UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

# SISTEMA INALÁMBRICO PARA TRANSMISIÓN DE SEÑALES DE AUDIO CON MÍNIMA LATENCIA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

MATÍAS FELIPE BECERRA ARAVENA

PROFESORA GUÍA:
SANDRA CÉSPEDES UMAÑA

MIEMBROS DE LA COMISIÓN:

CLAUDIO ESTÉVEZ MONTERO
JORGE SILVA SÁNCHEZ

*SANTIAGO DE CHILE*

*2016*

RESUMEN DEL TRABAJO DE TÍTULO PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

POR MATÍAS FELIPE BECERRA ARAVENA

PROFESORA GUÍA: SANDRA CÉSPEDES UMAÑA

FECHA: 11/04/2016

# SISTEMA INALÁMBRICO PARA TRANSMISIÓN DE SEÑALES DE AUDIO CON MÍNIMA LATENCIA

En el mundo de la música y el sonido, orientado a presentaciones en vivo, siempre es deseable contar con un retorno de alta calidad respecto a la señal de audio y la latencia, así como aislación acústica para resguardar la salud de los músicos y sonidistas. Por lo general, la tecnología empleada en música y sonido suele involucrar una alta cantidad de cables, los cuales dificultan la movilidad de los músicos y presentan riesgo de accidentes en el escenario, por lo cual es muy valorable para los involucrados poder prescindir de ellos con tecnología inalámbrica. Este trabajo propone un protocolo de comunicación inalámbrica con mínima latencia que soporte la transmisión de diferentes fuentes de sonido, su mezcla y retorno de la señal mezclada en el contexto de música en vivo. El protocolo propuesto se basa en conocimiento libre correspondiente a los estándares IEEE 802.11 para la capa física y acceso múltiple mediante TDMA, más aportes del autor de este trabajo para la capa de enlace, en cuanto a sincronización de nodos y configuración de parámetros del sistema de acuerdo a los requerimientos de las fuentes de sonido, topología de la red y características del canal. Luego de implementar y validar mediante simulaciones el trabajo realizado en OMNeT++, se obtienen resultados satisfactorios que incluso permiten holguras para adaptar el sistema a requerimientos más exigentes de calidad de audio o número de fuentes de sonido en caso de ser necesario. De este modo el presente trabajo deja el camino preparado para la confección de un prototipo en el futuro.

# DEDICATORIA

Cuando era niño, con algo así como 4 años, era víspera de y navidad mi papá estaba instalando el árbol de pascua. El tema de las luces del árbol era un tema de total intriga para mí, ya que había varios juegos enredados. Mi papá las probaba meticulosamente y resulta que solo algunas luces encendían (en ese entonces ni mi papá y menos yo teníamos idea de circuitos en serie o paralelo). Era de noche y yo estaba fascinado, era el problema más complejo que había visto en mi corta vida. Lo único que quería era participar y ayudar a reparar las luces para que el árbol de pascua brillara maravilloso como tenía que ser. Sin embargo era tarde y al parecer mi aporte solo dificultaba la tarea de mi papá, así que con algo de resignación me fui a acostar, confiando en mi papá que me dijo "no te preocupes, hijo, mañana lo arreglamos".

A la mañana siguiente, apenas me levanto y salgo de mi pieza, veo a mi papá ahí con el árbol de navidad con todas las luces prendidas y brillando… ¡todas las luces!... Estaba maravillado y espontáneamente exclamé emocionado:

***"¡Papá, eres casi un científico!"***

Creo que este debe haber sido mi primer acercamiento a la ciencia y al método científico. Había presenciado lo más impresionante de mi vida hasta entonces e inventé una de las frases más anecdóticas de mi familia.

Gracias papá, por mostrarme de una manera tan simpática el camino que terminaría siguiendo. Ahora tu *eres casi un científico* y yo... *un ingeniero.*

Gracias también a toda la gente que he tenido la fortuna – o mala fortuna, jajaja - de conocer. En especial a Mirta, mi mamá; a Patricio, mi papá; a Tomás, mi hermano; Macarena, mi hermana; Nicoleth, mi polola; Sandra, mi profesora; a todos mi amigos, a la FCFM, al Instituto Alonso de Ercilla y a la música, mi fiel compañera por siempre. Solo puedo decir que sin cualquiera de estas personas o elementos, no podría estar contando esta historia tal y como es ahora.

# TABLA DE CONTENIDO

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# 1 INTRODUCCIÓN

El presente trabajo consiste en el diseño de un protocolo de comunicaciones que permita comunicación inalámbrica para transmisión de señales de audio de alta calidad con mínima latencia. Para ello la solución propuesta se sustentará en la tecnología de capa física del estándar IEEE 802.11 y se propondrá un diseño de capa de acceso basado en TDMA, donde se concentrará el mayor valor de este trabajo. Ya con el diseño listo, se procederá a validar mediante simulaciones en OMNeT++, permitiendo tener una aproximación realista del desempeño del sistema propuesto y luego verificar si los requerimientos de calidad de servicio se cumplen, orientándolos a aplicaciones de música en vivo.

## 1.1 MOTIVACIÓN

La música en vivo muchas veces conlleva, para quienes la ejecutan, problemas de exposición a muchos decibeles por parte de los músicos ya que suele ser interpretada en espacios reducidos y cerrados, además de que algunos instrumentos emiten sonidos de volumen elevado (instrumentos de bronce, percusión, amplificadores eléctricos), lo cual puede resultar dañino para la salud. Esto mismo ocurre en los recitales, donde todo es amplificado para poder lograr alcance para miles de personas en un recinto. Frente a esto, una solución inmediata para resguardar la salud de los músicos y otros involucrados es usar tapones para los oídos, pero esto merma la calidad del sonido percibido por ellos, quienes siempre necesitan escuchar claramente todos los instrumentos involucrados para que su interpretación sea precisa. Este trabajo pretende diseñar y validar mediante simulación una red inalámbrica donde cada instrumento esté vinculado a un micrófono y que, luego de mezclar el sonido de cada instrumento, este se reproduzca en audífonos in-ear (para aislación acústica, idealmente con cancelación de ruido) donde cada músico tenga retorno completo y en tiempo real, obteniendo así retorno fiel y aislación acústica al mismo tiempo.

## 1.2 DEFINICIÓN DEL PROBLEMA

En la actualidad existen varias soluciones comerciales en el mercado [1] [2] que resuelven el problema de otorgar retorno inalámbrico de calidad y en tiempo real a los músicos de una banda, sin embargo son de alto costo y usan tecnología propietaria, la cual no es accesible para desarrollos futuros sobre dichas tecnologías y están protegidas por patentes. Este trabajo pretende acercarse al desempeño de las ofertas de mercado proponiendo un sistema basado en estándares libres y tecnología de fácil uso y acceso.

El retardo óptimo para aplicaciones musicales como grabación en estudio o retorno para presentaciones en vivo es algo que depende de la tolerancia y adaptación de cada músico a las condiciones que se enfrenta. Si bien no existen estudios que permitan generalizar al respecto, se ha observado que por lo general los músicos consideran que retardos de hasta 30 ms permiten

desarrollar adecuadamente grabaciones profesionales o presentaciones en vivo [3]. Así mismo, el Efecto Haas [4] explica cómo el cerebro humano percibe sonidos iguales con desfases en el tiempo de menos de 100 ms como eco o reverberación de un mismo sonido y no como sonidos independientes. Dicho esto y tras una verificación empírica, se toma como cota superior un retardo de 30 ms para el sistema propuesto, siempre valorando valores por debajo de esta cota. Como este sistema ofrece soluciones para músicos dentro de una sala de ensayo o en un escenario, se utilizará como parámetro de diseño un radio de 15 metros desde la consola (punto de acceso) entregando soluciones a redes que cumplan con estas dimensiones, pero no descartando una eventual modificación de este parámetro para alguna implementación particular.

Teniendo claro lo anterior, corresponde enfocarse en las redes inalámbricas, las cuales se caracterizan por poder utilizar el aire o espacio libre como medio de transmisión, lo cual es de gran utilidad (en el contexto de música en vivo los cables pueden provocar tropiezos, accidentes o pueden dañarse y perder conexión, por lo tanto prescindir de ellos es algo deseable), pero también trae problemas intrínsecos tales como (a) detección de nodos móviles dentro de la red; (b) impedimentos para que la información llegue a destino tales como reflexión, shadowing o absorción; (c) colisiones entre mensajes enviados simultáneamente por dos o más nodos de la red; y (d) optimización del uso del canal y ancho de banda disponibles. Para este trabajo solo se consideran relevantes los problemas (c) y (d), pues las dimensiones de la red son tales que muchos problemas típicos no existen.

Generalmente, las señales de audio digitales de alta definición se muestrean a 44.1 kHz, con una cuantización que va desde los 16 bits por muestra. Pensando en dichos valores, para transmitir una señal de audio a tiempo real y ser escuchada correctamente se requiere una tasa de transmisión de 705.6 kbps. Pensando en la red, si se tuvieran N nodos que emiten señales de audio se requeriría una tasa de transmisión de $N \cdot 705.6\ kbps$, lo cual establece el requisito del throughput de la red.

El principal desafío de este trabajo es entonces lograr obtener el servicio deseado sobre las condiciones descritas, utilizando la banda libre de frecuencias en Chile según ISM, correspondiente a la banda entre 2.4 GHz y 2.5 GHz [5]. Se pretende adaptar los distintos estándares libres y tecnologías relacionados a los requerimientos de este sistema para darle valor a esta solución desde un punto de vista académico.

# 2 REVISIÓN BIBLIOGRÁFICA Y ESTADO DEL ARTE

## 2.1 MARCO TEÓRICO Y CONCEPTOS GENERALES

Los conceptos necesarios para entender el presente trabajo son bastante variados, abarcando primero tres capas del modelo de interconexión de sistemas abiertos (OSI, por sus siglas en inglés), el cual define arquitecturas en la interconexión de sistemas de comunicaciones. Las primeras tres capas del modelo son la capa física, capa de enlace y capa de red, las cuales están directamente relacionadas con este trabajo y se definen a continuación:

- Capa Física: proporciona los medios mecánicos, eléctricos, funcionales y de procedimiento para activar, mantener y desactivar conexiones físicas. Se relaciona directamente con el manejo de bits a nivel de voltaje, ondas electromagnéticas, codificaciones, modulaciones, entre otros.
- Capa de Enlace: es responsable de la transferencia fiable de información a través de un circuito de transmisión de datos. Recibe peticiones de la capa de red y utiliza los servicios de la capa física. Se relaciona directamente con tecnologías de acceso al medio, definición de la estructura de tramas de datos, detección de errores y control de flujo.
- Capa de Red: Encargada de la conectividad, identificación y asignación de rutas para poder efectuar la comunicación entre dos nodos.

Además, existen algunos conceptos involucrados en las transmisiones inalámbricas, tales como:

- Canal: es el medio de transmisión por el cual viajan señales portadoras de información entre un emisor y un receptor. En este caso, el canal de interés corresponde al aire.
- Colisión: dado un canal, se entiende por colisión cuando dos nodos transmiten información al mismo tiempo siendo imposible rescatar la información que cada uno estaba enviando.

Ya con estos conceptos claros, se pueden definir algunos tipos de redes respecto a la extensión que tengan. En particular, este trabajo implementará una red de área local.

- Red de área local (LAN, por sus siglas en inglés): son redes de área local, lo cual se relaciona directamente con el tamaño de la red. Se usa principalmente en hogares, edificios u oficinas. El objetivo es interconectar y permitir la comunicación entre los distintos miembros de la red.

Una de los requerimientos fundamentales de esta propuesta es que la red emplee transmisión inalámbrica de datos. Para esto se tomará como referencia el estándar 802.11 de la IEEE [6] que define los protocolos y tecnologías necesarias para la transmisión de datos en redes inalámbricas.

- Estación (station): En el contexto de 802.11, corresponde a dispositivos computacionales que están habilitados para realizar transmisión inalámbrica. Las redes están diseñadas para transmitir información entre estaciones. Estas pueden eventualmente cambiar su ubicación pero seguir conectadas a la red.
- Punto de acceso (AP, de Access Point en inglés): Cumple la función de transformar las tramas 802.11 para que puedan comunicarse con el resto del mundo. Muchas veces hacen de puente

entre medios cableados e inalámbricos. Muchas topologías de redes inalámbricas usan al AP como interfaz entre las estaciones y las redes externas.

Dado que se aplicará transmisión inalámbrica, es necesario definir algunas técnicas para compartir el canal entre los distintos nodos de la red. Para esto se definen los siguientes conceptos:

- Multiplexión: La multiplexión es una técnica que tiene como objetivo la utilización de un mismo canal por varios flujos, logrando enviar información de varias fuentes a varios destinos por el mismo medio. A continuación se presentan las formas típicas de aplicar multiplexión.
    - Multiplexión por división en frecuencias (FDM, por sus siglas en inglés): El espectro de frecuencia del canal es dividido en bandas de frecuencia, de manera tal que a cada flujo se le asigna una de ellas logrando que transmita su señal sobre una señal portadora y que no interfieran entre sí. Es utilizada tanto con señales digitales como analógicas y su representación puede apreciarse en la figura 1, tanto para la multiplexión como para la demultiplexión.

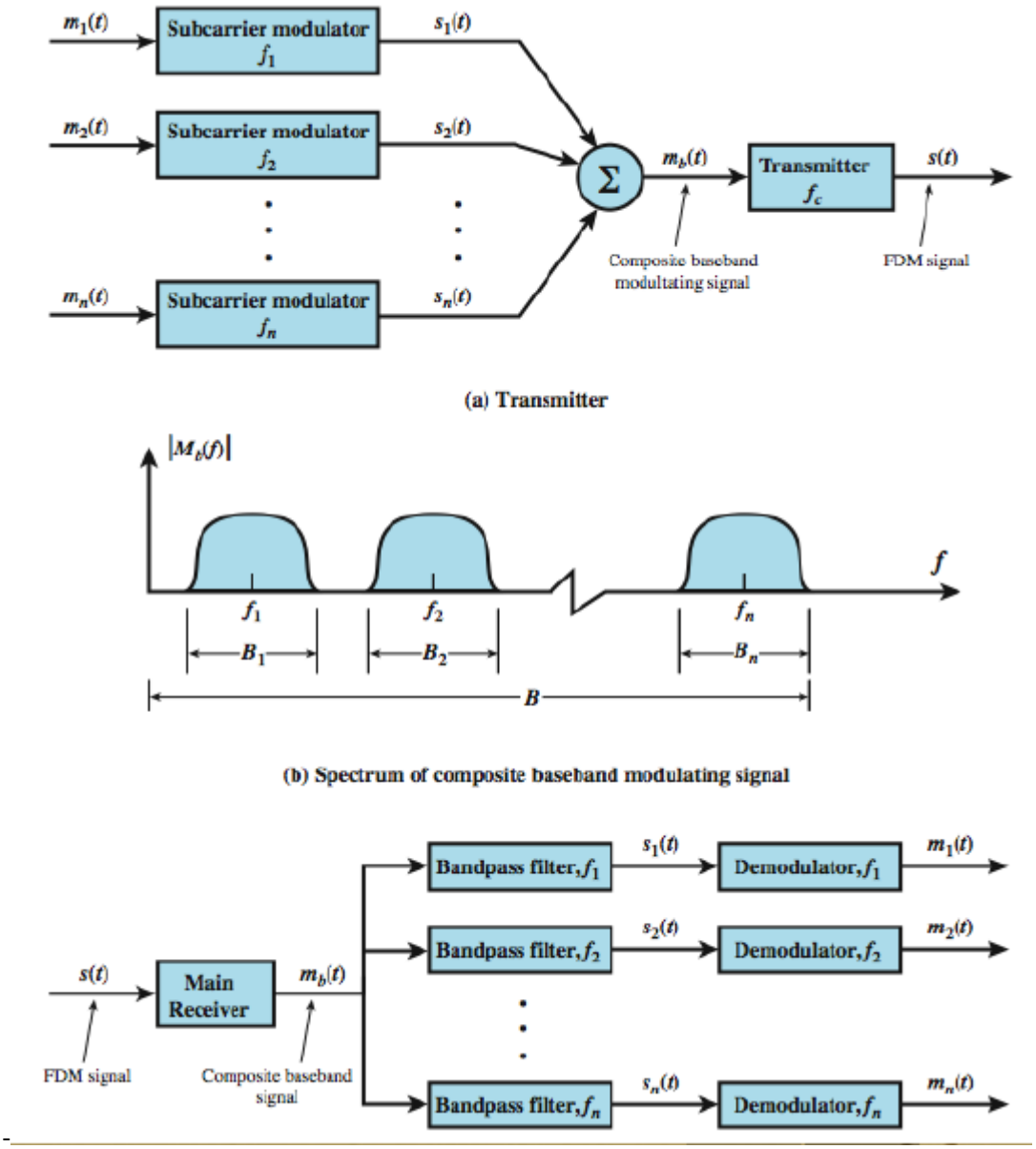


*Figura 1: Esquema de funcionamiento FDM. Tomado de [7]*

En la figura 1 se muestra un diagrama de cómo funciona FDM, donde se observa que cada flujo tiene una banda asociada, luego las señales se suman y se transmiten todas juntas para luego ser aisladas mediante filtros y demoduladas por cada nodo receptor.

- Multiplexión por división en el tiempo (TDM, de Time Division Mutiplexion en inglés): Es un sistema de multiplexión ampliamente usado para sistemas digitales, el que tiene como fin compartir el medio (asociado a una banda de frecuencia) entre distintos flujos asignando a cada flujo el canal completo durante un determinado intervalo de tiempo. En la figura 2 se muestra un esquema del funcionamiento de TDM para una red con *n* flujos, el cual se basa en time slots (tiempo asignado a cada flujo) y en frames (datos asociados a una ronda de transmisión de los *n* flujos). Es importante reconocer que si

un flujo no está enviando información, entonces el time slot asociado a ese flujo estará vacío pero igualmente ocupará parte del ancho de banda.

Cuando se emplea esta técnica de multiplexión para compartir el medio con muchos usuarios de forma coordinada se denomina Acceso Múltiple por División de Tiempo (TDMA, por su sigla en inglés).

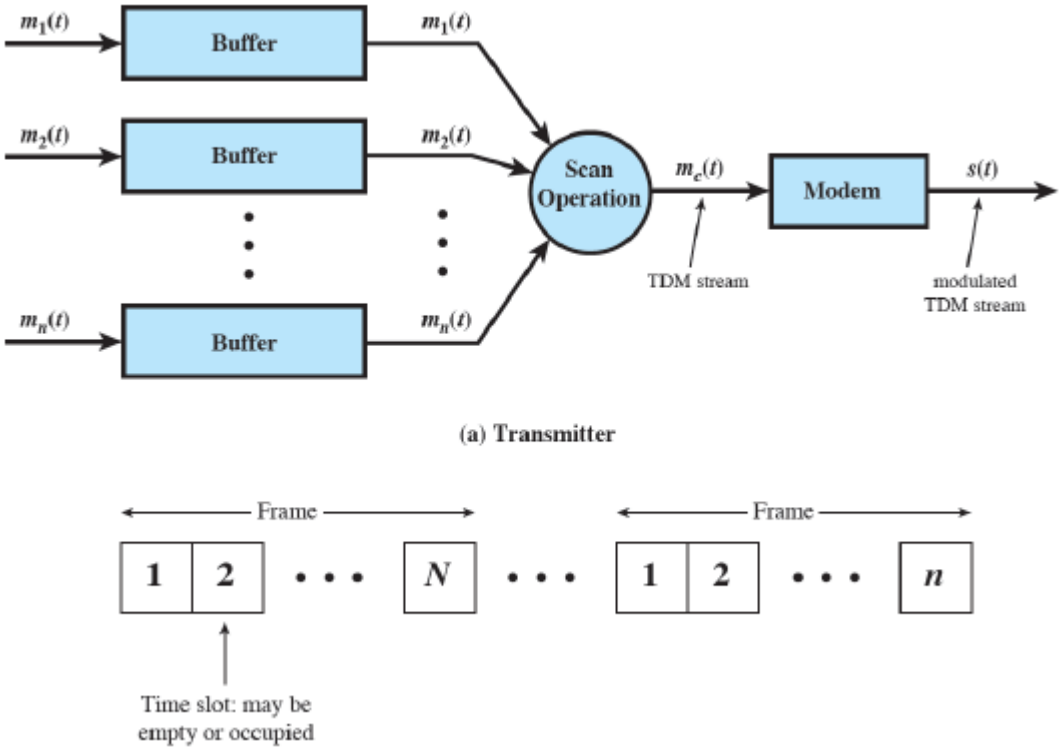


*Figura 2: Funcionamiento de TDM y estructura de un frame. N corresponde al N-ésimo timeslot Tomado de [7]*

Por otro lado queda introducir algunos conceptos necesarios para comprender e implementar la sincronización de la red.

- Tiempo de transmisión: Tiempo que se demora un emisor en poner la información en un canal. En comunicación digital, este tiempo se calcula con base en el tamaño en bits de la información y la capacidad en bits por segundo (bps) del canal.
- Tiempo de propagación: En comunicación digital, es el tiempo que tarda un bit en viajar mediante un canal, entre el emisor y el receptor. En el caso de transmisión inalámbrica viene dado por la velocidad de la luz en el aire y la distancia a recorrer.
- Tiempo de procesamiento: Tiempo que transcurre cuando alguna estación tiene que computar o procesar información para continuar con la comunicación.
- Latencia: Tiempo que transcurre entre que una señal es emitida y es recibida. Suele calcularse en función de los tiempos explicados recién.

Finalmente, un concepto importante relacionado con la calidad de servicio:

- Throughput: volumen de información neto que fluye a través de un sistema de comunicación. Para esto se considera solo la información relevante para el servicio que se está prestando, siendo la información de control y otra información "de apoyo" factores que reducen el throughput del sistema.

## 2.2 ESTADO DEL ARTE: AVANCES CIENTÍFICOS

En esta sección destacan principalmente los avances hechos por IEEE, particularmente algunos grupos de trabajo relacionados con 802 y artículos que desarrollan investigaciones al respecto. A continuación se muestran los principales aportes de estándares relacionados con el problema de este trabajo.

### 2.2.1 Estándares IEEE 802.1

Este grupo de trabajo describe la relación y el manejo de los distintos subgrupos de trabajo de los estándares IEEE 802. Para este trabajo resultan interesantes los subgrupos relacionados con Audio video Bridging (AVB), los cuales corresponden a:

- 802.1BA: AVB Systems [8]
- 802.1AS Timing and Synchronization for Time-Sensible Applications (gPTP) [9]
- 802.1Qat Stream Reservation Protocol (SRP) [10]
- 802.1Qav Forwarding and Queuing for Time Sensitive Applications (FQTSS) [10]

Estos estándares están orientados a soportar redes que involucran tráfico audiovisual, con las exigencias de latencia correspondiente, valiéndose principalmente de la tecnología desarrollada en 802.3 y 802.11n. Estas redes se caracterizan por ser escalables, tener tráfico variable y ser capaces de compatibilizar con otro tipo de tráfico (no AVB). Sin embargo, su aporte yace principalmente en definir protocolos para la interoperabilidad de estaciones con hardware conocido (tipo 802.3 y 802.11) y definir redes multi-salto adecuadas para transmisión audiovisual con ciertos requerimientos de retardo. En particular, existen dos clases de tráfico: (a) cuya latencia máxima permitida es de 2 ms; y (b) cuya latencia máxima permitida es de 50 ms. La principal limitante es que dada la tecnología usada, un salto inalámbrico, que está sustentado por 802.11n, tiene en el mejor de los casos un retardo asociado de 20 ms [11], lo cual sobrepasa la latencia tolerada para este trabajo.

### 2.2.2 Estándares IEEE 802.11

Este grupo de trabajo plantea un conjunto de estándares que permiten el acceso al medio de forma inalámbrica, es decir utilizando el aire como medio de transmisión de la información. Este estándar define los conceptos de estación, punto de acceso, medio inalámbrico y sistema de distribución. En términos muy generales, este estándar pretende homologar lo que hace el estándar 802.3, el cual está relacionado con las capas física y de enlace para conexiones cableadas (relacionado con la tecnología Ethernet), pero con transmisión inalámbrica en entornos LAN [12] [13].

Se han desarrollado varias modificaciones tales como 802.11e que pretende entregar calidad de servicio para aplicaciones de transmisión de voz o de audio, mediante múltiples colas y reservas del

medio. En el caso de 802.11g logra tasas de transmisión de hasta 54 Mbps. Finalmente 802.11n propone una alta tasa de transmisión, llegando hasta 100 Mbps ocupando un esquema de antenas de múltiples entradas/múltiples salidas (MIMO, por su sigla en inglés). Una de las principales características de estos estándares es el uso de una tecnología de acceso al medio basada en evitar el problema generado por colisiones, mediante protocolos como CSMA/CA, lo cual si bien reduce el problema de colisiones, genera retardos asociados a cada transmisión por el algoritmo de retroceso exponencial ante la presencia de colisiones.

### 2.2.3 Point Coordination Function en IEEE 802.11

IEEE 802.11 implementa dos formas de coordinación para el acceso: Distributed Coordination Function (DCF) y Point Coordination Function (PCF), los cuales operan en conjunto. De acuerdo a [14], DCF fue desarrollado para transmisión asíncrona, donde todas las estaciones utilizan CSMA/CA y retroceso exponencial para compartir el medio. Por su parte, PCF fue desarrollado para servicios que tienen requerimientos de tiempo, donde el AP determina qué estación tiene derecho a transmitir, permitiendo transmisiones en tiempo real.

PCF está basado en un esquema de polling que puede aplicarse mediante el algoritmo Round Robin aplicando prioridad de paquetes. Esto significa que básicamente el AP maneja un buffer donde van llegando paquetes de distintas estaciones, y basados en la prioridad que se le asigne a cada paquete y su tiempo de llegada al buffer, el AP reservará el canal para que cada estación pueda transmitir lo que necesita.

### 2.2.4 Implementación de TDMA sobre IEEE 802.11

Actualmente existen trabajos que proponen la aplicación de TDMA sobre redes basadas en 802.11 u 802.11e [15] [13]. Específicamente, en [15] se programa un software con la tecnología de acceso al medio mediante TDMA, aplicando sincronización entre los nodos de la red y aplicando los protocolos necesarios para lograr aplicar una comunicación multi-salto con mínima latencia. Si bien en este trabajo se tiene que el algoritmo de sincronización puede tardar 1 o 2 minutos, para efectos de este trabajo se espera que este tiempo disminuya bastante, pues esto es principalmente debido a la característica multi-saltos que posee la solución propuesta en [15] (lo cual no es parte de la solución propuesta en este trabajo). Como principal resultado se tiene una sincronización con time-slots del orden de los microsegundos y se logra que la transmisión de información utilizando el protocolo de control de transmisión (TCP, por sus siglas en inglés) logre ocupar prácticamente todo el ancho de banda disponible, presentando un buen avance respecto a lo ya existente.

### 2.2.5 SONET & SDH

Esta tecnología fue diseñada para soportar transmisión síncrona mediante un canal de fibra óptica, donde se pueden transmitir grandes cantidades de información de distintos tipos a la vez, principalmente tráfico de voz. Una de sus características principales es que es transmisión en tiempo real. Uno de los principales desafíos que se enfrentaron para implementar SONET/SDH fue que las fuentes que se debían sincronizar eran de distinto tipo. Sin embargo, SONET/SDH suele ser considerado como un protocolo de transporte en vez de uno de comunicaciones. Su ancho de banda

para un usuario varía entre 50 Mbps y 38,4 Gbps, dependiendo de las conexiones cruzadas que existan. Su mecanismo base de funcionamiento es por multiplexión por división de tiempo.

## 2.3 Estado del Arte: Servicios Disponibles en el mercado

Actualmente en el mercado existen variadas soluciones que resuelven problemas similares al que motiva este trabajo. Estas soluciones, sin embargo, no abarcan el problema completo, sino que lo dividen en tres etapas: (1) Emisión de una señal de audio y transmisión inalámbrica hasta un receptor (2) mezcla de todas las pistas de audio recibidas y (3) emisión de la mezcla de audio hacia un receptor que permite el monitoreo de los músicos. A continuación se resumen los principales servicios disponibles en el mercado.

### 2.3.1 Sistemas Inalámbricos

Los sistemas inalámbricos son típicamente usados para enviar la señal desde un transmisor (cuya señal viene de un micrófono) a un receptor, para luego enviar la señal a otro dispositivo analógicamente, típicamente una consola de mezcla, logrando así prescindir de cables y obtener facilidades de interpretación de música en vivo. En su mayoría son sistemas de uno a uno (1 transmisor envía únicamente a un receptor). Actualmente la marca Shure es líder en soluciones de este tipo, por lo cual se usa como referencia para entregar la siguiente información [1].

Las principales características son las siguientes:

- Calidad: Audio digital en 24 bits
- Frecuencia:
    - Se escanea el medio y se encuentra la banda más desocupada para sintonizar el sistema. La sintonización de los nodos es manual.
    - Ancho de banda: 7 MHz, 24 MHz, 36 MHz
    - Rangos: 470-650 MHz.
- Distancia: hasta 90 metros
- Cantidad de receptores: 1, 2 o 4
- Número de sistemas que pueden funcionar simultáneamente: 5 hasta 20
- Latencia: Bajos niveles de hasta 2.9 ms

### 2.3.2 Sistemas de Monitoreo

Los sistemas de monitoreo son utilizados para que los músicos puedan tener retorno adecuado de lo que están ejecutando todos los músicos, así como también proporcionar aislación del ruido exterior. Para esto un receptor es conectado a la consola de mezcla, donde se ingresa la señal que el músico desea escuchar (por ejemplo la mezcla de todos los músicos más lo que él mismo está tocando) y esta es enviada a un receptor, el cual está conectado a audífonos in-ear para que el músico pueda escuchar. Al igual que en a los sistemas inalámbricos, la marca Shure es líder en el mercado de monitoreo y se toma como referencia para este trabajo [2].

Las principales características de estos sistemas son las siguientes:

- Calidad: Audio digital en 24 bits

- Frecuencia:
    - Se escanea el medio y se encuentra la banda más desocupada para sintonizar el sistema. La sintonización de los nodos es manual.
    - Ancho de banda: hasta 72 MHz
    - Rangos: 470-860 MHz.
- Distancia: hasta 90 metros
- Comunicación uno a uno: Se conecta el transmisor a la consola de mezcla y luego esta envía la señal inalámbricamente hasta los receptores (audífonos in-ear).
- Personalización de la mezcla: El transmisor puede recibir varias entradas, llegando en algunos casos hasta 20, las cuales pueden ser manipuladas por el músico que recibe el sonido desde el dispositivo receptor (por ejemplo subirle el volumen a ciertos instrumentos).
- Latencia: Bajos niveles de hasta 0.7 ms

### 2.3.3 Consolas de Mezcla

Las consolas de mezcla tienen como fin superponer las señales de audio de entrada en una sola señal resultante (mezcla) para luego utilizar esta señal, por ejemplo, para amplificar el sonido de la mezcla completa por un solo set de parlantes o como interfaz para ingresar la señal a un computador. Algunas consolas de mezcla poseen otras características como *Phantom Power* (energizar micrófonos activos), *Panning* (enviar una señal a la izquierda o derecha en un sistema estéreo), balance de volúmenes de las entradas, agregar efectos, etc.

En el rubro de consolas digitales (que es lo que interesa para este trabajo), Yamaha es una de las marcas líderes en el mercado y se utiliza como referencia [16].

Las principales características se muestran en la siguiente tabla:

| **Tipo** | **Latencia [ms]** | **Entradas** | **Phantom** | **Ruido** | **Software** | **Conversión Análogo/Digital** |
|---|---|---|---|---|---|---|
| **Digital** | 2.5 | 8, 16, 24 | Sí/No | No | Sí | Sí |
| **Análogo** | ~0 | 8, 16, 24 | Sí/No | Posible | No | No |

*Tabla 1: Atributos de consolas de mezcla [16]*

# 3 OBJETIVOS Y METODOLOGÍA

## 3.1 OBJETIVO GENERAL

Diseñar un sistema inalámbrico para música en vivo, que permita la transmisión de las diferentes fuentes de sonido para ser mezcladas y retornadas, también inalámbricamente, a los músicos involucrados. El sistema deberá cumplir con requisitos estrictos de latencia que hagan imperceptible el retardo a los músicos que reciben el sonido mezclado y deberá asemejar el desempeño de productos comerciales actuales, pero basándose en tecnologías abiertas y estándares libres, para lo cual se demostrará el desempeño del sistema propuesto mediante simulaciones.

## 3.2 OBJETIVOS ESPECÍFICOS

1. Determinar los requerimientos para que el sistema propuesto tenga retardo imperceptible al oído humano una vez las fuentes de sonido han sido transmitidas, mezcladas y retornadas a los músicos.
2. Diseñar la arquitectura del sistema y el protocolo de capa de enlace que permita la comunicación inalámbrica para transmisión de datos en tiempo real.
3. Validar la factibilidad técnica de la solución propuesta en un entorno inalámbrico mediante la implementación y simulación en OMNeT++ del mecanismo propuesto.
4. Elaborar un análisis de desempeño para determinar si el sistema propuesto cumple con los requerimientos establecidos.

## 3.3 ANTECEDENTES GENERALES

El presente trabajo se sitúa en el contexto de las redes de comunicaciones inalámbricas. En la actualidad, un gran número de necesidades se han suplido mediante la implementación apropiada de distintas redes inalámbricas, tales como la Internet de las cosas, las redes ad-hoc o redes inalámbricas que aseguren calidad de servicio, como es el caso de este trabajo relacionado con transmisión de música en vivo. Este trabajo presenta un aporte considerable en la medida que muestra cómo el correcto diseño de redes inalámbricas puede entregar una solución satisfactoria a distintos problemas cotidianos del ser humano, alentando a que distintos estudiantes o académicos sigan desarrollando soluciones innovadoras para las distintas necesidades del ser humano.

## 3.4 ANTECEDENTES ESPECÍFICOS

Actualmente el problema de este trabajo se ha abordado de distintas maneras, tanto a nivel académico, como comercial, entregando un buen punto de partida para el desarrollo de este trabajo. Además, se aprecia que en el estado del arte no se resuelve completamente el problema, sino que lo más cercano es una solución modular (soluciones comerciales), de las cuales se

desconoce exactamente la tecnología implementada para lograrlo, lo cual genera otro aporte para este trabajo al generar conocimiento libre al respecto.

Para esto la profesora guía a puesto a disposición variada información respecto a estándares de IEEE y trabajos de investigación relacionados, como lo que se expuso en el estado del arte.

## 3.5 Hipótesis de Trabajo

El problema de obtener retorno con retardo imperceptible para músicos en el contexto de música en vivo, se puede resolver mediante un sistema inalámbrico que tome las señales digitalizadas (mediante micrófonos), las cuales son emitidas por distintas fuentes de audio (tales como amplificadores, voces o instrumentos acústicos), y la transmita mediante un protocolo de acceso al medio que cumple satisfactoriamente los requerimientos de calidad de servicio establecidos según retardo, calidad de señal y cobertura de la red inalámbrica. El sistema inalámbrico puede ser diseñado con base en tecnologías de fácil acceso y estándares libres, más aportes realizados por el autor de este trabajo.

## 3.6 Alcances

El presente trabajo está acotado a lo siguiente:

1. Utilizar como herramienta de trabajo estándares libres y tecnologías de fácil acceso.
2. Utilización de la banda de frecuencias libre según ISM.
3. Diseñar la arquitectura de capa de enlace del sistema.
4. Diseñar la arquitectura de capa de red del sistema considerando como máximo 16 nodos emisores y 10 receptores.
5. Ajustar parámetros y determinar condiciones que deben cumplirse para obtener el throughput deseado en el rango definido para un alcance de 15 metros, con los retardos apropiados.
6. Validar el modelo mediante simulación en el entorno de programación OMNet++

Por otro lado no se considerarán en este trabajo los siguientes aspectos:

1. Especificaciones del hardware involucrado, como micrófonos y audífonos. Se asume que la señal de todos los nodos emisores es igual y estándar.
2. Diseño de sistemas de baterías necesarios para la alimentación de los distintos dispositivos (nodos) involucrados.
3. Condiciones necesarias para el correcto funcionamiento cuando se implementa este tipo de red en habitaciones contiguas.
4. Condiciones necesarias para el correcto funcionamiento en presencia de obstáculos e interferencia.

## 3.7 INFRAESTRUCTURA DISPONIBLE

Este trabajo solo requiere de recursos intelectuales y computacionales. Los recursos computacionales constan del entorno de trabajo OMNeT++, el cual permite simular de manera realista distintas arquitecturas de red y permite implementar distintas bibliotecas, como por ejemplo 802.11, y desarrollar sobre estas lo necesario. Se utilizará un computador personal para este propósito.

# 4 SOLUCIÓN PROPUESTA

La red debe tener una arquitectura como la mostrada en la figura 3. En esta se aprecian claramente dos partes, la parte (a) que contiene a los nodos que envían la señal de audio al AP/consola y la parte (b) donde se encuentran los nodos que reciben la señal de audio mezclada por la consola y enviada por el AP. La figura 3 no representa la ubicación geográfica de los nodos y el AP ya que las fuentes de sonido y los receptores del sonido mezclado pueden estar ubicados en el mismo sitio (por ejemplo, un músico está generando sonido de audio y al mismo tiempo está recibiendo el retorno con su voz y todos los instrumentos mezclados).

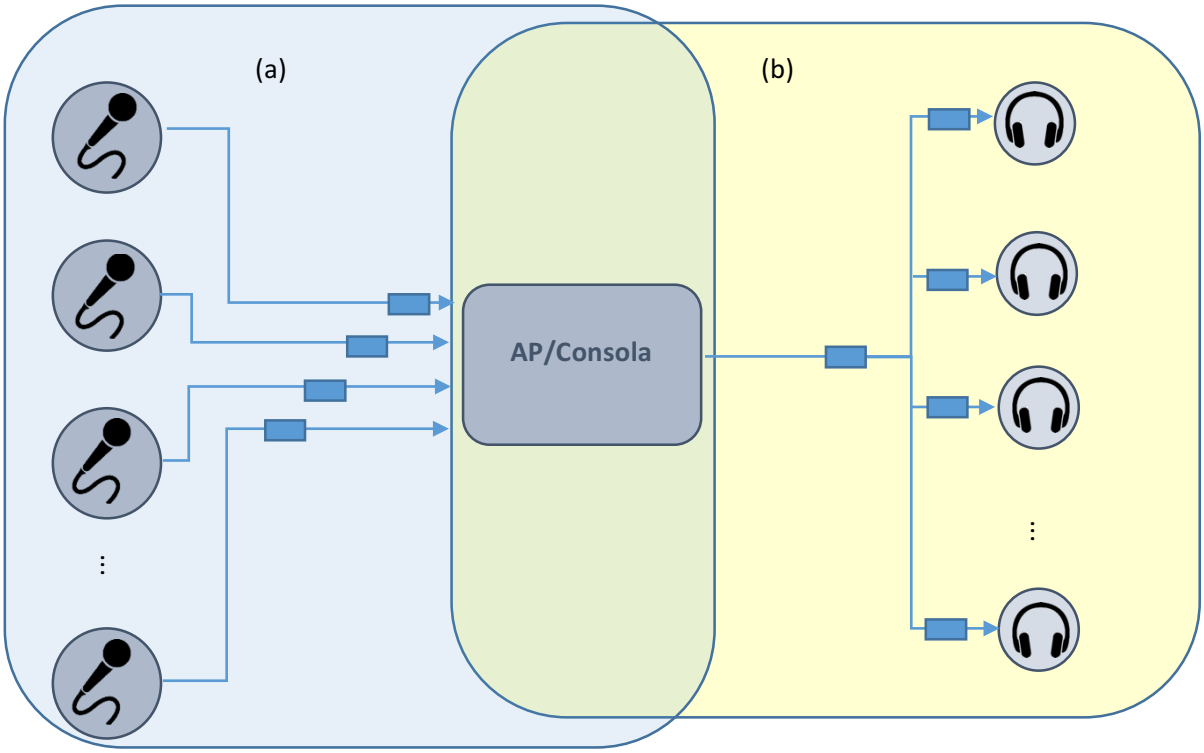


*Figura 3: Diagrama de la red. (a) representa la parte de la red que implementa TDMA y contiene a los nodos transmisores, que corresponden a micrófonos que emiten señales de audio con muestreo 44.1 kHz a 16 bits por muestra hacia la consola. (b) representa la parte de la red que consiste en la emisión de la señal de los emisores mezclada hacia cada uno de los nodos receptores, correspondientes a audífonos que reproducen el sonido mezclado. El AP/consola cumple tres funciones (1) receptor de las señales de (a) (2) mezclador de todas las señales y (3) emisor de la mezcla en (b)*

Para lograr dar un buen uso del canal y evitar totalmente el problema de colisiones, se propone utilizar un método híbrido de multiplexión entre FDM y TDM, siendo la división FDM para diferenciar a los nodos de (a) y los de (b), de acuerdo a la figura 1, mientras que TDMA se utilizará para la sección (a) de la red para compartir el canal entre todos los nodos involucrados, quienes necesitan enviar su señal de audio a la consola independientemente. Por otro lado, los músicos necesitan escuchar el mismo retorno cada uno (correspondiente a la mezcla de todos los instrumentos musicales que

están participando), por lo tanto la sección (b) no requiere multiplexión entre sus nodos y en consecuencia se transmitirá la mezcla de audio desde la consola mediante broadcast.

Como se mencionó anteriormente, la comunicación debe ser inalámbrica y debe cumplir con los requisitos principales, que de acuerdo a los modelos de capas son: (1) capa superior que se encarga de una correcta comunicación en este contexto, los cuales son throughput (relacionado con la calidad de la señal y ancho de banda requerido) y retardos permitidos para una correcta coordinación entre músicos; (2) capa de enlace o de control de acceso al medio (MAC, por sus siglas en inglés) que se encarga de controlar el acceso al medio de todos los miembros de la red, implementando los protocolos necesarios para que esto ocurra según las necesidades de la aplicación final; y (3) capa física que se encarga de entregar el soporte físico (a nivel de ondas electromagnéticas, cables transmisores y otros) a la comunicación que se establece. Es así que el trabajo se concentra principalmente en la capa de enlace, sin intervenir esfuerzos en aspectos de capas superiores ni en la capa física (específicamente en el radio transmisor/receptor ya que se usará como base lo existente en IEEE 802.11).

## 4.1 Requerimientos del Sistema

1. Calidad de sonido: A nivel de audio digital, se considera como alta definición cuando la señal de audio está muestreada al menos a 44.1 kHz y está cuantificada con al menos 16 bits por muestra. Esto establece una cota inferior para lograr calidad de alta definición de audio y se usará como referencia.
2. Cantidad de nodos: Como parámetro de diseño, se utilizan 16 nodos emisores como máximo en la red, esto fue elegido pensando en un mínimo requerido para aplicaciones de sonido y podría ser modificado en alguna implementación en el futuro (tomando en cuenta que su modificación afecta directamente a otros parámetros de diseño como time slot, time frame, throughput y retardo).
3. Throughput: Ya que estamos trabajando con señales digitales, se tiene que para lograr la alta definición mencionada anteriormente es necesario tener una tasa de transmisión de 705,6 kbps para una fuente de sonido. Como se ha definido que la red tiene máximo 16 fuentes de sonido se requiere un throughput máximo de 11,29 Mbps únicamente para enviar las señales de audio requeridas. La red puede requerir un throughput mayor, dado que pueden existir ineficiencias en la ocupación del canal por efecto de la tecnología TDMA o por agregar información de control. Afortunadamente, se sabe que IEEE 802.11g e IEEE 802.11n se logran velocidades de transmisión de hasta 54 Mbps y superiores a 100 Mbps respectivamente, y teniendo en cuenta que su tecnología de acceso al medio, CSMA/CA, se pretende reemplazar por TDMA, se espera alcanzar el throughput deseado sin problemas, puesto que el manejo de colisiones es un factor que disminuye bastante el throughput.
4. Latencia: En el rubro del audio y sonido, se sabe que el cerebro humano al detectar un mismo sonido con latencia, puede distinguirlo como dos sonidos distintos si es que ambos sonidos tienen un desfase en el tiempo igual o superior a 100 ms, generando lo que se conoce como eco. Para valores menores el cerebro interpreta esto como reverberación, sin

embargo, en el rubro de la música se sabe que hay ciertos niveles de reverberación que pueden llevar a los músicos a descoordinarse entre sí [3], [4]. Un nivel aceptable de reverberación, que no afecta la coordinación de los músicos al escuchar su propio sonido se establece en los 30ms, la cual será la cota superior aceptable para el diseño de este trabajo. Sin embargo, dado que el contexto de aplicación de este trabajo es para una presentación musical en vivo, la latencia se percibe doble. Para esto imagínese a modo de ejemplo que el percusionista dé una señal de partida, luego el resto de los músicos la percibirán con un retardo, digamos de 15 ms, y reaccionarán comenzando a interpretar una canción. El percusionista por su parte percibirá la respuesta 30ms retardada desde que él comenzó a tocar. De este modo, la cota superior para la latencia es de 15ms, en los cuales debe estar considerada la latencia que genera la consola de mezcla (se puede aproximar como 2,5 ms) dejando un margen de 12,5 ms para la latencia obtenida en el envío de los nodos al AP (parte (a) de la figura 3) junto a la del envío del AP a los nodos (parte (b) de la figura 3).

5. Utilización del medio: Para este trabajo se hace de vital importancia el contar con comunicación inalámbrica, pues en el contexto musical ya existen muchos cables involucrados y estos dificultan la movilidad de los músicos e incluso ponen en riesgo la comunicación y a los mismos músicos, en caso de accidentes por tropiezo o enredo. De este modo, los requisitos recién mencionados deben ser aplicables en el contexto de comunicación inalámbrica.
6. Dimensión de la red: Las distancias involucradas en esta red, dado que está orientado para música en vivo, tienen como cota máxima 15 m desde un nodo cualquiera al AP.

## 4.2 Abstracciones y Simplificaciones

Para la realización de este trabajo se considera lo siguiente:

1. El canal se encuentra en condiciones suficientes para el correcto funcionamiento de la red
2. Dadas las pequeñas dimensiones de la red y los retardos involucrados, se considerará que los nodos tienen posición fija a lo largo del tiempo y que esto no compromete la credibilidad de los resultados obtenidos, ya que los retardos en el tiempo involucrados son mínimos pues solo afectan al tiempo de propagación, lo cual es del orden de $10^{-8}s$. Además, esto siempre es cierto excepto para los micrófonos de algunos cantantes, quienes pueden desplazarse en el escenario (los otros micrófonos están fijos en los amplificadores o atriles). Las latencias por transmisión del sonido son del orden de $10^{-5}s$ dado que los micrófonos se encuentran a escasos centímetros (típicamente desde 1 cm hasta 5 cm, dependiendo del sonidista) de los distintos parlantes utilizados. Un caso particular son los micrófonos ambientales que se suelen utilizar en baterías a entre 30 cm 80 cm de distancia de los platos. Este caso puede imprimir un retardo del orden de $10^{-3}s$, pero por simplicidad se considerará cero para los cálculos realizados
3. Se asumirá que la consola existe y que para efectos de la red se puede representar por un bloque que recibe entre 1 y 16 señales de audio con las características ya señaladas, imprime un retardo de 2,5 ms (tal como se expuso en la sección 4) y emite una señal (distinta) de audio de las características conocidas hacia el bloque (b) de la red según la

Figura 3. Para una eventual implementación, la tecnología desarrollada en este trabajo debe ser programada en un AP y una consola de mezcla para que puedan implementarla correctamente.

4. Se considera que la información de audio existe y está representada por bits que se montan sobre paquetes generados en la capa superior (ver sección 8).
5. Se asume que los buffers definidos como parte de la solución (ver secciones 6.4.2 y 7) existen, tanto en el dispositivo que se conecta a un micrófono como en las entradas de la consola, y brindan los servicios requeridos para implementar la tecnología propuesta en este trabajo.
6. El formato de paquete utilizado contiene algunos campos de control que se usan para el algoritmo de sincronización únicamente. Para el streaming de audio los paquetes que van desde los nodos emisores al AP y del AP a los nodos receptores llevan únicamente información de audio, por lo cual esto es considerado al momento del cálculo del throughput requerido.

## 4.3 Capas Superiores

Las capas superiores (que frecuentemente se conforman por aplicación, transporte y red en arquitecturas basadas en la pila TCP/IP), que de ahora en adelante serán llamadas simplemente como capa superior, para este trabajo serán una abstracción de una fuente de sonido. En la realidad, el sonido estará siendo emitido desde un micrófono, y se requiere que la capa superior tome esta señal de audio y la deje en formato digital lista para luego enviarla a la capa de enlace. Para efectos de la sincronización, es necesario enviar unos paquetes con una cabecera que tenga información como la ID del nodo al que se está enviando información o algunos campos de control, sin embargo estos solo tienen un rol antes del streaming de audio, por lo que para efectos de throughput se puede considerar sin problemas que el ancho de banda es ocupado solo por datos.

La estructura en detalle de los paquetes se explicará en la sección 7.

## 4.4 Capa de Enlace

Esta es la capa que concentra la mayor contribución de este trabajo. Sus principales aportes son (1) la sincronización de todos los nodos de la red respecto al reloj del AP, para luego poder implementar (2) la transmisión con TDMA para acceso al medio y así soportar el streaming de datos y (3) la re-sincronización, que asegura que todos los nodos permanezcan sincronizados a lo largo de la transmisión.

### 4.4.1 Sincronización

El objetivo de la sincronización es que el AP logre sincronizar a cada uno de los nodos (i.e., fuentes de sonido) de la red. Luego de revisar aportes como los hechos en [15] y [13], se ha propuesto el siguiente algoritmo de sincronización. Primero se necesita estimar el "intervalo de tiempo" entre el AP y cada uno de los nodos de la red, luego se asignar a cada nodo su turno TDMA y se procurar que no ocurran colisiones de paquetes en el canal. A continuación se explica cómo opera la sincronización de la red en más detalle:

(0) Cada nodo tiene su propio reloj interno que no necesariamente es igual que el del AP. Además se asume que cada nodo posee una ID que toma valores entre 1 y 16 y que el AP las conoce. Como condición inicial pueden haber entre 1 y 16 nodos activos. La cantidad de nodos activos deberá determinarla el AP.

(1) El AP envía un mensaje a un determinado nodo esperando una respuesta de este. Esto se repite uno por uno con cada posible fuente de sonido presente en la red, recorriendo las Ids en orden creciente.

(2) (a) Si el nodo que está siendo llamado responde dentro de tiempo de espera establecido por el AP, entonces se procede a sincronizar a ese nodo y se pasa a (1) para sincronizar al siguiente nodo. La sincronización de un nodo con el AP se realiza calculando el tiempo que tarda en viajar un paquete desde un nodo cualquiera hasta el AP (intervalo de tiempo). Esto se logra registrando el tiempo, medido en el reloj del AP, que tarda un paquete de las mismas características de los que se utilizarán en el streaming en salir desde el AP hacia un nodo y que la respuesta correspondiente llegue, dividiendo este resultado por dos. El tiempo resultante es una aproximación que no considera retardo adicional por procesamiento en el receptor o movimiento del nodo durante la sincronización. Las distancias estimadas en tiempo entre el AP y cada nodo se guardan en una tabla de sincronización que solo el AP conoce.

(b) Si el nodo que está siendo llamado no responde dentro del tiempo de espera establecido por el AP, entonces se procede a reintentar la sincronización con un máximo de intentos definidos por el AP. Si se realizan el máximo de intentos y no se logra sincronizar al nodo, se pasa a (1) para sincronizar al siguiente.

(3) Una vez que el AP termina de sincronizar a los 16 nodos, procede a asignar los turnos de TDMA a cada uno de los nodos de la red y les informa a cada uno cuánto tiene que esperar para que comiencen a emitir la transmisión. También se les informa la duración del time slot y del time frame, para que puedan aplicar TDMA de buena manera.

### 4.4.2 Transmisión con TDMA

Se procederá a explicar por etapas los elementos involucrados en TDMA, recordando que un frame se define como se muestra en la Figura 2.

(1) El AP define los parámetros de time slot (tiempo que tiene un nodo para enviar información en su turno dentro de un frame) y el time frame (tiempo que transcurre para que cada nodo envíe su siguiente paquete desde que envió el anterior) de modo que se cumpla que:

$$t_{frame} = t_{slot} \cdot N,$$

Donde $N$ es el número de nodos. También se define el número de frames para re-sincronizar (en base a pruebas empíricas que dependen de la implementación involucrando hardware), y las características de los buffers necesarios para poner en cola los paquetes de audio que vengan de la capa superior antes de que sean enviados al AP y buffers necesarios para encolar los paquetes que van desde el AP a la consola.

El tamaño de los buffers necesarios, tanto para los que están entre una fuente y en el AP como para los que están en cada canal de la consola, se calcula mediante la siguiente expresión

$$buffer = t_{frame} \cdot 710\ kbps$$

Pensando en una cota inferior para que la información que se encola entre un time frame y otro quepa, sin desperdiciar recursos. De este modo cada nodo guarda la información correspondiente al audio generado, desde la última vez que fue su turno para utilizar el canal hasta el presente, la cual será enviada en el siguiente turno que le corresponda. Esto implica que una condición de diseño necesaria es que se cumpla que

$$T_{audio} * t_{frame} + b_{head} \leq C_{canal} * t_{slot}$$

Dondo $T_{audio}$ es la tasa de generación de audio, $t_{frame}$ es un time frame, $b_{head}$ es la cantidad de bits agregados de encabezado o control, $C_{canal}$ es la capacidad del canal y $t_{slot}$ es un time slot. En otras palabras, esto implica que la cantidad de información que cabe en un time slot en contexto de TDMA sea igual o mayor que la información de audio generada durante un time frame por una fuente de audio.

(2) El AP calcula los turnos en que cada nodo enviará su información. Por simplicidad, el orden de envío se asigna según las IDs de cada nodo, de forma creciente.

(3) El AP calcula los tiempos que debe esperar cada nodo según sus relojes propios desde que reciben la notificación del AP de que debe comenzar la transmisión y se los envía a cada fuente de sonido. Este tiempo se define como:

$$t_{ei} = t_{oi} + (i - 1) \cdot t_{frame}$$

Donde $t_{ei}$ es el tiempo que debe esperar el nodo i-ésimo, $i = 1,2 \dots 16$. Dependiendo del nodo y $t_{oi}$ es el tiempo de offset del nodo i-ésimo, el cual pretende compensar las distintas distancias en tiempo entre los distintos nodos y el AP para lograr una buena sincronización. El tiempo de offset se define como:

$$t_{oi} = max\{t_{dk}\}_{k=1}^{16} - t_{di}$$

Donde $t_{di}$ es el intervalo de tiempo entre el nodo i-ésimo y el AP, $t_{dk}$ es el intervalo de tiempo entre el nodo k-ésimo y el AP, calculados en la sincronización. Con esta fórmula para calcular los tiempos de offset se logra que los paquetes que envía cada nodo no colisionen durante la transmisión.

(4) Se procede al streaming, donde cada nodo toma su turno en TDMA según lo informado anteriormente y envía el siguiente paquete luego de que un time frame ha transcurrido.

(5) Para abordar efectos de una eventual des-sincronización por defectos de los relojes internos de los nodos u otros factores, se manda un frame distinto cada cierto número de frames que entrega una referencia del reloj del AP con la cual cada nodo puede aplicar el algoritmo de re-sincronización que se explica en 4.4.3.

### 4.4.3 Re-sincronización

El método de re-sincronización está basado en que, luego de realizada la sincronización, el AP es capaz de comunicarle a cada nodo su reloj como referencia gracias a los intervalos en tiempo

calculadas. De este modo, al transcurrir cierta cantidad de tiempo y asumiendo que el intervalo de tiempo entre el AP y los demás nodos no ha cambiado, cada nodo podrá realizar su pronóstico del reloj del AP según la referencia que manejan luego de la sincronización. Es así que si el AP les envía un paquete donde les comunica a cada nodo el valor que debería tener efectivamente esa predicción, entonces cada nodo podría corregir su propio reloj y programar correctamente el envío de sus siguientes paquetes en el streaming de audio.

# 5 ANÁLISIS DE LATENCIA Y SINCRONIZACIÓN DE LAS DISTINTAS FUENTES DE AUDIO

En esta sección se procede a analizar detalladamente la latencia que tendría un paquete generado por los nodos emisores y, dado que la aplicación es de música en vivo, también se procede a analizar por qué la sincronización del sistema es consistente en el tiempo  al momento de coordinar los envíos de los distintos nodos (instrumentos musicales) como al momento de mezclar los distintos canales de la consola, gracias al diseño propuesto en base a TDMA.

Para facilitar la explicación, se tomará como ejemplo una red de 4 nodos emisores y 1 nodo receptor. Los parámetros de TDMA serán time slot de 0.25 s y time frame de 1 s, ya que en la etapa de streaming (después de la sincronización) se ocupa todo el canal para enviar solamente datos. Se asume que la red ya se encuentra sincronizada.

En primera instancia, se tiene que cada fuente de sonido emite una señal digital de audio con las características señaladas en la sección 6.1 y almacena esta señal en un buffer cuya capacidad en bits está expresada en 6.4.2, tal como se muestra en la figura 4. Recalcando que el buffer tiene una capacidad tal que puede almacenar $t_f$ segundos (la duración de un time-frame), en este ejemplo se guardará un segundo de audio para lograr llenar el buffer. Cada vez que el buffer está lleno la información debe ser enviada a la capa superior y así lograr que no existan pérdidas de información, de modo que uno de los requerimientos del sistema es que TDMA haga coincidir los turnos de cada nodo emisor para enviar su información con los instantes en que se llena el buffer. Como la información solo se envía una vez que el buffer está lleno, se tiene un primer elemento de latencia generado. En este ejemplo, para llenar el buffer hay que generar un segundo de audio primero, por lo cual se añade una latencia de 1 segundo antes de llegar a la capa superior. En general, esta etapa añade una latencia de fuente $l_f = t_{frame}$ segundos (el tiempo de un time-frame).

Por simplicidad y dado que los costos son mínimos, se decidió que durante el primer time frame del streaming puedan haber pérdidas, ya que puede ocurrir, como se muestra en la figura 4, que un nodo tenga su buffer lleno antes de que le toque enviar por primera vez su información según los turnos de TDMA, frente a lo cual se descartan los bits más antiguos del buffer y se mantiene el buffer lleno.

Entonces, ya que cada nodo está enviando la información máxima que cabe en su buffer cuando le toca su turno en TDMA, cada nodo enviará exactamente los últimos $t_{frame}$ segundos de la señal de audio que generó antes de que le tocara su turno en TDMA (respecto al reloj de referencia del AP lo cual se logra con el algoritmo de sincronización) que es la cantidad de información que acumuló en el buffer desde la última vez que le tocó transmitir. De este modo se pretende que en el momento de mezclar el audio en la consola de mezcla la señal de cada canal sea consistente en el tiempo con la de los demás canales, o en otras palabras, que no exista desfase entre la música que se mezcla en cada canal (gracias a la sincronización) y que al mezclar cada canal tenga una señal continua en la consola (gracias a que cada nodo envía la información que cabe en su buffer).

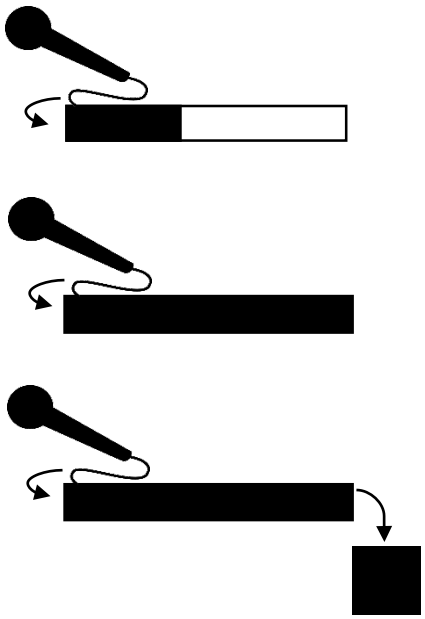

*Figura 4:* Muestra a una fuente de audio y su buffer en tres etapas: antes de que llegue su turno en TDMA, justo antes de que llegue su turno en TDMA y cuando tiene que descartar bits en el caso excepcional del primer time-frame.

Por otro lado, la capacidad de enlace disponible es mayor al utilizado por la red, lo cual implica que en un time slot existirá "espacio vacío", tal como se puede apreciar en la figura 5. Por ejemplo, si la capacidad de enlace disponible es de 24 Mbps, se tiene que el tiempo de bit es de 41,7 ns, con lo cual la expresión para la cantidad de bits que caben en un time slot está dada por

$$maxbits_{slot} = t_s / t_{bit}$$

Donde $t_s$ es el tiempo del time slot y $t_{bit}$ es el tiempo del bit según la capacidad del enlace disponible. Luego, es necesario contrastar esta cantidad con la cantidad de bits que cada nodo necesita poner en un time slot según la tasa de transmisión de audio, lo cual viene dado por

$$bits_{slot} = T_{audio} * t_{frame}$$

Donde $T_{audio}$ corresponde a la tasa de transmisión de audio y $t_{frame}$ es el tiempo de un time frame. La razón por la que se utiliza un time frame para calcular la cantidad de bits que van en un time slot es que, en el turno de un determinado nodo, este debe enviar la información que acumuló desde la última vez que fue su turno de enviar, es decir un time-frame atrás. Solo así se puede lograr que al

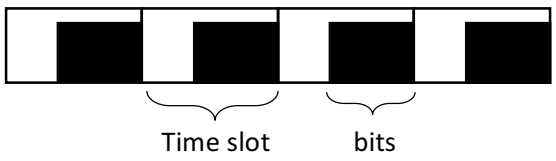


*Figura 5:* Estructura de un frame TDMA. Se muestra para una red con 4 nodos, donde se logra apreciar que sobra espacio en cada time-slot debido a que la capacidad del enlace es mayor a la requerida.

generar la mezcla de audio la señal de cada fuente de sonido sea reconstruida íntegramente (sin cortes de sonido). Esto también establece el tamaño mínimo que debe tener el buffer para que el diseño propuesto cumpla su cometido. Para efectos del ejemplo que se está utilizando, $maxbits_{slot} = 6\ Mb$ y $bits_{slot} = 0.71\ Mb$ observando que queda capacidad sin utilizar en esta configuración, tal como se esperaba.

La estructura de un time frame que se aprecia en el figura 5, que cuenta con capacidad disponible en cada time-slot incluso para la configuración máxima de nodos de la red, ya que 16 nodos utilizan menos de 13 Mbps y el canal soporta hasta 54 Mbps, entrega la ventaja de poder enviar información de control utilizando esta capacidad sin alterar la duración del frame. Esto será utilizado para enviar información necesaria para el algoritmo de re-sincronización, el cual solo necesita unos pocos bits de información y puede ejecutarse tal como se expuso en la sección 4.4.3.

Una vez la información es recibida por el AP, este envía los datos a la consola de mezcla. Como se dijo anteriormente la información de los distintos canales está correctamente sincronizada a pesar de que llega a la consola en distintos instantes. En la figura 6 se puede apreciar cómo llega la información a la consola separada por canales y se almacena en buffers del mismo tamaño que los de las fuentes de audio. En la figura 6a se aprecia la situación que ocurre cuando llega el primer paquete correspondiente al primer time-slot de toda la transmisión, en 6b se aprecia cuando llega el segundo y 6c cuando llega el cuarto y último (en este ejemplo) paquete del primer time-frame. Durante el primer time-frame existe pérdida de información en los canales 2 hasta el último, siendo el último canal el que pierde más información, correspondiendo a $t_s * (N - 1)$ segundos de audio (en este ejemplo se pierden 0.75 segundos para el canal 4). La figura 6d muestra cómo se encuentran los buffers de la consola al momento de comenzar el segundo time-frame. Desde este instante en adelante, no hay ningún tipo de pérdida de información y los buffers de cada canal siempre tienen señal de audio que mezclar, todo gracias al cálculo del tamaño de los paquetes que se envían en un time-slot.

Respecto a las latencias, se sabe que existen consolas en el mercado que añaden retardo de no más de $2.5\ ms$, tal como se indicó en la sección 4.3.3, por lo cual se añade la latencia de la consola $l_c = 2.5\ ms$. Por otro lado, existe una latencia para cada nodo emisor que se añade según el tiempo de propagación que toma en llegar al AP, y como las ondas electromagnéticas en el aire viajan a una velocidad cercana a 300.000 km/s y las dimensiones de la red son de máximo 15 m de distancia el tiempo de propagación es prácticamente cero. Por otro lado, como se explicó en 4.2, los retardos debido a la velocidad del sonido se consideran 0 para los cálculos puesto que por lo general el retardo asociado es muy pequeño por la corta distancia que el sonido viaja en el aire. Los tiempos de transmisión dependen de $bits_{slot}$ y la capacidad del canal, siendo en este caso $t_{tr} = \frac{bits_{slot}}{capacidad_{canal}} = 0.0295$ , resultado que depende directamente de la duración del time-frame.

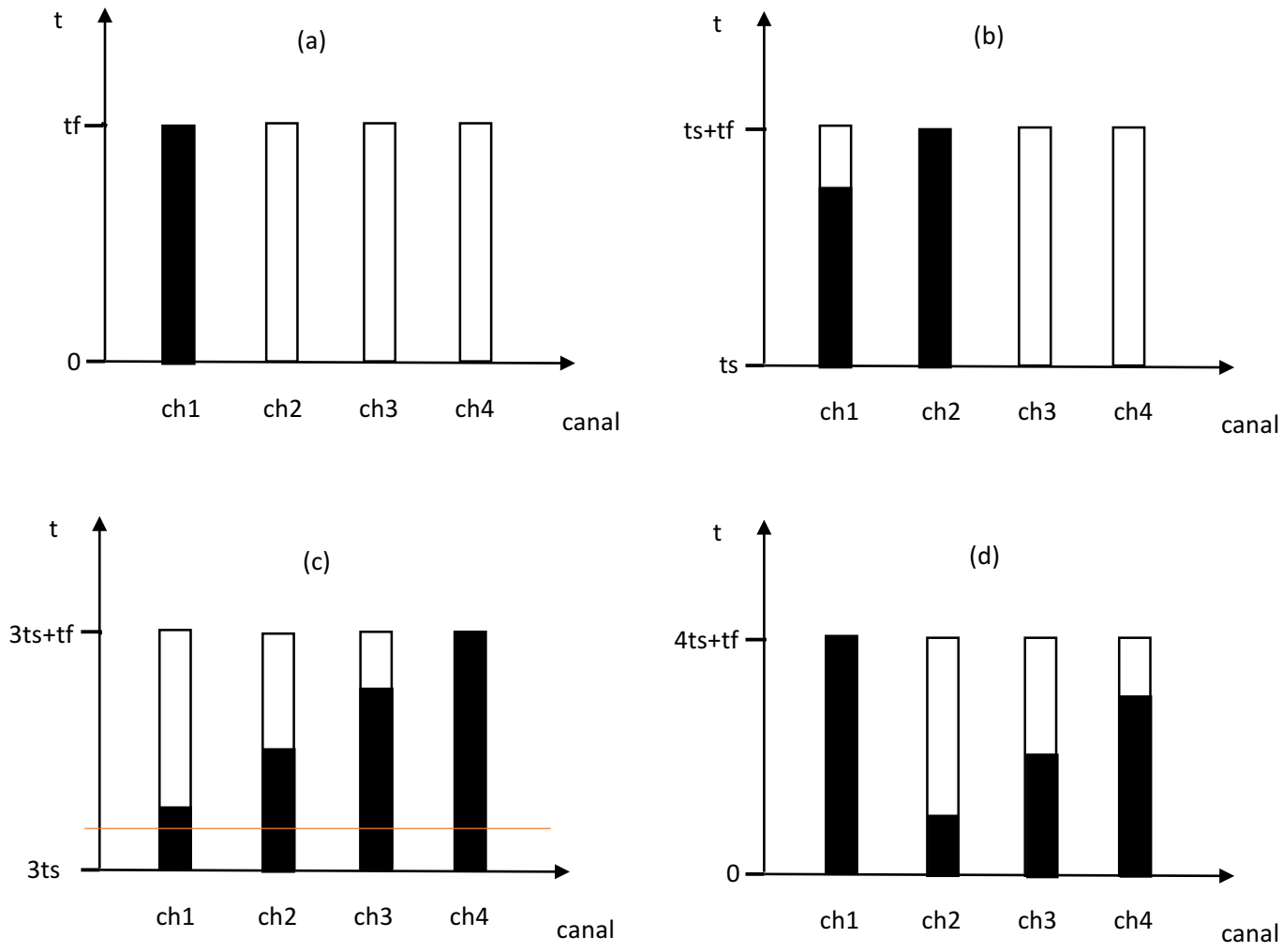


*Figura 6:* Se muestran los buffers de los canales que ingresan a la consola de mezcla. (a) muestra cuando llega el primer paquete de toda la transmisión, (b) muestra cuando llega el segundo paquete de toda la transmisión, (c) muestra cuando se completa el primer time frame de la transmisión y (d) muestra cuando llega el primer paquete del segundo time frame, donde el sistema entra en régimen permanente. La línea roja que se muestra en (c) permite identificar que los bits de los 4 canales fueron generados al mismo tiempo en las fuentes de audio, respecto a un mismo reloj de referencia.

Por otro lado pueden existir retardos por tiempos de procesamiento, pero como el sistema diseñado no involucra procesamiento significativo también se estima que el tiempo de procesamiento es cercano a cero.

Finalmente, la expresión para calcular la latencia total del sistema corresponde a:

$$l_{total} = l_f + t_{tr1} + l_c + t_{tr2}$$

Donde $l_f$ es la latencia añadida por la fuente, $t_{tr1}$ es el tiempo de transmisión entre las fuentes y el AP $l_c$ es la latencia añadida por la consola y $t_{tr2}$ es el tiempo de transmisión entre el AP y un nodo receptor. De este modo, para el ejemplo utilizado se tiene el siguiente valor para la latencia:

$$l_{total} = 1s + 0.0295s + 0.0025s + 0.0295s = 1.0615$$

Se observa que el parámetro que se puede manipular con mayor libertad es $l_f$, lo cual también influye en $t_{tr}$ y se utilizará posteriormente para lograr una configuración satisfactoria del sistema. Por último, es importante notar que $l_f$, $l_c$, $t_{tr1}$ y $t_{tr2}$ son constantes para cada nodo dado que los paquetes de datos enviados por cada nodo son de tamaño constante durante el streaming. De hecho, el único parámetro variable entre un nodo y otro es el tiempo de propagación, que como se mencionó anteriormente se aproxima a cero.

# 6 IMPLEMENTACIÓN EN OMNET++

Una vez que se comprende la solución a nivel conceptual sigue la implementación realizada para validación de desempeño, de lo cual trata esta sección. En este caso, dado que se trabaja en el entorno OMNeT++ [17], se genera una arquitectura particular basada en programación orientada a objetos ya que la lógica está escrita en lenguaje C++. Adicionalmente, hay conceptos propios de OMNeT++ que son programados en su propio lenguaje de programación y que también serán explicados a continuación. Los archivos generados para el diseño de la red, los nodos y la lógica involucrada pueden encontrarse en los anexos.

## 6.1 LA RED

La red es un concepto propio de OMNeT++, donde se define una red como un conjunto de módulos que se comunican entre sí y un canal que soporta la comunicación. Esto está escrito en lenguaje .ned y en este caso incorpora a todos los nodos que forman parte de la red, como se muestra en la figura 7. Específicamente, cada nodo (emisores, receptores y AP) corresponden a módulos compuestos, los cuales a su vez se componen de módulos simples.

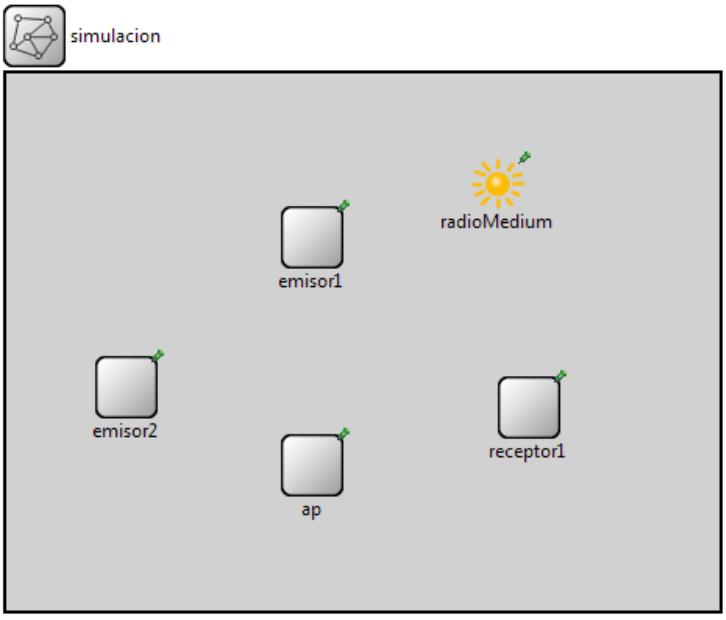


*Figura 7:* Visualización de la arquitectura de la red según el lenguaje .ned.

En la red pueden establecerse algunos parámetros iniciales para los módulos involucrados. En este caso se especifican, por ejemplo, las IDs de cada nodo.

Para este trabajo, se implementó una red con que soporta hasta 16 nodos tipo nodoTDMA (ver sección 6.2) que corresponden a emisores de información (pensando en una consola estándar de 16 canales), otro nodoTDMA configurado como AP, un nodoSimple utilizado como un segundo AP que transmite el broadcast en una frecuencia distinta a la de los emisores y receptores y un número

libre de nodos tipo nodoSimple (explicado en la sección 6.2) que recibe la información desde la consola (pasando por el segundo AP) mediante un broadcast. Se usa solo un receptor ya que para medir desempeño de la red solo es necesario un nodo ubicado lo más lejos posible, porque la comunicación en esta sección de la red es mediante broadcast.

## 6.2 Los Nodos

Los nodos están representados por módulos compuestos, los cuales están conformados por módulos simples. Los módulos compuestos definen algunos de sus parámetros, los cuales se pueden manipular desde el archivo correspondiente a la red, y pueden asignar valor a los parámetros de los módulos simples que lo componen. Es así como por ejemplo la red puede especificar una ID para cada nodo, y luego la capa superior puede acceder a este parámetro si es que así se especifica en el archivo .ned del nodo. En la figura 8 se puede apreciar un nodoTDMA, estructurado por capas.

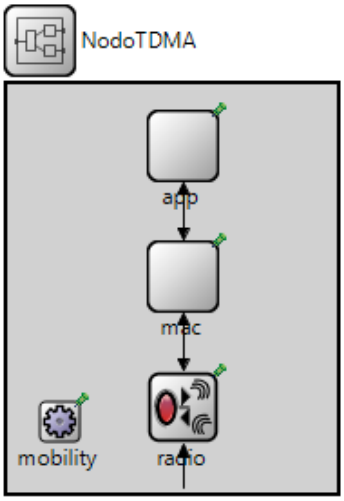


*Figura 8:* Visualización de un nodoTDMA según el lenguaje .ned

Un elemento muy importante es que los módulos compuestos pueden definir puertos y conexiones entre puertos. Los puertos corresponden a representaciones por donde puede entrar o salir información, según se especifique. Las conexiones son uniones entre puertos que se pueden explicitar en el archivo .ned del nodo. Como ejemplo en base a la figura 8, si se envía algo por el puerto de salida de la capa mac que está conectado con el radio, el mensaje automáticamente llegará al puerto de entrada del radio sin posibilidad de otra opción, pues así están establecidas las conexiones.

Para este trabajo se utilizan dos tipos de nodos: nodoTDMA, que aparece en la figura 8 y que se utilizan para representar a los nodos emisores y el AP, y nodoSimple, que representa a los nodos receptores y se componen solamente de la capa física y una capa superior que recibe los datos de la señal de audio emitida desde la consola y los ensambla para que esta pueda ser reproducida.

## 6.3 CAPA FÍSICA

Para este trabajo, la capa física es reutilizada del paquete INET [18], específicamente de los módulos que se utilizan para 802.11 como el radio que se aprecia en la Figura 8, pues satisface plenamente los requerimientos para este trabajo. Es por esto que no se entrará en mayores detalles al respecto, sin embargo es pertinente decir que tiene una complejidad considerable y puede ser revisada, en caso de que el lector lo desee, de la misma fuente original, pues es código abierto.

Los principales servicios que se obtienen de la capa física son:

a. Simulación realista de un canal inalámbrico.
b. Capacidad de emitir y recibir inalámbricamente información mediante antenas modeladas en INET.
c. Modulación y codificación de acuerdo a los estándares IEEE 802.11.
d. Configuración de la frecuencia de la señal portadora de acuerdo a IEEE 802.11
e. Configuración del ancho de banda según estándar IEEE 802.11.
f. Compatibilidad con otros módulos de OMNeT++ de acuerdo al modelo de capas, en particular con la capa MAC-TDMA diseñada en este trabajo.

## 6.4 CLASES INVOLUCRADAS PARA NODOS QUE SOPORTAN TDMA

Antes de revisar el flujo de paquetes e información, se introducirán todas las clases involucradas y sus principales funciones y características. Primero que todo, es necesario aclarar que toda la lógica detrás de este trabajo se encuentra escrita en leguaje C++, donde se definen clases. Adicionalmente algunas clases tienen asociado un código .ned, como es el caso de la representación de las capas de aplicación o MAC-TDMA, ya que al estar representadas en lenguaje .ned pueden ser utilizadas como sub-módulos para formar módulos compuestos (nodos) y posteriormente para diseñar una red y realizar simulaciones.

Las clases por definición poseen miembros, los cuales pueden representar parámetros (variables), algunos objetos (instancias de otras clases) y métodos (funciones que realizan alguna acción). Los métodos fundamentales son initialize(·) el cual actúa por defecto en todos los objetos involucrados cuando se inicia una simulación, y handleMessage(·) el cual actúa cada vez que un mensaje es recibido en un objeto. Lo anterior es fundamental, pues la simulación solo puede continuar si hay mensajes programados (eventos), y es en este método que se programan todos los mensajes correspondientes.

A continuación se detallan las clases involucradas y sus miembros principales.

### 6.4.1 Aplicación

Esta clase representa la capa superior de un nodoTDMA. Su objetivo es iniciar la comunicación de la red y, luego de que la red está sincronizada generar el tráfico (representando una señal de audio) que será enviado apropiadamente hacia la capa MAC-TDMA.

Método initialize(·): se encarga de inicializar las variables y objetos correspondientes y enviar un mensaje a la capa MAC-TDMA para que comience con el algoritmo de sincronización, esto último solo si el nodo corresponde a un AP.

Método handleMessage(·): se encarga de, una vez que esté lista la sincronización, enviar paquetes hacia la capa MAC-TDMA con solamente información correspondiente a una señal de audio, con el tamaño (en bits) adecuado y con la periodicidad establecida por TDMA. Para esto cada vez que se envía un paquete hacia la capa MAC-TDMA, se programa un auto-mensaje con el retardo adecuado, para que cuando este auto-mensaje llegue se envíe el siguiente paquete hacia la capa inferior.

### 6.4.2 MAC-TDMA

Esta clase se encarga de la mayor parte del trabajo necesario para establecer la comunicación en la red. Sus principales funciones son: (a) Inicializar los objetos necesarios para lograr establecer la comunicación en la red; (b) Sincronizar a todos los nodos de la red; (c) establecer las condiciones y estructuras necesarias para sostener la comunicación mediante TDMA; y (d) gestionar el correcto flujo de información entre los nodos y el AP, para luego enviar toda la información a la consola.

Esta clase es de complejidad mayor y cuenta con bastantes variables y métodos, por lo que a continuación se explicará lo esencial de su funcionamiento. Para más detalles puede consultar el código completo en el anexo.

El algoritmo de sincronización está implementado con distintos métodos que son llamados desde handleMessage(·), a continuación se explican los principales elementos:

Si es AP:

- Una vez que llega el mensaje de inicio de la capa superior, se procede a sincronizar al primer nodo con el método sincronizarNodo(·)
- Si llega la respuesta de algún nodo emisor al mensaje de sincronización enviado anteriormente, entonces se cancela el mensaje de reintentar sincronización programado, luego se llama a actualizarTablaSinc(·) que guarda el intervalo de tiempo entre el AP y el nodo a sincronizar y lo guarda en un arreglo. A continuación se llama al método sincronizarSgteNodo(·)
- Si llega un mensaje pidiendo que se reintente sincronizar a algún nodo, se procede a reintentar siempre y cuando no se haya sobrepasado el número máximo de intentos, de lo contrario se llama a sincronizarSgteNodo(·).
- Si llega un mensaje indicando que la sincronización está lista, se llama al método informarTurnos(·)
- Si llega un paquete con datos de audio se lo entrega a la consola y actualiza contadores de paquetes enviados y recibidos.

Si no es AP:

- Si llega un mensaje de sincronización, se llama al método responderSincronizacion(·).

- Si llega un mensaje informando los turnos correspondientes según TDMA, entonces se programa el primer mensaje de streaming de datos de audio enviando un mensaje a la capa superior.
- Si llega un paquete desde la capa superior con datos, se lo envía al AP.

Además es necesario explicar algunos métodos para comprender cómo la implementación se relaciona con la solución explicada en la sección 6.

- sincronizarNodo(·): Envía un mensaje de sincronización a todos los nodos, indicando a qué nodo se quiere sincronizar. Se registra en timeStamp el tiempo en que se envía ese paquete para poder calcular el intervalo de tiempo correspondiente. Finalmente se programa un mensaje para reintentar la sincronización, el cual está pensado en ser cancelado si se logra sincronizar correctamente.
- actualizarTablaSinc(·): estima el intervalo de tiempo entre el AP y el nodo a sincronizar tomando la resta simple entre el tiempo actual y el timeStamp. El resultado lo divide por dos y lo guarda en la tabla de sincronización.
- sincronizarSgteNodo(·): Actualiza la variable de estado que indica a qué nodo se está sincronizando, y procede a llamar a sincronizarNodo(·). Si ya se sincronizaron todos los nodos, entonces envía un mensaje indicando que la sincronización está lista.
- informarTurnos(·): de acuerdo a lo explicado en la sección 6, se procede a calcular los turnos en TDMA de cada nodo y los tiempos de offset que le corresponden a cada uno, luego se guarda este arreglo de tiempos en un tdmaPacket y se procede a enviarlo a todos los demás nodos para poder así comenzar con el streaming de datos de audio.
- responderSincronizacion(·): responde al mensaje de sincronización enviado por el AP.

En la figura 9 se puede aclarar totalmente cómo interactúan los distintos nodos de la parte (a) de la red según la figura 3 y cómo se llaman los métodos recién explicados a través del tiempo, pensando en el caso de una red de 3 nodos emisores donde no hay intentos de sincronización fallidos.

Para obtener más detalle acerca de la implementación de los métodos expuestos se pueden revisar los códigos adjuntos en el anexo.

### 6.4.3 Radio

La clase radio representa la capa física del nodo y es propia de INET, por lo que no será detallada en esta ocasión, pero cumple las funciones explicitadas en la sección 6.3. Como condición inicial de la simulación se le indican los parámetros necesarios para la comunicación, tales como ancho de banda y frecuencia portadora.

### 6.4.4 TDMA-packet

Esta clase es particular, ya que es del tipo “mensaje”. El objetivo de esto es que los distintos objetos puedan comunicarse entre sí, con las facilidades para encapsulado y entregar formatos correspondientes a cada aplicación. En esta ocasión, tdmaPacket representa un mensaje que cuenta con los siguientes atributos, utilizados solo para control:

a. timeStamp: Es heredado de cMessage y permite guardar libremente un tiempo de simulación en el momento que se desee.
b. msgKind: Es un int de uso libre heredado de cMessage. Se utiliza para guardar la id del nodo de destino en el contexto de sincronización.
c. bitLength: Campo que establece el tamaño en bits del paquete. Heredado de cMessage.
d. duracionFrame: Es el tiempo que dura un frame.
e. inicioTransmision: Tiempo en que inicia la transmisión utilizado como referencia para resincronizar.
f. Offset: vector que contiene el tiempo de offset que cada nodo debe esperar para empezar la transmisión, de acuerdo a TDMA.

Así mismo, existen métodos para leer y modificar los valores de estas variables.

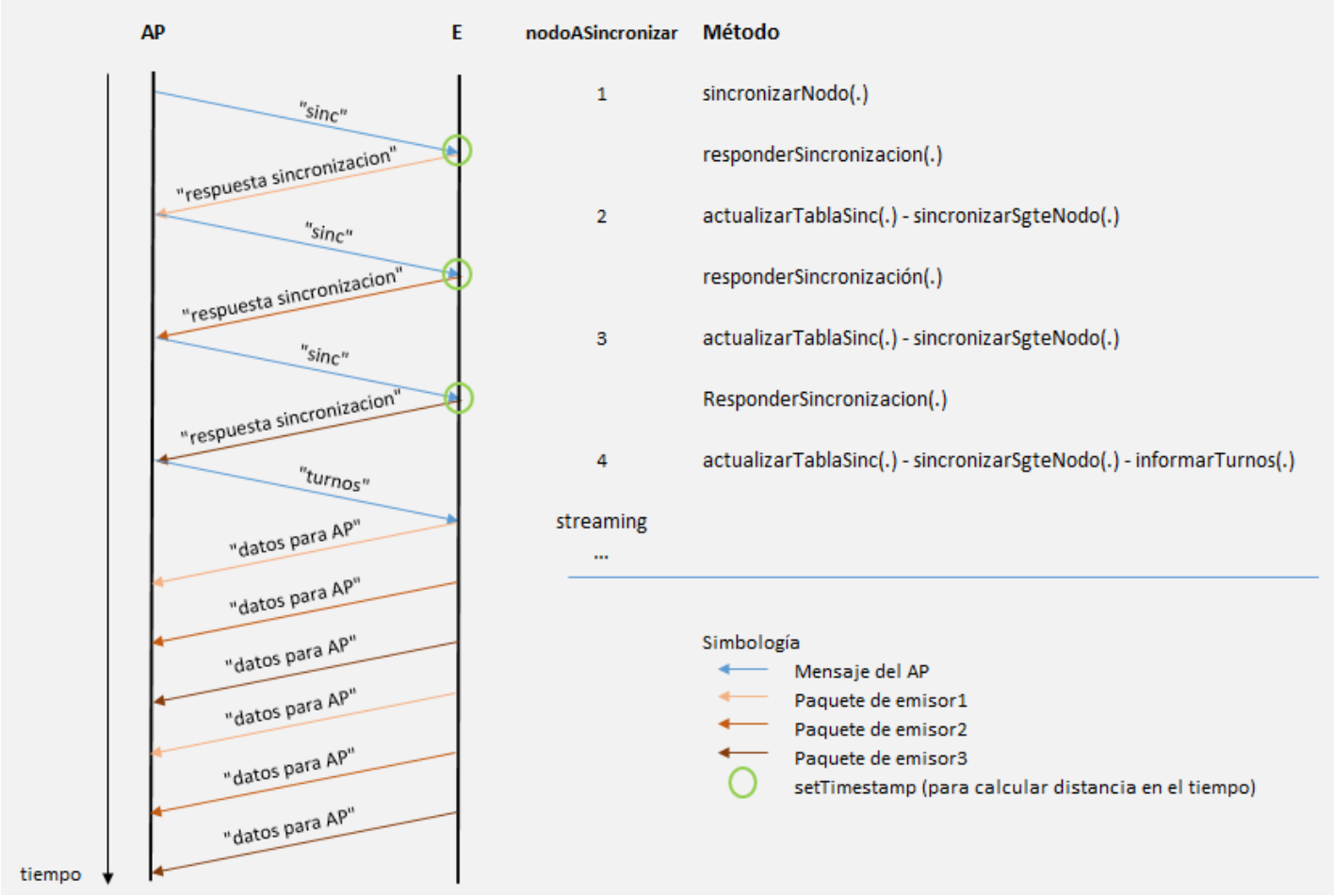


*Figura 9:* Red con un AP y 3 nodos emisores que muestra el flujo de paquetes e interacción de los nodos de la red respecto a la capa MAC.

## 6.5 Clases Involucradas Para Nodos Simples

### 6.5.1 NodoSimpleLogica

Esta clase representa la capa “todo en uno” de los nodos simples, la cual cumple principalmente funciones de aplicación y de acceso al medio.

El principal método es handleMessage(·), que tiene dos comportamientos dependiendo si es AP o no.

- Si es AP: si llega un mensaje con el nombre “procesar mezcla”, entonces se programa un mensaje a sí mismo que diga “enviar mezcla”, el cual llegará en 2.5 ms para simular el retardo de la consola. Si llega un mensaje con el nombre “enviar mezcla” entonces envía un paquete con el nombre “mezcla” con datos de audio por la interfaz radio, de acuerdo al tamaño de bits que debe tener un time slot (señalado en la sección 5) y a continuación programa un mensaje a sí mismo con el nombre “enviar audio” el cual tiene un retardo igual a un time-frame y corresponde al siguiente paquete.
- Si no es AP: Si recibe un paquete con el nombre “mezcla” entonces toma datos de la latencia desde que fue enviado desde el nodo emisor, aumenta el contador de paquetes recibidos y luego lo descarta.

## 6.6 Operación del Sistema

Para terminar de comprender por completo el funcionamiento de este sistema, se ilustrará cómo ocurre el flujo de mensajes y paquetes a través del tiempo. Para esto se tomará como ejemplo una red con dos nodos emisores, dos AP y un nodo receptor, como se muestra en la figura 10.

Se observa que al inicio, en la etapa de sincronización, el tiempo no se encuentra particionado por TDMA, luego, cuando se termina de sincronizar al último nodo y corresponde informar los turnos TDMA para comenzar el streaming, el tiempo se encuentra particionado, donde cada emisor nodo tiene el canal asignado durante un time-slot. Después de cada time-frame, el AP1 envía información al AP2 para poder medir la latencia en el tiempo (esto es solo relevante en la simulación, porque en la realidad la consola hace ese trabajo sin envío alguno). Además, luego de que transcurren cierto número de time-frames, se envía información de control para realizar la re-sincronización, aprovechando la capacidad ociosa del canal, como se discutió en la sección 7.

Paralelamente al TDMA se encuentra el broadcast de audio desde el AP2 (consola) hacia los receptores, el cual opera en otra frecuencia por lo cual no interfiere con el TDMA. En la figura puede observarse cómo para la parte de la red que realiza el broadcast se desperdicia gran parte del ancho de banda disponible, pues la capacidad del canal es muy superior a lo que se requiere para enviar la mezcla de audio (cerca de 30 veces mayor en base a un canal de 24 Mbps). El receptor no envía ningún tipo de información en ningún momento de la comunicación.

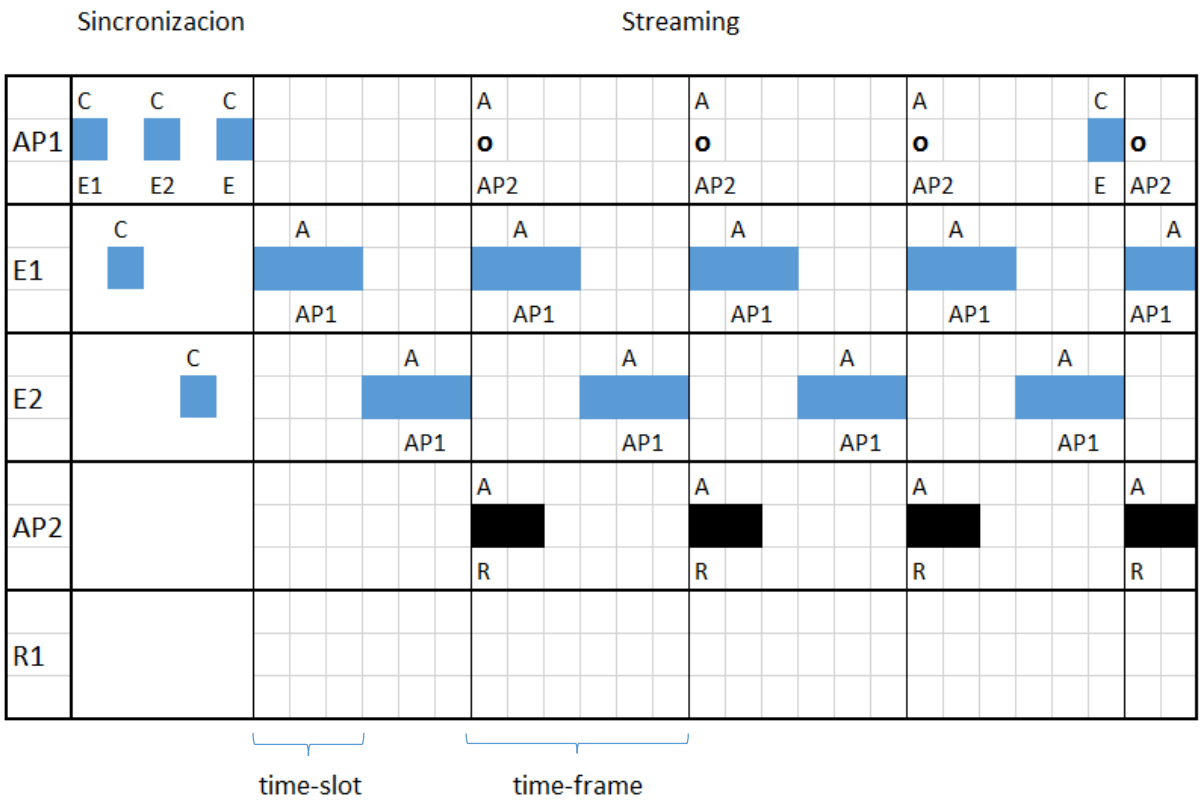


*Figura 10: Flujo de mensajes y paquetes en el tiempo. El gráfico muestra cómo se utiliza el canal en el tiempo (el tiempo se representa de izquierda a derecha). Se aprecia que los colores azul y negro representan tráfico en distinta frecuencia. Así mismo, sobre cada trozo de información se indica si esta es de tipo Audio (A) o control (C) y por debajo se indica a quién va dirigida la información, AP, Emisor (E) o receptor (R). Finalmente los círculos que envía el AP es información de control que se envía en la simulación mediante cable para medir las latencias.*

Cada vez que un mensaje es recibido por algún nodo se activan los métodos handleMessage(·) respectivos, los cuales se describieron anteriormente. Dado que cada mensaje tiene un nombre específico mediante el cual se reconoce qué rol cumple en la comunicación, es posible observar qué métodos se activan en los nodosTDMA que son los que poseen la lógica que soporta a este sistema. Esto puede verificarse en los códigos adjuntos en la sección de anexos.

# 7 SIMULACIONES, RESULTADOS Y DISCUSIÓN

Para todas las simulaciones se utilizaron las representaciones de radio IEEE 802.11 disponibles en INET [18]. En la siguiente tabla se especifican todos los parámetros utilizados y especificados en el archivo .ini. Los valores fueron elegidos según los requerimientos de la red y algunos valores por defecto propuestos por INET, tal como se muestran en la tabla 2:

| Parámetro | Valor |
|---|---|
| **Tecnología** | 802.11g |
| **Banda de Operación** | 2.4 GHz |
| **Tipo de Antena** | Isotrópica ideal |
| **Ancho de Banda** | 20 MHz |
| **Canal** | 1 o 6 dependiendo de la zona de la red según la figura 3 |
| **Capacidad del Canal** | 24 Mbps |
| **Tamaño del Encabezado** | 192 bits |
| **Potencia de Transmisión** | 20 mW |
| **Sensibilidad del Receptor** | -85 dB |
| **Detección de Energía del Receptor** | -85 dB |
| **Umbral SNIR del Receptor** | 4 dB |
| **Área de la Red** | 20m x 20m |

*Tabla 2: Parámetros generales de las distintas simulaciones.*

## 7.1 PRUEBAS DE DESEMPEÑO DE INICIALIZACIÓN DEL SISTEMA

Con el fin de verificar que funciona correctamente el algoritmo de sincronización, se procedió a sincronizar la red con distinto número de nodos emisores. Una de las cosas interesantes es observar que cuando el AP llama a un nodo que no está funcionando, procede a reintentar 3 veces cada 1 ms (parámetros de diseño) y si no hay respuesta continúa sincronizando al siguiente nodo, tal como se explicó en la sección 6.

Las configuraciones utilizadas para esta prueba se muestran en la tabla 3, indicando los nodos en funcionamiento con una X:

| #Config | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | E10 | E11 | E12 | E13 | E14 | E15 | E16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| **2** | | X | X | | X | | | | | | X | | | X | X | |
| **3** | X | X | X | X | | X | X | X | X | X | X | X | X | X | X | X |
| **4** | X | X | X | | | | | | | | | | | | | |

*Tabla 3: Escenarios para pruebas de sincronización.*

El algoritmo de sincronización está configurado de tal manera que el TDMA tendrá tantos participantes como nodos sincronizados existan. De este modo el sistema se configurará con 16, 6, 15 y 3 nodos para los 4 escenarios propuestos. Con el fin de observar este correcto funcionamiento

del algoritmo de sincronización se decidió registrar los datos de Intervalo de tiempo, que como se explicó en la sección 6 consiste en el tiempo de transmisión más tiempo de propagación al enviar un paquete estándar con información de audio. También se registró el arreglo del valor de Offset, que corresponde al tiempo que tiene que esperar cada nodo para enviar su primer paquete de audio una vez que se le informa su turno en TDMA.

La topología de la red se conforma con el AP al centro del área de la red y los nodos emisores se ubican aleatoriamente. Se realizó 1 simulación por escenario y los resultados obtenidos son los que se muestran en la tabla 4:

| #Conf | Nodo Emisor | Distancia en Tiempo [µs] | Offset [µs] | | #Conf | Nodo Emisor | Distancia en Tiempo [µs] | Offset [µs] |
|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 314,020100 | 270,730000 | | **2** | 14 | 314,018764 | 2400,021598 |
| **1** | 2 | 314,039971 | 600,007202 | | **2** | 15 | 314,029413 | 3000,010949 |
| **1** | 3 | 314,009949 | 1200,037224 | | **3** | 1 | 314,014114 | 330,590000 |
| **1** | 4 | 314,027168 | 1800,020005 | | **3** | 2 | 314,019977 | 600,027196 |
| **1** | 5 | 314,039691 | 2400,007482 | | **3** | 3 | 314,030556 | 1200,016617 |
| **1** | 6 | 314,015104 | 3000,032069 | | **3** | 4 | 314,012580 | 1800,034593 |
| **1** | 7 | 314,029245 | 3600,017928 | | **3** | 6 | 314,008633 | 2400,038540 |
| **1** | 8 | 314,029875 | 4200,017298 | | **3** | 7 | 314,029807 | 3000,017366 |
| **1** | 9 | 314,011593 | 4800,035580 | | **3** | 8 | 314,033258 | 3600,013915 |
| **1** | 10 | 314,031147 | 5400,016026 | | **3** | 9 | 314,024730 | 4200,022443 |
| **1** | 11 | 314,033789 | 6000,013384 | | **3** | 10 | 314,026502 | 4800,020671 |
| **1** | 12 | 314,028438 | 6600,018735 | | **3** | 11 | 314,014260 | 5400,032913 |
| **1** | 13 | 314,024325 | 7200,022848 | | **3** | 12 | 314,012576 | 6000,034597 |
| **1** | 14 | 314,011582 | 7800,035591 | | **3** | 13 | 314,009849 | 6600,037324 |
| **1** | 15 | 314,006138 | 8400,041035 | | **3** | 14 | 314,040525 | 7200,006648 |
| **1** | 16 | 314,047173 | 9000,000000 | | **3** | 15 | 314,021409 | 7800,025764 |
| **2** | 2 | 314,023867 | 164,950000 | | **3** | 16 | 314,047173 | 8400,000000 |
| **2** | 3 | 314,030331 | 600,010031 | | **4** | 1 | 314,020914 | 154,40000 |
| **2** | 4 | 314,040362 | 1200,000000 | | **4** | 2 | 314,008471 | 600,027883 |
| **2** | 1 | 314,028307 | 1800,012055 | | **4** | 3 | 314,036354 | 1200,000000 |

*Tabla 4: Resultados de Intervalo de tiempo y Offset.*

Con los resultados de la tabla 3 puede corroborarse al detalle el correcto funcionamiento del algoritmo, ya que esto es consistente con la implementación explicada en la sección 8 para calcular los tiempos de offset y asignar los turnos TDMA.

Como resultados globales del desempeño del algoritmo de sincronización se registraron los siguientes indicadores señalados en la tabla 5, donde se aprecia cómo la cantidad de nodos activos afecta en el valor del time frame y cómo la cantidad de reintentos realizados demora más el tiempo total de sincronización de la red, sin embargo aún en el escenario 4 donde se aprecia el mayor

tiempo de sincronización de la red el valor obtenido solo bordea las 4 centésimas de segundo, lo cual es muy positivo para la aplicación final de música en vivo.

| #Config | Cantidad de Nodos | Espera para reintentar [µs] | Reintentos Realizados | TimeFrame [µs] | Tiempo en Sincronizar la red [µs] |
|---|---|---|---|---|---|
| **1** | 16 | 1000 | 0 | 9600 | 5504,832492 |
| **2** | 6 | 1000 | 30 | 3600 | 31750,318214 |
| **3** | 15 | 1000 | 3 | 9000 | 8474,739071 |
| **4** | 3 | 1000 | 39 | 1800 | 40346,167832 |

*Tabla 5: Resultados globales de pruebas de sincronización.*

## 7.2 Pruebas de Desempeño de Latencia y Throughput

A continuación se muestra cómo se midió el desempeño del sistema respecto a la latencia y la razón de paquetes enviados y recibidos. De acuerdo a lo explicado en las secciones 6, 7 y 8, se procedió a simular el funcionamiento de la red con la configuración que se muestra en la figura 11 y presenta el escenario de máxima carga del sistema en cuanto a número de emisores y distancia máxima para computar latencias, ya que se monitorean los paquetes que van desde el emisor 16 hasta el receptor, donde ambos nodos están a la máxima distancia posible del AP. Los parámetros utilizados se encuentran en la tabla 6.

| Parámetro | Valor |
|---|---|
| **Tiempo de simulación (distinto de tiempo computacional)** | 40 minutos |
| **Cantidad de simulaciones** | 12 |
| **Nodos emisores** | 16 |
| **APs** | 2, en canales 1 y 6 respectivamente |
| **Nodos receptores** | 1 |
| **Distancia máxima al AP** | 15 m, aproximadamente |
| **Time-Slot** | 600 µs |
| **Time-Frame** | 9600 µs |

*Tabla 6: Parámetros de prueba de máxima carga.*

Dada la configuración y parámetros del sistema, se tiene que para una capacidad de canal de 24 Mbps la latencia máxima esperada del sistema en base a la teoría, viene dada por la expresión que fue deducida en la sección 7:

$$l_{total} = l_f + t_{tr1} + l_c + t_{tr2} = 9.6 + 0.284 + 2.5 + 0.284 = 12.668\, ms$$

Con lo cual se cumple teóricamente con la cota objetivo establecida para la latencia (15 ms).

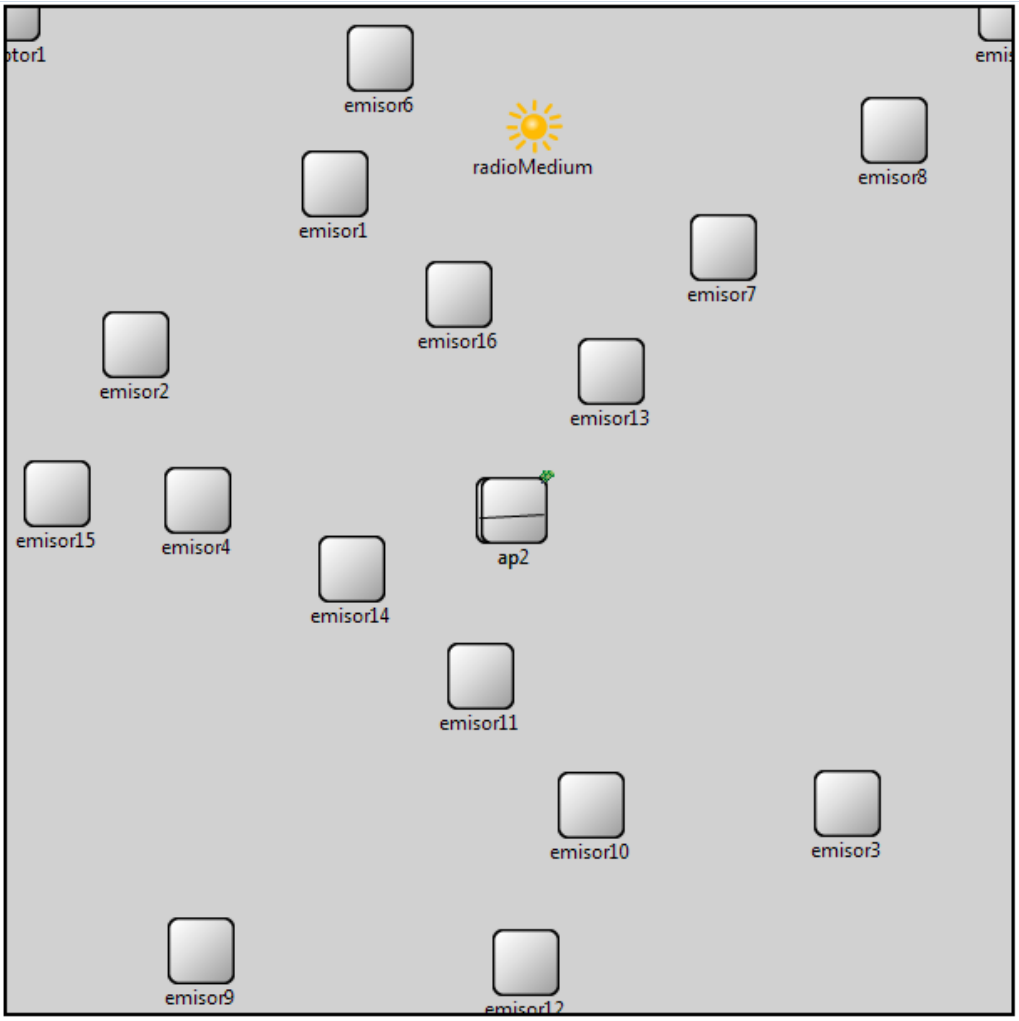


*Figura 11: Configuración de la red para la simulación de máxima carga.*

El hecho de realizar varias simulaciones permite generar escenarios donde la posición de los distintos nodos es aleatoria, excepto el receptor y el emisor16 que se sitúan lo más lejos posible del AP, el cual está al centro del área de la red, siempre con el fin de obtener la cota superior de la latencia del sistema. En estas simulaciones se recogieron datos de los paquetes emitidos por los nodos emisores, paquetes recibidos por AP1 y paquetes recibidos por el receptor, así como las latencias asociadas a un paquete que viaja desde un emisor al receptor, donde $l_f$ $y$ $l_c$ se consideran deterministas, por lo que se refleja el impacto de los tiempos de transmisión y los tiempos de procesamiento y propagación que el simulador entregue.

Además, para comprender mejor cómo afecta la variación de los distintos parámetros del sistema, se realizaron dos escenarios con capacidades del canal de 24 Mbps y 54 Mbps, ambos con los parámetros señalados en la tabla 6. Los resultados se encuentran en la tabla 7.

| Parámetro | Promedio (24 Mbps) | Desviación Estándar (24 Mbps) | Promedio (54 Mbps) | Desviación Estándar (54 Mbps) |
|---|---|---|---|---|
| **Paquetes enviados por cada emisor** | 249999 | 0,000000000 | 249999 | 0,000000000 |
| **Porcentaje de paquetes recibidos** | 100% | 0,000000000 | 100% | 0,000000000 |
| **Intervalo de tiempo** | 0,314025257 [ms] | 0,000009202 [ms] | 0,314025257 [ms] | 0,000009202 [ms] |
| **Latencia** | 12,72805051 [ms] | 0,000018520 [ms] | 12,40805097 [ms] | 0,000018332 [ms] |
| **Latencia Máxima** | 12.72809411 [ms] | 0,000000000 [ms] | 12,40809411 [ms] | 0,000000000 [ms] |
| **Tiempo en Sincronizar la Red** | 5,504873976 [ms] | 0,000099978 [ms] | 2,944880596 [ms] | 0,000098380 [ms] |

*Tabla 7: Resultados para máxima carga a 24 Mbps y 54 Mbps.*

Si se observa el valor obtenido para la latencia máxima a 24 Mbps, se tiene que es aproximadamente 0,059 ms superior a lo que se espera teóricamente, lo cual puede deberse a alguna configuración de OMNeT++ respecto a alguno de los tiempos de propagación, transmisión o procesamiento. Sin embargo, para efectos del objetivo de este trabajo se considera suficientemente cercano y satisfactorio.

Por otro lado, se aprecia cómo al aumentar la capacidad del canal se obtienen mejoras en los valores de intervalo de tiempo, latencia y tiempo de sincronización de la red, debido a que este aumento disminuye el tiempo de transmisión de los envíos de paquetes, lo cual entrega una holgura en la medida de que pueda aumentarse la capacidad del canal en una eventual implementación real. Además, se destaca que el factor que más afecta a la latencia guarda relación con el diseño de TDMA, siendo en este caso la contribución de la fuente de audio la que genera más latencia debido a que debe llenar su buffer antes de enviar un paquete (como se explicó en la sección 7), aportando una latencia de 9,6 ms para esta simulación (factor que depende de la calidad de audio y el número de nodos de la red), más 2,5 ms asociados a la consola de mezcla, siendo el tiempo de propagación el factor aleatorio para la latencia, que tiene directa relación con la posición de los nodos en la topología de la red. Esta variación puede apreciarse en la figura 12, donde se muestra un histograma con los valores del intervalo de tiempo medida por el AP, lo cual es una medida de los tiempos de transmisión y propagación. Así mismo, la latencia tiene asociada valores aleatorios que, aunque presentan una dispersión muy pequeña, puede apreciarse en la figura 13 cómo entregaron distintos valores para las distintas simulaciones realizadas.

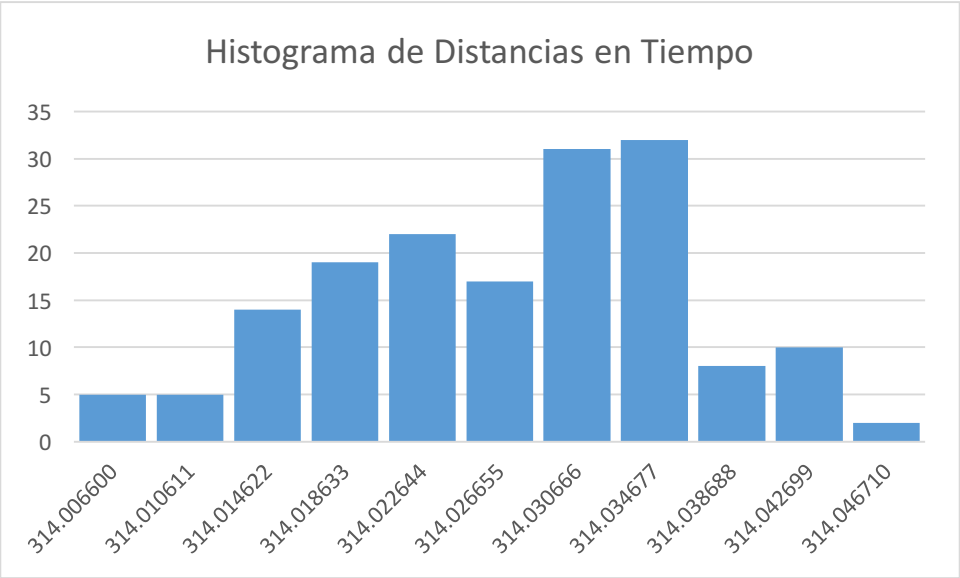


*Figura 12: Histograma de valores de Intervalo de tiempo para la prueba con 24 Mbps*

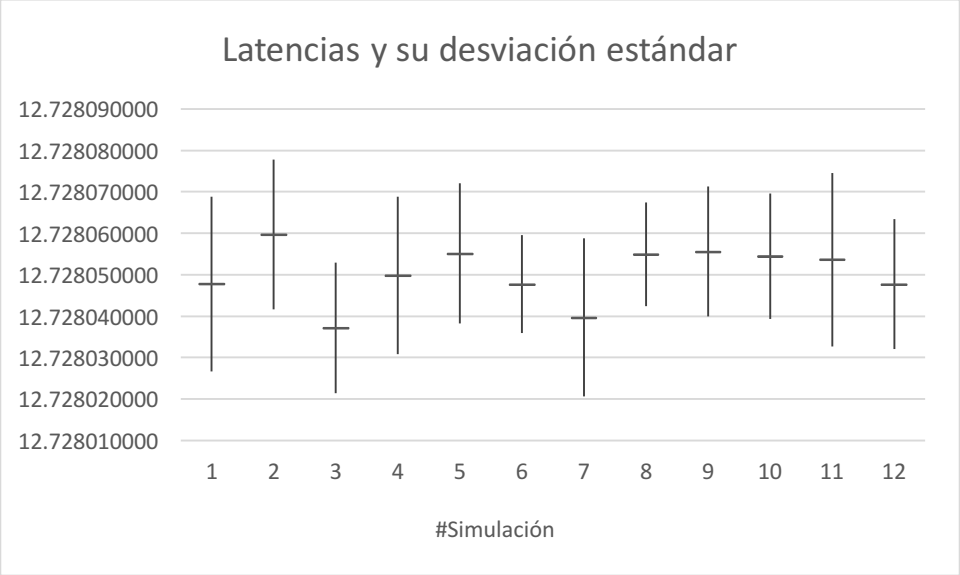


*Figura 13: Valor de latencias y su desviación estándar en las distintas simulaciones realizadas para 24 Mbps. La línea horizontal muestra la media y la línea vertical marca una desviación estándar hacia arriba y otra por debajo de la media.*

Los parámetros de diseño utilizados para el TDMA fueron pensados para cumplir con la cota establecida de la latencia y priorizar un buen throughput, lo que en este caso permite tener una razón de datos de audio vs datos de control del 97,26%, debido al encabezado de 192 bits que agrega 802.11g y que el tamaño de los paquetes es de 6816 bits dada la configuración utilizada para esta prueba de acuerdo a la tabla 6, con lo cual la capacidad mínima del canal para soportar este sistema es de 11,68 Mbps. Este excelente throughput, sumado a que no existe pérdida de paquetes

para la configuración dada en la tabla 2, entrega una holgura que podría aprovecharse para aumentar la calidad de la señal de audio transmitida (recordando que es la calidad mínima para ser considerada de alta definición) y para soportar una red con más nodos emisores, en la medida de que la capacidad del canal y la latencia soporten la nueva configuración y siempre recordando que existe un compromiso entre la latencia y el throughput, ya que a menor latencia, menos bits de audio tiene un paquete y baja el throughput del sistema.

## 7.3 Pruebas de Desempeño de Re-Sincronización

Para que la re-sincronización pueda ser verificada fue necesario des-sincronizar artificialmente los relojes internos de los distintos emisores. Para entender el razonamiento utilizado suponga que 2 relojes A y B, donde A es el reloj de referencia y B está desfasado según la expresión $d * hora_B = hora_A$, donde *d* es el desfase y corresponde a una constante. Por ejemplo si *d > 1* entonces el segundero de B avanza más lento que el de A, si *d < 1* avanza más rápido y si *d = 1* no existe el desfase.

Con fines ilustrativos, se simuló el efecto del método de sincronización tomando un nodo con d = 1.065, con lo cual se observa un peligro de desfase respecto a TDMA cada 3 frames, por lo que el método de sincronización envía el reloj de referencia cada 3 time-frames para evitar un desfase, tal como se aprecia en la figura 14, donde se muestra en color rojo cómo el desfase se vuelve muy perjudicial en caso de no re-sincronizar los relojes, mientras que al re-sincronizar (curva gris) el reloj desfasado se mantiene suficientemente cerca del reloj de referencia (curva azul)

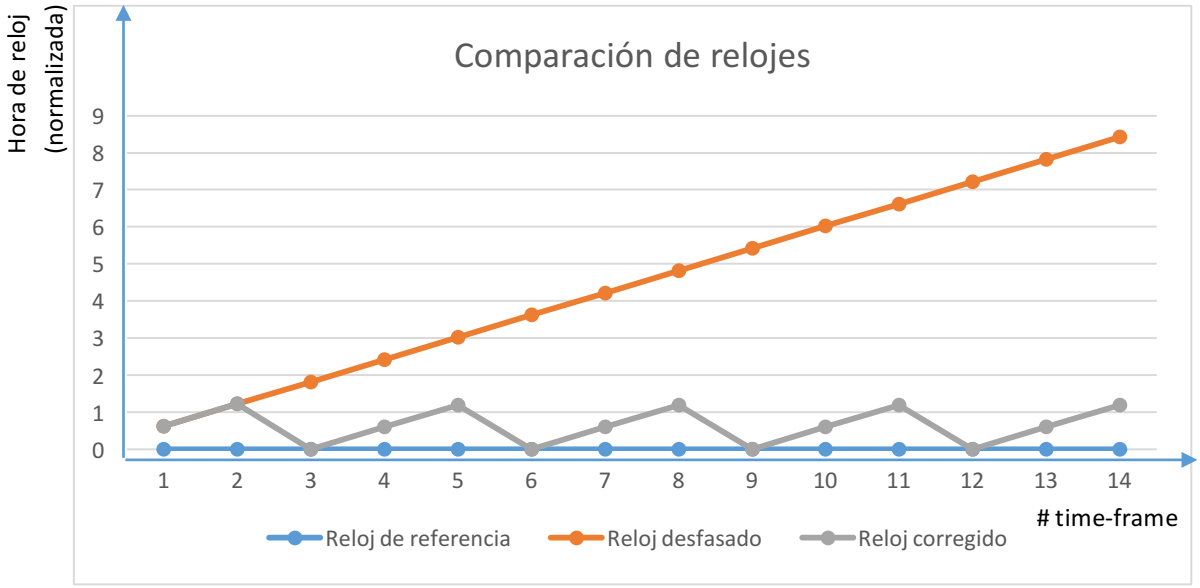


*Figura 14: gráfico comparativo de distintos relojes. Cada curva muestra el valor de la hora que marca el reloj menos la hora del reloj de referencia en función del time-frame en que se mide, para apreciar mejor el efecto del método de sincronización.*

En la práctica los desfases entre los relojes son muy pequeños y podrían incluso no ser significativos, sin embargo siempre dependen del hardware utilizado. Para configurar el método de re-sincronización de manera fiel a la realidad sería necesario estudiar empíricamente los desfases presentados y de ese modo asignar el parámetro de cada cuántos time frames el AP debe enviar su reloj de referencia para no reducir el throughput innecesariamente y mantener la red siempre sincronizada. Es por esto que también se decidió realizar las pruebas de máxima carga sin el método de re-sincronización, ya que no se sabe a priori cada cuántos time frames es necesario re-sincronizar y si esta suposición es errónea podría distorsionar los resultados obtenidos, confiando en que en la realidad es probable que solo se requiera sincronizar en excepcionales ocasiones en una transmisión. Con esto se muestra que el método propuesto funciona y solo requiere ser ajustado cuando se implemente el sistema a nivel de hardware.

Como se observó en las pruebas anteriores, la variación de posición de un nodo dentro de la red afecta mínimamente las distancias en tiempo entre los emisores y el AP, por lo cual su efecto puede despreciarse sin comprometer el desempeño de TDMA, siendo innecesario que sea abarcado por el método de re-sincronización.

# 8 CONCLUSIONES

En este trabajo se diseñó e implementó computacionalmente un protocolo de comunicaciones que permite comunicación inalámbrica para transmisión, mezcla y retorno de señales de audio de alta calidad con mínima latencia en el contexto de presentaciones de música en vivo. Se realizó un diseño sobre la capa física de IEEE 802.11 y se propuso una capa de enlace basada en TDMA, el cual se validó mediante simulaciones en OMNeT++.

Gracias a la validación del sistema obtenida con las simulaciones realizadas, se concluye que se cumplen satisfactoriamente los requisitos establecidos para su correcto desempeño respecto a throughput y latencia, lo cual incluso entrega una holgura que puede ser aprovechada para ampliar la capacidad del sistema soportando transmisión de audio de mejor calidad o soportando redes con mayor número de fuentes de sonido, siempre que la capacidad del canal lo permita. En la simulación realizada con la mayor carga posible para el sistema, se observó que el throughput alcanza el 97.26% de proporción de información de audio vs información de control, mientras la latencia asociada a un paquete desde que es creado por la fuente de audio hasta que es recibido por un receptor alcanza valores de a lo más 12.73 ms para los escenarios simulados, lo cual permitiría que el servicio de transmisión de audio entregado a los músicos sea de la calidad apropiada para un correcto desempeño respecto a la interpretación musical involucrada.

Junto a lo anterior, se propuso un algoritmo de sincronización que permite soportar la implementación de TDMA para acceso al medio. Este algoritmo se caracteriza por ser suficientemente rápido para la aplicación deseada, logrando sincronizar la red en cerca de 3 ms en el caso de carga completa de la red, y en cerca de 40 ms en caso de que la red tenga pocos nodos activos y se hayan tenido que realizar varios reintentos para asegurarse de que los nodos inactivos efectivamente estén inactivos. Recordando que la transmisión de audio comienza inmediatamente después de que la red termina de sincronizarse, se puede afirmar que para efectos prácticos el sistema comienza su modo de operación (streaming de audio) casi al mismo tiempo que este se enciende, facilitando que puedan incluirse nuevos emisores entre distintas canciones (pensando en música en vivo) simplemente reiniciando el sistema. Por otro lado, el método de re-sincronización entrega confiabilidad de que la transmisión ocurrirá de manera saludable, manteniendo la estructura TDMA independiente de posibles desfases asociados a los relojes del hardware involucrado.

Después de realizado este trabajo queda claro que el diseño propuesto permite un excelente aprovechamiento del canal y entrega una excelente calidad de servicio para redes inalámbricas de área local, y ya que está basado en tecnología IEEE 802.11 que actualmente está muy masificada y es ampliamente conocida, puede servir como referencia para distintas aplicaciones de redes WLAN que sean de interés en aplicaciones de tiempo real.

Finalmente, respecto a uno de los aspectos más interesantes relacionados a este trabajo, se concluye que de acuerdo a las evaluaciones de desempeño mediante simulaciones, el sistema propuesto permite acercarse de buena manera a las soluciones obtenidas en el mercado respecto

a la calidad de servicio entregada para aplicaciones de música en vivo. Para poder afirmar con seguridad que el proyecto tiene un potencial suficiente para ser una solución comercial en el futuro, queda manifiesta la intención de elaborar un prototipo y lograr establecer los requerimientos técnicos a nivel de hardware y de resiliencia frente a escenarios de operación adversos, para finalmente realizar la evaluación financiera correspondiente.

# 9 BIBLIOGRAFÍA


[8] IEEE. (2013*). IEEE Standard 802.1BA*: *Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks.* New York, NY, USA: IEEE-SA Standards Board.

[9] IEEE. (2011). *IEEE Standard 802.1AS: Audio Video Bridging Systems.* New York, NY, USA: IEEE-SA Standards Boards.

[10] IEEE. (2014). *IEEE Standard 802.1Q: Bridges and Bridged Networks.* New York, NY, USA: IEEE-SA Standards Boards.

[6] IEEE. (2012). *IEEE Standard 802.11 Wireless LAN Medium Access Control and Physical Layer Especifications.* New York, NY, USA: IEEE-SA Standards Boards.

[11] DeBeasi, P. (2009*). 802.11n: The End of Ethernet?.* Midvale, UT, USA: Burton Group.

[12] Gast, M. (2002). *802.11 Wireless Networks, The Definitive Guide.* San Francisco, CA, USA: O'Reilly.

[15] Djukic, P.; Mohapatra, P., (2012). "*Soft-TDMAC: A Software-Based 802.11 Overlay TDMA MAC with Microsecond Synchronization*," in IEEE Transactions on Mobile Computing, vol.11, no.3, pp.478-491.

[13] Costa, R.; Portugal, P.; Vasques, F.; Moraes, R., (2010) *"A TDMA-based mechanism for real-time communication in IEEE 802.11e networks,"* 2010 IEEE Conference on Emerging Technologies and Factory Automation (ETFA), pp.1-9, 13-16 Sept.

[19] A. K¨opke, M. Swigulski, K. Wessel, D. Willkomm, P. T. K. Haneveld, T. E. V. Parker, O. W. Visser, H. S. Lichte, & S. Valentin. (2008). *Simulating Wireless and Mobile Networks in OMNeT++ The MiXiM Vision.* OMNeT++ Workshop '08 Marseille, France: OMNeT++.

[17] A. Varga and R. Hornig, *An Overview of the OMNeT++ Simulation Environment*. Simutools, 2008, pp. 60:1–60:10.

[18] Inet Framework. (2012). *INET Framework for OMNeT++.* Diciembre 2015, de INET Framework. Sitio web: http://inet.omnetpp.org/doc/inet-manual-DRAFT.pdf.

[16] Yamaha. (2006*). Especificaciones técnicas de Consola de mezcla LS9*. Diciembre 2015, de Yamaha Corporation of America Sitio web: http://download.yamaha.com/api/asset/file/?language=en&site=ae.yamaha.com&asset_id=47011

[14] Li Feng; Jianqing Li; Xiaodong Lin, (2013)"*A New Delay Analysis for IEEE 802.11 PCF*," IEEE Transactions on Vehicular Technology, vol.62, no.8, pp.4064-4069, Oct.

[20] Sexton, M.J. (1989). *Synchronous networking-SONET and the SDH*. London, UK: IEE Colloquium on Changing Face of Telecommunications Transmission.

[7] Stallins, W. (2007). *Data and Computer Communications.* Upper Saddle River, NY, USA: Prentice Hall.

[1] Shure. (2015). *Catálogo de Sistemas Inalámbricos*. Diciembre 2015, de Shure Sitio web: http://es.shure.com/americas/products/wireless-systems

[2] Shure. (2015). *Catálogo de Sistemas de Monitoreo.* Diciembre 2015, de Shure Sitio web: http://es.shure.com/americas/products/personal-monitor-systems

[5] "ARTICLE 1 - Terms and Definitions". *life.itu.ch*. International Telecommunication Union. (2009). 1.15. *industrial, scientific and medical (ISM) applications (of radio frequency energy): Operation of equipment or appliances designed to generate and use locally radio frequency energy for industrial, scientific, medical, domestic or similar purposes, excluding applications in the field of telecommunications.*

[3] Hispasonic. (2015). *¿Cuál sería una latencia normal de entrada y salida?* Diciembre 2010, de Hispasonic Sitio web: http://www.hispasonic.com/foros/cual-seria-latencia-normal-entrada-salida/354572

[4] Haas, H. (1972). *The Influence of a Single Echo on the Audibility of Speech*, JAES Volume 20 Issue 2 pp. 146-159.

# 10 ANEXOS

A continuación se muestran los códigos más relevantes de la implementación de este trabajo. Se incluyen los archivos de configuración y estructuración de la red (en lenguaje .ned) y se incluye el código de la clase MacTDMA.cpp, que concentra la mayor parte de la lógica que soporta al sistema propuesto en este trabajo. Si se desea obtener más información acerca de la implementación, puede comunicarse con el autor de este trabajo.

## 10.1 ARCHIVO DE INICIALIZACIÓN DE LA RED OMNETPP.INI

```
[General]
network = simulacion
tkenv-plugin-path = ../../../etc/plugins

sim-time-limit = 1s
repeat = 1
**.vector-recording = false

**.constraintAreaMinX = 0m
**.constraintAreaMinY = 0m
**.constraintAreaMinZ = 0m
**.constraintAreaMaxX = 20m
**.constraintAreaMaxY = 20m
**.constraintAreaMaxZ = 0m

# Para todos los radios se usa 802.11g en la banda de 2.4GHz, parámetros por defecto
# Los otros parámetros por defecto son:
# antennaType = default("IsotropicAntenna");
# transmitter.headerBitLength = default(192b);
# transmitter.power = default(20mW);
# receiver.sensitivity = default(-85dBm);
# receiver.energyDetection = default(-85dBm);
# receiver.snirThreshold = default(4dB);

simulacion.ap.radio.channelNumber = 0
simulacion.ap.radio.carrierFrequency = 2.412GHz
simulacion.ap.radio.transmitter.bitrate = 24Mbps
simulacion.ap.radio.bandwidth = 20MHz

simulacion.ap2.radio.channelNumber = 5
simulacion.ap2.radio.carrierFrequency = 2.437GHz
simulacion.ap2.radio.transmitter.bitrate = 24Mbps
simulacion.ap2.radio.bandwidth = 20MHz

simulacion.receptor1.radio.channelNumber = 5
simulacion.receptor1.radio.carrierFrequency = 2.437GHz
simulacion.receptor1.radio.transmitter.bitrate = 24Mbps
simulacion.receptor1.radio.bandwidth = 20MHz

# el canal por defecto es 0, y carrierFrecuency por defecto es 2.412, por lo que se
omitirán

simulacion.emisor1.radio.transmitter.bitrate = 24Mbps
simulacion.emisor1.radio.bandwidth = 20MHz

simulacion.emisor2.radio.transmitter.bitrate = 24Mbps
simulacion.emisor2.radio.bandwidth = 20MHz
```

```
simulacion.emisor3.radio.transmitter.bitrate = 24Mbps
simulacion.emisor3.radio.bandwidth = 20MHz

simulacion.emisor4.radio.transmitter.bitrate = 24Mbps
simulacion.emisor4.radio.bandwidth = 20MHz

simulacion.emisor5.radio.transmitter.bitrate = 24Mbps
simulacion.emisor5.radio.bandwidth = 20MHz

simulacion.emisor6.radio.transmitter.bitrate = 24Mbps
simulacion.emisor6.radio.bandwidth = 20MHz

simulacion.emisor7.radio.transmitter.bitrate = 24Mbps
simulacion.emisor7.radio.bandwidth = 20MHz

simulacion.emisor8.radio.transmitter.bitrate = 24Mbps
simulacion.emisor8.radio.bandwidth = 20MHz

simulacion.emisor9.radio.transmitter.bitrate = 24Mbps
simulacion.emisor9.radio.bandwidth = 20MHz

simulacion.emisor10.radio.transmitter.bitrate = 24Mbps
simulacion.emisor10.radio.bandwidth = 20MHz

simulacion.emisor11.radio.transmitter.bitrate = 24Mbps
simulacion.emisor11.radio.bandwidth = 20MHz

simulacion.emisor12.radio.transmitter.bitrate = 24Mbps
simulacion.emisor12.radio.bandwidth = 20MHz

simulacion.emisor13.radio.transmitter.bitrate = 24Mbps
simulacion.emisor13.radio.bandwidth = 20MHz

simulacion.emisor14.radio.transmitter.bitrate = 24Mbps
simulacion.emisor14.radio.bandwidth = 20MHz

simulacion.emisor15.radio.transmitter.bitrate = 24Mbps
simulacion.emisor15.radio.bandwidth = 20MHz

simulacion.emisor16.radio.transmitter.bitrate = 24Mbps
simulacion.emisor16.radio.bandwidth = 20MHz
```

## 10.2 NodoTDMA.ned

```
package tdma_matias2.nodo_TDMA;

import inet.networklayer.diffserv.BehaviorAggregateClassifier;
import inet.physicallayer.common.packetlevel.Radio;
import inet.physicallayer.contract.packetlevel.IRadio;
import tdma_matias2.mac.MacTDMA;
import tdma_matias2.aplicacion.Aplicacion;
import inet.mobility.contract.IMobility;

module NodoTDMA
{
    parameters:
        @networkNode;
        int id;
        string radioType = default("Ieee80211ScalarRadio");
```

```
        int numRadios = default(1);
        string mobilityType = default(numRadios > 0 ? "StationaryMobility" : "");
        int tiempoDesfase;
        @display("bgb=182,225");
        int nNod;
    gates:
        input nodoTdmaIn @directIn;
        output nodoCableOut;
    submodules:
        mac: MacTDMA {
            @display("p=97,106");
            idNodo = id;
            nNodos = nNod;
        }
        app: Aplicacion {
            @display("p=97,34");
            nNodos = nNod;
            idNodo = id;
        }
        radio: <radioType> like IRadio {
            @display("p=97,178");
        }
        mobility: <mobilityType> like IMobility if mobilityType != "" {
            parameters:
                @display("p=29,186");
        }
    connections allowunconnected:
        nodoTdmaIn --> { @display("m=s"); } --> radio.radioIn;
        radio.upperLayerOut --> mac.radioIn;
        radio.upperLayerIn <-- mac.radioOut;
        mac.appOut --> app.macIn;
        mac.appIn <-- app.macOut;
              mac.cableOut --> nodoCableOut;
}
```

## 10.3 NODOSIMPLE.NED

```
package tdma_matias2.nodo_simple;

import inet.networklayer.diffserv.BehaviorAggregateClassifier;
import inet.physicallayer.common.packetlevel.Radio;
import inet.physicallayer.contract.packetlevel.IRadio;
import tdma_matias2.nodo_simple.NodoSimpleLogica;
import inet.mobility.contract.IMobility;

module NodoSimple
{
    parameters:
        @networkNode;
        bool esap;
        string radioType = default("Ieee80211ScalarRadio");
        int numRadios = default(1);
        string mobilityType = default(numRadios > 0 ? "StationaryMobility" : "");
        @display("bgb=182,225");
    gates:
        input nodoSimpleIn @directIn;
        input nodoSimpleCableIn;
    submodules:
        logica: NodoSimpleLogica {
            parameters:
                esAP = esap;
```

```
                @display("p=91,77");

        }
        radio: <radioType> like IRadio {
            @display("p=52,172");
        }
        mobility: <mobilityType> like IMobility if mobilityType != "" {
            parameters:
            @display("p=21,25");
        }
    connections allowunconnected:
        nodoSimpleIn --> { @display("m=s"); } --> radio.radioIn;
        radio.upperLayerOut --> logica.radioIn;
        radio.upperLayerIn <-- logica.radioOut;
        nodoSimpleCableIn --> logica.nodoSimpleLogicaIn;
```

## 10.4 SIMULACION.NED

```
package tdma_matias2.simulations;
import inet.physicallayer.common.packetlevel.Radio;
import tdma_matias2.mac.MacTDMA;
import tdma_matias2.nodo_TDMA.NodoTDMA;
import tdma_matias2.nodo_simple.NodoSimple;
import tdma_matias2.aplicacion.Aplicacion;
import inet.physicallayer.ieee80211.packetlevel.Ieee80211ScalarRadioMedium;

network simulacion
{
    parameters:
        int nNodos = 16;
        int bUniforme = 0.01;
        @display("bgb=20,20;bgs=30,m");
    submodules:
        radioMedium: Ieee80211ScalarRadioMedium {
        }
        ap: NodoTDMA {
            parameters:
                id = 0;
                tiempoDesfase = 0;
                @display("p=10,10");
                nNod = nNodos;
        }
        ap2: NodoSimple{
            parameters:
                esap = true;
                @display("p=10.1,10");
        }
        receptor1: NodoSimple{
            parameters:
                esap = false;
                @display("p=20,0");
        }
        emisor1: NodoTDMA {
            parameters:
                id = 1;
                nNod = nNodos;
        }
        emisor2: NodoTDMA {
            parameters:
                id = 2;
```

```
            nNod = nNodos;
        }
        emisor3: NodoTDMA {
            parameters:
                id = 3;
                nNod = nNodos;
        }
        connections allowunconnected:
            ap.nodoCableOut --> ap2.nodoSimpleCableIn;

}
```

## 10.5 MACTDMA.H

```
#ifndef MACTDMA_H_
#define MACTDMA_H_

#include <iostream>
#include <random>
#include <algorithm>
#include "cmessage.h"
#include "TdmaPacket_m.h"
#include <omnetpp.h>
#include <stdio.h>
#include <string.h>
#include "inet/common/INETDefs.h"
#include "inet/physicallayer/contract/packetlevel/IRadio.h"
#include "inet/physicallayer/contract/packetlevel/RadioControlInfo_m.h"
#include "inet/physicallayer/ieee80211/packetlevel/Ieee80211ControlInfo_m.h"
#include "inet/linklayer/ieee80211/mac/Ieee80211Mac.h"
#include "inet/common/INETDefs.h"
#include "inet/common/FSMA.h"
#include "inet/common/INETMath.h"
#include "inet/common/queue/IPassiveQueue.h"
#include "inet/common/lifecycle/ILifecycle.h"
#include "inet/physicallayer/contract/packetlevel/IRadio.h"
#include "inet/physicallayer/ieee80211/mode/IIeee80211Mode.h"
#include "inet/physicallayer/ieee80211/mode/Ieee80211ModeSet.h"
#include "inet/linklayer/base/MACProtocolBase.h"
#include "inet/linklayer/ieee80211/mac/Ieee80211Frame_m.h"
#include "inet/linklayer/ieee80211/mac/Ieee80211Consts.h"
#include "inet/linklayer/ieee80211/mac/IQoSClassifier.h"

class MacTDMA : public cSimpleModule
{
protected:

    // tiene las mismas stages que ieee80211, para compatibilizar con el radio
    virtual int numInitStages() const override { return 13; }
    virtual void initialize(int stage) override;
    virtual void handleMessage(cMessage *msg) override;
    virtual void finish() override;
    // sincroniza el nodo i-ésimo
    virtual void sincronizarNodo(int i);
    // si el nodo no es ap, con este método responde a los mensajes de sincronización del
AP
    virtual void responderSincronizacion();
    // con este método se actualiza la tabla de sincronización luego de sincronizar a
algún nodo
    virtual void actualizarTablaSinc(cMessage *msg);
```

```
    // con este método se continúa sincronizando al siguiente nodo que corresponda
    virtual void sincSgteNodo();
    // luego de que la sincronización está lista, con este método se informan los turnos
del TDMA a cada nodo
    virtual void informarTurnosTDMA();

    inet::ieee80211::IRadio *radio = nullptr;
    int appOutId = -1;
    int radioOutId = -1;
    int nodoCableOutId = -1;
    // mensaje usado para control
    cMessage *mensaje = nullptr;
    // indica si el nodo es AP o no
    bool esAP;
    // indica la id del nodo
    int idNodo;
    // indica la cantidad máxima de nodos de la red
    int nNodos = 0;
    // indica el largo de un time-slot
    simtime_t timeSlot ;
    simtime_t tiempoInicio;
    int paquetesEnviados;
    int paquetesRecibidos[1];

    //// cosas para sincronización ////

    cPacket *paquete;
    // este tdmaPacket se usa para control
    TdmaPacket *tpacket;
    // se usa para indicar que la sincronización está lista
    cMessage *sincronizacionLista;
    // se usa para reintentar la sincronizacion
    cMessage *reintentarSinc;
    // variable de estado que indica qué nodo se está sincronizando
    int nodoASincronizar;
    // número máximo de intentos de para re-sincronizar
    int nMaxIntentosSinc = 3;
    // variable de estado que indica cuántos intentos de re-sincronizar se han efectuado
    int nIntentosSinc;
    // tiempo de espera para volver a intentar sincronizar un nodo
    simtime_t esperaSincronizacion = SimTime(0.001);
    // arreglo de tiempo donde se guardan las distancias en el tiempo de cada nodo
    simtime_t tablaSinc[1];
    simtime_t offset[1];
    // arreglo con los tiempos que cada nodo debe esperar desde que comienza el
streaming.
    simtime_t tOffset;
    // reloj para la re-sincronizacion
    simtime_t reloj;
    // desfase del reloj
    simtime_t constante;
    // desfase del segundero del reloj
    double d;
    // entrega la hora desfasada del reloj
    virtual simtime_t decirHora(simtime_t hora, double d, simtime_t constante);
public:
    MacTDMA();
    virtual ~MacTDMA();
};

#endif /* APLICACION_H_ */
```

## 10.6 MacTDMA.cpp

```
#include <Mac/MacTDMA.h>

MacTDMA::MacTDMA() {
    // TODO Auto-generated constructor stub

}

MacTDMA::~MacTDMA() {
    cOwnedObject *Del=NULL;
    int OwnedSize=this->defaultListSize();
    for(int i=0;i<OwnedSize;i++){
            Del=this->defaultListGet(0);
            this->drop(Del);
            delete Del;
    }
}

Define_Module(MacTDMA);

void MacTDMA::initialize(int stage)
{
    if(stage == 0)
    {
        nNodos = par("nNodos");
        EV << "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX nNodos =
"<<nNodos<<endl;
        tpacket = new TdmaPacket("");
        tpacket->setTipo(0);
        timeSlot = SimTime(0.0006);
        d = 1 + 1/16;
        idNodo = par("idNodo");
        paquetesEnviados = 0;
        nIntentosSinc=0;        if(idNodo==0){
            esAP=true;
            nodoCableOutId = gate("cableOut")->getId();
        }
        else esAP = false;
        constante = 0;
        appOutId = gate("appOut")->getId();
        radioOutId = gate("radioOut")->getId();
        cModule *radioModule = gate("radioOut")->getNextGate()->getOwnerModule();
        //radioModule->subscribe(inet::ieee80211::IRadio::radioModeChangedSignal, this);
        //radioModule->subscribe(inet::ieee80211::IRadio::receptionStateChangedSignal,
this);
        //radioModule->subscribe(inet::ieee80211::IRadio::transmissionStateChangedSignal,
this);
        radio = check_and_cast<inet::ieee80211::IRadio *>(radioModule);
    }
    else if (stage == 4)
    {
        radio->setRadioMode(inet::ieee80211::IRadio::RADIO_MODE_TRANSCEIVER);
    }
}



void MacTDMA::handleMessage(cMessage *msg)
{
```

```
    if (esAP)
    {
        if (strcmp(msg->getName(),"iniciar sincronizacion")==0)
        {

            EV << "%%%%% iniciando sincronizacion %%%%%%.\n";
            nodoASincronizar = 1;
            sincronizarNodo(nodoASincronizar);
            delete msg;
        }
        else if(strcmp(msg->getName(),"respuesta sincronizacion")==0)
        {
            cancelEvent(reintentarSinc);
            cMessage *mensaje2 = msg;
            actualizarTablaSinc(mensaje2);
            sincSgteNodo();

        }
        else if(strcmp(msg->getName(),"reintentar sincronizacion")==0)
        {
            nIntentosSinc++;
            if(nIntentosSinc < nMaxIntentosSinc)
                sincronizarNodo(nodoASincronizar);
            else
                {
                EV << "%%%%%%% Nodo " << nodoASincronizar << " no se sincronizó
%%%%%%%"<<endl;
                // se asigna el valor 0 si el nodo no se logra sincronizar
                tablaSinc[nodoASincronizar-1] = 0;
                nIntentosSinc = 0;
                sincSgteNodo();
                }
            delete msg;
        }
        else if(strcmp(msg->getName(),"datos para AP")==0)
        {
            short numnodo = msg->getKind();
            double tiempo = SIMTIME_DBL((simTime() - tiempoInicio -
timeSlot*nNodos*paquetesRecibidos[7] )*10000.0);
            if (idNodo==16)
                EV<<"hola"<<endl;
            double tiempo2 = round(tiempo/6);
            int tiempo3 = (int)tiempo2 % nNodos;
            paquetesRecibidos[tiempo3]++;
            int paquetesRecibidos5 = paquetesRecibidos[6];
            if (tiempo3 == 15){
                // si es el nodo 16, se envía la paquete hacia el ap2 para medir su
latencia
                TdmaPacket *tpacket2 = dynamic_cast<TdmaPacket *>(msg);
                tpacket2->setName("procesar mezcla");
                tpacket2->setBitLength(0);
                tpacket2->setTimestamp(msg->getTimestamp());
                send(tpacket2 , nodoCableOutId);
                // grabo la hora
                if (paquetesRecibidos[7] % 3 == 0){
                    mensaje = new cMessage("enviar resinc");
                   scheduleAt(simTime() + timeSlot/2 , mensaje);
                }
            }
        }
        else if(strcmp(msg->getName(),"enviar resinc")==0){
```

```
            TdmaPacket mensaje1 = new TdmaPacket("resinc");
            mensaje1->setInicioTransmision(simTime() + tablaSinc[7]);
            send(radioOutId , mensaje1);
        }
        else
            delete msg;
    }
    else // si el nodo es un nodo emisor
    {
        if (strcmp(msg->getName(),"sinc")==0 && static_cast<int>(msg->getKind())==idNodo
)
        {
            responderSincronizacion();
            delete msg;
        }
        // si llega un mensaje informando los turnos, entonces se comunica a la capa de
aplicación que programe el streaming
        else if(strcmp(msg->getName(),"turnos")==0)
        {
            TdmaPacket *tpacket2 = dynamic_cast<TdmaPacket *>(msg);
            tOffset = tpacket->getOffset(idNodo-1);
            EV << "%%%%%%%%%%%% Nodo "<<idNodo<<" offset: "<< tOffset<< endl;
            tpacket2-> setName("programar streaming");
            send(tpacket2 , appOutId);
        }
        else if(strcmp(msg->getName(),"datos")==0)
        {// si se reciben datos desde la capa de aplicación, se envían al ap
            paquetesEnviados++;
            TdmaPacket *tpacket2 = dynamic_cast<TdmaPacket *>(msg);
            tpacket2->setName("datos para AP");
            tpacket2->setKind(idNodo);
            send(tpacket2 , radioOutId);
        }
        else if(strcmp(msg->getName(),"resinc")==0){
            if(idNodo == 16){
                TdmaPacket p = dynamic_cast<TdmaPacket*>(msg);
                constante = constante + decirHora(simTime(), d, constante) - p-
>getInicioTransmision();
            }
        }
        else
            delete msg;
    }
}

void MacTDMA::finish(){
    recordScalar("paquetesEnviados", paquetesEnviados);
    if(esAP){
        recordScalar("tiempoInicio", tiempoInicio);
        std::stringstream sstm;
        for(int i =0; i<16; i++){
            sstm.str("");
            sstm << "offset" << i;
            recordScalar( sstm.str().c_str(), offset[i]);
            sstm.str("");
            sstm << "sinc" << i;
            recordScalar( sstm.str().c_str(), tablaSinc[i]);
            sstm.str("");
            sstm << "paquetesRecibidosAP" << i;
            recordScalar( sstm.str().c_str(), paquetesRecibidos[i]);
        }
```

```
    }
}

void MacTDMA::sincronizarNodo(int nodo){

    cPacket *mensaje2 = new cPacket("sinc");
    mensaje2->setKind(static_cast<short>(nodo));  //va a preguntar por la i-ésima id
//    mensaje->setTimestamp(simTime());
    send(mensaje2,radioOutId);
    reintentarSinc = new cPacket("reintentar sincronizacion");
    scheduleAt(simTime()+esperaSincronizacion , reintentarSinc);

}
void MacTDMA::sincSgteNodo(){
    // se actualiza la variable nodo a sincronizar y se procede, si corresponde, a
sincronizar el siguiente nodo
    nodoASincronizar++;
    EV<<"NNNNNNNNNNNNNNNNNNNNNN " << nodoASincronizar << " NNNNNNNNNNNNNNNNNNN"<<endl;
    if (nodoASincronizar <= nNodos){
        sincronizarNodo(nodoASincronizar);
    }
    if(nodoASincronizar == (nNodos+1))
    {
        EV<<"/////////////////sincronizacion lista ////////////////7"<<endl;
//        tpacket = new TdmaPacket("turnos");
//        send(tpacket,radioOutId);
        informarTurnosTDMA();
    }
}
void MacTDMA::responderSincronizacion(){
    // si se recibe un mensaje de sincronización en un nodo cualquiera, se responde al AP
    cPacket *mensaje2 = new cPacket("respuesta sincronizacion");
    mensaje2->setTimestamp(simTime());
    mensaje2->setBitLength(nNodos*0.0006*710000);
    send(mensaje2,radioOutId);
}
void MacTDMA::actualizarTablaSinc(cMessage *msg){
    // actualiza la tabla de sincronización
    simtime_t distanciaTiempo = simTime() - msg->getTimestamp();
    tablaSinc[nodoASincronizar-1] = distanciaTiempo;
    EV<< "TTTTTTTTTTTTTTTT dist en el tiempo " <<  tablaSinc[nodoASincronizar-1] <<
"TTTTTTTTTTTTTTTTTTTTt"<<endl;
}

void MacTDMA::informarTurnosTDMA()
{
    // primero se calculan los tiempos de offset para cada nodo
    simtime_t suma = 0;
    double count = 0;
    simtime_t tmax = tablaSinc[0];
    TdmaPacket *tpacket2 = new TdmaPacket("turnos");
    tpacket2->setTipo(0);
    for (int i = 0; i<nNodos ; i++)
    {
        if(i>0)
            tmax = std::max(tablaSinc[i],tmax);
    }
    for(int i = 0; i<nNodos ; i++)
    {//acá es donde se le dice cuánto tiene que esperar cada nodo para enviar su primer
paquete, desde que reciban este mensaje.
        if(tablaSinc[i] != 0)
```

```
        {// solo se cuentan los nodos que fueron sincronizados
            suma = suma + tablaSinc[i];
            tpacket2->setOffset( i , timeSlot*count + tmax-tablaSinc[i]);
            offset[i] = timeSlot*count + tmax-tablaSinc[i];
            count = count + 1;
        }
    }
    // inicializo los contadores de paquetes recibidos
    for(int i =0 ; i<nNodos ; i++){
        paquetesRecibidos[i]=0;
    }
    tiempoInicio = simTime() + tablaSinc[0] + tpacket2->getOffset(0);
    tpacket2 -> setTimeSlot(timeSlot);
    // el time-frame está calculado en base a la cantidad de nodos sincronizados
    tpacket2->setDuracionFrame(timeSlot*count);
    EV << "%%%%%%%%%%%%%%%%%% tiempo del frame = "<< tpacket2->getDuracionFrame()<<endl;
    tpacket2->setInicioTransmision(tiempoInicio);
    EV << "%%%%%%%%%%%%%%%%%% se iniciará la transmisión en t="<< tiempoInicio <<endl;
    send(tpacket2,radioOutId);
}

simtime_t MacTDMA::decirHora(simtime_t hora, double d, simtime_t constante){
    return hora*d-constante;
}
```