CSS-VM: A Centralized and Semi-automatic System for VLAN Management

Fuliang Li
The Institute of Cyberspace and Network Science
Tsinghua National Laboratory for Information Science and Technology(TNList)
Beijing, China
Email: lfl09@mails.tsinghua.edu.cn

Jiahai Yang, Changqing An, Jianping Wu, Siyang Wang, Ning Jiang
The Institute of Cyberspace and Network Science
Tsinghua National Laboratory for Information Science and Technology(TNList)
Beijing, China
Email: {yang, jianping, acq}@cernet.edu.cn
xialanxuan1015@163.com, jiangning85@126.com

***Abstract*—VLANs (virtual local area networks) are widely used in many enterprises, campus, and data-center networks. Although VLANs can restrict broadcast domains and contain hosts in one or separate networks, the management of VLANs is an ad-hoc and error-prone work. In this paper, we design and implement a centralized and semi-automatic system for VLAN management (CSS-VM). Based on the physical network topology and user group (Examples groups are engineering, student cluster, faculty cluster, etc.) information, CSS-VM can decide the number of VLANs that each user group would be partitioned into and how to configure VLAN information on devices automatically. In addition, CSS-VM is able to calculate an optimal spanning tree for each VLAN and monitor the operating status of devices and links. Therefor, it does not need to enable the STP (Spanning Tree Protocol) on devices but still has the ability of avoiding bridging loops and quickly converging from device or link failure. We have evaluated CSS-VM on the topology and VLANs partition data of an operational enterprise network. Our results show that CSS-VM can obviously keep the broadcast traffic cost reasonable, efficiently partition and configure VLANs, quickly converge from link and device failures and intelligently make a balanced use of links.**

Keywords: Network management; VLAN; centralized; configuration;

I. INTRODUCTION

VLANs play an important role in many enterprises, campus, and data-center networks. Enterprise or campus network operators often group the users based on the roles they are playing. Users in a user group are in the same broadcast domain and have the same access permission, which will simplify administration and security tasks. Theoretically, we can create one VLAN for a whole user group. However, if one user group has too many users, the broadcast domain of this VLAN will be relatively large, and a mass of broadcast packets will occupy the bandwidth and therefore influence the network performance. In order to control the broadcast domain in a reasonable scope, further partition of such user group is needed. Obviously, the more VLANs a user group is partitioned, the smaller the broadcast domain is. However, limited by the hardware capacity, the total number of VLANs has an upper bound. Thus, it is important for the operators to seek the tradeoff between the hardware capacity and network performance.

Currently, both the ad-hoc fashion in VLAN design and the complicated configuration in VLAN implementation lead to a poor performance network. In the design phase, operators choose strategies in an ad-hoc fashion without systematic calculation and the results in an approach which may lead to much broadcast traffic, none-usage of some high performance links, and over-load of other links, which is often far from optimal designs. In the implementation phase, operators enable the parameters by manually inputting many commands. For a network which may be composed of many devices produced by different manufacturers, operators must configure the trunk links for VLANs prudentially.

In addition, in the enterprise networks, redundant and hierarchical network design is usually adopted to guarantee high availability of the network (as depicted in Fig.1). But such deployment may introduce bridging loops, which will cause the broadcast storms. So the STP (Spanning Tree Protocol) is enabled on the devices to avoid bridging loops. However, under the circumstance of large network with VLAN overlapping, the utilization of STP is not a good choice because of the overhead of running STP instances, especially when the number of VLANs is large.

In this paper, we design and implement a centralized and semi-automatic system for VLAN management. According to the physical network topology and user group information, our CSS-VM can calculate out how to partition the VLANs and configure VLAN information on each switch automatically. With the help of the monitoring mechanism, CSS-VM can not only quickly react to link and device failures, but also reconfigure the affected VLANs automatically. Moreover, CSS-VM will calculate one of the optimal spanning trees (based on our methodology of VLANs partition depicted in section III.A) for each VLAN and monitor the operating status of devices and links, so it does not need to enable the STP and still has the ability of avoiding bridging loops and quickly converging from device or link failures. We have evaluated our CSS-VM on the topology and VLANs partition data of an operational enterprise network. The results show that CSS-VM can obviously keep the broadcast cost reasonable, efficiently partition and configure VLANs, quickly converge from link and device failures and intelligently make a balanced use of links.

The remainder of this paper is organized as follows. Firstly,

978-3-901882-50-0

the related work is presented in Section II. Then we describe our methodology of VLANs partition and automatic configuration in Section III. Our monitoring mechanisms and adaptive adjusting algorithms for VLANs changing are described in section IV. The evaluation of CSS-VM is presented in section V. Finally our conclusion and future remarks are presented in section VI.

II. Related Work

Spanning tree protocol can not only block the redundant links to eliminate bridging loops, but also can make the layer-2 network load balance. In addition, when an interface breaks down, the protocol can activate the blocked interfaces and recalculate another optimal tree for each influenced VLAN. With the demand of network development, many kinds of spanning tree protocol have been proposed [1-3]. STP is an initialization for VLAN management, but limited by the convergence speed and the possibility of temporary bridging loops. RSTP (Rapid Spanning Tree Protocol) has accelerated the convergence speed. However, the whole layer-2 network can only have one spanning tree, which can be considerably influenced by the topological variation. MISTP/MSTP (Multi-instance/Multiple Spanning Tree Protocol) can reduce the communication cost and computing resources and achieve the goal of load balance. But it still requests devices to make intelligent decisions, which are non-compliance to the current design concept of devices, i.e., making the device as simple as possible. One of the benefits from CSS-VM is that we do not enable STP protocol on devices, which will obviously reduce the load of devices. Therefore, devices can provide more resources to forward data flows. We have evaluated the performance of CSS-VM, and the results illustrate that, without running STP instance, CSS-VM can also avoid bridging loops and quickly react to link and device failures within a tolerant time interval but with with a light load.

Since VLAN is widely used, many researchers have investigated the VLAN usage in enterprise and campus networks [4-8]. These works have revealed the traffic patterns of VLANs and frequent mistakes when configuring the *trunk* links. They also mine the dependencies hidden in VLANs. The most relevant work to this paper has shown the strategies and criteria of VLAN designing [9, 10]. But these works do not concern about the capacity of links, as well as the influence of exiting VLANs on links, which are not beneficial to load balance of links. CSS-VM consideres both the influence of existing VLANs on links and the capacity of links, which allows links to be averagely utilized.

Network configuration is a complicated and error-prine job, which brings burden on operators. Template-driven approaches for configuration are commonly used in ISP networks [11, 12]. L. Vanbever, et al. utilize programs to extract parameters from provisioning databases and then generate configuration snippets[13]. A. Greenberg, etc. propose a reverse engineering method to build provisioning databases from existing network configuration. X. Chen, et al. take advantage of database to abstract configuration information and declarative language to describe domain knowledge (domian knowledge is defined as dependencies and restrictions among network components) [15, 16]. Considering the good scalability of this approach, we integrate it in CSS-VM to automatically configure VLAN information on devices.

CSS-VM adopts a centralized mechanism to design, configure, maintain and adjust the VLANs. It is the first approach that integrates high-level decision making, middle-level adaptive adjusting and low-level configuration commands, which towards the top-down and systematic management of enterprise networks.

III. Methodology of VLANs Partition and Automatic Configuration

There are two prerequisites for centralized and semi-automatic VLAN management. For one thing, each switch should be configured with a management IP address. So our CSS-VM can configure the switches through SSH (Secure Shell) or other remote access ways. For another thing, some management VLANs should be created to manage the switches. Therefore, if a link breaks down, the reconfiguration commands calculated by CSS-VM can be sent to the influenced switches through the redundant links.

In addition, we reference the idea of $D\acute{E}COR$ which uses database to abstract configuration information [16]. There are three types of tables in the database: a) *Regular tables* store the basic information of devices(management IP, password, etc.), interfaces(interface description, group identifier,etc.), links(source interface description, destination interface description, *basic link cost*), *current link cost*, and etc.) *Configuration tables* store the configuration information for VLANs. When CSS-VM inserts records into the *configuration tables*,related CLI commands encapsulated in the scrips will be triggered and executed automatically on switches. c) *Status tables* store the status of devices and links. similarly with *configuration tables*, once a record is inserted into the *status table*, related query scrip will be triggered to check the status of device or link.

A. Methodology of VLANs partition

CSS-VM partitions each user group into one or more VLANs and calculates out the spanning tree for each VLAN based on the following information. First of all, network operators need to manually import the information of the physical network to the *regular tables*, including *phy_network table* (information about the subnets), *phy_device table* (information about the devices), *phy_interface table* (information about the interfaces attached to the devices) and *phy_link table* (information about the links connecting one device with another). CSS-VM will check whether the physical network topology is a connecting graph or not, and only if it is, the topology will be eventually preserved in the regular tables. Secondly, operators should classify the users based on the roles they are playing in the organization. The same kind of users will be divided into a user group. Users in the same user group may be scattered around different buildings, but due to their communication needs, operators must place them into a single logical subnet.

According to the network topology and user group information, CSS-VM will decide how to partition each user group. The partitioning job must comply with the *correctness criterion*, *feasibility criterion* and *performance and cost criteria*,

i.e. users in different user groups must be scheduled in different VLANs, users in a VLAN are limited by the size of the IP block and the broadcast traffic coat of each VLAN need be kept in a reasonable value [9, 10].

Without running STP instances (except management VLANs), CSS-VM builds spanning tree for each VLAN to realize load balance of links, as well as to avoid bridging loops. For a typical enterprise network topology (shown in Fig. 1) with several user groups and VLANs overlapping, several partition programs exist but with different effects. The question for CSS-VM is how to build optimal spanning trees for VLANs and achieve an optimal network performance in global, i.e., links with high capacity will be selected with high priority, links that have been used by other VLANs will be selected with low priority to avoid over loaded links, and the tree should generate least broadcast traffic.

We assume that each user group depicted in Fig.1 needs to be partitioned into two VLANs. The spanning trees shown in Fig. 2 are one of the optimal partitions, which can distribute traffic load to different links. As the centralized management center, CSS-VM can take full advantage of information of the whole network to pursue the partition program with the best performance.

CSS-VM adopts the broadcast traffic cost model, i.e., the broadcast cost of a VLAN is defined as $B = H * A * W$ [9]. *H* denotes the number of hosts in the VLAN, *A* denotes the average broadcast traffic (in pkt/s) generated by a host and *W* denotes the sum of links of the spanning tree. Note that this model does not consider the differences of physical links in capacity, so it is not reasonable in evaluating the broadcast traffic cost of VLANs (the reason depicted bellow). We redefine the broadcast traffic cost B of a VLAN as depicted in equation (1).

$$H * A * C \tag{1}$$

Here, H and A are given the same definition and we assume that broadcast traffic is generated at 2.12 packet/second/source [9], i.e., A=2.12. C denotes the sum of *link cost*. For the same broadcast traffic, links with higher bandwidth and devices with higher performance are less influenced compared with links with lower bandwidth and devices with lower performance. Therefor, we define C as the sum of the *link cost* and strictly choose the device from core layer as the root of the spanning tree rather than choosing the access layer device. In addition, for the link with the same capacity, a link which has carried one or more VLANs is much more influenced by broadcast traffic compared with the link which has not carried any or less VLANs. CSS-VM considers the influence of existing VLANs on current links. So the *link cost* in CSS-VM is composed of *basic link cost* and *current link cost*.We adopt the value of the *link cost* suggested by the STP as the *basic link cost*. For example, if the bandwidth of a link is 1 Gbps, the *basic link cost* of the link is 4, and if the bandwidth of a link is 100 Mbps, the *basic link cost* of the link is 19. We define the *current link cost* as shown in formula (2).

$$l_i.cur\_cost = (1 + p_i * \beta) * l_i.basic\_cost \tag{2}$$

If a link has been carrying some VLANs, when we calculate the broadcast traffic cost of a new VLAN which also contains this link, we should consider the influence of prior

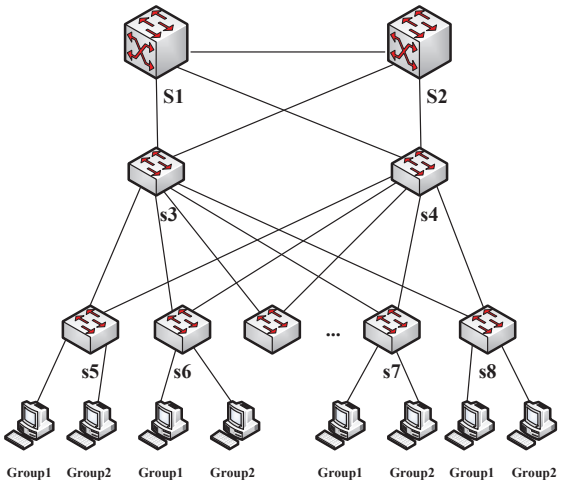

Fig. 1. A typical enterprise network topology with redundant links.

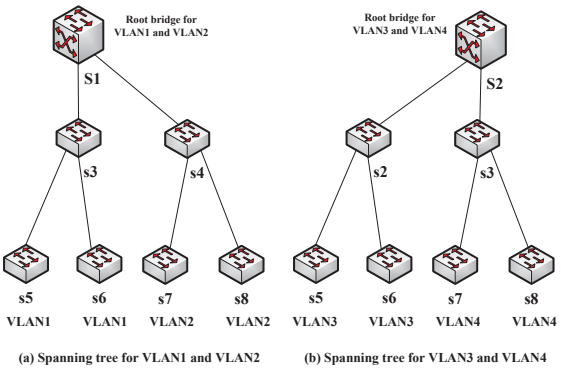

Fig. 2. An optimal VLAN partition for topology depicted in Figure 1.

VLANs on this link, i.e., the link will be more influenced by the broadcast traffic. CSS-VM utilizes equation (2) to calculate *current link cost* for each link. β denotes the tuning parameter (suggested value is 0.35 and details are stated in section V.E) and p_i denotes the number of VLANs that have been allowed on the link(i). Intuitively, when a link is added to a new VLAN, the cost of the link will increase percentage $\beta * p_i$ of the *basic link cost*.

Based on the redefined broadcast traffic cost model and two types of *link cost*, we propose a revised partition algorithms [10]. We omit the details of the algorithms for space reasons. We finally get the set (S) of VLANs for each user group. For each VLAN V in the set S, $V.Set_{access\_device}$ and $V.Set_{core\_device}$ store the devices related to the VLAN V, $V.Set_{access\_interface}$ stores all the *access* interfaces of VLAN V, $V.Set_{core\_interface}$ stores all the *trunk* interfaces and $V.Set_{span\_tree}$ stores all the *trunk links* of VLAN V. The information of each VLAN provides a foundation to configure devices automatically.

B. Methodology of Automatic Configuration

All the VLAN information calculated in III.A will be inserted into *configuration tables*. The ER diagram of the *configuration tables* is shown in Fig 3.

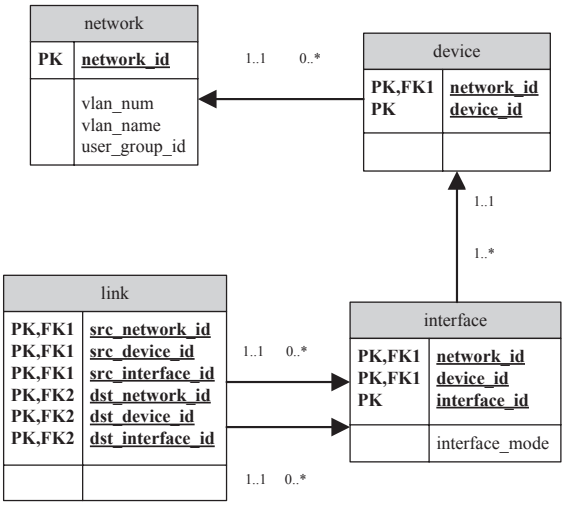

Fig. 3. The ER Diagram of configuration tables

We set a trigger in the *device table* and *interface table*. Once a record is inserted into the *device table* or *interface table*, a trigger will invoke a thread from the thread pooling. The thread calls a shell program to extract configuration parameters from related tables. Then it connects to the device and executes the configuration commands.

We take the device of Catalyst3750 (Cisco IOS Release 12.2(58) SE) as an example to illustrate. When we insert a record into the device table, a trigger will invoke a thread to call a shell program to extract $vlan\_num$ from the *network table*, the management address and password of the device from the $phy\_network\ table$. Then CSS-VM connects the device and executes the following commands.

vlan $vlan\_id$

Similarly, when we fill the *mode* field of the *interface table* with the value of *access* or *trunk*, relevant configuration commands will also be executed automatically. Once a connection with a device is created, CSS-VM will keep the connecting state for a period, which will save time for the next configuration task.

IV. Adjusting Algorithms for VLAN Changes

When a network is in the operating phase, operators need to constantly adjust the network for three reasons. First of all, physical links in the network may break down. This may be caused by upgrading the system of the devices, by devices outage or interfaces disabled. In summary, both interface failure and device failure result in link failure. Secondly, adding a user into a user group, or removing a user from a user group. Finally, adding a user group into the network, or moving a user group from one building to another. All these changes can be adjusted by CSS-VM. For space reasons, we only take the link failure as an example to illustrate.

Without enabling STP, when a physical link is out of order, devices cannot maintain spanning trees for influenced VLANs. CSS-VM must have the ability of monitoring and discovering failures, in addition, recalculating the new spanning tree for each influenced VLAN and updating related configuration automatically. The problem has been resolved by CSS-VM. For one thing, CSS-VM adopts both polling and asynchronous notification mechanisms to acquire the running state of physical links. For another thing, once CSS-VM receives a message about the link failure, it can make the devices ongoing work in a tolerant time interval.

A. Mechanism for Monitoring Link Failures

The **polling mechanism** needs CSS-VM actively and periodically to check the status of physical links. In order to achieve the goal above, CSS-VM will query the $link\_status\ field$ of the $link\_status\ table$, which will trigger inquiring scrip to check the status of interfaces. The inquiring scrip can be construed in two ways. First of all, the SNMP (Simple Network Management Protocol) request encapsulated in the scrip can be sent to get the value of *ifOperStatus*[17]. If the value of *ifOperStatus* is not *up(1)*, the *status field* of the physical link connecting to this interface will be set to linkdown and a link failure warning will be reported to CSS-VM. Secondly, CLI commands packaged in the scrip can be used to directly check the status of interfaces. Taking switch of Catalyst3750 (Cisco IOS Release 12.2(58) SE) as an example, the command of [**show interfaces** type module/number **status**] can reveal the status of interfaces (**connected** or **not connect**), through which we also can gain the status of physical links.

The **asynchronous notification mechanism** needs switches to support SNMP trap. We use MIB (Management Information Base) to define the trap message about an interface failure. When an interface breaks down, the switch will actively inform a trap message to CSS-VM. CSS-VM will translate the trap message and identity whether it is a warning on interface failure or not. If it is, CSS-VM will find out the physical link that the interface attached to and mark another interface of the physical link broken down. No matter which mechanism is chosen, when CSS-VM identifies an interface failure, it will recalculate spanning trees for all the influenced VLANs. But in view of occupying computing resources, the asynchronous notification mechanism is better than the polling mechanism.

Additionally, if an SNMP request or CLI command is executed without a response within the tolerant trials, it can be considered that the switch is broken down, and all the interfaces and physical links attached to this switch will be changed to *linkdown*.

B. Algorithms of Adjusting Influenced VLANs

When CSS-VM identifies a link failure, it must adjust the influenced VLANs as soon as possible. So when CSS-VM calculates the shortest path between any two devices, it utilizes the *basic link cost* rather than *current link cost*. This can simplify the procedure of recalculating the spanning tree for each influenced VLAN. Details of this algorithm are omitted for space reasons. .

V. Evaluation

We evaluate CSS-VM on the topology and VLANs partition data of an operational enterprise network with redundant links to guarantee high availability. The network is composed

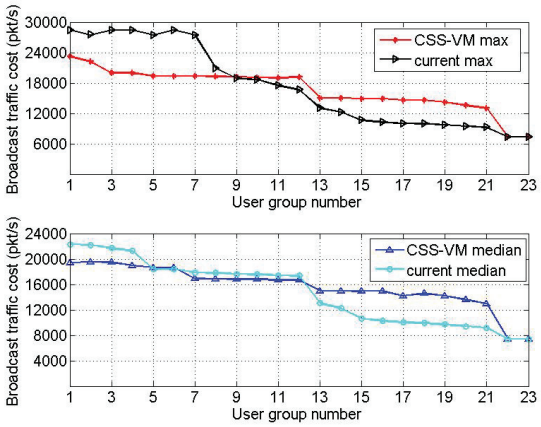

Fig. 4. The Maximum and Median Broadcast Traffic Costs of All User Groups.

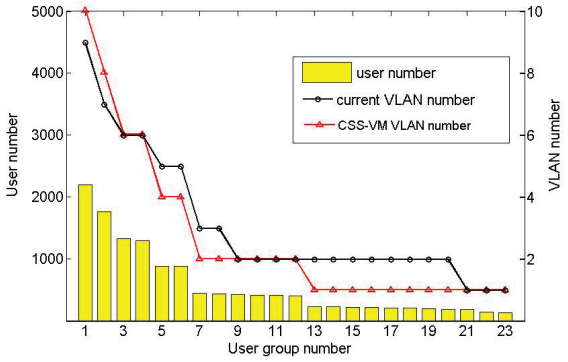

Fig. 5. The Number of Users and VLANs of All User Groups.

of 2 routers, 6 layer-3 switches, 660 layer-2 switches and 13200 hosts partitioned into 78 VLANs. As the head office, the company owns 3 buildings and 23 departments. About 78% of the user groups span only one building, and 74% of them only span one floor. The largest user group spans three buildings and five floors.

A. Keeping the Broadcast Traffic reasonable

One of the *performance criteria* is the broadcast traffic cost. As depicted in Fig.4, the rangeability of the maximum and median broadcast traffic costs produced by CSS-VM in each user group is less than current approache. Note that the variance of the broadcast traffic costs mainly depend on the number of users in each user group (referring Fig.5). CSS-VM authentically partitions some large user groups into several small VLANs with reasonable broadcast traffic costs. For example, the 1st user group with 2200 hosts is partitioned into 9 VLANs by current approach, with the largest and smallest broadcast traffic cost of 28828 pkt/s and 9000 pkt/s respectively, while CSS-VM partitions this user group into 10 VLANs, the largest broadcast traffic cost is 23232 pkt/s and the smallest is 19360 pkt/s. The median broadcast traffic cost of the 1st user group generated by CSS-VM is 19404 pkt/s, while current approach is 23022 pkt/s, which is obvious that the broadcast traffic costs produced by CSS-VM are closer to the median value than current approach. One may notice that CSS-VM also produces some broadcast traffic costs greater than current approach. Through our deep analysis, some small VLANs in a user group are merged by CSS-VM. For example, the 18th user group with 220 hosts is partitioned into two VLANs by current approach, and the broadcast traffic cost of each VLAN is 10100 pkt/s and 10000 pkt/s respectively. In contrast, CSS-VM merges these two small VLANs into a single VLAN with broadcast traffic cost of 19360 pkt/s.

B. Decreasing the Number of VLANs

Another *performance criterion* is the number of VLANs. CSS-VM allows devices to operate without running STP instance, so we do not have to care about the number of VLANs. But actually, too many VLANs will also increase the memory and processing requirement of CSS-VM. So the number of VLANs also need to be controlled within a certain value. As a result, CSS-VM creates 62 VLANs in total, while current approach partitions 71 VLANs. CSS-VM creates fewer VLANs than current approach. As depicted in Fig.5, some small user groups, which don't need to be partitioned into two or more VLANs, are partitioned into one or fewer VLANs by CSS-VM. In fact, most user groups are partitioned into VLANs in an unbalanced and unreasonable way by current approach, which make the broadcast traffic of each VLAN sometimes heavy and sometimes light (referring Fig.4). While CSS-VM can partition large VLANs into several small VLANs, but at the same time, it also merges small VLANs into bigger VLANs under the constraint of threshold. As a result, the number of VLANs shows lower variance. The results reveal that CSS-VM partitions the user groups in a more balanced and reasonable way than current approach does.

C. The Efficiency of Partitioning and Configuring VLANs

One of the most important *performance criteria* of CSS-VM is the efficiency of partitioning user groups and configuring VLANs. We deploy CSS-VM on a Linux server with Intel (R) Core (TM) i7 CPU and 4GB Memory. As depicted in Fig.6, CSS-VM spends 10 seconds partitioning the largest user group with 2200 users and utilizes about 111 seconds to configure VLAN information on devices automatically. While for the smallest user group with 110 users, it costs 0.98 second and 10 seconds for partition and configuration respectively. In order to reduce the configuration time, CSS-VM maintains a thread pooling for all the devices, so many configuration tasks can be executed synchronously on different devices. The thread pooling serves the user group one by one to avoid configuring the same device by two user group at the same time. We have made an interview with the enterprise network operators. For the largest user group, they will spend more than 3 hours accomplishing the configuration task. Automatic configuration can not only reduce the configuring time, but also liberate operators from the complicated and lousy configuration commands, which can also decrease the possibility of misconfiguration.

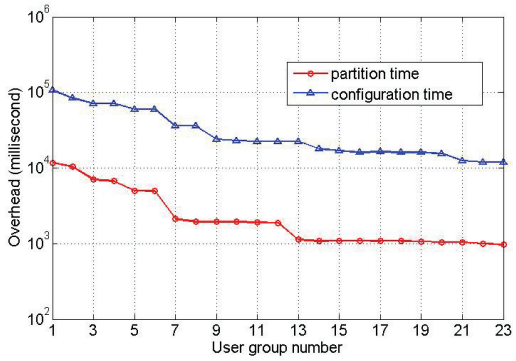

Fig. 6. The Number of Users and VLANs of All User Groups.

TABLE I. REACION TO INTERFACE FAILURE

type	Number of Influenced VLANs					
	1	3	5	7	9	12
Overhead(second)	2.49	2.5	2.61	2.82	2.97	3.2

TABLE II. REACION TO DEVICE FAILURE

type	Number of Influenced VLANs					
	1	3	5	7	9	12
Overhead(second)	2.55	2.64	2.71	2.95	3.16	3.4

TABLE III. OVERHEAD OF ADDING A USER TO A USER GROUP

type	Number of Existing VLANs in the User Group					
	1	2	4	6	8	10
Overhead (millisecond)	2812	2835	2859	2891	2997	3309

D. Reaction to Failures without Running STP Instances

CSS-VM maintains the spanning tree for each VLAN and has the ability of monitoring the status of interfaces and devices, so the devices do not need to enable STP, which will reduce the load generated by STP instances running on the devices. When an interface or a device failure is discovered, CSS-VM will react to it instantly. The converging speed is critical to evaluate the performance of CSS-VM. Both interface and device failures are deliberately designed to validate the convergence of CSS-VM. As shown in Table I, if an interface failure influences only one VLAN, it spends 2.49 seconds recovering the influenced VLAN, and when the number of influenced VLANs increases to 12, the overhead is up to 3.2 seconds. While if an interface failure influences 12 VLANs, which are all maintained by STP instances, BPDU packets will increase sharply and all the related devices will take part in recalculating spanning trees for the influenced VLANs, which will occupy vast bandwidth of links and bring in heavy load to devices. Consequently, BPDU packets may be dropped because of the network congestion, which may cause a long-period convergence. In addition, the normal data flow may also be dropped. Compared with running STP instances, CSS-VM can react to an interface failure within a tolerant time interval but with a lighter load.

Table II shows the overhead of reaction to the device failure. Device failure means all links directly connected to it are disabled, so it is a bit more complicated than link failure. We also evaluate the convergence of VLANs changes. Table III shows the overhead of adding a user to a user group.

TABLE IV. OVERHEAD OF ADDING A USER TO A USER GROUP

core links	Number of VLANs allowed on the core lniks					
	$\beta = 0$	$\beta = 0.1$	$\beta = 0.3$	$\beta = 0.35$	$\beta = 0.5$	$\beta = 1$
L-1	0	1	3	4	4	4
L-2	6	6	5	4	4	4
L-3	7	6	5	5	5	5
L-4	0	1	2	2	2	2
L-5	0	0	1	2	2	2

E. Sensitivity to Parameters

We firstly observe the sensitivity of our algorithms to the N and B_{max} parameters. With large N values, CSS-VM creates more VLANs and decreases the broadcast traffic costs, while with large B_{max} values, the results are opposite. Our results also illustrate that CSS-VM can make the tradeoff that reduce the broadcast traffic cost and decrease the number of VLANs [12]. We pay more attention on the sensitivity to the β parameter. If a *trunk* link is allowed one or more VLANs, CSS-VM will use β to increase the cost of the *trunk* link. So the redundant links with low cost will be chosen to construct spanning trees for new VLANs, which will be beneficial to load balance of links. The load balance of links is an important and troublesome issue when network environment is complicated. We assume that there are 13 small user groups in the same floor sharing 5 core links. The operators require that users in each user group are grouped into a single VLAN. The topology is depicted in Fig.1. As depicted in IV, when β is set to no less than 0.35, the utilization of the links is most balanced comparatively. So we regard 0.35 as the optimal tuning value of β in this network environment. In CSS-VM, through adjusting the value of β, the core links utilization can achieve the most reasonable status under any network topology, which we believe, is crucial to load balance of links.

VI. CONCLUSION

Due to the requirements of administration and security consideration, VLAN is widely used in enterprise, campus and data-center networks. But traditional methods of VLAN design and configuration are ad-hoc and error-prone. In this paper, we design and implement a centralized and semi-automatic system for VLAN management (CSS-VM). We have evaluated CSS-VM on the topology and VLANs partition data of an operational enterprise network. CSS-VM partitions each use group into one or more VLANs with reasonable broadcast traffic costs. It also keeps the number of VLANs in a smaller value than current approach. The efficiency of partitioning and automatically configuring VLANs is satisfying. Without running STP instance, CSS-VM still has the ability of avoiding bridging loops and quickly converging from network failure, which decreases the load of devices and make devices provide more resources for forwarding data flows.

ACKNOWLEDGMENT

This work is supported by the National Basic Research Program of China under Grant No. 2009CB320505, the Natural Science Foundation of China (NSFC) under Grant No. 61170211, the National Science and Technology Supporting Plan of China under Grant No. 2008BAH37B05, Specialized Research Fund for the Doctoral Program of Higher Education (SRFDP) under Grant No. 20110002110056.

REFERENCES

[1] *IEEE Standard 802.1D*, "Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Common specifications-Part 3: Media Access Control (MAC)Bridges", 1998.

[2] *IEEE Standard 802.1W*, "Rapid spanning tree configuration", 2001.

[3] *IEEE Standard 802.1S*, "Virtual Bridged Local Area Networks - Amendment3: Multiple Spanning Trees", 2002

[4] P. Garimella, Y. E. Sung, N. Zhang, and S. G. Rao, "Characterizing vlan usage in an operational network", *in Proc. of ACM SIGCOMM workshop on Internet network management (INM)*, 2007.

[5] M. B. Tariq, A. Mansy and N. Feamster, and M. Ammar, "Measuring VLAN-induced sharing in a campus network", *in Proc. of the 9th ACM SIGCOMM conference on Internet measurement conference (IMC)*, 2009.

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

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

[8] M. Yu, J. Rexford, X. Sun, S. Rao, and N. Feamster, "A survey of virtual LAN usage in campus networks", *in IEEE Communication Magazine*, 2011, Vol.49, pp.98-103.

[9] Y. E. Sung X. Sun, S. G. Rao, G. G. Xie, and D. A. Maltz, "Towards systematic design of enterprise networks", *in IEEE/ACM Transactions on Networking (TON)*, 2011,Vol.19, pp.695-708.

[10] X. Sun, Y. W. Sung, S. D. Krothapalli, and S. G. Rao, "A systematic approach for evolving VLAN designs", *in Proc. of the 29th conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2010.

[11] J. Gottlieb, A.Greenberg, J. Rexford, J. Wang, "Automated provisioning of BGP customers", *in IEEE Network Mag*, 2003, Vol.17, pp.44-55.

[12] W. Enck, P. Mcdaniel, A. Greenberg, S. Sen, P. Sebos, S. Spoerel, S. Rao, "Configuration management at massive scale: system design and experience", *in Proc of USENIX Annual Technical ConferenceUSENIX*, 2007.

[13] L. Vanbever, G. Pardoen, and O. Bonaventure, "Towards Validated Network Configurations with NCGuard", *In Proc. of Internet network managemet (INM) Workshop*, 2008.

[14] D. Caldwell, A. Gilbert, J. Gottlieb, A. Greenberg, G. Hjalmtysson, and J. Rexford, "The cutting EDGE of IP router configuration", *in ACM SIGCOMM Computer Communication Review*, 2004, Vol.34, pp.21-26.

[15] X. Chen, Y. Mao, ZM. Mao, and J. Van der Merwe, "Declarative configuration management for complex and dynamic networks", *in Proc. of the conference on Emerging networking experiments and technologies (CoNext)*, 2010.

[16] X. Chen, Y. Mao, ZM. Mao, J. Van der Merwe, "DECOR: DEClarative network management and OpeRation", *in ACM SIGCOMM Computer Communication Review*, 2010,Vol.40, pp.61-66.

[17] K. McCloghrie, and F. Kastenholz, "The Interfaces Group MIB", June 2000.