INSTITUTO UNIVERCITARIO AERONÁUTICO

Sistema de control de acceso discriminado para redes WiFi Implementado sobre hardware embebido

Trabajo Final de Grado

Porchietto Claudio Aníbal

01/01/2014

Declaración de derechos de autor

El presente informe fue desarrollado como trabajo de grado para la carrera Ingeniería en Telecomunicaciones de la Facultad de ingeniería del Instituto Universitario Aeronáutico de la ciudad de Córdoba, Argentina.

El autor, Claudio Porchietto, deja constancia por la presente la autorización y la disposición de este material para la comunidad.

Dedicatoria

Se lo dedico a mis padres y abuelos que con mucho sacrificio me dieron la oportunidad de ser profesional y a los profesores ya que sin ellos no podría haberlo hecho.

Muchas Gracias.

Claudio

Agradecimientos

Ante todo deseo agradecer a mis padres y hermano que me dieron su apoyo, y me alentaron en los momentos difíciles. Agradezco también a los profesores, quienes me guiaron en el proceso de aprendizaje y ayudaron a que hoy llegue a estas instancias.

Título del Proyecto

Sistema de control de acceso discriminado para redes WIFI Implementado sobre hardware embebido.

Hoja de aceptación del Trabajo Final

INSTITUTO UNIVERSITARIO AERONAUTICO –

FACULTAD DE INGENIERÍA

Aprobado por el Departamento de Telecomunicaciones en el cumplimiento de los requisitos exigidos para otorgar el título de Ingeniero en Telecomunicaciones.

Al Señor: Porchietto, Claudio Aníbal DNI: 30.470.134

Revisado por:

……

Tutor del trabajo.

……

Director Depto. Telecomunicaciones.

Tribunal Examinador.

……

Presidente del Tribunal Examinador.

……

Vocal del Tribunal Examinador.

Índice

Índice de ilustraciones.

Glosario

Access Point (AP): Entidad que contiene una estación (STA) y proporciona acceso a los servicios de distribución (DS), a través del medio inalámbrico (WM) para STA asociados. [1]

Basic Service Set (BSS): Un conjunto de estaciones (STA) que se han sincronizado correctamente con el servicio primitives14 JOIN y una STA que se ha utilizado la primitiva START. Por otra parte, un conjunto de STA que han utilizado la especificada primitiva de inicio coincidente perfiles de malla, donde el partido de los perfiles de malla se ha verificado a través del procedimiento de exploración. La pertenencia a un BSS no implica que la comunicación inalámbrica con todos los demás miembros de la BSS sea posible.[1]

Board: Es un PCB en particular que contiene todo el hardware necesario para hacer funcionar un dispositivo. Entre los elementos electrónicos que contiene se pueden encontrar fuentes de alimentación, SoC, memorias, interfaces entre otros. [2][3]

Distribution System (DS): Un sistema utilizado para interconectar un conjunto de basic service set (BSS) y redes de área local (LAN) integrados para crear un conjunto de servicios extendido (ESS).[1]

General Purpose Input/Output (GPIO): Es una solución flexible que controla por software señales digitales. Cada GPIO representa un BIT conectado a un pin en particular. El PCB determina como se conectan con el hardware externo los GPIOs. El firmware determina como se utilizan esos GPIOs.[4] [5]

Makefile: Los makefiles son los ficheros de texto que utiliza make para llevar la gestión de la compilación de programas**.**

Network Interface Controller (NIC): Una tarjeta de red o adaptador de red es un periférico que permite la comunicación con aparatos conectados entre sí. Aunque el término tarjeta de red se suele asociar a una tarjeta de expansión insertada en una ranura interna de un computador se suele utilizar para referirse también a dispositivos integrados en la placa madre de un equipo. Cada tarjeta de red tiene un número de identificación único de 48 bits, en hexadecimal llamado dirección MAC.[6]

Paquetes por Segundo (pps): cantidad de paquetes que se envía en un segundo.[6]

Random Access Memory (RAM): Es la memoria de trabajo del sistema operativo y la mayoría del software que corre en un procesador.

Read Only Memory (ROM): Es un medio de almacenamiento utilizado en ordenadores y dispositivos electrónicos, que permite sólo la lectura de la información y no su escritura. Al menos de manera fácil y rápida.

Round trip time (rtt): tiempo que tarda un paquete de datos enviado desde un emisor en volver a este mismo emisor habiendo pasado por el receptor de destino.[6][1]

System On a Chip (SoC): describe la tendencia cada vez más frecuente de usar tecnologías de fabricación que integran todos o gran parte de los módulos componentes de un ordenador o cualquier otro sistema informático o electrónico en un único circuito integrado o chip.[3][2]

Trivial File Transfer Protocol (TFTP): Es un protocolo muy simple que se usa para transferir archivos. Su nombre proviene a partir de esto. Se ha implementado en la parte superior del protocolo UDP por lo que se puede utilizar para mover archivos entre máquinas en diferentes redes. Tres modos de transferencia son compatibles actualmente: netascii (8 bits ASCII), octeto (raw 8 bit bytes) y mail (El modo correo es obsoleto y no debe aplicarse o utilizarse.)[7]

Virtual LAN (VLAN): Esta norma específica cómo el Servicio de MAC es apoyado por Virtual Bridged LAN, los principios de funcionamiento de las redes, y el funcionamiento de los puentes compatibles con VLAN, incluida la gestión, protocolos y algoritmos. Permitirá la interconexión y separación de las LAN individuales.[8]

Virtual Machine (VM): Una máquina virtual es un software que simula a una computadora y puede ejecutar programas como si fuese una computadora real.

Wi-Fi: es una marca de la Wi-Fi Alliance, la organización comercial que adopta, prueba y certifica que los equipos cumplen los estándares 802.11 relacionados a redes inalámbricas de área local.[1]

Wireless Station (STA): En el diseño de LANs cableadas se asume implícitamente que una dirección es equivalente a una ubicación física. En las redes inalámbricas, este no es siempre el caso. En el estándar IEEE 802.11, la unidad que tiene una dirección es una estación (STA). El término implica más que el origen y/o destino de un mensaje. La STA es un destino de mensaje, pero no (en general) una ubicación fija. La STA puede asumir múltiples características distintas, cada una de las cuales forma su función. Por ejemplo, una sola unidad con dirección puede ser simultáneamente una STA portátil, una STA dependiente, y una STA oculto. [1]

Listado de símbolos y convenciones.

AP: Access Point.

ARP: Address Resolution Protocol.

BSS: Basic Service Set.

BSSID: Basic Service Set Identifier.

CLS: Clear to Send.

DS: Distribution System.

GPIO: General Purpose Input/Output.

Mbit/s: Megabits por Segundo.

mSeg: Milésimas de Segundo.

NIC: Network Interface Controller.

pps: packets per second.

RAM: Random Access Memory.

ROM: Read Only Memory.

RTS: Request To Send.

Rtt: Round trip time.

RX: Receptor.

SoC: System On a Chip.

STA: Estación.

TFTP: Trivial File Transfer Protocol

TX: Transmisor.

VLAN: Virtual Local Área Network.

VM: Virtual Machine.

WLAN: Wireless Local Area Network.

Resumen

En la presente tesis se describe la experiencia de haber implementado sobre un hardware propietario un firmware embebido basado en Linux con el fin de crear un Access Point (AP) en el que discrimine dispositivos de usuarios públicos y usuarios privados.

El desarrollo se realizó en tres etapas. Primero se centralizo el esfuerzo en crear una plataforma de pruebas estable con el objetivo de poder realizar ensayos de rendimiento a fin de poder verificar el impacto de implementar cada uno de los sistemas de filtrado.

La segunda etapa se centró sobre la implementación de un AP en una RaspberryPI con un adaptador WLAN UBS como interfaz inalámbrica. Al poseer una capacidad de cálculo bruta mucho mayor que un AP tradicional, se pensó que este hardware no sería un cuello de botella al aplicar los sistemas de filtrado. Las primeras pruebas fueron contundentes al demostrar que el hardware del RaspberryPI **no posee** las capacidades suficientes para procesar los paquetes a tasas superiores a los 5 Mbit/s.

Estos resultados inesperados provocaron una desviación del plan de trabajo planteado en el anteproyecto, y dieron por concluida la segunda etapa forzando a trabajar directamente sobre el AP TL-730RE dando así comienzo de la tercera fase del proyecto, dado que no es útil o viable simular los sistemas en máquinas virtuales ni implementarlos sobre un RaspberryPI para facilitar el desarrollo. No obstante el objetivo general no cambia.

Con los prototipos de sistema planteados, los equipos móviles de los usufructuarios pertenecientes a los usuarios con privilegios en la red (autentificados) tendrán acceso a la red cableada privada y los usuarios sin privilegios o invitados solo tendrán acceso a una red pública. No obstante la performance puede ser un punto débil.

Para ello se analizó y testeo el framework Netfilter/Iptables y el protocolo VLAN implementados sobre el AP TL-730RE. La elección de la tecnología resultante de este trabajo, se basa en la comparación, contra el AP con firmware original, de las métricas obtenidas para cada nueva configuración del sistema de filtrado planteada sobre un mismo hardware, el AP TL-730RE.

Los resultados superaron ampliamente las expectativas. Al aplicar el filtrado tanto para VLAN como para NETFILTER, el impacto sobre la performances es mínimo. No obstante NETFILTER mostro una tendencia a decaer la performance en la medida que el número de usuarios registrados se incrementa. A pesar de esto, con 63 usuarios el prototipo logro un ancho de banda de 78,3 Mbit/s con una pérdida de paquetes del 0,063 por ciento.

El sistema VLAN no mostro diferencias al manejar distintas cantidades de usuarios y logro una tasa de 92,6 Mbit/s con una pérdida de paquetes del 0,015 por ciento.

Palabras clave

Netfilter/Iptables, Seguridad, Privacidad, Acceso, VLAN

1 Introducción

En este proyecto de grado se propuso implementar un prototipo de Access Point (AP) WI-FI en el que se discrimina dispositivos de usuarios públicos y usuarios privados, permitiendo incrementar la flexibilidad de acceso a la red privada desde cualquier punto de cobertura del Access Point sin comprometer la seguridad. De este modo se mejorará el uso de los recursos de la red.

Dicha implementación se realizó sobre hardware embebido con la distribución Linux OpenWrt release 14.07[3]. De este modo el equipamiento móvil de los usuarios perteneciente a los usuarios con privilegios en la red puede acceder a la red cableada privada y los usuarios sin privilegios o invitados pueden tener acceso solo a una red pública.

1.1 Objetivos del Proyecto

1.1.1 Objetivos generales

Permitir el acceso seguro a usuarios institucionales con dispositivos móviles a través de una red WI-FI pública sin comprometer la seguridad de la red privada.

1.1.2 Objetivos específicos

- Analizar y estudiar las tecnologías intervinientes.
- Implementar un prototipo con una RaspberryPI y un adaptador Wi-Fi.
- Verificar si es factible la implementación sobre un AP TL-WA730RE. De no ser factible la Implementación sobre el AP TL-WA730RE proponer los requerimientos mínimos que debería tener el hardware y que AP cumple con ellos.

1.2 Destinatarios

El presente proyecto se ha estructurado de forma que pueda ser utilizado como parte del Proyecto PIDDEF 41/11 “Monitoreo inteligente remoto de sistemas y redes para la auditoría y seguridad informática” cuyo promotor es el Ministerio de Defensa de la República Argentina, en conjunto con el Instituto Universitario Aeronáutico.

El sistema a implementar está pensado para usuarios móviles con privilegios que necesiten acceso a la red privada desde cualquier AP que brinde servicio público.

1.3 Beneficios esperados

Al aplicar el sistema seleccionado en una red se incrementará su flexibilidad permitiendo brindar acceso a la red privada desde cualquier punto de cobertura del AP sin comprometer la seguridad. De este modo se mejorará el uso de los recursos de la red.

El usuario tendrá la posibilidad de seleccionar por una aplicación web sí es un invitado e ingresar a la red pública o sí es una persona con privilegios y quiera acceder a la red privada

previa autenticación, de este modo se dota de seguridad y flexibilidad a la red con una inversión mínima. No obstante en este proyecto sólo se analiza e implementan los métodos de filtrado de paquetes quedando fuera la autenticación de los usuarios y la alta automatizada.

2 Marco Teórico

2.1 Interfaces de red de Linux

El kernel[9] Linux distingue universalmente entre dos tipos de interfaces de red:

2.1.1 Interfaces de red física

Las entidades eth0, eth8, radio0, wlan19, etc. Siempre representan un dispositivo de hardware de red real, como una NIC, WNIC o algún otro tipo de módem. Tan pronto como el controlador o driver de una interfaz de red física está cargado en el núcleo o kernel se hace presente y disponible.

Cualquier interfaz de red física es llamada por el sistema operativo para que el usuario configure el dispositivo de hardware y también para integrarlo en los programas y scripts.

2.1.2 Interfaces de red virtuales

Las entidades lo, eth0:1, eth0.1, vlan2, br0, pppoe-dsl, gre0, etc. son las interfaces de red virtuales que no representan un dispositivo de hardware existente, pero están vinculados a uno (de lo contrario es inútil). Las interfaces de red virtuales se inventaron para dar al administrador del sistema la máxima flexibilidad al configurar un sistema operativo basado en Linux. Una interfaz de red virtual se asocia generalmente con una interfaz de red física (eth6) u otra interfaz virtual (eth6.9) o ser independiente como la interfaz de bucle invertido lo.

Existen diversos tipos de interfaces de red virtuales:

Alias: eth4: 5, eth4: 6, etc.

Las IP alias son una forma obsoleta de gestionar múltiples direcciones IP/máscaras por interfaz. Herramientas más recientes, como iproute2 admiten varias “direcciones/prefijos” por interfaz, pero todavía los alias son necesarias para la compatibilidad hacia atrás.

VLAN: eth4.0, eth4.1, eth4.3, vlan0, etc.

Se crean para dividir una sola interfaz de red a nivel de capa 2 en múltiples interfaces virtuales. Los drivers de todas las tarjetas de red que participan deben ser compatibles con IEEE 802.1Q y ser configurados en consecuencia. Este estándar permite hasta 4096 VLANs (12bits)[8].

El documento de estándares de referencia es 801.2q especifica que VID valores 0 y 4095 no pueden utilizarse para el etiquetado de los paquetes, ya que denotan valores reservados - VID 0 es el valor por defecto vlan 'nativa' - dejando 4.094 valores válidos en el medio, aunque VID 1 es a menudo reservada para la gestión de la red. Esto significa vlan0 se puede utilizar como una VLAN dentro o entre dispositivos, pero no se puede etiquetar paquetes con él.

Existen tres niveles para de VLAN que engloban distintos mecanismos. Los más importantes son:

- VLAN de nivel 1 (por puerto). Se especifica qué puertos del switch pertenecen a la VLAN, los miembros de dicha VLAN son los que se conecten a esos puertos.
- VLAN de nivel 2 por direcciones MAC. Se asignan hosts a una VLAN en función de su dirección MAC.
- VLAN de nivel 2 por tipo de protocolo. La VLAN queda determinada por el contenido del campo tipo de protocolo de la trama MAC.
- VLAN de nivel 3 por direcciones de subred o IP.

Estos niveles se utilizan para indicar en que capa del modelo OSI trabaja para discriminar los paquetes.

Un puerto trunk o, switchport mode trunk, puede ser miembro de múltiples VLANs al mismo tiempo. Todos los paquetes que ingresen o salgan de él tienen que estar etiquetados.

Bridgeds: br0, br-lan, …

Se utilizan para hacer varias interfaces de red virtuales o físicos, actúan como si fueran una sola interfaz de red (casi lo opuesto a la VLAN). También se puede utilizar para VPN y las interfaces de puente. El puente Ethernet Linux puede ser usado para conectar múltiples dispositivos Ethernet juntos. La conexión es totalmente transparente: los hosts conectados a un dispositivo Ethernet pueden ver hosts conectados directamente a los otros dispositivos Ethernet.

Interfaces de túnel: pppoe-dsl, pppoa-dsl, tun0, vpn1, …

Utilizadas para enviar paquetes a través de un protocolo de túnel como GRE, IPSec, PPPoE, etc.

Interfaces inalámbricas: wlan0, wlan0_1, ath3, ath_monitor, …

Subsistema inalámbrico Linux: A nivel de kernel siempre hay una interfaz de red física para cada WNIC llamada la interfaz principal. Se las denota por phy0, phy1, etc. La interfaz principal es invisible para el usuario. Entonces, dependiendo del modo de operación inalámbrica de la interfaz principal se crean interfaces de red virtuales inalámbricas con diferentes propiedades. Las configuraciones pueden ser: ad-hoc (IBSS), managed, AP, WDS, mesh, monitor. Esto se hace automáticamente de forma predeterminada. Cuando se carga el controlador WNIC, siempre habrá la interfaz principal y (al menos) una interfaz virtual asociada.

2.2 Netfilter: iptables, ebtables

Netfilter es un framework de Linux incorporado en el kernel [10]que permite interceptar y modificar paquetes IP. Cuenta con herramientas en espacio de usuario que permiten manipular las tablas y las cadenas.

Hay tres conceptos básicos en iptables y ebtables:

- Reglas
- Cadenas
- Tablas

2.2.1 Reglas

Una regla especifica una condición y una acción. Una condición es una característica que un paquete debe cumplir para ejecutar una acción. Por ejemplo, si es TCP aceptar.

2.2.2 Cadenas

Una cadena es una lista ordenada de reglas. Para cada paquete se va comprobando si se le aplica cada regla de la cadena. Si una regla **NO** se aplica a un paquete, se pasa a la siguiente. Si una regla **SI** se aplica a un paquete, se ejecuta la acción definida en dicha regla y ya no se comprobaran más reglas de la cadena.

Una cadena puede tener definida una política, que es la acción que ejecuta de no cumplir ninguna regla. Por defecto todas las políticas son aceptar los paquetes.

Hay diversas cadenas predefinidas: PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING y BROUTING.

Cadena PREROUTING: Reglas que se aplican a los paquetes que llegan a la máquina. Esta cadena se ejecuta antes de comprobar si el paquete es para la propia máquina o hay que reenviarlo.

Cadena INPUT: Reglas que se aplican a los paquetes destinados a la propia máquina. Esta cadena se ejecuta justo antes de entregarlos a la aplicación local.

Cadena FORWARD: Reglas que se aplican a los paquetes que han llegado a la máquina pero van destinados a otra y hay que reenviarlos. Esta cadena se ejecuta antes de consultar la tabla de encaminamiento ya sea por un reenvío por un IP o por mac.

Cadena OUTPUT: Reglas que se aplican a los paquetes creados por la propia máquina. Esta cadena se ejecuta justo después de que la aplicación le pase los datos a enviar al kernel del sistema operativo y antes de consultar la tabla de encaminamiento.

Cadena POSTROUTING: Reglas que se aplican a los paquetes que salen de la máquina, tanto los creados por ella como los que se reenvían. Esta cadena se ejecuta después de consultar la tabla de encaminamiento.

Cadena BROUTING: Reglas que se aplican a los paquetes que llegan a la máquina. Esta cadena se ejecuta antes de comprobar si el paquete es para la propia máquina o hay que reenviarlo. Se ejecuta antes inclusive que la cadena PREROUTING.

2.2.3 Tablas:

Una tabla contiene un conjunto de cadenas. Cada tabla engloba las reglas.

Las tablas por defecto son: filter, nat mangle, raw, brouting.

- *Filter:* engloba las reglas de filtrado de paquetes, es decir, de las que deciden que un paquete continúe su camino o sea descartado.
- *Nat:* engloba las reglas de modificación de direcciones IP y puertos de los paquetes
- *Mangle:* engloba las reglas de modificación de algunos campos de las cabeceras del paquete.
- *Raw:* engloba las reglas que permiten marcar excepciones al seguimiento que hace el kernel de las "conexiones" de la máquina.
- *Broute:* engloba las reglas que modifican las direcciones mac origen y destino de los paquetes.

En la ilustración 1[11] se ve como está ubicada cada cadena en las capas de IP.

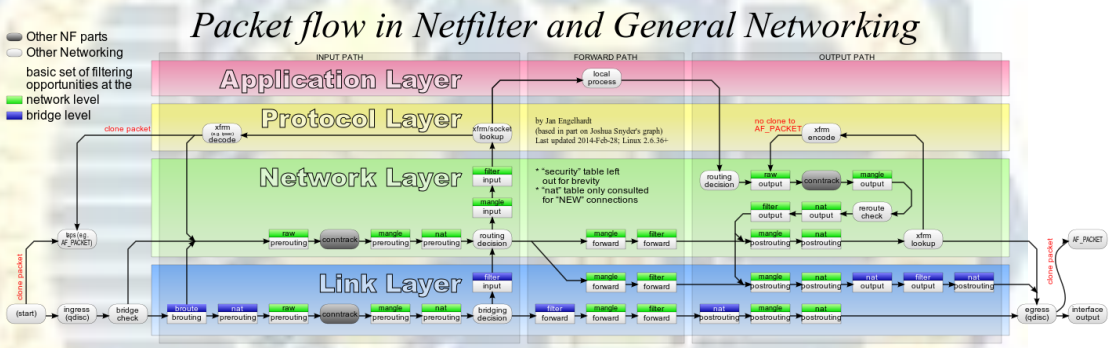


Ilustración 1 Flujo del paquete por Netfilter visto por capas

En la ilustración 2 se describe como transita un paquete que atraviesa la pila de protocolos en un kernel Linux con Netfilter y el procesado por las cadenas dentro de cada tabla.

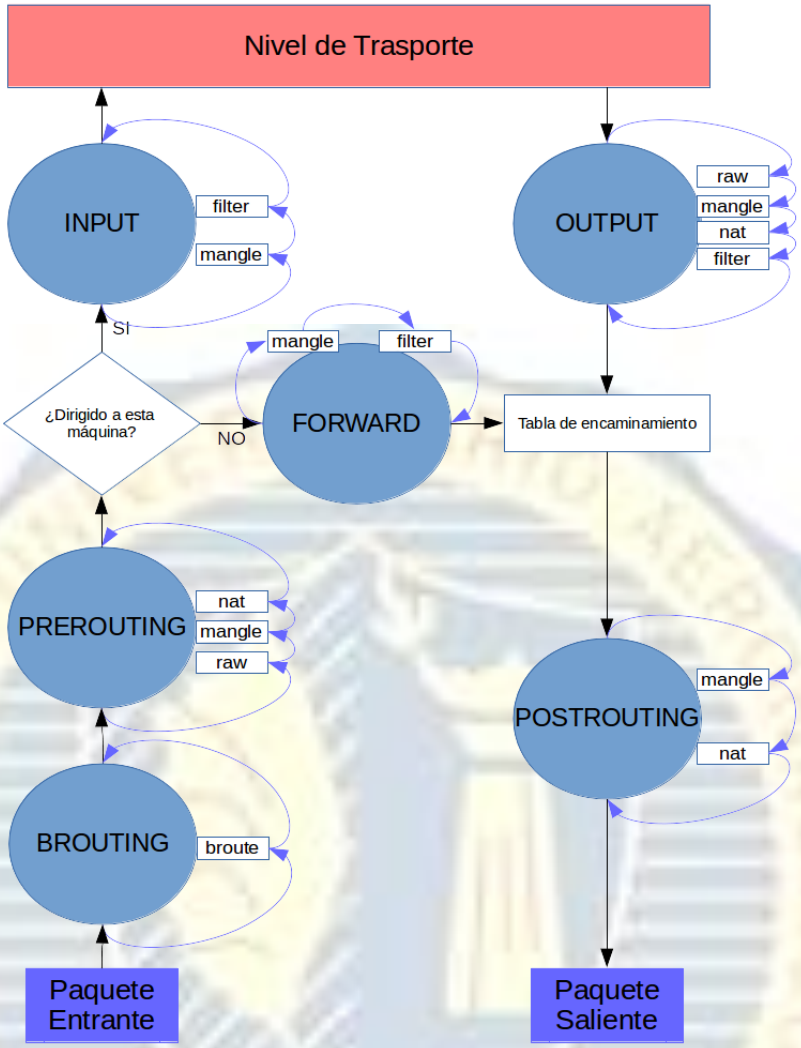


Ilustración 2 Diagrama de flujo de un paquete que atraviesa netfilter.

2.3 Redes de paquetes, teoría de colas y congestión.

Existen artículos que estudian la relación entre el tráfico de la red con el rendimiento y el retardo de la misma [13]. Surge del modelo de colas M/M/1 que el retardo estadístico tiende a infinito al acercarse a la capacidad del canal [12]. El rendimiento está definido como la tasa de datos que logra atravesar la red dividido el ancho de banda del canal.

2.3.1 Comportamiento ideal

En la Ilustración 3 se muestra el comportamiento ideal de la utilización de una red. La gráfica superior representa el rendimiento de la red, número de paquetes enviados al sistema en función de la carga ofrecida, número de paquetes transmitidos por el sistema, ambos parámetros normalizados al rendimiento máximo teórico de la red.

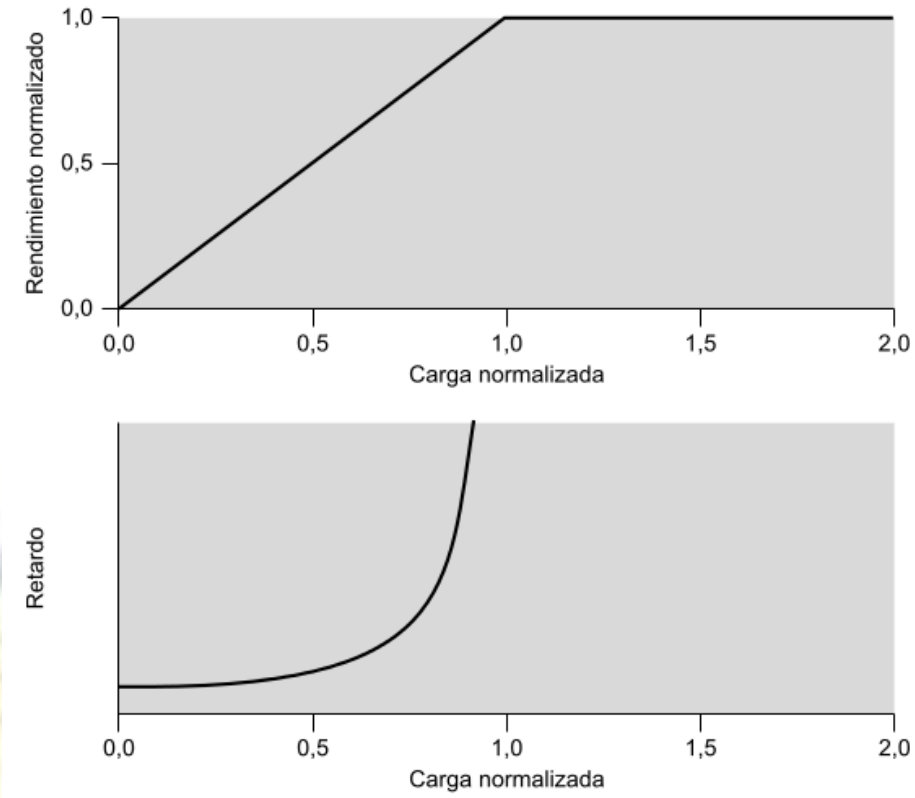


Ilustración 3 Comportamiento Ideal.

En el caso ideal, el rendimiento de la red crece hasta aceptar una carga igual a la capacidad total de la red, permaneciendo el rendimiento normalizado a valor 1.0 para cargas de entrada superiores. Obsérvese, sin embargo, lo que sucede con el retardo extremo a extremo medio experimentado por un paquete, incluso bajo esta suposición de funcionamiento ideal. Cuando la carga es despreciable, existe un retardo pequeño constante consistente en el retardo de propagación a través de la red desde el origen hasta el destino más un retardo de procesamiento en cada nodo. A medida que la carga de la red aumenta, al valor de retardo fijo anterior se suman el retardo estadístico de encontrar el canal ocupado en el momento de acceder.

2.3.2 Comportamiento real.

En el caso ideal se ha supuesto que las memorias o búferes son infinitas. En la práctica, las memorias son finitas, lo que provoca desbordamientos o pérdida de paquetes.

La carga de la red es pequeña cuando la tasa de servicio es muy superior a la tasa de paquetes. En este caso la colas son pequeñas y el retardo esta mayor mente dado por la tasa de servicio.

Conforme aumenta la carga las colas crecen. Esto provoca que el retardo en este punto este definido por la tasa de servicio y por el retardo estadístico de la cola provocando una congestión moderada. Estadísticamente la probabilidad de encontrar la cola llena

aumenta por lo que el rendimiento no crece proporcional a la carga y el porcentaje de pérdida de paquetes crece.

Eventualmente la red se satura y el retardo tiende a infinito.

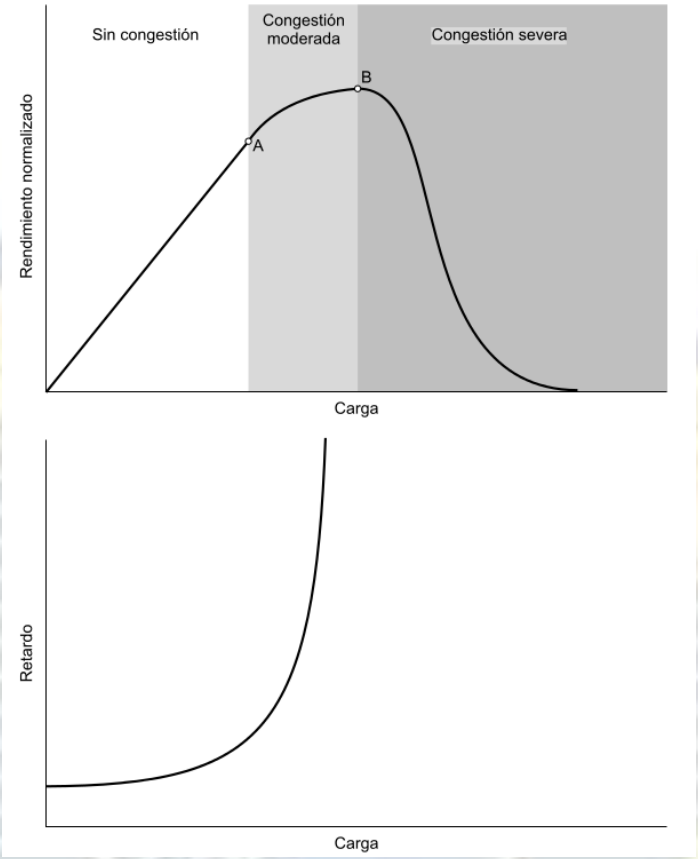


Ilustración 4 Efectos de congestión en la red.

3 Estudio técnico.

Dado la naturaleza de esta tesis se optó por trabajar directamente sobre el hardware propuesto para su implementación. Omitiendo lo planteado originalmente de simular el entorno con Virtual Box.

La simulación no presenta mejores facilidades de trabajo que la plataforma real ya que en la misma se tiene comunicación con cada uno de los elementos por medio de la red de gestión.

Además, la simulación si presenta grandes inconvenientes. Los módulos de kernel del firmware OpenWrt varían de un dispositivo a otro. Con lo que la configuración funcional para Virtual Box no funcionaría en el SoC Atheros.

Las pruebas preliminares arrojaron que aunque se estimaba que el RaspberryPI tendría un rendimiento muy superior al AP TP-Link el resultado fue todo lo contrario. El AB logrado por el RaspberryPI era cercano a un tercio del AB logrado por el AP. Tras un análisis superficial se detectó que el proceso *"kworker"* del sistema operativo Linux que corre en la RaspberryPI generaba numerosas interrupciones al procesador, hasta el punto de saturarlo. Esto es posible a un pequeño bug en el kernel o simplemente que el SoC BCM2835 del RaspberryPI no esté debidamente optimizado para el manejo de paquetes Ethernet.

Es por todos estos motivos que se va a trabajar directamente sobre el hardware del AP TP-LINK TL-730RE V1 para el desarrollo del sistema.

3.1 Entorno de desarrollo.

Es imperativo para el desarrollo y evaluación de los sistemas de discriminación de tráfico propuestos contar con una plataforma estable que permita verificar el impacto en la performance dada por las distintas configuraciones de AP. Se asume que aunque la configuración del switch cambia para los sistemas propuestos no afectara de manera significativa a las métricas.

De la teoría de cogestión vista en la sección 2.3 podemos extraer que los indicadores que se tienen que analizar son: RTT, pérdida de paquetes, AB RX y AB TX. Los gráficos de congestión surgen de poner en las restantes métricas en función del AB TX.

- Rtt vs AB TX.
- Paquet lost vs AB TX.
- AB RX vs AB TX.

Las cuaternas (rtt, paquet lost, AB RX, AB TX) son obtenidas mediante el Script "*test_red.sh*" ejecutado en la plataforma de desarrollo.

La plataforma de trabajo es la siguiente:

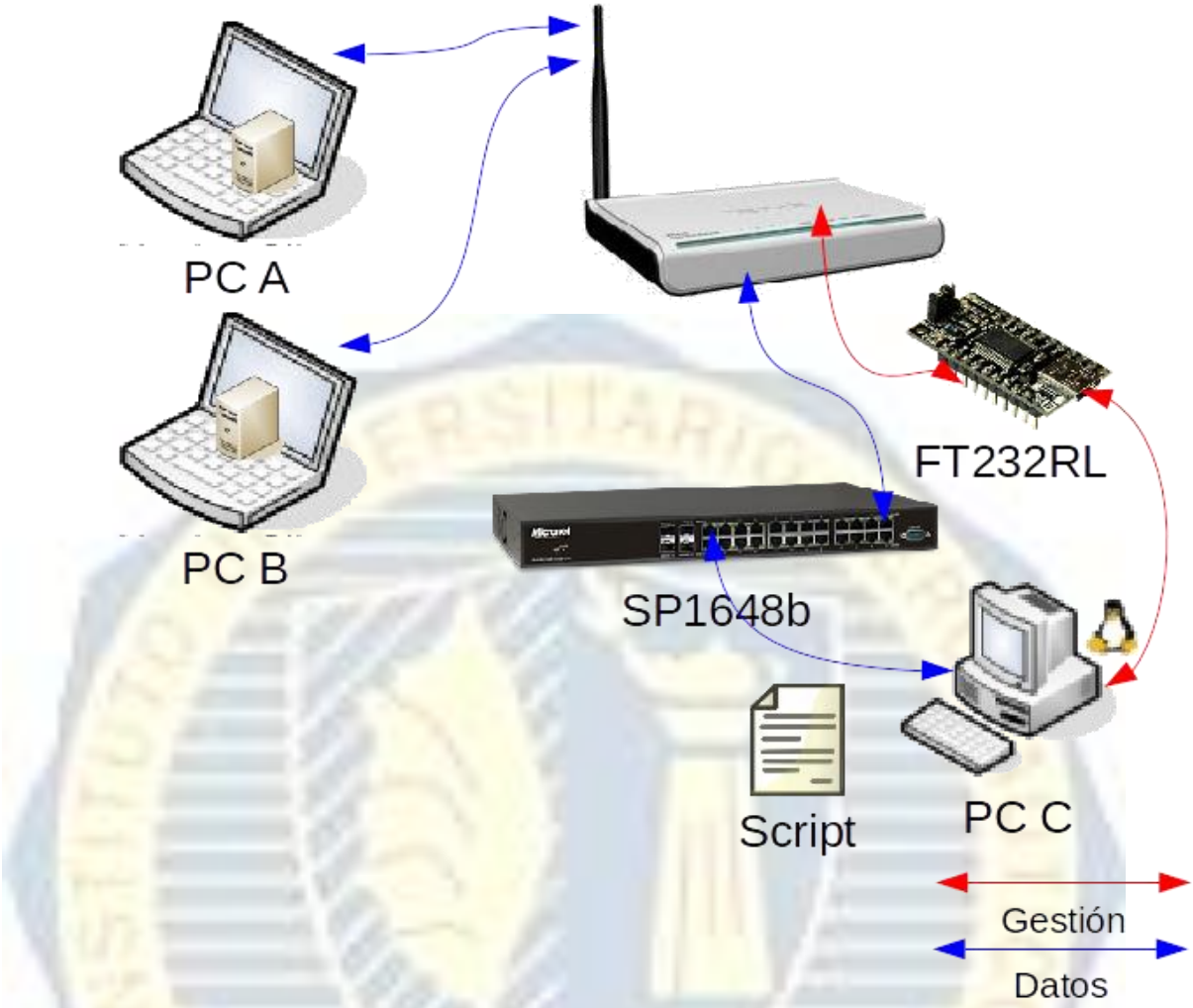


Ilustración 5 Plataforma de pruebas y desarrollo.

Para realizar un diagrama de congestión de la red se ejecuta el script "*test_red.sh*" en la PC de desarrollo indicada como PC C. El mismo genera un flujo de datos UDP a tasa fija. En cada iteración de la prueba genera un vector de cinco elementos: ancho de banda transmitido en Mbit/s, round trip en mseg, ancho de banda recibido por el servidor en Mbit/s, Jitter en mseg y porcentaje de paquetes perdidos. Para la próxima iteración se incrementa el AB TX entre la PC C y PC A. de este modo se obtienen todos los puntos de este sistema quinta dimensional. El Jitter es un valor extra que nos brinda la herramienta iperf, se decidió guardarlo junto a los datos relevantes para posteriores análisis fuera de esta tesis. Los mecanismos del script serán explicados en mayor profundidad más adelante.

El camino que siguen los paquetes entre el cliente y servidor de iperf es pasando por una interfaz Gigabit Ethernet al switch SP1648b. El switch redirige los paquetes por una interfaz Fast Ethernet al AP. El AP procesa los paquetes y a los que le corresponda pasar los trasmite a la PC A por una interfaz WiFi norma "n" a tasa fija de 150 Mbit/s donde iperf en modo servidor guara información de los paquetes que recibe entregándosela al script al final

de cada iteración de la prueba. De este modo se confecciona cada uno de los puntos del diagrama de congestión.

Se considera el que al no tener ningún sistema de filtrado el AP quien ejerce de cuello de botella o limitante del ancho de banda total del sistema es la modulación WiFi o el puerto Fast Ethernet, en cualquiera de los dos casos al realizar una prueba con el AP por defecto o como viene de fábrica nos dará una lectura base para poder comprar contra las lecturas de la distintas configuraciones de AP.

En el momento que el procesador del AP tenga que aplicar reglas para filtrar los paquetes o dirigirlos a las distintas VLAN se considera que aparecerá un nuevo cuello de botella. Este puede ser inapreciable y no afectar a la performance de la red o todo lo contrario. Justamente Esto es lo que se trata de determinar en esta tesis.

3.2 Script test_red.sh

El Script test_red.sh se ejecuta en PC "C" y conecta con los servidores Iperf alojados en PC "A" o "B".

Iperf es una herramienta que se utiliza para hacer pruebas en redes informáticas. Crear flujos de datos TCP y UDP según la prueba que realice. Iperf puede funcionar como cliente o como servidor debiendo estar cada uno en un extremo del enlace a medir.

Por ejemplo para probar el rendimiento de la red con la Configuración predeterminada desde el cliente:

#>iperf -c 192.168.66.39

Client connecting to 192.168.66.39, TCP port 5001
TCP window size: 64.0 KByte (default)

---

[3] local 192.168.66.35 port 62427 connected with 192.168.66.39 port 5001
[ID] Interval Transfer Bandwidth
[3] 0.0-11.1 sec 21.1 MBytes 16.0 Mbits/sec
Las pruebas de TCP y UDP son diferentes:

Desde el servidor:

#>iperf -s

Server listening on TCP port 5001
TCP window size: 64.0 KByte (default)

---

[4] local 192.168.66.39 port 5001 connected with 192.168.66.35 port 62427
[ID] Interval Transfer Bandwidth
[4] 0.0-11.1 sec 21.1 MBytes 16.0 Mbits/sec

El equipo cliente se conecta al equipo servidor en el puerto TCP 5001 y solo se mide el ancho de banda del cliente al servidor.

La herramienta admite muchos parámetros de configuración. Aquí se detallan los más utilizados:

- -f Formato de los datos, permite forzar la salida de datos a una unidad. Ej. Mbits/sec.
- -r Ancho de banda bidireccional, mide en ambas direcciones pero no de manera simultánea.
- -d Ancho de banda bidireccional simultaneo
- -w Tamaño de la ventana de TCP
- -p, permite especificar el puerto donde escuchar o preguntar para el cliente o servidor. Por defecto es el 5001
- -t, temporización, determina el tiempo que dura la prueba.
- -i intervalo que da entre informes sin parar la prueba.
- -u, -b Análisis UDP, configuración de ancho de banda.
- -P Análisis en paralelo, se utiliza para correr pruebas en paralelo.

UDP: Cuando se utiliza el protocolo UDP, Iperf en modo cliente permite al usuario especificar el tamaño de los datagramas y la tasa de transmisión. No importa si los paquetes se pierden no los retransmitirá. Iperf en modo servidor, el que está al otro extremo de la red, proporciona resultados del rendimiento y de los paquetes perdidos.

TCP: Cuando se utiliza TCP, Iperf mide el rendimiento de la carga útil sobre un flujo TCP. Admite configurar el tamaño de los paquetes y la cantidad de conexiones que genera.

El siguiente ejemplo muestra lo que ejecuta en cada una de las iteraciones del script. La opción –b permite utilizar el ancho de banda deseado.

Cliente

```
#>iperf -c 192.168.1.3 -u -b 10m
------------------------------------------------------------
Client connecting to 192.168.66.39, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 64.0 KByte (default)
------------------------------------------------------------
[ 3] local 192.168.66.35 port 55753 connected with 192.168.66.39 port 5001
[ ID] Interval Transfer Bandwidth
[ 3] 0.0-10.0 sec 11.9 MBytes 10.0 Mbits/sec
[ 3] Sent 8505 datagrams
[ 3] Server Report:
[ 3] 0.0-10.0 sec 11.9 MBytes 10.0 Mbits/sec 0.000 ms 0/ 8504 (0%)
[ 3] 0.0-10.0 sec 1 datagrams received out-of-order
```

Servidor

#>iperf -s -u -i 1

Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 64.0 KByte (default)

---

[3] local 192.168.66.39 port 5001 connected with 192.168.66.35 port 55753
[ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams

[3] 0.0- 1.0 sec 1.19 MBytes 10.0 Mbits/sec 0.000 ms 0/ 852 (0%)
[3] 1.0- 2.0 sec 1.19 MBytes 10.0 Mbits/sec 0.069 ms 0/ 850 (0%)
[3] 2.0- 3.0 sec 1.19 MBytes 9.98 Mbits/sec 0.000 ms 0/ 849 (0%)
[3] 3.0- 4.0 sec 1.19 MBytes 10.0 Mbits/sec 0.000 ms 0/ 851 (0%)
[3] 4.0- 5.0 sec 1.19 MBytes 10.0 Mbits/sec 0.000 ms 0/ 850 (0%)
[3] 5.0- 6.0 sec 1.19 MBytes 10.0 Mbits/sec 0.000 ms 0/ 850 (0%)
[3] 6.0- 7.0 sec 1.19 MBytes 10.0 Mbits/sec 0.000 ms 0/ 851 (0%)
[3] 7.0- 8.0 sec 1.19 MBytes 10.0 Mbits/sec 0.000 ms 0/ 850 (0%)
[3] 8.0- 9.0 sec 1.19 MBytes 10.0 Mbits/sec 0.000 ms 0/ 850 (0%)
[3] 9.0-10.0 sec 1.19 MBytes 10.0 Mbits/sec 0.000 ms 0/ 851 (0%)
[3] 0.0-10.0 sec 11.9 MBytes 10.0 Mbits/sec 0.000 ms 0/ 8504 (0%)
[3] 0.0-10.0 sec 1 datagrams received out-of-order

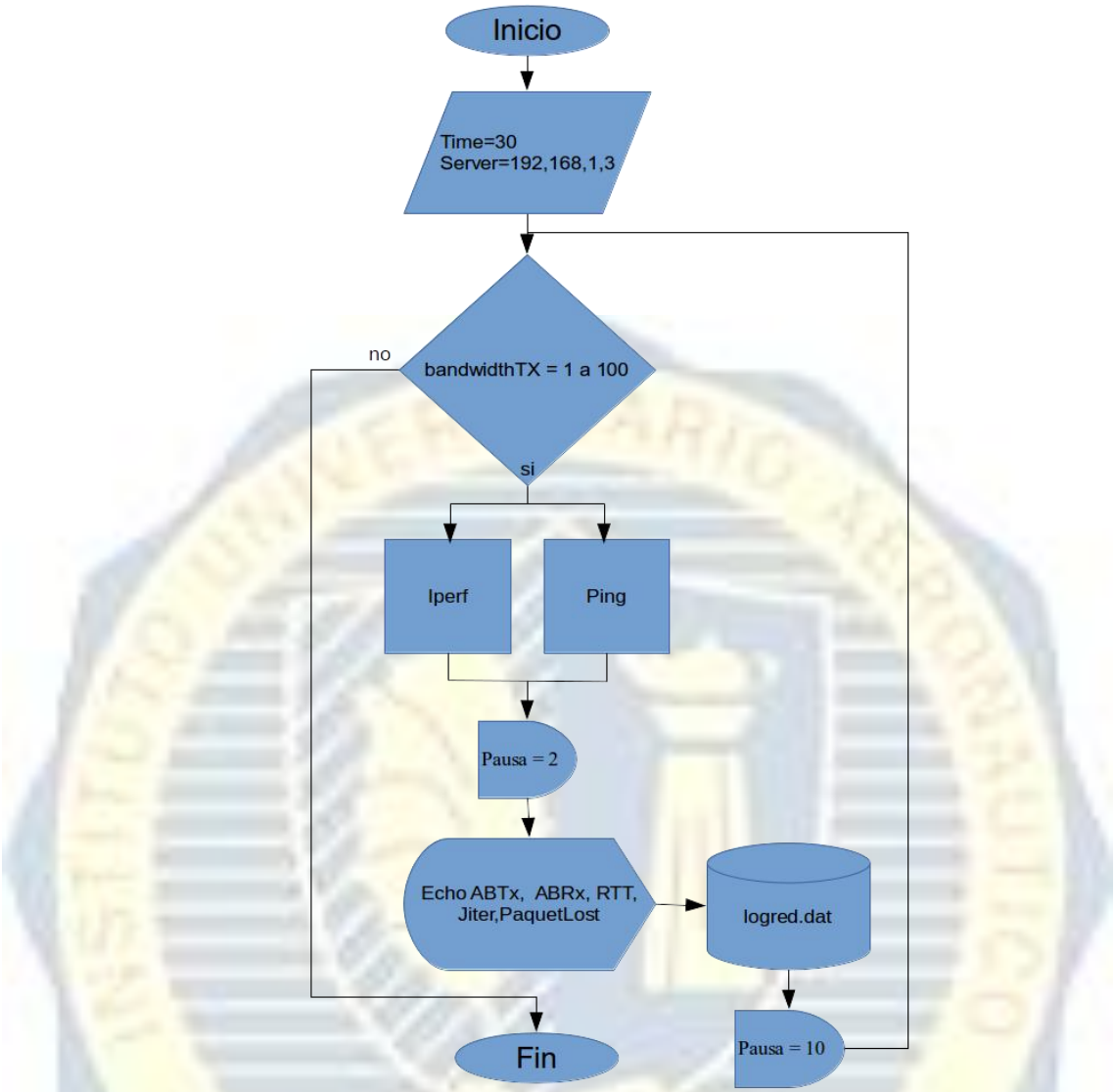


Ilustración 6 Script test_red.sh.

El Script visto en la ilustración 6 carga el canal con paquetes UDP con tasas que van desde 1 a 100 Mbit/s. Cada prueba de carga del canal dura lo que determine la variable "time" en segundos. Simultáneamente se ejecuta un ping a una tasa de 1 pps. El retardo de todos los paquetes de sondeo es promediado. Se consideró que el ancho de banda que ocupan los paquetes de sondeo de la herramienta "ping" no afecta la lectura de la herramienta "iperf" ya que sus magnitudes son muy diferentes.

El valor de la variable "time" se determinó de forma empírica, realizando numerosas pruebas sobre la plataforma hasta obtener resultados estables.

La información devuelta por "iperf" y ping es parseada con AWK y guardada en el archivo de texto plano logred.dat para luego levantar y procesar con Excel.

El resultado de este ensayo es una matriz de cinco columnas por cien filas. Procesándola con Excel se construye el diagrama de congestión. Comparando los diagramas de

las distintas configuraciones dará cual es el AB efectivo de la red para cada configuración y se podrá verificar cuanto impacta la carga de procesamiento de paquetes en el sistema, demostrando si es viable su implementación o no.

3.2.1 Pseudocódigo de test_red.sh

"Time" es igual a 30
"server" es igual a 192.168.1.3
Crea e inserta la cadena de texto "bandwidthTX(Mbit/s), rt(mseg), bandwidthRX(Mbit/s), Jiter(mseg), PaquetLost(%)" en el archivo logred.dat
mientras bandwidthTX este entre 1 y 100
ejecuta:
genera un flujo de datos UDP de ancho de banda "bandwidthTX" hacia "server" y simultáneamente medir el retardo contra "server"
espera 2 segundos
agrega "el valor bandwidthTX, el valor de RTT, el valor de bandwidthRX , el valor de Jiter , el valor de PaquetLost al final de logred.dat
espera 10 segundos
ejecuta la próxima iteración

3.2.2 Ejemplo de salida del script.

bandwidthTX(Mbits)	rt(mseg)	bandwidthRX(Mbits)	Jiter(mseg)	PaquetLost(%)
1	1,328	1	0,212	0
2	1,221	2	0,014	0
3	1,221	3	0,262	0
4	1,303	4	0,108	0
5	1,325	5	0,067	0
6	1,45	6	0,062	0
...	...	...	...	...
95	1,057	86,4	0,19	9,6
96	14,504	58,7	0,178	39
97	3,5316	87,1	13,312	8,9
98	1,4092	52,9	0,172	45
99	4,7213	85,7	0,528	10
100	1,39	72,6	1,431	24

3.3 Aspectos básicos de un AP

En general para que un dispositivo pueda considerase un AP tiene que contener una STA que provea acceso a las STA asociadas, por un medio inalámbrico, al BSS.

Como referencia se tomará el AP TL-WA730RE que contiene el board CA-F120 Rev 1.1

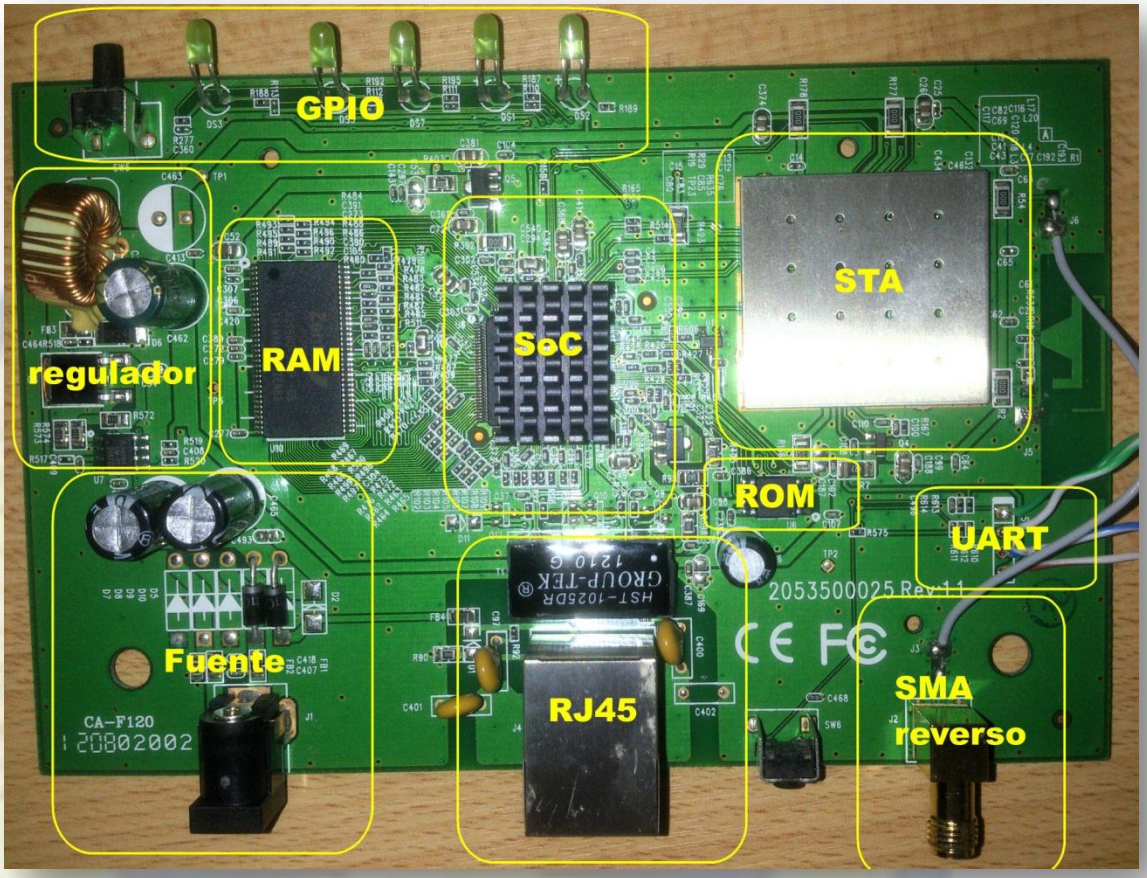


Ilustración 7 Board CA-F120.

Este board en particular tiene el SoC AR7240 de Atheros.

Los SoC se diseñan para fines no específicos. Es por ello que se puede encontrar el mismo SoC en distintos dispositivos y no todos GPIOs y buses del SoC están conectados.

En el AP de referencia se encuentran los siguientes elementos:

- **Fuente:** su fin es estabilizar la tensión de entrada.
- **GPIO:** compuesto por los LED y dos pulsadores que sirven para mostrar el estado del AP y enviar información al mismo. En el firmware original uno de los pulsadores se utiliza para enviarle una petición de reset y el otro para disponerlo como un repetidor. Cabe aclarar que el firmware es quien captura estas peticiones y no el hardware.

- **RAM:** Es la memoria volátil del AP. En este AP en particular contiene un ddr de 32MB.
- **Regulador:** convierte la tensión de entrada a las distintas tensiones que necesita el SoC, la STA y el resto de elementos.
- **RJ45:** conector para la interfaz Ethernet.
- **ROM:** Es la memoria no volátil. Este dispositivo en particular contiene S25FL032P, que es una memoria flash de 32 Mbit con SPI. Están divididos en 256 páginas de dos bloques de 64 Kbit. Esto da un total de 4 Mbyte utilizables para el firmware.
- **SMA:** conector para la antena WI-FI que opera a 2.4GHz
- **SoC:** Es el corazón de todo AP. Toda la información se procesa en esta unidad. En este SoC en particular el procesador es un MIPS de 24K.
- **STA:** es la interface física hacia el medio inalámbrico.
- **UART:** Todo sistema basado en Linux tiene una entrada y salida estándar del terminal. En este dispositivo en particular se redirige al UART. Es un puerto serie que opera entre 0 y 3.3v a 115200 Kbit/s de 8 bits sin paridad, un bit de parada y control de flujo por software o como se lo suele nombrar 115200 8N1.[14] Los usuarios finales no utilizan esta interfaz sólo se usa para depurar el código del firmware o tener un control más directo del mismo.

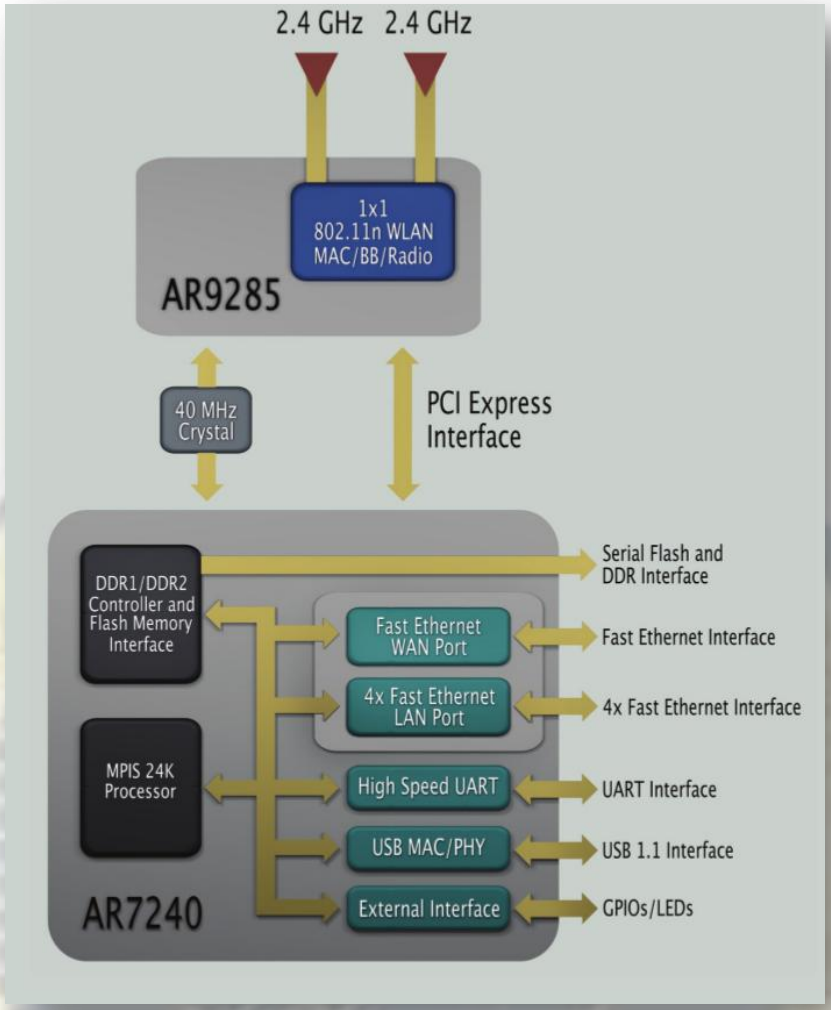


Ilustración 8 SoC atheros AR7240.

Dentro del SoC se puede distinguir[2]:

- **NIC:** Interfaz física hacia el medio alámbrico. Se presentan dos: Una destinada a WAN y otra conectada a un SWITCH. El mismo es también interno del SoC y tiene capacidad del gestionar 16 VLAN y trunking.
- **Interfaz USB 1.**1.
- **MIPS:** Es la CPU propiamente dicha.

3.3.1 Proceso de arranque de un AP

1. Se energiza la fuente.
2. Se estabiliza el regulador de tensión.
3. El SoC busca el sector 0 de la ROM.
4. Se carga el Bootloader.
5. El Bootloader inicia los dispositivos Ethernet y la UART.

6. El Bootloader envía un log por medio de la UART y espera una interrupción.
7. De no encontrar una interrupción carga el Firmware que está guardado en la ROM.
8. El firmware inicia todos los dispositivos y servicios del AP.

En el AP de referencia el Bootloader es U-Boot 1.1.4.

En su bootlog (Ver Anexos Log de inicio de AP) se puede apreciar el proceso de arranque del AP con el firmware OpenWrt 14.09 ya grabado. U-boot opera entre las líneas 1 y 29. En la línea 13 se programa como entrada estándar a serial que es la UART. En las líneas 14 y 15 se configura como salida estándar y salida de errores también a la UART es a partir de aquí donde se puede interrumpir a U-Boot (Su utilidad se verá más adelante.) una vez terminado el tiempo de interrupción U-Boot reinicia el AP ordenándole que cargue el sector 9f020000 que es donde está alojado el firmware. A partir de aquí es el Firmware quien tiene el control del dispositivo.

La necesidad de repasar estos conceptos es que para el proyecto es menester cambiar el firmware de los dispositivos. Los firmwares originales rara vez cumplen con los requerimientos. Y es aquí donde U-Boot toma gran importancia ya que da la posibilidad de escribir un firmware personalizado en el sector asignado en la ROM para el firmware orinal.

En el caso que el firmware que se escribió, este corrupto el Bootloader seguirá funcionando lo que nos dará la posibilidad de cargar otro firmware en lugar de dejar obsoleto el dispositivo.

3.3.2 Compilar el firmware OpenWrt

No se entrará en demasiado detalle en la compilación ya que no es el objetivo de esta tesis. Pero si se darán los parámetros que requiere firmware creado en este trabajo.

La toolchain o herramienta de compilación es Buildroot. Es una herramienta provista por la comunidad responsable del desarrollo de OpenWrt. Es muy similar a la compilación de un kernel Linux. Dicha toolchain corre sobre sistemas GNU/Linux en la estación de desarrollo. La comunidad desaconseja su utilización en ambientes Windows Y MacOSX.

Buildroot es un conjunto de Makefiles y parches que permite la generación de una herramienta de compilación cruzada y un binario del sistema raíz para un enrutador inalámbrico.

Para la compilación cruzada utiliza uClibc, una pequeña biblioteca estándar C. esto es necesario ya que los microprocesadores de la estación de desarrollo tiene una arquitectura X86 por lo general y los de los AP suelen tener arquitecturas MIPS, no tienen los set de instrucciones assembler compatibles.

El conjunto de herramientas de compilación que permiten la generación del binario se compone de: “gcc” como compilador, binutils como ensamblador y enlazador, y la biblioteca de estándar C.

Es posible configurar y compilar todo OpenWrt a mano no obstante Builroot contiene todos los parámetros y parches necesarios para la compilación cruzada. Lo que nos facilita y agiliza la compilación de un firmware.

3.3.2.1 La obtención de OpenWrt Buildroot

OpenWrt Buildroot está disponible a través de SVN. Para cualquier tipo de desarrollo OpenWrt usted debe conseguir la última versión de svn a través de:

$ Svn co https://svn.openwrt.org/openwrt/

3.3.2.2 Configurar una imagen

Mientras que el entorno de compilación de OpenWrt, Builroot, fue pensado principalmente para los desarrolladores, también es lo suficientemente simple que un usuario final sin experiencia puede construir fácilmente su propio firmware personalizado.

La ejecución del comando *"make menuconfig"* hará aparecer la pantalla del menú de configuración del OpenWrt, a través de este menú se puede seleccionar la plataforma que se está dirigiendo el firmware, qué versiones de la cadena de herramientas desea utilizar para construir y qué paquetes desea instalar en la imagen del firmware. OpenWrt contiene gran cantidad de paquetes. De todas maneras, se recomienda incluir sólo los paquetes de utilidad sino incrementará el tamaño del firmware y reduce su performance, el ancho de banda y la cantidad de clientes que se pueden gestionar.

Al igual que en la configuración del núcleo Linux, casi todas las opciones tienen tres alternativas, y/m/n que se representan de la siguiente manera:

- <*> (presionando y) Esto será incluido en la imagen de firmware.
- <M> (presionando m) Esto será compilado pero no incluido.
- <> (presionando n) Esto no será compilado.

3.3.2.3 Paquetes instalados

Arptables, ebtables, firewall, iptables: Es un framework disponible en el núcleo Linux que permite interceptar y manipular paquetes de red. Dicho framework permite realizar el manejo de paquetes en diferentes estados del procesamiento.

Busybox: Es un programa que combina muchas utilidades estándares de Unix en un solo ejecutable pequeño. Es capaz de proveer la mayoría de las utilidades que están especificadas para los sistemas Unix, además, de muchas de las utilidades que suelen verse en sistemas GNU/Linux.

Dropbear: Es un cliente/servidor ssh liviano. Da la posibilidad de interactuar con el firmware.

Iproute2: es un paquete que reemplaza al obsoleto net-tools. Con él se puede configurar la interfaz de red.

Iw / iwinfo: Utilidad para manipular la interface inalámbrica.

Una vez terminado con la configuración, es necesario guardar los cambios.

3.3.2.4 *Compilar una imagen*

Para comenzar a compilar el firmware, se escribe el comando *"make"* dentro del directorio que creo SVN. Por defecto OpenWrt sólo mostrará un resumen de alto nivel del proceso de compilación y no cada comando individual.

Cuando haya terminado, el firmware resultante estará en el directorio *"./bin"* y los paquetes en el directorio *"./bin/packages"* listo para ser grabado en el AP.

En el anexo se presenta el listado de los paquetes instalados en el firmware, explicando sólo los más importantes.

3.3.3 Firmar el Firmware.

Los AP comerciales con la finalidad que no se modifique su firmware introducen un chequeo de seguridad en el U-boot. Si el binario no está firmado U-boot propietario no descomprime el kernel. Evitando así que funcione el AP modificado.

Esto introduce a dos alternativas. Cargar un U-boot libre o firmar el firmware. Si la carga del gestor de arranque falla, convertirá en irrecuperable por los métodos aquí mencionados el AP. Además, el gestor de arranque sólo afecta el arranque del AP no es menester en todos los demás procesos. Es por ello que la opción electa es proceder a firmar el firmware.

Se emplea la herramienta TP 路由固件口修改工具.exe

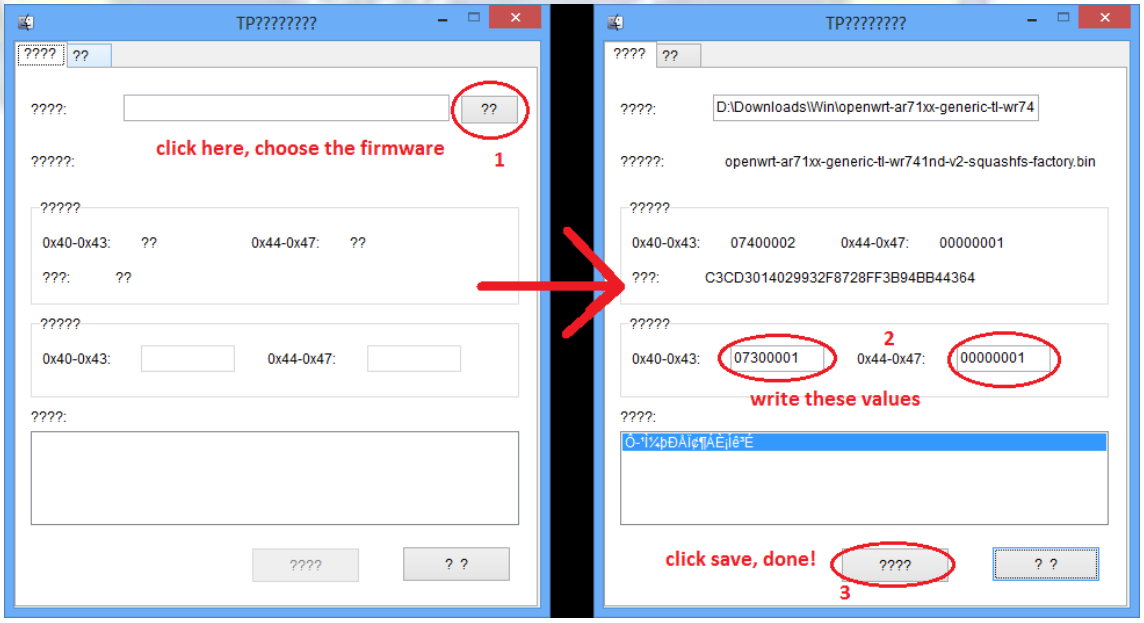


Ilustración 9 Firmador de Firmware.

3.3.4 Cargar un firmware personalizado.

Teniendo el binario ya compilado (el firmware) y firmado se procede a cargarlos en la ROM del AP. Si bien hay varias formas de hacer este procedimiento aquí se describe la forma de más bajo nivel o por decirlo de otra manera, la que menos requerimientos tiene. Con esta técnica no es necesario siquiera tener un firmware cargado y operativo en el AP por lo que también es útil en el proceso de recuperación.

En la ilustración 8 se describe un diagrama de conexiones para ayudar a visualizar globalmente el proceso.

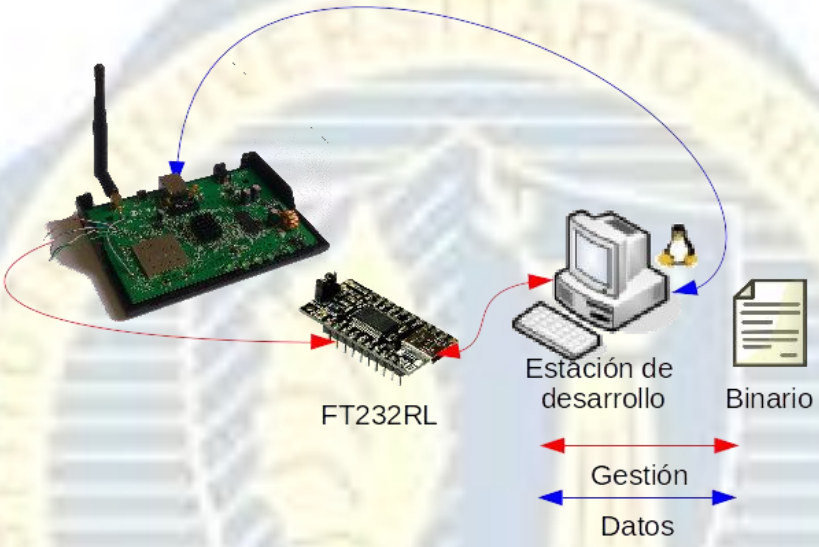


Ilustración 10 Plataforma de desarrollo y carga de firmware.

Pre requisitos: Es necesario tener el binario, un terminal y una interfaz de red.

La línea de gestión es una interface serial UART que conecta la terminal del AP a la PC de desarrollo. Se utiliza para dar instrucciones al AP y ver sus mensajes.

La línea de datos es Ethernet y se utiliza para transferir el binario. Si bien el binario se lo puede mandar por la línea de gestión su velocidad es mucho menor y por el tamaño del mismo ralentizaría la carga.

3.3.4.1 Interrumpir el Bootloader

El tiempo que opera el Bootloader está entre uno y dos segundos. De modo que es tiempo para interrumpirlo es corto.

Ya teniendo conectada la terminal, por medio de la UART, se energiza el dispositivo. Acto seguido se envía el código *"tpl"*. El Bootloader paraliza el proceso de arranque y muestra una consola para operar.

1. *U-Boot 1.1.4 (Sep 21 2010 - 13:23:41)*
2.
3. *AP91 (ar7240) U-boot*

4. *DRAM:*
5. *sri*
6. *#### TAP VALUE 1 = 9, 2 = 9*
7. *32 MB*
8. *id read 0x100000ff*
9. *flash size 4194304, sector count = 64*
10. *Flash: 4 MB*
11. *Using default environment*
12.
13. *In: serial*
14. *Out: serial*
15. *Err: serial*
16. *Net: ag7240_enet_initialize...*
17. *No valid address in Flash. Using fixed address*
18. *: cfg1 0xf cfg2 0x7014*
19. *eth0: 00:03:7f:09:0b:ad*
20. *eth0 up*
21. *No valid address in Flash. Using fixed address*
22. *: cfg1 0xf cfg2 0x7214*
23. *eth1: 00:03:7f:09:0b:ad*
24. *ATHRS26: resetting s26*
25. *ATHRS26: s26 reset done*
26. *eth1 up*
27. *eth0, eth1*
28. *Autobooting in 1 seconds*
29. *ar7240>*

En el bootlog muestra que la interrupción la toma justo antes de cargar el firmware. Después de haber inicializado el SoC, la RAM, el ROM, la entrada y salida estándar, y el Ethernet. Esto constituye un sistema con el que ya se puede interactuar.

3.3.4.2 Cargar TFTP en el ordenador

El servidor TFTP es quien entregará la imagen al U-BOOT.

1. *sudo apt-get install tftpd-hpa tftp*
2. *sudo cp Imagen_a_grabar.img /var/lib/tftpboot*

Si se quiere probar que opere correctamente el servidor TFTP se puede comprobar a través de la herramienta tftp haciendo una conexión al servidor local y descargando la imagen copiada al directorio de trabajo.

3. *tftp localhost*
4. *tftp> get Imagen_a_grabar.img*
5. *tftp> quit*
6. *cmp /var/lib/tftpboot Imagen_a_grabar.img Imagen_a_grabar.img*
7. *# no output other then a prompt means it worked correctly*

3.3.4.3 Situarlos en la misma red.

TFTP opera sobre la capa UDP/IP en el puerto 69. Por consiguiente es necesario que tanto el dispositivo como el terminal se puedan comunicar por red. Lo normal es que ambos estén en la misma sub red y conectado físicamente entre sí aunque esto no es un requisito.

Establecer la IP del terminal como 192.168.1.100 con el comando

8. *$ sudo ifconfig eth0 192.168.1.100/24*

Lo siguiente es fijar la IP del Bootloader. Conectados a la terminal UART

30. *ar7240> setenv ipaddr 192.168.1.123*
31. *ar7240> setenv serverip 192.168.1.100*

Para verificar la configuración se utiliza el comando "printenv"

32. *ar7240>printenv*
33. *bootargs=console=ttyS0,115200 root=31:02 rootfstype=jffs2 init=/sbin/init mtdpa)*
34. *bootcmd=bootm 0x9f020000*
35. *bootdelay=1*
36. *baudrate=115200*
37. *ethaddr=0x00:0xaa:0xbb:0xcc:0xdd:0xee*
38. *ipaddr=192.168.1.123*
39. *serverip=192.168.1.100*
40. *stdin=serial*
41. *stdout=serial*
42. *stderr=serial*
43. *ethact=eth1*

3.3.4.4 Cargar el binario a la memoria del dispositivo.

Se utiliza el comando "tftpboot"[7] para indicar que traiga desde el servidor tftp el binario y lo aloje en el sector de memoria 0x80000000.

44. *ar7240>tftpboot 0x80000000 OpenWrt-ar71xx-generic-tl-wa701n-v1-jffs2-factory.bin*
45. *Using eth1 device*
46. *TFTP from server 192.168.1.100; our IP address is 192.168.1.123*
47. *Filename 'OpenWrt-ar71xx-generic-tl-wa701n-v1-jffs2-factory.bin'.*
48. *Load address: 0x80000000*
49. *Retry count exceeded; starting again*
50. *eth0 link down*
51. *Using eth1 device*
52. *TFTP from server 192.168.1.100; our IP address is 192.168.1.123*
53. *Filename 'OpenWrt-ar71xx-generic-tl-wa701n-v1-jffs2-factory.bin'.*
54. *Load address: 0x80000000*
55. *Loading: ###*
56. *done*

```
57. Bytes transferred = 3932160 (3c0000 hex)
```

Hay que prestar especial atención al tamaño del archivo transferido ya que brinda el desplazamiento al momento de grabarlo en la ROM.

3.3.4.5 Borrar la memoria

Es necesario para evitar corrupción de datos limpiar la memoria ROM en los sectores que se alojará el nuevo firmware. Se utiliza el comando "erase" para señalar el primer sector a borrar y la cantidad de sectores que borrar.

```
58. ar7240> erase 0x9f020000 +0x3c0000
59.
60. First 0x2 last 0x3d sector size 0x10000
61.  61
62. Erased 60 sectors
```

Es preciso destacar que los sectores están expresados en valores hexadecimales. El primer sector se elige respetando el origen del firmware original con el fin de no pisar a U-Boot. Esta información se extrae desde "printenv".

3.3.4.6 Copiar el binario en la ROM

Se utiliza el comando "cp.b" con los parámetros sector origen, sector destino y tamaño del binario.

```
63. ar7240> cp.b 0x80000000 0x9f020000 0x3c0000
64. Copy to Flash... write addr: 9f020000
65. Done
```

Demás está decir que los valores están en hexadecimal.

3.3.4.7 Iniciar el nuevo firmware

Para iniciar el firmware desde el Bootloader se utiliza la herramienta "bootm" que solicita la dirección de memoria que contiene la imagen.

```
66. bootm 0x9f020000
```

Cabe aclarar que este es el comando que ejecuta U-Boot después de esperar el tiempo de interrupción. De esta forma cada vez que el AP se reinicie cargará el firmware modificado sin problemas.

En el anexo se presenta un listado completo de las funciones DAS U-Boot.

3.4 El Conmutador o Switch

Un conmutador o switch es un dispositivo digital lógico de interconexión de equipos. Su función es interconectar dos o más segmentos de red pasando datos de un segmento a otro de acuerdo con la dirección MAC de destino de las tramas en la red.

La función que cumple en la plataforma de pruebas es sólo apreciable en el sistema VLAN ya que va a ser el encargado de separar los paquetes en las distintas redes en función de cómo los enmascare el AP. Para las demás configuraciones no es necesario. Con el fin de que los retardos propios de la red no cambien se lo mantiene conectado en todas las pruebas.

Recordando lo mencionado en el apartado 2.1.2 El sistema VLAN está regido por el estándar 802.1Q. Consiste en subdividir un switch físico en dos o más switchs lógicos. Existen tres niveles para de VLAN.

- VLAN de nivel 1 (por puerto). Se especifica qué puertos del switch pertenecen a la VLAN, los miembros de dicha VLAN son los que se conecten a esos puertos.
- VLAN de nivel 2 por direcciones MAC. Se asignan hosts a una VLAN en función de su dirección MAC.
- VLAN de nivel 2 por tipo de protocolo. La VLAN queda determinada por el contenido del campo tipo de protocolo de la trama MAC.
- VLAN de nivel 3 por direcciones de subred o IP.

El caso de estudio sólo se centra en las VLAN de nivel 1 o por puerto.

Dicho switch virtual puede estar distribuido en la red. Esto se realiza por medio de un puerto trunk. Este puerto pertenece a más de una VLAN y cada paquete que transmite es etiquetado con un TAG correspondiente a la VLAN que pertenece.

De este modo se pueden generar dos VLAN una para usuario públicos y otra para los privados. Quedando su tráfico acotado a su switch virtual.

El modelo que se utiliza para las pruebas es Micronet SP1684B.

3.4.1 Especificaciones técnicas de SP1684B

Cuenta con soporte de 256 VLANs, SSH, 4 MB Buffer de paquetes, bus interno de 48Gbps, 8 MB Flash RAM[15] [16].

Ilustración 11 Switch SP1648b

3.5 El módulo UART

Su función es la de gestionar comunicaciones UART que son de corto alcance. No tienen modulación de línea en su lugar se utiliza banda base.

En el entorno de desarrollo se utiliza para acceder a la terminal de comando del AP. De este modo se tiene un control más preciso del mismo.

El módulo que se utilizó es el FT232RL

Ilustración 12 modulo USB-UART

3.5.1 Especificaciones técnicas de FT232RL [16]:

- Un solo chip USB para datos en serie asíncrono
- Interfaz de transferencia.
- Todo el protocolo USB manejado en el chip. No requiere programación firmware específico USB.
- Almacenamiento EEPROM totalmente integrado 1024 bits.
- Descriptores de dispositivo y configuración CBUS I/O.
- Resistencias terminales USB totalmente integrado.
- Generación de reloj totalmente integrado sin cristal externo requerido.
- Tasas de transferencia de datos de 300 baudios 3 MBaudios.
- RS422, RS485, RS232 a niveles TTL.
- 128 byte búfer de recepción y de transmisión de 256 bytes.
- Accionamiento de LED al transmitir y recibir señales.
- Soporte de interfaz UART para 7 u 8 bits de datos, 12 bits de parada y par/ impar/ marca/ espacio/sin paridad FIFO de recepción y transmisión de datos de alta tampones rendimiento.
- Número de serie USB.
- Soporta autobús impulsado, autoalimentado y alta potencia.
- Bus powered configuraciones USB.
- Integrado + 3.3V convertidor de nivel para USB de E/S.
- Convertidor de nivel integrado en el UART y CBUS.
- Para la interconexión a entre + 1.8V y + 5V la lógica.
- Verdadero/3.3V / 2.8V / salida de accionamiento 1.8V CMOS 5V y entrada TTL.
- Circuito integrado power-on- reset.
- Señal UART opción de inversión.
- + 3.3V (usando oscilador externo) a + 5.25V
- Oscilador interno de alimentación única operación.
- Bajo operativo y USB suspender actual.
- Bajo consumo de ancho de banda USB.

- UHCI / OHCI / EHCI controlador compatible anfitrión.
- USB compatible 2.0 Full Speed.
- -40 ° C a 85 ° C de temperatura de funcionamiento extendido

4 Desarrollo del Trabajo

4.1 Resumen Técnico

Se presentan dos diagramas en bloques: el primero muestra el esquema general del sistema implementando VLANs y el segundo del sistema implementado IPTABLES.

La principal diferencia entre los sistemas es que VLAN encamina y encapsula los paquetes al destino correcto e IPTABLES los filtra sino son para destinos preestablecidos. Será importante ver el impacto que tiene el hardware de cada una de estas técnicas dado que en los APs comerciales los recursos son muy limitados, en el orden de los 32Mb de RAM y 4Mb ROM.

4.1.1 Sistema VLAN

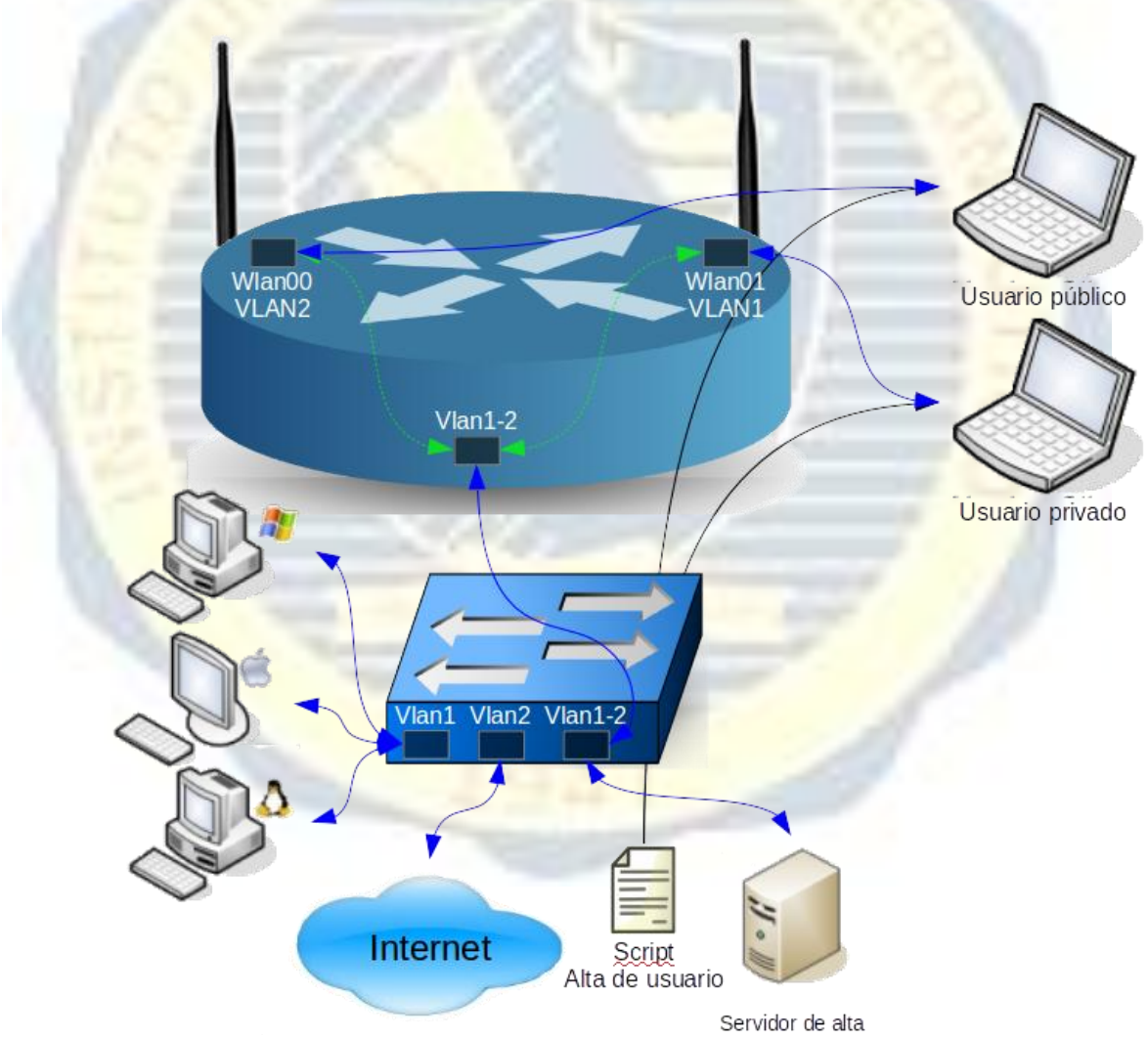


Ilustración 13 Sistema VLAN.

Un conmutador que soporta el protocolo 802.1Q es posible definir diferentes VLANs. Cada VLAN está identificada por un número entero comprendido entre 1 y 4095, además, cada

VLAN podrá tener una cadena descriptiva. Los identificadores numéricos de las VLANs tienen validez en toda la red de conmutadores.

Cuando los puertos sólo se encuentran en una VLAN el conmutador no necesita modificar las tramas, pues nunca hay ambigüedad. Cuando en un puerto no se modifican las tramas se dice que es un puerto sin etiquetar (untagged).

Cuando un enlace transporta tráfico perteneciente a varias VLANs los dos puertos en sus extremos deben estar en modo trunk. En este caso es necesario modificar todas las tramas que transporta el enlace, agregando una etiqueta, para poder identificar la VLAN a la que pertenece la trama. Cuando un puerto pertenece a varias VLANs se dice que es un puerto etiquetado (tagged).

El estándar 802.1Q define etiquetas de 4 bytes que se añaden a la cabecera de las tramas que se transmiten por un puerto etiquetado. De esta forma, cuando llega al conmutador de destino este sabe por qué puertos la debe propagar. Cuando se propague por un puerto no etiquetado se eliminará la etiqueta de 4 bytes y la trama recuperará su formato original.

Normalmente las estaciones de red están conectadas a puertos no etiquetados del conmutador, sólo se utilizan los puertos etiquetados para los enlaces trunk entre dos conmutadores.

Resumiendo:

- Un puerto no etiquetado sólo puede pertenecer a una VLAN, no existe ambigüedad sobre el tráfico que se envía/recibe.
- Un puerto que pertenece a varias VLANs es un puerto etiquetado. En este caso es necesario añadir una etiqueta de 4 bytes a cada trama.
- En un puerto no etiquetado se puede conectar cualquier host.
- En un puerto etiquetado se debe conectar un host con soporte para VLANs.

El prototipo contiene dos Interfaces virtuales cada una perteneciente a una VLAN. Cada interface tiene su propio BSSIDs. El puerto Ethernet será un puerto etiquetado. Esto obliga a que el switch también soporte el estándar 802.1Q. Este switch se encargará de separar los paquetes provenientes de cada VLAN en las LAN correspondientes. Así se estará hablando de red privada y red pública.

4.1.1.1 Diagrama de configuración para el AP TL730RE

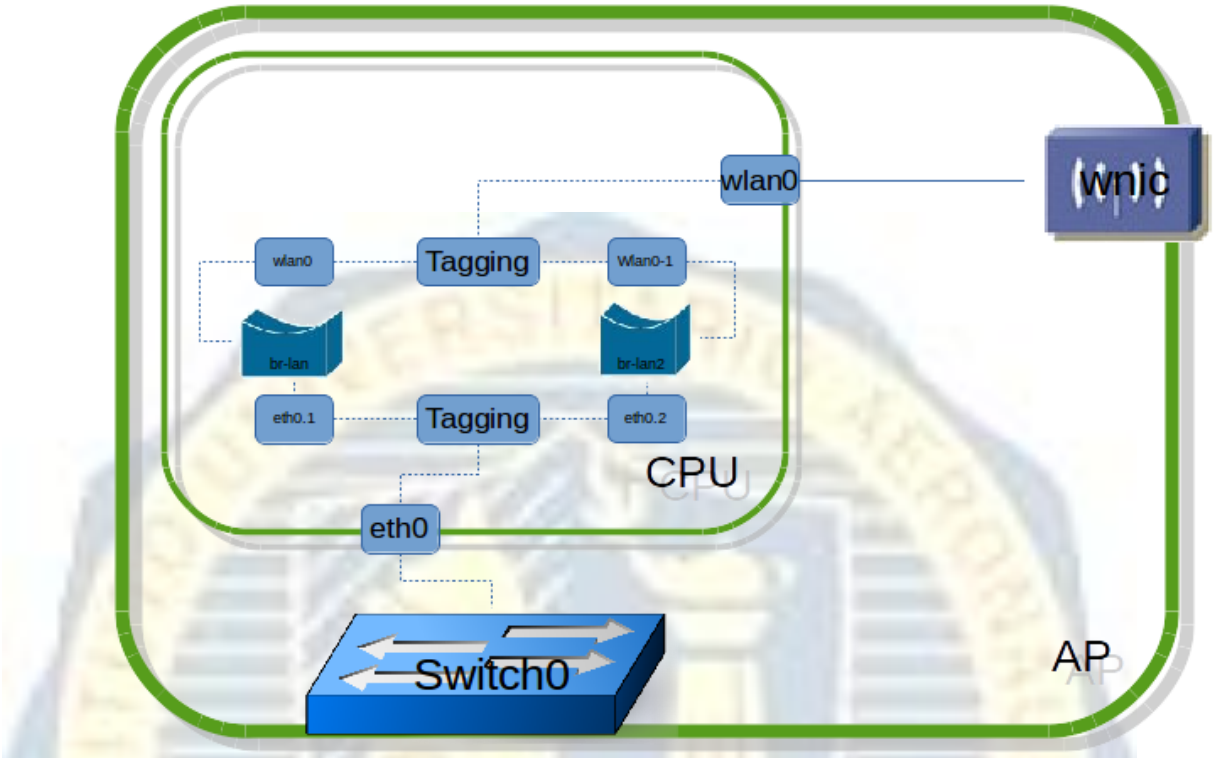


Ilustración 14 diagrama en bloque interno del AP.

En la ilustración 14 se describen los bloques físicos y lógicos por los que transita un paquete IP dentro del AP. Los bloques lógicos son generados por el kernel Linux e insumen tiempo de la CPU, mientras que los bloques físicos procesan los paquetes sin demandar recursos de la CPU.

Switch0 es un switch programable, es un bloque físico. Cuenta con cinco puertos. El puerto 0 está conectado a la NIC eth0 y el puerto 1 a la red externa. Ambos puertos están en modo trunking para dejar pasar los paquetes etiquetados por la CPU.

NIC eth0 es un puerto Ethernet, una NIC conectada al BUS PCI de la CPU.

La CPU tiene programados dos brigde o puentes, br-lan y br-lan2. A br-lan se vinculó eth0.1 y wlan0. Por otra parte al br-lan2 se vinculó eth0.2 y wlan0-1. Los dispositivos anteriormente mencionados son lógicos.

Cada una de las wlan tienen su propio bssid y essid de modo que los clientes ven la red como dos AP diferentes. Todo el tráfico queda lógicamente dividido sin posibilidad de saltarse la barrera.

4.1.1.2 Archivos de configuración para el AP TL730RE

4.1.1.2.1 /etc/config/network

```
config interface 'loopback'
        option ifname 'lo'
        option proto 'static'
        option ipaddr '127.0.0.1'
        option netmask '255.0.0.0'

config globals 'globals'
        option ula_prefix 'fd4b:b166:008c::/48'

config 'switch' 'eth0'
        option 'reset' '1'
        option 'enable_vlan' '1'

config 'switch_vlan'
        option 'vlan'       '1'
        option 'device'     'eth0'
        option 'ports'      '0t 1t'

config 'switch_vlan'
        option 'vlan'       '2'
        option 'device'     'eth0'
        option 'ports'      '0t 1t'


config 'switch_vlan'
        option 'vlan'       '0'
        option 'device'     'eth0'
        option 'ports'      '2 3 4'

config interface 'lan'
        option ifname 'eth0.1'
        option force_link '1'
        option type 'bridge'
        option proto 'static'
        option ipaddr '192.168.1.253'
        option netmask '255.255.255.0'
        option ip6assign '60'

config interface 'lan2'
        option ifname 'eth0.2'
        option force_link '1'
        option type 'bridge'
        option proto 'static'
        option ipaddr '192.168.1.252'
        option netmask '255.255.255.0'
        option ip6assign '60'
```

4.1.1.2.2 /etc/config/wireless

```
config wifi-device 'radio0'
        option type 'mac80211'
        option channel '4'
        option hwmode '11g'
        option path 'platform/bcm2708_usb/usb1/1-1/1-1.3/1-1.3:1.0'
        option noscan '1'
        option htmode 'HT40'
        option country 'AR'
        option rts '500'
        option distance '10'

config wifi-iface
        option device 'radio0'
        option network 'lan'
        option mode 'ap'
        option ssid 'OpenWrt'
        option encryption 'psk'
        option key 'qwerty123'

config wifi-iface
        option device 'radio0'
        option network 'lan2'
        option mode 'ap'
        option ssid 'OpenWrt2'
        option encryption 'psk'
        option key 'qwerty123'
```

4.1.1.2.3 /etc/config/firewall

```
config defaults
        option syn_flood       1
        option input           ACCEPT
        option output          ACCEPT
        option forward         REJECT
        option disable_ipv6    1

config zone
        option name            lan
        list   network         'lan'
        option input           ACCEPT
        option output          ACCEPT
        option forward         ACCEPT
```

4.1.2 Sistema IPTABLES

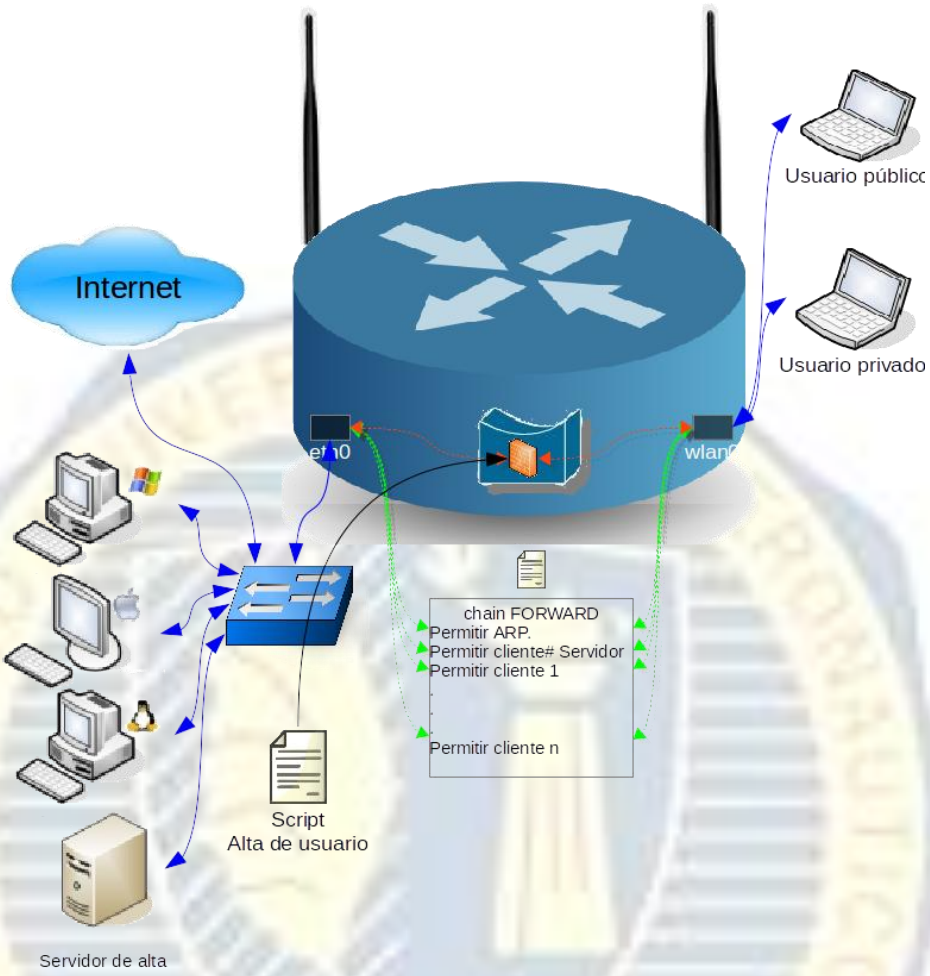


Ilustración 15 Sistema IPTABLES.

Este sistema implementa un firewall que filtra los paquetes sino coinciden con un origen destino correcto. Esto se realiza a través de reglas dentro de la cadena chain FORWARD para cada cliente.

La chain FORWARD procesa todos los paquetes que recibe el kernel y no están destinados a la máquina local. La misma se le dio la política de descartar todos los paquetes. De este modo el AP actúa como un contrafuego puro y duro. No deja pasar ningún tipo de tráfico, por lo que se crearon reglas particulares para permitir sólo algunos paquetes.

Un cliente con premisos se le crean reglas de entrada simple, sólo se comprueba si el paquete viene de ese cliente o si va a ese cliente.

Por el contrario un cliente sin permisos se le crean reglas dobles, se comprueba si el paquete es el cliente y va a la puerta de enlace o si el paquete es de la puerta de enlace y va al cliente. Esto bloquea todo el tráfico exceptuando el de salida de internet, con lo que no es capaz de ver el resto de la red.

4.1.2.1 Comandos para crear una pasarela segura a través del firewall por mac

ebtables -t filter -P FORWARD DROP

ebtables -t filter -A FORWARD -p ARP -j ACCEPT

ebtables -t filter -A FORWARD -s f4:6d:4:45:c7:1d -d f8:d1:11:26:33:a3 -j ACCEPT

ebtables -t filter -A FORWARD -s f8:d1:11:26:33:a3 -d f4:6d:4:45:c7:1d -j ACCEPT

4.1.2.2 Pseudocódigo

La chain FORWARD toma como política por defecto descartar todos los paquetes.

La chain FORWARD acepta todo los paquetes ARP

La chain FORWARD acepta todos los paquetes que tengan como origen PC1 y destino servidor

La chain FORWARD acepta todos los paquetes que tenga como origen servidor y destino PC1

4.2 Metodología

Se seleccionó la metodología espiral debido a que la misma se fundamenta en la evaluación de reducción del riesgo del proyecto dividiendo el proyecto en pequeños fragmentos y brindando la posibilidad cambio durante el proceso de desarrollo.

El método de investigación que se utilizó en este proyecto es un análisis teórico para luego hacer una verificación empírica.

Como metodología de desarrollo para la realización del trabajo de grado se realizaron las siguientes actividades:

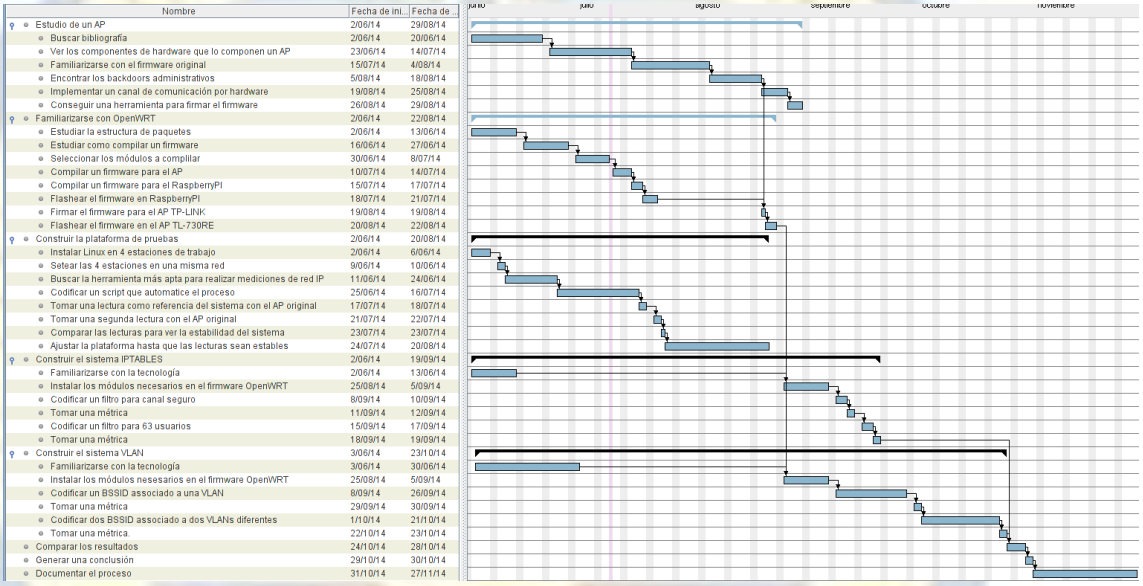
- Se entregaron informes parciales del trabajo para la constante revisión por parte del tutor y así supervisar el avance del mismo.
- Mediante el diagrama de Gantt propuesto se ajustó al calendario, respetando los tiempos y recursos planteados en cada etapa para poder lograr el mismo con éxito.

4.3 Actividades realizadas

- Se estudió los dos sistemas propuestos y las tecnologías intervinientes.
- Se rediagramó los sistemas en base al estudio previamente realizado.
- Se descartó implementar la plataforma de pruebas virtual con el firmware OpenWrt.
- Se instaló el firmware OpenWrt en la RaspberryPI.
- Se instaló el firmware OpenWrt en el AP TL730RE.
- Se implementó la plataforma de pruebas con el AP TL730RE con su firmware original como nodo central.
- Se extrajeron datos del ensayo referente a los requerimientos de hardware y la performance.
- Se reemplazó el AP por RaspberryPI.

- Se extrajeron datos del ensayo referente a los requerimientos de hardware y la performance.
- Se implementó VLAN en el AP TL730RE con el firmware OpenWrt para el prototipo.
- Se extrajeron datos del ensayo referente a los requerimientos de hardware y la performance.
- Implementar IPTABLES en el AP TL730RE con el firmware OpenWrt para el prototipo.
- Se extrajeron datos del ensayo referente a los requerimientos de hardware y la performance.
- Se seleccionó VLAN sobre IPTABLES.
- Se implementó VLAN con dos Estaciones y Dos segmentos de red.
- Se Probó VLAN en conjunto viendo la interferencia en los datos.

4.3.1 Diagrama de Gantt real



Nombre	Fecha de ini...	Fecha de ...
Estudio de un AP	2/06/14	29/08/14
Buscar bibliografía	2/06/14	20/06/14
Ver los componentes de hardware que lo componen un AP	23/06/14	14/07/14
Familiarizarse con el firmware original	15/07/14	4/08/14
Encontrar los backdoors administrativos	5/08/14	18/08/14
Implementar un canal de comunicación por hardware	19/08/14	25/08/14
Conseguir una herramienta para firmar el firmware	26/08/14	29/08/14
Familiarizarse con OpenWRT	2/06/14	22/08/14
Estudiar la estructura de paquetes	2/06/14	13/06/14
Estudiar como compilar un firmware	16/06/14	27/06/14
Seleccionar los módulos a compilar	30/06/14	8/07/14
Compilar un firmware para el AP	10/07/14	14/07/14
Compilar un firmware para el RaspberryPI	15/07/14	17/07/14
Flashear el firmware en RaspberryPI	18/07/14	21/07/14
Firmar el firmware para el AP TP-LINK	19/08/14	19/08/14
Flashear el firmware en el AP TL-730RE	20/08/14	22/08/14
Construir la plataforma de pruebas	2/06/14	20/08/14
Instalar Linux en 4 estaciones de trabajo	2/06/14	6/06/14
Setear las 4 estaciones en una misma red	9/06/14	10/06/14
Buscar la herramienta más apta para realizar mediciones de red IP	11/06/14	24/06/14
Codificar un script que automatice el proceso	25/06/14	16/07/14
Tomar una lectura como referencia del sistema con el AP original	17/07/14	18/07/14
Tomar una segunda lectura con el AP original	21/07/14	22/07/14
Comparar las lecturas para ver la estabilidad del sistema	23/07/14	23/07/14
Ajustar la plataforma hasta que las lecturas sean estables	24/07/14	20/08/14
Construir el sistema IPTABLES	2/06/14	19/09/14
Familiarizarse con la tecnología	2/06/14	13/06/14
Instalar los módulos necesarios en el firmware OpenWRT	25/08/14	5/09/14
Codificar un filtro para canal seguro	8/09/14	10/09/14
Tomar una métrica	11/09/14	12/09/14
Codificar un filtro para 63 usuarios	15/09/14	17/09/14
Tomar una métrica	18/09/14	19/09/14
Construir el sistema VLAN	3/06/14	23/10/14
Familiarizarse con la tecnología	3/06/14	30/06/14
Instalar los módulos nesesarios en el firmware OpenWRT	25/08/14	5/09/14
Codificar un BSSID associado a una VLAN	8/09/14	26/09/14
Tomar una métrica	29/09/14	30/09/14
Codificar dos BSSID associado a dos VLANs diferentes	1/10/14	21/10/14
Tomar una métrica.	22/10/14	23/10/14
Comparar los resultados	24/10/14	28/10/14
Generar una conclusión	29/10/14	30/10/14
Documentar el proceso	31/10/14	27/11/14

Ilustración 16 Diagrama de gannt.

4.4 Materiales utilizados

- Dos PCs que actúan servidores.
- Una RaspberryPI.
- Un adaptador Wi-Fi por USB.
- Dos estaciones móviles.
- Switch administrable.
- Un desarrollador.

4.5 Control de costos

No aplica.

4.6 Dificultades que se han presentado

Dado el carácter de ingeniería inversa del proyecto configurar la plataforma de pruebas fue una tarea más laboriosa de lo que se previó. Muchos de los bloques son tratados como caja negra. Sólo se visualizan sus entradas y salidas.

El primer gran obstáculo se presentó al inyectar paquetes UDP con ipref a una tasa constante y cercana al punto de saturación. Un fallo en el controlador “e1000” del kernel Linux 3.14 responsable de la nic Gigabit Ethernet provocó que se deshabilitara la IRQ para esa interfaz y pasara a recoger los paquetes por un método de poling. Esto hizo disminuir el rendimiento de la interfaz a cerca de 5 Mbit/s. Una vez identificado el problema la solución fue sencilla, instalar un kernel más actual en que el bug estaba corregido.

El segundo problema relevante que se encontró fue el muy pobre rendimiento del RaspberryPI. Muy inferior a lo que se esperaba, transformándola en inútil para el análisis de los sistemas de filtrado. Esto provocó una completa re diagramación del plan de trabajo obligando a trabajar directamente sobre el prototipo final.

Por último se observó que ante la misma configuración en pruebas largas la performance mermaba respecto a las pruebas cortas. Lo que contradice la lógica que dicta que pruebas largas debieran ser más estables y con menos error que pruebas cortas. Lo que en realidad sucedía es que el servicio de escritorio Network-Manager de Ubuntu solicitaba a la interfaz wnic un escaneo periódico de las redes inalámbricas. La interfaz deja de responder paquetes durante el período de escaneo, provocando una acumulación de paquetes en el buffer del AP y una lectura errónea. Una vez más la solución fue sencilla ya que se deshabilitó el servicio durante todo el escaneo.

4.7 Resultados alcanzados

Se presentan los resultados obtenidos más relevantes al aplicar la plataforma de evaluación sobre cada una de las configuraciones del AP de forma sintética y resumida. Mostrando sólo los datos más relevantes en formato de gráfica que marcan una tendencia.

Los resultados completos se encuentran en el archivo adjunto “Log de RED.xlsx”.

4.7.1 Descripción de cada prueba.

En cada una de los ensayos realizados se respetaron los parámetros por defecto del firmware correspondiente exceptuando lo particularmente descripto.

4.7.1.1 TP-LINK 3

Los parámetros de configuración para esta prueba fueron:

- Seguridad: WPA
- Clear to send: off
- Firmware: Original

Esta fue la tercera prueba con el mismo firmware sobre el hardware del AP TP-LINK TL-730RE.

4.7.1.2 TP-LINK 4

Los parámetros de configuración para esta prueba fueron:

- Seguridad: WPA
- Clear to send: 500
- Firmware: Original

Esta fue la cuarta prueba con el firmware original y la que mejor performance mostro de todas las pruebas con el firmware original. Esta métrica y configuración del canal inalámbrico es la que se va a utilizar de aquí en adelante como línea base para comparar los posteriores resultados.

4.7.1.3 RaspberryPI 1

Los parámetros de configuración para esta prueba fueron:

- Seguridad:WPA
- Clear to send: 500
- Firmware: OpenWrt 14.07

Esta fue la primera prueba con el firmware OpenWrt. El canal inalámbrico se lo configuro con los mismos parámetros que la prueba “TP-LINK 4”. Esta métrica se realizó para comparar la performance del RaspberryPI contra la métrica “TP-LINK 4” a fin de determinar si la plataforma tiene capacidades suficientes para manejar el flujo de datos. Los resultados fueron muy por debajo de lo espera.

4.7.1.4 RaspberryPI 2

Los parámetros de configuración para esta prueba fueron:

- Seguridad: none
- Clear to send: 500
- Firmware: OpenWrt 14.07

La prueba fue un intento de mejorar la performance del RaspberryPI eliminando la seguridad WPA. Se supuso que pudo ser un problema del módulo de criptografía. No obstante los resultados empeoraron.

4.7.1.5 OpenWrt 1

Los parámetros de configuración para esta prueba fueron:

- Seguridad: WPA
- Clear to send: 500
- Firmware: OpenWrt 14.07

Esta fue la primera prueba con el firmware OpenWrt en el hardware de AP. El canal inalámbrico se lo configuro con los mismos parámetros que la prueba "TP-LINK 4". Esta métrica se realizó para comparar la performance del firmware OpenWrt 14.07 contra el firmware original. Dicha comparación revela lo optimizado que esta un OpenWrt.

4.7.1.6 OpenWrt 2

Los parámetros de configuración para esta prueba fueron:

- Seguridad: WPA
- Clear to send: 500
- Firmware: OpenWrt 14.07

Esta prueba es casi idéntica a OpenWrt 1 solo se cambió el parámetro "becon interval" de 100ms a 350ms.

4.7.1.7 OpenWrt 3

Los parámetros de configuración para esta prueba fueron:

- Seguridad: WPA
- Clear to send: 500
- Firmware: OpenWrt 14.07

Esta prueba es casi idéntica a OpenWrt 1 solo se desactivo el servidor web y otro servicios no esenciales para comprobar si afectaba el rendimiento.

4.7.1.8 OpenWrt 4

Los parámetros de configuración para esta prueba fueron:

- Seguridad: WPA
- Clear to send: 500
- Firmware: OpenWrt 14.07

En esta prueba se etiquetan los paquetes que viajan entre el AP y el Switch con vlan1 por medio del switch interno del AP. Se pretende verificar que el etiquetado por hardware del AP y el etiquetado del Switch no imponen variaciones en la capacidad del canal que afecten las métricas de manera apreciable.

Su final es para distinguir si el impacto en la performance se debe al etiquetado de los paquetes o si al manejo interno del AP.

4.7.1.9 OpenWrt 5

Los parámetros de configuración para esta prueba fueron:

- Seguridad: WPA
- Clear to send: 500
- Firmware: OpenWrt 14.07

La prueba se realizó en similares condiciones que en OpenWrt 4. La diferencia fue que se desactivaron los servicios no esenciales. El objetivo de la misma es para junto a otras determinar si afecta en distintas condiciones a la capacidad del canal los servicios en segundo plano que hay corriendo en el AP.

4.7.1.10 OpenWrt 6

Los parámetros de configuración para esta prueba fueron:

- Seguridad: WPA
- Clear to send: 500
- Firmware: OpenWrt 14.07

Se implementó el sistema VLAN descripto en el apartado 4.1.1 y se generó tráfico por el canal seguro. Esta métrica se realizó para verificar el impacto que tiene sobre el canal el aplicar el sistema de filtrado.

4.7.1.11 OpenWrt 7

Los parámetros de configuración para esta prueba fueron:

- Seguridad: WPA
- Clear to send: 500
- Firmware: OpenWrt 14.07

Se implementó el sistema IPTABLES descripto en el apartado 4.1.2 con un solo cliente autorizado en el canal seguro. Esta métrica se realizó para verificar el impacto que tiene sobre el canal el aplicar el sistema de filtrado.

4.7.1.12 OpenWrt 8

Los parámetros de configuración para esta prueba fueron:

- Seguridad: WPA
- Clear to send: 500
- Firmware: OpenWrt 14.07

Se implementó el sistema IPTABLES descripto en el apartado 4.1.2 con 63 cliente autorizado en el canal seguro. El cliente que genera tráfico es el número 63. Esta métrica se realizó para verificar el impacto que tiene sobre el canal el aplicar el sistema de filtrado cuando el número de clientes es elevado.

4.7.2 Resultados de los ensayos.

4.7.2.1 Interpretación de las gráficas.

En las Ilustraciones 17, 18 y 19 se agrupan los doce ensayos descriptos en los apartados 4.7.1.1 al 4.7.1.12. Con el fin de tomar una dimensión global de como varían las métricas en función de cada configuración de AP.

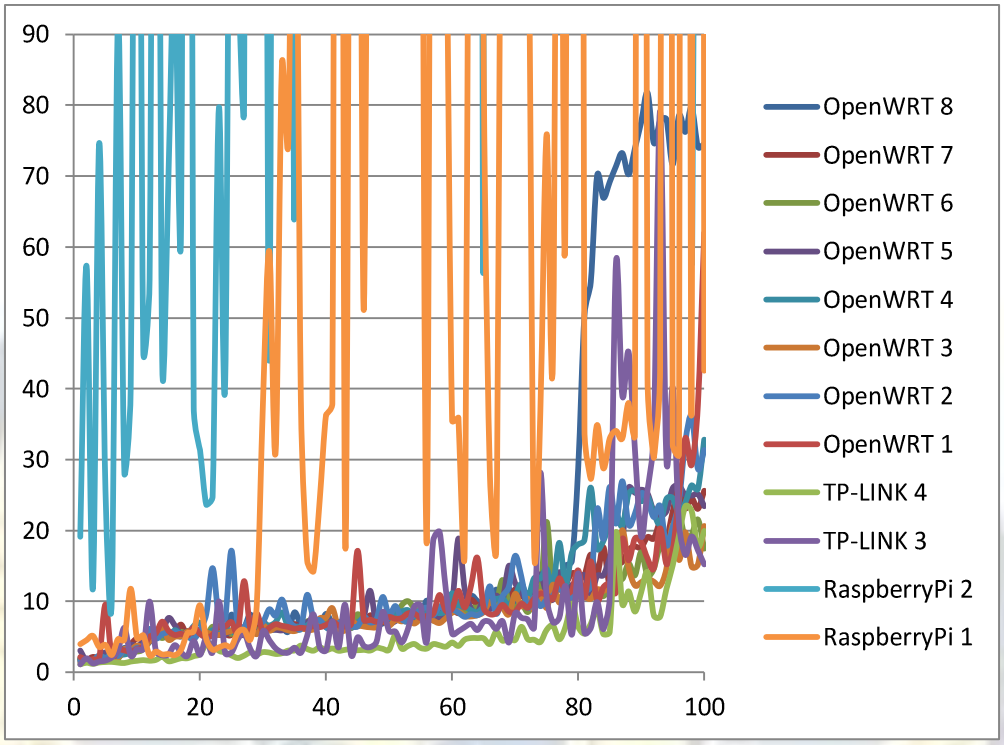


Ilustración 17 Ancho de banda transmitido [Mbit/s] vs retardo [mSeg].

La gráfica de ancho de banda transmitido contra retardo de la red revela a que tasa de transmisión la red se congestiona, si bien en la teoría cuando la red alcanza el punto de congestión el retado tiende al infinito, en la práctica no sucede así. El retardo se eleva abruptamente pero de un modo acotado.

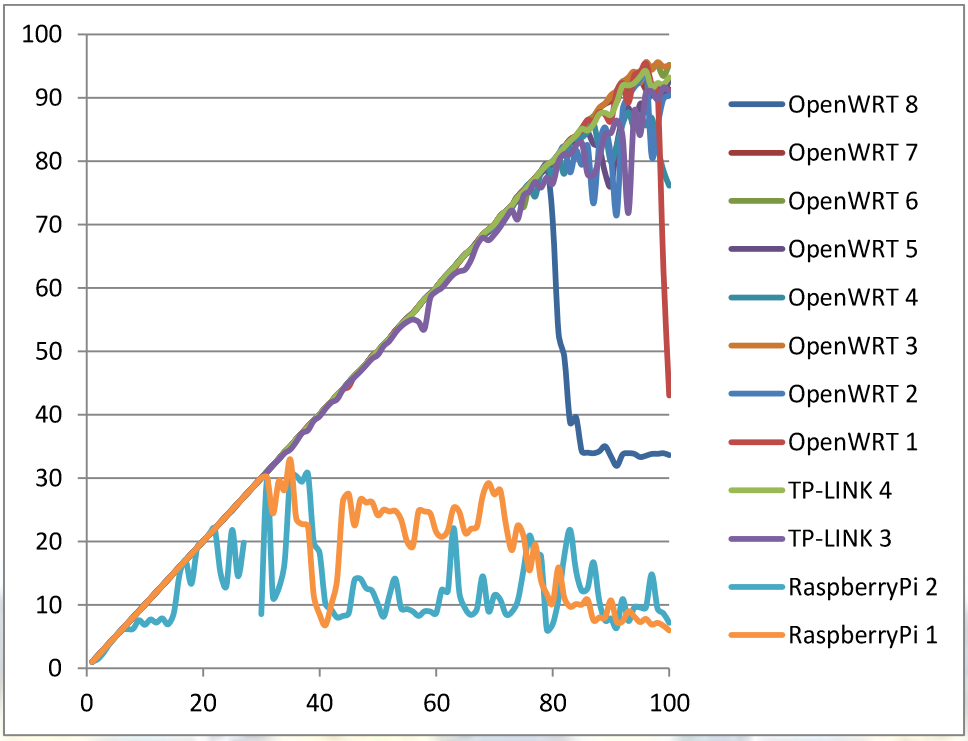


Ilustración 18 Ancho de banda transmitido [Mbit/s] vs Ancho de banda recibido [Mbit/s].

En la ilustración 18 y 19 se aprecia porque sucede esta diferencia entre la teoría y la realidad. La red decide descartar los paquetes que no puede manejar. Lo que se ve como una diferencia entre el ancho de banda transmitido y recibido.

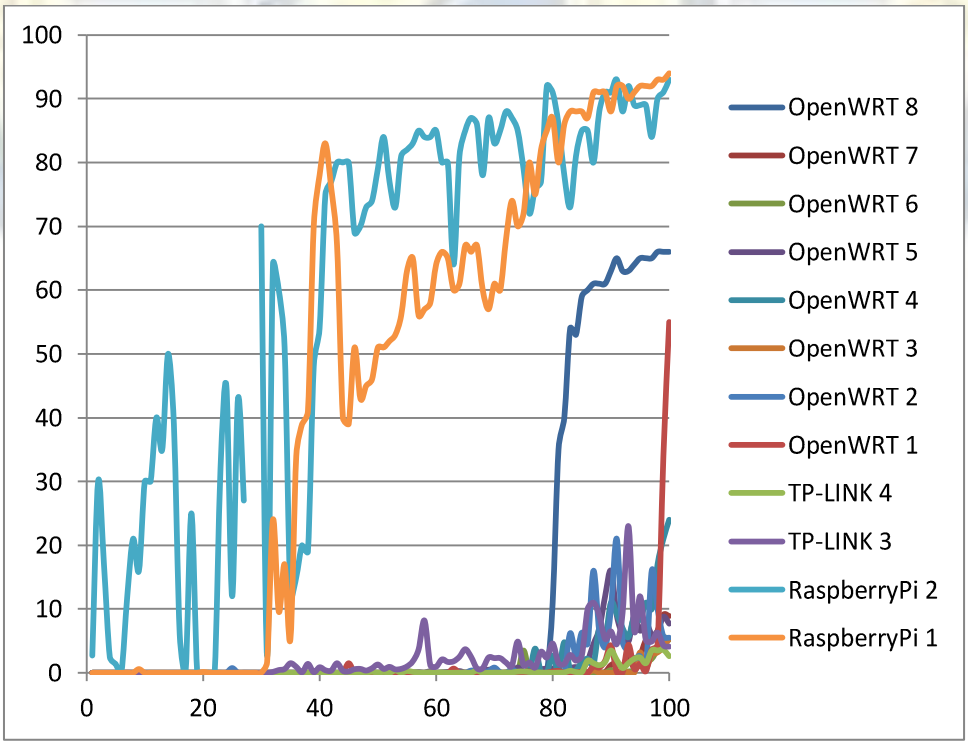


Ilustración 19 Ancho de banda transmitido [Mbit/s] vs Perdida de paquetes [%].

La gráfica ancho de banda transmitido contra paquetes perdidos es otra forma de ver la misma información que en la gráfica ancho de banda transmitido contra recibido. Ya que dos de las tres variables están vinculadas matemáticamente por la fórmula:

El ancho de banda transmitido es arbitrario por ende se puede expresar el ancho de banda recibido en función de la pérdida de paquetes.

4.7.2.2 Comparativa de sistemas de filtrado

A fin de poder visualizar mejor los ensayos más relevantes en las ilustraciones 20, 21 y 22 se agruparan los diagramas de congestión de las siguientes pruebas:

- TP-LINK 4: firmware original con cifrado WPA.
- OpenWRT6: sistema de filtrado con VLAN.
- OpenWRT7: sistemas de filtrado IPTABLES con un cliente.
- OpenWRT8: sistemas de filtrado IPTABLES 63 clientes.

La ilustración 15 muestra el creciente retardo estadístico de cada uno de los sistemas en función de la utilización del canal. Se aprecia claramente que el sistema OpenWRT8 se congestiona a partir de los 78 Mbit/s mientras que los otros sistemas no muestras signos evidentes.

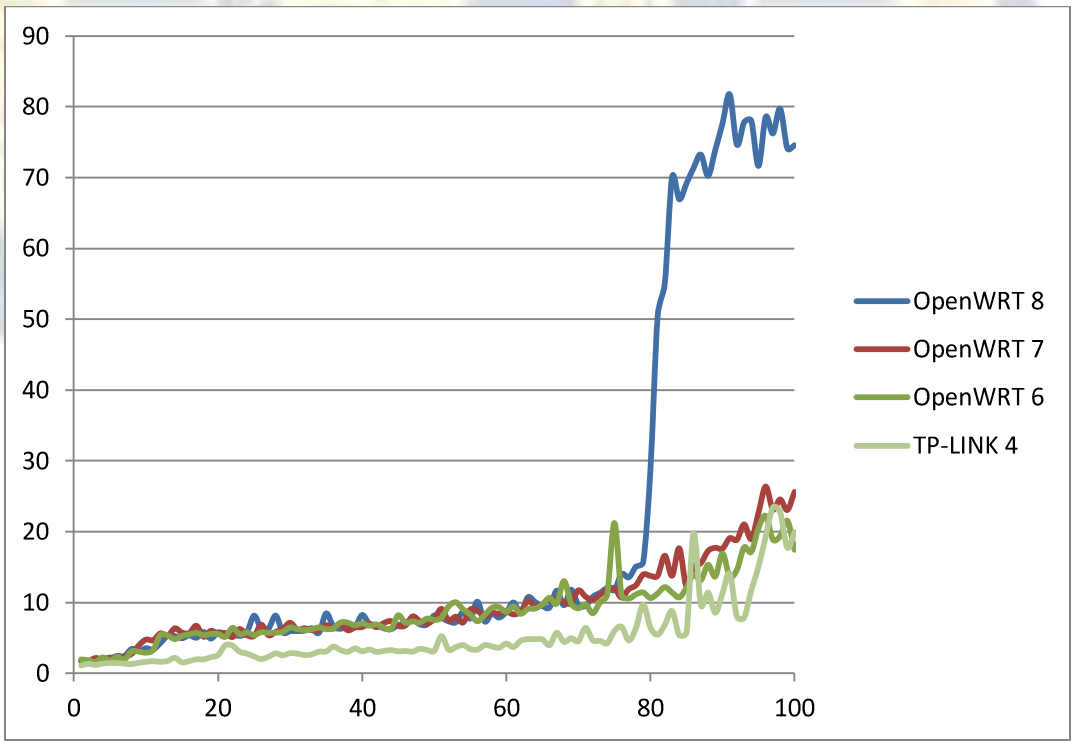


Ilustración 20 Ancho de banda transmitido [Mbit/s] vs retardo [mSeg]

Las ilustraciones 21 y 22 muestran la fiabilidad del canal a través de la pérdida de paquetes. En la realidad una gráfica es función de la otra.

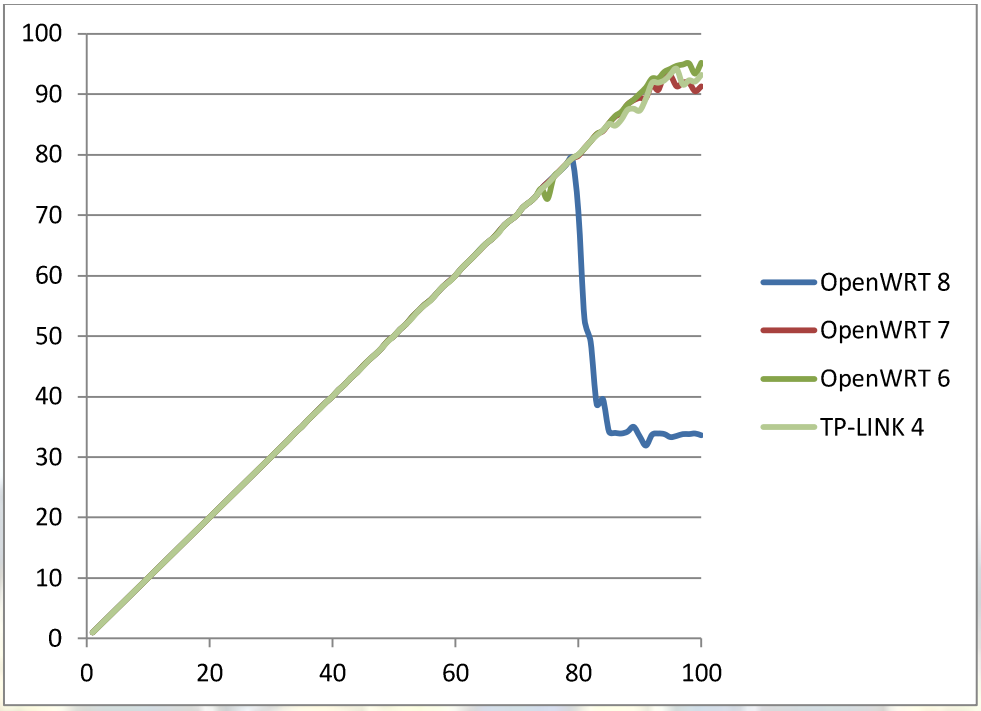


Ilustración 21 Ancho de banda transmitido [Mbit/s] vs Ancho de banda recibido [Mbit/s].

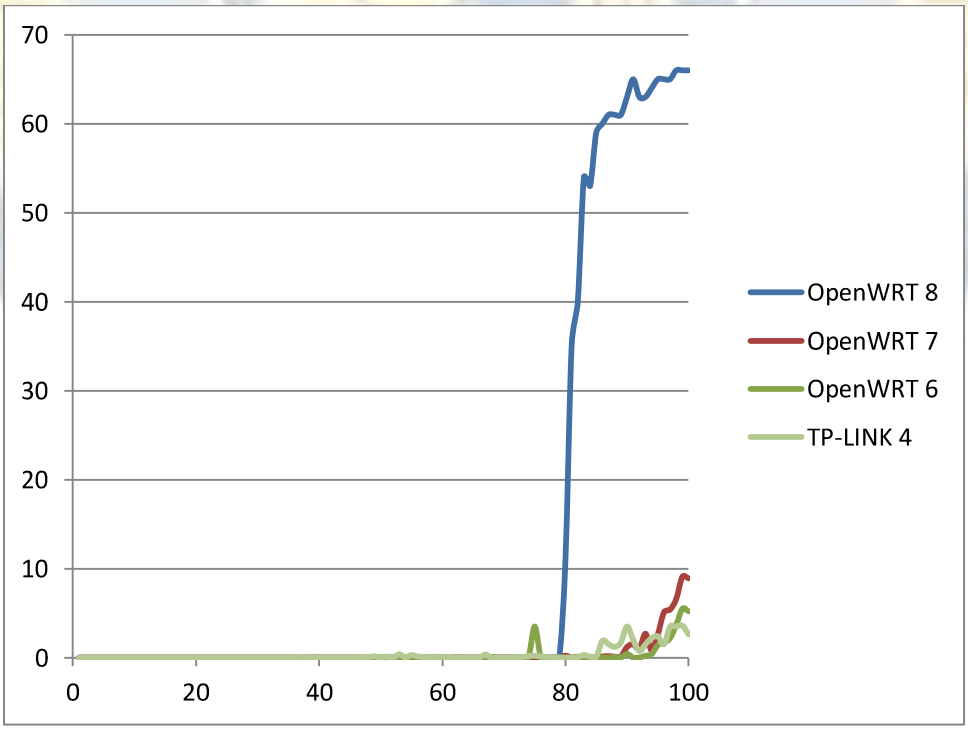


Ilustración 22 Ancho de banda transmitido [Mbit/s] vs Perdida de paquetes [%].

En la ilustración 22 se aprecia la ventaja de presentar la información de distintos modos. Si bien en las ilustraciones 20 y 21 no se destacan los signos de congestión en las

pruebas TP-LINK 4, OpenWRT6 y OpenWRT7, en la ilustración 22 se ve que el sistema Iptables (OpenWRT7) es más pesado de procesar. Congestiona con menos carga que VLAN aunque tenga un solo cliente.

El sistema IPTABLES aumenta la congestión a medida que aumenta el número de clientes. No obstante con 63 clientes conectados todavía puede manejar hasta cerca de 78 Mbit/s lo que es más que aceptables para muchas aplicaciones.

El sistema VLAN es independiente de la cantidad de usuarios y el impacto en la tasa de servicio es muy poco apreciable. Sin llegar a colapsar el sistema en todo el rango de la prueba.

4.7.2.3 Comparativa del Firmware original con OpenWrt

Las ilustraciones 23 y 24 muestran la comparativa de un mismo AP con una misma configuración pero con el firmware OpenWrt y el firmware Original.

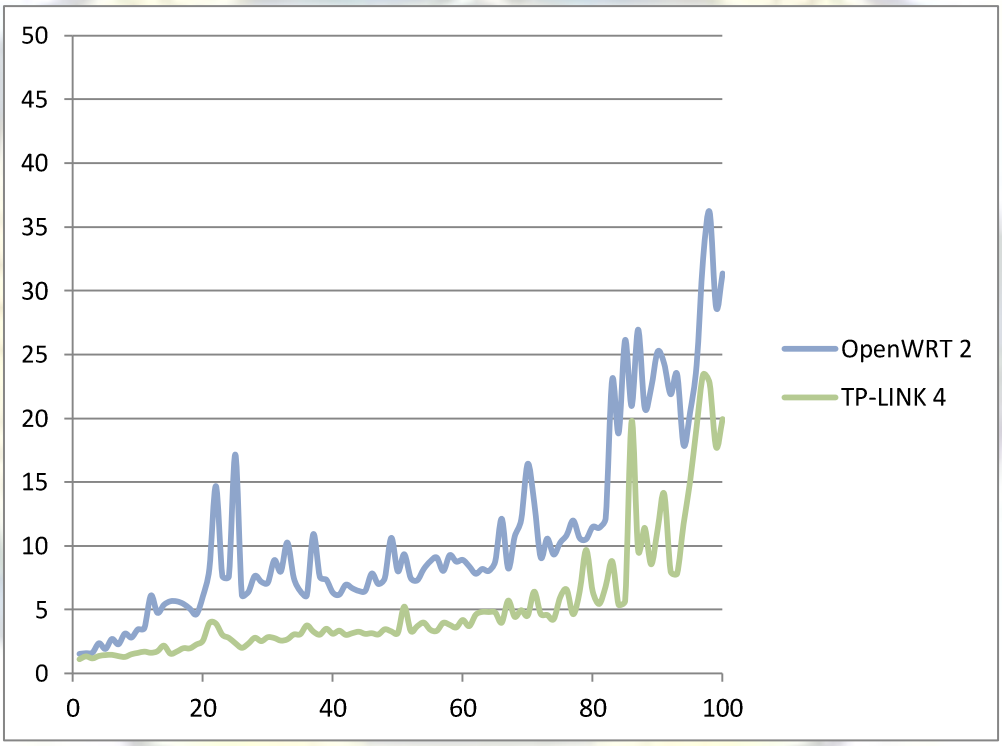


Ilustración 23 Ancho de banda transmitido [Mbit/s] vs retardo [mSeg].

En base a lo expuesto por las gráfica 18 se concluye que el kernel del firmware OpenWrt presenta mayores retardos que el kernel correspondiente al firmware original para procesar un paquete. Esto provoca un tiempo mayor de servicio y una congestión del sistema más temprana como se aprecia en la ilustración 19.

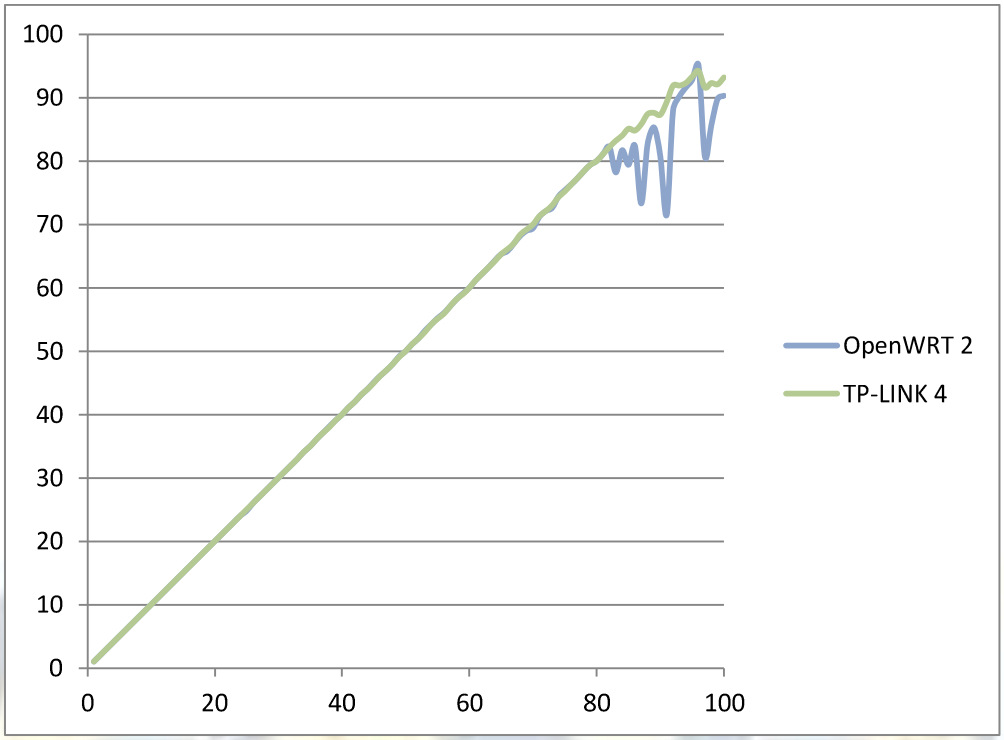


Ilustración 24 Ancho de banda transmitido [Mbit/s] vs Ancho de banda recibido [Mbit/s].

Este resultado indica que en posteriores revisiones del firmware se puede seguir optimizando el rendimiento.

4.7.2.4 Motivos para descartar el RaspberryPI

En base a los expuesto por las gráficas 17, 18 y 19 se descarta la aplicación con un RaspberryPI ya que sin aplicar ninguno de los métodos de filtrado muestra signos de congestión a partir de los 6 Mbit/s.

Las Ilustraciones 25, 26 y 27 contraponen el AP TP-730re contra la RaspberryPI con WPA activado y sin activar respectivamente. Todas las pruebas seleccionada no corresponden a los sistemas de filtrado.

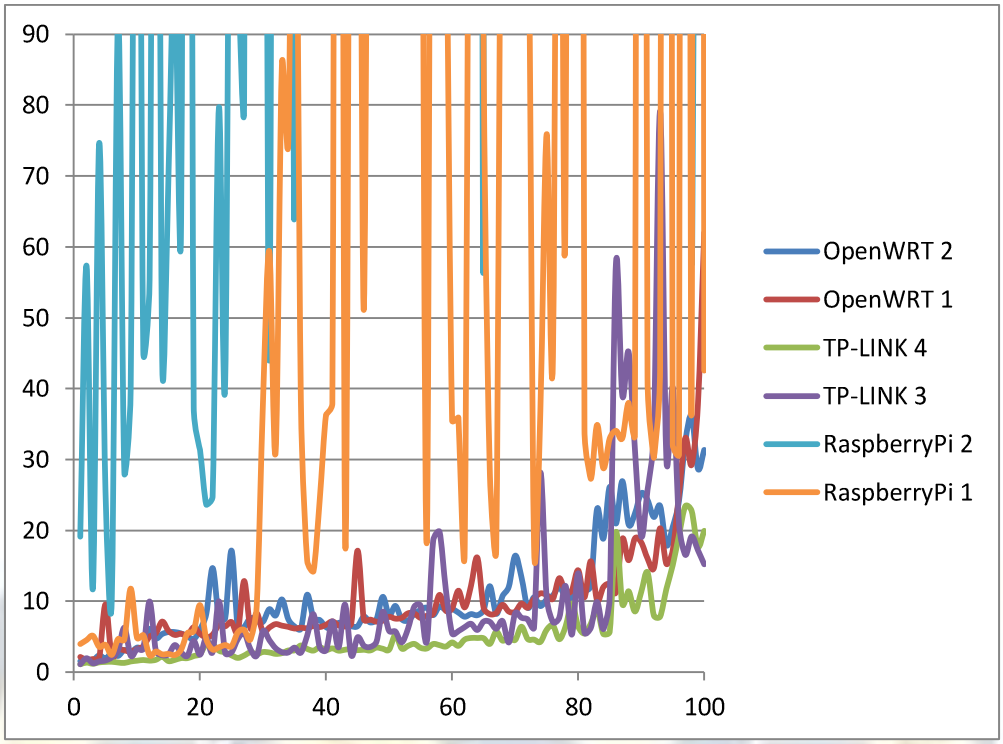


Ilustración 25 Ancho de banda transmitido [Mbit/s] vs retardo [mSeg].

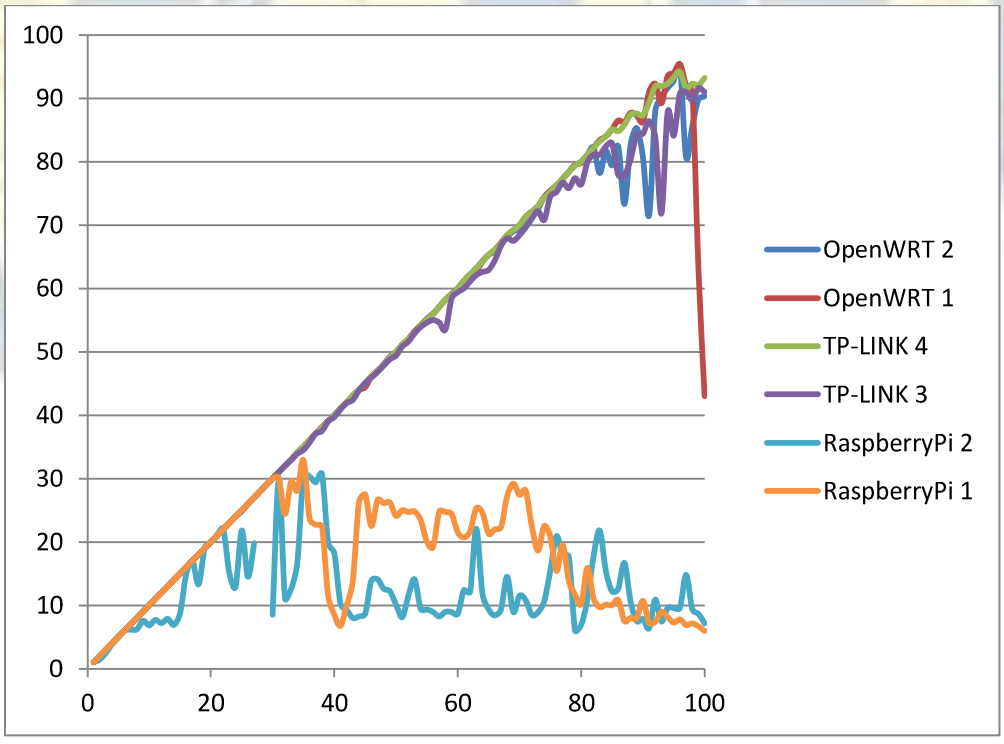


Ilustración 26 Ancho de banda transmitido [Mbit/s] vs Ancho de banda recibido [Mbit/s].

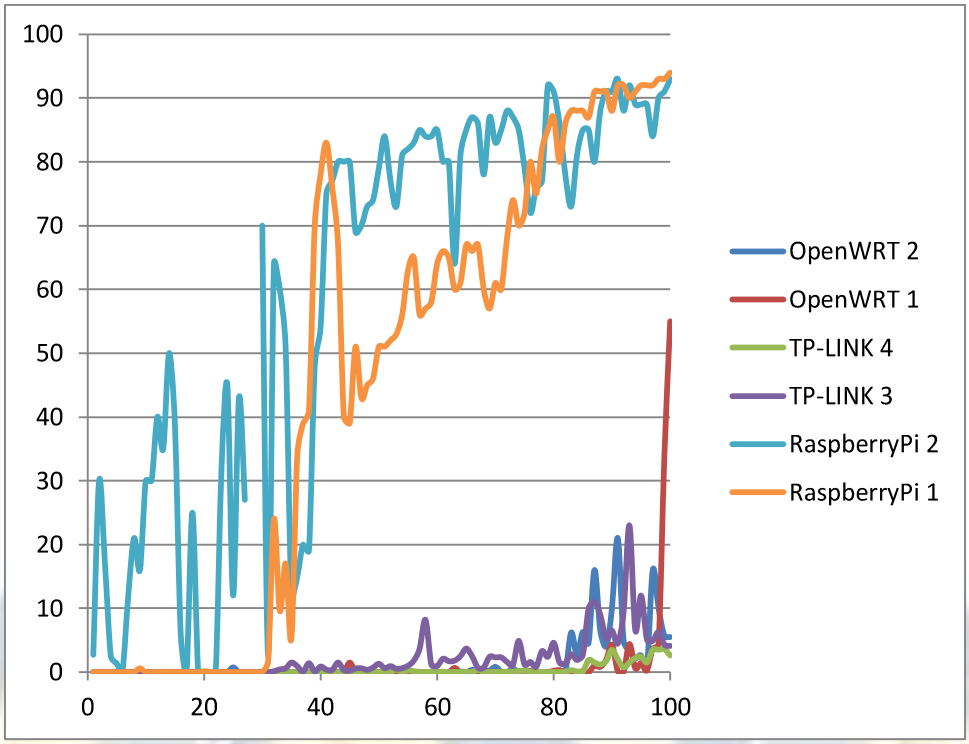


Ilustración 27 Ancho de banda transmitido [Mbit/s] vs Perdida de paquetes [%].

Se distingue claramente que por lo expuesto en las gráficas 25, 26 y 27 RaspberryPI se congestiona con una carga de 6 Mbit/s. muy por debajo del rendimiento del AP. Es por ello que se descartó la aplicación con un RaspberryPI ya que sin aplicar ninguno de los métodos de filtrado muestra signos de congestión a partir de los 6 Mbit/s.

5 Inversión requerida

No aplica.

6 Proyección de costos de operación y Mantenimiento

No aplica.

7 Análisis de viabilidad comercial

No aplica.

8 Análisis financiero

No aplica.

9 Estudio Ambiental

No aplica.

10 Estudio Social

No aplica.

11 Evaluación Económica

No aplica.

12 Conclusiones

Tras haber analizado implementado y ensayado con VLANs e IPTABLES se concluye que:

- La suposición de que el RaspberryPI presentaría una elevada performance con los sistemas de filtrado era errónea ya que sin aplicar ninguno de los métodos de filtrado muestra signos de congestión a partir de los 6 Mbit/s siendo esto muy inferior a lo posteriormente analizado en el AP TP-LINK TL-730RE.
- El sistema VLAN es independiente de la cantidad de usuarios y el impacto en la tasa de servicio es muy poco apreciable. Sin llegar a colapsar el sistema en todo el rango de la prueba.
- Con el filtrado por IPTABLES el ancho de banda del canal aumenta a medida que crece el número de usuarios registrados en el sistema.
- El AP TP-LINK TL-730RE demostró ser perfectamente capaz de manejar los dos sistemas de filtrado. Por ende como requerimientos mínimos para aplicar cualquiera de los dos sistemas se recomienda un procesador de 400MHz con 4 megas de RAM.
- Mientras el número de clientes sea bajo el AP TL-730RE procesa los paquetes con el sistema VLAN o el sistema IPTABLES robustamente, sin generar un gran aumento en el tiempo de servicio inclusive con un hardware tan limitado como el SoC analizado.
- El sistema IPTABLES aumenta la congestión a medida que crece el número de clientes. No obstante con 63 clientes conectados todavía puede manejar hasta cerca de 78Mbit/s Lo que es más que aceptables para muchas aplicaciones.
- En base a todo lo expuesto tomando sólo como referencia la performance del AP el sistema seleccionado es VLAN. Pese a ello dependiendo el entorno de red donde se aplique el sistema de filtrado, la selección podría variar ya que el sistema IPTABLES es más flexible y no requiere que la red posea un switch capaz de manejar VLAN.
- Ambos sistemas son totalmente factibles de instalar en AP comerciales sin disminuir significativamente su rendimiento.

13 Bibliografía

I. 802.11, «http://standards.ieee.org/,» 2012. [En línea]. Available: http://standards.ieee.org/about/get/802/802.11.html. [Último acceso: 10 2 2014].
1]

Atheros, «AR9002AP-1S AP/Router solution based on 802.11n,» 2014.
2]

adamyno, «OpenWrt Doc,» 23 11 2013. [En línea]. Available: http://wiki.OpenWrt.org/doc/start. [Último acceso: 26 2 2014].
3]

I. Wikimedia Foundation, 02 2014. [En línea]. Available: http://en.wikipedia.org/wiki/System-on-a-chip.
4]

L. Troval, 11 2013. [En línea]. Available: https://www.kernel.org/doc/Documentation/gpio/gpio.txt.
5]

I. 802.3, Marzo 2002. [En línea]. Available: http://standards.ieee.org/getieee802/download/802.3-2002.pdf.
6]

K. R. Sollins, «TFTP Revision 2,» 1992.
7]

I. 802.1Q, agosto 2011. [En línea]. Available: http://standards.ieee.org/about/get/802/802.1.html.
8]

L. Troval, 11 2013. [En línea]. Available: https://www.kernel.org/doc/Documentation/gpio/gpio.txt.
9]

P. N. Ayuso, «http://www.netfilter.org/,» 2014. [En línea]. Available: http://www.netfilter.org/. [Último acceso: 2014].
10]

Jengelh, «wikimedia,» wikipedia, 18 12 2011. [En línea]. Available: http://commons.wikimedia.org/wiki/File:Netfilter-packet-flow.svg. [Último acceso: 1 10 2014].
11]

W. STALLINGS, COMUNICACIONES Y REDES DE COMPUTADORES. Séptima edición, MADRID: PEARSON PRENTICE HALL, 2004.
12]

C.-Q. Y. a. A. V. Reddy, «A Taxonomy for Congestion Control Algorithms in Packet Switching Networks,» IEEE Networks, 1995.
13]

lorema, 3 2014. [En línea]. Available: http://wiki.OpenWrt.org/doc/networking/network.interfaces.
14]

Spansion®, «S25FL032P,» 2013.
15]

16] Future Technology Devices International Limited , «FT232R USB UART IC Datasheet,» Future Technology Devices International Limited (USA), ShangHai, 2010.

17] Bradford Liedel DBA ModemHelp Networks and Web Services. Mason, MI 48854 USA, 03 2014. [En línea]. Available: http://www.modemhelp.net/faqs/8n1.shtml.

18] micronet, 2013. [En línea]. Available: http://www.micronet.com.tw/mod/product/index.php?REQUEST_ID=cGFnZT1kZXRhaWwmUElEPTc1.

19] IEEE, «http://standards.ieee.org/,» Marzo 2002. [En línea]. Available: http://standards.ieee.org/getieee802/download/802.3-2002.pdf.

20] http://standards.ieee.org/, «http://standards.ieee.org/,» kñ, 1 4 2012. [En línea]. Available: http://standards.ieee.org/about/get/802/802.11.html. [Último acceso: 10 2 2014].

14 Anexos

14.1 Funciones DAS U-Boot

- reset - Reinicia la CPU
- ? - alias para 'help'
- base - Imprime la dirección base o el offset
- bootm - Inicia un firmware desde la memoria
- bootp - Inicia un firmware desde la red usado BootP/TFTP.
- cmp - compara dos archivos en memoria.
- cp - copia en memoria
- crc32 - calcula el checksum
- erase - borra la memoria FLASH
- flinfo - Imprime la información de la memoria FLASH
- go - Inicia una aplicación en la dirección 'addr'
- help - Imprime la lista de comandos.
- loadb - carga un archivo binario por el puerto serie (kermit mode)
- loady - carga un archivo binario por el puerto serie (ymodem mode)
- loop - infinite loop on address range
- md - memory display
- mm - memory modify (auto-incrementing)
- mtest - simple RAM test
- mw - memory write (fill)
- nm - memory modify (constant address)
- printenv - Imprime variables de entorno
- protec - habilita o deshabilita la protección contra escritura FLASH
- rarpboot - Inicia un firmware desde la red usado RARP/TFTP
- setenv - Establece variables de entorno
- tftpboot - Inicia un firmware desde la red usado TFTP
- version - Imprime la versión del U-Boot

14.2 Log de inicio de AP de referencia con OpenWrt

1. *U-Boot 1.1.4 (Sep 21 2010 - 13:23:41)*
2.
3. *AP91 (ar7240) U-boot*
4. *DRAM:*
5. *sri*
6. *#### TAP VALUE 1 = 8, 2 = 9*
7. *32 MB*
8. *id read 0x100000ff*
9. *flash size 4194304, sector count = 64*
10. *Flash: 4 MB*
11. *Using default environment*
12.
13. *In: serial*
14. *Out: serial*
15. *Err: serial*
16. *Net: ag7240_enet_initialize...*
17. *No valid address in Flash. Using fixed address*
18. *: cfg1 0xf cfg2 0x7014*
19. *eth0: 00:03:7f:09:0b:ad*
20. *eth0 up*
21. *No valid address in Flash. Using fixed address*
22. *: cfg1 0xf cfg2 0x7214*
23. *eth1: 00:03:7f:09:0b:ad*
24. *ATHRS26: resetting s26*
25. *ATHRS26: s26 reset done*
26. *eth1 up*
27. *eth0, eth1*
28. *Autobooting in 1 seconds*
29. *## Booting image at 9f020000 ...*
30. *   Uncompressing Kernel Image ... OK*
31.
32. *Starting kernel ...*
33.
34.
35.
36. *OpenWrt kernel loader for AR7XXX/AR9XXX*
37. *Copyright (C) 2011 Gabor Juhos <juhosg@OpenWrt.org>*
38. *Looking for OpenWrt image... found at 0xbf022000*
39. *Decompressing kernel... done!*
40. *Starting kernel at 80060000...*
41.
42. *[0.000000] Linux version 3.10.28 (fnord@tschunk) (gcc version 4.6.4 (OpenWrt/Linaro GCC 4.6-2013.05 r39535)) #1 Sun Feb 9 19:30:00 UTC 2014*

43. [0.000000] bootconsole [early0] enabled
44. [0.000000] CPU revision is: 00019374 (MIPS 24Kc)
45. [0.000000] SoC: Atheros AR7240 rev 2
46. [0.000000] Clocks: CPU:350.000MHz, DDR:350.000MHz, AHB:175.000MHz, Ref:5.000MHz
47. [0.000000] Determined physical RAM map:
48. [0.000000] memory: 02000000 @ 00000000 (usable)

14.3 Log de inicio de AP de referencia con Firmware original

1. *U-Boot 1.1.4 (Sep 21 2010 - 13:23:41)*
2.
3. *AP91 (ar7240) U-boot*
4. *DRAM:*
5. *sri*
6. *#### TAP VALUE 1 = 8, 2 = 8*
7. *32 MB*
8. *id read 0x100000ff*
9. *flash size 4194304, sector count = 64*
10. *Flash: 4 MB*
11. *Using default environment*
12.
13. *In: serial*
14. *Out: serial*
15. *Err: serial*
16. *Net: ag7240_enet_initialize...*
17. *No valid address in Flash. Using fixed address*
18. *: cfg1 0xf cfg2 0x7014*
19. *eth0: 00:03:7f:09:0b:ad*
20. *eth0 up*
21. *No valid address in Flash. Using fixed address*
22. *: cfg1 0xf cfg2 0x7214*
23. *eth1: 00:03:7f:09:0b:ad*
24. *ATHRS26: resetting s26*
25. *ATHRS26: s26 reset done*
26. *eth1 up*
27. *eth0, eth1*
28. *Autobooting in 1 seconds*
29. *## Booting image at 9f020000 ...*
30. *Uncompressing Kernel Image ... OK*
31.
32. *Starting kernel ...*
33.
34. *Booting AR7240(Python)...*
35. *Linux version 2.6.15--LSDK-7.3.0.300 gcc version 3.4.4 #12 Thu Aug 19 17:28:05 CST 2010*
36. *flash_size passed from bootloader = 4*
37. *CPU revision is: 00019374*
38. *Determined physical RAM map:*
39. *memory: 02000000 @ 00000000 (usable)*
40. *User-defined physical RAM map:*
41. *memory: 02000000 @ 00000000 (usable)*
42. *Built 1 zonelists*

43. *Kernel command line: console=ttyS0,115200 root=/dev/mtdblock2 rootfstype=squashfs init=/sbin/init mtdpartM*
44. *Primary instruction cache 64kB, physically tagged, 4-way, linesize 32 bytes.*
45. *Primary data cache 32kB, 4-way, linesize 32 bytes.*
46. *Synthesized TLB refill handler (20 instructions).*
47. *Synthesized TLB load handler fastpath (32 instructions).*
48. *Synthesized TLB store handler fastpath (32 instructions).*
49. *Synthesized TLB modify handler fastpath (31 instructions).*
50. *Cache parity protection disabled*
51. *PID hash table entries: 256 (order: 8, 4096 bytes)*
52. *Using 175.000 MHz high precision timer.*
53. *Dentry cache hash table entries: 8192 (order: 3, 32768 bytes)*
54. *Inode-cache hash table entries: 4096 (order: 2, 16384 bytes)*
55. *Memory: 30548k/32768k available (1368k kernel code, 2204k reserved, 291k data, 128k init, 0k highmem)*
56. *Mount-cache hash table entries: 512*
57. *Checking for 'wait' instruction... available.*
58. *NET: Registered protocol family 16*
59. *Reset button pressed.*
60. *Returning IRQ 48*
61. *AR7240 GPIOC major 0*
62. *squashfs: version 3.3 (2007/10/31) Phillip Lougher*
63. *squashfs: LZMA suppport for slax.org by jro*
64. *Initializing Cryptographic API*
65. *io scheduler noop registered*
66. *io scheduler deadline registered*
67. *Serial: 8250/16550 driver $Revision: #1 $ 1 ports, IRQ sharing disabled*
68. *serial8250.0: ttyS0 at MMIO 0x0 (irq = 19) is a 16550A*
69. *RAMDISK driver initialized: 1 RAM disks of 8192K size 1024 blocksize*
70. *PPP generic driver version 2.4.2*
71. *PPPoX init, max protocols:3*
72. *NET: Registered protocol family 24*
73. *pppox protocol 0 register. max:3*
74. *cmdlinepart partition parsing not available*
75. *Searching for RedBoot partition table*
76. *5 RedBoot partitions found on MTD device ar7240-nor0*
77. *Creating 5 MTD partitions on "ar7240-nor0":*
78. *0x00000000-0x00020000 : "boot"*
79. *0x00020000-0x00120000 : "kernel"*
80. *0x00120000-0x003e0000 : "rootfs"*
81. *0x003e0000-0x003f0000 : "config"*
82. *0x003f0000-0x00400000 : "art"*
83. *->Oops: flash id 0x10215 .*
84. *NET: Registered protocol family 2*
85. *IP route cache hash table entries: 512 (order: -1, 2048 bytes)*
86. *TCP established hash table entries: 2048 (order: 1, 8192 bytes)*

87. TCP bind hash table entries: 2048 (order: 1, 8192 bytes)
88. TCP: Hash tables configured (established 2048 bind 2048)
89. TCP reno registered
90. TCP bic registered
91. NET: Registered protocol family 1
92. NET: Registered protocol family 17
93. 802.1Q VLAN Support v1.8 Ben Greear <greearb@candelatech.com>
94. All bugs added by David S. Miller <davem@redhat.com>
95. VFS: Mounted root (squashfs filesystem) readonly.
96. Freeing unused kernel memory: 128k freed
97. init started: BusyBox v1.01 (2010.04.08-11:57+0000) multi-call binary
98. Algorithmics/MIPS FPU Emulator v1.5
99. ip_conntrack version 2.4 (256 buckets, 5120 max) - 244 bytes per conntrack
100. insmod: cannot open module `/lib/modules/2.6.15/kernel/ip_nat.ko': No such file or directory
101. insmod: cannot open module `/lib/modules/2.6.15/kernel/ipt_MASQUERADE.ko': No such file or directory
102. insmod: cannot open module `/lib/modules/2.6.15/kernel/ipt_conntrack.ko': No such file or directory
103. insmod: cannot open module `/lib/modules/2.6.15/kernel/ipt_iprange.ko': No such file or directory
104. insmod: cannot open module `/lib/modules/2.6.15/kernel/ipt_mac.ko': No such file or directory
105. insmod: cannot open module `/lib/modules/2.6.15/kernel/ipt_string.ko': No such file or directory
106. insmod: cannot open module `/lib/modules/2.6.15/kernel/iptable_nat.ko': No such file or directory
107. insmod: cannot open module `/lib/modules/2.6.15/kernel/ipt_time.ko': No such file or directory
108. insmod: cannot open module `/lib/modules/2.6.15/kernel/ipt_TRIGGER.ko': No such file or directory
109. insmod: cannot open module `/lib/modules/2.6.15/kernel/ip_conntrack_pptp.ko': No such file or directory
110. insmod: cannot open module `/lib/modules/2.6.15/kernel/ipt_TCPMSS.ko': No such file or directory
111. insmod: cannot open module `/lib/modules/2.6.15/kernel/ipt_multiurl.ko': No such file or directory
112. insmod: cannot open module `/lib/modules/2.6.15/kernel/ip_conntrack_h323.ko': No such file or directory
113. insmod: cannot open module `/lib/modules/2.6.15/kernel/ipt_MARK.ko': No such file or directory
114. insmod: cannot open module `/lib/modules/2.6.15/kernel/sch_htb.ko': No such file or directory
115. insmod: cannot open module `/lib/modules/2.6.15/kernel/sch_prio.ko': No such file or directory

116. insmod: cannot open module `/lib/modules/2.6.15/kernel/sch_sfq.ko': No such file or directory
117. insmod: cannot open module `/lib/modules/2.6.15/kernel/cls_basic.ko': No such file or directory
118. insmod: cannot open module `/lib/modules/2.6.15/kernel/cls_fw.ko': No such file or directory
119. insmod: cannot open module `/lib/modules/2.6.15/kernel/ts_kmp.ko': No such file or directory
120. insmod: cannot open module `/lib/modules/2.6.15/kernel/flashid.ko': No such file or directory
121. insmod: cannot open module `/lib/modules/2.6.15/kernel/harmony.ko': No such file or directory
122. Now flash open!
123. Now flash open!
124. sys_mode=3 mac:8106c000 ag7240_macs[0]:00000000 ag7240_macs[1]:8106c000
125. 2
126. sys_mode=3 mac:8030c000 ag7240_macs[0]:8030c000 ag7240_macs[1]:8106c000
127. 1
128. sys_mode 3 PHY4 auto-negotiation enable
129. Attention: PHY4 Class A setting in debug 5 register which is not described in DS
130. phy_setup ethUnit:0 phyUnit:4
131. Port 4, Neg Success
132.
133. (none) mips #12 Thu Aug 19 17:28:05 CST 2010 (none)
134. (none) login: ATHRS26: resetting s26
135. ATHRS26: s26 reset done
136. sys_mode 3 PHY0 auto-negotiation enable
137. Attention: PHY 0 Class A setting in debug 5 register which is not described in DS
138. sys_mode 3 PHY1 auto-negotiation enable
139. Attention: PHY 1 Class A setting in debug 5 register which is not described in DS
140. sys_mode 3 PHY2 auto-negotiation enable
141. Attention: PHY 2 Class A setting in debug 5 register which is not described in DS
142. sys_mode 3 PHY3 auto-negotiation enable
143. Attention: PHY 3 Class A setting in debug 5 register which is not described in DS
144. phy_setup ethUnit:1 phyUnit:0
145. phy_setup ethUnit:1 phyUnit:1
146. phy_setup ethUnit:1 phyUnit:2
147. phy_setup ethUnit:1 phyUnit:3
148. Port 0, Neg Success

149. Port 1, Neg Success
150. Port 2, Neg Success
151. Port 3, Neg Success
152. device eth0 entered promiscuous mode
153. phy_setup ethUnit:1 phyUnit:0
154. phy_setup ethUnit:1 phyUnit:1
155. phy_setup ethUnit:1 phyUnit:2
156. phy_setup ethUnit:1 phyUnit:3
157. Port 0, Neg Success
158. Port 1, Neg Success
159. Port 2, Neg Success
160. Port 3, Neg Success
161. ath_hal: module license 'Proprietary' taints kernel.
162. ath_hal: 0.9.17.1 (AR5416, DEBUG, REGOPS_FUNC, WRITE_EEPROM, 11D)
163. wlan: 0.8.4.2 (Atheros/multi-bss)
164. ath_rate_atheros: Copyright (c) 2001-2005 Atheros Communications, Inc, All Rights Reserved
165. ath_dev: Copyright (c) 2001-2007 Atheros Communications, Inc, All Rights Reserved
166. ath_pci: 0.9.4.5 (Atheros/multi-bss)
167. wifi0: Atheros 9285: mem=0x10000000, irq=48 hw_base=0xb0000000
168. ++++ phy_up is 1, fdx is 1, speed is 1
169. AG7240: enet unit:1 phy:0 is up...Mii 100Mbps full duplex
170. AG7240: enet unit 1 phy 0 mode 0x4c04
171. br0: port 1(eth0) entering learning state
172. br0: topology change detected, propagating
173. br0: port 1(eth0) entering forwarding state
174. wlan_me: Version 0.1
175. Copyright (c) 2008 Atheros Communications, Inc. All Rights Reserved
176. wlan: mac acl policy registered
177. Country ie is AR
178. maxrate = 150000
179. Country ie is AR
180. br0: port 1(eth0) entering disabled state
181. phy_setup ethUnit:1 phyUnit:0
182. phy_setup ethUnit:1 phyUnit:1
183. phy_setup ethUnit:1 phyUnit:2
184. phy_setup ethUnit:1 phyUnit:3
185. Port 0, Neg Success
186. Port 1, Neg Success
187. Port 2, Neg Success
188. Port 3, Neg Success
189. device ath0 entered promiscuous mode
190. br0: port 2(ath0) entering learning state
191. br0: topology change detected, propagating

192. br0: port 2(ath0) entering forwarding state
193. br0: port 2(ath0) entering disabled state
194. Country ie is AR
195. br0: port 2(ath0) entering learning state
196. br0: topology change detected, propagating
197. br0: port 2(ath0) entering forwarding state
198. ++++ phy_up is 1, fdx is 1, speed is 1
199. AG7240: enet unit:1 phy:0 is up...Mii 100Mbps full duplex
200. AG7240: enet unit 1 phy 0 mode 0x4c04
201. br0: port 1(eth0) entering learning state
202. br0: topology change detected, propagating
203. br0: port 1(eth0) entering forwarding state
204. br0: port 2(ath0) entering disabled state
205. Country ie is AR
206. br0: port 2(ath0) entering learning state
207. br0: topology change detected, propagating
208. br0: port 2(ath0) entering forwarding state
209.
*210. OPERMODE:0****
211.
212. PRODUCTID:7300001
213. receive set intr_mode:1
214.
215. CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7 | VT102 | Desconectado | ttyUSB0

14.4 Lista de paquetes del Firmware

1. *make[1] world*
2. *make[2] target/compile*
3. *make[3] -C target/linux compile*
4. *make[2] package/cleanup*
5. *make[2] package/compile*
6. *make[2] package/install*
7. *make[3] -C package/arptables install*
8. *make[3] -C package/bridge-utils install*
9. *make[3] -C package/busybox install*
10. *make[3] -C package/dropbear install*
11. *make[3] -C package/ebtables install*
12. *make[3] -C package/firewall install*
13. *make[3] -C package/gpio-button-hotplug install*
14. *make[3] -C package/hostapd install*
15. *make[3] -C package/hotplug2 install*
16. *make[3] -C package/iproute2 install*
17. *make[3] -C package/iptables install*
18. *make[3] -C package/iw install*
19. *make[3] -C package/iwinfo install*
20. *make[3] -C package/kernel install*
21. *make[3] -C package/libjson-c install*
22. *make[3] -C package/libnl-tiny install*
23. *make[3] -C package/libubox install*
24. *make[3] -C package/mac80211 install*
25. *make[3] -C package/mtd install*
26. *make[3] -C package/netifd install*
27. *make[3] -C package/opkg install*
28. *make[3] -C package/swconfig install*
29. *make[3] -C package/toolchain install*
30. *make[3] -C package/uboot-envtools install*
31. *make[3] -C package/ubus install*
32. *make[3] -C package/uci install*
33. *make[3] -C package/util-linux install*
34. *make[3] -C package/base-files install*
35. *make[2] package/rootfs-prepare*
36. *make[3] package/preconfig*
37. *make[2] target/install*
38. *make[3] -C target/linux install*
39. *make[2] package/index*

15 Códigos de computación

15.1 Script test_red.sh

```
#!/bin/bash
#script para medir el retardo a un punto de la red con distintas cargas de tráfico.
#Duración de la prueba.
time=30
server=192.168.1.3
echo 'bandwidthTX( Mbit/s), rt(mseg), bandwidthRX( Mbit/s), Jiter(mseg), PaquetLost(%)' > logred.dat
for bandwidthTX in `seq 1 100`;
do
#ifconfig eth3 down
#sleep 1
#ifconfig eth3 up
iperf -c $server -f m -t $time -b $bandwidthTX'M' 2> /dev/null |  awk 'BEGIN {OFS=", "} { if ( NR == 11 ) {gsub(/[%,(,)]/,"",$NF); print $7,  $9, $NF}}' > logred.tmp &
RTT=$(ping $server -c $time -i 1 | grep rtt | cut -d"=" -f2 | cut -d"/" -f2)
sleep 2
iperf=$(cat logred.tmp)
echo $bandwidthTX, $RTT, $iperf >> logred.dat
#tiempo entre puerbas.
sleep 10
done
rm logred.tmp
```