A Toolkit for Automating and Visualizing VLAN Configuration

Sunil D. Krothapalli, Xin Sun, Yu-Wei E. Sung, Suan Aik Yeo and Sanjay G. Rao
School of Electrical and Computer Engineering, Purdue University
West Lafayette, Indiana, USA
{skrothap, sun19, sungy, yeo, sanjay}@purdue.edu

ABSTRACT

Virtual Local Area Networks (VLANs) are extensively used in enterprise networks. However, their configuration remains an ad-hoc, complex and error-prone process today. We believe that to eliminate these difficulties, there is need for automation tools, and also need for visualization tools. In this paper, we report on our experience building a VLAN management toolkit, which automates and visualizes common VLAN configuration tasks. We begin by describing common misconfigurations, and their impact on network performance and security. We next present a set of algorithms that automate the VLAN configuration tasks. These algorithms form the back end of the toolkit. The front end of the toolkit consists of an interactive graphical user interface which provides visualization of VLAN operations at multiple granularities, and can be accessed remotely from a web browser. We are in the process of deploying the toolkit at a large campus network which has thousands of switches, and around 800 VLANs. Our initial operational experience shows that the toolkit is effective in both automating configuration tasks, and identifying common misconfigurations. In particular, we have found that (i) more than 40% of the VLANs in the network have redundant links that may lead to security and performance issues. (ii) more than 30% of the VLANs in the network have missing links which may result in connectivity issues and (iii) the root-bridge placements of more than 30% of the VLANs are not optimum, which again may result in performance issues. We believe these insights highlight the benefit and importance of such a toolkit.

Categories and Subject Descriptors

C.2.3 [**Computer Systems Organization**]: COMPUTER-COMMUNICATION NETWORKS—*Network Operations*

General Terms

Algorithms, Design, Management

Keywords

VLAN, Toolkit, Automation, Visualization

SafeConfig'09, November 9, 2009, Chicago, Illinois, USA.

1. INTRODUCTION

Managing Virtual Local Area Networks (VLANs) is one of the unique challenges facing operators of today's enterprise networks. VLANs are extensively used in enterprise networks and are often used to address groups of users as a single unit to ease management even though they are spread over physically disparate locations [9, 14]. In spite of the wide prevalence, VLAN configuration remains a tedious and complex process. On one hand, VLAN configuration is complex, because of the size and complexity of today's enterprise networks (some of them even surpass those of carrier networks [11, 12, 16]), and also because of the network-wide dependencies that are inherent to VLAN design. For example, a simple configuration such as adding a new host to a VLAN may require modifying the configuration of multiple switches in the network (a process called configuring the "trunk" links). On the other hand, there is a lack of tools for automating, visualizing or validating VLAN configuration. In fact, Almost all the VLAN configuration tasks are done in a complete manual and ad-hoc fashion today.

There are many potential sources of errors that arise from this ad-hoc approach to VLAN configuration. Such errors may result in serious connectivity issues, security holes and network inefficiencies. For instance, redundantly configured trunk links may falsely extend the traffic of a VLAN to the part of the network it is not supposed to transverse. This may increase the susceptibility of the network to ARP poisoning [17] and ARP storm [10] attacks. Further, the lack of visualization and validation tools makes it extremely difficult for operators to keep track of and troubleshoot their networks. Hence, there is an imminent need to develop systems that can assist network operators in configuring VLANs.

In this paper, we report on our experience designing and implementing a software toolkit which automates, visualizes and validates a set of common VLAN configuration tasks. The toolkit consists of a back end, which implements a set of algorithms for automating the VLAN configuration, and a front end, which provides an interactive graphical user interface for the operators to submit, view and validate their desired operations. The back end is running on one of our servers as a service process, while the front end can be remotely accessed using a web browser like Mozilla Firefox.

The toolkit is in the process of being deployed at Purdue University, and has begun to be used by the operators. The campus network of Purdue University has around 200 routers, 1300 switches and 800 VLANs. Our initial operational experience shows that the tool is effective in both automating VLAN operations, and identifying configuration errors. In particular, we found that (i) more than 40% of the VLANs in the network have redundant links that leads to security and performance problems; (ii) more than 30% of the VLANs have missing links which result in connectivity issues; and (iii) the root-bridge placements of more than 30% of the VLANs

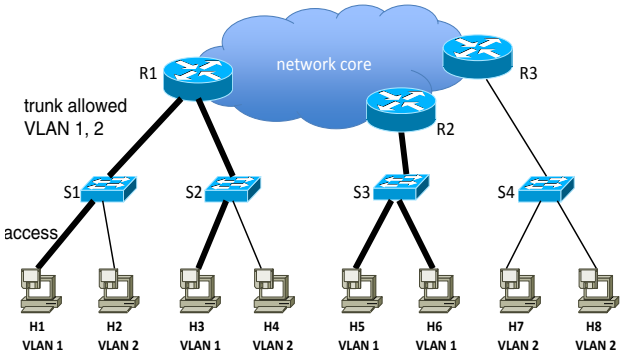

Figure 1: A sample enterprise VLAN setup

are not optimum, which again results in performance problems. We believe these insights highlight the benefit of the tool.

2. CHALLENGES IN ENTERPRISE VLAN CONFIGURATION

In this section, we first give a brief background of VLANs, and then highlight some of the key issues of VLAN configuration.

2.1 VLAN background

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. 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). As another example, the trunk link S4-R3 is configured to only permit traffic of VLAN 2. Typically, a separate spanning tree rooted at a **root-bridge** is constructed per VLAN. For example, the collection of bold links form the spanning tree of VLAN 1, and either R1 or R2 can be selected as the root-bridge. The root-bridge should be chosen judiciously to minimize the resulting spanning tree size, which in turn minimizes the broadcast traffic of the VLAN [3].

2.2 Challenges in VLAN configuration

In spite of the wide prevalence, VLAN configuration remains a tedious, complex and error-prone process today. We believe that to eliminate these difficulties, there is need for automation tools, and also need for visualization tools, as we discuss below.

2.2.1 *Need for automation tools*

Today almost all of the VLAN configuration tasks are performed in a complete manual and ad-hoc fashion. However, many of the tasks are not easy or straight-forward, as a single task typically requires making modifications to multiple dependent devices. Hence configuration errors are not uncommon. Considering the example of extending an existing VLAN to new locations. This happens frequently as departments expand, new facilities get set up, etc. However, to add a new host to the VLAN, the operator must correctly identify and configure all the *trunk* links that are on the path between the access switch of the new host, and the root-bridge of the VLAN, to allow traffic of this VLAN. Failure to do so may result in serious connectivity, security and/or performance issues, as we will describe in detail in Section 2.3. In addition, the operator may also need to change the root-bridge of the VLAN, to adapt to topology changes. This will require him to correctly calculate the spanning tree size for the current root-bridge, and potential alternatives, and choose one that results in the smallest spanning tree size. If the VLANs are large and span many buildings, it becomes very difficult for the operator to correctly identify (and configure) all the trunk links.

2.2.2 *Need for visualization and validation tools*

Operators are also in need of visualization and validation tools which can assist them in better understanding and troubleshooting their networks. VLANs in an enterprise can grow extremely large and span many buildings. For example, all the classroom machines in a university may be grouped into a single VLAN. Without proper visualization, it is very difficult for operators to keep track of important VLAN information, such as the number of hosts in the VLAN, the buildings the VLAN spans, the switches and routers involved, etc, Such information is crucial to operators when they troubleshoot their networks, or when they make design choices such as whether a large VLAN should be partitioned into smaller ones, and how. Further, operators also need tools to assist them in diagnosing their network problems. As we will see shortly, "missing links" and "redundant links" are the common configuration errors, and may cause serious performance and security issues. It would be desirable to have tools to automatically identify such misconfigurations in the network.

2.2.3 *Why existing approaches are insufficient*

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 for configuring trunk links. At a high level, VTP allows an operator to create VTP domains, and each switch can be configured to belong to 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 [8]. 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 [7]. 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 [9].

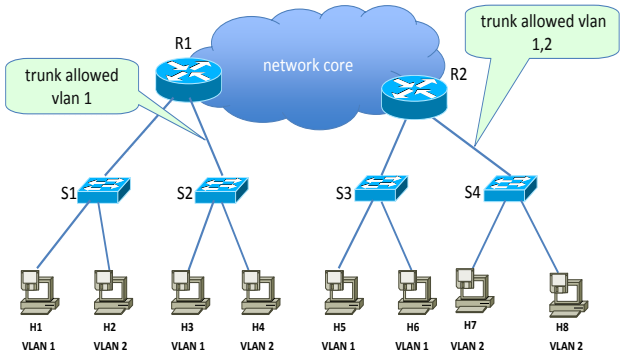

Figure 2: Examples of missing and redundant links that result from VLAN misconfigurations.

2.3 Common misconfigurations and their impact

Due to the complexity of VLAN configuration, and the lack of tools, it is not surprising that many potential errors may arise from the configuration process. In this section, we discuss two types of common errors that arise from configuring trunk links, and their impact on connectivity, security and performance.

2.3.1 Missing links

Figure 2 shows examples of misconfigurations. Host H4 which belongs to VLAN 2 was recently added to the network, but the trunk link R1-S2 was not configured to allow traffic of VLAN 2. We term such a misconfigured link a *missing link* of VLAN 2, since the link should have been configured to allow VLAN 2. Missing links usually cause *connectivity issues.* In this example, H4 is disconnected from the rest of the hosts in the network, because it cannot send or receive traffic through the link R1-S2.

2.3.2 Redundant links

Consider Figure 2 again. 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 not be configured for VLAN 1. Redundant links can cause two types of problems. First, they introduce *security holes* to the network. By extending VLAN traffic to places it is not supposed to be seen, redundant links increase the susceptibility of the network to ARP poisoning attacks, where the attacker inserts fake ARP messages to the network in order to spoof its IP address [17]. In addition, by not restricting the broadcast traffic well, redundant links also increase the susceptibility of the network to ARP storm attacks, where an attacker outside the network may perform a port scan of IPs of the network, causing a lot of ARP broadcast traffic to be generated [10]. Second, Redundant links may also cause *network inefficiencies*. More specifically, redundant links may unnecessarily see broadcast traffic of other VLANs, which may reduce the utilization of the links, or even cause serious congestion at the peak time.

3. ALGORITHMS FOR AUTOMATING VLAN CONFIGURATION

By interviewing the operators, we have identified a set of most common VLAN configuration tasks. These tasks can be broadly classified into, (i) configuration tasks and (ii) visualization and validation tasks. Configuration tasks normally involve extending an existing VLAN into new parts of a network, and deploying a new VLAN altogether in a network. Visualization and validation tasks usually range from viewing the spread of a VLAN to finding sub-optimal root bridge placements to detecting misconfigurations such as missing and redundant links.

In this section, we present a set of algorithms for automating those common VLAN configuration tasks. These tasks include (i) finding the VLAN spread, (ii) finding the optimal root-bridge of a VLAN, (iii) configuring a new VLAN, (iv) extending an existing VLAN to new locations (v) finding redundant links in a VLAN and (vi) finding missing links in a VLAN.

Input: The inputs to all the algorithms are: (i) Network configuration file of all the switches and routers in the network, which can be obtained by collecting the startup configurations from the switches and routers. (2) Network link information, which is needed for constructing the Layer-2 topology of the network. This information can be obtained through various sources. One way is through the use of Cisco Discovery Protocol (CDP) information in a network comprised of Cisco devices. (3) VLAN number, which specifies the VLAN that the algorithm should run on.

Initialization: The following are the initialization steps that are needed by every algorithm. These steps are performed at the start up time of the tool.

```
1: Create a graph g(i,j) = (V,E) from the input network
   configuration and links
2:
3: forall v1, v2 in V do
4:       find the shortest path between v1 and v2
5. end for
6:
```

Step 1 creates a graph of the network using the provided configuration files and link files. Steps 3-5 generate the shortest paths for every pair of nodes in the network using the Floyd Warshall's all-pairs shortest path algorithm. This shortest path information is stored in a matrix and will be used by every algorithm.

3.1 Finding the VLAN spread

Given the vast size of the enterprise networks, a typical VLAN spreads across multiple sites via core and access switches and routers. Finding the VLAN spread helps the operator visualize how and where a VLAN spreads in the network. We will describe the visualization process in Section 4. In this section, we present the algorithm for finding the VLAN spread.

```
1: L = (); S = ()
2:
3: Find the set of access switches which have at least an
     access link configured for the input VLAN and label the
     set as S.
4: Find the configured root-bridge (RB) for the VLAN.
5:
6: foreach as in set S do
7:       find the shortest path from as to RB.
8:       add the new links to set L which are not already
         present.
9: end for
10:
11: return (L)
```

Description of the algorithm: Steps 3-4 find the access switches and the root-bridge for the input VLAN, by parsing the input configuration files. Steps 6-9 find the shortest paths from each access switch to the root-bridge, using the shortest path matrix generated in the initialization steps. Step 11 returns the unique set of links from all the paths identified in previous steps to find the VLAN spread.

The VLAN spread algorithm forms the basic building block for most of other algorithms presented in this section.

3.2 Finding the optimal root bridge

Any router or switch in a network can be a potential root-bridge for a VLAN. However it is important to place the root-bridge judiciously so the resulting spanning tree can be minimized. We present our algorithm for finding the optimal root-bridge for a VLAN.

```
 7: L = (); S = ()
 8:
 9: Find the VLAN spread links using the algorithm
      described in section 3.1, and store them in L. Also
      calculate the set S of all switches from the input network
      configuration files.
10: foreach switch in S do
11:       find the spanning tree SPT with S as the root bridge
12:       if SPT is smaller than current minimum then
13:             set SPT as the current minimum
14:             set S as the current optimal root-bridge ORB
15:       end if
16: end for
17:
18: return (ORB)
```

Description of the algorithm: The algorithm first invokes the VLAN spread algorithm to find the set of links **L** where the input VLAN spreads. It also parses the network configuration files to find the set **S** of all the switches in the network. Next, the algorithm considers every possible switch in the set **S** to be a potential root-bridge for the VLAN. It selects each switch under consideration as the root bridge and creates a spanning tree for the input VLAN. The switch results in the minimal spanning tree is chosen to be the optimal root-bridge. The algorithm breaks ties arbitrarily in the cases more than one minimal spanning tree is found.

3.3 Creating a new VLAN

When operators create a new VLAN, an important task is to correctly identify the set of trunk links to be configured to allow this VLAN. We present our algorithm for automatically discovering the set of trunk links. This algorithm takes an additional input which is the set of access switches (**S**) that are to be included in the new VLAN.

```
 1: L = (); S = (the set of access switches )
 2:
 3: Find the optimal root bridge ORB using the algorithm
      described in section 3.2.
 4:
 5: foreach switch in set S do
 6:       find the shortest path from 'switch' to ORB.
 7:       add the new links to set L which are not already present
 8: end for
 9:
10: return (ORB, L)
```

Description of the algorithm: Given the set of access switches for the new VLAN, the algorithm first decides the optimal root-bridge placement, using the algorithm described in section 3.2. It then identifies the set of unique links **L** from all the shortest paths between the access switches **S** and the optimal root-bridge. The algorithm finally returns the set **L** which contains the list of trunk links the operator should configure to allow the traffic of the new VLAN.

3.4 Extending an existing VLAN

We present our algorithm for extending an existing VLAN to new part of the network. We focus on the task of finding the set of trunk links that need to be configured. This algorithm takes an additional input, which is the set of access switches **S** that the VLAN is being extended to.

```
 1: L = (); EL = (); S = (the set of new access switches)
 2:
 3: Find the VLAN spread links (L) using the algorithm
      described in section 3.1. Also find the configured
      root-bridge (RB).
 4:
 5: foreach switch in S do
 6:       find the shortest path from the RB to the switch
 7:       add links to EL which are not present in L
 8: end for
 9:
10: return (EL)
```

Description of the algorithm: The algorithm first finds out the configured root bridge, and the set of links **L** where the input VLAN spreads. The algorithm then computes the shortest paths from each switch in **S** to the root bridge. It then eliminates those links which are already present in **L** to find new links **EL** which need to be configured to allow traffic of the input VLAN. Finally it returns **EL**.

3.5 Finding redundant links in a VLAN

We present our algorithm for finding redundant links of a VLAN.

```
 1: L = (); L' = (); RL = ()
 2:
 3: Find the VLAN spread links L using the algorithm described
      in section 3.2. Also find the configured root-bridge RB
 4: Perform an enhanced Breadth First Search (BFS) on the
      network with RB being the source node. During the BFS,
      explore a link (i.e., edge) and add the link to L', only
      if it allows traffic of the input VLAN.
 5:
 6: RL =L' - L
 7:
 8: return (RL)
```

Description of the algorithm: The algorithm first finds the set of links and the root-bridge for the input VLAN. It then performs an enhanced breadth first search on the network starting at the root-bridge and explores an edge only if it is configured to allow the traffic of the input VLAN. Using this approach, it is guaranteed to find all the trunk links which are connected and allow traffic of the input VLAN. These trunk links are stored in set **L'**. Next, the set of VLAN spread links **L** (which are the links that connect the root-bridge and at least one access switch) are removed from **L'**. The

remaining links in **L'** are added to set **RL**. Finally, the algor returns **RL** as the set of redundant links of the VLAN. By co uring these links to disallow the VLAN, the unnecessary broa traffic of the VLAN will be eliminated.

3.6 Finding missing links in a VLAN

We present our algorithm for finding the missing links of a V

1: **L** = (); **ML** = ()
2:
3: Find the VLAN spread links L using the algorithm described in section 3.1
4:
5: **foreach** link *l* in L **do**
6: verify if *l* allows input VLAN
7: **if** *l* does not allow input VLAN **then**
8: add *l* to ML
9: **end if**
10: **end for**
11:
12: return (ML)

Description of the algorithm: Steps 1-3 find the VLAN spread links **L** using the algorithm described in section 3.1. Once the VLAN spread is found, the algorithm verifies the configuration of each trunk link in the set **L** to see if it permits traffic of the input VLAN. If the link does not allow traffic of the VLAN, it adds the link to **ML**, the set of missing links. Finally the algorithm returns **ML**.

4. VLAN MANAGEMENT TOOLKIT

We have developed a VLAN configuration toolkit which automates, visualizes and validates the common configuration tasks that we present in Section 3. In this section, we describes the design and implementation of the toolkit and all of its components. The entire software package has been developed using the Java Enterprise Edition (Java EE) framework, which is the premier platform for developing robust and scalable enterprise applications [4]. Most of the components are similar to the Java EE framework [2] except for the Enterprise Java Beans component, which is replaced by custom PerlC Modules component. The PerlC modules form the back end, i.e., the implementation of the algorithms we presented in Section 3, and replace the Enterprise Java Beans component of the Java EE framework. Each PerlC module is written in either Perl or C or sometimes a mixture of both, and is responsible for processing a particular VLAN configuration task submitted by the user. The result generated by the back end will be processed and finally displayed by the front end, which is the graphical user interface. The communication mechanism between the JSPs and PerlC modules is done through a root shell which is in contrast to the communication mode between the JSPs and EJBs in a Java EE framework.

We have already described the back end algorithms in Section 3. In this section, we provide detailed description of the front end of the toolkit, i.e., the graphical user interface.

4.1 Configuration interface

Figure 3 presents a snapshot of the Virtual LAN Management Tool. This interface can be accessed remotely using a web browser such as Mozilla Firefox. It provide a set of configuration actions the network operator can choose to perform on a VLAN. This includes all the VLAN configuration tasks presented in Section 3. The web interface also provides options to customize the output view and the anonymization functionalities.

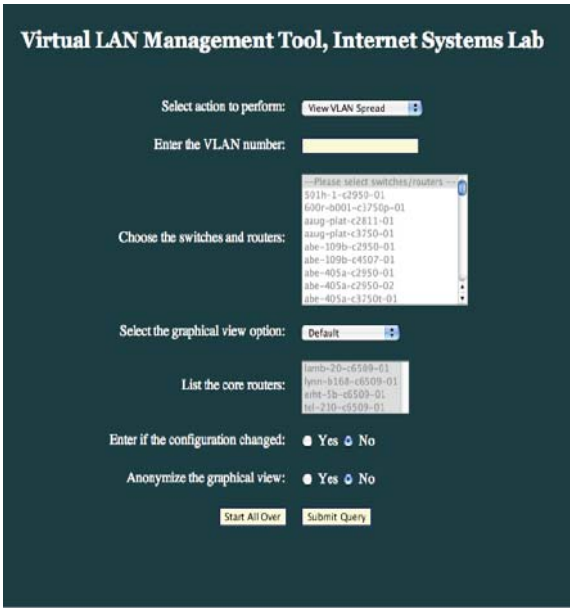

Figure 3: A snapshot of the web configuration interface of Virtual LAN Management Tool

4.2 Visualization Interface

The visualization interface of the system is powered by the ZGR Viewer [5], which is based upon the Zoomable Visual Transformation Machine [6], to generate high quality network graphs. Most of the features available in the software, except the network wide features, support both graphical and text based versions of the output. Both the interfaces present their own advantages. The graphical interface will help the operator gain a network wide qualitative view of how a VLAN is organized. On the other hand, the text based interface gives the exact details of which links are misconfigured or need new configurations.

For each action the network operator submitted through the configuration interface, a three-step process will be performed by our toolkit. First, the corresponding back end PerlC module will be invoked and will produce the configuration result in a link file. The link file contains both the control information as well as data information. The control information is used to give priority processing to nodes like root bridges to represent them uniquely in the output for easy viewing. Next, the link file will be fed into a module we term generator. The generator module will parse the input link file and produce a dot file, which is a graphical representation of nodes and links in text format. Finally, this graphical representation is fed as input to the Graphviz Neato tool, which is an open source visualization software [1]. The Neato tool will then generate a Scalable Vector Graphics (SVG) format file which is displayed to the operator using the ZVTM viewer applet.

4.3 Multiview support

The graphical interface supports two output views: (i) a *default view*, which shows the full topological spread of a VLAN, and (ii) a *core view*, which shows only the set of nodes and links attached to the core of the network. The core view is particular helpful when viewing the spread of large VLANs, as the default view may give too much information which a network operator may not really need and tend to clutter the output. By using core view, the op-

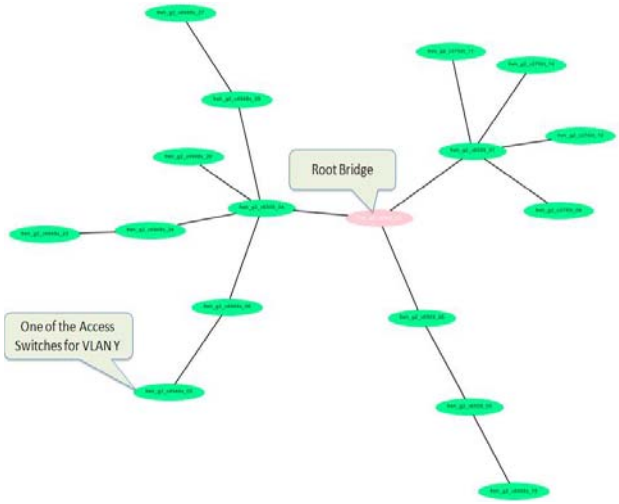

Figure 4: Sample VLAN spread generated using the toolkit

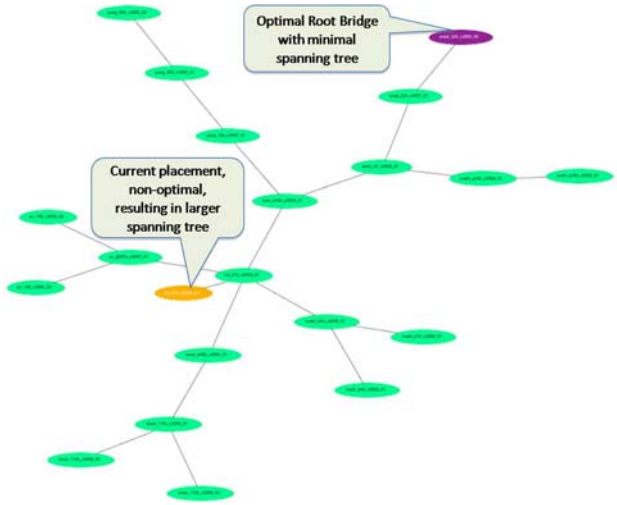

Figure 5: The placements of the optimal root-bridge, and the current root-bridge are shown. Optimum root-bridge results in smaller spanning tree of the VLAN.

erator can choose a list of routers and switches as the core, and change the granularity of the output. We note that the core view functionality can be easily disabled on flat networks where no such core exists.

5. OPERATIONAL EXPERIENCE

We are in the process of deploying the VLAN configuration toolkit at Purdue University, whose campus network has around 200 routers, 1300 switches and 800 VLANs. The campus operators have begin to use the tool, and have provided valuable feedback to us. In this section, we report our initial operational experience with the tool. We are also in the process of releasing this tool to the wider community and hope to have experience from more networks in the future.

We note that all the figures in this section are snapshots of the visualization interface displayed to the users. The text boxes on the figures were added manually later for clarification purpose.

5.1 VLAN Spread

Figure 4 depicts the spread of a VLAN, which is one of the tasks the operator can perform through the graphical user interface. As we can see in the graph, the VLAN spread shows all the nodes (i.e., routers and switches) in the network that the VLAN spreads to. The "leaves" in the graph are the access switches that connect to the end hosts of the VLAN. Note that the end hosts are not shown in the graph. The VLAN spread graph also shows the configured root-bridge of the VLAN.

5.2 Finding optimum root Bridge placement

Figure 5 shows an example where the current root-bridge placement is different from the optimum. In this case, the number of spanning tree links with optimal root-bridge is 19 as compared to 20 with the current placement. We have performed this operation on all the VLANs in the network. The result shows that the root-bridge placements of about 30% of the VLANs are not optimum.

5.3 VLAN Extension

The output of VLAN extension is a set of links to be configured to permit traffic of the VLAN. Figure 6 portrays a scenario where such configuration changes are needed as the VLAN is being extended to new access switches which do not already allow the VLAN. In the graph, the new access switches are drawn in dark color, and the trunk links which need to be configured are marked in bold.

5.4 Finding redundant trunk links

Figure 7 depicts the redundant links identified using the tool for a VLAN. The redundant links are marked in bold. We see that there is a lot of redundancy even for a single VLAN and this leads to superfluous broadcast traffic on all these links. Our toolkit does a good job in identifying all of the redundant links.

We have run the tool for every VLAN in the network. In summary, around 42% of the VLANs have at least one redundant link, and 5% have more than five redundant links. The main reason for the high presence of redundant links is the evolution of the network. As the network evolves, hosts may be removed, or be moved from one VLAN to another, etc., but the configurations on the trunk links often failed to be updated. This highlights the importance of having the VLAN configuration toolkit which automates the discovery of the trunk links.

5.5 Finding missing trunk links

Figure 8 shows an example where a VLAN has several links missing, i.e., these links should have been configured to allow the VLAN, but were not. The missing links are shown as dotted lines in the VLAN spread. From the figure, one could find that nodes A and B (which are switches) are disconnected from the rest of the hosts in the VLAN and hence all the hosts attached to the two nodes are also disconnected. If these hosts are servers which host critical services, it would result in severe unavailability issues.

We have run the tool for every VLAN in the network. We found that around 32% of the VLANs have at least one missing link, and about 2.5% have more than five missing links. We have manually checked the switch and router configuration files to validate these results, as well as discussed our findings with the operators. It turned out that most of the missing links were indeed configuration errors. This again highlights the importance and benefit of the VLAN configuration toolkit.

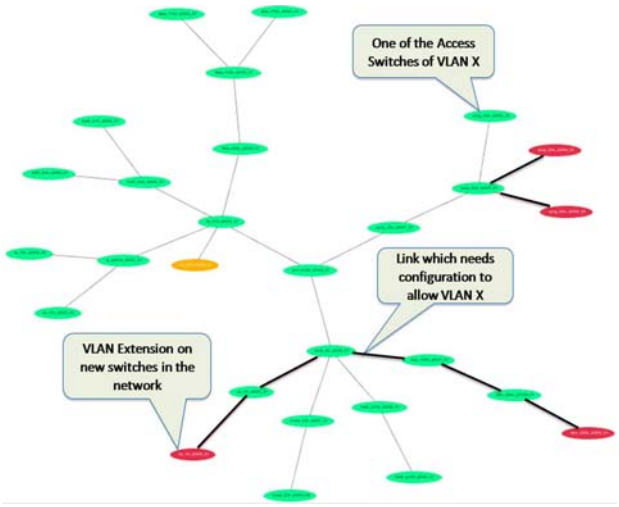

Figure 6: VLAN Extension to new switches. The new switches are drawn in dark, and the trunk links that need configuration are in bold.

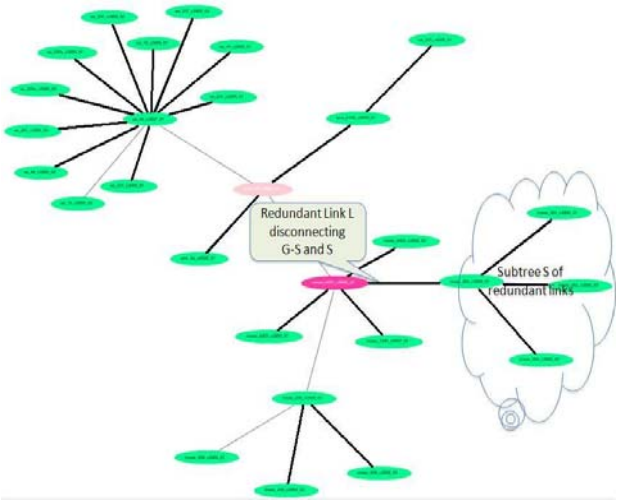

Figure 7: Redundant links in a VLAN. The links in bold are redundant. The node in dark color is the root-bridge.

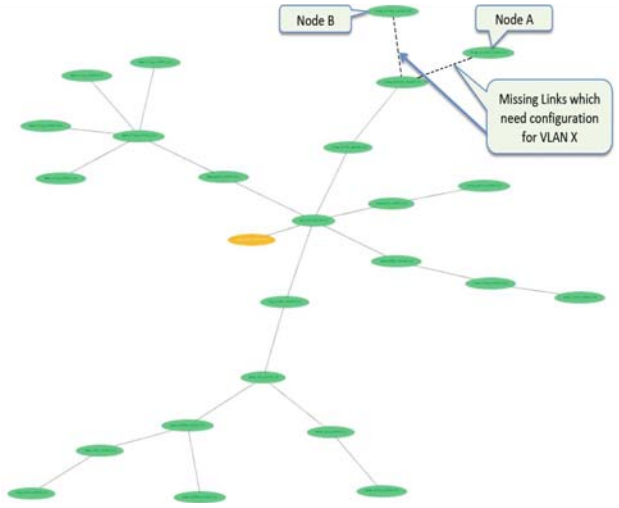

Figure 8: Missing links in a VLAN. The trunk links that should have been configured to allow the VLAN is shown in dotted links. Switches A and B are disconnected from the VLAN.

6. RELATED WORK

Several works have studied VLAN design in enterprise networks. Garimella et al. [9] characterizes VLAN usage in one operational network and exposes several degenerate designs. Sung et al. [16] shows the feasibility of adopting a systematic approach in the VLAN design of *greenfield* networks that are yet to be deployed. Other works include the use of traffic data [13, 15] to expose degenerate design patterns, understand VLAN traffic patterns, and correlate cross-layer faults. By contrast, this work focuses on designing systematic algorithms for automating common VLAN operational tasks. Further, the experience of designing and implementing the VLAN configuration toolkit, and the insight from its initial deployment are also unique. There also exists industry efforts like the Cisco VLAN Trunk Protocol [8] to manage VLANs, however such efforts are limited in functionality. For instance, VLANs that span multiple VTP domains still require manual configuration of trunk links (see Section 2.3). Our work complements these industry efforts by not only automating the trunk link configuration tasks, and also visualizing and validating the effect and impact of the tasks.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we presented our experience in designing and implementing a toolkit for VLAN configuration, which addressed plethora of unique challenges network operators face in managing VLANs in enterprise networks. The back end of the tool implements a set of algorithms that automate a range of common VLAN operation tasks. The front end of the tool provides an interactive graphical user interface which can be accessed remotely using a web browser. To the best of our knowledge, this is one of the first research works to develop tools for automating and visualizing enterprise VLAN operation.

We are in the process of deploying the tool at Purdue University. Our operational experience from the initial deployment shows that, the tool not only automates the configuration tasks, but also assists the operators in visualizing and validating the effect and impact of the tasks. The initial operational experience confirms the effectiveness of the tool, and also highlights its benefits.

In the future, we plan to deploy the tool in other networks and hope to gain more experiences. We also plan to augment the tool to help incremental network evolution, with advanced functionalities such as evaluating network designs and recommending the top K changes that operators must do to get maximum performance benefits.

8. REFERENCES

[1] Graphviz - graph visualization software. `http://www.graphviz.org/`.
[2] Java 2 enterprise edition technology center. `http://java.sun.com/developer/technicalArticles/J2EE/Intro/`.
[3] Spanning tree protocol root guard enhancement. `http://www.cisco.com/en/US/tech/tk389/tk621/technologies_tech_note09186a00800ae96b.shtml`.
[4] Sun developer network. `http://java.sun.com/javaee/`.
[5] Zgrviewer, a graphviz/dot viewer. `http://zvtm.sourceforge.net/zgrviewer.html`.
[6] Zvtm. `http://zvtm.sourceforge.net/`.

[7] Cisco. Troubleshooting vlan trunk protocol (VTP). Online document. http://www.cisco.com/application/pdf/paws/98155/tshoot-vlan.pdf, 2007.

[8] Cisco. Understanding vlan trunk protocol (VTP). Online document. http://www.cisco.com/application/pdf/paws/10558/21.pdf, 2007.

[9] P. Garimella, Y.-W. E. Sung, N. Zhang, and S. Rao. Characterizing vlan usage in an operational network. In *ACM SIGCOMM workshop on Internet Network Management (INM'07), Kyoto, Japan*, 2007.

[10] S. Kumar. Impact of distributed denial of service (DDoS) attack due to ARP storm. In *Proc. of 4th International Conference on Networking (ICN)*, 2005.

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

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

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

[14] S. K. Sadhukhan and D. Saha. Auditing campus-wide local area networks (lans) for virtual lan configurations using a simple network manager. Technical report, Indian Institute of Management (IIM).

[15] 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.

[16] 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.

[17] S. Whalen. An introduction to ARP spoofing. Online document. http://packetstormsecurity.nl/papers/protocols/intro_to_arp_spoofing.pdf, 2001.