A Systematic Approach for Evolving VLAN Designs

Xin Sun, Yu-Wei E. Sung, Sunil D. Krothapalli, and Sanjay G. Rao
Purdue University

***Abstract*—Enterprise networks are large and complex, and their designs must be frequently altered to adapt to changing organizational needs. The process of redesigning and reconfiguring enterprise networks is ad-hoc and error-prone, and configuration errors could cause serious issues such as network outages. In this paper, we take a step towards systematic evolution of network designs in the context of virtual local area networks (VLANs). We focus on VLANs given their importance and prevalence, the frequent need to change VLAN designs, and the time-consuming and error-prone process of making changes. We present algorithms for common design tasks encountered in evolving VLANs such as deciding which VLAN a new host must be assigned to. Our algorithms trade off multiple criteria such as broadcast traffic costs, and costs associated with maintaining spanning trees for each VLAN in the network, while honoring correctness and feasibility constraints on the design. Our algorithms also enable automatic detection of network-wide dependencies which must be factored when reconfiguring VLANs. We evaluate our algorithms on longitudinal snapshots of configuration files of a large-scale operational campus network obtained over a two year period. Our results show that our algorithms can produce significantly better designs than current practice, while avoiding errors and minimizing human work. Our unique data-sets also enable us to characterize VLAN related change activities in real networks, an important contribution in its own right.**

I. Introduction

Enterprise network operators must frequently change the design of their networks to reflect new organizational needs, that may arise due to the addition of new hosts, movement and reorganization of departments and personnel, revision of security policies, and upgrading of router hardware.

The high-level objectives (such as performance and security goals) that operators have for their networks are embedded in hundreds of low-level device configurations. Redesigning enterprise networks is challenging given the huge semantic gap between these high-level objectives, and low-level configurations. Configuration changes are required on a frequent basis, and a given redesign task may require changes to multiple device configurations. The presence of a large number of dependencies between configurations has been shown to be a key source of complexity in managing networks [1].

The frequency and complexity of configuration changes makes the overall process of redesigning and reconfiguring enterprise networks error-prone. Errors in changing configurations have been known to result in outages, business service disruptions, violations of Service Level Agreements (SLA) and cyber-attacks [2], [3], [4]. Further, ad-hoc practices in evolving networks may result in degradation of network "quality" leading to overly complex and "kludgy" designs with poor performance characteristics (e.g., unacceptably high broadcast traffic) [1], [5], [6], [7].

Despite the complexity, scale and importance of enterprise networks, the area has only recently started receiving attention from the research community. Recent efforts have advanced our understanding of these networks through studies that model operational practices in enterprise networks [1], [5], [6], [7], [8], [9]. There has been some work on systematic "top-down" approaches to enterprise design and configuration, but in the limited context of newly deployed (greenfield) networks [10], [11]. While these are important first efforts, the key challenge of redesigning and reconfiguring existing networks is yet to be considered.

In this paper, we take a first step to this end by considering issues in evolving the VLAN design of operational enterprise networks. We focus on VLANs as a case study since they are extensively used in enterprise networks, represent time-consuming tasks for network operators, and have only recently started receiving attention from the research community [7], [11], [12]. On the one hand, changes must be made frequently to the VLAN design given the constant addition and movement of hosts in the network. On the other hand, making these changes poses issues for operators at both the design and configuration levels. At the design level, operators must ensure that changes to VLAN design take into account network-wide performance criteria such as the need to constrain broadcast costs with VLANs, At the configuration level, operators must be careful to account for network-wide dependencies. In particular, even a simple change such as adding a host to a VLAN may require reconfiguration of multiple switches in the network (a process called configuring "trunk" links).

Motivated by these challenges, we design a set of algorithms and heuristics, that can automate the VLAN management tasks, at both the design and configuration levels. Our algorithms take the operator's high-level intent, (e.g., "add a host corresponding to a user in the sales department"), and make design decisions such as whether a new VLAN should be created, and which VLAN the new host must be assigned to. The algorithms also identify the set of devices whose configurations need to be modified, and the changes that need to be made. The results of the algorithms could then be integrated with existing tools such as [10] to create the final and new device level configurations.

We evaluate our algorithms on a large-scale campus network that contains tens of thousands of hosts, using a longitudinal data-set that includes daily snapshots of complete configura-

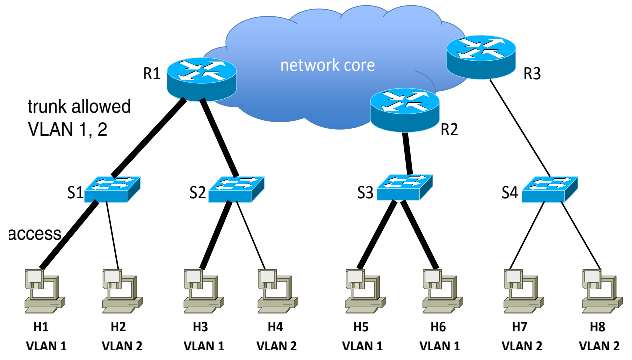


Fig. 1: A sample enterprise VLAN setup

tion files of all devices in the network, for a continuous period of two and a half years. As a first step, we taxonomize the VLAN related change activities that occur in the network, and characterize the frequency of the change activities, and discuss the types of events that result in these changes. We believe such a characterization of changes to operational networks is an important contribution of the paper in its own right. Second, employing the data-set, we evaluate the effectiveness of our algorithms in making judicious decisions while evolving VLAN designs, by comparing them with decisions made by the network operators. Our results show that our algorithms can result in substantially better performance, both in terms of lower broadcast costs for VLANs, as well as use of fewer VLANs thereby minimizing processing requirements on switches. Finally, we evaluate the benefits of our automated algorithms for reconfiguring VLANs by studying the presence of configuration errors in existing VLAN designs. Our analysis reveals several errors in configuration of trunk links which can both impact host connectivity and lead to links carrying unnecessary broadcast traffic. Such errors could be avoided using our automated algorithms. Overall, these results demonstrate the importance and effectiveness of adopting systematic approaches in managing the evolution of VLAN designs.

II. Background

In this paper we focus on issues in evolving the VLAN design of enterprise networks. We choose VLANs as a case study, given that they are extensively used today, given that changes must be made frequently to VLAN design, and since the management of VLANs is a time-consuming and error-prone process [7], [11]. In the rest of this section, we give a brief description of the technical challenges facing VLAN management.

A. VLAN essentials

Operators reduce the complexity of their configuration tasks by thinking about users as collective groups based on the role of each user in the organization (e.g., what resources they should be able to access). Example groups are engineering, sales, payroll, student cluster, faculty cluster, etc. In this paper, we name such a user group a *category*. Today, these logical groupings are most commonly implemented by VLANs, which take a set of users in physically disparate locations and place them into a single logical subnet, even if the users are connected to different switches. For instance, an enterprise policy may permit access only for all sales personnel, and it may be desirable to ensure these users receive IP addresses from the same subnet so that routing policies and packet filters can be applied to them as a group. Consider Fig. 1. S1∼S4 are switches, and R1∼R3 are routers. Notice that even though hosts H1 and H6 are physically separated, they are both part of VLAN 1. Likewise, hosts H2 and H8 belong to VLAN 2.

Each VLAN constitutes a separate broadcast domain. Therefore, it is important to ensure that broadcast traffic is properly constrained to reduce unnecessary traffic for increased performance and security. To achieve this, every link is configured to permit only traffic for appropriate VLANs. In Fig. 1, the link S1-H1 is configured as an *access* link and forwards only VLAN 1 traffic. The link S1-R1 is configured as a *trunk* link and permits traffic for multiple explicitly specified VLANs (in this case, VLANs 1 and 2). Typically, a separate spanning tree rooted at a *root bridge* is constructed per VLAN. For example, the collection of bold links forms the spanning tree of VLAN 1, and either R1 or R2 can be selected as the root bridge.

B. Considerations in VLAN design

In designing VLANs, there are several criteria that operators must consider as we discuss below:

Correctness criterion: Hosts in different categories must be placed in different VLANs. This is the correctness criterion for VLAN design. A category may be implemented as one or more VLANs.

Feasibility criterion: The maximum number of hosts a VLAN can have is determined by the size of the IP block it is assigned. For instance, a VLAN with a /24 IP block can have no more than 256 hosts.

Performance and cost criteria: Hosts in the same VLAN belong to the same broadcast domain, and it is important to keep the cost of broadcast traffic small. The cost depends on both (i) the number of hosts in the VLAN, and (ii) the span of the VLAN, i.e., how spread out the hosts of the VLAN are in the underlying network topology. This can be measured by the number of links in the spanning tree of the VLAN. More formally, we leverage the broadcast traffic cost model from [11], which is defined as follows. Let H denote the number of hosts in a VLAN, A denote the average broadcast traffic (in pkt/s) generated by a host in the VLAN, and W denote the number of links in the spanning tree of the VLAN. Then, the broadcast cost B of the VLAN is $B = H * A * W$.

Clearly, partitioning a category into more VLANs means smaller broadcast traffic of each VLAN, since it reduces the number of hosts in the VLANs. This will in turn lead to better network performance, since it reduces network congestion. However, the total number of VLANs a network can have is bounded by the hardware capacity. This is because a separate spanning tree is typically constructed and maintained for every VLAN in the network, and this increases the memory and processing requirements of individual switches. For example, a

Cisco Catalyst 2950 switch can only support up to 64 spanning tree instances [13]. Increasing the bound on number of VLANs usually requires upgrading the hardware, which translates to monetary costs. Thus, it is critical for the operator to make the tradeoff that best suits her budget, and her requirement on the network performance.

Other considerations: After the operator decides on the number of VLANs that should be assigned to a category, she must then decide which hosts are grouped into each VLAN. Clearly, different ways of grouping hosts may result in different spanning tree sizes, and thus different broadcast traffic costs. Finally, the placement of the root-bridge also impacts the spanning tree size. Hence, it is important to group the hosts and place the root-bridge judiciously to minimize the broadcast traffic.

III. Evolving VLAN designs

We begin by discussing common types of tasks that operators encounter when evolving VLAN designs. We next discuss the complexity that these tasks entail.

A. *Tasks encountered when evolving VLANs*

Through discussions with operators, and our characterization of change activities related to VLANs (see Section V-A), we identified a few common change operations related to VLAN design as we summarize below:

- *Adding/removing a host to/from a category:* These change operations are frequently encountered. Example scenarios include people joining and leaving a department, purchase of new machines and removal of outdated ones. While these changes might seem trivial, they are not straightforward to make as we discuss later in the section.
- *Adding/removing a category.* These change operations are more involved, and happen relatively less frequently. New host categories may be added when an enterprise grows and new departments are created. Likewise, categories get removed on occasions such as a merger of departments, or closure of facilities. When a category is added, a set of hosts belonging to that category may be added altogether, and one or more VLANs may need to be created. When a category is removed, all the VLANs and hosts of that category will need to be removed.
- *Configuring/reconfiguring a root-bridge* These changes usually happen when VLANs are created/removed. Our analysis of change activities in operational networks (Section V-A) also reveals interesting scenarios where root-bridges were changed because an entire department migrates to a new building. Hosts in the department were part of a group of VLANs, and new root-bridges were selected for these VLANs which were closer to the new building.

B. *Complexity in making changes to VLANs*

The process of redesigning and reconfiguring VLANs is complex and error-prone as we discuss below.

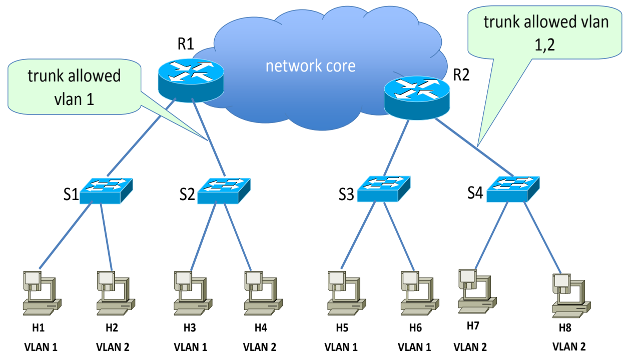


Fig. 2: Examples of missing and redundant links that result from VLAN misconfigurations.

1) *Choosing design strategies:* Many of the tasks described in the previous section require operators to make non-trivial decisions taking various factors into account. When adding a category, an operator must decide the number of VLANs to create, come up with an assignment of newly added hosts to each of the newly created VLANs, as as well as select a root-bridge for each VLAN. When adding a host to a category, an operator must decide whether to create a new VLAN or to add the host to an existing VLAN, and in the latter case, the operator will also need to decide which VLAN to add the host to. Finally, selecting a root-bridge requires the operator to make a decision of which switch to choose. In choosing the strategies, the operator must meet the correctness and feasibility criteria while optimizing for the performance criteria and take costs into account (Section II-B). Today, operators choose strategies in an ad-hoc fashion, which often results in far from optimal designs. For example, our analysis of the data-set we collect (Section V-A) shows that the root-bridge placement of over 30% of the VLANs in the campus network are not optimal.

2) *Configuring and reconfiguring the trunk links:* In addition to choosing design strategies, the operator must manually configure or reconfigure certain trunk links, for any of the common change operations. This in turn requires the operator to correctly identify network-wide configuration dependencies. Failure to do so could result in lose of connectivity, and unnecessary broadcast traffic. Figure 2 shows two examples of misconfiguration. First, host H4 which belongs to VLAN 2 was recently added, but the trunk link R1-S2 was not configured to allow VLAN 2. This caused H4 to be disconnected from the rest of the hosts in the network. We term such a misconfigured link a *missing link* of VLAN 2, since the link should have been configured to allow VLAN 2. Second, the trunk link R2-S4 was configured to allow both VLAN 1 and 2, however there was no host from VLAN 1 attached to S4. This caused all broadcast traffic generated by VLAN 1 to unnecessarily transverse this link. We term such a misconfigured link a *redundant link* of VLAN 1, since it should have not been configured for VLAN 1.

While there are automated tools available today to prevent such misconfigurations, they are inadequate. For instance, Cisco VLAN Trunk Protocol (VTP) is a Cisco proprietary protocol that seeks to eliminate the need of configuring trunk links. At a high level, VTP allows an operator to create VTP domains, and each switch can be configured to belong on one (and only one) VTP domain. When a VLAN is added to one switch in a VTP domain, all the other switches in the same domain will be automatically configured to trunk the VLAN [14]. VTP can completely eliminate the need for manual trunk link configuration only when all switches are configured to be in the same VTP domain. However doing so will require every switch to be part of the spanning tree of each VLAN. This may impose prohibitive CPU overhead on the switches, and may also introduce too much VTP protocol traffic [15]. If the network consists of multiple VTP domains, then the VLANs that span switches in multiple domains still require manual trunk link configuration. Our prior work shows that this scenario is fairly common [7].

IV. Algorithms for automating VLAN change configuration

In this section we present a set of algorithms that aim to automate the process of evolving VLAN designs. The first two algorithms automate the changes that require choosing design strategies, namely adding a new category and adding a host. The last algorithm automates the process of configuring trunk links.

A. Algorithms for adding a new category

Given a new category with a set of hosts, we present an algorithm that determines the number of new VLANs to create, and how the hosts should be grouped into the VLANs. In doing so, the algorithm meets the correctness and feasibility criteria (see section II-B). Further, the algorithm seeks to make the right tradeoff between broadcast traffic cost and the number of VLANs used, in a manner that reflects the operator's intention.

In order to let the operator express her intention of how the tradeoff should be made, we let her specify two parameters to the algorithm. One parameter is the maximum broadcast traffic she is willing to tolerate, which we denote as $\boldsymbol{B_{max}}$. Our algorithm should never create VLANs whose broadcast traffic is greater than B_{max}. The other parameter is the *total* number of VLANs that is desirable not to be exceeded, which we denote as $\boldsymbol{N}$. While this number is not a hard bound, the algorithm should avoid creating more than N VLANs whenever possible.

When adding a new category, our algorithm determines the number of new VLANs to create using the following logic. First, it creates a single VLAN, and puts all the hosts of the new category in it. If the resulting broadcast traffic cost is greater than B_{max}, the algorithm will always partition this VLAN into smaller VLANs. Otherwise, if the cost is smaller than αB_{max}, where α is a constant between 0 and 1, then the algorithm will never partition the VLAN any further. This is reasonable, because when the broadcast traffic is small enough, there is no need to further reduce it as it will make no practical difference to the network performance.

Now the question is whether the algorithm should partition the VLAN further if its broadcast traffic falls between αB_{max} and B_{max}. To make a reasonable choice, the algorithm must consider the number of new VLANs that can be created. Intuitively, if the network is in a stage where a lot of new VLANs can be created, then the algorithm should partition the VLAN in order to further reduce the broadcast traffic. On the other hand, if the network is in a stage where few new VLANs may be created, then the algorithm should not partition the new VLAN in order to save on the number of VLANs. Thus our algorithm compares the broadcast traffic cost of the new VLAN, with $(\alpha+\frac{(1-\alpha)P}{N})B_{max}$, and partitions the new VLAN only when its cost is higher. Here P refers to the number of *existing* VLANs in the network. We believe $(\alpha+\frac{(1-\alpha)P}{N})B_{max}$ is a reasonable threshold to compare with. For example, when $P = 0$, which means no VLAN has been created yet and up to N new VLANs can be created, the threshold reduces to αB_{max}. On the other hand, when $P = N$, which means creating new VLANs should be avoided, it becomes B_{max}.

We summarize the algorithm below, where B refers to the broadcast traffic cost of the new VLAN.

```
1: if $B < \alpha B_{max}$ then
2:     do not partition
3: else if $\alpha B_{max} \leq B < B_{max}$ && $B <= (\alpha + \frac{(1-\alpha)P}{N})B_{max}$ then
4:     do not partition
5: else
6:     partition
7: end if
```

More specifically, the algorithm has the following steps when trying to add a new category:

Step 1: The algorithm first considers the entire category as a single VLAN V and selects the optimum root-bridge. The optimal root-bridge can be selected simply by considering every switch in the network and choosing the one that results in minimum broadcast traffic. If V meets the feasibility criterion, and the broadcast traffic cost of V is smaller than min$\{B_{max}$, $(\alpha + \frac{(1-\alpha)P}{N})B_{max}\}$, then the algorithm returns V and stops. Otherwise, it stores V in a set L.

Step 2: The algorithm picks and deletes a VLAN V with the largest broadcast traffic from L, and runs the *partition algorithm* on V, which results in two smaller VLANs V_1 and V_2. (We will present this partition algorithm shortly).

Step 3: If V_1(or V_2) meets the feasibility criterion, and the broadcast traffic of V_1(or V_2) is greater than min$\{B_{max}$,$(\alpha + \frac{(1-\alpha)P}{N})B_{max}\}$, then V_1 (or V_2) is added to set L. Otherwise, it is added to set S. Note that P (the number of existing VLANs), is increased by one to reflect the addition of the new VLAN. Step 2 and Step 3 are repeated until L is empty.

Step 4: The algorithm randomly picks two VLANs from set S and merges them into one VLAN. If the resulting VLAN meets the feasibility criterion, and the broadcast traffic cost (with the best root-bridge selected) is smaller than min$\{B_{max}$,

$(\alpha + \frac{(1-\alpha)P}{N})B_{max}\}$, then the new VLAN is added to S, and the two original VLANs are deleted. Otherwise, the algorithm picks another pair of VLANs in S and repeats the step. The algorithm continues until all possible pairs are tried, and no further pair can be merged. This step is needed to prevent creation of overly small VLANs by the grouping algorithm used in the previous step.

Step 5: The algorithm returns S, which gives a grouping scheme of the new category.

We now describe in detail the partition algorithm that was used in Step 2. This algorithm partitions the hosts in a VLAN into *two* separate VLANs. Intuitively, it is best to group the hosts that are "close" to each other in the underlying topology into the same VLAN, as this will minimize the span of the resulting VLANs. To achieve this, we employ the *K-means* clustering algorithm to cluster nearby hosts together. In particular, suppose the target VLAN has H hosts, then each host is considered as a point in a H-dimensional space with a coordinate computed to be a vector of the length of the shortest layer-2 path to all of the hosts (including the host itself) in the VLAN. The K-means clustering algorithm is then invoked with K=2 to partition the VLAN into two smaller VLANs.

Finally, an operator may require that all hosts in a certain category be grouped into a single VLAN. For example, hosts in the category may need to run a broadcast-based application, and may need to broadcast to all other hosts in the same category. The above algorithm can be trivially augmented to accommodate such special categories, by adding additional conditions in Step 1.

B. Algorithms for adding a host to a category

Given a new host that is to be added to an existing category, we next discuss an algorithm which determines whether a new VLAN should be created to accommodate the new host, or whether the new host should be added to an existing VLAN in the same category. In the latter case, the algorithm also decides which VLAN to add the host to, if the category has multiple VLANs.

We assume the category the new host is in has K VLANs. We again let the operator specify the total number of VLANs N and the maximum acceptable broadcast traffic cost B_{max} as in the previous algorithm. The algorithm proceeds as follows.

For each VLAN in the category, the algorithm calculates the broadcast traffic of the resulting VLAN if the new host were to be added to it. If the broadcast traffic of all these VLANs become larger than $\min\{B_{max}, (\alpha + \frac{(1-\alpha)P}{N})B_{max}\}$, then the algorithm creates a new VLAN and adds the host to it, and stops. Here P denotes the number of VLANs currently used in the network, like in the previous algorithm. Otherwise, the algorithm adds the host to the VLAN for which the increase in broadcast traffic is the smallest, and stops.

C. Algorithms for configuring trunk links

As we discussed in Section III, changing the VLAN design may require reconfiguring trunk links. When creating a new VLAN, adding a host to a VLAN or configuring the root-bridge, certain trunk links will need to be configured to allow the VLAN. Also, when removing a host or root-bridge, certain trunk links will need to be reconfigured to eliminate unnecessary broadcast traffic. In doing so, a key task is to discover the network-wide dependencies. For instance, when removing a host from a VLAN, the algorithm must first identify all the trunk links on the shortest path between the host and the root-bridge of the VLAN. This in turn depends on the location of the access switch of the host, and the location of the root-bridge. It then needs to determine for each link whether the link is being used by other hosts, and only reconfigure those that are not. This depends on the location of all other hosts in the same VLAN. We next present an algorithm for discovering such dependencies when removing a host from a VLAN. Similar algorithms may be employed when adding a host or changing the root-bridge, and we omit the algorithms for space reasons.

Step 1: The algorithm finds the shortest layer-2 path between the access switch of the host that is to be removed, and the root-bridge of the VLAN. This can be done using classical algorithms such as the Floyd-Warshall algorithm. We denote the set of trunk links on this path as Q.

Step 2: For each remaining host in the same VLAN, the algorithm finds the shortest path from the access switch of the host to the root-bridge. We denote the set of trunk links on all these paths as T.

Step 3: The trunk links which are present in set Q but not present in set T are identified. The algorithm reconfigures these trunk link to no longer allow the VLAN.

V. Evaluation

We evaluate our algorithms on a large-scale campus network. The network consists of about 200 routers, 1300 switches, and tens of thousands of hosts grouped into hundreds of VLANs. A small number of routers form the core of the network. Typically, each building has a router with a link to one of the core routers. This link connects all hosts in the building to the rest of the network. The network has seen significant growth for the past few years. Our data-set includes daily snapshots of the configuration files of all switches and routers in the network, for two and a half years (from January 2007 to July 2009). The data-set also includes the physical topology of the network.

VLAN usage: While the campus IT operators provide routing services for the entire campus, each logical group such as the School of Engineering, the School of Liberal Arts, and the Libraries has its own administrators. Each administrative unit is given an IP address block and is free to assign addresses within that block to individual hosts. The operator policy requires that hosts in different administrative units must belong to different broadcast domains. VLANs are extensively used to meet this goal, as well as to constrain the size of broadcast domains. Most VLANs span a small section of the campus — about 50% of them span only one building. However, about 10% of the VLANs span 5+ buildings, and the largest VLAN spans over 60 buildings. VLANs with a large span correspond to administrative units that have hosts in most buildings on campus, e.g., hosts in all classrooms are administered together and are grouped into one VLAN.

A. Characterizing typical changes to VLAN design

Table I summarizes the main types of VLAN related changes and their frequency. The first column presents the type of change operation. The second column gives the total number of occurrences of each type of change operation during the entire 30-month period for which we have configuration data available. The remaining columns show a breakdown of the number of occurrences of each type of change operation over various six-month periods. The breakdown enables us to get a better sense of differences in change activities over time.

We have already described the first five types of change operations in Section III. We now briefly discuss the last two change operations. "Moving a host of a different VLAN to the access switch" refers to a scenario where a user moved out of her office and another user from a different department moved in. The new user's computer is part of a different VLAN, but it is now plugged into the same access switch and port as the previous user's computer was. This may be viewed as a combination of removing a host, and adding a new host. The access switch of a host may change (last row of Table I) if a user moves to a new office, or if the access switch itself has been upgraded. This operation may also be viewed as a combination of removing a host and adding a new host.

We observe from Table I that as expected, operations involving adding or removing entire categories are relatively infrequent. Operations involving changing the root-bridge of VLANs are also infrequent. In fact most changes of this type occurred during an episode when an entire department migrated to a new building. Hosts in the department were part of a group of VLANs, and new root-bridges were selected for these VLANs which were closer to the new building. Finally, other operations — particularly addition and removal of hosts — occur very frequently, with over 8 hosts added per day on average.

The table also shows that the frequency of change operations may be very different across periods. For instance, 60 categories were added from 01/2007 to 06/2007, however no new categories were added from 01/2008 to 06/2008. Finally, some change operations show correlations. For example, when a lot of categories were added, a lot of new hosts were added too.

B. Effectiveness of algorithms in meeting performance criteria

We evaluate how effective our algorithms are compared to the current approach in ensuring that the resulting VLAN designs perform well with respect to criteria such as constraining the broadcast costs, and minimizing the number of VLANs used. Our algorithms require several inputs such as a logical mapping from hosts to categories, the broadcast traffic generated by each host, and the B_{max}, N and α parameters. We present our methodology for obtaining these values, and then present our evaluation results.

Evaluation methodology: With help from the operators, we categorize the hosts on a large segment of the campus. Each category corresponds to a different administrative unit. In total, there are around 120 categories and 15000 hosts in this segment. Many categories are small, and the median category has only around 80 hosts. However, the largest category includes 2000+ hosts.

To study the sensitivity of our algorithms to different change patterns, we split our 30-month trace into multiple six-month segments. We only present results using the first six-month segment, given it has the most dynamic pattern, and contains complex change activities such as creation of categories (see Table I). We have run our algorithms over other trace segments as well, and similar trends hold. We do not present those results for lack of space.

We assume that broadcast traffic is generated at 2.12 packet/second/source, based on a measurement of an actual campus network [11]. To study the effectiveness of our algorithms compared to the current approach, we set N to be the same number of VLANs as what was used by the current approach at the end of the six-month trace. We set B_{max} to be 110% of the maximum broadcast traffic of all VLANs at the beginning of the trace, to allow some growth in VLAN size. Finally, in our evaluation we fix α at 0.5. While we believe these parameters are reasonable, we have also performed a sensitivity study to them.

Results: Figure 3 shows the maximum and median broadcast traffic costs across all VLANs, produced by our algorithms and by the current approach. The top two curves correspond to the maximum values, while the bottom two correspond to the medians. We observe that our algorithms reduce the maximum broadcast traffic by a factor of three. In addition, while our algorithms produce no VLANs with broadcast traffic greater than 120,000 pkt/s, the current approach produced eight such over-sized VLANs (not shown in the figure). Further analysis shows that the better performance of our approach is due to several factors. (i) Our algorithms create more VLANs for large categories. This reduces the broadcast traffic of the VLANs in those categories. For example, there was a big category added to the network on January 5th, with 254 hosts. The current approach created only a single VLAN for the category, resulting in a broadcast traffic cost of 205,699 pkt/s. In contrast, our algorithms create two VLANs, with the broadcast traffic of both VLANs smaller than 77,480 pkt/s (a factor of three reduction). (ii) Our algorithms assign hosts to VLANs in a more balanced way, thanks to the smart grouping algorithm (Section IV-A). For example, on January 12th, a large category with 392 hosts was added to the network. While our algorithms create the same number of VLANs (two) for this new category as the current approach did, the variation in VLAN sizes is lower. In particular, the broadcast traffic of the two VLANs created by our algorithm is 85,958 pkt/s and 89,409 pkt/s, while the broadcast traffic of the two VLANs created using the operator-based approach is 134,416 pkt/s and

Type of Change	Total	01/07-06/07	07/07-12/07	01/08-06/08	07/08-12/08	01/09-06/09
Add a new category	74	60	3	0	1	10
Remove a category	13	0	0	11	1	1
Add a new host	7619	4731	902	339	484	1163
Remove a host	4682	373	443	1016	932	1918
Change the root-bridge of a VLAN	30	1	1	0	1	27
Move host of different VLAN to access switch	630	125	167	55	147	136
Change the access switch of a host	145	16	44	59	15	11

TABLE I: Types of change operations and their frequency over a 30 month period

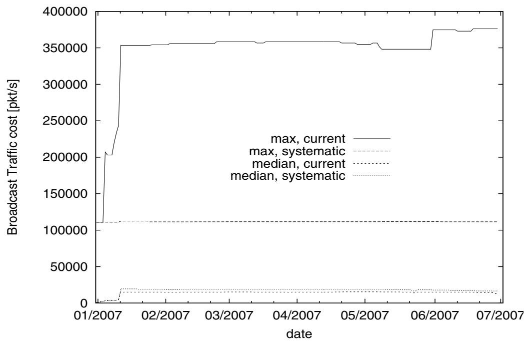


Fig. 3: The maximum and median broadcast traffic costs, produced by the current approach and by our algorithms.

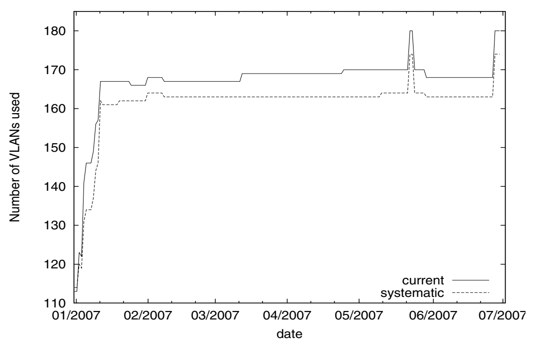


Fig. 4: The number of VLANs used in the network for the current approach and for our algorithms

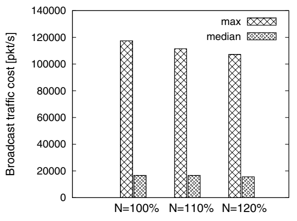
52,470 pkt/s. (iii) Our algorithms always choose the optimal root-bridge for the VLANs, while the current approach failed to do so around 30% of the time.

Figure 3 also shows that the median broadcast traffic produced by our algorithms is comparable to that produced by the current approach. This, combined with the fact that the maximum broadcast traffic is greatly reduced, confirms that our algorithms meet operator objectives better while evolving VLAN designs.

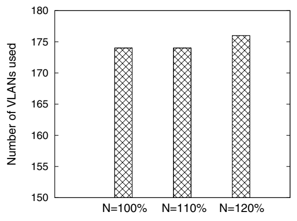
Figure 4 shows the number of VLANs used by our algorithms, and by the current approach. Interestingly, our algorithms actually use six fewer VLANs than the current approach. Further analysis reveals that the current approach sometimes created new VLANs unnecessarily, or created more VLANs than needed. For example, on January 6th, a new category with 136 hosts was added to the network. The current approach created two new VLANs for the category, with the broadcast traffic of each VLAN around 10,200 pkt/s. In contrast, our algorithms create only a single VLAN, with the broadcast traffic of 36,328 pkt/s.

We discussed the above results with the campus operators. They confirmed that the designs produced by our algorithms are reasonable, and expressed interest in adopting our systematic approach. Overall, these results highlight the effectiveness of our algorithms in choosing more judicious strategies when evolving VLAN designs.

Sensitivity to thresholds: We next study the sensitivity of our algorithms to the N and B_{max} parameters. We perform two sets of experiments. In the first set, we fix B_{max} to be 110% of the maximum broadcast traffic of all VLANs at the beginning of the six-month trace, and vary N to be



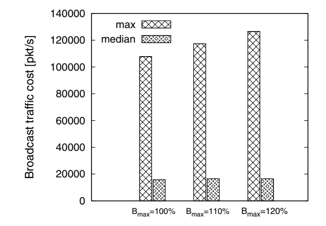
(a) The maximum and median broadcast traffic costs



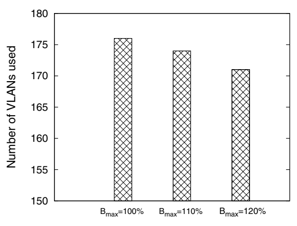
(b) Number of VLANs used

Fig. 5: Sensitivity of our algorithms to the N parameter.

100%, 110% and 120% of the number of VLANs used by the current approach at the end of the trace. Figure 5a shows the maximum and median broadcast traffic of all VLANs. Figure 5b shows the number of VLANs used at the end of the trace. Each bar in the graphs corresponds to a different N value. In the second set, we fix N to be the same number of VLANs as what was produced by the current approach at the end of the trace, and vary B_{max} to be 100%, 110% and 120% of the maximum broadcast traffic of all VLANs at the beginning of the trace. Figure 6a and Figure 6b show the maximum and median broadcast traffic, and the number of VLANs used. Each bar corresponds to a different B_{max} value. All the results shown in Figure 5 and Figure 6 are produced by our algorithms at the end of the six-month trace. These results demonstrate that our algorithms make smart tradeoffs that meet the operator objectives well. For example, with larger N values, the algorithms create more VLANs and reduce the broadcast traffic costs, as shown in Figure 5. With larger B_{max} values, the algorithms will allow more broadcast traffic in each



(a) The maximum and median broadcast traffic costs



(b) Number of VLANs used

Fig. 6: Sensitivity of our algorithms to the B_{max} parameter.

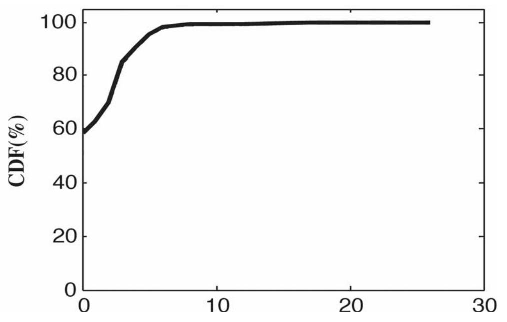


Fig. 7: CDF of the number of redundant links per VLAN.

VLAN, and create fewer VLANs, as shown in Figure 6.

C. Benefits of algorithms in avoiding configuration errors

In this section, we evaluate the benefits of our algorithms in avoiding configuration errors. We do so by evaluating the number of misconfigured trunk links generated with the current approach. Our evaluations are conducted using a one-day snapshot from our data-set. We consider two kinds of trunk link misconfigurations defined in Sec III-B2: (i) redundant links; and (ii) missing links.

Figure 7 shows the CDF on the number of redundant links per VLAN. Around 42% of all the VLANs have one or more redundant links, and 5% of them have more than 5 such links. These numbers signify the presence of significant amount of unnecessary broadcast traffic in the network. We discussed our findings with our operators. It turned out that most redundantly configured trunk links are due to network evolution. For example, when a host corresponding to a VLAN is removed, operators often do not update switch configurations to reflect host removal (as doing so in a correct fashion is viewed non-trivial today).

In addition, our analysis also discovered a few cases of missing links. The number of instances of such misconfigurations are relatively rare since errors of this form disconnect users from the network, causing operators to fix the errors in response to complaints from users. However, our approach can proactively determine the presence of such errors and help operators avoid them.

Overall, these results highlight the need for systematic algorithms to correctly and automatically identify network dependencies, and configure and reconfigure trunk links.

VI. Related work

Issues in VLAN design have not received much attention until very recently. In recent years, researchers have studied VLAN usage in enterprise networks using a single configuration snapshot [7] or traffic data [12], [16] to expose degenerate design patterns, understand VLAN traffic patterns, and correlate cross-layer faults. The closest related work to this paper is our own work [11] which shows the feasibility of applying systematic "top-down" approaches to VLAN design, but in the limited context of newly deployed (greenfield) networks [11]. In contrast, our focus in this paper is on algorithms for incrementally evolving existing VLAN designs, characterizing VLAN design changes in operational networks, algorithms for identifying network-wide configuration dependencies, and evaluation of our algorithms under real change patterns.

To our knowledge, the only other works that study longitudinal snapshots of network configuration are [17], [18]. However, their goal is to summarize and study configuration-level changes at a network-wide or interface-centric view. In contrast, our goal is orthogonal in that we characterize VLAN design-level changes and the causes behind them. A parallel work [19] looks at automatically reconfiguring networks to reduce complexity of network designs. Our work is orthogonal in that we focus on performance criteria such as broadcast traffic, and cost criteria such as number of VLANs used, rather than complexity of network design.

Our work complements recent proposals for alternate architectures for enterprise networks [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30]. These works seek to contain the complexity of enterprise design and configuration either through centralized decision making [20], [21], [22], [23], [24], [25], [26], or through alternatives to broadcast-based host discovery [27], [28]. While these architecture can effectively minimize broadcast traffic if deployed, our work is immediately relevant to existing networks and an initial prototype [31] has been deployed in a large campus network.

VII. Conclusions

In this paper, our primary contribution is to show the feasibility and importance of a systematic approach to evolving VLAN designs in operational enterprise networks, considering issues at both design level, and configuration level. When evaluated over a longitudinal data set of a large-scale operational campus network which included daily snapshots of network configurations for over two years, we found our approach can (i) result in VLANs with broadcast costs lower than current practice by 70%; (ii) use 3% fewer VLANs in the design resulting in lower processing and memory requirements on switches; and (iii) avoid errors in reconfiguring trunk links that are highly prevalent today - for instance, in our trace we found that 42% of VLANs has redundantly configured trunk links.

As a second contribution, we present algorithms that enable systematic evolution of VLAN designs. At the design level, our algorithms enable operators to make judicious decisions when evolving VLANs such as deciding which VLAN a host must be assigned to, trading off multiple criteria such as broadcast traffic costs, and costs associated with adding additional VLANs in the network, while honoring logical category constraints, and feasibility constraints that limit the number of hosts that may be assigned to any VLAN. At the configuration level, our algorithms automatically identify network-wide dependencies inherent in reconfiguring VLAN trunk links.

A third contribution of the paper is a characterization of VLAN related change activities of a large-scale operational campus network, by analyzing longitudinal snapshots of configuration files obtained over a two year period. Our characterization study provides a taxonomy of the types of changes, the frequency of changes, and explanation of causes behind the changes. We believe that the unique insights gleaned are important to the community in their own right.

Our future work is considering techniques to redesign (rather than just incrementally evolve) VLANs, considering costs of reconfiguration in doing so. We are also exploring extending our approach to other domains such as evolution of routing designs.

Acknowledgment

We thank Brad Devine, Duane Kyburz and other colleagues in the Information Technology Department at Purdue (ITaP) for providing access to the data, and for being generous with their time. This work was supported by NSF award CNS-0721488.

References

[1] T. Benson, A. Akella, and D. Maltz, "Unraveling the complexity of network management," in *Proc. of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

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

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

[4] S. Narain, "Network configuration management via model finding," in *Proc. Large Installations Systems Administration (LISA) Conference*, 2005.

[5] 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. of ACM Annual Conference of the Special Interest Group on Data Communication(SIGCOMM)*, 2008.

[6] D. Maltz, G. Xie, J. Zhan, H. Zhang, G. Hjalmtysson, and A. Greenberg, "Routing design in operational networks: A look from the inside," in *Proc. ACM SIGCOMM*, 2004.

[7] P. Garimella, Y.-W. E. Sung, N. Zhang, and S. Rao, "Characterizing vlan usage in an operational network," in *Proc. of ACM SIGCOMM INM workshop*, 2007.

[8] 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.

[9] Y.-W. E. Sung, C. Lund, M. Lyn, S. Rao, and S. Sen, "Modeling and understanding end-to-end class of service policies in operational networks," in *Proc. of ACM Annual Conference of the Special Interest Group on Data Communication(SIGCOMM)*, 2009.

[10] 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.

[11] Y.-W. E. Sung, S. G. Rao, G. G. Xie, and D. A. Maltz, "Towards systematic design of enterprise networks," in *Proc. of the ACM CoNEXT Conference*, 2008.

[12] A. Mansy, M. B. Tariq, N. Feamster, and M. Ammar, "Measuring vlan-induced dependencies on a campus network," in *Proc. ACM SIGCOMM IMC*, 2009.

[13] Cisco, "Catalyst 2950 desktop switch software configuration guide." [Online]. Available: http://www.cisco.com/en/US/docs/switches/lan/catalyst2950/software/release/12.1_11_yj4/configuration/guide/lrescg.html

[14] Cisco, "Understanding vlan trunk protocol (vtp)," 2007. [Online]. Available: http://www.cisco.com/application/pdf/paws/10558/21.pdf

[15] Cisco, "Troubleshooting vlan trunk protocol (vtp)," 2007. [Online]. Available: http://www.cisco.com/application/pdf/paws/98155/tshoot-vlan.pdf

[16] K. Sripanidkulchai, C. Issariyapat, and K. Meesublak, "Inference of network-wide vlan usage in small enterprise networks," in *Proc. of IEEE Workshop on Automated Network Management*, 2008.

[17] Y.-W. E. Sung, S. G. Rao, S. Sen, and S. Leggett, "Extracting network-wide correlated changes from longitudinal configuration data," in *Proc. of Passive and Active Measurement Conference (PAM)*, 2009.

[18] X. Chen, Z. M. Mao, and K. van der Merwe, "Towards automated network management: Network operations using dynamic views," in *Proc. of SIGCOMM INM Workshop*, 2007.

[19] T. Benson, A. Akella, and D. Maltz, "Phoenix: A system for automatically reconfiguring networks," Poster Session, USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2009.

[20] OpenFlow, "Stanford University," http://www.openflowswitch.org/.

[21] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking control of the enterprise," in *Proc. of ACM Annual Conference of the Special Interest Group on Data Communication(SIGCOMM)*, Kyoto, Japan, Aug. 2007.

[22] M. Casado, T. Garfinkel, A. Akella, M. Freedman, D. Boneh, N. McKeown, and S. Shenker, "SANE: A protection architecture for enterprise networks," in *Proc. USENIX Security*, 2006.

[23] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4D approach to network control and management," *ACM Computer Communication Review*, October 2005.

[24] J. Rexford, A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, G. Xie, J. Zhan, and H. Zhang, "Network-wide decision making: Toward a wafer-thin control plane," in *Proc. ACM SIGCOMM HotNets Workshop*, 2004.

[25] 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. NSDI*, 2005.

[26] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, "The case for separating routing from routers," in *Proc. ACM SIGCOMM Workshop on Future Directions in Network Architecture*, 2004.

[27] C. Kim, M. Caesar, and J. Rexford, "Floodless in SEATTLE: A scalable Ethernet architecture for large enterprises," in *Proc. of ACM Annual Conference of the Special Interest Group on Data Communication(SIGCOMM)*, 2008.

[28] A. Greenberg, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," in *Proc. of ACM Annual Conference of the Special Interest Group on Data Communication(SIGCOMM)*, 2009.

[29] TRILL, "IETF TRILL working group," http://www.ietf.org/html.charters/trill-charter.html, 2003.

[30] H. Ballani and P. Francis, "Conman: a step towards network manageability," in *Proc. ACM SIGCOMM*, 2007.

[31] S. D. Krothapalli, S. A. Yeo, Y.-W. E. Sung, and S. G. Rao, "Virtual man: A VLAN management system for enterprise networks," Demo Session, ACM SIGCOMM, 2009.