Escuela Politécnica Superior

Diseño del nodo de conexión a Internet de la red Guifi-Elx

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autora:
Elena San Miguel Pérez

Tutor:
José Ángel Bemá Gallano

Universitat d'Alacant
Universidad de Alicante

Junio 2017

Diseño del nodo de conexión a Internet de la red de Guifi-Elx

Autora

Elena San Miguel Pérez

Tutor

José Ángel Berná Galiano

Dpto. Física, Ingeniería de Sistemas y Teoría de la Señal

GRADO EN INGENIERÍA INFORMÁTICA

Escuela
Politécnica
Superior

Alicante, 19 de junio de 2017

Resumen

El presente proyecto contribuye a la mejora del servicio de acceso a Internet de la asociación Xarxa Lliure Elx, en beneficio de la asociación y los usuarios de la red libre ciudadana de Guifi-Elx. Para ello se analiza el proyecto social y tecnológico existente en la red y sobre él se desarrollan varias actividades de mejora.

La actividad sobre la que se dedica especial detalle y atención en el presente proyecto es el despliegue del servicio de QoS para controlar el tráfico a Internet de forma que se comparta el ancho de banda justa y eficientemente. Además se realizan otras tres actividades. La primera es el diseño y despliegue de un dispositivo de respaldo al gateway de la red, que garantizará la disponibilidad del servicio de acceso a Internet en caso de caída del gateway principal de la asociación. La segunda es el aumento del ancho de banda contratado de acceso a Internet. La tercera es la instalación de un servidor web para alojar un blog informativo que aporte visibilidad a la asociación en Internet.

Con este nuevo sistema aumentan el valor y la capacidad de crecimiento de la red y de la asociación. Además, el presente proyecto plantea nuevas posibilidades de mejora, pudiendo ser de interés y utilidad para otras asociaciones, colectivos u organizaciones.

A mi familia

Índice de contenidos

Índice de figuras

1. INTRODUCCIÓN

Este documento es la memoria del proyecto de **Diseño del nodo de conexión a Internet de la red Guifi-Elx**. El proyecto abarca el estudio del estado actual de la red y el desarrollo de una serie de mejoras incluyendo la introducción de un nuevo dispositivo de respaldo en la red y la implantación de nuevos servicios, destacando entre ellos el despliegue de la tecnología de control de QoS.

El documento se divide en cuatro capítulos a parte de la Introducción: Estudio del escenario, Mejora de la red, Despliegue de QoS en la red, y Conclusiones. En el Estudio del escenario se describe la red de Guifi-Elx como red comunitaria dentro del ecosistema de guifi.net. En el capítulo de Mejora de la red se plantean las mejoras necesarias en la red comunitaria, de forma que aumente el valor para la infraestructura y para la comunidad, y que se implantarán en el nodo de acceso a Internet de la red. A continuación, en el capítulo de Despliegue de QoS en la red se desarrollan las fases de implantación de QoS con las herramientas tc e iptables. Finalmente, en las Conclusiones se resumen los objetivos conseguidos y el trabajo futuro.

2. ESTUDIO DEL ESCENARIO

La motivación de este proyecto viene alimentada principalmente por la motivación de su contexto. La red de Guifi-Elx sobre la que se trabaja en este proyecto forma parte, al principio y al final, de un movimiento global por las redes de telecomunicaciones libres. El trabajo que se desarrolla en este proyecto es local, pero por su contextualización tiene repercusiones en todos los planos de abstracción del contexto, desde lo local a lo global. Este capítulo recorre desde guifi.net como proyecto global hasta el nodo de conexión a Internet como recurso local.

Guifi.net

A continuación se recogen tres definiciones de guifi.net de fuentes diferentes.

Guifi.net es el conjunto formado por la red, los individuos, colectivos, empresas, instituciones y administraciones que dan soporte o colaboran de manera que la red esté operativa y ofrezca conectividad a todos. Por lo tanto funciona como operador de telecomunicaciones. (*Procomún de la Red Abierta, Libre y Neutral «RALN», II. 2. c.*, entrada actualizada en julio de 2012.[1])

Guifi.net es un proyecto tecnológico, social y económico impulsado desde la ciudadanía que tiene por objetivo la creación de una red de telecomunicaciones abierta, libre y neutral basada en un modelo de procomún. El desarrollo de esta infraestructura mancomunada facilita el acceso a las telecomunicaciones en general y a la conexión a Internet de banda ancha en particular, de calidad, a un precio justo y para todo el mundo. Además, genera un modelo de actividad económica colaborativa, sostenible y de proximidad. [*¿Qué es guifi.net?*, en el blog de la web de guifi.net, entrada actualizada en abril de 2017.[2]]

Guifi.net es la comunidad que aglutina y da salida a las inquietudes de las personas que quieren desarrollar una red comunitaria. Inicialmente, surge de la necesidad de proporcionar acceso a las redes de banda ancha en zonas de Cataluña central. Posteriormente, también se ha desplegado en otros lugares donde los proveedores tradicionales no han llevado la cobertura, básicamente por criterios mercantiles.

[*Desenvolupar una xarxa WiFi municipal segons el model de xarxa oberta i neutra de Guifi.net*, Sergi Cabré Godall, Trabajo Fin de Carrera de la UOC, junio de 2012].

Desde el punto de vista y experiencia de la autora, guifi.net es un **ecosistema**, y para entenderlo se han de conocer sus componentes: infraestructura, habitantes, conocimiento, beneficios e historia. Aunque se puede hacer un análisis simplificado, no se puede obviar ninguna de sus partes.

Infraestructura

Es una red de telecomunicaciones. Los nodos se conectan entre ellos a nivel físico formando dos posibles topologías: malla o infraestructura. Los nodos pueden ser casas, oficinas, granjas, edificios públicos, etc. Los enlaces se crean sobre todo con tecnología inalámbrica, aunque también los hay de cable o fibra óptica. A diferencia de las redes comerciales tradicionales, la infraestructura se crea desde los nodos, y el valor está en su interconexión directa.

Guifi.net es una red de telecomunicaciones abierta, libre y neutral (RALN)[3]. Puede ampliarse siempre, por cualquier espacio (a excepción de los límites geofísicos) y por cualquier persona o entidad. Cada participante al conectarse libremente y añadiendo nuevos nodos y enlaces, extiende la red y obtiene conectividad. Además, todas las personas pueden usar la red para cualquier propósito, respetando que el resto también lo pueda hacer. La información sobre cómo funciona y sus componentes es pública y accesible. No se permite ningún tipo de discriminación, inspección o modificación sobre el tráfico o las actividades que hay en la red, ni sobre nadie. No hay ninguna entidad que tenga más poder y que por tanto pueda apropiarse de la red.

Es una red comunitaria; es un sistema de telecomunicaciones e informático al servicio de una comunidad geográfica, para apoyar, aumentar y extender las redes sociales humanas ya existentes.

Habitantes

Las redes sociales, entendidas como estructuras sociales formadas por personas o entidades conectadas y unidas entre sí por algún tipo de relación o interés común,[4] son la vida del ecosistema y por tanto la parte más importante y que le da sentido. Los habitantes son los actores que se relacionan formando estas redes. Individuales, colectivos, empresas, administraciones y universidades se organizan horizontalmente para diseñar, construir y mantener el proyecto social y colaborativo que resuelve sus necesidades telecomunicativas, y garantiza el derecho universal de comunicación y acceso a la información.

Guifi.net es una comunidad (agregación de comunidades) autogestionada y soberana. Las personas poseen y utilizan la infraestructura. El poder y la responsabilidad están distribuidos. Cada usuario o comunidad es titular y propietario de la parte de la infraestructura que ha construido. Cada usuario o colectivo es responsable del uso que hace de la red y del contenido que incorpora.

Los actores forman una comunidad global abierta en la que cualquiera puede participar en igualdad de condiciones. Hay muchas formas de participar en la comunidad, con diferentes niveles de implicación y diferentes roles (por ejemplo: usuarios, profesionales, técnicos, inversores), pero siempre como participantes activos, no meros consumidores.

Conocimiento

Otro componente imprescindible es el conocimiento. Guifi.net es una fuente de conocimiento accesible y dinámica, sobre telecomunicaciones, informática, economía, entre otras. Como pasa con la infraestructura, la comunidad posee y utiliza el conocimiento. En este ecosistema se pretende minimizar la distancia entre la tecnología y la sociedad. Esto se dinamiza de forma virtual o presencial: por chat[5], por listas de correo[6], en una biblioteca multimedia[7], el canal de tv online[8], en presentaciones en eventos públicos, en el encuentro anual SAX[9], en la wiki[10], en quedadas y eventos puntuales, en las webs de las asociaciones o comunidades, etc.

Se genera un punto de encuentro entre expertos y profesionales y aprendices y novatos. También es un lugar de experimentación, para poner en práctica lo teórico. Del mismo modo es un lugar para generar nuevo conocimiento, compartirlo y aprender en comunidad.

Como es el caso de este Trabajo Final de Grado, estudiantes e investigadores comparten sus proyectos relacionados con guifi.net en la wiki[11].

Beneficios

El ecosistema se organiza en base a la economía del común, en la que los beneficios se garantizan para la comunidad. A modo de resumen, destaco estos beneficios:

- Reducción de costes, ya que se comparten recursos.
- Mayor calidad en la comunicación, ya que se reduce la competencia el el espacio radioeléctrico.
- Mayor control de los dispositivos de red y enlaces, que pertenecen a la comunidad.
- Menor individualismo (aunque también cabría), para crear colectividad.

- Comunicación en un entorno más seguro.
- Libertad para hacer, pensar, expresar y comunicar.
- Alternativa en caso de catástrofe.
- Participación activa y abierta.
- Facilitación de la autoprestación de servicios.
- Acceso al entorno rural.
- Menor dependencia de recursos externos.

Además, el valor de una red (Internet o guifi.net) aumenta exponencialmente con el número de personas que participan y del contenido y servicios que se aportan.

Es imprescindible aclarar que aunque se respetan todas las libertades de los habitantes del ecosistema, aplicando el acuerdo del Procomún de la RALN, se podrá bloquear tráfico que que supongo un abuso de mensajes no solicitados, contenidos inapropiados o ilegales y quieran causar un perjuicio o restringir las libertades de otros usuarios.

Legalidad

La Fundación Privada para la Red Abierta, Libre y Neutral guifi.net[12] (la Fundación) es una entidad legal y reguladora de guifi.net. Es una capa legal que cubre a todo el ecosistema del mundo externo, Internet. Y se encarga de velar por los principios que rigen la red.

Es un sistema autónomo en Internet, y un proveedor de Internet para la red de guifi.net. Es la interfaz de guifi.net en el Punto Neutro de Internet en Cataluña (CATNIX)[13]. Es una asociación sin ánimo de lucro que no gobierna guifi.net ni toma decisiones por la comunidad. Es una entidad con unas funciones específicas dentro de guifi.net, sin establecer ninguna relación jerárquica.

Un poco de historia y realidad actual de la red[14]

El proyecto empezó en 2004, con un grupo de personas interesadas en acceder a Internet desde sus granjas en Cataluña, donde a ningún ISP comercial le salía rentable extender su red, y por tanto no se podía acceder a Internet.

Decidieron crear un enlace inalámbrico desde esos lugares “lejanos” hasta el pueblo, donde sí llegaba la red para acceder a Internet. Poco a poco la red fue creciendo, creando red

comunitaria, compartiendo el acceso a Internet, y creando y consumiendo servicios y contenido dentro de esta red.

En 2007 se creó la Fundación. En 2009 se desplegó el primer enlace de fibra óptica. En este mismo año guifi.net pasó a formar parte del CATNIX, y la Fundación queda inscrita en el registro de operadores de telecomunicaciones de la Comisión del Mercado de las Telecomunicaciones. Hasta entonces se accedía a Internet a través de contratos convencionales de acceso a Internet con los ISP comerciales. El proyecto de guifi.net es Premio Nacional de Telecomunicaciones, entre otros.

Actualmente hay 33000 nodos operativos, y 60000 km de enlaces formando la infraestructura de guifi.net. La extensión de la red es mundial. Es la red abierta más grande del mundo actualmente. La gráfica de la figura 1, extraída de la web de guifi.net[15], muestra la curva de crecimiento de los nodos operativos desde 2004 hasta el 2017, en el mundo entero.

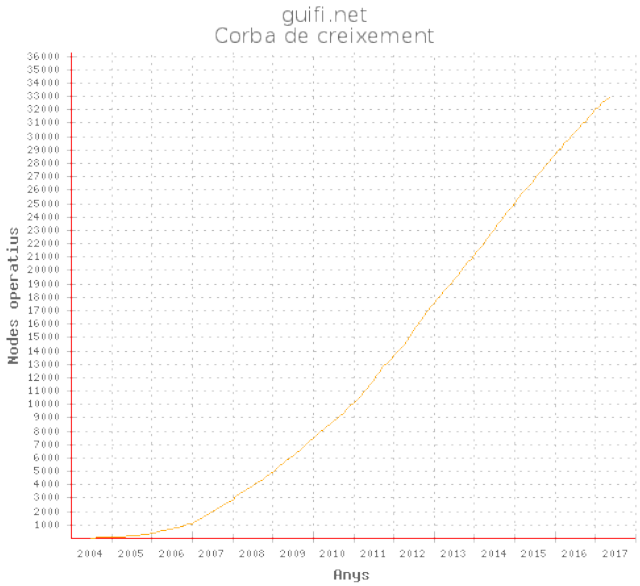


Figura 1: Curva de crecimiento de nodos operativos

En el mapa[16] de la figura 2 se ve la densidad de nodos en la Península Ibérica, donde están la mayoría de nodos de guifi.net.

Figura 2: Mapa de nodos en la Península Ibérica e Islas Baleares

Terminología propia de guifi.net

Guifi.net se utiliza una terminología específica que puede generar confusión y por tanto se definen a continuación una serie de conceptos que se utilizan en este documento.

Zona. Área geográfica o política. Las zonas tienen estructura jerárquica.

Comunidad. Es un grupo de individuos que se unen y organizan para gestionar el procomún. En este contexto el procomún son los elementos del ecosistema de guifi.net en una zona determinada. En una comunidad se generan redes sociales.

Nodo. En guifi.net es una ubicación donde hay algún aparato conectado a la red, como un ordenador o un encaminador. Un nodo tiene uno de estos estados: en construcción, proyectado, en pruebas, operativo, reservado.

Supernodo. Es un nodo, que en la topología de infraestructura establece enlaces troncales con otros supernodos y también con los nodos para que puedan acceder a la red.

Gateway. Encaminador que tiene interfaz externa hacia Internet. Está conectado al gateway de un ISP que da acceso a Internet.

La comunidad Guifi-Elx

Guifi-Elx es la comunidad Wi-Fi ciudadana de guifi.net en Elche. Nació en 2011, cuando un grupo de personas de Elche, libremente decidieron construir su propia red de telecomunicaciones independiente, abierta, libre y neutral. Esta red les permitiría conectarse entre ellos, aportar contenido y compartir recursos como el acceso a Internet. Desde entonces ha ido creciendo la comunidad, la infraestructura y su valor.

Se creó la red de Guifi-Elx como una isla de guifi.net en la zona de Elx. Es una isla porque sus nodos no tienen enlaces con otros nodos del resto de la red de guifi.net.[17] En el mapa[18] de la figura 3 se ven los nodos como puntos verdes y los enlaces como líneas verdes.

Figura 3: Mapa de nodos en la zona de Elx

A continuación se explica la **infraestructura de la red a fecha de febrero de 2017**.

La topología de esta red tiene una estructura jerárquica, en modo infraestructura, de manera que hay una capa troncal, formada por supernodos, y una capa de acceso, formada por los nodos.

Cada nodo o supernodo está instalado en el edificio de la casa de quien es el nodo. Y está formado por uno o varios encaminadores y una o varias antenas. Los nodos se conectan con enlaces Wi-Fi, a través de las antenas instaladas en las azoteas, al AP del supernodo con mejor conectividad. Y por cable proporcionan acceso a las viviendas. Los supernodos también proporcionan acceso a la red a una vivienda del edificio, por cable.

Para los enlaces inalámbricos se utiliza la tecnología Wi-Fi 5Ghz, de las bandas del espectro libres de licencia, para llegar a cualquier lugar de la ciudad y a las zonas rurales cercanas, y así facilitar el acceso universal y gratuito. Gratuito en referencia a que los usuarios no están obligados a pagar al contratar los servicios de una operadora comercial para acceder a la red, como sí pasa para acceder a Internet. Además, estos enlaces tienen un ancho de banda superior al que normalmente está disponible con un acceso a Internet convencional; son enlaces de 100 o 300Mbps.

La figura 4 representa la topología de la red de Guifi-Elx. Los supernodos están sobre los fondos verde y rosa en la figura 4. Los nodos están en la parte inferior de la topología. A la izquierda de los supernodos está el gateway de acceso a Internet, explicado más adelante en este capítulo en el apartado de la Asociación Xarxa Lliure Elx (en adelante, *la asociación*).

A continuación la descripción de cada nodo al detalle.

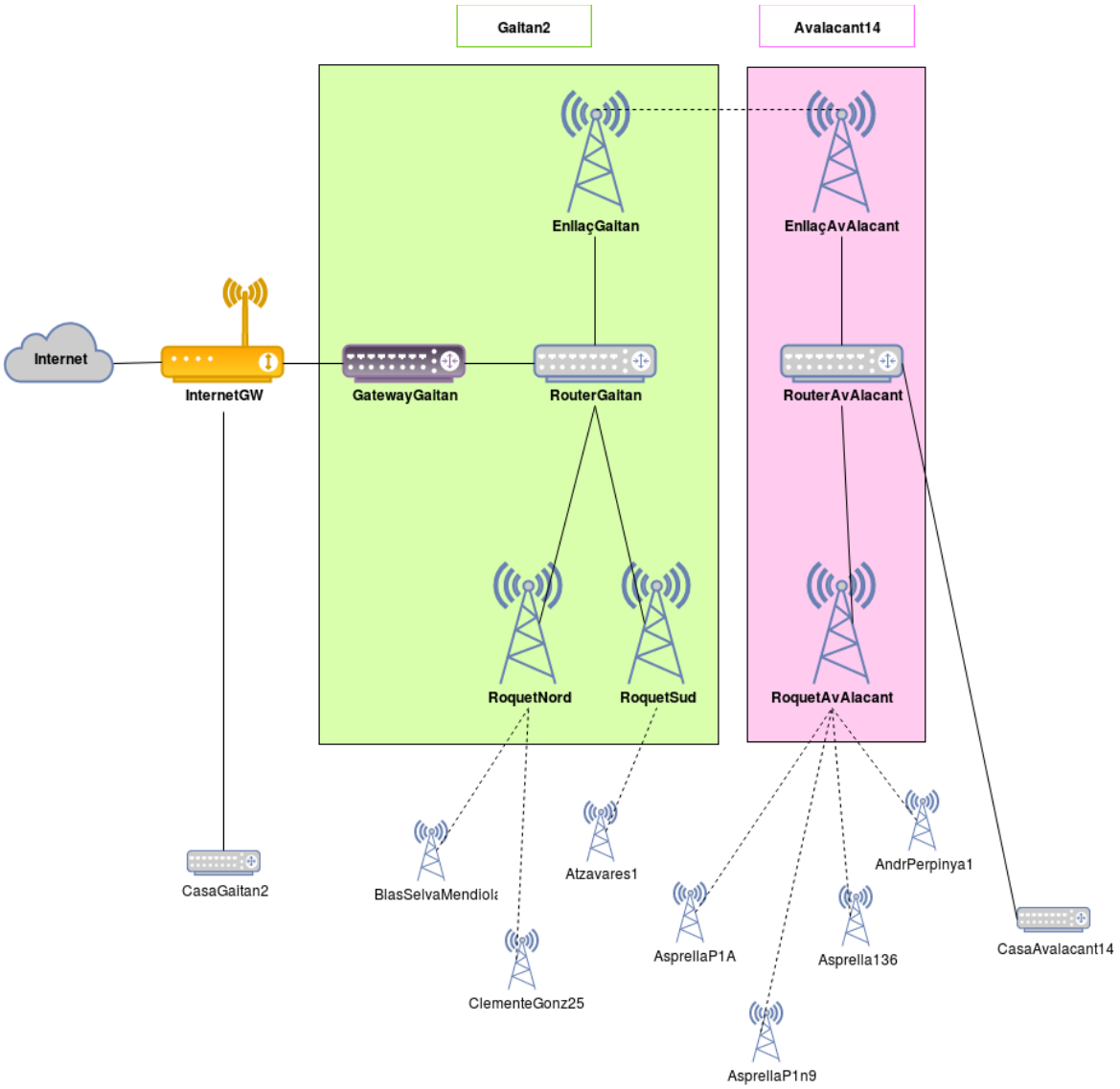


Figura 4: Nodos y supernodos de Guifi-Elx

Supernodos

Supernodo Gaitan2

Con fondo verde en la figura 4. La instalación está en la azotea de un edificio de 8 plantas. Hay mucha visibilidad, al norte hacia la el campus de la Universidad Miguel Hernández, y al sur hacia el Hospital, Santa Pola, La Marina, Ripoll. La figura 5 es una fotografía de la instalación de las antenas en un mástil en la azotea. Las figuras 6 y 7 son fotografías panorámicas de las vistas desde la azotea.

Figura 5: Instalación de antenas de Gaitan2

Figura 6: Vista noroeste desde Gaitan2

Figura 7: Vista hacia Ripoll desde Gaitan2

El supernodo está formado por 5 dispositivos:

- RouterGaitan. Es una RouterBoard Mikrotik RB750GL. El SO es RouterOs. Características del hardware[19]: Memoria 64MB. Procesador Atheros MIPS 24Kc 400MHz. Con 5 puertos Gigabit Ethernet. Con entrada PoE. Es el encaminador que interconecta el resto de dispositivos del supernodo.
- EnllaçGaitan. Es una radio Ubiquiti XM Nanostation M5. El SO es AirOs. Características del hardware[20]: Memoria 32MB SDRAM, 8MB Flash. Procesador Atheros MIPS 24Kc, 400MHz. Se alimenta con PoE pasivo. Con 2 puertos Fast Ethernet. Esta radio antena hace de puente para conectar con el supernodo Avalacant14, estableciendo un enlace troncal Wi-Fi a la radio antena homóloga del otro supernodo.
- RoquetNord y RoquetSud. Son dos radios Ubiquiti XM Roquet M5. El SO es AirOs. Características del hardware de las Roquet M5[21]: Memoria 64MB SDRAM, 8MB Flash. Procesador Atheros MIPS 24KC, 400MHz. Se alimentan con PoE pasivo. Con 1 puerto Fast Ethernet. Son las dos antenas sectoriales que hacen de AP para establecer enlace con los nodos.
- GatewayGaitan. PFSense 2.3.2-RELEASE (i386) FreeBSD 10.3-RELEASE-p5. CPU type: VIA Eden Processor 1000MHz. Con 3 puerots Fast Ethernet y 2 Gigabit Ethernet. Es el gateway de la asociación, para gestionar el acceso a Internet y la administración de la red.

Supernodo Avalacant14

Con fondo rosa en la figura 4. La instalación está en un edificio en la plaza Benidorm. La figura 8 es una fotografía de la instalación de antenas en un mástil en la azotea. La figura 9 es una fotografía de la vista desde a azotea.

Figura 8: Instalación de antenas de Avalacant14

Figura 9: Vista desde Avalacant

El supernodo está formado por tres dispositivos:

- RouterAvAlacant. Es una RouterBoard Mikrotik RB750. El SO es RouterOs. Características del hardware[22]: Memoria 64MB. Procesador Atheros MIPS 24Kc 400MHz. Con 5 puertos Gigabit Ethernet. Con entrada PoE. Es el encaminador entre el resto de dispositivos del supernodo.
- EnllaçAvAlacant. Es una radio Ubiquiti XM Nanostation Loco M5. El SO es AirOs. Características del hardware[23]: Memoria 32MB SDRAM, 8MB Flash. Procesador Atheros MIPS 24Kc, 400MHz. Con un puerto Fast Ethernet. Esta radio antena hace de puente para conectar con el supernodo Gaitan2, estableciendo un enlace troncal Wi-Fi a la radio antena homóloga del otro supernodo.

- RoquetAvAlacant. Es una radio Ubiquiti XM Roquet M5. El SO es AirOs. Características del hardware[24]: Memoria 64MB SDRAM, 8MB Flash. Procesador Atheros MIPS 24KC, 400MHz. Se alimentan con PoE pasivo. Con 1 puerto Fast Ethernet. Es la antena sectorial que hace de AP para establecer enlace con los nodos.

Nodos

Están representados en la parte inferior de la figura 4. Cada nodo está formado por una antena instalada la azotea del edificio, y que se conecta a través de un enlace Wi-Fi a la antena Roquet del supernodo para acceder al resto de la red de Guifi-Elx.

Además la antena conecta por Ethernet a las viviendas del edificio que quieran conectarse. A partir de aquí empieza la red privada que es gestionada por cada usuario. La instalación típica en la vivienda consiste en conectar un encaminador que haga de conmutador ethernet y de AP Wi-Fi para conectar diferentes dispositivos dentro de la casa.

Direccionamiento IP

La red Guifi-Elx es una red TCP/IP que utiliza IPv4 como protocolo de red (capa 3 en la pila de protocolos TCP/IP). Se sigue la propuesta de criterios de asignación de direcciones IPv4 de guifi.net[25] que evita conflictos con el direccionamiento en Internet y facilita al máximo la interconexión e interoperabilidad entre redes comunitarias. Por tanto, se reparten los tres bloques de direcciones privadas reservados por la IANA y definidos en la RFC1918[26] para diferentes usos en la red de guifi.net, de forma que se identifican tres capas: pública, de gestión y privada.

El bloque de direcciones 10.0.0.0/8 forma la *capa pública de guifi.net* (IMPORTANTE: *pública* dentro de la red de guifi.net y en ningún caso *pública* respecto a Internet). Esta capa permite conectividad de capa 3 más allá de la propia zona o comunidad, sin necesidad de utilizar NAT. El reparto de estas direcciones está automatizado para todo guifi.net con un autoservicio desasistido, para minimizar la intervención de administradores de red humanos y que sea lo más eficaz posible.

Para la zona de Elx, hay reservados dos rangos de direcciones[27]: 10.229.40.0/24 y 10.0.104.0/23. Las direcciones IP que se utilizan actualmente pertenecen al primer rango.

En la capa pública hay 6 subredes. Dos de ellas son las LAN de cada supernodo. Las otras cuatro son las subredes que forma cada AP en modo infraestructura en los supernodos y sus respectivos nodos cliente. En la figura 10 se presenta el direccionamiento de la capa pública en la topología de red.

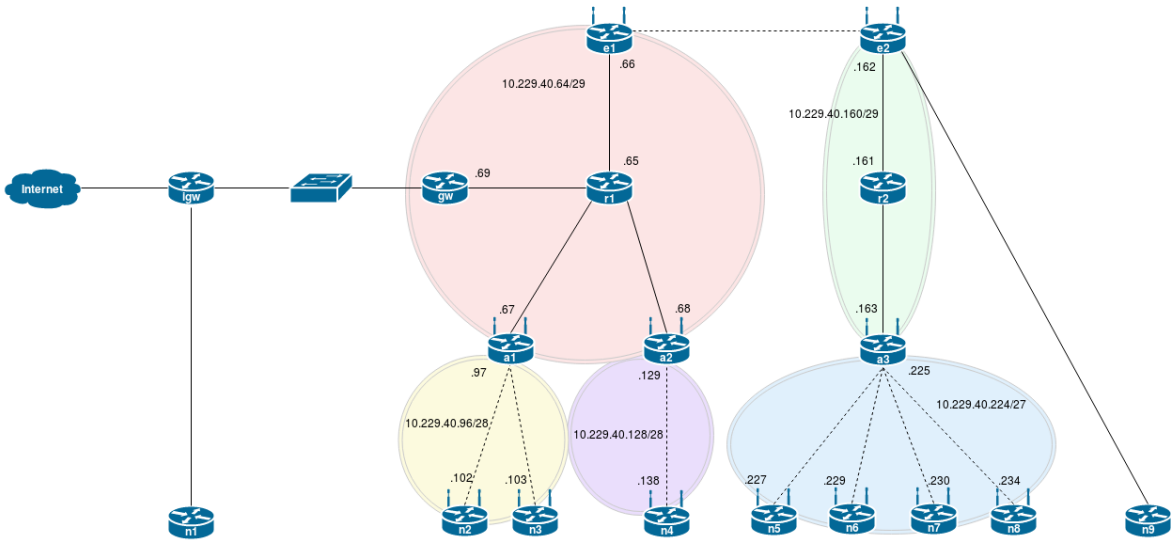


Figura 10: Direccionamiento de la capa pública de la red de Guifi-Elx

LAN del supernodo ElxGaitan2, 10.229.40.64/29:

- .65: RouterGaitan
- .66: EnllaçGaitan
- .67: RoquetNord
- .68: RoquetSud
- .69: GatewayGaitan

Subred del AP RoquetNord, 10.229.40.96/27:

- .97: RoquetNord
- .102: BlasSelvaMendiola
- .103: ClementeGonz25

Subred del AP RoquetSud, 10.229.40.128/27:

- .129: RoquetSud

- .138: Atzavares1

LAN del supernodo ElxAvAlacant14, 10.229.40.160/29:

- .161: RouterAvAlacant
- .162: EnllaçAvAlacant
- .163: RoquetAvAlacant
- .164: ServerAvAlacant

Subred del AP RoquetAvAlacant, 10.229.40.224/27:

- .225: RoquetAvAlacant
- .227: AsprellaP1A
- .229: AsprellaP1n9
- .230: Asprella136
- .234: AndrPerpinya1

El bloque de direcciones 172.16.0.0/12 forma la *capa de gestión*. Esta capa proporciona la conectividad de capa 3 dentro de la red comunitaria de Guifi-Elx. Las direcciones asignadas son únicas en la red de Guifi-Elx, y pueden ser usadas en otras redes y zonas; son direcciones privadas de la red Guifi-Elx. En la figura 11 se presenta el direccionamiento de la capa de gestión en la topología de red.

El bloque de direcciones 192.168.0.0/16 forma la capa privada: son para uso privado de los usuarios finales para su red local. En el caso del supernodo Gaitan2 se utiliza la subred 192.168.1.0/24 representada en la figura 12:

- .1: GatewayInternet
- .6: GatewayGaitan

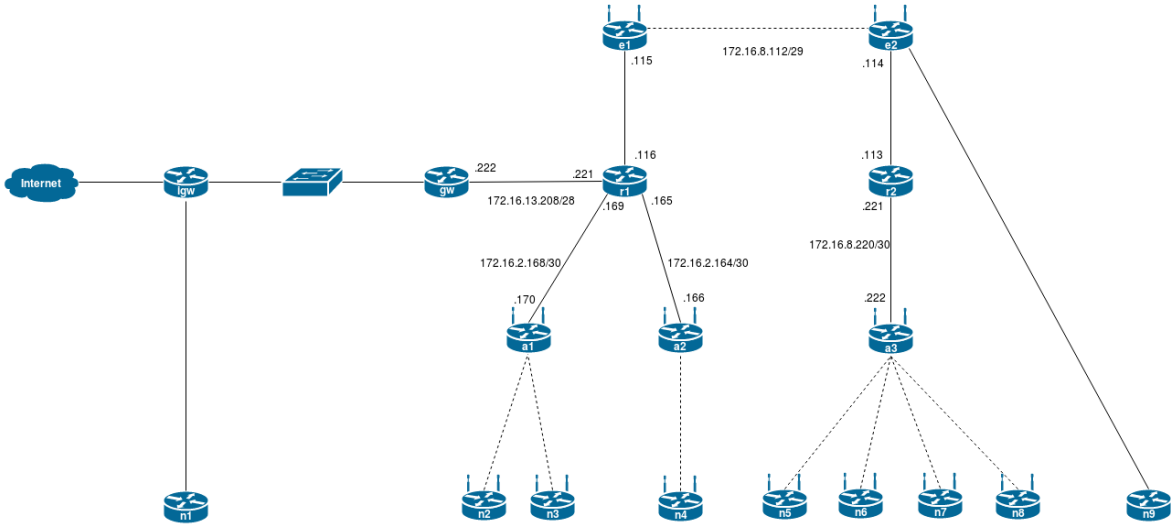


Figura 11: Direccionamiento de la capa de gestión de la red de Guifi-Elx

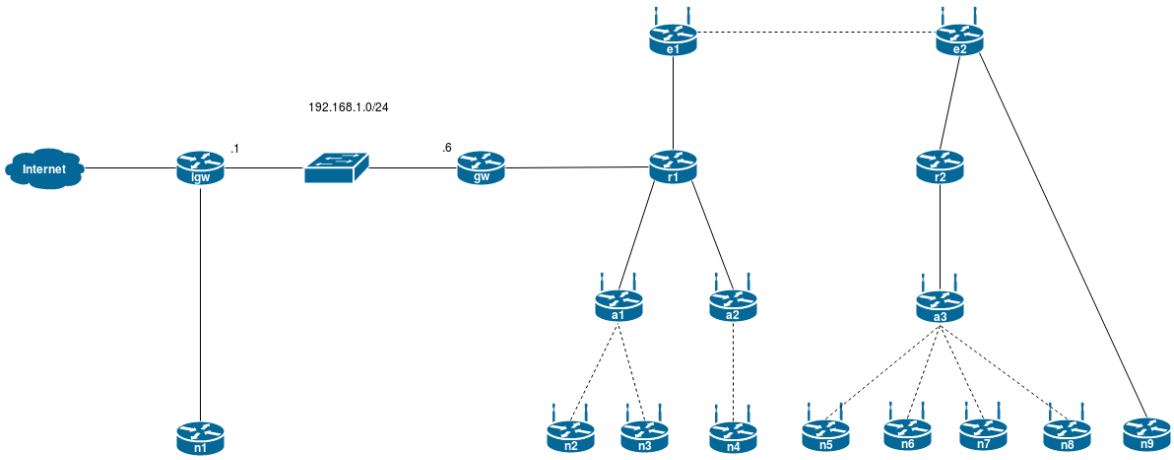


Figura 12: Direccionamiento de la capa privada del supernodo Gaitan2

Los protocolos de encaminamiento dinámico con que funciona la red son BGP y OSPF.

Actualmente el único servicio de red es acceso a Internet compartido.

La asociación Xarxa Lliure Elx

La asociación nació en 2011. Actualmente hay 9 socios, que pagan trimestralmente una cuota. La asociación se encarga del mantenimiento de la red. De forma que se garantice la conectividad de los nodos, actualizando el software de los dispositivos, haciendo pruebas de velocidad, manteniendo las instalaciones físicas de los supernodos y ayudando en la instalación de nuevos nodos. También se encarga de dinamizar la expansión de la red, a través de actividades o por Internet.

Además la asociación contrata con un ISP comercial (actualmente Orange) el servicio de acceso a Internet. A fecha de febrero de 2017, se hay contratados 50Mbps simétricos con FTTH. El gateway de acceso a Internet es propiedad de Orange. El gateway de la asociación hace de intermediario entre el gateway de acceso a Internet y la red de Guifi-Elx. Así, los nodos socios obtienen el servicio de acceso a Internet a través del gateway de la asociación.

El gateway de la asociación se encarga de los siguientes servicios: cortafuegos perimetral, almacenamiento de ficheros para gestión de la red, almacenamiento de copias de seguridad.

Financiación

Para cubrir los gastos monetarios iniciales se hizo un crowdfunding a través de la plataforma web Verkami[28]. Después, con las cuotas se cubren los gastos monetarios del mantenimiento de la capa troncal de la red. Los nodos son gestionados por cada usuario.

3. MEJORA DE LA RED

A partir de la red de Guifi-Elx en el estado explicado en el capítulo anterior, se propone llevar a cabo un proyecto de mejora de la red, el cual tiene cuatro objetivos principales.

Primero. Introducir en la red un dispositivo de respaldo al gateway de la asociación.

Segundo. Aumentar el ancho de banda de acceso a Internet contratado.

Tercero. Instalar un servidor web para alojar la web informativa de Guifi-Elx.

Cuarto. Desplegar un nuevo servicio en la red: QoS.

Servidor paralelo

Añadir un dispositivo de respaldo al gateway de la asociación consiste en configurar un nuevo dispositivo con los mismos servicios que hay en el gateway para que el funcionamiento de la red no se vea alterado en caso de caída del gateway. Lo llamaremos Servidor Paralelo. Se le asignan las direcciones IP de guifi.net que necesita: una pública de guifi.net y una de gestión. También se configura el routerGaitan para funcionar correctamente.

La asociación tiene un ordenador de placa reducida, un Odroid-U2. Sus características son: Procesador 1,7Ghz ARM Cortex-A9 Quad Cores. 2 puertos USB2.0. Con 1 puerto Fast Ethernet. Éste será el servidor paralelo. El sistema operativo elegido es Debian 8 (Jessie). Es un sistema operativo estable, con una comunidad activa, buena documentación y muy usado en guifi.net.

Una vez integrado en la red el servidor paralelo, la topología de red de Guifi-Elx cambia. Las figuras 13, 14, 15 y 16 ilustran el cambio. El direccionamiento asignado es:

- 10.229.40.70/29 ServidorParalel para la capa pública de guifi.net
- 172.16.8.74/30 SevidorParalel para la capa de gestión
- 192.168.1.7 SevidorParalel para la capa privada
- 172.16.8.74/30 RouterGaitan para la capa de gestión

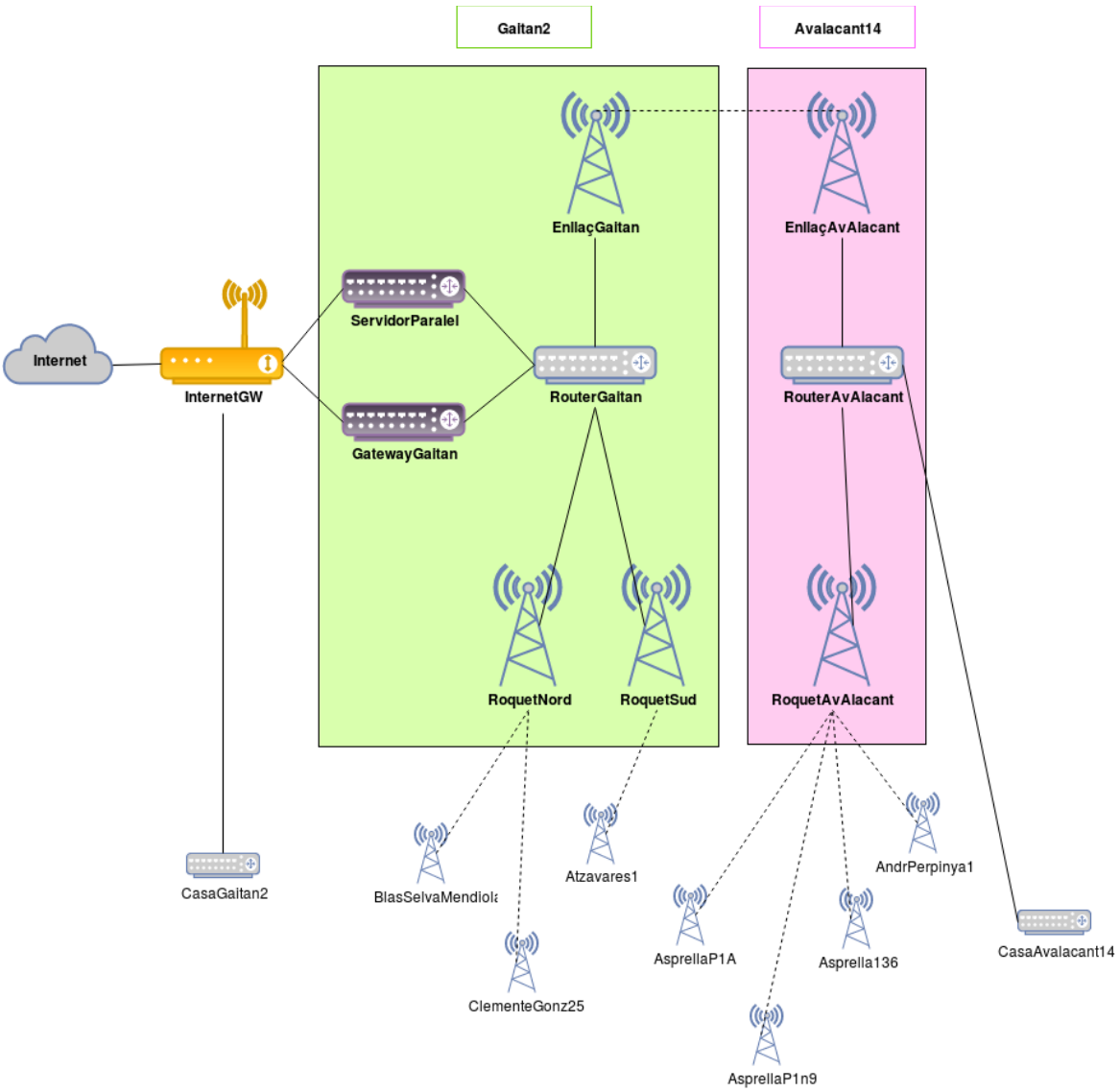


Figura 13: Nodos y supernodos de Guifi-Elx con el Servidor Paralelo

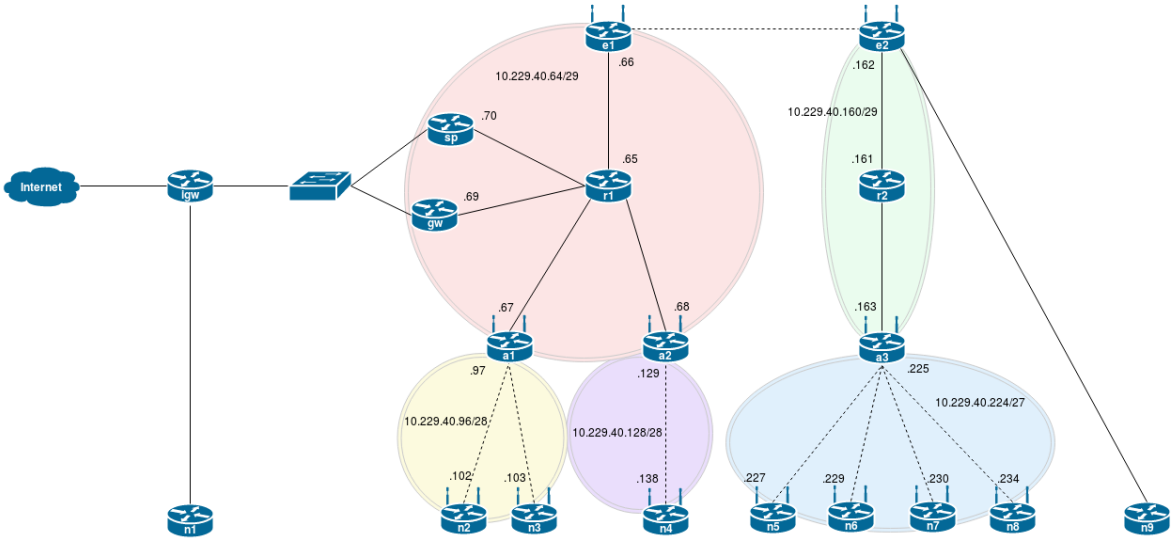


Figura 14: Direccionamiento de la capa pública de la red de Guifi-Elx con el Servidor Paralelo

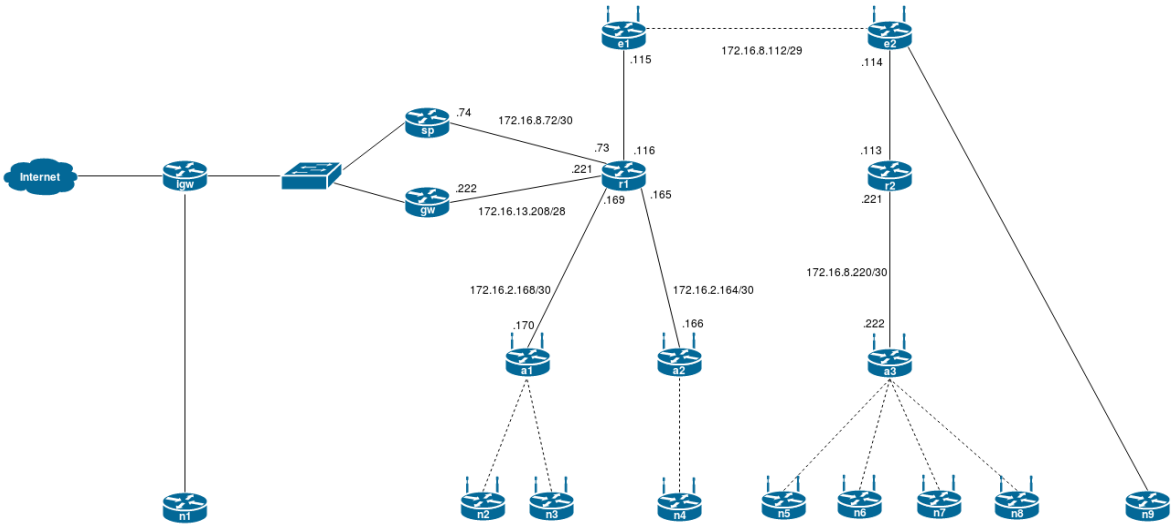


Figura 15: Direccionamiento de la capa de gestión de la red de Guifi-Elx con el Servidor Paralelo

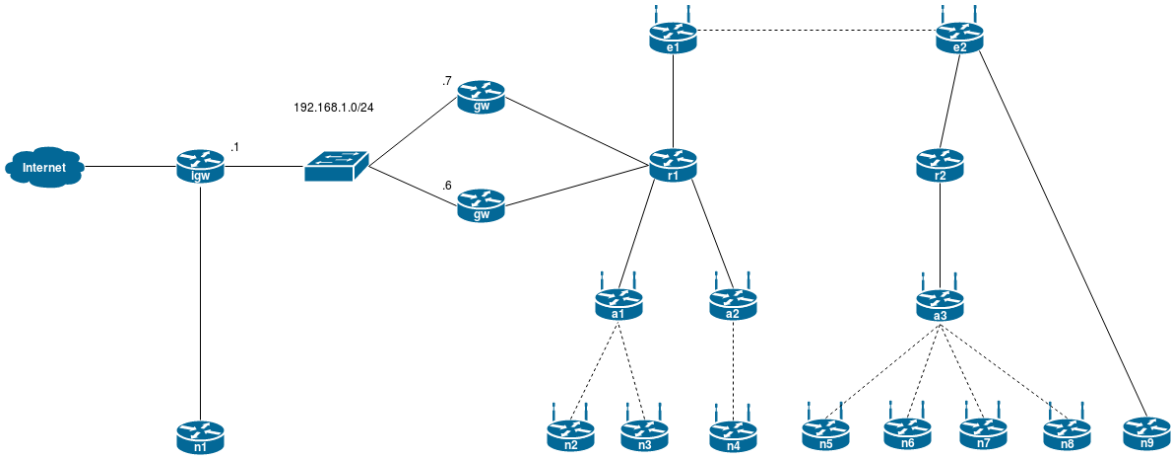


Figura 16: Direccionamiento de la capa privada del supernodo Gaitan2 con el Servidor Paralelo

El Anexo I es la guía de instalación y configuración del servidor paralelo con los servicios y aplicaciones básicas.

Aumento de ancho de banda a Internet

Se decide renovar contrato con Orange, de forma que ahora habrán disponibles 300Mbps simétricos de FTTH, en vez de 50Mbps simétricos de FTTH. Al cambiar Orange el dispositivo gateway de Internet en la casa por otro más potente, se han de volver a configurar los reenvíos de puerto.

Servidor web

Se instala un servidor web en el nuevo servidor paralelo para alojar la página web de Guifi-Elx, que está en desarrollo. El Anexo II de este documento es el proceso de instalación. La URL para acceder a la página web es: https://elx.guifi.net

4. DESPLIEGUE DE QOS EN LA RED

El Grupo de Trabajo de Ingeniería de Internet (IETF[29]) y la Unión Internacional de Telecomunicaciones (ITU[30]) definen la QoS (Quality of Service, Calidad de Servicio) así:

UIT-T G.1000: "Efecto global de la calidad de funcionamiento de un servicio, que determina el grado de satisfacción de los usuarios".

RFC 2386: "Conjunto de requisitos del servicio que debe cumplir la red en el transporte de un flujo".

A partir de varias definiciones se extrae que QoS se refiere a la capacidad de una red para ofrecer mejor servicio a un tráfico de red seleccionado[31]. Mediante diferentes técnicas de QoS se proporciona el nivel de servicio deseado para el tráfico de una aplicación, usuario o flujo de datos. De este modo se hace un uso más eficiente de los recursos, y se puede evitar la saturación.

Evaluar la QoS supone medir la calidad de la transmisión y la disponibilidad del servicio en cuestión, en términos de ancho de banda, pérdida de paquetes, retardo y variación del retardo.

Aunque la instalación en la red de soluciones de QoS no es necesaria, es recomendable para evitar los inconvenientes de una red de "mejor esfuerzo", que es en la que no se aplica ninguna estrategia de QoS. Sin QoS todos los paquetes son tratados igual, sin establecer ninguna prioridad de un tipo de tráfico sobre otro. Pero no todos los tipos de tráfico tienen los mismos requisitos. Por ejemplo, las aplicaciones de tiempo real requieren tiempos de respuesta muy bajos y un flujo de bits constante, mientras que el tráfico de correo electrónico es más tolerante a retardos. Por lo tanto, es beneficioso repartir los recursos según las necesidades.

Este proyecto no busca un procedimiento de QoS para aplicarlo a toda la red pública de Guifi-Elx, aunque es una propuesta muy interesante para desarrollar en el futuro. La comunidad de Guifi-Elx, formando parte de guifi.net, cumple el acuerdo de interconexión "Procomún de la Red Abierta Libre y Neutral"[32]. En este acuerdo se describe una base de cómo aplicar QoS en la red.

Este proyecto da una solución de QoS como servicio implantado para mejorar el servicio de acceso a Internet a los socios de la asociación. Por lo tanto se aplica en el gateway de la

asociación. El tráfico que se pretende controlar y gestionar es el tráfico entre los nodos socios e Internet. Todo este tráfico pasa por el gateway de la asociación, a excepción del tráfico a Internet del socio de CasaGaitan2 en el edificio donde está el supernodo, que al tener el gateway de Internet en casa este socio se conecta directamente al AP Wi-Fi del gateway de Internet. Por tanto, es en el gateway de la asociación donde se desplegará el servicio de QoS.

Uno de los mecanismos de QoS, aplicado en este proyecto, es clasificar el tráfico de red en categorías y diferenciar cómo es procesado el tráfico según la categoría a la que pertenece. Supone controlar el tráfico de la red, para establecer prioridades y para mejorar el aprovechamiento eficiente del recurso de ancho de banda y gestionar su reparto. De este modo se evitan abusos y saturación. Otro de los mecanismos aplicados es la garantía de un ancho de banda mínimo. Otros son los mecanismos de QoS que no se aplican en este proyecto, como la ingeniería de tráfico o el control de la congestión.

Los pasos a seguir para el despliegue del servicio de QoS son:

1- Analizar los requisitos de red.

2- Definir las políticas.

3- Diseñar la configuración de las políticas.

4- Probar las políticas.

5- Implementar las políticas.

6- Monitorizar y ajustar.

Estas fases son desarrolladas a lo largo del capítulo.

Análisis de requisitos de red

Este análisis se hace con todos los socios de la asociación. Se analizan los requisitos de red para ofrecer un buen servicio de acceso a Internet compartido. El análisis se basa en una serie de cuestiones:

- ¿Qué tipo de tráfico se quiere permitir hacia Internet?

- ¿Qué tipos de tráfico hace falta diferenciar? ¿Cómo se han de priorizar los diferentes tipos de tráfico?

- ¿Se establece ancho de banda mínimo garantizado?

- ¿Cómo se reparte el ancho de banda entre los socios?

- ¿Qué aplicaciones utilizan los usuarios? Como la navegación web, transferencia de archivos por FTP, distribución de archivos por P2P, videoconferencia...

A partir de estas preguntas básicas y de un análisis con las herramientas de PfSense para identificar los protocolos activos y usados en el tráfico hacia Internet, se establecen los requisitos de red consensuados y aprobados por los técnicos y socios de la asociación.

Se permitirá el tránsito de todo tipo de tráfico hacia Internet. La priorización del tráfico se basa en estas categorías por orden de importancia:

1. Tráfico de control y mantenimiento de la red: DNS, ICMP, NTP, SSH, puertos de administración y direcciones IP de administración de los dispositivos de red.
2. Voz sobre IP: SIP, IAX.
3. Comunicación en tiempo real basada en texto: cliente XMPP, cliente XMPPS, Retroshare, IRC.
4. Correo electrónico: IMAP, IMAP sobre TLS, POP3, POP3 sobre SSL, SMTP, SMTPS.
5. Web: HTTP, HTTPS.
6. Transferencia de archivos: FTP, TFTP.
7. Otros tráficos detectados.

Al hacer uso compartido del acceso a Internet es importante hacer un reparto justo y equilibrado del recurso a compartir: el ancho de banda. Lo más justo es hacer un reparto equitativo del ancho de banda entre los socios, es decir, entre los nodos socios. Por tanto, para que no haya un abuso del ancho de banda por parte de ningún nodo, es necesario establecer un ancho de banda mínimo garantizado. De forma que cada nodo tenga acceso al máximo de ancho de banda posible, y como mínimo a su parte proporcional.

El nodo CasaGaitan es un caso diferente. El tráfico a Internet de este nodo socio no pasa por el gateway de la asociación como el resto de tráfico del resto de nodos. Como el gateway de Internet está instalado en la casa, los usuarios que acceden a Internet desde la

casa lo hacen sin pasar por el gateway de la asociación. Para que este tráfico no tenga un trato diferente y mejor que el resto de nodos es necesario modificar la topología a nivel físico. Los cambios no se incluyen en este proyecto. Se proponen como proyecto futuro. Los cambios necesarios para que el tráfico sea tratado igual para todos los nodos socios son:

- No utilizar el gateway de Internet como AP del socio de CasaGaitan2. El gateway de Internet pasa a ser responsabilidad de la asociación.
- Instalar un dispositivo diferente para proporcionar el acceso a la red Guifi-Elx para el socio de CasaGaitan2. Este dispositivo es responsabilidad del socio y no de la asociación.
- Instalar un cable Ethernet desde este nuevo dispositivo en la casa hasta la azotea donde está el RouterGaitan.
- Configurar el nuevo dispositivo y el RouterGaitan para el correcto funcionamiento. Asignar direcciones IP de guifi.net al nuevo dispositivo.

El acceso de los nodos a la red de Guifi-Elx quedaría como se muestra en la figura 17.

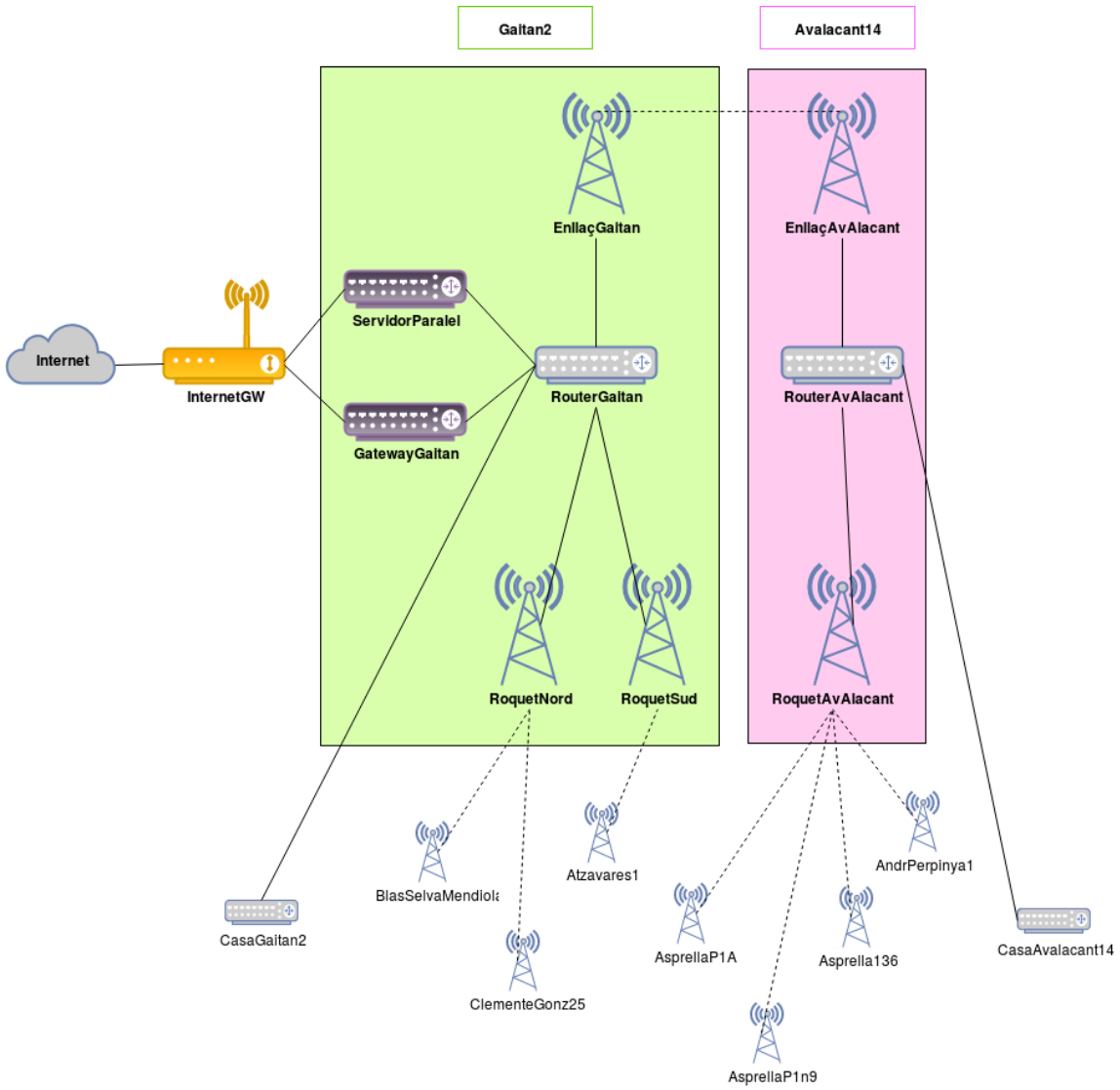


Figura 17: Nodos y supernodos de Guifi-Elx con cambio de dispositivo en CasaGaitan2

Definición de las políticas de QoS

A partir del análisis de requisitos de la red y de los socios se establecen las políticas de reparto de ancho de banda en el gateway de la asociación. Estas políticas se aplican en la interfaz de salida del gateway de la asociación hacia Internet.

Primero se diferencia el tráfico en dos tipos generales, en este orden de prioridad:

A- Tráfico de control y mantenimiento de la red: DNS, ICMP, NTP, SSH, puertos de administración y direcciones IP de administración de los dispositivos de red.

B- Tráfico masivo.

Para el tráfico masivo se hace un reparto equitativo del 90% del ancho de banda de acceso a Internet (300Mbps simétricos) entre los nodos socios (9 actualmente), de forma que cada nodo tenga un ancho de banda mínimo garantizado de (300*0,9/9=30) 30Mbps simétricos. El máximo posible sería el 90% de 300, 270Mbps, pero las capacidades físicas de la infraestructura limitan a 100Mbps, ya que las interfaces de los dispositivos son Fast Ethernet.

Además, para cada nodo socio, se clasifica el tráfico de la siguiente forma, priorizando en este orden y reservando un porcentaje de ancho de banda:

1. Voz sobre IP: SIP, IAX. 10%
2. Comunicación en tiempo real basada en texto: cliente XMPP, cliente XMPPS, Retroshare, IRC. 10%
3. Correo electrónico: IMAP, IMAP sobre TLS, POP3, POP3 sobre SSL, SMTP, SMTPS. 20%
4. Web: HTTP, HTTPS. 50%
5. Transferencia de archivos: FTP, TFTP. 10%
6. Otros tráficos detectados.

Configuración de las políticas de QoS

El siguiente paso de la instalación del servicio de QoS es la implementación de las políticas con las herramientas concretas del sistema operativo, tc e iptables. Este apartado del documento refleja el estudio de las herramientas y el proceso de generación de la configuración.

tc (traffic control) es una aplicación de GNU/Linux, para mostrar y manipular la configuración del control del tráfico en el kernel de Linux.[33]

El Control del Tráfico consiste en[34]:

- Shaping (moldear). Se controla la velocidad de transmisión del tráfico moldeado. Se puede usar tanto para reducir el ancho de banda disponible como para regular las ráfagas de tráfico para mejorar el comportamiento de la red. El moldeado sucede en sentido de salida del tráfico por una interfaz de red.

- Planificar. Planificando la transmisión de los paquetes se puede mejorar la interactividad del tráfico que lo necesite, mientras se garantiza ancho de banda para las transferencias masivas. Reordenar y priorizar son sinónimos en este caso, y sucede sólo en sentido de salida del tráfico por una interfaz de red.

- Policing. Se ocupa del tráfico entrante del mismo modo que el moldeado se ocupa de la transmisión de tráfico. Ocurre en sentido de entrada.

- Descartar. El tráfico que excede un determinado ancho de banda, puede ser descartado inmediatamente, tanto en entrada como en salida.

De estas técnicas de control del tráfico, se usan el moldeado y la priorización para implementar las políticas descritas anteriormente.

Para eludir la complejidad de tc, existen diversas herramientas que proporcionan una interfaz de configuración de nivel superior. Estas aplicaciones limitan el uso de tc e iptables según las funcionalidades que implementen, aunque también proporcionan un entorno de configuración más amable para los humanos. Algunos ejemplos son: tc-config[35], FireQOS[36], Wonder Shaper[37], tcng[38], Shorewall[39]. Sin embargo, como primera solución de este proyecto, se genera la configuración de QoS sin utilizar estas aplicaciones. Queda como propuesta para trabajo futuro el estudio y la utilización de estas herramientas.

Para moldear y priorizar el tráfico se manejan las colas de paquetes en la interfaz seleccionada. Con la gestión de colas se determina la forma en que ENVIAMOS datos. Es importante darse cuenta de que sólo se puede controlar el tratamiento de los datos que se transmiten. De la manera en que funciona Internet, no tenemos control directo sobre lo que la gente nos envía. Es como en el buzón de correo físico de casa. La única forma de influir al mundo para modificar la cantidad de correos que me envían sería contactando con todo el mundo.[40]

Por esto, en el escenario descrito en este proyecto se trata el tráfico saliente de la interfaz hacia Internet del gateway de la asociación, es decir, el tráfico hacia Internet. Así, como consecuencia, se controla el tráfico entrante de respuesta también.

El procesado de tráfico con la herramienta tc se controla con tres tipos de objetos: disciplinas de encolado (qdisc), clases (class) y filtros (filter).[41]

- Disciplinas de encolado (qdisc). Son elementales para entender el Control del Tráfico. Siempre que el kernel necesita enviar un paquete a una interfaz, éste es encolado hacia la qdisc configurada para la interfaz. Inmediatamente después, el kernel intenta coger tantos paquetes como pueda de la qdisc, para pasarlos al controlador del adaptador de red.

- Clases. Algunas qdisc pueden contener clases, que contienen más qdisc (entonces el tráfico puede ser encolado en una de las qdisc internas, que están dentro de las clases). Cuando el kernel intenta desencolar un paquete de una qdisc con clases, el paquete puede venir de una de las clases. Por ejemplo, una qdisc puede priorizar cierto tipo de tráfico tratando de desencolar unas clases antes que otras.

- Filtros. Las qdisc usan filtros para determinar dónde encolar un paquete. Cuando un paquete llega a una clase con subclases, necesita ser clasificado. El filtrado es uno de los métodos para conseguirlo. Se va llamando a todos los filtros asociados una clase hasta que uno de ellos devuelve un veredicto. Si no se hace ningún veredicto, pueden haber otros criterios disponibles. Esto es diferente para cada qdisc. Es importante destacar que los filtros están dentro de las qdisc, y por tanto no son los que controlan lo que pasa.

El proceso de control del tráfico se compone de tres pasos:[42]

- Se crean las colas. Con sus prioridades y parámetros.

- Se identifica y se marca el tráfico que pasa por la interfaz.

- Se clasifica el tráfico y se envía a la cola apropiada según la marca.

A menos que queramos una configuración QoS específica, el kernel de Linux usa el planificador de colas pfifo_fast por defecto para cada interfaz. Se puede consultar las qdisc configuradas para una determinada interfaz *$DEV*, con el comando:

```
tc qdisc show dev $DEV
```

La salida para la configuración por defecto es:

```
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap 1 2 2 2 1 2
0 0 1 1 1 1 1 1 1 1
```

La qdisc pfifo_fast es una qdisc de encolado simple sin clases, por lo que no se le pueden añadir otras qdisc con el comando tc. Sin embargo, esta cola tiene 3 'bandas', con prioridades diferentes. En cada banda se aplican las reglas FIFO: "primero en entrar, primero en salir" ("First In, First Out"). La prioridad de cada paquete procesado se basa en el campo ToS del paquete; modificar este campo puede ser útil. El kernel atiende a los bits ToS de los paquetes IP, y se encarga de insertar los paquetes en la banda correspondiente. Así, mientras haya paquetes esperando en la banda 0, la banda 1 no será procesada, y lo mismo para las bandas 1 y 2.[43]

A continuación se explica la construcción del guión de comandos de configuración de los parámetros de QoS descritos en las políticas con las herramientas tc e iptables.

Se declaran tres variables con los valores concretos del escenario. La interfaz del gateway sobre la que se aplicará la configuración.

```
DEV=eth0
```

El límite de velocidad de subida 100Mbps (100 y no 300 por las limitaciones de la infraestructura), establecido por el contrato de acceso a Internet, se expresa en kbits (kilobits por segundo) para trabajar con números enteros minimizando los errores por redondeo a las unidades.

```
UPRATE=100000
```

Y la cantidad de socios actualmente en la red.

```
N_SOCIOS=9
```

Se elimina cualquier configuración de qdisc anterior desde la raíz.

```
tc qdisc del dev $DEV root
```

La primera distinción de tráfico que hay en las políticas es tráfico A y B. Tráfico de control y mantenimiento de la red, y tráfico masivo. En la raíz se necesita una qdisc con subclases. Además, el tráfico B tiene límites de ancho de banda.

La qdisc prio es una posibilidad. Desencolará primero el tráfico A y luego el B. Pero esta disciplina no permite crear sólo dos bandas (clases) ni moldear. Cuando se intenta con el comando:

```
tc qdisc add dev $DEV root handle 1: prio bands 2
```

La salida indica que no es posible:

```
RTNETLINK answers: Invalid argument
```

La mejor opción es la disciplina de encolado htb, que permite controlar el uso del ancho de banda en las clases. El tráfico que no sea clase 1 (tráfico A) es clase 2 (tráfico B) por defecto.

```
tc qdisc add dev $DEV root handle 1: htb default 2
```

El tráfico A es procesado por la clase 1:1, con la mayor prioridad (prio 1) y clasificado por un filtro fw. Se explica el filtro más adelante. Además, este tráfico es gestionado por una disciplina sfq, de forma que ningún flujo domine sobre otro, mitigando un ataque DoS. Para hacer el reparto justo, la disciplina sfq identifica cada flujo con dirección de origen, dirección de destino y puertos de origen y destino. Esta clase tiene el máximo rate.

```
tc class add dev $DEV parent 1: classid 1:1 htb rate $
{UPRATE}kbit prio 1

tc qdisc add dev $DEV parent 1:1 handle 10: sfq perturb 10

tc filter add dev $DEV parent 1: protocol ip prio 1 handle 1
fw flowid 1:1
```

El tráfico B es el resto del tráfico, el tráfico masivo, por lo tanto se puede tomar como la clase por defecto (default 2). Con menor prioridad (prio 2), y con un límite de ancho de banda (rate) que permite prestar si está libre. El límite rate indica el máximo de ancho de banda que la clase podrá utilizar cuando el enlace esté lleno. Para el tráfico B este límite es el 90% del total. El límite ceil es el límite máximo al que la clase podrá transmitir, sin prestar, y que por defecto es igual al rate.

```
tc class add dev $DEV parent 1: classid 1:2 htb rate $
(($UPRATE*9/10))kbit prio 2
```

Para gestionar el tráfico B se utiliza una qdisc htb con identificador :20. Se explica más adelante.

```
tc qdisc add dev $DEV parent 1:2 handle 20: htb default 1
```

La prioridad en una clase indica la importancia de una clase sobre otra al utilizar el ancho de banda libre de otras clases que no lo usan y lo prestan. La prioridad no afecta al ancho de banda mínimo que se asigna a la clase (rate). La prioridad sirve para determinar el reparto del ancho de banda libre disponible. A menor valor del número, mayor prioridad.

La prioridad en un filtro afecta al orden en que se consultan los filtros asociados a una misma qdisc. A menor valor del número, mayor prioridad.[44]

La figura 18 muestra el árbol de objetos para controlar el tráfico de la interfaz creados hasta este punto.

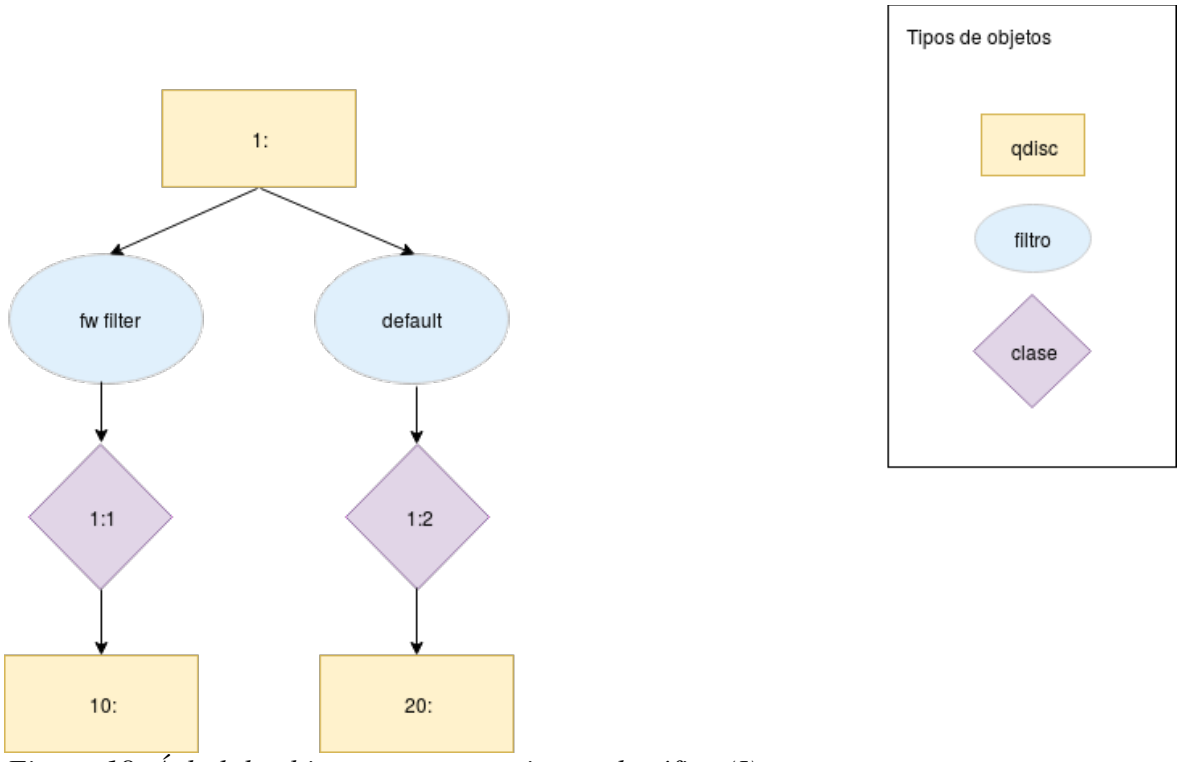


Figura 18: Árbol de objetos tc para gestionar el tráfico (I)

Para determinar qué clase ha de procesar cada paquete, se usan los filtros. Un filtro de paquetes es un software que examina la cabecera de los paquetes según van pasando, y decide la suerte del paquete completo.[45] Hay varios tipos de filtros tc (basic, bpf, cgroup, flow, fw, route, rsvp, tcindex, u32), explicados en el manual de Linux tc(8). El filtro fw permite clasificar paquetes según una marca (fwmark) asignada por el cortafuegos (con iptables). Si la marca coincide con el criterio del filtro tc, el filtro se aplica. Usar este filtrado con iptables en lugar del filtrado directamente con tc proporciona flexibilidad al separar la clasificación del filtrado. Además, es necesario utilizar el filtro fw para aplicar los criterios de filtrado antes de que se haga NAT de origen en la interfaz y los filtros que comparen la dirección IP de origen no tengan el efecto deseado. Más adelante se explica por qué el filtro fw da solución a esto.

La distribución estándar de iptables cuenta con varios módulos para el marcado: CONNMARK, MARK y IPMARK. CONNMARK se usa para marcar todos los paquetes de una misma conexión. MARK se usa para marcar a nivel de paquete. IPMARK se usa para marcar paquetes basándose en la IP, pudiendo emplear máscaras para reducir el número de líneas.[46]

Los paquetes del tráfico A tienen la marca 1. La pregunta es: ¿Marcar con MARK o CONNMARK? Hay escasa documentación sobre esta cuestión. Es en los foros de discusión en Internet donde se trata el tema. Como dice un usuario en el foro de discusión de Gentoo[47], en el caso en que todos los paquetes de una conexión deban estar marcados con el mismo valor, utilizar CONNMARK supone aprovechar el trabajo que hace el Sistema de Seguimiento de Conexiones del kernel Linux, conntrack[48], para etiquetar los paquetes. De forma que no hace falta hacer el procesado paquete a paquete por las reglas de iptables, lo que supondría una tarea redundante.

En el contexto en que se aplicarán estas reglas, una misma conexión siempre tendrá un mismo valor de marca. Por esto se usa CONNMARK. Se hace un filtrado de paquetes con estado.

Cómo funciona CONNMARK.[49] Netfilter tiene una tabla de estado en memoria, que usa para recordar el estado de una conexión y de este modo se pueden identificar los paquetes que pertenecen a una misma conexión. Una vez que se haya marcado el primer paquete de una conexión, se escribe esta marca en la tabla de estado del cortafuegos. Y así se podrán marcar los siguientes paquetes.

El sistema de seguimiento de conexión también funciona para los paquetes UDP. Aunque el protocolo UDP es no orientado a conexión, el sistema de seguimiento considera conexiones UDP teniendo en cuenta parámetros como IP origen/destino, puerto origen/destino, timestamp, entre otros.

Cómo utilizar iptables para marcar los paquetes con CONNMARK.[50] CONNMARK tiene varios parámetros: *set-mark* y *save-mark* se usan en los paquetes nuevos y *restore-mark* en paquetes relacionados.

Hay dos marcas diferentes: marcas de netfilter y marcas de seguimiento de la conexión. Connmark pone la marca de seguimiento de la conexión. *Restore-mark* copia la marca de seguimiento de conexión en la marca netfilter. Para filtrar los paquetes con tc es necesaria la marca netfilter. *Save-mark* hace lo contrario que restore, copia la marca netfilter a la marca de seguimiento de la conexión. Esto se debe usar sólo cuando sea necesario, no por defecto, para los paquetes de nueva conexión. De este modo sólo se marcan (set y save) los

paquetes NEW, y los paquetes ESTABLISHED y RELATED obtienen la marca de su conexión (restore).

Las reglas de marcado con iptables pertenecen a la tabla mangle[51], que se usa para la alteración especializada de paquetes. De las cadenas que tiene esta tabla, la que interesa en este escenario es la cadena POSTROUTING, para alterar los paquetes que salen por una interfaz determinada del dispositivo. En este escenario, se indica la interfaz de salida eth0 hacia Internet.

Es importante destacar que las reglas de la cadena de POSTROUTING de la tabla mangle se evalúan antes que las reglas de la cadena POSTROUTING de la tabla nat. Es necesario aplicar los filtros de tráfico por IP antes de hacer la transformación NAT a la dirección IP de la interfaz. La figura 19 muestra el proceso de atravesar las cadenas y tablas de iptables.

Cuando un paquete pasa por una interfaz se van examinando las reglas del cortafuegos hasta que el paquete coincide con el criterio de una regla que tenga una acción final (ACCEPT, DROP o RETURN), o hasta hasta llegar a la política por defecto. Se usa la acción RETURN para que no se compare cada paquete con más reglas de las necesarias.

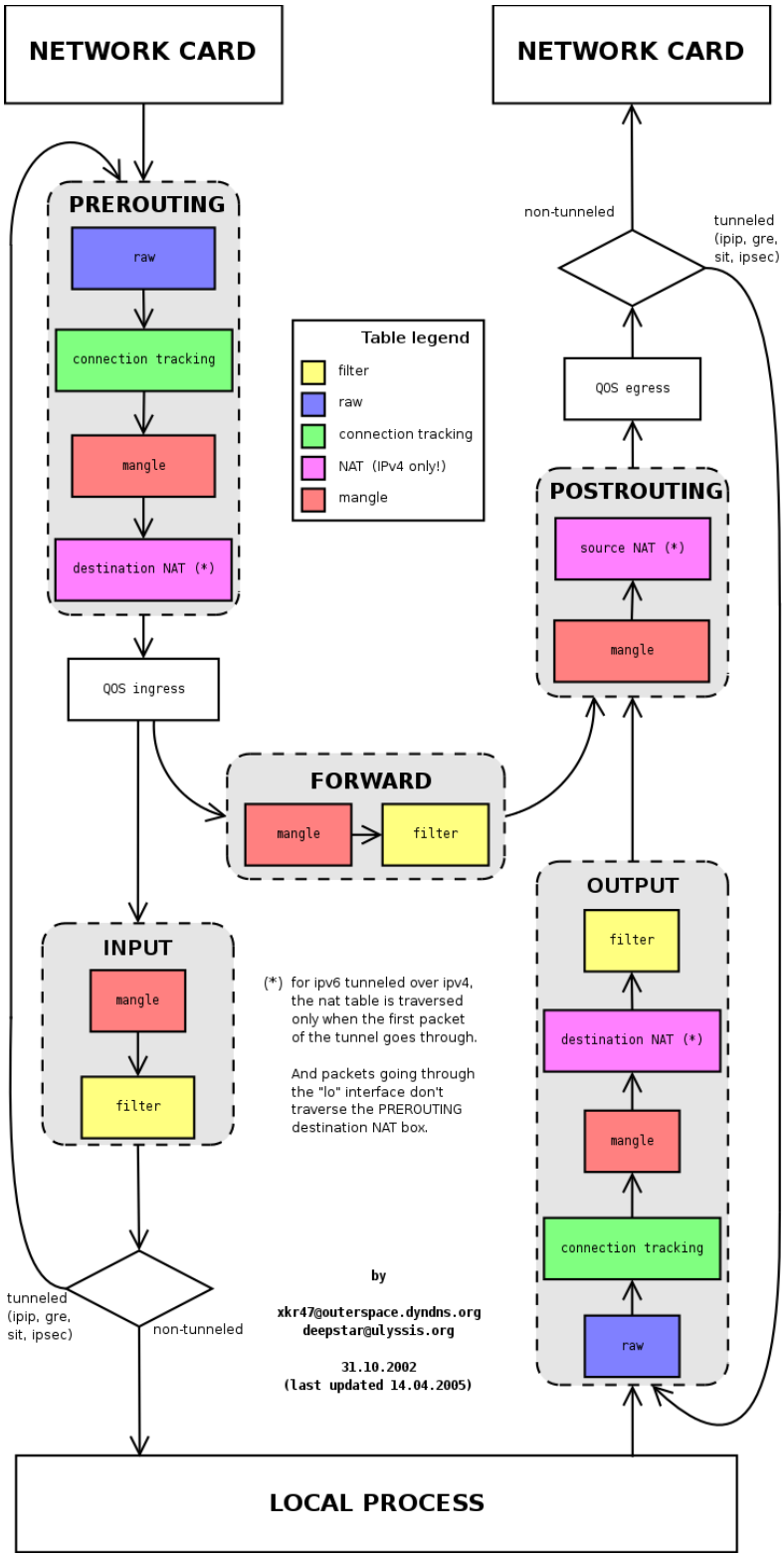


Figura 19: Diagrama de flujo del atravesar de los paquetes por las cadenas de iptables

Las reglas de iptables para el tráfico de control y mantenimiento:

Marcado de los paquetes de conexiones iniciadas:

```
iptables -t mangle -A POSTROUTING -j CONNMARK –restore-mark
```

DNS: 53/UDP, 53/TCP; NTP: 123/UDP, 123/TCP; SSH: 22/UDP, 22/TCP

```
iptables -t mangle -A POSTROUTING -o $DEV -p udp -m multiport
--dports 53,123,22 -m state --state NEW -j CONNMARK --set-mark 1

iptables -t mangle -A POSTROUTING -o $DEV -p udp -m multiport
--dports 53,123,22 -m state --state NEW -j RETURN

iptables -t mangle -A POSTROUTING -o $DEV -p tcp -m multiport
--dports 53,123,22 -m state --state NEW -j CONNMARK --set-mark 1

iptables -t mangle -A POSTROUTING -o $DEV -p tcp -m multiport
--dports 53,123,22 -m state --state NEW -j RETURN
```

Puerto de administración: *$ADMIN_PORT*/TCP

```
iptables -t mangle -A POSTROUTING -o $DEV -p tcp --sport
$ADMIN_PORT -m state --state NEW -j CONNMARK --set-mark 1

iptables -t mangle -A POSTROUTING -o $DEV -p tcp --sport
$ADMIN_PORT -m state --state NEW -j RETURN
```

ICMP:

```
iptables -t mangle -A POSTROUTING -p icmp -j MARK --set-mark
1
iptables -t mangle -A POSTROUTING -p icmp -j RETURN
```

Direcciones IP de dispositivos de red: 192.168.1.7, 172.16.8.73, 10.229.40.67, 10.229.40.68, 10.229.40.65, 10.229.40.161, 10.229.40.163, 172.16.10.77, 172.16.13.221, 172.16.8.115, 172.16.2.166, 172.16.2.170

```
iptables -t mangle -A POSTROUTING -s
192.168.1.7,172.16.8.73,10.229.40.67,10.229.40.68,10.229.40.65,10.
229.40.161,10.229.40.163,172.16.10.77,172.16.13.221,172.16.8.115,1
72.16.2.166,172.16.2.170 -m state --state NEW -j CONNMARK --set-
mark 1

iptables -t mangle -A POSTROUTING -s
192.168.1.7,172.16.8.73,10.229.40.67,10.229.40.68,10.229.40.65,10.
229.40.161,10.229.40.163,172.16.10.77,172.16.13.221,172.16.8.115,1
72.16.2.166,172.16.2.170 -m state --state NEW -j RETURN
```

Dentro del tráfico B se reparte el ancho de banda de forma equitativa entre las IP de los socios. La qdisc htb :20 gestiona este tráfico.

Los nodos que no sean socios podrán utilizar ancho de banda que los socios no necesitan ni usan. Por defecto el tráfico va a esta clase (default 99).

```
     tc class add dev $DEV parent 20: classid 20:99 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS  ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 99

     tc qdisc add dev $DEV parent 20:99 handle 999: pfifo_fast
```

Las direcciones IP se han de identificar manualmente, ya que serán las direcciones IP públicas de guifi.net de los nodos socios documentadas en el apartado Introducción - La comunidad Guifi-Elx - Direccionamiento IP. Se comprueba el direccionamiento de los paquetes que llegan al gateway con el analizador de paquetes tcpdump[52], ejecutando los comandos:

```
DEV_LAN=eth1
tcpdump -D
tcpdump -i $DEV_LAN -n -c 10 > out
```

La opción -D de tcpdump es para listar las interfaces sobre las que se puede analizar el tráfico. Con la opción -i se indica la interfaz elegida, la interfaz $DEV_LAN conectada a la red de guifi.net por la que pasa el tráfico antes de hacer NAT. Con la opción -n se evita la resolución DNS al mostrar los detalles de los paquetes, ya que no es necesario. Con la opción -c se indica la cantidad de paquetes a recoger.

Las direcciones IP de los ocho nodos socios cuyo tráfico pasa por el gateway de la asociación son:

10.229.40.102, BlasSelvaMendiola.

10.229.40.103, ClementeGonz25.

10.229.40.138, Atzavares1.

10.229.40.161, CasaAvalacant14.

10.229.40.227, AsprellaP1A.

10.229.40.229, AsprellaP1n9.

10.229.40.230, Asprella136.

10.229.40.234, AndrPerpinya1.

El nodo CasaGaitan todavía no tiene asignada IP pública de guifi.net.

Se crea una clase, una qdisc y un filtro por cada dirección IP. Se decide utilizar como identificador de la clase el último número de la dirección IP. Como el rango de direcciones públicas de guifi.net utilizado en la zona de Elx es 10.229.40.0/24, es el último número el único que varía, y por tanto es único para cada IP de cada socio. Si se utiliza otro rango de direcciones IP puede modificar el mecanismo de creación de identificadores. La clase, la qdisc y el filtro para un nodo socio son los siguientes. Para el resto de socios se ha de cambiar el número 102.

```
tc class add dev $DEV parent 20: classid 20:102 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 2

tc qdisc add dev $DEV parent 20:102 handle 102: pfifo_fast

tc filter add dev $DEV parent 1: protocol ip prio 1 handle
102 fw flowid 20:102
```

El tráfico clasificado en cada clase es gestionado por una qdisc pfifo_fast.

La figura 20 representa el árbol de objetos para controlar el tráfico de la interfaz creados hasta este punto.

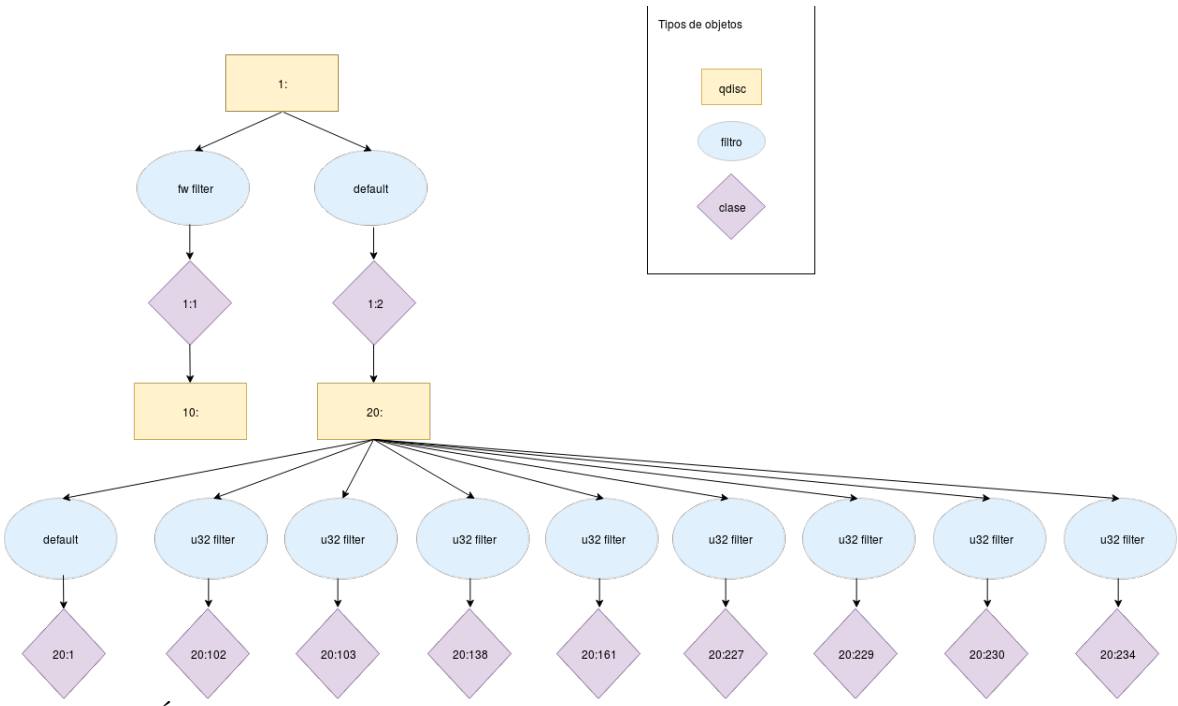


Figura 20: Árbol de objetos tc para gestionar el tráfico (II)

Finalmente, queda por configurar el control del tráfico masivo de cada nodo. Las políticas indican que se ha de clasificar, priorizar y moldear cada tipo de tráfico descrito. Esto se puede configurar en el gateway o en el dispositivo privado de cada socio. Se decide configurar en el dispositivo privado de cada socio por los siguientes motivos:

- Distribución del procesamiento. Así se evita un exceso de trabajo al gateway de la asociación.
- Mayor control por parte de los socios. La configuración personalizada para cada socio se configura en sus propios dispositivos.
- Eficiencia. En este escenario, el tratar el tráfico en el punto más cercano al origen permite mejorar el uso del ancho de banda durante el camino hasta el gateway.

La configuración para el tráfico masivo de los socios queda como propuesta de trabajo futuro. La configuración dependerá de las herramientas del dispositivo del socio. No obstante, en el Anexo III se describe el proceso de generación de esta configuración si se implementara en el gateway.

Guión de comandos de configuración de QoS

```
#!/bin/bash
#
# Inspidado de: https://www.iplocation.net/traffic-control

UPRATE=100000
N_SOCIOS=9
DEV=eth0
ADMIN_PORT=


start() {


  # En la raíz hay una disciplina de encolado htb. El tráfico que
no sea clase 1 (A) es clase 2 (B) por defecto.
  tc qdisc add dev $DEV root handle 1: htb default 2

  # El tráfico tipo B tiene tiene control del ancho de banda.
  tc class add dev $DEV parent 1: classid 1:2 htb rate $(( $UPRATE
* 9 / 10  ))kbit prio 2

  # El tráfico tipo A.
```

```
  tc class add dev $DEV parent 1: classid 1:1 htb rate $
{UPRATE}kbit prio 1

  # La qdisc sfq para hacer un reparto justo del ancho de banda
por flujos.
  tc qdisc add dev $DEV parent 1:1 handle 10: sfq perturb 10

  # Filtrado del tráfico A, en la banda (clase) más prioritaria.
El marcado de los paquetes es 1.
  tc filter add dev $DEV parent 1: protocol ip prio 1 handle 1 fw
flowid 1:1


  # Marcado de los paquetes de conexiones iniciadas.
  iptables -t mangle -A POSTROUTING -j CONNMARK --restore-mark

  # Marcado de los paquetes que inician sesiones de tráfico tipo
A.
  # A- Tráfico de control y mantenimiento.
  # DNS: 53/UDP, 53/TCP
  # NTP: 123/UDP, 123/TCP
  # SSH: 22/UDP, 22/TCP
  iptables -t mangle -A POSTROUTING -o $DEV -p udp -m multiport
--dports 53,123,22 -m state --state NEW -j CONNMARK --set-mark 1
  iptables -t mangle -A POSTROUTING -o $DEV -p udp -m multiport
--dports 53,123,22 -m state --state NEW -j RETURN
  iptables -t mangle -A POSTROUTING -o $DEV -p tcp -m multiport
--dports 53,123,22 -m state --state NEW -j CONNMARK --set-mark 1
  iptables -t mangle -A POSTROUTING -o $DEV -p tcp -m multiport
--dports 53,123,22 -m state --state NEW -j RETURN

  # Puerto de administración: $ADMIN_PORT/TCP
  iptables -t mangle -A POSTROUTING -o $DEV -p tcp --sport
$ADMIN_PORT -m state --state NEW -j CONNMARK --set-mark 1
  iptables -t mangle -A POSTROUTING -o $DEV -p tcp --sport
$ADMIN_PORT -m state --state NEW -j RETURN

  # ICMP
  iptables -t mangle -A POSTROUTING -p icmp -j MARK --set-mark 1
  iptables -t mangle -A POSTROUTING -p icmp -j RETURN

  # Direcciones IP de los dispositivos de red
  iptables -t mangle -A POSTROUTING -s
192.168.1.7,172.16.8.73,10.229.40.67,10.229.40.68,10.229.40.65,10.
229.40.161,10.229.40.163,172.16.10.77,172.16.13.221,172.16.8.115,1
72.16.2.166,172.16.2.170 -m state --state NEW -j CONNMARK --set-
mark 1
  iptables -t mangle -A POSTROUTING -s
192.168.1.7,172.16.8.73,10.229.40.67,10.229.40.68,10.229.40.65,10.
```

```
229.40.161,10.229.40.163,172.16.10.77,172.16.13.221,172.16.8.115,1
72.16.2.166,172.16.2.170 -m state --state NEW -j RETURN



  # Qdisc para repartir ancho de banda entre socios
  tc qdisc add dev $DEV parent 1:2 handle 20: htb default 99

  # Los nodos que no sean socios podrán utilizar ancho de banda
que los socios no necesitan ni usan. No es posible poner rate=0.
  tc class add dev $DEV parent 20: classid 20:99 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS  ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 99
  tc qdisc add dev $DEV parent 20:99 handle 999: pfifo_fast

  # El rate del tráfico B entre 9 socios.
  # El identificador de clase corresponde con el último número
decimal de la IP plica de guifi.net del nodo socio.

  iptables -t mangle -A POSTROUTING -s 10.229.40.102 -m state
--state NEW -j CONNMARK --set-mark 102
  iptables -t mangle -A POSTROUTING -s 10.229.40.102 -m state
--state NEW -j RETURN
  tc class add dev $DEV parent 20: classid 20:102 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 2
  tc qdisc add dev $DEV parent 20:102 handle 102: pfifo_fast
  tc filter add dev $DEV parent 1: protocol ip prio 1 handle 102
fw flowid 20:102

  iptables -t mangle -A POSTROUTING -s 10.229.40.133 -m state
--state NEW -j CONNMARK --set-mark 133
  iptables -t mangle -A POSTROUTING -s 10.229.40.133 -m state
--state NEW -j RETURN
  tc class add dev $DEV parent 20: classid 20:133 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS  ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 2
  tc qdisc add dev $DEV parent 20:133 handle 133: pfifo_fast
  tc filter add dev $DEV parent 1: protocol ip prio 1 handle 133
fw flowid 20:133


  iptables -t mangle -A POSTROUTING -s 10.229.40.138 -m state
--state NEW -j CONNMARK --set-mark 138
  iptables -t mangle -A POSTROUTING -s 10.229.40.138 -m state
--state NEW -j RETURN
  tc class add dev $DEV parent 20: classid 20:138 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS  ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 2
  tc qdisc add dev $DEV parent 20:138 handle 138: pfifo_fast
```

```
  tc filter add dev $DEV parent 1: protocol ip prio 1 handle 138
fw flowid 20:138

  iptables -t mangle -A POSTROUTING -s 10.229.40.161 -m state
--state NEW -j CONNMARK --set-mark 161
  iptables -t mangle -A POSTROUTING -s 10.229.40.161 -m state
--state NEW -j RETURN
  tc class add dev $DEV parent 20: classid 20:161 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS  ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 2
  tc qdisc add dev $DEV parent 20:161 handle 161: pfifo_fast
  tc filter add dev $DEV parent 1: protocol ip prio 1 handle 161
fw flowid 20:161

  iptables -t mangle -A POSTROUTING -s 10.229.40.227 -m state
--state NEW -j CONNMARK --set-mark 227
  iptables -t mangle -A POSTROUTING -s 10.229.40.227 -m state
--state NEW -j RETURN
  tc class add dev $DEV parent 20: classid 20:227 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS  ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 2
  tc qdisc add dev $DEV parent 20:227 handle 227: pfifo_fast
  tc filter add dev $DEV parent 1: protocol ip prio 1 handle 227
fw flowid 20:227

  iptables -t mangle -A POSTROUTING -s 10.229.40.229 -m state
--state NEW -j CONNMARK --set-mark 229
  iptables -t mangle -A POSTROUTING -s 10.229.40.229 -m state
--state NEW -j RETURN
  tc class add dev $DEV parent 20: classid 20:229 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS  ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 2
  tc qdisc add dev $DEV parent 20:229 handle 229: pfifo_fast
  tc filter add dev $DEV parent 1: protocol ip prio 1 handle 229
fw flowid 20:229

  iptables -t mangle -A POSTROUTING -s 10.229.40.230 -m state
--state NEW -j CONNMARK --set-mark 230
  iptables -t mangle -A POSTROUTING -s 10.229.40.230 -m state
--state NEW -j RETURN
  tc class add dev $DEV parent 20: classid 20:230 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS  ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 2
  tc qdisc add dev $DEV parent 20:230 handle 230: pfifo_fast
  tc filter add dev $DEV parent 1: protocol ip prio 1 handle 230
fw flowid 20:230

  iptables -t mangle -A POSTROUTING -s 10.229.40.234 -m state
--state NEW -j CONNMARK --set-mark 234
```

```
  iptables -t mangle -A POSTROUTING -s 10.229.40.234 -m state
--state NEW -j RETURN
  tc class add dev $DEV parent 20: classid 20:234 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS  ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 2
  tc qdisc add dev $DEV parent 20:234 handle 234: pfifo_fast
  tc filter add dev $DEV parent 1: protocol ip prio 1 handle 234
fw flowid 20:234


}

stop() {

# Borrado de objetos.
tc qdisc del dev $DEV root
iptables -t mangle -F

}

restart() {

# Self-explanatory.
    stop
    sleep 1
    start

}

show() {

  tc -s qdisc show dev $DEV
  iptables -t mangle -L POSTROUTING -v -n
  # -v Muestra un contador de cada flujo.
  # -n Muestra las direcciones IP y puertos en formato numérico.
}

case "$1" in

  start)

    echo -n "Starting bandwidth shaping: "
    start
    echo "done"
    ;;

  stop)

    echo -n "Stopping bandwidth shaping: "
    stop
```

```
    echo "done"
    ;;

  restart)

    echo -n "Restarting bandwidth shaping: "
    restart
    echo "done"
    ;;

  show)

    echo "Bandwidth shaping status for $IF:"
    show
    echo ""
    ;;

  *)

    pwd=$(pwd)
    echo "Usage: qos.script {start|stop|restart|show}"
    ;;

esac

exit 0
```

Pruebas de las políticas de QoS

La siguiente fase del despliegue de QoS es la ejecución de pruebas. Se proponen dos pruebas para evaluar la mejora.

Prueba 1. El tráfico de control y mantenimiento de la red tiene prioridad sobre otros tipos de tráfico. Siempre tiene ancho de banda disponible.

Prueba 2. El ancho de banda se reparte entre los nodos socios equitativamente.

Evaluar la QoS supone medir la calidad de la transmisión y la disponibilidad del servicio en cuestión, en términos de ancho de banda, pérdida de paquetes, retardo y variación del retardo.

Queda como trabajo futuro el diseño detallado y la ejecución de las pruebas, incluyendo la elección del escenario de pruebas y la adaptación tanto del escenario elegido como de la configuración generada.

Implementación de la políticas

La implementación de las políticas se hace de manera temporal en el servidor paralelo. Esto implica activar la configuración en este dispositivo. En el RouterGaitan se activa la ruta por defecto al ServidorParalelo en vez de al RouterGaitan.

El guión de comandos ejecutado en esta fase es una modificación del anterior, simplificando la jerarquía de objetos tc. En el Anexo IV se incluye este guión de comandos.

Monitorización

Se instala un servidor de monitorización escalable, distribuido, en tiempo real y saludable en el servidor paralelo para visualizar el uso de las diferentes clases tc creadas en la configuración. Netdata[53]. La figura 21 muestra el uso de las clases tc con el tráfico normal de la red. La primera gráfica ilustra el ancho de banda de cada clase, y la segunda gráfica, la cantidad de paquetes asociados a cada clase.

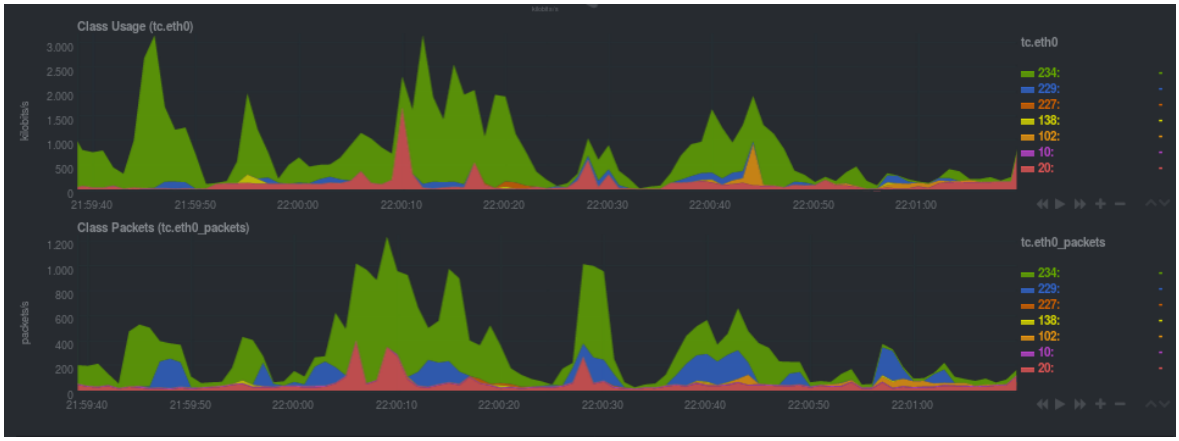


Figura 21: Uso de las clases de tc configuradas

Se accede a la información del dispositivo a través de la URL http://localhost:19999.

Además se visualiza el número de paquetes que llegan a las qdisc ejecutando el comando:

```
tc -s qdisc show dev $DEV
```

La figura 22 muestra la salida del anterior comando.

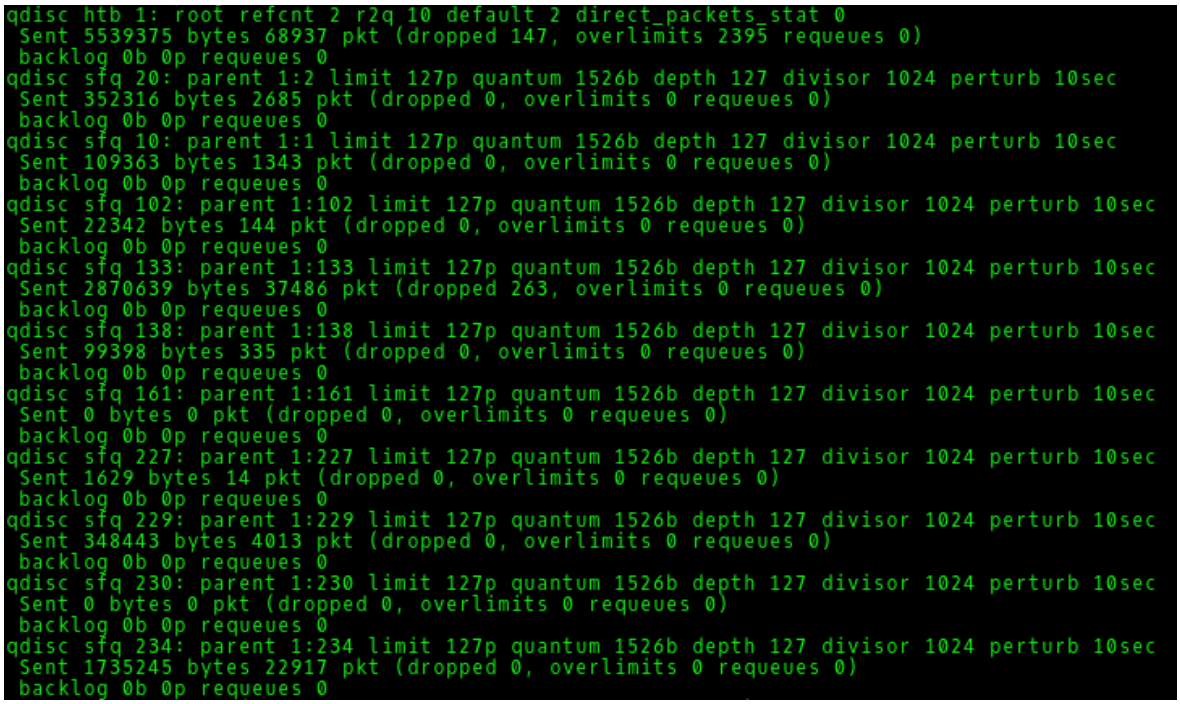


Figura 22: Salida del comando de monitorización de las qdisc

La cantidad de paquetes que han cumplido los criterios de las reglas de iptables se muestran ejecutando el comando:

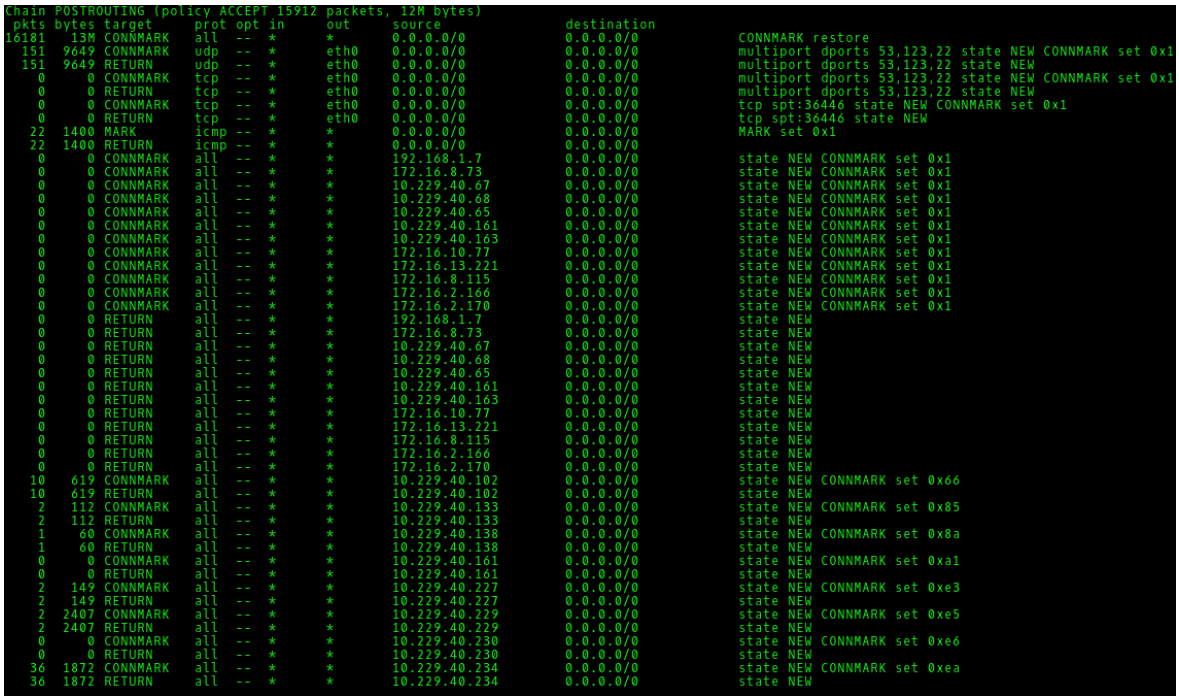
```
iptables -t mangle -L POSTROUTING -v -n
```

La figura 23 muestra la salida del comando anterior.

Por línea de comandos se puede utilizar la aplicación iftop[54] para visualizar el uso del ancho de banda en una interfaz. Para poder visualizar las direcciones IP antes de que se haga NAT se ha de monitorizar la interfaz hacia guifi.net, ejecutando el comando:

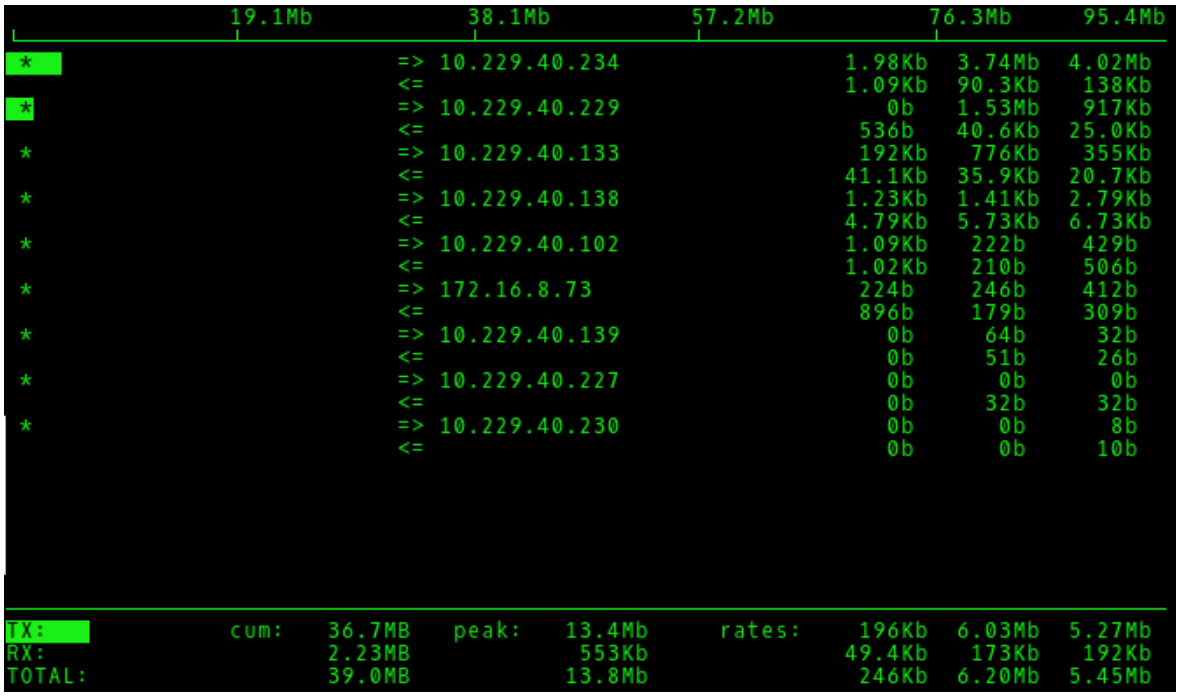
```
iftop -nN -i eth1
```

Si a continuación se pulsa la tecla ‘s’, se agruparán los flujos de datos por dirección IP de destino , que son las direcciones IP de los dispositivos de la red. La figura 24 muestra la salida del comando anterior.



```
Chain POSTROUTING (policy ACCEPT 15912 packets, 12M bytes)
 pkts bytes target     prot opt in     out     source               destination
16181   13M CONNMARK   all  --  *      *       0.0.0.0/0            0.0.0.0/0           CONNMARK restore
  151  9649 CONNMARK   udp  --  *      eth0    0.0.0.0/0            0.0.0.0/0           multiport dports 53,123,22 state NEW CONNMARK set 0x1
  151  9649 RETURN     udp  --  *      eth0    0.0.0.0/0            0.0.0.0/0           multiport dports 53,123,22 state NEW
    0     0 CONNMARK   tcp  --  *      eth0    0.0.0.0/0            0.0.0.0/0           multiport dports 53,123,22 state NEW CONNMARK set 0x1
    0     0 RETURN     tcp  --  *      eth0    0.0.0.0/0            0.0.0.0/0           multiport dports 53,123,22 state NEW
    0     0 CONNMARK   tcp  --  *      eth0    0.0.0.0/0            0.0.0.0/0           tcp spt:36446 state NEW CONNMARK set 0x1
    0     0 RETURN     tcp  --  *      eth0    0.0.0.0/0            0.0.0.0/0           tcp spt:36446 state NEW
   22  1400 MARK       icmp --  *      *       0.0.0.0/0            0.0.0.0/0           MARK set 0x1
   22  1400 RETURN     icmp --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 CONNMARK   all  --  *      *       192.168.1.7          0.0.0.0/0           state NEW CONNMARK set 0x1
    0     0 CONNMARK   all  --  *      *       172.16.8.73          0.0.0.0/0           state NEW CONNMARK set 0x1
    0     0 CONNMARK   all  --  *      *       10.229.40.67         0.0.0.0/0           state NEW CONNMARK set 0x1
    0     0 CONNMARK   all  --  *      *       10.229.40.68         0.0.0.0/0           state NEW CONNMARK set 0x1
    0     0 CONNMARK   all  --  *      *       10.229.40.65         0.0.0.0/0           state NEW CONNMARK set 0x1
    0     0 CONNMARK   all  --  *      *       10.229.40.161        0.0.0.0/0           state NEW CONNMARK set 0x1
    0     0 CONNMARK   all  --  *      *       10.229.40.163        0.0.0.0/0           state NEW CONNMARK set 0x1
    0     0 CONNMARK   all  --  *      *       172.16.10.77         0.0.0.0/0           state NEW CONNMARK set 0x1
    0     0 CONNMARK   all  --  *      *       172.16.13.221        0.0.0.0/0           state NEW CONNMARK set 0x1
    0     0 CONNMARK   all  --  *      *       172.16.8.115         0.0.0.0/0           state NEW CONNMARK set 0x1
    0     0 CONNMARK   all  --  *      *       172.16.2.166         0.0.0.0/0           state NEW CONNMARK set 0x1
    0     0 CONNMARK   all  --  *      *       172.16.2.170         0.0.0.0/0           state NEW CONNMARK set 0x1
    0     0 RETURN     all  --  *      *       192.168.1.7          0.0.0.0/0           state NEW
    0     0 RETURN     all  --  *      *       172.16.8.73          0.0.0.0/0           state NEW
    0     0 RETURN     all  --  *      *       10.229.40.67         0.0.0.0/0           state NEW
    0     0 RETURN     all  --  *      *       10.229.40.68         0.0.0.0/0           state NEW
    0     0 RETURN     all  --  *      *       10.229.40.65         0.0.0.0/0           state NEW
    0     0 RETURN     all  --  *      *       10.229.40.161        0.0.0.0/0           state NEW
    0     0 RETURN     all  --  *      *       10.229.40.163        0.0.0.0/0           state NEW
    0     0 RETURN     all  --  *      *       172.16.10.77         0.0.0.0/0           state NEW
    0     0 RETURN     all  --  *      *       172.16.13.221        0.0.0.0/0           state NEW
    0     0 RETURN     all  --  *      *       172.16.8.115         0.0.0.0/0           state NEW
    0     0 RETURN     all  --  *      *       172.16.2.166         0.0.0.0/0           state NEW
    0     0 RETURN     all  --  *      *       172.16.2.170         0.0.0.0/0           state NEW
   10   619 CONNMARK   all  --  *      *       10.229.40.102        0.0.0.0/0           state NEW CONNMARK set 0x66
   10   619 RETURN     all  --  *      *       10.229.40.102        0.0.0.0/0           state NEW
    2   112 CONNMARK   all  --  *      *       10.229.40.133        0.0.0.0/0           state NEW CONNMARK set 0x85
    2   112 RETURN     all  --  *      *       10.229.40.133        0.0.0.0/0           state NEW
    1    60 CONNMARK   all  --  *      *       10.229.40.138        0.0.0.0/0           state NEW CONNMARK set 0x8a
    1    60 RETURN     all  --  *      *       10.229.40.138        0.0.0.0/0           state NEW
    0     0 CONNMARK   all  --  *      *       10.229.40.161        0.0.0.0/0           state NEW CONNMARK set 0xa1
    0     0 RETURN     all  --  *      *       10.229.40.161        0.0.0.0/0           state NEW
    2   149 CONNMARK   all  --  *      *       10.229.40.227        0.0.0.0/0           state NEW CONNMARK set 0xe3
    2   149 RETURN     all  --  *      *       10.229.40.227        0.0.0.0/0           state NEW
    2  2407 CONNMARK   all  --  *      *       10.229.40.229        0.0.0.0/0           state NEW CONNMARK set 0xe5
    2  2407 RETURN     all  --  *      *       10.229.40.229        0.0.0.0/0           state NEW
    0     0 CONNMARK   all  --  *      *       10.229.40.230        0.0.0.0/0           state NEW CONNMARK set 0xe6
    0     0 RETURN     all  --  *      *       10.229.40.230        0.0.0.0/0           state NEW
   36  1872 CONNMARK   all  --  *      *       10.229.40.234        0.0.0.0/0           state NEW CONNMARK set 0xea
   36  1872 RETURN     all  --  *      *       10.229.40.234        0.0.0.0/0           state NEW
```

Figura 23: Salida del comando de monitorización de iptables



```
                 19.1Mb           38.1Mb           57.2Mb           76.3Mb           95.4Mb
└────────────────┴────────────────┴────────────────┴────────────────┴────────────────┘
*                                => 10.229.40.234                  1.98Kb   3.74Mb   4.02Mb
                                 <=                                1.09Kb   90.3Kb    138Kb
*                                => 10.229.40.229                     0b   1.53Mb    917Kb
                                 <=                                 536b   40.6Kb   25.0Kb
*                                => 10.229.40.133                   192Kb    776Kb    355Kb
                                 <=                                41.1Kb   35.9Kb   20.7Kb
*                                => 10.229.40.138                  1.23Kb   1.41Kb   2.79Kb
                                 <=                                4.79Kb   5.73Kb   6.73Kb
*                                => 10.229.40.102                  1.09Kb    222b     429b
                                 <=                                1.02Kb    210b     506b
*                                => 172.16.8.73                     224b     246b     412b
                                 <=                                 896b     179b     309b
*                                => 10.229.40.139                     0b      64b      32b
                                 <=                                   0b      51b      26b
*                                => 10.229.40.227                     0b       0b       0b
                                 <=                                   0b      32b      32b
*                                => 10.229.40.230                     0b       0b       8b
                                 <=                                   0b       0b      10b

TX:              cum:   36.7MB   peak:   13.4Mb      rates:     196Kb   6.03Mb   5.27Mb
RX:                     2.23MB            553Kb               49.4Kb    173Kb    192Kb
TOTAL:                  39.0MB           13.8Mb               246Kb   6.20Mb   5.45Mb
```

Figura 24: Ejecución de iftop por consola

5. CONCLUSIONES

La realización de este proyecto se enmarca dentro del trabajo valiente, comprometido y con ilusión que llevan a cabo los voluntarios de la comunidad de Guifi-Elx. Las actividades de mejora reflejadas en esta memoria se suman a la evolución continua de la red comunitaria.

Excepto el servicio de QoS, las otras tres actividades de mejora forman parte del estado actual de la red operativa, aportando valor y capacidad de crecimiento.

El aumento de ancho de banda a Internet contratado permite que éste sea un recurso menos limitado y se disminuya la competencia. Aún así, la infraestructura de red es Fast Ethernet. La máxima velocidad desde la red de Guifi-Elx es 100Mbps, y desde la CasaGaitan 300Mbps. Antes de este cambio, el máximo para cualquier nodo era 50Mbps. Para que los nodos de la red tengan disponibilidad máxima del ancho de banda a Internet hace falta una evolución de la infraestructura de red a Gigabit Ethernet. Esta mejora se está llevando a cabo.

La instalación del nuevo servidor paralelo proporciona resistencia a la red pudiendo hacer de gateway automáticamente cuando fuera necesario. Además es un servidor para desplegar los servicios que la comunidad pueda necesitar. Actualmente es el servidor web.

Tener una página web informativa es una herramienta muy útil de difusión de cualquier proyecto. La asamblea de Guifi-Elx considera necesario un blog informativo accesible desde Internet para cualquier persona interesada. Así se amplían los canales de comunicación. Siguiendo la idea de autoabastecimiento, se ha instalado en el nuevo dispositivo de red un servidor Wordpress, donde se puede generar contenido fácilmente. La página web se sirve por HTTPS, proporcionando cifrado y autenticidad del contenido web. La página web está en desarrollo.

Por último, el despliegue del servicio de QoS en la red se ha desarrollado en todas sus fases. En la fase de análisis de los requisitos de red (fase 1) y en la fase de definición de las políticas (fase 2) se invitó a participar a las personas que usan la red. En la fase de análisis era importante la opinión y la experiencia de los usuarios, y en la fase de definición de las políticas era importante el consenso de todos los usuarios. Sin embargo, la implicación fue mínima. En la fase de diseño de las políticas (fase 3) ha sido muy interesante a la vez que costoso el estudio de las herramientas tc e iptables para su aprovechamiento óptimo en la

implementación de las políticas. Aunque se ha diseñado una solución tratando el tráfico solamente en sentido de salida por la interfaz hacia Internet, se pueden estudiar los beneficios de moldear el tráfico en sentido de entrada también, como se propone en la Guía de Configuración de FireQOS[55]. Como trabajo futuro queda la realización de las pruebas detalladas sobre las métricas de QoS y la implementación definitiva del servicio de QoS en la red. También queda abierto el camino para producir la configuración de QoS con una herramienta de nivel de abstracción de configuración superior. Se destaca FireQoS por estar en desarrollo continuo actualmente, por haber sido útiles las discusiones que han tenido lugar en el repositorio del código de la aplicación, y porque las funcionalidades implementadas coinciden con los requisitos de este proyecto.

Además esta memoria incluye las guías de instalación del nuevo dispositivo de red y del servidor web. La documentación es esencial para contribuir al conocimiento colectivo y a la libertad de las personas para acceder a la información. Así como para el desarrollo de este proyecto se ha acudido a la documentación disponible de experiencias y trabajos previos, este documento queda disponible para ser consultado.

Se concluye pues, que la red libre ciudadana de Guifi-Elx ha sido mejorada, repercutiendo positivamente a nivel local en su comunidad y en la asociación Xarxa Lliure Elx, y a nivel global en guifi.net, que sigue creciendo por el bien común.

ANEXO I

Esta guía abarca la configuración básica de red y la instalación y configuración de servicios básicos. Está guía se basa en un ordenador de placa reducida Odroid-U2 con el sistema operativo Debian 8 (Jessie). La configuración se realiza por línea de comandos a través de una conexión ssh al sistema. Esta guía no abarca la instalación del sistema operativo ni la conexión por ssh. Se asume administración básica del sistema operativo por línea de comandos.

El primer paso es instalar todas las **actualizaciones del sistema** ejecutando los comandos:

```
apt-get update
apt-get upgrade
apt-get dist-upgrade
```

Este dispositivo necesita dos puertos físicos Ethernet para conectarse a la red por cable. Si la placa dispone de un puerto solo (como es el caso de la Odroid-U2), se necesita un **adaptador de USB 2.0 a Fast Ethernet** para utilizar uno de los puertos USB como un puerto Fast Ethernet. Después de conectar el adaptador a la placa se comprueba que automáticamente el sistema añade una nueva interfaz ejecutando el comando:

```
ifconfig
```

En la salida se ven las dos interfaces de red *eth0* y *eth1*.

Para configurar las **interfaces de red**, se ha modificar el fichero */etc/network/interfaces*:

```
    # La interfaz loopback
    auto lo
    iface lo inet loopback

    # Activa automáticamente. Interfaz LAN, que da a guifi.net.
Opción estática. IP de gestión.
    auto eth1
    allow-hotplug eth1
    iface eth1 inet static
            address 172.16.8.74
            netmask 255.255.255.252

    # Activa automáticamente. Interfaz LAN, que da a guifi.net.
Opción estática. IP pública de guifi.net.
    auto eth1:0
    allow-hotplug eth1:0
```

```
    iface eth1:0 inet static
            address 10.229.40.70
            netmask 255.255.255.248

    # Activa la eth0 automáticamente. Interfaz WAN, que da a
Internet. Opción dhcp u opción static.
    auto eth0
    allow-hotplug eth0
    # Opción ip por dhcp para la configuración inicial.
    #iface eth0 inet dhcp
    # Opción ip estática.
    iface eth0 inet static
           address 192.168.1.6
           netmask 255.255.255.0
           gateway 192.168.1.1
```

Los **servidores dns** también se indican en */etc/network/interfaces*, en la interfaz eth0. Se han elegido servidores públicos de distintas empresas: el DNS local del gateway de Internet, Google y Level 3.

```
    dns-nameservers 192.168.1.1 8.8.8.8 4.2.2.1
    dns-search guifi.net
```

Después de instalar resolvconf, para que los servidores indicados pasen al fichero de */etc/resolv.conf*, y después de reiniciar las interfaces, se comprueba que en el fichero */etc/resolv.conf* no hayan servidores DNS incorrectos.

El **cortafuegos** se configura con la aplicación iptables. Se crea un script *fw.proxy* con los comandos necesarios:

```
    !/bin/sh

    # Interfaz conectada a Internet.
    INTERNET="eth0"

    # Interfaz conectada a LAN.
    LAN_IN="eth1"

    # NO MODIFICAR ABAJO
    # Borrar antiguo cortafuegos.
    iptables -F
    iptables -X
    iptables -t nat -F
    iptables -t nat -X
    iptables -t mangle -F
    iptables -t mangle -X
```

```
    # Cargar los módulos IPTABLES para NAT y soporte IP conntrack.
    modprobe ip_conntrack
    modprobe ip_conntrack_ftp

    # Para cliente ftp de win xp.
    #modprobe ip_nat_ftp

    # Permitir ICMP PING
    iptables -A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT

    # Asegurar que se puede acceder vía ssh.
    iptables -A INPUT -p tcp --dport $ADMIN_PORT -j ACCEPT

    # Permitir el acceso a la web.
    iptables -A INPUT -p tcp --dport 80 -j ACCEPT
    iptables -A INPUT -p tcp --dport 443 -j ACCEPT

    # Consultas DNS (udp y tcp).
    iptables -A INPUT -i $LAN_IN -p udp --dport 53 -j ACCEPT
    iptables -A INPUT -i $LAN_IN -p tcp --dport 53 -j ACCEPT
    iptables -A INPUT -i $LAN_IN -p udp --dport 953 -j ACCEPT
    iptables -A INPUT -i $LAN_IN -p tcp --dport 953 -j ACCEPT

    # Política por defecto.
    iptables -P INPUT DROP
    iptables -P OUTPUT ACCEPT

    # Acceso ilimitado a loopback.
    iptables -A INPUT -i lo -j ACCEPT
    iptables -A OUTPUT -o lo -j ACCEPT

    # Permitir UDP, DNS y FTP Pasivo.
    iptables -A INPUT -i $INTERNET -m state --state
ESTABLISHED,RELATED -j ACCEPT

    # Poner este sistema como encaminador para el resto de la LAN.
    iptables -t nat -A POSTROUTING -o $INTERNET -j MASQUERADE
    iptables -A FORWARD -i $LAN_IN -j ACCEPT

    # Acceso ilimitado a LAN.
    iptables -A INPUT -i $LAN_IN -j ACCEPT
    iptables -A OUTPUT -o $LAN_IN -j ACCEPT

    # Descartar todo y guardar registro de ello.
    iptables -A INPUT -j LOG --log-level 4 --log-prefix ERROR:
```

Se le da permisos de ejecución al script y se ejecuta, ejecutando los comandos:

```
chmod 744 fw.proxy
```

```
./fw.proxy
```

Para que las reglas de iptables se apliquen en cada reinicio del sistema, se exporta la configuración en un fichero *iptables.up.rules* y se carga cuando se levantan las interfaces. El comando para exportar reglas iptables es:

```
iptables-save > iptables.up.rules
```

Para restaurar las reglas iptables se añade una línea en */etc/network/interfaces*:

```
post-up /sbin/iptables-restore < /<ruta>/iptables.up.rules
```

Nota: sustituir <*ruta*> por la ruta absoluta hasta el fichero *iptables.up.rules*.

Se configuran las **rutas del tráfico** en la tabla de rutas añadiendo las instrucciones en */etc/network/interfaces*:

```
    # Configurar las rutas. Las que vayan de 10.0.0.0 a
10.255.255.255 se envían hacia el router de guifi.net con ip
172.16.8.73.
    post-up route add -net 10.0.0.0 netmask 255.0.0.0 gw
172.16.8.73

   # Las que van destinadas de la 172.16.0.0 a la 172.31.255.255
también se envían hacia guifi.net.
    post-up route add -net 172.16.0.0 netmask 255.240.0.0 gw
172.16.8.73
```

Se instala **nmap** para explorar la red.

Se instala **tcpdump** para analizar el tráfico de la red por línea de comandos.

Se instalan dos aplicaciones para hacer **pruebas de velocidad**: speedtest-cli[56] e iperf. Speedtest-cli es una aplicación de línea de comandos para medir la velocidad de transmisión de datos hasta un servidor de speedtest.net cercano en Internet. Iperf es una aplicación que se ejecuta como cliente en un dispositivo y como servidor en otro, de forma que se evalúa un tramo de la red.[57]

Se indica el **nombre del disposivo** en */etc/hostname* y en */etc/hosts*.

Para que el dispositivo funcione como encaminador, pudiendo **reenviar paquetes**, se ha de indicar en */etc/sysctl.conf*, descomentando esta línea:

```
net.ipv4.ip_forward = 1
```

Para aplicar el cambio se reinicia el servicio:

```
/etc/init.d/procps restart
```

Y se comprueba que la funcionalidad de reenvío está activa (=1) ejecutando el comando:

```
cat /proc/sys/net/ipv4/ip_forward
```

Se gestionan los **usuarios del sistema**. Se cambia la contraseña del usuario root. Y se crea un nuevo usuario con permisos de root, para ejecutar comandos como root sólo cuando sea necesario. Se ejecutan los comandos:

```
passwd root
adduser <nuevo>
adduser <nuevo> sudo
```

Nota: sustituir <*nuevo*> por el nombre elegido para el usuario.

Se cambia el puerto ssh por defecto en /etc/ssh/sshd_config:

```
# What ports, IP address and protocols we listen for
Port <puerto>
```

Nota: sustituir <*puerto*> por el puerto elegido para administración.

Se establece la **zona horaria** ejecutando el comando:

```
dpkg-reconfigure tzdata
```

Se instala **squid3** para poder acceder al gateway de Internet desde la red de Guifi-Elx. Se añade al fichero */etc/squid3/squid.conf*:

```
    acl localnet1 src 10.0.0.0/8    # RFC1918 possible internal
network
    acl localnet2 src 172.16.0.0/12   # RFC1918 possible internal
network
    http_access allow localnet1
    http_access allow localnet2
```

Se instala bind9 para poder tener **DNS oficial de guifi.net**, configurado temporalmente como forwarder, editando */etc/bind/named.conf.options* :

```
    acl trusted {
        172.16.0.0/12;
        10.0.0.0/8;
        localhost;
```

```
    };

    options {
        directory "/var/cache/bind";

        recursion yes;
        allow-query { trusted; };

        forwarders {
                62.36.225.150;
                62.37.228.20;
                8.8.8.8;
                4.2.2.1;
                8.8.4.4;
                4.2.2.2;
        };
        forward only;

    //========================================================
        // If BIND logs error messages about the root key being
expired,
        // you will need to update your keys. See
https://www.isc.org/bind-keys
    //========================================================
        dnssec-enable yes;
        dnssec-validation yes;

        auth-nxdomain no;# conform to RFC1035
        listen-on-v6 { any; };
    };
```

Finalmente, se ha de añadir configuración en dos dispositivos más. Se configura en el gateway de Internet un **reenvío de puerto** para el puerto elegido de administración del dispositivo.

En el routerGaitan se configura la interfaz *wlan4* con la IP de gestión 172.16.8.73. Y se añade una **ruta por defecto de respaldo** en la tabla de rutas, hacia 172.16.8.74, que es la dirección IP de gestión del nuevo dispositivo, con distancia 2 (mayor que al gateway principal).

ANEXO II

Esta es la guía de instalación del servidor web HTTPS para servir la página web de Guifi-Elx, disponible desde Internet por la URL https://elx.guifi.net. Primero se instala la infraestructura LAMP para el servidor web, con Apache[58] y MySQL[59]; después se instala y configura Wordpress[60] como sistema de gestión de contenidos para publicar la web; después se habilita HTTPS en el servidor web con Certbot[61], como cliente de Let's Encrypt; y finalmente se instala el servicio de SFTP para instalar plugins y poder crear la web. En toda la guía se explica la instalación por línea de comandos.

INSTALACIÓN DE LA INFRAESTRUCTURA LAMP

Se ejecuta el comando:

```
apt-get install apache2 mysql-client mysql-server php5 php5-mysql php5-curl php5-gd
```

INSTALACIÓN DE WORDPRESS

Siguiendo las instrucciones de instalación de Wordpress[62] y el tutorial de www.linuxserve.com[63]:

Paso 1: Descargar y extraer

Se ejecutan los comandos:

```
cd <ruta_hasta_directorio_apache>
wget https://wordpress.org/latest.tar.gz
tar -xzvf latest.tar.gz
```

Nota: la **<*ruta_hasta_directorio_apache*>** configurada para el servidor apache por defecto es */var/www/html/*.

Paso 2: Crear una base de datos y un usuario

Se utiliza el cliente MySQL de línea de comandos ejecutando:

```
mysql -u <usuario_admin> -p

CREATE DATABASE <nom_bd_wp>;
```

```
     CREATE USER <usuari_wp>@localhost IDENTIFIED BY
'<contraseña_wp>';
     GRANT ALL PRIVILEGES ON <nom_bd_wp>.* TO
<usuari_wp>@localhost;
     FLUSH PRIVILEGES;
     EXIT
```

Nota: los nombres en cursiva se sustituyen por los nombres elegidos para la instalación. Se pueden seguir los consejos de la guía de Wordpress[64].

Paso 3: Configurar el fichero *wp-config.php*

Se ejecutan los comandos:

```
     cd <ruta_hasta_directorio_apache>/wordpress
     cp wp-config-sample.php wp-config.php
```

Se edita el el contenido de *wp-config.php* en la sección "MySQL settings", añadiendo los nombres elegidos, que están en negrita.

```
    // ** MySQL settings - You can get this info from your web
host ** //
    /** The name of the database for WordPress */
    define( 'DB_NAME', 'nom_bd_wp' );

    /** MySQL database username */
    define( 'DB_USER', 'usuari_wp' );

    /** MySQL database password */
    define( 'DB_PASSWORD', 'contraseña_wp' );

    /** MySQL hostname */
    define( 'DB_HOST', 'localhost' );
```

Utilizando el servicio de Wordpress de generación aleatoria de claves secretas[65], se modifica el fichero de configuración *wp-config.php*, en la sección "Authentication Unique Keys and Salts". Un ejemplo sería:

```
    define( 'AUTH_KEY',         'clave secreta aquí' );
    define( 'SECURE_AUTH_KEY',  'clave secreta aquí' );
    define( 'LOGGED_IN_KEY',    'clave secreta aquí' );
    define( 'NONCE_KEY',        'clave secreta aquí' );
    define( 'AUTH_SALT',        'clave secreta aquí' );
    define( 'SECURE_AUTH_SALT', 'clave secreta aquí' );
    define( 'LOGGED_IN_SALT',   'clave secreta aquí' );
    define( 'NONCE_SALT',       'clave secreta aquí' );
```

Paso 4: Colocar los ficheros en el servidor web

El directorio raíz de la web indicada en el fichero de configuración de apache2 indica el directorio del sistema de archivos local donde están los ficheros contenidos en la carpeta wordpress descargada y extraída en el paso 1. Se crea un fichero de configuración para esta web ejecutando los comandos:

```
cd /etc/apache2/sites-available
cp 000-default.conf elx.guifi.net.conf
cd ../sites-enabled/
ln -s ../sites-available/elx-guifi-net.conf elx-guifi-
net.conf
rm 000-default.conf
```

Se modifica */etc/apache2/sites-available/elx-guifi-net.conf* para añadir estas líneas.

```
DocumentRoot /var/www/html/wordpress
ServerName elx.guifi.net
```

Y finalmente se reinicia el servidor ejecutando el comando:

```
service apache2 restart
```

Paso 5: Ejecutar el script de instalación

Para poder acceder con el navegador web desde Internet a la instalación de Wordpress de forma gráfica a través de http://elx.guifi.net/wp-admin/install.php, hace falta configurar una redirección de puertos en el gateway de Internet para que el tráfico web (puertos 80 y 443) sea redirigido a los puertos correspondientes del servidor web, accesible en la red privada del supernodo de Gaitan2.

INSTALACIÓN DE HTTPS

Se habilita el protocolo HTTPS en el sistema operativo[66]. Para habilitar el módulo SSL, se ejecutan estos comandos:

```
a2enmod ssl
service apache2 restart
```

Se utiliza Certbot, un cliente automático para gestionar los certificados SSL/TLS del servidor web. Para instalarlo se sigue la guía para Debian 8 Jessie y Apache[67]. Se ejecuta el comando:

```
apt-get install python-certbot-apache -t jessie-backports
```

Se crea el fichero de configuración de apache para utilizar HTTPS para la web ejecutando los comandos:

```
cd ../sites-available/
cp default-ssl.conf elx-guifi-net-ssl.conf
cd ../sites-enabled/
ln -s ../sites-available/elx-guifi-net-ssl.conf elx-guifi-
net-ssl.conf
```

Y se modifica el fichero de configuración *elx-guifi-net-ssl.conf* indicando el mismo nombre de servidor que en *elx-guifi-net.conf*.

```
ServerName elx.guifi.net
```

Para conseguir un certificado y permitir que Certbot edite la configuración de Apache se ejecuta el comando:

```
certbot –apache
```

Se eligen las siguientes opciones de configuración de Certbot:

domain: elx.guifi.net

Renew certificate.

Secure option: redirect everithing to https.

De este modo los certificados de Let's Encrypt se renovarán automáticamente y se servirá la web sólo con HTTPS.

INSTALACIÓN DE SFTP

Se crea un usuario del sistema operativo para administrar Wordpress ejecutando el comando:

```
adduser <usr>
```

Nota: sustituir <*usr*> por el nombre de usuario elegido.

Se sigue la guía de ochobitshacenunbyte.com[68] para añadir soporte para SSH y SFTP en Wordpress. Para gestionar las actualizaciones e instalaciones de plugins de Wordpress con protocolos cifrados. Se ejecutan los comandos:

```
cd /var/www/html/wordpress/wp-content/plugins/
```

```
wget https://downloads.wordpress.org/plugin/ssh-sftp-updater-
support.0.7.1.zip
unzip ssh-sftp-updater-support.0.7.1.zip
```

Se da la propiedad de todo el árbol de Wordpress al usuario <*usr*>, y al grupo de usuarios de Apache www-data. Se ejecuta el comando:

```
chown -R <usr>:www-data /var/www/html/wordpress
```

Nota: sustituir <*usr*> por el nombre de usuario elegido anteriormente.

Se limitan los permisos del fichero de configuración ejecutando el comando:

```
chmod 440 /var/www/html/wordpress/wp-config.php
```

Se accede a https://elx.guifi.net/wp-admin/plugins.php con el usuario *usuario_wp* y se activa el plugin SSH SFTP Updater Support.

Se comprueba que funciona instalando un plugin desde https://elx.guifi.net/wp-admin/plugin-install.php escogiendo SSH2 y el host *localhost:*<*puerto*>. Siendo <*puerto*> el puerto de administración SSH el configurado para este dispositivo.

ANEXO III

A partir de la configuración generada en el capítulo de Despliegue de QoS en la red, se puede continuar con esta guía para la configuración del tráfico masivo de los nodos en el gateway de la asociación.

Para el tráfico masivo de cada nodo socio se crea un subárbol de objetos similar al subárbol de la qdisc 1:. El tráfico generado por cualquier otro nodo no socio de la red, clasificado en la clase 20:1 por defecto, se gestiona con una qdisc pfifo_fast por defecto. Los comandos de configuración del control del tráfico son homólogos para el tráfico de cada nodo socio, varían únicamente los identificadores.

Se toma el tráfico de la clase 20:**102** como muestra para explicar la configuración de las clases hermanas. Como se especifica en las políticas de QoS, para cada nodo socio se clasifica el tráfico en 6 tipos que tendrán diferentes prioridades y anchos de banda reservados:

1. Voz sobre IP: SIP, IAX. 10%
2. Comunicación en tiempo real basada en texto: cliente XMPP, cliente XMPPS, Retroshare, IRC. 10%
3. Correo electrónico: IMAP, IMAP sobre TLS, POP3, POP3 sobre SSL, SMTP, SMTPS. 20%
4. Web: HTTP, HTTPS. 50%
5. Transferencia de archivos: FTP, TFTP. 10%
6. Otros tráficos detectados.

La qdisc htb permite asignar diferentes prioridades y anchos de banda a las clases.

```
tc qdisc add dev $DEV parent 20:102 handle 102: htb default 7
```

El tipo de tráfico 6 es “Otros tráficos detectados”, es decir, el tráfico que no sea filtrado por los filtros para los tipos 1 a 5. Éstos últimos se clasifican según los filtros de marca del cortafuegos, tal y como se ha hecho para el tráfico A. El número de la marca de cortafuegos, del identificador de clase y de la prioridad empieza por 2. El 1 ya se utiliza para el tráfico A. El rate es el porcentaje correspondiente del rate de la clase madre; el ceil

es el mismo que la clase madre. La clase madre corresponde a todo el tráfico del nodo socio.

Tráfico tipo 1. Voz sobre IP: SIP, IAX. 10%.

```
tc class add dev $DEV parent 102: classid 102:2 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS / 10 ))kbit ceil $(( $UPRATE * 9 /
10 )) prio 2

tc filter add dev $DEV parent 102: protocol ip prio 1 handle
2 fw flowid 102:2
```

Reglas del cortafuegos para marcar el tráfico de voz sobre IP:

SIP: 5060/UDP, 5060/TCP; IAX2: 4569/UDP

```
iptables -t mangle -A POSTROUTING -o $DEV -p udp -m multiport
--dports 5060,4569 -j CONNMARK --set-mark 2

iptables -t mangle -A POSTROUTING -o $DEV -p udp -m multiport
--dports 5060,4569 -j RETURN

iptables -t mangle -A POSTROUTING -o $DEV -p tcp -m multiport
--dports 5060,4569 -j CONNMARK --set-mark 2

iptables -t mangle -A POSTROUTING -o $DEV -p tcp -m multiport
--dports 5060,4569 -j RETURN
```

Tráfico tipo 2. Comunicación en tiempo real basada en texto: cliente XMPP, cliente XMPPS, Retroshare, IRC. 10%.

```
tc class add dev $DEV parent 102: classid 102:3 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS / 10 ))kbit ceil $(( $UPRATE * 9 /
10 )) prio 3

tc filter add dev $DEV parent 102: protocol ip prio 1 handle
3 fw flowid 102:3
```

Reglas del cortafuegos para marcar el tráfico de comunicación en tiempo real basada en texto:

XMPP-client: 5222/TCP; XMPPS-client: 5223/TCP; XMPP Link-Local Messaging: 5298/UDP, 5298/TCP; Retroshare: 40571/UDP, 40571/TCP; IRC: 194/UDP, 194/TCP

```
iptables -t mangle -A POSTROUTING -o $DEV -p udp -m multiport
--dports 5298,40571,194 -j CONNMARK --set-mark 3
```

```
     iptables -t mangle -A POSTROUTING -o $DEV -p udp -m multiport
--dports 5298,40571,194 -j RETURN

     iptables -t mangle -A POSTROUTING -o $DEV -p tcp -m multiport
--dports 5222,5223,5298,40571,194 -j CONNMARK --set-mark 3

     iptables -t mangle -A POSTROUTING -o $DEV -p tcp -m multiport
--dports 5222,5223,5298,40571,194 -j RETURN
```

Tráfico tipo 3. Correo electrónico: IMAP, IMAPS, POP3, POP3S, SMTP, SMTPS. 20%.

```
     tc class add dev $DEV parent 102: classid 102:4 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS / 5 ))kbit ceil $(( $UPRATE * 9 /
10 )) prio 4

     tc filter add dev $DEV parent 102: protocol ip prio 1 handle
4 fw flowid 102:4
```

Reglas del cortafuegos para marcar el tráfico de correo electrónico:

IMAP: 143/UDP, 143/TCP; IMAP sobre SSL: 993/UDP, 993/TCP; POP3: 110/TCP, 110/UDP; POP3 sobre SSL: 995/UDP, 995/TCP; SMTP: 25/UDP, 25/TCP; SMTPS: 587/UDP, 587/TCP

```
     iptables -t mangle -A POSTROUTING -o $DEV -p udp -m multiport
--dports 143,993,110,995,25,587 -j CONNMARK --set-mark 4

     iptables -t mangle -A POSTROUTING -o $DEV -p udp -m multiport
--dports 143,993,110,995,25,587 -j RETURN

     iptables -t mangle -A POSTROUTING -o $DEV -p tcp -m multiport
--dports 143,993,110,995,25,587 -j CONNMARK --set-mark 4

     iptables -t mangle -A POSTROUTING -o $DEV -p tcp -m multiport
--dports 143,993,110,995,25,587 -j RETURN
```

Tráfico tipo 4. Web: HTTP, HTTPS. 50%.

```
     tc class add dev $DEV parent 102: classid 102:5 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS / 2 ))kbit ceil $(( $UPRATE * 9 /
10 )) prio 5

     tc filter add dev $DEV parent 102: protocol ip prio 1 handle
5 fw flowid 102:5
```

Reglas del cortafuegos para marcar el tráfico web:

HTTP: 80/UDP, 80/TCP; HTTPS: 443/UDP, 443/TCP

```
     iptables -t mangle -A POSTROUTING -o $DEV -p udp -m multiport
--dports 80,443 -j CONNMARK --set-mark 5

     iptables -t mangle -A POSTROUTING -o $DEV -p udp -m multiport
--dports 80,443 -j RETURN

     iptables -t mangle -A POSTROUTING -o $DEV -p tcp -m multiport
--dports 80,443 -j CONNMARK --set-mark 5

     iptables -t mangle -A POSTROUTING -o $DEV -p tcp -m multiport
--dports 80,443 -j RETURN
```

Tráfico tipo 5. Transferencia de archivos: FTP, TFTP, SFTP, FTPS. 10%.

```
     tc class add dev $DEV parent 102: classid 102:6 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS / 10 ))kbit ceil $(( $UPRATE * 9 /
10 )) prio 6

     tc filter add dev $DEV parent 102: protocol ip prio 1 handle
6 fw flowid 102:6
```

Reglas del cortafuegos para marcar el tráfico web:

FTP-data: 20/TCP; FTP: 21/TCP; TFTP: 69/UDP; FTPS-data: 989/UDP, 989/TCP; FTPS: 990/UDP, 990/TCP

```
     iptables -t mangle -A POSTROUTING -o $DEV -p udp -m multiport
--dports 69,989,990 -j CONNMARK --set-mark 6

     iptables -t mangle -A POSTROUTING -o $DEV -p udp -m multiport
--dports 69,989,990 -j RETURN

     iptables -t mangle -A POSTROUTING -o $DEV -p tcp -m multiport
--dports 20,21,989,990 -j CONNMARK --set-mark 6

     iptables -t mangle -A POSTROUTING -o $DEV -p tcp -m multiport
--dports 20,21,989,990 -j RETURN
```

Tráfico tipo 6. Otros tráficos detectados.

```
     tc class add dev $DEV parent 102: classid 102:6 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS / 100 ))kbit ceil $(( $UPRATE *
9 / 10 )) prio 6
```

La figura 25 muestra el subárbol de objetos para controlar el tráfico de un nodo socio en el gateway de la asociación.

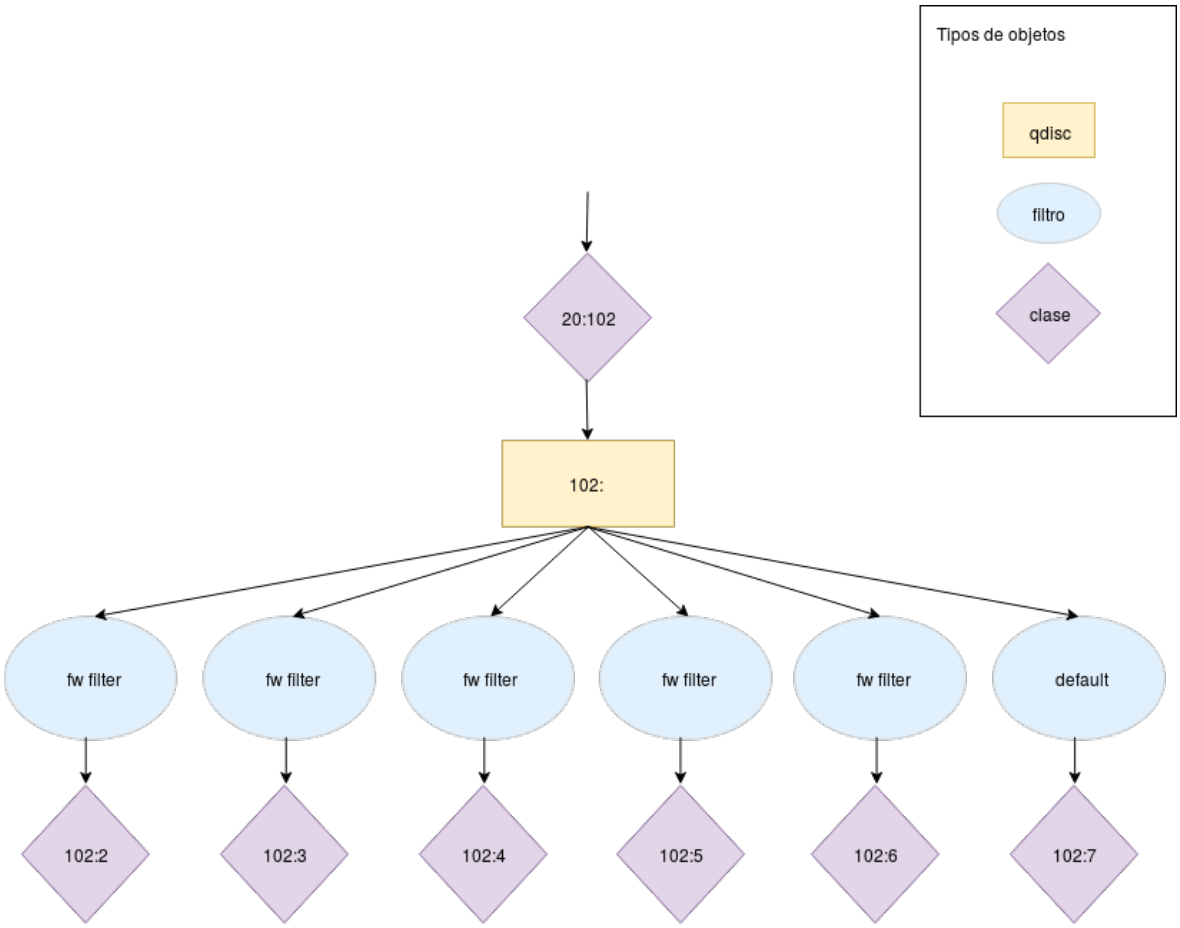


Figura 25: Subárbol de objetos tc para gestionar el tráfico del nodo socio .102

La figura 26 muestra el árbol de objetos para controlar el tráfico de la interfaz, ampliando el árbol descrito en el capítulo de Despliegue de QoS en la red. En las flechas de color gris se repite la configuración de la clase hermana 20:102.

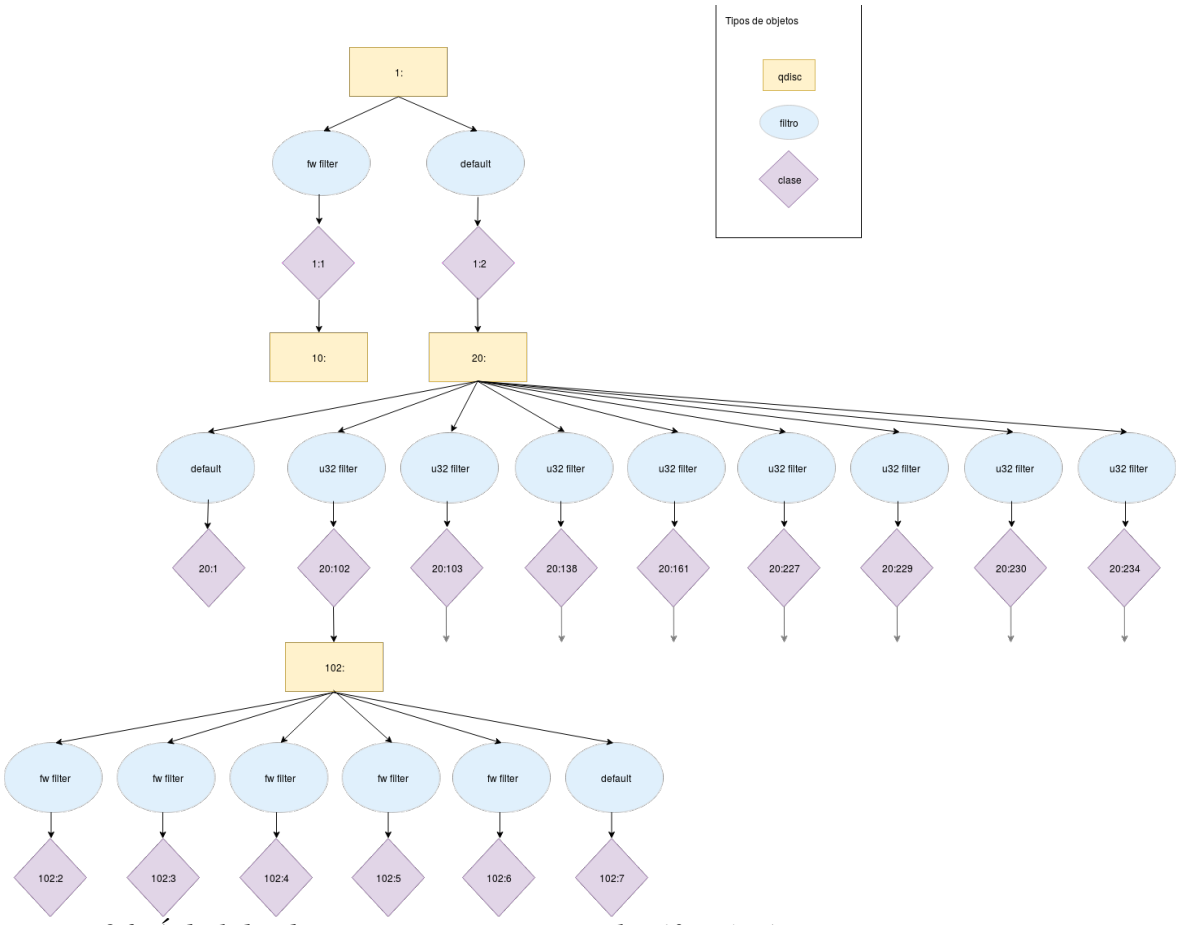


Figura 26: Árbol de objetos tc para gestionar el tráfico (III)

ANEXO IV

El guión de comandos ejecutado en el servidor paralelo, como implantación temporal del servicio es:

```
#!/bin/bash
#
# Inspired by: https://www.iplocation.net/traffic-control

UPRATE=100000
N_SOCIOS=9
DEV=eth0
ADMIN_PORT=



start() {

  # En la raíz hay una disciplina de encolado htb. El tráfico que
no sea clase 1 (A) es clase 2 (B) por defecto.
  tc qdisc add dev $DEV root handle 1: htb default 2

  # El tráfico tipo B tiene tiene control del ancho de banda.
  tc class add dev $DEV parent 1: classid 1:2 htb rate $(( $UPRATE
* 9 / 10 / $N_SOCIOS ))kbit ceil $(( $UPRATE * 9 / 10 ))kbit prio
99
  # Qdisc para repartir ancho de banda entre socios
  tc qdisc add dev $DEV parent 1:2 handle 20: sfq perturb 10

  # El tráfico tipo A.
  tc class add dev $DEV parent 1: classid 1:1 htb rate $
{UPRATE}kbit prio 1
  # La qdisc sfq para hacer un reparto justo del ancho de banda
por flujos.
  tc qdisc add dev $DEV parent 1:1 handle 10: sfq perturb 10
  # Filtrado del tráfico A, en la banda (clase) más prioritaria.
El marcado de los paquetes es 1.
  tc filter add dev $DEV parent 1: protocol ip prio 1 handle 1 fw
flowid 1:1


  # Marcado de los paquetes de conexiones iniciadas.
  iptables -t mangle -A POSTROUTING -j CONNMARK --restore-mark

  # Marcado de los paquetes que inician sesiones de tráfico tipo
A.
```

```
  # A- Tráfico de control y mantenimiento.
  # DNS: 53/UDP, 53/TCP
  # NTP: 123/UDP, 123/TCP
  # SSH: 22/UDP, 22/TCP
  iptables -t mangle -A POSTROUTING -o $DEV -p udp -m multiport
--dports 53,123,22 -m state --state NEW -j CONNMARK --set-mark 1
  iptables -t mangle -A POSTROUTING -o $DEV -p udp -m multiport
--dports 53,123,22 -m state --state NEW -j RETURN
  iptables -t mangle -A POSTROUTING -o $DEV -p tcp -m multiport
--dports 53,123,22 -m state --state NEW -j CONNMARK --set-mark 1
  iptables -t mangle -A POSTROUTING -o $DEV -p tcp -m multiport
--dports 53,123,22 -m state --state NEW -j RETURN

  # Puerto de administración: $ADMIN_PORT/TCP
  iptables -t mangle -A POSTROUTING -o $DEV -p tcp --sport
$ADMIN_PORT -m state --state NEW -j CONNMARK --set-mark 1
  iptables -t mangle -A POSTROUTING -o $DEV -p tcp --sport
$ADMIN_PORT -m state --state NEW -j RETURN

  # ICMP
  iptables -t mangle -A POSTROUTING -p icmp -j MARK --set-mark 1
  iptables -t mangle -A POSTROUTING -p icmp -j RETURN

  # Direcciones IP de los dispositivos de red
  iptables -t mangle -A POSTROUTING -s
192.168.1.7,172.16.8.73,10.229.40.67,10.229.40.68,10.229.40.65,10.
229.40.161,10.229.40.163,172.16.10.77,172.16.13.221,172.16.8.115,1
72.16.2.166,172.16.2.170 -m state --state NEW -j CONNMARK --set-
mark 1
  iptables -t mangle -A POSTROUTING -s
192.168.1.7,172.16.8.73,10.229.40.67,10.229.40.68,10.229.40.65,10.
229.40.161,10.229.40.163,172.16.10.77,172.16.13.221,172.16.8.115,1
72.16.2.166,172.16.2.170 -m state --state NEW -j RETURN

  # El rate del tráfico B entre 9 socios.
  # El identificador de clase corresponde con el último número
decimal de la IP plica de guifi.net del nodo socio.
  iptables -t mangle -A POSTROUTING -s 10.229.40.102 -m state
--state NEW -j CONNMARK --set-mark 102
  iptables -t mangle -A POSTROUTING -s 10.229.40.102 -m state
--state NEW -j RETURN
  #tc filter add dev $DEV parent 20:0 protocol ip prio 1 u32
classid 20:102 match ip src 10.229.40.102/32
  tc class add dev $DEV parent 1: classid 1:102 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 2
  tc qdisc add dev $DEV parent 1:102 handle 102: sfq perturb 10
  tc filter add dev $DEV parent 1: protocol ip prio 2 handle 102
fw flowid 1:102
```

```
  iptables -t mangle -A POSTROUTING -s 10.229.40.133 -m state
--state NEW -j CONNMARK --set-mark 133
  iptables -t mangle -A POSTROUTING -s 10.229.40.133 -m state
--state NEW -j RETURN
  #tc filter add dev $DEV parent 20:0 protocol ip prio 1 u32
classid 20:133 match ip src 10.229.40.133/32
  tc class add dev $DEV parent 1: classid 1:133 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS  ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 2
  tc qdisc add dev $DEV parent 1:133 handle 133: sfq perturb 10
  tc filter add dev $DEV parent 1: protocol ip prio 2 handle 133
fw flowid 1:133

  iptables -t mangle -A POSTROUTING -s 10.229.40.138 -m state
--state NEW -j CONNMARK --set-mark 138
  iptables -t mangle -A POSTROUTING -s 10.229.40.138 -m state
--state NEW -j RETURN
  #tc filter add dev $DEV parent 20:0 protocol ip prio 1 u32
classid 20:138 match ip src 10.229.40.138/32
  tc class add dev $DEV parent 1: classid 1:138 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS  ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 2
  tc qdisc add dev $DEV parent 1:138 handle 138: sfq perturb 10
  tc filter add dev $DEV parent 1: protocol ip prio 2 handle 138
fw flowid 1:138

  iptables -t mangle -A POSTROUTING -s 10.229.40.161 -m state
--state NEW -j CONNMARK --set-mark 161
  iptables -t mangle -A POSTROUTING -s 10.229.40.161 -m state
--state NEW -j RETURN
  #tc filter add dev $DEV parent 20:0 protocol ip prio 1 u32
classid 20:161 match ip src 10.229.40.161/32
  tc class add dev $DEV parent 1: classid 1:161 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS  ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 2
  tc qdisc add dev $DEV parent 1:161 handle 161: sfq perturb 10
  tc filter add dev $DEV parent 1: protocol ip prio 2 handle 161
fw flowid 1:161

  iptables -t mangle -A POSTROUTING -s 10.229.40.227 -m state
--state NEW -j CONNMARK --set-mark 227
  iptables -t mangle -A POSTROUTING -s 10.229.40.227 -m state
--state NEW -j RETURN
  #tc filter add dev $DEV parent 20:0 protocol ip prio 1 u32
classid 20:227 match ip src 10.229.40.227/32
  tc class add dev $DEV parent 1: classid 1:227 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS  ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 2
  tc qdisc add dev $DEV parent 1:227 handle 227: sfq perturb 10
```

```
  tc filter add dev $DEV parent 1: protocol ip prio 2 handle 227
fw flowid 1:227

  iptables -t mangle -A POSTROUTING -s 10.229.40.229 -m state
--state NEW -j CONNMARK --set-mark 229
  iptables -t mangle -A POSTROUTING -s 10.229.40.229 -m state
--state NEW -j RETURN
  #tc filter add dev $DEV parent 20:0 protocol ip prio 1 u32
classid 20:229 match ip src 10.229.40.229/32
  tc class add dev $DEV parent 1: classid 1:229 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS  ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 2
  tc qdisc add dev $DEV parent 1:229 handle 229: sfq perturb 10
  tc filter add dev $DEV parent 1: protocol ip prio 2 handle 229
fw flowid 1:229

  iptables -t mangle -A POSTROUTING -s 10.229.40.230 -m state
--state NEW -j CONNMARK --set-mark 230
  iptables -t mangle -A POSTROUTING -s 10.229.40.230 -m state
--state NEW -j RETURN
  #tc filter add dev $DEV parent 20:0 protocol ip prio 1 u32
classid 20:230 match ip src 10.229.40.230/32
  tc class add dev $DEV parent 1: classid 1:230 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS  ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 2
  tc qdisc add dev $DEV parent 1:230 handle 230: sfq perturb 10
  tc filter add dev $DEV parent 1: protocol ip prio 2 handle 230
fw flowid 1:230

  iptables -t mangle -A POSTROUTING -s 10.229.40.234 -m state
--state NEW -j CONNMARK --set-mark 234
  iptables -t mangle -A POSTROUTING -s 10.229.40.234 -m state
--state NEW -j RETURN
  #tc filter add dev $DEV parent 20:0 protocol ip prio 1 u32
classid 20:234 match ip src 10.229.40.234/32
  tc class add dev $DEV parent 1: classid 1:234 htb rate $
(( $UPRATE * 9 / 10 / $N_SOCIOS  ))kbit ceil $(( $UPRATE * 9 /
10 ))kbit prio 2
  tc qdisc add dev $DEV parent 1:234 handle 234: sfq perturb 10
  tc filter add dev $DEV parent 1: protocol ip prio 2 handle 234
fw flowid 1:234



}

stop() {

# Borrado de objetos.
tc qdisc del dev $DEV root
```

```
iptables -t mangle -F

}

restart() {

# Self-explanatory.
    stop
    sleep 1
    start

}

show() {

  tc -s qdisc show dev $DEV
  iptables -t mangle -L POSTROUTING -v -n
  # -v Muestra un contador de cada flujo.
  # -n Muestra las direcciones IP y puertos en formato numérico.
}

case "$1" in

  start)

    echo -n "Starting bandwidth shaping: "
    start
    echo "done"
    ;;

  stop)

    echo -n "Stopping bandwidth shaping: "
    stop
    echo "done"
    ;;

  restart)

    echo -n "Restarting bandwidth shaping: "
    restart
    echo "done"
    ;;

  show)

    echo "Bandwidth shaping status for $IF:"
    show
    echo ""
    ;;
```

```
  *)

    pwd=$(pwd)
    echo "Usage: qos.script {start|stop|restart|show}"
    ;;

esac

exit 0
```

BIBLIOGRAFÍA Y REFERENCIAS

ESTUDIO DEL ESCENARIO

Castellet Marques, J., 27 septiembre 2016. *2016-09-17 Presentación MUM Madrid*. En: guifi·media. [consulta: 27 febrero 2017]. Disponible en: http://media.guifi.net/media/2016-09-17-presentacion-al-mum-madrid

Digital Rights Lac, 8 abril 2016. *Redes comunitarias: Internet desde la primera milla*. En: Digital Rights Lac, Latin America & The Caribbean. Número 30. [consulta: 22 febrero 2017]. Disponible en: http://www.digitalrightslac.net /es/redes-comunitarias-internet-desde-la-primera-milla

Guifi.net. Guifi.net Wiki. 11 diciembre 2016, 00:13:11. [consulta: 2 marzo 2017]. Disponible en: http://ca.wiki.guifi.net/wiki/Guifi.net

Hub eur@ction project, Febrero de 2004. *Manifiesto De Las Comunidades Inalámbricas*. En: Derechos para Tod@s. Número 18. [consulta: 21 febrero 2017]. Disponible en: http://www.nodo50.org/derechosparatodos /DerechosRevista/Derechos18-Inalambricas.htm

DESPLIEGUE DE QOS EN LA RED

Andreasson, O., 2006. *CONNMARK target*. En: Andreasson, O., *Iptables Tutorial 1.2.2*. [consulta: 25 abril 2017]. Disponible en: http://security.maruhn.com /iptables-tutorial/x9125.html

Devera, M. y Don Cohen, 5 mayo 2002. *HTB Linux queuing discipline manual – user guide*. [consulta: 21 abril 2017]. Disponible en: http://luxik.cdi.cz /~devik/qos/htb/manual/userg.htm

Devera, M. y Hubert, B., 10 enero 2002. *HTB(8).* En: Manuales de Linux.

Eychenne, H. *IPTABLES(8)*. En: iptables 1.6.0.

Firewall mark classifier in tc(8). 21 octubre 2015. En: Manuales de Linux.

Fuster Monzó, J., 2004. *Filtrado de paquetes y QoS*. Universidad Politécnica de Valencia. Disponible en: http://www.redes-linux.com/manuales/ancho_banda/filter_qos.pdf

Graf, T., Maxwell, G., van Mook, R., van Oosterhout, M., Schroeder, P., Spaans, J. y Larroy, P., 19 mayo 2012, 12:40:48. *Linux Advanced Routing & Traffic Control HOWTO*. [consulta: 21 mayo 2017]. Disponible en: http://lartc.org/

Hertzog, R. y Mas R., 2015. 10.3. Quality of Service. En: Hertzog, R. y Mas R., *The Debian Administrator's Handbook*. ISBN: 979-10-91414-04-3. Disponible en: https://www.debian.org/doc/manuals/debian-handbook/index.en.html

Hombashi, T., 2016. *Welcome to tcconfig's documentation!*. [consulta: 29 abril 2017]. Disponible en: https://tcconfig.readthedocs.io/en/latest/index.html

Hubert, B., 16 diciempre 2001. *TC(8)*. En: Manuales de Linux.

Juniper Networks, Inc., 2012. *Chapter 3: Firewall Filters*. En: Juniper Networks, Inc., *JNCIA- Junos Study Guide - Part 2*.

Kuznetsov, A., Hadi Salim, J. y Hubert, B., 16 diciempre 2001. *PRIO(8)*. En: Manuales de Linux.

Netfilter Connmark. To Linux and beyond! [consulta: 25 abril 2017]. Disponible en: https://home.regit.org/netfilter-en/netfilter-connmark/

PING(8). 7 mayo 2014. En: System Manager's Manual: iputils.

Russel, R., 24 enero 2002. *7. Using iptables*. En: Russel, R., *Linux 2.4 Packet Filtering HOWTO*. [consulta: 26 abril 2017]. Disponible en: https://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO-7.html

Russel, S., 4 junio 2014. *The u32 filter*. En: Russel, S., *Linux-tc-notes*. [consulta: 21 mayo 2017]. Disponible en: http://linux-tc-notes.sourceforge.net/tc/doc/cls_u32.txt

Service Name and Transport Protocol Port Number Registry. [consulta: 24 mayo 2017]. Disponible en: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

Tsaousis, C., 23 diciembre 2016. *You should install QoS on all your servers*. [consulta: 16 junio 2017]. Disponible en: https://github.com/firehol/netdata/wiki/You-should-install-QoS-on-all-your-servers

Unión Internacional de Telecomunicaciones, noviembre 2001. *G.1000: Calidad de servicio de las comunicaciones: Marco y definiciones*. Recomendación UIT-T G.1000. Artículo número S 21744. Disponible en: https://www.itu.int/rec/T-REC-G.1000-200111-I/es

Universal 32bit classifier in tc(8). 25 septiembre 2015. En: Manuales de Linux.

Vegas Ayora, J., 3 diciempre 2007. *Estudio y propuestas de mejora de la red GUIFI.NET (II)*. Trabajo de fin de carrera. Escola Politècnica Superior de Castelldefels, Universitat Politècnica de Catalunya.

1 https://guifi.net/es/ProcomunXOLN
2 https://guifi.net/es/que_es
3 https://guifi.net/es/ProcomunXOLN
4 Isabel Ponce, *MONOGRÁFICO: Redes Sociales - Definición de redes sociales*. Online: http://recursostic.educacion.es/observatorio/web/eu/internet/web-20/1043-redes-sociales?start=1
5 https://xat.guifi.net/
6 https://llistes.guifi.net/sympa/
7 http://media.guifi.net
8 http://tv.guifi.net/
9 http://es.wiki.guifi.net/wiki/SAX
10 http://es.wiki.guifi.net/wiki/Portada
11 http://es.wiki.guifi.net/wiki/Proyecto_de_Fin_de_Carrera
12 https://fundacio.guifi.net/Fundaci%C3%B3n
13 http://www.catnix.net/es/miembros/
14 http://es.wiki.guifi.net/wiki/Guifi.net
15 https://guifi.net/es/guifi/menu/stats/nodes
16 https://guifi.net/es/node/17711/view/map
17 http://es.wiki.guifi.net/wiki/Empezar_una_isla_desde_0
18 https://guifi.net/ca/node/35962/view/map
19 https://routerboard.com/RB750GL
20 https://dl.ubnt.com/nanoM5_DS.pdf
21 https://dl.ubnt.com/rocketM5_DS.pdf
22 https://routerboard.com/RB750
23 https://dl.ubnt.com/loco_m5_datasheet.pdf
24 https://dl.ubnt.com/rocketM5_DS.pdf
25 https://guifi.net/es/node/22572
26 https://tools.ietf.org/html/rfc1918
27 https://guifi.net/ca/node/35962/view/ipv4
28 https://www.verkami.com/projects/344-xarxa-wi-fi-ciutadana-de-guifi-net-a-elx
29 https://www.ietf.org/
30 https://www.itu.int
31 http://docwiki.cisco.com/wiki/Quality_of_Service_Networking
32 https://guifi.net/es/ProcomunXOLN
33 Manual de Linux: tc(8)
34 Manual de Linux: tc(8)
35 https://tcconfig.readthedocs.io/en/latest/
36 https://firehol.org/
37 https://github.com/magnific0/wondershaper/
38 http://tcng.sourceforge.net/
39 http://www.shorewall.net/
40 Linux Advanced Routing & Traffic Control, http://lartc.org/
41 Manual de Linux: tc(8)
42 Manual de Linux: tc(8)
43 Linux Advanced Routing & Traffic Control, http://lartc.org/
44 Manual de Linux: tc-u32(8)
45 https://www.netfilter.org/documentation/HOWTO/es/packet-filtering-HOWTO-3.html
46 Manual de Linux: iptables(8)
47 https://forums.gentoo.org/viewtopic-t-739610.html
48 https://www.netfilter.org/projects/conntrack-tools/index.html
49 http://www.system-rescue-cd.org/networking/Load-balancing-using-iptables-with-connmark/
50 https://github.com/firehol/firehol/issues/23
51 Manual de Linux: iptables(8)
52 http://www.tcpdump.org/
53 https://github.com/firehol/netdata/wiki
54 http://www.ex-parrot.com/pdw/iftop/
55 https://firehol.org/tutorial/fireqos-new-user/
56 https://github.com/sivel/speedtest-cli
57 http://ca.wiki.guifi.net/wiki/Servidor_Iperf
58 https://httpd.apache.org/
59 https://www.mysql.com/

60 https://wordpress.org/
61 https://certbot.eff.org/
62 https://codex.wordpress.org/Installing_WordPress
63 http://www.linuxserve.com/2015/05/install-latest-wordpress-version-on.html
64 https://codex.wordpress.org/Installing_WordPress
65 https://api.wordpress.org/secret-key/1.1/salt/
66 https://howto.biapy.com/en/debian-gnu-linux/servers/apache-2/enable-the-https-protocol-with-apache-2-on-debian
67 https://certbot.eff.org/#debianjessie-apache
68 https://www.ochobitshacenunbyte.com/2016/01/21/soporte-ssh-sftp-wordpress/