Instituto Politécnico Nacional

Centro de Innovación y Desarrollo Tecnológico en Cómputo

Diseño y desarrollo de una aplicación gráfica para la configuración de redes virtuales de área local en un sistema operativo LINUX

TESIS QUE PARA OBTENER EL GRADO DE MAESTRÍA EN TECNOLOGÍA DE CÓMPUTO

PRESENTA:

Ing. Luis Alberto Marín Sánchez

DIRECTORES DE TESIS:

M. en C. Marlon David González Ramírez

M. en C. Eduardo Vega Alvarado

México, D. F. Julio, 2011

SIP-14

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 11:00 horas del día 24 del mes de junio de 2011 se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del CIDETEC para examinar la tesis de grado titulada:

"DISEÑO Y DESARROLLO DE UNA APLICACIÓN GRÁFICA PARA LA CONFIGURACIÓN DE REDES VIRTUALES DE ÁREA LOCAL EN UN SISTEMA OPERATIVO LINUX"

Presentada por el alumno:

MARÍN	SÁNCHEZ	LUIS ALBERTO
Apellido paterno	materno	nombre(s)

Con registro: B 0 9 1 3 6 4

aspirante al grado de:

MAESTRÍA EN TECNOLOGÍA DE CÓMPUTO

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***SU APROBACIÓN DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

M. EN C. JUAN CARLOS GONZÁLEZ ROBLES
Presidente

M. EN C. MAURICIO OLGUÍN CARBAJAL
Secretario

M. EN C. EDUARDO VEGA ALVARADO
Primer Vocal
(Director de Tesis)

M. EN C. MARLON DAVID GONZÁLEZ RAMÍREZ
Segundo Vocal
(Director de Tesis)

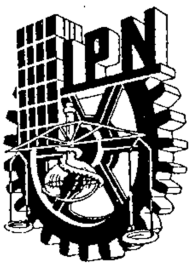
M. EN C. ADAUTO ISRAEL ORTIZ ROMERO
Tercer Vocal

DR. ROLANDO FLORES CARAPIA
Suplente

EL PRESIDENTE DEL COLEGIO

DR. VICTOR MANUEL SILVA GARCÍA

S. E. P.
INSTITUTO POLITECNICO NACIONAL
CENTRO DE INNOVACION Y DESARROLLO TECNOLOGICO EN CUMPUTO



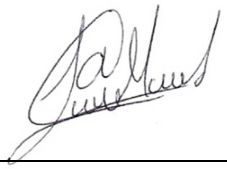
INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México, D.F el día 25 del mes julio del año 2011, el que suscribe Luis Alberto Marín Sánchez alumno del Programa de Maestría en Tecnología de Cómputo con número de registro B091364, adscrito al Centro de Innovación y Desarrollo Tecnológico en Cómputo, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del M. en C. Marlon David González Ramírez y el M. en C. Eduardo Vega Alvarado y cede los derechos del trabajo intitulado Diseño y desarrollo de una aplicación gráfica para la configuración de redes virtuales de área local en un sistema operativo LINUX, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección lams851@hotmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.



Luis Alberto Marín Sánchez

Resumen

En la actualidad las redes de cómputo se han convertido en una necesidad para el ser humano, ya que ofrecen una gran cantidad de servicios para llevar a cabo casi cualquier trabajo, sobre todo para manipular información. Cada vez existen más variantes tecnológicas para dar soluciones a las dificultades presentadas durante la implementación de dichas redes; entre ellas está la creación de las redes virtuales de área local llamadas VLAN. Este tipo de redes permite crear redes lógicas independientes a partir de una red física a través de un conmutador con la finalidad de reducir el dominio de difusión, lo que lleva a una mejor administración de la red.

Ante la variedad de fabricantes existentes en el mercado, y el hecho de que cada tecnología trabaja de distinta forma, es necesario adquirir un conocimiento previo para hacer uso de los equipos y poder llevar acabo alguna implementación. Por otra parte, existe una alternativa que en los últimos años ha tenido un gran avance en el área de redes, dada la gran cantidad de personas que hacen aportaciones en ella con la finalidad de compartirlas, para que de este modo siga el continuo crecimiento en dicha área. Estamos hablando del sistema operativo LINUX.

En LINUX se pueden implementar redes virtuales de área local haciendo uso de una herramienta llamada ***vconfig***, la cual permite crear interfaces de red virtual, asociadas a una tarjeta de red física y que se pueden configurar al igual que cualquier otra tarjeta de red. En este sistema operativo solo existe la configuración en modo línea de comando, por lo que esta tesis tiene la finalidad de utilizar esta herramienta y proporcionar una interface gráfica con un ambiente más práctico y eficiente para poder llevar acabo esta misma configuración.

El presente trabajo muestra las vulnerabilidades que puede presentar un sistema de software libre, específicamente enfocándonos a las distribución Open Suse 11.2; pero además muestra las ventajas y desventajas de hacer uso de equipos que tienen un respaldo por parte del fabricante y de otros que son genéricos. Por otra parte, ofrece la creación de una nueva herramienta gráfica para la configuración básica de las redes virtuales de área local en un sistema operativo LINUX. La finalidad de crear esta herramienta es ofrecer a los usuarios una alternativa más para poder llevar a cabo este tipo de implementaciones, ya que el uso de una interface gráfica permite llevar a cabo configuraciones de una forma más práctica y sencilla.

Abstract

Nowadays computer networks are a necessity for people, because these networks offer a vast quantity of services designed to perform any kind of work, and the most important thing is that we can manipulate that information. Due to technological advances there are more network platform vendors who have given solutions to the difficulties that have arisen in the development of Networks implementation. Among them is the creation of virtual local area networks known as VLAN. Such networks allow the creation of independent logical networks from a physical network through a switch, with the aim of reducing the broadcasting domain, giving us the opportunity of having a better network administration.

With such a great variety of brands on the market, and taking into account the fact that every technology works in a different way, the acquisition of previous knowledge is necessary, so we can use the pieces of equipment in order to perform any implementation. There is another alternative, which in recent years has had a successful progress in the networking area since there are many people making contributions, sharing the growth in this field. We are talking about the LINUX operating system.

With LINUX you can implement Virtual Local Area Networks with the usage of a tool called vconfig, which allows the creation of virtual network interfaces, associated with a physical network card, and these can be then configured like any other network card. In this operating system that configuration is possible only in the command line mode, so the objective of this thesis is to use this tool as a base and provide a graphical interface with a more practical and efficient environment, so the same configuration can be carried out.

This thesis shows the vulnerabilities of a free software system, focusing specifically on the distribution of Open Suse 11.2, but also shows the advantages and disadvantages of using computers that have a back-up from the manufacturer and others that are generic. Additionally, it offers the creation of a new graphical interface as a tool for basic configuration of Virtual Local Area Networks in a LINUX operating system. The purpose of creating this tool is to offer users an alternative to develop these kinds of implementations, since the usage of a graphical interface allows to configure in a more practical and simple way.

Agradecimientos

A mis padres por brindarme el apoyo para realizar mis objetivos y metas que he logrado en mi vida, por los innumerables sacrificios que han hecho por mí. Estoy infinitamente agradecido.

A mis directores de tesis Marlon David González Ramírez y Eduardo Vega Alvarado, por brindarme su apoyo y confianza para realizar este trabajo de tesis.

Al IPN-CIDETEC por ser la institución que me permitió realizar mis estudios de posgrado en tecnología de cómputo.

A mis hermanos por ser las personas que siempre han sido un ejemplo a seguir.

Un agradecimiento especial a Arianne Mancilla Ponce por su paciencia, cariño, y confianza, que ha estado conmigo en todo momento. Gracias.

A mis amigos que siempre me brindaron su apoyo incondicional en los buenos y malos momentos.

ÍNDICE GENERAL

I. Planteamiento del problema

El gran avance tecnológico que ha tenido el área de redes de computadoras se manifiesta en la continua aparición de nuevas tecnologías. En este entorno, CISCO es una empresa principalmente dedicada a la fabricación, venta, mantenimiento y consultoría de equipos de telecomunicaciones, cuya infraestructura le permite ser el proveedor de servicios número uno, dominando la mayor parte del mercado mundial. CISCO desarrolló la plataforma de Redes Virtuales de Área Local (VLAN), lo que le permite ser líder en dicha tecnología.

La implementación de las VLAN se puede realizar desde dos entornos de configuración: línea de comando y aplicación web, en ambas se utiliza la línea de comando solo que la diferencia está en que una permite un entorno gráfico interactivo y la otra es un ambiente de consola. Cabe destacar que las dos se basan en el sistema operativo llamado IOS (*Sistema Operativo de Interconexión de Redes)* el cual permite programar y administrar equipos tales como conmutadores y enrutadores por medio de un ambiente gráfico *(web).*

Durante los últimos años un gran porcentaje de usuarios ha migrado a LINUX, pero lamentablemente aún existe temor a conocer el sistema, ya que se tiene la idea de que sus entornos son para usuarios expertos en áreas de programación; por otra parte, la implementación de las VLAN en este sistema llega a ser una tarea complicada y confusa, ya que depende de que tan relacionado está el usuario con dicho sistema y qué tipo de información ha adquirido de él.

En LINUX se utiliza una herramienta llamada VCONFIG; con esta herramienta se implementan las VLAN accediendo a la consola y utilizando la línea de comando. Ello implica tener conocimiento sobre la configuración y uso de comandos para realizar dicha actividad, lo cual se complica debido a la poca información veraz documentada para hacer uso eficiente de esta herramienta y muchas otras más. Este tipo de problemas trae como consecuencia que los usuarios emigren a otras alternativas que permitan una mejor interacción entre el usuario y sistema, donde los entornos en los cuales se trabaja les sean más sencillos, aunque ofrezcan una menor estabilidad.

El enfoque de esta tesis está dirigido a aquellas personas que inician desarrollos en LINUX, y no han tenido la oportunidad de interactuar con el

sistema, de tal forma que cuando tengan la necesidad de usarlo puedan acceder y encontrarse con ambientes sencillos y fáciles de implementar.

II. Objetivo General

Diseño, desarrollo y documentación de una aplicación en un ambiente gráfico, para la configuración de Redes Virtuales de Área Local (VLAN) desde el sistema operativo LINUX.

III. Objetivos particulares:

1.- Estudio y evaluación de lenguajes visuales sobre un entorno LINUX, en los escritorios KDE y GNOME.

2.- Análisis de los entornos de desarrollo en LINUX.

3.- Análisis de las distribuciones LINUX Fedora y Open Suse.

4.- Análisis del uso de la herramienta VCONFIG para la construcción de VLAN.

5.- Desarrollar la herramienta gráfica para administración básica de VLAN.

IV. Justificación

Microsoft Windows, siendo uno de los sistemas operativos más usados en la actualidad presenta una desventaja muy importante, su uso requiere el pago de una licencia; por su parte, LINUX es un sistema operativo de código abierto, y cualquier usuario puede tener acceso a la mayor parte de sus distribuciones. Lamentablemente se ha propagado poco la idea de hacer uso del software libre, por lo que poca gente conoce dichos sistemas. Existen comunidades de desarrolladores que se encargan de implementar software de código abierto y lo

distribuyen libremente; este software tiene la ventaja de que los usuarios pueden acceder al código para implementar adaptaciones y mejoras.

En este trabajo se desarrolla una opción para crear y eliminar VLAN's, donde el usuario que esté haciendo desarrollos en este campo tenga una herramienta práctica y simple para llevar a cabo la implementación de las VLAN sobre el sistema operativo LINUX, aprovechando el soporte que el núcleo (*Kernel*) de LINUX tiene para la norma IEEE 802.1Q Para llevar a cabo dicha aplicación es necesario verificar la distribución y núcleo con los cuales se trabaje; si ya está instalada la herramienta o será necesario instalar la herramienta correspondiente.

V. Organización de la tesis

En el Capítulo 1 se mencionan y explican tanto los conceptos generales sobre los elementos que intervienen en las VLAN, como el entorno de desarrollo para la aplicación, ofreciendo un panorama general para el lector. El Capítulo 2 se enfoca al desarrollo de la aplicación y las herramientas que se requieren, mostrando el manejo de los archivos utilizados para su programación. En el Capítulo 3 se muestra la configuración de las VLAN haciendo uso tanto de la herramienta existente en la línea de comandos, como de la aplicación desarrollada. En el Capítulo 4 se compara la interfaz de línea de comando y la aplicación gráfica, recalcando los aspectos importantes de cada una de ellas; así mismo, se incluyen las pruebas realizadas para verificar su operatividad. Por último, se presentan los resultados y conclusiones obtenidas, así como una recomendación sobre trabajos futuros en esta línea de desarrollo.

Capítulo 1

1.1 Introducción

El manejo e intercambio de información es fundamental cuando se requiere llevar a cabo actividades que involucren equipos de cómputo compartiendo recursos entre los usuarios. Una solución a esta problemática es el desarrollo de las Redes de Área Local *(LAN);* en este tipo de redes se tiene una serie de consideraciones con respecto a la conexión que se lleva a cabo entre los distintos nodos, mismos que pueden estar en ubicaciones geográficamente distintas, para obtener el mejor camino al establecer una comunicación entre al menos dos usuarios. Lo anterior se denomina conmutación de redes; inicialmente, las comunicaciones se basaban en dispositivos llamados concentradores o *hubs*, los cuales presentaban dificultades al transferir información, originando que en la red existiera un gran número de choques denominados colisiones. Este problema se redujo en gran medida mediante la segmentación de la red, al implementarse dispositivos tales como el puente (*bridge*) y el conmutador (*switch*), los cuales permiten segmentar los dominios de colisión alcanzando una mayor eficiencia[1].

Existe un tipo de red que puede ser implementada dentro de una LAN, denominada Red Virtual de Área Local (*VLAN: Virtual Local Area Network*). Este tipo de redes es muy útil, ya que permite segmentar a través de dominios de difusión *(Broadcast).* Los dominios de difusión son grupos de dispositivos que envían y reciben mensajes a toda la red; al implementarse esta tecnología es posible tener una configuración lógica de la red a través de los conmutadores, los cuales están interconectados a los enrutadores (*Router*). Estos últimos dispositivos enlazan a las diferentes VLAN, independientemente de su cableado físico[2]; es decir, la configuración es de tipo lógico.

Una red de área local que está interconectada por muchos conmutadores puede dividirse en muchas VLAN, que tienen el comportamiento de pequeñas LAN independientes [3]. Para configurar una VLAN es necesario que los equipos soporten el estándar IEEE 802.1Q [4], aunque algunos equipos pueden trabajar con protocolos como ISL (*Inter Switch Link*), protocolo propietario de CISCO que mantiene información sobre el tráfico entre enrutadores y conmutadores, y el VLT (*Virtual LAN Trunk*) de 3Com, que contiene la información sobre las VLAN. Los

principales problemas para la implementación de las VLAN se derivan de la poca información disponible al respecto; se encuentran artículos sobre el tema en la IEEE, pero el acceso a este tipo de información no es abierto.

1.2 Redes de computadoras

Una red de computadoras es una interconexión de dispositivos de cómputo que permite tener acceso al hardware y software para compartir recursos tales como impresoras, red, correo electrónico, información, etc. Las redes de computadoras se originaron por una necesidad del intercambio de información entre el personal de determinadas áreas de trabajo, y con el paso del tiempo se fueron desarrollando, de tal forma que todos los equipos estuvieran comunicados tanto en áreas pequeñas como en un edificio, o cualquier parte del mundo como dos países que se encuentren en áreas geográficamente muy distantes.

Cuando inicialmente se implementaron las redes las actividades a realizar se volvían más simples, eficientes y sobre todo se reducían costos. Posteriormente se dio origen a distintos tipos de redes que se fueron clasificando con base en las necesidades que se requerían. Así surgen las redes LAN, abarcando áreas pequeñas; sin embargo, cuando se requería el intercambio de la información no solo dentro de un edificio o un área determinada las LAN dejan de ser eficientes. Por tal motivo surgen las redes MAN *(Metropolitan Area Network)*, que se pueden extender en ciudades enteras o también partes de una. Por último se tienen las WAN *(Wide Area Network)* o redes de área amplia; estas redes se comunican con otras redes en áreas geográficas muy grandes, permitiendo que las empresas pudieran comunicarse entre sí a muy grandes distancias. La figura 1.1 muestra la red más sencilla entre dos equipos de cómputo.

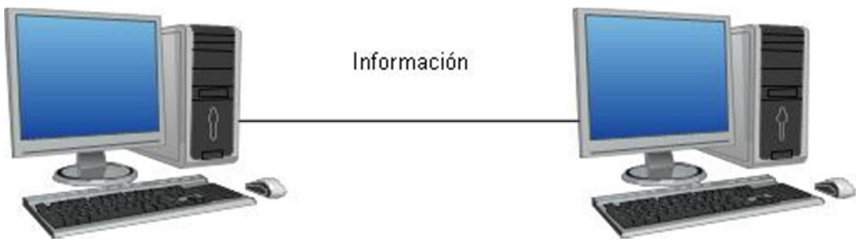


Figura 1. 1 Red punto a punto

1.2.1 Redes de área local

Una LAN es una red de comunicación cuyo alcance es pequeño, solo abarca un área no mayor a la de un edificio, o un conjunto de oficinas[5]. (Ver figura 1.2) Algunas tecnologías LAN son: Ethernet, Token Ring, FDDI. Actualmente las LAN juegan un papel muy importante en el intercambio de información; lo que ha permitido que se haya extendido el uso de estas redes es el aumento en la utilidad de las computadoras personales, estaciones de trabajo y servidores. Otro factor muy importante es que los precios del hardware y software han ido disminuyendo significativamente con el paso del tiempo, dejando la computación virtualmente al alcance de todos.

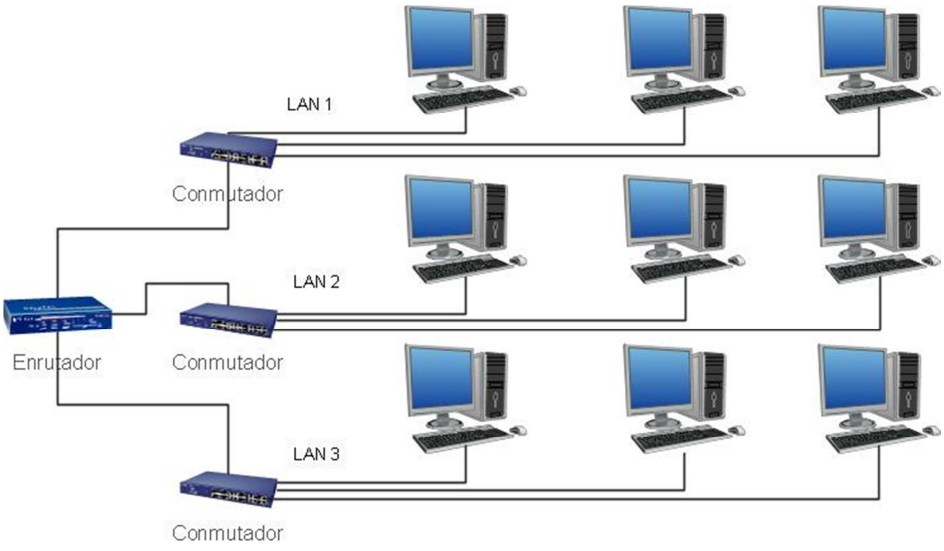


Figura 1. 2 Red LAN

1.2.2 Modelo OSI

El modelo OSI *(Sistema de Interconexión Abierto)* es un modelo de referencia para comunicaciones de red, el cual tuvo su origen en el año de 1984 debido a que existían distintos fabricantes los cuales introducían nuevas tecnologías y productos, pero empleaban especificaciones e implementaciones diferentes, lo que era un problema para la interoperabilidad de las redes. El

modelo de referencia OSI está dividido en siete capas, cada una con una función específica para llevar a cabo la tarea de la comunicación de red.

7. Aplicación: Esta es la capa más cercana al usuario final, en forma de programas de uso común, tales como procesadores de texto, hojas de cálculo, navegadores de Internet, etc. En este nivel se tienen acceso a los servicios de red, aunque esta capa no proporciona servicios a alguna otra.

6. Presentación: Su tarea principal es que la información enviada hacia la capa de aplicación se pueda leer por la capa de aplicación de otro sistema, es decir se encarga de traducir múltiples formatos de datos basándose en un formato común; otra tarea importante es el cifrado y descifrado de los datos.

5. Sesión: Aquí se establece, administra y finaliza la sesión entre dos equipos; también sincroniza el diálogo entre las capas de presentación de los dos equipos y administra el intercambio de datos.

4. Transporte: Se encarga del servicio de transporte de datos entre dos computadoras: establece, mantiene y finaliza adecuadamente los circuitos virtuales; para proporcionar un servicio fiable se emplea la detección y recuperación de errores en el transporte y la información en el control de flujo.

3. Red: Esta capa proporciona conectividad y la selección de la ruta entre dos computadoras las cuales pueden estar en zonas geográficamente separadas; también se ocupa del direccionamiento lógico.

2. Enlace de datos: Proporciona tránsito fiable de datos a través de un medio físico, al tiempo que se ocupa del direccionamiento físico, la topología de la red, el acceso al medio, notificación de errores y el control del flujo.

1. Física: Se ocupa de los medios eléctricos, mecánicos y el procedimiento para mantener y desactivar un enlace físico entre los sistemas finales, por ejemplo los niveles de voltaje, velocidad y distancias de transmisión, conectores, etc.

1.2.3 Topologías físicas

Esta topología define cómo se interconectan físicamente los dispositivos de una red; esto es, cómo es que se encuentran enlazados entre sí. Una red tiene dos tipos de topologías: la física y la lógica. Como su nombre lo indica, la física se refiere al medio físico, mientras que la lógica especifica el modo de comunicación interna de los dispositivos.

Algunas de las topologías físicas más comunes son:

Bus: Todas las computadoras se interconectan mediante un cable único, es decir, el medio físico es una línea recta cuyo final tiene un dispositivo terminador que finaliza y balancea al medio (ver figura 1.3).

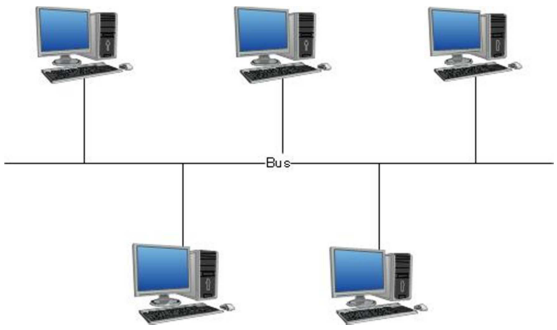


Figura 1. 3 Topología de Bus

Estrella: Una de las topologías más usadas, en ella todos los dispositivos se conectan a un solo punto, normalmente un dispositivo concentrador o distribuidor (ver figura 1.4).

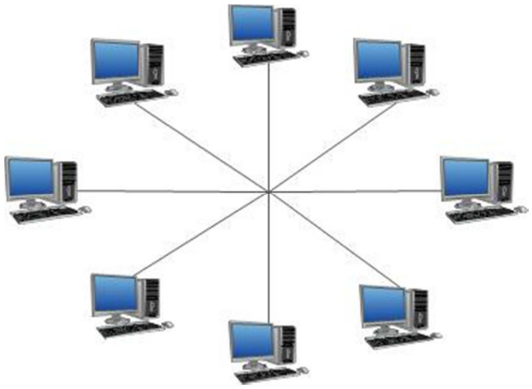


Figura 1. 4 Topología en Estrella

Anillo: Los equipos están conectados en un anillo, sin principio o fin, dado que los equipos están conectados de forma circular; existen dos topologías de este tipo: la de anillo simple, con un solo sentido de transmisión, y la de doble anillo, en donde la información puede viajar en ambos sentidos (ver figura 1.5).

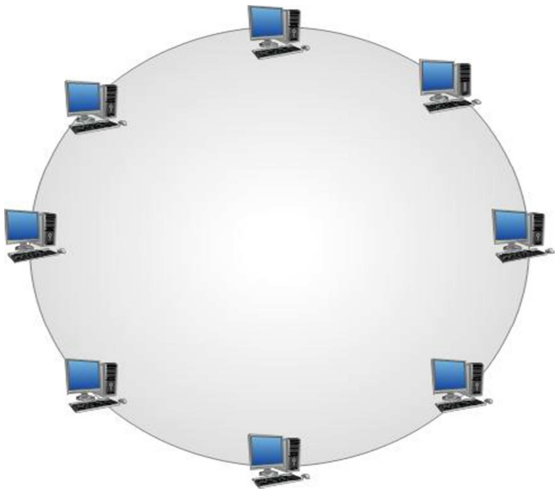


Figura 1. 5 Topología en Anillo

Jerárquica: Se parte de un punto troncal, de donde se derivan más puntos, la conexión de los dispositivos se pude visualizar como un árbol invertido (ver figura 1.6).

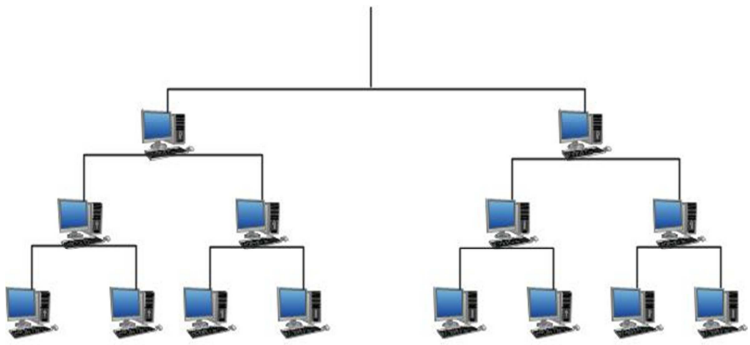


Figura 1. 6 Topología Jerárquica

Malla: Esta topología es más eficiente que las ya mencionadas, solo que es más costosa, ya que en ella cada nodo tiene interconexión con el resto de los nodos de tal forma que si un enlace se cae se tienen más conexiones activas [6] (ver figura 1.7).

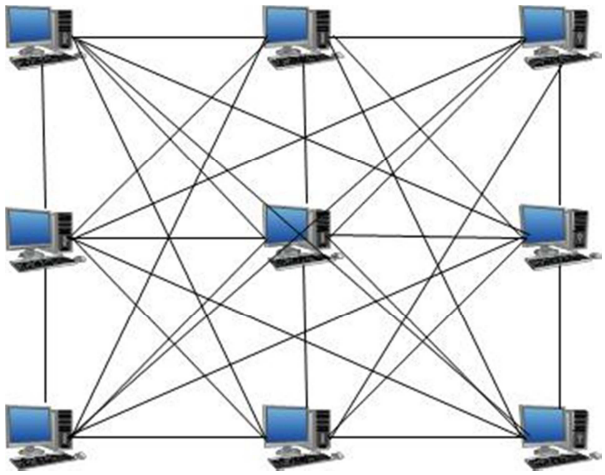


Figura 1. 7 Topología de Malla

1.2.4 Topologías lógicas

Una topología lógica de red se refiere a la manera en que los equipos en dicha red interpretan la comunicación a través del medio, aunque físicamente esta comunicación pueda ser diferente. Las topologías lógicas más comunes son de difusión y de transmisión de *tokens*.

Difusión: En esta variante cada equipo dirige sus datos a una interfaz de red en particular, a una dirección de multidifusión o a una dirección de difusión en el medio de red. No existe un orden que los equipos deban seguir para utilizar la red, en este caso el primero que llegue es el primero que puede tener acceso; Ethernet funciona de esta manera (Ver figura 1.8).

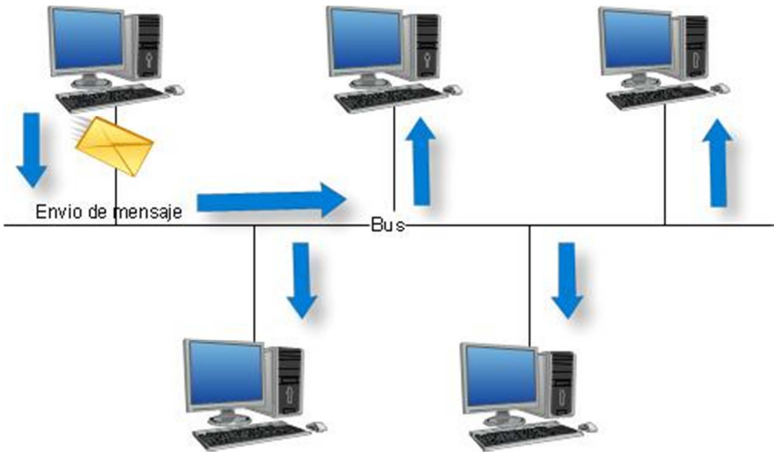


Figura 1. 8 Topología de difusión

Transmisión de tokens: En esta topología se obtiene el control de acceso a la red mediante un token o elemento, el cual representa el turno para transmitir, y se envía secuencialmente a cada equipo de la red. Cuando un equipo recibe el token, dicho equipo queda habilitado para enviar datos por la red. Si el equipo no tiene datos que enviar, el token se pasa al siguiente nodo, y este proceso se repite cíclicamente. Token ring y FDDI son dos ejemplos de tipos de redes que utilizan la transmisión de tokens sobre topologías físicas en anillo (Ver figura 1.9).

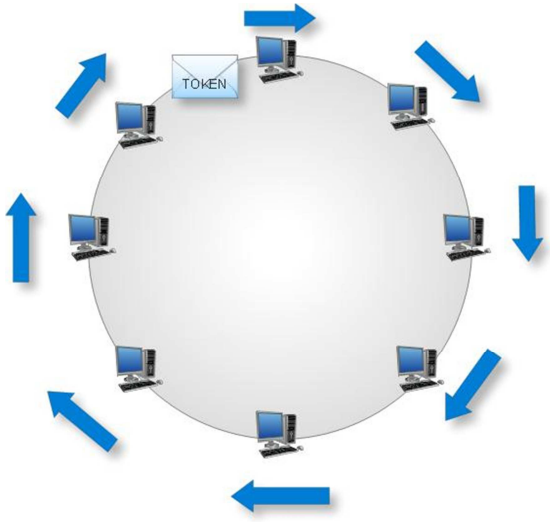


Figura 1. 9 Transmisión de Token

1.3 Redes Virtuales de Área Local (VLAN)

Una red LAN esta interconectada de forma física, permitiendo que los dispositivos ubicados en el mismo segmento de la red se comuniquen entre sí, algunos de los problemas asociados a este tipo de redes es que no existe confidencialidad entre los equipos de la LAN, en este caso el ancho de banda no se aprovecha de una forma eficiente ya que todos los equipos se encuentran en un solo dominio de colisión. En una red VLAN ocurre lo contrario, ya que la interconexión es lógica y no física, de tal forma que las VLAN pueden interconectar equipos que se encuentren en distintos segmentos de red y lograr una comunicación (ver figura 1.10); esto se logra utilizado software especial de administración.

Una VLAN es considerada un dominio de difusión, esto quiere decir que si un equipo transmite información que está asignado a una VLAN, la información será recibida por aquellos equipos que pertenezcan a la misma, esto se logra a través de la asociación de puertos de un conmutador. Para entender cómo es que las VLAN funcionan es necesario saber cómo es que se construyen estas.

La principal diferencia entre una interconexión lógica y una física es que los usuarios que pertenecen a la VLAN pueden ser distribuidos a través de la LAN, incluso si se encuentran en diferentes conmutadores, esto es, en distintos segmentos de red. Esto permite reubicar físicamente a los usuarios dentro de la misma LAN sin perder el grupo de trabajo al que pertenecen**.** Así, una VLAN es un dominio de difusión que aísla de forma lógica pequeños segmentos de LAN, administrando así de una forma más eficiente el ancho de banda. Los equipos de cómputo que pertenezcan a la VLAN se pueden comunicar con otras VLAN utilizando dispositivos llamado enrutadores (*routers*). Una VLAN se puede crear tomando la dirección MAC, de puerto o de IP, de cada uno de los dispositivos que van a formar parte de la red virtual. La figura 1.8 muestra la creación tres VLAN, la VLAN 5 está definida en los puertos 1, 4, 7 del conmutador, la VLAN 10 en los puertos 2, 5, 8 y la VLAN 15 en los puertos 3, 6, 9 en donde cada uno de ellas es un dominio de difusión independiente lo cual conlleva una mejor administración de la red y un mejor funcionamiento. Las VLAN funcionan sobre el estándar Ethernet ya que estas utilizan el encapsulamiento 802.1q, este añade información adicional a la trama de Ethernet.

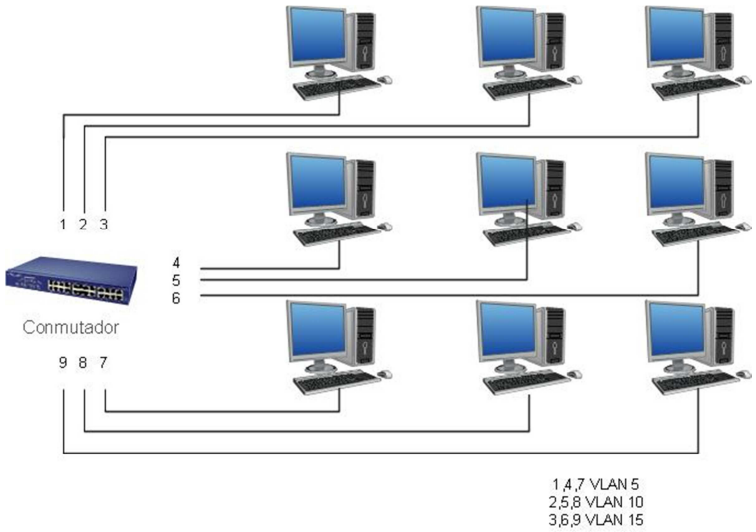


Figura 1. 10 VLAN

Las ventajas que ofrecen las VLAN son las siguientes:

- Permiten movilidad física de los usuarios dentro de los grupos de trabajo.
- Dominios lógicos: Los grupos de trabajo pueden definirse a través de uno o varios segmentos físicos,
- Mejor administración del ancho de banda: Las redes virtuales pueden restringir dominios de difusión a los dominios lógicos donde han sido generados.
- Permiten una mayor seguridad: Los accesos desde y hacia los dominios lógicos, pueden ser restringidos, en función de las necesidades específicas de cada red, proporcionando un alto grado de seguridad.
- Costos: Las VLAN pueden ser implementadas bajo conmutadores que tengan soporte para estas, lo cual permite que se usen los mismos equipos y no se tenga que invertir más en cuestiones de cableado o de equipos extra para hacer más eficiente la red, solo se requiere de configuración vía software.
- Conectividad: Es posible expandir las VLAN a través de conmutadores y enrutadores, aunque estos estén situados geográficamente en lugares distintos

Otros beneficios que las VLAN ofrecen es que existen grupos de trabajo virtuales, permiten reducir los costos administrativos cuando la red requiere llevar acabo cambios o traslados de los usuarios, en cuanto a los dominios de difusión permite tener un mejor control de la red.

1.3.1 Trama Ethernet 802.3

Para transmitir información sobre el estándar Ethernet, los dispositivos que se implementan en la red construyen una trama, es decir un paquete de información de control y datos el cual viaja a través de una red de cómputo. Un mensaje pequeño será enviado en una sola trama de Ethernet, pero cuando se requiere mandar una gran cantidad de información esta es dividida en múltiples tramas. La figura 1.11 muestra el formato básico de la trama Ethernet IEEE 802.3

PREAMBULO	SDF	DESTINO	ORIGEN	LONGITUD TIPO	DATOS RELLENO	FCS
7	1	6	6	2	46 - 1500	4

Figura 1. 11 Estructura de trama Ethernet IEEE 802.3

Campos de la trama Ethernet.

Preámbulo (7 octetos): Este campo es un patrón alternativo de 1 y 0 y se utiliza para la sincronización de tiempo, esto para las primeras versiones de Ethernet, en cuanto a las versiones más actuales ya son síncronas, por lo que la información de temporización es redundante, pero esta se mantiene por razones de compatibilidad. El patrón binario es 10101010

SDF (Delimitador de trama de inicio, 1 octeto): este campo marca el final de la información de temporización, el patrón binario es 10101011

Destino (6 octetos) Dirección MAC destino: Este campo contiene la dirección MAC del destinatario.

Origen (6 octetos) Dirección MAC Origen: Contiene la dirección MAC de origen

Longitud Tipo (2 octetos): En este caso si el valor es menor que 1536 en decimal, indica longitud. La interpretación de longitud se emplea donde la capa LLC proporciona la identificación del protocolo.

- Tipo (Ethernet): El tipo especifica el protocolo de capa superior que recibe al dato después de que se haya completado el procesamiento.
- Longitud (IEEE 802.3) La longitud indica el número de bytes de datos que siguen a este campo. Si el valor es igual o superior a 1536 decimal, indica tipo, y el contenido del campo de datos lo decodifica el protocolo indicado

Datos y relleno (46 a 1500 octetos): este campo puede ser de cualquier longitud que no provoque que la trama exceda su tamaño máximo. La unidad máxima de transferencia es decir la MTU para Ethernet es de 1500 octetos, por lo que los datos no deben exceder este tamaño
.
FCS (4 octetos) Secuencia de verificación de datos: Esta secuencia contiene un valor CRC de 4 bytes creado por el dispositivo emisor, y el dispositivo receptor lo vuelve a calcular para comprobar daños en las tramas.

En la versión DIX de Ethernet que se desarrolló antes de la adopción de la versión IEEE 802.3 de Ethernet, los campos preámbulo de delimitador de trama de inicio se combinaron en un solo campo el campo longitud/tipo fue enumerado solo como longitud en las primeras versiones del IEEE y solo como tipo en la versión DIX.

1.3.2 Estándar IEEE 802.1Q

Para el año 1996 se creó un subcomité de IEEE 802.1, de donde surgió el estándar IEEE 802.1Q con la finalidad de regular las VLAN. Este estándar es un mecanismo que permite múltiples redes puenteadas que puedan compartir la misma conexión de red física aislando la información de otra red. El estándar 802.1Q se usa para identificar una VLAN; en él se añaden 4 bytes a cada una de las tramas Ethernet, de forma que cada trama se identifica como un miembro de la red. Los conmutadores tienen la tarea de buscar en los 4 bytes el destino donde debe ser enviada la trama. Un conmutador identifica una trama VLAN por los primeros 2 bytes de la cabecera de la trama, mientras que los siguientes 2 bytes están divididos en 3 partes. La primera parte (3 bits) indica la prioridad de la trama (*Pri*); el bit siguiente, llamado CFI (*Indicador de Formato Canónico*), indica si el orden de los bits es canónico o no; la última parte es de 12 bits, e indica el cliente a quien pertenece la trama (ver figura 1.12).

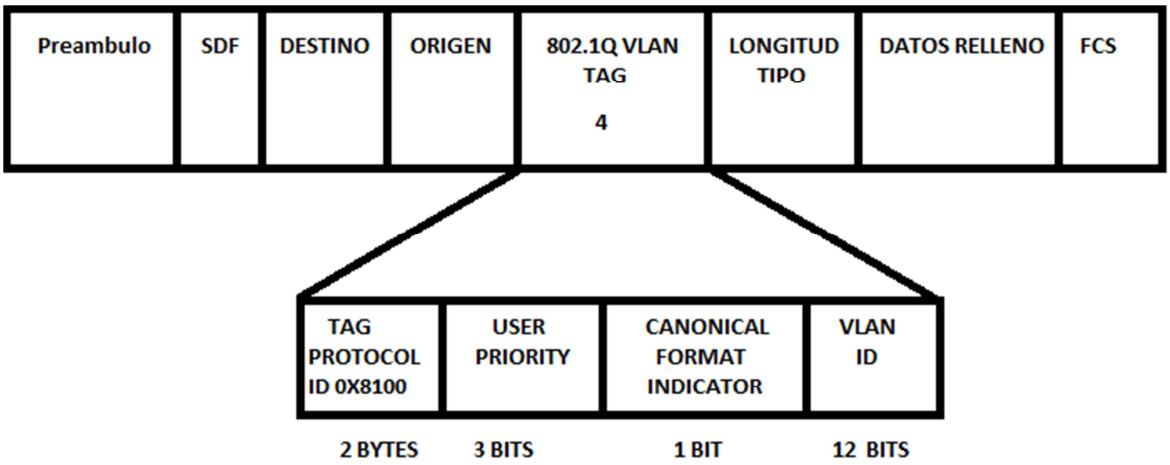


Figura 1. 12 Trama 802.1q

1.3.3 Tipos de VLAN

- VLAN basadas en puerto: Se conocen también como VLAN estáticas. Sólo se define una VLAN para un puerto y se dispone de múltiples VLAN para su asignación a un conmutador. Las VLAN basadas en puerto no reconocen el direccionamiento de nivel 3, todas las VLAN que admiten el tráfico de Capa 3, tal como IP, IPX o AppleTalk, deben estar definidas en la misma red dentro de la misma VLAN.

- VLAN basadas en MAC: Permite tener una configuración dinámica, usando las direcciones MAC que son generadas por los conmutadores.

- VLAN basadas en la capa 3: Están basadas en el protocolo IP, por lo que los conmutadores no realizan tareas de enrutamiento, simplemente verifican las direcciones IP.

- VLAN por protocolo: Se especifica a la VLAN que tipo de protocolo es el que está disponible para permitir el intercambio de información (IP, IPX).

1.4 Protocolos VLAN

El hacer uso de las VLAN implica el manejo de ciertos protocolos para su funcionamiento, para permitir que las redes creadas trabajen de forma adecuada.

Existen principalmente dos protocolos: el protocolo de árbol de expansión y el protocolo de compartición para VLAN.

1.4.1 Protocolo de Árbol de Expansión (STP)

El protocolo de árbol de expansión fue estandarizado como IEEE 802.1 Para efectos de controlar los enlaces redundantes en redes se utilizan dispositivos llamados puentes (*bridges*); este algoritmo se implementa como software instalado en las estaciones de los conmutadores, y así se pueden detectar múltiples enlaces y también es posible deshabilitarlos.

Este protocolo se presenta en los conmutadores ya que estos dispositivos toman decisiones con base en las direcciones MAC, donde la información se envía a un puerto destino específico. Cuando se envía un mensaje de difusión este mensaje se duplica en cada nodo de la red, de tal forma que la trama se duplica de forma indefinida, este protocolo se encarga de mantener la red libre de estos ciclos. Y presenta cuatro estados:

- Bloqueo: Se prohíbe el reenvío de tramas, y no se actualizan las tablas de direcciones MAC.

- Escucha: No proporciona reenvío de tramas o de direcciones, permite al puente estar dentro del algoritmo.

- Aprendizaje: Este estado permite al puente construir tablas de direcciones MAC, y las tramas son descartadas.

- Envío: El estado permite tres funciones al puente; reenvío de las tramas, construcción de las tablas MAC y participación en el algoritmo [7].

1.4.2 Protocolo de Compartición VLAN (VTP)

Este protocolo permite reducir la administración de una red conmutada, cuando se configura una nueva VLAN y se está usando un servidor VTP la VLAN es distribuida en todos los conmutadores que estén dentro del dominio, esto permite que no se necesita configurar la misma VLAN en el resto de los conmutadores. Este protocolo es propietario de CISCO y esta implementado en la mayor parte de los conmutadores CISCO Catalyst.

Existen distintos modo para hacer uso del protocolo:

- Modo servidor: puede crear, modificar y eliminar VLAN y especificar otros parámetros de configuración. VTP servidor es el modo por defecto.
- Modo cliente: clientes se comportan de la misma manera que los servidores de VTP, pero no se puede crear, cambiar o eliminar VLAN en un cliente VTP.
- Modo Transparente: Solo crea, elimina y modifica VLAN, pero ni comparte su información con el resto de los conmutadores [8].

Existe una función de este protocolo que es conocida como recorte VTP y permite determinar que redes VLAN tienen miembros en un conmutador, eliminando el tráfico de difusión que no es necesario en otros conmutadores.

1.5 Tecnologías de conmutación

Existen dos tecnologías que permiten disminuir la congestión y el tráfico de red, así como también incrementar el ancho de banda, conocidas como conmutación y puenteo en referencia al uso de puentes y conmutadores, los cuales trabajan en la capa dos del modelo OSI. Su funcionamiento está basado en el reenvío de tramas basadas en las direcciones MAC para poder efectuar la operación de conmutación. Cuando la dirección MAC es desconocida para el dispositivo, este hace que la trama inunde toda la red y pueda así alcanzar el destino deseado.

Los puentes son dispositivos de capa dos, diseñados para crear dos o más segmentos LAN, siendo cada uno un dominio de colisión. Su finalidad es filtrar el tráfico, permitiendo la conexión con otros segmentos de la misma LAN. Los puentes solo tienen dos puertos, lo cual permite dividir el dominio de colisión en dos partes. El funcionamiento de estos dispositivos está basado en direcciones MAC, por lo cual no afecta la lógica del direccionamiento de la capa tres. Estos dispositivos construyen tablas con las direcciones MAC que están localizadas en un segmento de red y en otras redes, lo que permite filtrar la información y entregarla de forma selectiva.

Los conmutadores trabajan en capa dos, siendo equivalentes a un puente multipuerto, estos dispositivos utilizan micro segmentación para reducir el número de colisiones y permiten aumentar el ancho de banda, permitiendo también

comunicaciones tipo dúplex. Estos dispositivos permiten interconectar dos o más segmentos de red, trabajando con direcciones MAC.

Con base en los avances tecnológicos se han construido puentes y conmutadores con capacidades mayores, lo cual lleva a un caso en donde se tiene un nodo en cada puerto de un puente, lo que permite reducir los dominios de colisión a tal punto que dichas colisiones son mínimas. Finalmente se tiene un dispositivo que funciona como un puente pero con varios puertos y realiza exactamente la función mencionada anteriormente; el conmutador separa en segmentos físicos llamados microsegmentos.

Estos microsegmentos facilitan la creación de un segmento dedicado, que permite que el ancho de banda sea el mismo para cada usuario de la red, caso contrario a lo que ocurre con el puente, donde el ancho de banda se divide entre los usuarios de la red. En el conmutador cada usuario accede al ancho de banda completo; la microsegmentación reduce las colisiones y permite que cada estación conectada a la red tenga mayor capacidad para establecer comunicación.

Se tiene la topología de la figura 1.13 y se desea enviar información del equipo A al C; cada vez que llega información al conmutador, este actualiza su tabla con la direcciones MAC, en un inicio la tabla se encuentra vacía por tal motivo el conmutador inunda la red enviando mensajes por el resto de los puertos como muestra la figura 1.14. Cuando el equipo a quien va dirigida la información en este caso es el C le llega este mensaje, este responde al equipo que desea tener comunicación, es decir el equipo A, y envía un mensaje como muestra la figura 1.15.

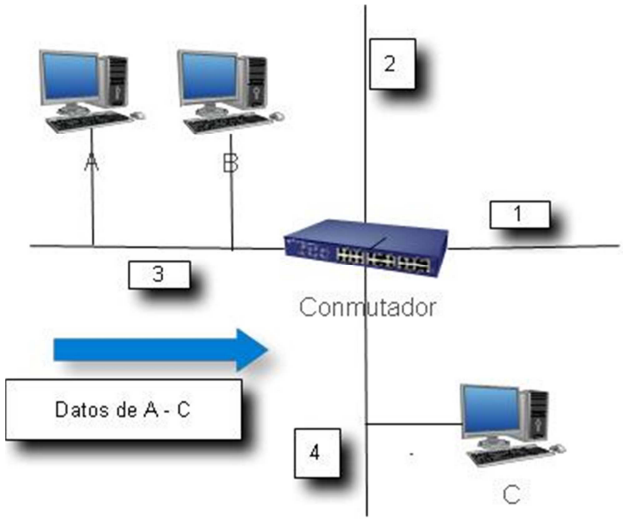


Figura 1. 13 Datos de A – C

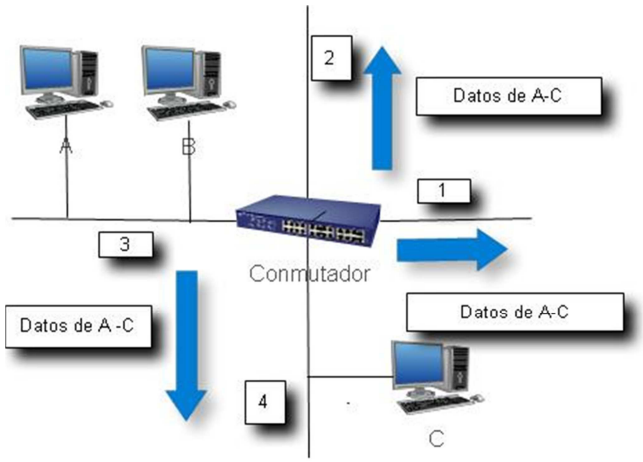


Figura 1. 14 Inundación de datos a los puertos del conmutador

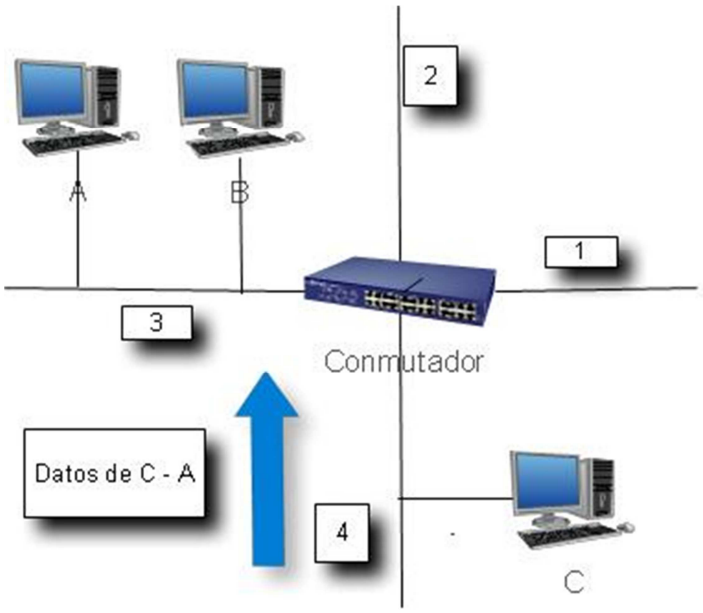


Figura 1. 15 Respuesta al mensaje de inundación

1.6 Tipos de conmutación

- Conmutación de almacenamiento y envío: El conmutador lee la información de toda la trama de datos, con la finalidad de detectar errores, verifica el destino y posteriormente procede a transmitir. Si encuentra un error cuando realiza la lectura de la trama la descarta. La latencia que presenta esta conmutación no representa un problema a considerar.

- Conmutación por método de corte: La información llega al conmutador, y a medida que el tráfico fluye, lee desde el inicio la trama hasta obtener la dirección MAC y corta a través de dicha dirección sin leer el resto del contenido; por obvias razones este método disminuye la latencia de la transmisión, pero no cuenta con detección de errores.

- Conmutación de liberación de fragmentos: Se basa en filtrar los fragmentos que han colisionado antes de reiniciar el envío. Esta conmutación espera antes de enviar, hasta determinar que el paquete recibido no es un fragmento de colisión.

1.7 Sistema Operativo LINUX

LINUX es un sistema operativo compatible con Unix. Una de sus características más importantes y que lo hacen diferente a otros sistemas es que su uso es libre, esto es, que no se tiene que pagar algún tipo de licencia para su utilización, a menos que se requiera soporte técnico. El sistema está formado por un núcleo (kernel) y una gran cantidad de programas y bibliotecas que permiten su funcionamiento. Este sistema se derivó del proyecto GNU, por tal motivo es correcto referirse al sistema como GNU/LINUX.

LINUX ha sido programado y depurado por una gran cantidad de programadores alrededor del mundo. El núcleo del sistema está en continuo desarrollo bajo la coordinación de Linus Torvalds, creador del proyecto. La calidad de los programas y aplicaciones crece día a día, siendo distribuidos bajo la licencia GPL, donde el código fuente esta accesible y puede tener alguna modificación pero siempre bajo esta licencia. LINUX es un sistema de distribución libre, en el que los archivos y programas requeridos para su funcionamiento se encuentran en una gran cantidad de servidores en Internet. Sin embargo, buscar los archivos y programas necesarios, instalarlos y después configurarlos es una tarea complicada; debido a esto existen las distribuciones de LINUX.

Una distribución es una recopilación de ciertos programas y archivos que están organizados y preparados para llevar acabo su instalación. Existe una gran cantidad de distribuciones y la mayor parte de ellas se pueden encontrar en servidores de Internet sin costo alguno. Por mencionar algunas de ellas esta Fedora, Open Suse, Ubuntu, CentOS, Mandriva, etc. Con la finalidad de escoger una distribución de LINUX para desarrollar el proyecto se determinó hacer uso de las dos más populares. Después de una serie de pruebas sobre cada una de ellas se llegó a las conclusiones contenidas en la Tabla 1.1 [9].

Aunque ambas distribuciones tiene la capacidad de poder instalar una gran cantidad de aplicaciones para implementaciones de red, estas al querer usarlas en una distinta puede requerir de una configuración adicional, es un hecho que cuando se desea instalar una aplicación siempre existen archivos específicos para cada grupo de distribuciones, es por ello que se tiene que tomar estas precauciones ya que si el archivo es incorrecto el sistema operativo indicara que el archivo que desea instalar no es el adecuado.

CARÁCTERÍSTICAS	FEDORA	OPEN SUSE
INSTALADOR	Instalación práctica, no requiere mucha asistencia del usuario.	Instalador muy flexible, permite configurar muchos parámetros de forma sencilla.
ESCRITORIO	GNOME	KDE
CONFIGURACIÓN	Para usuarios con más experiencia en sistemas LINUX, la mayor parte de configuración requiere de comandos, aunque no en todos los casos. Su instalador gráfico KPackagekit, no es tan práctico. En cuanto al instalador yum es muy práctico bajo línea de comandos, pero requiere de mayor práctica.	No requiere conocimientos amplios, es más sencillo de usar, ya que su herramienta yast permite hacer la mayor parte de configuración de forma gráfica. El instalador Zypper es muy práctico bajo la línea de comandos pero también requiere de un mayor conocimiento sobre los paquetes.
DETECCIÓN DE HARDWARE	Presenta problemas con las tarjetas de video ATI, en cuanto a equipos de cómputo del laboratorio no se tuvo problemas.	Presenta problemas con las tarjetas de video ATI, en cuanto a equipos de cómputo del laboratorio no se tuvo ningún problema.
INFORMACIÓN	Existe mucha información en Internet sobre esta distribución.	Hay más información sobre esta distribución comparada con fedora.
ESTABILIDAD	Se detectaron algunos problemas al actualizar el kernel, ya que indicaba paquetes rotos.	No se presentaron problemas con esta distribución.
TARJETAS DE RED	Cuenta con una tarjeta de red, al crear VLAN sobre esta tarjeta no presento dificultades	Se trabajó con este equipo debido a que este contaba con tres tarjetas de red, y permitía una mejor manipulación de la herramienta desarrollada. No presento problemas en la configuración de las VLAN
RED	Permite configurar el equipo como un enrutador, modificar archivo etc/sysctl.conf. Net.ipv4.ip_forward = 1, Posteriormente el enrutamiento se tiene que hacer manualmente	Permite configurar el equipo como un enrutador hay que modificar el archivo el cual está en etc/sysconfig/sysctl.conf. Net.ipv4.ip_forward = 1

Tabla 1.1 Comparación entre las distribuciones Open Suse y Fedora.

1.8 Entornos de desarrollo

1.8.1 Anjuta

Anjuta es un entorno para el escritorio GNOME, que incluye una gran cantidad de herramientas, entre ellas un asistente para aplicaciones, un depurador interactivo, editor de código fuente, control de versiones, y un diseñador de interfaces gráficas. Permite programar en diferentes lenguajes tales como: C++, java, python, etc. Anjuta fue diseñado para ser fácil de usar, pero es lo suficientemente potente como para satisfacer todas las necesidades de programación, Se han añadido muchas características nuevas en el transcurso de su desarrollo, entre ellas se puede mencionar glade que permite el desarrollo de las interfaces gráficas de usuario[10].

La dificultad que se encontró fue que glade no está integrado dentro de anjuta, se tiene que ejecutar cada programa independiente. Presento muchas dificultades para el uso de la distribución Fedora 12 ya que requiere de muchas dependencias para su buen funcionamiento, al hacer pruebas sencillas si no se tenía la biblioteca adecuada mandaba mensajes de error con el nombre la biblioteca requerida, una vez instalada, esta otra requería de una más**,** esto ocurría con la mayor parte de los distintos proyectos que puede generar este entorno. Otro punto importante es que al actualizar todas las bibliotecas anjuta tenía que funcionar de forma correcta ya no requería de más bibliotecas pero se complicó ya que no desplegaba la interfaz gráfica [10].

1.8.2 Monodevelop

Monodevelop es un entorno diseñado específicamente para C# y otros lenguajes .NET; este entorno permite generar fácilmente aplicaciones de escritorio, www ASP .NET en LINUX, Windows, y Mac OS desde su versión superior a la 2.2. Monodevelop permite que las aplicaciones hechas en Visual Studio puedan ejecutarse en LINUX, ya que mantiene una base de código único para las distintas plataformas. El entorno incluye manejo de clases y asistencia, permite completar código y tiene un diseñador de interfaces gráficas [11]. Su interfaz de usuario es muy amigable ya que la creación de los formularios es muy práctica para el desarrollo de una interfaz gráfica, Este entorno de desarrollo es también una muy buena opción, pero se dejó de lado debido a que el lenguaje que se deseaba usar es C++.

1.8.3 QT Creator

QT es un entorno de desarrollo multiplataforma, que proporciona un editor de código para C++, un diseñador de interfaces de usuario, facilidades para construcción de proyectos y herramientas de gestión. QT está escrito en C++; sin embargo es posible utilizar QT con otros lenguajes de programación a través de *bindings* (adaptaciones de bibliotecas para su uso en un lenguaje distinto a aquel en el que han sido escritas). En QT existen bindings para lenguajes como C#, PHP, Python, Ruby, etc. QT se utiliza en una amplia variedad de dispositivos, tales como: computadoras de escritorio, teléfonos celulares, lectores electrónicos, computadoras de automóvil, etc. Algunas aplicaciones conocidas que utilizan QT son: el entorno de escritorio KDE y toda su suite de aplicaciones (mensajero instantáneo Kopete, software oficina KOffice, y el navegador web Konqueror, Skype, Google Maps, y la herramienta de modelado Maya. QT está disponible bajo 3 diferentes licencias:

- GPL (Aplicación de código abierto, los cambios realizados al código fuente de Qt deben ser compartidos con la comunidad.
- LGPL Es posible crear aplicaciones de código cerrado, los cambios realizados al código fuente de QT deben ser compartidos con la comunidad.
- Comercial: Es posible crear aplicaciones de código cerrado, los cambios realizados al código fuente de QT pueden mantenerse cerrados [12].

Este entorno de desarrollo facilita la programación ya que no requiere de muchas dependencias, es necesario instalar las bibliotecas de QT que en el caso de Open Suse es libqt4-devel y por supuesto el entorno de desarrollo que es QT Creator. Para el caso de instalar en fedora 12 las bibliotecas son qt-devel qt-config. En ambos casos se requiere del compilador de C++ el cual se puede encontrar en el paquete build-essential.

El manejo de este entorno es mucho más amigable y practico, no se presentó ningún problema para desarrollar la interfaz, ya que este fue probado en tres distribuciones, Fedora, Open Suse y Ubuntu, en todas funciono correctamente, la compilación de la interfaz gráfica en estas distribuciones no tuvo ningún problema siempre y cuando tuviera instalados los paquetes necesarios y en cuestión de los escritorio, funciono para KDE y GNOME.

1.8.4 Comparativa de entornos de desarrollo

Se seleccionaron tres entornos de desarrollo para probar su funcionalidad y su capacidad para poder desarrollar la interfaz gráfica. La tabla 1.2 se muestra algunas de las características que presentaron.

CARACTERÍSTICAS	QT CREATOR	ANJUTA	MONODEVELOP
DEPENDENCIAS	Requiere solo de las bibliotecas de QT y el compilador g++	Requiere de muchas bibliotecas además de las de GTK+ para sus distintas aplicaciones	NO requiere muchas dependencias., utiliza bibliotecas GTK+
CONFIGURACIÓN	No necesita parámetros para configuración	Requiere de ciertas características a configurar	Tiene un uso fácil, no requiere configuración extra
LENGUAJES SOPORTADOS	En general C, C++, pero puede soportar otros lenguajes con ayuda de bindings	C, C++, java , python,	C#, .NET
LICENCIA	GPL, LGPL	GPL	GPL
APRENDIZAJE	Es muy práctico y tiene buena documentación sobre su uso	Demasiado confuso, y con muchos errores	Es práctico
DISEÑADOR DE INTERFACES GRÁFICAS	Muy buen diseñador integrado	Existe un diseñador pero es independiente	Diseñador integrado

Tabla 1.2 Entornos de desarrollo

1.8.5 Lenguajes de programación

LINUX dispone de una gran variedad de lenguajes disponibles para desarrollar programas o aplicaciones y que esta pueda ser distribuida de forma libre; entre los más importantes están los siguientes:

- Perl: Es un lenguaje de programación orientada a objetos que está enfocado a principiantes, es decir, a aquellos que tienen conocimientos

básicos en programación. Este lenguaje es muy potente para la manipulación de textos y cadenas.

- Python: Está enfocado para principiantes, permite la programación orientada a objetos y básicamente maneja scripts para determinadas aplicaciones y www.
- PHP: Está enfocado a principiantes y permite la programación orientada a objetos, su principal aplicación es en www, es muy popular para las aplicaciones que requieran bases de datos basadas en red.
- Java: Enfocado a principiantes, tiene un campo muy grande de aplicaciones y en especial la www.
- C: No está enfocado a principiantes, permite la programación de sistemas y distintas aplicaciones, es un lenguaje muy popular.
- C++: No está enfocado a principiantes, programación orientada a objetos, tiene muchas aplicaciones.

Capítulo 2

2.1 Interfaz Gráfica

La interfaz gráfica a desarrollar tiene la finalidad de proporcionar a los usuarios una herramienta para implementar VLAN de una forma sencilla en un sistema operativo LINUX, de tal forma que el administrador de la red no necesita profundizar en el uso de los comandos que requiere este tipo de sistemas. Se propone que la interfaz gráfica tenga las funciones principales existentes bajo línea de comandos, que son:

- Agregar VLAN: Permite agregar la VLAN con un identificador numérico, permitiendo su visualización en una ventana.

- Eliminar VLAN: Las VLAN que se creen podrán ser eliminadas desde la misma interfaz gráfica.

- Cargar módulo 802.1q: Cargar el módulo antes de comenzar el proceso de administración de las VLAN.

- Visualizar Número de tarjetas de Red: La interfaz gráfica permitirá ver cuantas tarjetas tiene instaladas el sistema donde se esté ejecutando la interfaz.

- Configurar las VLAN: Esta parte permitirá enviar los parámetros básicos, tales como dirección ip y máscara de subred a la VLAN.

Debido a que la interfaz gráfica hace uso del comando vconfig se describe a continuación:

Descripción: El comando vconfig permite crear y eliminar VLAN. Las VLAN creadas en LINUX son tarjetas de red virtuales asociadas a una tarjeta de red física.

Las opciones que permite este comando son las siguientes:

- Add [nombre de interfaz] [identificador de VLAN]

  Con esto podemos añadir la VLAN respecto a la tarjeta de red física

- Rem [VLAN]

  Permite eliminar la VLAN.

- Set_flag [VLAN] 0|1

Cuando se pone en uno se reordena la cabecera de Ethernet, la tarjeta de red aparecerá como un dispositivo Ethernet sin VLAN´s.

Cuando está en 0 que es por defecto, los encabezados de Ethernet no cambian el orden.

Esto se puede modificar ya que algunos programas de filtrado de paquetes pueden tener algún problema.

La interfaz gráfica solo manipular las comando para agregar y eliminar, aunque también se configuran pero se hace uso del comando ifconfig.

2.1.1 Requerimientos y limitaciones

Para poder crear las VLAN en LINUX es necesario que el sistema operativo haya levantado el módulo 802.1q, posteriormente tener instalado el paquete **vlan-utils**, que trae el comando **vconfig**, este ya está precargado en la mayor parte de las distribuciones actuales, para verificar que esté instalado es necesario ejecutar el comando **vconfig** en una consola de LINUX.

En cuanto a los requerimientos de la interfaz se necesita lo siguiente

- Contar con elementos de interacción para el usuario, tales como botones, campos para introducir y visualizar información de configuración, etc.
- Capacidad para funcionar en diferentes distribuciones de LINUX, siempre y cuando el kernel tenga soporte para las VLAN.
- El software debe ser de fácil aplicación para el usuario, permitiéndole realizar sus tareas de forma sencilla y concisa.
- Permitir las funciones básicas de la implementación de las VLAN mencionadas anteriormente: añadir, configurar y eliminar.
- Si el equipo en el que está instalado el sistema presenta más de dos tarjetas de red, permitir llevar acabo las funciones básicas en cualquier tarjeta de red y que el usuario pueda seleccionar entre ellas, recordando que no todos los módulos para las tarjetas de red en LINUX permiten el soporte para implementar las VLAN. En el caso de algunas tarjetas de red de determinados fabricantes no existe el módulo para detectar y manejar la tarjeta en LINUX.
- La MTU (Unidad máxima de transferencia) puede presentar conflictos [13].

2.1.2 Herramientas a usar

El diseño de la interfaz requiere de un sistema LINUX con:

- Entorno de programación QT Creator versión 2.0.1.
- Bibliotecas QT 4.7.0
- Paquete VLAN.
- Comandos (vconfig, ifconfig).
- Equipo de cómputo con distribución de LINUX con kernel superior a 2.4.14 y más de 2 tarjetas de red instaladas en el sistema.

Los elementos mencionados anteriormente se requieren debido a que en el caso del entorno de programación y las bibliotecas son las versiones que están en uso actualmente, el paquete vlan es necesario porque permite hacer uso del comando vconfig, en cuestión del kernel tiene que ser superior a esa versión ya que es la que tiene el soporte para el estándar 802.1q. Por último se menciona que requerimos más de dos tarjetas de red para hacer uso de cualquier tarjeta de red del sistema.

2.2 Diseño y Desarrollo de la Interfaz Gráfica

El desarrollo llevara un orden específico, se iniciara con la construcción de las interfaces gráficas y después se programará cada una de ellas. El diagrama de la Figura 2.1 muestra a grandes rasgos de las funciones que lleva acabo el sistema a desarrollar.

La ventana principal incluye los siguientes botones de propósito específico que muestran la funcionalidad de los objetos que se añadirán a la interfaz (ver imagen 2.2):

- Botón de activación del módulo: Al ejecutar este botón el sistema habilita el módulo 802.1q en el sistema operativo LINUX, de modo contrario no se podrán implementar las VLAN. Si bien en la mayoría de las distribuciones ya viene cargado el módulo, algunas requieren hacerlo manualmente.

- Botón tarjetas de red: Indica al usuario cuantas tarjetas de red tiene instaladas en el sistema, mismas que podrá visualizar en la interface para seleccionar entre ellas y crear las VLAN.

- Botón Añadir / eliminar: Este botón ejecuta una nueva ventana o formulario con diversos campos para la configuración y administración.

- Botón salir: Si el usuario desea salir del programa puede dar clic sobre el botón o simplemente cerrar la ventana.

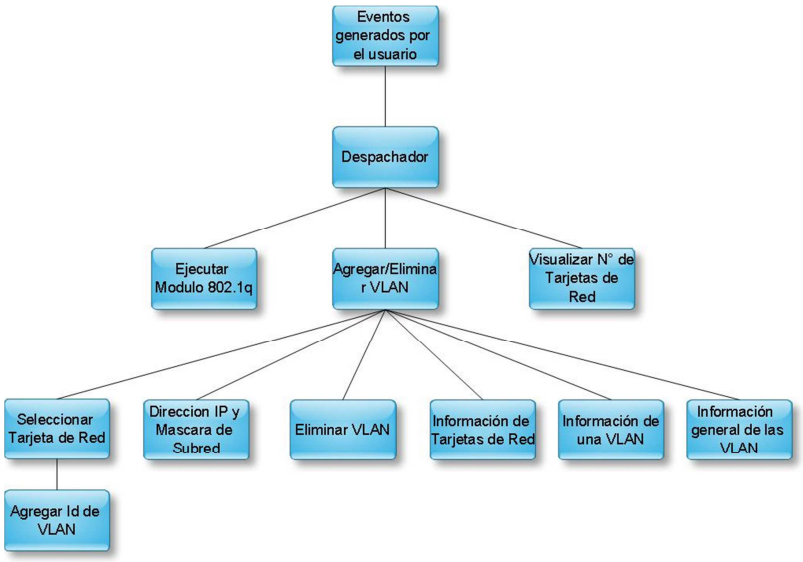


Figura 2. 1 Diagrama de estructura de cajas [14].

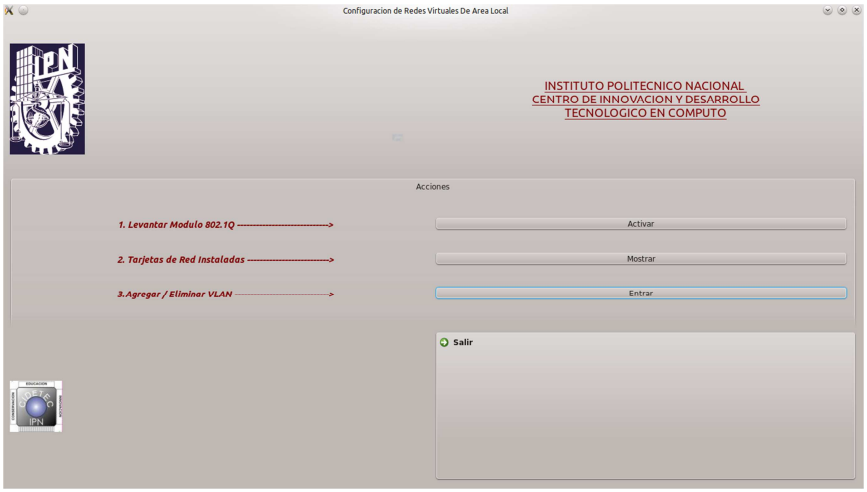


Figura 2. 2 Ventana Principal de la Interface Gráfica

La figura 2.3 muestra la ventana en ejecución al dar clic en el botón de agregar /eliminar VLAN:

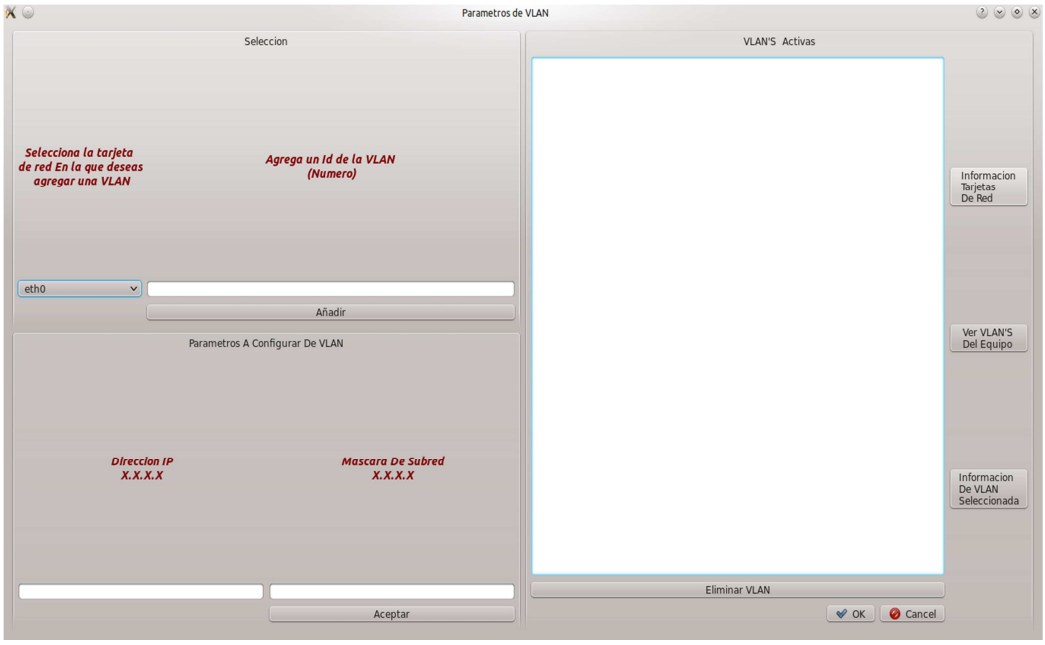


Figura 2. 3 Configuración de parámetros de las VLAN

Esta ventana permite llevar acabo la mayor parte de la configuración. El título **selección** contiene dos parámetros, uno permite escoger la tarjeta de red sobre la cual se requiere crear la VLAN y el otro es para agregar el identificador de la misma. El siguiente título, **parámetros a configurar de vlan**, sirve para introducir datos como la dirección ip y la máscara de subred de un VLAN. En la interface aparece otro título más **vlan activas**, en esta parte se registra cada una de las VLAN que el usuario vaya agregando con la finalidad de que pueda visualizar los procesos que se han ejecutado; en caso de que el usuario desee eliminar alguna VLAN tendrá que seleccionarla y dar clic en **eliminar VLAN**.

Por otra parte también hay la posibilidad de ver la configuración de las tarjetas red, de las VLAN que se han agregado y algunos parámetros de cada VLAN; el botón con título **configuración de tarjetas de red** permite ver la configuración general de las tarjetas de red (ver figura 2.4). El botón de título **ver vlan del equipo** permite ver el archivo de configuración de las VLAN (ver figura 2.5); por último, el botón restante permite ver la configuración específica de la VLAN que el usuario seleccione (ver figura 2.6).

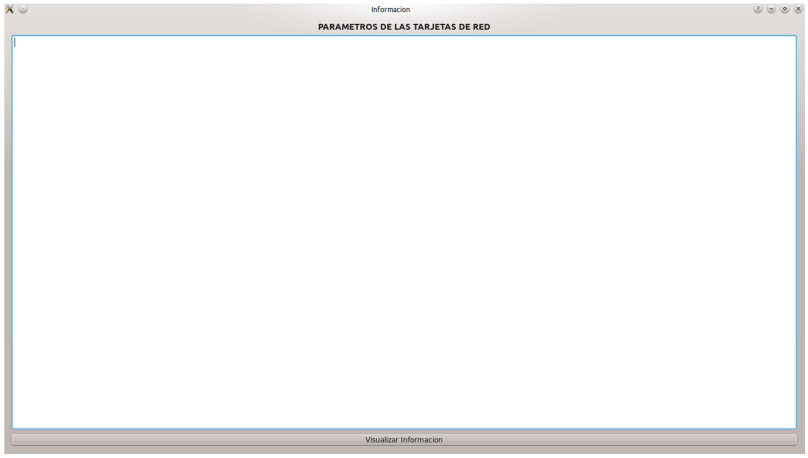


Figura 2. 4 Estado de las Tarjetas de Red

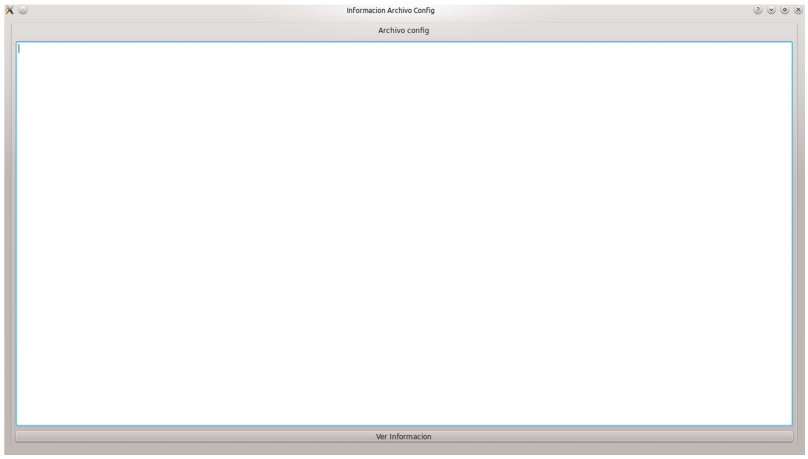


Figura 2. 5 Archivo de configuración de las VLAN

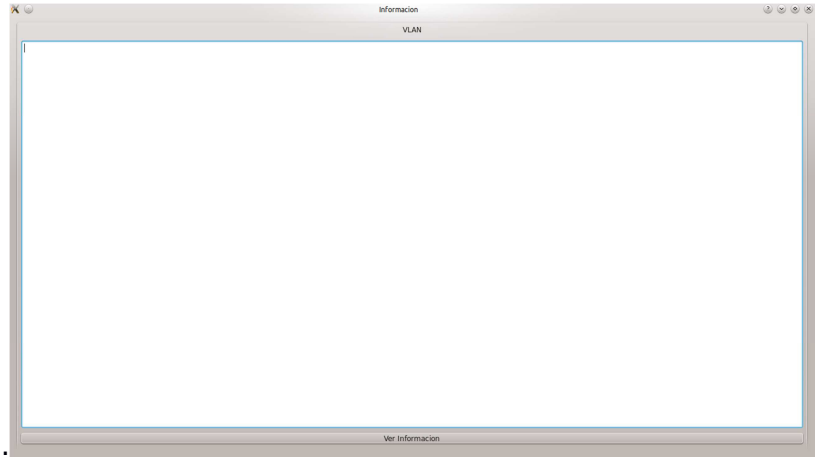


Figura 2. 6 Estado de una VLAN especifica

2.2.1 Programación en QT

QT es una biblioteca multiplataforma que sirve principalmente para desarrollar interfaces gráficas de usuario, utilizando el lenguaje C++ para el desarrollo de los proyectos. El funcionamiento de la interface gráfica es denominado programación dirigida por eventos, esto quiere decir que el programador desarrolla el código a ejecutar en respuesta a los diferentes eventos existentes de la interface; en este caso son llamados **slots**, que pueden ser acciones como dar clic sobre un botón, elegir la opción de un menú, abrir o cerrar ventanas, etc. Las ventanas son clases, y cada uno de los componentes llamados **widgets** también son clases, los eventos conocidos como slots son los métodos de estas ventanas.

En este modelo de programación no existe un control específico sobre la secuencia del programa ya que existen muchas alternativas para que el usuario interaccione con el mismo. Así, el objetivo del programador será asociar a cada evento el comportamiento adecuado [15].

2.2.2 Gestión de archivos en QT Creator

QT Creator gestiona los archivos de configuración del programa de la siguiente forma:

La raíz del programa es el nombre del proyecto en este caso se le llamo interfaz1, de ahí se deriva un archivo que es el que contiene el resto de las subcarpetas con una extensión .pro (interfaz1.pro); si se desea abrir el proyecto completo este es el archivo que hay que ejecutar. Posteriormente se encuentran tres capetas más, mismas que se describen a continuación:

Cada proyecto que se genera en QT crear de manera automática los archivos necesarios solo para el primer formulario y estos están agrupados en carpetas, esto permite un mejor control sobre los archivos para poner código fuente de una forma adecuada.

Estas carpetas hacen referencia al desarrollo de la interfaz.

Sources: En ella se encuentran los archivos de implementación, con extensión .cpp, incluyendo al archivo main; en ellos se escribe el código para los métodos y funciones.

Para la interfaz se usó cinco archivos con los siguientes nombres:

- main.cpp
- widget.cpp
- infovlan.cpp
- información.cpp
- dialog.cpp
- allinfovlan.cpp

En estos archivos se implementaron los métodos para la ejecución de vconfig y la lectura de los archivos del sistema

Forms: Contiene todos los formularios que se utilizan en el proyecto, estos archivos tienen extensión .ui (*user interface*, interface de usuario). En el desarrollo se tiene los siguientes archivos.

- widget.ui
- infovlan.ui
- información.ui
- dialog.ui
- allinfovlan.ui

Estas son la interfaces que se desarrollaron

Headers: Contiene los archivos con extensión .h. Como el lenguaje de programación es C++, estos son los llamados archivos de cabecera, que contienen las declaraciones de las constantes, variables y funciones de las que consta el modulo, así como de las clases adicionales que genere el programador.

Aquí se tiene los siguientes archivos

- widget.h
- infovlan.h
- información.h
- dialog.h
- allinfovlan.h

En ellas se declararon las clases y las variables que en su mayor parte son de tipo string.

La figura 2.7 muestra la creación de un nuevo proyecto y la forma en que se agrupan los archivos.

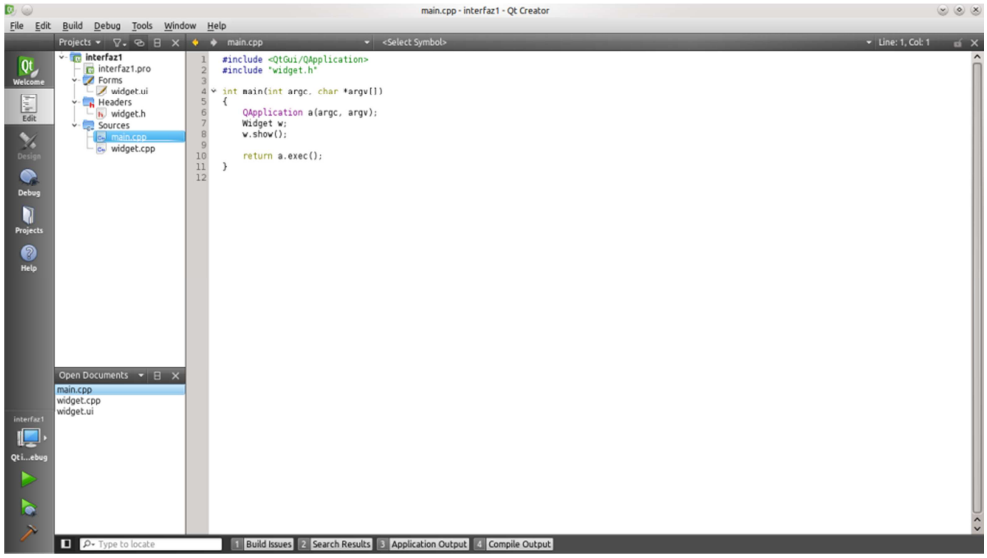


Figura 2. 7 Gestión de un Proyecto en QT

2.2.3 Clase widget

Esta es la clase principal y contiene 3 archivos:

- Widget.ui: Es el diseño principal de la interface del usuario; la Figura 2.8 muestra las herramientas necesarias para el diseño de dicha interfaz.

-

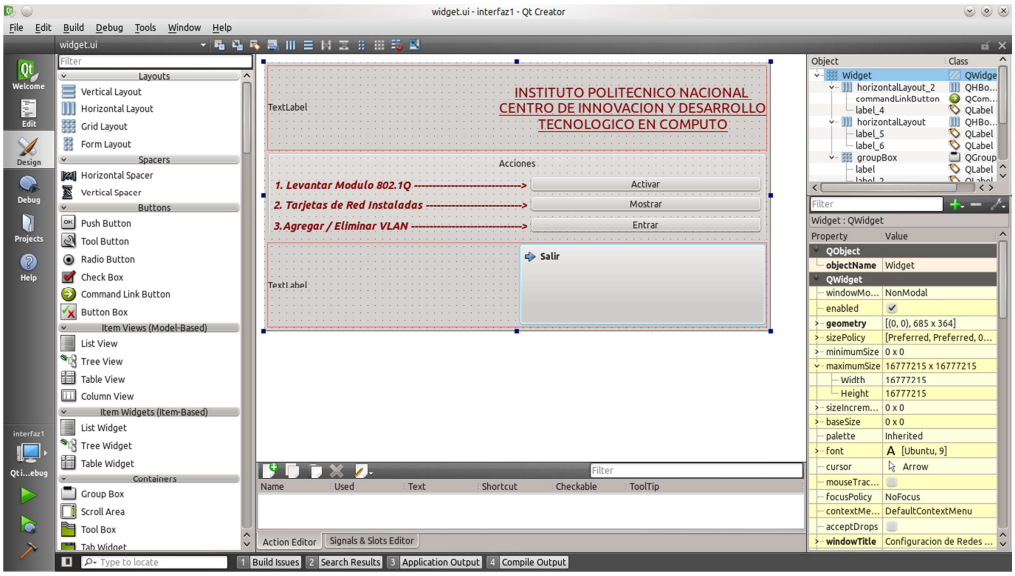


Figura 2. 8 Diseño de la ventana principal en QT

- Widget.h: Es el archivo de cabecera, donde se declara una clase llamada **Widget**, conteniendo la definición de los elementos del formulario que se añadieron a la interface; estos slots se declaran como privados.

- Widget.cpp: Este archivo contiene los métodos de la clase Widget. Incluye el método usado para enviar datos tal como si se enviaran desde la consola línea de comando; esto se logra haciendo uso de la subrutina **system**, ya que la configuración de las VLAN se hace por medio de la consola de LINUX, como se mencionó previamente.

2.2.3.1 Proceso

Cuando se inicia el programa inmediatamente se pide la ejecución de dos comandos que se encuentran dentro del constructor, para que el programa quede precargado con la información del número de tarjetas de red que tiene instaladas el sistema. Esto se logra ejecutando el comando **ifconfig –a**, al añadir –a se lee en general todas las tarjetas aunque estén deshabilitadas y se crea un archivo con la información haciendo uso del operador **>**, que en LINUX permite crear archivos de texto; el archivo se denomina InfTarjetas.txt. Una vez teniendo este archivo se hace uso de otro comando llamado **grep,** cuya función es realizar una búsqueda sobre archivos y encontrar líneas que concuerden con un patrón especifico, por omisión si se ejecuta sin parámetros imprime dichas líneas. Al ejecutar **grep -c** más el patrón deseado se obtiene el número de líneas que contienen dicho patrón.

Verificando el archivo de configuración de las tarjetas de red, se ve que por omisión LINUX pone etiquetas de tipo **eth+subíndice** en las tarjetas ethernet. El subíndice comienza en cero si existe una sola tarjeta de red, y conforme se van añadiendo otras tarjetas se va incrementando. Haciendo pruebas con los archivos, cada tarjeta de red ocupa un determinado espacio en el archivo de texto, de manera independiente; por lo tanto, es posible leer el archivo de texto con el comando **grep** y contar cuantas tarjetas tiene instaladas el sistema. Este dato se asigna a una variable para usarlo en el programa y posteriormente visualizarlo en el objeto llamado **comboBox** que contendrá las tarjetas de red del sistema.

Si el usuario activa el botón de levantar el módulo se envía a la consola el comando **modprobe 8021q**, haciendo uso de **system**; es necesario mencionar que cuando se utiliza el comando **vconfig** esto se ejecuta automáticamente, pero como en ocasiones no se conoce el tipo de distribución LINUX que se está usando es necesario que se pueda cargar antes.

2.2.4 Clase dialog

La siguiente ventana aparece al activar el botón **agregar/eliminar VLAN**, y contiene los archivos siguientes:

dialog.ui: La figura 2.9 muestra el diseño de la interfaz, con todos los elementos necesarios para realizar las operaciones básicas de configuración de las VLAN.

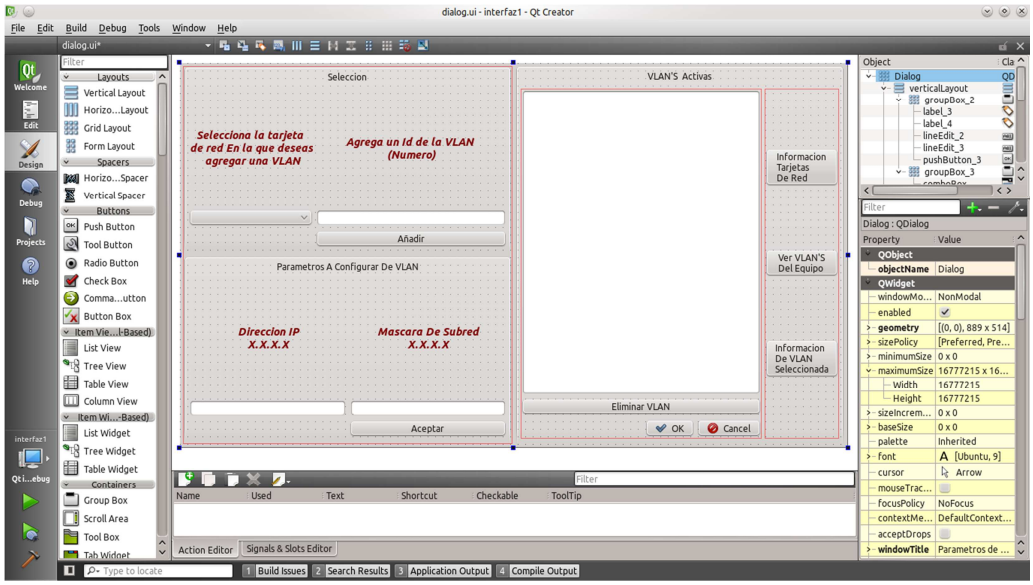


Figura 2. 9 Diseño de la ventana de parámetros

dialog.h: Declara una clase llamada dialog con todos los elementos o slots privados, así como la declaración de las variables para captura de los datos que el usuario requiere ingresar en el sistema.

dialog.cpp: Incluye los métodos para configurar las VLAN.

2.2.4.1 Proceso

Este formulario es el más completo, ya que aquí se llevan a cabo todas las operaciones. Inicia con el primer objeto llamado **GroupBox** que contiene los siguientes objetos:

- ComboBox
- LineEdit
- PushButton
- 2 labels

El ComboBox es cargado con la información de las tarjetas de red por medio de una función en el programa, llamada **asignarnum** que recibe este parámetro a partir de la lectura del archivo Ntarjetas.txt. Para configurar una VLAN es necesario agregar una identificación, para ello existe un objeto LineEdit, donde el usuario ingresara el id (id es el identificador que se le asigna a la vlan) por medio de una variable de tipo string. Los datos son enviados por medio de la subrutina system haciendo uso del comando vconfig. Estos datos son comprobados mediante el valor retornado por dicha subrutina, ya que system funciona de la siguiente forma en C:

El proceso espera a que finalice la ejecución de la subrutina y devuelve la salida del programa ejecutado.

- int system (cadena)

- const char *cadena;

Cadena es el comando a ejecutar.

El estado de salida del programa ejecutado es -1, o 127 en caso de error, y envía un 0 si la cadena es correcta, con base en esto se determina si el comando enviado ha sido correcto, de lo contrario se envía un mensaje al usuario para que vuelva a intentarlo. Cuando el usuario ingresa el id de la VLAN es necesario dar clic en el botón **aceptar** para que el comando se ejecute; si es exitoso se enviará un mensaje y se visualizará del lado derecho la VLAN agregada al sistema, en caso contrario se enviara un mensaje al usuario del error generado; todas las VLAN agregadas estarán dentro de un objeto llamado QlistWidget.

Para configurar cada VLAN con datos como la dirección ip y la máscara de subred es necesario seleccionarla, para que la interface pueda leer el tipo de dato a configurar, mismo que es almacenado en una variable tipo cadena. Estos datos se encuentran en otro GroupBox que contiene lo siguiente:

- PushButton
- 2 labels
- 2 LineEdit

Otro proceso es la eliminación de las VLAN; en este caso, la forma de enviar el comando correcto es que el usuario seleccione la VLAN a eliminar y de clic en el botón **eliminar**, una vez haciendo esto se tiene el dato especifico de la VLAN y es almacenado en otra variable string, para enviar la cadena correspondiente mediante la subrutina system. Existen otros tres procesos diseñados con el propósito de ofrecer información al usuario sobre los procesos que ya realizó; dichos procesos se ejecutan mediante botones visualizando una ventana independiente para cada uno de ellos, y se describen en los apartados siguientes.

2.2.5 Clase información

Esta ventana se ejecuta al dar clic sobre el botón **información de tarjetas de red**, y contiene los siguientes archivos:

informacion.ui: Es la interfaz de usuario, en la Figura 2.10 se muestra el diseño.

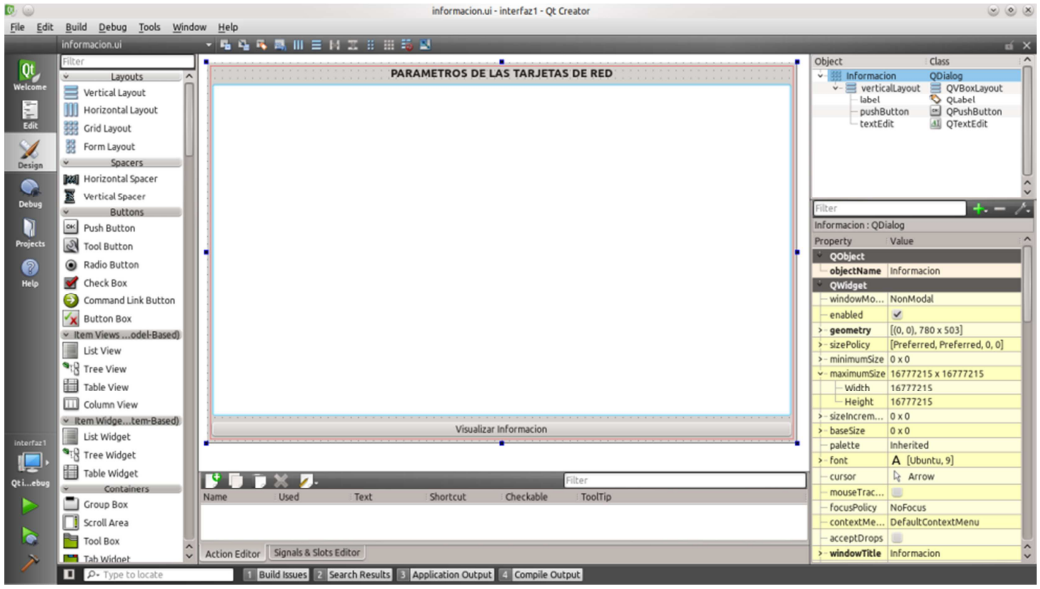


Figura 2. 10 Diseño de ventana que muestra configuración de las tarjetas de red

informacion.h: Se declara la clase información

Informacion.cpp: Contiene el método para poder leer un archivo de texto en QT.

El proceso de este método consiste en ejecutar el comando **ifconfig**, el cual permite visualizar la información sobre las tarjetas de red, misma que se guarda en un archivo de texto llamado ifconfig.txt. Una vez teniendo este dato se procede a hacer la lectura de dicho archivo para mostrarla en pantalla. El mismo procedimiento se aplica para las dos ventanas restantes.

2.2.6 Clase allinfovlan

El botón **ver VLAN del equipo** genera la pantalla mostrada en la Figura 2.11, y contiene los archivos siguientes:

allinfovlan.ui: Interfaz del usuario.

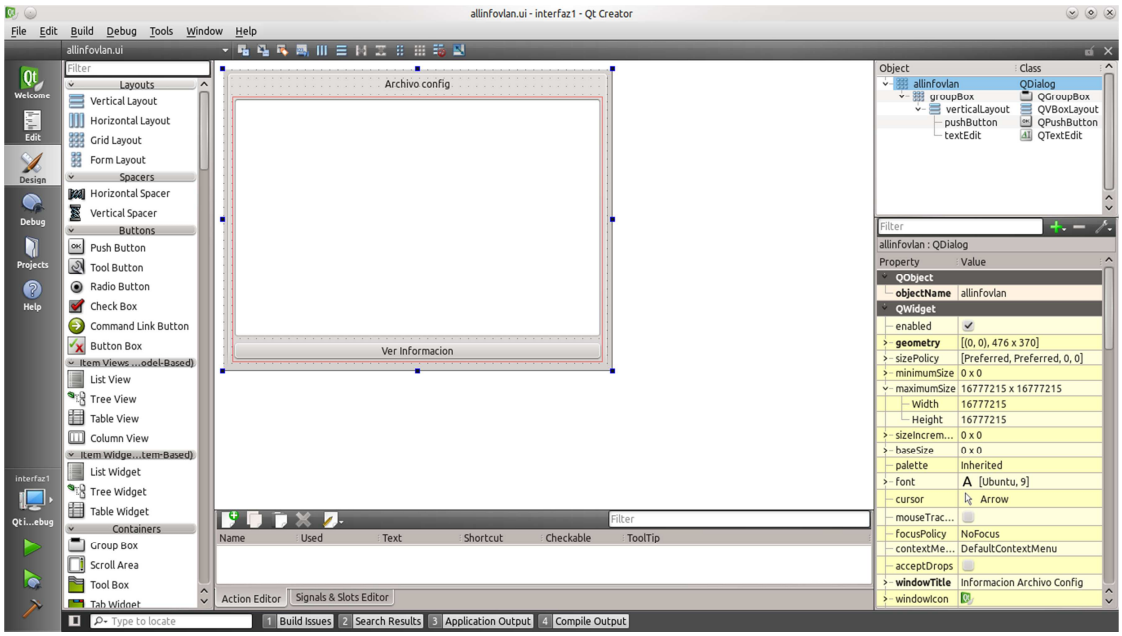


Figura 2. 11 Diseño de ventana que muestra información de las VLAN

allinfovlan.h: Se declara la clase allinfovlan

allinfovlan.cpp: Realiza la lectura de un archivo de nombre ***config*** que se encuentra en el directorio /proc/net/vlan para mostrar la información en pantalla; este directorio contiene los procesos de las VLAN en curso; cuando el usuario va agregando VLAN al sistema este archivo se crea automáticamente con la configuración de cada una de ellas.

2.2.7 Clase infovlan

Por último el botón **información de VLAN seleccionada** abre la pantalla mostrada en la Figura 2.12, y contiene los siguientes archivos:

infovlan.ui: Interface de usuario

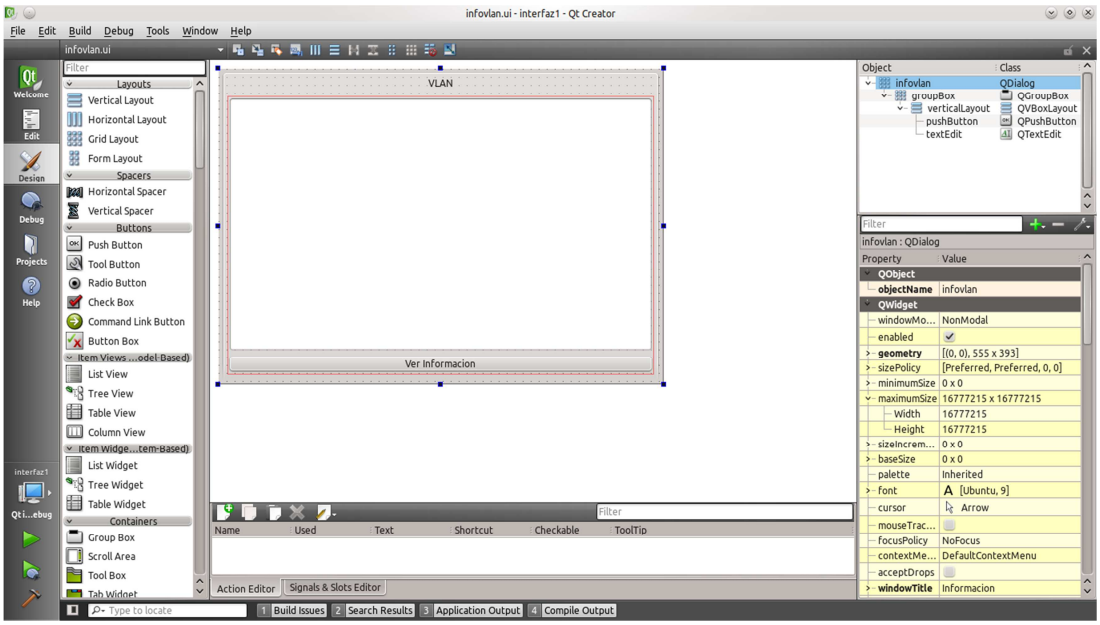


Figura 2. 12 Diseño de ventana que muestra información de una sola VLAN

infovlan.h: Se declara la clase infovlan

infovlan.cpp: Aquí se encuentra el método para leer los archivos. En este proceso el usuario puede seleccionar una VLAN específica para visualizar sus datos, cada vez que un usuario agrega una VLAN se crea un archivo de configuración de la misma en el directorio /proc/net/vlan.

Capítulo 3

3.1 Implementación de VLAN en LINUX modo línea de comando

La implementación de VLAN en LINUX normalmente se hace vía software, con el uso de ciertas herramientas ejecutadas mediante la consola o línea de comandos de cualquier distribución de LINUX. El proceso para crear las VLAN es similar al de la configuración de una tarjeta de red en LINUX, ya que estas al crearlas, se añaden tarjetas de red virtuales asociadas a la tarjeta de red física existente, puede existir más de una VLAN en una tarjeta de red. Cabe recordar que LINUX, siendo un sistema operativo de código libre, puede presentar dificultades con respecto a las tarjetas de red, ya que no todas ellas tienen soporte para VLAN. Las herramientas que se utilizan para la implementación permiten enviar diferentes parámetros a los archivos de configuración, y son:

- Ifconfig: Permite ver el estado de las interfaces de red que se encuentran en el equipo, así como su configuración y otros parámetros más.

- vconfig: Permite agregar, configurar y eliminar las VLAN al sistema.

3.1.1 Arquitectura de la red

La Figura 3.1 muestra un esquema de configuración de una red pequeña en la cual se llevará a cabo la implementación de las VLAN. Para ello se requiere:

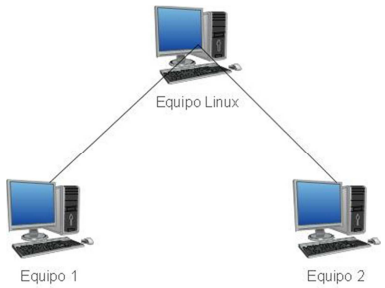


Figura 3. 1 Arquitectura de la red

- Un equipo con sistema operativo LINUX, con dos o más tarjetas de red tipo Ethernet.
- Dos equipos con sistema operativo LINUX o Windows.

- Dos cables cruzados para interconectar los equipos.

3.1.2 Proceso en modo línea de comando

El procedimiento consiste en asignar direcciones IP estáticas al equipo LINUX. Este cuenta con tres tarjetas de red, mismas que el sistema operativo identifica como ***eth0***, ***eth1*** y ***eth2***, en donde serán configuradas las VLAN. Para añadir las VLAN es necesario ejecutar una terminal y agregar los comandos siguientes (ver Figura 3.2):

- vconfig add eth1 2
  Añade la VLAN 2 de la tarjeta eth1

- vconfig add eth2 5
  Añade la VLAN 5 de la eth2

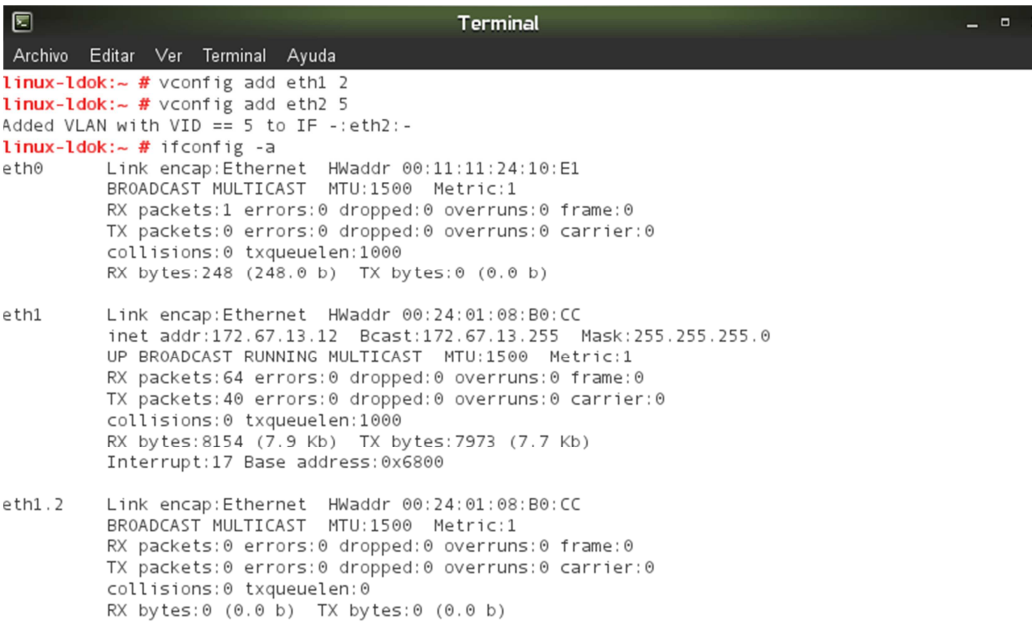


Figura 3. 2 Comando ifconfig

Para asignar los parámetros a cada VLAN es necesario usar el comando ifconfig:

ifconfig eth1.2 172.67.13.9 netmask 255.255.255.0 broadcast 172.67.13.255 up
ifconfig eth2.5 172.67.12.9 netmask 255.255.255.0 broadcast 172.67.12.255 up

La Figura 3.3 muestra las VLAN creadas:

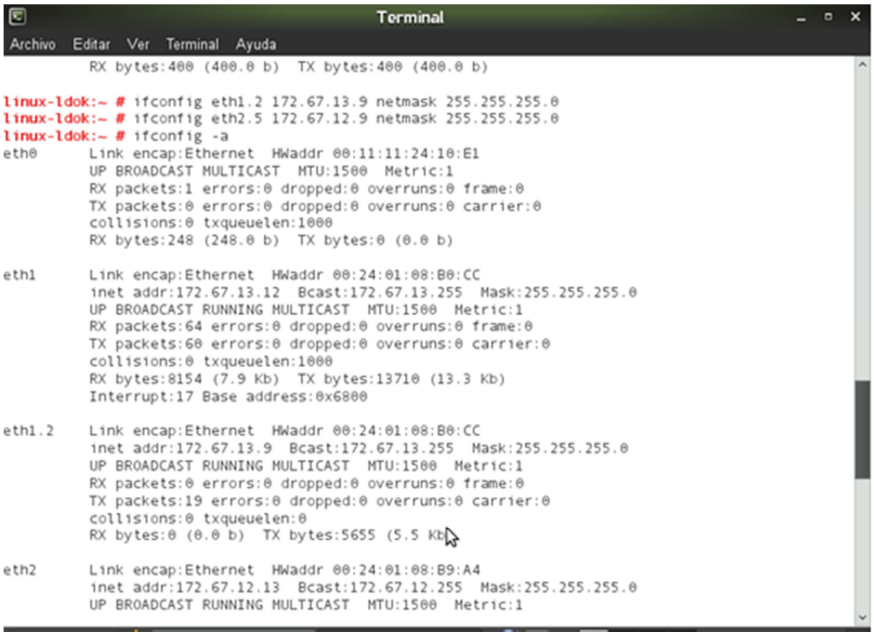


Figura 3. 3 Visualización de las VLAN

La configuración de los otros equipos se muestra en las Figuras 3.4 y 3.5:

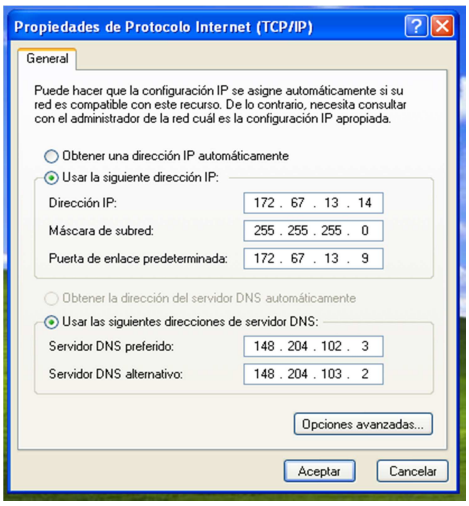


Figura 3. 4 Configuración de Tarjeta de red equipo 1

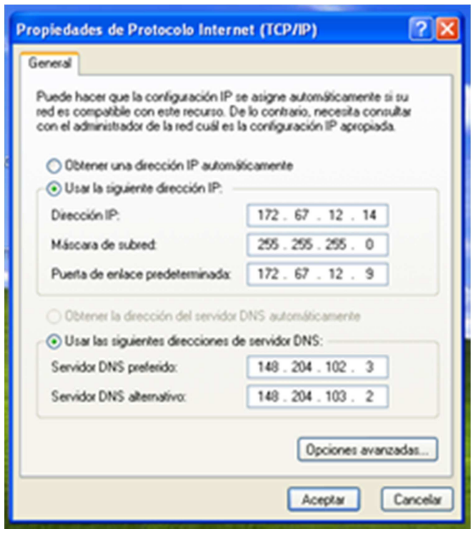


Figura 3. 5 Configuración de Tarjeta de red equipo 2

Una vez configurados, se prueba si existe conexión con ellos. Desde LINUX se envían los comandos:

- Ping 172.67.13.14
- Ping 172.67.12.14

Si la conexión es correcta, se observan en la pantalla los resultados mostrados en la Figura 3.6:

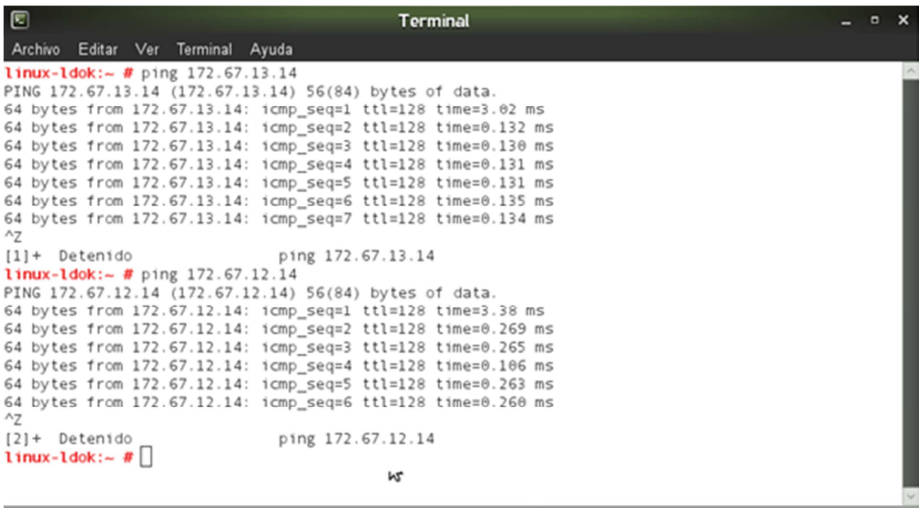


Figura 3. 6 Comando ping del equipo LINUX

Ahora desde el host 1 envía un ping a la VLAN 2 (Figura 3.7):

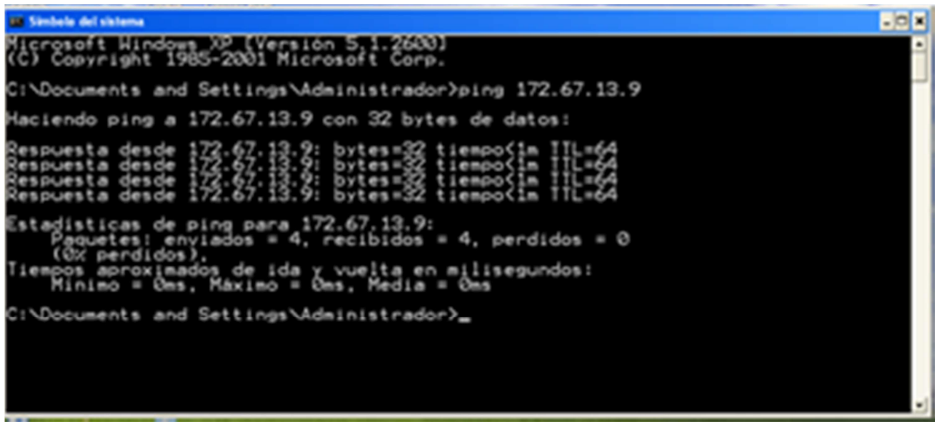


Figura 3. 7 Comando ping del equipo 1

Desde el host 2 se envía un ping a la VLAN 5 (Figura 3.8):

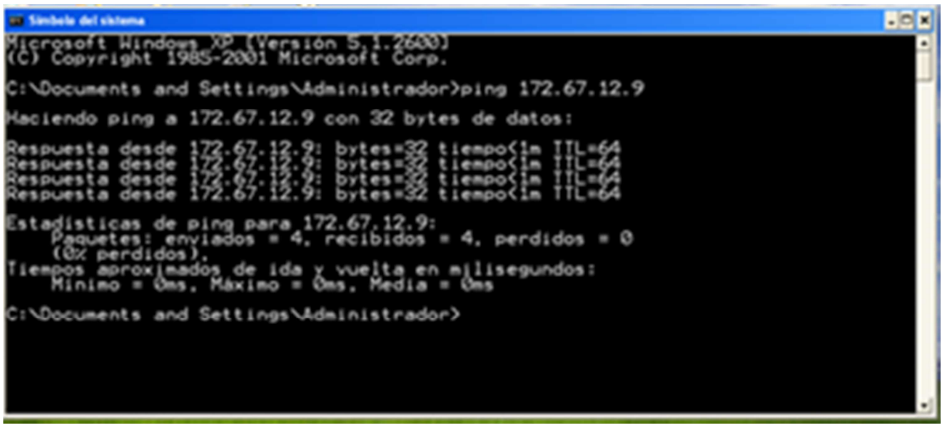


Figura 3. 8 Comando ping del equipo 2

Uno de los objetivos de las VLAN es que no se puedan ver entre ellas, ya que solo los equipos que pertenecen a la misma VLAN podrán comunicarse entre sí; en este caso si se envía un ping del equipo 1 al equipo 2 no habrá conexión, tal como se muestra en la Figura 3.9, donde se comprueba que no hay conectividad. La forma de que dos VLAN o más puedan verse entre ellas es poniendo un enrutador el cual permitirá que estas puedan tener comunicación.

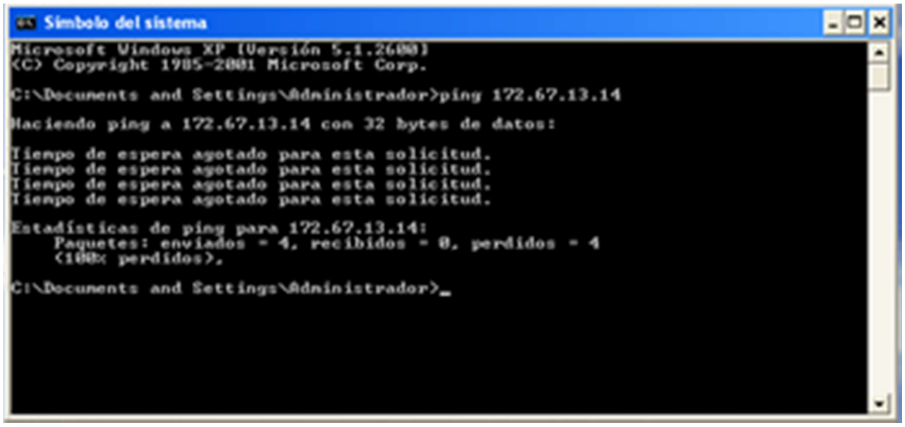


Figura 3. 9 Comando ping entre las VLAN

Hasta ahora el equipo con sistema operativo Open Suse es el encargado de gestionar las VLAN, por lo que para interconectar ambas VLAN es necesario habilitar el enrutamiento dentro del mismo sistema; esto se logra de la siguiente manera:

Existe un archivo en la dirección /etc/sysconfig/sysctl, el cual por omisión tiene la configuración que se muestra en la Figura 3.10.

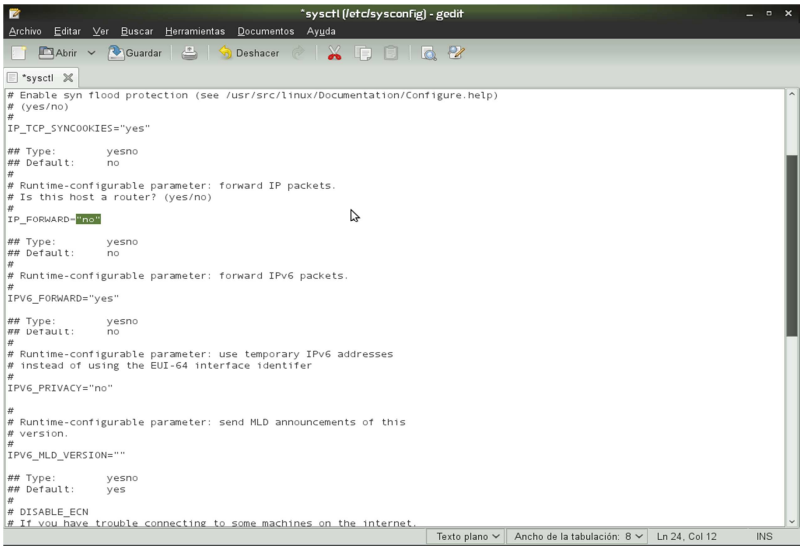


Figura 3. 10 Archivo sysctl

En el archivo sysctl se busca la línea IP_FORWARD=”no”, se cambia por IP_FORWARD=”yes”, y se graba nuevamente el archivo (Figura 3.11).

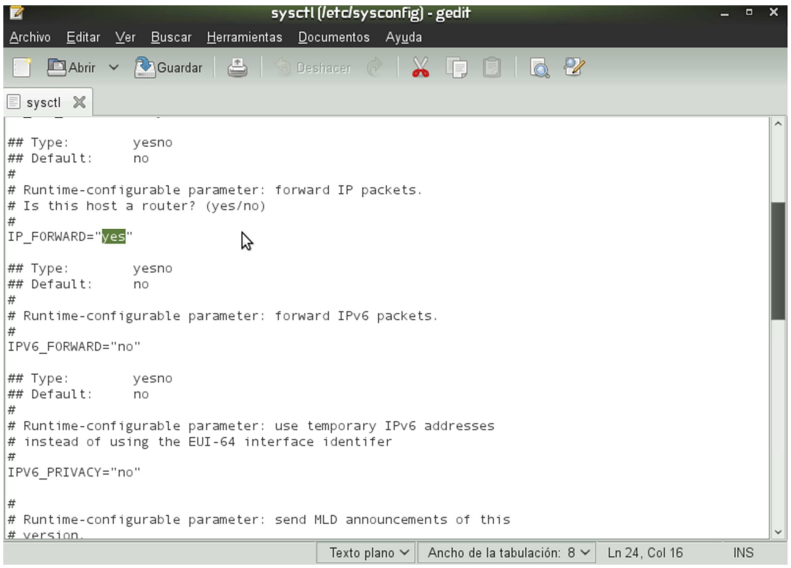


Figura 3. 11 Modificación al archivo sysctl

Finalmente se reinicia el equipo, es posible que al reiniciarlo se requiere configurar nuevamente las VLAN. Ahora ya existe conexión entre host 1 y host 2; esto se puede hacer probar enviando un ping entre equipo 1 y equipo 2 y viceversa, como se muestra en las Figuras 3.12 y 3.13:

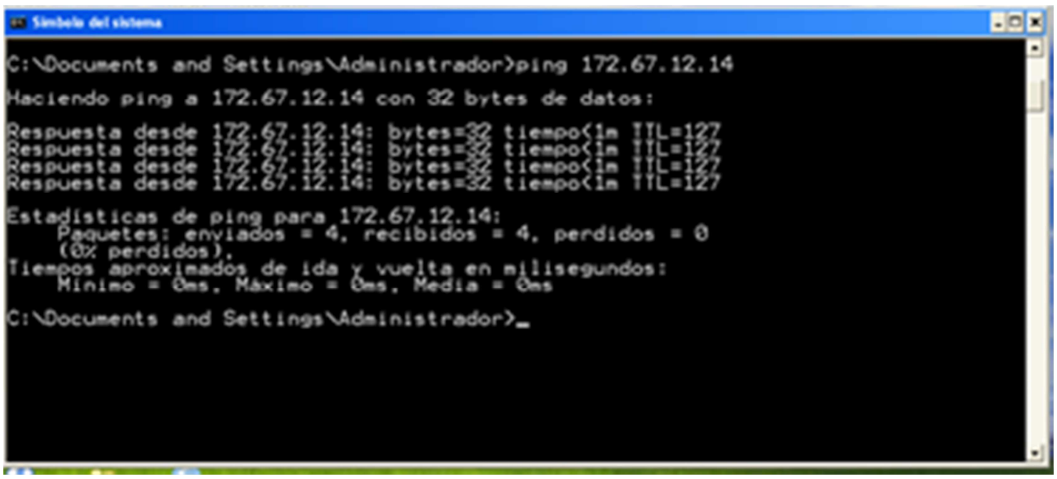


Figura 3. 12 Comando ping de equipo 1 a 2

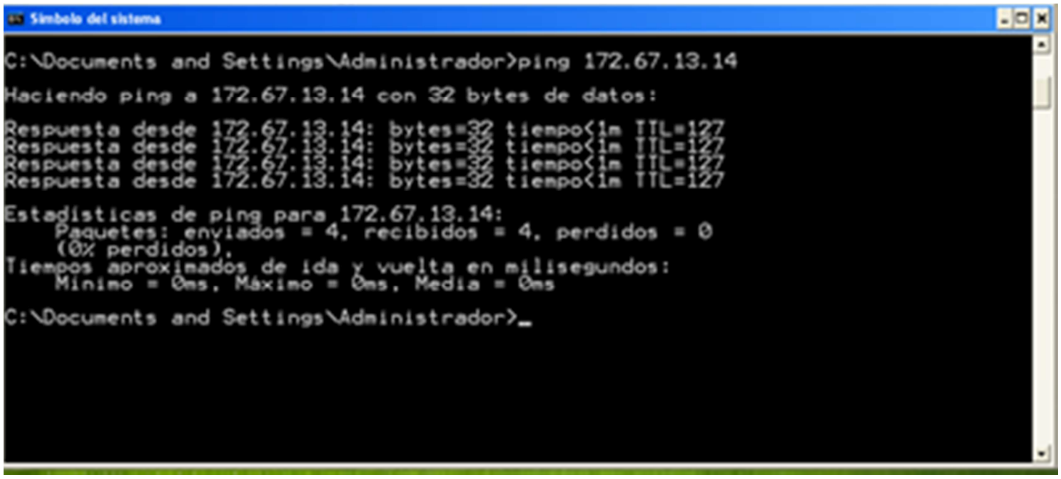


Figura 3. 13 Comando ping de equipo 2 a 1

3.2 Implementación de VLAN en la interfaz desarrollada

3.2.1 Proceso en modo gráfico

La arquitectura de la red sigue siendo la mostrada en la Figura 3.1. Se selecciona la aplicación a utilizar, en este caso el programa es interface1, ubicado en la carpeta del mismo nombre, como se muestra en la Figura 3.14.

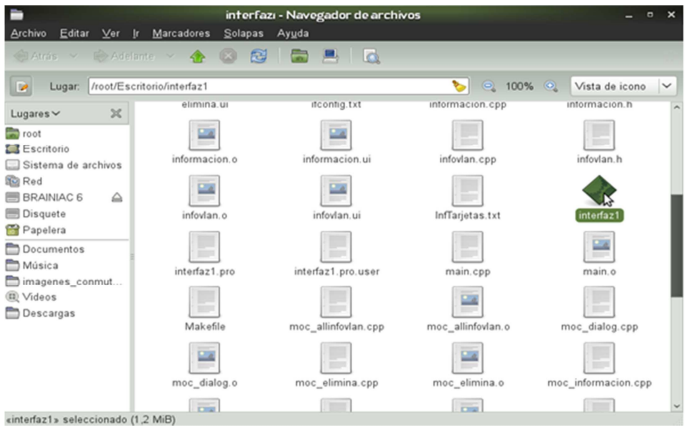


Figura 3. 14 Archivo Ejecutable

Al abrir la aplicación se muestra una pantalla con varias opciones (Figura 3.15), una de las cuales es ejecutar el módulo 802.1Q; así mismo, se pueden ver

las tarjetas de red en el equipo, Agregar y Eliminar VLAN, y la opción Salir. Al ejecutar la activación del módulo se recibe el mensaje mostrado en la Figura 3.16.

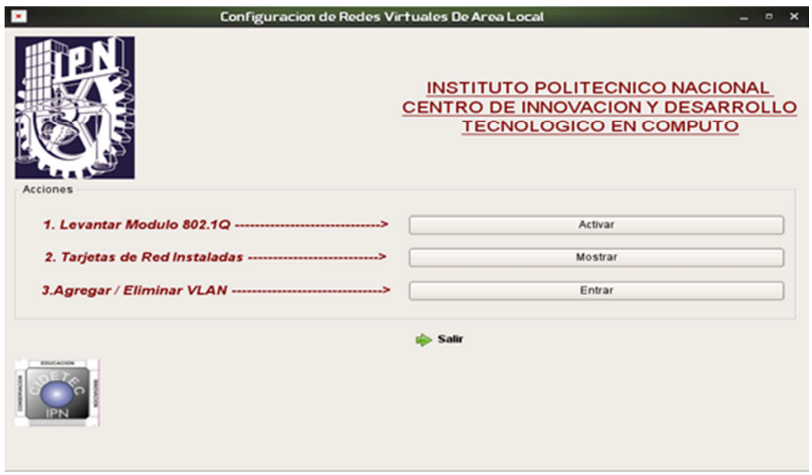


Figura 3. 15 Pantalla principal

Figura 3. 16 Activación del módulo 802.1q

Para iniciar el proceso hay que seleccionar el botón Agregar/Eliminar VLAN, con lo que aparece la ventana mostrada en la Figura 3.17, en la cual se inicia la captura de las VLAN a agregar; para el desarrollo se agregan la VLAN 2 en la tarjeta eth1 y la VLAN 5 en la eth2:

- VLAN2 eth1.2
- VLAN5 eth2.5

Figura 3. 17 Configuración de parámetros de las VLAN

Se indica la tarjeta en donde se agregará la VLAN y el ID de la misma y se da clic en el botón **aceptar**, con lo que aparece el mensaje de la Figura 3.18:

Figura 3. 18 Selección de tarjeta de red

Una vez agregadas las VLAN aparecen en el cuadro del lado derecho; entonces se puede asignar su dirección IP y su máscara de Red, y para pasar los parámetros se da clic en aceptar, como se muestra en la Figura 3.19:

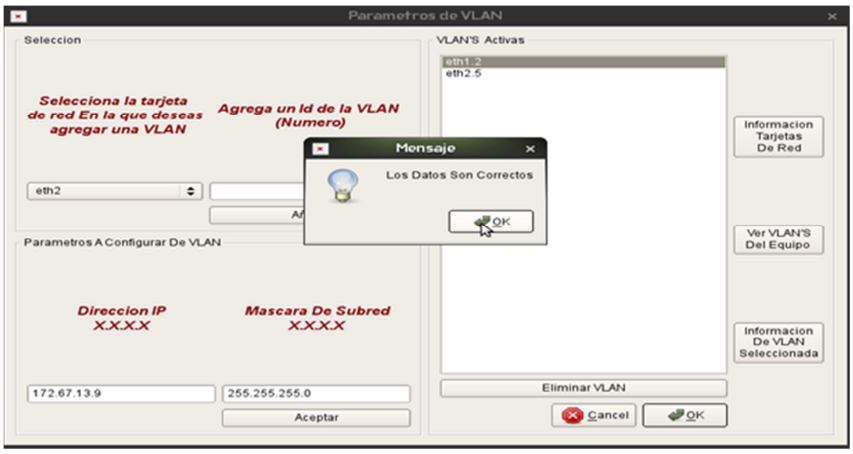


Figura 3. 19 Comprobación de datos correctos

Para este caso, los parámetros son VLAN2 172.67.13.9 y VLAN5 172.67.12.9, ambos con máscara de 24 (255.255.255.0):

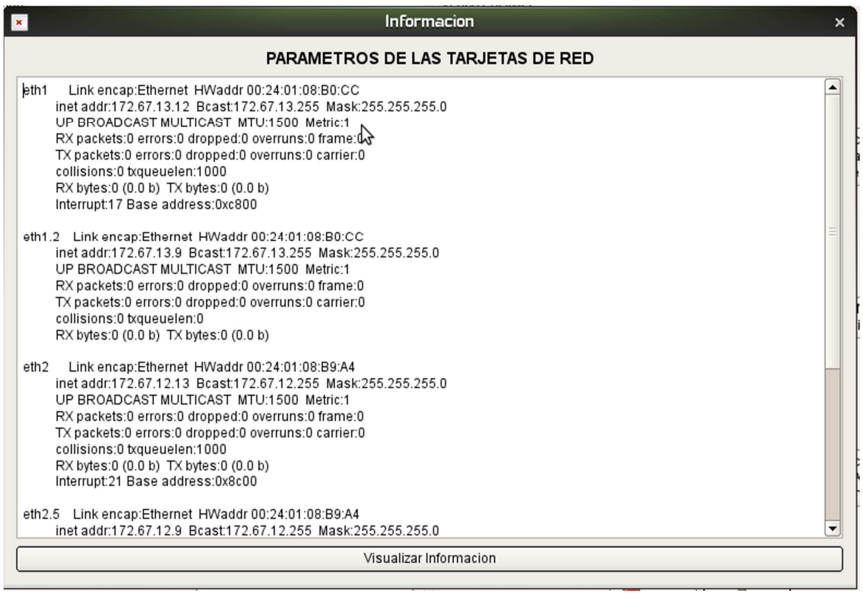


Figura 3. 20 Información de las tarjetas de red

La opción **Información de las tarjetas de red** permite verificar si los datos enviados por la interface son correctos (Figura 3.20); en caso de error se vuelven a asignar los parámetros. Una vez que el equipo LINUX configura las VLAN, se requiere configurar los equipo 1 y 2, con sus respectivas direcciones y puertas de enlace, como se muestra en las Figuras 3.21 y 3.22. Posteriormente se conectan

estos equipos al equipo LINUX mediante cables cruzados, de la siguiente forma: Fastethernet 1 al equipo 1 y Fastethernet 2 al equipo 2.

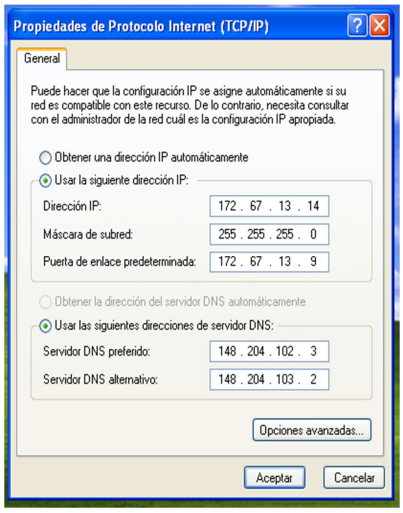


Figura 3. 21 Configuración del equipo 1

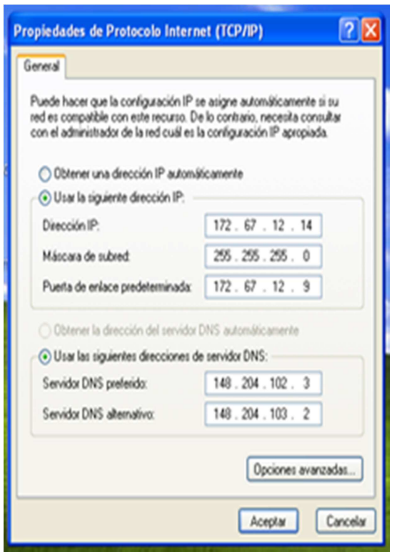


Figura 3. 22 Configuración de equipo 2

Para comprobar que hay conectividad entre los dos equipos y el equipo LINUX se mandan comandos ping, como se muestra en la Figura 3.23:

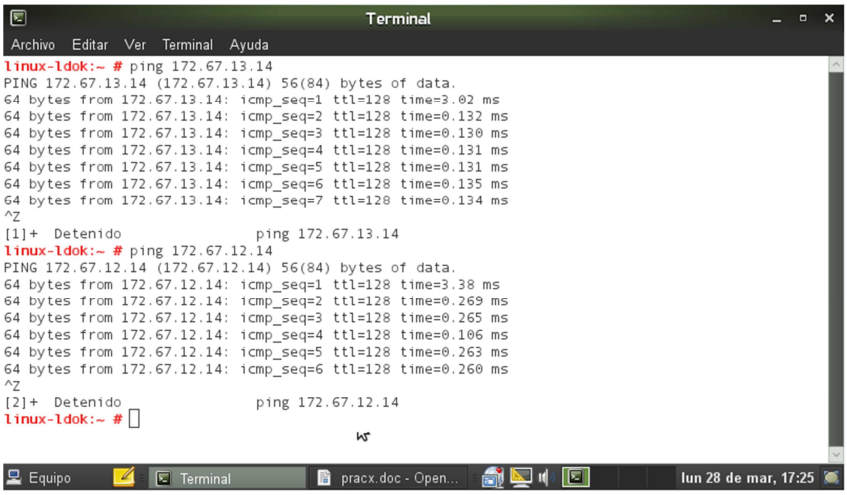


Figura 3. 14 Ping de Equipo LINUX a equipo 1 y 2

Ahora desde el equipo 1 se envía el ping al equipo LINUX: Ping 172.67.13.9 VLAN 2; el resultado debe ser el mismo (Ver Figura 3.24).

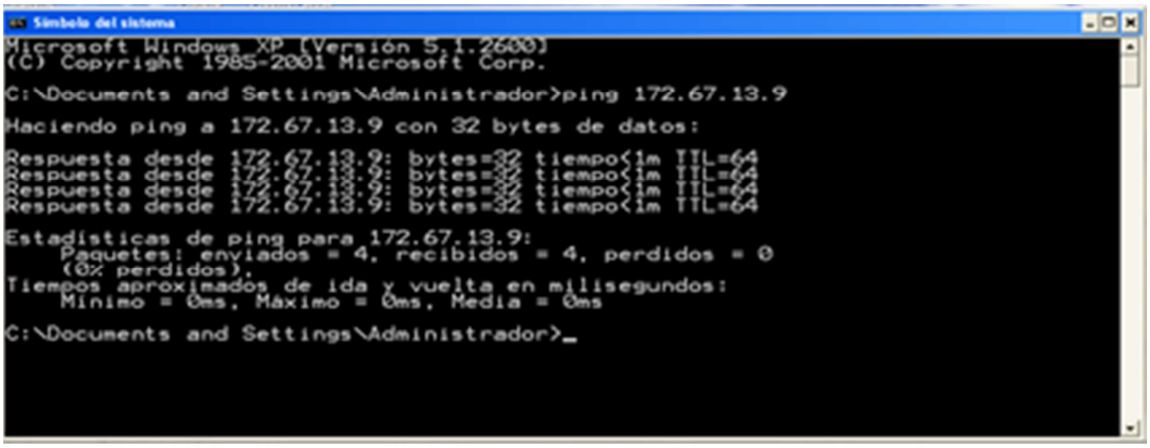


Figura 3. 24 Ping de Equipo 1 a Equipo LINUX

Desde el equipo 2 se envía ping al equipo LINUX (Figura 3.25):

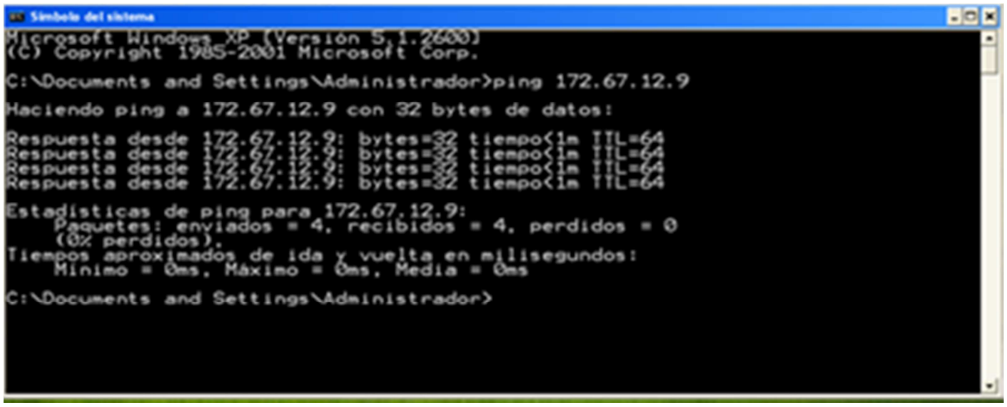


Figura 3. 15 Ping de equipo 2 a equipo LINUX

Para interconectar ambas VLAN es necesario habilitar el enrutamiento en Open Suse; para ello la línea IP_FORWARD=”no” en el archivo **sysctl** se cambia por IP_FORWARD=”yes”, y se guarda nuevamente el archivo (Figura 3.26):

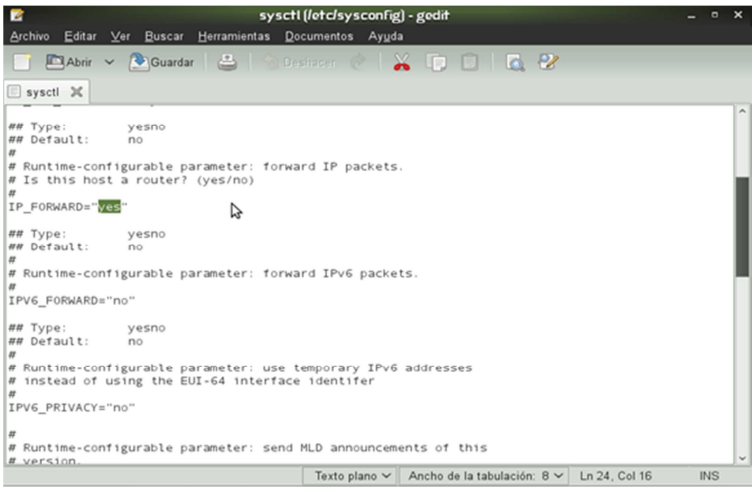


Figura 3. 26 Archivo sysctl

Finalmente se reinicia el equipo, es posible que al reiniciarlo se tengan que volver a configurar las VLAN, ya que no se tiene la opción de poder guardar la configuración de las VLAN.

Usando el comando ping entre los equipos 1 y 2 se comprueba que hay conectividad entre ellos, como se muestra en las Figuras 3.27 y 3.28 donde cada uno de los equipos envía ping al equipo con el cual no tenía conexión.

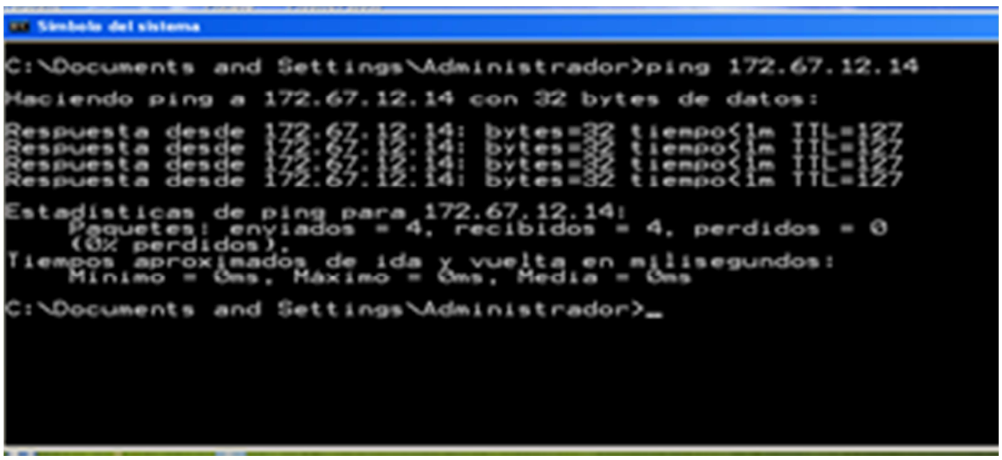


Figura 3. 27 Ping Equipo 1 a equipo 2

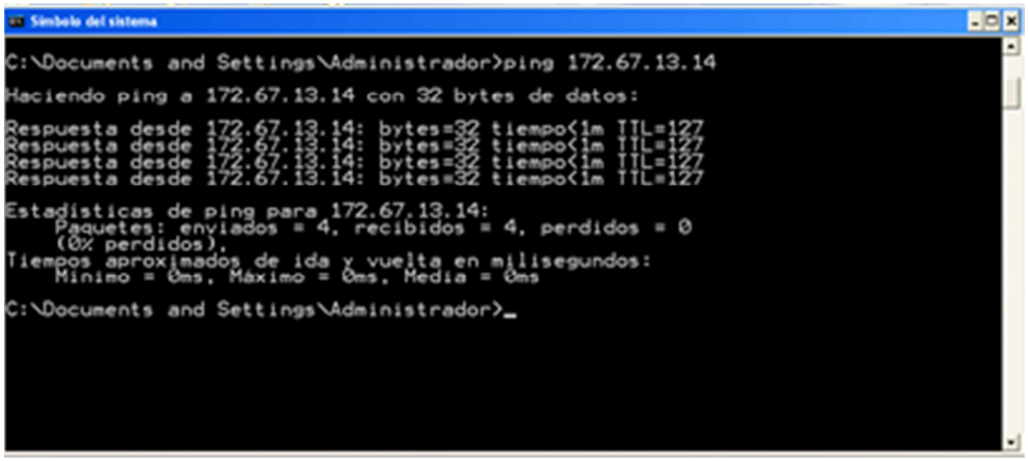


Figura 3. 28 Ping de Equipo 2 a equipo 1

Capítulo 4

4.1 Comparación de los métodos para la configuración de VLAN

El capítulo anterior muestra la configuración de las VLAN haciendo uso de dos tipos de aplicación, una ya existente en modo línea de comando y la otra desarrollada en esta tesis con una interfaz gráfica. Con las pruebas hechas anteriormente se pueden comparar dichas interfaces, para determinar cuáles son las ventajas y desventajas que presenta el uso de cada una de ellas.

La interfaz gráfica permite configurar las funciones más comunes del comando vconfig en la consola, en este caso es necesario conocer la ejecución de cada comando y como se envían los parámetros a ellos; esta es una de las ventajas más importantes que la interface gráfica ofrece, permitiendo que la configuración de las VLAN sea un proceso sencillo y práctico de realizar.

Por otra parte, cabe mencionar que en la actualidad y con la idea de simplificar el uso de las computadoras para cualquier persona, llámese novato o experto, se ha vuelto común utilizar la interfaz gráfica para que el usuario interactúe y establezca un contacto más fácil e intuitivo con dicha computadora; esto no quiere decir que se sustituya a la línea de comando, simplemente existe una alternativa más para que el usuario pueda llevar a cabo sus actividades.

Algunas de las dificultades que se pueden encontrar haciendo uso de la línea de comandos son:

- Es muy común equivocarse al teclear comandos u órdenes que sean muy largas y/o con muchos parámetros.

- Se requiere de un tiempo considerable para aprender el uso de los programas (aplicaciones) necesarios.
- Estas órdenes se pueden ejecutar en una gran cantidad de opciones; en ocasiones también contienen una gran cantidad de configuraciones, y cada una de ellas es diferente respecto al programa usado.

Algunas de las ventajas que presenta la línea de comando respecto a la interfaz gráfica son:

- La línea de comando permite tener acceso a todas opciones que tiene el comando vconfig, en la interfaz gráfica solo estarán disponibles las opciones necesarias para configurar las VLAN.

- Después de un determinado tiempo y con el conocimiento correspondiente permite automatizar los procesos mediante scripts.

- La respuesta de la línea de comandos es más rápida que la interfaz gráfica.

La Interfaz gráfica por su parte presenta:

- Proporciona una mejor visión de lo que está sucediendo.
- Evita los errores del teclado.
- Involucra al usuario a comprender lo que está realizando.
- Requiere de menos aprendizaje y memorización.

4.2 Pruebas

Las pruebas que se llevaron a cabo demuestran que la herramienta **vconfig** funciona en la interfaz gráfica; es decir, que el mismo resultado obtenido al crear las VLAN en modo línea de comando ocurre usando la interfaz gráfica. La forma de hacer esta prueba es por medio de un conmutador configurable; para este desarrollo se utilizó un equipo Catalyst de la serie 3560 de CISCO el cual tiene soporte para las VLAN, así como una computadora con LINUX el cual funge como un enrutador, con sistema operativo Open Suse en su versión 11.2. Se hace uso del conmutador de CISCO para apoyar la verificación de conectividad entre las VLAN que se crean en LINUX, así como poder añadir más de dos equipos de cómputo en cada VLAN. Es necesario mencionar que la idea es reemplazar este conmutador CISCO por uno desarrollado en LINUX, pero en este caso se hace uso solo para verificar que la interfaz gráfica está cumpliendo su función.

Cabe mencionar que la misma prueba que se realizó con el conmutador configurable primero se realizó con otro propietario de 3com, este dispositivo no es configurable y el resultado fue el siguiente.

- No existe comunicación entre los equipos hacia las VLAN, debido a que este equipo no tienen el soporte para VLAN, si el equipo de cómputo se requería comunicar al equipo LINUX, primero tiene que pasar la información por el conmutador y es imposible hacer la conexión, por dicha razón se hace uso del conmutador de CISCO ya que este si tiene este soporte.

La topología usada para llevar acabo las pruebas se muestra en la Figura 4.1; como se observa en dicha figura, se crean dos VLAN en LINUX haciendo uso de la interfaz gráfica, cada una con sus respectivos parámetros. Para activarlas en la tarjeta de red usada, (en este caso eth0), si el equipo tiene más de dos tarjetas de red se hace uso de la que se desee y la configuración sigue siendo la misma. Se aplica el siguiente comando en consola:

Ifconfig eth0 0.0.0.0 up

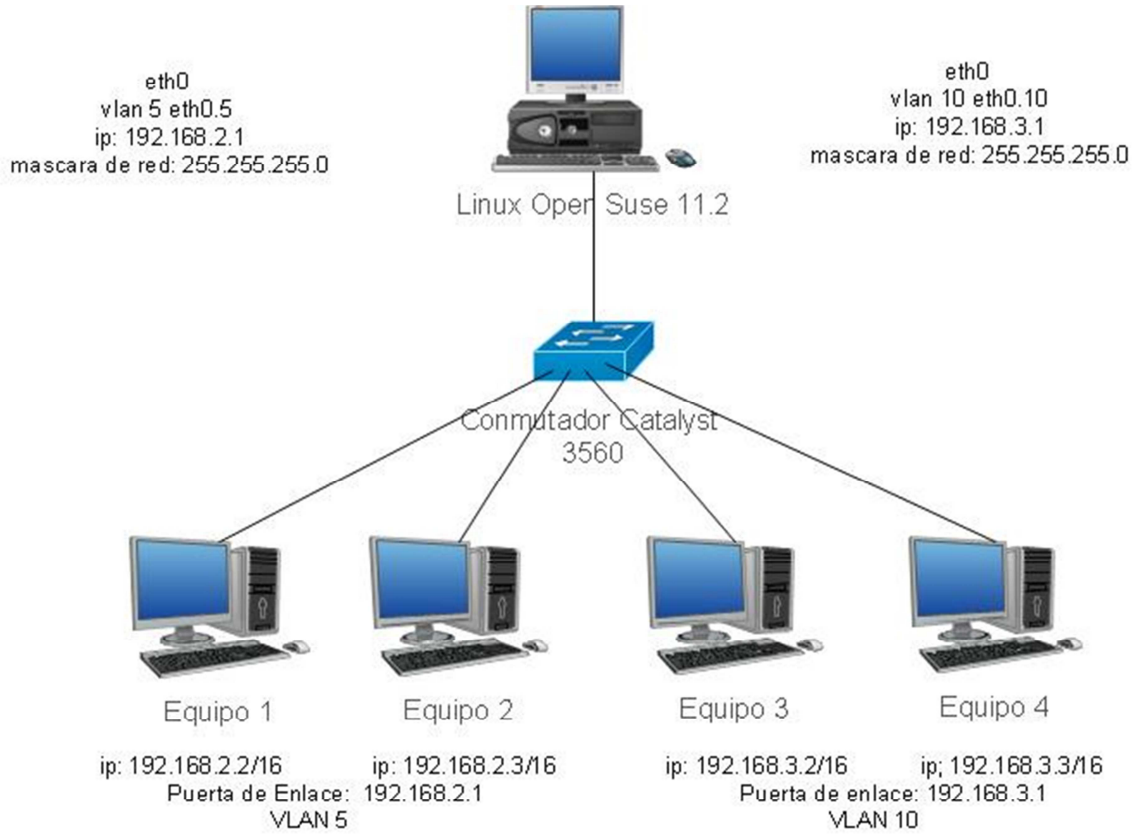


Figura 4. 1 Topología de Pruebas

Con la información de la figura 4.1 se hace uso de la interfaz gráfica para agregar las VLAN correspondientes, una vez hecho esto se configura el conmutador CISCO creando las mismas VLAN y configurando uno de los puertos en modo troncal que significa que puede transportar múltiples VLAN sobre el mismo puerto.

Configuración del conmutador CISCO Catalyst 3560

La siguiente configuración que se muestra es para poder declarar las dos VLAN existentes en el sistema Linux, cabe aclarar que es necesario que las VLAN agregadas en el sistema LINUX en este caso con identificador 5 y10 tienen que ser la mismas en el conmutador, ya que de no ser así se presentaran conflictos de comunicación. Por lo tanto se agrega la VLAN 5 y 10 en conmutador configurable donde estas utilizaran 4 puertos del conmutador, dos para las VLAN 5 y otros dos para la VLAN 10 posteriormente se utilizara un último puerto que es el que permitirá el paso de las dos VLAN a través del sistema LINUX.

Al acceder al conmutador por medio del puerto serial se realiza una conexión para poder configurar el equipo, el modo en el que inicia no permite hacer configuraciones por lo tanto debemos de entrar al modo privilegiado (#) escribiendo el comando**, configure terminal**, y se procede agregar cada VLAN. Una vez que las VLAN se han agregado en el conmutador es necesario determinar que puertos tendrán acceso a cada VLAN. En este caso se asignó el puerto 2 y 3 a la VLAN 5 y 4, 5 a la VLAN 10, esto se logra accediendo a cada puerto con el comando **interface fastethernet0/N°,** donde N° es el número del puerto, posteriormente se le indica que VLAN tendrá acceso por dicho puerto con el comando **switchport access VLAN**#, por último se configura el puerto 10 que estará conectado al equipo LINUX el cual permitirá el paso de múltiples VLAN. Como nota importante si se utiliza un conmutador diferente que tiene soporte para VLAN, es necesario revisar el manual de configuración para obtener los resultados esperados.

```
switch>
switch#configure terminal
switch(config)#vlan 5
switch(config-vlan)#name vlan5
switch(config-vlan)#exit
switch(config)#vlan 10
switch(config-vlan)#name vlan10
switch(config-vlan)#exit
switch(config)#exit
```

```
switch#wr mem
switch#conf terminal
switch(config)#interface fastethernet0/2
switch(config-if)#switchport access vlan 5
switch(config-if)#no shutdown
switch(config-if)#exit
switch(config)#interface fastethernet0/3
switch(config-if)#switchport access vlan 5
switch(config-if)#no shutdown
switch(config-if)#exit
switch(config)#interface fastethernet0/4
switch(config-if)#switchport access vlan 10
switch(config-if)#no shutdown
switch(config-if)#exit
switch(config)#interface fastethernet0/5
switch(config-if)#switchport access vlan 10
switch(config-if)#no shutdown
switch(config-if)#exit
switch(config)#interface fastethernet0/10
switch(config-if)#switchport trunk encapsulation dotq1
switch(config-if)#switchport trunk allowed vlan 5,10
switch(config-if)#switchport mode trunk
switch(config-if)#no shutdown
switch(config-if)#exit
switch(config)#exit
switch#wr mem
switch#
```

Finalmente se configura cada uno de los equipos de cómputo con sus respectivos parámetros, dirección ip y puerta de enlace. Al llevar acabo esto, se puede probar la conectividad de los equipos de cómputo dependiendo a la VLAN que pertenecen. Enviando un ping entre equipos que pertenezcan a la misma VLAN y el resultado será la existencia de comunicación, sucederá lo contrario si dos equipos se intentan comunicar y pertenecen a distinta VLAN.

Por último los equipos deben verificar comunicación hacia la VLAN, si existe comunicación, indicara que las VLAN agregadas por medio de la interfaz gráfica fue satisfactoria.

Capítulo 5

5.1 Conclusiones y resultados

Con base en lo expuesto anteriormente, en este trabajo se llevó a efecto el diseño e implementación de una interfaz gráfica para la configuración de VLAN en un sistema operativo LINUX, tomando en consideración el uso de una distribución de uso libre, tal como es el caso de Open Suse 11.2. Al respecto, se encontraron ciertas problemáticas que fueron tratadas en el desarrollo de esta tesis.

- Existe muy poca información sobre desarrollos e investigaciones sobre el tema, se encontraron tesis respecto a las VLAN pero no orientadas con el uso del software libre [23], [24], [25], a excepción de una tesis de maestría que aporta una metodología para la implementación de VLAN en LINUX [26]. Con el desarrollo de esta tesis se complementa para seguir desarrollos en sistemas operativos de uso libre.

- Si bien se encuentra información sobre el sistema operativo LINUX, los libros y manuales disponibles no profundizan en el tema de las VLAN; la mayor parte de la información se encuentra en páginas de Internet, foros y artículos, los cuales no siempre provienen de fuentes confiables.

- Existe una gran cantidad de distribuciones que se pueden usar de manera libre y cada una de ellas tiene distintas formas de trabajar sobre implementaciones de red.

- La mayor parte de las investigaciones y desarrollos en el campo de las redes son de organismos propietarios tales como CISCO, esto conlleva a que la mayor parte de las implementaciones se desarrollen con el fabricante.

- En el caso de llevar acabo la implementación de las VLAN en LINUX existen pocos documentos formales que indiquen la posibilidad de llevarlas a cabo.

- Normalmente, si un administrador de red utiliza las tecnologías de ciertos propietarios es porque tiene las herramientas necesarias para llevar a efecto su actividad con facilidad. Los propietarios, al no dar a conocer su tecnología, provocan que los usuarios descarten otras tecnologías desconocidas sin saber qué ventajas o desventajas tienen sobre estas.

- Por otra parte las empresas no invierten en capacitación para tecnologías que no se han aplicado en el campo laboral, ya que se prefiere asegurar tecnologías utilizadas previamente, con un soporte garantizado.

La distribución Open Suse en su versión 11.2, requiere de una herramienta que permita la funcionalidad de un conmutador como es el caso de CISCO, ya que se comprueba que el uso de un conmutador de este tipo permite realizar la funcionalidad de las VLAN, si se tiene este conmutador en el sistema LINUX sería posible hacer un análisis para comprarlo con el ya existente en el mercado.

Si el equipo LINUX se desea utilizar como conmutador existe un conflicto, las VLAN como tal no cumplen su función al 100%, tal como el concepto lo indica. Esto debido a que en LINUX no existe dicha aplicación de origen; por ello se requiere hacer uso de un conmutador CISCO, tal como se realizó en las pruebas, con ello las VLAN funcionan de forma correcta haciendo que el equipo LINUX trabaje como un enrutador.

La elección de un entorno de desarrollo en LINUX para programar una interfaz de usuario puede llegar a ser un poco complicada y requiere de tiempo para alcanzar un uso adecuado de ellos. En este caso QT Creator es la herramienta que permitió desarrollar la aplicación debido a su facilidad de uso y al no tener tantas dependencias con otro tipo de bibliotecas, caso contrario que ocurrió con Anjuta.

La programación de QT creator en LINUX se pude llevar acabo de una manera práctica haciendo uso del lenguaje C++, en este caso si lo requerido es enviar información hacia la línea de comandos mediante una interfaz gráfica, es posible haciendo uso de la función system

Cabe destacar que una de las aportaciones más grandes de este trabajo es la de brindar a los usuarios la implementación de las VLAN en un ambiente más agradable y fácil de desarrollar, sin que presenten dificultades para su actividad. Finalmente, aquellos que decidan hacer uso de una distribución diferente tendrán la oportunidad de hacer uso de esta herramienta cumpliendo con las características y pequeñas modificaciones que se mencionan en este trabajo.

La interfaz gráfica no tiene todos los complementos que el comando vconfig puede ejecutar bajo la línea de comandos pero es un hecho que tiene las configuraciones esenciales para poder hacer funcionar un sistema creando las VLAN. El resto de las configuraciones son opcionales para el usuario.

El objetivo general de esta tesis es ofrecer una herramienta opcional para los usuarios que les permita crear configurar y eliminar las VLAN dando el mismo funcionamiento que ofrece el comando **vconfig** bajo la línea de comandos; lamentablemente aún no se puede llevar acabo la implementación al 100% ya que Open Suse requiere de una implementación más para que pueda funcionar como un conmutador y permita la misma funcionalidad que ofrecen otros fabricantes.

5.2 Trabajos a futuro

De las complicaciones que se presentaron al llevar a la práctica la implementación de las VLAN, se desprende la necesidad de desarrollar un conmutador en software para el sistema Operativo LINUX, ya que esto permitirá una manipulación óptima y completa de dichas redes; esto debido a que una computadora personal con la distribución Open Suse en su versión 11.2 no funge como un conmutador de capa dos.

Así, otro trabajo a futuro sería probar el uso de la interfaz gráfica ya sobre el conmutador de LINUX y en distintas distribuciones, para verificar su adecuado funcionamiento

Por otro lado se requiere que se profundice en lo que refiere a información sobre la herramienta vconfig ya que esta se podría adecuar de manera que las VLAN en LINUX cumplieran totalmente con su objetivo, ya teniendo esto se requiere configurar las VLAN solo con conmutadores LINUX sin la necesidad de que intervengan conmutadores de otro fabricante. En ese sentido, es necesario que se verifique si el encapsulamiento es el adecuado al estándar 802.1q.

La interfaz gráfica solo cumple con las funciones básicas de las VLAN (crear, configurar y eliminar), sería muy útil implementar una serie de opciones para poder configurar las tarjetas de red del equipo sobre la misma interfaz, además que el sistema pueda guardar configuraciones sin la necesidad de que cada vez que arranque el sistema se tenga que volver hacer.

Referencias

[1]. CISCO Systems, “CCNA 1,2, Pearson, tercera edición, 2004

[2]. D. D. Chowdhury, “High Speed Technology Handbook”, Springer, Berlin, 2000.

[3]. G. Held, “Ethernet Networks”, John Wiley, Tercera edición, 1998.

[4]. Enterasys Networks, 802.1Q VLAN, Enterasys, USA, 2002.

[5]. W. Stallings, “Local & Metropolitan Area Networks”, Quinta Edición, 1997.

[6]. J. E. Goldman, “Local Area Networks”, John Wiley, Segunda Edición, USA, 2000.

[7]. G. C. Sackett, “IBM`s Token-Ring Networking Handbook”, Mc Graw Hill, USA, 1993.

[8]. CISCO Systems, “Understanding VLAN Trunk Protocol (VTP)”, Julio 13, 2007, USA, disponible en, http://www.CISCO.com/application/pdf/paws/10558/21.pdf

[9]. R. Martínez, “Sobre LINUX”, España, 2011, disponible en http://www.LINUX-es.org/sobre_LINUX.

[10]. N. Kimar, “Anjuta IDE Manual”, Johannes Schmid, Versión 2.1.0, USA, 2007, disponible en http://projects.gnome.org/anjuta/.

[11]. Members of the MonoDevelop community, “Documentation”, .2009, disponible en, http://monodevelop.com/Documentation.

[12]. Nokia Corporation, “QT reference documentation”, 2008, disponible en http://doc.qt.nokia.com/4.7/index.html,

[13]. P. Frieden, “VLANS on LINUX”, Marzo 11, 2004, disponible en http://www.LINUXjournal.com/article/7268.

[14]. E. Yourdon & Larry L. Constantine. Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design. , Upper Saddle River, NJ, USA, Prentice-Hall, Inc., 1979.

[15]. J. Blanchette, “C++ GUI Programming with QT 4”, Prentice Hall, Segunda Edicion, 2008.

[16]. CISCO Systems, “IEEE 802.1Q Configuration”, CISCO Press, 2003. Disponible en http://doc.trecom.tomsk.su/CISCO/cc/td/doc/product/access/mar_3200/mar_conf/m511m80.htm.

[17]. M. Zhu, M. Molle, “Design and Implementation of Application-based Secure VLAN”, en Proceedings of the 29th Annual IEEE International Conference on Local Computer Network, 2004.

[18]. J. Barstch, G. Hillier, M. Hilzinger, J. Meixner, M. Nagorny, S. Olschner, M. Schäfer, J. D. Schmidt, A. Schnell, C. Schuszter, A. Schröter, R. Walter, “SUSE LINUX 9.1, Manual del Usuario”, Open SUSE, 2004, pp. 104-117.

[19]. CISCO Systems, “Configuring Routing Between VLANs with 802.1Q Encapsulation”, CISCO Press, 2006. Disponible en http://www.CISCO.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fswtch_c/swprt6/xcfvl80q.htm.

[20]. CISCO Systems, “CISCO Security Response: CISCO VLAN Trunking Protocol Vulnerabilities”, Document ID: 71306, 2006, disponible en http://www.CISCO.com/warp/public/707/CISCO-sr-20060913-vtp.shtml.

[21]. V. Heuven, “VLANs and GVRP on LINUX: quickly from specification to prototype”, 10th International LINUX System Technology Conference, 2003.

[22]. A. Ruiz, “Separando la red con VLANes”, Ministerio de Educación y Ciencia, 2007.

[23]. A. O. Camargo, L. C. Eleno, E. Palomares, “Diseño y Reconstrucción de LAN IIM, con uso de Firewalls, VLAN´s y WiFi”, Tesis de Licenciatura, UNAM 2006.

[24]. O. Hernández, “Implementación de VLAN en una red académica (UPIITA)”, Tesis de Licenciatura, IPN, 2005.

[25]. G. M. Rojas, L. F. López, “Análisis de Enrutamiento Dinámico y VLAN para interconectar Redes en Red”, Tesis de Licenciatura, UNAM, 2000.

[26]. M. D. González, “Metodología para la Implementación y Administración de Redes Virtuales de Área Local, Tesis de Maestría, IPN, 2007.

Anexo I

Glosario de términos

Administrador de Red: Persona a cargo de la operación, mantenimiento y administración de una red.

Ancho de banda: Cantidad de bits que pueden viajar por un medio físico (cable coaxial, par trenzado, fibra óptica, etc.) por unidad de tiempo, de forma que mientras mayor sea el ancho de banda más rápido se transmitirá la información.

Anjuta: Entorno de desarrollo integrado que permite programar en lenguajes tales como C, C++, java, python, en sistemas GNU/LINUX.

Broadcast (Difusión): Paquete de datos enviado a todos los nodos de una red.

CISCO: Empresa multinacional dedicada a la fabricación, venta, mantenimiento y consultoría de equipos de telecomunicaciones.

Concentrador: Dispositivo que permite centralizar el cableado de una red, así como ampliarla.

Conmutador: Dispositivo para interconectar dispositivos de red a alta velocidad de manera inteligente. Puede agregar ancho de banda, acelerar el tráfico de paquetes y reducir el tiempo de espera. Los conmutadores son más inteligentes que los concentradores *(hubs)* y ofrecen un ancho de banda más dedicado para los usuarios o grupos de usuarios. Un conmutador envía los paquetes de datos solamente a la computadora correspondiente, con base en la información que cada paquete contiene.

Encapsulamiento: Envoltura de datos en una cabecera de un protocolo en particular.

Enrutador: Dispositivo de capa de red que usa una o más métricas para determinar cuál es la ruta óptima a través de la cual se debe enviar el tráfico de red. Los enrutadores envían paquetes de una red a otra basándose en la información de capa de red.

IEEE (*Instituto de Ingeniería Eléctrica y Electrónica*): Organización profesional cuyas actividades incluyen el desarrollo de estándares de comunicaciones y redes. Los estándares de IEEE son los de mayor importancia para las redes de área local en la actualidad.

IEEE 802.1q: Estándar que fue definido en el RFC 2674 en el año de 1998, para especificar la implementación de las VLAN.

IOS (*Sistema Operativo de Interconexión*): Sistema operativo creado por CISCO Systems para programar y mantener equipos de interconexión de redes informáticas.

IP (*Protocolo de Internet*): Protocolo de capa de red del modelo TCP/IP que ofrece un servicio de interconexión de redes no orientado a conexión. El IP brinda funciones de direccionamiento, especificación del tipo de servicio, fragmentación y reensamblaje, y seguridad.

LAN (*red de área local*): Red de datos de alta velocidad y bajo nivel de errores que cubre un área geográfica relativamente pequeña (hasta unos pocos cientos de metros). Las LAN conectan estaciones de trabajo, periféricos, terminales y otros dispositivos en un solo edificio u otra área geográficamente limitada. Los estándares de LAN especifican el cableado y señalización en las capas físicas y de enlace de datos del modelo OSI. Ethernet, FDDI y Token Ring son tecnologías de LAN ampliamente utilizadas.

Latencia (*Latency*): Es el tiempo o lapso necesario para que un paquete de información se transfiera de un lugar a otro. La latencia, junto con el ancho de banda, son determinantes para la velocidad de una red.

Mac: (*Control de Acceso al Medio*): Parte de la capa de enlace de datos para controlar el uso del medio de transmisión; incluye una dirección de 6 bytes (48 bits) que comprenden 24 bits para el origen y 24 para el destino, así como el método para obtener permiso para transmitir.

Monodevelop: Entorno de desarrollo integrado para programar en C# y plataformas .NET.

OSI: (Modelo de *Sistema de Interconexión de Sistemas Abiertos*): Modelo de arquitectura de red desarrollado por ISO e UIT-T. El modelo está compuesto por siete capas (Física, Enlace de Datos, Red, Transporte, Sesión, Presentación y Aplicación), cada una de las cuales especifica funciones de red individuales, tales como el direccionamiento, el control de flujo, el control de errores, el encapsulamiento y la transferencia confiable de mensajes. **Ping: (*búsqueda de direcciones de Internet*)**: Mensaje de eco del protocolo ICMP y su respuesta. A menudo se usa en redes IP para probar el alcance de un dispositivo de red.

QT Creator: Entorno de desarrollo integrado que permite programar en lenguaje C, C++.

Red: Agrupación de computadoras, impresoras, enrutadores, conmutadores y otros dispositivos que se pueden comunicar entre sí a través de algún medio de transmisión.

Spanning Tree Protocol: Protocolo definido para evitar ciclos redundantes en una red de cómputo.

VLAN: (*LAN virtual*): Grupo de dispositivos de una LAN que están configurados (usando el software de administración) de tal modo que se pueden comunicar como si estuvieran conectados al mismo cable, cuando, en realidad, están ubicados en una serie de segmentos de LAN distintos.

Manual para generar el archivo ejecutable de la aplicación

Es necesario cumplir correctamente los pasos siguientes para poder generar el archivo ejecutable de la aplicación y hacer uso de la misma.

Para Open Suse 11.2

1.- Copiar la carpeta con nombre *interfaz1* en el directorio que el usuario desee.

2.- Instalar el paquete *build-essential* para poder compilar. Se puede instalar con *Zypper* o *yast*.

Comando a ejecutar:

zipper install build-essential

3.- Instalar las bibliotecas de QT incluidas en el paquete libqt4-devel.

Comando a ejecutar en la consola:

zypper install libqt4-devel

4.- Instalados estos dos paquetes es posible ejecutar los comandos necesarios, dirigiéndose al directorio donde el usuario coloco la carpeta llamada interfaz1, que contiene los archivos de la interfaz gráfica.

Ejecutar lo siguiente en una consola:

qmake –project

qmake

make

Con estos comandos se compila el proyecto y se genera el archivo ejecutable que permitirá abrir la interfaz gráfica, el cual estará dentro de la misma carpeta de los archivos.

5.- Ejecutar el siguiente comando para iniciar la aplicación:

./interfaz1

6.- Si el usuario desea entrar a la carpeta interfaz1 de forma gráfica, pude dirigirse al directorio y dar doble clic sobre el archivo interfaz1, de igual forma se ejecutará.

La interfaz gráfica se puede probar en otra distribución de LINUX. Si es el caso, se necesita buscar el nombre del paquete correspondiente a la distribución es decir las bibliotecas de QT y el compilador de C++; una vez cumplido esto el procedimiento es el mismo.

NOTA: Es necesario que la aplicación se ejecute con jerarquía de súper usuario, de esta forma se tendrá un buen funcionamiento.

Como crear una VLAN con la aplicación

1.- Ejecutar la aplicación

2.- Dar clic sobre la activación del módulo 802.1q.

3.- Si el usuario lo desea puede ver cuantas tarjetas de red tiene instaladas el sistema.

4.- Dar clic sobre el botón añadir/eliminar VLAN, con lo que se abrirá una nueva ventana.

5.- Si se tiene más de una tarjeta de red, seleccionar en cuál de ellas se desea asociar la VLAN, el sistema por defecto tiene seleccionada la eth0.

6.- Añadir el identificador numérico de la VLAN y dar clic en aceptar.

7.- Hasta este punto la VLAN ha sido agregada pero no se ha dado de alta, si se desea verificar que existe, basta con dar clic en el botón ver VLAN del equipo.

8.- Para configurar la dirección IP y la máscara de subred es necesario seleccionar la VLAN en la pantalla y continuar con la asignación de los parámetros, para después dar clic en aceptar; ahora el usuario puede ver la configuración de tarjetas de red, y la VLAN recién creada aparecerá con los datos asignados.

9.- Para cambiar los parámetros de la VLAN se selecciona la red y se capturan los nuevos parámetros que el usuario desee, posteriormente se da clic en aceptar y se verifica que los datos se hayan actualizado.

9.- Este mismo procedimiento se aplica para todas las VLAN que se desee agregar o modificar.

Recomendaciones sobre el uso de la interfaz gráfica

Se recomienda a los usuarios que hagan uso de la aplicación leer el siguiente apartado para evitar conflictos.

- Es necesario que el usuario cargue el módulo 8021q antes de realizar cualquier configuración, para evitar problemas en la comunicación
- El usuario deberá entrar a una sesión como usuario *root* o súper usuario, especialmente si desea ejecutar la aplicación desde consola, de lo contrario tendrá problemas para hacer la configuración.
- La interfaz gráfica tiene una ventana que se ejecuta al dar clic sobre el botón **añadir/eliminar VLAN,** una vez que el usuario haga configuraciones sobre ella es necesario mantenerla abierta por si se desea hacer modificaciones, ya que la interfaz no guarda las configuraciones de las VLAN, si esta se cierra se perderá la información pero las VLAN seguirán en el sistema.
- Para eliminar las VLAN y leer la información sobre una VLAN específica es necesario seleccionar la VLAN deseada, de lo contrario se puede generar un error en el sistema.
- Para pasar los parámetros de dirección IP y la máscara de subred el usuario necesita seleccionar la VLAN sobre la cual desea trabajar.

Anexo II

Código fuente

- **Clase allinfovlan**
- **Allinfovlan.cpp:**

```
#include "allinfovlan.h"
#include "ui_allinfovlan.h"
#include <QFile>
#include <QTextStream>

allinfovlan::allinfovlan(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::allinfovlan)
{
    ui->setupUi(this);
}
allinfovlan::~allinfovlan()
{
    delete ui;
}
void allinfovlan::on_verInformacion_clicked()
{
    QFile file_for_reading("/proc/net/vlan/config");
    file_for_reading.open(QIODevice::ReadOnly);
    QTextStream text_stream_for_reading(&file_for_reading);
    ui->textEdit->setText (text_stream_for_reading.readAll ());
    file_for_reading.close();
}
```

- **Allinfovlan.h**

```
#ifndef ALLINFOVLAN_H
#define ALLINFOVLAN_H
#include <QDialog>

namespace Ui {
    class allinfovlan;
}
class allinfovlan : public QDialog
{
    Q_OBJECT
public:
    explicit allinfovlan(QWidget *parent = 0);
    ~allinfovlan();
private:
    Ui::allinfovlan *ui;
private slots:
    void on_verInformacion_clicked();
};
#endif // ALLINFOVLAN_H
```

- **Clase Dialog**
- **Dialog.cpp**

```
#include "dialog.h"
#include "ui_dialog.h"
#include "widget.h"
#include <QDebug>
#include <QMessageBox>
#include <informacion.h>
#include <allinfovlan.h>
#include <infovlan.h>

int val;
int correctoa;
int correctob;

Dialog::Dialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Dialog)
{
    ui->setupUi(this);
}

Dialog::~Dialog()
{
    delete ui;
}
void Dialog::on_buttonBox_accepted()
{
    accept ();
}
void Dialog::on_buttonBox_rejected()
{
 reject();
}
void Dialog::asignarnum (int a)
{
    for (int i = 0; i <a ; i++ )
          ui->comboBox->addItem (" eth" + QString::number (i));  //+ " "+str1
}
void Dialog::on_comboBox_activated(int index)
{
    QMessageBox::information (this,"mensaje","SE HA SELECCIONADO LA INTERFAZ
eth"+QString::number (index));
    val=index;
}
void Dialog::on_aVlan_editingFinished()
{
    str1 = ui->aVlan->text ();
    if(str1=="0")
       QMessageBox::information (this,"Mensaje","Error");
}
void Dialog::on_aDirIp_editingFinished()
{
    str2 = ui->aDirIp->text ();
```

```
}
void Dialog::on_aNetmask_editingFinished()
{
   str3 = ui->aNetmask->text ();
}
void Dialog::on_anadirVlan_clicked() //agregar Id VLAN
{
   if(str1=="" || str1=="0")
      QMessageBox::information (this,"Mensaje","Ingresa un ID de la VLAN");
   else
   {
      agregar="vconfig add eth"+QString::number (val) +" "+ str1;
      const char *a = agregar.toAscii ().data ();
      correctoa = system(a);
      if(correctoa==0)
      {
          ui->listWidget->addItem (new QListWidgetItem("eth"+QString::number (val) +"."+ str1));
          ui->aVlan->clear ();
          QMessageBox::information (this,"Mensaje","VLAN eth"+QString::number (val)+"."+str1+"
"+"Agregada");
          str1="";
      }
      else
      {
         QMessageBox::information (this,"Aviso","El Dato del ID es Incorrecto Vuelve A Intentarlo");
         ui->aVlan->clear ();
      }
   }
}
void Dialog::on_aceptarIpNetmask_clicked()//configurar
{
   QString cvlan;
   QString space = " ";
   int bandera=0;
   if(str2=="" || str3=="")
      QMessageBox::information (this,"Mensaje","Ingresa Datos");
   else
   {
   QListWidgetItem *vl  = new QListWidgetItem(ui->listWidget);
   vl = ui->listWidget->currentItem ();
   cvlan = vl->data (Qt::DisplayRole).toString ();
   }
   QListIterator<QString> Iter(dirip);
   while(Iter.hasNext ())
   {
      QString var = Iter.next ();
      if(var==str2)
      {
       QMessageBox::information (this,"Mensje","La Direccion IP ya Fue Asgnada");
      bandera=1;
      }

   }
   if(cvlan=="")
      QMessageBox::information (this,"Mensaje","Selecciona Una VLAN");
```

```
    else if(bandera==0)
    {
    configurar = "ifconfig"+space+cvlan+space+str2+" "+"netmask"+" "+str3;
                const char *b = configurar.toAscii ().data ();
                correctob = system(b);
                if(correctob == 0)
                {
                   QMessageBox::information (this,"Mensaje","Los Datos Son Correctos");
                   dirip.append (str2);
                 }
                else
                {
                 QMessageBox::information (this,"Mensaje","Datos Incorrectos vuelve a Intentarlo");
                }
                ui->aDirIp->clear ();
                ui->aNetmask->clear ();
                str2="";
                str3="";
     }
}
void Dialog::on_eliminarVlan_clicked()//BORRAR
{
    QString rvlan="";
    QString espacio=" ";
    QListWidgetItem *item = new QListWidgetItem(ui->listWidget);
    item = ui->listWidget->currentItem ();
    rvlan =item->data (Qt::DisplayRole).toString ();
    if(rvlan=="")
       QMessageBox::information (this,"Mensaje","Selecciona una VLAN");
    else
    {
    eliminar = "vconfig rem"+espacio+rvlan;
    const char *e = eliminar.toAscii ().data ();
    system(e);
         delete item;
          QMessageBox::information (this,"Mensaje","Vlan"+espacio+ rvlan+" "+"Eliminada" );
      }
}
void Dialog::on_ifconfig_clicked()
{
    Informacion inf;
    inf.exec ();
}
void Dialog::on_verAllVlan_clicked()
{
    allinfovlan vlanequipo;
    vlanequipo.exec ();
}
void Dialog::on_verVlan_clicked()
{
    infovlan selectvlan;
    QString svlan;
    QString espacio=" ";
    QListWidgetItem *item = new QListWidgetItem(ui->listWidget);
    item = ui->listWidget->currentItem ();
```

```
    svlan =item->data (Qt::DisplayRole).toString ();
    if(svlan=="")
        QMessageBox::information (this,"Mensaje","Selecciona una VLAN");
    else
    {
    selectvlan.asignarvlan (svlan);
    selectvlan.exec ();
    }
}
```

- **Dialog.h**

```
#ifndef DIALOG_H
#define DIALOG_H
#include <QDialog>

namespace Ui {
    class Dialog;
}
class Dialog : public QDialog
{
    Q_OBJECT
public:
    QString str1;
    QString str2;
    QString str3;
    QString agregar;
    QString configurar;
    QString alta;
    QString eliminar;
    QList<QString> dirip;
    explicit Dialog(QWidget *parent = 0);
    ~Dialog();
    void asignarnum (int a);

private:
    Ui::Dialog *ui;

private slots:
    void on_aceptarIpNetmask_clicked();
    void on_aNetmask_editingFinished();
    void on_aDirIp_editingFinished();
    void on_verVlan_clicked();
    void on_verAllVlan_clicked();
    void on_ifconfig_clicked();
    void on_eliminarVlan_clicked();
    void on_anadirVlan_clicked();
    void on_aVlan_editingFinished();
    void on_comboBox_activated(int index);
    void on_buttonBox_rejected();
    void on_buttonBox_accepted();
};
#endif // DIALOG_H
```

- **Clase información**
- **Información.cpp**

```
#include "informacion.h"
#include "ui_informacion.h"
#include <QFile>
#include <QTextStream>
#include <QTextEdit>

Informacion::Informacion(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Informacion)
{
    ui->setupUi(this);
}
Informacion::~Informacion()
{
    delete ui;
}
void Informacion::on_verInformacion_clicked()
{
    system("ifconfig >ifconfig.txt");
        QFile file_for_reading("ifconfig.txt");
        file_for_reading.open(QIODevice::ReadOnly);
        QTextStream text_stream_for_reading(&file_for_reading);
        ui->textEdit->setText (text_stream_for_reading.readAll ());
        file_for_reading.close();
}
```

- **Información.h**

```
#ifndef INFORMACION_H
#define INFORMACION_H
#include <QDialog>

namespace Ui {
    class Informacion;
}
class Informacion : public QDialog
{
    Q_OBJECT
public:
    explicit Informacion(QWidget *parent = 0);
    ~Informacion();

private:
    Ui::Informacion *ui;

private slots:
    void on_verInformacion_clicked();
};
#endif // INFORMACION_H
```

- **Clase infovlan**
- **infovlan.cpp**

```
#include "infovlan.h"
#include "ui_infovlan.h"
#include <QDebug>
#include <QString>
#include <QMessageBox>
#include <QFile>
#include <QTextStream>

infovlan::infovlan(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::infovlan)
{
    ui->setupUi(this);
}

infovlan::~infovlan()
{
    delete ui;
}
void infovlan::asignarvlan (QString b)
{
dato=b;
}
void infovlan::on_verInformacion_clicked()
{
    selvlan = "/proc/net/vlan/"+dato;
    QFile file_for_reading(selvlan);
    file_for_reading.open(QIODevice::ReadOnly);
    QTextStream text_stream_for_reading(&file_for_reading);
    ui->textEdit->setText (text_stream_for_reading.readAll ());
    file_for_reading.close();
    QMessageBox::information (this,"Mensaje","Correcto");

}
```

- **Infovlan.h**

```
#ifndef INFOVLAN_H
#define INFOVLAN_H
#include <QDialog>

namespace Ui {
    class infovlan;
}
class infovlan : public QDialog
{
    Q_OBJECT

public:
    explicit infovlan(QWidget *parent = 0);
    ~infovlan();
```

```
    void asignarvlan(QString b);
    QString dato;
    QString selvlan;

private:
    Ui::infovlan *ui;

private slots:
    void on_verInformacion_clicked();
};
#endif // INFOVLAN_H
```

- **Clase widget**
- **Widget.cpp**

```
#include "widget.h"
#include "ui_widget.h"
#include <QMessageBox>
#include <stdio.h>
#include <stdlib.h>
#include <QString>
#include <QDebug>
#include <dialog.h>

QString qStr;
Widget::Widget(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Widget)
{
    system("ifconfig> InfTarjetas.txt");
    system("grep -c eth InfTarjetas.txt>NTarjetas.txt");
    FILE *archivo;
          char caracteres[10];
          int numero;
          archivo = fopen("NTarjetas.txt","r");
          if (archivo == NULL)
                exit(1);
          printf("\nEl contenido del archivo de prueba es \n\n");
          while (feof(archivo) == 0)
          {
                fgets(caracteres,10,archivo);
                if(feof(archivo)==0)
                printf("%s\n\n",caracteres);
                //fclose(archivo);
          }
          printf("%c\n\n",caracteres[0]);
          numero=caracteres[0];
          num=numero-'0';
          printf("TIENES %d TARJETAS DE RED \n\n ",num);
          qStr = QString::number(num);
    ui->setupUi(this);
   QImage image1("CIDETEC.jpg");
   QImage image2("POLITECNICO.jpg");
    ui->imagenCidetec->setPixmap (QPixmap::fromImage (image1));
```

```
    ui->imagenIpn->setPixmap (QPixmap::fromImage (image2));
}
Widget::~Widget()
{
    delete ui;
}
void Widget::on_activaModulo_clicked()
{
 int correcto;
 correcto = system("modprobe 8021q");
QMessageBox::information (this,"mensaje","Modulo 802.1Q Activo");
}
void Widget::on_salir_clicked()
{
    close();
}
void Widget::on_muestraTarjetas_clicked()
{
      QMessageBox::information(this,"Mensaje",qStr + " Tarjetas de Red Instaladas");
}

void Widget::on_agregarEliminar_clicked()
{
    if(num==0)
    QMessageBox::information(this,"Aviso","No existen Tarjetas De Red En EL Equipo");
     else
    {
    Dialog tar;
    tar.asignarnum (num);
    tar.exec ();
    }
}
```

- **Widget.h**

```
#ifndef WIDGET_H
#define WIDGET_H
#include <QWidget>

namespace Ui {
    class Widget;
}
class Widget : public QWidget
{
    Q_OBJECT

public:
    int num;
    explicit Widget(QWidget *parent = 0);
    ~Widget();

private:
    Ui::Widget *ui;
```

```
private slots:
    void on_agregarEliminar_clicked();
    void on_muestraTarjetas_clicked();
    void on_salir_clicked();
    void on_activaModulo_clicked();
};
#endif // WIDGET_H
```

- **Programa principal**
- **main.cpp**

```
#include <QtGui/QApplication>
#include "widget.h"
#include "dialog.h"
#include <QSplashScreen>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Widget w;
    w.show();
    return a.exec();
}
```

- **Recursos.qrc**

```
<RCC>
    <qresource prefix="/new/prefix1">
        <file>CIDETEC.jpg</file>
        <file>POLITECNICO.jpg</file>
    </qresource>
</RCC>
```