MECANISMOS DE BALANCEO DE CARGA EN MPLS CON RSVP-TE Y OSPF

ANEXOS

WILMAR RAFAEL MUÑOZ BRAVO
MARCO ANTONIO TRUJILLO CASTRILLÓN

Director:
Ing. Oscar J. Calderón C.

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
GRUPO I + D NUEVAS TECNOLOGÍAS EN TELECOMUNICACIONES
DEPARTAMENTO DE TELECOMUNICACIONES
POPAYÁN
2006

CONTENIDO

ANEXO A: Código del Mecanismo DLCCM

A.1 Funciones de Asignación, División-Clasificación, Balanceo y Control

Las funciones de Asignación, División-Clasificación, Balanceo y Control están implementadas en el archivo "lbmplsV01.sh", el cual se ejecuta en el LER1, siempre que los LSPs estén previamente configurados. El código se muestra a continuación.

```
#!/bin/bash
ingres="0xa2"    # Identificador del LSP creado desde LER1->LER2
egres="0xa1"     # Identificador del LSP creado desde LER2->LER1
report=/root/bin/report

#Aumentar el número de conexiones por defecto que soporta conntrack
echo "16344" > /proc/sys/net/ipv4/ip_conntrack_max

#Variables Globales
# Siempre Max1 debe ser mayor o igual a Max2.
numConMov=0 # Numero de conexiones a mover
mlsp=1   # LSP1
ulsp=2    # LSP2
numMapLines=0
numUmapLines=0
numReMapLines=0
PERIODO=30
Max1=1000   #1M
Max2=1000   #1M
C=`echo "$Max1 * 90 / 100" | bc`    #Umbral C1=90%MaxBW
C2=`echo "$Max2 * 90 / 100" | bc`  #Umbral C2=90%MaxBW
M=`echo "$Max1 * 70 / 100" | bc`    #Umbral M=70%MaxBW
holdtime=12      # Holdtime por default
vacearLs=0       # Para acelerar el proceso de mover toda la carga del Ls -> Lp
export numConMov mlsp ulsp numMapLines numUmapLines numReMapLines PERIODO Max C M
holdtime vacearLs
```

```
# Limpiar registros temporales.
rm -rf register_LSP1
rm -rf register_LSP2


# Esta función es un componente de la función de colección que accede a la base de datos de
#OSPF-TE,  calcula la tasa de transmisión de datos asociada a cada enlace,  los compara con las
#velocidades calculadas  por la función de colección local y extrae el valor máximo, que posterior
#mente utiliza  algoritmo de balanceo LCM-M.

readtun(){
# Limpiar registros temporales.
rm -rf max_reserva
rm -rf db_tun_lsp
rm -rf max_bw_if
touch max_reserva
touch db_tun_lsp
touch max_bw_if

# Crear el patrón de búsqueda que se utiliza para filtrar la información relevante de la base de
datos de OSPF-TE
echo “Router-Address:” > patron
echo “Link-ID:” >> patrón
echo “Maximum Bandwidth:” >> patron
echo “Maximum Reservable Bandwidth:” >> patron
echo “Resource class/color:” >> patron


# Calcular las velocidades locales para cada tunnel
tunnel -L | grep eth | awk 'BEGIN {OFS=":"} {print $2, $8 }' | cut -d ")" -f 1 | sort +1nr > tunnel.txt
echo "test1"
numtun=`cat tunnel.txt | wc -l`
echo "test2"
vtysh -c "show ip ospf database opaque-area" > dbospf.txt        #  Leer la base de datos de OSPF-
TE
grep -f patron dbospf.txt > dbospf1.txt      # Filtrar información relevante (Link-ID, Sub-TLVs TE)
numline=`grep -f patron dbospf.txt | wc -l` # numero de lineas totales de archivo dbospf.txt
num=`echo "$numline / 5 " | bc`                # Nuemro total de enlaces tramo a tramo.
      for i in `seq 1 $numtun`; do
```

```
        echo "Numero de tuneles $numtun"
        echo "Numero del tunel a evaluar $i"
                lsp_id[$i]=`head -n $i tunnel.txt | tail -n 1 | cut -d ":" -f 1`
                nameIftun[$i]=`head -n $i tunnel.txt | tail -n 1 | cut -d ":" -f 2`
                export lsp_id nameIftun
                echo "lsp_id ${lsp_id[$i]}, nameIftun ${nameIftun[$i]}"
                tmp1=`head -n $i file | tail -n 1` #Los datos se extraen del rtest2 file, La primera linea
                lspidtmp=`echo $tmp1 | cut -d " " -f 3` # Se extrae el lsp_ip
                if [ ${lsp_id[$i]} -eq $lspidtmp ]; then
                        path=`echo $tmp1 | awk '{print $6 }' | awk 'BEGIN {FS=":0"} { print $1 }' | sed s/:/" "/g`
                fi

                for j in $path; do # A partir del numero de saltos calcular el numero de enlaces por salto
                        #Adecua el dato para ser comparado con lo que se extrae de la dbospf
                        link_rtest=`echo $j | awk 'BEGIN {FS="."} {print $1, $2, $3}' | tr " " "."`
                        echo "Esto es el tramo a analizar $path"
                        echo "Son los saltos modificados en tramos de enlaces $link_rtest"
                        for k in `seq 1 $num`; do # Iterar por el numero de grupos de 5 lineas extraidas de la
db-ospf
                                numline=`echo "$k * 5" | bc` # Mostrar grupos de lineas multiplos de 5
                                head -n $numline dbospf1.txt | tail -n 5 > tmp1.txt # Leer siempre las ultimas 5
lineas
                                link_id=`grep Link-ID: tmp1.txt | awk 'BEGIN {FS="."; OFS="."} {print $1, $2,
$3}'`
                                link_ospf=`echo $link_id | awk 'BEGIN {FS=":"} {print $2}'`
                                echo "Link_id de ospfdb $link_ospf"
                                echo "Link_id de rtest $link_rtest"
                                if [ $link_rtest = $link_ospf ]; then
                                        id_tx=`cat tmp1.txt | grep -f patron3 | awk 'BEGIN {FS=":"} {print $2}'`
                                if [ $id_tx = $ingres ]; then
                                        grep -f patron1 tmp1.txt > tmp2.txt
                                        max_bw_if=`head -n 1 tmp2.txt | awk 'BEGIN {FS=":"} {print $2}' | cut -d "("
-f 1`
                                        max_rate=`tail -n 1 tmp2.txt | awk 'BEGIN {FS=":"} {print $2}' | cut -d "(" -f
1`
                                        echo "he encontrado el link $link_ospf con velocidad $max_rate"
                                        echo $max_rate  >> max_reserva_$i # Ancho de banda utilizado para lsp
$i
                                        echo $max_bw_if >> max_bw_if_$i
```

```
                        unset max_rate
                        unset max_bw_if
                    fi
                fi
                done
            done
            echo "Terminando el Camino nuemro $i"
        done

            echo "hacer lecturas locales.."
        echo "Numero de tuneles $numtun"

            for i in `seq 1 $numtun`; do
        echo "Numero del tunel a evaluar $i"
            if [ ! -s $report/${nameIftun[$i]} ] ; then
                echo "Los datos no estan listos....waiting 1 seg"
                sleep 1; #Esperar 1 segundos.
                echo "....OK"
            fi
                 #adicionar la velocidad local o del primer tramo del Tunnel
            ratetunn[$i]=`cat $report/${nameIftun[$i]}` #Velocidad del tunel local.
           echo "lsp_id ${lsp_id[$i]}, nameIftun ${nameIftun[$i]}, ratetunn ${ratetunn[$i]}"
           #adicionar la velocidad local o del primer tramo del Tunnel
            echo ${ratetunn[$i]} >> max_reserva_$i  # Adicionar la velocidad del primer tramo.
            max_bw_rsv_lsp=`cat max_reserva_$i | sort -nr | head -n 1` # Se extrae la carga mayor
            echo ${lsp_id[$i]}:$max_bw_rsv_lsp >> db_tun_lsp  # Base de datos de todos los tuneles
lsp.
            max_bw_if=`cat max_bw_if_$i | sort -n` # Se extrae la carga menor, para fijar M y C
            #echo "Base de datos parcial ${link_id[$i]}:$max_bw_rsv_lsp"
            db_parcial=`cat db_tun_lsp`
            echo "base de datos parcial $db_parcial"
            rm -rf max_reserva_$i
            rm -rf max_bw_if_$i
            touch max_reserva_$i
            touch max_bw_if_$i
           done
           database=`cat db_tun_lsp`
            echo Base de datos final $database
```

```
}


# La función “dctrafico” implementa las funciones de división de tráfico por flujos y lo clasifica acorde a 5 parámetros que son:
# La dirección IP origen, dirección IP destino, puerto origen, puerto destino y tipo de protocolo, y los almacena en
# registros temporales que son utilizados luego por la función de Asignación y la función de Balanceo.

dctrafico(){
        echo "escaneando flujos..."
    #grep tcp /proc/net/ip_conntrack | awk '{print $1, $5, $6, $7, $8 }' | sed /src=127.0.0.1/d | uniq > tcp_flowtmp
    #sed -e 's/src=//g' -e 's/dst=//g' -e 's/sport=//g' -e 's/dport=//g' -e '/192.168.20[1-5]/d' tcp_flowtmp >  tcp_flow
    grep udp /proc/net/ip_conntrack | awk '{print $1, $4, $5, $6, $7 }' | sed /src=127.0.0.1/d  | uniq > udp_flowtmp
        sed -e 's/src=//g' -e 's/dst=//g' -e 's/sport=//g' -e 's/dport=//g' -e '/192.168.20[1-5]/d' udp_flowtmp > udp_flow
        cat udp_flow > TotalFlowtmp
        #cat tcp_flow >> TotalFlowtmp
        cat TotalFlowtmp | sort > TotalFlow
        touch register_LSP1
        touch register_LSP2
        if [ -s register_LSP1 ] || [ -s register_LSP2 ]; then
                        cat register_LSP2 | sort > register_LSP2tmp
                        cat register_LSP2tmp > register_LSP2
                        cat register_LSP1 | sort > register_LSP1tmp
                                cat register_LSP1tmp > register_LSP1
                        diff -w TotalFlow register_LSP2 | grep \> | sed s/\>//g | sort | uniq > umapLinesLSP2tmp
                        diff -w TotalFlow register_LSP1 | grep \> | sed s/\>//g | sort | uniq > umapLinesLSP1tmp
                        cat umapLinesLSP2tmp | sed s/^[^a-z]//g > umapLinesLSP2
                        cat umapLinesLSP1tmp | sed s/^[^a-z]//g > umapLinesLSP1
                        diff -w register_LSP2 umapLinesLSP2 | grep \< | sed s/\</g | sort | uniq > register_LSP2org
```

```
                diff -w register_LSP1 umapLinesLSP1 | grep \< | sed s/\</g | sort | uniq >
register_LSP1org
                cat register_LSP2org | sed s/^[^a-z]//g > register_LSP2
                cat register_LSP1org | sed s/^[^a-z]//g > register_LSP1
                diff -w TotalFlow register_LSP2 | grep \< | sed s/\</g | sort | uniq >
TotalesNoLSP2
                cat TotalesNoLSP2 | sed s/^[^a-z]//g > newCon
                diff -w newCon register_LSP1 | grep \< |  sed s/\</g > TotNoLSP2NoLSP1
                cat TotNoLSP2NoLSP1 | sed s/^[^a-z]//g | sort > register_LSP1New
                # Verificar las conexiones existentes
                echo "Adicionando lineas"
                cat register_LSP1New >> register_LSP1       # Adicionar las nuevas
conexiones al final
                cat register_LSP1 | sed s/^[^a-z]//g > register_LSP1tmp # Borrar espacios
al inicio de linea
                cat register_LSP1tmp | sort | uniq > register_LSP1 # ordenar
                for i in `seq 1 2`; do # Numero de LSPs
                        echo "Eliminar las conexiones del lsp$i.."
                        cat umapLinesLSP$i > umapLines
                        ulsp=$i
                        numUmapLines=`cat umapLines | wc -l`
                        echo "Numero de conexiones eliminar: $numUmapLines"
                        umaping
                done
                numMapLines=`cat register_LSP1New | wc -l`
                echo "numMapLines $numMapLines; si numMapLines !=0 => hay nuevas
conexiones"
                mlsp=1
                cat register_LSP1New > mapLines
                maping
        #fi
    else
        echo "register_LSP1 y LSP2 Estan vacios, Mapear todas las conexiones a
register_LSP1"
        echo "Llamar initmark....OK"
        initmark
    fi

}
```

```
initTables(){
        echo "Iniciar inittables"
        iptables -t mangle -F
        tunnel -m -s 192.168.200.188/32 -d 192.168.206.1/32 -p icmp -l 200
        tunnel -m -s 192.168.200.185/32 -d 192.168.206.1/32 -p icmp -l 201
}

initmark(){
        echo "Entrando a Init mark"
        src_addr=0.0.0.0/0
        dst_addr=192.168.207.0/24
        mpls=1 #- Define el LSP por defecto
        cat TotalFlow > register_LSP1
        cat register_LSP1 | sed s/^[^a-z]//g > register_LSP1tmp # Borrar espacios al inicio de linea
        cat register_LSP1tmp > register_LSP1
        touch register_LSP2
        cat register_LSP1 > mapLines   # En este caso mapeara por defecto todas las conexines al
LSP1
        numMapLines=`cat register_LSP1 | wc -l`
                mlsp=1  # Mapear al LSP1
                maping
        echo "Saliendo de initMark"
}



# Las funciones maping, umaping y remaping conforman la función de Asignación, se encargan de
hacer el mapeo, la eliminación
#  y la re-asignacion de FECs sobre los los LSPs, esta función es llamada por el control cuantas
veces lo requiera
# la función de balanceo

maping(){
if [ ! $numMapLines -eq 0 ];then
      for i in `seq 1 $numMapLines`; do         # Se itera hasta que mueve todas las conexiones
            connection=`head -n $i mapLines | tail -n 1`
            echo "Mapear la $connection Numero $i"
            proto=`echo $connection | cut -d " " -f 1`
```

```
            s_ip=`echo $connection | cut -d " " -f 2`
            d_ip=`echo $connection | cut -d " " -f 3`
            s_port=`echo $connection | cut -d " " -f 4`
            d_port=`echo $connection | cut -d " " -f 5`
iptables -t mangle -I PREROUTING -s $s_ip -d $d_ip -p $proto --sport $s_port --dport $d_port -j MARK --set-mark $mlsp
iptables -t mangle -I OUTPUT -s $s_ip -d $d_ip -p $proto --sport $s_port --dport $d_port -j MARK --set-mark $mlsp

        done
fi
}

umaping(){
echo "Entrando a umaping....."
if [ ! $numUmapLines -eq 0 ];then
echo "Entrando a realizar umaping..."
        for i in `seq 1 $numUmapLines`; do        # Se itera hasta que mueve todas las conexiones
            connection=`head -n $i umapLines | tail -n 1`
            echo "UMapear la $connection Numero $i"
            proto=`echo $connection | cut -d " " -f 1`
            s_ip=`echo $connection | cut -d " " -f 2`
            d_ip=`echo $connection | cut -d " " -f 3`
            s_port=`echo $connection | cut -d " " -f 4`
            d_port=`echo $connection | cut -d " " -f 5`
iptables -t mangle -D PREROUTING -s $s_ip -d $d_ip -p $proto --sport $s_port --dport $d_port -j MARK --set-mark $ulsp
iptables -t mangle -D OUTPUT -s $s_ip -d $d_ip -p $proto --sport $s_port --dport $d_port -j MARK --set-mark $ulsp

        done
echo "....umaping....OK"
fi
}

remaping(){
echo "Entrando a remaping..."
if [ ! $numReMapLines -eq 0 ];then
        for i in `seq 1 $numReMapLines`; do        # Se itera hasta que mueve todas las conexiones
```

```
		connection=`head -n $i remapLines | tail -n 1`
		echo "ReMapear la $connection Numero $i"
		proto=`echo $connection | cut -d " " -f 1`
		s_ip=`echo $connection | cut -d " " -f 2`
		d_ip=`echo $connection | cut -d " " -f 3`
		s_port=`echo $connection | cut -d " " -f 4`
		d_port=`echo $connection | cut -d " " -f 5`
iptables -t mangle -D PREROUTING -s $s_ip -d $d_ip -p $proto --sport $s_port --dport $d_port -j MARK --set-mark $ulsp
iptables -t mangle -D OUTPUT -s $s_ip -d $d_ip -p $proto --sport $s_port --dport $d_port -j MARK --set-mark $ulsp
iptables -t mangle -I PREROUTING -s $s_ip -d $d_ip -p $proto --sport $s_port --dport $d_port -j MARK --set-mark $mlsp
iptables -t mangle -I OUTPUT -s $s_ip -d $d_ip -p $proto --sport $s_port --dport $d_port -j MARK --set-mark $mlsp

		done
echo ".....remaping...OK"
fi
}


# Es la función encargada de ejecutar el control de mecanismo DLCCM, determina el momento que debe llamar la función
# de balanceo, como también el sentido que debe tener el balanceo de carga, si es del primario al secundario o viceversa.

control(){
#Si control es '0' P->S
#Si control es '1' S->P
control=0
deltaBW=0
vacearLs=0 # Por defecto esta deshabilitado
Min=`echo "$Max1 * 1 / 100" | bc` # Minimo 20Kbsp
export control deltaBW bwTCP bwUDP bwTCP bwUDP
#consutal estado de los LSPs
numLSP=`cat db_tun_lsp | wc -l`
for i in `seq 1 $numLSP`; do
		head -n $i db_tun_lsp > db_tun_lsp1
```

```
        LSP[$i]=`tail -n 1 db_tun_lsp1`
        echo "LSP_ID + Velocidad ${LSP[$i]}"
        export LSP
done
var1=`echo "${LSP[1]}" | cut -d ":" -f 2`
var2=`echo "${LSP[2]}" | cut -d ":" -f 2`
var3=`echo "$var2 <= $Min" | bc` #20K
if [ $var3 -eq 1  ];then
        var2=0;
fi
Lp=$var1
Ls=$var2
echo "esta es var1 $var1 y el de var2: $var2"
export Lp Ls
unset result
result=1
echo "var1=Lp $var1"
echo "var2=Ls $var2"
result=`echo "$var1 > $var2" | bc`
echo "result=$result; si result=1 => Lp > Ls ; Si result=0 => Lp < Ls"
        if [ $result -eq 1 ]; then
                echo "Lp $Lp >= Ls $Ls...OK"
                unset result
                result=`echo "$var1 >= $M" | bc`
                if [ $result -eq 1 ]; then
                        echo "Lp $Lp >= M $M"
                        unset result
                        result=`echo "$var1 <= $C" | bc`
                        if [ $result -eq 1 ]; then
                                echo "Lp $Lp <= C $C"
                                echo "No hacer nada, porque esta en trafico normal"
                        else
                        control=0 # BalanceoC
                        echo "BalanceoC"
                        echo "Lp $Lp > C $C"
                        deltaBW=`echo "$Lp - $C" | bc`
                        echo "deltaBW $deltaBW"
                        echo "Llamar balanceo"
                        balanceo
```

```
                fi
        else                    # BalancearM
                echo "BalancearM"
                echo "LP < M"
                if [ ! $Ls -eq 0 ]; then
                        deltaBW=`echo "$M - $Lp" | bc`
                        deltaTMP=`echo "$deltaBW <= $Ls" | bc`
                        if [ ! $deltaTMP -eq 1  ];then
                                deltaBW=$Ls
                        fi
                        control=1
                        echo "Llamar balanceo"
                        balanceo
        fi      fi
else
        echo "Lp No  es Mayor que Ls"
        if [ $Ls -ge $C2 ] && [ $Lp -gt $M ]; then
                echo "Estado de congestion para Ls"
        else
                unset result
                suma=`echo "$var1 + $var2" | bc`
                echo "Lp + Ps = $suma"
                result=`echo "$suma < $C" | bc`
                echo "Lp + Ls ($suma) < C $C"
                if [ $result -eq 1 ]; then
                        echo "VacearS"
                        control=1       # VacearS
                        deltaBW=$Ls
                        vacearLs=1
                        echo "VacearS"
                        echo "este es el varlor de $Ls"
                        if [ ! $suma -eq 0 ];then
                                balanceo
                        else
                                echo "No hacer nada porque no hay trafico suma es cero"
                        fi
                else
                        echo "BalanceoR"
                        echo "Lp $Lp + Ls $Ls No es < C $C"
```

```
                                control=1          # BalanceoR
                                deltaBW=`echo "($Ls - $Lp)/2" | bc`
                                balanceo
                        fi
                fi
        fi

}

readrateLSP(){
        if [ ! -s $report/enable ]; then
                echo "Los datos de las interfaces no estan listos..."
                echo "waiting...2 seg"
                sleep 2 ;
                echo "...OK"
        fi
        currentRateLSP1=`cat $report/T11680eth1`
        currentRateLSP2=`cat $report/T11680eth2`
        echo "Lectura del las interfaces: T11680eth1=$currentRateLSP1;
T11680eth2=$currentRateLSP2"
        export currentRateLSP2 currentRateLSP1
}

# Ejecuta los procesos de balanceo de carga.

balanceo(){
#Se asume un ancho de banda de 10Kbp de una conexion tcp y 20Kbs de una udp
state=1         # Finaliza el balanceo cuando "state=0"
inicio=0# Inicio por defecto esta en cero.
while [ $state -eq 1 ];do
readrateLSP # Funcion que lee la velocidad del LSP local
echo "Velocidad LSP1 $currentRateLSP1 Velocidad LSP2 $currentRateLSP2"
if [ $control -eq 0 ];then # Si => P->S
        echo "Balancear P->S"
        cat register_LSP1 > register
        if [ $inicio -eq 0 ]; then
                rateInit=$currentRateLSP1
                inicio=1
        fi
```

```
        mlsp=2
        ulsp=1
else
        echo "Balancear Ls->Lp"
        cat register_LSP2 > register
        if [ $inicio -eq 0 ]; then
                rateInit=$currentRateLSP2
                inicio=1
        fi
        mlsp=1
        ulsp=2
fi
        numConMov=0 # Por defecto el numero de conexiones a mover es cero
        numConT=`cat register | wc -l`
        numConMov=1;            #  Define el balanceo uno a uno
        if [ $vacearLs -eq 1 ]; then # Acelarar el proceso VacearS
                numConMov=$numConT
        fi
        echo "Numero de connections a balancear $numConMov"
        head -n $numConMov register > remapLines
        numConRes=`echo "$numConT - $numConMov" | bc`  # El Numero de conexiones Totales
- Movidas = Restantes
        tail -n $numConRes register > register_LSPRes
        numReMapLines=`cat remapLines | wc -l`
        echo "Llamando remaping...."
        remaping        # Llamar remaping
        echo ".....Remaping...OK"


        if [ $control -eq 0 ];then # Si => P->S; con el fin de borrar las conexiones movidas de Uno y
Pasarlas al Otro
                cat register_LSPRes > register_LSP1  # Actualiza El registro del LSP1
                cat remapLines >> register_LSP2 # Adiciona al final las conexiones movidas al
registro del LSP2
                cat register_LSP1 | sed s/^[^a-z]//g | sort > register_LSP1tmp  #Borrar espacion al
inicio de linea
                cat register_LSP2 | sed s/^[^a-z]//g | sort > register_LSP2tmp  # Y ordenar.
                cat register_LSP1tmp > register_LSP1
                cat register_LSP2tmp > register_LSP2
```

```
            sleep $holdtime     # Esperar 8 segundos
                echo "Esperar $holdtime seg...."
        # Comparar si el balanceo fue exitoso
        readrateLSP     # Lear los la velocidad en la interfaz local
                echo "Velocidad inicial $rateInit Velocidad final $currentRateLSP1"
                #eval=`echo "$rateInit >= $currentRateLSP1" | bc`
                if [ $rateInit -ge $currentRateLSP1 ]; then
                deltaBWtmp=`echo "$rateInit - $currentRateLSP1" | bc` # Comparar si el balanceo
fue exitoso
                        echo "Delta actual $deltaBWtmp"
                resultado=`echo "$deltaBWtmp >= $deltaBW" | bc`
                        echo " deltaBWtmp $deltaBWtmp >= deltaBW $deltaBW"
                        echo "resultado $resultado"
                    NoConexionesLSP1=`cat register_LSP1 | wc -l`
                         if [ $NoConexionesLSP1 -le 0 ];then
                     echo "Velocidad del LSP1 ha llegado ha cero"
                     resultado=1
                fi

                else
                resultado=0
                deltaBWtmp=$deltaBW
                fi
        else
                #
                cat register_LSPRes > register_LSP2  # Actualiza El registro del LSP2
          cat remapLines >> register_LSP1 # Adiciona al final las conexiones movidas al registro
del LSP1
                cat register_LSP2 | sed s/^[^a-z]//g | sort > register_LSP2tmp
                cat register_LSP1 | sed s/^[^a-z]//g | sort > register_LSP1tmp
          cat register_LSP1tmp > register_LSP1
          cat register_LSP2tmp > register_LSP2
                sleep $holdtime         # Espera 2 segundos
                echo "Esperar $holdtime seg"
          # Comparar si el balanceo fue exitoso
                readrateLSP     # Lear los la velocidad en la interfaz local
                echo "Velocidad inicial $rateInit Velocidad final $currentRateLSP2"
                #eval=`echo "$rateInit >= $currentRateLSP2" | bc`
                if [ $rateInit -ge $currentRateLSP2 ]; then
```

```
                deltaBWtmp=`echo "$rateInit - $currentRateLSP2" | bc`   # Comparar si el balanceo
fue exitoso
                resultado=`echo "$deltaBWtmp >= $deltaBW" | bc`
                        echo "Delta actual $deltaBWtmp"
                echo " deltaBWtmp $deltaBWtmp >= deltaBW $deltaBW"
                echo "resultado $resultado"
                NoConexionesLSP2=`cat register_LSP2 | wc -l`
                if [ $NoConexionesLSP2 -le 0 ];then
                        #if [ $currentRateLSP2 -le 10 ];then
                                echo "Velocidad del LSP2 ha llegado ha cero"
                                resultado=1
                        fi

                else
                echo "EL balanceo no fue exitoso"
                resultado=0
          deltaBWtmp=$deltaBW
                fi
        fi

        if [ $resultado -eq 1  ];then
                state=0         # Fin del balanceo
      fi
done    # Fin del algorimo de balanceo

}

#- Main
dctrafico
initTables
#initmark
while :
        do
        dctrafico
        readtun
        control
        echo "Balaceo completado...Esperar $PERIODO seg"
        sleep $PERIODO
done
```

El proceso de balanceo puede ser ejecutado de la siguiente manera:

[root@LSR1 bin]# ./lbmplsV01.sh &>/dev/null

A.2 Función de Colección LER de Ingreso

La función de Colección “fcoleccionIngressV01.sh” es implementada en el LER1, tiene diferencias respecto a la implementa en los LSRs, debido que esta función únicamente debe hacer al calculo de la tasa de transmisión de datos de cada uno de los LSPs configurados y almacenarlo en un archivo de texto, que es accedido por la función de control cuando lo requiere la función de balanceo.

```
#!/bin/bash
report=/root/bin/report
database=/proc/net/dev
interfaces[1]=0
interfaces[2]=0
interfacesOld[1]=0
interfacesOld[2]=0
touch $report/enable
if [ ! -d $report ];then
        mkdir $report
fi
#Se escaneara la velocidad de los tuneles LSP
scanIf() {
rm $report/enable # Bloquear la lectura de los datos
numIf=`cat /proc/net/dev | cut -d ":" -f 1 | awk '/T11680eth/ {print $1}' | wc -l`
if [ $numIf -gt 0 ];then
    for i in `seq 1 $numIf`; do
        #Leer interfaces
        interfaces[$i]=`grep eth$i $database | cut -d ":" -f 2 | awk '{ print $9 }' | head -n 1`
        velocidad=`echo " 8 * (${interfaces[$i]} - ${interfacesOld[$i]}) / 10000 " | bc`
        interfacesOld[$i]=${interfaces[$i]}  # Guardar los bytes para el proximo reporte
        echo "$velocidad" > $report/T11680eth$i
            echo "$velocidad" >> $report/report_T11680eth$i
```

```
        done
fi
echo "1" > $report/enable # Habilita la lectura de los datos.
}


## Main Program ##
while :
            do
            scanIf
            sleep 10
done
```

Para que la función de colección en el LER1 de ingreso se active automáticamente al inicio del sistema se debe crear un archivo de inicio llamado “fcolecciond” dentro del directorio “/etc/init.d/”, cuyo contenido debe ser:

```
#!/bin/bash
start(){
        echo "staring colector"
        sh /root/bin/fcoleccionIngressV01.sh&
}
stop(){
        echo "stoping colector"
}
case "$1" in
        start)
                start
        ;;
        stop)
                stop
        ;;
        *)
        echo $"Usage: $0 {start|stop}"
        exit 1
esac
exit 0
```

A.3 Función de Colección y Distribución en un LSRs

Las funciones de Colección y Distribución están implementadas en el archivo "fcoleccionCoreV01.sh" en el Core de la red MPLS, la cual tiene la función de realizar el calculo de la tasa de transmisión de datos de salida en cada una de las interfaces de un LSR y almacenarlo en un archivo de texto que es accedido por la función de distribución control cuando lo requiere la función de balanceo.

```
#!/bin/bash
#Version final
report=/root/bin/report
database=/proc/net/dev
interfaces[0]=0             #Velocidad inicial
interfaces[1]=0             #Velocidad inicial
interfacesOld[0]=0
interfacesOld[1]=0
delta=$[10000000*5/10000]   #Solo difundir LSAs Opacos si es mayor al delta (5Kbps)
if [ ! -d $report ];then
        mkdir $report
fi
scanIf() {
numIf=`cat /proc/net/dev | cut -d ":" -f 1 | awk '/eth/ {print $1}' | wc -l`
numIf=`echo "$numIf - 1" | bc`
        for i in `seq 0 $numIf`; do
                #Leer interfaces
                interfaces[$i]=`grep eth$i $database | cut -d ":" -f 2 | awk '{ print $9 }'`
                velocidad=`echo " 8 * (${interfaces[$i]} - ${interfacesOld[$i]}) / 10000 " | bc`
                interfacesOld[$i]=${interfaces[$i]}  # Guardar los bytes para el proximo reporte
                echo "$velocidad" > $report/rate_eth$i
                #difundir=`echo "$velocidad >= $delta" | bc`
                echo "conf t " > $report/opaque_lsa
                echo "interface eth$i" >> $report/opaque_lsa
                echo "mpls-te link max-rsv-bw $velocidad" >> $report/opaque_lsa
                #if [ $difundir -eq 1 ];then #Difundir LSA-TE Opaco
                        /usr/local/bin/vtysh < $report/opaque_lsa &>/dev/null
                #fi
        done
}
```

```
## Main Program ##
while :
        do
        scanIf
        sleep 10
        echo "`date`"
done >> $report/logfile
```

Para que la función de colección se active automáticamente al inicio del sistema se debe crear un archivo de inicio llamado “fcolecciond” dentro del directorio “/etc/init.d/”, es similar al utilizado por el LER1 de ingreso para ejecutar la misma función de Colección, únicamente debe cambiar en nombre del código fuente “fcoleccionIngressV01.sh” por “fcoleccionCoreV01.sh”, como se muestra a continuación.

```
#!/bin/bash
start(){
    echo "iniciar función de colección"
    sh /root/bin/fcoleccionCoreV01.sh &
}
stop(){
    echo "para función de colección"
}
case "$1" in
    start)
        start
    ;;
    stop)
        stop
    ;;
    *)
    echo $"Usage: $0 {start|stop}"
    exit 1
esac
exit 0
```

Una vez creado el archivo “fcolecciond” se deben asignar permisos de ejecución:

[root@LSR1 root]# chmod +x /etc/init.d/fcolecciond

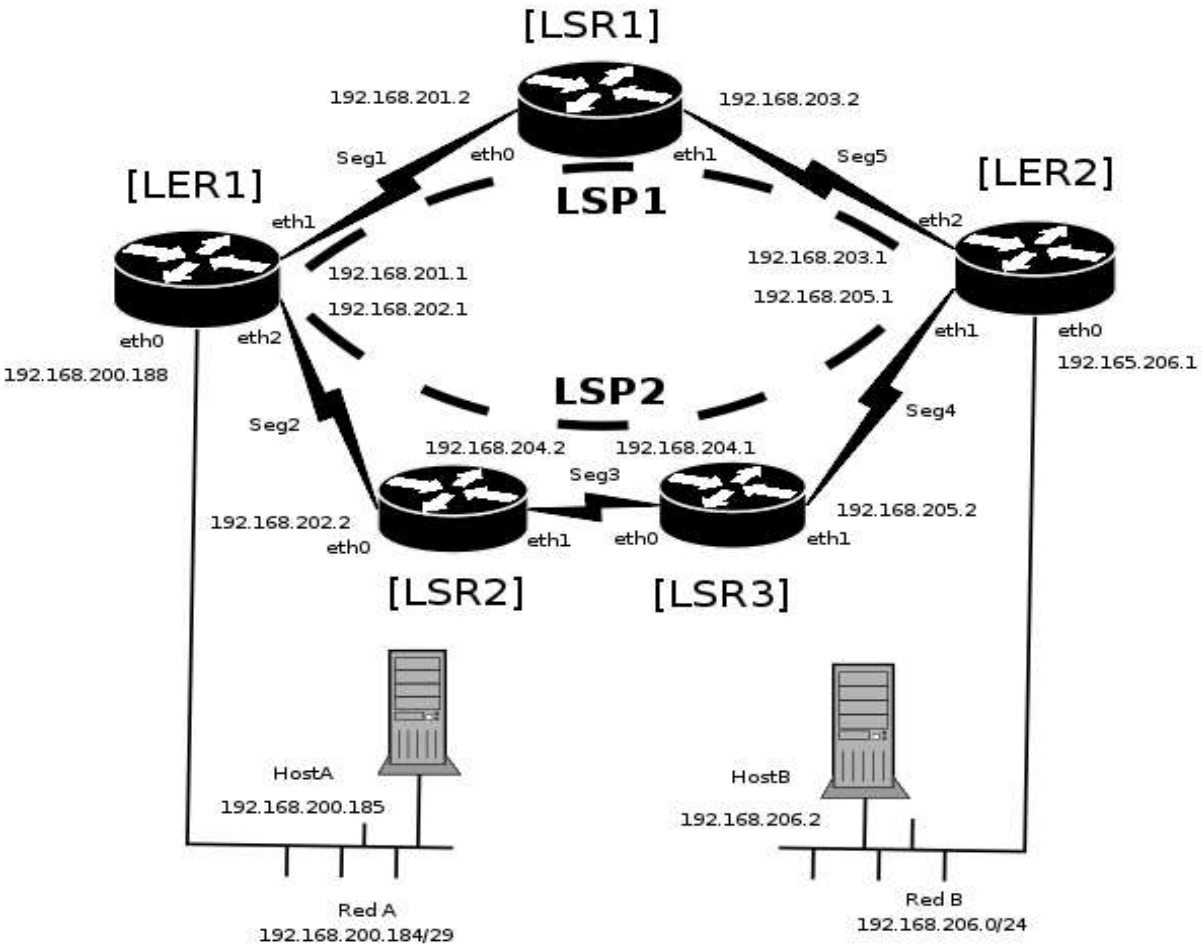
Finalmente se debe crear un enlace simbolico al run-level 3, que es el nivel de ejecución por defecto que tiene redhat:

[root@LSR1 root]# ln -s /etc/init.d/fcolecciond /etc/rc3.d/S99fcolecciond

ANEXO B: CONFIGURACION DE LA RED EXPERIMIENTAL MPLS BAJO UN AMBIENTE LINUX

Debido a los escasos recursos hardware con los que se contaban para desarrollar el proyecto de grado, se optó por utilizar un sistema de máquinas virtuales que permitiera en una sola máquina física disponer de varios sistemas operativos en representación de cada uno de los nodos requeridos en la red experimental MPLS. El diagrama topológico de la red experimental permite la creación de dos LSPs, los cuales son suficientes para ejecutar las pruebas de balanceo de carga del mecanismo DLCCM. El diagrama topológico de la red se muestra en la Figura 1.

Figura 1: Diagrama Topológico de la red Experimental MPLS



El hostA y el hostB son máquinas físicas que operan como clientes finales, los cuales tienen instalado el sistema operativo debian 3.1, y están localizados en dos sub-redes diferentes interconectadas por el domino MPLS. Los nodos LER1, LER2, LSR1, LSR2, LSR3 denominados como dominio MPLS, están implementados bajo el sistema de máquinas virtuales utilizando el software VMWare. Las máquinas pertenecientes al dominio MPLS corren el sistema operativo redHat 9.0. La guía de instalación y configuración de los sistemas operativos y el software WMWare puede ser consultada en los sitios web de cada uno de los respectivos proyectos.

El software VMWare permite la creación de segmentos lógicos, que son utilizados para interconectar las maquinas virtuales dentro del dominio MPLS. Para la creación y configuración del dominio MPLS primero se deben crear las diferentes maquinas virtuales en este caso el LER1, LER2, LSR1, LSR2 y LSR3, luego instalar el sistema operativo correspondiente a cada maquina virtual, este proceso es semejante a la instalación de un sistema operativo en una máquina física, solamente se requiere Introducir el CD1 de instalación de redhat 9.0 en la unidad de CD ROM, iniciar la maquina virtual y continuar la instalación estándar de redhat 9.0. Finalmente se debe crear un grupo virtual con 5 segmentos virtuales, al cual deben adicionase las máquinas virtuales previamente creadas y configuradas, como también realizar una correcta interconexión entre las máquinas virtuales dentro del dominio MPLS. Las Figuras 2, 3 y 4 muestran como debe estar configurado el dominio virtual MPLS.

Figura 2: Dominio Virtual MPLS

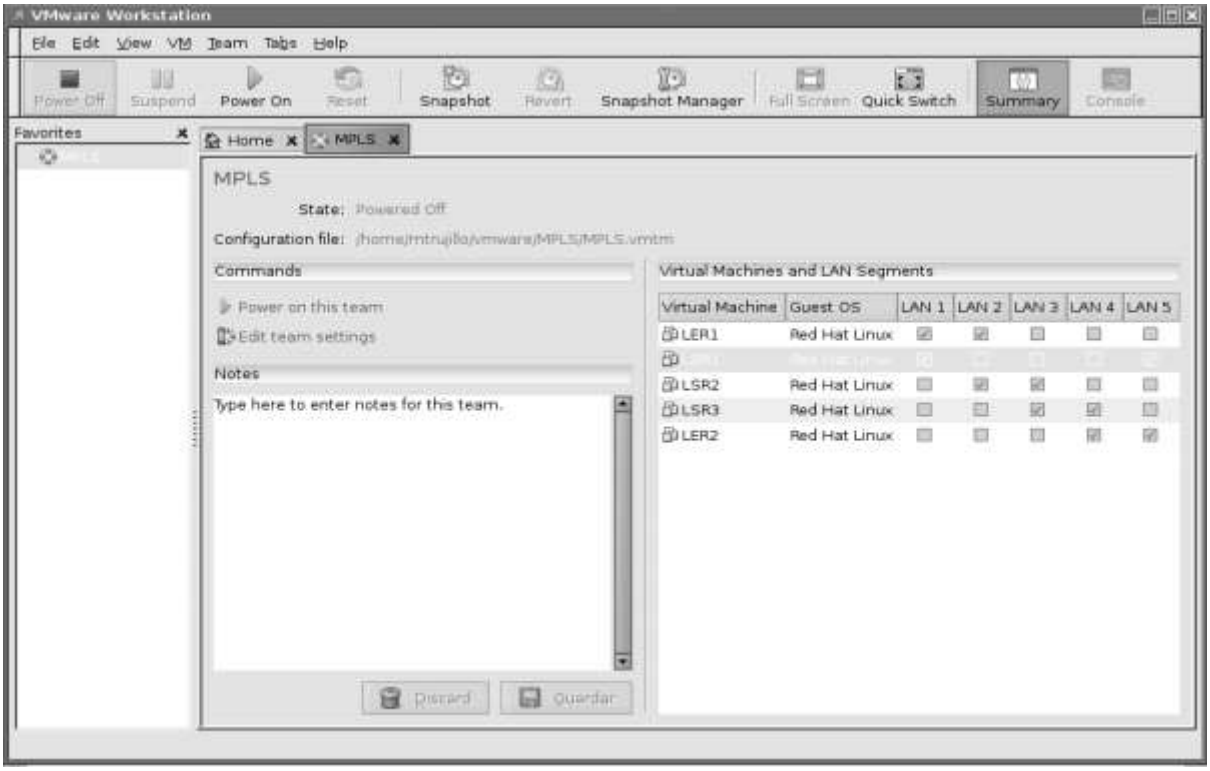


Figura 3: Interconexión de las Maquinas Virtuales dentro del Dominio MPLS

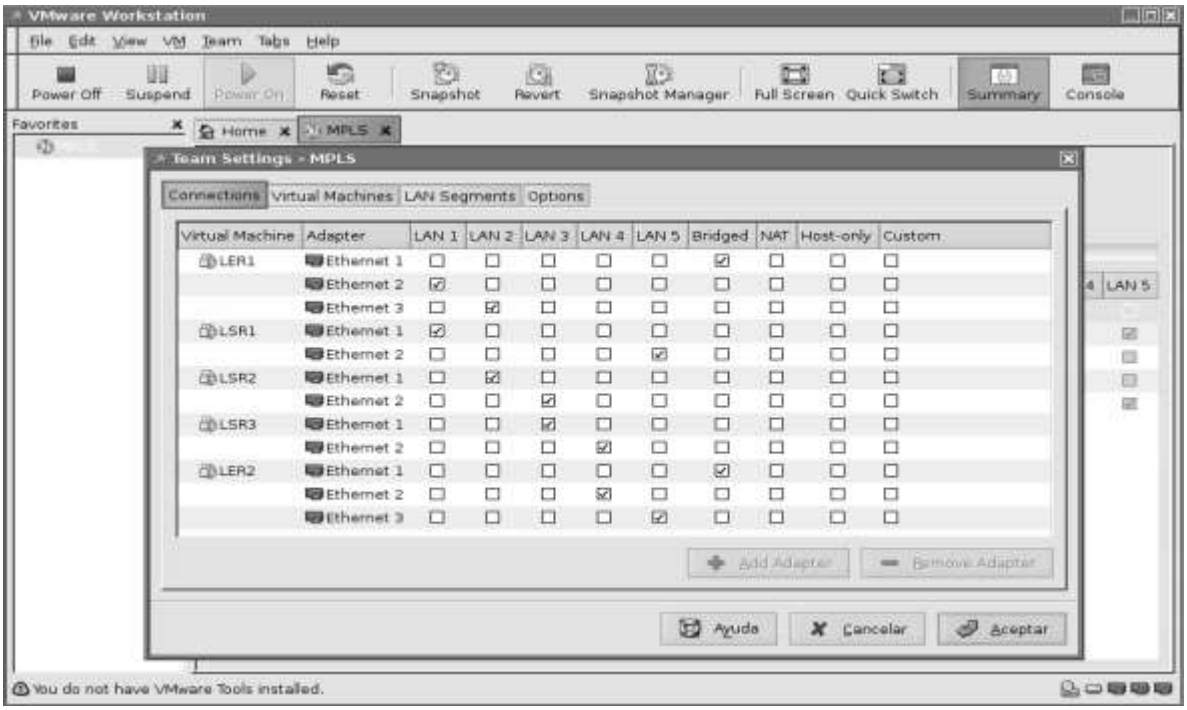
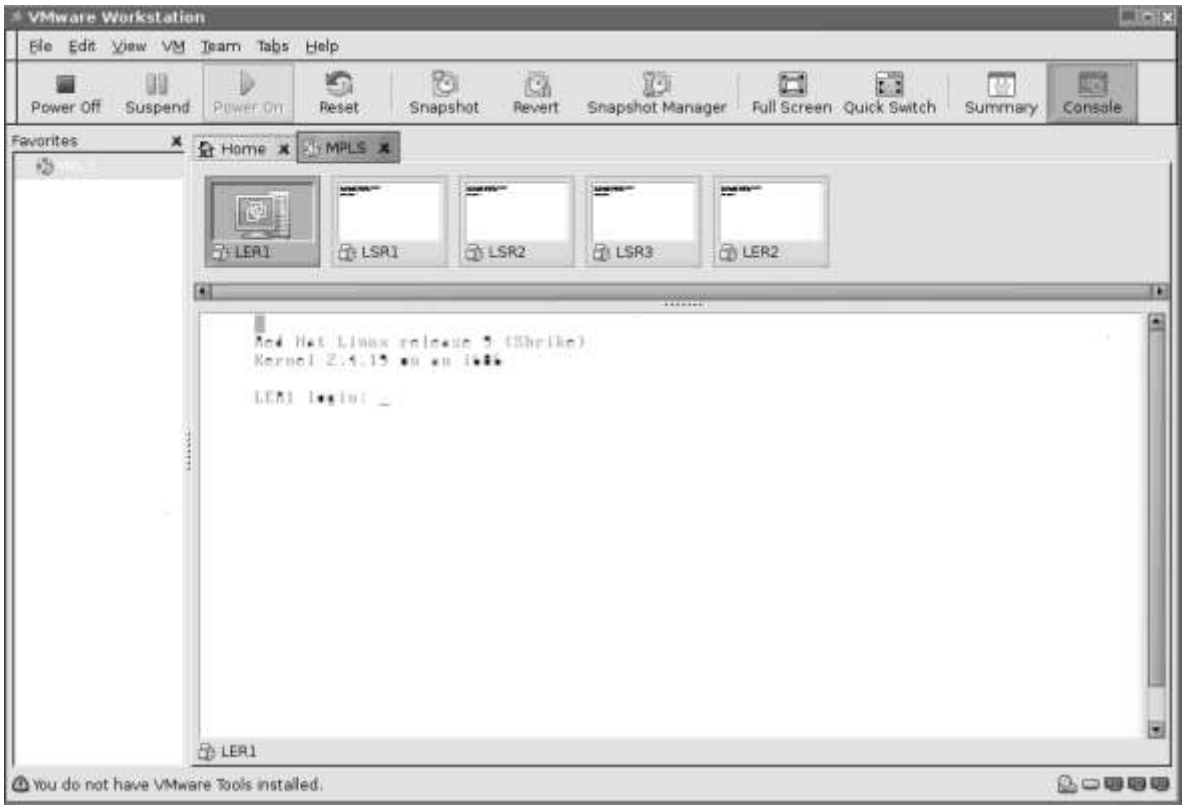


Figura 4: Dominio Virtual MPLS en Funcionamiento



Una vez creado y configurado el Dominio MPLS, se debe instalar y configurar el software de enrutamiento IP en cada uno de los respectivos nodos de la red Experimental MPLS, el cual corresponde al plano de control en términos de la arquitectura MPLS. El software de enrutamiento que utiliza el mecanismo DLCCM, es el desarrollado por el proyecto quagga.

B.1 Instalación y Configuración del Software de Enrutamiento Quagga

Este apartado únicamente abordará los aspectos de configuración de quagga utilizados en la implementación del mecanismo DLCCM.

Descargar código fuente de quagga:

```
[root@LER1 root]# wget http://www.quagga.net/download/quagga-0.99.4.tar.gz
```

Descomprimir, Configurar e Instalar Quagga:

El software de enrutamiento quagga tiene por defecto deshabilitadas las extensiones para ingeniera de tráfico, por lo tanto en la configuración se deben habilitar los LSAs Opacos, las extensiones TE y la interfaz de comandos vtysh, como se muestra a continuación:

```
[root@LER1 root]#mv quagga-0.99.4.tar.gz /usr/src/
[root@LER1 root]#cd /usr/src/
[root@LER1 src]# tar zxvf quagga-0.99.4.tar.gz
quagga-0.99.4/
....
[root@LER1 src]# cd quagga-0.99.4
[root@LER1 quagga-0.99.4]# ./configure --enable-opaque-lsa --enable-ospf-te --enable-vtysh
[root@LER1 quagga-0.99.4]# make
[root@LER1 quagga-0.99.4]# make install
```

Para habilitar los demonios de zebrad y ospfd al inicio del sistema, se deben crear dos archivos que inicien los servicios y estos archivos deben tener un enlace simbólico al nivel de ejecución No 3.

Archivo de inicio de zebra:

```
[root@LER1 root]# cat /etc/init.d/quagga
#!/bin/bash
case "$1" in
start)
        /usr/local/sbin/zebra &
;;
stop)
        kill `cat /var/run/zebra.pid`
;;
restart)
```

```
  $0 stop && sleep 3
  $0 start
;;
reload)
  $0 stop
  $0 start
;;
status)
        su - quagga -c "/usr/local/sbin/zebra --help"
;;
*)
echo "Usage: $0 {start|stop|restart|reload|status}"
exit 1
esac
```

Crear un enlace simbólico al nivel de ejecución No 3:

```
[root@LER1 root]# ln -s /etc/init.d/quagga /etc/rc3.d/S85quagga
```

El proceso anterior debe repetirse en cada uno de los nodos que conforman el dominio MPLS.

Luego de habilitar las extensiones TE y los LSAs Opacos, se debe configurar las interfaces de red en cada unos de los nodos, como también zebra y ospf.

Configuración LER1

Interfaz de red eth0:

```
[root@LER1 root]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=static
IPADDR=192.168.200.188
NETMASK=255.255.255.248
GATEWAY=192.168.200.190
ONBOOT=yes
```

Interfaz de red eth1:

```
[root@LER1 root]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
ONBOOT=yes
BOOTPROTO=static
IPADDR=192.168.201.1
NETMASK=255.255.255.252
```

Interfaz de red eth2:

```
[root@LER1 root]# cat /etc/sysconfig/network-scripts/ifcfg-eth2
DEVICE=eth2
ONBOOT=yes
BOOTPROTO=static
IPADDR=192.168.202.1
NETMASK=255.255.255.252
```

Configuración zebra:

```
[root@LER1 root]# cat /usr/local/etc/zebra.conf
!
! Zebra configuration saved from vty
!   2006/07/11 08:36:59
!
hostname puffy
password puffy++
enable password puffy++
!
interface eth0
 ip address 192.168.200.188/29
!
interface eth1
 ip address 192.168.201.1/30
!
interface eth2
 ip address 192.168.202.1/30
!
interface gre0
!
interface lo
```

```
!
interface teql0
!
interface tunl0
!
ip forwarding
!
line vty
!
```

Configuración ospf:

```
[root@LER1 root]# cat /usr/local/etc/ospfd.conf
!
! Zebra configuration saved from vty
!   2006/07/11 08:36:59
!
hostname ospfd
password zebra
log stdout
!
!
!
interface eth0
!
interface eth1
!
interface eth2
!
interface gre0
!
interface lo
!
interface teql0
!
interface tunl0
!
router ospf
```

```
 ospf router-id 192.168.201.1
 passive-interface eth0
 network 192.168.200.188/29 area 0.0.0.200
 network 192.168.201.0/30 area 0.0.0.200
 network 192.168.202.0/30 area 0.0.0.200
 area 0.0.0.200 nssa translate-candidate
 capability opaque
!
line vty
!
```

Configuración LER2

Interfaz de red eth0:

```
[root@LER2 root]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.206.3
IPADDR=192.168.206.1
NETMASK=255.255.255.0
NETWORK=192.168.206.0
ONBOOT=yes
```

Interfaz de red eth1:

```
[root@LER2 root]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
ONBOOT=yes
BOOTPROTO=static
IPADDR=192.168.205.1
NETMASK=255.255.255.252
GATEWAY=192.168.205.2
```

Interfaz de red eth2:

```
[root@LER2 root]# cat /etc/sysconfig/network-scripts/ifcfg-eth2
DEVICE=eth2
ONBOOT=yes
BOOTPROTO=static
IPADDR=192.168.203.1
```

NETMASK=255.255.255.252
GATEWAY=192.168.203.2

Configuración de zebra:

```
[root@LER2 root]# cat /usr/local/etc/zebra.conf
!
! Zebra configuration saved from vty
!   2006/07/12 06:46:42
!
hostname Router
password zebra
enable password zebra
!
interface eth0
 ip address 192.168.206.1/24
!
interface eth1
 ip address 192.168.205.1/30
!
interface eth2
 ip address 192.168.203.1/30
!
interface gre0
!
interface lo
!
interface teql0
!
interface tunl0
!
ip forwarding
!
line vty
!
```

Configuración de ospf:

```
[root@LER2 root]# cat /usr/local/etc/ospfd.conf
!
! Zebra configuration saved from vty
!   2006/07/12 06:46:42
!
hostname ospfd
password zebra
log stdout
!
!
!
interface eth0
!
interface eth1
!
interface eth2
!
interface gre0
!
interface lo
!
interface teql0
!
interface tunl0
!
router ospf
 ospf router-id 192.168.203.1
 network 192.168.203.0/30 area 0.0.0.200
 network 192.168.205.0/30 area 0.0.0.200
 network 192.168.206.0/24 area 0.0.0.200
 area 0.0.0.200 nssa translate-candidate
 capability opaque
!
line vty
!
```

Configuración LSR1

Interfaz de red eth0:

```
[root@LSR1 root]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.201.3
IPADDR=192.168.201.2
NETMASK=255.255.255.252
NETWORK=192.168.201.0
ONBOOT=yes
```

Interfaz de red eth1:

```
[root@LSR1 root]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
BOOTPROTO=static
BROADCAST=192.168.203.3
IPADDR=192.168.203.2
NETMASK=255.255.255.252
NETWORK=192.168.203.0
ONBOOT=yes
```

Configuración de zebra:

```
[root@LSR1 root]# cat /usr/local/etc/zebra.conf
!
! Zebra configuration saved from vty
!   2006/07/12 06:40:01
!
hostname Router
password zebra
enable password zebra
!
interface eth0
 ip address 192.168.201.2/30
!
interface eth1
 ip address 192.168.203.2/30
!
```

```
interface gre0
!
interface lo
!
interface teql0
!
interface tunl0
!
ip forwarding
!
line vty
!
```

Configuración de ospf:

```
[root@LSR1 root]# cat /usr/local/etc/ospfd.conf
!
! Zebra configuration saved from vty
!   2006/07/12 06:40:01
!
hostname ospfd
password zebra
log stdout
!
!
!
interface eth0
 mpls-te link metric 0
 mpls-te link max-bw 10000
 mpls-te link unrsv-bw 0 1.25e+06
 mpls-te link unrsv-bw 1 1.25e+06
 mpls-te link unrsv-bw 2 1.25e+06
 mpls-te link unrsv-bw 3 1.25e+06
 mpls-te link unrsv-bw 4 1.25e+06
 mpls-te link unrsv-bw 5 1.25e+06
 mpls-te link unrsv-bw 6 1.25e+06
 mpls-te link unrsv-bw 7 1.25e+06
```

```
 mpls-te link rsc-clsclr 0xa1
!
interface eth1
 mpls-te link metric 0
 mpls-te link max-bw 10000
 mpls-te link unrsv-bw 0 1.25e+06
 mpls-te link unrsv-bw 1 1.25e+06
 mpls-te link unrsv-bw 2 1.25e+06
 mpls-te link unrsv-bw 3 1.25e+06
 mpls-te link unrsv-bw 4 1.25e+06
 mpls-te link unrsv-bw 5 1.25e+06
 mpls-te link unrsv-bw 6 1.25e+06
 mpls-te link unrsv-bw 7 1.25e+06
 mpls-te link rsc-clsclr 0xa2
!
interface gre0
!
interface lo
!
interface teql0
!
interface tunl0
!
router ospf
 ospf router-id 192.168.201.2
 network 192.168.201.0/30 area 0.0.0.200
 network 192.168.203.0/30 area 0.0.0.200
 area 0.0.0.200 nssa translate-candidate
 capability opaque
  mpls-te
  mpls-te router-address 192.168.201.2
!
line vty
!
```

Configuración LSR2

Interfaz de red eth0:

[root@LSR2 root]# cat /etc/sysconfig/network-scripts/ifcfg-eth0

DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.202.3
IPADDR=192.168.202.2
NETMASK=255.255.255.252
NETWORK=192.168.202.0
ONBOOT=yes

Interfaz de red eth1:

[root@LSR2 root]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
BOOTPROTO=static
BROADCAST=192.168.204.3
IPADDR=192.168.204.2
NETMASK=255.255.255.252
NETWORK=192.168.204.0
ONBOOT=yes

Configuración de zebra:

[root@LSR2 root]# cat /usr/local/etc/zebra.conf
!
! Zebra configuration saved from vty
!   2006/07/12 07:10:14
!
hostname Router
password zebra
enable password zebra
!
interface eth0
 ip address 192.168.202.2/30
!
interface eth1
 ip address 192.168.204.2/30
!
interface gre0
!
interface lo
!

```
interface teql0
!
interface tunl0
!
ip forwarding
!
line vty
!
```

Configuración ospf:

```
[root@LSR2 root]# cat /usr/local/etc/ospfd.conf
!
! Zebra configuration saved from vty
!   2006/07/12 07:10:14
!
hostname ospfd
password zebra
log stdout
!
!
!
interface eth0
 mpls-te link metric 0
 mpls-te link max-bw 10000
 mpls-te link unrsv-bw 0 1.25e+06
 mpls-te link unrsv-bw 1 1.25e+06
 mpls-te link unrsv-bw 2 1.25e+06
 mpls-te link unrsv-bw 3 1.25e+06
 mpls-te link unrsv-bw 4 1.25e+06
 mpls-te link unrsv-bw 5 1.25e+06
 mpls-te link unrsv-bw 6 1.25e+06
 mpls-te link unrsv-bw 7 1.25e+06
 mpls-te link rsc-clsclr 0xa1
!
interface eth1
 mpls-te link metric 0
 mpls-te link max-bw 10000
 mpls-te link unrsv-bw 0 1.25e+06
```

```
 mpls-te link unrsv-bw 1 1.25e+06
 mpls-te link unrsv-bw 2 1.25e+06
 mpls-te link unrsv-bw 3 1.25e+06
 mpls-te link unrsv-bw 4 1.25e+06
 mpls-te link unrsv-bw 5 1.25e+06
 mpls-te link unrsv-bw 6 1.25e+06
 mpls-te link unrsv-bw 7 1.25e+06
 mpls-te link rsc-clsclr 0xa2
!
interface gre0
!
interface lo
!
interface teql0
!
interface tunl0
!
router ospf
 ospf router-id 192.168.202.2
 network 192.168.202.0/30 area 0.0.0.200
 network 192.168.204.0/30 area 0.0.0.200
 area 0.0.0.200 nssa translate-candidate
 capability opaque
  mpls-te
  mpls-te router-address 192.168.202.2
!
line vty
!
```

Configuración LSR3

Intefaz de red eth0:

```
[root@LSR3 root]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.204.3
IPADDR=192.168.204.1
NETMASK=255.255.255.252
NETWORK=192.168.204.0
```

ONBOOT=yes

Interfaz de red eth1:

[root@LSR3 root]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
BOOTPROTO=static
BROADCAST=192.168.205.3
IPADDR=192.168.205.2
NETMASK=255.255.255.252
NETWORK=192.168.205.0
ONBOOT=yes

Configuración de zebra:

[root@LSR3 root]# cat /usr/local/etc/zebra.conf
!
! Zebra configuration saved from vty
!   2006/07/12 06:42:07
!
hostname Router
password zebra
enable password zebra
!
interface eth0
 ip address 192.168.204.1/30
!
interface eth1
 ip address 192.168.205.2/30
!
interface gre0
!
interface lo
!
interface teql0
!
interface tunl0
!
ip forwarding

```
!
line vty
!
```

Configuración de ospf:

```
[root@LSR3 root]# cat /usr/local/etc/ospfd.conf
!
! Zebra configuration saved from vty
!   2006/07/12 06:42:07
!
hostname ospfd
password zebra
log stdout
!
!
!
interface eth0
 mpls-te link metric 0
 mpls-te link max-bw 10000
 mpls-te link unrsv-bw 0 1.25e+06
 mpls-te link unrsv-bw 1 1.25e+06
 mpls-te link unrsv-bw 2 1.25e+06
 mpls-te link unrsv-bw 3 1.25e+06
 mpls-te link unrsv-bw 4 1.25e+06
 mpls-te link unrsv-bw 5 1.25e+06
 mpls-te link unrsv-bw 6 1.25e+06
 mpls-te link unrsv-bw 7 1.25e+06
 mpls-te link rsc-clsclr 0xa1
!
interface eth1
 mpls-te link metric 0
 mpls-te link max-bw 10000
 mpls-te link unrsv-bw 0 1.25e+06
 mpls-te link unrsv-bw 1 1.25e+06
 mpls-te link unrsv-bw 2 1.25e+06
 mpls-te link unrsv-bw 3 1.25e+06
 mpls-te link unrsv-bw 4 1.25e+06
 mpls-te link unrsv-bw 5 1.25e+06
```

```
 mpls-te link unrsv-bw 6 1.25e+06
 mpls-te link unrsv-bw 7 1.25e+06
 mpls-te link rsc-clsclr 0xa2
!
interface gre0
!
interface lo
!
interface teql0
!
interface tunl0
!
router ospf
 ospf router-id 192.168.204.1
 network 192.168.204.0/30 area 0.0.0.200
 network 192.168.205.0/30 area 0.0.0.200
 area 0.0.0.200 nssa translate-candidate
 capability opaque
  mpls-te
  mpls-te router-address 192.168.204.1
!
line vty
!
```

Pruebas de Conexión

Una vez configurado el software de enrutamiendo en todo el dominio MPLS se deben hacer pruebas de conexión.

Pruebas hostA(alcatraz)->LER1:

Pruebas de ping:

```
mtrujillo@alcatraz:~$ ping LER1 -c 2
PING LER1.unicauca.edu.co (192.168.200.188) 56(84) bytes of data.
64 bytes from LER1.unicauca.edu.co (192.168.200.188): icmp_seq=1 ttl=64 time=0.633 ms
64 bytes from LER1.unicauca.edu.co (192.168.200.188): icmp_seq=2 ttl=64 time=0.739 ms

--- LER1.unicauca.edu.co ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1006ms
```

rtt min/avg/max/mdev = 0.633/0.686/0.739/0.053 ms
mtrujillo@alcatraz:~$

Pruebas de traceroute:

mtrujillo@alcatraz:~$ traceroute LER1
traceroute to LER1.unicauca.edu.co (192.168.200.188), 30 hops max, 38 byte packets
 1 LER1 (192.168.200.188) 11.811 ms 5.754 ms 1.036 ms
mtrujillo@alcatraz:~$

Pruebas hostA(alcatraz)->LSR1:

Prueba de ping:

mtrujillo@alcatraz:~$ ping LSR1 -c 2
PING LSR1.unicauca.edu.co (192.168.201.2) 56(84) bytes of data.
64 bytes from LSR1.unicauca.edu.co (192.168.201.2): icmp_seq=1 ttl=63 time=2.92 ms
64 bytes from LSR1.unicauca.edu.co (192.168.201.2): icmp_seq=2 ttl=63 time=3.06 ms

--- LSR1.unicauca.edu.co ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 2.929/2.997/3.066/0.087 ms
mtrujillo@alcatraz:~$

Prueba de traceroute:

mtrujillo@alcatraz:~$ traceroute LSR1
traceroute to LSR1.unicauca.edu.co (192.168.201.2), 30 hops max, 38 byte packets
 1 LER1 (192.168.200.188) 10.698 ms 6.820 ms 1.054 ms
 2 LSR1 (192.168.201.2) 6.663 ms 4.956 ms 2.616 ms
mtrujillo@alcatraz:~$

Pruebas hostA(alcatraz)->LSR2:

Prueba de ping:

mtrujillo@alcatraz:~$ ping LSR2 -c 2
PING LSR2.unicauca.edu.co (192.168.202.2) 56(84) bytes of data.
64 bytes from LSR2.unicauca.edu.co (192.168.202.2): icmp_seq=1 ttl=63 time=6.18 ms
64 bytes from LSR2.unicauca.edu.co (192.168.202.2): icmp_seq=2 ttl=63 time=2.36 ms

--- LSR2.unicauca.edu.co ping statistics ---

2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 2.360/4.273/6.186/1.913 ms
mtrujillo@alcatraz:~$

Prueba de traceroute:

mtrujillo@alcatraz:~$ traceroute LSR2
traceroute to LSR2.unicauca.edu.co (192.168.202.2), 30 hops max, 38 byte packets
 1 LER1 (192.168.200.188) 10.079 ms 6.614 ms 7.912 ms
 2 LSR2 (192.168.202.2) 9.574 ms 8.986 ms 3.992 ms
mtrujillo@alcatraz:~$

Pruebas hostA(alcatraz)->LSR3:

Prueba de ping:

mtrujillo@alcatraz:~$ ping LSR3 -c 2
PING LSR3.unicauca.edu.co (192.168.204.1) 56(84) bytes of data.
64 bytes from LSR3.unicauca.edu.co (192.168.204.1): icmp_seq=1 ttl=62 time=34.4 ms
64 bytes from LSR3.unicauca.edu.co (192.168.204.1): icmp_seq=2 ttl=62 time=3.63 ms

--- LSR3.unicauca.edu.co ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 3.633/19.023/34.413/15.390 ms
mtrujillo@alcatraz:~$

Prueba de traceroute:

traceroute to LSR3.unicauca.edu.co (192.168.204.1), 30 hops max, 38 byte packets
 1 LER1 (192.168.200.188) 13.052 ms 1.045 ms 1.154 ms
 2 LSR2 (192.168.202.2) 7.988 ms 6.028 ms 5.655 ms
 3 LSR3 (192.168.204.1) 9.160 ms 7.828 ms 6.896 ms
mtrujillo@alcatraz:~$

Pruebas hostA(alcatraz)->LER2:

En estas pruebas solo se prueba la ruta más corta, [hostA-LER1-LSR1-LER2].

Prueba de ping:

mtrujillo@alcatraz:~$ ping LER2 -c 2
PING LER2.unicauca.edu.co (192.168.206.1) 56(84) bytes of data.
64 bytes from LER2.unicauca.edu.co (192.168.206.1): icmp_seq=1 ttl=62 time=8.96 ms
64 bytes from LER2.unicauca.edu.co (192.168.206.1): icmp_seq=2 ttl=62 time=4.27 ms

--- LER2.unicauca.edu.co ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 4.277/6.622/8.967/2.345 ms
mtrujillo@alcatraz:~$

Prueba de traceroute:

mtrujillo@alcatraz:~$ traceroute LER2
traceroute to LER2.unicauca.edu.co (192.168.206.1), 30 hops max, 38 byte packets
 1 LER1 (192.168.200.188) 13.572 ms 4.680 ms 0.854 ms
 2 LSR1 (192.168.201.2) 5.952 ms 2.317 ms 2.148 ms
 3 LER2 (192.168.206.1) 10.269 ms 4.494 ms 7.184 ms
mtrujillo@alcatraz:~$

Pruebas hostA(alcatraz)->LER2:

Se prueba la ruta más larga, [hostA-LER1-LSR2-LSR3-LER2], para lo cual se baja la interfaz eth1 del LER1.

[root@LER1 root]# ifconfig eth1 down
[root@LER1 root]#

Con lo anterior se elimina la ruta más corta para hacer que el tráfico siga la ruta más larga.

Prueba de ping:

mtrujillo@alcatraz:~$ ping LER2 -c 2
PING LER2.unicauca.edu.co (192.168.206.1) 56(84) bytes of data.
64 bytes from LER2.unicauca.edu.co (192.168.206.1): icmp_seq=1 ttl=61 time=27.8 ms
64 bytes from LER2.unicauca.edu.co (192.168.206.1): icmp_seq=2 ttl=61 time=9.78 ms

--- LER2.unicauca.edu.co ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1139ms
rtt min/avg/max/mdev = 9.789/18.829/27.869/9.040 ms
mtrujillo@alcatraz:~$

Prueba de traceroute:

mtrujillo@alcatraz:~$ traceroute LER2

```
traceroute to LER2.unicauca.edu.co (192.168.206.1), 30 hops max, 38 byte packets
 1  LER1 (192.168.200.188)  25.375 ms  1.026 ms  1.052 ms
 2  LSR2 (192.168.202.2)  6.885 ms  4.469 ms  6.114 ms
 3  LSR3 (192.168.204.1)  28.946 ms  8.278 ms  7.043 ms
 4  LER2 (192.168.206.1)  13.608 ms  9.037 ms  9.006 ms
mtrujillo@alcatraz:~$
```

Pruebas hostA (alcatraz)->hostB:

En estas pruebas solo se prueba la ruta más corta, [hostA-LER1-LSR1-LER2-hostB].

Prueba de ping:

```
mtrujillo@alcatraz:~$ ping hostB -c 2
PING hostB.unicauca.edu.co (192.168.206.2) 56(84) bytes of data.
64 bytes from hostB.unicauca.edu.co (192.168.206.2): icmp_seq=1 ttl=252 time=13.2 ms
64 bytes from hostB.unicauca.edu.co (192.168.206.2): icmp_seq=2 ttl=252 time=6.40 ms

--- hostB.unicauca.edu.co ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1008ms
rtt min/avg/max/mdev = 6.400/9.829/13.258/3.429 ms
mtrujillo@alcatraz:~$
```

Prueba de traceroute:

```
mtrujillo@alcatraz:~$ traceroute hostB
traceroute to hostB.unicauca.edu.co (192.168.206.2), 30 hops max, 38 byte packets
 1  LER1 (192.168.200.188)  1.783 ms  0.474 ms  0.973 ms
 2  LSR1 (192.168.201.2)  2.062 ms  2.292 ms  2.038 ms
 3  LER2 (192.168.203.1)  3.893 ms  3.691 ms  3.985 ms
 4  hostB (192.168.206.2)  15.121 ms  8.956 ms  10.027 ms
mtrujillo@alcatraz:~$
```

Pruebas hostA(alcatraz)->hostB:

Se prueba la ruta más larga, [hostA-LER1-LSR2-LSR3-LER2-hostB], para lo cual se baja la interfaz eth1 del LER1.

```
[root@LER1 root]# ifconfig eth1 down
[root@LER1 root]#
```

Con lo anterior se elimina la ruta más corta para hacer que el tráfico siga la ruta más larga.

Prueba de ping:

```
mtrujillo@alcatraz:~$ ping hostB -c 2
PING hostB.unicauca.edu.co (192.168.206.2) 56(84) bytes of data.
64 bytes from hostB.unicauca.edu.co (192.168.206.2): icmp_seq=1 ttl=251 time=10.2 ms
64 bytes from hostB.unicauca.edu.co (192.168.206.2): icmp_seq=2 ttl=251 time=6.33 ms

--- hostB.unicauca.edu.co ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 6.333/8.300/10.268/1.969 ms
```

Prueba de traceroute:

```
mtrujillo@alcatraz:~$ traceroute hostB
traceroute to hostB.unicauca.edu.co (192.168.206.2), 30 hops max, 38 byte packets
 1  LER1 (192.168.200.188)  7.884 ms  1.189 ms  0.616 ms
 2  LSR2 (192.168.202.2)  11.666 ms  6.534 ms  3.907 ms
 3  LSR3 (192.168.204.1)  14.735 ms  5.854 ms  7.525 ms
 4  LER2 (192.168.205.1)  15.309 ms  10.964 ms  8.940 ms
 5  hostB (192.168.206.2)  11.531 ms  9.865 ms  31.761 ms
mtrujillo@alcatraz:~$
```

Las pruebas anteriores indican que el enrutamiento entre la red A (192.168.200.184/29) y la red B (192.168.206.0/24) esta funcionando correctamente. A continuación se presentada detalladamente la instalación y configuración del demonio rsvpd, el cual tiene integrada la arquitectura MPLS.

B.2 Instalación y Configuración de rsvpd-MPLS-Linux

Los siguientes pasos que se presentan a continuación deben ser ejecutados correctamente a fin de obtener una correcta configuración de un nodo MPLS. El motivo por el cual se utiliza Ret Hat 9.0 se debe a que el parche que habilita las

funcionalidades de MPLS fue diseñado específicamente para redhat, el mismo parche fue probado en fedora Core1, Core2, debian 3.0, 3.1, sin obtenerse resultados satisfactorios.

Instalación

1. Descargar las fuentes del kernel de Linux (linux-2.4.19.tar.bz2) en el directorio "/usr/src/", desde http://www.kernel.org.

2. El directorio "/usr/src" puede contener un enlace simbólico a una versión del Kernel diferente a la que se requiere, si esto ocurre debe ser borrado dicho enlace simbólico.

3. Descomprimir las fuentes del kernel y crear un enlace simbólico llamado linux.

```
[root@LER1 root]# cd /usr/src/
[root@LER1 src]# tar jxvf linux-2.4.19.tar.bz2
[root@LER1 src]# ln -s linux-2.4.19 linux
```

4. Ahora crear la arquitectura de enlaces:

```
[root@LER1 src]# cd linux-2.4.19
[root@LER1 linux-2.4.19]# make menuconfig
```

También se puede utilizar "make xconf" en lugar de "make menuconfig", pero en los dos casos se debe dar "exit" inmediatamente.

5. Lista de Software

- Red Hat 7 o Red Hat 9. Aunque en cualquier versión de Red Hat superior o igual a Red Hat 7.0 el parche MPLS funciona correctamente.

El siguiente software puede descargase de: http://dsmpls.atlantis.ugent.be

- KERNEL 2.4.19 MPLS172.patch,
- iptables-1.2.4-0.2-dscp.tgz,
- DSMPLS+IP.patch

- iproute2-current.tar.gz and rsvpd.0.70-rc2.tgz.

6. Parche MPLS: Mover el “KERNEL 2.4.19 MPLS172.patch” dentro del directorio “/usr/src” y ejecutar el siguiente comando dentro del directorio de las fuentes del kernel:

```
[root@LER1 linux-2.4.19]# patch -p1 < ../KERNEL 2.4.19 MPLS172.patch
```

7. Parche DSMPLS+IP.patch: Mover el parche “DSMPLS+IP.patch” dentro del directorio “/usr/src” y dentro del directorio de las fuentes del kernel ejecutar lo siguiente:

```
[root@LER1 linux-2.4.19]# patch -p0 < ../DSMPLS+IP.patch
```

8. Instalación de iptable-1.2.4: Mover iptables-1.2.4-0.2-dscp.tgz dentro del directorio “/home/rsvp/” y descomprimirlo:

```
[root@LER1 rsvp]# tar zxvf iptables-1.2.4-0.2-dscp.tgz
```

9. Dentro del directorio “/home/rsvp/iptables-1.2.4-0.2-dscp” ejecutar:

```
[root@LER1 iptables-1.2.4-dscp]# make patch-o-matic
```

10. Con el comando anterior el parche solicitara confirmación de los parches que serán instalados. Responder “y” a los parches “dscp.patch” y “ftos.patch”.
IMPORTANTE**:** inmediatamente después de confirmar la Instalación del parche ftos.patch debe ejecutar 'q' para salir de “patch-o-matic”.

11. Ahora puede compilar e instalar:

```
[root@LER1 iptables-1.2.4-dscp]# make
[root@LER1 iptables-1.2.4-dscp]# make install
```

12. Instalación del kernel de Linux: Para configurar el kernel, entrar al directorio “usr/src/linux-2.4.19” y ejecutar:

```
[root@LER1 linux-2.4.19]# make xconfig
```

13. Compilar el kernel con mínimo las siguientes opciones de networking (Habilitar en el "Prompt for development and/or incomplete code/drivers")

Las opciones listadas a continuación son obligatorias:

- Network packet Filtering
- TCP/IP networking
    1. advanced router
    2. IP: policy routing
    3. IP: use netlter MARK value as routing key
- suport MPLS

- Configuración de IP: Netlter
    1. IP tables support (habilitar todas las suboptions amenos que usted sepa que esta haciendo, asegurase que el soporte para "DSCP match" y "MPLS target" este disponible.

- QoS and/or fair queuing (habilitar todas las sub-opciones amenos que sepa que esta haciendo).

- Network device options (En la Configuración del dominio MPLS se habilitaron todas las opciones de “network options” excepto las opciones experimentales).

14. - Compilar e instalar el Kernel de Linux:

```
[root@LER1 linux-2.4.19]# make dep
[root@LER1 linux-2.4.19]# make clean
[root@LER1 linux-2.4.19]# make bzImage
[root@LER1 linux-2.4.19]# make modules
[root@LER1 linux-2.4.19]# make modules install
```

15. Mover bzImage y System.map al directorio “/boot” y crear un nuevo enlace simbólico para System.map

```
[root@LER1 linux-2.4.19]# cp arch/i386/boot/bzImage /boot/vmlinuz-2.4.19
[root@LER1 linux-2.4.19]# cp System.map /boot/System.map-2.4.19
```

16. Opcionales:

```
[root@LER1 boot]# cd /boot
```

```
[root@LER1 boot]# rm vmlinuz
[root@LER1 boot]# ln –s vmlinuz-2.4.19 vmlinuz
[root@LER1 boot]# rm System.map
[root@LER1 boot]# ln –s System.map-2.4.19 System.map
```

Opciones Finales

17. Editar el "/etc/grub/grub.conf" y adicionar las líneas para la nueva imagen. Un ejemplo de la Configuración del grub se muestra a continuación:

```
title Red Hat 9.0 - LER1 (2.4.19)
root (hd0,0)
kernel /vmlinuz-2.4.19 ro root=LABEL=/
```

Ahora debe ejecutar el comando "grub" y dentro del promt ejecutar reboot para actualizar la Configuración.

```
[root@LER1 boot]# grub

   GRUB  version 0.93  (640K lower / 3072K upper memory)

 [ Minimal BASH-like line editing is supported.  For the first word, TAB
   lists possible command completions.  Anywhere else TAB lists the possible
   completions of a device/filename.]

grub> reboot
```

18. Chequear el kernel:

```
[root@LER1 boot]# dmesg | grep -i mpls
MPLS version 1.172 11/14/2002 jleu@mindspring.com
```

19. Instalación de rsvp-te:RSVP-TE:

```
[root@LER1 root]# cd /usr/include/
[root@LER1 root]# mv linux linux.old
[root@LER1 root]# ln -s /usr/src/linux/include/linux linux
```

22. Poner las fuentes del demonio rsvp (svpd.0.70-rc2.tgz) en el "/home" y ejecutar el siguiente comando:

```
[root@LER1 root]#  tar zxvf rsvpd.0.70-rc2.tgz
```

Esto crea un directorio "/home/rsvpd" que contiene todos los requerimientos necsarios para la Instalación de rsvpd y MPLS-Lunux.

25. [root@LER1 root]# cd /home/rsvpd

26. Ahora compilar las fuentes de rsvpd:

```
[root@LER1 rsvpd]# make clean
[root@LER1 rsvpd]# make
```

27. Una vez compiladas las fuentes del demonio rsvpd, el archivo binario de rsvpd se localiza en el directorio "/home/rsvpd/rsvpd/" y otras herramientas útiles de MPLS-Linux se encuentran en el directorio "/home/rsvpd/labeltest".

28. Ahora se deben copiar los binarios de las herramientas de MPLS-Linux y el demonio rsvpd, al directorio "/usr/loca/bin/" con lo cual se hacen ejecutables.

```
[root@LER1 labeltest]# cp mplsadm /usr/local/bin/
[root@LER1 labeltest]# cp rapirecv_auto /usr/local/bin/
[root@LER1 labeltest]# cp rtest2 /usr/local/bin/
[root@LER1 labeltest]# cp tc /usr/local/bin/
[root@LER1 labeltest]# cp tunnel /usr/local/bin/
[root@LER1 rsvpd]# cp rsvpd /usr/local/bin/
```

29. Antes de correr el demonio rsvpd se deben realizar algunas configuraciones:

```
[root@LER1 rsvpd]# cd /home/rsvpd/
[root@LER1 rsvpd]# vi label.conf
```

30. Chequear que el nombre de las interfaces en el archivo "label.conf" correspondan con las interfaces activas y salvar en "/etc/bel.conf".

```
[root@LER1 rsvpd]# cp label.conf /etc/
```

31. Editar el archivo "ds config", chequeando que la variable IFACES contenga los nombres de la interfaces activas en el nodo. Ejemplo: IFACES='eth0 eth1 eth2'
Ejecutar el script "ds config"

```
[root@LER1 rsvpd]# ./ds_config
```

El proceso anterior debe realizarse en cada uno de los nodos que conforman la red Experimental MPLS.

Configuración del Dominio MPLS

Una vez realizada la configuración, puesta en funcionamiento del enrutamiento IP, instalado el demonio rsvpd y habilitadas las funcionalidades de MPLS en el kernel de Linux, se completan todos los requerimientos para configurar el Dominio MPLS, el cual acorde a la Figura 1, esta conformado por un LER de Ingreso (LER1), un LER de Egreso (LER2), y tres LSRs (LSR1, LSR2, LSR3) distribuidos en el Core MPLS.
Lo primero que se debe hacer es habilitar el demonio rsvpd en el Dominio MPLS, para lo cual debe ejecutarse el siguiente comando en todos los nodos.

```
[root@LER1 root]# rsvpd -D
09:40:57.913  RSVP Version 4.2a4tl-0.70-rc2, Compiled on: May 28 2006 at 18:16
09:40:57.945  Log level:7  Debug mask:7  Start time: Wed Sep 13 09:40:57 2006
09:40:57.993  Physical, Virtual, and API interfaces are:
09:40:57.996   0 API                    0.0.0.0 (0) NoIS
09:40:57.997   1 eth0                 192.168.200.188 (5 NoIS M
09:40:57.998   2 eth1                  192.168.201.1 (6) NoIS M
09:40:57.999   3 eth2                  192.168.202.1 (7) NoIS M
rtap [v3.1] RSVP Test Application
RSVP API version 5.02

Enter ? or command:
T1>
```

Adicionalmente en los LERs se deben configurar los LSPs explícitos, lo cual se hace utilizando una de las extensiones de ingeniería de tráfico disponible en RSVP-TE e implementada en el demonio rsvpd. Haciendo uso de las utilidades disponibles en el demonio rsvpd, se debe habilitar el LER de egreso (LER2) para

que una vez reciba un mensaje “PATH” envíe automáticamente el mensaje “RESV”, como se muestra a continuación:

En el LER de egreso:

```
[root@LER2 root]# rapirecv_auto &
```

En el LER de Ingreso se deben enviar los mensajes “PATH” solicitando la creación de los LSPs mediante la utilidad “rtest2”.

```
[root@puffy root]# rtest2 -f /root/bin/file
```

El archivo “file” define los LSPs explícitos, así pueden ser creados varios LSPs simultáneamente, con un simple comando. El formato de configuración de los LSP en el archivo “file” es el siguiente:

<dst_ip> <src_ip> <lsp_id> <PHBID> <IntServ> <hop_ip1:hop_id2:hop_ip3:0> <SE style>

dst_ip: Dirección IP de destino.

src_ip: Dirección IP de Origen.

lsp_id: Identificador del LSP.

PHBID: Comportamiento salto a salto, utilizado generalmente con Diff -Serv.

IntServ: Define el uso de señalización en servicios integrados.

hop_ip[1,2,3]: Dirección IP de cada salto que conforma el LSP explicito, el máximo número de saltos soportado es 19.

SE style: Define el tipo de reserva Explícitamente Compartido (SE:Shared Explicit) para el LSP, el cual le permite hacer una reserva simple, pero el receptor puede identificar qué transmisores podrían compartir dicha reserva.

El contenido del archivo “file” para la implementación del Dominio MPLS, acorde a la topología de red mostrada en la Figura 1, se muestra a continuación:

```
[root@puffy root]#cd bin
[root@puffy root]#cat file
192.168.206.1 192.168.201.1 200 0 0 192.168.201.2:192.168.203.1:0 0
192.168.206.1 192.168.202.1 201 0 0 192.168.202.2:192.168.204.1:192.168.205.1:0 0
```

Luego de creados los LSPs se debe ejecutar el archivo “lbmplsV01.sh”, con lo cual finaliza la implementación y puesta en funcionamiento del mecanismo de balanceo

de carga en redes MPLS. La Figura 5 muestra la creación de los LSPs primario y secundario.

Figura 5: Creación de los LSPs Primario y Secundario.

```
[root@LER1 root]# rtest2 -f bin/file
192.168.201.2
192.168.203.1
Path Message to: 192.168.206.1, Port: 200
Rapi_Session: session id=1 ,fd=4
use etid 0
192.168.202.2
192.168.204.1
192.168.205.1
Path Message to: 192.168.206.1, Port: 201
Rapi_Session: session id=2 ,fd=4
use etid 0
Goto admin mode
Enter 'session id' of the LSP to tear down
_
```

Para verificar que los LSPs se hayan creado correctamente se puede utilizar el comando “túnel”, como se muestra en la Figura 6.

Figura 6: Verificación de los LSPs

```
[root@LER1 root]# tunnel -L

LSPID          Destination (type label/ phb/         viface)
E 200        192.168.206.1 ( gen 11680/  BE/     T11680eth1)
E 201        192.168.206.1 ( gen 11680/  BE/     T11680eth2)
[root@LER1 root]# _
```

Utilizando la opción “-c” con el comando “tunnel”, es posible visualizar en número de paquetes que cruzan a través de cada LSP, como se muestra en la Figura 7.

Figura 7: Visualizar de los Contadores de Paquetes de LSP

```
[root@LER1 root]# tunnel -L

LSPID          Destination (type label/ phb/         viface)
E 200        192.168.206.1 ( gen 11680/  BE/     T11680eth1)
E 201        192.168.206.1 ( gen 11680/  BE/     T11680eth2)
[root@LER1 root]#
[root@LER1 root]# tunnel -L -c

LSPID          Destination (type label/ phb/         viface)  Packets     Bytes
E 200        192.168.206.1 ( gen 11680/  BE/     T11680eth1)        0         0
E 201        192.168.206.1 ( gen 11680/  BE/     T11680eth2)        0         0
[root@LER1 root]# _
```

Con el proceso anterior se ha completado la instalación y configuración de la Red Experimental MPLS, así como la creación de los LSPs que serán utilizados por el mecanismo de balanceo de carga DLCCM.