

VLANs on Linux

Paul Frieden

To begin, we must have a more formal definition of what a LAN is. LAN stands for local area network. Hubs and switches usually are thought of as participating in a single LAN. Normally, if you connect two computers to the same hub or switch, they are on the same LAN. Likewise, if you connect two switches together, they are both on the same LAN.

A LAN includes all systems in the broadcast domain. That is, all of the systems on a single LAN receive a broadcast sent by any member of that LAN. By this definition, a LAN is bordered by routers or other devices that operate at OSI Layer 3.

Now that we've defined a LAN, what is a VLAN? VLAN stands for virtual LAN. A single VLAN-capable switch is able to participate in multiple LANs at once.

This functionality alone has a variety of uses, but VLANs become far more interesting when combined with trunking. A trunk is a single physical connection that can carry multiple VLANs. Each frame that crosses the trunk has a VLAN identifier attached to it, so it can be identified and kept within the correct VLAN.

Trunks can be used between two switches, between a switch and a router or between a switch and a computer that supports trunking. When connecting to a router or computer, each VLAN appears as a separate virtual interface.

When using trunks, it is important to consider that all the VLANs carried over the trunk share the same bandwidth. If the trunk is running over a 100Mbps interface, for example, the combined bandwidth of all the VLANs crossing that trunk is limited to 100Mbps.

Advantages of VLANs

VLANs provide a number of benefits to a network designer. The first advantage is the number of devices required to implement a given network topology can be reduced. Without VLANs, if your network design requires ten machines divided into five different LANs, you would need five different switches or hubs, and most of the ports would be wasted. With VLANs, this work could be done with one device.

Most routers and standard computers can support a limited number of physical network interfaces. Although dual and quad-port Ethernet adapters are available, these are expensive. For example, a quad-port Ethernet card may cost \$400. VLAN capable switches start at around \$500, but they support many more interfaces.

Depending on the scenario, VLANs and trunks can provide an effective way of segmenting a network without the expense and complexity of managing many physical interfaces.

Types of Trunks

Several trunk encapsulations are available. Trunks can be carried across a variety of interface types, but this article deals only with Ethernet. The two main protocols for carrying VLANs over Ethernet are ISL and 802.1q. ISL was created by Cisco prior to the standardization of 802.1q and is proprietary. 802.1q, on the other hand, is an open standard and is widely supported. Hereafter, references to trunking mean 802.1q-over-Ethernet. As a side note, 802.1q is defined on only 100Mbps or higher Ethernet; it does not support 10Mbps.

How VLANs Work

Trunks using the 802.1q protocol work by adding a 4-byte VLAN identifier to each frame. This is used on both ends to identify to which VLAN each individual frame belongs. When a switch receives a

VLANs on Linux

Paul Frieden

tagged unicast frame, it looks up the outgoing port using both the destination MAC address and the VLAN identifier. When a broadcast frame is received, it is flooded out to all active ports participating in that VLAN.

When a VLAN-aware router or computer receives a tagged frame, it examines the tag to determine to which virtual interface the frame belongs. This virtual interface can have an IP address and behaves basically the same as a normal physical interface.

Some switches have the concept of a native VLAN on a trunk connection. Packets sent out from the trunk port on this VLAN are untagged. Likewise, untagged packets received on this port are associated with this VLAN. Native VLANs on both ends of a trunk must match. A native-VLAN mismatch on the two ends of the trunk causes problems using the native VLAN configured on each end.

Security Considerations for VLANs and Trunks

For all the benefits of VLANs and trunking, some risks must be weighed. As opposed to physical separation between network segments, VLANs rely on the switch to do the right thing. It is possible that a misconfiguration or a bug could cause the VLAN barriers to be broken.

Two risks are associated with VLANs. In the first, a packet leaks from one VLAN to another, possibly revealing sensitive information. In the second, a specially crafted packet is injected into another VLAN. Any attack that could cause the VLAN barriers to break requires a machine directly attached to the physical network. This means that only a local machine can execute an attack against the switch. When the switch is configured properly, the chances of these problems happening are slim, but the possibility still exists. It is up to you to examine your needs and your security policy to determine if VLANs are right for you.

It is beyond the scope of this article to describe exactly how to configure your switch securely, but most vendors provide documentation outlining best practices. Briefly, you should configure at least the following:

- Disable trunking and trunk negotiation on all ports except those absolutely necessary.
- Enable MAC flood protection on all ports.
- Isolate the management VLAN from workstations and servers.

Linux and VLANs

Linux has long been able to connect to VLAN trunks with a kernel patch, and the functionality was integrated into the mainstream kernel in 2.4.14. Kernel 2.6 also supports VLAN trunking.

In order to use 802.1q trunking, simply set the CONFIG_VLAN_8021Q option when configuring your kernel. Depending on what Ethernet card you have, you may need to patch the driver to make VLANs work correctly. This process is discussed in greater detail later in the article.

MTU Issues

As mentioned earlier, 802.1q works by tagging each frame with a 4-byte VLAN identifier. However, some Ethernet drivers assume the maximum frame size is 1,500 bytes. The addition of the 4-byte tag does not leave as much room for data. Thus, although small packets are sent and received correctly, large packets fail. The solution is either to drop the MTU of the VLAN device or to correct the assumptions of the driver.

VLANs on Linux

Paul Frieden

Patches are available on the Linux VLAN Web site for a variety of cards (see Resources). Several drivers work correctly out of the box (or tar.gz, as the case may be), including the e100 driver for Intel-based cards.

Linux Configuration

Configuring VLANs under Linux is a process similar to configuring regular Ethernet interfaces. The main difference is you first must attach each VLAN to a physical device. This is accomplished with the vconfig utility. If the trunk device itself is configured, it is treated as native. For example, these commands define VLANs 2-4 on device eth0:

```
vconfig add eth0 2
vconfig add eth0 3
vconfig add eth0 4
```

The vconfig program can set a variety of other options, including device-naming conventions. Hereafter, these are assumed to be at their defaults.

Once the virtual interfaces are defined, they can be used in the same way as other interfaces. The standard utilities, such as ifconfig and route, all accept VLAN interfaces and behave as expected. For example, all VLAN interfaces can be listed with ifconfig -a.

Depending on your distribution, support may be available for automatically configuring VLANs on startup. Debian 3.0 or greater supports this support, but Red Hat and Fedora currently do not. For other distributions, you simply need to write a script that executes vconfig prior to the main network startup scripts.

Switch Configuration

Because the configuration interfaces for different brands of switches all are different, the focus of this section is the common Cisco 2924. All switch configurations are from this model but should work with little change on other IOS-based switches. A variety of configuration commands are related to trunking, but only the most basic are covered here. The samples also assume the ports all have a default configuration. Specifically, this means all ports are configured as access ports in VLAN 1.

This article focuses on the Linux side of the configuration, so only a basic explanation of the switch commands are given. Listing 1 is a configuration fragment that could be entered into a Cisco Catalyst 2924 switch. See Resources for URLs to complete documentation of these commands.

Listing 1. Configuring a Cisco Catalyst 2924 Switch

```
interface FastEthernet 0/1
  switchport mode trunk
  switchport trunk encapsulation dot1q
  switchport trunk native vlan 1
interface FastEthernet 0/2
  switchport access vlan 2
```

The commands here are fairly self explanatory if you are familiar with the VLAN terminology presented earlier. Briefly, the first section converts the first port into a trunk running 802.1q encapsulation with native VLAN 1. The second section simply moves port 2 into VLAN 2.

VLANs on Linux

Paul Frieden

It is important to see how VLANs are configured and operating on the switch. The first task is to see the status of a particular port. This can be done with `show interfaces <interface> switchport` command.

Listing 2. `show interfaces <interface> switchport`

```
#show interfaces FastEthernet 0/1 switchport
Name: Fa0/1
Switchport: Enabled
Administrative mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
Negotiation of Trunking: Disabled
Access Mode VLAN: 0 ((Inactive))
Trunking Native Mode VLAN: 1 (VLAN0001)
Trunking VLANs Enabled: ALL
Trunking VLANs Active: 1-5
Pruning VLANs Enabled: 6-1001
...
```

Probably the most useful command is the `show vlan` command. It shows you a table indicating which ports are in which VLANs.

Example

The best way to see how VLANs work is by example. Imagine you work for Widgets, Inc. There are about 20 people from several departments working at your location. Ten people work in engineering, two people are in accounting, five people in sales and three people in marketing. Widgets, Inc. currently has a flat network, one in which all the machines are on the same LAN. All of these machines are connected to a Cisco 2924 switch and reside in the 10.0.0.0/24 private network.

Figure 1. Widgets, Inc.'s Private Network

To improve security, you have convinced management to let you segment the network. You already have a Linux firewall running Debian 3.0 facing the Internet, but now you need to extend it to segment the network. The first snag is you have been given only a minimal budget for the project.

After some consideration, you have decided to separate the inside network into four segments: Management, Sales & Marketing, Accounting and Engineering and a DMZ for your assorted servers. The management VLAN has no workstations associated with it and is used only for the switch's configuration interface.

Figure 2. The Segmented Network

Your existing firewall cannot accommodate three more physical interfaces. You recently read an interesting article about how to use VLANs with Linux, which gives you an idea. With VLANs, the new topology can be implemented with the existing interfaces. In fact, the physical layout of your network doesn't change at all. Using VLANs adds a management network to the mix, bringing the total to five.

Figure 3. The Segmented Network with VLANs

VLANs on Linux

Paul Frieden

You also have decided to subnet your existing IP addresses for the new segments. Using a subnet mask of 255.255.255.224 gives you plenty of IPs for each segment and leaves you several spare subnets to use later. You already are using DHCP to assign IP addresses, so client reconfiguration is not an issue.

Listing 3. Assigning IP Addresses

Description	VLAN	IP Subnet
Management	1	10.0.0.0/27
DMZ	2	10.0.0.32/27
Accounting	3	10.0.0.64/27
Engineering	4	10.0.0.96/27
Sales & Marketing	5	10.0.0.128/27

Preparation

Because the network changes here can cause a loss of connectivity, it is important to have everything prepared beforehand. Ensure that your firewall meets the prerequisites above. It also is recommended that you have a serial console connection available before you begin. Obviously, these kinds of changes should be done after business hours.

Preparation is the most important part of a network project. In this case, it is important to have everything planned out well in advance. You should have planned out your firewall policy, server configuration, DNS update and so on. Think about all the functions required for the daily operation of your network, and consider how the changes described here might effect them. For example, reducing the DHCP lease time several days in advance allows the workstations to retrieve their new leases more quickly.

Firewall Configuration

The first step towards the new network configuration is to establish the trunk between the firewall and the switch. On Debian, the `vlan` package contains the required utilities. Most other distributions also offer a package containing these utilities. Compile and install your kernel as you normally would, and enable 802.1q support (`CONFIG_VLAN_8021Q`).

The Debian interfaces file, located in `/etc/network/interfaces`, provides support for creating VLAN interfaces. Each interface is defined as normal, with the addition of a `vlan_native_interface` line. If your distribution does not support defining VLAN interfaces, you need to have a script define them before network startup. Listing 4 shows a Debian interfaces file, using DHCP to retrieve the IP for the outside interface.

Listing 4. A Debian Interfaces File

```
auto lo
iface lo inet loopback
auto eth0 eth1 vlan2 vlan3 vlan4 vlan5
iface eth0 inet dhcp

# VLAN 1 - native management VLAN
iface eth1 inet static
    address 10.0.0.1
    netmask 255.255.255.224
    vlan_raw_device eth1
```

VLANs on Linux

Paul Frieden

```
# VLAN 2 - DMZ
iface vlan2 inet static
    address 10.0.0.33
    netmask 255.255.255.224
    vlan_raw_device eth1

# VLAN 3 - Accounting
iface vlan3 inet static
    address 10.0.0.65
    netmask 255.255.255.224
    vlan_raw_device eth1

# VLAN 2 - DMZ
iface vlan2 inet static
    address 10.0.0.33
    netmask 255.255.255.224
    vlan_raw_device eth1

# VLAN 3 - Accounting
iface vlan3 inet static
    address 10.0.0.65
    netmask 255.255.255.224
    vlan_raw_device eth1

# VLAN 4 - Engineering
iface vlan4 inet static
    address 10.0.0.97
    netmask 255.255.255.224
    vlan_raw_device eth1

# VLAN 5 - Sales & Marketing
iface vlan5 inet static
    address 10.0.0.129
    netmask 255.255.255.224
    vlan_raw_device eth1
```

If you were using a distribution other than Debian, you could put lines similar to the ones in Listing 5 in a startup script that runs before network configuration.

Listing 5. Startup Script for Non-Debian Distributions

```
vconfig add eth1 2
vconfig add eth1 3
vconfig add eth1 4
vconfig add eth1 5
```

Once the new interfaces are defined, you can bring them up using `ifup <device name>`. You also need to `ifdown` and `ifup eth1` to set the correct IP and netmask.

Switch Configuration

Before you begin configuration, make sure the IP address of the switch falls within the new management subnet. The IP configuration is associated with a virtual interface. This is normally VLAN1.

VLANs on Linux

Paul Frieden

Listing 6. IP Address for VLAN1

```
interface VLAN1 ip address 10.0.0.2 255.255.255.224
```

The firewall is connected to port 1 on the switch, which is referred to as FastEthernet 0/1 in IOS notation. The first task is to set the encapsulation and native VLAN, then you can enable the trunk.

Listing 7. Enabling the Trunk

```
interface FastEthernet 0/1
  switchport trunk encapsulation dot1q
  switchport trunk native vlan 1
  switchport mode trunk
```

Once the trunk is active, you need to move ports from the default VLAN into their new one. This is done by entering the interface configuration and issuing `switchport access vlan <vlan id>`. Although not necessary, it is helpful to physically group VLANs to make them easier to manage.

Listing 8. Moving the Ports

```
interface FastEthernet0/2 switchport access vlan 2
interface FastEthernet0/3 switchport access vlan 2
interface FastEthernet0/4 switchport access vlan 3
interface FastEthernet0/5 switchport access vlan 3
interface FastEthernet0/2 switchport access vlan 3
```

Once your changes are complete, you can see which ports are in which VLAN by using the `show vlan` command.

Finishing Up

The first order of business is to test whether you can move packets of all sizes successfully without MTU issues. Packets above 1,476 bytes should trigger any MTU issue you have. This can be tested by pinging from the firewall to a machine on a non-native VLAN. If small packets work but large packets do not, you most likely have an MTU issue.

Because you are using DHCP, you now need to update your `dhcpd.conf` file to reflect the new subnets. Once it is restarted, client machines start to receive their new IP addresses.

Without a policy, a firewall is useless. Unfortunately, defining that policy is beyond the scope of this article. However, a variety of effective tools are freely available for this purpose.

Now that everything is working, we need to make sure the switch's new configuration is written to memory. This is done from enable mode using the `write memory` command.

Conclusion

As you can see, VLAN trunking can be a valuable tool. I hope you have learned where it can be useful, the risks and benefits of using it and the basics of its configuration. Even though this document focuses on a Cisco 2924 switch, it shouldn't be difficult to translate the configuration here to any switch that supports 802.1q trunks.

I would like to give special thanks to Cheryl Lehman for helping to make my first article readable and to Randall Shutt for reviewing the content.