplicity, that the largest possible IP address is 29 (0.0.0.29).

The script will first loop through the Policy Blocks of each Policy Column's Policy Body, and convert the destination ranges within them into strings, starting with the first column in the Policy Array, the column of 0-4. The first Policy Block in this Policy Column's Policy Body contains the destination range 0-9, which is flagged as accessible, so its range is converted into a string of the format ‘0-9’. The next Policy Block contains the destination range 10-19, and is unflagged, so it is not converted to string format, and the last block, of destination range 20-29, is flagged as inaccessible, so it is also not converted into string format. Having converted all of the accessible destination IP ranges of the Policy Body into strings, the individual strings are joined into one comma-separated string, and the pointer of the 0-4 Policy Column is modified to point to this string of “0-9”.

The next Policy Column, range 5-9, is examined next, which produces a destination string of “0-9,15-19”, and the Policy Column of range 10-15 also produces a destination string of “0-9,15-19”. The remaining columns of 16-19 and 20-29 each produce a destination string of “0-9”, and with their completion, the script proceeds to compile the Policy Columns together into source strings.

To form Policy Units, the script pulls Policy Columns from The Policy Array that have identical strings of accessible destination ranges, and groups them together into a string of source ranges, followed by their corresponding string of accessible destination ranges. This is accomplished by first taking the destination range string of the first Policy Column in the Policy Array, in this case the 0-4 column, recording that column's source range in string format just as

above with the Policy Blocks, and removing this Column from the Policy Array. Next, the script loops through the remaining Policy Columns in the array, looking for identical destination range strings; when a match is found, the source range of that Column is recorded into the source ranges string, and the Column is removed from the Policy Array. After the entire Policy Array has been scanned, the sources string and the destinations string are joined with a semicolon between the two as a delimiter to separate them. This string now represents one Policy Unit, and it is recorded in a text file. After the first pass of the Policy Array, the columns that have been matched and removed are those of ranges 0-4, 16-19, and 20-29, so the first Policy Unit recorded is "0-4,16-19,20-29;0-9", and the Policy Columns remaining in the Array are those of ranges 5-9 and 10-15. On the second sweep of the Policy Array, these two Policy Columns are found to have identical destination strings, so the second and last Policy Unit recorded is "5-9,10-15;0-9,15-19".

3.2 Categories

The category model is an extension of the policy unit model by which policy units can be grouped together into larger groupings of IPs to further increase ease and efficiency of network management. More specifically, the policy units are grouped together by putting them into sets such that each policy unit is present in two categories, and the intersection of any two categories will give exactly one of the policy units, such that each policy unit can be received from one intersection of two of the categories. (This model could conceivably also be framed such that each policy unit is received from an intersection of three sets, four sets, or so forth.) As with policy units, we define two kinds of categories:

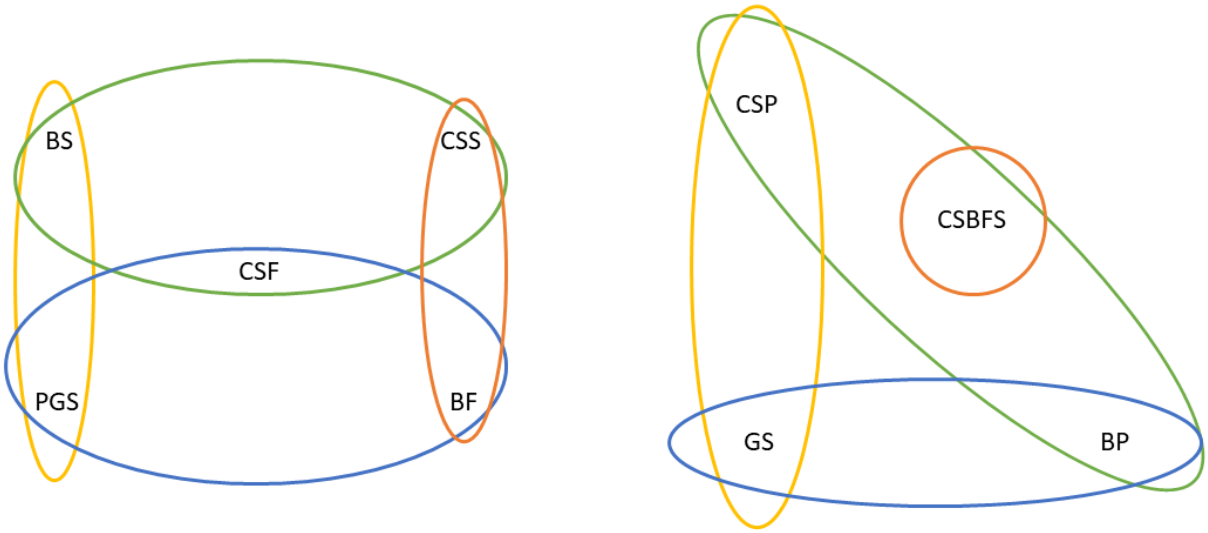
- **Source Categories,** which are generated based on Source Policy Units, and
- **Destination Categories,** which are generated based on Destination Policy Units.

3.2.1 Model

To illustrate the Category model, we will expand on the example given in section 3.1.1. Starting with the source policy units from the example network, we have the Computer Science Faculty (CSF), Computer Science Students (CSS), Biology Faculty (BF), Biology Students (BS), and Printers and Grading Server (PGS) source policy units. Forming these policy units into categories gives us four source categories: the first category contains the CSF, CSS, and BS

policy units; the second category contains the CSF, BF, and PGS policy units; the third category contains the CSS and BF policy units; and the fourth category contains the BS and PGS policy units (illustrated in Fig. 4(a)).

Similarly, we can work from the destination policy units of the example network, which are Computer Science Printers (CSP), Biology Printer (BP), Grade Server (GS), and Computer Science and Biology Faculty and Students (CSBFS). Forming these policy units into categories gives us four destination categories: the first category contains the CSP, BP, and CSBFS policy units; the second category contains the CSP and GS policy units; the third category contains the BP and GS policy units; and the fourth category contains the CSBFS policy unit (illustrated in Fig. 4(b)).



(a) Source Categories (b) Destination Categories

Figure 4, Categories in the Example Network

3.2.2 Data Structure of Categories

The Category model is in the form of a set. As described above, each category consists of a set of policy units; more specifically, each category is a set of all the IPs of each policy unit that it contains. These sets are stored in the form of comma-separated strings. Categories themselves are, likewise, stored in a text file, with each line of the file containing all the IP ranges of a particular category.

3.2.3 Algorithms

The intent of the algorithm used to arrange policy units into categories is to minimize the number of categories needed to store all of the policy units. This is accomplished by looping through the policy units, and incrementing two index variables to determine which two categories to add each policy unit to, as shown in Algorithm 1. The *i* variable increments with each new policy unit, reverting to zero each time it reaches the last existing category (or, when *i*==*j*), and the *j* variable increments each time *i* reverts back to zero. Each incrementation of *j* adds a new category for the network, and the total number of categories is equal to *j*+1. This setup distributes policy units into categories such that a new category is added only when there are no unique combinations left among the existing categories to pair with one another, and each new category created by an incrementation of *j* is paired with each of the pre-existing categories before the creation of the next category.

The input for Algorithm 3 is a file of Policy Units, as produced by Algorithm 2, containing all of the source hosts IPs and the destination host IPs they can access within each source policy unit, and containing all of the destination host IPs and the source IPs that can access them within each destination policy unit. If source policy units are given as input, the output will be source categories, and if destination policy units are given as input, the output will be destination categories.

Algorithm 3: Form Categories from Policy Units

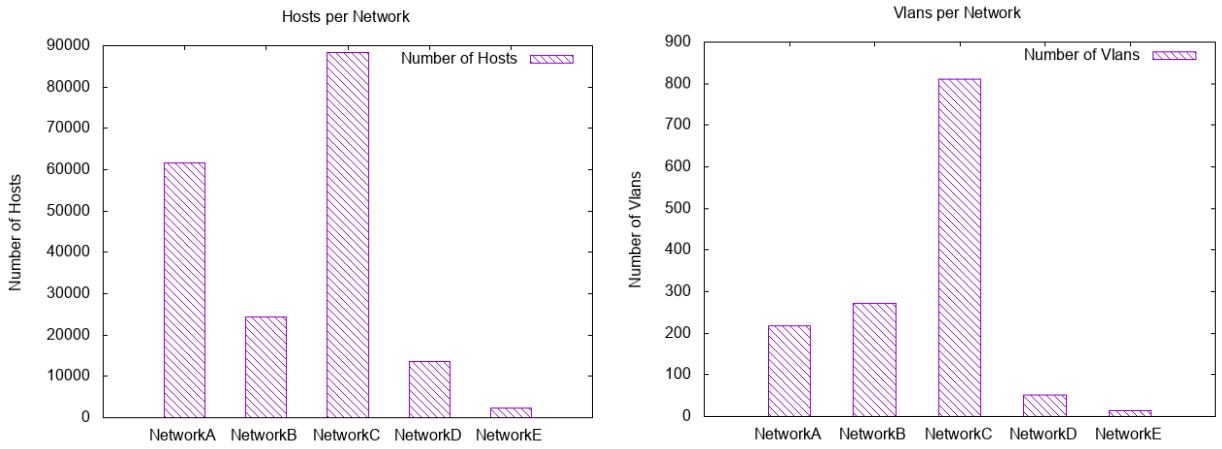
Input: Policy Units file (PUF)
Output: Categories file

```
i = 0
j = 1
FOR each policyUnit in PUF DO
    append policyUnit to categories[i]
    append policyUnit to categories[j]
    i++
    IF i == j DO
        i = 0
        j++
    END IF
```

END FOR

4. Evaluation Results

In this section, we examine and compare various properties of the policy units produced from five enterprise networks, referred to here as Networks A, B, C, D, and E. Fig. 5(a) shows the number of IP addresses in each network, and Fig. 5(b) shows the number of VLANs in each network. Among these five networks, Network C is the largest, with more IPs and VLANs than the other four networks, while Network E is the smallest and Network D is the second-smallest. Networks A and B are not as clear, however, as Network A has many more IPs than B, but Network B has a few more VLANs than A. This could be because Network A is a more loosely-structured network than B, despite its larger quantity of IPs. The properties of these networks that we examine with respect to policy units are the number of IP addresses included in each policy unit and the number of VLANs that are included in each policy unit, as well as the number of policy units that appear within each VLAN. Comparisons are also made between source and destination policy units within the same networks. Finally, we look at the number of source and destination policy units and categories in each network. From these examinations, we intend to learn more about the nature of policy units and categories in the context of enterprise networks.



(a) Number of hosts in each network (b) Number of VLANs in each network

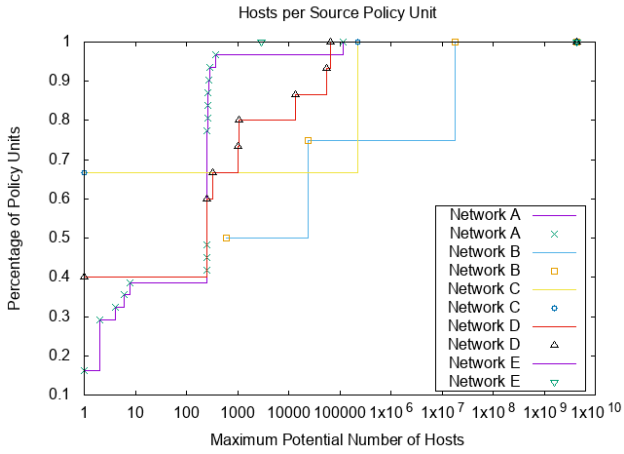
Figure 5, Network Properties

4.1 Source Policy Units

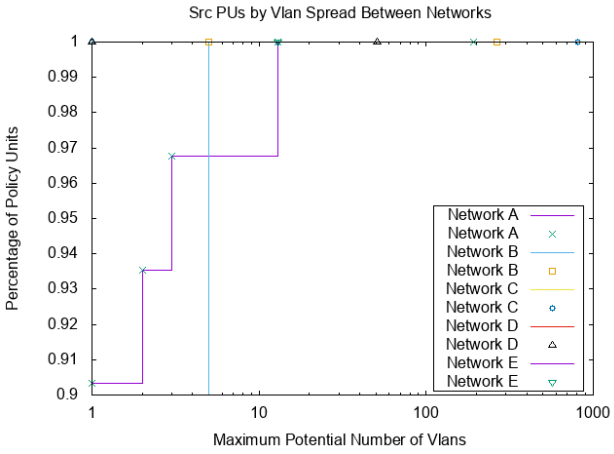
In our analysis of source policy units across the five networks, we looked at how many

IPs were covered by each source policy unit (Fig. 6(a)), how many VLANs were at least partially included in each source policy unit (Fig. 6(b)), and how many source policy units were at least partially included in each VLAN in each network (Fig. 6(c)). Fig. 6(a) shows that source policy units in our sample networks tend to be either very small, covering no more than ten IPs each, or large, covering upwards of 500 IPs each. Fig. 6(b) shows that nearly all source policy units in our sample networks cover no more than 11 VLANs each, and Fig. 6(c) shows that nearly all VLANs in our sample networks include no more than 15 source policy units each.

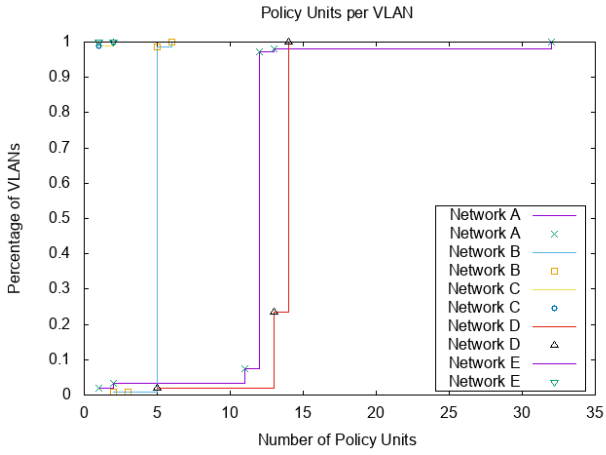
Overall, the observable trends are that source policy units tend to be either large or very small, and that there tend to be neither very many VLANs per source policy unit, nor very many source policy units per VLAN. This might imply that many of the VLANs that are included in policy units might contain a large span of IPs, in order for the source policy units which cover few VLANs to include a large number of IPs.



(a) Maximum potential number of hosts per Source Policy Unit in each network



(b) Maximum potential number of VLANS included per Source Policy Unit in each network



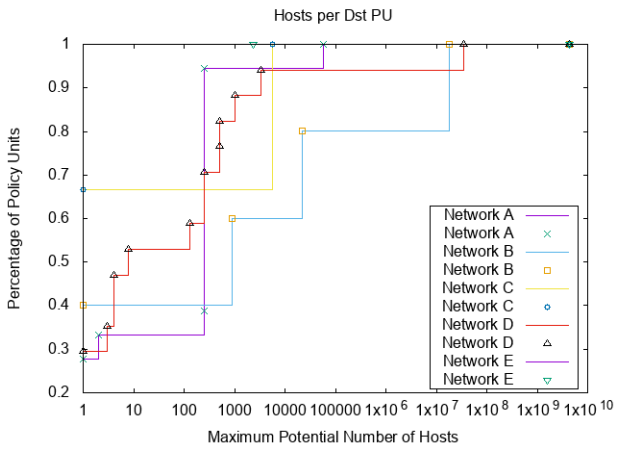
(c) Maximum potential number of Source Policy Units that include each VLAN in each network

Figure 6, Source Policy Units

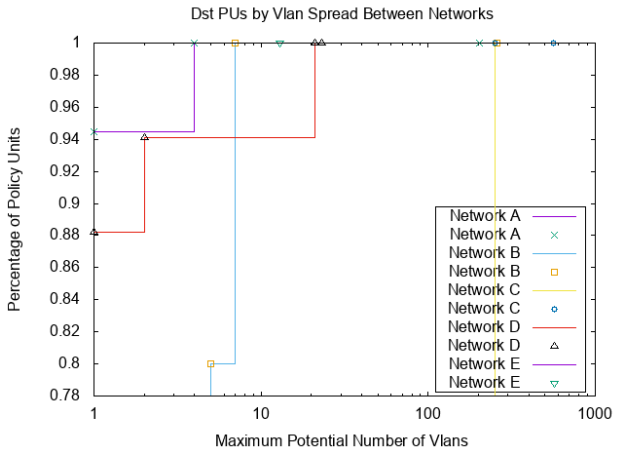
4.2 Destination Policy Units

In our analysis of destination policy units, we looked at how many IPs were covered by each destination policy unit (Fig. 7(a)), and how many VLANs were covered by each destination policy unit (Fig. 7(b)). Fig. 7(a) shows that the destination policy units in our sample networks come in a wide range of sizes, with many covering between one and ten IPs each, most covering between 100 and 100,000 IPs each, and a few covering more than 10,000,000 IPs each. Fig. 7(b) shows that, while the number of VLANs in our sample covered by each destination policy unit varies between each network, the apparent trend is for each destination policy unit to include no more than 12 VLANs each.

Overall, the observable trends are that destination policy units come in a great range of numbers of IPs covered by each, and that most destination policy units cover a fairly small number of VLANs each. As with the source policy units, this may imply that the individual VLANs within networks might tend to cover a wide range of IPs, in order for the destination policy units to cover a large number of IPs without covering a large number of VLANs. In the section below discussing categories, Fig. 13 shows that each network contains more source policy units than destination policy units.



(a) Maximum potential number of hosts per Destination Policy Unit in each network



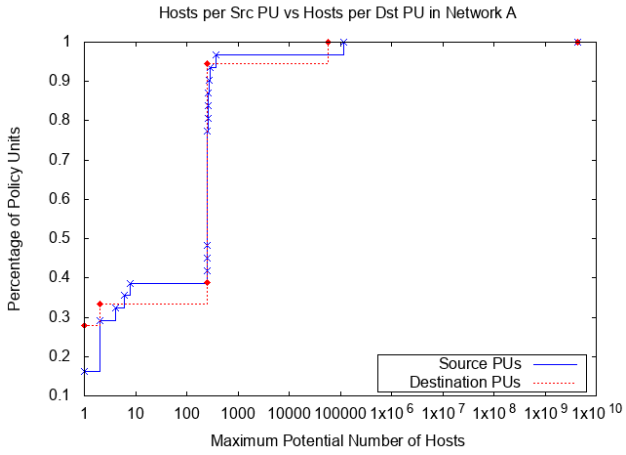
(b) Maximum potential number of VLANS included per Destination Policy Unit in each network

Figure 7, Destination Policy Units

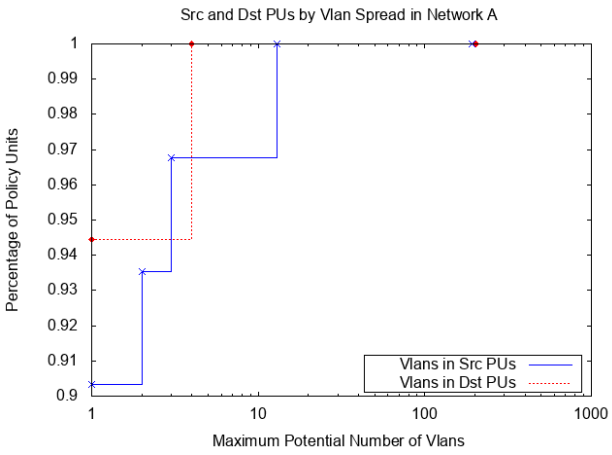
4.3 Analysis of Policy Units in each Network

In addition to comparing policy units from different networks to one another, we also compare source and destination policy units from the same network to one another, to look for similarities or differences in their distributions. We also look at how the policy unit distributions of a network compare to VLAN distributions of that network.

Network A. In Network A, Fig. 8(a) shows us that the numbers of IPs in each source policy unit and in each destination policy unit are similar in this network, and Fig. 8(b) shows us that source policy units have a greater variety of VLAN coverage counts than destination policy units, and that source policy units also tend to include more VLANs each than destination policy units.



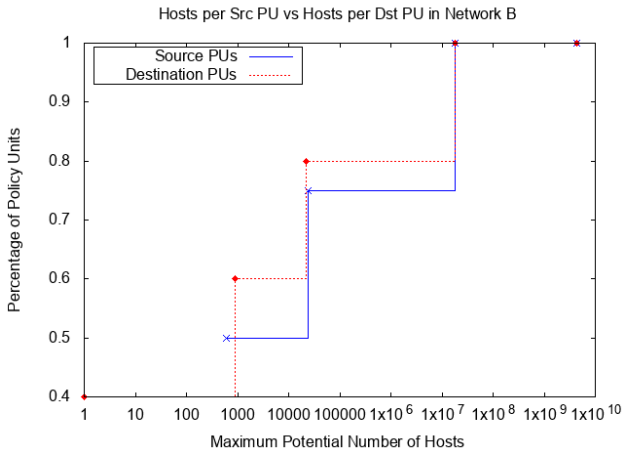
(a) Maximum potential number of hosts per Source and Destination Policy Units in Network A



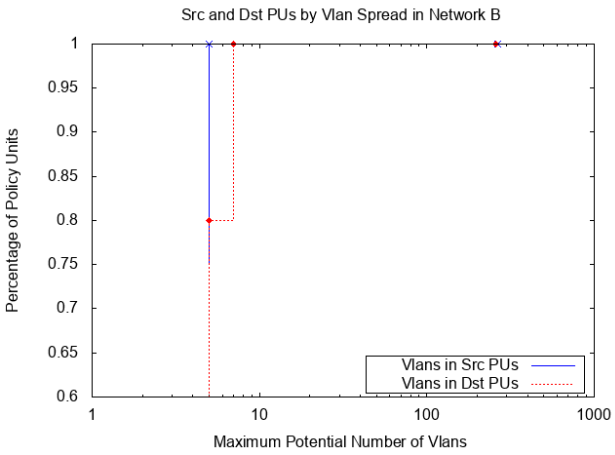
(b) Maximum potential number of VLANS included per Source and Destination Policy Unit in Network A

Figure 8, Network A

Network B. In Network B, Fig. 9(a) shows us that source and destination policy units in this network tend to cover similar numbers of IPs, and Fig. 9(b) shows us that most source and destination policy units in this network include five VLANs each, with some destination policy units covering seven each.



(a) Maximum potential number of hosts per Source and Destination Policy Units in Network B



(b) Maximum potential number of VLANS included per Source and Destination Policy Unit in Network B

Figure 9, Network B

Network C. In Network C, Fig. 10 shows us that source and destination policy units in this network tend to share a relatively similar spread of numbers of included IPs, with many source policy units covering many more IPs than their destination policy unit counterparts. Information gathered from this network about VLANs per policy unit indicates that most of the policy units cover no VLANs, with a few policy units covering hundreds of VLANs each.

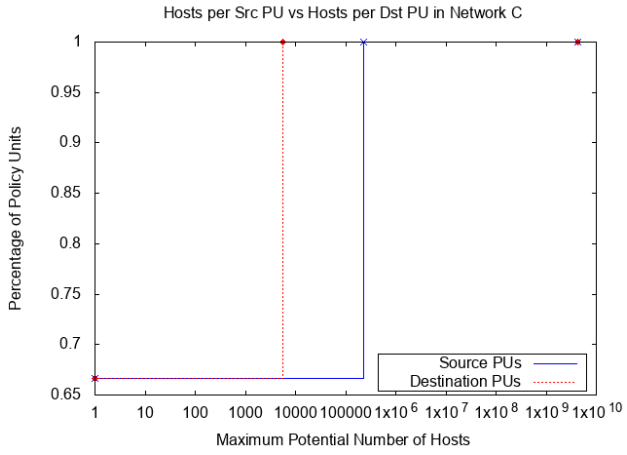
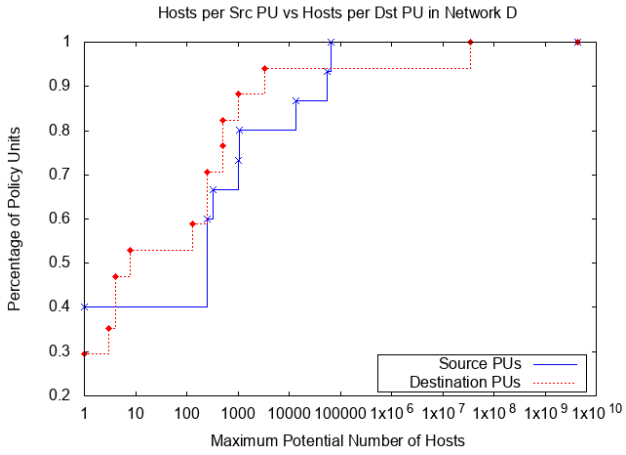


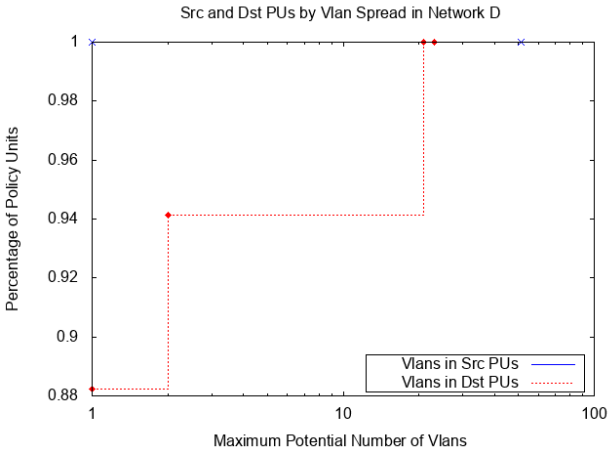
Figure 10: Maximum potential number of hosts per Source and Destination Policy Units in Network C

Network D. In Network D, Fig. 11(a) shows us that in this network, the source and destination policy units tend to cover similar quantities of IPs each, and Fig. 11(b) shows us that,

while nearly all of the source policy units cover just one VLAN each, some of the destination policy units branch out to cover two each, and a few more cover 11 each.



(a) Maximum potential number of hosts per Source and Destination Policy Units in Network D



(b) Maximum potential number of VLANS included per Source and Destination Policy Unit in Network D

Figure 11, Network D

Network E. In Network E, Fig. 12 shows us that the source and destination policy units in this network tend to cover similar numbers of IPs each. Information gathered from this network about VLANs per policy unit indicate that each source and destination policy unit include exactly 13 VLANs each.

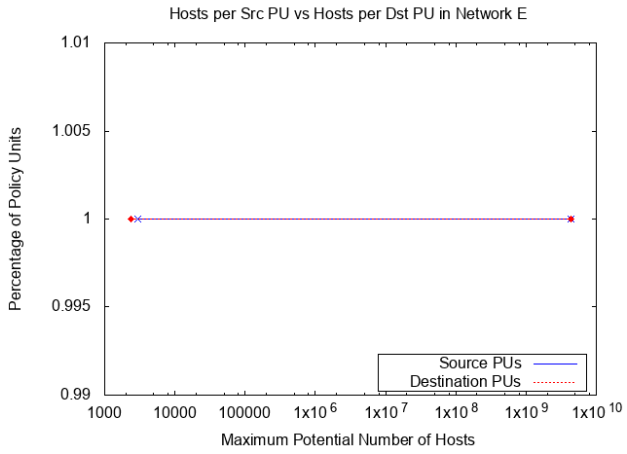


Figure 12: Maximum potential number of hosts per Source and Destination Policy Units in Network E

In each of these five networks, it may be noted that the number of IPs covered by each source policy unit and by each destination policy unit bear similarities to one another, so we might theorize from this that source and destination policy units within a network will tend to loosely mirror one another in this aspect. Similarities in VLAN coverage also appear to potentially be present, but to a lesser extent.

It can be noted that networks A and D have the greatest variance in policy unit sizes, while networks B, C, and E have only a few sizes of policy units. This correlates to the number of policy units in each network, as seen in Fig. 13 below: networks A and D have relatively many policy units, while networks B, C, and E have relatively few policy units.

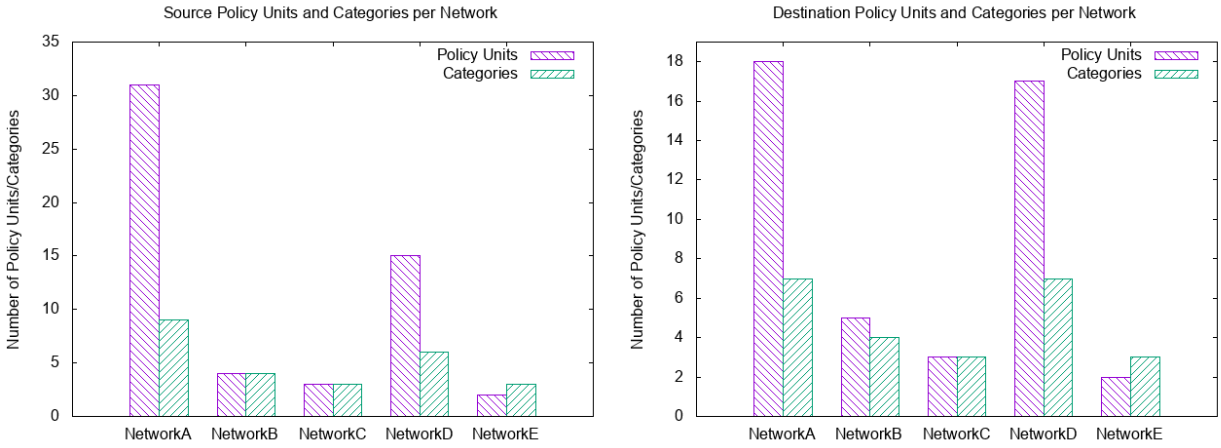
4.4 Policy Categories

Fig. 13 shows that the number of policy categories in a network can be much smaller than the number of policy units in that network, especially if the network has many policy units. Networks with one or two policy units will have more policy categories than policy units, networks with three or four policy units will have the same number of policy categories as policy units, and networks with five or more policy units will have fewer policy categories than policy units. This tells us that the usefulness of the policy category model grows with the number of policy units in a network, and is not likely to be useful for particularly small or unrestrictive networks. This figure also shows that, in each network, there are more source policy units and categories than destination policy units and categories.

Referring back to the number of hosts and VLANs in each network (Fig. 5), we can further analyze the distribution of policy units and categories in each network. Network A, which has the second-most IPs and the third-most VLANs, leads in the number of source and destination policy units and source categories, and ties with Network D for the most destination categories. Network B, which has the third-most IPs and the second-most VLANs, ranks third in numbers of source and destination policy units and categories. Network C, which has the most IPs and VLANs, has the second-fewest source and destination policy units, and ties with Network E for the fewest source and destination categories. Network D, which has the second-fewest IPs and VLANs, has the second-most number of source and destination policy units and source categories, and ties with Network A for most destination categories. Network E, which has the fewest IPs and VLANs, has the fewest source and destination policy units, and ties with

Network C for fewest source and destination categories.

It is interesting to note that Network C, which has relatively many IPs and VLANs, has relatively few policy units and categories, and Network D, which has relatively few IPs and VLANs, has relatively many policy units and categories, while Network A, which has relatively many IPs and VLANs, also has relatively many policy units and categories, and Network E, which has relatively few IPs and VLANs, also has relatively few policy units and categories. (Meanwhile, Network B remains fairly consistently in the middle of the data set for these aspects.) This implies that there is little correlation between the size of a network and the number of rules it has; large networks can have many rules or few rules, and small networks can also have few or many rules. A conclusive analysis of this lack of correlation would, however, require further research with larger sample sizes.



(a) Number of Source Policy Units and Source Categories in each network

(b) Number of Destination Policy Units and Destination Categories in each network

Figure 13, Policy Units and Categories per network

5. Future Work and Conclusion

We have seen that the application of our models can greatly reduce the management size of a network by reducing tens of thousands of IPs down to merely tens of policy units, with each policy unit consisting of IP addresses that receive the same treatment in regard to policy rules, and then further grouping those policy units into even fewer categories, within which they might all receive similar overarching policy treatments. Further research could be conducted to apply these models to more networks for a more comprehensive analysis of the effectiveness of the models, and later perhaps to develop the models into a usable tool for network management, or

integrate the models into an existing tool, and test how helpful the models are in practice to network managers in terms of efficiency, ease of use, applicability to their networks, etc.

With the application of policy unit and category models to network management, it is possible to reduce the amount of time and effort required to modify and maintain network policy rules, which would in turn reduce the opportunities for human error in the network maintenance process. Human error, as discussed previously, can lead to major security breaches, so a reduction of opportunities for human error is an increase in the security of a network. With security being such a significant matter in networking, the models proposed here have the potential to play an important role in the future of network management.

References

[1] L. H. Newman. (2016, December 14). *Hack Brief: Hackers Breach a Billion Yahoo Accounts. A Billion* [Online]. Available: https://www.wired.com/2016/12/yahoo-hack-billion-users/

[2] L. H. Newman. (2017, October 03). *Yahoo's 2013 Email Hack Actually Compromised Three Billion Accounts* [Online]. Available: https://www.wired.com/story/yahoo-breach-three-billion-accounts/

[3] A. Peterson. (2014, December 18). *The Sony Pictures hack, explained* [Online]. Available: https://www.washingtonpost.com/news/the-switch/wp/2014/12/18/the-sony-pictures-hack-explained

[4] C. Saran. (2004, February 10). *Human error, not software, the main cause of network failure* [Online]. Available: https://www.computerweekly.com/news/2240054454/Human-error-not-software-the-main-cause-of-network-failure

[5] D. E. Comer, "Configuration And Operation," in *Automated Network Management Systems*, *1*st ed. Upper Saddle River, NJ: Pearson Education, Inc., 2007, ch. *4*, sec. *4.11*, pp. *48-49*

[6] A. Gupta, "Network Management: Current Trends and Future Perspectives," *Journal of Network and Systems Management*, vol. *14*, no. *4*, pp. *483-491*, Dec, 2006.

[7] D. E. Comer, "The Network Management Challenge," in *Automated Network Management Systems*, 1st ed. Upper Saddle River, NJ: Pearson Education, Inc., 2007, ch.

1, sec. *1.6*, pp. *3*

[8] D. E. Comer, "Network Automation: Questions And Goals," in *Automated Network Management Systems*, 1st ed. Upper Saddle River, NJ: Pearson Education, Inc., 2007, ch. *14*, sec. *14.4*, pp. *242*

[9] K. Ikehara, K. ikeda, K. Motomura, Y. Inoue, "Proposal of a hierarchical network management method based on network management protocol monitoring," *2000 IEEE/IFIP Network Operations and Management Symposium 'The Networked Planet: Management Beyond 2000'*, Honolulu, HI, 2000, pp. *973-974*.

[10] U. H. Rao, S. Mohapatra, "Deploying Network Management Solutions in Enterprises," *INC2010: 6th International Conference on Networked Computing*, Gyeonju, South Korea, 2010, pp. *11-13*.

[11] R. Ward, P. Skeffington, "Network Management Security," *Proceedings of the Sixth Annual Computer Security Applications Conference*, Tuscon, AZ, 1990, pp. *173-180*.

[12] Z. Kan, C. Hu, Z. Wang, G. Wang, X Huang, "NetVis: A Network Security Management Visualization Tool Based On Treemap," *Workshop on Visualization for Cyber Secur*ity, Atlanta, GA, 2013, pp. *41-48*.

[13] H. Kim, N. Feamster. (2013, February 14). Improving Network Management with Software Defined Networking. *Mag*. [Online]. pp. 114-119. Available: https://ieeexplore.ieee.org/document/6461195

[14] A. Gelberger, N. Yemini, R. Giladi, "Performance Analysis of Software-Defined Networking (SDN)," *2013 IEEE 21st International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems*, San Francisco, CA, 2013, pp. *389-393*.

[15] M. H. Raza, S. C. Sivakumar, A. Nafarieh, B. Robertson, "A Comparison of Software Defined Network (SDN) Implementation Strategies," *2nd International Workshop on Survivable and Robust Optical Networks (IWSRON)*, 2014, pp. *1050-1055*.

[16] M. Raza, V. Samineni, W. Robertson, "Physical and Logical Topology Slicing Through SDN," *2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, Vancouver, BC, Canada, 2016, pp. *1-4*.

[17] C. S. Gomes, F. S. Dantas Silva, E. P. Neto, K. B. Costa, J. B. da Silva, "Towards a Modular Interactive Management Approach for SDN Infrastructure Orchestration," *2016*

IEEE Conference on Networking Function Virtualization and Software Defined Networks (NFV-SDN), Palo Alto, CA, 2016.