2013-12

Minimizing Network Complexity through Integrated Top-Down Design

Sun, Xin

http://hdl.handle.net/10945/46775

Minimizing Network Complexity through Integrated Top-Down Design

Xin Sun
School of Computing and Information Sciences
Florida International University
xinsun@cs.fiu.edu

Geoffrey G. Xie
Department of Computer Science
Naval Postgraduate School
xie@nps.edu

ABSTRACT

The network design process today remains ad-hoc and largely complexity agnostic, often resulting in suboptimal networks characterized by excessive amounts of dependencies and commands in device configurations. The unnecessarily high configuration complexity can lead to a huge increase in both the amount of manual intervention required for managing the network and the likelihood of configuration errors, and thus must be avoided. In this paper we present an integrated top-down design approach and show how it can minimize the unnecessary configuration complexity in realizing user reachability control, a key network design objective that involves designing three distinct network elements: VLAN, IP address, and packet filter. Capitalizing on newly-developed abstractions, our approach integrates the design of the three elements into a unified framework by systematically modeling how the design of one element may impact the complexity of other elements. Our approach goes substantially beyond the current "divide-and-conquer" approach that designs each element in complete isolation, and enables minimizing the combined complexity of all elements. Specifically, two new optimization problems are formulated, and novel algorithms and heuristics are developed to solve the formulated problems. Evaluation on a large campus network shows that our approach can effectively reduce the packet filter complexity and VLAN trunking complexity by more than 85% and 70%, respectively, when compared to the ad-hoc approach currently used by the operators.

Categories and Subject Descriptors

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

Keywords

Network complexity; Top-down design; Reachability control; VLAN; IP address allocation

1 Introduction

Recent research [7,22,31] and vendor documents [1,27] reveal that multiple, distinct routing designs are possible to meet the same set of enterprise network operational requirements (e.g., security policy represented by a reachability matrix [35]). Moreover, the configuration complexity of these designs can vary greatly. In other words, some designs may incur much higher configuration complexity than others while accomplishing the same objectives. The *unnecessarily high* configuration complexity is highly undesirable as it can lead to a huge increase in both the amount of manual intervention required for managing the network and the likelihood of configuration errors. For example, a research report [20] discloses that 80% of enterprise IT budget is devoted to maintaining the status quo. Despite this investment, configuration errors account for 50-80% of network outages [18, 20] and enable 65% of all successful cyber-attacks [28]. There is a general perception that complexity is the primary cause of high human costs, and interviews and anecdotal evidence suggest that an operator's ability to run a network decreases as the network becomes more complex [7].

Thus, an important open research question arises: *Is it possible to systematically identify, among all designs that can meet given operational requirements, the one(s) with the minimum amount of configuration complexity?*

The current state of network design practice by operators is mostly ad hoc and, in particular, does not rigorously formulate the goal of minimizing network complexity. As a result, a large number of existing production networks may not be optimal in terms of configuration complexity [22, 33], likely causing a huge increase in operational costs. Having recognized the importance of the problem and associated challenges, researchers have recently begun to investigate this problem in the specific context of enterprise network design [32, 33]. These approaches focus primarily on meeting the specific objective of *user reachability control* (essentially implementing a subnet-level reachability matrix). They enable an operator to formulate an individual design task, such as grouping hosts of his/her network into different VLANs, into a model of optimizing a desired performance metric subject to a set of correctness and feasibility constraints.

While these recent advances in systematic network design create a major opportunity to address the complexity problem, the current approaches suffer from a critical limitation: they employ a "divide-and-conquer" (i.e., stage-by-stage) strategy that simply models individual design steps in complete isolation even when the steps together implement a common goal. For example, totally independent formulations and optimality criteria are used for VLAN design and packet filter design [33] even though the two design steps share a common objective of user reachability control. While these formulations can potentially minimize the complexity of configuration at each design stage in isolation, the overall complexity may still be unnecessarily high. This is because the design choices made at an early stage (e.g., VLAN design or IP address allocation)

CoNEXT'13, December 9–12, 2013, Santa Barbara, California, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2101-3/13/12 ...$15.00.
http://dx.doi.org/10.1145/2535372.2535376.

can significantly impact the available design space of a later stage (e.g., packet filters), potentially resulting in a substantial amount of unnecessary complexity.

We observe that in practice, in addition to reachability control, many other common and important operational objectives are also achieved through designing multiple networking elements at different stages. Two prominent examples are (i) quality of service (QoS) which involves end-to-end traffic engineering, source policing, and per link bandwidth management; and (ii) resiliency which requires planning of both physical and logical topologies. For these important objectives, the current "divide-and-conquer" design approach is incapable of eliminating all the unnecessary complexity.

In this paper, we investigate a novel integrated top-down methodology that jointly designs multiple networking elements involved in achieving a common objective. The key components of our approach include: (i) for a given design objective, identifying all the networking elements that may be involved in its implementation, and their interactions (i.e., how the design of one element could affect the design of others); (ii) characterizing the source of complexity in each element, leveraging recently-developed complexity metrics; (iii) formulating the design problem as one of minimizing the total complexity of all involved elements, subject to correctness and feasibility constraints; and (iv) developing specific algorithms and heuristics to solve the formulated problems. As such, this new approach goes substantially beyond the state-of-the-art "divide-and-conquer" approaches. It requires not only entirely new formulations and algorithms, but also fundamentally new abstractions and models in order to integrate multiple design steps into a unified framework.

Our integrated design methodology is general and can be applied to a variety of network design objectives and scenarios. In order to demonstrate its feasibility and power at sufficient depths, in this paper we focus on one concrete application of the methodology: user reachability control. We choose reachability control because (i) security is of vital importance to virtually every enterprise network, and (ii) the design involves at least three networking elements, and as such it is both highly challenging and at the same time may benefit greatly from the new approach.

Similarly, while our approach is agnostic to the type of network complexity metric used[1], the focus of this paper is on minimizing the *configuration complexity*, specifically the amount of *command dependencies* [7] in the router configurations of the resulting network. According to recent studies [7, 31, 32], these dependencies are directly linked to the operational cost as they require substantial manual effort to configure correctly in the initial implementation and manage in subsequent evolutions, and if not maintained properly, can lead to serious issues such as application performance degradation and security breaches.

We evaluate the benefits of the new approach in the context of the heuristics we have developed for solving the formulated design problem of user reachability control. The evaluation is conducted on a large university campus network with a few thousand user hosts. The results show that our approach can effectively reduce the number of packet filter rules and the number of VLAN trunk ports by more than 85% and 70%, respectively, when compared to the ad-hoc approach currently used by the operators.

The rest of the paper is organized as follows. In Section 2, we briefly survey the state of the art. In Section 3, we first substantiate the need for the integrated design approach using a detailed example of reachability control design. We then demonstrate how the integrated approach can be applied to reachability control, and formally introduce an integrated design framework for this problem. In Sections 4 and 5, we present new formulations and novel heuristics to accomplish the two design problems identified by the framework: joint design of VLAN and packet filter, and joint design of IP allocation and packet filter. Section 6 describes a through evaluation of our heuristics in a large campus network setting. Possible extensions and open issues are discussed in Section 7. Finally, we conclude the paper and briefly outline our plan for future work in Section 8.

2 State of Art of Network Design

In this section, we overview the current state of the art of enterprise network design, specifically focusing on the more recent developments in top-down design techniques. Our aim is to not only discuss related work, but also provide a historical perspective of the proposed integrated design approach before we present detailed examples to substantiate how the new approach may reduce complexity in the next section.

2.1 Operational Practice and Tools

The operational community has a rich history of crafting the art of network design and reconfiguration. Nonetheless, the state of the practice by operators is still defined predominantly by ad-hoc, manual decision making. Notable efforts to simplifying network design involve template-based approaches that codify and promote best practices [1, 2, 3, 4] and abstract languages to specify configurations in a vendor-neutral fashion [12]. There are also tools such as PRESTO [13] to convert a network design into device-vendor-specific configuration commands. These approaches merely model the low-level mechanisms and their configuration. They do not model network-wide operator intent such as reachability and manageability. A logic-based approach to configuration generation based on model-finding is presented in [25]. The focus is on the generation of configuration parameters conforming to correctness rules distilled from best practices, and the system does not take complexity into consideration. Many works have approached the problem of minimizing the number of rules in a single access control list (ACL) (e.g., [23]). In contrast, we focus on minimizing the total number of packet filter rules required for a given network to meet all its reachability control requirements.

Finally, various design guidelines including those for a top-down network design approach [27] can be found in the literature. These guidelines provide practical insights into the trade-offs of different design choices regarding topology, hardware and protocols. However, considerable manual effort is required to determine how to apply these guidelines to the design of a network of medium to large size.

2.2 Systematic Multi-Stage Design

Systematic network design, characterized by the use of a formal model to generate configuration that is *provably* correct and additionally *optimizes* certain performance metrics, has emerged as a potential solution to the challenges facing the operational community. Early efforts on this front focus on tasks encountered in carrier networks, such as configuring BGP policies [9, 14, 16, 17], optimizing OSPF weights [29], and redundancy planning.

More recent studies [32, 33] target enterprise networks specifically. They employ a "divide-and-conquer" strategy and perform network design in a stage-by-stage fashion, e.g., effectively treating VLAN design and packet filter design as two completely independent tasks. While these studies have advanced the state of the art of systematic network design, their models may produce designs with unnecessarily high configuration complexity, as we will elaborate

[1] Section 7 provides a brief discussion of other potential complexity metrics.

in Section 3. It is this unnecessary complexity that this work seeks to expose and minimize.

It should be noted that the recent progress in systematic network design owes largely to new abstractions from related work in several areas, including (i) characterization of the designs of production networks (e.g., [22]), (ii) static analysis of network properties (e.g., [21, 35]), and (iii) the formulation of new configuration complexity metrics [7].

2.3 Software Defined Networking

To combat network complexity, researchers have started investigating new software-defined networking (SDN) architectures based on logically centralized controllers and declarative configuration languages (e.g., Frenetic [15]). These approaches have the potential to simplify network design by shifting complexity away from configuration of many individual devices to programming of few centralized controllers. While SDN has shown potential, challenging problems such as the need to update network devices in a consistent fashion [30] must be resolved before it can be widely deployed.

More importantly, SDN operators must carry out a similar design task of translating high-level reachability control requirements into flow rules. Since these flow rules will be installed on demand in the Ternary Content Addressable Memory (TCAM) of switches and once installed, checked for each packet passing through, it is desirable to minimize the number of such rules required. In this way, the design methodology and heuristics presented in this paper also apply to an SDN setting, as further discussed in Section 7.

3 An Integrated Design Framework for Reachability Control

We now apply the integrated top-down approach as described in Section 1 to the user reachability control problem. With an illustrative example scenario, we identify the networking elements that are involved in realizing this important design objective, understand the source of configuration complexity of each element, and capture how the designs of individual elements may interact with each other and impact the overall complexity. We then present a framework for achieving an integrated design of user readability control.

3.1 An Illustrative Example Scenario

Our example is based on the toy network shown in Fig. 1a. There are two departments: Engineering and Financial. Each department has users in multiple locations as shown. In addition, there is a set of servers. The reachability control policy is that the servers should only be accessed by Financial users. The following design steps are needed to implement the policy: (i) grouping the hosts into VLANs; (ii) assigning subnet addresses to VLANs; and (iii) installing a filter restricting access to the servers. We are given the following design constraints: at most three VLANs can be created; and the available IP blocks are 10.0.1/24, 10.0.2/24 and 10.0.3/24.

Fig. 1b illustrates a first possible design, where hosts are grouped into VLANs solely based on their physical locations. Unfortunately, this grouping scheme makes configuring the filter difficult: it is not possible to express the rules at the level of subnet prefixes for VLAN 10 or VLAN 20, because they both contain hosts from both departments. As such, we have to write filter rules at the level of IPs to permit individual Financial hosts, which results in an explosive growth of rules. We note that each packet filter is a sequential collection of filter rules, and each filter rule contains a pattern to be matched against packet headers, and an action (i.e., permit or deny) to be applied to packets whose header matches the corresponding pattern. The pattern part may be configured to match specific values of all or any subset of the five header fields: source and destination IPs and ports, and protocol; and thus creates static dependencies on those filed values. Such dependencies must be manually configured and maintained and thus are a major source of configuration complexity.

Fig. 1c depicts a different design, where hosts in the same VLAN belong to the same department. This design enables expressing filter rules at the level of subnet prefixes, which significantly reduces the number of rules and thus simplifies the filter configuration. However, this design suffers from a different kind of configuration complexity: it requires configuring a large number of VLAN *trunk ports*, as denoted by the bold lines in the figure. Trunk ports are the switch ports that connect to another switch. Since each VLAN constitutes a separate broadcast domain, it is important to properly constrain broadcast traffic to eliminate unnecessary broadcast traffic for increased performance and security. More specifically, every switch-to-switch link (called "trunk link") must be configured to only allow traffic of appropriate VLANs. This is achieved by manually configuring the corresponding trunk ports to permit those specific VLANs. VLAN trunk ports configuration is widely considered a major source of complexity, as operators must manually identify and configure the correct set of VLANs to allow for each port. (Readers are referred to Sec.II-A of [32] for a more detailed explanation of trunk ports).

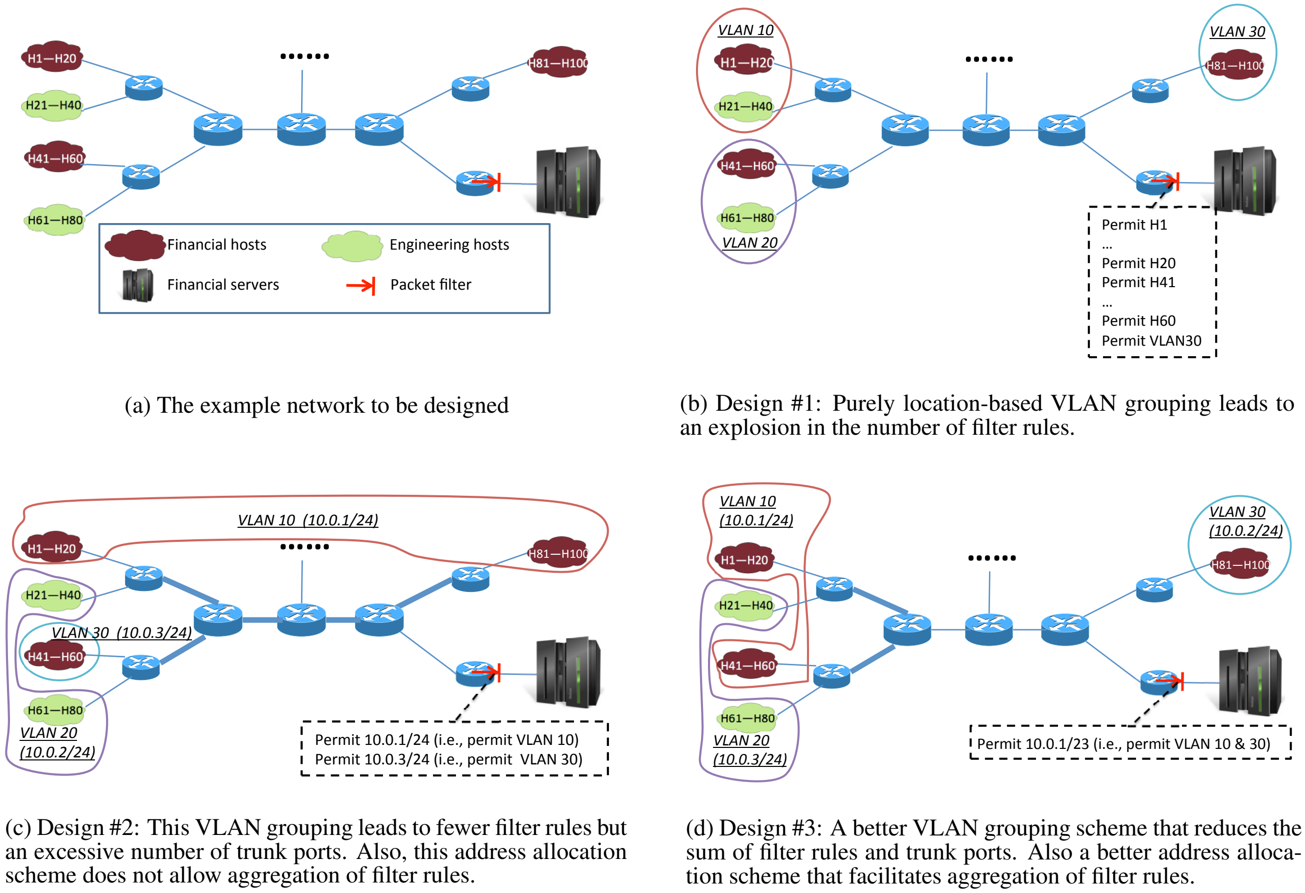
Fig. 1d shows a third design. The design still enables expressing filter rules at the subnet prefix level, but also significantly reduces the amount of trunk port configuration by requiring fewer VLAN trunk ports. Furthermore, the IP allocation scheme of this design is better than the previous one as it facilitates prefix aggregation when expressing rules, i.e., it allows aggregating the two original filter rules ("permit 10.0.1/24" and "permit 10.0.2/24") into a single /23 rule ("permit 10.0.1/23") as shown.

We make two observations from this example. First, for the same target network, there exist multiple designs that are all *correct*. For our example network, there are at least 6 different designs (three different VLAN grouping schemes coupled with two different address allocation schemes.) However, different designs have different levels of configuration complexity. Second, the complexity of the resulting network is determined by both the VLAN configuration complexity (characterized by the number of trunk ports) and the packet filter complexity (characterized by the number of filter rules). Furthermore, the packet filter design is directly impacted by both the VLAN grouping scheme and the IP address allocation scheme. Thus, a design approach clearly will not work well if it treats VLAN grouping and IP allocation as total independent tasks and ignore their inherent interactions. For example, current top-down design approaches (e.g., [32, 33]) consider VLAN design as an isolated task, and thus they will solely seek to minimize the number of trunk ports without considering how doing so will impact the packet filter design, i.e., they will pick the first design shown in Fig. 1b for our toy example, which is clearly not the best.

3.2 Elements of Reachability Control and Their Interactions

Realizing a host-level reachability control consists of designing the following networking elements: VLAN, IP address allocation, routing, and packet filters. We discuss below the role of each of these elements, their impact on configuration complexity, and their inherent interactions.

First, the VLAN design directly determines the number of trunk ports that need to be configured and maintained in the resulting network, a major source of operational complexity. Furthermore, the VLAN design also significantly impacts the packet filter complexity, as it determines how hosts are grouped into subnets. Intuitively, if the VLANs align well with reachability policy boundaries (e.g.,



(a) The example network to be designed

(b) Design #1: Purely location-based VLAN grouping leads to an explosion in the number of filter rules.

(c) Design #2: This VLAN grouping leads to fewer filter rules but an excessive number of trunk ports. Also, this address allocation scheme does not allow aggregation of filter rules.

(d) Design #3: A better VLAN grouping scheme that reduces the sum of filter rules and trunk ports. Also a better address allocation scheme that facilitates aggregation of filter rules.

Figure 1: multiple designs with different complexity characteristics exist for one network.

hosts in the same VLAN are subject to the same policies), then policies may be efficiently expressed at the level of subnet prefixes, resulting in the minimal number of filter rules. On the other hand, if VLANs are ill-aligned with policy boundaries (e.g., a single VLAN contains hosts with completely different policy requirements), then filter rules may have to be expressed at the level of individual IP addresses, leading to a large number of rules.

Second, the IP allocation scheme determines how packet filter rules may be aggregated and thus impacts the number of filter rules in the resulting network. Intuitively, a good IP allocation scheme can minimize the number of filter rules by assigning aggregatable IP blocks to VLANs/subnets that are subject to similar reachability policies, so that a single filter rule can cover multiple subnets by using an aggregated prefix. In contrast, a naive IP allocation scheme that assigns IP prefixes solely based on the physical location of the subnets is not likely to minimize complexity. As another real-world example, we have observed that some operational networks employ an IP allocation scheme that matches the third octet of every subnet prefix address to the corresponding VLAN ID, e.g., VLAN 100 will be assigned prefix x.y.100/24. Such naive approaches treat IP allocation as an isolated design task and try to simplify the allocation scheme itself, but they fail to systematically consider how the IP allocation will impact the aggregation of filter rules.

Third, the routing design also impacts packet filter complexity. As a principle to ensure the correctness of design, if traffic between two subnets S_i and S_j is subject to filtering, then a filter must be placed on *every* possible layer-three path between the two subnets [33]. Routing design determines the layer-three topology, and thus impacts the number of packet filters needed and where they should be installed.

Finally, packet filters implement the reachability control policy, and is another major source of configuration complexity. As discussed above, its configuration complexity is impacted by the VLAN design, IP address allocation scheme, and routing design.

Scope of this work: We observe that in practice, packet filters are typically only placed at the network edge, i.e., on the gateway routers of subnets. This design pattern has two major benefits. First, it guarantees that traffic to/from a subnet will always be filtered while simplifying the filter placement. That is, it relieves operators from having to find out all the possible layer-three paths between subnets. Second, all policies regarding a particular subnet are implemented in a single location (i.e., its gateway router), which maximizes the opportunity to compress the filter rules through prefix aggregation, and thus simplifies the configuration of filter rules. Given this observation, in this paper we assume that packet filters can only be placed on the gateway routers of subnets. This assumption gives the additional benefit that we no longer need to consider routing design further, since the filter placement is now fixed and independent of the layer-three topology. We do wish to acknowledge that systematic routing design is a challenging research problem on its own. We leave a more comprehensive investigation of designs where filters may be placed anywhere in the network to future work, and focus on VLAN design, IP allocation, and packet filter design in this paper.

3.3 Formulating the Reachability Control Design Problem

Following the integrated design methodology described in Section 1, we now present a design framework for reachability control, which integrates the design of the individual elements identified above. In doing so, our goal is to enable the design process to be fully automated, and require only high-level specifications from operators. We first present a new abstraction that facilitates specifying and modeling reachability policy, and then present the framework.

3.3.1 A New Abstraction for Specifying Reachability Policies

An essential input to our framework is the reachability control policies, and it is important to consider how they should be specified. The current "divide-and-conquer" design approach [33] requires reachability policies to be specified at the VLAN/subnet level, i.e., it requires operators to specify a reachability matrix where each cell (i, j) denotes the reachability between VLAN i and VLAN j. This abstraction works for the "divide-and-conquer" approach which assumes that the VLAN design has already been completed before designing packet filters. However, it does not work for our approach which integrates the design of VLANs and packet filters, i.e., our framework cannot use such a VLAN-level reachability matrix as input because VLANs themselves are to be determined by the solution. In addition, we believe that the VLAN-level reachability matrix is too low level as a policy abstraction, and it is tedious for operators to specify reachability policy using it.

In this work, we introduce a new abstraction for specifying reachability policy: a reachability matrix at "user role" level. We define a *user role* as a logical category that a set of users or servers belong to. Example user roles include faculty users, Computer Science users, financial servers, etc. Note that a user may have multiple roles, e.g., a CS professor can have both roles of CS users and faculty users. Each cell (i, j) of the reachability matrix specifies reachability policy between a user role i and another user role j. The advantage of this abstraction is that it allows policies to be specified at a high level and independent of any design details.

3.3.2 Formulating Reachability Control Design

We formulate the design problem of reachability control as follows. We assume we are given the physical topology of the network, and the set of users/servers and their network locations. For each user/server, we are given its user roles. We are given the reachability matrix at the user-role level as described above. In addition, we are given the maximum number of VLANs that can be created (denoted by N), and the available IP blocks. The design framework includes tasks of (i) mapping the set of users to at most N VLANs, (ii) assigning IP blocks from the available IP space to the created VLANs, and (iii) configuring packet filters to enforce the reachability policies. Our goal is to minimize the total configuration complexity of the resulting network. As discussed above, the configuration complexity (denoted as C_{total}) includes both VLAN-related complexity (denoted by C_v), measured by the number of trunk ports, and filter-related complexity (denoted by C_f), measured by the number of filter rules. Formally, we model the total configuration complexity as:

$$C_{total} = W_v * C_v + W_f * C_f \tag{1}$$

where W_v and W_f are the weight factors given to the two complexity categories, and can be customized by operators. For example, if the operators of a network consider VLAN trunk ports more difficult to configure and maintain than filter rules, they can assign W_v a higher value than W_f.

We notice that, while the VLAN grouping scheme and the IP allocation scheme both impact the packet filter complexity, VLAN design and IP allocation scheme are completely independent of each other, i.e., the design choices made in VLAN grouping won't affect the available design space of the IP allocation scheme, and vice versa. Given this insight, we are able to formulate the design of reachability control as two joint design problems in order to make it more tractable:

- Joint design of VLANs and packet filters;
- Joint design of IP allocation scheme and packet filters.

The output of the first joint design includes the VLAN grouping scheme, and an intermediate representation of packet filter rules expressed in terms of individual VLANs and hosts. This intermediate representation of packet filters then becomes part of the input to the second joint design. The output of the second joint design includes the IP allocation scheme, and complete packet filters expressed in terms of IP address blocks. In the next two sections, we formulate and solve the two joint design problems.

4 Joint Design of VLANs and Packet Filters

We first present models for formulating the joint design problem, and then develop heuristics for solving the formulated problem.

4.1 Formulating the Joint Design Problem

This design task is to map hosts to a set of VLANs and to derive packet filter rules expressed in terms of individual VLANs and hosts. There are several important considerations in doing so, as we detail below.

VLAN count: The total number of VLANs that can be created in the design is determined by the hardware used in the network. This is because each VLAN runs its own instance of spanning tree, which consumes the memory and CPU resource of the switches. For example, a Cisco Catalyst 2950 switch can only support up to 64 spanning tree instances [11]. To model this constraint, we simply assume that operators will specify the maximum number of VLANs that can be created, which is denoted by N.

VLAN size: A VLAN becomes a separate subnet at layer three, and thus the number of hosts that a VLAN can have is bounded by the size of the IP address block assigned to the corresponding subnet (assuming NAT is not used). For example, it is a common practice to limit the maximum size of a VLAN to that of a /24 subnet, i.e., at most 254 hosts. We assume the operators will specify the maximum VLAN size, denoted by *MAX-VLAN-SIZE*.

Correctness criteria: To ensure the correctness of the design, the following two conditions must be satisfied. First, the given reachability policies must be correctly implemented through packet filters. Second, all hosts in the same VLAN must have full reachability toward each other, since they are all in the same broadcast domain.

Configuration complexity: This design will determine the VLAN configuration complexity C_v (i.e., the total number of VLAN trunk ports in the resulting network). Further, it will also impact the packet filter configuration complexity. Note that the filter rules generated by this design are expressed in terms of individual VLANs and hosts. The VLANs and hosts will be assigned IP addresses in the second joint design, and thus the filter rules could be further aggregated when converted to the IP representation in that design. Thus, we model the total configuration complexity introduced by this design (denoted by C'_{total}) as follows:

$$C'_{total} = W_v * C_v + W_f * C'_f \tag{2}$$

W_v, W_f and C_v have been defined for Equation (1). C'_f is the configuration complexity of the packet filters generated by this design task, measured as the total number of filter rules. Clearly $C'_f \geq C_f$

as the joint design of IP allocation and packet filters may further reduce the number of filter rules through prefix aggregation.

Now we can formally formulate this joint design problem as:

Minimize: C'_{total}
Subject to:
- the correctness criteria, and
- the constraints on VLAN number and size.

4.2 Heuristic for Solving the Joint Design Problem

We present the details of our heuristic that works in a step-by-step fashion. For ease of understanding, we use a running example to illustrate the various algorithmic operations. The example network setup is shown in Fig. 2a. There are eight user roles: Biology, Computer Science, IT, Faculty, Students, managers, operators, and servers. The reachability policies are also shown in the graph. We are given that $N = 6$, and *MAX-VLAN-SIZE* $= 254$. For simplicity, we assume that the operator has chosen to set $W_v = W_f = 1$.

4.2.1 Step 1: Map Policy Groups to VLANs

As illustrated in Section 3.1, it is often desirable for a VLAN to contain hosts subject to the same reachability policy, because doing so enables filter rules to be written at the level of an entire VLAN. To capture this insight in the design process, we leverage the abstraction of "policy groups" introduced by recent works [7, 31] including our own for network modeling. A *policy group* abstracts the set of hosts that are (i) subject to the same reachability policy towards other hosts and (ii) have full reachability among themselves. Clearly the set of policy groups forms a partition of all hosts. It is easy to see that a policy group is an atomic unit in deriving filter rules, i.e., if a packet filter allows traffic from one host in a policy group, it must also allow traffic from all the other hosts of the same policy group. Thus, the use of policy groups in the design process simplifies the reasoning of reachability control by allowing us to reason about groups of hosts together instead of individual ones. We believe the set of policy groups can be straightforwardly derived from the inputs of user roles and the role-level reachability matrix, but omit the details due to lack of space.

As a reasonable starting point of the design, we initially let each policy group become a separate VLAN. We then derive the filter rules. As mentioned in Sec.3.2, we have assumed that packet filters can only be placed on the gateway routers of the VLANs to be protected. Thus the filter rules can be determined in a straightforward way: for each VLAN, the corresponding packet filter permits all other VLANs (i.e., policy groups) that can communicate with this VLAN, according to the reachability matrix. We assume an implicit deny in the end of a packet filter, following the vendor convention. Filters that simply permit all traffic are omitted.

Fig. 2b illustrate the design after this step. Seven policy groups are identified straightforwardly from the inputs: CS faculty (shown as CS-F on graph) which resides in two different locations, CS students (CS-S), Biology faculty (Bio-F), Biology students (Bio-S), IT managers (IT-M), IT operators (IT-O) and servers (SVR). Each policy group has been placed in a separate VLAN. For example, the entire CS-Faculty policy group becomes VLAN *V4*. The corresponding VLAN trunk ports to be configured are shown by the bold links connecting those ports. The packet filters are also shown, and as expected all filter rules are expressed at the VLAN level. Finally, the amount of configuration complexity in terms of filter rules and trunk ports after this step is also shown.

4.2.2 Step 2: Selectively Partition VLANs with Large Span

For each VLAN created in Step 1, we now evaluate whether it is beneficial (i.e., leading to smaller C'_{total}) to partition it into two smaller VLANs. If so, we will execute the partitioning, and iteratively evaluate for the resulting two smaller VLANs. We repeat this step for every VLAN until we cannot further reduce C'_{total} by partitioning existing VLANs. Our insight for this step is as follow.

On one hand, partitioning a VLAN that has a large span could potentially reduce C'_{total} as it could significantly reduce the number of trunk ports (i.e, C_v). Consider *V4* (CS-Faculty policy group) in Fig. 2b as an example. By partitioning it into two smaller VLANs, i.e., the new *V4* and *V8* in Fig. 2c, we eliminate the need for any trunk port for this VLAN, and thus reduce C_v. On the other hand, partitioning a VLAN could also potentially increase C'_{total} as it could lead to more filters and/or filter rules required, i.e., an increase in C'_f. There are two reasons for this. First, after the partitioning it may be necessary to install a *new* packet filter to protect a newly created VLAN. For example, in Fig. 2c there is a new packet filter that protects the newly created *V8*, which introduces 6 new rules. Second, it may be necessary to add additional rules in the *existing* filters, to permit a newly created VLAN. For example, in Fig. 2c a rule "permit V8" is added to four existing filters.

More specifically, we employ the K-means clustering algorithm (with K=2) to decide how a VLAN should be partitioned into two, such that the reduction in C_v is maximized. In configuring the clustering algorithm, we let each host in the VLAN be a node, and the distance between two nodes be the length of the shortest layer-two path between the corresponding hosts. The clustering algorithm then groups nearby hosts into the same VLAN and thus minimizes the need of trunk ports.

For our running example, we find that by partitioning the old VLAN *V4* in Fig. 2b into two smaller VLANs *V4* and *V8* in fig.2c, we reduces C_v (i.e., the number of trunk ports) by 12, but increases C'_f (i.e., the number of filter rules) by 10. As we assume $W_v = W_f = 1$, the total complexity is reduced by 2, according to Equation (2). Hence, we execute the partitioning since it is beneficial to do so. We also find that it is not beneficial to partition any other VLAN. Fig. 2c shows the resulting design after this step.

4.2.3 Step 3: Partition VLANs with Too Many Hosts

This step ensures that the constraint on VLAN size is met. It checks each VLAN in the current design to see whether it contains more hosts than the specified *MAX-VLAN-SIZE*. If so, it again uses the K-means clustering algorithm described in the previous step to partition the VLAN into two. This process iterates until all VLANs have been reduced to a size no larger than the *MAX-VLAN-SIZE*.

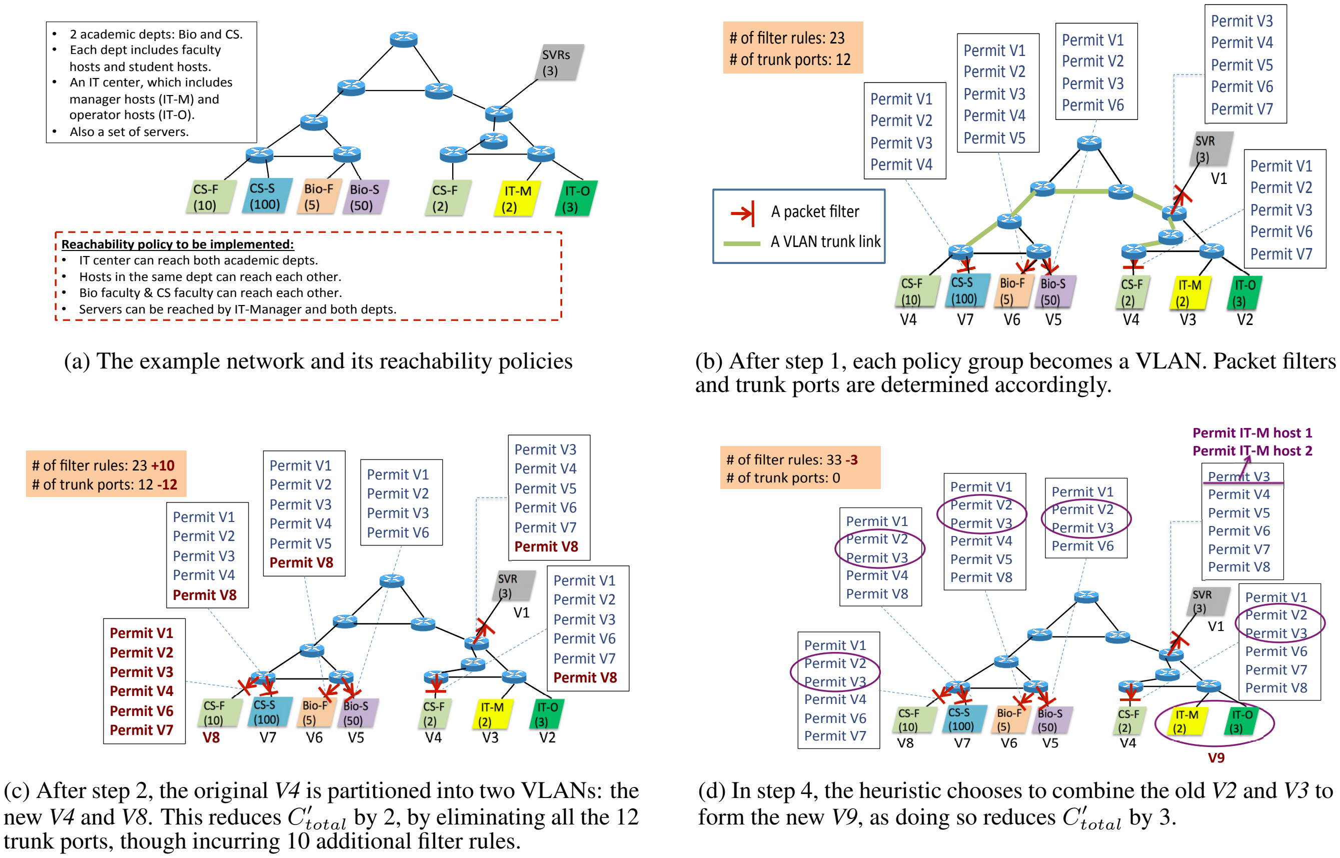
For our running example, Since none of the VLANs contains more than 254 hosts, this step will not partition any VLAN.

4.2.4 Step 4: Selectively Combine Multiple VLANs

This step has two purposes: further reducing the total complexity C'_{total}, and also ensuring that the constraint on the VLAN count is met. It achieves both by selectively combining pairs of VLANs in an iterative process as described below.

For every *eligible* pair of VLANs, the heuristic evaluates the complexity impact of combining them. A pair of VLANs is eligible to be combined if (i) the sum of the hosts in both VLANs is not greater than *MAX-VLAN-SIZE*, and (ii) the hosts in both VLANs have full reachability toward each other. For every eligible VLAN pair, we calculate the potential change in C'_{total} if the two were combined into a single new VLAN. We then select the pair with the maximum reduction in C'_{total} to execute the combining. We repeat this process until the following two conditions are *both* met:

- The total number of VLANs is not greater than N (i.e., the maximum number of VLANs that can be created); and,
- It is not possible to further reduce C'_{total} by combining any more eligible pair of VLANs.


• 2 academic depts: Bio and CS.
• Each dept includes faculty hosts and student hosts.
• An IT center, which includes manager hosts (IT-M) and operator hosts (IT-O).
• Also a set of servers.
SVRs (3)
CS-F (10)
CS-S (100)
Bio-F (5)
Bio-S (50)
CS-F (2)
IT-M (2)
IT-O (3)
Reachability policy to be implemented:
• IT center can reach both academic depts.
• Hosts in the same dept can reach each other.
• Bio faculty & CS faculty can reach each other.
• Servers can be reached by IT-Manager and both depts.
# of filter rules: 23
# of trunk ports: 12
A packet filter
A VLAN trunk link
Permit V1 Permit V2 Permit V3 Permit V4
Permit V1 Permit V2 Permit V3 Permit V4 Permit V5
Permit V1 Permit V2 Permit V3 Permit V6
Permit V3 Permit V4 Permit V5 Permit V6 Permit V7
Permit V1 Permit V2 Permit V3 Permit V6 Permit V7
SVR (3) V1
V4 V7 V6 V5 V4 V3 V2
# of filter rules: 23 +10
# of trunk ports: 12 -12
Permit V1 Permit V2 Permit V3 Permit V4 Permit V8
Permit V1 Permit V2 Permit V3 Permit V4 Permit V5 Permit V8
Permit V1 Permit V2 Permit V3 Permit V6
Permit V3 Permit V4 Permit V5 Permit V6 Permit V7 Permit V8
Permit V1 Permit V2 Permit V3 Permit V6 Permit V7 Permit V8
Permit V1 Permit V2 Permit V3 Permit V4 Permit V6 Permit V7
V8 V7 V6 V5 V4 V3 V2
# of filter rules: 33 -3
# of trunk ports: 0
Permit IT-M host 1
Permit IT-M host 2
Permit V1 Permit V2 Permit V3 Permit V4 Permit V8
Permit V1 Permit V2 Permit V3 Permit V4 Permit V5 Permit V8
Permit V1 Permit V2 Permit V3 Permit V6
Permit V3 Permit V4 Permit V5 Permit V6 Permit V7 Permit V8
Permit V1 Permit V2 Permit V3 Permit V6 Permit V7 Permit V8
Permit V1 Permit V2 Permit V3 Permit V4 Permit V6 Permit V7
V8 V7 V6 V5 V4 V9

(a) The example network and its reachability policies

(b) After step 1, each policy group becomes a VLAN. Packet filters and trunk ports are determined accordingly.

(c) After step 2, the original *V4* is partitioned into two VLANs: the new *V4* and *V8*. This reduces C'_{total} by 2, by eliminating all the 12 trunk ports, though incurring 10 additional filter rules.

(d) In step 4, the heuristic chooses to combine the old *V2* and *V3* to form the new *V9*, as doing so reduces C'_{total} by 3.

Figure 2: An running example for illustrating the operations of our heuristics of our heuristics for joint design of VLAN and packet filters.

To understand why combining VLANs could possibly lead to reduction in C'_{total}, consider *V2* and *V3* in Fig. 2c as an example. If we combine those two VLANs into the new *V9* as illustrated in Fig. 2d, then for all packet filters that need to permit both *V2* and *V3* by using two separate rules, they now only need to permit the new *V9* using a single rule, leading to a reduction of rules.

However, this benefit does not come without potential penalty. The penalty is two-fold. Fist, if there is any packet filter that permits only one of the two original VLANs, then it cannot permit the combined new VLAN. For example, in Fig. 2c the packet filter protecting *V1* (i.e., the servers) only permits *V3* but not *V2*. So after *V2* and *V3* are combined to form the new *V9* as shown in Fig. 2d, the filter cannot simply change to permit *V9* instead, because doing so would wrongfully grant access to hosts in the original *V2*. Hence, the filter now has to permit *individual hosts* in *V3* as shown in Fig. 2d, leading to an increase in the number of filter rules. Second, combining two VLANs could also require configuring additional VLAN trunk ports, if the two VLANs are in different locations. Though in our example this is not the case as *V2* and *V3* connect to the same switch.

For our running example, the heuristics will choose to first combine *V2* and *V3* to form the new *V9*, because doing so results in a reduction in C'_f by 3 while keeping C_v unchanged. This is illustrated in Fig. 2d. In fact this is the only pair of VLANs that will result in a reduction in C'_{total} if combined. All other VLAN pairs when combined will cause C'_{total} to increase. However, since the total number of VLANs after combining *V2* and *V3* is 7, which is greater than the given limit of $N = 6$, another pair of VLANs has to be combined. The heuristic will again evaluate all eligible pairs and then choose to combine *V1* and *V4* to form the new VLAN *V10*, as doing so results in the least increase in C'_{total} (C_v and C'_f will be increased by four and two respectively). After that, both conditions listed above are met and this step stops.

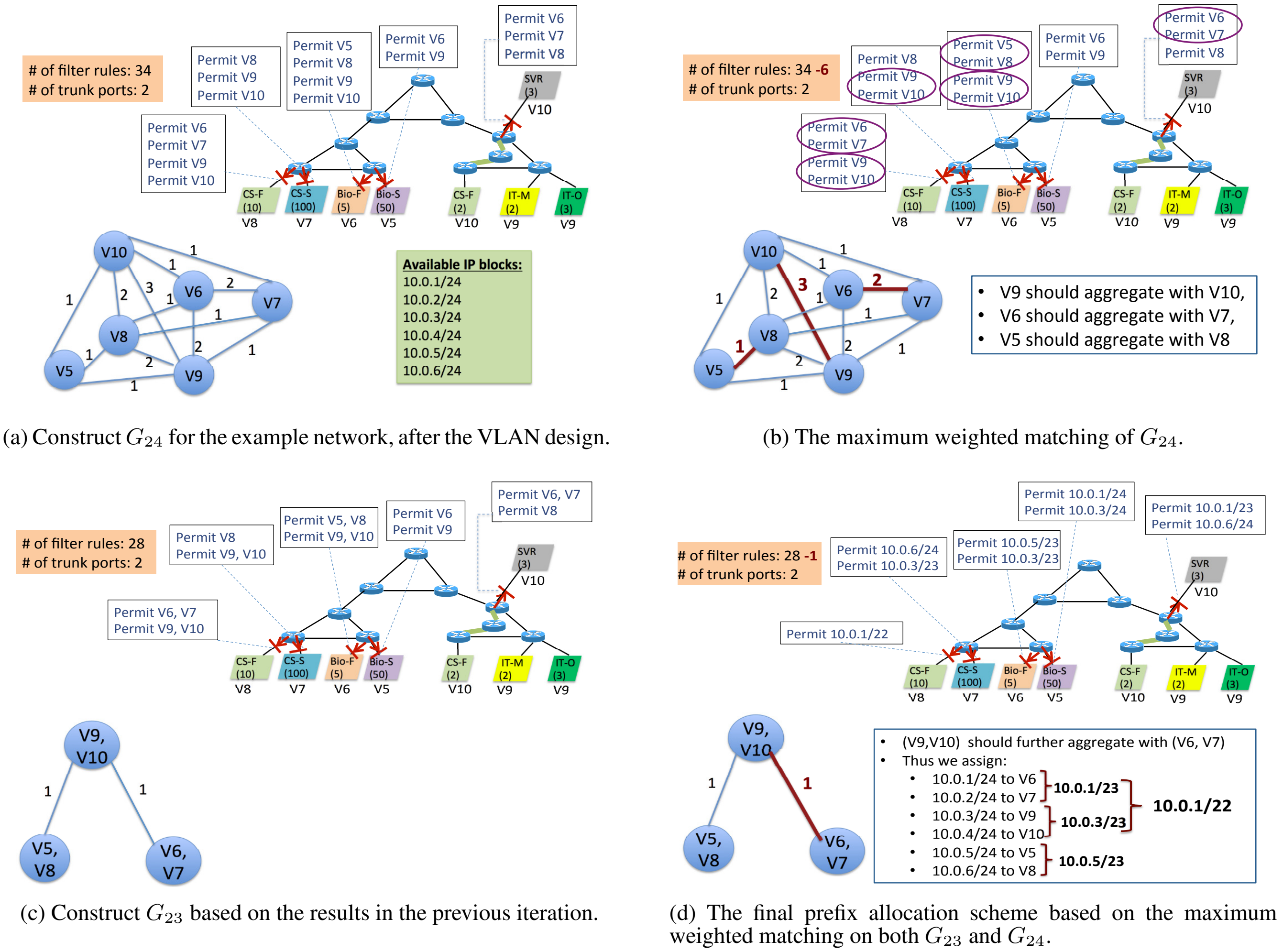
5 Joint Design of IP Allocation and Packet Filters

We first formulate the joint design problem, and then present a heuristic solution based on finding the maximum weighted matching on a graph. In describing the heuristic, we continue to use the same running example from the previous section.

5.1 Formulating the Joint Design Problem

This design task is performed after the joint design of VLANs and packet filters that is presented in the previous section. The inputs are: (i) the VLAN grouping scheme; (ii) the packet filters in the intermediate representation (i.e., expressed in terms of individual VLANs and hosts); and (iii) available IP blocks. The goal of this design is to find the optimal scheme of allocating IP prefixes to VLANs such that the resulting number of filter rules is minimized.

For example, in Fig. 3a, the packet filter that protects *V6* contains four rules to separately permit *V5*, *V8*, *V9* and *V10*. However, if two aggregatable prefixes (say 10.0.1/24 and 10.0.2/24) are assigned to *V5* and *V8*, then the two VLANs can be permitted together in one rule that permits the aggregated prefix 10.0.1/23. Even better, if prefixes 10.0.3/24 and 10.0.4/24 are also assigned to *V9* and *V10*, then further aggregation can be achieved and the filter will need only a single rule "permit 10.0.1/22" to permit all four VLANs. Further, since multiple packet filters are typically involved, the ad-



(a) Construct G_{24} for the example network, after the VLAN design.

(b) The maximum weighted matching of G_{24}.

(c) Construct G_{23} based on the results in the previous iteration.

(d) The final prefix allocation scheme based on the maximum weighted matching on both G_{23} and G_{24}.

Figure 3: The example continued from previous section for illustrating operations of our heuristic for assigning IP prefixes to VLANs.

dress allocation scheme should prioritize the assignment of aggregatable prefixes based on how frequently the candidate VLANs appear together and receive the same treatment (i.e., permit or deny) in all filters, in order to minimize the total number of filter rules.

Our focus in designing the address allocation scheme is on the VLAN/prefix level, i.e., we focus on assigning IP prefixes to VLANs. We do not consider how IPs should be assigned to *individual* hosts *inside* each VLAN. Even though carefully assigning IPs to individual hosts could potentially enable intra-VLAN IP aggregation and reduce the number of filter rules concerning individual hosts, we believe this may be too low level that it is impractical to require operators to configure and track individual IP assignment. In practice, DHCP is often used so that hosts will receive their IPs automatically from the IP blocks assigned to their VLANs/subnets. Thus, we simply assume that individual IP assignment to hosts is done randomly, and do not consider possible aggregation of individual host IPs. We do wish to note that the heuristic presented below can be directly applied to finding the optimal individual host IP assignment, if the operators wish to do so.

Correctness criteria: To ensure the correctness of the design, the following two conditions must be satisfied. First, the prefixes assigned to VLANs must be chosen from the pool of available IP blocks. Second, when aggregating filter rules, the given reachability control policies must be correctly implemented.

Configuration complexity: This design will determine the final packet filter complexity C_f (i.e., the total number of filter rules in the resulting network).

Formally, the joint design of prefix allocation and packet filters can be formulated as follows:

Minimize: C_f
Subject to: the correctness criteria.

5.2 Heuristic for Solving the Joint Design Problem

The key idea for solving the problem is to model it as *finding the maximum weighted matching on a graph*. Our solution works in iterations over the prefix lengths, starting from /32, then /31, then /30, and so on (i.e., in each iteration the prefix length is decreased by 1). In the iteration that concerns prefix length l, we construct a graph G_l whose vertices are prefixes of length l that are available and can be assigned to VLANs. There is an edge between two vertices, if the corresponding VLANs *appear together and receive the same treatment* in at least one filter. The weight of an edge is defined as the number of filters in which the two corresponding VLANs appear together and receive the same treatment.

To illustrate, Fig. 3a shows our running example continued from the previous section. Note that only rules concerning an entire VLAN are shown here, and rules concerning individual hosts are omitted, as the focus here is on assigning prefixes to VLANs. The graph G_l is empty for iterations concerning prefix length from /32 to /25, as the VLANs in this particular example are all of /24. In the iteration concerning /24, the corresponding graph G_{24} is shown in the lower left part of Fig. 3a. The numbers shown on the edges are the weights. For example, the weight of the edge (*V9*, *V10*) is 3, as the two VLANs being permitted together in three packet filters.

Now the design problem of finding the best allocation scheme of prefixes of length l can be solved by finding the maximum weighted matching on G_l. A *matching* on a graph is defined as a subset of edges such that none of them share a common vertex. The *maximum weighted matching* is defined as a matching for which the sum of the weights of the matched edges is as large as possible. We note there exist algorithms (e.g., [5]) that take O(n^3) to find the maximum weighted matching on a general undirected graph, where n is the number of vertices. Now in our context, for each edge included in the maximum weighted matching, the corresponding two VLANs should be assigned aggregatable prefixes. It is easy to see that by maximizing the weight of the selected matching on G_l, we maximize the opportunity to reduce the number of filter rules through prefix aggregation.

We leverage the algorithm described in [5] to find the maximum weighted matching for our running example, and the result is marked in red on the G_{24} graph shown in Fig. 3b. According to the result, we should assign aggregatable prefixes to *V9* and *V10*, to *V6* and *V7*, and to *V5* and *V8*. Doing so will reduce the number of filter rules by 6 as illustrated in Fig. 3b.

The process of constructing G_l and finding the maximum weighted matching on it continues for larger prefixes. It stops when G_l does not have any edge. For our example, after the above iteration of /24, we are left with three /23 prefixes, which are aggregated prefixes of the corresponding pairs of VLANs as specified by the maximum weighted matching in the previous iteration of /24. So the new graph G_{23} can be constructed as shown in Fig. 3c. G_{23} has three vertices, which are the three VLAN pairs each receiving aggregatable prefixes in the previous iteration. There is an edge between two vertices, if all four involved VLANs appear together and receive the same treatment in at least one filter. The weight of an edge is the number of filters in which all four involved VLANs appear together and receive the same treatment. For the running example, either edge could be the maximum weighted matching for G_{23}, and our heuristic will randomly pick one, say the edge ({*V9*,*V10*}, {*V6*,*V7*}) as shown in Fig. 3d. This means that aggregatable /23 prefixes will be assigned to the two pairs of VLANs {*V9*,*V10*} and {*V6*,*V7*}, so that they can be further aggregated to a /22 prefix. As a result, whenever a filter needs to permit all those four VLANs, it can simply permit the aggregated /22 prefix in a single rule. The process stops after /23 for the example network.

Based on these results, for our running example it is best to assign 10.0.1/24 to *V9*, 10.0.2/24 to *V10*, 10.0.3/24 to *V6*, 10.0.4/24 to *V7*, 10.0.5/24 to *V5*, and 10.0.6/24 to *V8*. This prefix allocation scheme reduces the number of filter rules by 7, which is the maximum reduction that can be achieved through prefix aggregation. The final design is shown in Fig. 3d.

6 Evaluation

We evaluate our integrated design framework for reachability control on a large university campus network. The network has a few thousand hosts and is assigned a /16 IP space. Our dataset includes configuration files of all devices (most are layer-2 and layer-3 switches), as well as the complete physical topology data obtained through the Cisco Discovery Protocol (CDP).

VLANs are extensively used in this network, with a total number of 69. Each VLAN is assigned a /24 prefix address, so the maximum number of hosts a VLAN can contain is 254. A large number of packet filters (i.e., access-control-list, or ACLs) is present. The vast majority of filters are installed on the gateway routers of VLANs, and there is no filter in the network core. This matches our assumption of filter placement (Section 3.2) very well.

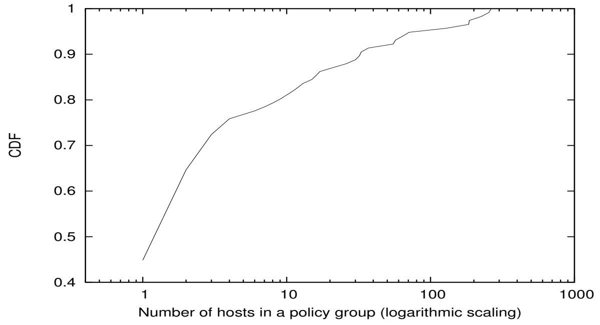


Figure 4: CDF on the number of hosts in each policy group.

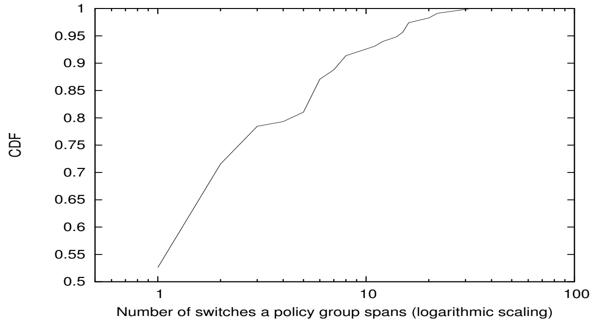


Figure 5: CDF on the number of switches that a policy group spans.

6.1 Characterizing Policy Groups

Although our design framework allows operators to specify reachability policies using the user-role-level reachability matrix (Section 3.3.1), unfortunately for the campus network under study a complete and up-to-date documentation of all reachability policies is not available. Thus, we take an alternative approach and reverse engineer all the policy groups based on the device configuration files, using the methodology presented in [8]. We are able to identify a total of 116 policy groups in this network and, furthermore, make the following interesting observations about them.

First, the majority of policy groups are small, many with fewer than 10 hosts. However, the largest policy group includes 264 hosts which is larger than any single VLAN. 23 policy groups contain 10 hosts or more, and 10 of them contains 50 hosts or more. The size distribution of the policy groups is shown in Fig. 4. Further investigation shows that many of the small policy groups are servers, special purpose (e.g., VoIP) boxes, and management hosts (e.g., operators granting their own office desktops special privilege so that they can log on to remote switches and routers right from their office). The two largest policy groups consist of student dorm hosts and computers in classrooms, respectively. These two groups of hosts are of large volume and span many buildings, but hosts in each group are subject to the same reachability policies.

Second, we investigate the footprint of these policy groups, by measuring the number of switches they span. More specifically, for each policy group we measure the number of switches that one or more of its hosts directly connect to. The results are summarized in Fig. 5. We see that, while half of the policy groups connect to only a single switch, 20% of them span 5 or more switches, and 10% of

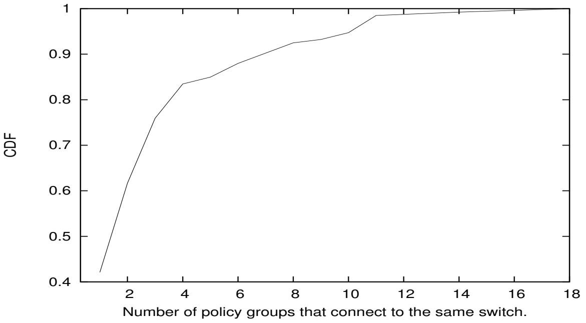


Figure 6: CDF on the number of policy groups that a switch connect to.

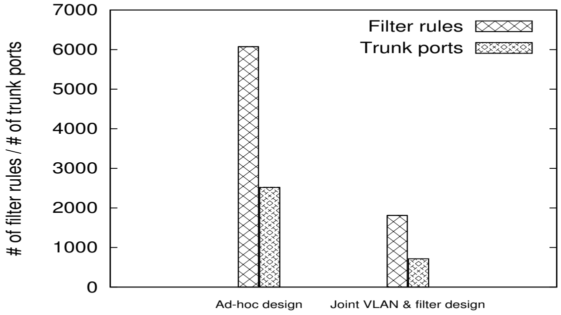


Figure 7: Comparing the number of filter rules and the number of VLAN trunk ports produced by our joint VLAN/filter design heuristic to those by the ad-hoc design approach.

them span 10 or more switches. The largest policy group spans 32 switches, which turns out to be the dorm machines. Next, We measure for every switch the number of policy groups that connect to the switch. The results are summarized in Fig. 6. We see that, while 40% of the switches connect to only a single policy group, 20% of the switches connect to 4 or more policy groups, and 10% of the switches connect to 7 or more policy groups. The maximum number of policy groups that a switch connects to is 18. These results show that the “divide-and-conquer” approach [32, 33] that solely seeks to minimize the number of VLAN trunk ports will not work well for minimizing the overall configuration complexity. This is because that approach will group hosts purely based on their physical locations (i.e., the switches they connect to) and thus will likely place multiple policy groups connecting to the same switch in the same VLAN. This is particularly true when the number of VLANs that can be created is smaller than the number of policy groups, which is the case here. As a result, many filter rules will have to be expressed at the individual IP level, leading to an explosion in the number of rules as illustrated by the example design in Fig. 1b.

6.2 Evaluating the Joint Design of VLANs and Packet Filters

We now evaluate the effectiveness of our heuristic for a joint design of VLANs and packet filters. In the current campus network, packet filters are placed on over 70 layer-3 switches and routers, with more than 6000 rules in total. It is surprising to find that the majority of those rules are at the individual IP level. On the other hand, the current network also has a large number (2500+) of VLAN trunk ports, needed to connect hosts in different physical locations into the same VLAN. Due to these facts, the current network has a high degree of configuration complexity. We do wish to note that the campus network is well managed by a dedicated team of highly skilled operators, and that many hours of design time have been spent on finding a solution to reachability control. We believe that these observations confirm that for a large-scale network with fine-grain reachability control requirements, it is just too difficult for any operators to manually search for the optimal design that minimizes the complexity. This highlights the need for our integrated top-down design approach.

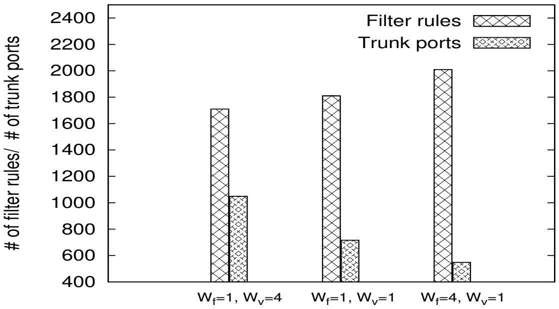


Figure 8: Sensitivity of the results to the W_f and W_v parameters.

To execute our design heuristic, we set N (the maximum number of VLANs allowed in the design) to be 69, the same number of VLANs used in the current network. The *MAX-VLAN-SIZE* is set to 254, which also matches the current network design. We set the weight factors W_f and W_v (for packet filter complexity and VLAN trunk port complexity, respectively) to be 1.0. We then run our heuristic (implemented by a set of Perl scripts) on our dataset. The heuristic runs sufficiently fast and completes the design in less than two minutes on a PC with a quad-core i7 CPU. The resulting configuration complexity is shown in Fig. 7. There are two clusters of bars, corresponding to the ad-hoc design approach operators used to produce the current network and our integrated top-down design approach, respectively. In each cluster, the first bar shows the total number of filter rules in the resulting design, and the second bar shows the total number of VLAN trunk ports. The results show that (i) our heuristic effectively reduces the total number of filter rules down to 1809, which is only 30% of the number of filter rules in the current network; and (ii) our heuristics also reduces the number of VLAN trunk ports down to 716, which is only 29% of the number of trunk ports in the current network.

We next study the sensitivity of the results to the W_f and W_v values. For this purpose, we consider two alternative design scenarios. In the first scenario, the complexity of configuring VLAN trunk ports is considered four times higher than that of configuring filter rules, and thus we set $W_f = 1$ and $W_v = 4$. In the second scenario, the complexity of configuring filter rules is considered four times higher than that of configuring VLAN trunk ports, and thus we set $W_f = 4$ and $W_v = 1$. We run the heuristics with these two additional setups on the same dataset, and Fig. 8 summarizes the results. Each cluster of bars corresponds to a specific choice of W_f and W_v. In each cluster, the first bar shows the total number of filter rules in the resulting design, and the second bar shows the total number of VLAN trunk ports. We make two observations. First, For all settings, the total configuration complexity is substantially lower than that of the current network. This shows that our heuristic effectively reduces the complexity regardless of the choice of W_f or W_v values, and thus can be applied to a wide range of design scenarios. Second, the results also show that our heuristic can intelligently trade off the two complexity factors for different design scenarios, and produce the best design for each scenario. For example, when VLAN trunk ports are given a higher complexity weight, the produced design uses less trunk ports, at the cost of

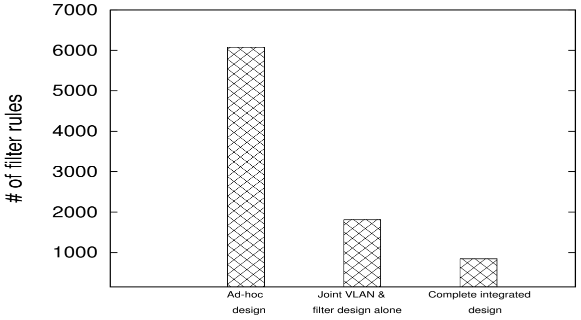


Figure 9: The number of filter rules produced by the current ad-hoc design, the joint design of VLANs and filters alone, and the full integrated design framework including the prefix allocation step.

using more filter rules. In contrast, when packet filters are given a higher complexity weight, the produced design uses fewer filter rules, at the cost of more VLAN trunk ports.

6.3 Evaluating the Joint Design of IP Allocation & Filters

We next evaluate the effectiveness of our heuristic for a joint design of IP address allocation and packet filters. Our heuristic takes as input the packet filters produced in the joint design of VLANs and filters, and those filters are expressed in terms of individual VLANs and hosts. Note that the focus of this paper is on allocating prefix addresses to VLANs, and the heuristic only designs the IP allocation scheme at the prefix level (Section 5.1). Again our heuristic runs relatively fast and completes the design in less than one minute. The result is shown in Fig. 9. This figure shows the number of filter rules in three designs: (i) the current network, (ii) the joint design of VLANs and packet filters alone, with $W_f = W_v = 1$, and (iii) the full integrated design including the IP prefix allocation step. (We do not show the VLAN trunk port data in this figure since they are not impacted by the prefix allocation heuristic.) We see that by integrating the prefix allocation design and the packet filter design, our heuristic is able to further reduce the total number of filter rules down to 841. This halves the total number of filter rules (including both VLAN-level and host-level rules) produced by the joint design of VLANs and packet filters alone, and is only 14% of the number of filter rules in the current network. Together, the total amount of configuration complexity (including both filter rules and VLAN trunk ports) incurred by our integrated top-down design approach is only 18% of that incurred by the current ad-hoc design approach. Overall, these results demonstrate the effectiveness of our integrated design approach in reducing network configuration complexity.

7 Discussion and Open Issues

Applying the integrated approach to other operation objectives: We consider the presented design methodology as briefly outlined in Section 1 an important contribution in its own right. Even though the problem formulations, algorithms and heuristics developed in this paper are specific to the objective of user reachability control, the integrated design methodology is general, neither limited nor tied to that specific context. In fact, we observe that virtually every network operation objective involves designing multiple networking elements; therefore, we expect our integrated methodology to have wide applicability in top-down network design.

Take QoS for instance. A QoS solution typically involves end-to-end traffic engineering (e.g., through routing) and per-link traffic management (e.g., through policing, marking and shaping of packets on routers) [34]. Intuitively, a more sophisticated routing design such as routing traffic of different QoS classes over different paths, which has its own complexity to implement and maintain, can achieve a more predictable and simpler traffic pattern at each router, and subsequently simplify the per-link traffic management requirement. On the other hand, a simplistic routing design that allows all classes of traffic on all links would likely complicate the traffic management task. Thus, when designing a QoS solution, it is important to jointly consider routing and traffic management in order to minimize the overall complexity.

Considering other aspects of network complexity: Our integrated design methodology is not tied to the configuration complexity metric used in this paper. In principle, the approach will work with any complexity metric that is quantifiable based on design parameters. For example, Chun *et al.* [10] have proposed to measure the amount of dependencies between states maintained at different routers. Conceivably, the collection of states required at each networking device can be inferred from the choices of protocols and other such decisions at design time. If that is indeed the case, one may use the state-centric metric in place of the configuration-centric metric in formulating new optimization problems similar to those presented in Sections 4 and 5. An interesting open question is how much the set of optimal design choices would vary from metric to metric.

It is noteworthy that our literature search is unable to identify additional complexity metrics subject to the criteria of being (i) objectively quantifiable and (ii) directly linked to design choices. On a positive note, the networking community is increasingly aware of the importance of developing formal models and metrics for defining and quantifying network complexity. A new group has been formed within the Internet Research Task Force (IRTF) to specifically promote research in this direction. Particularly of interest is the call by this group to develop high-level complexity metrics to help design networks with more predictable behaviors and less resembling of complex nonlinear systems where a small local perturbation may lead to a cascading system wide failure [6].

Applying the integrated approach to optimize SDN flow rule generation: Recent research [19, 24, 26] on SDN advocates that the controller platform should provide a "one big switch" abstraction to the applications running on it. This abstraction enables the application programmers to specify policies at a high level (i.e., network level) and let the controller translate those policies into low-level (i.e., switch-level) flow rules and install them on individual switches. In doing so, a fundamental constraint is the limited TCAM space on the commodity switches where the rules will be stored. Thus, it is desirable to minimize the number of flow rules required. Existing proposals on this front again take a "divide-and-conquer" approach and assume that the IP allocation scheme has been decided before generating and distributing the rules, even though how the IP addresses are assigned can have a significant impact on how flow rules may be aggregated. We believe that our integrated methodology can be applied to jointly design IP allocation and rule generation and distribution to minimize the resulting number of rules. The heuristics presented in this paper may be leveraged in that context as well. We leave a thorough investigation in this direction to future work.

Optimality vs. tractability: We consider the formulations and algorithms presented in this paper only one candidate solution of a spectrum of possible integrated design frameworks for reachability control. For example, it may be feasible to formulate VLAN design, IP address allocation and routing design into a single optimization problem. Broadly speaking, we observe that two competing factors, *optimality* (in terms of how many networking elements are unified) and *tractability* (whether a practical solution can be found), are at play with the integrated approach. An interesting open question is whether a class of design points ("sweet spots") exists that strikes the right balance between the two factors.

8 Conclusions and Future Work

We have shown the importance and effectiveness of an integrated top-down network design methodology for systematically identifying, among all designs that meet a given operational objective, the one(s) with the minimum configuration complexity. The approach enables us to rigorously formulate two new optimization problems as part of a design framework for accomplishing a network's reachability control policy while avoiding unnecessary configuration complexity. The power of the new formulations comes from a unified model that captures the intricate interplays between design decisions concerning VLANs, IP addresses, and packet filters. While this paper focused on reachability control as an application, we believe that the integrated design methodology is applicable not only to a variety of design objectives and tasks for today's networks, but also to the emerging SDN paradigm.

To the best of our knowledge, this is the first work that investigates systematically reducing network complexity through top-down design. Our work builds on top of, but goes fundamentally beyond, prior research on network complexity [7, 10, 31], which focused on complexity measurement, quantification and modeling. Furthermore, this work is the first to reveal a fundamental limitation of the commonly accepted "divide-and-conquer" design approach [32,33] in containing network complexity. We consider this insight a major advance in the state of the art in top-down network design.

For future work, we will seek to (i) extend the unified reachability control design framework by modeling also the task of routing design, (ii) evaluate the framework on additional datasets of enterprise networks, (iii) validate the generality of the integrated design methodology by applying it to other operation objectives such as QoS and resiliency, as well as to the "one big switch" model of SDN, (iv) incorporate other types of network complexity metrics, including those with a higher level semantics about network behaviors than the configuration driven metrics, and last but not the least, (v) investigate how the presented approach, currently targeting new ("green-field") networks, can be adapted to support evolving and redesigning existing ("brown-field") networks.

9 Acknowledgments

We thank the CoNEXT reviewers, our shepherd Jim Kurose, Michael Behringer, Alexander Clemm and Alberto Gonzalez Prieto for their feedback that greatly helped improve this work.

10 References

[1] Cisco IP solution center. http://www.cisco.com/en/US/products/sw/netmgtsw/ps4748/index.html.

[2] Intelliden. http://www.intelliden.com/.

[3] Opsware. http://www.opsware.com/.

[4] Voyence. http://www.voyence.com/.

[5] Maximum Weighted Matching. http://jorisvr.nl/maximummatching.html, 2008.

[6] M. Behringer and G. Huston. A framework for defining network complexity. Internet Draft (work in progress), http://tools.ietf.org/html/draft-irtf-ncrg-complexity-framework-00, 2013.

[7] T. Benson, A. Akella, and D. Maltz. Unraveling the complexity of network management. In *Proc. USENIX NSDI*, 2009.

[8] T. Benson, A. Akella, and D. A. Maltz. Mining policies from enterprise network configuration. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, 2009.

[9] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and Jacobus van der Merwe. Design and implementation of a Routing Control Platform. In *Proc. USENIX NSDI*, 2005.

[10] B.-G. Chun, S. Ratnasamy, and E. Kohler. NetComplex: A Complexity Metric for Networked System Designs. In *Proc. Usenix NSDI*, 2008.

[11] Cisco. Catalyst 2950 desktop switch software configuration guide.

[12] Distributed Management Task Force, Inc. http://www.dmtf.org.

[13] W. Enck, P. McDaniel, S. Sen, P. Sebos, S. Spoerel, A. Greenberg, S. Rao, and W. Aiello. Configuration management at massive scale: System design and experience. In *Proc. USENIX*, 2007.

[14] N. Feamster and H. Balakrishnan. Detecting BGP configuration faults with static analysis. In *Proc. USENIX NSDI*, 2005.

[15] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker. Frenetic: a network programming language. In *Proceedings of ACM SIGPLAN international conference on Functional programming*.

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

[17] T. G. Griffin and J. L. Sobrinho. Metarouting. In *Proc. ACM SIGCOMM*, 2005.

[18] Juniper Networks. What is behind network downtime? http://www-935.ibm.com/services/tw/gts/pdf/200249.pdf, 2008.

[19] N. Kang, Z. Liu, J. Rexford, and D. Walker. Optimizing the one big switch abstraction in software-defined networks. In *Proc. ACM CoNEXT*, 2013.

[20] Z. Kerravala. As the value of enterprise networks escalates, so does the need for configuration management. The Yankee Group Report, 2004.

[21] A. R. Khakpour and A. X. Liu. Quantifying and querying network reachability. In *Proc. IEEE ICDCS*, 2010.

[22] F. Le, G. G. Xie, D. Pei, J. Wang, and H. Zhang. Shedding light on the glue logic of the Internet routing architecture. In *Proc. ACM SIGCOMM*, 2008.

[23] A. Liu, E. Torng, and C. Meiners. Firewall compressor: An algorithm for minimizing firewall policies. In *Proc. IEEE INFOCOM*, 2008.

[24] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker. Composing software-defined networks. In *Proc. USENIX NSDI*, 2013.

[25] S. Narain. Network configuration management via model finding. In *Proc. LISA Conference*, 2005.

[26] Nicira. Networking in the era of virtualization, 2012.

[27] P. Oppenheimer. *Top-Down Network Design (3rd Edition)*. Cisco Press, 2010.

[28] J. Pescatore. Taxonomy of software vulnerabilities. The Gartner Group Report, 2003.

[29] R. Rastogi, Y. Breitbart, M. Garofalakis, and A. Kumar. Optimal configuration of OSPF aggregates. *IEEE/ACM Transaction on Networking*, 2003.

[30] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker. Abstractions for network update. In *Proceedings of the ACM SIGCOMM*, 2012.

[31] X. Sun, S. Rao, and G. Xie. Modeling complexity of enterprise routing design. In *Proc. ACM CoNEXT*, 2012.

[32] X. Sun, Y.-W. E. Sung, S. Krothapalli, and S. Rao. A Systematic Approach for Evolving VLAN Design. In *Proc. IEEE INFOCOM*, 2010.

[33] E. Sung, X. Sun, S. Rao, G. G. Xie, and D. Maltz. Towards systematic design of enterprise networks. *IEEE/ACM Trans. Networking*, 19(3):695–708, June 2011.

[34] V. Tabatabaee, B. Bhattacharjee, R. La, and M. A. Shayman. Differentiated traffic engineering for QoS provisioning. In *IEEE INFOCOM 2005*, 2005.

[35] G. G. Xie, J. Zhan, D. A. Maltz, H. Zhang, A. Greenberg, G. Hjalmtysson, and J. Rexford. On static reachability analysis of IP networks. In *Proc. IEEE INFOCOM*, 2005.