ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ESTUDIO, PRUEBAS Y SIMULACIÓN DEL ESTÁNDAR IEEE 802.11AC BASÁNDOSE EN MU-MIMO (MIMO MULTIUSER)

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

LLANGARÍ ARIZO FELIPE ANDRÉS

MEJÍA QUISHPE ENRIQUE JAVIER

DIRECTOR: M.Sc. RICARDO XAVIER LLUGSI CAÑAR

CO-DIRECTOR: M.Sc. CARLOS ROBERTO EGAS ACOSTA

Quito, marzo 2016

DECLARACIÓN

Nosotros, Llangarí Arizo Felipe Andrés y Mejía Quishpe Enrique Javier, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; ya que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normativa institucional vigente.

Llangarí Arizo Felipe Andrés

Mejía Quishpe Enrique Javier

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por Llangarí Arizo Felipe Andrés y Mejía Quishpe Enrique Javier bajo nuestra supervisión.

M.Sc. Ricardo Llugsi
DIRECTOR DEL PROYECTO

M.Sc. Carlos Egas
CO DIRECTOR DEL PROYECTO

AGRADECIMIENTOS

Agradezco a mi familia, a mi novia, y a todos los amigos que han estado durante este proyecto.

Felipe Llangarí A.

AGRADECIMIENTOS

A todas y cada una de las personas que de una u otra forma contribuyeron al desarrollo de este proyecto.

Enrique Mejía Q.

DEDICATORIA

Este triunfo es dedicado para Marina, José, Pepe, Luz y Paola.

Felipe Llangarí A.

DEDICATORIA

A Wilman, Yolanda, Estefania,
Jhonn,Stalyn, Sarai y Sebastian.

Enrique Mejía Q.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La forma de comunicarse inalámbricamente en un ambiente de área local ha evolucionado constantemente, de tal manera que puedan ser utilizadas por aplicaciones que demandan una mayor capacidad de red. Es así que el IEEE (*Institute of Electrical and Electronics Engineers*, Instituto de Ingenieros Eléctricos y Electrónicos) ha desarrollado modificaciones al estándar original IEEE 802.11 para lograr mayores prestaciones y utilidades, publicando la enmienda IEEE 802.11ac.

El presente documento, que consta de cinco capítulos, busca identificar los aspectos más importantes en los que el estándar muestra avances en capacidad, latencia, velocidades, entre otros.

En el capítulo I se detalla el estándar IEEE 802.11ac, características de capa física, de sub-capa MAC (*Medium Access Control*, Control de Acceso al Medio), la técnica de transmisión MU-MIMO (*MIMO MULTIUSER*, MIMO Multi-usuario) entre otras particularidades que establece el estándar.

En el capítulo II se realiza la simulación del estándar IEEE 802.11ac utilizando el software computacional MATLAB®. Se analiza el modelamiento de canal, se describen las funciones, herramientas y parámetros empleados en la programación.

En el capítulo III se desarrollan un conjunto de pruebas ejecutadas en escenarios diferentes, en donde se analizan las velocidades que se pueden alcanzar utilizando el AP (*Access Point*, Punto de Acceso) Cisco AIR-CAP3602E-A-K9 integrado con el módulo Cisco AIR-RM3000AC-A-K9 en el estándar IEEE 802.11ac.

En el capítulo IV se presentan los resultados obtenidos en la parte práctica y se comparan con los datos teóricos obtenidos en el capítulo II.

Finalmente, en el capítulo V se muestran las conclusiones y recomendaciones que se obtuvieron durante la ejecución del proyecto.

PRESENTACIÓN

Con los avances de las comunicaciones inalámbricas, se ha buscado conseguir transmisiones que proporcionen mayores velocidades, brindar servicio a más usuarios y radios de cobertura más amplios. Cada cierto tiempo la industria ha lanzado sucesivas modificaciones con crecientes tasas de transmisión de datos y capacidades, por lo tanto, resulta importante conocer beneficios y ventajas de los nuevos estándares.

En la actualidad se cuenta el estándar IEEE 802.11ac, conocido también como la quinta generación WiFi, que es considerado el siguiente paso en la evolución de las comunicaciones inalámbricas. Operando en la banda de 5 GHz es capaz de alcanzar velocidades de hasta 1 Gbps añadiendo canales de RF (*Radio Frecuency*, Radio Frecuencia) más amplios, modulación de alta densidad (por ejemplo 256-QAM (*Quadrature Amplitude Modulation*, Modulación de Amplitud en Cuadratura)) y técnicas mejoradas de administración de ancho de banda (por ejemplo RTS (*Ready To Sent*, Listo para enviar), CTS (*Clear To Send*, Libre para enviar)). La técnica MU-MIMO (*Multiuser Multiple-Input Multiple-Output,* Múltiples Entradas-Múltiples Salidas Multiusuario) también está considerada dentro del estándar, donde un AP permite la transmisión simultánea a varios usuarios.

El estándar 802.11ac ofrece *VHT* (*Very High Throughput,* Muy Alto Rendimiento), para alcanzar este beneficio se requieren modificaciones sustanciales en la capa física y en la sub-capa MAC. Todo esto permite contar con servicios tales como: *streaming* de vídeo, búsquedas de bases de datos, transferencia de archivos y voz sobre Wi-Fi que son aplicaciones que demandan cada vez mayores capacidades de una red inalámbrica.

Con el presente proyecto se busca comprobar las velocidades efectivas teóricas utilizando el AP Cisco AIR-CAP3602E-A-K9 y el módulo AIR-RM3000AC-A-K9. Además, comprobar y analizar las características del estándar en un ambiente de simulación. Actualmente, el estándar IEEE 802.11ac es nuevo y poco implementado en el país, por lo que el presente estudio ayudará a tener una amplia

visión de su funcionamiento y operación del mismo; y así, ser un aporte en futuras implementaciones a gran escala de este tipo de tecnología.

CAPÍTULO I

ESTÁNDAR IEEE 802.11

1.1 ESTÁNDAR 802.11

La especificación IEEE 802.11 surge en el año de 1997, es un estándar que define las características de una WLAN (*Wireless Local Área Network*, Red de Área Local Inalámbrica). Este estándar se inicia junto con el grupo WECA (*Wireless Ethernet Compatibility Alliance*, Alianza de Compatibilidad Ethernet Inalámbrica), el propósito de este grupo era el de garantizar la compatibilidad entre dispositivos de diferentes vendedores que utilizan 802.11. WECA desarrolló un test de interoperabilidad Wi-Fi (*Wireless Fidelity,* Fidelidad Inalámbrica*)* y actualmente a la alianza se la conoce como Wi-Fi.

1.1.1 CONCEPTOS BÁSICOS

En comunicaciones inalámbricas se utilizan términos y conceptos particulares como:

- **Estación:** Componente que se conecta al medio inalámbrico.
- **BSS (*Basic Service Set,* Conjunto de Servicios Básicos):** Conjunto de estaciones que se comunican entre sí mediante un procedimiento específico. Cuando todas las estaciones en el BSS son estaciones móviles y no hay conexión a una red cableada el BSS se denomina IBSS (*Independent Basic Service Set*, Conjunto de Servicios Básicos Independiente). Un IBSS generalmente es de naturaleza temporal con un número pequeño de estaciones que es creado para un propósito en particular. Cuando un BSS incluye AP, el BSS se denomina BSS de infraestructura.

- **ESS (*Extended Service Set,* Conjunto de Servicios Ampliados):** Se puede interconectar un conjunto de BSSs mediante un DS (*Distribution System*, Sistema de Distribución) para dar lugar a un conjunto de servicios ampliados. Cada BSS tiene un AP que presenta la funcionabilidad de una estación y permite el acceso al sistema de distribución.

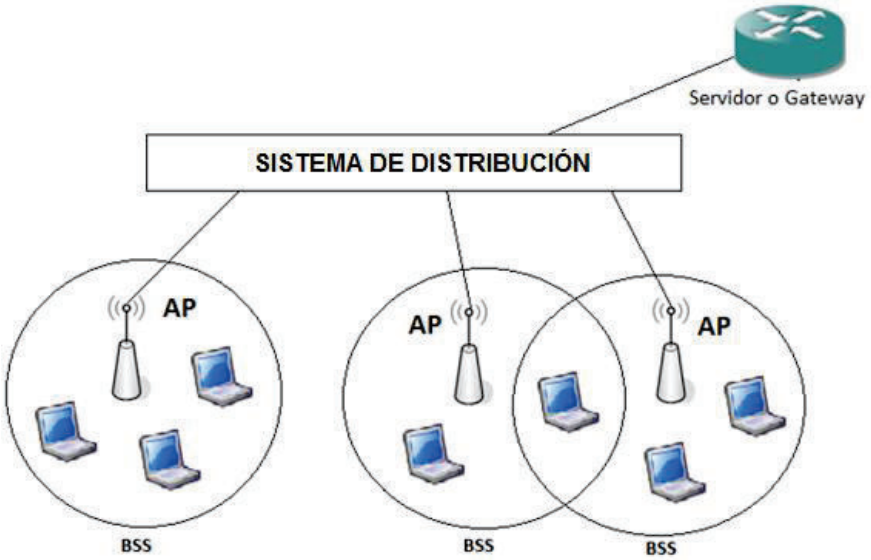


Figura 1.1 Conjunto de servicios extendidos (ESS)

1.1.1.1 Servicios de Distribución

En el proceso de conexión a una red inalámbrica se establecen los siguientes términos:

- **Asociación:** La estación establece una conexión lógica con el punto de acceso.
- **Re-asociación:** Una estación deja la asociación con un AP para asociarse con otro.
- **Des-asociación:** El servicio que permite a cualquiera de las partes, AP o estación móvil, a terminar la asociación.
- **Distribución:** Los datos se llevan desde el origen hasta el destino. El AP utiliza el servicio de distribución para determinar si la trama debe ser enviada dentro de su BSS o si la trama debe ser encaminada para la entrega a una estación asociada con un AP o red diferente.

- **Integración:** Servicio que permite la traslación de una red WLAN a otra no 802.11

1.1.1.2 Servicios de Estación

- **Autenticación:** Para que una estación pueda unirse a la red debe pasar por una serie de pruebas que permitan saber si quién se quiere conectar es quien dice ser. El estándar 802.11 brinda dos tipos de autenticación:

    - OSA (*Open System Authentication,* Autenticación Abierta): Cualquier estación que quiera unirse a la red será aceptado.
    - SKA (*Shared Key Authentication,* Autenticación de llave compartida): Para poder autenticarse en la red, la estación debe saber la clave de acceso.

- **Des-autenticación:** El proceso cuando una estación deja la red.
- **Privacidad:** Implica la seguridad de los datos.
- **Entrega de datos:** La estación transmite y recibe datos.

1.1.2 EVOLUCIÓN DEL ESTÁNDAR IEEE 802.11

La necesidad de mayores velocidades de transmisión, mecanismos de seguridad más eficientes, añadir QoS (*Quality of Service*, Calidad de Servicio), proporcionar una mejor interoperabilidad, ha motivado el desarrollo de nuevos estándares para redes inalámbricas que, tomando como base el estándar 802.11, agregan características que satisfagan las necesidades de los usuarios.

Estos estándares han sido desarrollados por los respectivos grupos de trabajo 802.11 de la IEEE, los cuales verifican si los estándares propuestos cumplen los siguientes criterios: mercado potencial grande, compatibilidad y viabilidad técnica y económica; la Figura 1.2 presenta la evolución de los estándares IEEE.

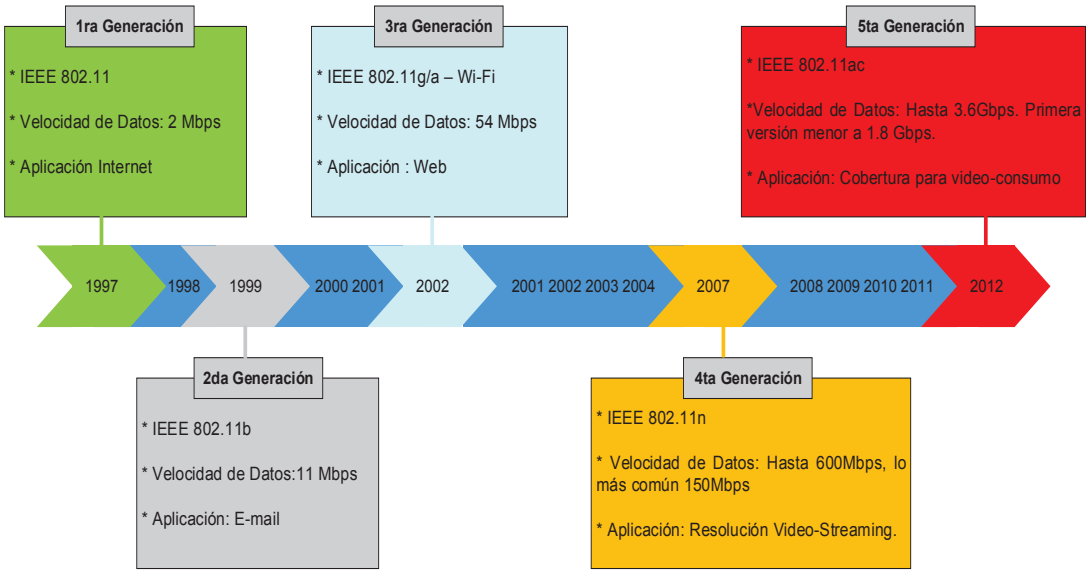


Figura 1.2 Evolución de IEEE 802.11 [1]

1.1.2.1 **IEEE 802.11.**

El estándar IEEE 802.11 ofrece tasas de transmisión de datos máximas de 1 y 2 Mbps. Presenta tres alternativas para la capa física: DSSS (*Direct Sequence Spread Spectrum*, Espectro Ensanchado por Secuencia Directa), FHSS (*Frecuency Hopping Spread Spectrum,* Espectro Ensanchado por Salto de Frecuencia) y una técnica por infrarrojo. Opera en la banda de los 2.4 GHz.

1.1.2.2 **IEEE 802.11b.**

Este estándar opera en la banda no licenciada de 2.4 GHz con velocidades de transmisión de 5.5 y 11 Mbps, tiene un alcance teórico hasta 300 m, y en ambiente interno de 30 a 50 metros con bajo ruido [2].

Utiliza un esquema DSSS y dispone de tres diferentes tipos de modulación BPSK (*Binary Phase Shift Keying*, Desplazamiento de Fase Binaria), QPSK (*Quadrature Phase Shift Keying,* Desplazamiento de Fase en Cuadratura), CCK (*Complementary Code Keying,* Código Complementario) que dependen de la tasa de velocidad máxima.

1.1.2.3 IEEE 802.11a

Opera en la banda de 5 GHz y su velocidad máxima de transmisión es de 54 Mbps. Utiliza OFDM (*Orthogonal Frecuency Division Multiplexing,* Multiplexación por División de Frecuencia Ortogonales) y diferentes técnicas de modulación: BPSK, QPSK Y QAM para las diferentes velocidades.

1.1.2.4 IEEE 802.11g

Este estándar combina características de sus predecesores, por ejemplo: utiliza la banda de 2.4 GHz al igual que 802.11b y un esquema de modulación OFDM como 802.11a. Puede alcanzar velocidades teóricas hasta 54 Mbps.

1.1.2.5 IEEE 802.11n

El estándar 802.11n se desarrolla sobre los estándares anteriores añadiendo técnicas MIMO para alcanzar velocidades de datos significativamente altas de hasta 600 Mbps. Para aquello se requiere de un sistema de múltiples antenas tanto en el transmisor como en el receptor; trabaja utilizando de uno a cuatro *spatial streams*[1] con un ancho de banda de canal de 40 MHz. Utiliza los mismos tipos de modulación que 802.11a y 802.11g. Su operación es en las bandas de 2.4 y 5 GHz.

1.1.3 COMPARACIÓN DE ESTÁNDARES

En la siguiente tabla se detalla las características de los estándares 802.11:

Estándar IEEE	802.11b	802.11g	802.11a	802.11n	802.11ac
Máxima Velocidad de datos (Mbps)	11	54	54	>100	>500 (Canales de 80 Mhz)
Frecuencia	2.4 GHz	2.4GHz	5GHz	2.4GHz y 5GHz	5 GHz
Ancho de Canal (MHz)	20	20	20	20 y 40 (40 es opcional)	20,40,80,160, y 80 + 80
Tecnología	SISO	SISO	SISO	MIMO	MIMO/MU-MIMO

[1] Spatial Stream: Flujo espacial producto de la división de una secuencia de datos determinado por el número de antenas.

Estándar IEEE	802.11b	802.11g	802.11a	802.11n	802.11ac
Técnica de Transmisión	DSSS	DSSS/OFDM	OFDM	(OFDM)	(OFDM)
Número Máximo de secuencias espaciales	1	1	1	4	8
Técnica Beamforming	No	No	No	Si	Si

Tabla 1.1 Comparación de estándares inalámbricos

1.2 IEEE 802.11ac

En el año de 2008 se forma el grupo de trabajo TGac (*Task Group 802.11ac*, Grupo de Trabajo 802.11ac) que empieza a desarrollar enmiendas sobre 802.11. Al igual que los desarrollos anteriores dentro de las redes de acceso inalámbrico, 802.11ac está diseñado para ser totalmente compatible con los estándares anteriores, por lo que 802.11ac se posiciona como el sucesor de 802.11n.

En particular, la norma prevee un rendimiento máximo de al menos 500 Mbps para un único usuario, y al menos 1Gbps en el caso de múltiples usuarios. Esto representa un aumento de cinco veces la tasa máxima alcanzable (por usuario) en comparación con la enmienda anterior 802.11n [3]. Además, es el primer estándar en usar la técnica Multi-usuario de MIMO, la cual permite que un punto de acceso puede transmitir hacia múltiples clientes a la vez. Las características del estándar 802.11ac son las siguientes:

1.2.1 TRANSMISIÓN BEAMFORMING

Esta tecnología permite ajustar las señales de radiofrecuencia y direccionarlas por el mejor camino que deberían tomar para alcanzar a un dispositivo cliente. 802.11ac especifica un único método de transmisión *beamforming* que se basa en una retroalimentación explicita para permitir uno o varios usuarios MIMO.

El transmisor, un AP por ejemplo, envía señales a cada uno de los usuarios y se mantiene a la espera de que los receptores proporcionen una retroalimentación explicita con una medida del canal.

Esta información es utilizada para crear una matriz de orientación (Q) que se usará para pre-codificar los datos a transmitir, creando un conjunto de haces (*beams*) dirigidos para optimizar la recepción en uno o varios receptores. La estación que transmite los datos utilizando la matriz de orientación es conocida como 802.11ac *beamformer* (VHT *beamformer*). Por otro lado, la estación que responde con la retroalimentación se la llama VHT *beamformee* [3].

En la Figura 1.3 se muestra el concepto de *MU-Beamfoming,* donde un AP atiende a un conjunto de usuarios mediante la formación de varios haces.

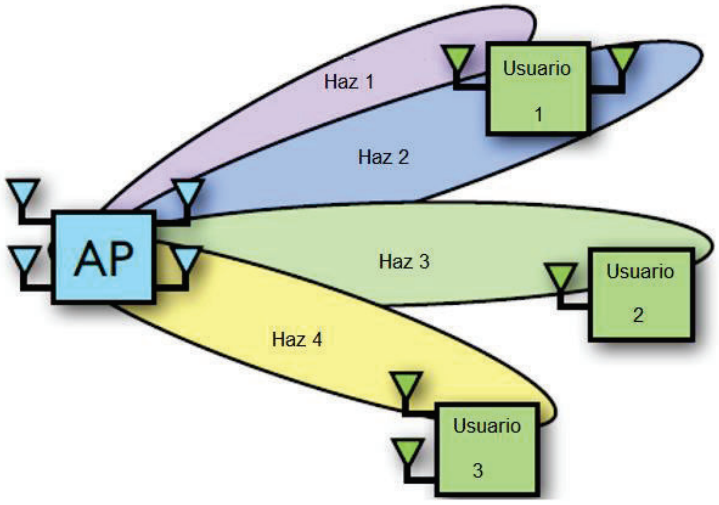


Figura 1.3 Download MU-beamforming [3]

1.2.1.1 **Funcionamiento**

El proceso de *beamforming* para 802.11ac trabaja de la siguiente manera:

- La estación *beamformer* transmite una trama de anuncio NDP (*Null Data Packet,* Paquete de Datos Nulo) que se utiliza para tomar el control del canal

e identificar a los equipos *beamformees*. Las estaciones *beamformees* responderán a la trama NPD mientras que las demás aplazarán el acceso al canal hasta que la secuencia de sondeo se complete.

- El equipo *beamformee* analiza los campos en la trama NDP recibida y calcula una matriz de retroalimentación (*feedback)*, que permite al *beamformer* calcular la matriz de orientación.
- El equipo *beamformer* recibe la matriz de *feedback* y calcula la matriz de orientación para dirigir las transmisiones hacia la estación *beamformee.*

Con la matriz de orientación, el *beamformer* puede transmitir en una determinada dirección como se muestra en la Figura 1.4. Sin *beamforming* la energía es radiada en todas las direcciones más o menos de igual manera. Si el transmisor aplica una matriz de orientación (Q) se enviará la energía en una dirección preferida. La combinación de la matriz de orientación y el canal (H) determina si una señal se hace más fuerte o más débil.

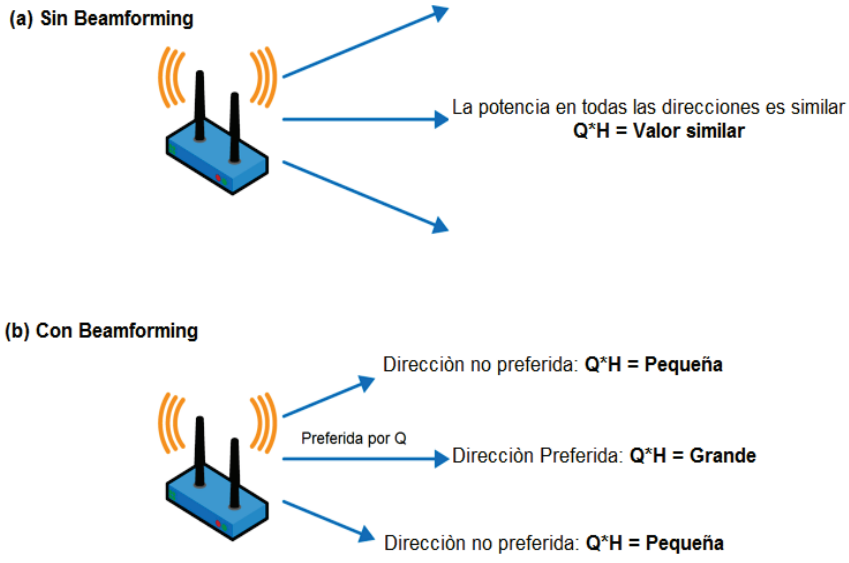


Figura 1.4 Beamforming [4]

1.2.1.2 Formación del haz para un solo usuario (Single-User Beamforming)

En una transmisión *beamforming* se busca formar un solo haz para un único usuario, el propósito es dar forma a una transmisión desde un transmisor hacia un

solo receptor. El *beamformer* envía un NDP. El *beamformee* analiza la trama recibida y calcula una matriz de *feedback* que se reenvía dentro de una trama de respuesta.

La calibración del canal se realiza en un solo procedimiento, en donde el *beamformer* y el *beamformee* miden de manera conjunta el canal para proporcionar los datos necesarios y calcular la matriz de orientación.

1.2.1.3 Formación del haz para varios usuarios (Multi-User Beamforming)

En este caso, el beamformer necesita una respuesta de todos los dispositivos *beamformee.* Cada *beamformee* aporta con información en una matriz de *feedback* y el *beamformer* usa dichas matrices para originar una matriz de orientación.
El procedimiento de "escucha" comienza exactamente como en el caso de un solo usuario, sin embargo, para recuperar la matriz de retroalimentación de cada beamformee es necesario una nueva trama, la trama *Beamforming Report Poll*, con ello se asegura que se recojan las respuestas de todos los beamformee*s* como se muestra en la Figura 1.5. En este ejemplo, el *beamformer* necesita dos tramas *Beamforming Report Poll* para obtener las matrices del segundo y tercer *beamformee* (no se requiere para la primera estación, pues ya se conoce su estado por medio de la trama NDP).

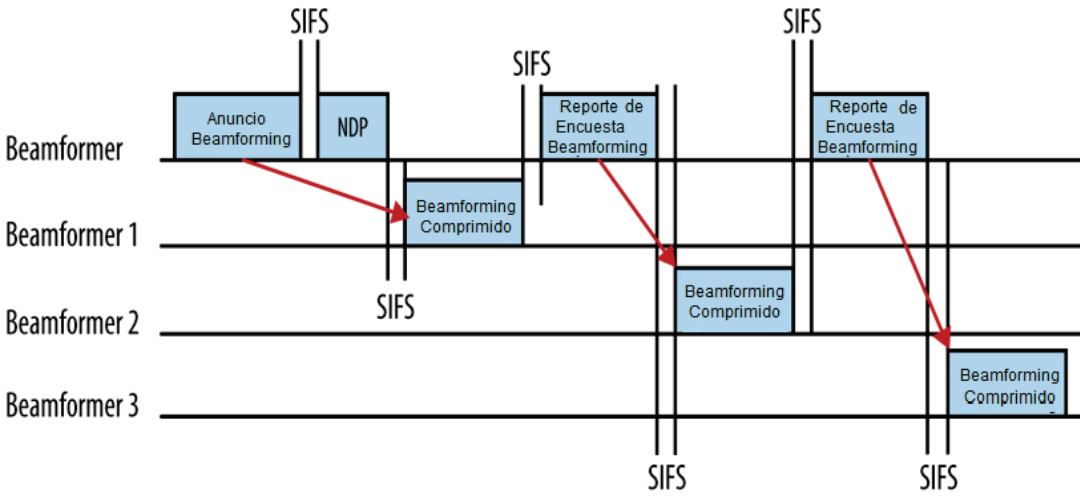


Figura 1.5 Beamforming Multi-Usuario [4]

1.2.2 THROUGHPUT

Cada vez que se establece un nuevo estándar 802.11, la mayoría de las organizaciones y los clientes desean saber que tan rápido va a ser. Con el estándar 802.11n que trabaja en 2.4 y 5GHz se tiene dispositivos finales los cuales no alcanzan las velocidades máximas establecidas, ya sea, por factores inherentes a la transmisión o por cuestiones de fabricación. Para el estándar 802.11ac las velocidades de datos teóricas definidas se presentan en la Tabla 1.2.

Velocidades de datos para Estándar 802.11ac						
MCS[2]	Velocidades de Transmisión (Mbps) (Canal 20 MHz, 1 x SS)		Ancho de Banda de Canal	SS (Secuencias Espaciales)	Velocidades de Transmisión (Mbps) (Canal 160 MHz,8xSS)	
	LGI[3]	SGI			LGI	SGI
0	6.5	7.2	x2.1 para 40 MHZ x4.5 para 80 MHz x9.0 para 160 MHz	x2 para 2 SS x3 para 3 SS x4 para 4 SS x5 para 5 SS x6 para 6 SS X7 para 7 SS X8 para 8 SS	468.0	520.0
1	13.0	14.4			939.0	1040.0
2	19.5	21.7			1404.0	1560.0
3	26.0	28.9			1872.0	2080.0
4	39.0	43.3			2808.0	3120.0
5	52.0	57.8			3744.0	4160.0
6	58.5	65.0			4212.0	4680.0
7	65.0	72.2			4680.0	5200.0
8	78.0	86.7			5616.0	6240.0
9	(86.7)	(96.3)			6240.0	6933.3

Tabla 1.2 Velocidades de datos 802.11ac [5]

Es importante distinguir que el estándar 802.11ac presenta dos fases denominadas *Wave*. *Wave* 1 ofrece 3 veces el rendimiento del estándar 802.11n. Próximamente se espera los productos de la fase *Wave* 2 con los cuales se podrá alcanzar hasta 3.47 Gbps teóricamente.

Un resumen con las características de 802.11ac *Wave* 1, 802.11ac *Wave* 2 y 802.11ac IEEE se presentan a continuación:

[2] MCS Index (*Modulation and Coding Scheme*, Esquema de Codificación y Modulación): determina el tipo de modulación y la tasa de codificación.
[3] GI (Guard Interval, Intervalo de Guarda): El intervalo de guarda entre símbolos ayuda al receptor a superar los efectos multi-trayectoria. L=Long, S=Short.

	802.11ac Wave 1	802.11ac Wave 2	802.11ac Especificación IEEE
Frecuencia de Operación	5 GHz	5 GHz	5 GHz
MIMO	Single User (SU)	Multi User (MU)	Multi User (MU)
Velocidad	1.3 Gbps	2.34 Gbps – 3.47 Gbps	6.9 Gbps
Ancho de banda	20,40, 80 MHz	20,40, 80, 80+80, 160 MHz	20,40, 80, 80+80, 160 MHz
Modulación	256 QAM	256 QAM	256 QAM
Spatial Streams	3	3-4	8

Tabla 1.3 Características Wave 1 y 2 [6]

1.2.3 OFDM

Multiplexación por División de Frecuencias Ortogonal (OFDM) es una técnica de multiplexado multi-portadora donde el ancho de banda disponible es dividido en canales de banda estrecha adyacentes, llamados subportadoras o tonos, y los flujos de datos con altas velocidades es dividido en varios flujos de bajas velocidades que se multiplexan en las subportadoras y transmitidas simultáneamente. Los datos son modulados mediante cualquier técnica de modulación digital como BPSK, QPSK y QAM.

En general la señal OFDM se crea mediante la asignación de los datos digitales a subportadoras, luego, utilizando IFFT (*Inverse Fast Fourier Transform*, Transformada Rápida Inversa de Fourier) para representar la señal en el dominio del tiempo. La IFFT ordena todas las componentes de las señales en elementos sinusoidales individuales de amplitud y frecuencia específicas. Estas operaciones se invierten en el receptor para recuperar los datos originales utilizando la FFT[4] (*Fast Fourier Transform,* Transformada Rápida de Fourier).

[4] FFT convierte una señal del dominio del tiempo al dominio de frecuencia. IFFT convierte una señal del dominio de la frecuencia al dominio del tiempo.

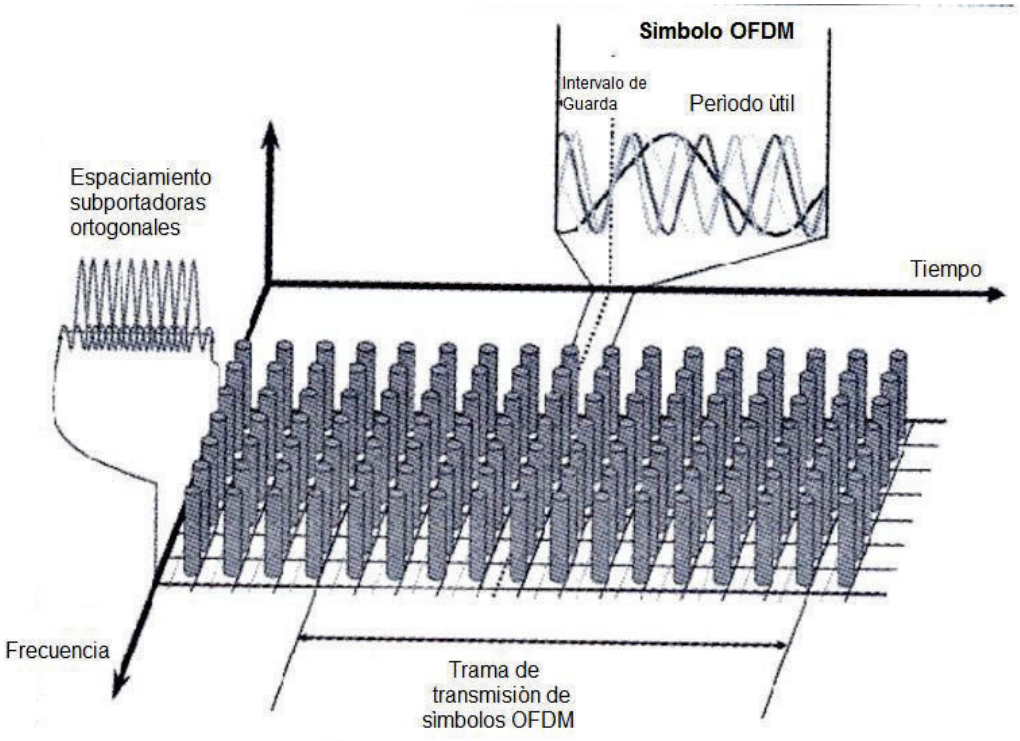


Figura 1.6 Transmisión OFDM [7]

OFDM brinda un mayor rendimiento y beneficios como por ejemplo reduce el ISI (*Inter Symbol Interference*, Interferencia Inter Símbolo), supera los problemas asociados con canales multi-trayectoria, uso eficiente del espectro, entre otras. Esta técnica es utilizada en varias ramas de telecomunicaciones como: estándares WLAN (IEEE 802.11a/g/n/ac), WMAN (*Wireless Metropolitan Area Networks*, Redes de Área Metropolitana Inalámbricas), WIMAX (*Worldwide Interoperability for Microwave Access*, Interoperabilidad Mundial para Acceso por Microondas) LTE (*Long Term Evolution*, Evolución a Largo Plazo), etc.

1.2.3.1 **OFDM en IEEE 802.11ac**

Para 802.11ac, OFDM se la usa tal como lo hace 802.11a y 802.11n, de hecho 802.11ac utiliza las especificaciones de 802.11n con las modificaciones necesarias para alcanzar sus objetivos.

El ancho de banda se divide en subportadoras OFDM, cada una con un ancho de banda de 312.5 KHz [7]. Cada subportadora es utilizada como una transmisión independiente y OFDM distribuye los datos de entrada entre dichas subportadoras. Pocas subportadoras son reservadas (*pilot carriers*), éstas no llevan información de usuarios y en cambio son usadas como medida del canal.

Ancho de Banda (MHz)	20	40	80	160
FFT Size[5]	64	128	256	512
Número de subportadoras de datos	52	108	234	468
Número de subportadoras pilotos	4	6	8	16
Total número de subportadoras	56	114	242	484
Transmisión de subportadoras	±(1 - 28)	±(2 - 58)	±(2 - 122)	±(6 - 126) ±(130 - 250)

Tabla 1.4 Subportadoras OFDM en 802.11ac [8]

La Figura 1.7 muestra las subportadoras piloto en un canal de 20 MHz. Las subportadoras en la mitad del ancho de banda (DC) y en los extremos superiores e inferiores son anuladas para reducir los problemas en circuitos analógicos de banda base y para evitar interferencias con los canales adyacentes, respectivamente.

Los intervalos de guarda en las señales OFDM proporcionan resistencia al ISI y a los errores de sincronización de tiempo debido a la ortogonalidad de las subportadoras. En los estándares IEEE 802.11, la duración de símbolo es de 4 µseg, 20% de esa duración (800 nseg) es el intervalo de guarda que lleva un prefijo cíclico de la señal [8].

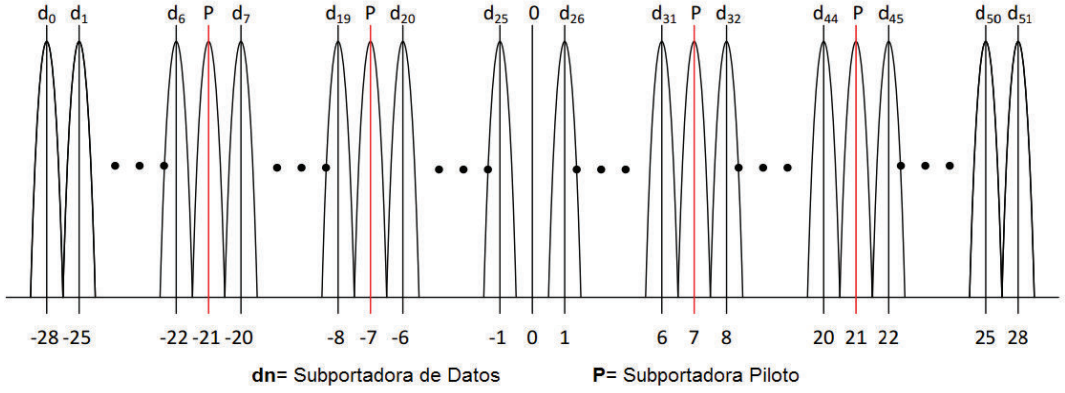


Figura 1.7 Subportadoras OFDM en un canal de 20 MHz en 802.11 [8]

[5] FFT Size: Define el número de subportadoras que existen en un determinado ancho de banda de canal

1.2.4 MIMO

Se define como una tecnología de comunicación en donde el transmisor y el receptor poseen varias antenas, con lo cual se logra incrementar la eficiencia espectral de dicha comunicación inalámbrica. MIMO toma ventaja de la propagación multi-camino o multi-trayecto para mejorar el rendimiento del sistema, ofrece un incremento significativo en el *throughput* y alcance de la comunicación sin ancho de banda y potencia de transmisión adicional.

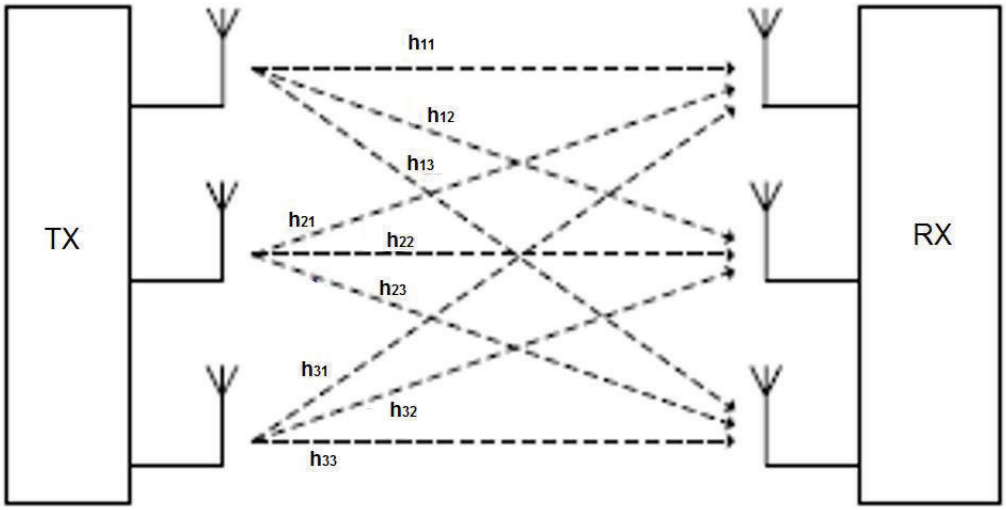


Figura 1.8 Sistema MIMO 3x3

Existen diferentes sistemas MIMO que se pueden utilizar. Estas son: SISO (*Single Input Single Output,* Única Entrada Única Salida), MISO (*Multiple Input Single Output*, Múltiple Entrada Única Salida) y SIMO (*Single Input Multiple Output*, Única entrada Múltiple Salida). Los modos de acceso al canal de radio, la manera de utilizarlo y su complejidad varían dependiendo del número de antenas que se tenga tanto en el transmisor como en el receptor; de acuerdo a lo expuesto se identifica los siguientes tipos de sistemas:

- **SISO:** Este tipo de sistemas maneja una sola antena transmisora y una sola antera receptora. La ventaja de este tipo de sistemas es su simplicidad, no necesitan métodos de procesamiento en cuanto a diversidad se refiere. La

interferencia y el desvanecimiento tendrá mayor impacto en este tipo de sistemas que en sistemas MIMO.

Figura 1.9 Esquema SISO [9]

- **MISO:** Este tipo de sistema destina múltiples antenas transmisoras y una sola antena receptora, también denominado como diversidad de transmisión. La ventaja de usar MISO es que las múltiples antenas, la codificación y procesamiento ocurre en el transmisor. Esto tiene un impacto positivo sobre el tamaño, costo y duración de la batería en el lado del receptor.

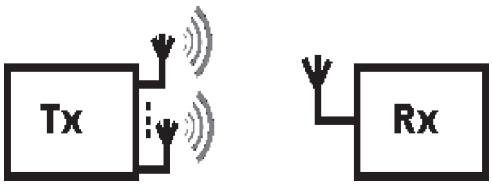


Figura 1.10 Esquema MISO [9]

- **SIMO:** Este tipo de sistema destina una sola antena transmisora y múltiples antenas receptoras. Su implementación es relativamente fácil; sin embargo, posee un alto procesamiento en el lado del receptor limitando el tamaño, el costo y el consumo de batería del mismo. También es conocido como diversidad en recepción.

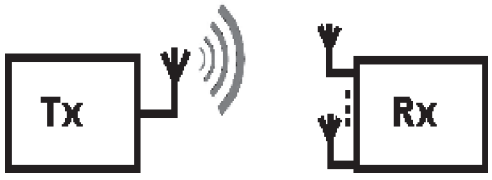


Figura 1.11 Esquema SIMO [9]

1.2.5 TIPOS DE MIMO

1.2.5.1 SU-MIMO

Conocido también como MIMO punto a punto, MIMO de un solo usuario o simplemente MIMO. Es la tecnología descrita en la sección 1.2.4, MIMO consigue grandes mejoras en la eficiencia espectral, fiabilidad y capacidad del canal utilizando técnicas como multiplexación espacial[6] o STC (*Space Time Coding,* Codificación Espacio Temporal).

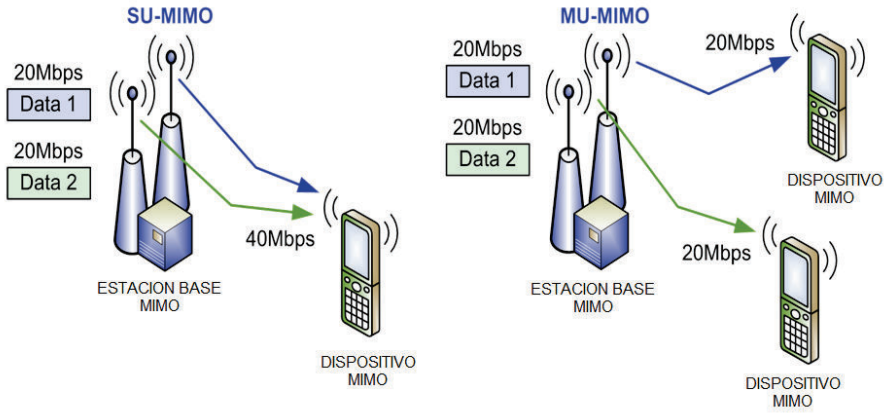


Figura 1.12 SU-MIMO vs MU-MIMO [10]

La diferencia con MU-MIMO radica en que, SU-MIMO no explota la diversidad de usuarios, es decir, un dispositivo transmite varios *spatial streams* al mismo tiempo pero dirigido a un solo usuario a la vez.

1.2.5.2 MU-MIMO

El avance más significativo en 802.11ac es la tecnología MU-MIMO, que permite asignar múltiples flujos espaciales a diferentes usuarios de manera simultánea, lo que aumenta el rendimiento total y la capacidad del sistema WLAN. MU-MIMO basa

[6] Multiplexación Espacial: Transmisión de flujos diferentes de señales codificadas individualmente (conocido como “corriente espacial”) a través de múltiples antenas en paralelo.

su transmisión en las capacidades *beamforming* para establecer hasta cuatro enlaces dirigidos de RF simultáneamente.

Un AP requiere de técnicas de *beamforming* para dirigir la máxima señal hacia el cliente deseado (cliente A) mientras lo reduce al mínimo hacia otro (cliente B) y viceversa. Las especificaciones 802.11ac definen un único método para transmisión *beamforming*, en este método el punto de acceso envía una señal especial de "escucha" a todas las estaciones, las cuales estiman el canal y envían un reporte de vuelta al punto de acceso.

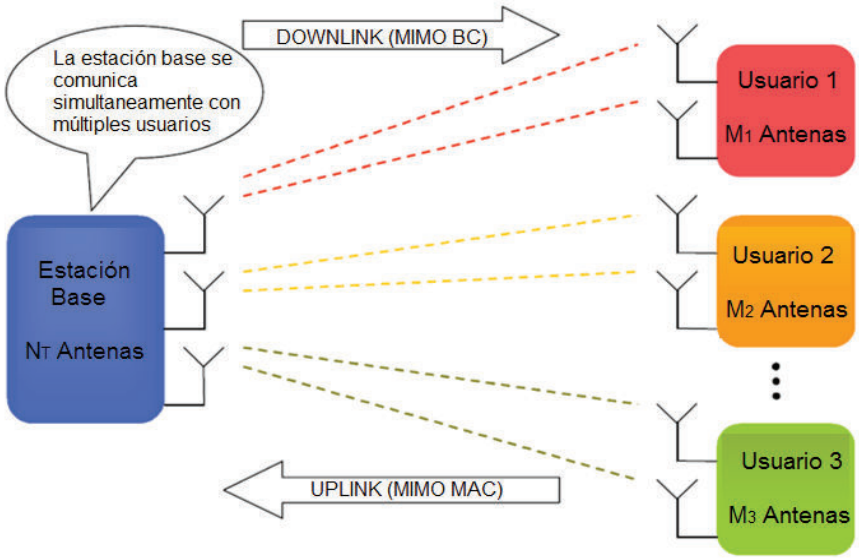


Figura 1.13 Transmisión MU-MIMO

Nuevos términos se asocian con este tema:

- SDMA (*Space-Division Multiple Access*, Acceso Múltiple por División Espacial): Con esta tecnología se logra que varios transmisores envíen señales separadas en el espacio, al mismo tiempo y en la misma banda a varios receptores.

- Enlace descendente MU-MIMO (*Downlink* MU-MIMO): el AP transmite a varios dispositivos receptores a la vez.

Para los enlaces ascendentes (*uplink*) y descendentes (*downlink*) se estable un determinado modo de transmisión:

- MIMO BC *(Broadcast Channels,* Canal de Difusión*)*: Representa el enlace descendente para una topología punto a multi-punto, aquí se incluye SDMA y técnicas avanzadas de pre-codificación.

- MIMO MAC: En este caso la transmisión se realiza desde los usuarios hacia la estación base, el AP actúa como receptor, por lo que también incluirá algoritmos avanzados para la recepción de la señal y el conocimiento del estado del canal. [11]

Una de las principales limitaciones en las velocidades de MU-MIMO es la interferencia inter-usuario, que es causada por *beamformees* que están muy cercanos el uno al otro. La mitigación de la interferencia entre usuarios es un obstáculo importante para las aplicaciones prácticas de MU-MIMO. Se espera que esta característica esté disponible en productos futuros como parte de 802.11ac *wave* 2.

1.2.5.3 **Transmisión MIMO Multi-Usuario**

Una vez que se “escucha” al canal multi-usuario, el AP puede iniciar una transmisión hacia varios dispositivos. El estándar 802.11ac permite hasta 4 transmisiones MU-MIMO simultáneas, cada una de estas transmisiones puede tener su propia velocidad de modulación, codificación y diferente número de *spatial streams.*

Cuando se transmite un conjunto de tramas MU-MIMO, el estándar trata a cada usuario de manera separada. Por ejemplo, en el caso de una transmisión MIMO de dos usuarios, cada secuencia de datos de los usuarios se manejará de manera independiente en el sistema.

La tarea más importante del receptor, en este tipo de transmisión, es la de determinar cómo tomar los *streams* de datos que le "pertenecen" de un conjunto de varios *streams* de todos los usuarios y al mismo tiempo ignorar los datos de los demás usuarios. Al decodificar las transmisiones, un receptor puede procesar no sólo sus tramas, sino también las demás (*streams* interferentes) existentes durante la transmisión. En 802.11ac no se impone ningún requisito para que una estación decodifique o no los flujos de datos interferentes, pero hacerlo reducirán los efectos de interferencia.

1.2.5.4 Acuses de Recibo en MU-MIMO

MIMO Multi usuario en 802.11ac trabaja únicamente en dirección desde el punto de acceso hacia los clientes (*downlink*). El punto de acceso puede transmitir una trama multiusuario a varios receptores al mismo tiempo, pero los *acknowledgement* (acuses de recibo) debe transmitirse individualmente desde los clientes hacia el AP (*uplink*). Cada trama que se transmite en 802.11ac es una trama agregada, por lo que el estándar utiliza el procedimiento de BA[7] (*Block Acknowledgement*, Bloque de Acuse de recibo) definido en 802.11n.

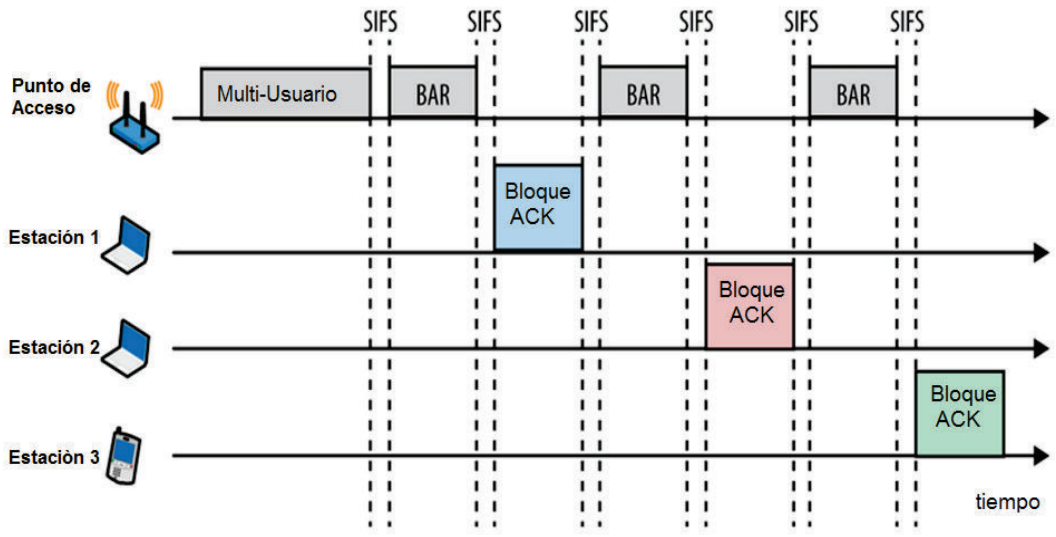


Figura 1.14 Acuse de recibo MU-MIMO [4]

[7] BA: Permite realizar una confirmación para un grupo o bloque de tramas en lugar de una confirmación o ACK por cada trama.

Para graficar de mejor manera lo expuesto, se presenta la Figura 1.14, en donde después de ganar el control del canal, el punto de acceso envía una trama a todos los dispositivos clientes. En el BA, el transmisor mantiene el control del canal e individualmente solicita acuses de recibo donde él lo requiera. En la figura, se puede apreciar que el AP continua con su transmisión de datos con peticiones explicitas de acuses de bloque a cada uno de los tres receptores.

1.2.6 CAPA FÍSICA

1.2.6.1 Frecuencia de Operación

El estándar especifica su operación en la banda no licenciada de 5 GHz, donde existe relativamente menor interferencia y mayor número de canales disponibles en comparación con la banda de 2,4 GHz. Por lo tanto, se espera el aprovechamiento de estas dos características para brindar un mayor rendimiento.

1.2.6.2 Ancho de Banda del Canal

El canal de 20 y 40 MHz en 802.11n se amplía a 80 y 160 MHz en el estándar 802.11ac, permitiendo una mejora significativa, al tener un canal más amplio se aumenta la velocidad de transmisión.

Las características son las siguientes:

- El estándar 802.11ac requiere que todos los dispositivos toleren anchos de banda de 20, 40 y 80 MHz en la banda de 5 GHz, y que la compatibilidad en 160 MHz sea opcional.
- Los canales de 80 MHz son el resultado de la combinación de dos canales de 40 MHz adyacentes que no se superpongan.
- El canal opcional de 160 MHz puede formarse mediante la combinación de dos canales adyacentes o no adyacentes de 80 MHz.

La Figura 1.15 muestra la asignación de canal para la región de los Estados Unidos.

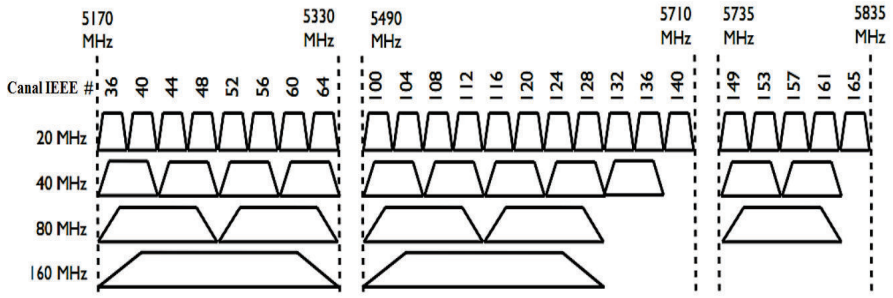


Figura 1.15 Asignación de Canales en EE.UU [3]

1.2.6.3 Codificación, Modulación e Intervalos de Guarda

El índice MCS asocia técnicas de modulación y codificación, los mismos que proporcionan una variada gama de velocidades en función del ancho del canal, el número de flujos espaciales, y el intervalo de guarda. En 802.11ac se establece únicamente 10 índices MCS, como se muestra en la Tabla 1.5, los MCS del 0 al 7 son obligatorios (similar a 802.11n). El MCS 8 y el MCS 9 son nuevos y serán habilitados de acuerdo al avance de la tecnología. Al igual que sus predecesores 802.11ac utiliza un código de corrección de errores. Una de las características principales de un código corrector de errores es añadir información redundante en proporción descrita por el *code rate*, por ejemplo si R=1/2 entonces se transmite un bit de datos de usuario por cada dos bits del canal.

Altos valores de *code rate* tienen más datos y menos redundancia a riesgo de no poder recuperar la información en el caso de existir demasiados errores en la transmisión, pero se aumenta la velocidad de transmisión

Se aumenta el orden de modulación de 64-QAM (802.11n) a 256-QAM, que proporciona un incremento en las tasas de datos en un 33% con respecto al anterior estándar [8]. Esta variación se logra mediante la representación de ocho bits codificados por símbolo (en cada sub portadora) en vez de seis como en el estándar anterior. Sin embargo, es importante señalar, que se requiere un mayor SNR (*Signal to Noise Ratio*, Relación Señal al Ruido) por que los símbolos en el diagrama de constelación están más cerca uno del otro, lo que los vuelve más

susceptibles al ruido. En la Tabla 1.5 se muestran las combinaciones de modulación y code rate con su respectivo índice MCS.

Índice MCS	Modulación	Code Rate (R)
0	BPSK	1/2
1	QPSK	1/2
2	QPSK	3/4
3	16-QAM	1/2
4	16-QAM	3/4
5	64-QAM	2/3
6	64-QAM	3/4
7	64-QAM	5/6
8	256-QAM	3/4
9	256-QAM	5/6

Tabla 1.5 Valores MCS para 802.11ac

Se mantiene el intervalo de guarda 802.11n, el intervalo de guarda de 800 nseg es de tipo mandatorio mientras el de 400 nseg es opcional.

Los valores de MCS que el estándar considera como “no aplica” son mostrados en la Tabla 1.6.

	20 MHz	80 MHz	160 MHz
MCS 6	N/A	3 y 7 SS	N/A
MCS 9	1, 2, 4, 5, 7, y 8 SS	6 SS	3 SS

Tabla 1.6 Valores MCS N/A [12]

1.2.6.4 Estructura de la trama 802.11ac

Una de los puntos más importantes que surgió al momento de construir la trama 802.11ac, fue la necesidad de que ésta sea compatible con los estándares WLAN anteriores. Para cumplir con este requerimiento, el formato de la trama VHT es similar al formato *mixed-mode*[8] de 802.11n y empieza con los mismos campos que los de la trama 802.11a.

[8] Mixed mode: En este modo de operación de 802.11n los paquetes se transmiten con un preámbulo compatible los estándares 802.11a y 802.11g. El resto del paquete tiene un nuevo formato de secuencia.

Otro detalle a tomar en cuenta es la necesidad de permitir transmisiones MU-MIMO: el preámbulo debe ser capaz de describir el número de *spatial streams* con el que se va a trabajar y permitir que varios receptores puedan prepararse para recibir dichas tramas. Para alcanzar este requerimiento se necesita de una nueva cabecera, ya que el campo de HT-SIG (*High Throughput Signal Field*, Campo de Señalización de Alto Rendimiento) usado en el estándar 802.11n no se puede aplicar a los nuevos tamaños de ancho de canal o un gran número de *spatial streams*.

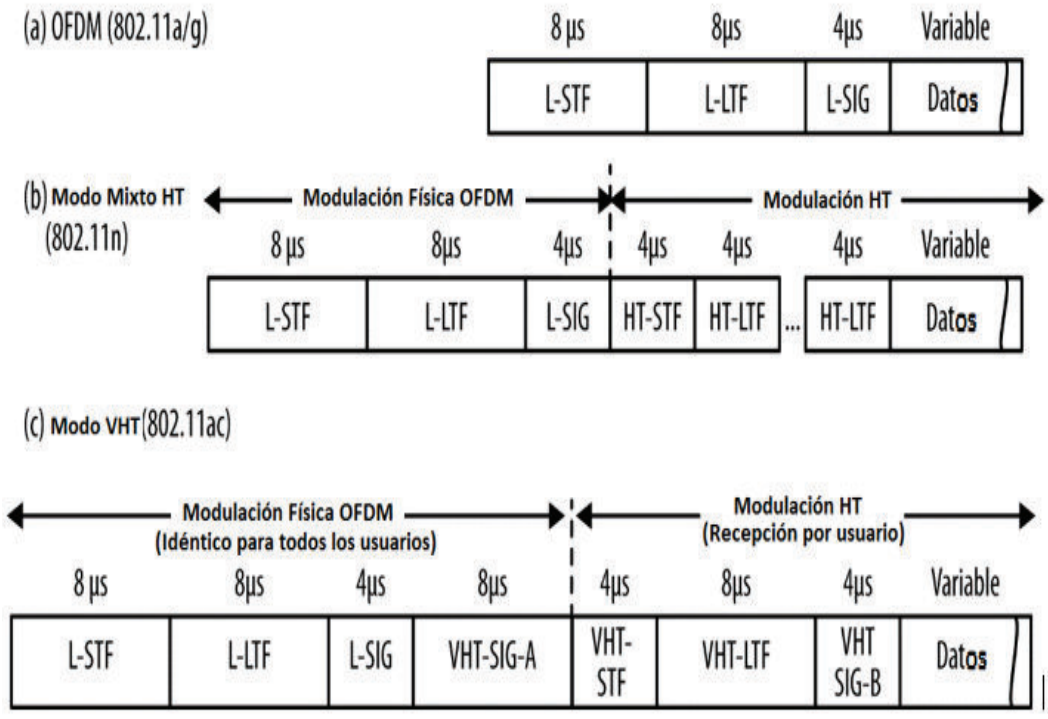


Figura 1.16 Formatos de tramas 802.11 [12]

La Figura 1.16 muestra el formato original no HT (*High Throughput*, Alto Rendimiento) de una trama OFDM (a), el formato de una trama en modo *mix-mode* (b) y el formato VHT (c). La trama VHT consiste de un *legacy preamble* (preámbulo legado), un preámbulo VHT y *un data payload* (carga útil de datos).

Del preámbulo legado forman parte: L-STF (*Legacy Short Training Field,* Campo abreviado de entrenamiento heredado), L-LTF (*Legacy Long Training Field,* Campo

prolongado de entrenamiento heredado) y L-SIG (*Legacy Signal,* Señal Heredada). Estos campos son los mismos que en los estándares 802.11a y 11n, permiten a los dispositivos 802.11 sincronizar las tramas de datos y evitar interferencias por parte de otras estaciones.

Para continuar con los campos: VHT-SIG-A, VHT-STF, VHT-LTF, VHT-SIG-B y finalmente el campo de DATOS. La siguiente sección da una breve explicación de cada uno de los campos mencionados.

1.2.6.4.1 L-STF (Legacy Short Training Field, Campo abreviado de entrenamiento heredado)

Formado por 12 símbolos OFDM para anchos de banda de 20 MHz, ayudan al receptor a identificar que una trama 802.11 está a punto de empezar, sincronizar *timers*, y seleccionar una antena. Cualquier dispositivo 802.11 que sea capaz de operar en OFDM puede decodificar este campo.

1.2.6.4.2 L-LTF (Legacy Long Training Field, Campo prolongado de entrenamiento heredado)

En 802.11ac se utiliza para decodificar los campos L-SIG y VHT-SIG-A. Consiste en 52 subportadoras para canales de 20 MHz.

1.2.6.4.3 L-SIG (Legacy Signal, Señal Heredada)

Es el último campo del preámbulo legado, en 802.11a se utiliza para describir la velocidad de los datos y la longitud de la trama en bytes, estos parámetros lo utilizan los receptores para calcular el tiempo de transmisión de la trama. Los dispositivos 802.11ac establecen una velocidad de datos de 6 Mbps y por ende se establece un valor de longitud de trama. Este valor de longitud obliga a los dispositivos con estándares anteriores a esperar hasta que la transmisión de la trama haya terminado.

1.2.6.4.4 VHT-SIG-A (Very High Throughput Signal A, Señal A de muy alto rendimiento)

Es el primer campo VHT y su formato varía dependiendo si la transmisión es para uno o varios usuarios. Consiste en dos símbolos de 24 bits (VHT-SIG-A1 y VHT-SIG-A2), cada uno lleva los parámetros necesarios para que una estación 802.11ac decodifique el resto de la trama. Figura 1.17.

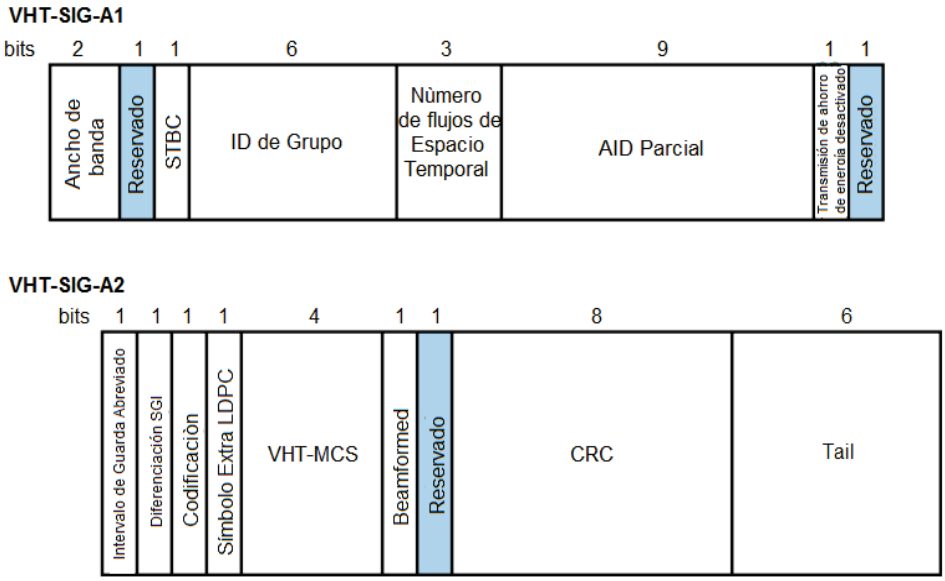


Figura 1.17 Campo VHT-SIG-A (Single User) [12]

El detalle de cada sub campo se explica a continuación:

- *Ancho de banda (2 bits):* Formado por 2 bits: 0 para 20, 1 para 40, 2 para 80 y 3 para 160 MHz.

- *STBC (1 bit):* Si el *payload* se codifica con STBC el campo será 1, de lo contrario 0.

- *Group ID (Group Identifier,* Identificador de Grupo) *(6 bits):* Cuando se transmite una trama de un solo usuario, el valor de este campo será 0 o 63. Un ID de grupo con un valor de 0 se utiliza para tramas que son enviadas al AP, en cambio, cuando el valor es 63 significa que las tramas son enviadas a un cliente.

- *Número de flujos de espacio-temporal (3 bits):* Indica el número de flujos espacio-temporales contando desde 0, es decir si su valor es de 3 entonces se tiene 4 flujos espacio-temporales.

- *AID Parcial* (*Partial Association Identifier*, Identificador de Asociación Parcial) *(9 bits):* Para transmisiones hacia un AP, este campo está formado por los últimos 9 bits del BSSID (*Basic Service Set Identifier*, Identificador del Conjunto de Servicios Básicos). Para transmisiones hacia un cliente, el *Partial* AID está formado por la combinación del ID de asociación y el BSSID del AP al cual está conectado.

- *Transmisión de ahorro de energía desactivado (1 bit):* Este campo indica si a los dispositivos clientes se les permite o no adoptar un estado de "apagado". Será 0 si al dispositivo cliente se le permite entrar en dicho estado.

- *SGI* (*Short Guard Interval,* Intervalo de Guarda Abreviado) *(1 bit):* El valor de este campo se establece en 1 para indicar que el intervalo de guarda abreviado de 400 nseg se utiliza en el *payload* de la trama 802.11ac, de lo contrario el valor será 0.

- *Diferenciación Intervalo de Guarda (1 bit):* Indica si se requiere un símbolo adicional cuando se utiliza SGI.

- *Codificación (1 bit):* Toma el valor de 0 si se usa una codificación convolucional[9] y 1 cuando se usa LDPC.

- *Símbolo extra LDPC (1 bit):* La codificación LDPC puede crear la necesidad de utilizar un símbolo ODFM adicional para transmitir el campo de datos.
  Este campo toma el valor de 1 para indicar que se necesita un símbolo extra.

[9] Código Convolucional: Es un código de detección de errores donde cada símbolo de *m* bits de información al codificarse se convierte en un símbolo de *n* bits. Donde *m/n* es la tasa del código y n≥m.

- *MCS (4 bits):* Contiene el índice MCS para el *payload*. Los posibles valores se muestran en la primera columna de la Tabla 1.5.

- *Beamformed (1bit):* Cuando se utiliza una matriz *beamforming* para la transmisión el valor del campo será 1, caso contrario será 0.

- *CRC (Cyclic Redundancy Code,* Codificación por Redundancia Cíclica*) (8 bits):* Permite al receptor detectar errores en el campo VHT SIG-A.

- *Tail (6 bits):* Consta de 6 ceros para terminar el proceso del codificación convolucional.

1.2.6.4.5 VHT-STF (VHT Short Training Field, Campo abreviado de entrenamiento VHT)

Este campo es similar a L-STF. Se utiliza para el control automático de ganancia en transmisiones MIMO y para ajustar los tiempos de sincronización.

1.2.6.4.6 VHT-LTF (VHT Legacy Long Training Field, Campo prolongado de entrenamiento heredado VHT)

Formado por una secuencia de símbolos que establecen la demodulación de la trama, empezando por el campo *VHT SIG-B*. El campo VHT-LTF además es utilizado por el receptor para la estimación de canal MIMO. El número de símbolos VHT-LTFs (1, 2, 4, 6, u 8) depende del número de *spatial streams* como se muestra en la Tabla 1.7. Por ejemplo, en un enlace con cinco *streams,* seis símbolos son necesarios.

VHT-LTFs								
Número de *Spatial Streams*	1	2	3	4	5	6	7	8
Número de Símbolos VHT-LTFs	1	2	4	4	6	6	8	8

Tabla 1.7 Flujos y Símbolos VHT-LTF

1.2.6.4.7 VHT-SIG-B (Very High Throughput Signal B, Señal B de muy alto rendimiento)

Este campo lleva información de la longitud del campo de DATOS y del índice MCS que utiliza cada usuario en una transmisión multi-usuario. En el caso de existir un único usuario, VHT-SIG-B solo incluirá información de la longitud del campo de DATOS. El número de bits varía dependiendo del ancho de banda tal como lo muestra la Figura 1.18.

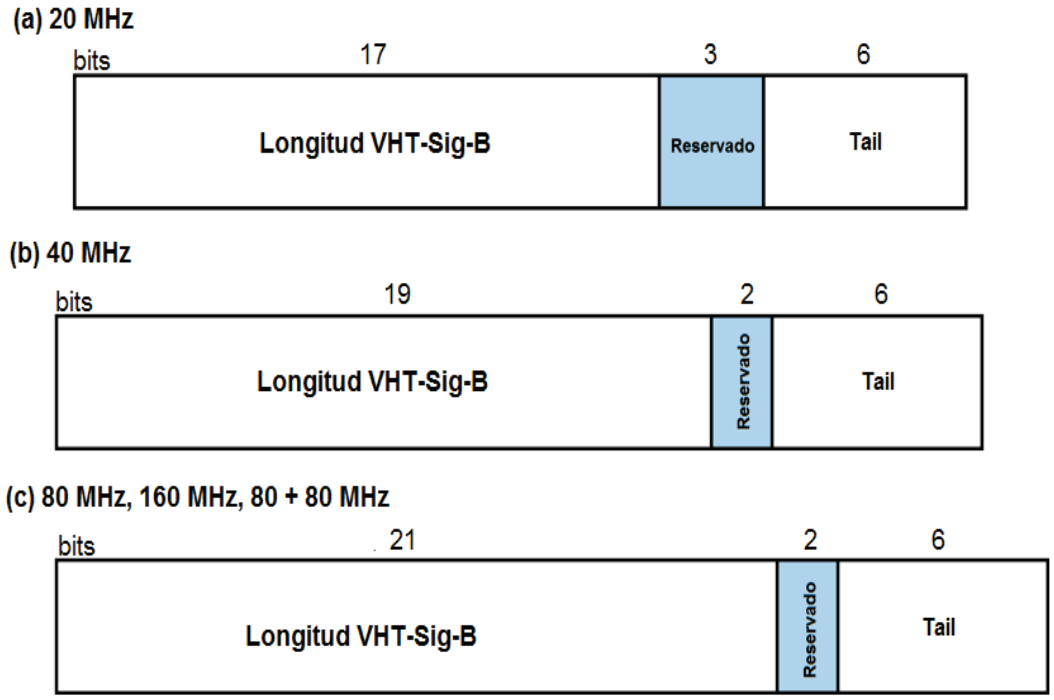


Figura 1.18 VHT-SIG—B para diferentes Anchos de Banda [12]

Los canales con mayor ancho de banda tienen la capacidad de transportar más datos, por lo que el campo VHT-SIG-B se repite, tal como muestra la Figura 1.19. Para un canal de 40 MHz el campo se repite una vez. Para un canal de 80 MHz el canal se repite dos veces y se anexa un conjunto de bits 0. Para uno de 160 u 80+80 MHz el campo se repite dos veces, también se añade bits 0 de relleno y después la estructura resultante se la repite una vez más. Con estas repeticiones se asegura que se ocupe exactamente un símbolo.

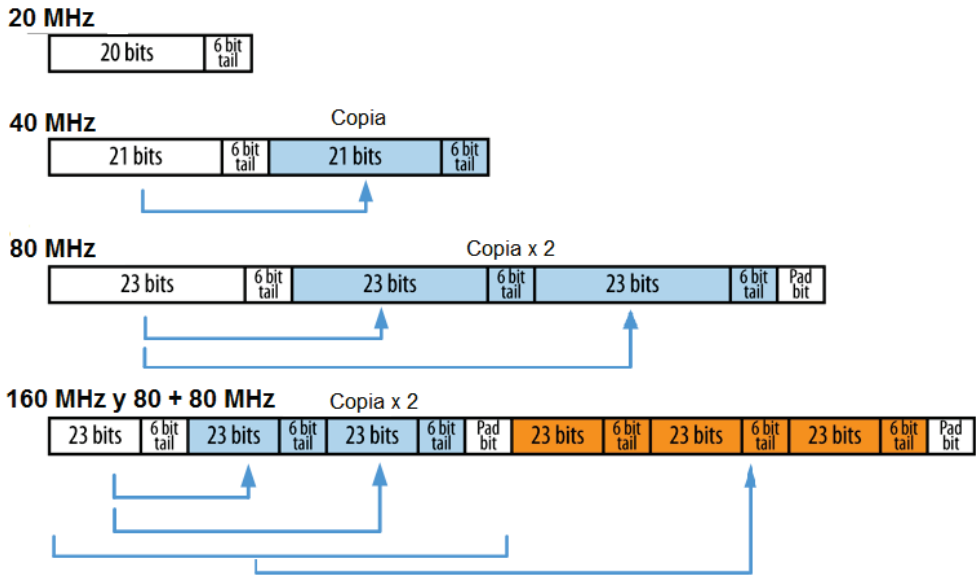


Figura 1.19 Campo VHT-SIG-B de acuerdo al ancho de banda de canal [12]

1.2.6.4.8 Campo de Datos

Consiste en un número variable de símbolos OFDM que básicamente lleva la carga útil o PSDU (*Physical Layer Service Data Unit,* Unidad de Datos de Servicio de Capa Física), cuyo número de símbolos que se transmitirán depende del campo L-SIG. La Figura 1.20 muestra los elementos que conforman este campo: *Service,* PSDU, *PHY Pad* (Relleno de capa física) y *Tail*. El sub campo *Service* contiene un código inicializador del *scrambler*[10] de siete bits, un bit reservado para futuras aplicaciones y ocho bits de CRC[11] para detección de errores. PSDU es de longitud variable y contiene la unidad de datos de servicios PLCP (*Physical Layer Convergence Protocol*, Protocolo de Convergencia de Capa Física). *PHY Pad* posee un número variable de bits de relleno utilizados para crear símbolos OFDM completos. *Tail* que contiene bits empleados para finalizar el proceso de codificación convolucional.

[10] Scrambler: Permite asegurar que las señales transmitidas tengan un comportamiento estadístico razonable y evita la aparición de largas cadenas de un mismo símbolo, o la repetición periódica de ciertas secuencias.
[11] CRC: Técnica utilizada para la detección de errores de datos digitales, pero no para corregirlos.

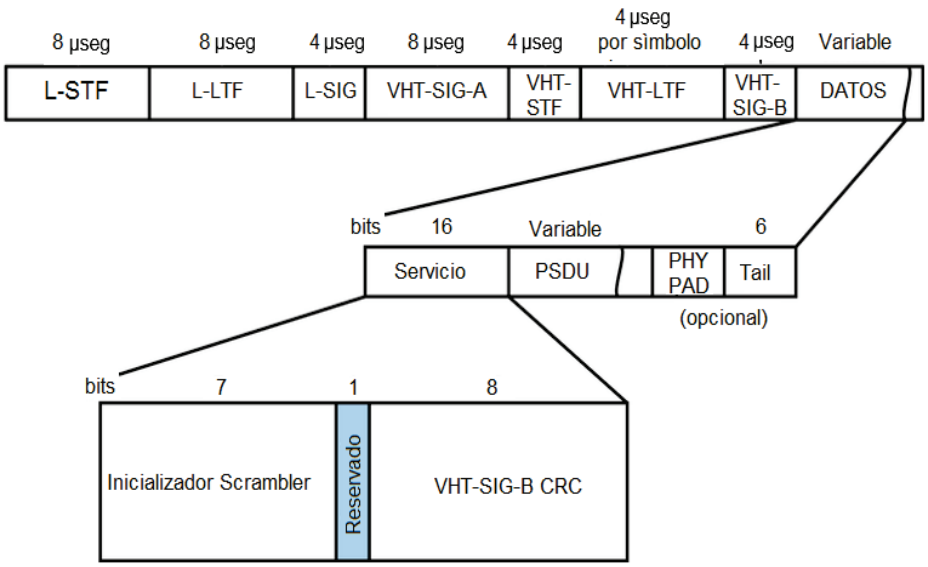


Figura 1.20 Campo DATOS [12]

1.2.6.4.9 Variaciones para MU-MIMO

Para poder transmitir varios *streams* es necesario realizar algunos cambios a la trama original de 802.11ac. Los cambios en el campo VHT-SIG-A se representan en la figura 1.21.

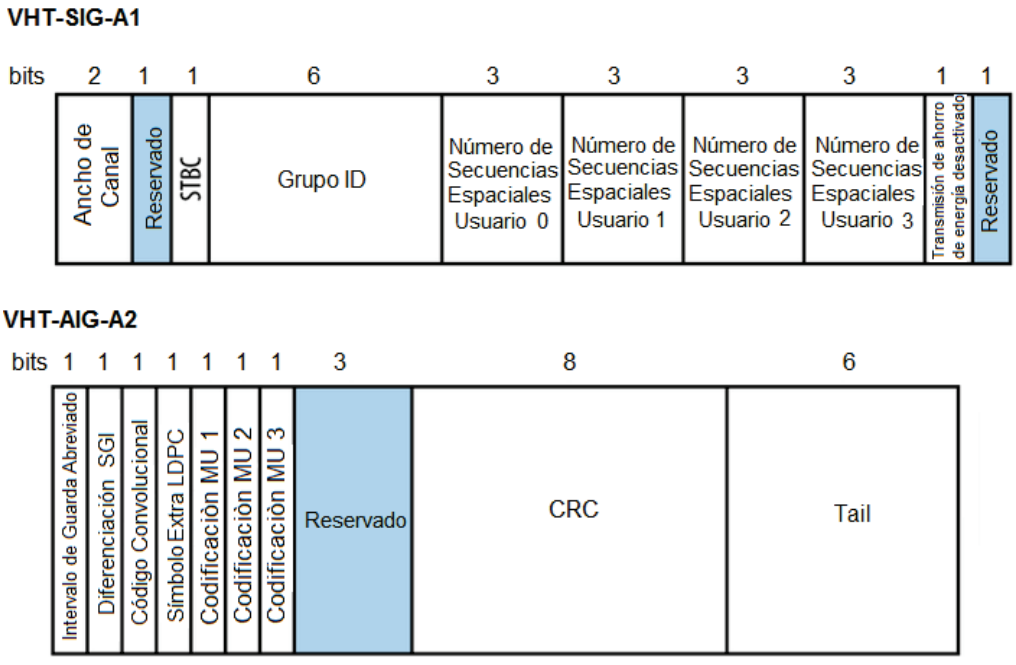


Figura 1.21 VHT-SIG-A Multi-Usuario [12]

- *Número de secuencias espaciales (12 bits, 3 para cada usuario):* Una transmisión multi usuario puede ser enviada a un máximo de 4 usuarios a la vez. Estos sub campos indican a los receptores cuantos *streams* deben esperar en la transmisión.

- *Multi-user coding (3 bits):* En la versión multi-usuario existe un bit que indica si ha sido utilizada una codificación convolucional o LDPC. Los valores de MCS para cada *stream* de los usuarios se mueven al campo VHT-SIG-B.

En este tipo de transmisiones multi-usuario, cada usuario tiene su propio campo VHT-SIG-B. A continuación la Figura 1.22 presenta los formatos para *single-user* y *multi-user* para 20, 40 y 80 MHz respectivamente.

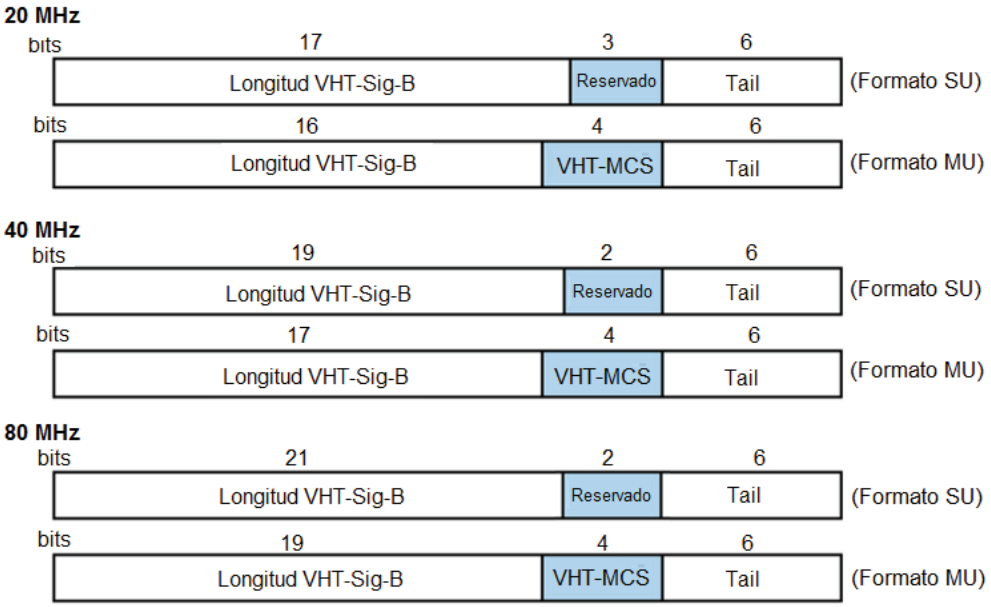


Figura 1.22 VHT-SIG-B Multi-Usuario [12]

1.2.7 SUB CAPA MAC

La sub capa MAC es la encargada de coordinar que los dispositivos puedan tener acceso al canal de transmisión. La mayoría de los cambios implementados en la sub capa MAC de 802.11ac se relacionan con las nuevas características

implementadas en la capa física, además de mantener el formato utilizado por sus predecesores. Existen dos cambios significativos:

- Se amplía la longitud máxima de trama, de 8000 a alrededor de 11000 bytes.
- Reúsa el campo HT Control de 802.11n pero definiendo un nuevo formato para este campo. Cuando el campo HT Control empieza con 0, el formato es idéntico a 802.11n y es del tipo HT. Cuando empieza con 1 es del tipo VHT.

La Figura 1.23 muestra el formato de la trama de sub capa MAC campo HT Control cuando es del tipo VHT.

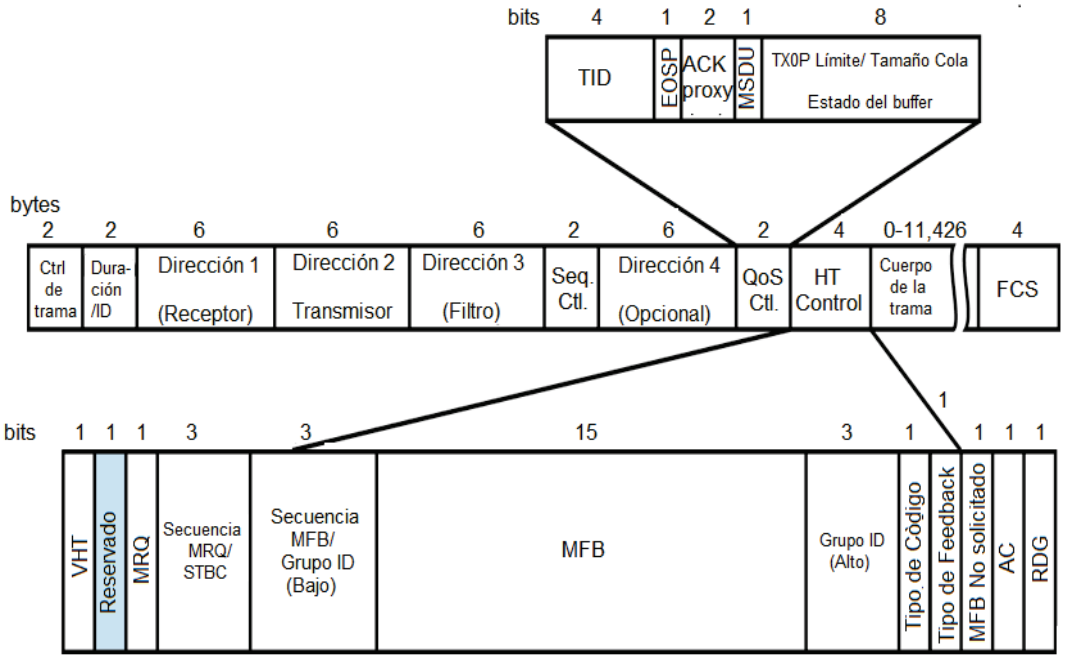


Figura 1.23 Formato de Trama de Sub capa MAC 802.11ac [13]

1.2.7.1 Agregación de Tramas

La agregación de tramas busca mejorar el rendimiento de la red, permitiendo que un dispositivo transmita múltiples tramas al momento de acceder al canal, disminuyendo así su sobrecarga de transmisión. En 802.11ac todas las tramas utilizan el formato A-MPDU (*Aggregated MAC Protocol Data Unit*, Agregación de Unidad de Datos de Protocolo MAC) lo que significa que la capa MAC debe hacerse cargo de todo el procedimiento de *framing* (entramado), y que la capa física trabaje

únicamente con la longitud total de lo que está transportando. Todas las tramas de datos 802.11ac son enviadas con el formato A-MPDU, incluso si están conformadas por una sola trama. [13].

Debido a las altas velocidades potenciales que maneja el estándar, describir la longitud de la trama requiere de muchos bits, por lo que la longitud máxima de transmisión se define por tiempo, 5.484 mseg [8]. En lugar de representar la longitud con un gran número de bytes en la cabecera PLCP, la cual se transmite a la menor velocidad de datos posible, 802.11ac desplaza ese indicador de longitud a los delimitadores MPDU que son transmitidos a altas velocidades de datos como parte del *payload.*

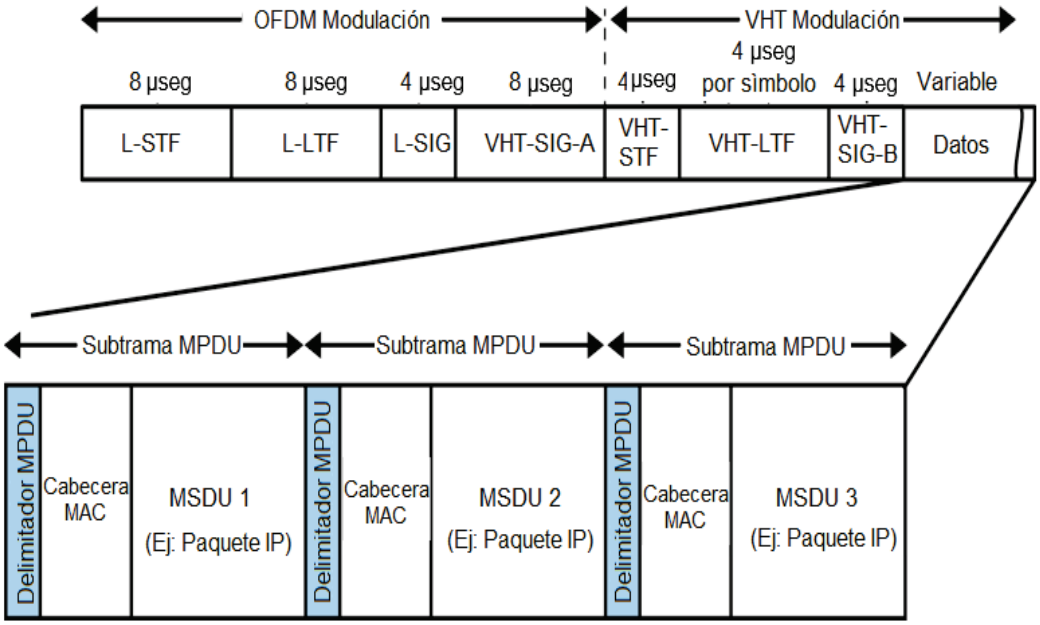


Figura 1.24 Formato A-MPDU [13].

1.2.7.2 **Tramas de Administración**

El estándar 802.11ac posee tramas de administración que permiten establecer conexiones con redes del mismo tipo. Estas tramas están conformadas por elementos que informan sobre las capacidades y operaciones VHT.

1.2.7.2.1 Elemento de Información de Capacidades VHT

Es el elemento principal utilizado en las tramas de administración para configurar el funcionamiento y operación de redes 802.11ac, Figura 1.25. Su estructura es simple: dos campos que describen las características y velocidades que el transmisor puede manejar.

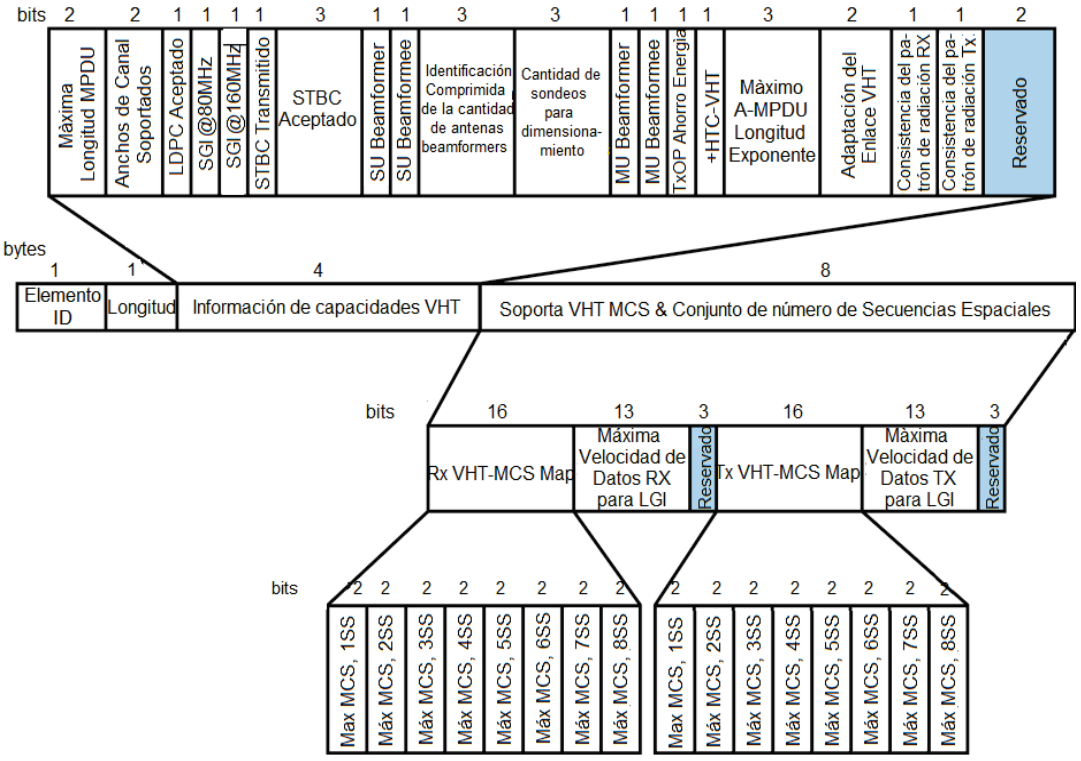


Figura 1.25 Elemento de Información de Capacidades VHT [13]

A continuación se describe cada uno de los campos que lo conforman:

- *Máximo longitud de MPDU*: Las tramas MAC 802.11ac pueden tener una de las tres longitudes: 3895, 7991 o 11454 bytes. Que corresponden a los valores de 0, 1 y 2 respectivamente. El valor de 3 es reservado [13].

- *Anchos de Banda Soportado*: Este campo es habilitado para poder identificar si el transmisor soporta anchos de banda de 160 Mhz, así como los anchos de banda 80+80 Mhz.

- *Rx LDPC*: Para detectar si el transmisor puede recibir tramas LDPC.

- *Intervalo de guarda abreviado (80 MHz y 160 MHz):* Estos campos se establecen en 1 si el transmisor puede recibir tramas usando el intervalo de guarda abreviado.

- *Tx STBC*: Permite identificar si la transmisión se producirá con tramas codificadas en STBC.

- *Rx STBC:* Permite identificar cuantos *spacial streams* son compatibles en recepción para tramas codificadas con STBC. El campo toma el valor de 1 para un *spacial streams*. El valor 0 indica que STBC no es compatible y los valores del 5 al 7 son reservados.

- *SU Beamformer y SU Beamformee:* Permite seleccionar si el transmisor puede operar como *beamformer* de un solo usuario o como *beamformee* para intercambiar paquetes con otra estación.

- Identificación comprimida de la cantidad de antenas beamformers y cantidad de sondeos para dimensionamiento: Estos campos se utilizan para indicar el número máximo de antenas que pueden participar en la medición del canal en el proceso de *beamforming*.

- *MU Beamformer y MU Beamformee:* Cuando el valor de estos campos es 1, el transmisor es capaz de operar como *beamformer o beamformee multi-usuario.*

- *VHT TxOp Power-Save:* Un AP puede activar este campo, con el valor de 1, para habilitar el ahorro de energía durante una transmisión VHT. Las estaciones asociadas a la red también utilizarán el valor de 1 para indicar que esta característica está habilitada.

- *+HTC-VHT(High Throughput Control-VHT, Control de Alto rendimiento - VHT)*: Se establece en 1 para indicar que el transmisor es capaz de recibir el campo VHT Control.

- *Longitud máxima del exponente A-MPDU:* Este campo puede tomar valores de 0 a 7. Indica la longitud a la que puede transmitirse una A-MPDU.

- *Capacidad de adaptación de enlace VHT:* Determina el MCS más apropiado para un enlace que utiliza retroalimentación.

- *Consistencia del patrón de radiación de las antenas de transmisión y recepción*: Estos campos toman el valor de 1 si el patrón de radiación de la antena transmisora no cambia. Una de las razones más comunes por la que los patrones de radiación varíen es la transmisión beamforming.

- *RX y TX VHT-MCS Mapeo:* Posee dos bits para representar tres opciones: el valor 0 indica un MCS de 0 a 7, el valor 1 agrega el MCS 8 y el valor 2 agrega el MCS 9. El valor de 3 es reservado. El campo de dos bits se repite ocho veces para que el transmisor puede especificar los MCS máximos admitidos para cada *spatial stream*.

- *Más alta tasa de datos soportada para Tx y Rx:* Representan las velocidades de datos más altas con las que se puede trabajar. Cada campo posee 13 bits.

1.2.7.2.2 Elemento de Información de Operación VHT

Presenta información del canal y de velocidades básicas soportadas por el transmisor, Figura 1.26. Las velocidades básicas son aquellas que son compatibles por todos los clientes que están conectados al AP. La parte inicial de este elemento se refiere a los canales utilizados por el transmisor, lo componen los siguientes campos:

- Ancho de Canal (1 byte): Para operaciones en 20 y 40 MHz el valor del campo se establece en 0, 1 para 80 MHz, 2 para canales de 160 MHz y 3 para canales de 80+80 MHz. Los valores restantes son reservados.

- Frecuencia Central de Canal 0 (1 byte): Utilizado solo para cuando se tiene canales de 80 y 160 MHz, para transmitir la frecuencia central de la BSS.

- Frecuencia Central de Canal 1 (1 byte): Este campo se utiliza solo para operaciones en 80+80 MHz y se usa para transmitir la frecuencia central del canal del segundo segmento.

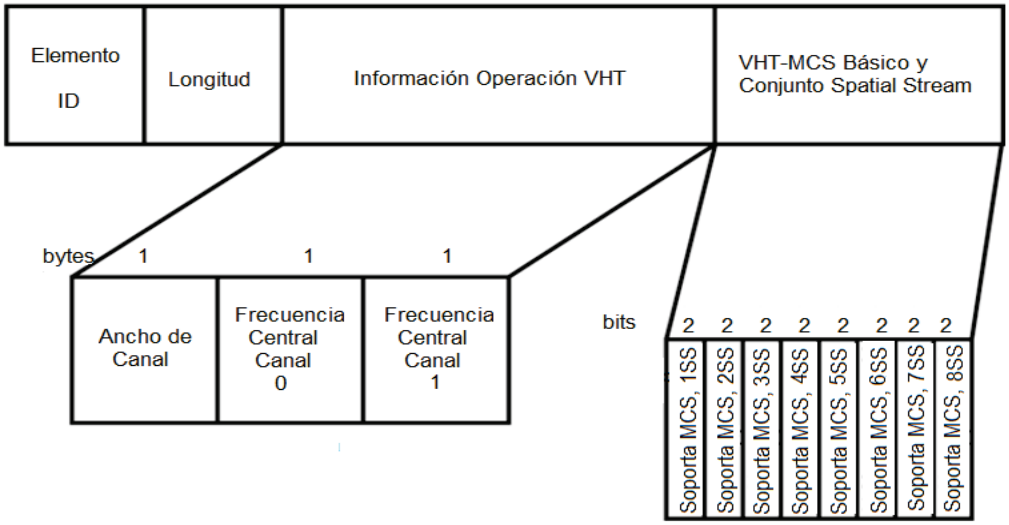


Figura 1.26 Elemento de Información de Operación VHT [13]

1.2.7.3 **Procedimientos de Acceso al Medio**

El hecho de que en este nuevo estándar estén disponibles nuevos anchos de banda conlleva a que existan también nuevas reglas de acceso al medio de comunicación que determinen si el canal está libre o no.

1.2.7.3.1 SubCanales Primarios y Secundarios

De manera similar que en 802.11n, el estándar 802.11ac consta de un canal de 40 MHz que requiere de un sub canal primario de 20 MHZ. Adicionalmente, los canales

de 80 MHZ tienen un subcanal primario de 40 MHz (que a su vez incluye un canal primario de 20 MHz) y un subcanal secundario de 40 MHz. Lo mismo se aplica para los canales de 160 MHz y 80 + 80 MHz que consisten de subcanales primarios y secundarios de 80 MHz. En la figura 1.27 se representa la relación entre subcanales primarios y secundarios.

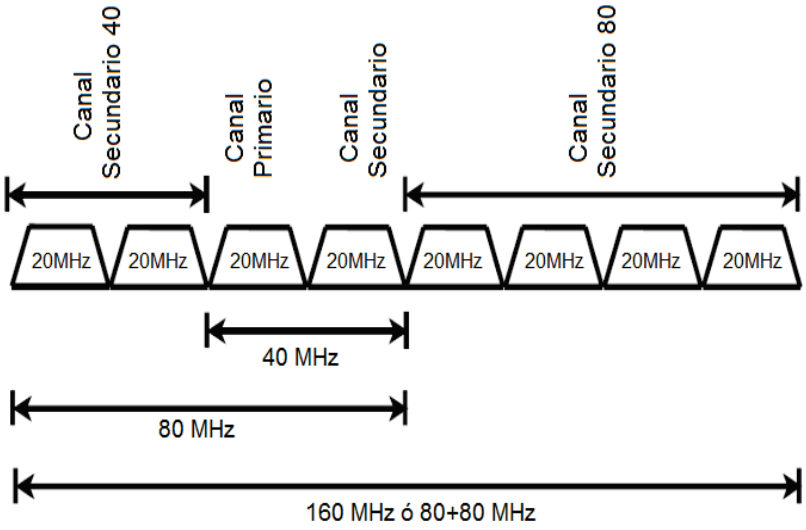


Figura 1.27 Canales Primarios y Secundarios [3]

En todos los casos, el subcanal primario se utiliza para la detección de portadora con el fin de garantizar que ningún otro dispositivo está transmitiendo. La presencia del sub canal primario de 20 MHz también es necesaria para garantizar la coexistencia y compatibilidad hacia "atrás" con los dispositivos 802.11. Solo el subcanal primario realiza CCA[12] (*Clear Channel Assessment,* Evaluación de Disponibilidad del Canal) que consiste en la detección de paquetes desde el preámbulo, por el contrario no se requiere que el sub canal secundario realice CCA.

1.2.7.3.2 Acceso estático y Dinámico al Canal

El estándar 802.11ac extiende las políticas de acceso propuestas en 802.11n para el caso de los canales de 80 y 160 MHz. Para que una estación sea capaz de

[12] CCA: Es un mecanismo de detección de portadora en ambientes WLAN. CCA involucra dos funciones: detección de portadora y detección de energía.

transmitir una PPDU *(Phy Protocol Data Unit,* Unidad de Datos de Protocolo de capa Física*)* a 80MHz se deben cumplir dos condiciones:

- El canal primario sigue las reglas EDCA[13] (*Enhanced Distributed Channel Access,* Mejora del Acceso al Canal Distribuido) por lo que necesita estar inactivo por un tiempo DIFS[14] *(DCF Inter Frame Space,* Espacio entre tramas DCF) más la duración del contador de *backoff*[15]*.*

- Los tres sub canales secundarios deben haber estado inactivos durante el periodo de PIFS[16] (*PCF Inter-Framing Spacing,* Espacio entre tramas PCF) inmediatamente antes de que el contador de *backoff* expire. [3]

En el caso que cualquier subcanal secundario este ocupado, la estación puede seguir cualquiera de las reglas de acceso estático o dinámico al canal como se menciona en 802.11n:

- *Acceso Estático al canal:* En el caso que una estación 802.11ac intente transmitir en un canal de 80 MHz. Si el subcanal secundario está ocupado la estación elegirá un periodo de espera (*backoff*) al azar dentro del tamaño actual de la ventana de contención para reiniciar el proceso de contienda e intentar transmitir solo si la interferencia no está presente en cualquiera de los subcanales.

- *Acceso Dinámico al Canal:* La estación 802.11ac puede intentar transmitir en un canal más pequeño, usando un canal de 20 o 40 MHz, esto dependerá del CCA de cada sub canal. Esto es un enfoque que proporciona mayor flexibilidad, que permite que la asignación de recursos sea más eficiente porque la estación aún puede transmitir a través de una fracción

[13] EDCA: Acceso al canal para estaciones con tramas con diferentes prioridades. Una estación con tráfico de alta prioridad espera un poco menos antes de enviar su paquete, en promedio, que una estación con tráfico de baja prioridad.
[14] DIFS: utilizado por las estaciones que esperan acceder al canal en un proceso de contingencia.
[15] Algoritmo de Backoff: Algoritmo que consiste en esperar un tiempo aleatorio antes de volver a intentar una transmisión.
[16] PIFS: Utilizado por PCF durante las operaciones libres de contención.

del ancho de banda original. Todas las transmisiones siempre tienen que incluir el canal primario con el fin de informar al receptor que canales el transmisor usará. [3]

1.2.7.3.3 Mecanismo RTS (Ready To Sent, Listo para enviar) /CTS (Clear To Send, Libre para enviar)

Se plantea un *handshake*[17] para evitar colisiones y así manejar de mejor manera la asignación de canal estático y dinámico. Este *handshake* consiste en una modificación a los mecanismos RTS/CTS que proporciona información sobre la cantidad actual de ancho de banda disponible. Por ejemplo, si se tiene un escenario donde un AP desea transmitir a un cliente sobre un canal de 80MHz, en primer lugar el AP verifica si el canal está libre, si es así, entonces se envían varios RTS en formato PPDU 802.11a (un RTS para cada subcanal de 20MHz). Una vez realizado lo anterior se espera que cualquier dispositivo cercano reciba un RTS en su canal primario.

Antes de que el cliente responda con un CTS se comprueba si cualquiera de los sub-canales en la banda de 80MHz está ocupado, el cliente solo responde con un CTS en los subcanales que están libres e informa al AP el ancho de banda total.

1.2.7.3.4 Administración de Ancho de Banda Dinámico

Sólo los anchos de banda de 20, 40, 80 y 160 MHz están permitidos en 802.11ac, 60 y 120 MHz no están permitidos. Con este aumento de ancho de banda, un canal disponible tiene una mayor flexibilidad. Sin embargo, lo que se busca es optimizar el uso del canal con ancho de banda más amplio de manera eficiente. Una red 802.11ac incluye un canal principal de 20 MHz. Se accede a este canal primario mediante detección de portadora para asegurarse de que el canal está libre de interferencias de otras redes.

[17] Handshake: Técnica utilizada para que un comunicación se establezca de manera correcta.

Otro uso para el canal principal es la convivencia y la compatibilidad hacia atrás con estándares anteriores Wi-Fi. Una vez que se obtiene el acceso al canal principal, se pueden añadir canales secundarios obteniéndose un ancho de banda más amplio.

El estándar 802.11n define un mecanismo de *handshake* CTS / RTS en modo estático. Esto se muestra en la Figura 1.28, en este caso el transmisor no tiene ninguna interferencia en los canales primarios y secundarios y envía una señal de RTS, sin embargo, el receptor posee interferencia en el canal secundario y no puede responder dicha petición con un CTS por lo que ninguna transmisión se produce en este caso.

Por otro lado, en el modo dinámico definido en 802.11ac, la interferencia de canal se mide por canal, y el receptor puede enviar señales CTS por canal para indicar qué canales están libres de interferencias. En este caso se permite que la transmisión tenga lugar en el canal principal, lo que mejora la utilización de ancho de banda y el rendimiento general de la red.

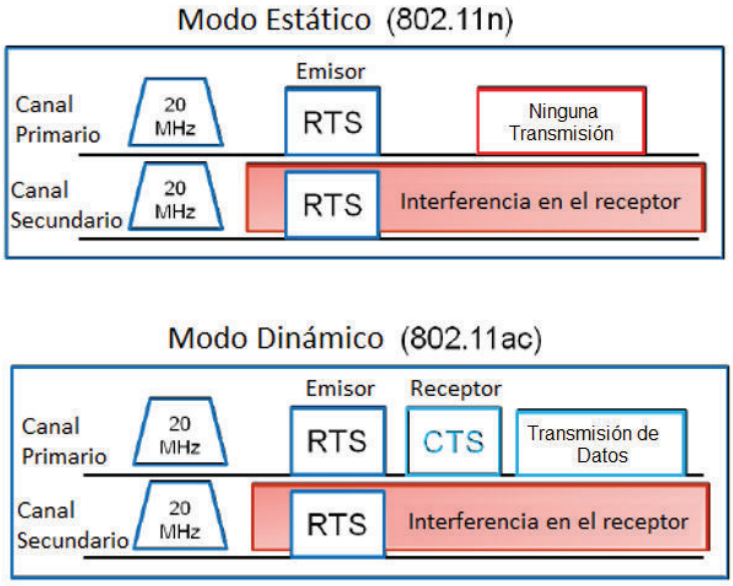


Figura 1.28 Administración de ancho de banda [14]

CAPÍTULO II

SIMULACIÓN DEL ESTÁNDAR IEEE 802.11AC Y MU-MIMO

2.1 INTRODUCCIÓN

En este capítulo se realiza la simulación del estándar 802.11ac, así como la técnica MU-MIMO por medio del software computacional Matlab® R2015b. La simulación se realiza a nivel de capa física estableciendo modelos de canal inalámbrico con distribuciones de *Rician* y *Rayleigh*.

Se plantea un programa en el cual el usuario selecciona el tipo de modelo de canal, la cantidad de usuarios, la cantidad de antenas por usuario, el valor de MCS*,* se elige el valor de SNR (*Signal Noise Ratio,* Relación Señal a Ruido) y el valor de ancho de banda de canal. El proceso de simulación del estándar 802.11ac permite obtener resultados matemáticos y representaciones gráficas. Estos resultados posteriormente son comparados en el capítulo 4 con los datos obtenidos en la parte práctica.

2.2 CANAL INALÁMBRICO

El canal inalámbrico se define como el medio por el cual viajan las señales desde un transmisor a un receptor, siendo el medio de transmisión el aire. En este tipo de comunicaciones la señal enviada se ve afectada por fenómenos como el ruido, pérdida de la trayectoria, dispersión, reflexión, múltiples reflexiones, absorción, entre otros. Los fenómenos se generan debido a la presencia de obstáculos en el canal inalámbrico, ver Figura 2.1

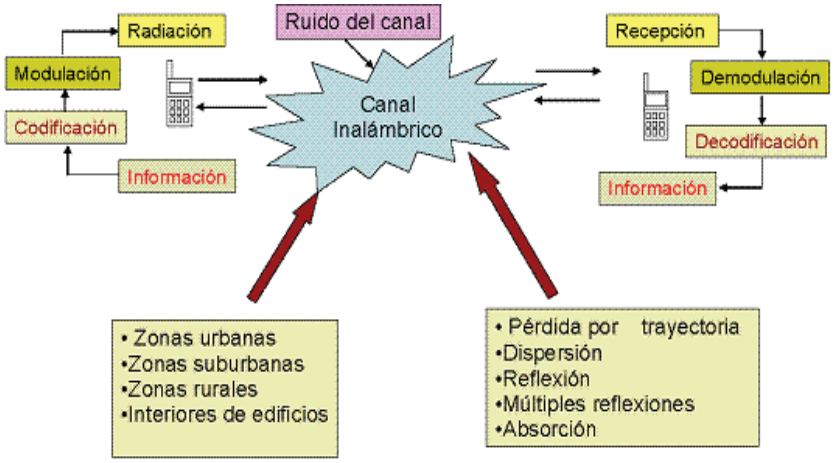


Figura 2.1 Diagrama de comunicación. [15]

2.2.1 DESVANECIMIENTO (*FADING*)

El efecto de desvanecimiento agrupa diferentes fenómenos que afectan a la señal que viaja por el canal inalámbrico, provocando un cambio de amplitud, fase o frecuencia en la señal. A continuación se presenta un resumen de los tipos de *fading*.

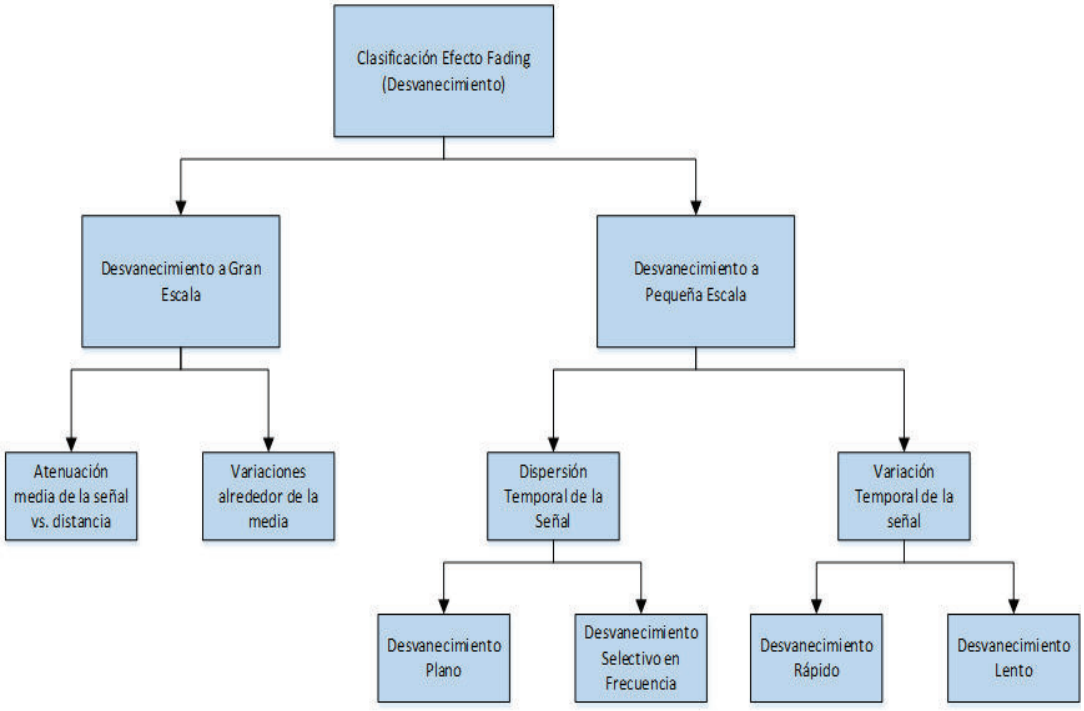


Figura 2.2 Clasificación Fading

2.2.2 DESVANECIMIENTO A GRAN ESCALA

El desvanecimiento a gran escala representa la atenuación de la potencia promedio de la señal en función de la distancia recorrida, siempre que dicha distancia sea mucho mayor a la longitud de onda (d>>λ).

2.2.3 DESVANECIMIENTO A PEQUEÑA ESCALA [16]

El desvanecimiento a pequeña escala, es un término para describir la rápida variación de la amplitud y de fase de una señal en un pequeño periodo de tiempo o durante una distancia corta con relación a su longitud de onda λ.

2.2.3.1 **Dispersión Temporal**

La dispersión temporal permite calcular los diferentes tiempos y por lo tanto diversas atenuaciones con las cuales las componentes multitrayectoria llegan al receptor. Para explicar el efecto de la dispersión temporal se relaciona el parámetro de ancho de banda coherente[18] (Bc) con el ancho de banda de la señal (Bw).

2.2.3.1.1 Desvanecimiento Plano (Flat Fading)

El desvanecimiento plano se presenta cuando el ancho de banda coherente es mayor al ancho de banda de la señal.

$$Bc > Bw \qquad (2.1)$$

2.2.3.1.2 Desvanecimiento Selectivo en frecuencia (Frequency Selective Fading)

El desvanecimiento selectivo se presenta cuando el ancho de banda coherente es menor al ancho de banda de la señal.

[18] Ancho de Banda Coherente: Rango de frecuencias en el cual el canal se puede considerar plano (todas sus componentes en frecuencia experimentan la misma atenuación).

Este tipo de desvanecimiento es más complejo de diseñar, ya que cada componente de la señal se modela por separado.

$$Bc < Bw \qquad (2.2)$$

2.2.3.2 **Variación Temporal del Canal**

La variación temporal del canal se presenta debido al movimiento que existe entre el transmisor y el receptor, producto de ello se originan alteraciones en la amplitud y la fase del canal.

2.2.3.2.1 Desvanecimiento Rápido (Fast Fading)

El desvanecimiento rápido ocurre cuando el tiempo de símbolo[19] (Ts) es mucho mayor al tiempo coherente[20] (Tc).

$$Ts \gg Tc \qquad (2.3)$$

2.2.3.2.2 Desvanecimiento Lento (Slow Fading)

Se tiene este tipo de *fading* cuando el tiempo de símbolo (Ts) es mucho menor al tiempo de coherencia (Tc), aquí se puede asumir que el canal es estático[21] durante uno o varios periodos de la señal transmitida.

$$Ts \ll Tc \qquad (2.4)$$

---

[19] Tiempo de Símbolo: Tiempo mínimo que debe mantenerse un símbolo en el medio de transmisión para poder distinguirlo.
[20] Tiempo Coherente: Intervalo de tiempo para el cual el comportamiento del canal no varía.
[21] Canal Estático: En este entorno se considera que el canal permanece constante durante la transmisión.

2.2.3.3 **Modelo Rayleigh y Rician [17]**

2.2.3.3.1 Modelo de Desvanecimiento Rayleigh

Un desvanecimiento *Rayleigh* se produce cuando no existe línea de vista entre el transmisor y el receptor. La función de densidad de probabilidad que describe la distribución de *Rayleigh* viene dada por la siguiente relación matemática.

$$f(r) = \frac{r}{\sigma^2}\exp\left(-\frac{r^2}{2\sigma^2}\right) \qquad r \geq 0 \qquad (2.5)$$

Donde

r = Envolvente de la señal recibida.

$\frac{r^2}{2}$ = Potencia instantánea.

σ^2 = Potencia media de la señal.

2.2.3.3.2 Modelo de Desvanecimiento de Rician

El desvanecimiento de *Rician* implica que existe un trayecto predominante o LOS, este modelo es utilizado en ambientes abiertos o en ambientes *indoor*. La función de densidad de probabilidad de la distribución *Rice* es descrita con la fórmula:

$$f(r) = \frac{r}{\sigma^2}\exp\left(\frac{-\left(r^2+A^2\right)}{2\sigma^2}\right) I_0\left(\frac{Ar}{\sigma^2}\right), \qquad r \geq 0, A \geq 0 \qquad (2.6)$$

r = Representa la envolvente de la señal recibida

$2\sigma^2$ = Potencia media de la señal multitrayecto

Ar = Amplitud pico de la señal dominante.

I_0= Función de Bessel modificada de primera especie.

$\frac{r^2}{2}$= Potencia Instantanea.

2.2.3.3.3 Factor de Rician (k)

El factor de *Rician* es la relación de la potencia que existe entre la señal en la ruta dominante y la potencia de la señal en las rutas dispersas.
Así, cuando k=0 se tiene un canal con desvanecimiento *Rayleigh* ya que no se tiene una componente LOS.

$$k = \frac{potencia\ en\ la\ ruta\ dominante}{potencia\ en\ las\ rutas\ dispersas} \qquad (2.7)$$

Para k=∞, se tiene un canal con AWGN[22] (*Additive White Gaussian Noise,* Ruido Gaussiano Blanco Aditivo).

2.3 MODELAMIENTO DE CANAL INALÁMBRICO

2.3.1 MODELOS DE CANAL WLAN

El diseño de un sistema de comunicación inalámbrico requiere conocer en detalle la propagación de las señales por el canal. Los canales inalámbricos que utilizan diversidad de antenas en transmisión y recepción añaden mayor complejidad al estudio y a la caracterización del canal dado que se debe relacionar los distintos sub-canales generados entre cada antena transmisora y receptora con los factores propios que afectan a una transmisión de este tipo, como por ejemplo el desvanecimiento.

En el estudio *TGn* (*Task Group 802.11n*, Grupo de Trabajo 802.11n) *Channel Models* [18] realizado por la IEEE para el modelado del canal de 802.11n parten de los trabajos desarrollados por Medbo[23] y Saleh-Valenzuela[24].

---

[22] AWGN: Modelo básico usado para representar el ruido térmico en los canales de comunicación. La señal recibida es igual a la transmitida más ruido con densidad de potencia espectral plana.
[23] Medbo: Modelo que determina los ambientes físicos para una WLAN, se propone perfiles de escenarios, los cuales varian en dimensiones, en condiciones LOS/NLOS.
[24] Saleh-Valenzuela: Modelo propuesto para entornos multitrayectoria, consiste en que las múltiples reflexiones que llegan al receptor del sistema no lo hacen de manera independiente, sino de manera agrupada (*Clústers*).

En el modelo de Medbo se definen los ambientes en los cuales se produce la transmisión (lugar físico). En la Tabla 2.1 se presentan los tipos de ambientes o entornos de estudio y la condición de propagación de la señal transmitida asociados a cada modelo (A al F).

Entorno	Condición	Modelo
Desvanecimiento plano[25]	LOS[26]/NLOS[27]	A
Residencial	LOS/ NLOS	B
Residencial / Oficina pequeña	LOS/ NLOS	C
Oficina típica	LOS/ NLOS	D
Oficina Grande	LOS/ NLOS	E
Espacios Grandes (interior/exterior)	LOS/ NLOS	F

Tabla 2.1 Clasificación y asignación de Modelos. [18]

El modelo de Saleh-Valenzuela se utiliza en ambientes que presentan componentes multitrayectoria, es decir se tiene señales que llegan al receptor a diferentes tiempos producidos por obstáculos presentes entre el transmisor y receptor.

En dicho estudio se define el concepto de *taps* que son las componentes de la señal, expresadas en pulsos de potencia, que se generan cuando existen multitrayectorias. Un conjunto de taps forma un *clúster*[28].

La Tabla 2.2 muestra la cantidad de *clústers* por modelo, como se cita en [18] para un ancho de banda de canal de 40MHz.

[25] Desvanecimiento Plano: El canal conserva el espectro de la señal transmitida y trata a todas las frecuencias por igual.
[26] LOS (*Line Of Sight*, Línea de Vista): Escenario donde existe un trayecto dominante entre el receptor y transmisor
[27] NLOS (*Non Line Of Sight*, Sin Línea de Vista): Escenario donde no existe un trayecto dominante entre el receptor y transmisor
[28] Clúster: Es la agrupación de múltiples componentes de potencia de la señal que llegan al receptor.

Modelo	Número de Clústers
A	1
B	2
C	2
D	3
E	4
F	6

Tabla 2.2 Número de clústers por modelo.

Los modelos expuestos describen el comportamiento del canal por donde viaja la señal y explican las características de los *taps* que se originan en la transmisión. Los *taps* tienen un valor de potencia así como un tiempo de retardo específico al arribar al receptor. El número de *taps* depende de cada modelo, los cuales tienen valores de AoA (*Angle of Arrival*, Ángulo de Arribo) y AoD (*Angle of Departure,* Ángulo de Partida) definidos en [18].

2.3.2 MODELO DE CANAL 802.11ac

El TGn de la IEEE desarrolló el modelo de canal 802.11n compatible con MIMO y canales de 40 MHz. El TGac describe las modificaciones a los modelos de canal propuestos por TGn para generar el modelo de canal 802.11ac. [19]

2.3.2.1 Modificaciones para ampliar el ancho de banda del sistema

Los modelos de canal propuestos por TGn asumen un valor de 10 nseg como separación entre *taps* y se utilizan en sistemas con ancho de banda de canal de hasta 40 MHz. Para el modelo de canal 802.11ac, que cuenta con anchos de banda de canal más amplios, el espaciamiento entre *taps* del PDP[29] (*Power Delay Profile,* Perfil de Retardo de Potencia) se dividirá por un factor (k) igual a $2^{\left(log_2 \frac{BW}{40}\right)}$, donde BW es el nuevo ancho de banda en MHz. [19]. Estos valores se resumen en la Tabla 2.3 para un ancho de banda de hasta 160 MHz.

[29] PDP: Determina la forma en la que se distribuye la potencia de la señal entre los distintos trayectos que lo componen.

Ancho de Canal	Factor k	Separación entre Taps
BW ≤ 40 MHz	1	10 nseg
40 MHz < BW ≤ 80 MHz	2	5 nseg
80 MHz < BW ≤ 160 MHz	4	2.5 nseg

Tabla 2.3 Separación entre *taps* para 40, 80 y 160 MHz

El hecho de que exista un nuevo espaciamiento entre *taps* conlleva a que se generen nuevos *taps.* Estos nuevos *taps* se generan basándose en una interpolación lineal[30] [19] en los PDP definidos en TGn. La Figura 2.4 ilustra el procedimiento para obtener los nuevos *taps* en un ancho de canal de 160 MHz (k=4) para un hipotético par de *taps* separados 10ns. Los *taps* TGn son las flechas negras mostradas a los extremos y los nuevos *taps* interpolados se indican mediante las flechas azules. En este caso, 3 nuevos *taps* son añadidos: 2.5 nseg, 5.0 nseg y 7.5 nseg después del TGn tap i-esimo. La potencia para cada nuevo PDP *tap* se deriva de la línea que conecta la potencia de los *taps* i-ésimo e i-ésimo+1 usando la interpolación lineal descrita anteriormente. Este procedimiento se realiza para todos los TGn PDP taps i-ésimo para i=1 a (n_taps-1), donde n_taps es el número de PDP *taps* del *clúster* interpolado.

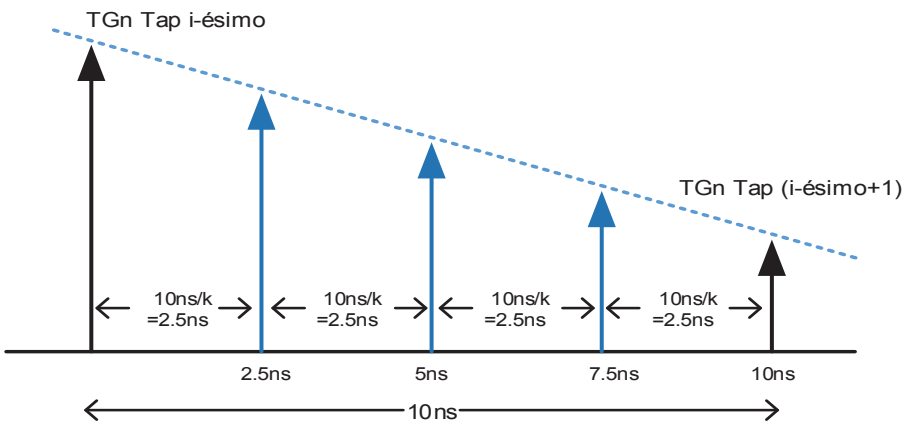


Figura 2.3 Agregación de nuevos *Taps.* [19]

A continuación, se presenta los resultados para el *clúster* 1 del modelo B, en donde los valores de la Tabla 2.4 son los descritos en [18] para anchos de banda de canal

[30] Interpolación Lineal: Es un procedimiento mediante el cual se obtiene un valor dentro de un intervalo, conociendo los valores extremos.

de 40 MHz, mientras que los valores de las Tabla 2.5 y 2.6 se obtiene a partir de la interpolación lineal para 80 MHz y 160 MHz respectivamente, los mismos que están sombreados de color amarillo.

40 MHz	**Número de tap**	1	2	3	4	5
	Retardo Taps [nseg]	0	10	20	30	40
	Potencia [dBm]	0	-5.4	-10.8	-16.2	-21.7

Tabla 2.4 Taps para 40 MHz - Modelo B

80 MHz	**Número de tap**	1	2	3	4	5	6	7	8	9
	Retardo Taps [nseg]	0	5	10	15	20	25	30	35	40
	Potencia [dBm]	0	-2.7	-5.4	-8.1	-10.8	-13.5	-16.2	-18.95	-21.7

Tabla 2.5 Taps para 80 MHz - Modelo B

160 MHz	**Número de tap**	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	Retardo Taps [nseg]	0	2.5	5	7.5	10	12.5	15	17.5	20	22.5	25	27.5	30	32.5	35	37.5	40
	Potencia [dBm]	0	-1.35	-2.7	-4.05	-5.4	-6.75	-8.1	-9.45	-10.8	-12.15	-13.5	-14.85	-16.2	-17.57	-18.95	-20.32	-21.7

Tabla 2.6 Taps para 160 MHz - Modelo B

En los anexos A, B y C se presentan los valores de *taps* para los modelos A, B y C.

2.3.2.2 **MIMO de orden superior**

Los modelos de canal 802.11n fueron originalmente concebidos para sistemas MIMO 4*4, y se basan en el modelo de correlación de canal de Kronecker. Los modelos de canal 802.11ac utilizan el modelo idéntico de correlación de canal de Kronecker para simulaciones de canales MIMO de orden superior. [19]

2.3.2.2.1 Correlación Espacial y Modelo de Kronecker

La correlación se presenta como un parámetro clave en el uso de múltiples antenas dado que condiciona la eficiencia de los sistemas de diversidad. La correlación como un parámetro matemático se explica en el anexo D, en esta sección se busca explicar la correlación espacial para indicar el grado de similitud estadística entre dos canales cuyas antenas transmisoras y/o receptoras se encuentran separadas una cierta distancia. [20]

La Figura 2.4 presenta un escenario formado por dos enlaces, los cuales tienen trayectorias diferentes siendo, por lo tanto, no correlacionados entre sí.

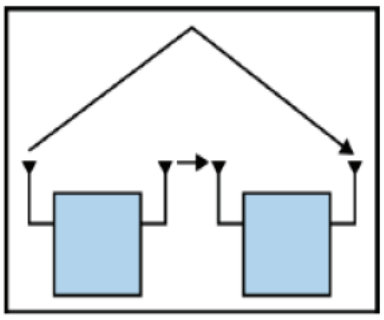


Figura 2.4 Enlaces no correlacionados [21]

La correlación está influenciada por diversos factores, principalmente por las características físicas de las antenas en el transmisor y en el receptor, por ejemplo: la topología de los arreglos de antena transmisor y receptor, la separación entre las antenas, el AoA y AoD, el diagrama de radiación, la dispersión angular, entre otros. [22]. La dispersión angular es un parámetro que permite determinar cómo se agrupan las múltiples trayectorias sobre una dirección.

Para un sistema MIMO formado por 2 antenas receptoras (m) y dos antenas transmisoras (n) (ver Figura 2.5), donde d es la separación de las antenas transmisoras y Δr es la separación de las antenas receptoras. Cada uno de los subcanales h_{mn} relacionan las antenas transmisoras con las antenas receptoras.

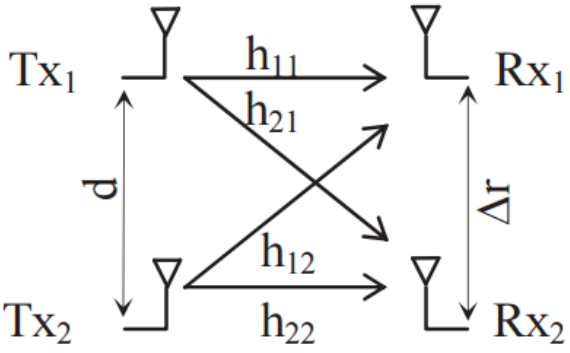


Figura 2.5 Sistema MIMO 2x2 [23]

El grado de independencia entre los enlaces está definido por la matriz de correlación R, que para el sistema MIMO 2x2 es:

$$\boldsymbol{R} = \begin{vmatrix} \rho_{11}^{11} & \rho_{11}^{12} & \rho_{11}^{21} & \rho_{11}^{22} \\ \rho_{12}^{11} & \rho_{12}^{12} & \rho_{12}^{21} & \rho_{12}^{22} \\ \rho_{21}^{11} & \rho_{21}^{12} & \rho_{21}^{21} & \rho_{21}^{22} \\ \rho_{22}^{11} & \rho_{22}^{12} & \rho_{22}^{21} & \rho_{22}^{22} \end{vmatrix} \qquad (2.8)$$

Donde ρ_{ij}^{pq} representa el coeficiente de correlación entre h_{ij} y h_{pq}, donde i, p son antenas receptoras y j, q son antenas transmisoras.

Si se considera el comportamiento de los canales como un proceso estacionario[31], la correlación depende únicamente de la distancia de separación entre las antenas y no de su posición por lo que se puede expresar la matriz de correlación de la siguiente manera:

$$\boldsymbol{R} = \begin{vmatrix} 1 & \rho_{11}^{12}(\Delta r) & \rho_{11}^{21}(0) & \rho_{11}^{22}(\Delta r) \\ \rho_{12}^{11}(\Delta r) & 1 & \rho_{12}^{21}(-\Delta r) & \rho_{12}^{22}(0) \\ \rho_{21}^{11}(0) & \rho_{21}^{12}(-\Delta r) & 1 & \rho_{21}^{22}(\Delta r) \\ \rho_{22}^{11}(\Delta r) & \rho_{22}^{12}(0) & \rho_{22}^{21}(\Delta r) & 1 \end{vmatrix} \qquad (2.9)$$

El coeficiente de correlación viene dado por la siguiente expresión:

$$\rho_{x,y} = \langle x|y\rangle = \frac{E[x.y]-E[x].E[y]}{\sqrt{(E[|x|^2]-|E[x]^2|).(E[|y|^2]-|E[y]^2|)}} \qquad (2.10)$$

Donde x, y son las envolventes de la tensión medida, correspondiente a h_{ij} y h_{pq} respectivamente y $E[\,]$ es el valor esperado.

El modelo de Kronecker o de correlación separable asume que la correlación en transmisión es independiente del receptor, la correlación en recepción es independiente del transmisor, AoA y AoD se suponen separables, por lo tanto la ecuación de la matriz de correlación se obtiene mediante la ecuación 2.11 [19]:

[31] Proceso Estacionario: Todos los elementos que actúan en la transmisión permanecen fijos

$$\mathbf{R} = R_t \otimes R_r \qquad (2.11)$$

Donde:

$\otimes$: Representa el producto de Kronecker.

N_t : Número de antenas transmisoras.

N_t : Número de antenas receptoras.

R_t : Representa la matriz de correlación en el transmisor, de tamaño $N_t - por - N_t$.

R_r : Representa matriz de correlación en el receptor, de tamaño $N_r - por - N_r$.

2.3.2.3 **Modificaciones en AoA y AoD para MU-MIMO**

El parámetro AoA define el ángulo con el que llegan las componentes de la señal a los receptores con respecto a un plano de referencia, por otro lado, el ángulo AoD establece el valor del ángulo de salida de la señal desde el transmisor.

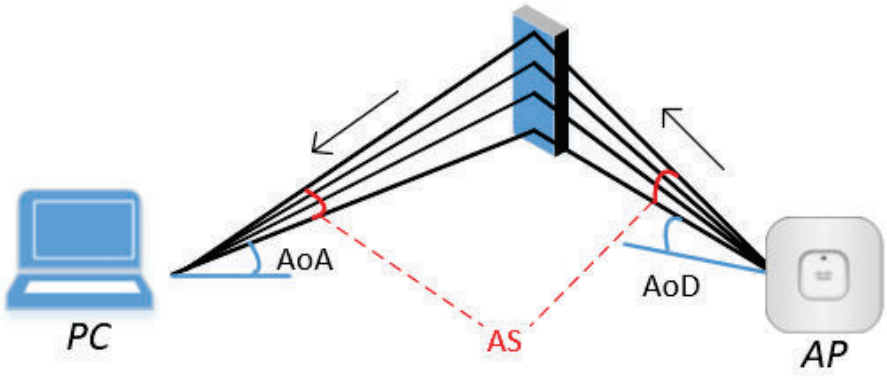


Figura 2.6 Ángulos AoA y AoD.

En [18] se especifica los *clústers* AoAs y AoDs para transmisiones SU-MIMO. En 802.11ac se requiere implementar la tecnología MU-MIMO, para lograrlo se debe generar ángulos *offset*[32] dependiendo de la cantidad de usuarios, estos ángulos se adicionarán a los valores AoA y AoD ya existentes en 802.11n.

[32] Ángulos de Offset: Desplazamientos pseudo aleatorios que se añaden a los *clusters* AoAs y AoDs definidos en TGn. Simulan la diversidad de la posición y orientación del usuario con respecto al AP.

Los canales multiusuario son modelados de acuerdo a la siguiente modificación para cada cliente:

- Aplicar un desfase aleatorio uniforme de ±180° a los ángulos AoA y AoD de cada modelo para condiciones de LOS y NLOS.

2.3.2.4 Generación del canal MU-MIMO

La Figura 2.7 representa un escenario MU-MIMO *downlink*, que consta de un AP y tres estaciones separadas espacialmente. Se asume que el AP tiene N antenas transmisoras (Ntx), las tres estaciones poseen N antenas receptoras (Nrx1, Nrx2 y Nrx3 respectivamente). En el canal MU-MIMO se puede considerar que se tiene tres canales SU-MIMO independientes entre el AP y cada estación, denotado como H1, H2 y H3 respectivamente. Cada canal SU-MIMO asume diferentes correlaciones espaciales, tanto para el AP como para las estaciones. La correlación espacial entre las antenas de diferentes estaciones se asume como cero.

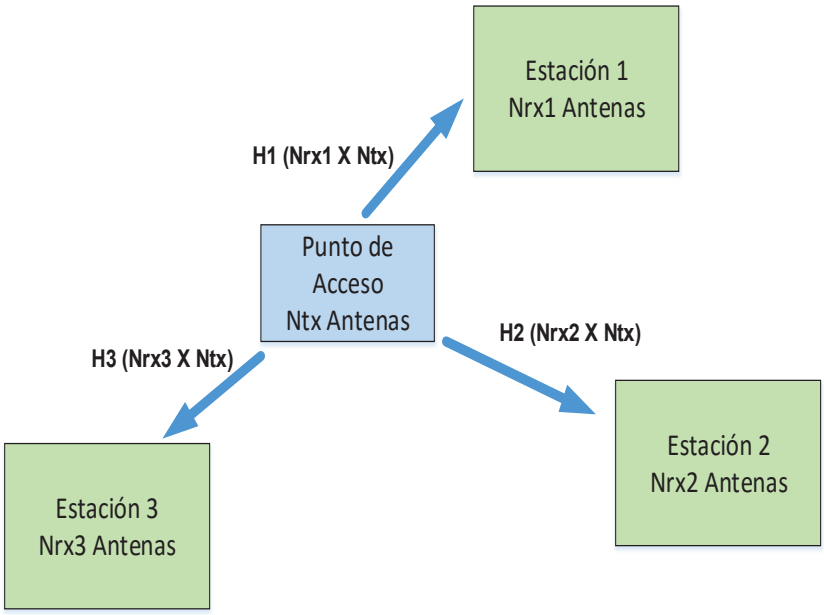


Figura 2.7 Escenario Multi-user MIMO (MU-MIMO) [19]

Para generar el canal MU-MIMO se agrupan los canales SU-MIMO de cada estación (H1, H2, H3), obteniendo una matriz del tamaño $(N_{rx1} + N_{rx2} + N_{rx3})x\ N_{tx}$ En la Figura 2.8 se presenta la matriz MU-MIMO a obtenerse.

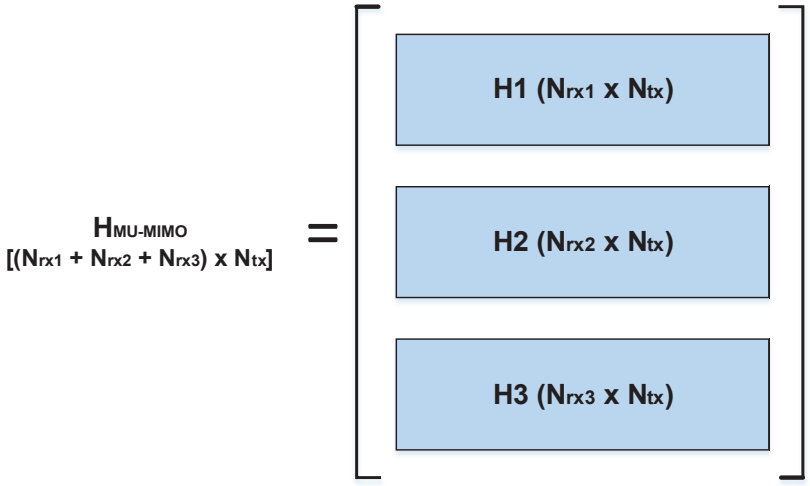


Figura 2.8 Matriz MU-MIMO [19]

2.4 SIMULACIÓN DEL ESTÁNDAR IEEE 802.11AC

2.4.1 CONDICIONES PARA LA SIMULACIÓN

Para construir la simulación del estándar IEEE 802.11ac es importante definir los parámetros y asunciones a considerar:

- Para el desarrollo de la simulación se eligieron los modelos A, B y C (citados en las Tablas 2.1 y 2.2) estos permiten condiciones de LOS o NLOS por el escenario en el que fueron desarrollados. Además de contar con las variables suficientes para la simulación de un canal de comunicaciones con modelos de desvanecimiento de *Rician* y *Rayleigh*. Los modelos elegidos son representativos y ayudan a que la simulación no se extienda demasiado.

  El modelo A consta de un solo *tap* que cubre escenarios *flat fading*, mientras que B y C son selectivos en frecuencia. Se aplicará un desvanecimiento tipo *Rayleight* para los modelos A y C. Al modelo B le corresponderá un desvanecimiento tipo *Rician.*

- Se utiliza el modelo de canal 802.11ac expuesto en 2.3.2

- Se dispone de un máximo de 4 usuarios como determina el estándar 802.11ac.
- Se establece un máximo de 3 *spatial streams* por usuario y 8 *spatial streams* en el transmisor (AP).
- El número de antenas en el transmisor será igual a la suma de las antenas de cada receptor.
- Se emplean anchos de banda de canal de 20, 40, 80 y 160 MHz.
- Los índices MCS están en el rango de 0 a 9. Este dato conjuntamente con la cantidad de *spatial streams*, el número de subportadoras de datos OFDM y el ancho de banda de canal permiten calcular el valor de *data rate*[33] del sistema.
- El diagrama de bloques del sistema propuesto para el análisis de 802.11ac con MU-MIMO se presenta en la Figura 2.9:

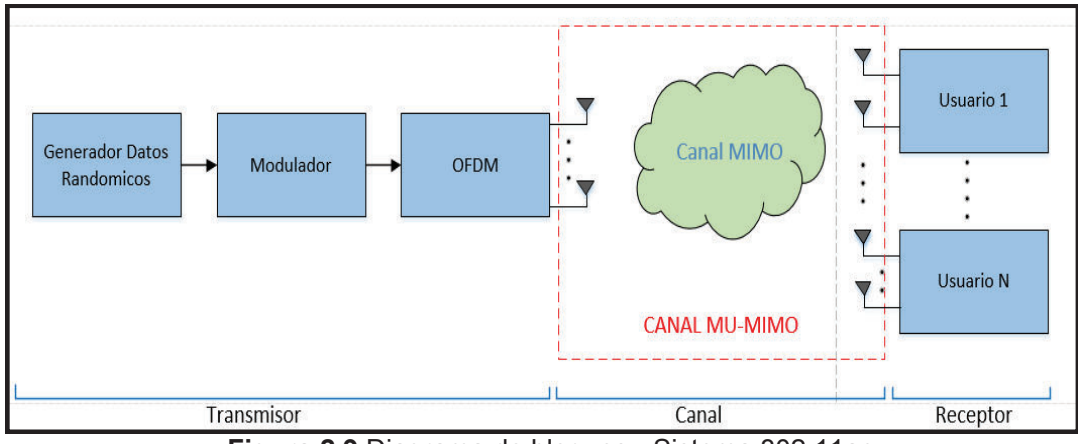


Figura 2.9 Diagrama de bloques - Sistema 802.11ac

2.4.2 ESTRUCTURA DEL PROGRAMA

La simulación del estándar 802.11ac está compuesta por funciones que realizan una tarea específica, para tener una idea clara de la estructura del programa a continuación se presenta el diagrama de flujo del mismo.

[33] Data Rate: Es la tasa máxima de transmisión de datos en una red, expresada en bits por segundo.

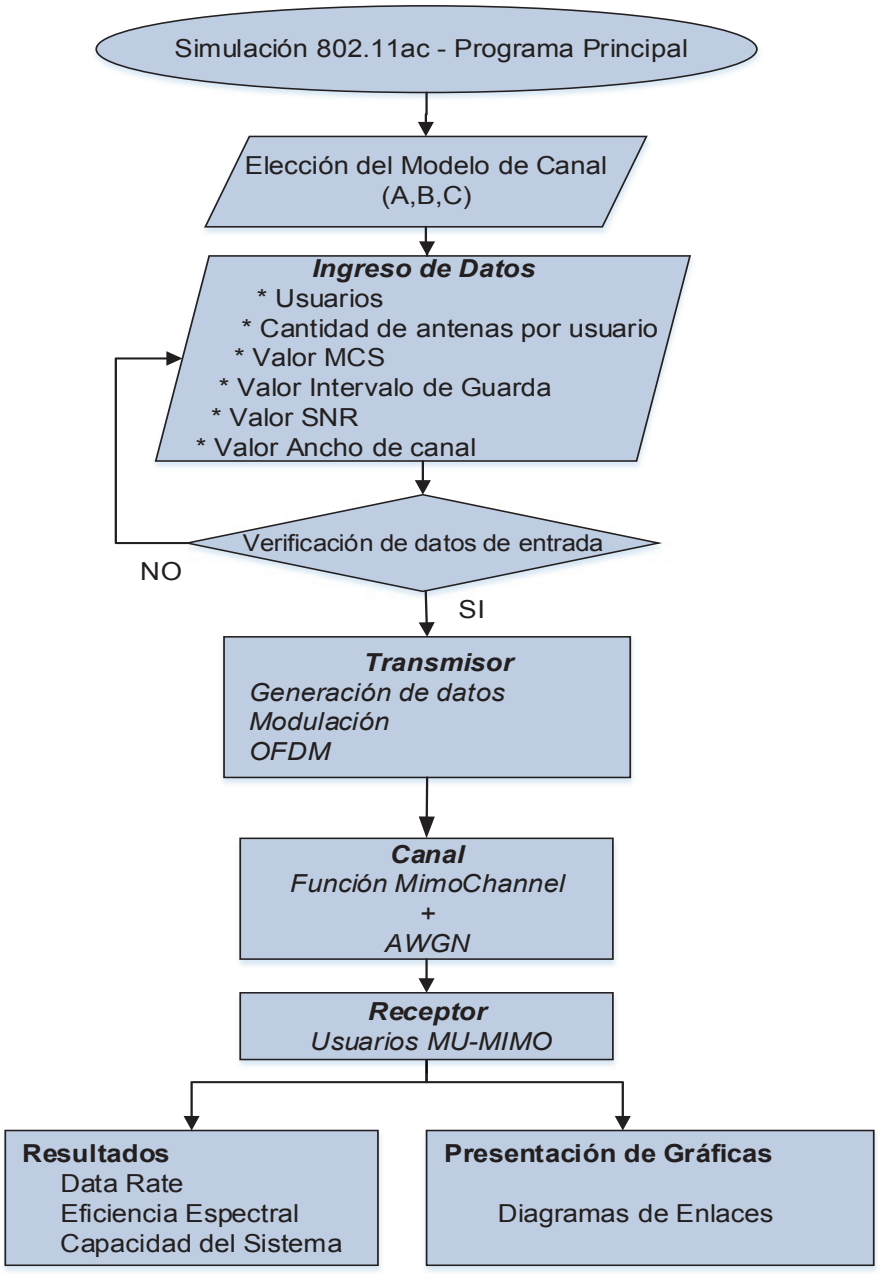


Figura 2.10 Diagrama de Flujo – Simulación 802.11ac

La Figura 2.11 presenta la distribución y el orden en que se ejecutan las funciones en cada uno de los modelos a simular. Las funciones comunes a todos los modelos son las mostradas en color azul.

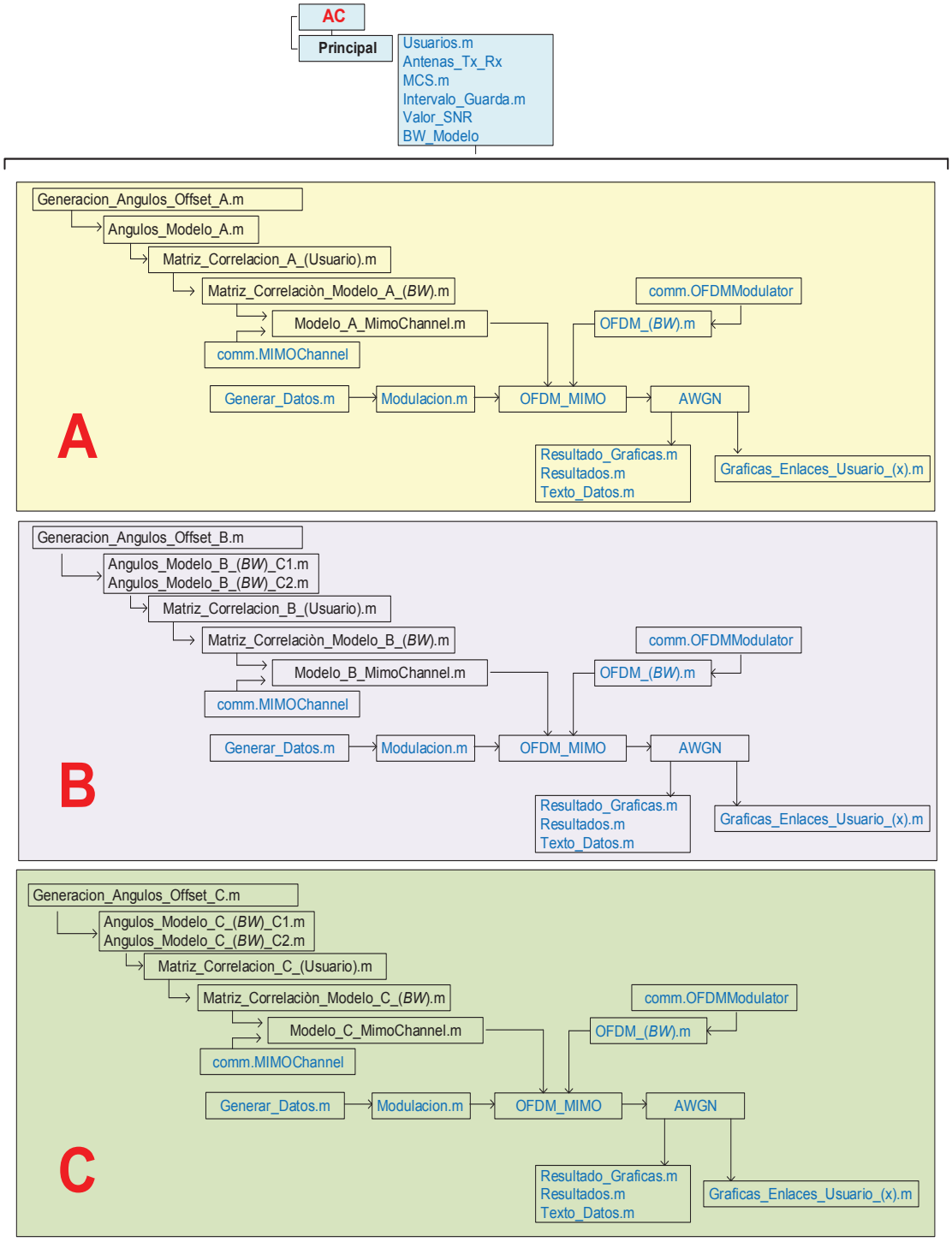


Figura 2.11 Funciones del Programa 802.11ac

El código fuente de las funciones presentadas en la Figura 2.11 se encuentra en el anexo E.

Para iniciar la simulación del Estándar IEEE 802.11ac, desde el *Command Window*[34] de Matlab® se debe digitar las letras “AC”, instantáneamente se muestra una presentación como la observada en la Figura 2.12.

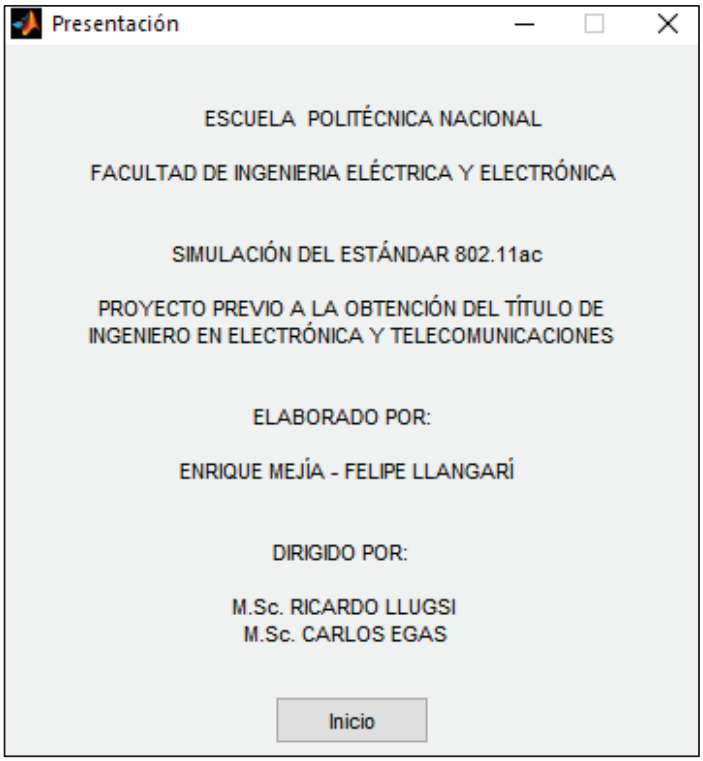


Figura 2.12 Presentación de la simulación 802.11ac

2.4.2.1 Función Principal

La función Principal realiza la elección del modelo de canal (A, B o C), los cuales permiten simular entornos con las condiciones de LOS/NLOS como se expone en la Tabla 2.1.

Esta función verifica el dato ingresado por teclado, y una vez seleccionado el modelo se encarga de “llamar” a las funciones propias y comunes de cada modelo de canal. El diagrama de flujo de esta función se presenta en la Figura 2.13.

[34] Command Window: Ventana donde se ejecutan las instrucciones de Matlab® y se presentan los resultados de ser el caso.

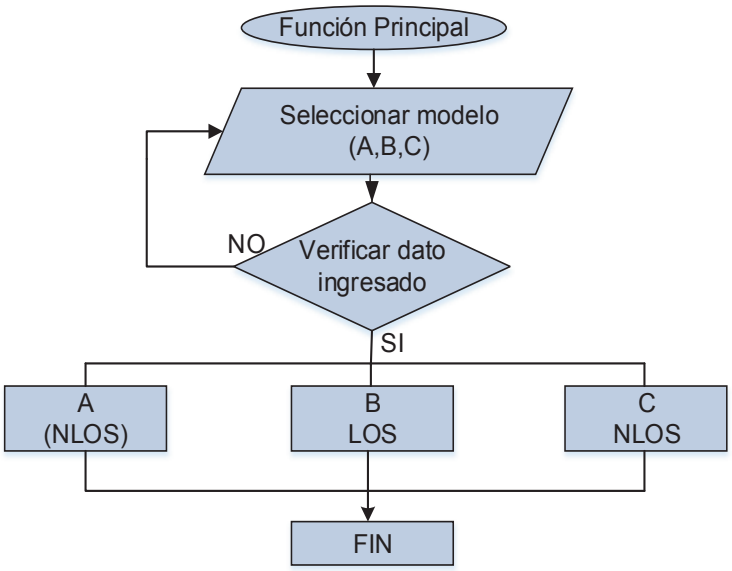


Figura 2.13 Diagrama de Flujo – Programa Principal

2.4.2.2 **Función Usuarios**

En esta función se ingresa la cantidad de usuarios (1-4) donde 4 es la cantidad máxima que permite el estándar 802.11ac para poder representar MU-MIMO, en caso de exceder o errar en estos valores se debe reintentar el ingreso, la Figura 2.14 indica el proceso de esta función.

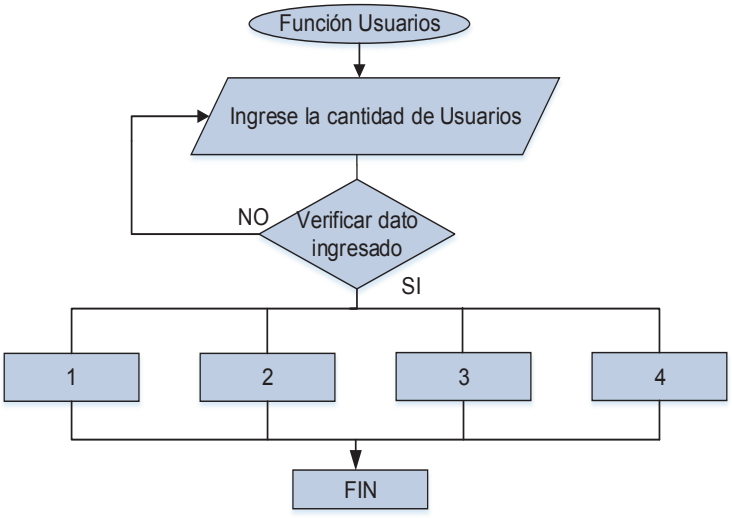


Figura 2.14 Diagrama de Flujo – Función Usuarios

2.4.2.3 **Función Antenas_Tx_Rx**

La función Antenas_Tx_Rx define el número de antenas según el valor ingresado en la función Usuarios, con la condición de 3 antenas máximo por usuario y, donde el número de antenas en el AP es la suma de las antenas de cada usuario.

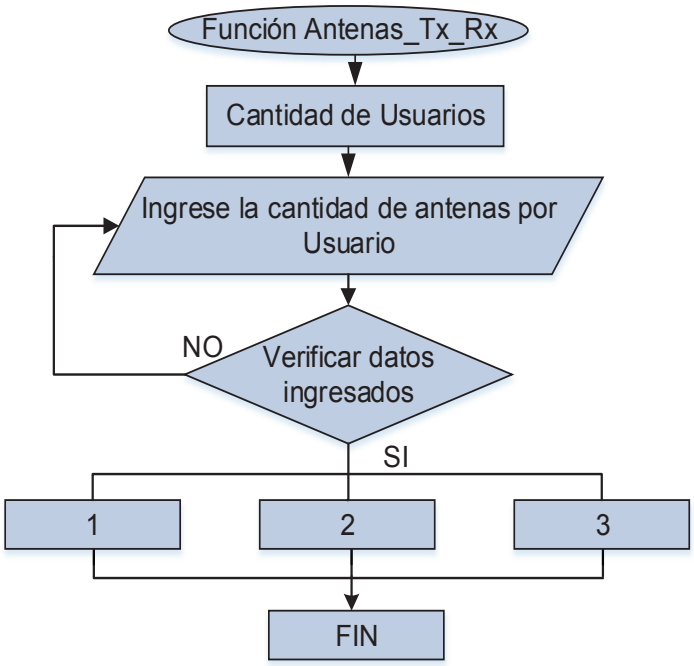


Figura 2.15 Diagrama de Flujo – Función Antenas_Tx_Rx

2.4.2.4 **Función MCS**

La función MCS permite ingresar el índice MCS disponible para el estándar 802.11ac, estos valores se explican en la Tabla 1.5. La Figura 2.16 representa el diagrama de flujo de esta función.

Los resultados que se obtiene según el valor ingresado son los siguientes:

- Tipo de Modulación (Mod)
- Número de bits codificados por subportadora (Nbpscs)
- Valor de Codificación (Coding)

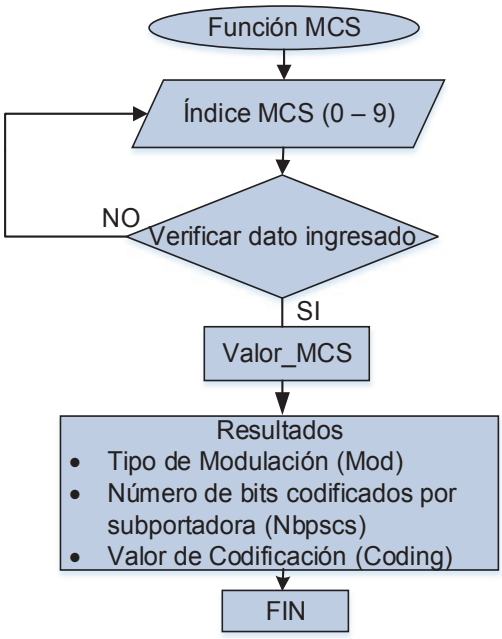


Figura 2.16 Diagrama de Flujo – Función MCS

2.4.2.5 **Función Intervalo de Guarda**

En esta función se solicita que se ingrese el valor del intervalo de guarda, para 802.11ac se tiene dos valores como se menciona en 1.2.6.3. Además, la función consta de una sentencia que verifica el valor ingresado, la Figura 2.17 representa el proceso que realiza esta función.

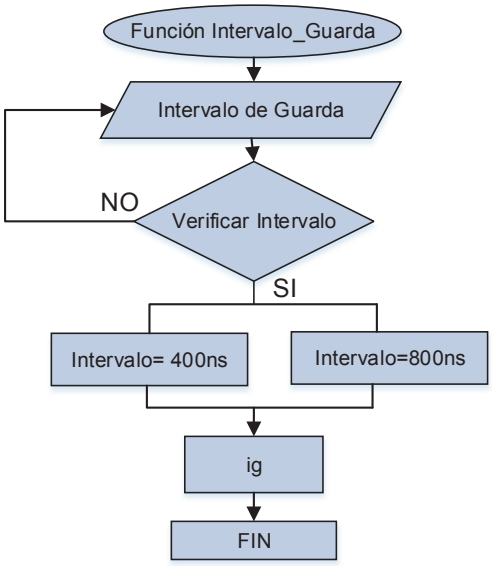


Figura 2.17 Diagrama de Flujo – Función Intervalo_Guarda

2.4.2.6 **Función Valor_SNR**

La función Valor_SNR solicita el ingreso del valor de SNR. Tomando como referencia la Figura 2.19 este valor debe estar entre 0 y 50 dB. La figura además muestra los valores de MCS que se pueden alcanzar para diferentes valores de SNR. El valor de SNR se utiliza para el cálculo de capacidad del canal y en la función AWGN, la Figura 2.18 representa un diagrama de flujo del proceso de la función.

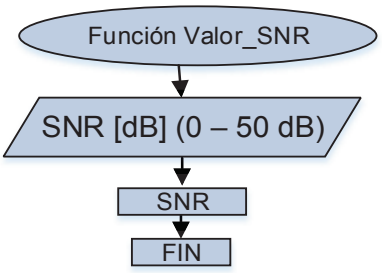


Figura 2.18 Diagrama de Flujo – Función Valor_SNR

Estándar	BW	1	2	3	4	5	6	7	8	9	10
802.11ac	20MHz	None	MCS 0	MCS 0	MCS 0	MCS 1	MCS 1	MCS 1	MCS 1	MCS 2	MCS 2
802.11ac	40MHz	None	None	None	None	MCS 0	MCS 0	MCS 0	MCS 1	MCS 1	MCS 1
802.11ac	80MHz	None	None	None	None	None	None	None	MCS 0	MCS 0	MCS 0
802.11ac	160MHz	None	None	None	None	None	None	None	None	None	None
	SNR in dB	11	12	13	14	15	16	17	18	19	20
802.11ac	20MHz	MCS 3	MCS 3	MCS 3	MCS 3	MCS 4	MCS 4	MCS 4	MCS 5	MCS 5	MCS 6
802.11ac	40MHz	MCS 1	MCS 2	MCS 2	MCS 3	MCS 3	MCS 3	MCS 3	MCS 4	MCS 4	MCS 4
802.11ac	80MHz	MCS 1	MCS 1	MCS 1	MCS 1	MCS 2	MCS 2	MCS 3	MCS 3	MCS 3	MCS 3
802.11ac	160MHz	MCS 0	MCS 0	MCS 0	MCS 1	MCS 1	MCS 1	MCS 1	MCS 2	MCS 2	MCS 3
	SNR in dB	21	22	23	24	25	26	27	28	29	30
802.11ac	20MHz	MCS 6	MCS 6	MCS 6	MCS 6	MCS 7	MCS 7	MCS 7	MCS 7	MCS 8	MCS 8
802.11ac	40MHz	MCS 5	MCS 5	MCS 6	MCS 6	MCS 6	MCS 6	MCS 6	MCS 7	MCS 7	MCS 7
802.11ac	80MHz	MCS 4	MCS 4	MCS 4	MCS 5	MCS 5	MCS 6	MCS 6	MCS 6	MCS 6	MCS 6
802.11ac	160MHz	MCS 3	MCS 3	MCS 3	MCS 4	MCS 4	MCS 4	MCS 5	MCS 5	MCS 6	MCS 6
	SNR in dB	31	32	33	34	35	36	37	38	39	40
802.11ac	20MHz	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9
802.11ac	40MHz	MCS 7	MCS 8	MCS 8	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9
802.11ac	80MHz	MCS 7	MCS 7	MCS 7	MCS 7	MCS 8	MCS 8	MCS 9	MCS 9	MCS 9	MCS 9
802.11ac	160MHz	MCS 6	MCS 6	MCS 6	MCS 7	MCS 7	MCS 7	MCS 7	MCS 8	MCS 8	MCS 9
	SNR in dB	41	42	43	44	45	46	47	48	49	50
802.11ac	20MHz	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9
802.11ac	40MHz	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9
802.11ac	80MHz	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9
802.11ac	160MHz	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9	MCS 9

Tipo de Modulación
Ninguno = Plomo
BPSK = Rojo
QPSK = Naranja
16-QAM = Amarillo
64-QAM = Azul
256-QAM = Verde

Figura 2.19 Valor de MCS en función de SNR [24]

2.4.2.7 **Función BW_Modelo**

La función BW_Modelo se utiliza para ingresar el valor de ancho de banda de canal y se dispone de una función para cada modelo que se denominan de la siguiente manera:

- Función BW_Modelo_A
- Función BW_Modelo_B
- Función BW_Modelo_C

2.4.2.7.1 Función BW_Modelo_A

En esta función se ingresa el valor de ancho de banda de canal (20, 40, 80, 160 MHz), y permite la ejecución de las funciones que dependen del valor ingresado, en el diagrama de flujo de la Figura 2.20 se presenta el proceso que sigue esta función.

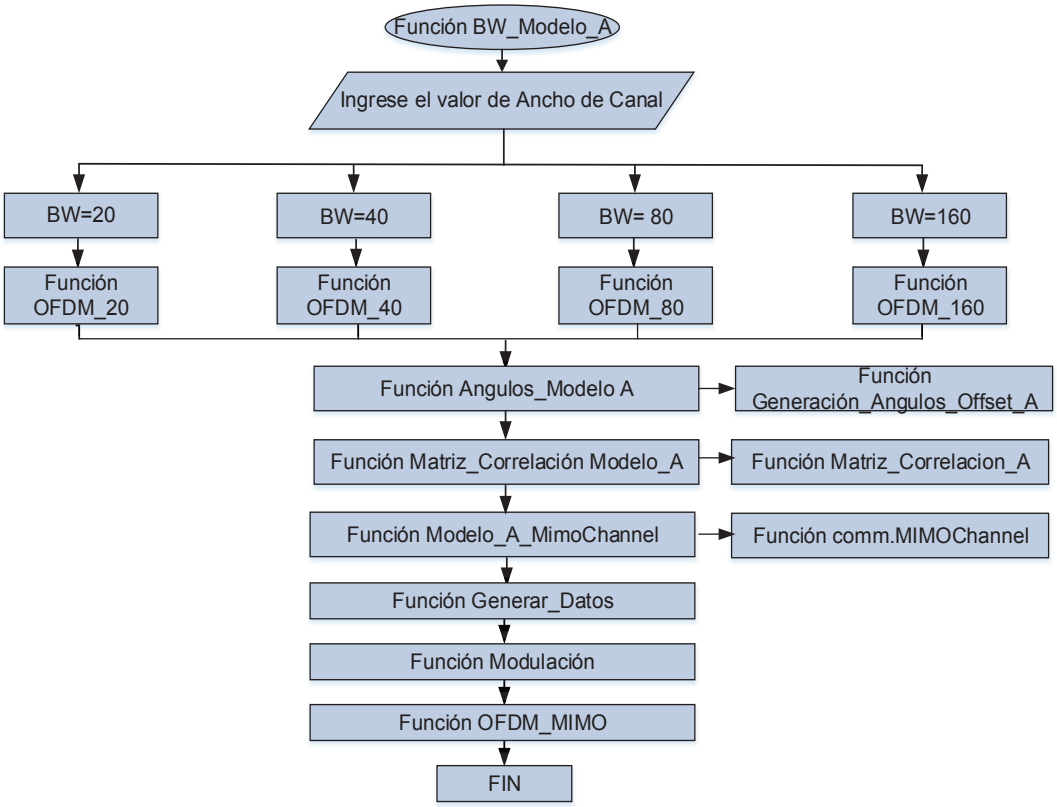


Figura 2.20 Diagrama de Flujo – Función BW_Modelo_A

2.4.2.7.2 Función BW_Modelo_B

En esta función se ingresa el valor de ancho de banda de canal (20, 40, 80, 160 MHz) y permite llamar a las funciones que dependen del valor ingresado, en el diagrama de flujo de la Figura 2.21 se explica el orden en que las funciones se ejecutan. El modelo B tiene dos clusters por lo cual las funciones para los ángulos AoA y AoD se originan según el ancho de banda y el cluster.

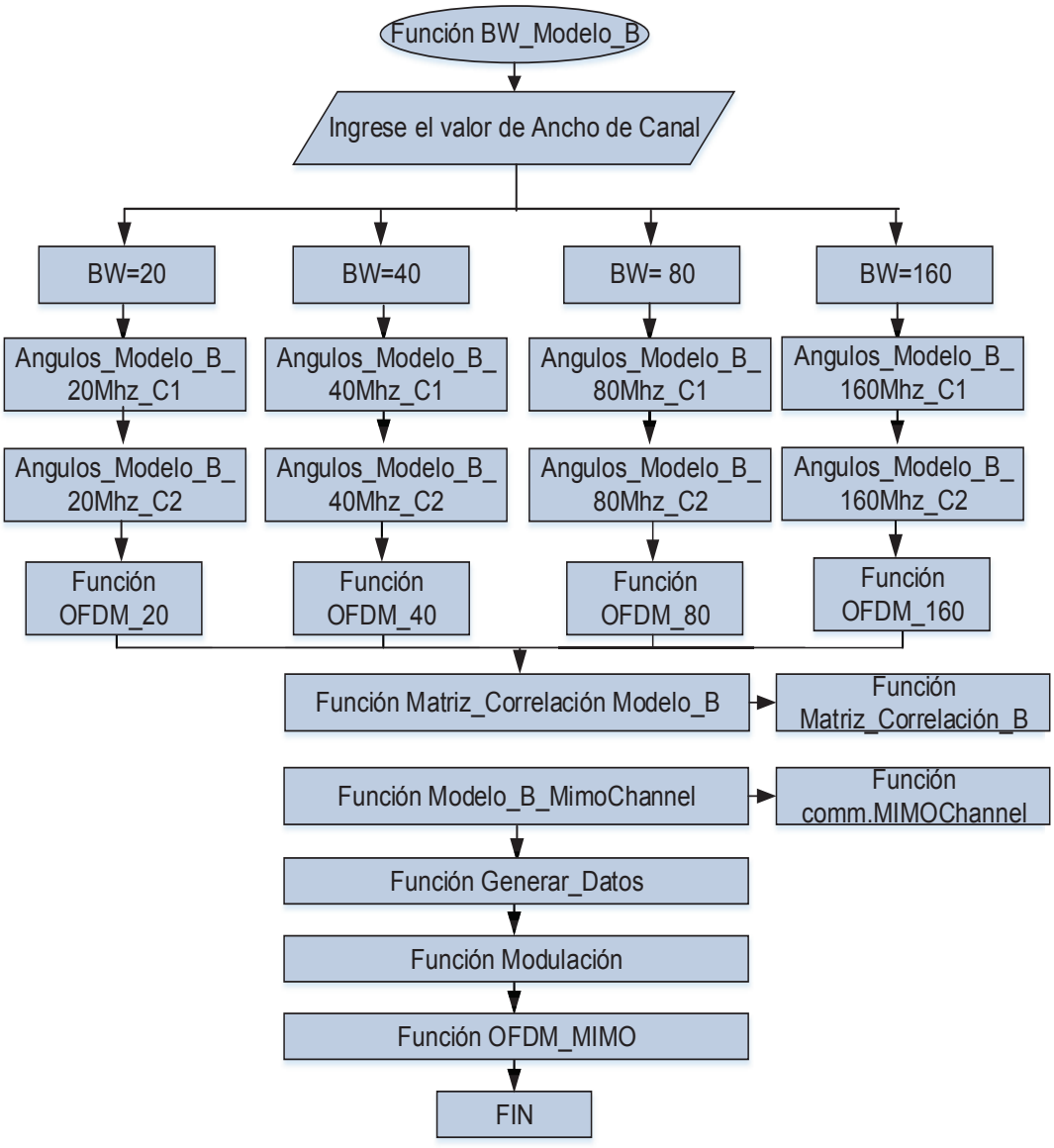


Figura 2.21 Diagrama de Flujo – Función BW_Modelo_B

2.4.2.7.3 Función BW_Modelo_C

La función BW_Modelo_C permite el ingreso del valor de ancho de banda de canal (20, 40, 80, 160 MHz) y permite llamar a las funciones asociadas al modelo C, en el diagrama de flujo de la Figura 2.22 se explica el orden en que las funciones se ejecutan. El modelo C de forma similar al modelo B consta de dos clusters por lo cual las funciones de ángulos AoA y AoD se forman dependiendo el ancho de banda y el cluster.

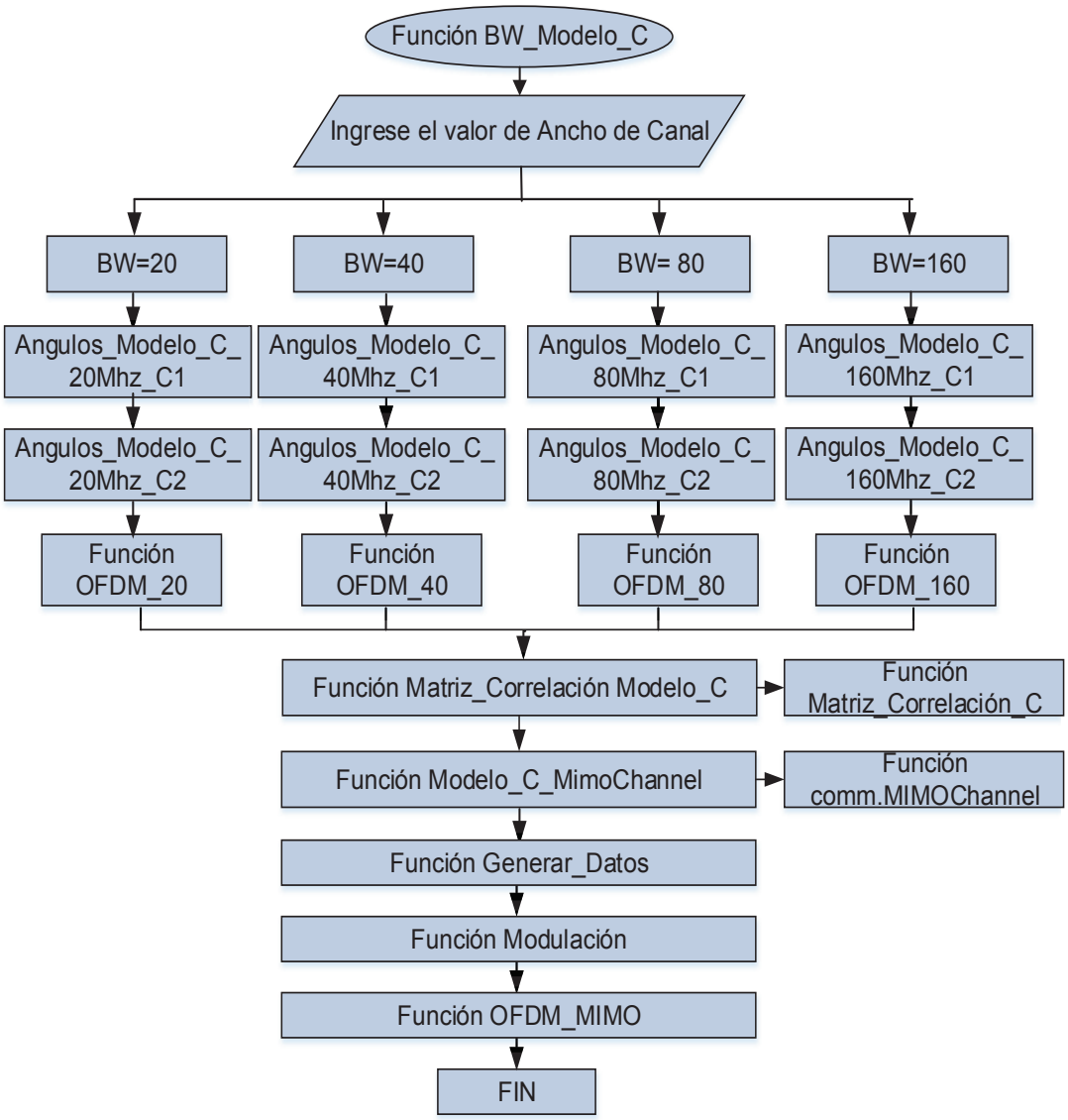


Figura 2.22 Diagrama de Flujo – Función BW_Modelo_C

2.4.2.8 Función Generación_Angulos_Offset

Para simular MU-MIMO, como se gráfica en la Figura 2.7, se debe generar ángulos *offset* para cada usuario, ver apartado 2.3.2.3, los cuales se añaden a los ángulos AoA y AoD establecidos para cada modelo.

Se utiliza un valor diferente para iniciar el generador de números aleatorios. Una secuencia de valores se genera por cada valor de inicialización, un valor por cada cliente MU-MIMO. Los valores de inicialización se determinan de acuerdo a los parámetros que se muestran en la Tabla 2.7:

Ángulo *Downlink*	Ángulo *Uplink*	Valor Inicialización
Ángulo AoD para LOS	Ángulo AoA para LOS	39161
Ángulo AoD para NLOS	Ángulo AoA para NLOS	2803
Ángulo AoA para LOS	Ángulo AoD para LOS	45191
Ángulo AoA para NLOS	Ángulo AoD para NLOS	13367

Tabla 2.7 Valores que genera los ángulos offset. [19]

Las siguientes líneas de comando explican como se generan los ángulos *offset* para 4 usuarios en condiciones LOS:

```
x = 4;                                  %Número de usuarios

generar_valores_AoD = 39161;            % Valor de inicialización.
rango_OFFSET = 360;                     % Rango del ángulo Down Link/AoD (+/-180°).

rand('seed',generar_valores_AoD);       % Generación aleatoria de valores.
valor_OFFSET = [(rand(x,1)-0.5)*rango_OFFSET];  % Operación para generar ángulo OFFSET.
```

Figura 2.23 Línea de comandos para generar ángulos *offset*

La función Generación_Angulos_Offset se establece para cada modelo y se denominan de la siguiente manera:

- Función Generación_Angulos_Offset _A
- Función Generación_Angulos_Offset _B
- Función Generación_Angulos_Offset _C

El diagrama de flujo de la Figura 2.24 muestra el proceso de generación de ángulos *offset,* dicho proceso es común para todos los modelos a excepción del valor de inicialización que se toma de acuerdo a la Tabla 2.7.

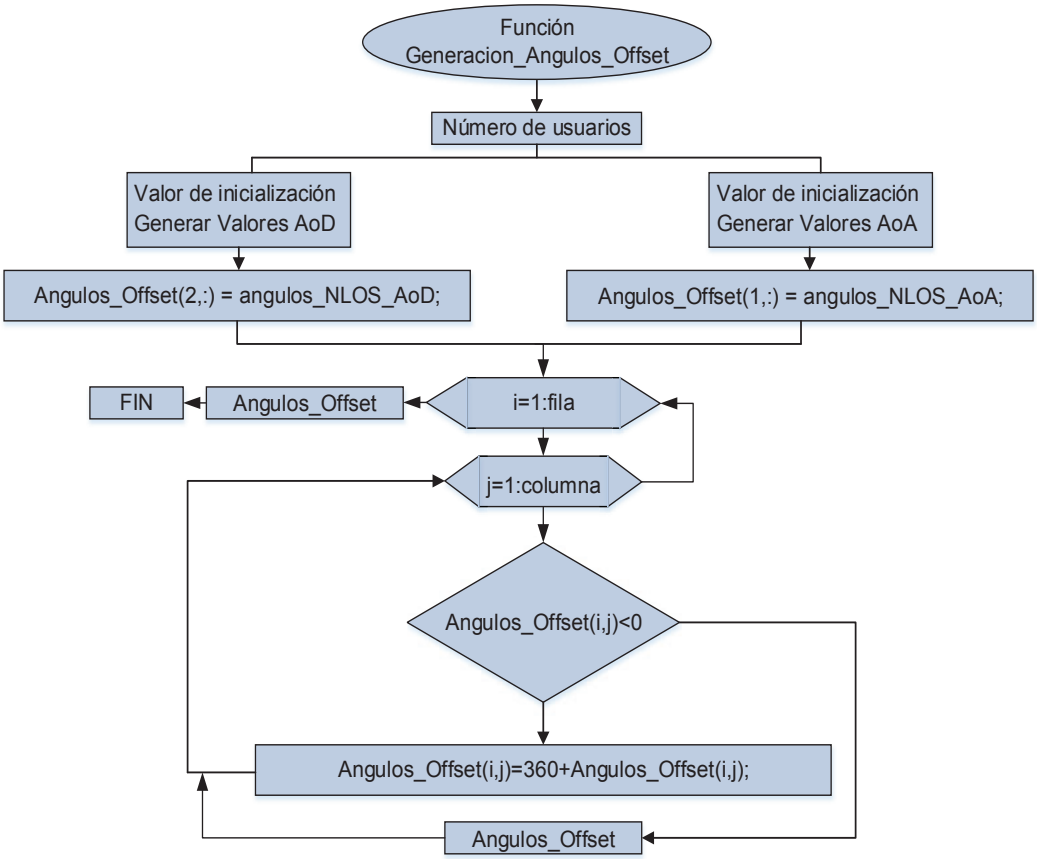


Figura 2.24 Diagrama de Flujo – Función Generación_Angulos_Offset

2.4.2.9 **Función Ángulos_Modelo**

Las funciones “Angulos_Modelo” se asocian a cada modelo y utilizan los resultados de las funciones Generación_Angulos_Offset, se las denomina de la siguiente manera:

- Función Angulos_Modelo_A
- Función Angulos_Modelo_B
- Función Angulos_Modelo_C

2.4.2.9.1 Función Angulos_Modelo_A

La función Angulos_Modelo_A permite adicionar los valores de ángulos offset (obtenidos para cada usuario) a los ángulos AoA y AoD del modelo A, para el caso que se en tenga un usuario los valores de los ángulos serán los mimos a lo establecido en el anexo A.

Este modelo cuenta únicamente de un *tap* por lo cual la función es similar para todos los anchos de banda, en el siguiente diagrama de flujo se puede apreciar el proceso de esta función.

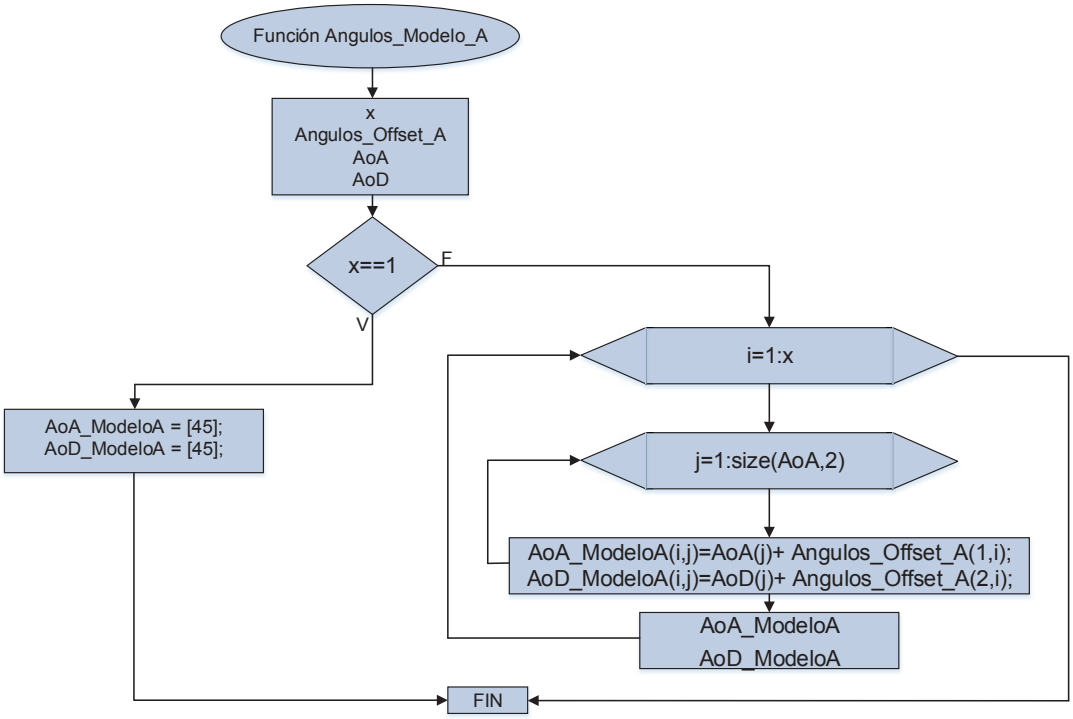


Figura 2.25 Diagrama de Flujo – Función Angulos_Modelo_A

2.4.2.9.2 Función Angulos_Modelo_B

En el modelo B, para añadir los ángulos *offset* de cada usuario a los ángulos AoA y AoD se ha dispuesto de funciones que dependen del valor de ancho de banda de canal y del *clúster* ya que la longitud de los vectores de AoA y AoD son diferentes

para cada caso (ver Tabla 2.8). Las funciones para 20 MHz y 40 MHz son similares pero se ha dispuesto separarlas para diferenciarlas de mejor manera.

Ancho de Banda	Clúster 1 (C1)	Clúster 2 (C2)
20 MHz	Angulos_Modelo_B_20MHz_C1	Angulos_Modelo_B_20MHz_C2
40 MHz	Angulos_Modelo_B_40MHz_C1	Angulos_Modelo_B_40MHz_C2
80 MHz	Angulos_Modelo_B_80MHz_C1	Angulos_Modelo_B_80MHz_C2
160 MHz	Angulos_Modelo_B_160MHz_C1	Angulos_Modelo_B_160MHz_C2

Tabla 2.8 Funciones Ángulos Modelo B

En la Figura 2.26, a manera de ejemplo, se presenta el diagrama de flujo de la función Angulos_Modelo_B_80MHz_C1 en donde se ilustra el proceso que se realiza, este procedimiento siguen de manera similar las funciones de Tabla 2.8, para el caso de que exista un solo usuario los valores de AoA y AoD se mantienen como se presenta en el anexo B.

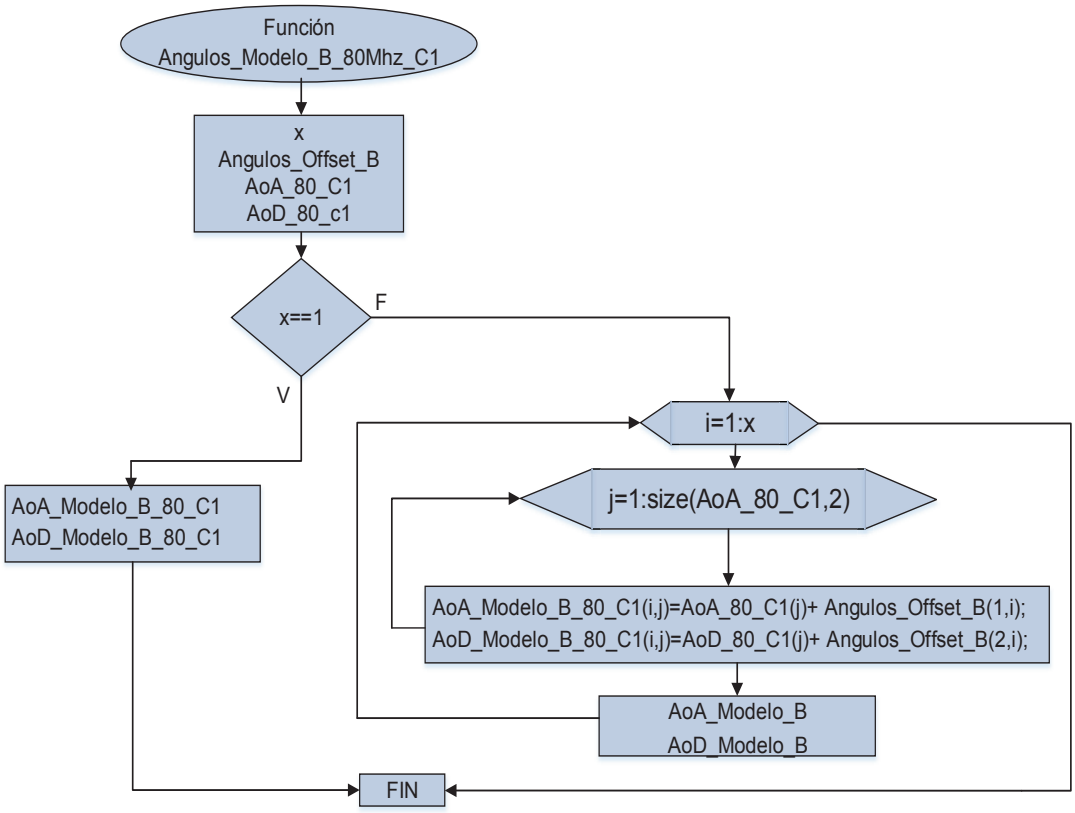


Figura 2.26 Diagrama de Flujo – Función Angulos_Modelo_B_80MHz_C1

2.4.2.9.3 Función Angulos_Modelo_C

En el modelo C para añadir los ángulos *offset* obtenidos para cada usuario a los ángulos AoA y AoD se ha dispuesto de funciones que dependen del valor de ancho de banda de canal y del *clúster*, ya que la longitud de los vectores de AoA y AoD son diferentes para cada caso (ver Tabla 2.9). Las funciones para 20 MHz y 40 MHz son similares pero se ha dispuesto separarlas para diferenciarlas de mejor manera.

Ancho de Canal	Cluster 1 (C1)	Cluster 2 (C2)
20 MHz	Angulos_Modelo_C_20MHz_C1	Angulos_Modelo_C_20MHz_C2
40 MHz	Angulos_Modelo_C_40MHz_C1	Angulos_Modelo_C_40MHz_C2
80 MHz	Angulos_Modelo_C_80MHz_C1	Angulos_Modelo_C_80MHz_C2
160 MHz	Angulos_Modelo_C_160MHz_C1	Angulos_Modelo_C_160MHz_C2

Tabla 2.9 Funciones Ángulos Modelo C

En la Figura 2.27 se presenta el diagrama de flujo de la función Angulos_Modelo_C_80MHz_C1 que explica el proceso que realizan las funciones de la Tabla 2.9. En el caso de que exista un solo usuario los valores de AoA y AoD se mantienen como se presenta en el anexo C.

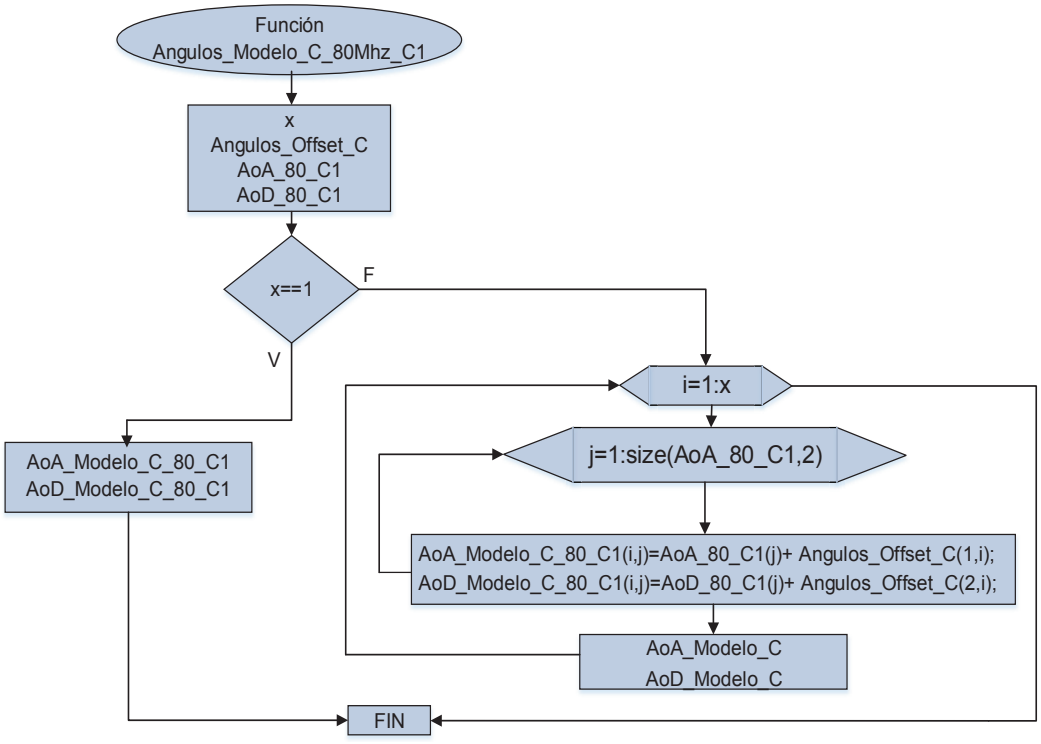


Figura 2.27 Diagrama de Flujo – Función Angulos_Modelo_C_80MHz_C1

2.4.2.10 **Función *comm.OFDMModulator* [25]**

La función *comm.OFDMModulator* es propia de Matlab® 2015b, y simula el efecto de OFDM para el estándar 802.11ac, esta función depende del ancho de banda de canal y del número de antenas de los usuarios. En la Figura 2.28 se indica los tipos de subportadoras OFDM.

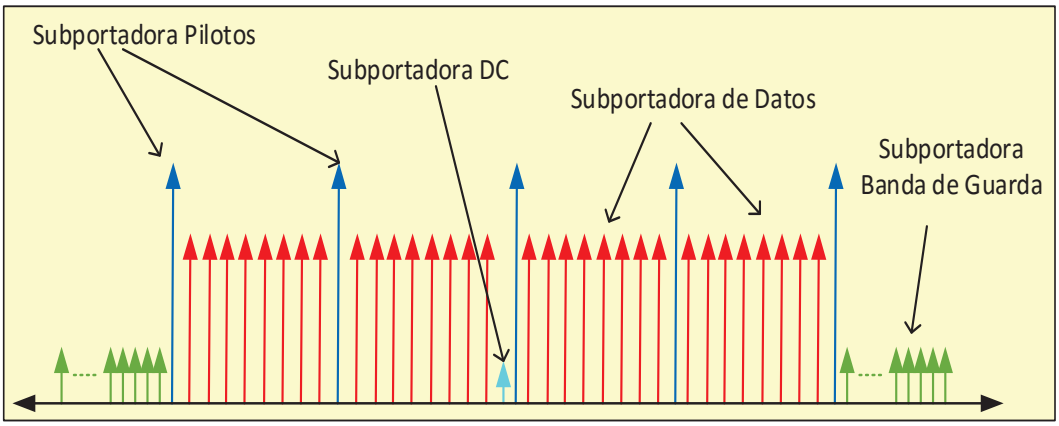


Figura 2.28 Tipos de subportadoras OFDM

2.4.2.10.1 Parámetros OFDMModulator

- **FFTLength:** Representa el tamaño de la transformada de Fourier que se establece según el ancho de banda de canal.
- **NumGuardBandCarriers:** Indica el número de subportadoras de banda de guarda que depende del ancho de banda de canal.
- **PilotInputPort:** Esta es una propiedad lógica que controla si se puede especificar los índices de las subportadoras piloto. El valor por defecto es *false*.
- **PilotCarrierIndices:** Si el parámetro PilotInputPort es *true* entonces permite ingresar los índices de las subportadoras piloto según el ancho de canal.
- **CyclicPrefixLength:** Especifica la longitud del prefijo cíclico[35] OFDM. El valor por defecto es 16.

[35] Prefijo Cíclico: El prefijo cíclico es una copia de la última parte del símbolo OFDM haciendo que la señal transmitida sea periódica y así evitar la interferencia intersimbólica y entre portadoras.

- **NumSymbols:** Especifica el número de símbolos OFDM, el valor de este parámetro por defecto es 1 y debe ser un número entero positivo.
- **NumTransmitAntennas:** Determina el número de antenas utilizadas para trasmitir una señal OFDM. El valor debe ser un número entero positivo menor o igual a 64. El valor por defecto es 1.

En la simulación se define una función OFDM para cada ancho de banda de canal, estas son: OFDM_20, OFDM_40, OFDM_80, OFDM_160. Los parámetros a ingresar en cada una de ellas se muestran en la Tabla 2.10.

Parámetro	OFDM_20	OFDM_40	OFDM_80	OFDM_160
FFTLength	64	128	256	512
NumGuardBandCarriers	[4;4]	[7;7]	[7;7]	[14;14]
PilotInputPort	True	True	True	True
PilotCarrierIndices	[11;25;39;53]	[11;39;53; 75;89;117]	[25;63;89; 117;139; 167;203;231]	[25;53;89;117; 139;167;203; 231;281;309; 245;373;395; 423;459;487]
CyclicPrefixLength	0	0	0	0
NumSymbols	1	1	1	1
NumTransmitAntenas	Depende de la cantidad de antenas de cada usuario			

Tabla 2.10 Parámetros para la función OFDMModulator

2.4.2.11 Función Generar_Datos

Esta función, cuyo diagrama de flujo se presenta en la Figura 2.29, crea datos aleatorios que será la información que se transmite en el sistema para cada usuario y se genera por medio de la siguiente instrucción.

Datos_RX_1 = randi[0 Mod-1],Ntramas*NDatos,1,RX_1)

Donde:

randi : Función de Matlab para generar número aleatorios.

[0 Mod-1]: Se generan datos aleatorios del 0 hasta el valor de modulación ingresado – 1, por ejemplo: si el MCS ingresado es 7 se tiene

una Modulación de 64 QAM y, por lo tanto se obtendrá números aleatorios entre 0 y 63 como se explica en [26].

Ntramas: Se ha seleccionado un valor de 500 tramas.

NDatos: Este valor se obtiene de la función OFDMModulator. Representa el número de subportadoras de datos que tiene cada ancho de banda de canal (ver Tabla 1.4).

RX_1: Es el número de antenas para el usuario 1.

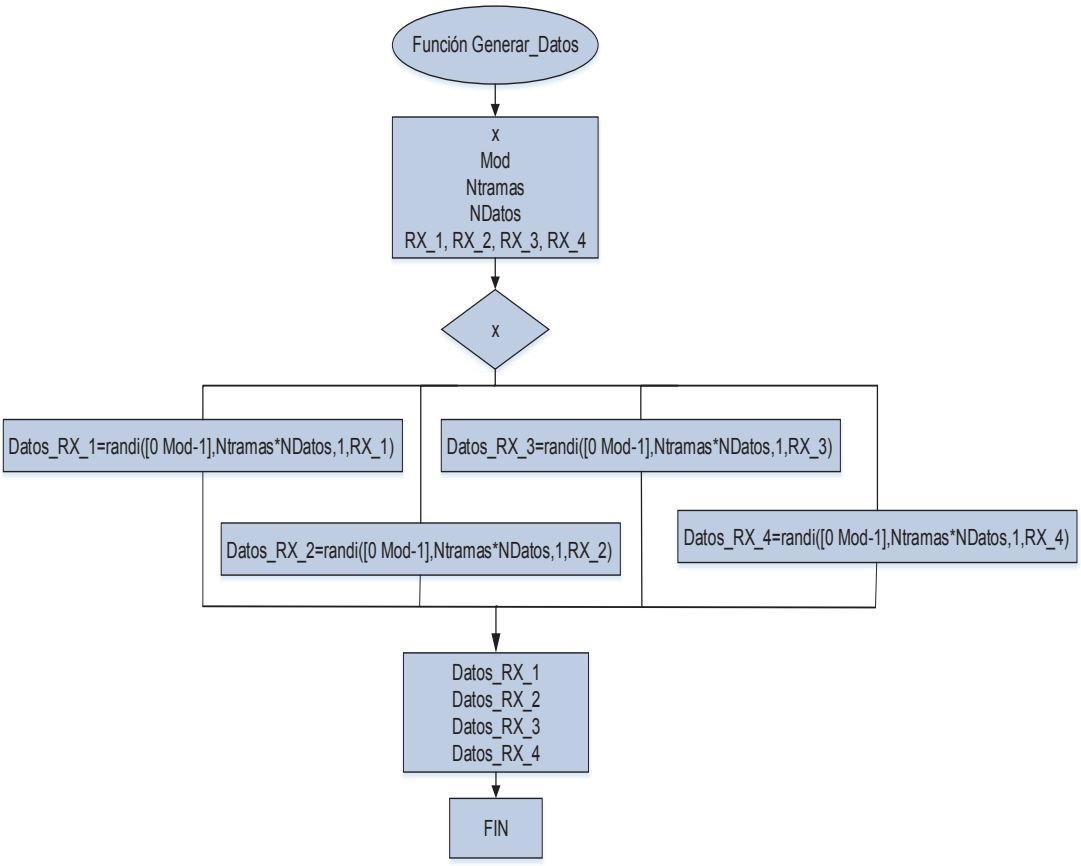


Figura 2.29 Diagrama de Flujo – Función Generar_Datos

2.4.2.12 **Función Modulación**

La función realiza la modulación de los datos generados para cada usuario, para ello se utiliza la sentencia *qammod* de Matlab® :

```
Datos_Mod_1 = qammod(Datos_RX_1(:),Mod)
```

Donde:

qammod: Función de Matlab® para modular en cuadratura.
Datos_RX_1(:): Datos aleatorios generados para el usuario 1.
Mod: Orden de la modulación.

El diagrama de flujo de esta función se presenta en la Figura 2.30:

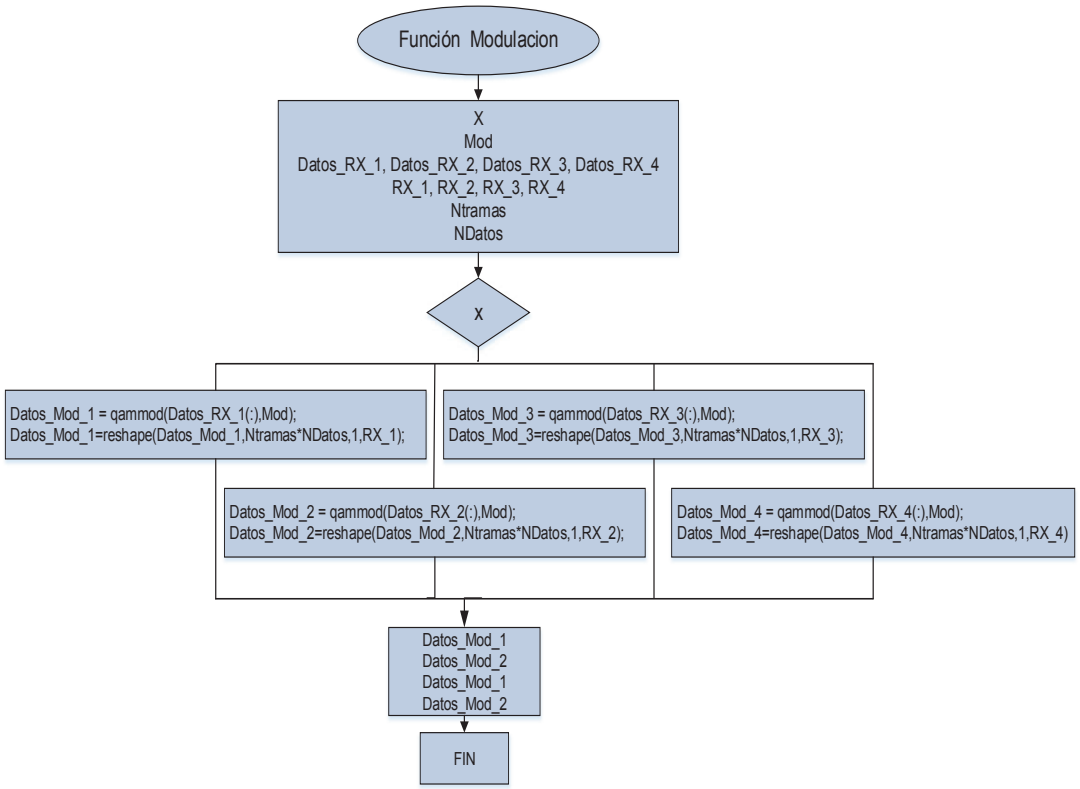


Figura 2.30 Diagrama de Flujo – Función Modulación

2.4.2.13 **Función Matriz_Correlacion_Modelo**

Estas funciones se crean para cada modelo y contienen los parámetros de cada uno de ellos. El modelo A tiene un solo *tap* por lo cual se determina una sola función que no depende del valor de ancho de banda. Para los modelos B y C que tienen

2 clústers cada uno, se asignan funciones para cada ancho de banda. En la siguiente tabla se presentan las funciones para cada modelo.

Modelo A	Modelo B (BW=20,40,80,160 MHz)	Modelo C (BW=20,40,80,160 MHz)
Matriz_Correlacion _Modelo_A	Matriz_Correlacion_Modelo_B_20	Matriz_Correlacion_Modelo_C_20
	Matriz_Correlacion_Modelo_B_40	Matriz_Correlacion_Modelo_C_40
	Matriz_Correlacion_Modelo_B_80	Matriz_Correlacion_Modelo_C_80
	Matriz_Correlacion_Modelo_B_160	Matriz_Correlacion_Modelo_C_160

Tabla 2.11 Funciones Matriz_Correlacion_Modelo

El proceso que realizan estas funciones es el de asignar la función que calcula la matriz de correlación de transmisión y recepción a cada usuario. En la Figura 2.31 se muestra el procedimiento que sigue el modelo B, el cual es similar para los demás casos de la Tabla 2.11.

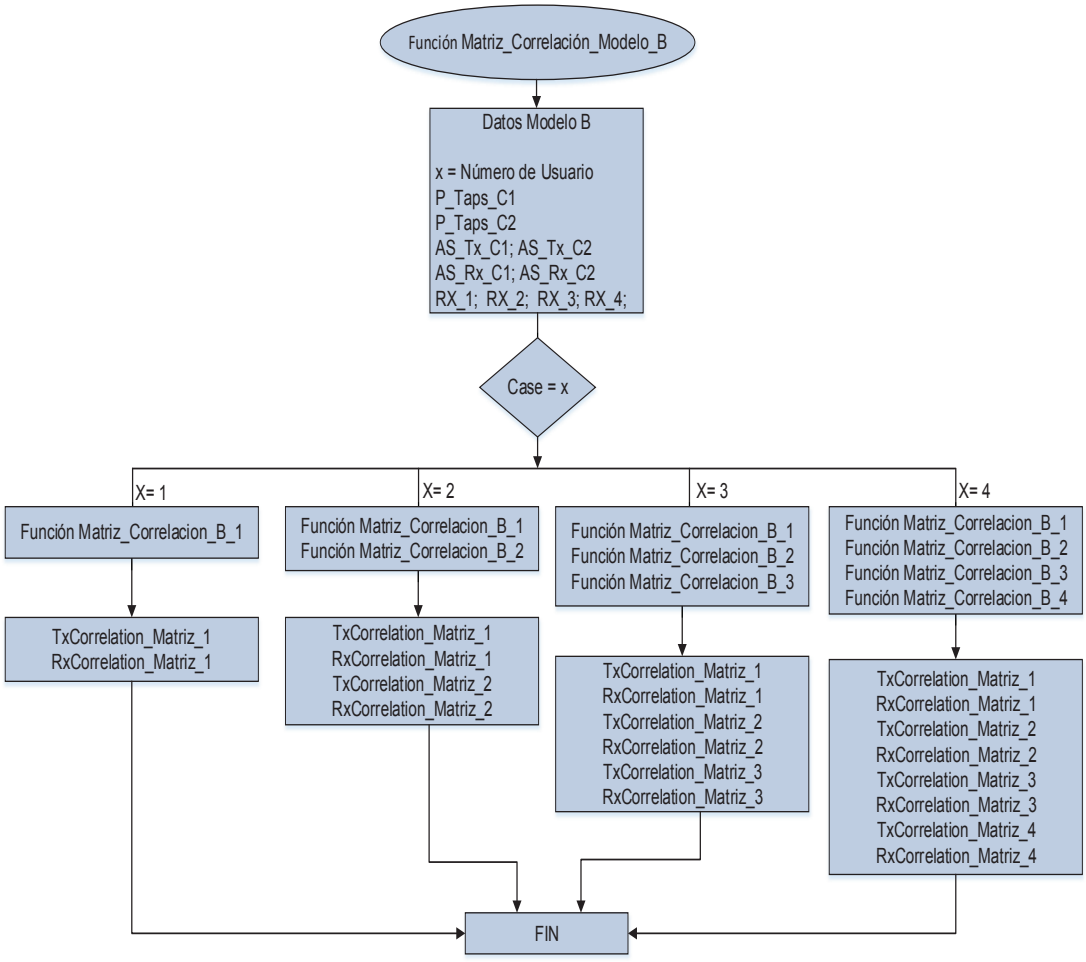


Figura 2.31 Diagrama de Flujo – Función Modulación

Una vez que se ingresa el modelo, el valor del ancho de banda de canal y el número de usuarios, se determinan las funciones de matriz de correlación en función de estos parámetros.

2.4.2.14 **Función para calcular la Matriz de Correlación**

Esta función determina las matrices de correlación en transmisión y recepción para cada usuarios. La función es la siguiente:

```
function [TxCorrelationMatriz_1, RxCorrelationMatriz_1] ...
= Matriz_Correlacion_B_1(TX_1,RX_1,P_Taps_C1,P_Taps_C2,TxSpacing,RxSpacing,…
                         AS_Tx_C1, AS_Tx_C2, AoD_C1, AoD_C2, ...
                         AS_Rx_C1, AS_Rx_C2, AoA_C1, AoA_C2)
```

Nota: Función tomada del Modelo B para el usuario 1. El nombre de las variables cambiaran de acuerdo al número de usuario y cantidad de clústers de cada modelo.

Donde:

TX_1:	Antena transmisora del usuario 1.
RX_1:	Antena receptora del usuario 1.
P_Taps_C1, P_Taps_C2:	Potencia de los taps de los clústers 1 y 2 respectivamente.
TxSpacing, RxSpacing:	Separación entre antenas transmisoras y receptoras.
AS_Tx_C1, AS_Tx_C2:	Angular spread para el transmisor de los clústers 1 y 2 respectivamente.
AoD_C1, AoD_C2:	Ángulos AoD de los clústers 1 y 2 respectivamente.
AS_Rx_C1, AS_Rx_C2:	Angular spread para el receptor de los clústers 1 y 2 respectivamente.
AoA_C1, AoA_C2	Ángulos AoD de los clústers 1 y 2 respectivamente.

Cada usuario de cada modelo tiene su propia función para generar su matriz de correlación como se detalla en la Tabla 2.12.

Modelo A	Modelo B	Modelo C
Matriz_Correlacion_A_1	Matriz_Correlacion_B_1	Matriz_Correlacion_C_1
Matriz_Correlacion_A_2	Matriz_Correlacion_B_2	Matriz_Correlacion_C_2
Matriz_Correlacion_A_2	Matriz_Correlacion_B_3	Matriz_Correlacion_C_3
Matriz_Correlacion_A_2	Matriz_Correlacion_B_4	Matriz_Correlacion_C_4

Tabla 2.12 Funciones Matriz_Correlacion_Modelo

2.4.2.15 **Función comm.MIMOChannel [27]**

La función *comm.MimoChannel* es propia de Matlab® 2015b [27], permite generar y simular la transmisión de una señal a través de un canal MIMO en desvanecimiento de *Rayleigh* o *Rician*, donde para realizar la correlación espacial la función utiliza el modelo de Kronecker mencionado en 2.3.2.2.1.

Los parámetros a ingresar en la función *comm.MimoChannel* se detallan a continuación.

2.4.2.15.1 Parámetros comm.MimoChannel

- **SampleRate:** Específica la frecuencia de muestreo de la señal de entrada, el valor por defecto es de 1 Hz.
- **PathDelays:** Indica los tiempos de retardo de los *taps* en segundos, en el caso de que se tenga un valor escalar se considerara un canal *flat fading*, mientras que, cuando sea un vector se tendrá un canal selectivo en frecuencia.
- **AveragePathGains:** Representa el valor total promedio de las ganancias de los taps en dB, es de igual tamaño que el vector *PathDelays.*
- **FadingDistribution:** Determina el tipo de desvanecimiento, las opciones son *Rayleigh* o *Rician*. Por defecto se establece un desvanecimiento tipo *Rayleigh;* para *Rician:* en este proyecto se ha establecido la letra “R”.
- **KFactor:** Especifica el factor K de un canal con desvanecimiento, es un valor tipo escalar, real y positivo o un vector positivo de la misma longitud que el

parámetro *PathDelays*. Esta propiedad se aplica cuando se establece *FadingDistribution* como *Rician*. El valor predeterminado de esta propiedad es de 3. En el caso de que el KFactor sea igual a 0 se tiene un desvanecimiento *Rayleight.*

- **MaximunDopplerShift:** Determina el máximo Efecto *Doppler* en hercios, el valor por defecto en Matlab® es de 0.001 Hz, este parámetro es aplicable a todos los *taps* de la señal.
- **DopplerSpectrum:** Especifica la forma del espectro D*oppler* para el canal, y se habilita únicamente si se determina algún valor en *MaximunDopplerShift*, el valor por defecto en Matlab® es *Jakes*.
- **SpatialCorrelation:** Determina las matrices de correlación de transmisión y recepción, de las cuales se extrae el valor de antenas transmisoras y receptoras. Este parámetro por defecto es *true.*
- **TransmitCorrelationMatrix:** Ingresa la matriz de correlación en transmisión. Esta función se habilita cuando *SpatialCorrelation* es establecida como *true*.
- **ReceiveCorrelationMatrix:** Ingresa la matriz de correlación en recepción. Esta función se habilita cuando *SpatialCorrelation* es establecida como *true.*
- **PathForDopplerDisplay:** Se aplica cuando en *Visualization* se desea obtener el gráfico de *Doppler Spectrum*.

Las funciones de la Tabla 2.13 generan un canal MIMO para cada usuario en los tres modelos usando la función *comm.MIMOChannel*, además de formar la matriz MU-MIMO.

Modelo	Función
Modelo A	Modelo_A_MimoChannel
Modelo B	Modelo_B_MimoChannel
Modelo C	Modelo_C_MimoChannel

Tabla 2.13 Funciones MIMOChannel

La Figura 2.32 muestra el diagrama de flujo de la función MIMOChannel para el modelo B, el proceso es similar para todos los modelos tomando en cuenta las particularidades de cada uno de ellos.

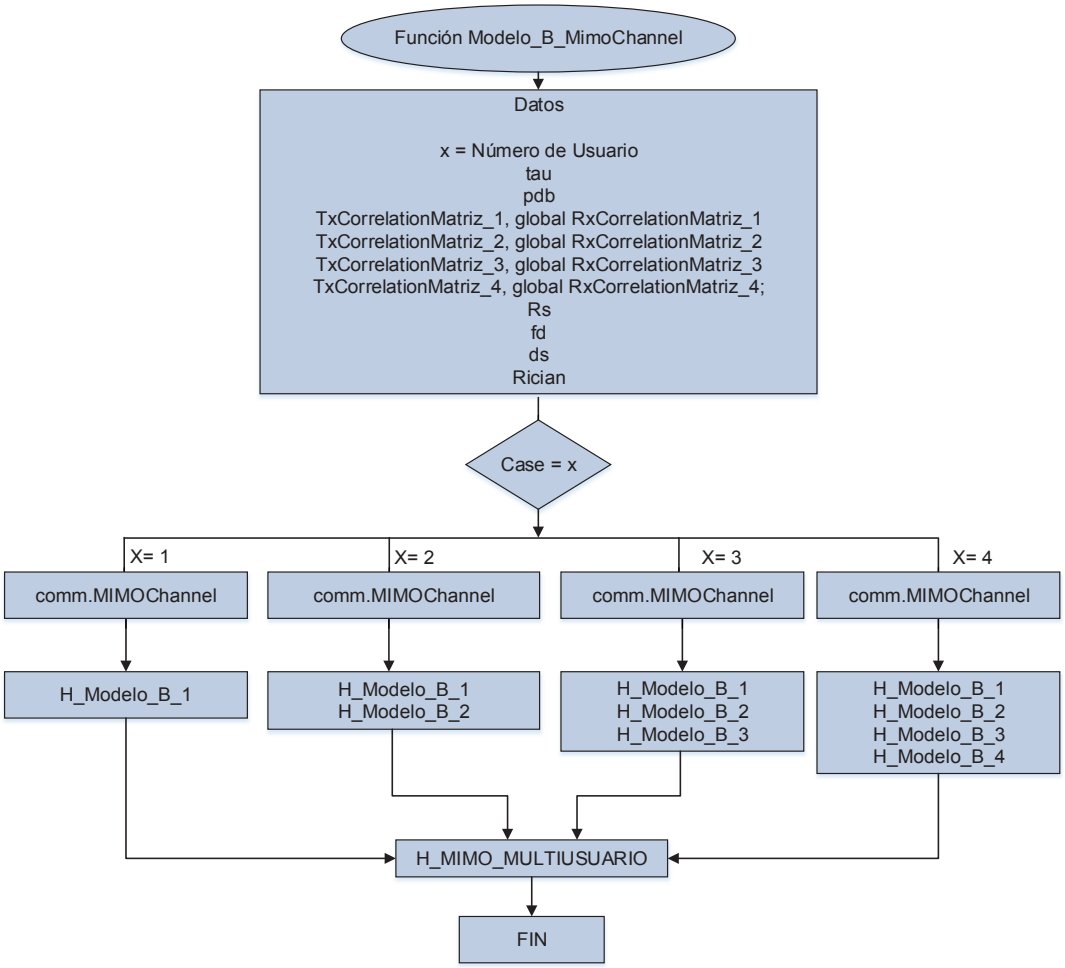


Figura 2.32 Función Modelo_A_MimoChannel

El efecto MU-MIMO se obtiene asumiendo un canal SU-MIMO por usuario como se expone en 2.3.2.4. Así por ejemplo, para un escenario con 4 usuarios se tiene que la matriz MU-MIMO de este modelo se genera de la siguiente manera:

H_MIMO_MULTIUSUARIO={H_Modelo_B_1;H_Modelo_B_2;H_Modelo_B_3;H_Modelo_B_4}

```
H_MIMO_MULTIUSUARIO =

    [1x1 comm.MIMOChannel]
    [1x1 comm.MIMOChannel]
    [1x1 comm.MIMOChannel]
    [1x1 comm.MIMOChannel]
```

Figura 2.33 Función H_MIMO_MULTIUSUARIO – Modelo B

2.4.2.16 **Función OFDM_MIMO**

Esta función depende de los resultados generados por las funciones: Modulación, comm.OFDMModulator y comm.MIMOChannel. El procedimiento de esta función es modular en OFDM los datos de entrada y su resultado ingresarlo por el canal MIMO de manera independiente para cada usuario (ver Figura 2.34).

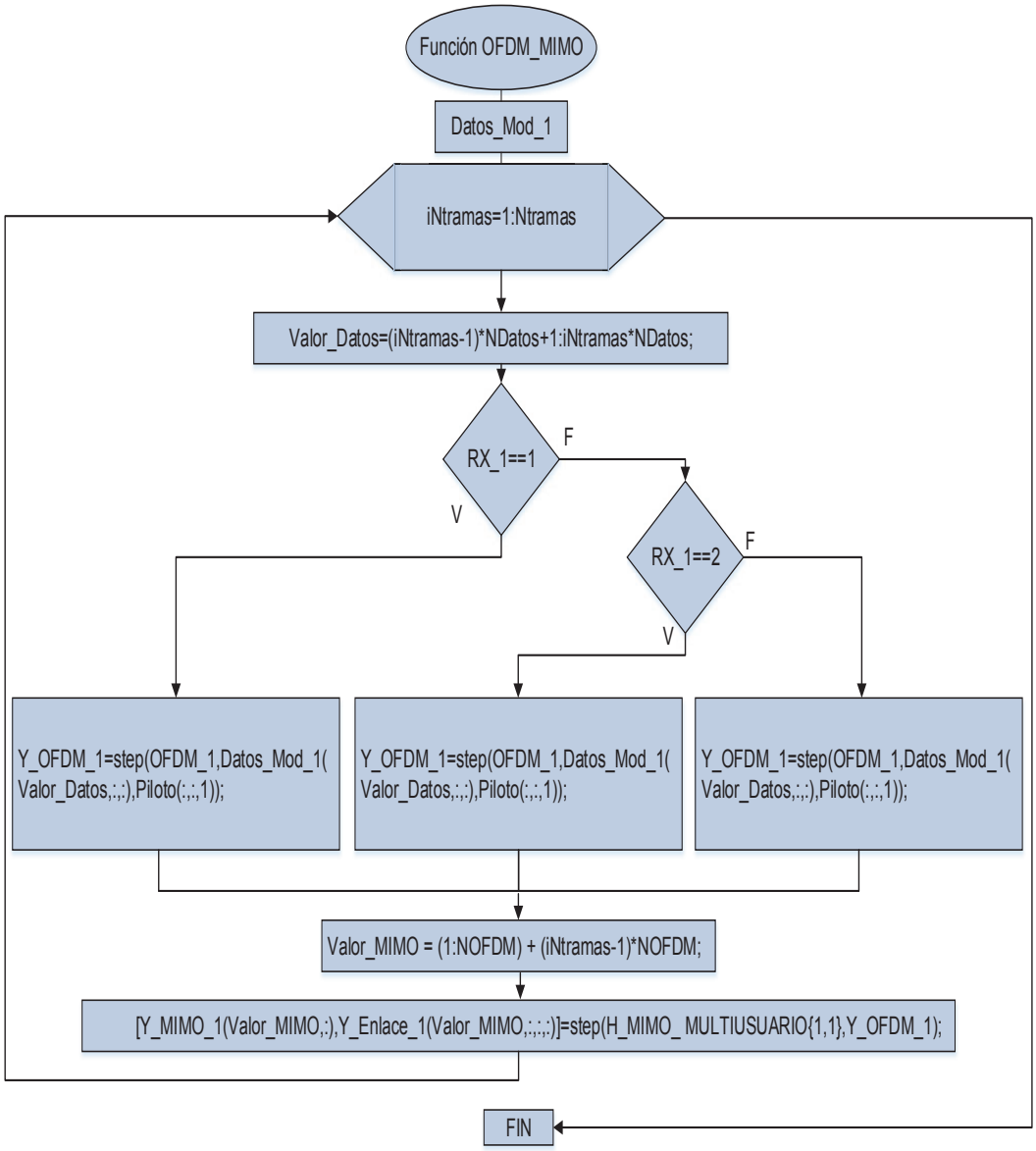


Figura 2.34 Diagrama de flujo - Función OFDM_MIMO

2.4.2.17 **Función AWGN**

Por medio de esta función se adiciona ruido AWGN al resultado obtenido del proceso OFDM_MIMO, se usa la función *awgn* de Matlab® que utiliza el valor de SNR en dB. Los valores de SNR se toman de la Figura 2.19. El diagrama de flujo de esta función se presenta a continuación.

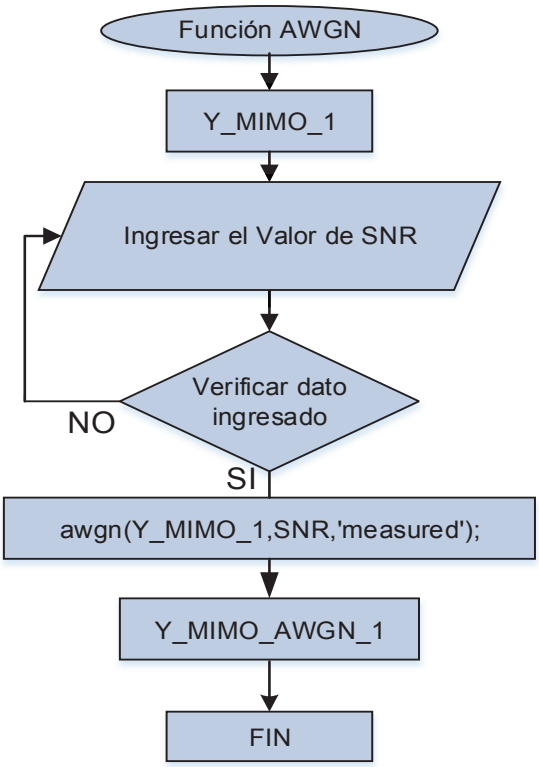


Figura 2.35 Diagrama de flujo - Función AWGN

2.4.2.18 **Función Resultados**

Esta función calcula los valores de Data Rate, Capacidad del Canal y la Eficiencia del Sistema basándose en las siguientes fórmulas.

- Data Rate

$$Ncbps\_1 = Nbpscs * NDatos \quad (2.12)$$

$$Ndbps\_1 = Ncbps\_1 * Coding \quad (2.13)$$

$$Data\_rate\_1 = \frac{Ndbps\_1}{ts} * RX\_1 * ig \quad (2.14)$$

Donde:

Nbpscs = Bits codificados por subportadora
Ncbps_1 = Bits codificados por símbolo OFDM
NDatos = Subportadoras de datos
Ndbps_1 = Bits de datos por símbolo OFDM
Coding = Valor de code rate
RX_1 = Número de antenas
ts = Duración del símbolo 4µseg
ig = Intervalo de guarda (1 = 800 nseg, 1.1= 400 nseg)
Data_rate_1 = Velocidad de transmisión

- Capacidad del Canal

$$C = BW * log_2 (1 + SNR\_Valor) \qquad (2.15)$$

Donde:

C = Capacidad del Canal
BW = Ancho de banda de canal
SNR_Valor = Relación señal al ruido

- Eficiencia del Sistema

$$E = \frac{Data\_rate}{BW} \qquad (2.16)$$

Data_rate = Velocidad de transmisión del sistema
E = Eficiencia del Sistema

CAPÍTULO III

DESARROLLO DE PRUEBAS

3.1 INTRODUCCIÓN

El rendimiento o performance de una red de datos depende de varios factores, entre los cuales se encuentran: velocidad efectiva o *throughput*, la tecnología utilizada por los dispositivos, obstáculos físicos, distancia entre los dispositivos, interferencias Wireless, entre otros.

Por un lado, con el uso de los tipos avanzados de modulación (256-QAM), se tiene la ventaja de transmitir más bits por unidad de señal. Aunque en dichas transmisiones se pueden producir un número mayor de errores por los factores citados en el párrafo anterior.

En la etapa de planificación de la red se hace necesario realizar todas las mediciones correspondientes para minimizar los efectos negativos que degradan a la red, asegurando así su buen desempeño.

En este capítulo se realiza un conjunto de evaluaciones prácticas, en un ambiente inalámbrico en modo infraestructura, utilizando el Access Point Cisco AIR-CAP3602E-A-K9, el módulo Cisco AIR-RM3000AC-A-K9 y dispositivos finales compatibles para comprobar velocidades efectivas teóricas definidas en el estándar IEEE 802.11ac.

Las pruebas se efectuarán en ambientes externos e internos, donde el punto de acceso (AP) operará en modo autónomo. En esta fase de pruebas se irá variando parámetros como: tipo de antenas según el escenario, distancias, ancho de banda, MCS e intervalos de guarda.

3.2 MODOS DE OPERACIÓN WLAN

3.2.1 MODO INFRAESTRUCTURA

Una topología en este modo es aquella en la que los dispositivos finales se conectan a un equipo (AP) central que los administra.

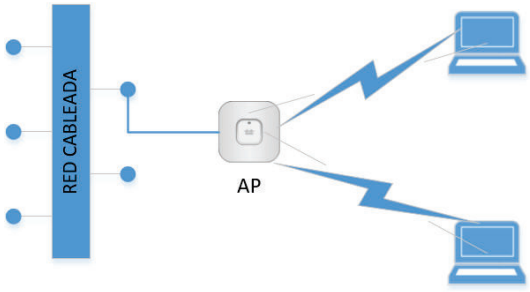


Figura 3-1 Modo Infraestructura

3.2.2 MODO AD-HOC

También conocida como modo “peer to peer”. Este tipo de redes no necesita un punto de acceso centralizado que los administre, en su lugar los dispositivos finales se conectan entre sí.

Figura 3-2 Modo AD-HOC

3.2.3 MODOS DE OPERACIÓN DEL AP

Las siguientes formas en las que funcionan u operan los puntos de acceso se consideran para equipos de las características utilizadas en este trabajo.

3.2.3.1 AUTONOMO

Conocido también como *Stand Alone,* en este modo el AP actúa independientemente sin necesidad de equipos adicionales. El cambio de un modo a otro consiste en instalar en el AP el sistema operativo (IOS) compatible con el modo deseado.

3.2.3.2 LIGHTWEIGHT

En el modo LAP (*Lightweight Access Point*) el AP es administrado por una WLC (*Wireless Lan Controller, Controladora de LAN Inalámbrica*) que esté presente en la red, de este último equipo el AP adopta las configuraciones (SSIDs, ganancia de antenas, ancho de banda, frecuencia de operación, seguridades, entre otras) necesarias para su funcionamiento.

3.3 EQUIPOS UTILIZADOS

Como se menciona anteriormente, en esta fase de pruebas se utiliza el equipo AIR-CAP3602I-A-K9 que hace las funciones de AP, el módulo AIR-RM3000AC-A-K9 que proporciona las características 802.11ac (mostrados en la Figura 3.3) y dispositivos finales de usuario compatibles con dicho estándar.

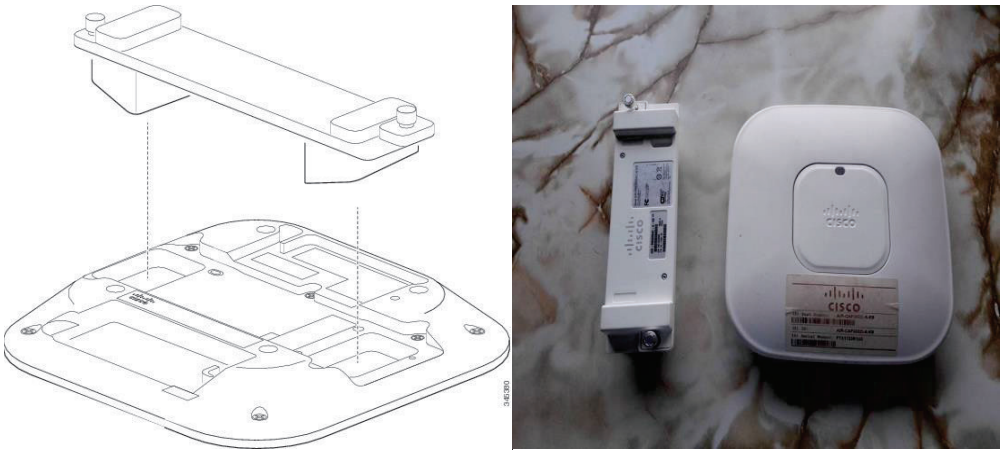


Figura 3.3 CISCO AIR-CAP3602I-A-K9 y AIR-RM3000AC-A-K9

3.3.1 AIR-CAP3602I-A-K9 y AIR-RM3000AC-A-K9

Los puntos de acceso Cisco de la serie 3600 tienen las siguientes características:

- Radios de banda dual (2.4 y 5GHz).
- Opción de antenas internas en el modelo 3602I (AIR-CAP3602I-x-K9, AIR-AP3602I-UXK9), o externas en el modelo 3602E (AIR-CAP3602E-x-K9, AIR-AP3602E-UXK9).
- Interoperabilidad con clientes 802.11.

En un principio se planteó utilizar el AP Cisco modelo AIR-CAP3602E-A-K9, debido a que la empresa auspiciante de este proyecto estaba interesada en utilizar estos equipos para una implementación WLAN. Pero por decisiones internas de dicha empresa se adquiere los APs Cisco modelo AIR-CAP3602I-A-K9. Dado que las especificaciones entre los dos modelos de la serie 3600 son similares en sus características (Figura 3.4) y de acuerdo a lo antes expuesto, para realizar las pruebas se utiliza el AP Cisco AIR-CAP3602I-A-K9.

Item	Specification
Part Numbers	**The Cisco Aironet 3600i Access Point: Indoor environments, with internal antennas** • AIR-CAP3602I-x-K9 - Dual-band controller-based 802.11a/g/n • AIR-CAP3602I-xK910 - Eco-pack (dual-band 802.11a/g/n) 10 quantity access points **The Cisco Aironet 3600e Access Point: Indoor, challenging environments, with external antennas** • AIR-CAP3602E-x-K9 - Dual-band controller-based 802.11a/g/n • AIR-CAP3602E-xK910 - Eco-pack (dual-band 802.11a/g/n) 10 quantity access points

Figura 3.4 Especificaciones serie 3600 [28]

El AP Cisco AIR-CAP3602I-A-K9 por sí solo brinda los radios de 802.11 a/g/n mas no las características de 802.11ac, por lo tanto, es necesario incluir el módulo AIR-RM3000AC-A-K9 para que en conjunto funcionen bajo el estándar en estudio. En la Figura 3.5 se indica los tipos de módulos que se pueden añadir al AP Cisco AIR-CAP3602I-A-K9, y se enmarcan las especificaciones del módulo AIR-RM3000AC-A-K9.

Item	Specification
Module Options	Cisco Aironet Wireless Security Module • Provides full-spectrum scanning for, wIPS for comprehensive detection and mitigation of over the wire network attacks, Cisco CleanAir technology detecting devices causing network interference, rogue device detection, context (location) awareness, and radio resource management (RRM) solutions • Provides full scanning of all 2.4- and 5-GHz channels while the Access Point is serving data clients on the integrated radios Cisco Aironet IEEE 802.11ac Wave 1 Module • Supports the IEEE 802.11ac specification and the features defined by the Wi-Fi Alliance for the first wave of Wi-Fi CERTIFIED 11ac • 3x3:3SS (spatial streams), 80-MHz wide channels, 256 quadrature amplitude modulation (QAM), and data rates up to 1.3 Gbps • Wi-Fi Alliance certified - http://www.wi-fi.org/certified-products-advanced-search Cisco Universal Small Cell 5310 (limited availability) • 3GPP band 1 (2100 MHz), 16 users, voice (R99), packet data (HSPA/HSDPA+) • 3GPP band 2/5 (band 2 – 1930 and band 5 - 869), 16 users, voice (R99), packet data (HSPA/HSDPA+)

Figura 3.5 Opciones de módulos para la serie 3600 [28]

Para las pruebas de velocidades efectivas se debe tomar en cuenta las particularidades propias de los equipos a utilizar (AIR-CAP3602I-A-K9, AIR-RM3000AC-A-K9), los dos equipos ofrecen el estándar 802.11ac con las características de *Wave* 1 como se menciona en el Capítulo I (Tabla 1.3) .

La Figura 3.6 muestra las capacidades 802.11 soportadas por el módulo AIR-RM3000AC-A-K9:

Item	Specification
802.11 Capabilities	• 3 x 3 multiple-input multiple-output (MIMO) with three spatial streams • Maximal ratio combining (MRC) • 802.11ac, 802.11n, and 802.11a/g beamforming • 20-, 40- and 80-MHz channels • PHY data rates up to 1.3 Gbps (80-MHz with three spatial streams) • Packet aggregation: A-MPDU (Tx/Rx), A-MSDU (Tx/Rx) • 802.11 dynamic frequency selection (DFS) • Cyclic shift diversity (CSD) support

Figura 3.6 Especificaciones 802.11 soportadas por el módulo AIR-RM3000AC-A-K9 [29]

En la Figura 3.7 se presentan los "*data rates*" soportados por el equipo Cisco AIR-RM3000AC-A-K9, en donde se considera las velocidades de datos para 802.11ac tomando en cuenta variables como: MCS, ancho de banda, número de *spatial streams* e intervalos de guarda. Por ejemplo: si se tiene un ancho de banda de canal de 40 MHz, con 2 spatial streams, un MCS de 9 y un intervalo de guarda de 400 nseg, teóricamente se logra un *data rate* de 400 Mbps.

Item	Specification							
40-MHz Rate (Mbps)	802.11ac data rates (5 GHz):							
	MCS Index[3]	Spatial Streams	GI[4] = 800ns				GI = 400ns	
			20-MHz Rate (Mbps)	40-MHz Rate (Mbps)	80-MHz Rate (Mbps)	20-MHz Rate (Mbps)	40-MHz Rate (Mbps)	80-MHz Rate (Mbps)
	0	1	6.5	13.5	29.3	7.2	15	32.5
	1	1	13	27	58.5	14.4	30	65
	2	1	19.5	40.5	87.8	21.7	45	97.5
	3	1	26	54	117	28.9	60	130
	4	1	39	81	175.5	43.3	90	195
	5	1	52	108	234	57.8	120	260
	6	1	58.5	121.5	263.3	65	135	292.5
	7	1	65	135	292.5	72.2	150	325
	8	1	78	162	351	86.7	180	390
	9	1	NA	180	390	NA	200	433.3
	0	2	13	27	58.5	14.4	30	65
	1	2	26	54	117	28.9	60	130
	2	2	39	81	175.5	43.3	90	195
	3	2	52	108	234	57.8	120	260
	4	2	78	162	351	86.7	180	390
	5	2	104	216	468	115.6	240	520
	6	2	117	243	526.5	130	270	585
	7	2	130	270	585	144.4	300	650
	8	2	156	324	702	173.3	360	780
	9	2	NA	360	780	NA	400	866.7
	0	3	19.5	40.5	87.8	21.7	45	97.5
	1	3	39	81	175.5	43.3	90	195
	2	3	58.5	121.5	263.3	65	135	292.5
	3	3	78	162	351	86.7	180	390
	4	3	117	243	526.5	130	270	585
	5	3	156	324	702	173.3	360	780
	6	3	175.5	364.5	NA	195	405	NA
	7	3	195	405	877.5	216.7	450	975
	8	3	234	486	1053	260	540	1170
	9	3	260	540	1170	288.9	600	1300

Figura 3.7 Velocidades de transmisión soportadas por el módulo AIR-RM3000AC-A-K9 [29]

Con el propósito de llevar a cabo las pruebas del estándar se hace uso de 2 adaptadores USB 802.11ac: WIFI AC1200 USB ADAPTER y AWUS036AC AC1200 WIRELESS ADAPTER

3.3.2 WIFI AC1200 USB ADAPTER

Sus especificaciones principales son:

- Compatibilidad con los estándares: 802.11a/b/g/n/ac.
- Interface USB 2.0/3.0
- Esquemas de modulación soportados: CCK, DQPSK, DBPSK, QPSK, 16 QAM, 64 QAM y 256 QAM.
- Esquemas de encriptación: WEP, WPA/WPA2 (TKIP/AES).
- Realtek 8812AU Chipset Inside para confiabilidad. [29]
- AC1200 Dual-Band para alto rendimiento, hasta 866.7 Mbps.
- Compatible con: Microsoft Windows 10 / 8.1 / 8 / 7 / Vista / XP / Mac OS 10.4-10.10 / Linux.

Figura 3.8 Adaptador USB WiFi AC1200

3.3.3 AWUS036AC AC1200 WIRELESS ADAPTER

El adaptador inalámbrico ALFA 802.11ac AWUS036AC trae consigo la próxima generación de tecnología inalámbrica 802.11ac a sus computadoras laptop/desktop. AWUS036AC permite mayor velocidad, amplia cobertura de la

señal, comunicaciones más estables y compatibilidad en las comunicaciones inalámbrica. [31]

Figura 3.9 AWUS036AC AC 1200 WIRELESS ADAPTER

Las especificaciones del equipo antes citado se presentan en la tabla 3.1:

Especificaciones de Hardware	
Interface	USB 3.0
Tipo de antenas	Conector hembra desmontable RP-SMA
Antena	Antena dipolo de banda dual
Especificaciones Inalámbricas	
Estándar	IEEE 802.11a, IEEE802.11b, IEEE802.11g, IEEE802.11n, IEEE 802.11ac
Frecuencia	2.4GHz / 5GHz
Velocidad de datos	802.11a : 6,9,12,18,24,36,48,54Mbps 802.11b : 1, 2, 5.5, 11Mbps 802.11g : 6,9,12,18,24,36,48,54Mbps 802.11n : up to 300Mbps 802.11ac: up to 867Mbps
Seguridad Wireless	WEP 64/128 bit,WPA-PSK,WPA2.PSK, Cisco CCX
Sistema Operativo	Windows XP, Vista, 7, 8.1 Mac 10.5, 10.6, 10.7, 10.8, 10.9 Linux

Tabla 3.1 Especificaciones AWUS036AC AC 1200 WIRELESS ADAPTER [31]

3.4 DESARROLLO DE PRUEBAS

El conjunto de pruebas y escenarios presentados en este apartado tienen como objetivo determinar el máximo *throughput* que el AP CISCO AIR-CAP3602I-A-K9 y el módulo CISCO AIR-RM3000AC-A-K9 ofrecen a los dispositivos finales. Para llevar a cabo tal objetivo se consideran:

- Dos tipos de ambientes: interno y externo.

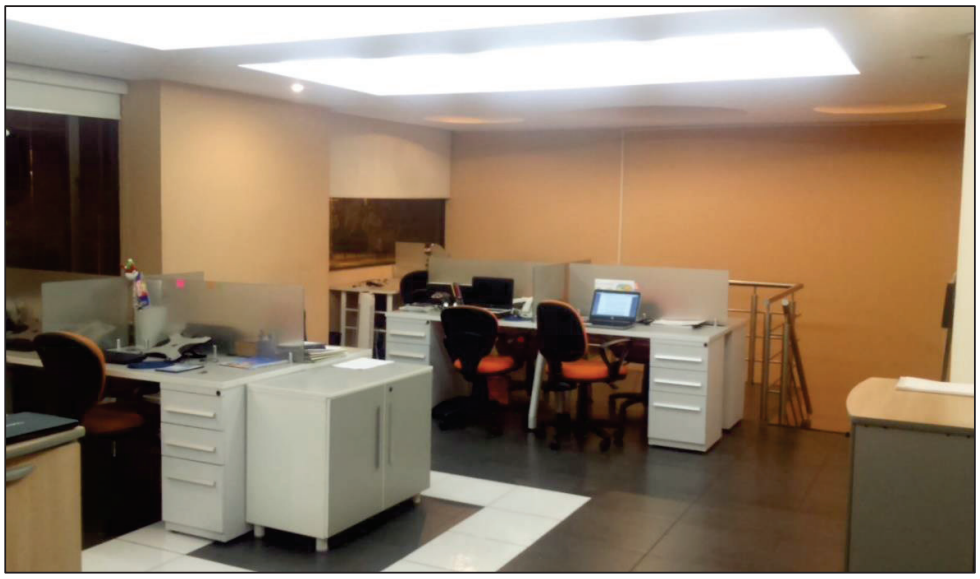


Figura 3.10 Ambiente interno

Figura 3.11 Ambiente externo

- Topología de red tipo infraestructura mostrada en la Figura 3.12

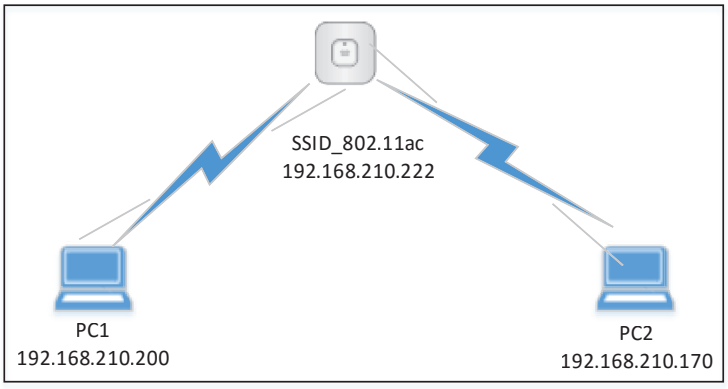


Figura 3.12 Topología implementada

- Adaptadores 802.11ac (detallados en 3.2.3 y 3.3.3) y computadoras cuyas características se presentan en la Tabla 3.2.

COMPUTADOR	TIPO	S.O	PROCESADOR	MEMORIA
PC1	Laptop	Windows 8 64bits	Intel(R) Celeron (R) @2.16GHz	4 GB
PC2	Laptop	Windows 8 64bits	Intel (R) Core (TM) i7 @2 GHz	8 GB

Tabla 3.2 Características computadores utilizados

- Ocho escenarios (ver Tabla 3.3) que se configuran en el AP y están en función de las variables: ancho de banda de canal, *spatial streams* e intervalos de guarda.

	Ancho de Canal [MHz]	Spatial Stream	Intervalo de Guarda (L = 800 nseg, S = 400 nseg)
Escenario 1	40	1	L
Escenario 2	40	1	S
Escenario 3	40	2	L
Escenario 4	40	2	S
Escenario 5	80	1	L
Escenario 6	80	1	S
Escenario 7	80	2	L
Escenario 8	80	2	S

Tabla 3.3 Configuraciones en el AP

- El uso de la herramienta Iperf3.exe[36].

[36] IPERF: Es una herramienta multiplataforma utilizada para realizar pruebas de calidad del enlace entre estaciones.

3.4.1 CONFIGURACIÓN DEL AP

Ingresando vía web al AP se pueden distinguir las interfaces de radio: Radio0-802.11N de 2.4 GHz, Radio1-802.11N de 5 GHz, y Radio2.802.11AC de 5GHz. Las tres interfaces por defecto se encuentran inhabilitadas (*shutdown*), para habilitarlas se debe configurar un SSID[37] y seleccionar la interfaz de radiación.

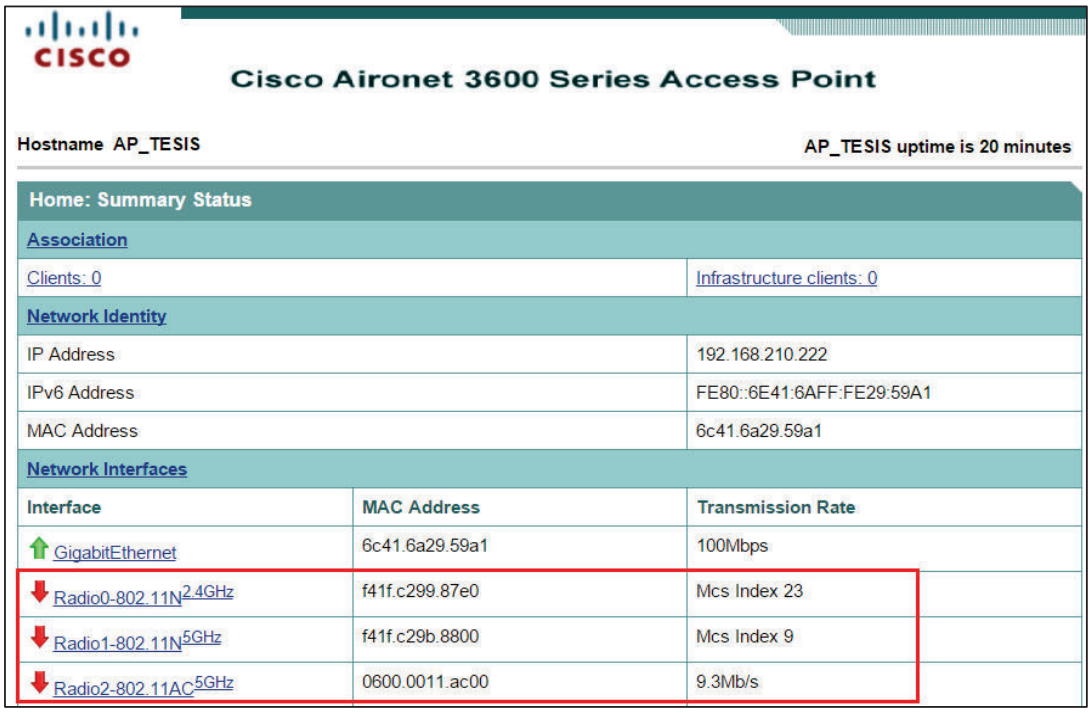


Figura 3.13 Interfaces de radio inhabilitadas

3.4.1.1 Configuración SSID

Se realiza el siguiente procedimiento:

- Se ingresa vía web al AP.
- En la pestaña de Security ➔ Ingresar a la opción SSID Manager
- Asignar un nombre en la casilla de SSID (SSID_802.11ac en este caso) y seleccionar la interfaz de radio Radio2.802.11AC de 5GHz.

[37] SSID: Nombre asignado a una red Wi-Fi (inalámbrica).

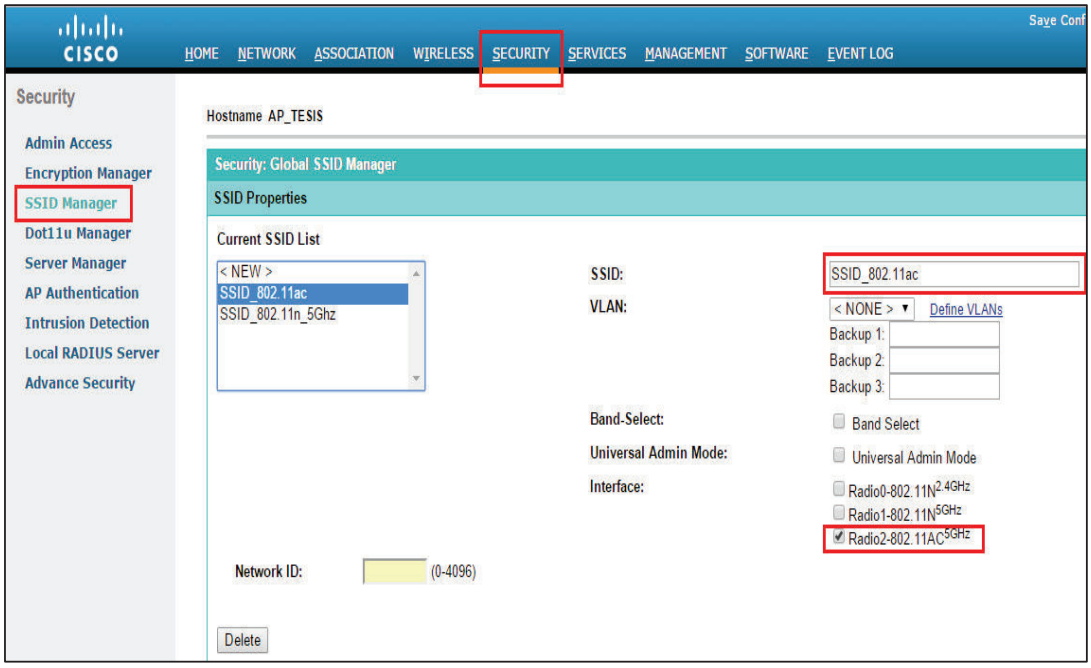


Figura 3.14 Configuración SSID

- En el Radio2-802.11AC elegir "*Single BSSID"* y seleccionar el SSID creado (SSID_802.11ac), así también en la opción de *Set Infrastructure SSID.*

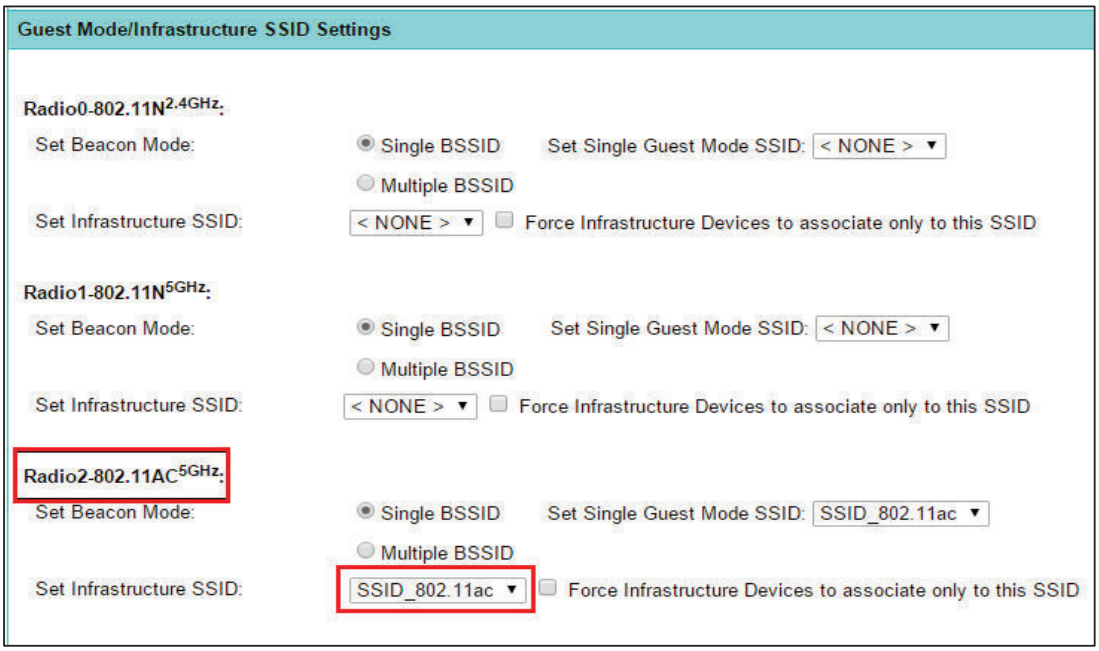


Figura 3.15 Asignación SSID al radio 802.11ac

Un AP Cisco de la serie 3600 en modo autónomo no es compatible con los tipos de seguridades TKIP y TKIP+AES para 802.11ac. [32]

3.4.1.2 Configuración de los escenarios en el AP

Se ingresa al AP vía SSH (*Secure Shell*) y se verifica la configuración del radio Dot11Radio2 ejecutando el comando s*how running-Configuración*

Los parámetros comunes de configuración para todos los escenarios son los siguientes:

- **ssid SSID_802.11ac** : Indica el nombre del SSID asignado a la interfaz de radio 802.11ac.
- **speed** : Habilita las velocidades disponibles en 802.11ac, se debe ingresar al menos una *speed basic* y una velocidad de 802.11n [33].
- **channel**: Muestra el canal de frecuencia elegido.
- **station-role root**: Indica el modo de operación del equipo.

A continuación se presenta una captura con la configuración de los parámetros antes mencionados:

```
interface Dot11Radio2
 no ip address
 !
 ssid SSID_802.11ac
 !
 antenna gain 5
 antenna a-antenna
 peakdetect
 no dfs band block
 beamform ofdm
 speed  basic-6.0 9.0 12.0 18.0 24.0 36.0 48.0 54.0 m0. m1. m2. m3. m4. m5. m6. m7. m8. m9. m10. m11. m12.
m13. m14. m15. m16. m17. m18. m19. m20. m21. m22. m23. a1ss9 a2ss9 a3ss9
 channel width 40-below
 channel 5765
 station-role root
 monitor frames endpoint ip address 0.0.0.1 port 10 truncate 0
 bridge-group 1
 bridge-group 1 subscriber-loop-control
 bridge-group 1 spanning-disabled
 bridge-group 1 block-unknown-source
 no bridge-group 1 source-learning
 no bridge-group 1 unicast-flooding
!
```

Figura 3.16 Parámetros comunes de configuración

Los parámetros que varían de acuerdo al escenario son los siguientes.

- **antenna**: Indica la cantidad de antenas habilitadas.
- **guard-interval**: Permite seleccionar los intervalos de guarda: *long*= 800 nseg, *any*= 800 nseg o 400 nseg dependiendo de los dispositivos finales.
- **channel width**: Establece el valor de ancho de banda de canal 802.11ac

A continuación se presenta una captura con la configuración de los parámetros antes mencionados:

```
interface Dot11Radio2
 no ip address
 !
 ssid SSID_802.11ac
 !
 antenna gain 5
 antenna a-antenna
 peakdetect
 no dfs band block
 beamform ofdm
 guard-interval long
 speed  basic-6.0 9.0 12.0 18.0 24.0 36.0 48.0 54.0 m0. m1. m2. m3. m4. m5. m6. m7. m8. m9. m10. m11. m12.
m13. m14. m15. m16. m17. m18. m19. m20. m21. m22. m23. a1ss9 a2ss9 a3ss9
 channel width 40-below
 channel 5765
 station-role root
 monitor frames endpoint ip address 0.0.0.1 port 10 truncate 0
 bridge-group 1
 bridge-group 1 subscriber-loop-control
 bridge-group 1 spanning-disabled
 bridge-group 1 block-unknown-source
 no bridge-group 1 source-learning
 no bridge-group 1 unicast-flooding
!
```

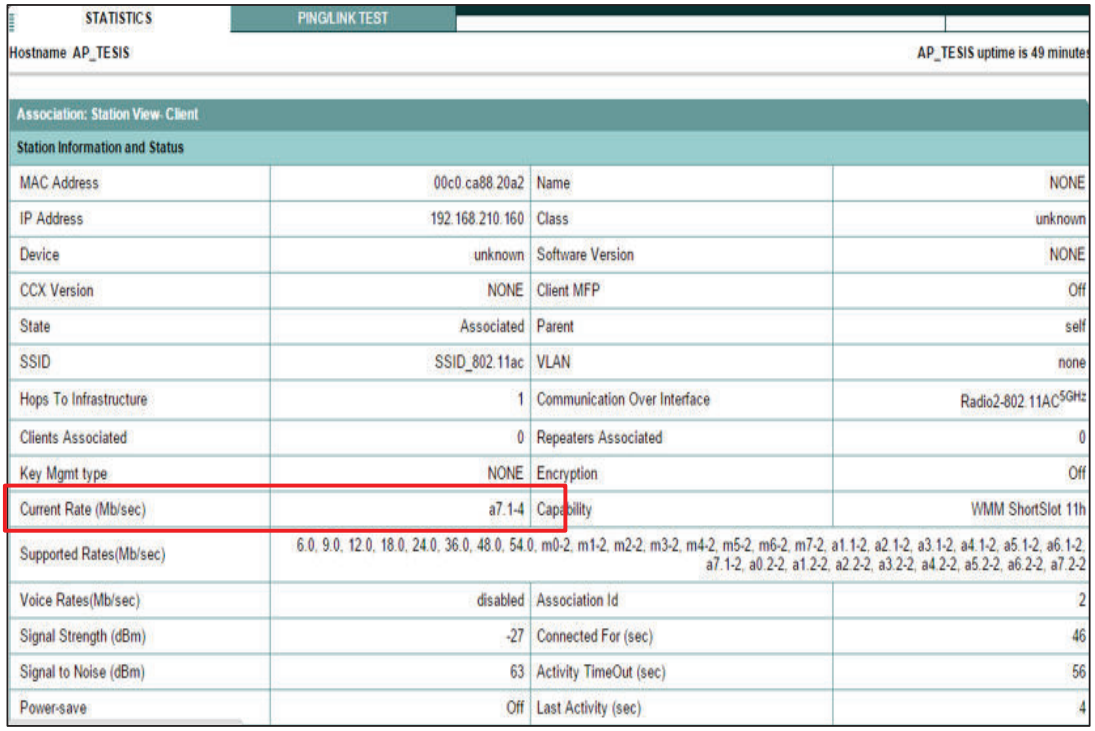
Figura 3.17 Parámetros variables

Mediante la interfaz gráfica del AP se puede determinar el índice MCS con el que un dispositivo final está trabajando. Este valor viene dado por la casilla "*Current Rates (Mb/sec)*" señalado en la Figura 3.18. Esta información será de utilidad cuando se realice el análisis y comparación de velocidades de transmisión.

El AP Cisco utiliza el formato aX.YcZ para mostrar la tasa de datos, la manera de interpretar dicha información es la siguiente:

- X = MCS, 0-9
- Y = Número de spatial streams, 1-3.

- Z = Ancho de banda de canal 2, 4, 8 (2 = 20Mhz, 4=40Mhz, 8=80Mhz)
- c = b para *Beamform*, s para STBC, - ninguno



STATISTICS | PING/LINK TEST

Hostname AP_TESIS — AP_TESIS uptime is 49 minutes

Association: Station View- Client

Station Information and Status

MAC Address	00c0.ca88.20a2	Name	NONE
IP Address	192.168.210.160	Class	unknown
Device	unknown	Software Version	NONE
CCX Version	NONE	Client MFP	Off
State	Associated	Parent	self
SSID	SSID_802.11ac	VLAN	none
Hops To Infrastructure	1	Communication Over Interface	Radio2-802.11AC^{5GHz}
Clients Associated	0	Repeaters Associated	0
Key Mgmt type	NONE	Encryption	Off
Current Rate (Mb/sec)	a7.1-4	Capability	WMM ShortSlot 11h
Supported Rates(Mb/sec)	6.0, 9.0, 12.0, 18.0, 24.0, 36.0, 48.0, 54.0, m0-2, m1-2, m2-2, m3-2, m4-2, m5-2, m6-2, m7-2, a1.1-2, a2.1-2, a3.1-2, a4.1-2, a5.1-2, a6.1-2, a7.1-2, a0.2-2, a1.2-2, a2.2-2, a3.2-2, a4.2-2, a5.2-2, a6.2-2, a7.2-2		
Voice Rates(Mb/sec)	disabled	Association Id	2
Signal Strength (dBm)	-27	Connected For (sec)	46
Signal to Noise (dBm)	63	Activity TimeOut (sec)	56
Power-save	Off	Last Activity (sec)	4

Figura 3.18 Interfaz gráfica del punto de acceso CISCO

3.4.2 PROCEDIMIENTO

Los pasos a seguir tanto para el ambiente interno y externo son los mismos, y se detallan a continuación:

- Se establece la topología de red propuesta en la Figura 3.12 tomando en cuenta un direccionamiento IP estático y seleccionando el canal menos congestionado. Para seleccionar un canal inalámbrico existen varios *software* que dan una visión del uso del espectro inalámbrico; en este caso el programa computacional que se utiliza es el *CommView for Wifi 7.1*

  La Figura 3.19 muestra el espectro de 5 GHz en donde se puede apreciar las redes Wi-Fi y los canales en los cuales operan. La mayoría de las redes ocupan el mismo rango de canales interfiriéndose entre ellas.

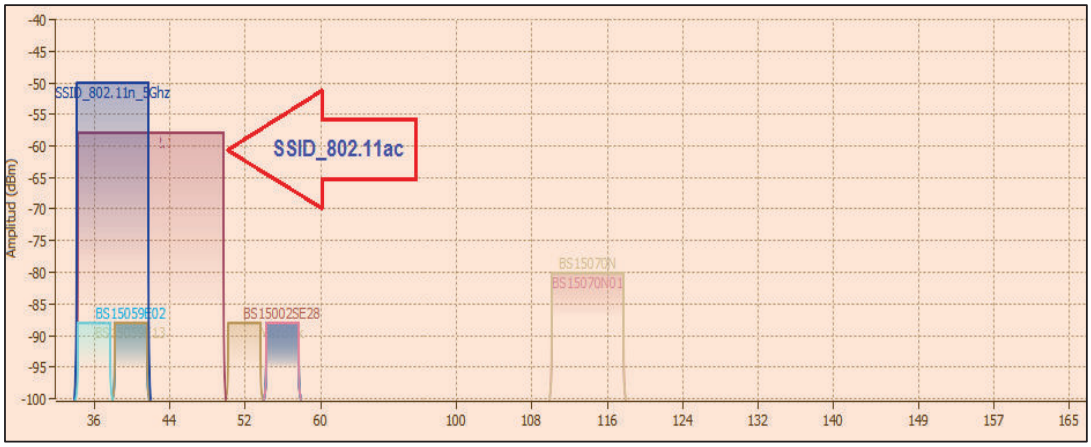


Figura 3.19 SSID_802.11ac con redes interferentes

Para evitar las interferencias provocadas por las demás redes se configura al SSID_802.11ac para que trabaje en el rango de frecuencias de 5735 a 5835 MHz. Figura 3.20.

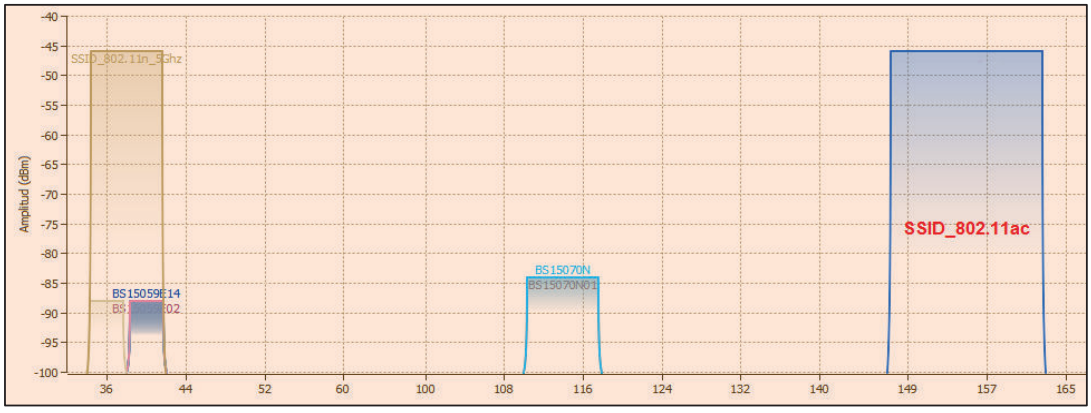


Figura 3.20 SSID_802.11ac sin redes interferentes

- Se configura la PC2 como cliente mediante el comando *iperf –c 192.168.210.200 –u –b <valor1>M –t 30 –i 1.* Figura 3.21.

  Donde:

  - *192.168.210.200* : dirección IP del host servidor.
  - *–u* : uso de paquetes UDP.
  - *–b* : ancho de banda a utilizar, 1 Mbps por default para UDP.

- *<valor1>* : velocidad nominal de transmisión.
- *M* : Mbps.
- *–t <valor>* : tiempo en segundos de la transmisión.
- *–i <valor>* : intervalo de tiempo en segundos entre reportes.

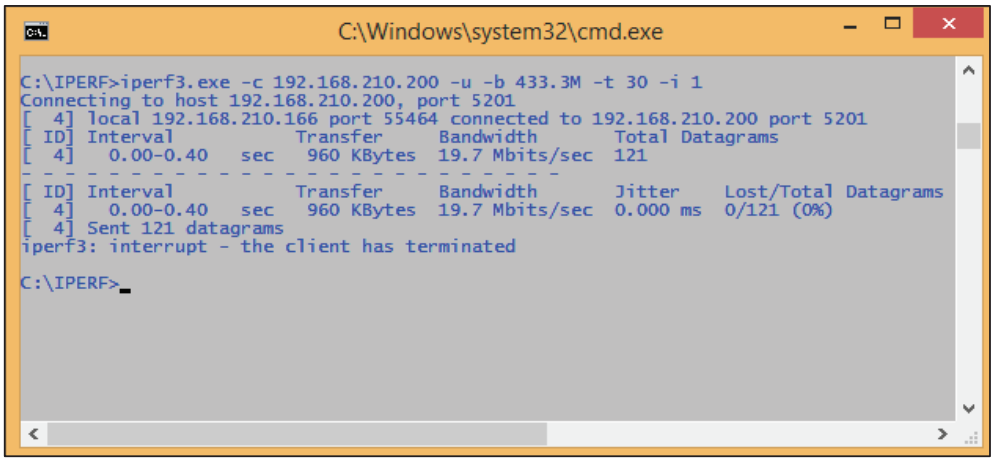


Figura 3.21 Configuración Iperf como cliente

La variable *<valor1>* dependerá del MCS definido en los equipos CISCO, por ejemplo, cuando el punto de acceso opera a 80 MHz, un *spatial stream*, intervalos de guarda de 400 nseg y MCS igual a 9 la velocidad nominal bajo estas condiciones es de 433.3 Mbps.

- Se configura la PC1 como servidor utilizando el comando *iperf –s.* Figura 3.22.

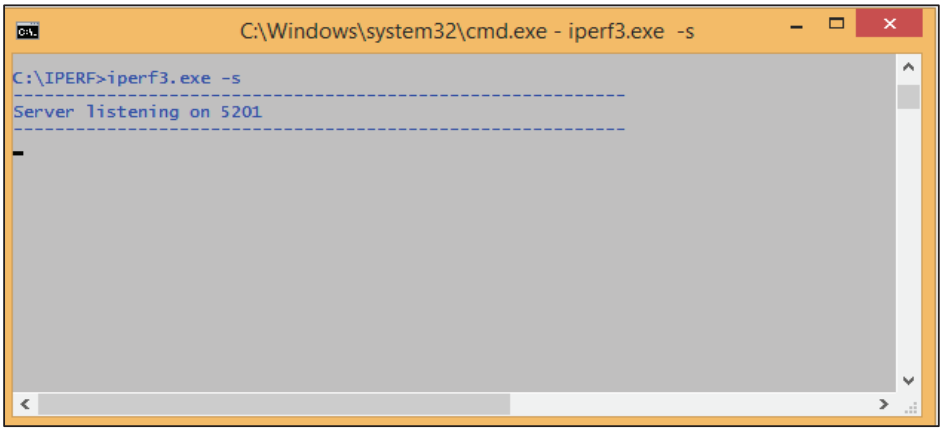


Figura 3.22 Configuración Iperf como servidor

- La distancia entre la PC1 y AP, PC2 y AP varian en 1, 5 y 10 metros para los dos ambientes.

3.4.3 EJECUCIÓN DE LAS PRUEBAS

3.4.3.1 Ambiente Interno

En este ambiente se presenta un ejemplo de la medición del *throughput* cuando el punto de acceso CISCO AIR-CAP3602I-A-K9 y el módulo CISCO AIR-RM3000AC-A-K9 trabajan con la siguiente configuración:

- Ancho de canal: 40 MHz
- Número de *spatial stream*: 1
- Intervalo de guarda: 400 nseg
- Frecuencia de operación: 5765 GHz

La Figura 3.23 muestra el estado del adaptador inalámbrico utilizado, se aprecia el SSID al cual está conectado (SSID_802.11ac) y la máxima velocidad teórica a la cual puede transmitir para este ejemplo (200,0 Mbps).

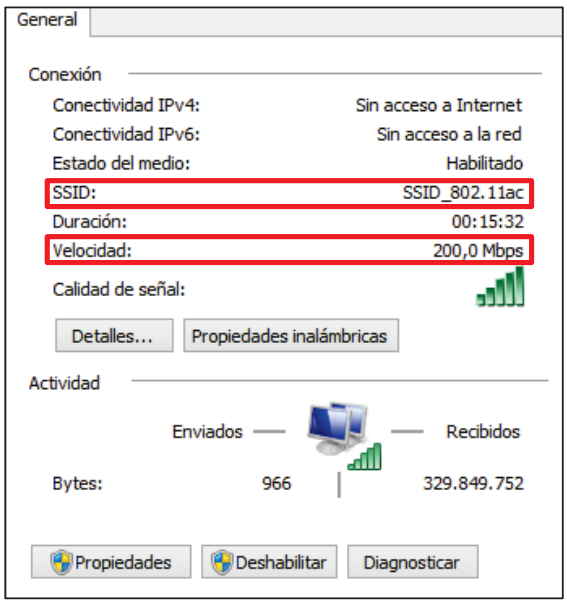
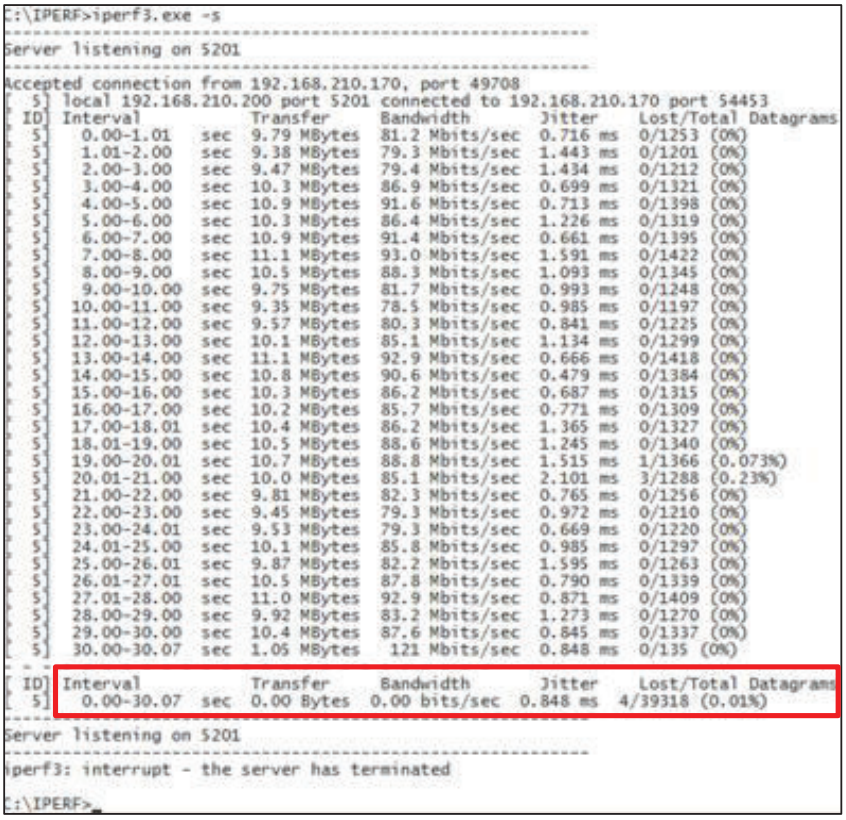


Figura 3.23 Estado adaptador inalámbrico AWUS036AC AC 1200 WIRELESS ADAPTER

En la Figura 3.24 se aprecia los resultados de la aplicación Iperf cuando la PC1 actúa como servidor.



```
C:\IPERF>iperf3.exe -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.210.170, port 49708
[  5] local 192.168.210.200 port 5201 connected to 192.168.210.170 port 54453
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-1.01   sec  9.79 MBytes  81.2 Mbits/sec  0.716 ms  0/1253 (0%)
[  5]   1.01-2.00   sec  9.38 MBytes  79.3 Mbits/sec  1.443 ms  0/1201 (0%)
[  5]   2.00-3.00   sec  9.47 MBytes  79.4 Mbits/sec  1.434 ms  0/1212 (0%)
[  5]   3.00-4.00   sec  10.3 MBytes  86.9 Mbits/sec  0.699 ms  0/1321 (0%)
[  5]   4.00-5.00   sec  10.9 MBytes  91.6 Mbits/sec  0.713 ms  0/1398 (0%)
[  5]   5.00-6.00   sec  10.3 MBytes  86.4 Mbits/sec  1.226 ms  0/1319 (0%)
[  5]   6.00-7.00   sec  10.9 MBytes  91.4 Mbits/sec  0.661 ms  0/1395 (0%)
[  5]   7.00-8.00   sec  11.1 MBytes  93.0 Mbits/sec  1.591 ms  0/1422 (0%)
[  5]   8.00-9.00   sec  10.5 MBytes  88.3 Mbits/sec  1.093 ms  0/1345 (0%)
[  5]   9.00-10.00  sec  9.75 MBytes  81.7 Mbits/sec  0.993 ms  0/1248 (0%)
[  5]  10.00-11.00  sec  9.35 MBytes  78.5 Mbits/sec  0.985 ms  0/1197 (0%)
[  5]  11.00-12.00  sec  9.57 MBytes  80.3 Mbits/sec  0.841 ms  0/1225 (0%)
[  5]  12.00-13.00  sec  10.1 MBytes  85.1 Mbits/sec  1.134 ms  0/1299 (0%)
[  5]  13.00-14.00  sec  11.1 MBytes  92.9 Mbits/sec  0.666 ms  0/1418 (0%)
[  5]  14.00-15.00  sec  10.8 MBytes  90.6 Mbits/sec  0.479 ms  0/1384 (0%)
[  5]  15.00-16.00  sec  10.3 MBytes  86.2 Mbits/sec  0.687 ms  0/1315 (0%)
[  5]  16.00-17.00  sec  10.2 MBytes  85.7 Mbits/sec  0.771 ms  0/1309 (0%)
[  5]  17.00-18.01  sec  10.4 MBytes  86.2 Mbits/sec  1.365 ms  0/1327 (0%)
[  5]  18.01-19.00  sec  10.5 MBytes  88.6 Mbits/sec  1.245 ms  0/1340 (0%)
[  5]  19.00-20.01  sec  10.7 MBytes  88.8 Mbits/sec  1.515 ms  1/1366 (0.073%)
[  5]  20.01-21.00  sec  10.0 MBytes  85.1 Mbits/sec  2.101 ms  3/1288 (0.23%)
[  5]  21.00-22.00  sec  9.81 MBytes  82.3 Mbits/sec  0.765 ms  0/1256 (0%)
[  5]  22.00-23.00  sec  9.45 MBytes  79.3 Mbits/sec  0.972 ms  0/1210 (0%)
[  5]  23.00-24.01  sec  9.53 MBytes  79.3 Mbits/sec  0.669 ms  0/1220 (0%)
[  5]  24.01-25.00  sec  10.1 MBytes  85.8 Mbits/sec  0.985 ms  0/1297 (0%)
[  5]  25.00-26.01  sec  9.87 MBytes  82.2 Mbits/sec  1.595 ms  0/1263 (0%)
[  5]  26.01-27.01  sec  10.5 MBytes  87.8 Mbits/sec  0.790 ms  0/1339 (0%)
[  5]  27.01-28.00  sec  11.0 MBytes  92.9 Mbits/sec  0.871 ms  0/1409 (0%)
[  5]  28.00-29.00  sec  9.92 MBytes  83.2 Mbits/sec  1.273 ms  0/1270 (0%)
[  5]  29.00-30.00  sec  10.4 MBytes  87.6 Mbits/sec  0.845 ms  0/1337 (0%)
[  5]  30.00-30.07  sec  1.05 MBytes   121 Mbits/sec  0.848 ms  0/135 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-30.07  sec  0.00 Bytes  0.00 bits/sec  0.848 ms  4/39318 (0.01%)
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
iperf3: interrupt - the server has terminated

C:\IPERF>_
```

Figura 3.24 Resultado de Iperf como Servidor

Adicionalmente, se muestra la configuración del AP en la Figura 3.25 donde se distinguen las variables antes mencionadas.

```
interface Dot11Radio2
 no ip address
 !
 ssid SSID_802.11ac
 !
 antenna gain 5
 antenna a-antenna
 peakdetect
 no dfs band block
 beamform ofdm
 speed  basic-6.0 9.0 12.0 18.0 24.0 36.0 48.0 54.0 m0. m1. m2. m3. m4. m5. m6.
m7. m8. m9. m10. m11. m12. m13. m14. m15. m16. m17. m18. m19. m20. m21. m22. m23
. a1ss9 a2ss9 a3ss9
 channel width 40-below
 channel 5765
 station-role root
 monitor frames endpoint ip address 0.0.0.1 port 10 truncate 0
 bridge-group 1
 bridge-group 1 subscriber-loop-control
 bridge-group 1 spanning-disabled
 bridge-group 1 block-unknown-source
 no bridge-group 1 source-learning
 no bridge-group 1 unicast-flooding
```

Figura 3.25 Configuración del punto de acceso Cisco

3.4.3.2 Ambiente Externo

De igual manera, se presenta un ejemplo de la medición del *throughput* para el ambiente externo. En este caso el punto de acceso tiene la siguiente configuración:

- Ancho de canal: 80 MHz
- Número de *spatial stream*: 1
- Intervalo de guarda: 800 nseg
- Frecuencia de operación: 5765 GHz

En la Figura 3.26 se presenta el estado del adaptador inalámbrico donde se aprecia el SSID al cual está conectado (SSID_802.11ac) y la máxima velocidad teórica a la cual puede transmitir para la configuración anterior (390,0 Mbps). En la Figura 3.27 se muestra la configuración del punto de acceso

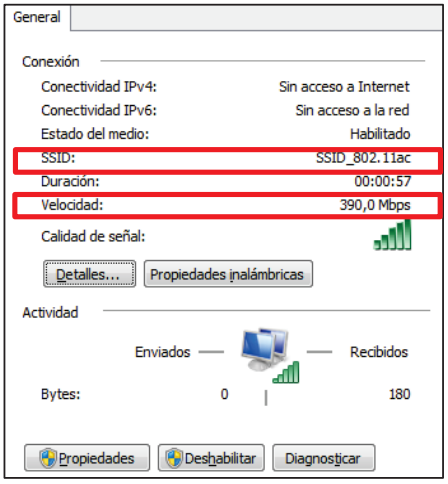


Figura 3.26 Estado del adaptador inalámbrico USB WiFi AC1200

```
interface Dot11Radio2
 no ip address
 !
 ssid SSID_802.11ac
 !
 antenna gain 5
 antenna a-antenna
 peakdetect
 no dfs band block
 beamform ofdm
 guard-interval long
 speed  basic-6.0 9.0 12.0 18.0 24.0 36.0 48.0 54.0 m0. m1. m2. m3. m4. m5. m6. m7. m8.
11. m12. m13. m14. m15. m16. m17. m18. m19. m20. m21. m22. m23. a1ss9 a2ss9 a3ss9
 channel width 80
 channel 5765
 station-role root
 monitor frames endpoint ip address 0.0.0.1 port 10 truncate 0
 bridge-group 1
 bridge-group 1 subscriber-loop-control
 bridge-group 1 spanning-disabled
 bridge-group 1 block-unknown-source
 no bridge-group 1 source-learning
 no bridge-group 1 unicast-flooding
```

Figura 3.27 Configuración del punto de acceso Cisco

En la Figura 3.28 se aprecia los resultados de la aplicación Iperf cuando la PC2 actúa como cliente.

```
C:\Users\Franklin\Desktop\IPERF>iperf3.exe -c 192.168.210.200 -u -b 390M -t 30 -i 1
Connecting to host 192.168.210.200, port 5201
[  4] local 192.168.210.160 port 54765 connected to 192.168.210.200 port 5201
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.00   sec  8.19 MBytes  68.7 Mbits/sec  1048
[  4]   1.00-2.00   sec  10.1 MBytes  84.3 Mbits/sec  1287
[  4]   2.00-3.00   sec  9.45 MBytes  79.3 Mbits/sec  1210
[  4]   3.00-4.00   sec  9.65 MBytes  80.9 Mbits/sec  1235
[  4]   4.00-5.00   sec  9.52 MBytes  79.8 Mbits/sec  1218
[  4]   5.00-6.00   sec  8.76 MBytes  73.5 Mbits/sec  1121
[  4]   6.00-7.00   sec  10.2 MBytes  85.4 Mbits/sec  1303
[  4]   7.00-8.00   sec  9.29 MBytes  77.9 Mbits/sec  1189
[  4]   8.00-9.00   sec  9.35 MBytes  78.4 Mbits/sec  1197
[  4]   9.00-10.00  sec  9.81 MBytes  82.3 Mbits/sec  1256
[  4]  10.00-11.00  sec  9.54 MBytes  80.0 Mbits/sec  1221
[  4]  11.00-12.00  sec  10.1 MBytes  85.1 Mbits/sec  1298
[  4]  12.00-13.00  sec  9.34 MBytes  78.3 Mbits/sec  1195
[  4]  13.00-14.00  sec  9.14 MBytes  76.7 Mbits/sec  1170
[  4]  14.00-15.00  sec  9.58 MBytes  80.3 Mbits/sec  1226
[  4]  15.00-16.00  sec  9.23 MBytes  77.4 Mbits/sec  1181
[  4]  16.00-17.00  sec  9.62 MBytes  80.7 Mbits/sec  1232
[  4]  17.00-18.00  sec  9.51 MBytes  79.8 Mbits/sec  1217
[  4]  18.00-19.00  sec  9.80 MBytes  82.2 Mbits/sec  1254
[  4]  19.00-20.00  sec  9.66 MBytes  81.0 Mbits/sec  1236
[  4]  20.00-21.00  sec  9.12 MBytes  76.5 Mbits/sec  1168
[  4]  21.00-22.01  sec  7.84 MBytes  65.1 Mbits/sec  1003
[  4]  22.01-23.00  sec  8.20 MBytes  69.5 Mbits/sec  1050
[  4]  23.00-24.00  sec  8.95 MBytes  75.0 Mbits/sec  1145
[  4]  24.00-25.00  sec  9.12 MBytes  76.5 Mbits/sec  1168
[  4]  25.00-26.00  sec  9.59 MBytes  80.4 Mbits/sec  1227
[  4]  26.00-27.00  sec  8.68 MBytes  72.8 Mbits/sec  1111
[  4]  27.00-28.00  sec  9.22 MBytes  77.2 Mbits/sec  1180
[  4]  28.00-29.00  sec  9.43 MBytes  79.1 Mbits/sec  1207
[  4]  29.00-30.00  sec  9.52 MBytes  79.8 Mbits/sec  1218
```

Figura 3.28 Resultado de Iperf como Cliente

CAPÍTULO IV

ANÁLISIS DE RESULTADOS

4.1 INTRODUCCIÓN

En este capítulo se presenta los resultados obtenidos en la simulación desarrollada en el capítulo II para los modelos A, B y C; se muestran los enlaces entre las antenas de transmisión-recepción y los valores de data rate por medio de curvas representadas gráficamente.

De igual manera se muestra los valores de velocidades alcanzadas con el punto de acceso AIR-CAP3602I-A-K9 y el módulo AIR-RM3000AC-A-K9 para las pruebas realizadas tanto para ambientes internos como externos. En este apartado se tomaron medidas para 8 escenarios en ambientes internos y externos obteniendo el valor promedio para cada caso.

Finalmente, se realiza la comparación de los valores obtenidos de la parte práctica con los calculados en la simulación.

4.2 RESULTADOS DE LA SIMULACIÓN

La simulación desarrollada para el estándar 802.11ac se aplica a los tres modelos WLAN elegidos, a continuación se presenta un ejemplo para cada uno de ellos.

4.2.1 Simulación – Modelo A

Para el modelo A, con dos usuarios y dos antenas para cada uno de ellos (4 *spatial streams)* se establece un índice MCS de 9 (Modulación= 256QAM, Codificación=5/6), un valor de intervalo de guarda de 400 nseg, un valor de SNR

de 33 dB y valor de ancho de banda de canal de 40 MHz. En la Figura 4.1 se presentan los datos mencionados.

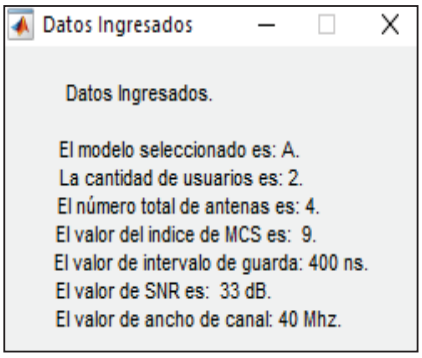


Figura 4.1 Datos Ingresados

- Resultados gráficos de los enlaces

En este modelo se define la condición de NLOS (Desvanecimiento Rayleigh), así en las Figuras 4.2 y 4.3 se presentan los enlaces entre las antenas transmisoras y las antenas receptoras de cada usuario, donde además se señala que para Δt1 se produce un desvanecimiento considerable y para Δt2 se tiene períodos de tiempo sin desvanecimiento, los valores de amplitud para cada usuario son diferentes esto debido a las condiciones aleatorias asumidas en la simulación.

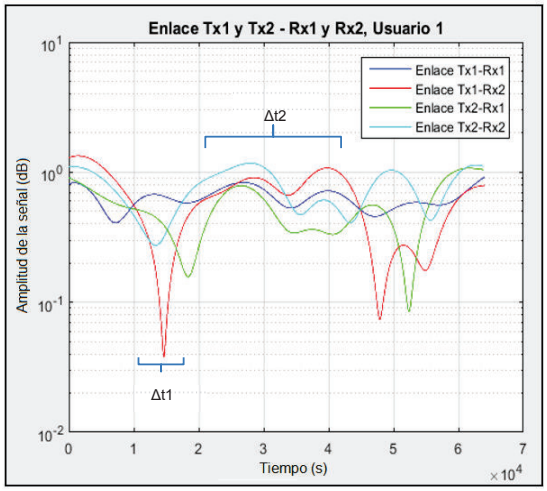


Figura 4.2 Usuario 1 – Enlaces

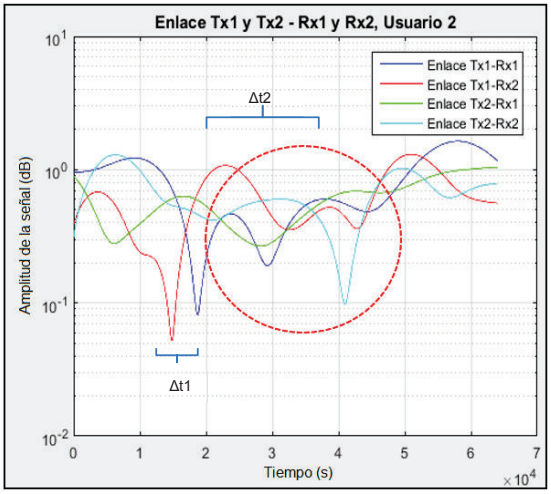


Figura 4.3 Usuario 2 – Enlaces

- Resultado Data_Rate

El valor de *Data_Rate* obtenido en la simulación para el modelo A con los datos ingresados (ver Figura 4.1) es igual al valor que asigna el estándar 802.11ac en [34]. A continuación se presentan los resultados calculados de Data Rate, Capacidad del Canal y Eficiencia del Sistema de la simulación por medio de las fórmulas de la sección 2.4.2.18.

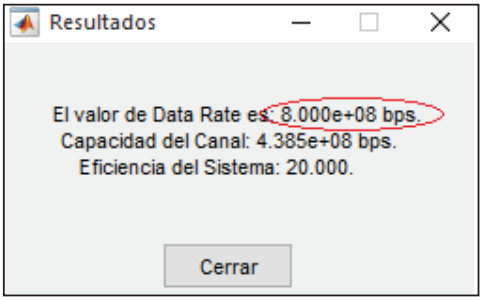


Figura 4.4 Simulación – Resultado Calculado

En la Figura 4.5 se indica el valor que establece el estándar 802.11ac donde se señalan los parámetros de ancho de banda, cantidad de *spatial stream,* valor de índice MCS y el intervalo de guarda con los cual se comprueba el resultado obtenido en la simulación igual a 800 Mbps.

MCS : Index

802.11n							802.11ac				
HT MCS Index	Spatial Streams	Modulation & Coding	Data Rate (GI = 800ns) 20MHz	Data Rate (GI = 400ns) 20MHz	Data Rate (GI = 800ns) 40MHz	Data Rate (GI = 400ns) 40MHz	Data Rate (GI = 800ns) 80MHz	Data Rate (GI = 400ns 80MHz	Data Rate (GI = 800ns) 160MHz	Data Rate (GI = 400ns) 160MHz	VHT MCS Index
24	4	BPSK 1/2	26	28.9	54	60	117	130	234	260	0
25	4	QPSK 1/2	52	57.8	108	120	234	260	468	520	1
26	4	QPSK 3/4	78	86.7	162	180	351	390	702	780	2
27	4	16-QAM 1/2	104	115.6	216	240	468	520	936	1040	3
28	4	16-QAM 3/4	156	173.3	324	360	702	780	1404	1560	4
29	4	64-QAM 2/3	208	231.1	432	480	936	1040	1872	2080	5
30	4	64-QAM 3/4	234	260	486	540	1053	1170	2106	2340	6
31	4	64-QAM 5/6	260	288.9	540	600	1170	1300	2340	2600	7
	4	256-QAM 3/4	312	346.7	648	720	1404	1560	2808	3120	8
	4	256-QAM 5/6	n/a	n/a	720	800	1560	1733.3	3120	3466.7	9

Figura 4.5 MCS Index – Modelo A [34]

4.2.2 Simulación – Modelo B

En este ejemplo para el modelo B, se dispone de 4 usuarios con 1, 2, 1, y 3 antenas respectivamentedos (7 *spatial streams).* Se ha elegido un índice MCS de 8 (Modulación= 256QAM, Codificación=3/4), un valor de intervalo de guarda de 800 nseg, un valor de SNR de 37 dB y ancho de banda de canal igual a 160 MHz. La Figura 4.6 detalla los parámetros ingresados.

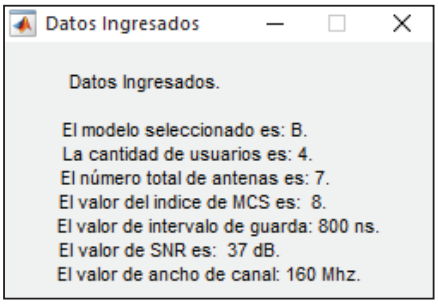


Figura 4.6 Datos Ingresados

- Resultado gráficos de los enlaces

En el modelo B se estable la condición LOS (Desvanecimiento Rician). Donde el usuario 1 tiene una antena (Enlace Tx1-Rx1), el usuario 2 con dos antenas (Enlaces: Tx1-Rx1, Tx1-Rx2, Tx2-Rx1, Tx2-Rx2), el usuario 3 con una antena (Enlace Tx1-Rx1) y el usuario 4 con tres antenas (Enlaces: Tx1-Rx1, Tx1-Rx2, Tx1-Rx3, Tx2-Rx1, Tx2-Rx2, Tx2-Rx3, Tx3-Rx1, Tx3-Rx2, Tx3-Rx3). Las Figuras 4.7, 4.8, 4.9 y 4.10 representan los enlaces entre las antenas transmisoras y las antenas receptoras de cada usuario, donde se observa el efecto MU-MIMO y se determina la semejanza entre los enlaces participantes en la transmisión, así mientras aumenta la cantidad de antenas para un usuario aumenta la cantidad de enlaces y por lo tanto se puede tener la probabilidad de que exista una mayor similitud entre las señales (comparando las Figuras 4.3 y 4.10 por medio de la circunferencia roja).

En las figuras también se observa que para la condición de LOS en el período $\Delta t1$ no se produce cambios considerables en los enlaces esto debido a que se tiene una señal dominante para un desvanecimiento Rician.

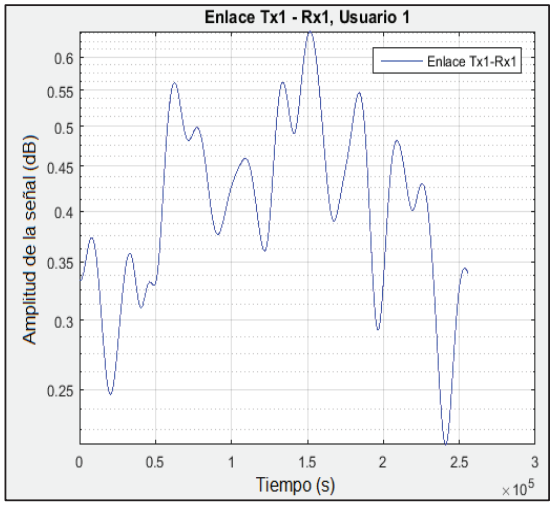


Figura 4.7 Usuario 1 – Enlaces

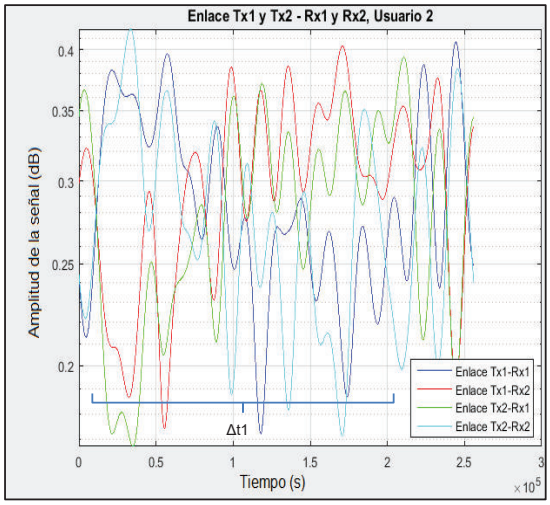


Figura 4.8 Usuario 2 – Enlaces

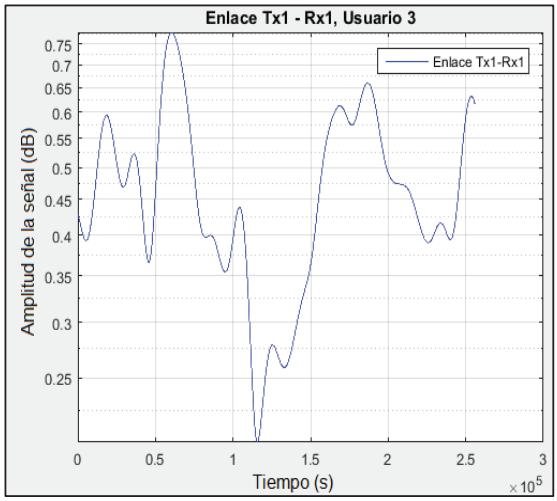


Figura 4.9 Usuario 3 – Enlaces

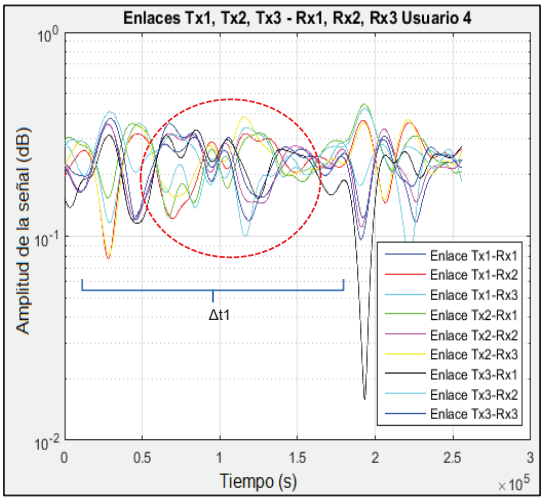


Figura 4.10 Usuario 4 – Enlaces

- Resultado

El valor de *Data_Rate* obtenido en la simulación para el modelo B se presenta en la Figura 4.11, siendo igual al valor que determina el estándar 802.11ac en [34]. A continuación se presentan los resultados calculados de Data Rate, Capacidad del Canal y Eficiencia del Sistema de la simulación por medio de las fórmulas de la sección 2.4.2.18.

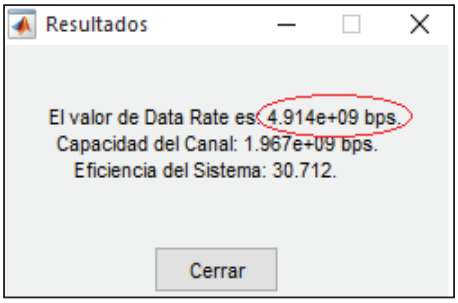


Figura 4.11 Resultado – Modelo B

En la Figura 4.12 se indica el valor que establece el estándar 802.11ac donde se señalan los parámetros de ancho de banda, cantidad de *spatial stream,* valor de índice MCS y el intervalo de guarda, con lo cual se comprueba el resultado obtenido en la simulación igual a 4.914 Gbps.

MCS : Index

802.11n											802.11ac
HT MCS Index	Spatial Streams	Modulation & Coding	Data Rate (GI = 800ns) 20MHz	Data Rate (GI = 400ns) 20MHz	Data Rate (GI = 800ns) 40MHz	Data Rate (GI = 400ns) 40MHz	Data Rate (GI = 800ns) 80MHz	Data Rate (GI = 400ns) 80MHz	Data Rate (GI = 800ns) 160MHz	Data Rate (GI = 400ns) 160MHz	VHT MCS Index
	7	BPSK 1/2					204.8	227.5	409.5	455	0
	7	QPSK 1/2					409.5	455	819	910	1
	7	QPSK 3/4					614.3	682.5	1228.5	1365	2
	7	16-QAM 1/2					819	910	1638	1820	3
	7	16-QAM 3/4					1228.5	1365	2457	2730	4
	7	64-QAM 2/3					1638	1820	3276	3640	5
	7	64-QAM 3/4					n/a	n/a	3685.5	4095	6
	7	64-QAM 5/6					2047.5	2275	4095	4550	7
	7	256-QAM 3/4					2457	2730	4914	5460	8
	7	256-QAM 5/6					2730	3033.3	5460	6066.7	9

Figura 4.12 MCS Index – Modelo B

4.2.3 Simulación – Modelo C

En este ejemplo para el modelo C, se dispone de 3 usuarios con 2,1 y 3 antenas respectivamentedos (6 *spatial streams).* Se escogido un índice MCS de 6 (Modulación= 64-QAM, Codificación=3/4), un valor de intervalo de guarda de 400 nseg, un valor de SNR de 24 dB y ancho de banda de canal igual a 80 MHz. La Figura 4.13 detalla estos parámetros ingresados.

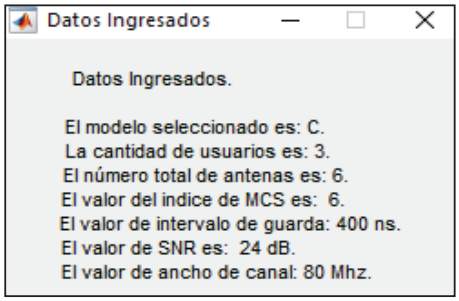


Figura 4.13 Datos Ingresados

- Resultado gráficos de los enlaces

En este modelo se define la condición de NLOS (Desvanecimiento Rayleigh), así en las Figuras 4.14, 4.15 y 4.16 se presentan los enlaces entre las antenas transmisoras y las antenas receptoras de cada usuario, donde además se señala que para Δt1 se produce un desvanecimiento considerable y para Δt2 se tiene períodos de tiempo sin desvanecimiento, los valores de amplitud para cada usuario son diferentes esto debido a las condiciones aleatorias asumidas en la simulación.

En las figuras se puede observar que mientras aumenta la cantidad de antenas para un usuario, aumenta la cantidad de enlaces y, por lo tanto, se puede tener la probabilidad de que existan señales muy semejantes (circunferencia roja).

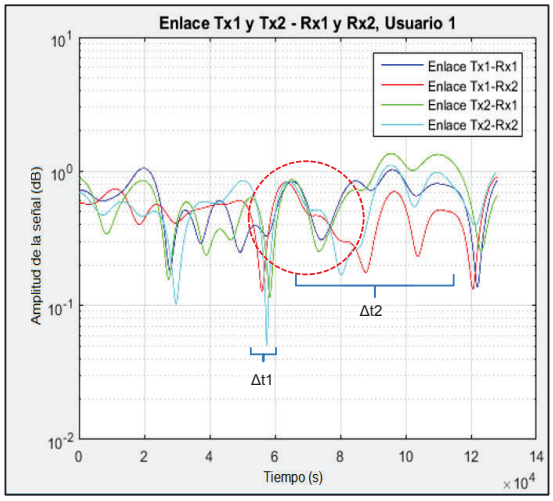


Figura 4.14 Usuario 1 – Enlaces

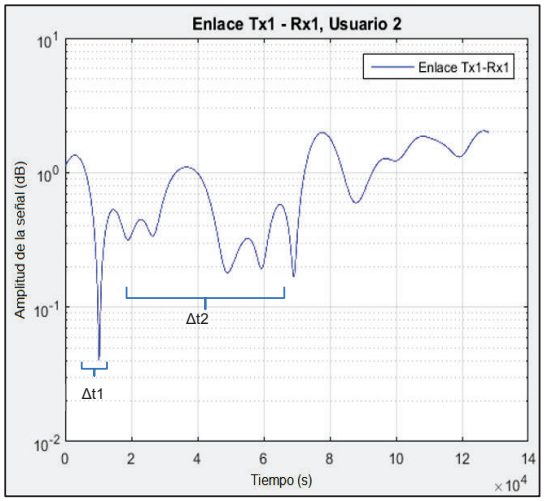


Figura 4.15 Usuario 2 – Enlaces

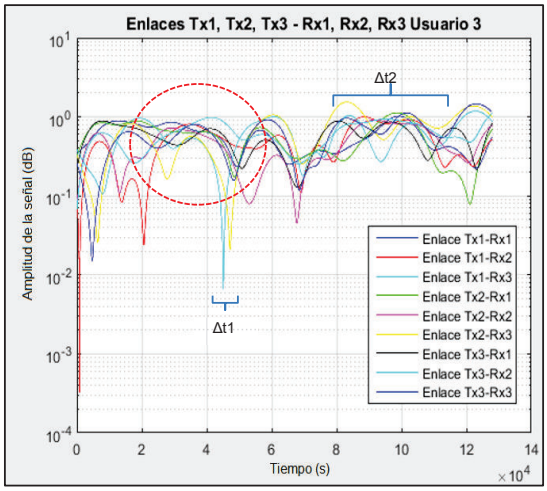


Figura 4.16 Usuario 3 – Enlaces

- Resultado

El valor de *Data_Rate* obtenido en la simulación para el modelo A con los datos ingresados (ver Figura 4.1) es igual al valor que asigna el estándar 802.11ac en [34]. A continuación se presentan los resultados calculados de Data Rate, Capacidad del Canal y Eficiencia del Sistema de la simulación por medio de las fórmulas de la sección 2.4.2.18.

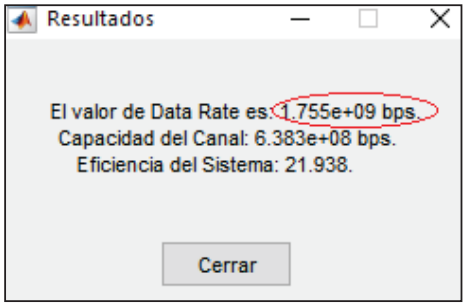


Figura 4.17 Resultados Modelo C

En la Figura 4.18 se indica el valor que establece el estándar 802.11ac donde se señalan los parámetros de ancho de banda, cantidad de *spatial stream,* valor de índice MCS y el intervalo de guarda, con lo cual se comprueba el resultado obtenido en la simulación igual a 1.755 Gbps.

MCS : Index

802.11n											802.11ac
HT MCS Index	Spatial Streams	Modulation & Coding	Data Rate (GI = 800ns) 20MHz	Data Rate (GI = 400ns) 20MHz	Data Rate (GI = 800ns) 40MHz	Data Rate (GI = 400ns) 40MHz	Data Rate (GI = 800ns) 80MHz	Data Rate (GI = 400ns) 80MHz	Data Rate (GI = 800ns) 160MHz	Data Rate (GI = 400ns) 160MHz	VHT MCS Index
	6	QPSK 1/2					351	390	702	780	1
	6	QPSK 3/4					526.5	585	1053	1170	2
	6	16-QAM 1/2					702	780	1404	1560	3
	6	16-QAM 3/4					1053	1170	2106	2340	4
	6	64-QAM 2/3					1404	1560	2808	3120	5
	6	64-QAM 3/4					1579.5	1755	3159	3510	6
	6	64-QAM 5/6					1755	1950	3510	3900	7
	6	256-QAM 3/4					2106	2340	4212	4680	8
	6	256-QAM 5/6					n/a	n/a	4680	5200	9

Figura 4.18 MCS Index – Modelo C

4.3 RESULTADOS DE LA PARTE PRÁCTICA

Los resultados de la parte práctica para los dos ambientes internos y externos se detallan a continuación.

4.3.1 Ambiente Interno

La parte práctica en el ambiente interno, se realizó en dos pisos (Planta Baja, Piso1), para lo cual se presenta mapas de calor del AP que indican la cobertura en estás areás.

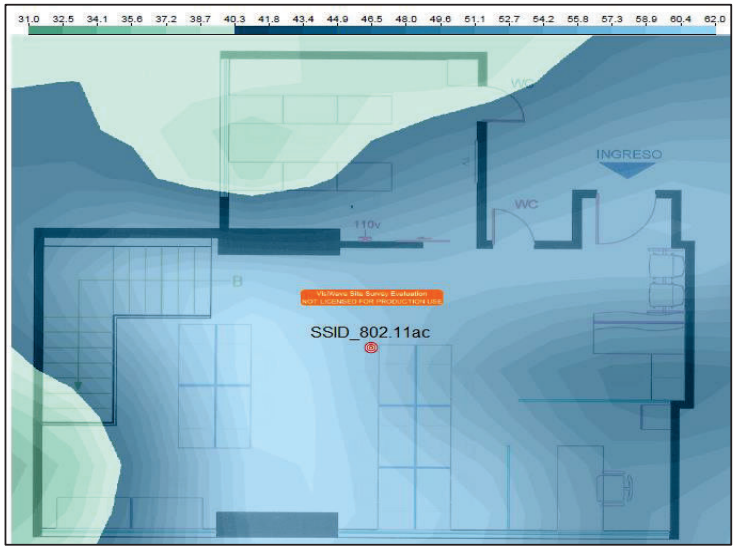


Figura 4.19 Mapa de Calor – Primer Piso

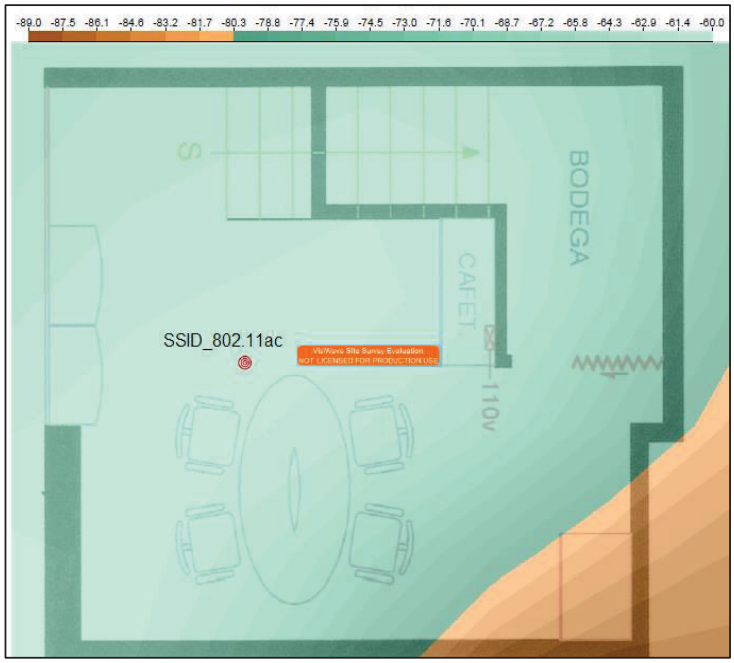


Figura 4.20 Mapa de Calor – Planta Baja

La herramienta que se utilizó para generar los mapas de calor del AP es VisiWave Site Survey. En la Figura 4.19 se observa que la intensidad de la señal del SSID_802.11ac esta entre 31.0 a 62 dBm lo cual es aceptable, en cambio para la planta baja (Figura 4.20) la intensidad está entre 60 a 90 dBm lo cual significa que la señal esta en un nivel bajo, en estas figuras el cuadro de color naranja indica un mensaje del programa que detalla que es una versión de evaluación.

Para distancias de 1 y 5 metros entre el AP y los dispositivos las pruebas se realizaron en el Piso 1, mientras que para 10 metros el AP y PC1 se encontraban en el piso 1 y PC2 en la planta baja. En el AP se configuró los 8 escenarios que se explican en Tabla 3.3.

Los resultados obtenidos con la herramienta IPERF para los ocho escenarios a las distancias mencionadas anteriormente se muestran en el anexo F, y en las Figuras 4.21, 4.22 y 4.23 se grafican los valores medios de cada escenario,

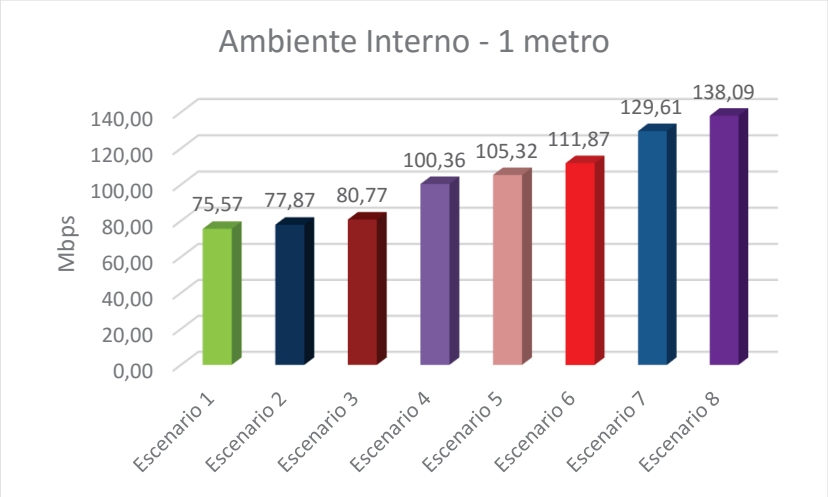


Figura 4.21 Valores medios – 1 metro

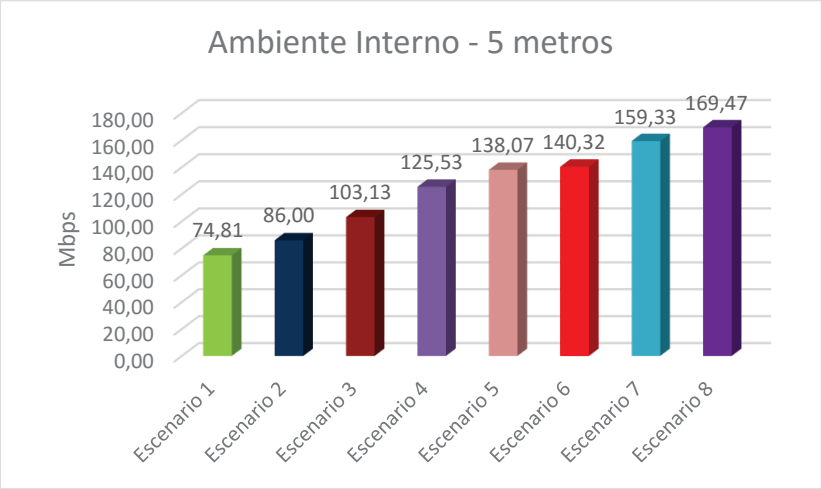


Figura 4.22 Valores medios – 5 metros

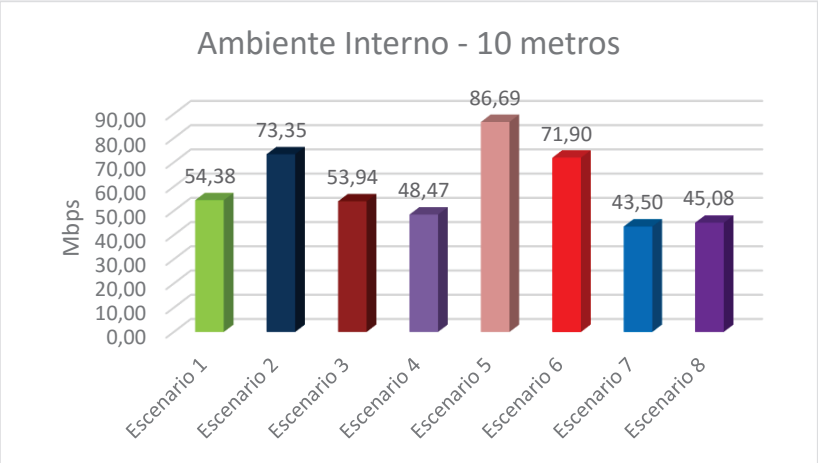


Figura 4.23 Valores medios – 10 metros

Se puede observar que para 5 metros se tiene un valor más elevado de las velocidades alcanzadas, en comparación a 1 y 10 metros. Esto se debe que cuando los dispositivos están tan cerca del AP el nivel de la señal es bajo debido al patrón de radiación de las antenas, el cual es omnidireccional y a una distancia de 10 metros el nivel disminuye por los obstáculos presentes.

4.3.2 Ambiente Externo

En este ambiente se realizaron las mediciones para los 8 escenarios indicados en la Tabla 3.3 de forma similar que en el ambiente interno. A continuación se presenta la cobertura del AP en está área.

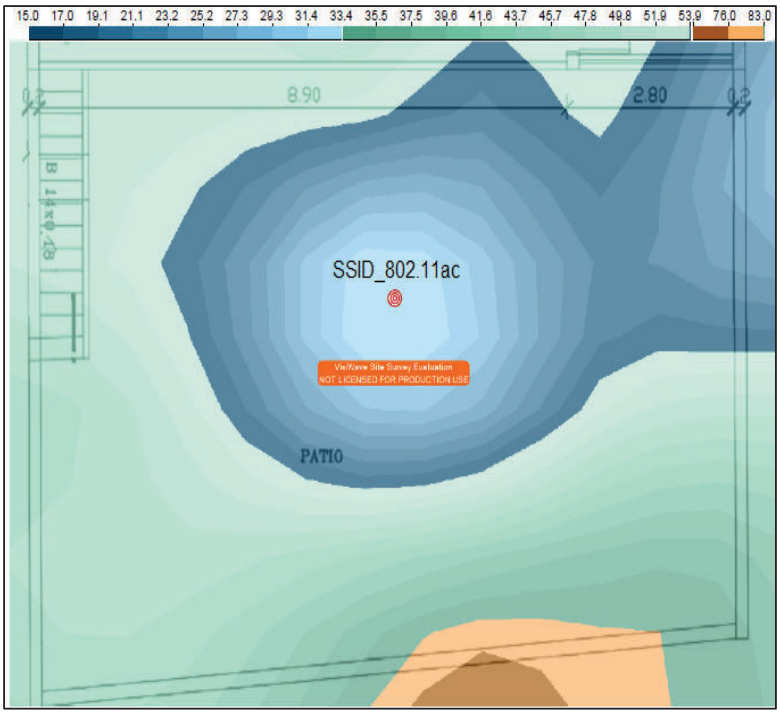


Figura 4.24 Mapa de Calor – Ambiente Externo

Los valores obtenidos con la herramienta IPERF se presentan en el anexo G, las siguientes Figuras 4.25, 4.26, y 4.27 representan los valores medios de los 8 escenarios para las distancias de 1 metro, 5 metros y 10 metros.

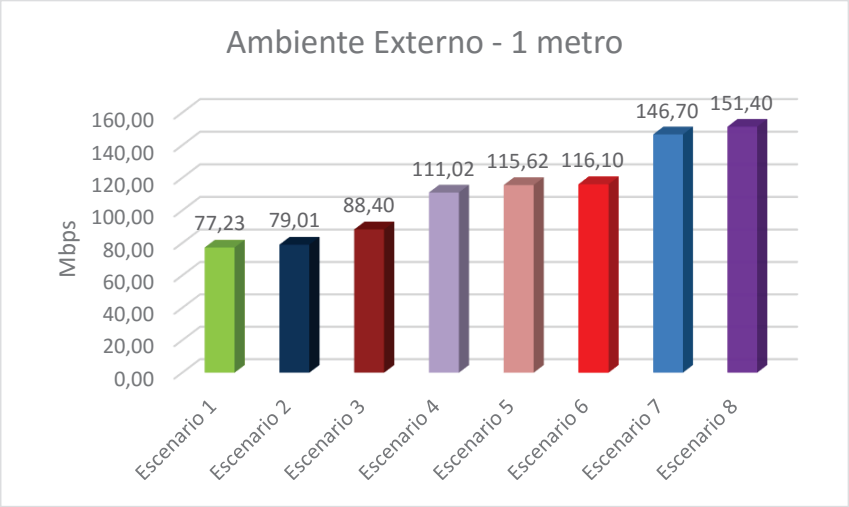


Figura 4.25 Valores medios – 1 metro

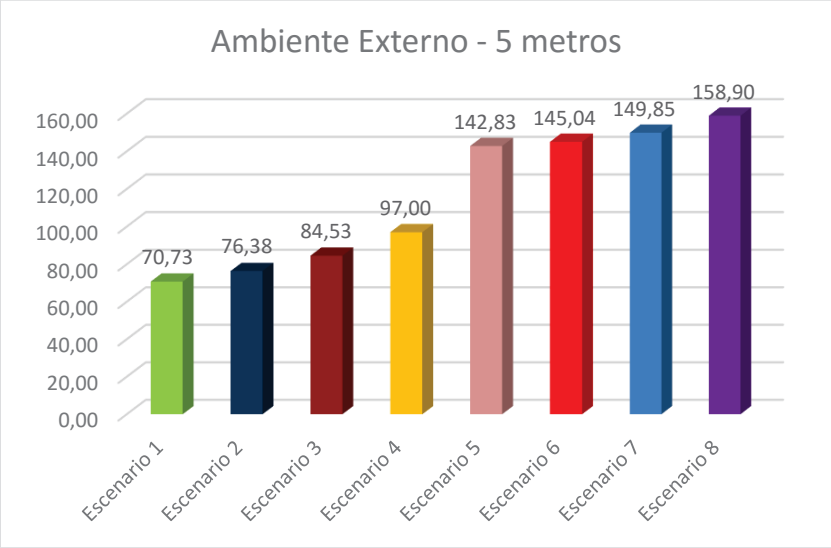


Figura 4.26 Valores medios – 5 metros

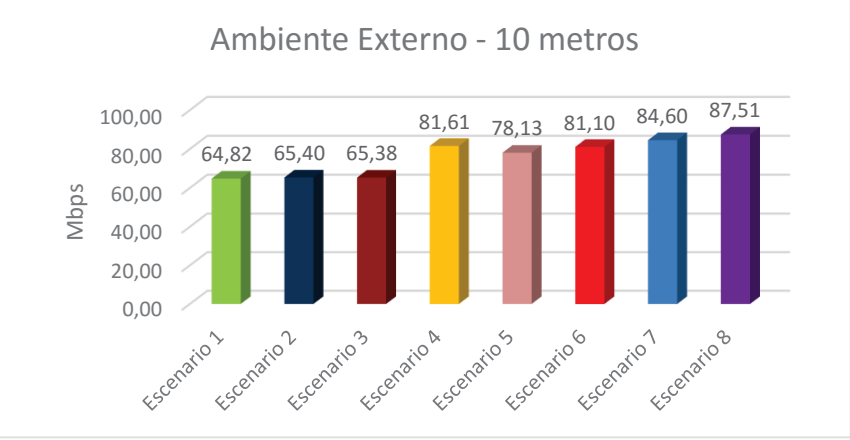


Figura 4.27 Valores medios – 10 metros

Estos resultados muestran de igual forma que para una distancia de 5 metros los valores son máximos y además a 10 metros ya no existe una variación de las velocidades en cada escenario como en el ambiente interno sino al contrario se produce un aumento continuo.

4.4 COMPARACIÓN DE RESULTADOS

En este apartado se realiza la comparación de los valores obtenidos en la parte práctica con los obtenidos en la simulación que son aproximadamente iguales a los que se define en el estándar 802.11ac.

En las Figuras 4.28 y 4.29 se muestran las comparaciones de los valores obtenidos en el desarrollo de las pruebas con los valores del Estándar 802.11ac.

Los *data rates* que se pueden lograr, en teoría en el estándar, para un ambiente real como los descritos anteriormente no pueden ser alcanzados por los dispositivos finales que se utilizaron.

En este punto es importante tomar en cuenta que tanto el punto de acceso como los dispositivos negocian el estado del enlace y las velocidades a transmitir, por tanto cuando la tasa de errores se incrementa la solución es cambiar la velocidad de modulación, el tipo de modulación o ambas de tal manera que la calidad del enlace se vea menos afectado posible por las condiciones de transmisión.

Además, el MCS máximo utilizado en una comunicación inalámbrica en condiciones de distancia, obstáculos, interferencias, viene dado por el dispositivo con el MCS más bajo. Por ejemplo, si intentamos establecer una comunicación con el conjunto CISCO AIR-CAP3602I-A-K9 y CISCO AIR-RM3000AC-A-K9, cuyo MCS más alto es 9, la máxima velocidad teórica a alcanzar sería 1300 Mbps, y un cliente 802.11ac que implemente un MCS máximo de 3 la velocidad máxima teórica sería de 390 Mbps en las mismas condiciones. El MCS utilizado en una comunicación Wi-Fi lo establecerá el dispositivo con menores prestaciones. [35]

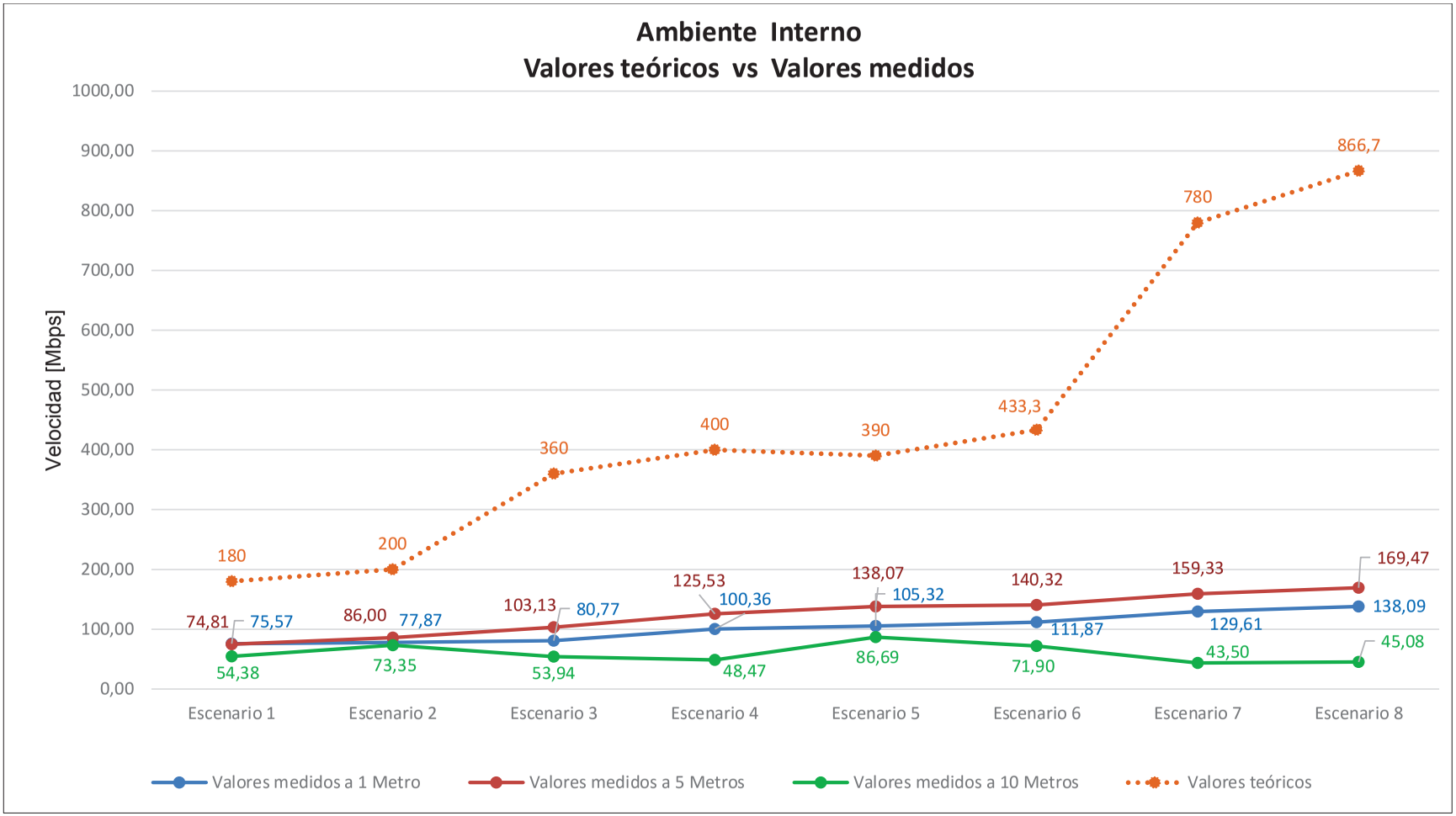


Figura 4.28 Comparación valores teóricos con valores medidos en ambiente interno

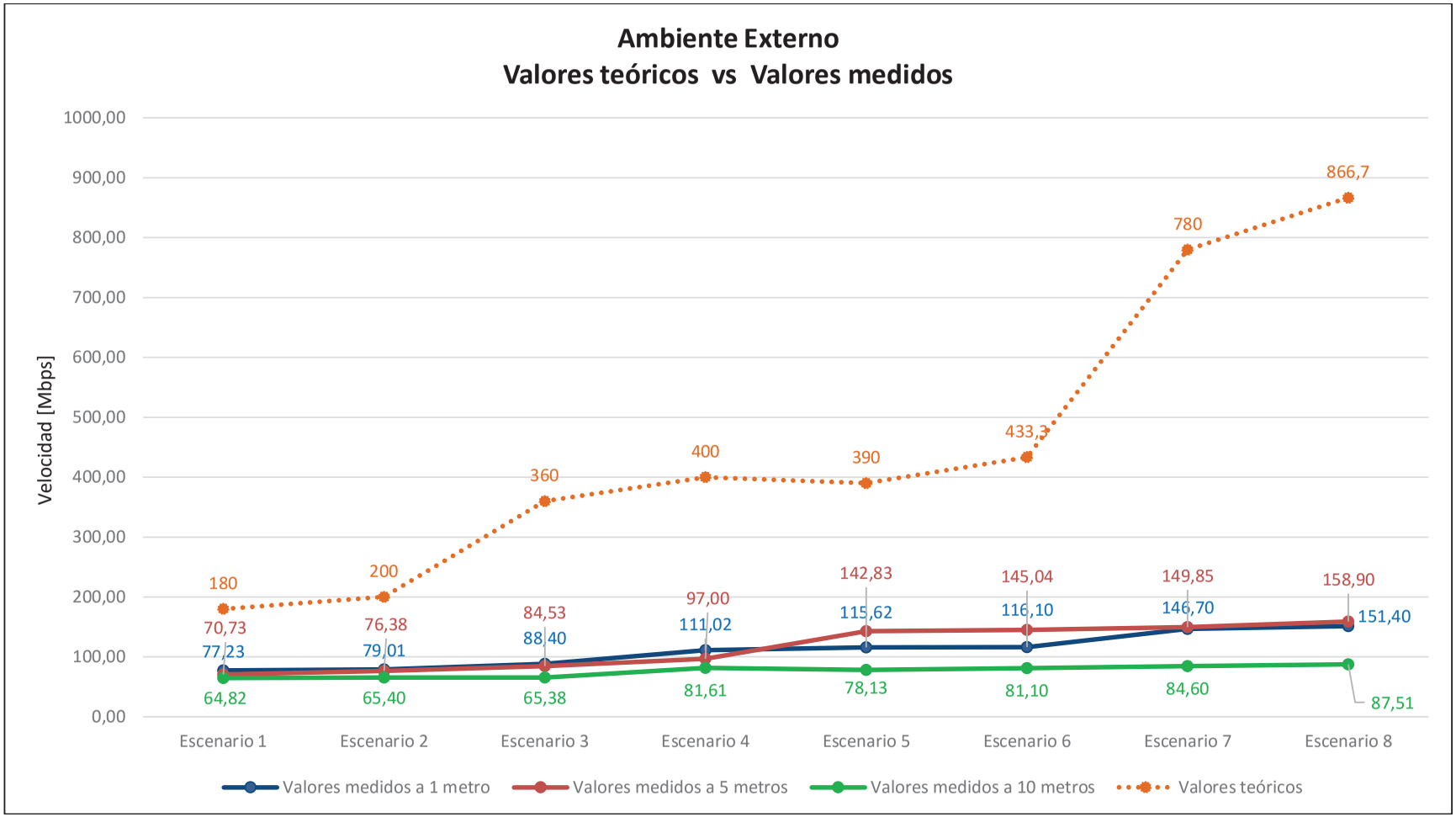


Figura 4.29 Comparación valores teóricos con valores medidos en ambiente externo

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- En este trabajo se presenta una descripción detallada de las avances propuestos por el estándar IEEE 802.11ac en relación a sus predecesores. Las mejoras en el rendimiento se deben al uso de un mayor ancho de banda de canal, mayor número de *spatial streams*, mayor orden de modulación, manejo dinámico del ancho de banda y MIMO multiusuario; los cuales constituyen la clave para lograr las altas tasas de velocidades de datos que se obtienen en esta enmienda.

- Esta nueva tecnología afronta grandes retos, entre ellos, requisitos de hardware superior, compatibilidad con estándares anteriores y un bajo costo de dispositivos finales. El estándar 802.11ac es compatible con los estándares 11a y 11n por operar en la misma banda de frecuencia, utilizar OFDM, MIMO y STBC.

- La trama 802.11ac consiste en muchos campos de preámbulo, seguido de los símbolos que representan los datos. Cada campo es construido por el transmisor para cumplir un propósito específico, algunos de estos campos contienen información acerca de la trama transmitida, y otros se utilizan para las estimaciones de canal. Los datos transmitidos se codifican con diferentes técnicas y operaciones para añadir robustez; estas operaciones se invierten en el lado del receptor para extraer los datos de nuevo.

- En la actualidad se torna difícil encontrar dispositivos finales capaces de abarcar todas las características y opciones que ofrece la norma, incluso la fase Wave 1 del estándar, que está vigente, es soportada por equipos terminales avanzados.

- De los resultados gráficos obtenidos de la simulación se observa que, al aumentar el ancho de banda de canal la correlación entre las señales transmitidas será mayor. Lo mencionado es importante ya que de darse este caso las señales se interferirán unas con otras haciendo muy difícil la recuperación de los datos enviados en el lado del receptor.

- Los equipos con los cuales se desarrollaron las pruebas (CISCO AIR-CAP3602I-A-K9 y AIR-RM3000AC-A-K9) se configuran de manera particular para que ofrezcan las características del estándar IEEE 802.11ac; por ejemplo, se necesita que se configure un ancho de banda de 40 MHz en 802.11n para que opere en IEEE 802.11ac con 80 MHz.

- Otra particularidad de configuración del AP CISCO AIR-CAP3602I-A-K9 y del módulo AIR-RM3000AC-A-K9 es que trabajando en modo autónomo no son compatibles con los algoritmos de seguridad TKIP y TKIP + AES. Esta falencia se supera al integrar estos equipos a un equipo adicional (WLC) para que este último sea el que administre, entre otras cosas, el tema de seguridad.

- De los resultados obtenidos en la parte práctica se puede concluir que mientras se trabaja con frecuencias altas en ambientes WLAN, el área de cobertura a la que puede servir un AP disminuye en comparación con estándares que trabajan en la banda de 2.4 GHz. De igual manera, al alejar demasiado los equipos del AP el throuhput disminuye hasta llegar un punto que el terminal de usuario se desasocia de la red.

- En las mediciones realizadas se observa que los valores de velocidades de datos a distancias de 1 metro son menores en relación a las velocidades

obtenidas a distancias de 5 metros. Esto se debe al patrón de radiación de las antenas omnidireccionales del AP CISCO AIR-CAP3602I-A-K9, dicho patrón se asemeja a la forma de una dona en donde los equipos más cercanos “verán” menor potencia de radiación que los que estén un poco más distantes.

- Analizando los valores medidos en la parte práctica; por ejemplo, para el escenario 5 en un ambiente interno a 5 metros de distancia entre los dispositivos se obtuvo un valor de velocidad de 138.07 Mbps, comparando con el valor teórico que es igual a 390 Mbps en las mismas condiciones se concluye que la diferencia entre estos valores se debe a que el valor teórico establecido en el estándar IEEE 802.11ac es calculado en condiciones ideales mientras que los medidos en la parte práctica se alcanzaron en un ambiente con presencia de obstáculos provocando la degradación de la señal y por ende su velocidad de transmisión.

- En la simulación y en la parte práctica se trató que las condiciones del canal de transmisión sean lo más ideales posibles con el objeto que los resultados estén lo más próximos a los que dictamina la enmienda.

- Con los equipos terminales utilizados no se logra alcanzar la máxima velocidad establecida por el estándar, de hecho la brecha que existe entra los resultados obtenidos y las velocidad teóricas es considerable. Este comportamiento se debe que los equipos, tanto AP como dispositivos finales, negocian las condiciones idóneas para poder enviar y recibir datos tomándose como referencia al equipo con las menores capacidades.

5.2 **Recomendaciones**

- Se puede tomar el presente trabajo de titulación como referencia y base con miras a continuar con el estudio de este estándar de la IEEE en temas de

métodos de recepción, demodulación de datos, respuestas de feedback para el conocimiento del canal, algoritmos y técnicas para mejorar y simular beamforming, entre otras características que no fueron parte de este proyecto.

- Para continuar con el estudio y ejecución de pruebas de mediciones de velocidades de transmisión de datos se recomienda utilizar nuevos APs que en la actualidad ya no necesitan de un módulo o accesorio adicional para operar con IEEE 802.11ac, sino que, ya tienen integrado el radio para este estándar.

BIBLIOGRAFÍA

[1] S. Luz, "Todo lo que debes saber sobre el nuevo estándar Wi-Fi." Marzo 2012 [Online]. Available: http://www.redeszone.net/2012/03/29/802-11ac-todo-lo-que-debes-saber-sobre-el-nuevo-estándar-wi-fi/. [Accessed 10 Diciembre 2015]

[2] S. Sinche, "Comunicaciones Inalámbricas.", Quito, 2013.

[3] O. Bejarano, E. Knightly, Minyoung Park, "IEEE 802.11ac: from channelization to multi-user MIMO." in Communications Magazine, IEEE , vol.51, no.10, pp.84-90, October 2013

[4] M. Gast, " Chapter 4," in 802.11ac: A Survival Guide, Editorial O´Reilly Media, 2013.

[5] "802.11ac In-Depth", 2010 [Online]. Available: http://www.arubanetworks.com/pdf/technology/whitepapers/WP_80211acInDepth.pdf 2010. [Accessed 08 Diciembre 2015]

[6] "Cisco 802.11ac Wave 2 FAQ," 2015 [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/802-11ac-solution/q-and-a-c67-734152.html. [Accessed 20 Noviembre 2015]

[7] D. Martinez, "Radiodifusión digital terrestre análisis del estándar DVB-T," [Online]. Available: http://www.scielo.org.ve/scielo.php?script=sci_arttext&pid=S1316-48212006000400003&lng=en&nrm=iso&tlng=es. [Accessed 19 Noviembre 2015]

[8] N. AL-Ghazu, "A Study of the Next WLAN Standard IEEE 802.11ac Physical Layer ", pp. 9 -30, Sweden, 2013.

[9] I. Poole, "MIMO Formats - SISO, SIMO, MISO, MU-MIMO," [Online]. Available: http://www.radio-electronics.com/info/antennas/mimo/formats-siso-simo-miso-mimo.php. [Accessed 15 Diciembre 2015]

[10] R. Hazra, "Multi User MIMO," Mayo 2012 [Online]. Available: http://technopits.blogspot.com/2012/05/dido-mu-mimo-is-it-same-or-is-it-not_03.html. [Accessed 02 Diciembre 2015]

[11] J. Bono, "Capítulo 3: Tecnología MIMO," [Online]. Available: http://www.academia.edu/13579593/G._Cap%C3%ADtulo_3_-_Tecnolog%C3%ADa_MIMO.

[12] M. Gast, "Chapter 2," in 802.11ac: A Survival Guide, Editorial O'Reilly Media, 2013.

[13] M. Gast, "Chapter 3," in 802.11ac: A Survival Guide, Editorial O´Reilly Media, 2013.

[14] QUALCOMM, "IEEE802.11ac: The Next Evolution of Wi-Fi," pp. 7, May 2012.

[15] V. Pinilla, "Boletín No. 29," Julio 2010 [Online]. Available:http://fmre.mx/boletinesx/2010/boletin1029.html. [Accessed 18 Noviembre 2015]

[16] I. Bernal, "Desvanecimiento", de Folleto Comunicaciones Inalámbricas, pp. 10-34, Quito, 2005.

[17] L. Fernandez, "Estudio y Simulación del canal Móvil para Bandas de LTE con distribuciones Rician y Rayleigh en el Modelo de Propagación OKUMURA-HATA en base a Matlab.", de Tesis, pp. 58-65.

[18] V. Laurent Schumacher, "TGn Channel Models," in IEEE Wireless LANs, pp. 5 - 41, 2004.

[19] H. Greg Breit, "TGac Channel Model Addendum," in IEEE Wireless LANS, pp. 5 - 15, 2010.

[20] O. Fernandez, "Correlación Espacial en canales MIMO," in Caracterización Experimental y Modelado de Canal MIMO para aplicaciones WLAN y WMAN, pp. 54, Santander, 2007,

[21] L. Cano, "Configuración y Evaluación de redes 802.11n," in Estudios de Ingeniería de Telecomunicaciónes, España, 2013.

[22] M. Velasco, R. López y D. Rojas, "Análisis de la correlación en un arreglo de antenas," México, 2014.

[23] R. Jaramillo y R. Torres, "Resultados Experimentales de Correlación Espacial de un Sistema MIMO en un Entorno Exterior-Interior A 2.4 Ghz," Santander.

[24] F. Networks, «White Paper: IEEE 802.11ac Migration Guide,» 2015.

[25] Matlab, "comm.OFDMModulator System Object," 2015 [Online]. Available: http://www.mathworks.com/help/comm/ref/comm.ofdmmodulator-class.html?searchHighlight=ofdmmodulator. [Accessed 10 Diciembre 2015]

[26] Matlab, "16-QAM with MATLAB Functions,» 2013 [Online]. Available: http://www.mathworks.com/help/comm/ug/digital-modulation.html. [Accessed 10 Diciembre 2015]

[27] Matlab, "comm.Mimochannel System Object," 2015. [Online]. Available: http://www.mathworks.com/help/comm/ref/comm.mimochannel-class.html. [Accessed 10 Diciembre 2015]

[28] Cisco, “Cisco Aironet 3600 Series Access Point Data Sheet,” 2013.

[29] Cisco, “Cisco Aironet Access Point Module for 802.11ac,” 2013.

[30] Chipset, “WiFi AC Adapter Dongle,” [Online]. Available: http://www.amazon.com/Adapter-Wireless-Dual-Band-802-11ac-Network/dp/B00LE3BJRA. [Accessed 1 Diciembre 2015]

[31] “AWUS036AC,” [Online]. Available: http://www.alfa.com.tw/products_show.php?pc=134&ps=208. [Accessed 1 Diciembre 2015]

[32] Cisco, “Configuring the Access Point for the First Time,” [Online]. Available: http://www.cisco.com/c/en/us/td/docs/wireless/access_point/15__4_JB/configuration/guide/scg15-2-4-Book/scg15-2-4-chap4-first.html#pgfId-1238489 [Accessed 2 Diciembre 2015]

[33] C. Systems, “Cisco IOS Configuration Guide for Autonomous Cisco Aironet Access Points,” de Configuring Radio Settings, pp. 13.

[34] “MCS: Index,” [Online]. Available: http://www.mcsindex.com/. [Accessed 15 Diciembre 2015]

[35] R. Telemáticas, “Velocidad de las redes WiFi N en entornos residenciales,” [Online]. Available: http://redestelematicas.com/velocidad-de-las-redes-wifi-n-en-entornos-residenciales/. [Accessed 20 Diciembre 2015]

ANEXOS

ANEXO A – MODELO A

Tablas con los valores de los *taps* para el Modelo A, con anchos de banda de canal de 20,40, 80 y 160 MHz.

Número de Tap	1
Retardo de Taps [ns]	0
Potencia [dBm]	0
AoA [°]	45
AS (Receptor) [°]	40
AoD [°]	45
AS (Transmisor) [°]	40

Tabla A.1 Modelo A - Valores para Ancho de Canal de 20, 40, 80 y 160 MHz.

ANEXO B – MODELO B

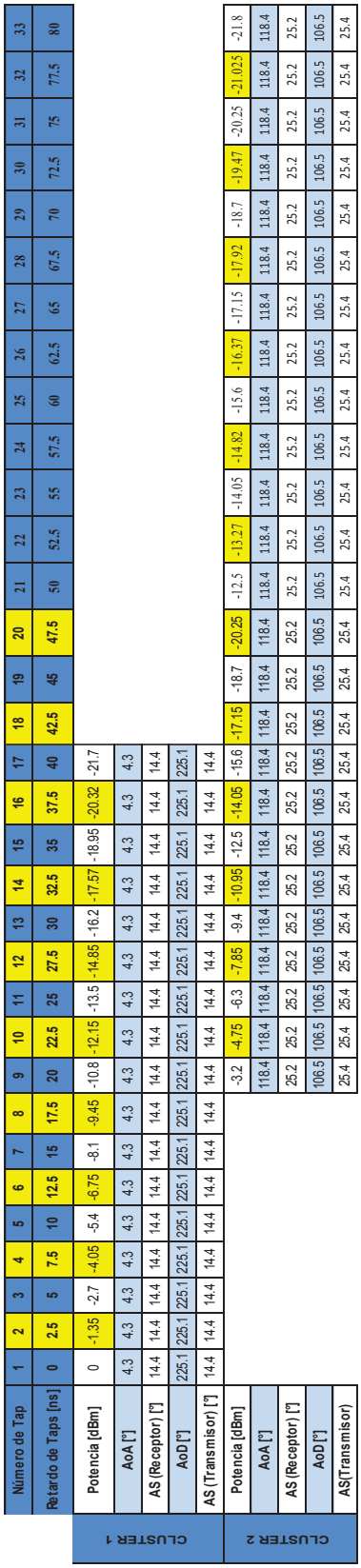
Tablas con los valores de los *taps* para el Modelo B, con anchos de banda de canal de 20,40, 80 y 160 MHz.

	Número de Tap	1	2	3	4	5	6	7	8	9
	Retardo de Taps [ns]	0	10	20	30	40	50	60	70	80
CLUSTER 1	Potencia [dBm]	0	-5.4	-10.8	-16.2	-21.7				
	AoA [°]	4.3	4.3	4.3	4.3	4.3				
	AS (Receptor) [°]	14.4	14.4	14.4	14.4	14.4				
	AoD [°]	225.1	225.1	225.1	225.1	225.1				
	AS (Transmisor) [°]	14.4	14.4	14.4	14.4	14.4				
CLUSTER 2	Potencia [dBm]			-3.2	-6.3	-9.4	-12.5	-15.6	-18.7	-21.8
	AoA [°]			118.4	118.4	118.4	118.4	118.4	118.4	118.4
	AS (Receptor) [°]			25.2	25.2	25.2	25.2	25.2	25.2	25.2
	AoD [°]			106.5	106.5	106.5	106.5	106.5	106.5	106.5
	AS (Transmisor) [°]			25.4	25.4	25.4	25.4	25.4	25.4	25.4

Tabla B.1 Modelo B - Valores para Ancho de Canal de 20 y 40 MHz.

	Número de Tap	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	Retardo de Taps [ns]	0	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80
CLUSTER 1	Potencia [dBm]	0	-2.7	-5.4	-8.1	-10.8	-13.5	-16.2	-18.95	-21.7								
	AoA [°]	4.3	4.3	4.3	4.3	4.3	4.3	4.3	4.3	4.3								
	AS (Receptor) [°]	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4								
	AoD [°]	225.1	225.1	225.1	225.1	225.1	225.1	225.1	225.1	225.1								
	AS (Transmisor) [°]	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4								
CLUSTER 2	Potencia [dBm]					-3.2	-4.75	-6.3	-7.85	-9.4	-10.95	-12.5	-14.05	-15.6	-17.15	-18.7	-20.25	-21.8
	AoA [°]					118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4
	AS (Receptor) [°]					25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2
	AoD [°]					106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5
	AS (Transmisor) [°]					25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4

Tabla B.2 Modelo B - Valores Interpolados Ancho de Canal 80 MHz.



	Número de Tap	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
	Retardo de Taps [ns]	0	2.5	5	7.5	10	12.5	15	17.5	20	22.5	25	27.5	30	32.5	35	37.5	40	42.5	45	47.5	50	52.5	55	57.5	60	62.5	65	67.5	70	72.5	75	77.5	80
CLUSTER 1	Potencia [dBm]	0	-1.35	-2.7	-4.05	-5.4	-6.75	-8.1	-9.45	-10.8	-12.15	-13.5	-14.85	-16.2	-17.57	-18.95	-20.32	-21.7																
	AoA [°]	4.3	4.3	4.3	4.3	4.3	4.3	4.3	4.3	4.3	4.3	4.3	4.3	4.3	4.3	4.3	4.3	4.3																
	AS (Receptor) [°]	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4																
	AoD [°]	225.1	225.1	225.1	225.1	225.1	225.1	225.1	225.1	225.1	225.1	225.1	225.1	225.1	225.1	225.1	225.1	225.1																
	AS (Transmisor) [°]	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4	14.4																
CLUSTER 2	Potencia [dBm]									-3.2	-4.75	-6.3	-7.85	-9.4	-10.95	-12.5	-14.05	-15.6	-17.15	-18.7	-20.25	-12.5	-13.27	-14.05	-14.82	-15.6	-16.37	-17.15	-17.92	-18.7	-19.47	-20.25	-21.025	-21.8
	AoA [°]									118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4	118.4
	AS (Receptor) [°]									25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2	25.2
	AoD [°]									106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5	106.5
	AS(Transmisor)									25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4	25.4

Tabla B.3 Modelo B - Valores Interpolados Ancho de Canal 160 MHz

ANEXO C – MODELO C

Tablas con los valores de los *taps* para el Modelo C, con anchos de banda de canal de 20,40, 80 y 160 MHz.

	Número de Tap	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	Retardo de Taps [ns]	**0**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**110**	**140**	**170**	**200**
CLUSTER 1	Potencia [dBm]	0	-2.1	-4.3	-6.5	-8.6	-10.8	-13.0	-15.2	-17.3	-19.5				
	AoA [°]	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3				
	AS (Receptor) [°]	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6				
	AoD [°]	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5				
	AS (Transmisor) [°]	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7				
CLUSTER 2	Potencia [dBm							-5.0	-7.2	-9.3	-11.5	-13.7	-15.8	-18.0	-20.2
	AoA [°]							332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3
	AS (Receptor) [°]							22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4
	AoD [°]							56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4
	AS (Transmisor)[°]							22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5

Tabla C.1 Modelo C – Ancho de Canal 20 y 40 MHz.

	Número de Tap	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
	Retardo de Taps [ns]	0	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	100	110	125	140	155	170	185	200
CLUSTER 1	Potencia [dBm]	0	-1.05	-2.1	-3,2	-4.3	-5,4	-6.5	-7.55	-8.6	-9.7	-10.8	-11.9	-13.0	-14.1	-15.2	-16.25	-17.3	-18.4	-19.5								
	AoA [°]	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3								
	AS (Receptor) [°]	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6								
	AoD [°]	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5								
	AS (Transmisor) [°]	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7								
CLUSTER 2	Potencia [dBm													-5.0	-6.1	-7.2	-8.25	-9.3	-10.4	-11.5	-12.6	-13.7	-14.75	-15.8	-16.9	-18.0	-19.1	-20.2
	AoA [°]													332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3
	AS (Receptor) [°]													22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4
	AoD [°]													56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4
	AS (Transmisor)[°]													22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5

Tabla C.2 Modelo C – Valores Interpolados Ancho de Canal 80 MHz

	Número de Tap	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
	Retardo de Taps [ns]	0	2,5	5	7,5	10	12,5	15	17,5	20	22,5	25	27,5	30	32,5	35	37,5	40	42,5	45	47,5	50	52,5	55	57,5
CLUSTER 1	Potencia [dBm]	0	-0.525	-1.05	-1.575	-2.1	-2.65	-3.2	-3.75	-4.3	-4.85	-5.4	-5.95	-6.5	-7.025	-7.55	-8.075	-8.6	-9.15	-9.7	-10.25	-10.8	-11.35	-11.9	-12.45
	AoA [°]	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3
	AS (Receptor) [°]	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6
	AoD [°]	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5
	AS (Transmisor) [°]	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7
CLUSTER 2	Potencia [dBm																								
	AoA [°]																								
	AS (Receptor) [°]																								
	AoD [°]																								
	AS (Transmisor)[°]																								

	Número de Tap	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53
	Retardo de Taps [ns]	60	62,5	65	67,5	70	72,5	75	77,5	80	82,5	85	87,5	90	95	100	105	110	117,5	125	132,5	140	147,5	155	162,5	170	177,5	185	192,5	200
CLUSTER 1	Potencia [dBm]	-13	-13.55	-14.1	-14.65	-15.2	-15.72	-16.25	-16.78	-17.3	-17.85	-18.4	-18.95	-19.5																
	AoA [°]	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3	290.3																
	AS (Receptor) [°]	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6	24.6																
	AoD [°]	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5																
	AS (Transmisor) [°]	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7	24.7																
CLUSTER 2	Potencia [dBm	-5.0	-5.55	-6.1	-6.65	-7.2	-7.725	8.25	-8.775	-9.3	-9.85	-10.4	-10.95	-11.5	-12.05	-12.6	-13.15	-13.7	-14.22	-14.75	-15.275	-15.8	-16.35	-16.9	-17.45	-18.0	-18.55	-19.1	-19.65	-20.2
	AoA [°]	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3	332.3
	AS (Receptor) [°]	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4	22.4
	AoD [°]	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4
	AS (Transmisor)[°]	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5

Tabla C-3 Modelo C – Valores Interpolados 160 MHz

ANEXO D - Correlación

En probabilidad y estadística, la correlación indica la fuerza y la dirección de una relación lineal y proporcionalidad entre dos variables estadísticas. Se considera que dos variables cuantitativas están correlacionadas cuando los valores de una de ellas varían sistemáticamente con respecto a los valores homónimos de la otra: si tenemos dos variables (A y B) existe correlación si al aumentar los valores de A lo hacen también los de B y viceversa. La correlación entre dos variables no implica, por sí misma, ninguna relación de causalidad

- **Coeficientes de correlación**

Existen diversos coeficientes que miden el grado de correlación, adaptados a la naturaleza de los datos. El más conocido es el coeficiente de correlación de Pearson(introducido en realidad por Francis Galton), que se obtiene dividiendo la covarianza de dos variables entre el producto de sus desviaciones estándar. Otros coeficientes son:

- **Interpretación geométrica**

Dados los valores muestrales de dos variables aleatorias $X(x_1, \dots, x_n)$ e$Y(y_1, \dots., y_n)$, que pueden ser consideradas como vectores en un espacio a n dimensiones, pueden construirse los "vectores centrados" como:

$$X(x_1 - \bar{x}, \dots, x_n - \bar{x})\ e\ Y(y_1 - \bar{y}, \dots, y_n - \bar{y})$$

El coseno del ángulo alfa entre estos vectores es dada por la fórmula siguiente:

$$\cos(\alpha) = \frac{\sum_{i=1}^{N}(x_i - \bar{x}).(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}.\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}}$$

Pues $\cos(\alpha)$ es el coeficiente de correlación muestral de Pearson. El coeficiente de correlación es el coseno entre ambos vectores centrados:
Si r = 1, el ángulo $\alpha = 0°$, ambos vectores son colineales (paralelos).
Si r = 0, el ángulo $\alpha = 90°$, ambos vectores son ortogonales.
Si r =-1, el ángulo $\alpha = 180°$, ambos vectores son colineales de dirección opuesto.

Por supuesto, desde el punto vista geométrico, no hablamos de correlación lineal: el coeficiente de correlación tiene siempre un sentido, cualquiera sea su valor entre -1 y 1. Nos informa de modo preciso, no tanto sobre el grado de dependencia entre las variables, sino sobre su distancia angular en la hiperesfera a n dimensiones.
La Iconografía de las correlaciones es un método de análisis multidimensional que reposa en esta idea. La correlación lineal se da cuando en una nube de puntos se encuentran o se distribuyen alrededor de una recta.

La fórmula de correlación para dos series distintas con cierto desfase "k", está dada por la fórmula:

$$r_k = \frac{\sum_{i=1}^{N-k}(x_i - \bar{x}).(y_{i+k} - \bar{y})}{\sqrt{\sum_{i=1}^{N-k}(x_i - \bar{x})^2} . \sqrt{\sum_{i=k+1}^{N}(y_i - \bar{y})^2}}$$

ANEXO E – Código fuente de la simulación

Función Principal

```
%Función Principal: Se elige el modelo de canal y se llama a las funciones del
programa.

function [Resultado] = Principal(modelo)
% Elección del modelo (A,B,C).
tic
close all
clear all
clc

[Rs] = Parametros();

global modelo
modelo = input ('Ingrese el modelo a seleccionar:  ','s'); % Ingresa el modelo (A
- C).
m=isletter(modelo);


% Verifica el ingreso correcto del dato (Modelo A,B o C).
while m<1
        fprintf('Error - Se puede ingresar los Modelos A, B o C  \n\n');
        modelo = input ('Ingrese el modelo a seleccionar:  ','s');
        m=isletter(modelo);
end

while m>0
switch modelo
    case 'A'                                    % Datos del caso A.
        clc
        fprintf('Elección de Modelo A.  \n\n');
        [x] = Usuarios();                       % Llamada a la función Usuarios.
        [TX] = Antenas_Tx_Rx();                 % Llamada a la función Antenas_Tx_Rx.
        [AoA_ModeloA,AoD_ModeloA] = Angulos_Modelo_A(); %Llamada a la función.
.Ángulos_Modelo_A
        [Valor_MCS] = MCS();                    % Llamada a la función MCS.
        [intervalo] = Intervalo_Guarda();       % Llamada a la función Intervalo_Guarda.
        [SNR] = Valor_SNR();                    % Llamada a la función Valor_SNR.
        [BW] = BW_Modelo_A();                   % Llamada a la función BW_Modelo_A.
        [Y_MIMO_AWGN_1] = AWGN();               % Llamada a la función AWGN.
        [Usuario_graf] = Resultado_Gráficas();  % Llamada a la función Resultado_Graficas.
        [Data_rate] = Resultados();             % Llamada a la función Resultados.
        [datos] = Texto_Datos();                % Llamada a la función Texto_datos.
        toc
        break

    case 'B'                                    % Datos del Caso B.
        clc
        fprintf('Elección de Modelo B.  \n\n');
        [x] = Usuarios();                       % Llamada a la función Usuarios.
        [TX] = Antenas_Tx_Rx();                 % Llamada a la función Antenas_Tx_Rx.
        [valor_MCS] = MCS();                    % Llamada a la función MCS.
        [intervalo] = Intervalo_Guarda();       % Llamada a la función Intervalo_Guarda.
        [SNR] = Valor_SNR();                    % Llamada a la función Valor_SNR.
        [BW] = BW_Modelo_B();                   % Llamada a la función BW_Modelo_B.
        [Y_MIMO_AWGN_1] = AWGN();               % Llamada a la función AWGN.
        [Usuario_graf] = Resultado_Gráficas();  % Llamada a la función Resultado_Gráficas.
        [Data_rate] = Resultados();             % Llamada a la función Resultados.
        [datos] = Texto_Datos();                % Llamada a la función Texto_datos.
        toc
        break

    case 'C'                                        % Datos del Caso C.
        clc
        fprintf('Elección de Modelo C.  \n\n');
```

```
        [x] = Usuarios();                       % Llamada a la función Usuarios.
        [TX] = Antenas_Tx_Rx();                 % Llamada a la función Antenas_Tx_Rx.
        [valor_MCS] = MCS();                    % Llamada a la función MCS.
        [intervalo] = Intervalo_Guarda();       % Llamada a la función Intervalo_Guarda.
        [SNR] = Valor_SNR();                    % Llamada a la función Valor_SNR.
        [BW] = BW_Modelo_C();                   % Llamada a la función BW_Modelo_C.
        [Y_MIMO_AWGN_1] = AWGN();               % Llamada a la función AWGN.
        [Usuario_graf] = Resultado_Gráficas();  % Llamada a la función Resultados_Gráficas.
        [Data_rate] = Resultados();             % Llamada a la función Resultados.
        [datos] = Texto_Datos();                % Llamada a la función Texto_datos.
        toc

        break


    % Mensaje de error, reingrerso de datos.
    otherwise
        fprintf('Error - Se puede ingresar los Modelos A, B o C  \n\n');
        modelo = input ('Ingrese el modelo a seleccionar:  ','s');

        end
    end
end
```

Función Usuarios

```
%Función Usuarios: Determina la cantidad de usuarios para el sistema.

function [n] = Usuarios(n)

% Variable Global
global x

n = input ('Ingrese el número de usuarios: ','s');  % Ingresa el número de clientes.
valor=isletter(n);                                  % Determino si el valor ingresado es
letra.

while valor > 0
      [Error_Letra] = Errores();                    % Mensaje de error.
      n = input ('Ingrese el número de usuarios: ','s');
      valor=isletter(n);
end

while valor < 1
x=str2double(n);

    if x>0 & x<5                    % Entrega del número de usuarios si esta en el rango.
(1-4)
        sprintf('La cantidad de usuarios es: %i \n\n',x);
        break
    % Mensaje de error si el número de usuarios no corresponden al determinado por el
estándar.
    else
       fprintf('No válido, para el estándar 802.11ac se define un número máximo de 4
usuarios para el uso de MU-MIMO.');
    end
    n = input ('Ingrese el número de usuarios: ','s');

    end
end
```

Función Antenas_Tx_Rx

```
%Función Antenas_Tx_Rx: Selección de la cantidad de receptoras por usuario y el número
total de antenas transmisoras para el punto de acceso.

function [TX] = Antenas_Tx_Rx(valor)
```

```
% Variables Globales
global TX
global RX_1, global RX_2, global RX_3, global RX_4
global x

switch x


    % Ingreso de Datos del caso con 1 usuario.
    case 1
        %Ingreso del número de antenas del usuario.
        RX_1= input ('\nIngrese el número de antenas del primer usuario: ');

        % Mensaje de error si tiene mas de 3 antenas.
        while RX_1>3
           disp ('Error - El usuario debe tener un máximo de 3 antenas.')
           RX_1= input ('\nIngrese el número de antenas del primer usuario: ');
        end
           % Determina la cantidad de antenas que tiene el transmisor.
           TX=RX_1;

 % Ingreso de datos del caso con 2 usuarios.
    case 2
        % Ingreso del número de antenas para cada usuario.
        RX_1 = input ('\nIngrese el número de antenas del primer usuario: ');
        RX_2 = input ('Ingrese el número de antenas del segundo usuario: ');
        % Mensaje de error del usuario 1 si tiene mas de 3 antenas.
        while RX_1>3
           disp ('Error - El usuario 1 debe tener un máximo de 3 antenas.')
           RX_1= input ('\nIngrese el número de antenas del primer usuario: ');
        end
        % Mensaje de error del usuario 2 si tiene mas de 3 antenas.
        while RX_2>3
           disp ('Error - El usuario 2 debe tener un máximo de 3 antenas.');
           RX_2= input ('\nIngrese el número de antenas del segundo usuario: ');
        end
           % Determina la cantidad de antenas que tiene el Transmisor.
           TX=RX_1+RX_2;

    % Ingreso de datos del caso con 3 usuarios.
    case 3
        % Ingreso del número de antenas para cada usuario.
        RX_1 = input ('\nIngrese el número de antenas del primer usuario: ');
        RX_2 = input ('Ingrese el número de antenas del segundo usuario: ');
        RX_3 = input ('Ingrese el número de antenas del tercer usuario: ');
        % Mensaje de error del usuario 1 si tiene mas de 3 antenas.
        while RX_1>3
           disp ('Error - El usuario 1 debe tener un máximo de 3 antenas.');
           RX_1= input ('\nIngrese el número de antenas del primer usuario: ');
        end
        % Mensaje de error del usuario 2 si tiene mas de 3 antenas.
        while RX_2>3
           disp ('Error - El usuario 2 debe tener un máximo de 3 antenas.');
           RX_2= input ('\nIngrese el número de antenas del segundo usuario: ');
        end
        % Mensaje de error del usuario 3 si tiene mas de 3 antenas.
        while RX_3>3
           disp ('Error - El usuario 3 debe tener un máximo de 3 antenas.');
           RX_3= input ('\nIngrese el número de antenas del tercer usuario: ');
        end
           % Determina la cantidad de antenas que tiene el transmisor.
           TX = RX_1 + RX_2 + RX_3;


       % Condición para que las antenas del transmisor no sea mayor a 8.
         while TX>8
           disp ('Error - La máxima cantidad de antenas de todos los usuarios es 8');
           RX_1 = input ('\nIngrese el número de antenas del primer usuario: ');
           RX_2 = input ('Ingrese el número de antenas del segundo usuario: ');
           RX_3 = input ('Ingrese el número de antenas del tercer usuario: ');
           TX = RX_1 + RX_2 + RX_3;
         end


    % Ingreso de datos del caso con 4 usuarios.
    case 4
```

```
        RX_1 = input ('\nIngrese el número de antenas del primer usuario: ');
        RX_2 = input ('Ingrese el número de antenas del segundo usuario: ');
        RX_3 = input ('Ingrese el número de antenas del tercer usuario: ');
        RX_4 = input ('Ingrese el número de antenas del cuarto usuario: ');
        % Mensaje de error del usuario 1 si tiene mas de 3 antenas.
        while RX_1>3
           disp ('Error - El usuario 1 debe tener un máximo de 3 antenas.');
           RX_1= input ('\nIngrese el número de antenas del primer usuario: ');
        end
        % Mensaje de error del usuario 2 si tiene mas de 3 antenas.
        while RX_2>3
           disp ('Error - El usuario 2 debe tener un máximo de 3 antenas.');
           RX_2= input ('\nIngrese el número de antenas del segundo usuario: ');
        end
        % Mensaje de error del usuario 3 si tiene mas de 3 antenas.
        while RX_3>3
           disp ('Error - El usuario 3 debe tener un máximo de 3 antenas.');
           RX_3= input ('\nIngrese el número de antenas del tercer usuario: ');
        end
        % Mensaje de error del usuario 4 si tiene mas de 3 antenas.
         while RX_4>3
           disp ('Error - El usuario 4 debe tener un máximo de 3 antenas.');
           RX_4= input ('\nIngrese el número de antenas del cuarto usuario: ');
         end
        % Determina la cantidad de antenas que tiene el transmisor
           TX = RX_1 + RX_2 + RX_3 + RX_4;


         % Condición para que las antenas del transmisor no sea mayor a 8.
         while TX>8
           disp ('Error - La máxima cantidad de antenas de todos los usuarios es 8');
           RX_1 = input ('\nIngrese el número de antenas del primer usuario: ');
           RX_2 = input ('Ingrese el número de antenas del segundo usuario: ');
           RX_3 = input ('Ingrese el número de antenas del tercer usuario: ');
           RX_4 = input ('Ingrese el número de antenas del cuarto usuario: ');
           TX = RX_1 + RX_2 + RX_3 + RX_4;
        end
    end
end
```

Función MCS

```
%Función MCS: Permite seleccionar el valor de indice MCS (Esquema de Modulación y
Codificación).

function [Valor_MCS] = MCS(index)

% Variables Globales.
global Valor_MCS
global Mod
global Coding
global m
global Nbpscs

% Ingresa el valor de índice de modulación (MCS).
indice = input ('\n Ingrese el valor de MCS (Esquema de Modulación y Codificación):
','s');
%Determino si el valor ingresado es letra.
MCS=isletter(indice);

% Mensaje de error al ingresar el valor de MCS.
while MCS>0
      [Error_Letra] = Errores();
      indice= input ('\nIngrese el valor de MCS (Esquema de Modulación y Codificación):
','s');
      MCS=isletter(indice);
end

while MCS<1

if MCS == 0                          %Sentencia if para determinar el valor de MCS.
Valor_MCS=str2double(indice);        %Se convierte de carácter a número.
```

```
    if Valor_MCS>=0 & Valor_MCS<10

switch Valor_MCS

    % Caso 1: Índice MCS=0
    case 0
        disp('El valor de Modulación es BPSK.')
        disp('El valor de Codificación es 1/2.')
        Mod = 2;                % Valor de Modulación.
        m='BPSK';               % Tipo de Modulación.
        Nbpscs=1;               % Número de bits codificados por subportadora.
        Coding = 1/2;           % Valor de Codificación.
        break

    % Caso 2: Índice MCS=1
    case 1
        disp('El valor de Modulación es QPSK.')
        disp('El valor de Codificación es 1/2.')
        Mod = 4;                % Valor de Modulación.
        m='QPSK';               % Tipo de Modulación.
        Nbpscs=2;               % Número de bits codificados por subportadora.
        Coding = 1/2;           % Valor de Codificación.
        break

    % Caso 3: Índice MCS=2
    case 2
        disp('El valor de Modulación es QPSK.')
        disp('El valor de Codificación es 3/4.')
        Mod = 4;                % Valor de Modulación.
        m='QPSK';               % Tipo de Modulación.
        Nbpscs=2;               % Número de bits codificados por subportadora.
        Coding = 3/4;           % Valor de Codificación.
        break

    % Caso 4: Índice MCS=3
    case 3
        disp('El valor de Modulación es 16 QAM.')
        disp('El valor de Codificación es 1/2.')
        Mod = 16;               % Valor de Modulación.
        m='16 QAM';             % Tipo de Modulación.
        Nbpscs=4;               % Número de bits codificados por subportadora.
        Coding = 1/2;           % Valor de Codificación.
        break

    % Caso 5: Índice MCS=4
    case 4
        disp('El valor de Modulación es 16 QAM.')
        disp('El valor de Codificación es 3/4.')
        Mod = 16;               % Valor de Modulación.
        m='16 QAM';             % Tipo de Modulación.
        Nbpscs=4;               % Número de bits codificados por subportadora.
        Coding = 3/4;           % Valor de Codificación.
        break

    % Caso 6: Índice MCS=5
    case 5
        disp('El valor de Modulación es 64 QAM.')
        disp('El valor de Codificación es 2/3.')
        Mod = 64;               % Valor de Modulación.
        m='64 QAM';             % Tipo de Modulación.
        Nbpscs=6;               % Número de bits codificados por subportadora.
        Coding = 2/3;           % Valor de Codificación.
        break

    % Caso 7: Índice MCS=6
    case 6
        disp('El valor de Modulación es 64 QAM.')
        disp('El valor de Codificación es 3/4.')
        Mod = 64;               % Valor de Modulación.
        m='64 QAM';             % Tipo de Modulación.
        Nbpscs=6;               % Número de bits codificados por subportadora.
        Coding = 3/4;           % Valor de Codificación.
        break
```

```
    % Caso 8: Índice MCS=7
    case 7
       disp('El valor de Modulación es 64 QAM.')
       disp('El valor de Codificación es 5/6.')
       Mod = 64;               % Valor de Modulación.
       m='64 QAM';             % Tipo de Modulación.
       Nbpscs=6;               % Número de bits codificados por subportadora.
       Coding = 5/6;           % Valor de Codificación.
       break

    % Caso 9: Índice MCS=8
    case 8
       disp('El valor de Modulación es 256 QAM.')
       disp('El valor de Codificación es 3/4.')
       Mod = 256;               % Valor de Modulación.
       m='256 QAM';             % Tipo de Modulación.
       Nbpscs=8;                % Número de bits codificados por subportadora.
       Coding = 3/4;            % Valor de Codificación.
       break

    % Caso 10: Índice MCS=9
    case 9
       disp('El valor de Modulación es 256 QAM.')
       disp('El valor de Codificación es 5/6.')
       Mod = 256;               % Valor de Modulación.
       m='256 QAM';             % Tipo de Modulación.
       Nbpscs=8;                % Número de bits codificados por subportadora.
       Coding = 5/6;            % Valor de Codificación.
       break
end
    % Mensaje de error si los valores ingresados son erróneos.
    else
    display ('Los valores que se pueden ingresar son de 0 a 9');
    indice= input ('\nIngrese el valor de MCS (Esquema de Modulación y Codificación):
','s');
    end
end
end
end
```

Función intervalo de Guarda

```
%Función Interalo_Guarda: Permite seleccionar el valor de intervalo de guarda.

function [intervalo] = Intervalo_Guarda(inter_guarda)
% Variables Globales
global ig
global inter_guarda

%Ingresa el valor de intervalo de guarda (400 y 800).
intervalo = input ('\nIngrese el intervalo de guarda (400,800)ns: ','s');
alfa=isletter(intervalo);

% Mensaje de error
while alfa > 0
      [Error_Letra] = Errores();
      intervalo = input ('\nIngrese el intervalo de guarda (400ns - 800ns): ','s');
      alfa=isletter(intervalo);

end
% Casos posibles del intervalo de guarda.
while alfa < 1
inter_guarda=str2double(intervalo);

        % Caso 1 Intervalo=400
         if inter_guarda==400
            ig = 1.111112;      % Valor a usar en el cálculo de Data_Rate.
            break


         % Caso 1 Intervalo=800
```

```
        else if inter_guarda==800
            ig = 1;            % Valor a usar en el cálculo de Data_Rate.
            break

        % Mensaje de error
        else
                disp('El valor de intervalo de guarda ingresado es incorrecto')

        end
        intervalo = input ('\nIngrese el intervalo de guarda (400ns - 800ns): ','s');
        end
    end
end
```

Función Valor_SNR

```
%Función SNR: Se ingresa el valor de SNR.

function [SNR] = Valor_SNR(valor)
% Variable Global.
global SNR

% Ingreso del SNR.
snr = input ('\nIngrese el valor de SNR en dB: ','s');
valor=isletter(snr);

% Mensaje de error.
while valor > 0
      [Error_Letra] = Errores();
      snr = input ('\nIngrese el valor de SNR en dB: ','s');
      valor=isletter(snr);
end

% Verificación de rangos.
while valor < 1
SNR=str2double(snr);
    % Verificación del rango de SNR(entre 0 y 50).
    if SNR>0 & SNR<50
      sprintf('El valor de SNR es: %i \n\n',SNR);
        break
    % Mensaje de error si esta fuera de rango.
    else
       fprintf('No válido, para la simulación se debe ingresar un valor entre 0 y 50 dB');
    end
    snr = input ('\nIngrese el valor de SNR en dB: ','s');
end
```

Función BW_Modelo_A

```
%Función BW_ModeloA: Determina el ancho de canal a seleccionar.

function [BW] = BW_ModeloA(LOS)
% Variables Globales
global BW
global Nsd

% Ingreso del valor de ancho de banda.
BW_A = input ('\nIngrese el valor de ancho de canal (20, 40, 80, 160 MHz): ','s');
valor=isletter(BW_A);
% Mensaje de error a datos ingresados.
while valor > 0
      [Error_Letra] = Errores();
      BW_A = input ('\nIngrese el valor de ancho de canal (20, 40, 80, 160 MHz): ','s');
      valor=isletter(BW_A);
end

while valor < 1
BW=str2double(BW_A);
switch BW
     % Caso 1 BW=20 MHz
```

```
     case 20
        [OFDM_1] = OFDM_20();                       % Llamada función OFDM_20.
        % Llamada función Matriz_Correlacion_Modelo_A.
        [TxCorrelationMatriz_1_1] = Matriz_Correlacion_Modelo_A();
        [H_Modelo_A_1] = Modelo_A_MimoChannel();    % Llamada función Modelo_A_MimoChannel.
        [Datos_RX_1] = Generar_Datos();             % Llamada función Generar_Datos.
        [Datos_Mod_1] = Modulación();               % Llamada función Modulación.
        [Y_MIMO_1] = OFDM_MIMO();                   % Llamada función OFDM_MIMO.
      break
    % Caso 1 BW=40 MHz
    case 40

        [OFDM_1] = OFDM_40();                       % Llamada función OFDM_40.
        % Llamada función Matriz_Correlacion_Modelo_A.
        [TxCorrelationMatriz_1_1] = Matriz_Correlacion_Modelo_A();
        [H_Modelo_A_1] = Modelo_A_MimoChannel();    % Llamada función Modelo_A_MimoChannel.
        [Datos_RX_1] = Generar_Datos();             % Llamada función Generar_Datos.
        [Datos_Mod_1] = Modulación();               % Llamada función Modulación.
        [Y_MIMO_1] = OFDM_MIMO();                   % Llamada función OFDM_MIMO.

       break
    % Caso 1 BW=80 MHz
    case 80

        [OFDM_1] = OFDM_80();                       % Llamada función OFDM_80.
        % Llamada función Matriz_Correlacion_Modelo_A.
        [TxCorrelationMatriz_1_1] = Matriz_Correlacion_Modelo_A();
        [H_Modelo_A_1] = Modelo_A_MimoChannel();    % Llamada función Modelo_A_MimoChannel.
        [Datos_RX_1] = Generar_Datos();             % Llamada función Generar_Datos.
        [Datos_Mod_1] = Modulación();               % Llamada función Modulación.
        [Y_MIMO_1] = OFDM_MIMO();                   % Llamada función OFDM_MIMO.
       break
    % Caso 1 BW=160 MHz
    case 160

        [OFDM_1] = OFDM_160();                      % Llamada func. OFDM_160.
        % Llamada función Matriz_Correlacion_Modelo_A.
        [TxCorrelationMatriz_1_1] = Matriz_Correlacion_Modelo_A();
        [H_Modelo_A_1] = Modelo_A_MimoChannel();    % Llamada función Modelo_A_MimoChannel.
        [Datos_RX_1] = Generar_Datos();             % Llamada función Generar_Datos.
        [Datos_Mod_1] = Modulación();               % Llamada función Modulación.
        [Y_MIMO_1] = OFDM_MIMO();                   % Llamada función OFDM_MIMO.

       break
    % Mensaje de error si BW esta fuera de rango.
    otherwise
        disp('El valor de ancho de canal es incorrecto')
        BW_A = input ('\nIngrese el valor de ancho de canal (20, 40, 80, 160 MHz): ','s');
    end
end
```

Función BW_Modelo_B

```
%Función BW_ModeloB: Determina el ancho de canal a seleccionar.

function [BW] = BW_Modelo_B(NLOS)
% Variables Globales
global BW
global Nsd

% Ingreso del valor de ancho de banda.
BW_B = input ('\nIngrese el valor de ancho de canal (20, 40, 80, 160 MHz): ','s');
valor=isletter(BW_B);
% Mensaje de error a datos ingresados.
while valor > 0
      [Error_Letra] = Errores();
      BW_B = input ('\nIngrese el valor de ancho de canal (20, 40, 80, 160 MHz): ','s');
      valor=isletter(BW_B);
end
```

```
while valor < 1
BW=str2double(BW_B);
switch BW
    % Caso 1 BW=20 MHz
    case 20

        [OFDM_1] = OFDM_20();                       % Llamada función OFDM_20.
        % Llamada función Ángulos_Modelo_B_20MHz_C1
        [AoA_Modelo_B_20_C1] = Angulos_Modelo_B_20MHz_C1();
        % Llamada función Ángulos_Modelo_B_20MHz_C1
        [AoA_Modelo_B_20_C2] = Angulos_Modelo_B_20MHz_C2();
        % Llamada función Matriz_Correlación_Modelo_B_20
        [TxCorrelationMatriz_1] = Matriz_Correlacion_Modelo_B_20();
        [H_Modelo_B_1] = Modelo_B_MimoChannel();   % Llamada función Modelo_B_MimoChannel.
        [Datos_RX_1] = Generar_Datos();            % Llamada función Generar_Datos.
        [Datos_Mod_1] = Modulación();              % Llamada función Modulación.
        [Y_MIMO_1] = OFDM_MIMO();                  % Llamada función OFDM_MIMO.
        break
    % Caso 1 BW=40 MHz
    case 40

        [OFDM_1] = OFDM_40();                       % Llamada función OFDM_40.
        % Llamada función Angulos_Modelo_B_40MHz_C1
        [AoA_Modelo_B_20_C1] = Angulos_Modelo_B_40MHz_C1();
        % Llamada función Angulos_Modelo_B_40MHz_C1
        [AoA_Modelo_B_20_C2] = Angulos_Modelo_B_40MHz_C2();
        % Llamada función Matriz_Correlacion_Modelo_B_40
        [TxCorrelationMatriz_1] = Matriz_Correlacion_Modelo_B_40();
        [H_Modelo_B_1] = Modelo_B_MimoChannel();   % Llamada función Modelo_B_MimoChannel.
        [Datos_RX_1] = Generar_Datos();            % Llamada función Generar_Datos.
        [Datos_Mod_1] = Modulación();              % Llamada función Modulación.
        [Y_MIMO_1] = OFDM_MIMO();                  % Llamada función OFDM_MIMO.

        break
    % Caso 1 BW=80 MHz
    case 80

        [OFDM_1] = OFDM_80();                       % Llamada función OFDM_80.
        % Llamada función Angulos_Modelo_B_80MHz_C1
        [AoA_Modelo_B_20_C1] = Angulos_Modelo_B_80MHz_C1();
        % Llamada función Angulos_Modelo_B_80MHz_C1
        [AoA_Modelo_B_20_C2] = Angulos_Modelo_B_80MHz_C2();
        % Llamada función Matriz_Correlacion_Modelo_B_80
        [TxCorrelationMatriz_1] = Matriz_Correlacion_Modelo_B_80();
        [H_Modelo_B_1] = Modelo_B_MimoChannel();   % Llamada función Modelo_B_MimoChannel.
        [Datos_RX_1] = Generar_Datos();            % Llamada función Generar_Datos.
        [Datos_Mod_1] = Modulación();              % Llamada función Modulación.
        [Y_MIMO_1] = OFDM_MIMO();                  % Llamada función OFDM_MIMO.
        break
    % Caso 1 BW=160 MHz
    case 160

        [OFDM_1] = OFDM_160();                       % Llamada func. OFDM_160
        % Llamada función Angulos_Modelo_B_160MHz_C1
        [AoA_Modelo_B_20_C1] = Angulos_Modelo_B_160MHz_C1();
        % Llamada función Angulos_Modelo_B_160MHz_C1
        [AoA_Modelo_B_20_C2] = Angulos_Modelo_B_160MHz_C2();
        % Llamada función Matriz_Correlacion_Modelo_B_160
        [TxCorrelationMatriz_1] = Matriz_Correlacion_Modelo_B_160();
        [H_Modelo_B_1] = Modelo_B_MimoChannel();   % Llamada función Modelo_B_MimoChannel.
        [Datos_RX_1] = Generar_Datos();            % Llamada función Generar_Datos.
        [Datos_Mod_1] = Modulación();              % Llamada función Modulación.
        [Y_MIMO_1] = OFDM_MIMO();                  % Llamada función OFDM_MIMO.
        break

    otherwise
        disp('El valor de ancho de canal es incorrecto')
        BW_B = input ('\nIngrese el valor de ancho de canal (20, 40, 80, 160 MHz): ','s');

end

end
```

Función BW_Modelo_C

```
%Función BW_ModeloC: Determina el ancho de canal a seleccionar.

function [BW] = BW_Modelo_C(NLOS)
% Variables Globales
global BW
global Nsd

% Ingreso del valor de ancho de banda.
BW_C = input ('\nIngrese el valor de ancho de canal (20, 40, 80, 160 MHz): ','s');
valor=isletter(BW_C);
% Mensaje de error a datos ingresados.
while valor > 0
      [Error_Letra] = Errores();
      BW_C = input ('\nIngrese el valor de ancho de canal (20, 40, 80, 160 MHz): ','s');
      valor=isletter(BW_C);
end

while valor < 1
BW=str2double(BW_C);
switch BW
    % Caso 1 BW=20 MHz
    case 20

        [OFDM_1] = OFDM_20();                       % Llamada función OFDM_20.
        % Llamada función Angulos_Modelo_C_20MHz_C1
        [AoA_Modelo_C_20_C1] = Angulos_Modelo_C_20MHz_C1();
        % Llamada función Angulos_Modelo_C_20MHz_C1
        [AoA_Modelo_C_20_C2] = Angulos_Modelo_C_20MHz_C2();
        % Llamada función Matriz_Correlacion_Modelo_C_20
        [TxCorrelationMatriz_1] = Matriz_Correlacion_Modelo_C_20();
        [H_Modelo_C_1] = Modelo_C_MimoChannel();   % Llamada función Modelo_C_MimoChannel.
        [Datos_RX_1] = Generar_Datos();            % Llamada función Generar_Datos.
        [Datos_Mod_1] = Modulación();              % Llamada función Modulación.
        [Y_MIMO_1] = OFDM_MIMO();                  % Llamada función OFDM_MIMO.
        break
    % Caso 1 BW=40 MHz
    case 40

        [OFDM_1] = OFDM_40();                       % Llamada función OFDM_40
        % Llamada función Angulos_Modelo_C_40MHz_C1
        [AoA_Modelo_C_20_C1] = Angulos_Modelo_C_40MHz_C1();
        % Llamada función Angulos_Modelo_C_40MHz_C1
        [AoA_Modelo_C_20_C2] = Angulos_Modelo_C_40MHz_C2();
        % Llamada función Matriz_Correlacion_Modelo_C_40
        [TxCorrelationMatriz_1] = Matriz_Correlacion_Modelo_C_40();
        [H_Modelo_C_1] = Modelo_C_MimoChannel();   % Llamada función Modelo_C_MimoChannel.
        [Datos_RX_1] = Generar_Datos();            % Llamada función Generar_Datos.
        [Datos_Mod_1] = Modulación();              % Llamada función Modulación.
        [Y_MIMO_1] = OFDM_MIMO();                  % Llamada función OFDM_MIMO.

        break
    % Caso 1 BW=80 MHz
    case 80

        [OFDM_1] = OFDM_80();                       % Llamada función. OFDM_80
        % Llamada función Angulos_Modelo_C_80MHz_C1
        [AoA_Modelo_C_20_C1] = Angulos_Modelo_C_80MHz_C1();
        % Llamada función Angulos_Modelo_C_80MHz_C1
        [AoA_Modelo_C_20_C2] = Angulos_Modelo_C_80MHz_C2();
        % Llamada función Matriz_Correlacion_Modelo_C_80
        [TxCorrelationMatriz_1] = Matriz_Correlacion_Modelo_C_80();
        [H_Modelo_C_1] = Modelo_C_MimoChannel();   % Llamada función Modelo_C_MimoChannel.
        [Datos_RX_1] = Generar_Datos();            % Llamada función Generar_Datos.
        [Datos_Mod_1] = Modulación();              % Llamada función Modulación.
        [Y_MIMO_1] = OFDM_MIMO();                  % Llamada función OFDM_MIMO.
        break
    % Caso 1 BW=160 MHz
    case 160

        [OFDM_1] = OFDM_160();                       % Llamada función OFDM_160
        % Llamada función Angulos_Modelo_C_160MHz_C1
        [AoA_Modelo_C_20_C1] = Angulos_Modelo_C_160MHz_C1();
```

```
        % Llamada función Angulos_Modelo_C_160MHz_C1
        [AoA_Modelo_C_20_C2] = Angulos_Modelo_C_160MHz_C2();
        % Llamada función Matriz_Correlacion_Modelo_C_160
        [TxCorrelationMatriz_1] = Matriz_Correlacion_Modelo_C_160();
        [H_Modelo_C_1] = Modelo_C_MimoChannel();   % Llamada función Modelo_C_MimoChannel.
        [Datos_RX_1] = Generar_Datos();            % Llamada función Generar_Datos.
        [Datos_Mod_1] = Modulación();              % Llamada función Modulación.
        [Y_MIMO_1] = OFDM_MIMO();                  % Llamada función OFDM_MIMO.
        break
    % Mensaje de error si BW esta fuera de rango.
    otherwise
        disp('El valor de ancho de canal es incorrecto')
        BW_B = input ('\nIngrese el valor de ancho de canal (20, 40, 80, 160 MHz): ','s');

end

end
```

Función Generación_Ángulos_Offset

Función Generación_Ángulos_Offset_A

```
%Función Generación_Angulos_Offset_A: Se genera los ángulos offset según la cantidad de
usuarios.

function [Angulos_Offset_A] = Generacion_Angulos_Offset_A(t)

%Determina el valor de ángulo OFFSET que se va aplicar a cada AoD para poder tener la
diversidad de usuarios para un ambiente LOS.
global x

generar_valores_AoD = 2803;                        % Valor definido en Matlab.
rango_OFFSET = 360;                                % Especifica el rango del ángulo AoD (+/-
180°).

rand('seed',generar_valores_AoD);                  % Generación aleatorio de valores.
valor_OFFSET = [(rand(x,1)-0.5)*rango_OFFSET];     % Operación para generar ángulo OFFSET.
angulos_NLOS_AoD = valor_OFFSET';                  % Cambio de orientación del vector.

%Determina el valor de ángulo OFFSET que se va aplicar a cada AoA para poder tener la
diversidad de usuarios en un ambiente LOS.

generar_valores_AoA = 13367;                       % Valor definido en Matlab.
rango_OFFSET = 360;                                % Especifica el rango del ángulo AoA (+/-
180°).

rand('seed',generar_valores_AoA);                  % Generación aleatorio de valores.
valor_OFFSET = [(rand(x,1)-0.5)*rango_OFFSET];     % Operación para generar ángulo OFFSET.
angulos_NLOS_AoA = valor_OFFSET';                  % Cambio de orientación del vector.

Angulos_Offset_A(1,:) = ángulos_NLOS_AoA;          % Formo la matriz Ángulos_Offset con los
valores de AoA.
Angulos_Offset_A(2,:) = ángulos_NLOS_AoD;          % Formo la matriz Ángulos_Offset con los
valores de AoD.

Angle_Offset=size(Angulos_Offset_A);               % Tamaño de la matriz Ángulos_Offset_A.
fila = Angle_Offset(1,1);
columna = Angle_Offset(1,2);

% Cambio de los valores negativos a un valor posiitvo.
for i=1:fila
    for j=1:columna
        if Angulos_Offset_A(i,j)<0
        Angulos_Offset_A(i,j)=360+Angulos_Offset_A(i,j);
        end
    end
    j=1;
end
end
```

Función Generación_Ángulos_Offset_B

```
function [Angulos_Offset_B] = Generacion_Angulos_Offset_B(t)

%Determina el valor de ángulo OFFSET que se va aplicar a cada AoD para poder tener la
diversidad de usuarios para un ambiente NLOS.
global x

generar_valores_AoD = 39161;                              % Valor definido en Matlab.
rango_OFFSET = 360;                                       % Especifica el rango del ángulo
AoD (+/-180°).

rand('seed',generar_valores_AoD);                         % Generación randomico de valores.
valor_OFFSET = [(rand(x,1)-0.5)*rango_OFFSET];            % Operación para generar ángulo
OFFSET.
angulos_LOS_AoD = valor_OFFSET';                          % Cambio de orientación del vector.

%Determina el valor de ángulo OFFSET que se va aplicar a cada AoA para poder tener la
diversidad de usuarios para un ambiente NLOS.

generar_valores_AoA = 45191;                              % Valor definido en Matlab.
rango_OFFSET = 360;                                       % Especifica el rango del ángulo
AoA (+/-180°).

rand('seed',generar_valores_AoA);                         % Generación aleatoria de valores.
valor_OFFSET = [(rand(x,1)-0.5)*rango_OFFSET];            % Operación para generar ángulo
OFFSET.
angulos_LOS_AoA = valor_OFFSET';                          % Cambio de orientación del vector.

Angulos_Offset_B(1,:) = angulos_LOS_AoA;                  % Formo la matriz Angulos_Offset.
con los valores de AoA.
Angulos_Offset_B(2,:) = angulos_LOS_AoD;                  % Formo la matriz Angulos_Offset.
con los valores de AoD.

Angle_Offset=size(Angulos_Offset_B);                      % Tamaño de la matriz
Angulos_Offset_B.
fila = Angle_Offset(1,1);
columna = Angle_Offset(1,2);

% Cambio de los valores negativos a un valor posiitvo.
for i=1:fila
    for j=1:columna
        if Angulos_Offset_B(i,j)<0
        Angulos_Offset_B(i,j)=360+Angulos_Offset_B(i,j);
        end
    end
    j=1;
end
end
```

Función Generación_Ángulos_Offset_C

```
function [Angulos_Offset_C] = Generacion_Angulos_Offset_C(t)

%Determina el valor de ángulo OFFSET que se va aplicar a cada AoD para poder tener la
diversidad de usuarios para un ambiente NLOS.
global x

generar_valores_AoD = 2803;                               % Valor definido en Matlab.
rango_OFFSET = 360;                                       % Especifica el rango del ángulo AoD
(+/-180°).

rand('seed',generar_valores_AoD);                         % Generación randomico de valores
valor_OFFSET = [(rand(x,1)-0.5)*rango_OFFSET];            % Operación para generar ángulo
OFFSET.
angulos_NLOS_AoD = valor_OFFSET';                         % Cambio de orientación del vector

%Determina el valor de ángulo OFFSET que se va aplicar a cada AoA para poder tener la
diversidad de usuarios para un ambiente NLOS.

generar_valores_AoA = 13367;                              % Valor definido en Matlab.
rango_OFFSET = 360;                                       % Especifica el rango del ángulo AoA
(+/-180°).
```

```
rand('seed',generar_valores_AoA);                       % Generación aleatoria de valores.
valor_OFFSET = [(rand(x,1)-0.5)*rango_OFFSET];          % Operación para generar ángulo
OFFSET.
angulos_NLOS_AoA = valor_OFFSET';                       % Cambio de orientación del vector.

Angulos_Offset_C(1,:) = angulos_NLOS_AoA;               % Formo la matriz Angulos_Offset con
los valores de AoA.
Angulos_Offset_C(2,:) = angulos_NLOS_AoD;               % Formo la matriz Angulos_Offset con
los valores de AoD.

Angle_Offset=size(Angulos_Offset_C);                    % Tamaño de la matriz
Angulos_Offset_B
fila = Angle_Offset(1,1);
columna = Angle_Offset(1,2);

% Cambio de los valores negativos a un valor posiitvo.
for i=1:fila
    for j=1:columna
        if Angulos_Offset_C(i,j)<0
        Angulos_Offset_C(i,j)=360+Angulos_Offset_C(i,j);
        end
    end
    j=1;
end
end
```

Ángulos_Modelo

Función Ángulo_Modelo_A

```
%Función Angulos_ModeloA: Sirve para adicionar el ángulo offset al ángulo AoA y ángulo AoD
del modelo A para un ancho de canal de 40, 80 y 160 MHz.

function [AoA_ModeloA,AoD_ModeloA] = Angulos_Modelo_A(usuarios)
% Variables Globales
global AoA_ModeloA
global AoD_ModeloA
global x

[Angulos_Offset_A]= Generacion_Angulos_Offset_A;    % Se llama al resultado de angulos
generados.

AoA = [45];                                         % Ángulos definidos para 802.11ac
Modelo A.
AoD = [45];                                         % Ángulos definidos para 802.11ac
Modelo A.

%Los ángulos OFFSET se suman a los ángulos del estándar para cada usuario.
    if x==1;
        AoA_ModeloA = [45];                         % Ángulo AoA para un solo usuario.
        AoD_ModeloA = [45];                         % Ángulo AoD para un solo usuario.

    else  x>1;
        for i=1:x;
            for j=1:size(AoA,2);
                AoA_ModeloA(i,j)=AoA(j)+ Angulos_Offset_A(1,i); %Matriz ángulos AoA +
Offset.
                AoD_ModeloA(i,j)=AoD(j)+ Angulos_Offset_A(2,i); %Matriz ángulos AoD +
Offset.
            end
        end
    end
end
```

Función Ángulo_Modelo_B

Ángulos_Modelo_B_20MHz_C1

```
%Función Angulos_Modelo_B_20MHz_C1: Se adiciona el ángulo offset al ángulo AoA y ángulo AoD
del modelo B para un ancho de canal de 20MHz para el cluster 1.
```

```
function [AoA_Modelo_B_20_C1,AoD_Modelo_B_20_C1] = Angulos_Modelo_B_20MHz_C1(usuarios)
% Variables Globales
global AoA_Modelo_B_20_C1
global AoD_Modelo_B_20_C1
global x
% Se llama al resultado de ángulos generados.
[Angulos_Offset_B]= Generacion_Angulos_Offset_B;
% Ángulos definidos para 802.11ac Modelo B Cluster 1 20MHz.
AoA_20_C1 = [4.3 4.3 4.3 4.3 4.3 -inf -inf -inf -inf];
% Ángulos definidos para 802.11ac Modelo B Cluster 1 20MHz.
AoD_20_C1 = [225.1 225.1 225.1 225.1 225.1 -inf -inf -inf -inf];

%Los ángulos OFFSET se suman a los ángulos del estándar para cada usuario.
    if x==1;
        % Ángulo AoA para un solo usuario.
        AoA_Modelo_B_20_C1 = [4.3 4.3 4.3 4.3 4.3 -inf -inf -inf -inf];
        % Ángulo AoD para un solo usuario.
        AoD_Modelo_B_20_C1 = [225.1 225.1 225.1 225.1 225.1 -inf -inf -inf -inf];

    else  x>1;
        for i=1:x;
            for j=1:size(AoA_20_C1,2);
                % Matriz ángulos AoA + Offset.
                AoA_Modelo_B_20_C1(i,j)=AoA_20_C1(j)+ Angulos_Offset_B(1,i);
                % Matriz ángulos AoD + Offset.
                AoD_Modelo_B_20_C1(i,j)=AoD_20_C1(j)+ Angulos_Offset_B(2,i);
            end
        end
    end
end
```

Angulos_Modelo_B_20MHz_C2

```
%Función Angulos_Modelo_B_20MHz_C2: Se adiciona el ángulo offset al ángulo AoA y ángulo AoD
del modelo B para un ancho de canal de 20MHz para el Cluster 2.

function [AoA_Modelo_B_20_C2] = Angulos_Modelo_B_20MHz_C2(Usuarios)
% Variables Globales
global AoA_Modelo_B_20_C2
global AoD_Modelo_B_20_C2
global x
% Función de números de usuarios.
[Angulos_Offset_B]= Generacion_Angulos_Offset_B;
% Ángulos definidos para 802.11ac Modelo B Cluster 2 20MHz.
AoA_20_C2 = [-inf -inf 118.4 118.4 118.4 118.4 118.4 118.4 118.4];
% Ángulos definidos para 802.11ac Modelo B Cluster 2 20MHz.
AoD_20_C2 = [-inf -inf 106.5 106.5 106.5 106.5 106.5 106.5 106.5];

% Los ángulos OFFSET se suman a los ángulos del estándar para cada usuario.
    if x==1;
        % Ángulo AoA para un solo usuario.
        AoA_Modelo_B_20_C2 = [-inf -inf 118.4 118.4 118.4 118.4 118.4 118.4 118.4];
        % Ángulo AoD para un solo usuario.
        AoD_Modelo_B_20_C2 = [-inf -inf 106.5 106.5 106.5 106.5 106.5 106.5 106.5];

    else x>1;
        for i=1:x;
            for j=1:size(AoA_20_C2,2);
                % Matriz ángulos AoA + Offset.
                AoA_Modelo_B_20_C2(i,j)=AoA_20_C2(j)+ Angulos_Offset_B(1,i);
                % Matriz ángulos AoD + Offset.
                AoD_Modelo_B_20_C2(i,j)=AoD_20_C2(j)+ Angulos_Offset_B(2,i);
            end
        end
    end
end
```

Angulos_Modelo_B_40MHz_C1

```
function [AoA_Modelo_B_40_C1,AoD_Modelo_B_40_C1] = Angulos_Modelo_B_40MHz_C1(usuarios)
%Función para adicionar el ángulo offset al ángulo AoA y ángulo AoD del modelo B para un
ancho de canal de 40MHz para el Cluster 1.
```

```
% Variables Globales
global AoA_Modelo_B_40_C1
global AoD_Modelo_B_40_C1
global x
% Se llama al resultado de ángulos generados.
[Angulos_Offset_B]= Generacion_Angulos_Offset_B;
% Ángulos definidos para 802.11ac Modelo B Cluster 1 40MHz.
AoA_40_C1 = [4.3 4.3 4.3 4.3 4.3 -inf -inf -inf -inf];
% Ángulos definidos para 802.11ac Modelo B Cluster 1 40MHz.
AoD_40_C1 = [225.1 225.1 225.1 225.1 225.1 -inf -inf -inf -inf];

% Los ángulos OFFSET se suman a los ángulos del estándar para cada usuario.
    if x==1;
        % Ángulo AoA para un solo usuario.
        AoA_Modelo_B_40_C1 = [4.3 4.3 4.3 4.3 4.3 -inf -inf -inf -inf];
        % Ángulo AoD para un solo usuario.
        AoD_Modelo_B_40_C1 = [225.1 225.1 225.1 225.1 225.1 -inf -inf -inf -inf];

    else  x>1;
        for i=1:x;
            for j=1:size(AoA_40_C1,2);
                 % Matriz ángulos AoA + Offset.
                AoA_Modelo_B_40_C1(i,j)=AoA_40_C1(j)+ Angulos_Offset_B(1,i);
                 % Matriz ángulos AoD + Offset.
                AoD_Modelo_B_40_C1(i,j)=AoD_40_C1(j)+ Angulos_Offset_B(2,i);
            end
        end
    end
end
```

Angulos_Modelo_B_40MHz_C2

```
%Función para adicionar el ángulo offset al ángulo AoA y ángulo AoD del modelo B para un
ancho de canal de 40MHz para el Cluster 2.

function [AoA_Modelo_B_40_C2,AoD_Modelo_B_40_C2] = Angulos_Modelo_B_40MHz_C2(usuarios)
% Variables Globales
global AoA_Modelo_B_40_C2
global AoD_Modelo_B_40_C2
global x
% Se llama al resultado de angulos generados.
[Angulos_Offset_B]= Generacion_Angulos_Offset_B;
% Ángulos definidos para 802.11ac Modelo B Cluster 2 40MHz.
AoA_40_C2 = [-inf -inf 118.4 118.4 118.4 118.4 118.4 118.4 118.4];
% Ángulos definidos para 802.11ac Modelo B Cluster 2 40MHz.

AoD_40_C2 = [-inf -inf 106.5 106.5 106.5 106.5 106.5 106.5 106.5];

%Los ángulos OFFSET se suman a los ángulos del estándar para cada usuario.
    if x==1;
        % Ángulo AoA para un solo usuario.
        AoA_Modelo_B_40_C2 = [-inf -inf 118.4 118.4 118.4 118.4 118.4 118.4 118.4];
        % Ángulo AoD para un solo usuario.
        AoD_Modelo_B_40_C2 = [-inf -inf 106.5 106.5 106.5 106.5 106.5 106.5 106.5];
    else x>1;
        for i=1:x;
            for j=1:size(AoA_40_C2,2);
                % Matriz ángulos AoA + Offset.
                AoA_Modelo_B_40_C2(i,j)=AoA_40_C2(j)+ Angulos_Offset_B(1,i);
                % Matriz ángulos AoD + Offset.
                AoD_Modelo_B_40_C2(i,j)=AoD_40_C2(j)+ Angulos_Offset_B(2,i);
            end
        end
    end
end
```

Angulos_Modelo_B_80MHz_C1

```
%Función Angulos_Modelo_B_80MHz: Adiciona ángulo offset al ángulo AoA y ángulo AoD del
modelo B para un ancho de canal de 80MHz para el Cluster 1.

function [AoA_Modelo_B_80_C1] = Angulos_Modelo_B_80MHz_C1(usuarios)
```

```
% Variables Globales
global AoA_Modelo_B_80_C1
global AoD_Modelo_B_80_C1
global x
% Se llama al resultado de angulos generados.
[Angulos_Offset_B]= Generacion_Angulos_Offset_B;
% Ángulos definidos para 802.11ac Modelo B Cluster 1 80MHz.
AoA_80_C1 = [4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 -inf -inf -inf -inf -inf -inf -inf -inf];
% Ángulos definidos para 802.11ac Modelo B Cluster 1 80MHz.
AoD_80_C1 = [225.1 225.1 225.1 225.1 225.1 225.1 225.1 225.1 225.1 -inf -inf -inf -inf -inf
-inf -inf -inf];

% Los ángulos OFFSET se suman a los ángulos del estándar para cada usuario.
    if x==1;
        % Ángulo AoA para un solo usuario.
        AoA_Modelo_B_80_C1 = [4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 -inf -inf -inf -inf -inf
-inf -inf -inf];
        % Ángulo AoD para un solo usuario.
        AoD_Modelo_B_80_C1 = [225.1 225.1 225.1 225.1 225.1 225.1 225.1 225.1 225.1 -inf -
inf -inf -inf -inf -inf -inf -inf];

    else x>1;
        for i=1:x;
            for j=1:size(AoA_80_C1,2);
                % Matriz ángulos AoA + Offset.
                AoA_Modelo_B_80_C1(i,j)=AoA_80_C1(j)+ Angulos_Offset_B(1,i);
                % Matriz ángulos AoD + Offset.
                AoD_Modelo_B_80_C1(i,j)=AoD_80_C1(j)+ Angulos_Offset_B(2,i);
            end
        end
    end
end
```

Angulos_Modelo_B_80MHz_C2

```
%Función Angulos_Modelo_B_80MHz: Adiciona ángulo offset al ángulo AoA y ángulo AoD del
modelo B para un ancho de canal de 80MHz para el Cluster 2.

function [AoA_Modelo_B_80_C2,AoD_Modelo_B_80_C2] = Angulos_Modelo_B_80MHz_C2(usuarios)
% Variables Globales
global AoA_Modelo_B_80_C2
global AoD_Modelo_B_80_C2
global x
% Se llama al resultado de angulos generados.
[Angulos_Offset_B]= Generacion_Angulos_Offset_B;
% Ángulos definidos para 802.11ac Modelo B Cluster 2 80MHz.
AoA_80_C2 = [-inf -inf -inf -inf 118.4 118.4 118.4 118.4 118.4 118.4 118.4 118.4 118.4
118.4 118.4 118.4 118.4];
% Ángulos definidos para 802.11ac Modelo B Cluster 2 80MHz.
AoD_80_C2 = [-inf -inf -inf -inf 106.5 106.5 106.5 106.5 106.5 106.5 106.5 106.5 106.5
106.5 106.5 106.5 106.5];

%Los ángulos OFFSET se suman a los ángulos del estándar para cada usuario.

    if x==1;
        % Ángulo AoA para un solo usuario.
        AoA_Modelo_B_80_C2 = [-inf -inf -inf -inf 118.4 118.4 118.4 118.4 118.4 118.4 118.4
118.4 118.4 118.4 118.4 118.4 118.4];
        % Ángulo AoD para un solo usuario.
        AoD_Modelo_B_80_C2 = [-inf -inf -inf -inf 106.5 106.5 106.5 106.5 106.5 106.5 106.5
106.5 106.5 106.5 106.5 106.5 106.5];

    else x>1;
        for i=1:x;
            for j=1:size(AoA_80_C2,2);
                % Matriz ángulos AoA + Offset.
                AoA_Modelo_B_80_C2(i,j)=AoA_80_C2(j)+ Angulos_Offset_B(1,i);
                % Matriz ángulos AoD + Offset.
                AoD_Modelo_B_80_C2(i,j)=AoD_80_C2(j)+ Angulos_Offset_B(2,i);
            end
        end
    end
end
```

Angulos_Modelo_B_160MHz_C1

```
%Función Angulos_Modelo_B_160MHz: Adiciona ángulo offset al ángulo AoA y ángulo AoD del
modelo B para un ancho de canal de 160MHz para el Cluster 1.

function [AoA_Modelo_B_160_C1,AoD_Modelo_B_160_C1] = Angulos_Modelo_B_160MHz_C1(Usuarios)
% Variables Globales
global AoA_Modelo_B_160_C1
global AoD_Modelo_B_160_C1
global x
% Se llama al resultado de ángulos generados.
[Angulos_Offset_B]= Generacion_Angulos_Offset_B;
% Ángulos definidos para 802.11ac Modelo B Cluster 1 160MHz.
AoA_160_C1 = [4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 -inf -inf
-inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf];
% Ángulos definidos para 802.11ac Modelo B Cluster 1 160MHz.
AoD_160_C1 = [225.1 225.1 225.1 225.1 225.1 225.1 225.1 225.1 225.1 225.1 225.1 225.1 225.1
225.1 225.1 225.1 -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -
inf -inf -inf];

% Los ángulos OFFSET se suman a los ángulos del estándar para cada usuario.

    if x==1;
        % Ángulo AoA para un solo usuario.
        AoA_Modelo_B_160_C1 = [4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3
4.3 4.3 -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf];
        % Ángulo AoD para un solo usuario.
        AoD_Modelo_B_160_C1 = [225.1 225.1 225.1 225.1 225.1 225.1 225.1 225.1 225.1 225.1
225.1 225.1 225.1 225.1 225.1 225.1 -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf
-inf -inf -inf -inf -inf -inf];

    else x>1;
        for i=1:x;
            for j=1:size(AoA_160_C1,2);
                % Matriz ángulos AoA + Offset.
                AoA_Modelo_B_160_C1(i,j)=AoA_160_C1(j)+ Angulos_Offset_B(1,i);
                % Matriz ángulos AoD + Offset.
                AoD_Modelo_B_160_C1(i,j)=AoD_160_C1(j)+ Angulos_Offset_B(2,i);
            end
        end
    end
end
```

Angulos_Modelo_B_160MHz_C2

```
%Función Angulos_Modelo_B_160MHz: Adiciona ángulo offset al ángulo AoA y ángulo AoD del
modelo B para un ancho de canal de 80MHz para el Cluster 2.

function [AoA_Modelo_B_160_C2,AoD_Modelo_B_160_C2] = Angulos_Modelo_B_160MHz_C2(usuarios)
% Variables globales
global AoA_Modelo_B_160_C2
global AoD_Modelo_B_160_C2
global x
% Se llama al resultado de ángulos generados.
[Angulos_Offset_B]= Generacion_Angulos_Offset_B;
% Ángulos definidos para 802.11ac Modelo B Cluster 2 80MHz.
AoA_160_C2 = [-inf -inf -inf -inf -inf -inf -inf -inf 118.4 118.4 118.4 118.4 118.4
118.4 118.4 118.4 118.4 118.4 118.4 118.4 118.4 118.4 118.4 118.4 118.4 118.4 118.4
118.4 118.4 118.4 118.4];
% Ángulos definidos para 802.11ac Modelo B Cluster 2 80MHz.
AoD_160_C2 = [-inf -inf -inf -inf -inf -inf -inf -inf 106.5 106.5 106.5 106.5 106.5 106.5
106.5 106.5 106.5 106.5 106.5 106.5 106.5 106.5 106.5 106.5 106.5 106.5 106.5
106.5 106.5 106.5 106.5];

% Los ángulos OFFSET se suman a los ángulos del estándar para cada usuario.
    if x==1;
        % Ángulo AoA para un solo usuario.
        AoA_Modelo_B_160_C2 = [-inf -inf -inf -inf -inf -inf -inf -inf 118.4 118.4 118.4
118.4 118.4 118.4 118.4 118.4 118.4 118.4 118.4 118.4 118.4 118.4 118.4 118.4 118.4
118.4 118.4 118.4 118.4 118.4 118.4 118.4];
        % Ángulo AoD para un solo usuario.
        AoD_Modelo_B_160_C2 = [-inf -inf -inf -inf -inf -inf -inf -inf 106.5 106.5 106.5
106.5 106.5 106.5 106.5 106.5 106.5 106.5 106.5 106.5 106.5 106.5 106.5 106.5 106.5
```

```
106.5 106.5 106.5 106.5 106.5 106.5 106.5];

    else x>1;
        for i=1:x;
            for j=1:size(AoA_160_C2,2);
                % Matriz ángulos AoA + Offset.
                AoA_Modelo_B_160_C2(i,j)=AoA_160_C2(j)+ Angulos_Offset_B(1,i);
                % Matriz ángulos AoD + Offset.
                AoD_Modelo_B_160_C2(i,j)=AoD_160_C2(j)+ Angulos_Offset_B(2,i);
            end
        end
    end
end
```

Función Ángulo_Modelo_C

Angulos_Modelo_C_20MHz_C1

```
%Función para adicionar el ángulo offset al ángulo AoA y ángulo AoD del modelo C para un
ancho de canal de 20MHz para el Cluster 1.

function [AoA_Modelo_C_20_C1,AoD_Modelo_C_20_C1] = Angulos_Modelo_C_20MHz_C1(usuarios)
% Variables globales
global AoA_Modelo_C_20_C1
global AoD_Modelo_C_20_C1
global x
%Se llama al resultado de ángulos generados.
[Angulos_Offset_C]= Generacion_Angulos_Offset_C;
%Ángulos definidos para 802.11ac Modelo C Cluster 1 20MHz.
AoA_20_C1 = [290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 -inf -inf -inf -
inf];
%Ángulos definidos para 802.11ac Modelo C Cluster 1 20MHz.
AoD_20_C1 = [13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 -inf -inf -inf -inf];

%Los ángulos OFFSET se suman a los ángulos del estándar para cada usuario.

    if x==1;
        %Ángulo AoA para un solo usuario.
        AoA_Modelo_C_20_C1 = [290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 -
inf -inf -inf -inf];
        %Ángulo AoD para un solo usuario.
        AoD_Modelo_C_20_C1 = [13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 -inf -inf -
inf -inf];

    else  x>1;
        for i=1:x;
            for j=1:size(AoA_20_C1,2);
                %Matriz ángulos AoA + Offset.
                AoA_Modelo_C_20_C1(i,j)=AoA_20_C1(j)+ Angulos_Offset_C(1,i);
                %Matriz ángulos AoD + Offset.
                AoD_Modelo_C_20_C1(i,j)=AoD_20_C1(j)+ Angulos_Offset_C(2,i);
            end
        end
    end
end
```

Angulos_Modelo_C_20MHz_C2

```
%Función para adicionar el ángulo offset al ángulo AoA y ángulo AoD del modelo C para un
ancho de canal de 40MHz para el Cluster 2.

function [AoA_Modelo_C_20_C2,AoD_Modelo_C_20_C2] = Angulos_Modelo_C_20MHz_C2(usuarios)
% Variables globales
global AoA_Modelo_C_20_C2
global AoD_Modelo_C_20_C2
global x
% Se llama al resultado de ángulos generados.
[Angulos_Offset_C]= Generacion_Angulos_Offset_C;
% Ángulos definidos para 802.11ac Modelo C Cluster 2 20MHz.
AoA_20_C2 = [-inf -inf -inf -inf -inf -inf 332.3 332.3 332.3 332.3 332.3 332.3 332.3
332.3];
```

```
% Ángulos definidos para 802.11ac Modelo C Cluster 2 20MHz.
AoD_20_C2 = [-inf -inf -inf -inf -inf -inf 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4];

%Los ángulos OFFSET se suman a los ángulos del estándar para cada usuario.
    if x==1;
        % Ángulo AoA para un solo usuario.
        AoA_Modelo_C_20_C2 = [-inf -inf -inf -inf -inf -inf 332.3 332.3 332.3 332.3 332.3
332.3 332.3 332.3];
        % Ángulo AoD para un solo usuario.
        AoD_Modelo_C_20_C2 = [-inf -inf -inf -inf -inf -inf 56.4 56.4 56.4 56.4 56.4 56.4
56.4 56.4];

    else  x>1;
        for i=1:x;
            for j=1:size(AoA_20_C2,2);
                % Matriz ángulos AoA + Offset.
                AoA_Modelo_C_20_C2(i,j)=AoA_20_C2(j)+ Angulos_Offset_C(1,i);
                % Matriz ángulos AoD + Offset.
                AoD_Modelo_C_20_C2(i,j)=AoD_20_C2(j)+ Angulos_Offset_C(2,i);
            end
        end
    end
end
```

Angulos_Modelo_C_40MHz_C1

```
%Función para adicionar el ángulo offset al ángulo AoA y ángulo AoD del modelo C para un
ancho de canal de 40MHz para el Cluster 1.

function [AoA_Modelo_C_40_C1,AoD_Modelo_C_40_C1] = Angulos_Modelo_C_40MHz_C1(usuarios)
% Variables globales
global AoA_Modelo_C_40_C1
global AoD_Modelo_C_40_C1
global x
% Se llama al resultado de ángulos generados.
[Angulos_Offset_C]= Generacion_Angulos_Offset_C;
% Ángulos definidos para 802.11ac Modelo C Cluster 1 40MHz.
AoA_40_C1 = [290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 -inf -inf -inf -
inf];
% Ángulos definidos para 802.11ac Modelo C Cluster 1 40MHz.
AoD_40_C1 = [13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 -inf -inf -inf -inf];

%Los ángulos OFFSET se suman a los ángulos del estándar para cada usuario.

    if x==1;
        % Ángulo AoA para un solo usuario.
        AoA_Modelo_C_40_C1 = [290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 -
inf -inf -inf -inf];
        % Ángulo AoD para un solo usuario.
        AoD_Modelo_C_40_C1 = [13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 -inf -inf -
inf -inf];

    else  x>1;
        for i=1:x;
            for j=1:size(AoA_40_C1,2);
                % Matriz ángulos AoA + Offset.
                AoA_Modelo_C_40_C1(i,j)=AoA_40_C1(j)+ Angulos_Offset_C(1,i);
                % Matriz ángulos AoD + Offset.
                AoD_Modelo_C_40_C1(i,j)=AoD_40_C1(j)+ Angulos_Offset_C(2,i);
            end
        end
    end
end
```

Angulos_Modelo_C_40MHz_C2

```
%Función para adicionar el ángulo offset al ángulo AoA y ángulo AoD del modelo C para un
ancho de canal de 40MHz para el Cluster 2.
function [AoA_Modelo_C_40_C2,AoD_Modelo_C_40_C2] = Angulos_Modelo_C_40MHz_C2(usuarios)
% Variables globales
global AoA_Modelo_C_40_C2
```

```
global AoD_Modelo_C_40_C2
global x
% Se llama al resultado de angulos generados.
[Angulos_Offset_C]= Generacion_Angulos_Offset_C;
% Ángulos definidos para 802.11ac Modelo C Cluster 2 40MHz.
AoA_40_C2 = [-inf -inf -inf -inf -inf -inf 332.3 332.3 332.3 332.3 332.3 332.3 332.3
332.3];
% Ángulos definidos para 802.11ac Modelo C Cluster 2 40MHz.
AoD_40_C2 = [-inf -inf -inf -inf -inf -inf 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4];

% Los ángulos OFFSET se suman a los ángulos del estándar para cada usuario.

    if x==1;
        % Ángulo AoA para un solo usuario.
        AoA_Modelo_C_40_C2 = [-inf -inf -inf -inf -inf -inf 332.3 332.3 332.3 332.3 332.3
332.3 332.3 332.3];
        % Ángulo AoD para un solo usuario.
        AoD_Modelo_C_40_C2 = [-inf -inf -inf -inf -inf -inf 56.4 56.4 56.4 56.4 56.4 56.4
56.4 56.4];

    else  x>1;
        for i=1:x;
            for j=1:size(AoA_40_C2,2);
                % Matriz ángulos AoA + Offset.
                AoA_Modelo_C_40_C2(i,j)=AoA_40_C2(j)+ Angulos_Offset_C(1,i);
                % Matriz ángulos AoD + Offset.
                AoD_Modelo_C_40_C2(i,j)=AoD_40_C2(j)+ Angulos_Offset_C(2,i);
            end
        end
    end
end
```

Angulos_Modelo_C_80MHz_C1

```
function [AoA_Modelo_C_80_C1,AoD_Modelo_C_80_C1] = Angulos_Modelo_C_80MHz_C1(usuarios)
%Función para adicionar el ángulo offset al ángulo AoA y ángulo AoD del modelo C para un
ancho de canal de 80MHz para el Cluster 1.
% Variables globales
global AoA_Modelo_C_80_C1
global AoD_Modelo_C_80_C1
global x
% Se llama al resultado de ángulos generados.
[Angulos_Offset_C]= Generacion_Angulos_Offset_C;
% Ángulos definidos para 802.11ac Modelo C Cluster 1 80MHz.
AoA_80_C1 = [290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3
290.3 290.3 290.3 290.3 290.3 -inf -inf -inf -inf -inf -inf -inf];
% Ángulos definidos para 802.11ac Modelo C Cluster 1 80MHz.
AoD_80_C1 = [13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5
13.5 13.5 13.5 13.5 -inf -inf -inf -inf -inf -inf -inf -inf];

%Los ángulos OFFSET se suman a los ángulos del estándar para cada usuario.

    if x==1;
        % Ángulo AoA para un solo usuario.
        AoA_Modelo_C_80_C1 = [290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3
290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 -inf -inf -inf -inf -inf -inf -inf -
inf];
        % Ángulo AoD para un solo usuario.
        AoD_Modelo_C_80_C1 = [13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5
13.5 13.5 13.5 13.5 13.5 13.5 -inf -inf -inf -inf -inf -inf -inf -inf];

    else  x>1;
        for i=1:x;
            for j=1:size(AoA_80_C1,2);
                % Matriz ángulos AoA + Offset.
                AoA_Modelo_C_80_C1(i,j)=AoA_80_C1(j)+ Angulos_Offset_C(1,i);
                % Matriz ángulos AoD + Offset.
                AoD_Modelo_C_80_C1(i,j)=AoD_80_C1(j)+ Angulos_Offset_C(2,i);
            end
        end
    end
end
```

Angulos_Modelo_C_80MHz_C2

```
%Función para adicionar el ángulo offset al ángulo AoA y ángulo AoD del modelo C para un
ancho de canal de 80MHz para el Cluster 2.
function [AoA_Modelo_C_80_C2,AoD_Modelo_C_80_C2] = Angulos_Modelo_C_80MHz_C2(usuarios)
% Variables globales
global AoA_Modelo_C_80_C2
global AoD_Modelo_C_80_C2
global x
% Se llama al resultado de ángulos generados.
[Angulos_Offset_C]= Generacion_Angulos_Offset_C;
% Ángulos definidos para 802.11ac Modelo C Cluster 2 80MHz.
AoA_80_C2 = [-inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf 332.3 332.3 332.3
332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3];
% Ángulos definidos para 802.11ac Modelo C Cluster 2 80MHz.
AoD_80_C2 = [-inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf 56.4 56.4 56.4
56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4];

% Los ángulos OFFSET se suman a los ángulos del estándar para cada usuario.
    if x==1;
        % Ángulo AoA para un solo usuario.
        AoA_Modelo_C_80_C2 = [-inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf
332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3];
        % Ángulo AoD para un solo usuario.
        AoD_Modelo_C_80_C2 = [-inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf
56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4];

    else  x>1;
        for i=1:x;
            for j=1:size(AoA_80_C2,2);
                % Matriz ángulos AoA + Offset.
                AoA_Modelo_C_80_C2(i,j)=AoA_80_C2(j)+ Angulos_Offset_C(1,i);
                % Matriz ángulos AoD + Offset.
                AoD_Modelo_C_80_C2(i,j)=AoD_80_C2(j)+ Angulos_Offset_C(2,i);
            end
        end
    end
end
```

Angulos_Modelo_C_160MHz_C1

```
function [AoA_Modelo_C_160_C1,AoD_Modelo_C_160_C1] = Angulos_Modelo_C_160MHz_C1(usuarios)
%Función para adicionar el ángulo offset al ángulo AoA y ángulo AoD del modelo C para un
ancho de canal de 160MHz para el Cluster 1.
% Variables globales
global AoA_Modelo_C_160_C1
global AoD_Modelo_C_160_C1
global x
% Se llama al resultado de ángulos generados.
[Angulos_Offset_C]= Generacion_Angulos_Offset_C;
% Ángulos definidos para 802.11ac Modelo C Cluster 1 160MHz.
AoA_160_C1 = [290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3
290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3
290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 -inf -inf -inf -inf -inf -inf -inf -
inf -inf -inf -inf -inf -inf -inf -inf -inf];
% Ángulos definidos para 802.11ac Modelo C Cluster 1 160MHz.
AoD_160_C1 = [13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5
13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5
13.5 13.5 13.5 13.5 -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -
inf -inf];

% Los ángulos OFFSET se suman a los ángulos del estándar para cada usuario.
    if x==1;
        % Ángulo AoA para un solo usuario.
        AoA_Modelo_C_160_C1 = [290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3
290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3
290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 290.3 -inf -inf -inf -inf
-inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf];
        % Ángulo AoD para un solo usuario.
        AoD_Modelo_C_160_C1 = [13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5
13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5 13.5
13.5 13.5 13.5 13.5 13.5 13.5 13.5 -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -
inf -inf -inf -inf -inf];
```

```
    else  x>1;
        for i=1:x;
            for j=1:size(AoA_160_C1,2);
                % Matriz ángulos AoA + Offset.
                AoA_Modelo_C_160_C1(i,j)=AoA_160_C1(j)+ Angulos_Offset_C(1,i);
                % Matriz ángulos AoD + Offset.
                AoD_Modelo_C_160_C1(i,j)=AoD_160_C1(j)+ Angulos_Offset_C(2,i);
            end
        end
    end
end
```

Angulos_Modelo_C_160MHz_C2

```
%Función para adicionar el ángulo offset al ángulo AoA y ángulo AoD del modelo C para un
ancho de canal de 1600MHz para el Cluster 2.

function [AoA_Modelo_C_160_C2,AoD_Modelo_C_160_C2] = Angulos_Modelo_C_160MHz_C2(usuarios)
% Variables globales
global AoA_Modelo_C_160_C2
global AoD_Modelo_C_160_C2
global x
% Se llama al resultado de ángulos generados.
[Angulos_Offset_C]= Generacion_Angulos_Offset_C;
% Ángulos definidos para 802.11ac Modelo C Cluster 2 160MHz.
AoA_160_C2 = [-inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -
inf -inf -inf -inf -inf -inf -inf -inf -inf 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3
332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3
332.3 332.3 332.3 332.3 332.3 332.3];
% Ángulos definidos para 802.11ac Modelo C Cluster 2 160MHz.
AoD_160_C2 = [-inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -
inf -inf -inf -inf -inf -inf -inf -inf -inf 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4
56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4
56.4 56.4];

% Los ángulos OFFSET se suman a los ángulos del estándar para cada usuario.
    if x==1;
        % Ángulo AoA para un solo usuario.
        AoA_Modelo_C_160_C2 =[-inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -
inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf 332.3 332.3 332.3 332.3 332.3
332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3
332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3 332.3];
        % Ángulo AoD para un solo usuario.
        AoD_Modelo_C_160_C2 =[-inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -
inf -inf -inf -inf -inf -inf -inf -inf -inf -inf -inf 56.4 56.4 56.4 56.4 56.4 56.4
56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4 56.4
56.4 56.4 56.4 56.4 56.4];

    else  x>1;
        for i=1:x;
            for j=1:size(AoA_160_C2,2);
                % Matriz ángulos AoA + Offset.
                AoA_Modelo_C_160_C2(i,j)=AoA_160_C2(j)+ Angulos_Offset_C(1,i);
                % Matriz ángulos AoD + Offset.
                AoD_Modelo_C_160_C2(i,j)=AoD_160_C2(j)+ Angulos_Offset_C(2,i);
            end
        end
    end
end
```

OFDM_20

```
%Función OFDM_20: Se genera la función OFDM con datos ingresados para un ancho de canal de
20MHz.
%Cada función OFDM se genera por usuario ya que se ingresa el valor de antenas de cada
usuario.

function [OFDM_1] = OFDM_20(Usuarios)
```

```
% Variables Globales
global x
global RX_1, global RX_2, global RX_3, global RX_4
global OFDM_1,global OFDM_2, global OFDM_3, global OFDM_4
global NDatos
global NOFDM
global Piloto
% Subportadoras Piloto para 40MHz, para las antenas de los usuarios.
Piloto=cat(3,[11;25;39;53],[11;25;39;53],[11;25;39;53]);

switch x

    case 1
    % Función OFDM para usuario 1.
    OFDM_1 =
comm.OFDMModulator('FFTLength',64,'NumGuardBandCarriers',[4;4],'CyclicPrefixLength',0,'Pilo
tInputPort',true,...
    'PilotCarrierIndices',[11;25;39;53],'NumSymbols',1,'NumTransmitAntennas',RX_1);
    Info_OFDM_1=info(OFDM_1);                   % Detalla la información del tamaño la
función OFDM_1.

    NDatos = Info_OFDM_1.DataInputSize(1);      % Valor de subportadoras de datos.
    NOFDM = Info_OFDM_1.OutputSize(1);          % Valor de subportadoras salida OFDM
usuario 1.

     case 2
     % Función OFDM para usuario 1.
     OFDM_1 =
comm.OFDMModulator('FFTLength',64,'NumGuardBandCarriers',[4;4],'CyclicPrefixLength',0,'Pilo
tInputPort',true,...
    'PilotCarrierIndices',[11;25;39;53],'NumSymbols',1,'NumTransmitAntennas',RX_1);
    Info_OFDM_1=info(OFDM_1);        % Detalla la información del tamaño la función OFDM_1.

    NDatos = Info_OFDM_1.DataInputSize(1);  % Valor de subportadoras de datos.
    NOFDM = Info_OFDM_1.OutputSize(1);    % Valor de subportadoras salida OFDM usuario 1.
    % Función OFDM para usuario 2.
    OFDM_2 =
comm.OFDMModulator('FFTLength',64,'NumGuardBandCarriers',[4;4],'CyclicPrefixLength',0,'Pilo
tInputPort',true,...
    'PilotCarrierIndices',[11;25;39;53],'NumSymbols',1,'NumTransmitAntennas',RX_2);
    Info_OFDM_2=info(OFDM_2);      % Detalla la información del tamaño la función OFDM_2.

     case 3
    % Función OFDM para usuario 1.
    OFDM_1 =
comm.OFDMModulator('FFTLength',64,'NumGuardBandCarriers',[4;4],'CyclicPrefixLength',0,'Pilo
tInputPort',true,...
    'PilotCarrierIndices',[11;25;39;53],'NumSymbols',1,'NumTransmitAntennas',RX_1);
    Info_OFDM_1=info(OFDM_1);       % Detalla la información del tamaño la función OFDM_1.

    NDatos = Info_OFDM_1.DataInputSize(1);  % Valor de subportadoras de datos.
    NOFDM = Info_OFDM_1.OutputSize(1);   % Valor de subportadoras salida OFDM usuario 1.
    % Función OFDM para usuario 2.
    OFDM_2 =
comm.OFDMModulator('FFTLength',64,'NumGuardBandCarriers',[4;4],'CyclicPrefixLength',0,'Pilo
tInputPort',true,...
    'PilotCarrierIndices',[11;25;39;53],'NumSymbols',1,'NumTransmitAntennas',RX_2);
    Info_OFDM_2=info(OFDM_2);        % Detalla la información del tamaño la función OFDM_2.
    % Función OFDM para usuario 3.
    OFDM_3 =
comm.OFDMModulator('FFTLength',64,'NumGuardBandCarriers',[4;4],'CyclicPrefixLength',0,'Pilo
tInputPort',true,...
    'PilotCarrierIndices',[11;25;39;53],'NumSymbols',1,'NumTransmitAntennas',RX_3);
    Info_OFDM_3=info(OFDM_3);        % Detalla la información del tamaño la función OFDM_3.

    case 4
    % Función OFDM para usuario 1.
     OFDM_1 =
comm.OFDMModulator('FFTLength',64,'NumGuardBandCarriers',[4;4],'CyclicPrefixLength',0,'Pilo
tInputPort',true,...
    'PilotCarrierIndices',[11;25;39;53],'NumSymbols',1,'NumTransmitAntennas',RX_1);
    Info_OFDM_1=info(OFDM_1);       % Detalla la información del tamaño la función OFDM_1.

    NDatos = Info_OFDM_1.DataInputSize(1);  % Valor de subportadoras de datos.
    NOFDM = Info_OFDM_1.OutputSize(1);    % Valor de subportadoras salida OFDM usuario 1.
```

```
    % Función OFDM para usuario 2.
    OFDM_2 =
comm.OFDMModulator('FFTLength',64,'NumGuardBandCarriers',[4;4],'CyclicPrefixLength',0,'Pilo
tInputPort',true,...
    'PilotCarrierIndices',[11;25;39;53],'NumSymbols',1,'NumTransmitAntennas',RX_2);
    Info_OFDM_2=info(OFDM_2);        % Detalla la información del tamaño la función OFDM_2.
    % Función OFDM para usuario 3.
    OFDM_3 =
comm.OFDMModulator('FFTLength',64,'NumGuardBandCarriers',[4;4],'CyclicPrefixLength',0,'Pilo
tInputPort',true,...
    'PilotCarrierIndices',[11;25;39;53],'NumSymbols',1,'NumTransmitAntennas',RX_3);
    Info_OFDM_3=info(OFDM_3);        % Detalla la información del tamaño la función OFDM_3.
    % Función OFDM para usuario 4.
    OFDM_4 =
comm.OFDMModulator('FFTLength',64,'NumGuardBandCarriers',[4;4],'CyclicPrefixLength',0,'Pilo
tInputPort',true,...
    'PilotCarrierIndices',[11;25;39;53],'NumSymbols',1,'NumTransmitAntennas',RX_4);
    Info_OFDM_4=info(OFDM_4);        % Detalla la información del tamaño la función OFDM_4.

 end

end
```

OFDM_40

```
%Función OFDM_40: Se genera la función OFDM con datos ingresados para un ancho de canal de
40MHz.
%Cada función OFDM se genera por usuario ya que se ingresa el valor de antenas de cada
usuario.

function [OFDM_1] = OFDM_40(Usuarios)
% Variables Globales
global x
global RX_1, global RX_2, global RX_3, global RX_4
global OFDM_1,global OFDM_2, global OFDM_3, global OFDM_4
global NDatos
global NOFDM
global Piloto

% Subportadoras Piloto para 40MHz, para las antenas de los usuarios.
Piloto=cat(3,[11;39;53;75;89;117],[11;39;53;75;89;117],[11;39;53;75;89;117]);

switch x

    case 1
    % Función OFDM para usuario 1.
    OFDM_1 =
comm.OFDMModulator('FFTLength',128,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...
    'PilotCarrierIndices',[11;39;53;75;89;117],'NumSymbols',1,'NumTransmitAntennas',RX_1);
    Info_OFDM_1=info(OFDM_1);        % Detalla la información del tamaño la función OFDM_1.

    NDatos = Info_OFDM_1.DataInputSize(1);  % Valor de subportadoras de datos.
    NOFDM = Info_OFDM_1.OutputSize(1);    % Valor de subpportadoras salida OFDM usuario 1.

     case 2
     % Función OFDM para usuario 1.
     OFDM_1 =
comm.OFDMModulator('FFTLength',128,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...
    'PilotCarrierIndices',[11;39;53;75;89;117],'NumSymbols',1,'NumTransmitAntennas',RX_1);
    Info_OFDM_1=info(OFDM_1);        % Detalla la información del tamaño la función OFDM_1.

    NDatos = Info_OFDM_1.DataInputSize(1);   % Valor de subportadoras de datos.
    NOFDM = Info_OFDM_1.OutputSize(1);   % Valor de subpportadoras salida OFDM usuario 1.
    % Función OFDM para usuario 2.
    OFDM_2 =
comm.OFDMModulator('FFTLength',128,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...
    'PilotCarrierIndices',[11;39;53;75;89;117],'NumSymbols',1,'NumTransmitAntennas',RX_2);
    Info_OFDM_2=info(OFDM_2);        % Detalla la información del tamaño la función OFDM_2.
```

```
     case 3
    % Función OFDM para usuario 1.
    OFDM_1 =
comm.OFDMModulator('FFTLength',128,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...
    'PilotCarrierIndices',[11;39;53;75;89;117],'NumSymbols',1,'NumTransmitAntennas',RX_1);
    Info_OFDM_1=info(OFDM_1);      % Detalla la información del tamaño la función OFDM_1.

    NDatos = Info_OFDM_1.DataInputSize(1);    % Valor de subportadoras de datos.
    NOFDM = Info_OFDM_1.OutputSize(1);   % Valor de subpportadoras salida OFDM usuario 1.
    %Función OFDM para usuario 2.
    OFDM_2 =
comm.OFDMModulator('FFTLength',128,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...
    'PilotCarrierIndices',[11;39;53;75;89;117],'NumSymbols',1,'NumTransmitAntennas',RX_2);
    Info_OFDM_2=info(OFDM_2);       % Detalla la información del tamaño la función OFDM_2.
    %Función OFDM para usuario 3.
    OFDM_3 =
comm.OFDMModulator('FFTLength',128,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...
    'PilotCarrierIndices',[11;39;53;75;89;117],'NumSymbols',1,'NumTransmitAntennas',RX_3);
    Info_OFDM_3=info(OFDM_3);       % Detalla la información del tamaño la función OFDM_3.

     case 4
    % Función OFDM para usuario 1.
     OFDM_1 =
comm.OFDMModulator('FFTLength',128,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...
    'PilotCarrierIndices',[11;39;53;75;89;117],'NumSymbols',1,'NumTransmitAntennas',RX_1);
    Info_OFDM_1=info(OFDM_1);       % Detalla la información del tamaño la función OFDM_1.

    NDatos = Info_OFDM_1.DataInputSize(1);      % Valor de subportadoras de datos.
    NOFDM = Info_OFDM_1.OutputSize(1);    % Valor de subpportadoras salida OFDM usuario 1.
    % Función OFDM para usuario 2.
    OFDM_2 =
comm.OFDMModulator('FFTLength',128,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...
    'PilotCarrierIndices',[11;39;53;75;89;117],'NumSymbols',1,'NumTransmitAntennas',RX_2);
    Info_OFDM_2=info(OFDM_2);       % Detalla la información del tamaño la función OFDM_2.
    % Función OFDM para usuario 3.
    OFDM_3 =
comm.OFDMModulator('FFTLength',128,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...
    'PilotCarrierIndices',[11;39;53;75;89;117],'NumSymbols',1,'NumTransmitAntennas',RX_3);
    Info_OFDM_3=info(OFDM_3);       % Detalla la información del tamaño la función OFDM_3.
    % Función OFDM para usuario 4.
    OFDM_4 =
comm.OFDMModulator('FFTLength',128,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...
    'PilotCarrierIndices',[11;39;53;75;89;117],'NumSymbols',1,'NumTransmitAntennas',RX_4);
    Info_OFDM_4=info(OFDM_4);       % Detalla la información del tamaño la función OFDM_4.

 end

end
```

OFDM_80

```
%Función OFDM_80: Se genera la función OFDM con datos ingresados para un ancho de canal de
80MHz.
%Cada función OFDM se genera por usuario ya que se ingresa el valor de antenas de cada
usuario.

function [OFDM_1] = OFDM_80(Usuarios)
% Variables globales
global x
global RX_1, global RX_2, global RX_3, global RX_4
global OFDM_1,global OFDM_2, global OFDM_3, global OFDM_4
global NDatos
global NOFDM
global Piloto
% Subportadoras Piloto para 40MHz, para las antenas de los usuarios.
Piloto=cat(3,[25;63;89;117;139;167;203;231],[25;63;89;117;139;167;203;231],[25;63;89;117;13
```

```
9;167;203;231]);

switch x

    case 1
    % Función OFDM para usuario 1.
    OFDM_1 =
comm.OFDMModulator('FFTLength',256,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...

'PilotCarrierIndices',[25;63;89;117;139;167;203;231],'NumSymbols',1,'NumTransmitAntennas',R
X_1)
    Info_OFDM_1=info(OFDM_1);        % Detalla la información del tamaño la función OFDM_1.

    NDatos = Info_OFDM_1.DataInputSize(1);      % Valor de subportadoras de datos.
    NOFDM = Info_OFDM_1.OutputSize(1);     % Valor de subpportadoras salida OFDM usuario 1.

     case 2
    % Función OFDM para usuario 1.
    OFDM_1 =
comm.OFDMModulator('FFTLength',256,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...

'PilotCarrierIndices',[25;63;89;117;139;167;203;231],'NumSymbols',1,'NumTransmitAntennas',R
X_1);
    Info_OFDM_1=info(OFDM_1);       % Detalla la información del tamaño la función OFDM_1.

    NDatos = Info_OFDM_1.DataInputSize(1);      % Valor de subportadoras de datos.
    NOFDM = Info_OFDM_1.OutputSize(1);    % Valor de subpportadoras salida OFDM usuario 1.
    % Función OFDM para usuario 2.
    OFDM_2 =
comm.OFDMModulator('FFTLength',256,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...

'PilotCarrierIndices',[25;63;89;117;139;167;203;231],'NumSymbols',1,'NumTransmitAntennas',R
X_2);
    Info_OFDM_2=info(OFDM_2);        % Detalla la información del tamaño la función OFDM_2.

     case 3
    % Función OFDM para usuario 1.
    OFDM_1 =
comm.OFDMModulator('FFTLength',256,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...

'PilotCarrierIndices',[25;63;89;117;139;167;203;231],'NumSymbols',1,'NumTransmitAntennas',R
X_1);
    Info_OFDM_1=info(OFDM_1);        % Detalla la información del tamaño la función OFDM_1.

    NDatos = Info_OFDM_1.DataInputSize(1);      % Valor de subportadoras de datos.
    NOFDM = Info_OFDM_1.OutputSize(1);   % Valor de subpportadoras salida OFDM usuario 1.
    % Función OFDM para usuario 2.
    OFDM_2 =
comm.OFDMModulator('FFTLength',256,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...

'PilotCarrierIndices',[25;63;89;117;139;167;203;231],'NumSymbols',1,'NumTransmitAntennas',R
X_2);
    Info_OFDM_2=info(OFDM_2);       % Detalla la información del tamaño la función OFDM_2.
    % Función OFDM para usuario 3.
    OFDM_3 =
comm.OFDMModulator('FFTLength',256,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...

'PilotCarrierIndices',[25;63;89;117;139;167;203;231],'NumSymbols',1,'NumTransmitAntennas',R
X_3);
    Info_OFDM_3=info(OFDM_3);       % Detalla la información del tamaño la función OFDM_3.

     case 4
    % Función OFDM para usuario 1.
    OFDM_1 =
comm.OFDMModulator('FFTLength',256,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...

'PilotCarrierIndices',[25;63;89;117;139;167;203;231],'NumSymbols',1,'NumTransmitAntennas',R
X_1);
```

```
    Info_OFDM_1=info(OFDM_1);       % Detalla la información del tamaño la función OFDM_1.

    NDatos = Info_OFDM_1.DataInputSize(1);      % Valor de subportadoras de datos.
    NOFDM = Info_OFDM_1.OutputSize(1);   % Valor de subpportadoras salida OFDM usuario 1.
    % Función OFDM para usuario 2.
    OFDM_2 =
comm.OFDMModulator('FFTLength',256,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...

'PilotCarrierIndices',[25;63;89;117;139;167;203;231],'NumSymbols',1,'NumTransmitAntennas',R
X_2);
    Info_OFDM_2=info(OFDM_2);       % Detalla la información del tamaño la función OFDM_2.
    % Función OFDM para usuario 3.
    OFDM_3 =
comm.OFDMModulator('FFTLength',256,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...

'PilotCarrierIndices',[25;63;89;117;139;167;203;231],'NumSymbols',1,'NumTransmitAntennas',R
X_3);
    Info_OFDM_3=info(OFDM_3);       % Detalla la información del tamaño la función OFDM_3.
    % Función OFDM para usuario 4.
    OFDM_4 =
comm.OFDMModulator('FFTLength',256,'NumGuardBandCarriers',[7;7],'CyclicPrefixLength',0,'Pil
otInputPort',true,...

'PilotCarrierIndices',[25;63;89;117;139;167;203;231],'NumSymbols',1,'NumTransmitAntennas',R
X_4);
    Info_OFDM_4=info(OFDM_4);       % Detalla la información del tamaño la función OFDM_4.

 end

end
```

OFDM_160

```
%Función OFDM_160: Se genera la función OFDM con datos ingresados para un ancho de canal de
160MHz.
%Cada función OFDM se genera por usuario ya que se ingresa el valor de antenas de cada
usuario.

function [OFDM_1] = OFDM_160(Usuarios)
% Variables Globales
global x
global RX_1, global RX_2, global RX_3, global RX_4
global OFDM_1,global OFDM_2, global OFDM_3, global OFDM_4
global NDatos
global NOFDM
global Piloto
% Subportadoras Piloto para 40MHz, para las antenas de los usuarios.
Piloto=cat(3,[25;53;89;117;139;167;203;231;281;309;345;373;395;423;459;487],[25;53;89;117;1
39;167;203;231;281;309;345;373;395;423;459;487],[25;53;89;117;139;167;203;231;281;309;345;3
73;395;423;459;487]);

switch x

    case 1
    % Función OFDM para usuario 1.
    OFDM_1 =
comm.OFDMModulator('FFTLength',512,'NumGuardBandCarriers',[14;14],'CyclicPrefixLength',0,'P
ilotInputPort',true,...

'PilotCarrierIndices',[25;53;89;117;139;167;203;231;281;309;345;373;395;423;459;487],'NumSy
mbols',1,'NumTransmitAntennas',RX_1)
    Info_OFDM_1=info(OFDM_1);        % Detalla la información del tamaño la función OFDM_1.

    NDatos = Info_OFDM_1.DataInputSize(1);     % Valor de subportadoras de datos.
    NOFDM = Info_OFDM_1.OutputSize(1);     % Valor de subpportadoras salida OFDM usuario 1.

     case 2
    % Función OFDM para usuario 1.
    OFDM_1 =
comm.OFDMModulator('FFTLength',512,'NumGuardBandCarriers',[14;14],'CyclicPrefixLength',0,'P
ilotInputPort',true,...
```

```
'PilotCarrierIndices',[25;53;89;117;139;167;203;231;281;309;345;373;395;423;459;487],'NumSy
mbols',1,'NumTransmitAntennas',RX_1);
    Info_OFDM_1=info(OFDM_1);        % Detalla la información del tamaño la función OFDM_1.

    NDatos = Info_OFDM_1.DataInputSize(1);     % Valor de subportadoras de datos.
    NOFDM = Info_OFDM_1.OutputSize(1);     % Valor de subpportadoras salida OFDM usuario 1.
    % Función OFDM para usuario 2.
    OFDM_2 =
comm.OFDMModulator('FFTLength',512,'NumGuardBandCarriers',[14;14],'CyclicPrefixLength',0,'P
ilotInputPort',true,...

'PilotCarrierIndices',[25;53;89;117;139;167;203;231;281;309;345;373;395;423;459;487],'NumSy
mbols',1,'NumTransmitAntennas',RX_2);
    Info_OFDM_2=info(OFDM_2);        % Detalla la información del tamaño la función OFDM_2.

     case 3
    % Función OFDM para usuario 1.
    OFDM_1 =
comm.OFDMModulator('FFTLength',512,'NumGuardBandCarriers',[14;14],'CyclicPrefixLength',0,'P
ilotInputPort',true,...

'PilotCarrierIndices',[25;53;89;117;139;167;203;231;281;309;345;373;395;423;459;487],'NumSy
mbols',1,'NumTransmitAntennas',RX_1);
    Info_OFDM_1=info(OFDM_1);        % Detalla la información del tamaño la función OFDM_1.

    NDatos = Info_OFDM_1.DataInputSize(1);     % Valor de subportadoras de datos.
    NOFDM = Info_OFDM_1.OutputSize(1);     % Valor de subpportadoras salida OFDM usuario 1.
    % Función OFDM para usuario 2.
    OFDM_2 =
comm.OFDMModulator('FFTLength',512,'NumGuardBandCarriers',[14;14],'CyclicPrefixLength',0,'P
ilotInputPort',true,...

'PilotCarrierIndices',[25;53;89;117;139;167;203;231;281;309;345;373;395;423;459;487],'NumSy
mbols',1,'NumTransmitAntennas',RX_2);
    Info_OFDM_2=info(OFDM_2);       % Detalla la información del tamaño la función OFDM_2.
    % Función OFDM para usuario 3.
    OFDM_3 =
comm.OFDMModulator('FFTLength',512,'NumGuardBandCarriers',[14;14],'CyclicPrefixLength',0,'P
ilotInputPort',true,...

'PilotCarrierIndices',[25;53;89;117;139;167;203;231;281;309;345;373;395;423;459;487],'NumSy
mbols',1,'NumTransmitAntennas',RX_3);
    Info_OFDM_3=info(OFDM_3);       % Detalla la información del tamaño la función OFDM_3.

     case 4
    % Función OFDM para usuario 1.
     OFDM_1 =
comm.OFDMModulator('FFTLength',512,'NumGuardBandCarriers',[14;14],'CyclicPrefixLength',0,'P
ilotInputPort',true,...

'PilotCarrierIndices',[25;53;89;117;139;167;203;231;281;309;345;373;395;423;459;487],'NumSy
mbols',1,'NumTransmitAntennas',RX_1);
    Info_OFDM_1=info(OFDM_1);        % Detalla la información del tamaño la función OFDM_1.

    NDatos = Info_OFDM_1.DataInputSize(1);  % Valor de subportadoras de datos.
    NOFDM = Info_OFDM_1.OutputSize(1);   % Valor de subpportadoras salida OFDM usuario 1.
    % Función OFDM para usuario 2.
    OFDM_2 =
comm.OFDMModulator('FFTLength',512,'NumGuardBandCarriers',[14;14],'CyclicPrefixLength',0,'P
ilotInputPort',true,...

'PilotCarrierIndices',[25;53;89;117;139;167;203;231;281;309;345;373;395;423;459;487],'NumSy
mbols',1,'NumTransmitAntennas',RX_2);
    Info_OFDM_2=info(OFDM_2);        % Detalla la información del tamaño la función OFDM_2.
    % Función OFDM para usuario 3.
    OFDM_3 =
comm.OFDMModulator('FFTLength',512,'NumGuardBandCarriers',[14;14],'CyclicPrefixLength',0,'P
ilotInputPort',true,...

'PilotCarrierIndices',[25;53;89;117;139;167;203;231;281;309;345;373;395;423;459;487],'NumSy
mbols',1,'NumTransmitAntennas',RX_3);
    Info_OFDM_3=info(OFDM_3);       % Detalla la información del tamaño la función OFDM_3.
    % Función OFDM para usuario 4.
    OFDM_4 =
comm.OFDMModulator('FFTLength',512,'NumGuardBandCarriers',[14;14],'CyclicPrefixLength',0,'P
```

```
ilotInputPort',true,...

'PilotCarrierIndices',[25;53;89;117;139;167;203;231;281;309;345;373;395;423;459;487],'NumSy
mbols',1,'NumTransmitAntennas',RX_4);
    Info_OFDM_4=info(OFDM_4);        % Detalla la información del tamaño la función OFDM_4.

 end

end
```

Generar_Datos

```
%Función Generar_Datos: Se genera el ingreso de datos de los usuarios.

function [Datos_RX_1] = Generar_Datos(Usuarios)
% Variables globales
global x
global Mod
global RX_1, global RX_2, global RX_3, global RX_4
global Datos_RX_1, global Datos_RX_2, global Datos_RX_3, global Datos_RX_4
global NDatos
global Ntramas

rng default;

switch x

    case 1
        % Genera datos caso 1 usuario.
        Datos_RX_1 = randi([0 Mod-1],Ntramas*NDatos,1,RX_1);

   case 2
        % Genera datos caso 2 usuarios.
        Datos_RX_1 = randi([0 Mod-1],Ntramas*NDatos,1,RX_1);
        Datos_RX_2 = randi([0 Mod-1],Ntramas*NDatos,1,RX_2);

    case 3
        % Genera datos caso 3 usuarios.
        Datos_RX_1 = randi([0 Mod-1],Ntramas*NDatos,1,RX_1);
        Datos_RX_2 = randi([0 Mod-1],Ntramas*NDatos,1,RX_2);
        Datos_RX_3 = randi([0 Mod-1],Ntramas*NDatos,1,RX_3);

    case 4
        % Genera datos caso 4 usuarios.
        Datos_RX_1 = randi([0 Mod-1],Ntramas*NDatos,1,RX_1);
        Datos_RX_2 = randi([0 Mod-1],Ntramas*NDatos,1,RX_2);
        Datos_RX_3 = randi([0 Mod-1],Ntramas*NDatos,1,RX_3);
        Datos_RX_4 = randi([0 Mod-1],Ntramas*NDatos,1,RX_4);
    end
end
```

Función Modulación

```
%Función Modulación: Se realiza la modulación a los datos ingresados, este proceso se lo
hace por usuario.

function [Datos_Mod_1] = Modulación(Usuarios)
% Variables Globales
global Mod
global x
global Datos_RX_1, global Datos_RX_2, global Datos_RX_3, global Datos_RX_4
global Datos_Mod_1, global Datos_Mod_2, global Datos_Mod_3, global Datos_Mod_4
global RX_1, global RX_2, global RX_3, global RX_4
global Ntramas
global NDatos

switch x
```

```
    case 1
        % Se modula los datos del usuario 1.
        Datos_Mod_1 = qammod(Datos_RX_1(:),Mod);
        % Reordena los datos según el número de antenas.
        Datos_Mod_1 = reshape(Datos_Mod_1,Ntramas*NDatos,1,RX_1);

    case 2
        %Se modula los datos del usuario 1.
        Datos_Mod_1 = qammod(Datos_RX_1(:),Mod);
        % Reordena los datos según el número de antenas.
        Datos_Mod_1 = reshape(Datos_Mod_1,Ntramas*NDatos,1,RX_1);
        %Se modula los datos del usuario 2.
        Datos_Mod_2 = qammod(Datos_RX_2(:),Mod);
        % Reordena los datos según el número de antenas.
        Datos_Mod_2 = reshape(Datos_Mod_2,Ntramas*NDatos,1,RX_2);

    case 3
        %Se modula los datos del usuario 1.
        Datos_Mod_1 = qammod(Datos_RX_1(:),Mod);
        % Reordena los datos según el número de antenas.
        Datos_Mod_1 = reshape(Datos_Mod_1,Ntramas*NDatos,1,RX_1);
        %Se modula los datos del usuario 2.
        Datos_Mod_2 = qammod(Datos_RX_2(:),Mod);
        % Reordena los datos según el número de antenas.
        Datos_Mod_2 = reshape(Datos_Mod_2,Ntramas*NDatos,1,RX_2);
        %Se modula los datos del usuario 3.
        Datos_Mod_3 = qammod(Datos_RX_3(:),Mod);
        % Reordena los datos según el número de antenas.
        Datos_Mod_3 = reshape(Datos_Mod_3,Ntramas*NDatos,1,RX_3);

    case 4
        %Se modula los datos del usuario 1.
        Datos_Mod_1 = qammod(Datos_RX_1(:),Mod);
        % Reordena los datos según el número de antenas.
        Datos_Mod_1 = reshape(Datos_Mod_1,Ntramas*NDatos,1,RX_1);
        %Se modula los datos del usuario 2.
        Datos_Mod_2 = qammod(Datos_RX_2(:),Mod);
        % Reordena los datos según el número de antenas.
        Datos_Mod_2 = reshape(Datos_Mod_2,Ntramas*NDatos,1,RX_2);
        %Se modula los datos del usuario 3.
        Datos_Mod_3 = qammod(Datos_RX_3(:),Mod);
        % Reordena los datos según el número de antenas.
        Datos_Mod_3 = reshape(Datos_Mod_3,Ntramas*NDatos,1,RX_3);
        %Se modula los datos del usuario 4.
        Datos_Mod_4 = qammod(Datos_RX_4(:),Mod);
        % Reordena los datos según el número de antenas.
        Datos_Mod_4 = reshape(Datos_Mod_4,Ntramas*NDatos,1,RX_4);
    end
end
```

Función Matriz_Corrrelación_A_1

```
function [TxCorrelationMatriz_1, RxCorrelationMatriz_1] ...
    = Matriz_Correlacion_A_1(TX_1,RX_1,P_Taps,TxSpacing,RxSpacing,AS_Tx,AoD_1,AS_Rx,AoA_1)
% Cálculo de las matrices de correlación en el transmisor y en el receptor para MIMO.
% Esta función es obtenida de la función del estándar 802.11n.
% Datos de entrada por usuario, antenas, espaciamiento entre antenas, ángulos AoA y AoD,
dispersión angular.

Np = length(P_Taps);  % Número de rutas.

% Grupos concatenados para cada ruta.
ASTx = [AS_Tx];
AoD = [AoD_1];
ASRx = [AS_Rx];
AoA = [AoA_1];

MaxNcTx = size(ASTx, 1);   %Máximo número de cluster en el transmisor.
MaxNcRx = size(ASRx, 1);   %Máximo número de cluster en el receptor.
```

```
validClustersTx = ~isinf(ASTx);
validClustersRx = ~isinf(ASRx);

NcTx = sum(validClustersTx, 1); %Número actual de cluster en el transmisor para cada ruta.
NcRx = sum(validClustersRx, 1); %Número actual de cluster en el receptor para cada ruta.

% Inicialice las matrices de correlación de tranmisión y recepción.
TxCorrelationMatriz_1 = zeros(TX_1, TX_1, Np);
RxCorrelationMatriz_1 = zeros(RX_1, RX_1, Np);

% Sentencia según el número de rutas.
for ip = 1:Np

    % Cálculo de la matriz de correlación en el transmisor.
    Rt_XX = zeros(1, TX_1);
    Rt_XY = zeros(1, TX_1);

    phi0Tx = AoD(:,ip) * pi/180;
    sigmaTx = ASTx(:, ip) * pi/180;
    delta_theta = 180 * pi/180;

    QTx = calculateQ(NcTx(ip), P_Taps(ip),sigmaTx, delta_theta);

    for ic = 1:MaxNcTx
        if validClustersTx(ic, ip) == 1  % Chequeo del cluster.
            if NcTx(ip) == 1
                iq = 1;
            else
                iq = ic;
            end
            Rt_XX = Rt_XX + QTx(iq) * calculateR_XX(TX_1, TxSpacing, ...
                                sigmaTx(ic), phi0Tx(ic), delta_theta);
            Rt_XY = Rt_XY + QTx(iq) * calculateR_XY(TX_1, TxSpacing, ...
                                sigmaTx(ic), phi0Tx(ic), delta_theta);
        end
    end
        TxCorrelationMatriz_1(:,:,ip) = toeplitz(Rt_XX + sqrt(-1)* Rt_XY); %Matriz
Correlación Transmisor.
    for it = 1:TX_1
        TxCorrelationMatriz_1(it,it,ip) = 1;    %Matriz Correlación Transmisor.
    end

    %  Cálculo de la matriz de correlación en el receptor.
    Rr_XX = zeros(1, RX_1);                 %Matriz de ceros, según las antenas usuario 1.
    Rr_XY = zeros(1, RX_1);                 %Matriz de ceros, según las antenas usuario 1.

    phi0Rx = AoA(:, ip) * pi/180;
    sigmaRx = ASRx(:, ip) * pi/180;

    QRx = calculateQ(NcRx(ip), P_Taps(ip), sigmaRx, delta_theta);

    for ic = 1:MaxNcRx
        if validClustersRx(ic, ip) == 1  % Chequeo del cluster.
            if NcRx(ip) == 1
                iq = 1;
            else
                iq = ic;
            end
            Rr_XX = Rr_XX + QRx(iq) * calculateR_XX(TX_1, RxSpacing, ...
                                sigmaRx(ic), phi0Rx(ic), delta_theta);
            Rr_XY = Rr_XY + QRx(iq) * calculateR_XY(TX_1, RxSpacing, ...
                                sigmaRx(ic), phi0Rx(ic), delta_theta);
        end
    end
        RxCorrelationMatriz_1(:,:,ip) = toeplitz(Rr_XX + sqrt(-1)* Rr_XY);       %Matriz
Correlación Receptor.
    for ir = 1:RX_1
        RxCorrelationMatriz_1(ir,ir,ip) = 1;    %Matriz Correlación Receptor.
    end
end

function Q = calculateQ(Nc, P_Taps, sigma, delta_theta)

if Nc == 1
    Q = 1/(1-exp(-sqrt(2)*delta_theta/sigma(sigma>-Inf)));
```

```
elseif Nc == 2
    Q = zeros(1, 2);
    Q(1) = 1/( 1-exp(-sqrt(2)*delta_theta/sigma(1)) ...
        + (sigma(2)*10^(P_Taps/10))/(sigma(1)*10^(P_Taps/10)) ...
        * (1-exp(-sqrt(2)*delta_theta/sigma(2))) );
    Q(2) = Q(1) * (sigma(2)*10^(P_Taps/10))/(sigma(1)*10^(P_Taps/10));
end

function R_XX = calculateR_XX(N, spacing, sigma, phi0, delta_theta)


R_XX = zeros(1, N);
R_XX(1,1) = 1;

for n = 2:N
    D = (n-1) * 2*pi* spacing;
    R_XX(1, n) = besselj(0,D);
    mInf = 100;
    sum_m = 0;
    for m = 1:mInf
        sum_m = sum_m + besselj(2*m,D)* 1/((sqrt(2)/sigma)^2+(2*m)^2) * ...
            cos(2*m*phi0).* ( sqrt(2)/sigma + exp(-sqrt(2)/sigma*delta_theta) ...
            * (-sqrt(2)/sigma*cos(2*m*delta_theta) + 2*m*sin(2*m*delta_theta) ) );
    end
    R_XX(1, n) = R_XX(1, n) + sum_m * 4/(sqrt(2)*sigma);
end

function R_XY = calculateR_XY(N, spacing, sigma, phi0, delta_theta)

R_XY = zeros(1, N);
R_XY(1,1) = 0;

for n = 2:N
    D = (n-1) * 2*pi* spacing;
    R_XY(1, n) = 0;
    mInf = 100;
    sum_m = 0;
    for m = 0:mInf
        sum_m = sum_m + besselj(2*m+1,D)* 1/((sqrt(2)/sigma)^2+(2*m+1)^2) ...
            * sin((2*m+1)*phi0) .* ( sqrt(2)/sigma + exp(-sqrt(2)/sigma*delta_theta) ...
            * (-sqrt(2)/sigma*cos((2*m+1)*delta_theta) + (2*m+1)*sin((2*m+1)*delta_theta) )
);
    end
    R_XY(1, n) = R_XY(1, n) + sum_m * 4/(sqrt(2)*sigma);
end
```

Función Matriz_Corrrelación_A_2

```
function [TxCorrelationMatriz_2, RxCorrelationMatriz_2] ...
    = Matriz_Correlacion_A_2(TX_2,RX_2,P_Taps,TxSpacing,RxSpacing,AS_Tx,AoD_2,AS_Rx,AoA_2)
% Cálculo de las matrices de correlación en el transmisor y en el receptor para MIMO.
% Esta función es obtenida de la función del estándar 802.11n.
% Datos de entrada por usuario, antenas, espaciamiento entre antenas, ángulos AoA y AoD,
dispersión angular.

Np = length(P_Taps);  % Número de rutas

% Grupos concatenados para cada ruta.
ASTx = [AS_Tx];
AoD = [AoD_2];
ASRx = [AS_Rx];
AoA = [AoA_2];

MaxNcTx = size(ASTx, 1);   %Máximo número de cluster en el transmisor.
MaxNcRx = size(ASRx, 1);   %Máximo número de cluster en el receptor.

% Ningún clusters para cada ruta.
validClustersTx = ~isinf(ASTx);
validClustersRx = ~isinf(ASRx);
```

```
NcTx = sum(validClustersTx, 1); % Número actual de cluster en el transmisor para cada ruta.
NcRx = sum(validClustersRx, 1); % Número actual de cluster en el receptor para cada ruta.

% Inicialice las matrices de correlación de tranmisión y recepción.
TxCorrelationMatriz_2 = zeros(TX_2, TX_2, Np);
RxCorrelationMatriz_2 = zeros(RX_2, RX_2, Np);

% Sentencia según el número de rutas.
for ip = 1:Np

    % Cálculo de la matriz de correlación en el transmisor.
    Rt_XX = zeros(1, TX_2);
    Rt_XY = zeros(1, TX_2);

    phi0Tx = AoD(:,ip) * pi/180;
    sigmaTx = ASTx(:, ip) * pi/180;
    delta_theta = 180 * pi/180;

    % Cálculo del coeficiente de normalización aproximado.
    QTx = calculateQ(NcTx(ip), P_Taps(ip),sigmaTx, delta_theta);

    for ic = 1:MaxNcTx
        % Chequeo de cluster válidos.
        if validClustersTx(ic, ip) == 1
            if NcTx(ip) == 1
                iq = 1;
            else
                iq = ic;
            end
            Rt_XX = Rt_XX + QTx(iq) * calculateR_XX(TX_2, TxSpacing, ...
                                sigmaTx(ic), phi0Tx(ic), delta_theta);
            Rt_XY = Rt_XY + QTx(iq) * calculateR_XY(TX_2, TxSpacing, ...
                                sigmaTx(ic), phi0Tx(ic), delta_theta);
        end
    end
    %Matriz Correlación Transmisor.
        TxCorrelationMatriz_2(:,:,ip) = toeplitz(Rt_XX + sqrt(-1)* Rt_XY);
    for it = 1:TX_2
    %Matriz Correlación Transmisor.
        TxCorrelationMatriz_2(it,it,ip) = 1;

    end

    %  Cálculo de la matriz de correlación en el receptor.
    Rr_XX = zeros(1, RX_2); %Matriz de ceros, según las antenas usuario 2.
    Rr_XY = zeros(1, RX_2); %Matriz de ceros, según las antenas usuario 2.

    phi0Rx = AoA(:, ip) * pi/180;
    sigmaRx = ASRx(:, ip) * pi/180;

    QRx = calculateQ(NcRx(ip), P_Taps(ip), sigmaRx, delta_theta);

    for ic = 1:MaxNcRx
        if validClustersRx(ic, ip) == 1  % Chequeo del cluster.
            if NcRx(ip) == 1
                iq = 1;
            else
                iq = ic;
            end
            Rr_XX = Rr_XX + QRx(iq) * calculateR_XX(TX_2, RxSpacing, ...
                                sigmaRx(ic), phi0Rx(ic), delta_theta);
            Rr_XY = Rr_XY + QRx(iq) * calculateR_XY(TX_2, RxSpacing, ...
                                sigmaRx(ic), phi0Rx(ic), delta_theta);
        end
    end
    RxCorrelationMatriz_2(:,:,ip) = toeplitz(Rr_XX + sqrt(-1)* Rr_XY);
    for ir = 1:RX_2
        RxCorrelationMatriz_2(ir,ir,ip) = 1;    %Matriz de Correlación del Receptor.

    end
end

function Q = calculateQ(Nc, P_Taps, sigma, delta_theta)

if Nc == 1
```

```
    Q = 1/(1-exp(-sqrt(2)*delta_theta/sigma(sigma>-Inf)));
elseif Nc == 2
    Q = zeros(1, 2);
    Q(1) = 1/( 1-exp(-sqrt(2)*delta_theta/sigma(1)) ...
        + (sigma(2)*10^(P_Taps/10))/(sigma(1)*10^(P_Taps/10)) ...
        * (1-exp(-sqrt(2)*delta_theta/sigma(2))) );
    Q(2) = Q(1) * (sigma(2)*10^(P_Taps/10))/(sigma(1)*10^(P_Taps/10));
end

function R_XX = calculateR_XX(N, spacing, sigma, phi0, delta_theta)
% Eq. (14) of [2]

R_XX = zeros(1, N);
R_XX(1,1) = 1;

for n = 2:N
    D = (n-1) * 2*pi* spacing;
    R_XX(1, n) = besselj(0,D);
    mInf = 100;
    sum_m = 0;
    for m = 1:mInf
        sum_m = sum_m + besselj(2*m,D)* 1/((sqrt(2)/sigma)^2+(2*m)^2) * ...
            cos(2*m*phi0).* ( sqrt(2)/sigma + exp(-sqrt(2)/sigma*delta_theta) ...
            * (-sqrt(2)/sigma*cos(2*m*delta_theta) + 2*m*sin(2*m*delta_theta) ) );
    end
    R_XX(1, n) = R_XX(1, n) + sum_m * 4/(sqrt(2)*sigma);
end

function R_XY = calculateR_XY(N, spacing, sigma, phi0, delta_theta)

R_XY = zeros(1, N);
R_XY(1,1) = 0;

for n = 2:N
    D = (n-1) * 2*pi* spacing;
    R_XY(1, n) = 0;
    mInf = 100;
    sum_m = 0;
    for m = 0:mInf
        sum_m = sum_m + besselj(2*m+1,D)* 1/((sqrt(2)/sigma)^2+(2*m+1)^2) ...
            * sin((2*m+1)*phi0) .* ( sqrt(2)/sigma + exp(-sqrt(2)/sigma*delta_theta) ...
            * (-sqrt(2)/sigma*cos((2*m+1)*delta_theta) + (2*m+1)*sin((2*m+1)*delta_theta) )
);
    end
    R_XY(1, n) = R_XY(1, n) + sum_m * 4/(sqrt(2)*sigma);
end
```

Función Matriz_Corrrelación_A_3

```
function [TxCorrelationMatriz_3, RxCorrelationMatriz_3] ...
    = Matriz_Correlacion_A_3(TX_3,RX_3,P_Taps,TxSpacing,RxSpacing,AS_Tx,AoD_3,AS_Rx,AoA_3)
% Cálculo de las matrices de correlación en el transmisor y en el receptor para MIMO.
% Esta función es obtenida de la función del estándar 802.11n.
% Datos de entrada por usuario, antenas, espaciamiento entre antenas, ángulos AoA y AoD,
dispersión angular.

Np = length(P_Taps);  % Número de rutas

% Grupos concatenados para cada ruta.
ASTx = [AS_Tx];
AoD = [AoD_3];
ASRx = [AS_Rx];
AoA = [AoA_3];

MaxNcTx = size(ASTx, 1);   %Máximo número de cluster en el transmisor
MaxNcRx = size(ASRx, 1);   %Máximo número de cluster en el receptor

validClustersTx = ~isinf(ASTx);
validClustersRx = ~isinf(ASRx);
```

```
NcTx = sum(validClustersTx, 1); % Número actual de cluster en el transmisor para cada ruta.
NcRx = sum(validClustersRx, 1); % Número actual de cluster en el receptor para cada ruta.

% Inicialice las matrices de correlación de tranmisión y recepción.
TxCorrelationMatriz_3 = zeros(TX_3, TX_3, Np);
RxCorrelationMatriz_3 = zeros(RX_3, RX_3, Np);

% Sentencia según el número de rutas.
for ip = 1:Np

    % Cálculo de la matriz de correlación en el transmisor.
    Rt_XX = zeros(1, TX_3);
    Rt_XY = zeros(1, TX_3);

    phi0Tx = AoD(:,ip) * pi/180;
    sigmaTx = ASTx(:, ip) * pi/180;
    delta_theta = 180 * pi/180;

    % Cálculo del coeficiente de normalización aproximado.
    QTx = calculateQ(NcTx(ip), P_Taps(ip),sigmaTx, delta_theta);

    for ic = 1:MaxNcTx
        if validClustersTx(ic, ip) == 1  % Chequeo del cluster.
            if NcTx(ip) == 1
                iq = 1;
            else
                iq = ic;
            end
            Rt_XX = Rt_XX + QTx(iq) * calculateR_XX(TX_3, TxSpacing, ...
                                sigmaTx(ic), phi0Tx(ic), delta_theta);
            Rt_XY = Rt_XY + QTx(iq) * calculateR_XY(TX_3, TxSpacing, ...
                                sigmaTx(ic), phi0Tx(ic), delta_theta);
        end
    end
    %Matriz Correlación Transmisor.
        TxCorrelationMatriz_3(:,:,ip) = toeplitz(Rt_XX + sqrt(-1)* Rt_XY);
    for it = 1:TX_3
    %Matriz Correlación Transmisor.
        TxCorrelationMatriz_3(it,it,ip) = 1;
    end

    %  Cálculo de la matriz de correlación en el receptor.
    Rr_XX = zeros(1, RX_3); %Matriz de ceros, según las antenas del usuario.
    Rr_XY = zeros(1, RX_3); %Matriz de ceros, según las antenas del usuario.

    phi0Rx = AoA(:, ip) * pi/180;
    sigmaRx = ASRx(:, ip) * pi/180;

    QRx = calculateQ(NcRx(ip), P_Taps(ip), sigmaRx, delta_theta);

    for ic = 1:MaxNcRx
        if validClustersRx(ic, ip) == 1  % Chequeo del cluster.
            if NcRx(ip) == 1
                iq = 1;
            else
                iq = ic;
            end
            Rr_XX = Rr_XX + QRx(iq) * calculateR_XX(TX_3, RxSpacing, ...
                                sigmaRx(ic), phi0Rx(ic), delta_theta);
            Rr_XY = Rr_XY + QRx(iq) * calculateR_XY(TX_3, RxSpacing, ...
                                sigmaRx(ic), phi0Rx(ic), delta_theta);
        end
    end
    RxCorrelationMatriz_3(:,:,ip) = toeplitz(Rr_XX + sqrt(-1)* Rr_XY);
    for ir = 1:RX_3
        RxCorrelationMatriz_3(ir,ir,ip) = 1;    %Matriz de Correlación del Receptor

    end
end

function Q = calculateQ(Nc, P_Taps, sigma, delta_theta)

if Nc == 1
    Q = 1/(1-exp(-sqrt(2)*delta_theta/sigma(sigma>-Inf)));
elseif Nc == 2
```

```
    Q = zeros(1, 2);
    Q(1) = 1/( 1-exp(-sqrt(2)*delta_theta/sigma(1)) ...
        + (sigma(2)*10^(P_Taps/10))/(sigma(1)*10^(P_Taps/10)) ...
        * (1-exp(-sqrt(2)*delta_theta/sigma(2))) );
    Q(2) = Q(1) * (sigma(2)*10^(P_Taps/10))/(sigma(1)*10^(P_Taps/10));
end

function R_XX = calculateR_XX(N, spacing, sigma, phi0, delta_theta)

R_XX = zeros(1, N);
R_XX(1,1) = 1;

for n = 2:N
    D = (n-1) * 2*pi* spacing;
    R_XX(1, n) = besselj(0,D);
    mInf = 100;
    sum_m = 0;
    for m = 1:mInf
        sum_m = sum_m + besselj(2*m,D)* 1/((sqrt(2)/sigma)^2+(2*m)^2) * ...
            cos(2*m*phi0).* ( sqrt(2)/sigma + exp(-sqrt(2)/sigma*delta_theta) ...
            * (-sqrt(2)/sigma*cos(2*m*delta_theta) + 2*m*sin(2*m*delta_theta) ) );
    end
    R_XX(1, n) = R_XX(1, n) + sum_m * 4/(sqrt(2)*sigma);
end

function R_XY = calculateR_XY(N, spacing, sigma, phi0, delta_theta)

R_XY = zeros(1, N);
R_XY(1,1) = 0;

for n = 2:N
    D = (n-1) * 2*pi* spacing;
    R_XY(1, n) = 0;
    mInf = 100;
    sum_m = 0;
    for m = 0:mInf
        sum_m = sum_m + besselj(2*m+1,D)* 1/((sqrt(2)/sigma)^2+(2*m+1)^2) ...
            * sin((2*m+1)*phi0) .* ( sqrt(2)/sigma + exp(-sqrt(2)/sigma*delta_theta) ...
            * (-sqrt(2)/sigma*cos((2*m+1)*delta_theta) + (2*m+1)*sin((2*m+1)*delta_theta) )
);
    end
    R_XY(1, n) = R_XY(1, n) + sum_m * 4/(sqrt(2)*sigma);
end
```

Función Matriz_Corrrelación_A_4

```
function [TxCorrelationMatriz_4, RxCorrelationMatriz_4] ...
    = Matriz_Correlacion_A_4(TX_4,RX_4,P_Taps,TxSpacing,RxSpacing,AS_Tx,AoD_4,AS_Rx,AoA_4)
% Cálculo de las matrices de correlación en el transmisor y en el receptor para MIMO.
% Esta función es obtenida de la función del estándar 802.11n.
% Datos de entrada por usuario, antenas, espaciamiento entre antenas, ángulos AoA y AoD,
dispersión angular.$

Np = length(P_Taps);  % Número de taps.

% Grupos concatenados para cada ruta.
ASTx = [AS_Tx];
AoD = [AoD_4];
ASRx = [AS_Rx];
AoA = [AoA_4];

MaxNcTx = size(ASTx, 1);   %Máximo número de cluster en el transmisor.
MaxNcRx = size(ASRx, 1);   %Máximo número de cluster en el receptor.

validClustersTx = ~isinf(ASTx);
validClustersRx = ~isinf(ASRx);

NcTx = sum(validClustersTx, 1); % Número actual de cluster en el transmisor para cada ruta.
NcRx = sum(validClustersRx, 1); % Número actual de cluster en el receptor para cada ruta.
```

```
% Inicialice las matrices de correlación de tranmisión y recepción.
TxCorrelationMatriz_4 = zeros(TX_4, TX_4, Np);
RxCorrelationMatriz_4 = zeros(RX_4, RX_4, Np);

% Sentencia según el número de rutas.
for ip = 1:Np

    % Cálculo de la matriz de correlación en el transmisor.
    Rt_XX = zeros(1, TX_4);
    Rt_XY = zeros(1, TX_4);

    phi0Tx = AoD(:,ip) * pi/180;
    sigmaTx = ASTx(:, ip) * pi/180;
    delta_theta = 180 * pi/180;

    % Cálculo del coeficiente de normalización aproximado.
    QTx = calculateQ(NcTx(ip), P_Taps(ip),sigmaTx, delta_theta);

    for ic = 1:MaxNcTx
        if validClustersTx(ic, ip) == 1
            if NcTx(ip) == 1
                iq = 1;
            else
                iq = ic;
            end
            Rt_XX = Rt_XX + QTx(iq) * calculateR_XX(TX_4, TxSpacing, ...
                                sigmaTx(ic), phi0Tx(ic), delta_theta);
            Rt_XY = Rt_XY + QTx(iq) * calculateR_XY(TX_4, TxSpacing, ...
                                sigmaTx(ic), phi0Tx(ic), delta_theta);
        end
    end
    %Matriz Correlación Transmisor.
        TxCorrelationMatriz_4(:,:,ip) = toeplitz(Rt_XX + sqrt(-1)* Rt_XY);
    for it = 1:TX_4
        %Matriz Correlación Transmisor.
        TxCorrelationMatriz_4(it,it,ip) = 1;

    end

    % Cálculo de la matriz de correlación en el transmisor.
    Rr_XX = zeros(1, RX_4);
    Rr_XY = zeros(1, RX_4);

    phi0Rx = AoA(:, ip) * pi/180;
    sigmaRx = ASRx(:, ip) * pi/180;

    % Cálculo del coeficiente de normalización aproximado.
    QRx = calculateQ(NcRx(ip), P_Taps(ip), sigmaRx, delta_theta);

    for ic = 1:MaxNcRx
        if validClustersRx(ic, ip) == 1
            if NcRx(ip) == 1
                iq = 1;
            else
                iq = ic;
            end
            Rr_XX = Rr_XX + QRx(iq) * calculateR_XX(TX_4, RxSpacing, ...
                                sigmaRx(ic), phi0Rx(ic), delta_theta);
            Rr_XY = Rr_XY + QRx(iq) * calculateR_XY(TX_4, RxSpacing, ...
                                sigmaRx(ic), phi0Rx(ic), delta_theta);
        end
    end
    %Matriz Correlación Transmisor.
    RxCorrelationMatriz_4(:,:,ip) = toeplitz(Rr_XX + sqrt(-1)* Rr_XY);
    for ir = 1:RX_4
    %Matriz Correlación Transmisor.
        RxCorrelationMatriz_4(ir,ir,ip) = 1;
    end
end

function Q = calculateQ(Nc, P_Taps, sigma, delta_theta)

if Nc == 1
    Q = 1/(1-exp(-sqrt(2)*delta_theta/sigma(sigma>-Inf)));
elseif Nc == 2
```

```
    Q = zeros(1, 2);
    Q(1) = 1/( 1-exp(-sqrt(2)*delta_theta/sigma(1)) ...
        + (sigma(2)*10^(P_Taps/10))/(sigma(1)*10^(P_Taps/10)) ...
        * (1-exp(-sqrt(2)*delta_theta/sigma(2))) );
    Q(2) = Q(1) * (sigma(2)*10^(P_Taps/10))/(sigma(1)*10^(P_Taps/10));
end

function R_XX = calculateR_XX(N, spacing, sigma, phi0, delta_theta)

R_XX = zeros(1, N);
R_XX(1,1) = 1;

for n = 2:N
    D = (n-1) * 2*pi* spacing;
    R_XX(1, n) = besselj(0,D);
    mInf = 100;
    sum_m = 0;
    for m = 1:mInf
        sum_m = sum_m + besselj(2*m,D)* 1/((sqrt(2)/sigma)^2+(2*m)^2) * ...
            cos(2*m*phi0).* ( sqrt(2)/sigma + exp(-sqrt(2)/sigma*delta_theta) ...
            * (-sqrt(2)/sigma*cos(2*m*delta_theta) + 2*m*sin(2*m*delta_theta) ) );
    end
    R_XX(1, n) = R_XX(1, n) + sum_m * 4/(sqrt(2)*sigma);
end

function R_XY = calculateR_XY(N, spacing, sigma, phi0, delta_theta)

R_XY = zeros(1, N);
R_XY(1,1) = 0;

for n = 2:N
    D = (n-1) * 2*pi* spacing;
    R_XY(1, n) = 0;
    mInf = 100;
    sum_m = 0;
    for m = 0:mInf
        sum_m = sum_m + besselj(2*m+1,D)* 1/((sqrt(2)/sigma)^2+(2*m+1)^2) ...
            * sin((2*m+1)*phi0) .* ( sqrt(2)/sigma + exp(-sqrt(2)/sigma*delta_theta) ...
            * (-sqrt(2)/sigma*cos((2*m+1)*delta_theta) + (2*m+1)*sin((2*m+1)*delta_theta) )
);
    end
    R_XY(1, n) = R_XY(1, n) + sum_m * 4/(sqrt(2)*sigma);
end
```

Función Matriz_Corrrelación_B_1, B_2, B_3, B_4

```
function [TxCorrelationMatriz_1, RxCorrelationMatriz_1] ...
    = Matriz_Correlacion_B_1(TX_1, RX_1,  P_Taps_C1,  P_Taps_C2, TxSpacing, RxSpacing, ...
                                AS_Tx_C1, AS_Tx_C2, AoD_C1, AoD_C2, ...
                                AS_Rx_C1, AS_Rx_C2, AoA_C1, AoA_C2)
% Cálculo de las matrices de correlación en el transmisor y en el receptor para MIMO.
% Esta función es obtenida de la función del estándar 802.11n.
% Datos de entrada por usuario, antenas, espaciamiento entre antenas, ángulos AoA y AoD,
dispersión angular.


Np = length( P_Taps_C1);  % Número de rutas.

% Grupos concatenados para cada ruta.
AS_Tx = [AS_Tx_C1; AS_Tx_C2];
AoD = [AoD_C1; AoD_C2];
AS_Rx = [AS_Rx_C1; AS_Rx_C2];
AoA = [AoA_C1; AoA_C2];

MaxNcTx = size(ASTx, 1);   %Máximo número de cluster en el transmisor.
MaxNcRx = size(ASRx, 1);   %Máximo número de cluster en el receptor.

validClustersTx = ~isinf(ASTx);
validClustersRx = ~isinf(ASRx);
```

```
NcTx = sum(validClustersTx, 1); % Número actual de cluster en el transmisor para cada ruta.
NcRx = sum(validClustersRx, 1); % Número actual de cluster en el receptor para cada ruta.

% Inicialice las matrices de correlación de tranmisión y recepción.
TxCorrelationMatriz_1 = zeros(TX_1, TX_1, Np);
RxCorrelationMatriz_1 = zeros(RX_1, RX_1, Np);

% Sentencia según el número de rutas.
for ip = 1:Np

    % Cálculo de la matriz de correlación en el transmisor.
    Rt_XX = zeros(1, TX_1);
    Rt_XY = zeros(1, TX_1);

    phi0Tx = AoD(:,ip) * pi/180;
    sigmaTx = AS_Tx(:, ip) * pi/180;
    delta_theta = 180 * pi/180;

    % Cálculo del coeficiente de normalización aproximado.
    QTx = calculateQ(NcTx(ip),  P_Taps_C1(ip),  P_Taps_C2(ip), sigmaTx, delta_theta);

    for ic = 1:MaxNcTx
        if validClustersTx(ic, ip) == 1  % Chequeo del cluster.
            if NcTx(ip) == 1
                iq = 1;
            else
                iq = ic;
            end
            Rt_XX = Rt_XX + QTx(iq) * calculateR_XX(TX_1, TxSpacing, ...
                                sigmaTx(ic), phi0Tx(ic), delta_theta);
            Rt_XY = Rt_XY + QTx(iq) * calculateR_XY(TX_1, TxSpacing, ...
                                sigmaTx(ic), phi0Tx(ic), delta_theta);
        end
    end
    %Matriz Correlación Transmisor.
    TxCorrelationMatriz_1(:,:,ip) = toeplitz(Rt_XX + sqrt(-1)* Rt_XY);
    for it = 1:TX_1
        %Matriz Correlación Transmisor.
        TxCorrelationMatriz_1(it,it,ip) = 1;
    end

    % Cálculo de la matriz de correlación en el transmisor.
    Rr_XX = zeros(1, RX_1);
    Rr_XY = zeros(1, RX_1);

    phi0Rx = AoA(:, ip) * pi/180;
    sigmaRx = AS_Rx(:, ip) * pi/180;

     % Cálculo del coeficiente de normalización aproximado.
    QRx = calculateQ(NcRx(ip),  P_Taps_C1(ip),  P_Taps_C2(ip), sigmaRx, delta_theta);

    for ic = 1:MaxNcRx
        if validClustersRx(ic, ip) == 1
            if NcRx(ip) == 1
                iq = 1;
            else
                iq = ic;
            end
            Rr_XX = Rr_XX + QRx(iq) * calculateR_XX(TX_1, RxSpacing, ...
                                sigmaRx(ic), phi0Rx(ic), delta_theta);
            Rr_XY = Rr_XY + QRx(iq) * calculateR_XY(TX_1, RxSpacing, ...
                                sigmaRx(ic), phi0Rx(ic), delta_theta);
        end
    end
    RxCorrelationMatriz_1(:,:,ip) = toeplitz(Rr_XX + sqrt(-1)* Rr_XY);
    for ir = 1:RX_1
        RxCorrelationMatriz_1(ir,ir,ip) = 1;
    end
end

function Q = calculateQ(Nc,  P_Taps_C1,  P_Taps_C2, sigma, delta_theta)

if Nc == 1
    Q = 1/(1-exp(-sqrt(2)*delta_theta/sigma(sigma>-Inf)));
elseif Nc == 2
```

```
    Q = zeros(1, 2);
    Q(1) = 1/( 1-exp(-sqrt(2)*delta_theta/sigma(1)) ...
        + (sigma(2)*10^( P_Taps_C2/10))/(sigma(1)*10^( P_Taps_C1/10)) ...
        * (1-exp(-sqrt(2)*delta_theta/sigma(2))) );
    Q(2) = Q(1) * (sigma(2)*10^( P_Taps_C2/10))/(sigma(1)*10^( P_Taps_C1/10));
end

function R_XX = calculateR_XX(N, spacing, sigma, phi0, delta_theta)

R_XX = zeros(1, N);
R_XX(1,1) = 1;

for n = 2:N
    D = (n-1) * 2*pi* spacing;
    R_XX(1, n) = besselj(0,D);
    mInf = 100;
    sum_m = 0;
    for m = 1:mInf
        sum_m = sum_m + besselj(2*m,D)* 1/((sqrt(2)/sigma)^2+(2*m)^2) * ...
            cos(2*m*phi0).* ( sqrt(2)/sigma + exp(-sqrt(2)/sigma*delta_theta) ...
            * (-sqrt(2)/sigma*cos(2*m*delta_theta) + 2*m*sin(2*m*delta_theta) ) );
    end
    R_XX(1, n) = R_XX(1, n) + sum_m * 4/(sqrt(2)*sigma);
end

function R_XY = calculateR_XY(N, spacing, sigma, phi0, delta_theta)

R_XY = zeros(1, N);
R_XY(1,1) = 0;

for n = 2:N
    D = (n-1) * 2*pi* spacing;
    R_XY(1, n) = 0;
    mInf = 100;
    sum_m = 0;
    for m = 0:mInf
        sum_m = sum_m + besselj(2*m+1,D)* 1/((sqrt(2)/sigma)^2+(2*m+1)^2) ...
            * sin((2*m+1)*phi0) .* ( sqrt(2)/sigma + exp(-sqrt(2)/sigma*delta_theta) ...
            * (-sqrt(2)/sigma*cos((2*m+1)*delta_theta) + (2*m+1)*sin((2*m+1)*delta_theta) )
);
    end
    R_XY(1, n) = R_XY(1, n) + sum_m * 4/(sqrt(2)*sigma);
end
```

Función Modelo_A_MimoChannel

```
%Función Modelo_A_MimoChannel: Esta función genera canal MIMO para los usuarios del
sistema.

function [H_Modelo_A_1] = Modelo_A_MimoChannel(Usuarios)
% Variables Globales
global x
global tau
global pdb
global TxCorrelationMatriz_1, global RxCorrelationMatriz_1
global TxCorrelationMatriz_2, global RxCorrelationMatriz_2
global TxCorrelationMatriz_3, global RxCorrelationMatriz_3
global TxCorrelationMatriz_4, global RxCorrelationMatriz_4
global H_MIMO_MULTIUSUARIO
global Rs
global fd
global ds

% Asignamos a R el parámetro Rician.
R='Rician';

switch x

    case 1
        % Configuración función MIMOChannel para el usuario 1 en el modelo A.
```

```
    H_Modelo_A_1= comm.MIMOChannel('SampleRate',Rs, ...
    'PathDelays',tau, ...
    'AveragePathGains',pdb, ...
    'MaximumDopplerShift',fd, ...
    'DopplerSpectrum',ds, ...
    'TransmitCorrelationMatrix',TxCorrelationMatriz_1, ...
    'ReceiveCorrelationMatrix',RxCorrelationMatriz_1,...
    'PathGainsOutputPort',true);

     % Formación matriz MU-MIMO.
     H_MIMO_MULTIUSUARIO ={H_Modelo_A_1};

case 2
    % Configuración función MIMOChannel para el usuario 1 en el modelo A.
     H_Modelo_A_1 = comm.MIMOChannel('SampleRate',Rs, ...
    'PathDelays',tau, ...
    'AveragePathGains',pdb, ...
    'MaximumDopplerShift',fd, ...
    'DopplerSpectrum',ds, ...
    'TransmitCorrelationMatrix',TxCorrelationMatriz_1, ...
    'ReceiveCorrelationMatrix',RxCorrelationMatriz_1,...
    'PathGainsOutputPort',true);
    % Configuración función MIMOChannel para el usuario 2 en el modelo A.
     H_Modelo_A_2 = comm.MIMOChannel('SampleRate',Rs, ...
    'PathDelays',tau, ...
    'AveragePathGains',pdb, ...
    'MaximumDopplerShift',fd, ...
    'DopplerSpectrum',ds, ...
    'TransmitCorrelationMatrix',TxCorrelationMatriz_2, ...
    'ReceiveCorrelationMatrix',RxCorrelationMatriz_2,...
    'PathGainsOutputPort',true);
    % Formación matriz MU-MIMO.
    H_MIMO_MULTIUSUARIO ={H_Modelo_A_1;H_Modelo_A_2};

case 3
    % Configuración función MIMOChannel para el usuario 1 en el modelo A.
     H_Modelo_A_1 = comm.MIMOChannel('SampleRate',Rs, ...
    'PathDelays',tau, ...
    'AveragePathGains',pdb, ...
    'MaximumDopplerShift',fd, ...
    'DopplerSpectrum',ds, ...
    'TransmitCorrelationMatrix',TxCorrelationMatriz_1, ...
    'ReceiveCorrelationMatrix',RxCorrelationMatriz_1,...
    'PathGainsOutputPort',true)
    % Configuración función MIMOChannel para el usuario 2 en el modelo A.
     H_Modelo_A_2 = comm.MIMOChannel('SampleRate',Rs, ...
    'PathDelays',tau, ...
    'AveragePathGains',pdb, ...
    'MaximumDopplerShift',fd, ...
    'DopplerSpectrum',ds, ...
    'TransmitCorrelationMatrix',TxCorrelationMatriz_2, ...
    'ReceiveCorrelationMatrix',RxCorrelationMatriz_2,...
    'PathGainsOutputPort',true)
    % Configuración función MIMOChannel para el usuario 3 en el modelo A.
     H_Modelo_A_3 = comm.MIMOChannel('SampleRate',Rs, ...
    'PathDelays',tau, ...
    'AveragePathGains',pdb, ...
    'MaximumDopplerShift',fd, ...
    'DopplerSpectrum',ds, ...
    'TransmitCorrelationMatrix',TxCorrelationMatriz_3, ...
    'ReceiveCorrelationMatrix',RxCorrelationMatriz_3,...
    'PathGainsOutputPort',true)
   % Formación matriz MU-MIMO
   H_MIMO_MULTIUSUARIO ={H_Modelo_A_1;H_Modelo_A_2;H_Modelo_A_3};

case 4
    % Configuración función MIMOChannel para el usuario 1 en el modelo A.
    H_Modelo_A_1 = comm.MIMOChannel('SampleRate',Rs, ...
    'PathDelays',tau, ...
    'AveragePathGains',pdb, ...
    'MaximumDopplerShift',fd, ...
    'DopplerSpectrum',ds, ...
    'TransmitCorrelationMatrix',TxCorrelationMatriz_1, ...
    'ReceiveCorrelationMatrix',RxCorrelationMatriz_1,...
    'FadingDistribution',R,...
```

```
        'DirectPathDopplerShift',0,...
        'DirectPathInitialPhase',0,....
        'PathGainsOutputPort',true);
        % Configuración función MIMOChannel para el usuario 2 en el modelo A.
         H_Modelo_A_2 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
        'AveragePathGains',pdb, ...
        'MaximumDopplerShift',fd, ...
        'DopplerSpectrum',ds, ...
        'TransmitCorrelationMatrix',TxCorrelationMatriz_2, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_2,...
        'FadingDistribution',R,...
        'DirectPathDopplerShift',0,...
        'DirectPathInitialPhase',0,....
        'PathGainsOutputPort',true);
        % Configuración función MIMOChannel para el usuario 3 en el modelo A.
         H_Modelo_A_3 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
        'AveragePathGains',pdb, ...
        'MaximumDopplerShift',fd, ...
        'DopplerSpectrum',ds, ...
        'TransmitCorrelationMatrix',TxCorrelationMatriz_3, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_3,...
        'FadingDistribution',R,...
        'DirectPathDopplerShift',0,...
        'DirectPathInitialPhase',0,....
        'PathGainsOutputPort',true);
        % Configuración función MIMOChannel para el usuario 4 en el modelo A.
         H_Modelo_A_4 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
        'AveragePathGains',pdb, ...
        'MaximumDopplerShift',fd, ...
        'DopplerSpectrum',ds, ...
        'TransmitCorrelationMatrix',TxCorrelationMatriz_4, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_4,...
        'FadingDistribution',R,...
        'DirectPathDopplerShift',0,...
        'DirectPathInitialPhase',0,....
        'PathGainsOutputPort',true);
        % Formación matriz MU-MIMO.
         H_MIMO_MULTIUSUARIO ={H_Modelo_A_1;H_Modelo_A_2;H_Modelo_A_3;H_Modelo_A_4};

    end
end
```

Función Matriz_Corrrelación_C_1, C_2, C_3, C_4

```
function [TxCorrelationMatriz_1, RxCorrelationMatriz_1] ...
    = Matriz_Correlacion_C_1(TX_1, RX_1,  P_Taps_C1,  P_Taps_C2, TxSpacing, RxSpacing, ...
                             AS_Tx_C1, AS_Tx_C2, AoD_C1, AoD_C2, ...
                             AS_Rx_C1, AS_Rx_C2, AoA_C1, AoA_C2)
% Cálculo de las matrices de correlación en el transmisor y en el receptor para MIMO.
% Esta función es obtenida de la función del estándar 802.11n.
% Datos de entrada por usuario, antenas, espaciamiento entre antenas, ángulos AoA y AoD,
dispersión angular.


Np = length( P_Taps_C1);  % Número de rutas

% Grupos concatenados para cada ruta
AS_Tx = [AS_Tx_C1; AS_Tx_C2];
AoD = [AoD_C1; AoD_C2];
AS_Rx = [AS_Rx_C1; AS_Rx_C2];
AoA = [AoA_C1; AoA_C2];

MaxNcTx = size(ASTx, 1);   %Máximo número de cluster en el transmisor.
MaxNcRx = size(ASRx, 1);   %Máximo número de cluster en el receptor.

validClustersTx = ~isinf(ASTx);
validClustersRx = ~isinf(ASRx);
```

```
NcTx = sum(validClustersTx, 1); % Número actual de cluster en el transmisor para cada ruta.
NcRx = sum(validClustersRx, 1); % Número actual de cluster en el receptor para cada ruta.

% Inicialice las matrices de correlación de tranmisión y recepción.
TxCorrelationMatriz_1 = zeros(TX_1, TX_1, Np);
RxCorrelationMatriz_1 = zeros(RX_1, RX_1, Np);

% Sentencia según el número de rutas
for ip = 1:Np

    % Cálculo de la matriz de correlación en el transmisor.
    Rt_XX = zeros(1, TX_1);
    Rt_XY = zeros(1, TX_1);

    phi0Tx = AoD(:,ip) * pi/180;
    sigmaTx = AS_Tx(:, ip) * pi/180;
    delta_theta = 180 * pi/180;

    % Cálculo del coeficiente de normalización aproximado.
    QTx = calculateQ(NcTx(ip),  P_Taps_C1(ip),  P_Taps_C2(ip), sigmaTx, delta_theta);

    for ic = 1:MaxNcTx
        if validClustersTx(ic, ip) == 1  % Chequeo del cluster.
            if NcTx(ip) == 1
                iq = 1;
            else
                iq = ic;
            end
            Rt_XX = Rt_XX + QTx(iq) * calculateR_XX(TX_1, TxSpacing, ...
                                sigmaTx(ic), phi0Tx(ic), delta_theta);
            Rt_XY = Rt_XY + QTx(iq) * calculateR_XY(TX_1, TxSpacing, ...
                                sigmaTx(ic), phi0Tx(ic), delta_theta);
        end
    end
    %Matriz Correlación Transmisor.
    TxCorrelationMatriz_1(:,:,ip) = toeplitz(Rt_XX + sqrt(-1)* Rt_XY);
    for it = 1:TX_1
        %Matriz Correlación Transmisor.
        TxCorrelationMatriz_1(it,it,ip) = 1;
    end

    % Cálculo de la matriz de correlación en el transmisor.
    Rr_XX = zeros(1, RX_1);
    Rr_XY = zeros(1, RX_1);

    phi0Rx = AoA(:, ip) * pi/180;
    sigmaRx = AS_Rx(:, ip) * pi/180;

     % Cálculo del coeficiente de normalización aproximado.
    QRx = calculateQ(NcRx(ip),  P_Taps_C1(ip),  P_Taps_C2(ip), sigmaRx, delta_theta);

    for ic = 1:MaxNcRx
        if validClustersRx(ic, ip) == 1
            if NcRx(ip) == 1
                iq = 1;
            else
                iq = ic;
            end
            Rr_XX = Rr_XX + QRx(iq) * calculateR_XX(TX_1, RxSpacing, ...
                                sigmaRx(ic), phi0Rx(ic), delta_theta);
            Rr_XY = Rr_XY + QRx(iq) * calculateR_XY(TX_1, RxSpacing, ...
                                sigmaRx(ic), phi0Rx(ic), delta_theta);
        end
    end
    RxCorrelationMatriz_1(:,:,ip) = toeplitz(Rr_XX + sqrt(-1)* Rr_XY);
    for ir = 1:RX_1
        RxCorrelationMatriz_1(ir,ir,ip) = 1;
    end
end

function Q = calculateQ(Nc,  P_Taps_C1,  P_Taps_C2, sigma, delta_theta)

if Nc == 1
    Q = 1/(1-exp(-sqrt(2)*delta_theta/sigma(sigma>-Inf)));
elseif Nc == 2
```

```
    Q = zeros(1, 2);
    Q(1) = 1/( 1-exp(-sqrt(2)*delta_theta/sigma(1)) ...
        + (sigma(2)*10^( P_Taps_C2/10))/(sigma(1)*10^( P_Taps_C1/10)) ...
        * (1-exp(-sqrt(2)*delta_theta/sigma(2))) );
    Q(2) = Q(1) * (sigma(2)*10^( P_Taps_C2/10))/(sigma(1)*10^( P_Taps_C1/10));
end

function R_XX = calculateR_XX(N, spacing, sigma, phi0, delta_theta)

R_XX = zeros(1, N);
R_XX(1,1) = 1;

for n = 2:N
    D = (n-1) * 2*pi* spacing;
    R_XX(1, n) = besselj(0,D);
    mInf = 100;
    sum_m = 0;
    for m = 1:mInf
        sum_m = sum_m + besselj(2*m,D)* 1/((sqrt(2)/sigma)^2+(2*m)^2) * ...
            cos(2*m*phi0).* ( sqrt(2)/sigma + exp(-sqrt(2)/sigma*delta_theta) ...
            * (-sqrt(2)/sigma*cos(2*m*delta_theta) + 2*m*sin(2*m*delta_theta) ) );
    end
    R_XX(1, n) = R_XX(1, n) + sum_m * 4/(sqrt(2)*sigma);
end

function R_XY = calculateR_XY(N, spacing, sigma, phi0, delta_theta)

R_XY = zeros(1, N);
R_XY(1,1) = 0;

for n = 2:N
    D = (n-1) * 2*pi* spacing;
    R_XY(1, n) = 0;
    mInf = 100;
    sum_m = 0;
    for m = 0:mInf
        sum_m = sum_m + besselj(2*m+1,D)* 1/((sqrt(2)/sigma)^2+(2*m+1)^2) ...
            * sin((2*m+1)*phi0) .* ( sqrt(2)/sigma + exp(-sqrt(2)/sigma*delta_theta) ...
            * (-sqrt(2)/sigma*cos((2*m+1)*delta_theta) + (2*m+1)*sin((2*m+1)*delta_theta) )
);
    end
    R_XY(1, n) = R_XY(1, n) + sum_m * 4/(sqrt(2)*sigma);
end
```

Función Modelo_B_MimoChannel

```
%Función Modelo_B_MimoChannel: Esta función genera canal MIMO para los usuarios del
sistema.

function [H_Modelo_B_1] = Modelo_B_MimoChannel(Usuarios)
% Variables Globales
global x
global tau
global pdb
global TxCorrelationMatriz_1, global RxCorrelationMatriz_1
global TxCorrelationMatriz_2, global RxCorrelationMatriz_2
global TxCorrelationMatriz_3, global RxCorrelationMatriz_3
global TxCorrelationMatriz_4, global RxCorrelationMatriz_4
global H_MIMO_MULTIUSUARIO
global Rs
global fd
global ds
%Asignamos a R el parámetro Rician.
R='Rician';

switch x

    case 1
        % Configuración función MIMOChannel para el usurio 1 en el modelo B.
        H_Modelo_B_1 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
```

```
    'AveragePathGains',pdb, ...
    'MaximumDopplerShift',fd, ...
    'DopplerSpectrum',ds, ...
    'FadingDistribution',R,...
    'KFactor',1,...
    'TransmitCorrelationMatrix',TxCorrelationMatriz_1, ...
    'ReceiveCorrelationMatrix',RxCorrelationMatriz_1,...
    'DirectPathDopplerShift',0,...
    'DirectPathInitialPhase',0,....
    'PathGainsOutputPort',true)
    %Matriz que contiene el canal MIMO del usuario 1.
    H_MIMO_MULTIUSUARIO ={H_Modelo_B_1};

case 2
    % Configuración función MIMOChannel para el usuario 1 en el modelo B.
    H_Modelo_B_1 = comm.MIMOChannel('SampleRate',Rs, ...
    'PathDelays',tau, ...
    'AveragePathGains',pdb, ...
    'MaximumDopplerShift',fd, ...
    'DopplerSpectrum',ds, ...
    'FadingDistribution',R,...
    'KFactor',1,...
    'TransmitCorrelationMatrix',TxCorrelationMatriz_1, ...
    'ReceiveCorrelationMatrix',RxCorrelationMatriz_1,...
    'DirectPathDopplerShift',0,...
    'DirectPathInitialPhase',0,....
    'PathGainsOutputPort',true);
    % Configuración función MIMOChannel para el usuario 2 en el modelo B.
    H_Modelo_B_2 = comm.MIMOChannel('SampleRate',Rs, ...
    'PathDelays',tau, ...
    'AveragePathGains',pdb, ...
    'MaximumDopplerShift',fd, ...
    'DopplerSpectrum',ds, ...
    'FadingDistribution',R,...
    'KFactor',1,...
    'TransmitCorrelationMatrix',TxCorrelationMatriz_2, ...
    'ReceiveCorrelationMatrix',RxCorrelationMatriz_2,...
    'DirectPathDopplerShift',0,...
    'DirectPathInitialPhase',0,....
    'PathGainsOutputPort',true);
    % Matriz MU-MIMO que contiene el canal MIMO del usuario 1 y 2.
    H_MIMO_MULTIUSUARIO ={H_Modelo_B_1;H_Modelo_B_2};

case 3
    % Configuración función MIMOChannel para el usuario 1 en el modelo B.
     H_Modelo_B_1 = comm.MIMOChannel('SampleRate',Rs, ...
    'PathDelays',tau, ...
    'AveragePathGains',pdb, ...
    'MaximumDopplerShift',fd, ...
    'DopplerSpectrum',ds, ...
    'FadingDistribution',R,...
    'KFactor',1,...
    'TransmitCorrelationMatrix',TxCorrelationMatriz_1, ...
    'ReceiveCorrelationMatrix',RxCorrelationMatriz_1,...
    'DirectPathDopplerShift',0,...
    'DirectPathInitialPhase',0,....
    'PathGainsOutputPort',true);
    % Configuración función MIMOChannel para el usuario 2 en el modelo B.
    H_Modelo_B_2 = comm.MIMOChannel('SampleRate',Rs, ...
    'PathDelays',tau, ...
    'AveragePathGains',pdb, ...
    'MaximumDopplerShift',fd, ...
    'DopplerSpectrum',ds, ...
    'FadingDistribution',R,...
    'KFactor',1,...
    'TransmitCorrelationMatrix',TxCorrelationMatriz_2, ...
    'ReceiveCorrelationMatrix',RxCorrelationMatriz_2,...
    'DirectPathDopplerShift',0,...
    'DirectPathInitialPhase',0,....
    'PathGainsOutputPort',true);
    % Configuración función MIMOChannel para el usuario 3 en el modelo B.
    H_Modelo_B_3 = comm.MIMOChannel('SampleRate',Rs, ...
    'PathDelays',tau, ...
    'AveragePathGains',pdb, ...
    'MaximumDopplerShift',fd, ...
```

```
        'DopplerSpectrum',ds, ...
        'FadingDistribution',R,...
        'KFactor',1,...
        'TransmitCorrelationMatrix',TxCorrelationMatriz_3, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_3,...
        'DirectPathDopplerShift',0,...
        'DirectPathInitialPhase',0,....
        'PathGainsOutputPort',true);
        %Matriz MU-MIMO que contiene el canal MIMO del usuario 1,2 y 3.
        H_MIMO_MULTIUSUARIO ={H_Modelo_B_1;H_Modelo_B_2;H_Modelo_B_3};

    case 4
        % Configuración función MIMOChannel para el usuario 1 en el modelo B.
        H_Modelo_B_1 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
        'AveragePathGains',pdb, ...
        'MaximumDopplerShift',fd, ...
        'DopplerSpectrum',ds, ...
        'FadingDistribution',R,...
        'KFactor',1,...
        'TransmitCorrelationMatrix',TxCorrelationMatriz_1, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_1,...
        'DirectPathDopplerShift',0,...
        'DirectPathInitialPhase',0,....
        'PathGainsOutputPort',true);
        % Configuración función MIMOChannel para el usuario 2 en el modelo B.
        H_Modelo_B_2 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
        'AveragePathGains',pdb, ...
        'MaximumDopplerShift',fd, ...
        'DopplerSpectrum',ds, ...
        'FadingDistribution',R,...
        'KFactor',1,...
        'TransmitCorrelationMatrix',TxCorrelationMatriz_2, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_2,...
        'DirectPathDopplerShift',0,...
        'DirectPathInitialPhase',0,....
        'PathGainsOutputPort',true);
        % Configuración función MIMOChannel para el usuario 3 en el modelo B.
        H_Modelo_B_3 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
        'AveragePathGains',pdb, ...
        'MaximumDopplerShift',fd, ...
        'DopplerSpectrum',ds, ...
        'FadingDistribution',R,...
        'KFactor',1,...
        'TransmitCorrelationMatrix',TxCorrelationMatriz_3, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_3,...
        'DirectPathDopplerShift',0,...
        'DirectPathInitialPhase',0,....
        'PathGainsOutputPort',true);
        % Configuración función MIMOChannel para el usuario 4 en el modelo B.
        H_Modelo_B_4 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
        'AveragePathGains',pdb, ...
        'MaximumDopplerShift',fd, ...
        'DopplerSpectrum',ds, ...
        'FadingDistribution',R,...
        'KFactor',1,...
        'TransmitCorrelationMatrix',TxCorrelationMatriz_4, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_4,...
        'DirectPathDopplerShift',0,...
        'DirectPathInitialPhase',0,....
        'PathGainsOutputPort',true);
        %Matriz MU-MIMO que contiene el canal MIMO del usuario 1,2,3 y 4.
        H_MIMO_MULTIUSUARIO ={H_Modelo_B_1;H_Modelo_B_2;H_Modelo_B_3;H_Modelo_B_4};
    end
end
```

Función Modelo_C_MimoChannel

```
%Función Modelo_C_MimoChannel_20: Esta función genera canal MIMO para los usuarios del
sistema.

function [H_Modelo_C_1] = Modelo_C_MimoChannel_20(Usuarios)
% Variables Globales
global x
global tau
global pdb
global TxCorrelationMatriz_1, global RxCorrelationMatriz_1
global TxCorrelationMatriz_2, global RxCorrelationMatriz_2
global TxCorrelationMatriz_3, global RxCorrelationMatriz_3
global TxCorrelationMatriz_4, global RxCorrelationMatriz_4
global H_MIMO_MULTIUSUARIO
global Rs
global fd
global ds

switch x

    case 1
        % Configuración función MIMOChannel para el usuario 1 en el modelo C.
        H_Modelo_C_1 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
        'AveragePathGains',pdb, ...
        'NormalizePathGains',false,...
        'MaximumDopplerShift',fd, ...
        'DopplerSpectrum',ds, ...
        'TransmitCorrelationMatrix',TxCorrelationMatriz_1, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_1,...
        'PathGainsOutputPort',true);
        % Matriz que contiene el canal MIMO del usuario 1.
        H_MIMO_MULTIUSUARIO ={H_Modelo_C_1};

    case 2
        % Configuración función MIMOChannel para el usuario 1 en el modelo C.
        H_Modelo_C_1 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
        'AveragePathGains',pdb, ...
        'NormalizePathGains',false,...
        'MaximumDopplerShift',fd, ...
        'DopplerSpectrum',ds, ...
        'TransmitCorrelationMatrix',TxCorrelationMatriz_1, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_1,...
        'PathGainsOutputPort',true);
        % Configuración función MIMOChannel para el usuario 2 en el modelo C.
        H_Modelo_C_2 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
        'AveragePathGains',pdb, ...
        'NormalizePathGains',false,...
        'MaximumDopplerShift',fd, ...
        'DopplerSpectrum',ds, ...
        'TransmitCorrelationMatrix',TxCorrelationMatriz_2, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_2,...
        'PathGainsOutputPort',true);
        % Matriz MU-MIMO que contiene el canal MIMO del usuario 1 y 2.
        H_MIMO_MULTIUSUARIO ={H_Modelo_C_1;H_Modelo_C_2};

    case 3
        % Configuración función MIMOChannel para el usuario 1 en el modelo C.
         H_Modelo_C_1 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
        'AveragePathGains',pdb, ...
        'NormalizePathGains',false,...
        'MaximumDopplerShift',fd, ...
        'DopplerSpectrum',ds, ...
        'TransmitCorrelationMatrix',TxCorrelationMatriz_1, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_1,...
        'PathGainsOutputPort',true);
        % Configuración función MIMOChannel para el usuario 2 en el modelo C.
        H_Modelo_C_2 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
        'AveragePathGains',pdb, ...
        'NormalizePathGains',false,...
        'MaximumDopplerShift',fd, ...
        'DopplerSpectrum',ds, ...
```

```
        'TransmitCorrelationMatrix',TxCorrelationMatriz_2, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_2,...
        'PathGainsOutputPort',true);
        % Configuración función MIMOChannel para el usuario 3 en el modelo C.
        H_Modelo_C_3 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
        'AveragePathGains',pdb, ...
        'NormalizePathGains',false,...
        'MaximumDopplerShift',fd, ...
        'DopplerSpectrum',ds, ...
        'TransmitCorrelationMatrix',TxCorrelationMatriz_3, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_3,...
        'PathGainsOutputPort',true);
        % Matriz MU-MIMO que contiene el canal MIMO del usuario 1,2 y 3.
        H_MIMO_MULTIUSUARIO ={H_Modelo_C_1;H_Modelo_C_2;H_Modelo_C_3}

    case 4
        % Configuración función MIMOChannel para el usuario 1 en el modelo C.
        H_Modelo_C_1 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
        'AveragePathGains',pdb, ...
        'NormalizePathGains',false,...
        'MaximumDopplerShift',fd, ...
        'DopplerSpectrum',ds, ...
        'TransmitCorrelationMatrix',TxCorrelationMatriz_1, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_1,...
        'PathGainsOutputPort',true);
        % Configuración función MIMOChannel para el usuario 2 en el modelo C.
        H_Modelo_C_2 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
        'AveragePathGains',pdb, ...
        'NormalizePathGains',false,...
        'MaximumDopplerShift',fd, ...
        'DopplerSpectrum',ds, ...
        'TransmitCorrelationMatrix',TxCorrelationMatriz_2, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_2,...
        'PathGainsOutputPort',true);
        % Configuración función MIMOChannel para el usuario 3 en el modelo C
        H_Modelo_C_3 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
        'AveragePathGains',pdb, ...
        'NormalizePathGains',false,...
        'MaximumDopplerShift',fd, ...
        'DopplerSpectrum',ds, ...
        'TransmitCorrelationMatrix',TxCorrelationMatriz_3, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_3,...
        'PathGainsOutputPort',true);
        % Configuración función MIMOChannel para el usuario 4 en el modelo C.
        H_Modelo_C_4 = comm.MIMOChannel('SampleRate',Rs, ...
        'PathDelays',tau, ...
        'AveragePathGains',pdb, ...
        'NormalizePathGains',false,...
        'MaximumDopplerShift',fd, ...
        'DopplerSpectrum',ds, ...
        'TransmitCorrelationMatrix',TxCorrelationMatriz_4, ...
        'ReceiveCorrelationMatrix',RxCorrelationMatriz_4,...
        'PathGainsOutputPort',true);
        % Matriz MU-MIMO que contiene el canal MIMO del usuario 1,2,3 y 4.
        H_MIMO_MULTIUSUARIO ={H_Modelo_C_1;H_Modelo_C_2;H_Modelo_C_3;H_Modelo_C_4};
    end
end
```

Función OFDM_MIMO

```
%Función OFDM_MIMO: Se realiza la función paso entre el canal MIMO de cada modelo y el
resultado de una función paso entre los datos
%modulados de la función OFDM independiente del canal.

function [Y_MIMO_1] = OFDM_MIMO(Usuarios)
% Variables Globales
```

```
global x
global H_MIMO_MULTIUSUARIO
global Y_MIMO_1, global Y_MIMO_2, global Y_MIMO_3, global Y_MIMO_4
global Y_Enlace_1, global Y_Enlace_2, global Y_Enlace_3, global Y_Enlace_4
global RX_1, global RX_2, global RX_3, global RX_4
global Datos_Mod_1, global Datos_Mod_2, global Datos_Mod_3, global Datos_Mod_4
global OFDM_1, global OFDM_2, global OFDM_3, global OFDM_4
global NDatos, global NOFDM
global Ntramas, global Nsamp
global tau
global Piloto

%Tamaño de los valores Y_MIMO_1 y Y_Enlace_1  para obtener los datos de salida del canal y
realizar las gráficas.
Y_MIMO_1 = zeros(Nsamp,RX_1); Y_Enlace_1 = zeros(Nsamp,length(tau),RX_1,RX_1);
%Tamaño de los valores Y_MIMO_2 y Y_Enlace_2  para obtener los datos de salida del canal y
realizar las gráficas.
Y_MIMO_2 = zeros(Nsamp,RX_2); Y_Enlace_2 = zeros(Nsamp,length(tau),RX_2,RX_2);
%Tamaño de los valores Y_MIMO_3 y Y_Enlace_3  para obtener los datos de salida del canal y
realizar las gráficas.
Y_MIMO_3 = zeros(Nsamp,RX_3); Y_Enlace_3 = zeros(Nsamp,length(tau),RX_3,RX_3);
%Tamaño de los valores Y_MIMO_4 y Y_Enlace_4  para obtener los datos de salida del canal y
realizar las gráficas.
Y_MIMO_4 = zeros(Nsamp,RX_4); Y_Enlace_4 = zeros(Nsamp,length(tau),RX_4,RX_4);

switch x

    case 1

        for iNtramas = 1:Ntramas

         Valor_Datos = (iNtramas-1)*NDatos+1:iNtramas*NDatos;
%Función que selecciona la cantidad de subportadoras piloto para las antenas del usuario 1.
           if RX_1==1
                Y_OFDM_1 = step(OFDM_1,Datos_Mod_1(Valor_Datos,:,:),Piloto(:,:,1));
            elseif RX_1==2
                Y_OFDM_1 = step(OFDM_1,Datos_Mod_1(Valor_Datos,:,:),Piloto(:,:,[1:2]));
            else
                Y_OFDM_1 = step(OFDM_1,Datos_Mod_1(Valor_Datos,:,:),Piloto(:,:,[1:3]));
           end

         Valor_MIMO = (1:NOFDM) + (iNtramas-1)*NOFDM;

        [Y_MIMO_1(Valor_MIMO,:),Y_Enlace_1(Valor_MIMO,:,:,:)] =
step(H_MIMO_MULTIUSUARIO{1,1},Y_OFDM_1);
     end

    case 2

        for iNtramas = 1:Ntramas

         Valor_Datos = (iNtramas-1)*NDatos+1:iNtramas*NDatos;
%Función que selecciona la cantidad de subportadoras piloto para las antenas del usuario 1.
           if RX_1==1
                Y_OFDM_1 = step(OFDM_1,Datos_Mod_1(Valor_Datos,:,:),Piloto(:,:,1));
            elseif RX_1==2
                Y_OFDM_1 = step(OFDM_1,Datos_Mod_1(Valor_Datos,:,:),Piloto(:,:,[1:2]));
            else
                Y_OFDM_1 = step(OFDM_1,Datos_Mod_1(Valor_Datos,:,:),Piloto(:,:,[1:3]));
            end
%Función que selecciona la cantidad de subportadoras piloto para las antenas del usuario 2.
            if RX_2==1
                Y_OFDM_2 = step(OFDM_2,Datos_Mod_2(Valor_Datos,:,:),Piloto(:,:,1));
            elseif RX_2==2
                Y_OFDM_2 = step(OFDM_2,Datos_Mod_2(Valor_Datos,:,:),Piloto(:,:,[1:2]));
            else
                Y_OFDM_2 = step(OFDM_2,Datos_Mod_2(Valor_Datos,:,:),Piloto(:,:,[1:3]));
            end

         Valor_MIMO = (1:NOFDM) + (iNtramas-1)*NOFDM;
        % Paso de la señal OFDM del usuario 1 y 2 por el canal MIMO
        [Y_MIMO_1(Valor_MIMO,:),Y_Enlace_1(Valor_MIMO,:,:,:)] =
step(H_MIMO_MULTIUSUARIO{1,1},Y_OFDM_1);
        [Y_MIMO_2(Valor_MIMO,:),Y_Enlace_2(Valor_MIMO,:,:,:)] =
step(H_MIMO_MULTIUSUARIO{2,1},Y_OFDM_2);
```

```
    end

    case 3

        for iNtramas = 1:Ntramas

       Valor_Datos = (iNtramas-1)*NDatos+1:iNtramas*NDatos;
%Función que selecciona la cantidad de subportadoras piloto para las antenas del usuario 1.
          if RX_1==1
              Y_OFDM_1 = step(OFDM_1,Datos_Mod_1(Valor_Datos,:,:),Piloto(:,:,1));
           elseif RX_1==2
              Y_OFDM_1 = step(OFDM_1,Datos_Mod_1(Valor_Datos,:,:),Piloto(:,:,[1:2]));
           else
              Y_OFDM_1 = step(OFDM_1,Datos_Mod_1(Valor_Datos,:,:),Piloto(:,:,[1:3]));
           end
%Función que selecciona la cantidad de subportadoras piloto para las antenas del usuario 2.
           if RX_2==1
              Y_OFDM_2 = step(OFDM_2,Datos_Mod_2(Valor_Datos,:,:),Piloto(:,:,1));
           elseif RX_2==2
              Y_OFDM_2 = step(OFDM_2,Datos_Mod_2(Valor_Datos,:,:),Piloto(:,:,[1:2]));
           else
              Y_OFDM_2 = step(OFDM_2,Datos_Mod_2(Valor_Datos,:,:),Piloto(:,:,[1:3]));
           end
%Función que selecciona la cantidad de subportadoras piloto para las antenas del usuario 3.
           if RX_3==1
              Y_OFDM_3 = step(OFDM_3,Datos_Mod_3(Valor_Datos,:,:),Piloto(:,:,1));
           elseif RX_3==2
              Y_OFDM_3 = step(OFDM_3,Datos_Mod_3(Valor_Datos,:,:),Piloto(:,:,[1:2]));
           else
              Y_OFDM_3 = step(OFDM_3,Datos_Mod_3(Valor_Datos,:,:),Piloto(:,:,[1:3]));
           end

        Valor_MIMO = (1:NOFDM) + (iNtramas-1)*NOFDM;
       % Paso de la señal OFDM del usuario 1, 2 y 3 por el canal MIMO
       [Y_MIMO_1(Valor_MIMO,:),Y_Enlace_1(Valor_MIMO,:,:,:)] =
step(H_MIMO_MULTIUSUARIO{1,1},Y_OFDM_1);
       [Y_MIMO_2(Valor_MIMO,:),Y_Enlace_2(Valor_MIMO,:,:,:)] =
step(H_MIMO_MULTIUSUARIO{2,1},Y_OFDM_2);
       [Y_MIMO_3(Valor_MIMO,:),Y_Enlace_3(Valor_MIMO,:,:,:)] =
step(H_MIMO_MULTIUSUARIO{3,1},Y_OFDM_3);
    end

    case 4

    for iNtramas = 1:Ntramas

       Valor_Datos = (iNtramas-1)*NDatos+1:iNtramas*NDatos;
%Función que selecciona la cantidad de subportadoras piloto para las antenas del usuario 1.
          if RX_1==1
              Y_OFDM_1 = step(OFDM_1,Datos_Mod_1(Valor_Datos,:,:),Piloto(:,:,1));
           elseif RX_1==2
              Y_OFDM_1 = step(OFDM_1,Datos_Mod_1(Valor_Datos,:,:),Piloto(:,:,[1:2]));
           else
              Y_OFDM_1 = step(OFDM_1,Datos_Mod_1(Valor_Datos,:,:),Piloto(:,:,[1:3]));
           end
%Función que selecciona la cantidad de subportadoras piloto para las antenas del usuario 2.
           if RX_2==1
              Y_OFDM_2 = step(OFDM_2,Datos_Mod_2(Valor_Datos,:,:),Piloto(:,:,1));
           elseif RX_2==2
              Y_OFDM_2 = step(OFDM_2,Datos_Mod_2(Valor_Datos,:,:),Piloto(:,:,[1:2]));
           else
              Y_OFDM_2 = step(OFDM_2,Datos_Mod_2(Valor_Datos,:,:),Piloto(:,:,[1:3]));
           end
%Función que selecciona la cantidad de subportadoras piloto para las antenas del usuario 3.
           if RX_3==1
              Y_OFDM_3 = step(OFDM_3,Datos_Mod_3(Valor_Datos,:,:),Piloto(:,:,1));
           elseif RX_3==2
              Y_OFDM_3 = step(OFDM_3,Datos_Mod_3(Valor_Datos,:,:),Piloto(:,:,[1:2]));
           else
              Y_OFDM_3 = step(OFDM_3,Datos_Mod_3(Valor_Datos,:,:),Piloto(:,:,[1:3]));
           end
%Función que selecciona la cantidad de subportadoras piloto para las antenas del usuario 4.
           if RX_4==1
              Y_OFDM_4 = step(OFDM_4,Datos_Mod_4(Valor_Datos,:,:),Piloto(:,:,1));
           elseif RX_4==2
```

```
                Y_OFDM_4 = step(OFDM_4,Datos_Mod_4(Valor_Datos,:,:),Piloto(:,:,[1:2]));
            else
                Y_OFDM_4 = step(OFDM_4,Datos_Mod_4(Valor_Datos,:,:),Piloto(:,:,[1:3]));
            end

         Valor_MIMO = (1:NOFDM) + (iNtramas-1)*NOFDM;
        % Paso de la señal OFDM del usuario 1, 2, 3 y 4 por el canal MIMO
        [Y_MIMO_1(Valor_MIMO,:),Y_Enlace_1(Valor_MIMO,:,:,:)] =
step(H_MIMO_MULTIUSUARIO{1,1},Y_OFDM_1);
        [Y_MIMO_2(Valor_MIMO,:),Y_Enlace_2(Valor_MIMO,:,:,:)] =
step(H_MIMO_MULTIUSUARIO{2,1},Y_OFDM_2);
        [Y_MIMO_3(Valor_MIMO,:),Y_Enlace_3(Valor_MIMO,:,:,:)] =
step(H_MIMO_MULTIUSUARIO{3,1},Y_OFDM_3);
        [Y_MIMO_4(Valor_MIMO,:),Y_Enlace_4(Valor_MIMO,:,:,:)] =
step(H_MIMO_MULTIUSUARIO{4,1},Y_OFDM_4);
     end
  end
end
```

Función AWGN

```
%Función AWGN: Se agrega un valor de ruido AWGN a la señal que pasa por el canal.

function [Y_MIMO_AWGN_1] = AWGN(Usuarios)
%Variables Globales
global x
global Y_MIMO_1, global Y_MIMO_2, global Y_MIMO_3, global Y_MIMO_4
global Y_MIMO_AWGN_1, global Y_MIMO_AWGN_2, global Y_MIMO_AWGN_3, global Y_MIMO_AWGN_4
global SNR

switch x

    case 1
        % Asignación de ruido blanco a la salida del canal MIMO usuario 1
        Y_MIMO_AWGN_1 = awgn(Y_MIMO_1,SNR,'measured');

    case 2
        % Asignación de ruido blanco a la salida del canal MIMO usuario 1
        Y_MIMO_AWGN_1 = awgn(Y_MIMO_1,SNR,'measured');
        % Asignación de ruido blanco a la salida del canal MIMO usuario 2
        Y_MIMO_AWGN_2 = awgn(Y_MIMO_2,SNR,'measured');

    case 3
        % Asignación de ruido blanco a la salida del canal MIMO usuario 1
        Y_MIMO_AWGN_1 = awgn(Y_MIMO_1,SNR,'measured');
        % Asignación de ruido blanco a la salida del canal MIMO usuario 2
        Y_MIMO_AWGN_2 = awgn(Y_MIMO_2,SNR,'measured');
        % Asignación de ruido blanco a la salida del canal MIMO usuario 3
        Y_MIMO_AWGN_3 = awgn(Y_MIMO_3,SNR,'measured');

    case 4
        % Asignación de ruido blanco a la salida del canal MIMO usuario 1
        Y_MIMO_AWGN_1 = awgn(Y_MIMO_1,SNR,'measured');
        % Asignación de ruido blanco a la salida del canal MIMO usuario 2
        Y_MIMO_AWGN_2 = awgn(Y_MIMO_2,SNR,'measured');
        % Asignación de ruido blanco a la salida del canal MIMO usuario 3
        Y_MIMO_AWGN_3 = awgn(Y_MIMO_3,SNR,'measured');
        % Asignación de ruido blanco a la salida del canal MIMO usuario 4
        Y_MIMO_AWGN_4 = awgn(Y_MIMO_4,SNR,'measured');

end
end
```

Función Resultados

```
%Función Velocidad: Determina el cálculo de la velocidad para cada usuario.

function [Data_rate] = Resultados(datos)
% Variables Globales
global Coding
global Nbpscs
global x
global RX_1, global RX_2, global RX_3, global RX_4
global SNR
global BW
global ig
global valor
global NDatos

g= x;
switch g

    case 1

       Ncbps = Nbpscs * NDatos;                  % Bits codificados por símbolo OFDM
       Ndbps = Ncbps * Coding;                   % Multiplicación por valor de code rate.
% Bit rate/duración de símbolo (4*10^-6), Multiplicado por número de antenas RX,
Multiplicado por intervalo de guarda.
       Data_rate = (Ndbps/(4*10^-6)) * RX_1 * ig;
       SNR_Valor=(10^(SNR/10));                          % Valor SNR en escalares.
       C = (BW*10^6)*log2(1 + SNR_Valor);                % Capacidad del Canal.
       E = Data_rate/(BW*10^6);                          % Eficiencia Espectral.
       valor=sprintf('El valor de Data Rate es: %2.3e bps.\nCapacidad del Canal: %2.3e
bps.\n Eficiencia del Sistema: %2.3f.',Data_rate,C,E);

       [Valor] = Texto_Resultados();

    case 2

       Ncbps_1 = Nbpscs * NDatos;               % Bits codificados por símbolo OFDM.
       Ndbps_1 = Ncbps_1 * Coding;              % Multiplicación por valor de code rate.
% Bit rate/duración de símbolo (4*10^-6), Multiplicado por Número de antenas RX,
Multiplicado por intervalo de guarda.
       Data_rate_1 = Ndbps_1/(4*10^-6) * RX_1 * ig;
       Ncbps_2 = Nbpscs * NDatos;               % Bits codificados por símbolo OFDM.
       Ndbps_2 = Ncbps_2 * Coding;              % Multiplicación por valor de code rate.
% Bit rate/duración de símbolo (4*10^-6), Multiplicado por Número de antenas RX,
Multiplicado por intervalo de guarda.
       Data_rate_2 = Ndbps_2/(4*10^-6) * RX_2 * ig;
       SNR_Valor=(10^(SNR/10));                          % Valor SNR en escalares.
       C = (BW*10^6)*log2(1 + SNR_Valor);                % Capacidad del Canal.
       Data_rate = Data_rate_1  + Data_rate_2;           % Bit rate total del sistema.
       E = Data_rate/(BW*10^6);                          % Eficiencia Espectral del sistema.
       valor=sprintf('El valor de Data Rate es: %2.3e bps.\n Capacidad del Canal: %2.3e
bps.\n Eficiencia del Sistema: %2.3f.',Data_rate,C,E);
       [Valor] = Texto_Resultados();

    case 3

       Ncbps_1 = Nbpscs * NDatos;                  % Bits codificados por símbolo OFDM.
       Ndbps_1 = Ncbps_1 * Coding;                 % Multiplicación por valor de code
rate.
% Bit rate/duración de símbolo (4*10^-6), Multiplicado por Número de antenas RX,
Multiplicado por intervalo de guarda.
       Data_rate_1 = Ndbps_1/(4*10^-6) * RX_1 * ig;
       Ncbps_2 = Nbpscs * NDatos;                  % Bits codificados por símbolo OFDM.
       Ndbps_2 = Ncbps_2 * Coding;                 % Multiplicación por valor de code rate.
% Bit rate/duración de símbolo (4*10^-6), Multiplicado por número de antenas RX,
Multiplicado por intervalo de guarda.
       Data_rate_2 = Ndbps_2/(4*10^-6) * RX_2 * ig;
       Ncbps_3 = Nbpscs * NDatos;                  % Bits codificados por símbolo OFDM.
       Ndbps_3 = Ncbps_3 * Coding;                 % Multiplicación por valor de code rate.
% Bit rate/duración de símbolo (4*10^-6), Multiplicado por número de antenas RX,
Multiplicado por intervalo de guarda.
       Data_rate_3 = Ndbps_3/(4*10^-6) * RX_3 * ig;

       SNR_Valor=(10^(SNR/10));                    % Valor SNR en escalares.
       C = (BW*10^6)*log2(1 + SNR_Valor);          % Capacidad del Canal.
       Data_rate = Data_rate_1  + Data_rate_2 + Data_rate_3;  %Bit rate total del sistema.
       E = Data_rate/(BW*10^6);                    %Eficiencia Espectral del sistema.
```

```
       valor=sprintf('El valor de Data Rate es: %2.3e bps.\n Capacidad del Canal: %2.3e
bps.\n Eficiencia del Sistema: %2.3f.',Data_rate,C,E);


       [Valor] = Texto_Resultados();

    case 4

       Ncbps_1 = Nbpscs * NDatos;                  % Bits codificados por símbolo OFDM.
       Ndbps_1 = Ncbps_1 * Coding;                 % Multiplicación por valor de code rate.
% Bit rate/duración de símbolo (4*10^-6), Multiplicado por número de antenas RX,
Multiplicado por intervalo de guarda.
       Data_rate_1 = Ndbps_1/(4*10^-6) * RX_1 * ig;
       Ncbps_2 = Nbpscs * NDatos;                  % Bits codificados por símbolo OFDM.
       Ndbps_2 = Ncbps_2 * Coding;                 % Multiplicación por valor de code rate.
% Bit rate/duración de símbolo (4*10^-6), Multiplicado por número de antenas RX,
Multiplicado por intervalo de guarda.
       Data_rate_2 = Ndbps_2/(4*10^-6) * RX_2 * ig;
       Ncbps_3 = Nbpscs * NDatos;                  % Bits codificados por símbolo OFDM.
       Ndbps_3 = Ncbps_3 * Coding;                 % Multiplicación por valor de code rate.
% Bit rate/duración de símbolo (4*10^-6), Multiplicado por número de antenas RX,
Multiplicado por intervalo de guarda.
       Data_rate_3 = Ndbps_3/(4*10^-6) * RX_3 * ig;
       Ncbps_4 = Nbpscs * NDatos;                  % Bits codificados por símbolo OFDM.
       Ndbps_4 = Ncbps_4 * Coding;                 % Multiplicación por valor de code rate.
% Bit rate/duración de símbolo (4*10^-6), Multiplicado por número de antenas RX,
Multiplicado por intervalo de guarda.
       Data_rate_4 = Ndbps_4/(4*10^-6) * RX_4 * ig;
       SNR_Valor=(10^(SNR/10));                    % Valor SNR en escalares.
       C = (BW*10^6)*log2(1 + SNR_Valor);          % Capacidad del Canal.
       Data_rate = Data_rate_1  + Data_rate_2 + Data_rate_3 + Data_rate_4;     %Bit rate
total del sistema.
       E = Data_rate/(BW*10^6);                    %Eficiencia Espectral del sistema.

       valor=sprintf('El valor de Data Rate es: %2.3e bps.\n Capacidad del Canal: %2.3e
bps.\n Eficiencia del Sistema: %2.3f.',Data_rate,C,E);
       [Valor] = Texto_Resultados();


end

end
```

ANEXOS F – Valores Ambiente Interno

Valores de las velocidades obtenidas con la herramienta IPERF, para 1, 5 y 10 metros.

	ESCENARIO 1	ESCENARIO 2	ESCENARIO 3	ESCENARIO 4	ESCENARIO 5	ESCENARIO 6	ESCENARIO 7	ESCENARIO 8
BW	40MHz	40MHz	40MHz	40MHz	80MHz	80MHz	80MHz	80MHz
Spatial Streams	1ss	1SS	2SS	2SS	1SS	1SS	2SS	2SS
Intervalo de guarda	800ns	400ns	800ns	400ns	800ns	400ns	800ns	400ns
Throughput	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps
1	73,9	70,4	86,1	94,3	89,2	99,7	108	113
2	74,3	60,9	93,1	100	84,1	120	147	119
3	66,5	68	80,6	61,7	71,6	118	161	108
4	78,7	69	89,4	104	86,7	105	167	114
5	73,8	57,9	81,8	81,6	90,6	87,7	142	126
6	70,4	73	77,9	104	88,7	100	111	121
7	76,9	57,6	74,9	118	90,5	94,4	154	124
8	65,9	67,2	76,4	111	112	98,4	109	133
9	76,7	53,8	85	111	104	110	110	141
10	84,7	74,7	91,8	131	97,1	108	168	140
11	68,4	79,5	90,9	103	113	118	129	166
12	65,7	63,2	79,7	127	114	113	128	147
13	76,6	75,3	81,5	149	107	124	124	149
14	79,3	72,7	66,8	124	105	119	121	114
15	76,5	89	72,4	98,3	120	121	138	143
16	72,4	94,8	62,5	102	106	126	153	172
17	83	91,5	65,1	85,7	113	115	115	165
18	85,5	92	77,6	81,6	111	117	118	175
19	72,1	88,2	90,7	76,1	121	121	120	149
20	69,7	84,1	78,5	81	113	116	116	153
21	77,2	84,1	83,9	86,4	111	102	144	143
22	81,3	89,3	82,7	85,8	127	96,3	94	122
23	77,3	77,4	80,9	81,1	121	98,7	96,4	163
24	75	63,1	83,8	98,6	104	130	141	164
25	79,3	85,4	55,9	98,2	105	111	177	152
26	78,1	83,6	81,3	80,4	120	106	102	149
27	74,8	102	83,6	110	110	126	120	140
28	83,5	89,6	89,7	115	116	125	128	109
29	76,2	97	89,1	110	106	122	125	94,7
30	73,4	81,7	89,4	101	102	108	122	134
Promedio	75,57	77,87	80,77	100,36	105,32	111,87	129,61	138,09

Tabla F.1 Ambiente Interno – 1 Metro

	ESCENARIO 1	ESCENARIO 2	ESCENARIO 3	ESCENARIO 4	ESCENARIO 5	ESCENARIO 6	ESCENARIO 7	ESCENARIO 8
BW	40MHz	40MHz	40MHz	40MHz	80MHz	80MHz	80MHz	80MHz
Spatial Streams	1ss	1SS	2SS	2SS	1SS	1SS	2SS	2SS
Intervalo de guarda	800ns	400ns	800ns	400ns	800ns	400ns	800ns	400ns
Throughput	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps
1	73,6	86	94,1	74	120	121	137	165
2	70	79,9	108	112	130	121	151	167
3	67,8	79,8	103	120	130	107	150	179
4	61,9	86,1	93,8	125	133	143	153	175
5	65,9	92,8	103	112	126	73,6	148	182
6	75	85	98,3	115	123	124	163	179
7	75,4	91,1	104	116	139	140	184	184
8	69	94,2	96,3	117	140	147	158	175
9	65	87,6	108	128	136	144	155	203
10	73	81,5	112	130	132	151	162	168
11	72,7	79,5	103	143	142	147	175	146
12	84,1	79,1	107	143	143	136	171	159
13	80,8	86,4	104	126	135	155	183	161
14	81,7	91,9	109	126	140	144	170	172
15	82,4	90,9	104	133	136	133	171	180
16	78,3	90,9	99,7	135	136	159	132	179
17	77,4	85,7	118	149	145	160	158	169
18	75,9	85,7	107	130	153	164	157	180
19	78	88,8	98	129	149	138	133	157
20	78	86,4	114	132	135	145	143	170
21	78,9	89,1	102	132	141	149	152	132
22	80	88,7	107	134	142	143	159	157
23	73,6	78,5	100	128	145	147	170	149
24	75,9	78,8	100	129	143	143	157	175
25	81,7	81,1	103	124	143	145	159	163
26	77,7	86	95,9	128	146	137	161	178
27	80,6	80,2	96,9	122	143	140	164	170
28	67,3	89,8	101	123	132	147	168	187
29	64,2	91,3	101	123	135	153	168	154
30	78,4	87,2	103	128	149	153	168	169
Promedio	74,81	86	103,13	125,53	138,07	140,32	159,33	169,47

Tabla F.2 Ambiente Interno – 5 Metros

	ESCENARIO 1	ESCENARIO 2	ESCENARIO 3	ESCENARIO 4	ESCENARIO 5	ESCENARIO 6	ESCENARIO 7	ESCENARIO 8
BW	40MHz	40MHz	40MHz	40MHz	80MHz	80MHz	80MHz	80MHz
Spatial Streams	1ss	1SS	2SS	2SS	1SS	1SS	2SS	2SS
Intervalo de guarda	800ns	400ns	800ns	400ns	800ns	400ns	800ns	400ns
Throughput	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps
1	62	77,5	52,1	31,2	80,8	31,4	35,6	37,3
2	58,3	68,6	58,6	37,9	71,1	60,3	37,2	26,7
3	71,1	70,3	62,7	40,6	90	50,9	49,4	24,6
4	59,7	72,7	55,1	26,4	116	61,4	48,2	33,1
5	49,6	76,1	58,3	28	91,1	74,8	45,5	48
6	36,6	67,9	53,6	27,6	78	69	37,4	56,1
7	47,2	56,6	56,4	34,5	82,6	79,1	31,3	65,5
8	47,8	65,3	61,7	37,6	96,4	67,1	28,4	57,2
9	51,7	63,6	55,3	46,2	115	68,4	38,1	41,7
10	58,5	73,8	56,6	47	86,1	56,5	43,8	34,9
11	45	82,5	54,1	52,5	122	91,4	48,5	49
12	44,9	79,9	59,2	54,1	93,2	71,2	47,6	45,9
13	65,8	76,9	57,8	58,3	121	68,9	42,8	42
14	55,2	85,7	57,1	55	107	76,8	44,1	42,1
15	46,9	81,3	57	48,4	98,9	85,3	48,1	41,6
16	55	71,6	63,3	59,7	121	88,8	42,4	47,2
17	48,2	71,8	43,9	64,5	109	65,5	35,8	42,5
18	78,3	78,6	53,8	60,1	65,8	97,3	41,3	40,5
19	53,1	57,9	54,3	51,5	73,5	89,4	38,4	43,9
20	59,5	55,8	46,8	56	84,1	106	28,4	43,2
21	63,3	73,3	51,1	52,2	112	67,2	41	59,7
22	52,4	75,8	49,9	54,7	92,2	72,9	47,4	52,8
23	46,8	72,8	54,9	58,4	68,3	101	49,8	47
24	64,8	83	48,4	52,9	56,1	82,8	64,4	46,2
25	66,1	88,7	48,5	57,6	69,4	86,1	45	53,4
26	58,7	71,4	44,9	52,2	85,5	59,7	51,8	50,7
27	38,7	74	48,3	52	51,9	65,5	53,2	53,1
28	49,9	79	48,3	51,5	59,2	56,8	51	44,8
29	51,1	76,4	49,9	52,2	58,5	50,1	48,5	39,1
30	45,1	71,6	56,2	53,2	45	55,4	40,7	42,6
Promedio	54,38	73,35	53,94	48,47	86,69	71,9	43,5	45,08

Tabla F.3 Ambiente Interno – 5 Metros

ANEXO G – Valores Ambiente Externo

Valores de las velocidades obtenidas con la herramienta IPERF, para 1, 5 y 10 metros para el Ambiente Externo.

	ESCENARIO 1	ESCENARIO 2	ESCENARIO 3	ESCENARIO 4	ESCENARIO 5	ESCENARIO 6	ESCENARIO 7	ESCENARIO 8
BW	40MHz	40MHz	40MHz	40MHz	80MHz	80MHz	80MHz	80MHz
Spatial Streams	1ss	1SS	2SS	2SS	1SS	1SS	2SS	2SS
Intervalo de guarda	800ns	400ns	800ns	400ns	800ns	400ns	800ns	400ns
Throughput	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps
1	64,8	56,2	82,3	114	91,6	121,2	117	125
2	73,4	65,4	87	92,5	102	90,2	169	151
3	68,4	53,4	87,8	100	102	130	181	160
4	89,5	92,9	93,9	132	114	105	169	186
5	73,4	86,8	88,1	109	120	124	146	178
6	79,8	90,6	87,8	117	106	123	132	188
7	74,6	90,8	72,7	107	89	119	127	176
8	68,3	86	76,7	149	100	121	132	128
9	76,1	84,3	92,4	106	116	121	127	121
10	79,1	78,8	92,5	93,3	112	122	117	120
11	79,6	63,3	83,3	103	126	103	147	162
12	81,7	79,6	89,7	98,6	121	129	150	167
13	64,9	83,1	86,5	117	143	121	158	131
14	76,9	78,4	89,7	112	100	118	153	126
15	73	107	85,8	123	115	120	159	155
16	78	72,1	82	110	125	122	125	151
17	80,5	77,7	95,8	93,9	124	128	144	168
18	75,3	92,4	97,1	142	114	98	155	187
19	75,9	105	94,6	126	127	122	171	152
20	76,7	102	91	117	102	130	158	129
21	74,7	88,3	89,9	111	117	112	163	150
22	81,9	89,7	89,3	100	105	120	165	123
23	82,2	69	89,1	111	124	126	150	141
24	77,9	63,7	89,4	118	115	104	145	126
25	76,7	55	96,8	95,6	133	124	165	158
26	75,6	61,5	87,6	126	114	118	148	134
27	86,8	56,6	85,2	109	133	96,5	129	164
28	84,9	75,6	87,5	99,3	122	102,1	125	164
29	87	84,6	83,7	99,8	133	103	137	179
30	79,4	80,4	96,7	98,7	123	110	137	142
Promedio	77,23	79,01	88,4	111,02	115,62	116,1	146,7	151,4

Tabla G.1 Ambiente Externo – 1 Metro

	ESCENARIO 1	ESCENARIO 2	ESCENARIO 3	ESCENARIO 4	ESCENARIO 5	ESCENARIO 6	ESCENARIO 7	ESCENARIO 8
BW	40MHz	40MHz	40MHz	40MHz	80MHz	80MHz	80MHz	80MHz
Spatial	1ss	1SS	2SS	2SS	1SS	1SS	2SS	2SS
Intervalo de	800ns	400ns	800ns	400ns	800ns	400ns	800ns	400ns
Throughput	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps
1	72,4	75,9	76,8	89,7	134	145	79,7	116
2	67	77,1	88	97	166	155	82	111
3	61,6	76,1	90,9	98	178	160	53,1	136
4	63,4	71,5	87,4	95,4	167	162	84,8	179
5	66,6	74,4	78	103	154	151	111	146
6	67,7	76,6	79,2	102	138	171	130	193
7	61,3	74,4	88,2	89,5	142	142	118	196
8	65,9	80,8	75	58,7	139	134	104	152
9	64,5	72,1	93,5	65,9	160	138	155	142
10	71,8	76,8	85,2	81,7	127	123	177	175
11	66,9	77,7	91	106	140	126	169	176
12	66,1	80,6	83,7	100	133	136	180	167
13	79,2	82,8	78,6	108	130	142	185	167
14	93,7	81,5	74,7	97,9	140	128	203	162
15	88,7	90,1	86,2	97,7	127	153	176	140
16	73,2	81,4	94,1	120	155	157	177	112
17	74,3	82,6	84,3	93	116	163	119	153
18	78,8	77,4	82	104	134	151	144	131
19	68,9	67	77,6	108	135	140	186	144
20	72,5	73,4	92,4	112	172	142	189	184
21	70,4	75,9	84	116	110	139	174	168
22	70,6	59,2	95,5	104	148	147	191	157
23	73,8	79,8	58,2	105	145	139	169	205
24	73,1	66,5	69,2	95,9	126	135	193	200
25	69,4	68,7	89,3	95,9	117	98,1	180	202
26	62,5	72,2	108	93,1	154	120	187	123
27	67	77,6	90,2	102	147	171	177	147
28	66,7	84,1	97,5	94,8	155	155	182	167
29	68,6	73,7	89,1	84,7	178	164	110	146
30	75,2	83,6	68,2	91,1	118	164	110	170
Promedio	70,73	76,38	84,53	97	142,83	145,04	149,85	158,9

Tabla G.2 Ambiente Externo – 5 Metros

	ESCENARIO 1	ESCENARIO 2	ESCENARIO 3	ESCENARIO 4	ESCENARIO 5	ESCENARIO 6	ESCENARIO 7	ESCENARIO 8
BW	40MHz	40MHz	40MHz	40MHz	80MHz	80MHz	80MHz	80MHz
Spatial Streams	1ss	1SS	2SS	2SS	1SS	1SS	2SS	2SS
Intervalo de guarda	800ns	400ns	800ns	400ns	800ns	400ns	800ns	400ns
Throughput	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps	Mbps
1	59,6	64,8	51,8	73,3	68,7	86,4	14,7	53,1
2	55,2	64,4	54,1	71,7	84,3	83,8	36,2	48,5
3	55,8	71,8	79,2	76,3	79,3	81,9	79	34,4
4	65,5	40,5	75,4	79,3	80,9	81,1	89,7	89,9
5	63,2	54,1	72,4	78,9	79,8	82,3	89,1	91,9
6	65,1	45,5	65	71,3	73,5	83,9	88,9	92,3
7	65,9	39,6	78,2	66,7	85,4	81,9	90,6	90,4
8	58,9	53,7	62,6	67,8	77,9	82,7	91,9	95,2
9	68,2	57,5	52,5	74,5	78,4	81,7	89	92,9
10	59,2	63,4	70,2	74	82,3	81,7	88,2	92,9
11	60,6	73,9	61,4	90,2	80	85,1	87,4	89,2
12	67,3	60,7	66,3	89,8	85,1	80,9	84,9	92,4
13	75,4	72	66,3	86,2	78,3	87,1	89,8	92,7
14	71	78,6	75,6	86,1	76,7	81	88,9	93,3
15	70,6	72,5	68,7	75,8	80,3	85,5	86,2	92,3
16	64,8	71,3	59,1	39,1	77,4	77,5	87,3	92,3
17	61,4	70,5	80,8	96,3	80,7	81,9	89,3	84,3
18	63,5	70,6	80,8	103	79,8	85,5	92,5	89,7
19	65,3	38,9	82,6	103	82,2	85,5	87,8	89,3
20	60,1	60,5	84,2	104	81	83,6	88,7	90,2
21	67,8	60,5	71,6	91,9	76,5	83,6	90	95,9
22	71	74,8	67,8	92,3	65,1	79,6	88,9	92,1
23	69,9	77,7	66,7	92,1	69,5	105	91,6	88,1
24	60,1	72,9	55,5	96,4	75	97	87,2	97,9
25	50,4	73,1	36,8	94,3	76,5	81,2	92,7	91,6
26	75,2	73,1	22,4	91	80,4	80,9	89,5	91,6
27	88,1	74,2	68,4	82,6	72,8	84,9	90,6	97
28	55,8	71,8	76,7	74,8	77,2	82,8	89,7	94,6
29	67,8	83,6	68	81,4	79,1	82,1	89,7	93,7
30	61,9	75,5	40,4	44,1	79,8	82,1	88	95,6
Promedio	64,82	65,4	65,38	81,61	78,13	81,1	84,6	87,51

Tabla G.3 Ambiente Externo – 10 Metros