Trabajo Final
Comunicaciones Inalámbricas 2014

Matías Richart - 3.953.948-4

18 de febrero de 2015

Estudio de un transmisor y receptor 802.11 en GNU Radio

Introducción

Existe un trabajo reciente [1] en el que se implementa en GNU Radio un receptor OFDM según el estándar 802.11. En particular se basa en 802.11a/g/p los cuales tienen una capa física muy similar. Por otro lado, también esta disponible un transmisor 802.11 el cual se describe en [2].

El código del transmisor/receptor está disponible bajo la licencia GPLv3.

Instalación

Como primer paso del trabajo se realizó la instalación y configuración del transmisor y receptor. Esto resultó bastante sencillo, siguiendo detalladamente los pasos indicados por los autores del trabajo en [3]. Se debe tener en cuanta que un requerimiento excluyente es utilizar la versión de GNU Radio 3.7 o posterior.

Luego de la instalación, como sugieren los autores, se puede probar el sistema utilizando el ejemplo `wifi_loopback.grc`. Este ejemplo simula un canal simple y conecta a través de este el transmisor con el receptor. Por lo tanto, es posible probar la instalación sin necesidad de un USRP. Existen otros ejemplos para utilizar con un dispositivo USRP, estos son `wifi_rx.gr`, `wifi_tx.gr` y `wifi_transceiver.gr`, los cuales analizaremos mas adelante.

Estudio

A partir de los artículos que describen el transmisor y receptor, se estudió la implementación en GNU Radio. A continuación se describen brevemente los aspectos que resultaron mas interesantes de la misma.

Receptor

La implementación del receptor puede ser dividida en varios pasos: la detección de una trama, la alineación con los símbolos, la corrección de frecuencia y fase, la estimación del canal y la posterior decodificación de la trama. En la Figura 1 se muestra el diagrama del receptor.

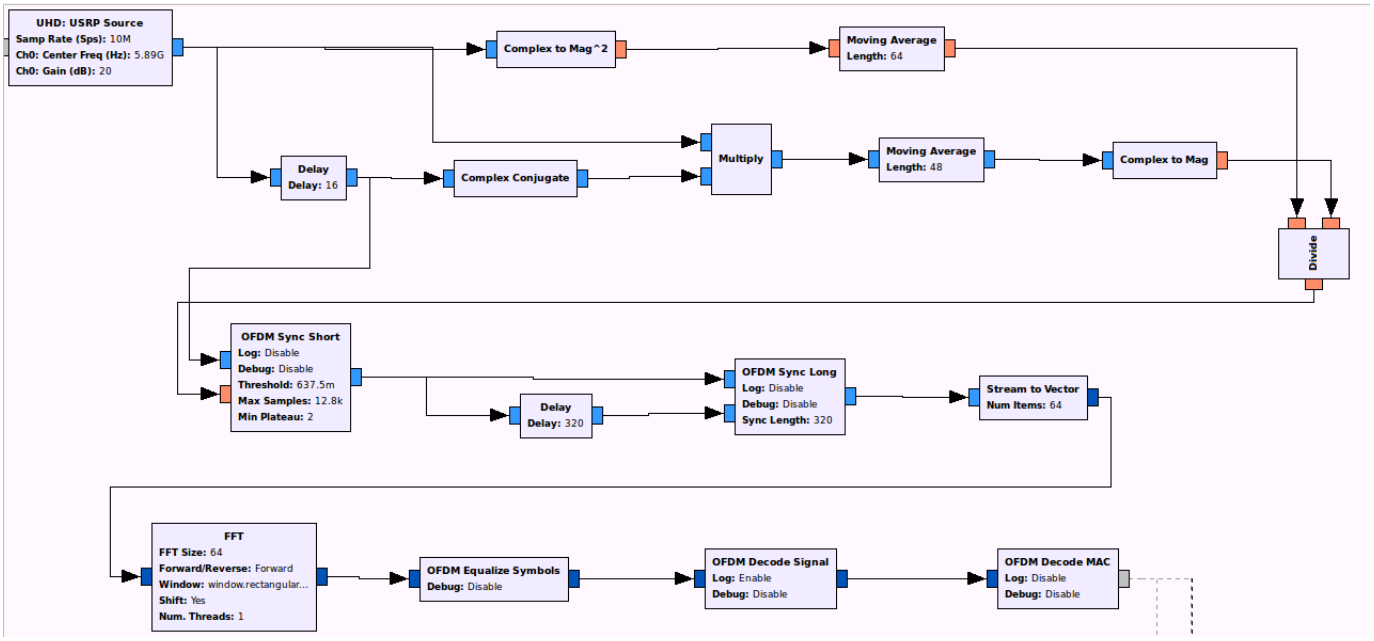


Figura 1: Implementación del receptor 802.11 en GNU Radio Companion

Para poder entender como se realizan estos pasos es importante conocer la composición de una trama 802.11 y en particular del preámbulo OFDM PLCP y el cabezal OFDM PLCP. El PLCP es el Physical Layer Convergence Procedure, es decir que estamos analizando como es el preámbulo y cabezal de la trama a nivel de capa física (recordar que el estándar 802.11 define tanto la capa física como la capa de enlace). En la Figura 2 (obtenida del estándar) podemos ver como es la composición de la trama. Para las primeras etapas de detección y alineación nos va a interesar el preámbulo. En la Figura 3 se puede ver un diagrama del mismo con los tiempos de cada símbolo para una tasa de muestreo de 20MHz (por defecto en 802.11). Este preámbulo esta compuesto por 12 símbolos, los primeros 10 determinan la *short training sequence* (secuencia corta) y los últimos dos la *long training sequence* (secuencia larga). Los símbolos de la secuencia corta consisten de 16 muestras cada uno, es decir que la secuencia tiene un largo de 160 muestras. Los dos símbolos de la secuencia larga se expanden 64 muestras cada uno y además están precedidos por un prefijo cíclico de 32 muestras (GI2 en la figura).

La detección de una trama es la primer tarea que realiza el receptor. Para esto lo que se hace es detectar la presencia de la *short training sequence*. Para esto, en la implementación estudiada, se utiliza la autocorrelación de la secuencia corta siguiendo el algoritmo propuesto en [4]. La idea del algoritmo es, aprovechando

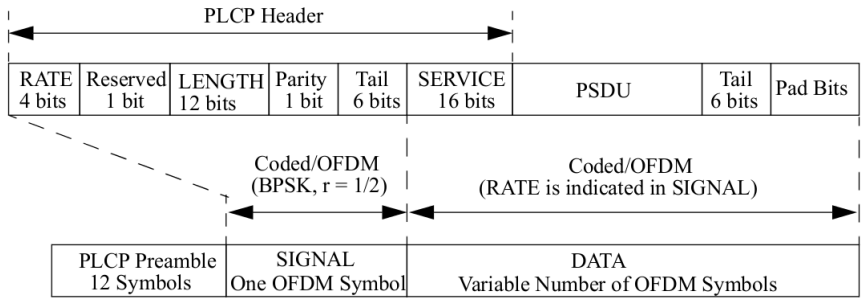


Figura 2: Formato de la trama 802.11

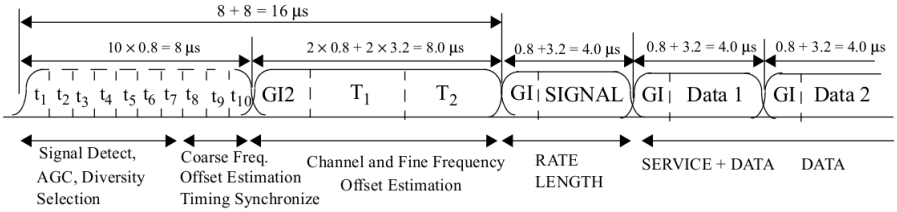


Figura 3: Preámbulo OFDM PLCP

la periodicidad de la secuencia, calcular la autocorrelación A de las muestras recibidas $r[n]$ con un retardo de 16, sobre una ventana de largo L.

$$A[n] = \sum_{k=0}^{L-1} r[n-k] * r[n-k-16]$$

Luego, este valor es normalizado, utilizando la energía promedio de la señal recibida $E[n]$ y obtenemos $C[n]$. Cuando este valor supera un cierto umbral configurable, se determina que se ha detectado un nueva trama.

$$E[n] = \sum_{k=0}^{L-1} r[n-k] * r[n-k]$$

$$C[n] = \frac{|A[n]|}{E[n]}$$

En la implementación estudiada el largo de la ventana es $L = 48$ y se determina que hay una nueva trama cuando durante 3 muestras $C[n]$ es mayor que el umbral. Por cada trama detectada, se pasa a los siguientes bloques un número fijo de muestras. Se puede ver en la Figura 1 que el bloque que realiza esta comparación es `OFDM Sync Short`.

El siguiente bloque en la implementación es `OFDM Sync Long`. Este bloque se encarga de la corrección de frecuencia y del alineamiento de los símbolos.

Para la corrección de frecuencia se utiliza también la secuencia corta. En este caso se comparan las muestras que deberían ser iguales, es decir las separadas

una distancia de 16. Si hay corrimiento en frecuencia estas muestras estarán rotadas, por lo que para cada muestra se mide esta diferencia y se toma un promedio. Con esto se calcula la corrección a realizar.

El alineamiento consiste en detectar donde comienzan los símbolos OFDM, y de esta manera poder pasar estos datos al bloque que realiza la FFT. Para esto se utiliza la secuencia larga de entrenamiento (*long training sequence*) que como mencionamos consiste en un patrón de 64 muestras que se repite dos veces y media. Un detalle importante es que este patrón de las secuencias de entrenamiento es conocido y dado por el estándar. Entonces, es posible utilizar un filtro apareado para detectar esta secuencia. Esto genera tres valores altos (picos) en el cálculo de la correlación (ver Figura 4) y es lo que utiliza la implementación estudiada para sincronizarse perfectamente con la trama y poder determinar las muestras en donde comienza cada símbolo de datos.

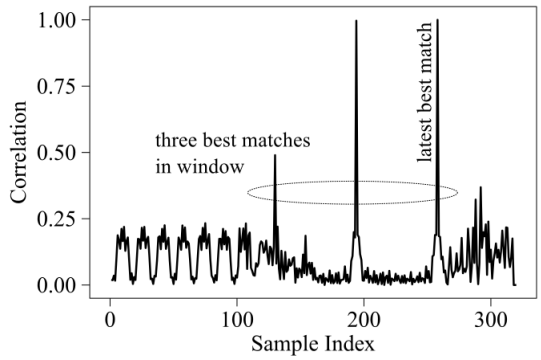


Figura 4: Comportamiento característico de la autocorrelación (obtenido de [1]).

El índice de los picos se calcula:

$$N_p = \underset{n \in \{0,\ldots,N_{preamble}\}}{\arg\max_3} \sum_{k=0}^{63} s[n+k]LT[k]$$

Donde LT es el patrón de la secuencia larga, $N_{preamble}$ es la suma del largo de la secuencia corta y larga y $\arg\max_3$ devuelve los 3 índices mas altos que maximizan la expresión. De esta manera sabemos que la muestra que se encuentra 64 posiciones a partir de la muestra donde se da el tercer pico es el comienzo del cabezal de la trama. De esta manera el bloque `OFDM Sync Long` puede de manera precisa descartar el preámbulo y enviar al siguiente bloque el resto de la trama.

Los símbolos son recibidos luego por el bloque `FFT` el cual aplica la Transformada de Fourier y pasamos a trabajar en frecuencia. Cada símbolo ahora consiste en 64 subportadoras. Luego de este bloque, se encuentra el bloque `OFDM Equalize Symbols` el cual se encarga de corrección en fase y una simple estimación del canal. Para la corrección de fase se utilizan las sub-portadoras piloto (*pilot subcarriers*) de cada símbolo. El estándar indica que por cada símbolo, cuatro de las sub-portadoras son dedicadas a señales piloto la cuales codifican una secuencia conocida utilizando BPSK. Este bloque además quita de cada símbolo

OFDM el *Guard Interval* (el cual consiste en el prefijo cíclico), la sub-portadora de la frecuencia central y las cuatro sub-portadoras piloto. Con lo cual al final de este bloque obtenemos las 48 sub-portadoras con datos.

Luego, se pasa a decodificar el campo SIGNAL de la trama, el cual como se puede ver en la Figura 2 contiene información del largo y la modulación de la trama. Este campo esta modulado con BPSK y codificado utilizando código convolucional de tasa $\frac{1}{2}$. Con estos datos se decodifica este campo y si el resultado es correcto (notar que contiene 1 bit de paridad), se pasa el resto de las muestras al siguiente bloque. El flujo de muestras se etiqueta con la información de largo y modulación obtenida de este campo. De esta manera el siguiente bloque conoce que mecanismo utilizar para decodificar los datos.

Finalmente se llega al último bloque `OFDM Decode MAC`. Este bloque recibe las 48 subportadoras y el dato de que modulación y codificación es utilizada y realiza los pasos inversos al transmisor:

1. Demodulación.
2. *Deinterleaving.*
3. Decodificación del código convolucional.
4. *Descrambling*

Transmisor

La implementación del transmisor se encuentra mucho menos documentada que la del receptor, sin embargo parece ser mas sencilla. En este caso hay que seguir los pasos descritos en el estándar e ir agregando los distintos elementos que forman la trama. A diferencia del receptor, no es necesario ningún mecanismo complejo como el utilizado para detectar o sincronizarse con una trama.

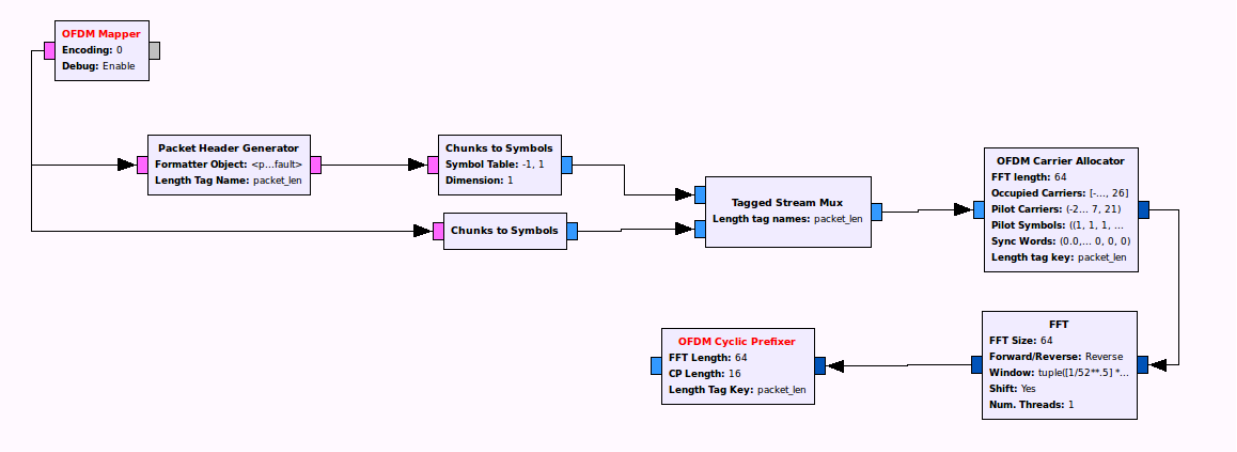


Figura 5: Implementación del transmisor 802.11 en GNU Radio Companion

Una explicación clara del proceso para transmitir una trama OFDM se puede ver en el apartado 18.3.2.2 del estándar 802.11 [5]. A continuación explicaremos

brevemente que partes de este proceso realiza cada bloque de la Figura 5. Es importante notar que en este caso la descripción de cada bloque se basa en un análisis mínimo del código y no en documentación precisa.

El bloque `OFDM Mapper` recibe como parámetro la MCS (modulación y codificación) a utilizar y se encarga de realizar varios procedimientos sobre los datos a enviar:

1. Genera el campo de datos, el cual incluye el campo SERVICE y los Tail y Pad bits.
2. Realiza el *Scrambling* de los bits.
3. Reinicia en cero los bits de Tail.
4. Codifica los datos con un código convolucional dependiendo de la tasa deseada.
5. Realiza el *Interleaving* de los bits.
6. Divide a los bits en grupos y convierte a cada grupo en un complejo dependiendo de la modulación deseada.

Todos estos pasos se corresponden con los pasos c)-i) del apartado 18.3.2.2.

Luego el bloque `Packet Header Generator` genera cabezal de la trama, lo cual incluye el campo SIGNAL y el campo SERVICE. El cabezal debe ser modulado con BPSK y esto es lo que hace el bloque `Chunks to Symbols` superior. El resto de la trama se modula con el esquema indicado como parámetro y esto es lo que hace el bloque `Chunks to Symbols` inferior. Finalmente se une el cabezal al resto de la trama.

Luego el bloque `OFDM Carrier Allocator` agrega las sub-portadoras piloto y el bloque `FFT` realiza la Transformada de Fourier Inversa y pasamos al dominio de tiempo.

Por último el bloque `OFDM Cyclic Prefixer` agrega los prefijos cíclicos (o *guard intervals*) a cada símbolo de la trama.

Pruebas con USRP

Como segunda parte del trabajo se realizaron varias pruebas de la implementación utilizando dispositivos USRP y dispositivos WiFi comerciales. El objetivo de estas pruebas es poder evaluar las posibilidades y limitantes de la implementación, no es un objetivo medir el rendimiento.

Según la documentación de los autores del receptor, la implementación tiene algunas falencias que limitan las funcionalidades del mismo. Por ejemplo:

- No están implementados mecanismos de señalización, por lo que no es posible asociarse a otro dispositivo.
- El receptor no puede generar tramas ACK en el tiempo exigido por el estándar.

- No está implementado el mecanismo de acceso al medio CSMA/CA, definido en el modo DFC.
- Solo soporta modulación BPSK y QPSK debido a que carece de un mecanismo de estimación de canal adecuado que solucione problemas de diferencias de magnitud de la señal.

Debido a estas carencias, las pruebas que son posibles de realizar son limitadas. Nos enfocaremos en pruebas similares a las que realizan los autores en los artículos [1], [6], [7]. Estos trabajos prueban la interoperabilidad del receptor implementado y en particular [7] prueba el sistema emisor/receptor pero solo con el estándar 802.11p.

En nuestro trabajo se prueba la interoperailidad del emisor/receptor con el estándar 802.11g.

Interoperabilidad Tarjeta WiFi - USRP

El dispositivo WiFi utilizado es un router inalámbrico hogareño *TP-LINK WDR4300* utilizando el sistema operativo *OpenWrt 14.07*. Este router cuenta con una tarjeta *Atheros AR9341* y se utiliza el driver ath9k de linux. Se utiliza el *USRP N210* con una daughterboard *SBX 400-4400 MHz Rx/Tx* por lo que las pruebas se realizarán utilizando el estándar 802.11 en el rango de frecuencias de 2.4Ghz. El USRP se conecta a una máquina con el sistema operativo *Ubuntu 14.04*, con procesador Intel Core i5 y una interfaz gigabit ethernet.

A continuación se detallan las pruebas realizadas. Las primeras pruebas consistieron en evaluar, de la forma mas rápida posible, la interoperabilidad. Luego de confirmada la interoperabilidad, se realizaron algunas pruebas mas exigentes.

Las pruebas se separaron en utilizar el USRP como receptor y utilizarlo como transmisor.

Prueba del transmisor

Debido a las carencias mencionadas anteriormente, la prueba mas sencilla y rápida para probar la comunicación fue configurar la tarjeta WiFi en modo monitor. Esto sirve para recibir todas las tramas en el medio así como para transmitir tramas sin necesidad de asociación como veremos mas adelante.

Para la transmisión de tramas desde el USRP se utilizó el ejemplo `wifi_tx.grc`. Este ejemplo contiene la implementación del transmisor encapsulada en los bloques `OFDM MAC` y `WIFI PHY Hier`. El bloque `OFDM MAC` es la implementación de la capa MAC del estándar y es donde se pueden configurar las direcciones MAC de origen, destino y difusión. El bloque `WIFI PHY Hier` es un bloque jerárquico que encapsula la *flow graph* del transmisor, la cual estudiamos en la sección anterior. En la entrada `app in` del bloque `OFDM MAC` se pueden conectar distintos bloques para generar tráfico. Para esta primer prueba se utilizó el bloque `Message Strobe` (ya es parte del ejemplo), el cual permite generar mensajes de forma periódica y conteniendo un texto dado.

Del lado del USRP la configuración consiste en colocar la dirección IP adecuada en la interfaz a la cual se conecta el mismo. El USRP N210 al iniciar utiliza la IP 192.168.10.2/24. Por lo que se configura la interfaz ethernet de la máquina con la dirección IP 192.168.10.1/24.

```
sudo ifconfig eth0 192.168.10.1/24
```

Además, en la implementación, se debe elegir el canal (en nuestro caso el 1) y colocar el `Sample Rate` en 20MHz (valor por defecto de 802.11g).

Para observar la correcta transmisión de tramas por parte del USRP se eligió un canal con poco tráfico y se configuró la tarjeta WiFi para escuchar todo el tráfico en ese canal. En el dispositivo WiFi se debe agregar una interfaz en modo monitor y obligar a que esta interfaz escuche en el canal 1. El canal que utiliza esta interfaz depende del canal en el cual esté sintonizada la interfaz física correspondiente. La manera mas sencilla para lograr esto es asegurarse que no haya ninguna otra interfaz lógica configurada y asignar el canal deseado a la nueva interfaz. Finalmente ejecutamos la herramienta tcpdump y podemos ver las tramas generadas por el USRP.

```
iw phy phy0 interface add mon0 type monitor
ifconfig mon0 up
ifconfig wlan0 down
iw dev mon0 set channel 1
tcpdump -i mon0
```

Los resultados obtenidos en este caso fueron que es posible ver las tramas generadas por el USRP pero solo si la tasa a la que se transmiten es elevada (se genera una trama cada 5ms). Analizando los registro de la tarjeta WiFi se detectó que la mayoría de las tramas son descartadas por errores en la comprobación del CRC. Se puede utilizar el siguiente comando para ver esta información:

```
cat /sys/kernel/debug/ieee80211/phy0/ath9k/recv
```

En la siguiente tabla se puede ver un resumen de los resultados obtenidos. El bloque `Message Strobe` utilizado permite configurar el tiempo entre mensajes generados. Se puede apreciar que al aumentar la cantidad de mensajes por segundo generados, aumenta la probabilidad de recibir un mensaje correctamente.

Tiempo entre mensajes	Cantidad de mensajes generados	Mensajes correctamente recibidos	Mensajes descartados por error de CRC	Tasa de recepción
10ms	2798	1	2784	0,035 %
5ms	3107	4	3081	0,13 %
1ms	15894	2395	10888	15 %

Una posible explicación del problema es que el emisor implementado no realiza el mecanismo de CSMA por lo que existe una alta tasa de colisiones. Mas allá de haber elegido un canal sin mucho tráfico (ninguna AP en ese canal ni en los cercanos) se detectó que muchos dispositivos generan *Probe Requests* recorriendo todos los canales por lo que no existe ningún canal totalmente libre de tráfico. El aumento de las recepciones al aumentar la tasa de transmisión se puede explicar

por el hecho de que al transmitir mas seguido, el resto de los dispositivos (que si realizan el CSMA) observan el medio ocupado mas tiempo y retasan sus transmisiones.

Prueba del receptor

Para la recepción se utilizó el ejemplo `wifi_rx.gr`. Este ejemplo consiste en una *flow graph* de GNU Radio Companion donde se puede ver la implementación del receptor separada en bloques y donde las tramas recibidas son pasadas al bloque `Wireshark Connector` para generar un archivo de trazas `.pcap`.

Del lado de la tarjeta WiFi se utilizó la biblioteca `pcap` para realizar inyección de paquetes y de esta manera resolver el problema de que la tarjeta solo transmite si esta asociada a una red AdHoc o a un AP. De esta manera se pueden generar tramas de broadcast fácilmente para que la tarjeta no espere un ACK de las tramas transmitidas. Además, se debe configurar la tarjeta inalámbrica para que transmita las tramas usando una modulación y codificación que la implementación del receptor acepte. Para los experimentos se utilizó un *data rate* de 6Mbps que equivale a modulación BPSK con un *code rate* de $\frac{1}{2}$.

La configuración del router en este caso es muy similar:

```
iw phy phy0 interface add mon0 type monitor
ifconfig mon0 up
ifconfig wlan0 down
iw dev mon0 set channel 1
iw dev mon0 set bitrates legacy-2.4 6
```

En este caso las tramas son generadas por la aplicación packetspammer la cual se compiló especificamente para el hardware y software utilizado. Se configuró el tamaño de los mensajes en el menor posible (52 bytes de datos) y se probó con tiempos entre tramas similares a la prueba anterior.

En la siguiente tabla vemos los resultados obtenidos. En este caso, gracias a los mensajes de debug que genera la implementación podemos determinar dos casos en los que se descartan los mensajes, al decodificar el campo SIGNAL de la trama, si el bit de paridad es incorrecto y si el cómputo del CRC no coincide con el contenido en el cabezal MAC de la trama.

Tiempo entre mensajes	Cantidad de mensajes generados	Mensajes correctamente recibidos	Mensajes descartados por error de CRC	Mensajes descartados por bit paridad	Tasa de recepción
40ms	3000	2534	38	255	84 %
20ms	3000	2245	58	220	75 %
10ms	3000	2010 Para	95	246	67 %
5ms	3000	1824	140	174	61 %
1ms	3000	543	129	549	18 %

Se puede apreciar que en este caso los resultados son totalmente opuestos. Se obtiene una tasa de recepción mucho mas alta y la misma disminuye al aumentar la tasa a la que se generan las tramas. Esto se debe a que en este caso el

transmisor si utiliza el mecanismo CSMA, por lo que las colisiones son mucho menores. El problema con tasas de tráfico altas en este caso se deben a carencias en la implementación del receptor. El receptor no logra decodificar todas las tramas en el tiempo requerido y por lo tanto se pierde la recepción de muchas de estas al no poder detectarlas.

Es importante aclarar que estos experimentos se ejecutaron solo una vez, por lo que no tienen valor estadístico. Debido a la variabilidad del medio lo correcto es realizar los experimentos varias veces.

Comunicación entre aplicaciones

Como segunda prueba, se realizó una comunicación directa entre la tarjeta inalámbrica y el dispositivo USRP. En el caso anterior simplemente se escuchaban las tramas en el medio, los datos de las tramas no eran enviados a capas superiores, en este caso la comunicación se realiza entre aplicaciones. El objetivo de los experimentos es poder generar tráfico usando la herramienta MGEN[8], la cual permite generar distintos tipos de tráfico y medir, por ejemplo, la tasa de pérdida de paquetes.

Para poder realizar esto es necesario configurar las dos puntas de la comunicación de manera especial. Las configuraciones que se detallan a continuación son el resultado final de varias pruebas y análisis de problemas encontrados.

Del lado de la tarjeta inalámbrica (WiFi1) la misma no aceptará tramas que no tengan como dirección destino su dirección MAC. Además no aceptará tramas que provengan de una dirección MAC origen de un dispositivo al cual no está asociado. Debido a que no es posible asociar el dispositivo USRP (USRP1) a la tarjeta, los autores de la implementación en gnuradio sugieren modificar el firmware de la tarjeta para simular esta asociación. Debido a que esto parece ser complejo, lo que se realizó fue utilizar un segundo dispositivo (WiFi2) con una tarjeta inalámbrica el cual se asociará a WiFi1 (lo mas sencillo es crear una red Ad-Hoc) y el dispositivo USRP1 simplemente generará tramas con dirección origen la dirección de WiFi2. Con esto logramos que WiFi1 acepte las tramas de USRP1 y envíe los datos a la capa de red.

A continuación se detalla la configuración del router WiFi1 para crear una red Ad-Hoc. Archivo /etc/config/wireless:

```
config wifi-device 'radio0'
        option type 'mac80211'
        option channel '1'
        option hwmode '11g'
        option path 'platform/ar934x_wmac'
        option htmode 'NONE'
        option log_level '2'

config wifi-iface
        option device 'radio0'
        option network 'adhoc0'
        option mode 'adhoc'
```

```
        option ssid 'comina'
        option encryption 'none'
```

Archivo /etc/config/network:

```
config interface 'adhoc0'
        option proto 'static'
        option ipaddr '192.168.201.1'
        option netmask '255.255.255.0'
```

Además se debe ejecutar el siguiente comando para generar tramas que indiquen que no se generen tramas ACK del lado del receptor. De esta manera el emisor no esperará por ACKs.

```
iw wlan0 set noack_map 0x009
```

También es necesario configurar el bitrate al que deseamos transmitir.

```
iw dev mon0 set bitrates legacy-2.4 6
```

Del lado del USRP se debe configurar el bloque `OFDM MAC` para que las direcciones origen y destino sean las correctas. Se debe usar además el bloque `TUNTAP PDU` el cual crea una interfaz (tap0) por la cual el emisor recibe los datos a enviar y por la cual el receptor envía los datos hacia la aplicación. Esta interfaz se debe configurar con la dirección IP adecuada para simular ser el dispositivo WiFi2. Además es necesario agregar un entrada estática en la tabla de ARP para evitar que el sistema operativo envié mensajes de consulta ARP. Para el caso que el USRP funciona como receptor también se debe modificar la dirección MAC de la interfaz tap0 para que coincida con el destino de las tramas enviadas por la tarjeta. Este es un ejemplo de configuración que se debe realizar en la máquina donde se encuentra el USRP.

```
ifconfig tap0 <IP-Wifi2>
arp -s <IP-Wifi1> <MAC-WiFi1>
ip link set dev tap0 address <MAC-WiFi2>
```

El bloque `OFDM MAC` se debe configurar con:

```
SRC MAC: <MAC-WiFi2>
DST MAC: <MAC-WiFi1>
BSS MAC: <BSSID de la red AdHoc>
```

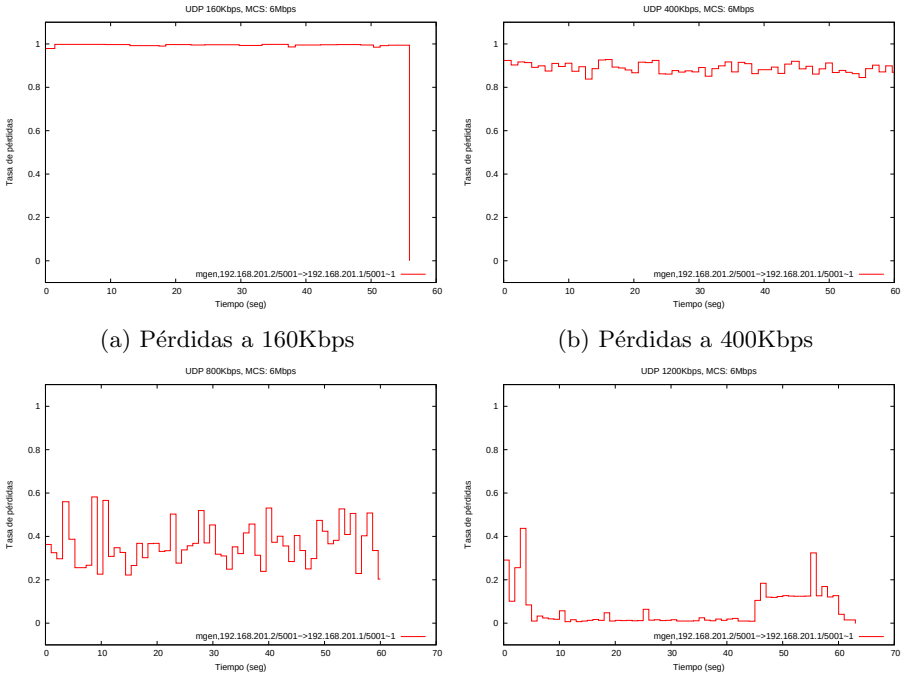
Además de estas configuraciones se debió cambiar parte del código de la implementación para ajustarse a la tarjeta inalámbrica utilizada. El problema es que en todos los casos probados la tarjeta siempre espera una trama con el campo *QoS Control* y el mismo no es agregado por el emisor USRP. Para esto se modificó el módulo ofdm_mac.cc y se agregó este campo y se modificaron las banderas necesarias. Esto además sirvió para generar tramas con la bandera NO-ACK la cual se encuentra dentro de este campo y es necesaria para que el receptor WiFi no genere ACKs. Por último, dependido si la tarjeta está en modo Ad-Hoc o AP es necesario setear una bandera en las tramas, lo cual también debe hacerse en el código.

Finalmente se genera tráfico UDP utilizando la herramienta MGEN, la cual incluye en los datos de los mensajes un número de secuencia con el cual es posible medir la pérdida de mensajes. Además cada mensaje recibido es registrado junto al momento de recepción con lo que se puede medir el throughput.

UDP desde el USRP a la tarjeta WiFi

En esta prueba se genera tráfico UDP a una tasa constante desde el USRP hacia el dispositivo WiFi. Se realizaron varias pruebas variando la tasa entre 160Kbps y 1200Kbps de datos (entre 200 y 1500 paquetes por segundo, cada paquete de 100 Bytes de datos). El *data rate* (la modulación y codificación) utilizado es de 6Mbps (BPSK 1/2). Los dispositivos se encuentran a una distancia de 1 metro y el USRP se configura con una ganancia de transmisión de 15dB.

En la Figura 6 vemos las pérdidas a distintas tasas de generación de tráfico.



(a) Pérdidas a 160Kbps

(b) Pérdidas a 400Kbps

(c) Pérdidas a 800Kbps

(d) Pérdidas a 1200Kbps

Figura 6: Pérdidas variando la tasa de mensajes UDP generados.

Vemos que cuanto mayor sea la tasa de generación de tráfico aumenta la tasa de recepción de mensajes. Como mencionamos anteriormente esto se debe a la falta del mecanismo CSMA. Con una tasa menor a 160Kbps no es posible recibir ningún mensaje.

También vemos que para la tasa mas alta, en un momento aumenta la tasa de pérdidas. Esto se debe a que el transmisor no puede cumplir con los tiempo necesarios para transmitir todos los mensajes, por lo tanto los buffers en gnu-radio comienzan a llenarse y a partir de cierto momento algunos mensajes son

descartados. Por lo tanto, si se intenta aumentar aún mas el tráfico comenzarán a aumentar las pérdidas.

Una de las desventajas de este problema es que no podemos generar tráfico a una tasa lo suficientemente alta (por ej. mayor a 6Mbps) como para evaluar la transmisión a distintas modulaciones y poder observar el aumento en la velocidad.

Por último, en la Figura 7 vemos el throughput obtenido cuando se genera tráfico a una tasa de 1200Kbps. Vemos que se logra un throughput muy cercano a la tasa a la que se transmite.

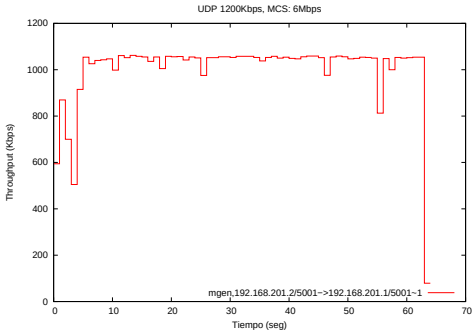
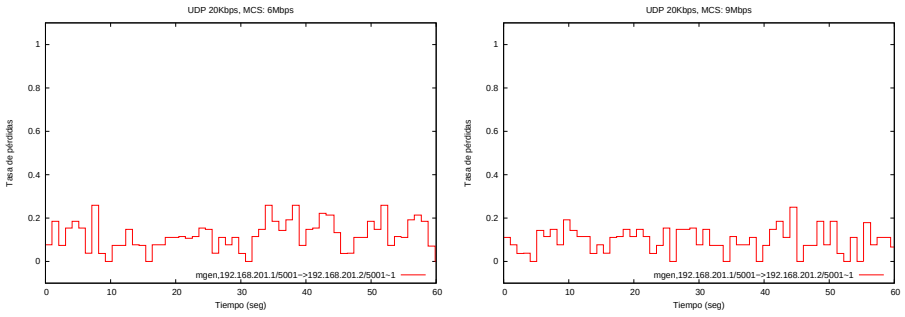


Figura 7: Throughput a 1200Kbps

Tráfico UDP desde la tarjeta WiFi al USRP

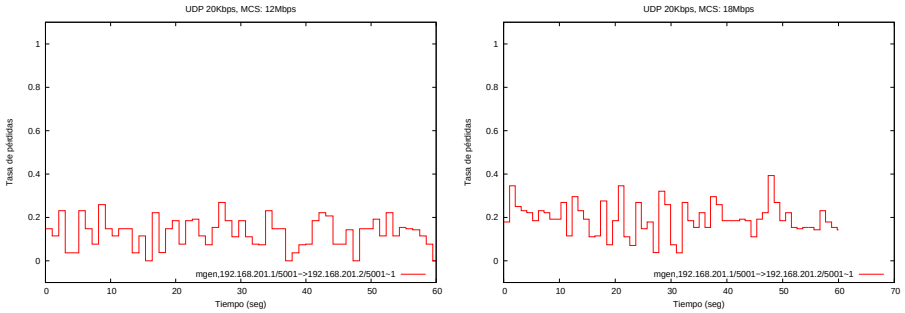
en este caso se genera tráfico UDP a una tasa constante de 20Kbps de datos (25 paquetes por segundo, cada paquete de 100 Bytes de datos) y variamos el *data rate* (es decir variamos la modulación y codificación) de los mensajes enviados entre 6 (BPSK 1/2), 9 (BPSK 3/4), 12 (QPSK 1/2), 18 (QPSK 3/4) y 24 (16-QAM) Mbps. Los dispositivos se encuentran a una distancia de 1 metro, el router utiliza su potencia máxima (19dBm) y el USRP una ganancia de recepción de 5dB.

En la Figura 8 mostramos los resultados del conteo de mensajes UDP perdidos. No se incluye el caso de 24Mbps (modulación 16-QAM) porque las perdidas son totales. Esto se debe, como se mencionó anteriormente, a que la implementación del receptor utilizada no tiene un algoritmo de estimación del canal lo suficientemente bueno para aceptar modulaciones que utilicen variaciones en la amplitud para codificar información. En la figura vemos que las pérdidas son bajas para todos los esquemas de modulación evaluados, aumentando levemente para el caso de 18Mbps.



(a) Pérdidas a 6Mbps

(b) Pérdidas a 9Mbps



(c) Pérdidas a 12Mbps

(d) Pérdidas a 18Mbps

Figura 8: Pérdidas variando la modulación y codificación.

En las siguientes figuras se puede ver la forma de la constelación recibida para las modulaciones BPSK, QPSK y 16-QAM. Se observa claramente por qué en el caso de QAM no es posible decodificar ninguna trama.

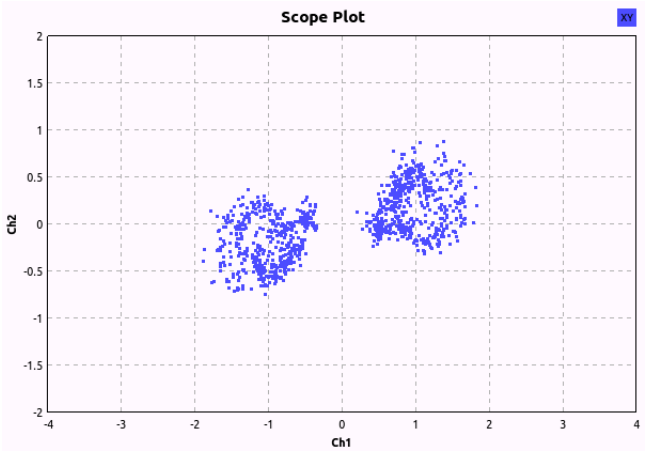


Figura 9: Constelación utilizando BPSK (6Mbps).

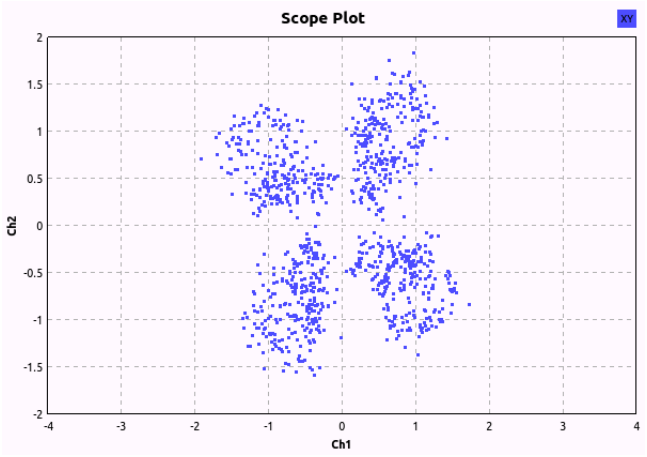


Figura 10: Constelación utilizando QPSK (12Mbps).

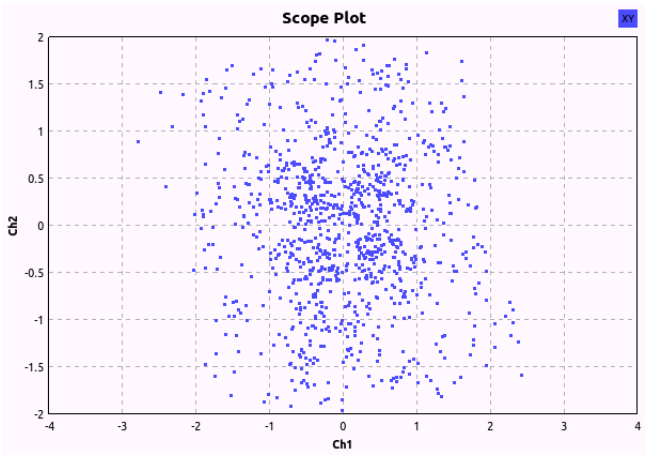
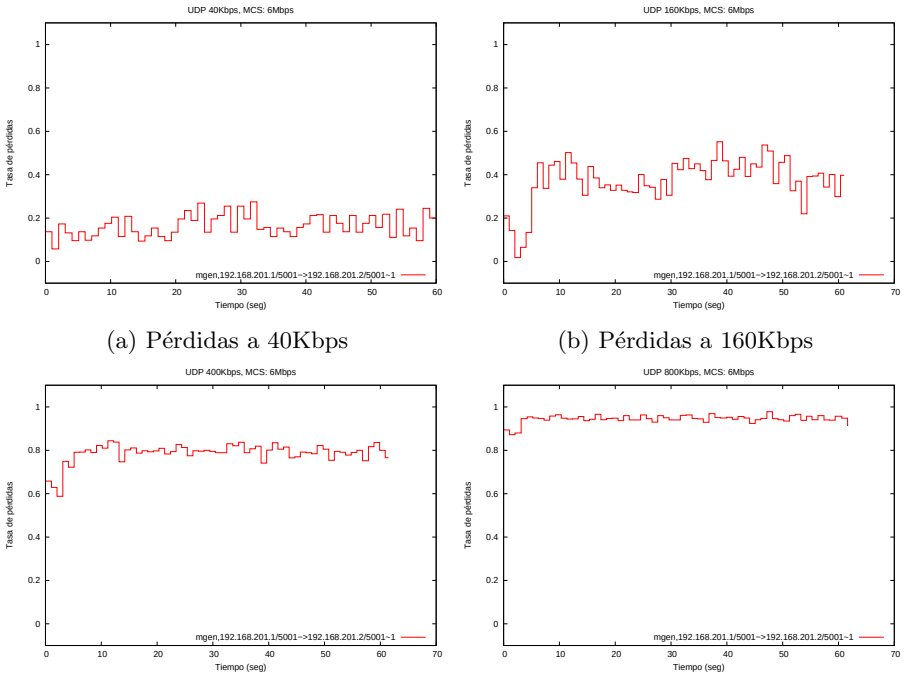


Figura 11: Constelación utilizando 16-QAM (24Mbps).

También se realizaron pruebas aumentado la cantidad de mensajes generados por segundo, obteniendo los mismos resultados que para la prueba utilizando el packetspammer: disminuye considerablemente el rendimiento del receptor al aumentar la tasa de mensajes. Para todos los casos se generó trafico durante el mismo intervalo de tiempo.

Es importante mencionar que se experimentó también con trafico TCP. Sin embargo esto no funciona debido a que al ser tan alta la probabilidad de pérdidas desde el USRP a la tarjeta WiFi nunca se puede completar la conexión.



(a) Pérdidas a 40Kbps

(b) Pérdidas a 160Kbps

(c) Pérdidas con 400Kbps

(d) Pérdidas a 800Kbps

Figura 12: Pérdidas variando la tasa de mensajes UDP generados.

Conclusiones

En este trabajo se experimentó con la implementación en GNU Radio de un sistema transmisor/receptor 802.11. Se estudió y documentó la implementación y posteriormente se realizaron pruebas de interoperabilidad del sistema con una tarjeta 802.11 comercial. De las pruebas realizadas se puede concluir que tanto el transmisor como el receptor cumplen con el estándar siendo posible comunicar implementaciones distintas.

La implementación cuenta con una serie de limitantes, siendo una de las mas importantes no contar con el mecanismo CSMA para el sensado del medio. Esto conlleva a que se generen colisiones y como consecuencia altas tasas de pérdida en las transmisiones. Una de los problemas mas importantes que genera esta limitante es la imposibilidad de realizar una comunicación TCP de manera sencilla. Otra limitante importante, no solo de la implementación sino mas general del SDR, es la latencia de procesamiento. Como se observó en varios experimentos el sistema no pudo soportar tráfico superior a 1200Kbps, tanto en la transmisión como en la recepción. Esto esta muy por debajo del tráfico soportado por hardware 802.11 comercial. Parece un área de investigación importante estudiar como mejorar estos tiempos de procesamiento ya sea utilizando hardware mas potente o técnicas de software mas avanzadas.

Bibliografía

[1] Bastian Bloessl, Michele Segata, Christoph Sommer, and Falko Dressler. An IEEE 802.11a/g/p OFDM Receiver for GNU Radio. In *ACM SIGCOMM 2013, 2nd ACM SIGCOMM Workshop of Software Radio Implementation Forum (SRIF 2013)*, pages 9–16, Hong Kong, China, August 2013. ACM.

[2] P Fuxjäger, A Costantini, D Valerio, P Castiglione, G Zacheo, T Zemen, and F Ricciato. Ieee 802.11 p transmission using gnuradio. In *6th Karlsruhe Workshop on Software Radios (WSR)*, pages 1–4. Citeseer, 2010.

[3] Bastian Bloessl. An ieee 802.11 a/g/p transceiver for gnu radio. https://github.com/bastibl/gr-ieee802-11, 2013.

[4] Chia-Horng Liu. On the design of ofdm signal detection algorithms for hardware implementation. In *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE*, volume 2, pages 596–599. IEEE, 2003.

[5] IEEE Std 802.11-2012, IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: WLAN MAC and PHY specifications. Online, March 2012.

[6] Bastian Bloessl, Michele Segata, Christoph Sommer, and Falko Dressler. Decoding IEEE 802.11a/g/p OFDM in Software using GNU Radio. In *19th ACM International Conference on Mobile Computing and Networking (MobiCom 2013), Demo Session*, pages 159–161, Miami, FL, October 2013. ACM.

[7] Bastian Bloessl, Michele Segata, Christoph Sommer, and Falko Dressler. Towards an open source ieee 802.11 p stack: A full sdr-based transceiver in gnu radio. In *VNC*, pages 143–149, 2013.

[8] Multi-generator (mgen). http://www.nrl.navy.mil/itd/ncs/products/mgen.