UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

MODELADO, ANÁLISIS Y SIMULACIÓN DE PROTOCOLOS DE COMUNICACIÓN PARA REDES MÓVILES E INALÁMBRICAS

María Elena Villapol Blanco

ELÉCTRICA Y ELECTRÓNICA
LATACUNGA

MODELADO, ANÁLISIS Y SIMULACIÓN DE PROTOCOLOS DE COMUNICACIÓN PARA REDES MÓVILES E INALÁMBRICAS

María Elena Villapol Blanco

Modelado, Análisis y Simulación de Protocolos de Comunicación para Redes Móviles e Inalámbricas
María Elena Villapol Blanco, PhD.
Primera edición electrónica. Diciembre de 2014
ISBN: 978-9978-301-47-0
Par revisor: Comisión Editorial de la ESPE

Universidad de las Fuerzas Armadas - ESPE
Grab. Roque Moreira Cedeño
Rector
Crnl. Francisco Armendáriz Saénz
Vicerrector Académico General
Crnl. Ricardo Urbina
Vicerrector de Investigación

Publicación autorizada por:
Comisión Editorial de la Universidad de las Fuerzas Armadas - ESPE

Grab. Mauro Pazmiño, MBA.	Presidente
Washington Sandoval, Ph.D.	Víctor Hugo Abril, Ph.D.
Mónica Jadán, Mg.	Ana V. Guamán, MSc.
Betzabé Maldonado, Mg.	Eddie Galarza, Mg.
Kléver Bravo, Dr.	José Albuja, Dr.

Edición y producción
David Andrade Aguirre

Diseño
Pablo Zavala A.

Derechos reservados. Se prohibe la reproducción de esta obra por cualquier medio impreso, reprográfico o electrónico.
El contenido, uso de fotografías, gráficos, cuadros, tablas y referencias es de **exclusiva responsabilidad** del autor.

Los derechos de esta edición electrónica son de la **Universidad de las Fuerzas Armadas - ESPE,** para consulta de profesores y estudiantes de la universidad e investigadores en: htpp//www.repositorio.espe.edu.ec.

Dedicatoria

A la memoria de mis padres, Jesús Miguel y Alecia María.

A mis hijos, María Victoria y José Francisco, quienes son la pasión de mi vida.

Agradecimientos

Al Consejo de Desarrollo Científico y Humanístico de la UCV por su ayuda económica a través de los proyectos de investigación aprobados que sustentaron este trabajo y el soporte económico para mi pasantía sabática en la University of Central Florida.

A los árbitros que revisaron los artículos que soportan esta investigación por sus comentarios que fueron de gran ayuda en la depuración del trabajo realizado.

A mis compañeros de trabajo, David, Karima y Ana que tomaron su tiempo en leer este documento y me dieron sus comentarios y sugerencias.

1. Prólogo

1.1. Antecedentes

A medida que han ido evolucionando las redes de computadoras, los protocolos de comunicación se han ido tornando más complejos. Así durante la década de los años 70, hubo un gran interés en la comunicación entre computadoras surgiendo dos principales problemas del desarrollo de diferentes protocolos de comunicación. Uno fue la incompatibilidad de los productos de diferentes fabricantes y el otro fue la complejidad de las tareas de comunicación. En consecuencia, al final de los 70, la *Organización Internacional para la Estandarización* (*International Organization for Standardization, ISO)* [26], la cual es un cuerpo internacional para el desarrollo de estándares en diversas materias, estableció un subcomité para comenzar a trabajar en la solución de estos problemas. Una de la metas del subcomité fue desarrollar un modelo estructurado, que abarcara las funciones de comunicación y promoviera la interoperabilidad entre computadoras de diferentes vendedores. El resultado del trabajo de la ISO fue el Modelo de Referencia llamado *Interconexión de Sistemas Abiertos* (*Open Systems Interconnection, OSI)* [28]. OSI proporciona un patrón funcional y conceptual, el cual les permite a las personas alrededor del mundo desarrollar estándares.

Un ejemplo, que revela la complejidad de la comunicación entre computadoras es ARPANET [14], predecesora de Internet. Esta red tiene sus inicios en un proyecto cuyo propósito fue conectar las computadoras de varias universidades e institutos de investigación de los Estados Unidos y el cual fue financiado por la *Agencia de Proyectos de Investigación Avanzados (Advanced Research Projects Agency, ARPA)* del *Departamento de Defensa (Department of Defense, DoD)*, con la finalidad facilitar el intercambio de información entre los investigadores. Dicha red, que nació en el año 1969 con un enlace que conectó a un *host* en *Stanford Research Institute (SRI)* con otro localizado en *University of California, Los Ángeles (UCLA)*, entre los cuales se transmitió una simple palabra, LO (realmente se deseaba transmitir LOGIN), hoy es la red más grande del planeta [36]. Aunque para probar que las máquinas podían comunicarse se necesitaron simple reglas de comunicación, a partir de que dicha comunicación pudo establecerse exitosamente, se fueron creando protocolos para regular el intercambio de información. Esto forzó a la comunidad de Internet a crear un modelo para estructurar la forma en que estas reglas o protocolos se organizaban. Dicho modelo es conocido como el modelo TCP/IP [56] y presenta similitudes con el modelo OSI. Por ejemplo, ambos usan capas estructuradas para dividir las tareas de comunicación, y las funcionalidades de las capas son similares. Ellos, sin embargo,

presentan diferencias. Por ejemplo, el modelo OSI define algunos conceptos importantes involucrados en la arquitectura de protocolos (tal como, interfaces y servicios), mientras que en el modelo TCP/IP, estos conceptos no están bien definidos. Los conceptos del modelo OSI hacen que el proceso de verificación y especificación de protocolos sea más fácil.

El rápido desarrollo de las redes de computadoras y el auge del desarrollo de los protocolos de comunicación hicieron que surgiera el concepto de *ingeniería de protocolos de comunicación*. La ingeniería de protocolos de comunicación hace énfasis en el desarrollo de los protocolos de comunicación [40] y trata de responder a la necesidad de desarrollar protocolos cada vez más complejos dadas las exigencias de comunicación de los usuarios (email, conexión remota, transferencia de archivos) así como también las capacidades, bondades y debilidades de los medios de comunicación, pasando por la interconexión de redes vía dispositivos intermedios, tales como los encaminadores. Las actividades asociadas a la ingeniería de protocolos incluyen aquellas involucradas en el diseño y mantenimiento de protocolos usando *métodos formales*. El término *formal* se refiere a aquellas técnicas basadas en matemáticas [2].

Dada la evolución tan rápida de la tecnología, muchas veces no se cuenta con el tiempo necesario para el desarrollo de un nuevo protocolo, lo cual involucra un conjunto de actividades envueltas en la ingeniería de protocolos, que comienza con la definición y análisis de sus requerimientos y que finaliza con la prueba y verificación del mismo. Es por ello que algunas tecnologías han sido implementadas por los fabricantes de los equipos aun antes de que el estándar que la soporta sea formalmente aprobado, tal es el caso del estándar IEEE 802.11n [46], cuya versión *draft* se implementó en varios dispositivos.

Un desarrollo de los protocolos de comunicación emparejado con la rápida evolución de la tecnología puede tener consecuencias negativas. Así, ya algunos artículos de investigación han mostrado la existencia de errores en ciertos protocolos. Por ejemplo, Gordon et al. [58] encontraron inconsistencias en el servicio prestado por *Class 2 Wireless Transaction Protocol (WTP*), al analizar dicho protocolo usando un método formal como lo son las *Redes de Petri Coloreadas (Colored Petri Nets, CPNs)* [32]. Han et al. [22], por su parte, encontraron un abrazo mortal cuando una conexión TCP es liberada antes de ser totalmente establecida. En esta investigación también se usaron las CPNs para verificar el protocolo. Más recientemente, Blanco et al. [5] encontraron que la especificación de la activación de la operación de la Capa MAC/ Física Alternativa (AMP) en Bluetooth 3.0 está incompleta.

Otra consecuencia de lo antes dicho es que los documentos que describen los protocolos usan mayormente narrativa para especificar las funciones y procedimientos, resultando en especificaciones ambiguas y difíciles de leer. Por lo cual, Villapol et al. [61] presentan un modelo formal del *Protocolo para la Reservación de Recursos* (*Resource Reservation Protocol, RSVP*), el cual es verificado de acuerdo a un conjunto de propiedades propias y otras que deberían estar presentes en cualquier protocolo.

Pese a que existen pocos trabajos que reflejen el uso de todas las actividades envueltas en la ingeniería de protocolos para el desarrollo de algún protocolo de comunicación, en los últimos años, los métodos formales han sido aplicados a las actividades envueltas en la misma, tales como, la especificación y verificación de protocolos. Por ejemplo, un trabajo inicial en la ingeniería de protocolos ha sido reportado por Billington [2]. Documentos más recientes puede ser encontrados en las memorias de *IFIP6 Work Group 6.1 Annual Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE)* [49].

Existe un amplio rango de métodos formales, en este libro se usan las *Redes de Petri Coloreadas (Colored Petri Nets, CPNs)*. Hay varias razones para usar las CPNs para el modelado y análisis de protocolos. Las principales se resumen a continuación. Las Redes de Petri son una técnica madura, lo cual se muestra en lo miles de artículos en revistas y reportes de investigación generados en más de 40 años de trabajo teórico y práctico [48]. Las Redes de Petri están soportadas por un estándar internacional [27] y varios libros de textos [47][51][52]. Las Redes de Petri constituyen una herramienta gráfica que permite análisis formal. En particular, las CPNs han sido usadas para modelar diversos protocolos de comunicación, tal como se refleja en [1][32].

La ingeniería de protocolos basada en el uso de técnicas formales presenta el problema de que se requiere de personas que estén familiarizadas con las mismas y que a su vez tengan el conocimiento de la tecnología a modelar. Esto ha constituido una limitante para que se amplíe el uso de estas técnicas en el desarrollo de los protocolos de comunicación hoy en día. Por el contrario, el estudio de las tecnologías de comunicación basadas en simulación son comúnmente usadas. La simulación tiene a su favor que existen diversas herramientas, tales como NS-2 [43] y Opnet [44], que permiten evaluar una vasta cantidad de tecnologías de redes de comunicación. El problema con la simulación es que no permite la verificación ni del protocolo ni del mecanismo que se está evaluando y por el contrario confía en las pruebas funcionales realizadas por los programadores para validar las implementaciones de los mismos.

Dado el panorama expuesto anteriormente se definirá el problema de la investigación descrita en este libro y la motivación de la misma.

1.2. Problema

En la última década ha habido un auge de las redes móviles e inalámbricas. Para finales del siglo pasado ya las tecnologías de redes de computadoras eran lo suficientemente maduras para finalmente converger con las tecnologías de comunicación inalámbricas. El resultado de dicha convergencia es la posibilidad de conectar múltiples tipos de dispositivos usando diversas interfaces de comunicación soportadas por tecnologías de *Redes de Área Personal Inalámbricas* (*Wireless Personal Area Netrwoks, WPAN*) hasta aquellas usadas para conectarse a una *Red de Área Amplia Inalámbrica* (*Wireless Wide Area Networks, WWAN*). Así, la segunda década de

este siglo se presenta con una gran cantidad de tecnologías de comunicación, muchas de estas relacionadas a las redes de comunicaciones móviles e inalámbricas.

Una de estas tecnologías inalámbricas surgió a finales del siglo pasado para interconectar dispositivos que se encontraran muy cercanos (aproximadamente a unos 5 metros de distancia entre ellos). Dicha tecnología se denomina Bluetooth [8] y ha tenido un éxito particular ya que permite el intercambio de información de voz, datos, audio, imagen y video entre dispositivos sin necesidad de tener una infraestructura particular (es decir, se basa en el tipo de comunicación denominada ad hoc). Bluetooth, sin embargo, se caracteriza por ser complejo y esto se puede observar al leer el documento estándar de una de las versiones de esta tecnología más utilizadas hoy en día, como lo es la versión 2 [8].

Otra tecnología de inicios de este siglo es WiMax, soportada por el estándar de la IEEE 802.16 [25] y cuyo objetivo es la interconexión de dispositivos fijos o móviles separados a distancias de varios kilómetros, distancias que pueden abarcar una ciudad. WiMax tiene entre sus características la adaptabilidad, por ejemplo, las técnicas de codificación de la información no son fijas sino que pueden cambiarse de acuerdo a las condiciones del canal. Por otra parte, el tamaño de la Unidad de Dato de Protocolo (*Protocol Data Unit, PDU*) también puede cambiarse. Esta condición de WiMax de ser adaptativa puede ser manejada por los implementadores para mejorar el rendimiento de una red que use esta tecnología.

De lo ante expuesto se generan varias interrogantes acerca del funcionamiento de esta dos tecnologías, Bluetooth y WiMax. La primera está relacionada a si una de las funciones del protocolo fundamental de Bluetooth, es decir el establecimiento de la conexión a nivel del protocolo Bandabase, se comporta de acuerdo a lo especificado en el documento estándar. La segunda interrogante está relacionada con una propuesta para la construcción de PDUs de la subcapa de *Control de Acceso al Medio* (*Medium Access Control, MAC*) de WiMax, la cual está basada en información de retroalimentación y que explota la adaptabilidad de esta tecnología a nivel de la subcapa MAC. Una interrogante relacionada a esta propuesta es si el mecanismo mejora el rendimiento de la red. La otra interrogante es si la propuesta se comporta de acuerdo a lo especificado en su diseño y conforma las funciones del estándar que la soportan.

1.3. Justificación

No es extraño, hoy en día, que alguien haya presentado problemas para establecer la comunicación entre algún dispositivo móvil usando cierta tecnología de comunicación (por ejemplo, Bluetooth), sobre todo cuando estos dispositivos son de fabricantes diferentes. Por otra parte, algunas veces sucede que aunque la comunicación se puede establecer la red no se comporta como se espera, bien desde el punto de vista subjetivo del usuario (*Calidad de la Experiencia, Quality of Expierience, QoE*) o desde el punto de vista del rendimiento ofrecido por de la red (*Calidad de Servicio, Quality of Service, QoS*). Aunque las causas de esto pueden ser

múltiples, una de ella puede ser que algunos de los protocolos que integran la tecnología tengan algún problema en su definición.

Por otra parte, existe una complejidad propia de las redes móviles e inalámbricas que viene determinada por su funcionamiento sobre un medio de transmisión que se puede ver afectado por muchos factores adversos, tales como la interferencia de diversas fuentes [41], cuyos efectos son muchas veces difíciles de predecir. Por lo cual, el concepto de adaptabilidad se ha introducido en estas redes. La adaptabilidad suma complejidad al desarrollo de las tecnologías de comunicación al dejar abierta la puerta para el desarrollo de múltiples mecanismos que soporten las mismas. Por ello es necesario estudiar estas nuevas propuestas desde el punto de vista de la ingeniería de protocolos como desde el punto de vista del rendimiento de la red.

1.4. Objetivo

El objetivo de este libro es presentar en detalle el modelado y análisis del protocolo Bandabase de Bluetooth usando las técnicas formales, CPN y lógica temporal y el modelado, simulacion y análisis del mecanismo de construcción de PDUs de la subcapa MAC (MPDUs) de WiMax adaptativa basada en información de retroalimentación usando técnicas de simulación y técnicas formales (CPN y lógica temporal).

1.5. ¿A Quién va Dirigido este Libro?

El libro está dirigido a investigadores y estudiantes de pregrado y postgrado que deseen entender en profundidad el estudio de los protocolos y mecanismos de comunicación usando un método formal, como son las CPNs, o la simulación. El mismo puede ser usado en asignaturas de pregrado y postgrado, tales como la evaluación de rendimiento, diseño de protocolos y especificación formal de protocolos. También puede ser de gran ayuda para aquellos estudiantes que estén desarrollando sus trabajos especiales de grado, trabajos de grado de maestría o tesis doctorales en esta área.

1.6. Estructura del Libro

Con la finalidad de cumplir los objetivos planteados el libro se encuentra estructurado de la siguiente forma. El Capítulo 1 presenta una introducción al mismo. Luego, en el Capítulo 2 se describen los conceptos fundamentales relacionados a la especificación y verificación de protocolos. Una descripción de las CPNs, lógica temporal y las herramientas usadas en este libro se realiza en el Capítulo 3. En el Capítulo 4 se describe el modelado y análisis del protocolo Bandabase de Bluetooth. El mecanismo de construcción de MPDUs de WiMax adaptativa basada en retroalimentación es introducido en el Capítulo 5, para entonces presentar un estudio del rendimiento de esta propuesta en función de parámetros de rendimiento de la red y de parámetros de subjetividad del usuario. En el Capítulo 6, el mecanismo anterior es modificado presentando una propuesta de construcción de MPDUs de WiMax adaptativa basada en los bloques básicos de retransmisión de la sub capa MAC, conocidos como bloques ARQ. Para su diseño se hace uso de

las CPNs que permiten además verificar el mecanismo, en función de las propiedades básicas de las Redes de Petri y de ciertas propiedades definidas en este documento. En la verificación se usan las funciones de búsqueda del grafo de estado y la técnica de chequeo de modelos.

2. Especificación y Verificación de Protocolos

2.1. Introducción

La especificación y verificación formal de protocolos implica un conjunto de actividades, que se inician con la descripción de la arquitectura de protocolo hasta la verificación del modelo propuesto [23]. Billington et al. [3] presentan estas actividades como un conjunto de pasos sistemáticos y los denominan *metodología de verificación de protocolos*, la cual se ha aplicado exitosamente a un conjunto de protocolos de comunicación [22][58][61]. En este libro, la metodología se utiliza como marco de referencia para las actividades de modelado y análisis del establecimiento de la conexión Bandabase de Bluetooth, y del mecanismo de construcción de MPDUs de WiMax adaptativa. Algunos conceptos relacionados con el modelo OSI [28][29] también son introducidos como ayuda a la descripción de la metodología.

2.2. Definiciones de la Arquitectura de Protocolos

El modelo OSI divide las actividades de la comunicación en siete capas estructuradas jerárquicamente [28][29]. Cada capa realiza un conjunto de las funciones previstas para proporcionar algunos servicios a la capa inmediatamente superior. La Figura 2.1 muestra los componentes de la jerarquía de protocolos según lo especificado por OSI [29]. El modelo OSI se refiere a una capa genérica como la *capa N*, la capa superior como la *capa N+1* y la capa inferior como la *capa N-1*. Las capacidades de la *capa N* se dividen entre las piezas activas conocidas como *entidades del protocolo N* (o solamente entidades). Dos entidades separadas por una red se comunican remotamente siguiendo un conjunto de las reglas preestablecidas, que se conocen como *protocolo* de comunicación (protocolo par). Así, las entidades del protocolo que trabajan en la *capa N* funcionan conjuntamente para proporcionar un servicio a la *capa N+1*.

Un *servicio* es una capacidad de una capa (proveedor del servicio) que se proporciona a la capa localizada por encima (es decir el usuario del servicio) a través del *punto de acceso al servicio (Service Access Point, SAP)* (ver la Figura 2.2). Un SAP es una dirección, que identifica el límite entre dos capas adyacentes. La comunicación entre las capas adyacentes se puede describir en un nivel abstracto como la ocurrencia de las *primitivas de servicio* (ver [29]). Las primitivas de servicio proporcionan una manera abstracta de describir la interacción entre el usuario del servicio y el abastecedor del servicio.

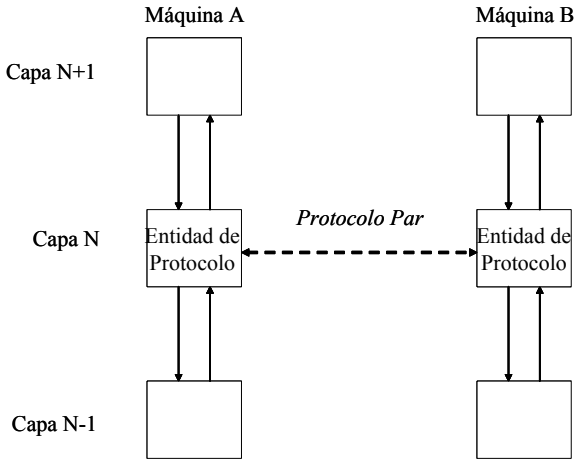


Figura 2.1: Jerarquía de capas.

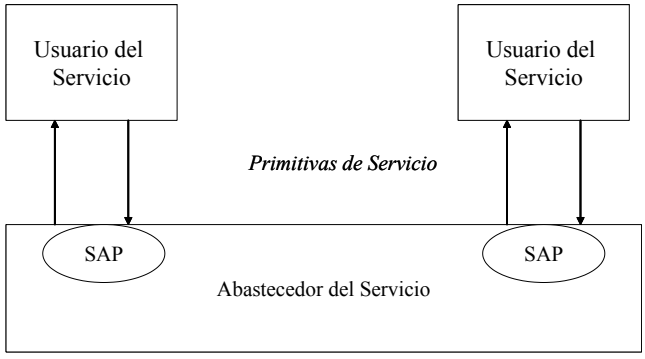


Figura 2.2: Aprovisionamiento del servicio en OSI.

2.3. Descripción de la Metodología

La Figura 2.3 muestra las actividades involucradas en la metodología para la verificación de protocolos. No todas las actividades que son parte de la metodología y que son descritas en este capítulo son implementadas, debido a que están fueran del alcance del libro.

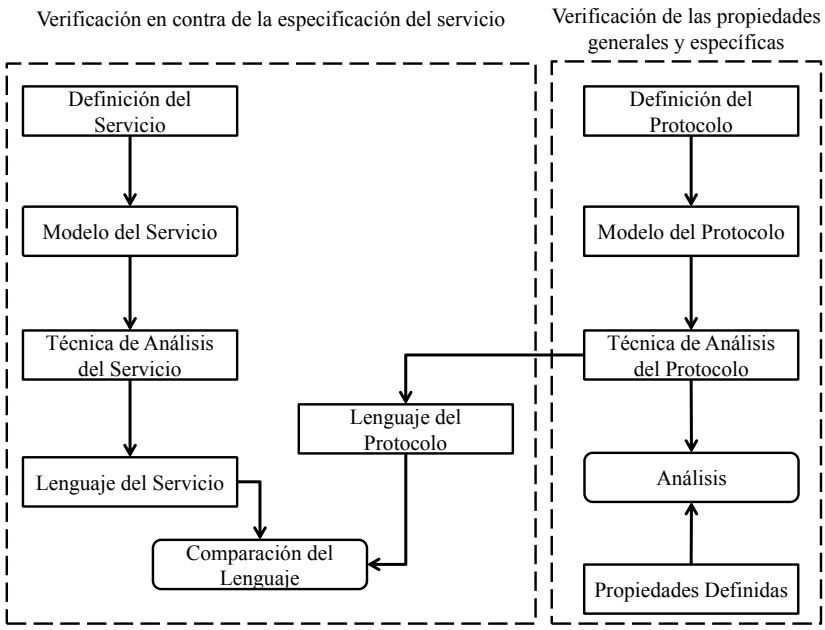


Figura 2.3: Metodología de verificación de protocolos.

2.4. Descripción de la Arquitectura

Aunque la descripción de la arquitectura no está señalada en la Figura 2.3 ni explícitamente descrita en [3] es una actividad que debe realizarse al comenzar a estudiar un protocolo de comunicación. Por este motivo la misma se detallará en esta sección.

La relación entre los protocolos se puede ilustrar usando una pila de protocolos que muestra la ubicación de cada uno de ellos en la estructura jerárquica. Esta arquitectura se denomina la arquitectura de la jerarquía de protocolos (o solamente arquitectura de protocolos). Sin embargo, algunas veces es también importante mostrar la interacción de los protocolos con otros mecanismos, que apoyan su funcionalidad (es decir la arquitectura funcional). Un ejemplo de un mecanismo, el cual no es un protocolo, es la expedición de paquetes de una cola basado en una disciplina particular de servicio.

El protocolo, sus funcionalidades y la arquitectura que soporta al mismo se describen usualmente en el documento de la especificación del protocolo. A veces no es posible incluir todos los componentes de una arquitectura de red particular. Por ejemplo, la arquitectura actual de Internet incluye muchos protocolos y aplicaciones. En este caso, la arquitectura puede mostrar solamente los componentes que son relevantes para modelar y analizar el protocolo de interés.

Después de determinar la relación entre los diversos mecanismos y protocolos implicados, es posible identificar el servicio que es proporcionado por el protocolo a otros componentes de la arquitectura (definición del servicio) y que será modelado. También, se puede conocer con claridad qué características del protocolo (definición del protocolo) necesitan ser modeladas.

2.4.1. Especificaciones del Servicio y del Protocolo

La especificación describe el servicio o el protocolo usando, por ejemplo, narrativa, diagramas de bloque o tablas de estado. Ejemplos de los documentos que incluyen tales especificaciones son los documentos de los estándares del ITU [1] y del IETF (i.e. Requerimientos para Comentarios o Request For Comments, RFCs).

Especificación del Servicio

La *especificación del servicio* describe el servicio que se proporciona al usuario. Esto se da a menudo como una secuencia de eventos que son posibles en una interfaz abstracta entre el usuario (una aplicación u otro protocolo) y el protocolo. La especificación del servicio se define en un nivel más alto de abstracción que la especificación del protocolo.

La especificación del servicio puede estar descrita en el documento de la especificación del protocolo o en un documento separado. Sin embargo, algunos documentos estándares no incluyen una especificación explícita del servicio [10] o el servicio no es especificado en términos de primitivas de servicio [7].

Una vez que los componentes de la arquitectura de protocolos y su relación se han establecido, es posible identificar los servicios que otros componentes de la arquitectura le proporcionan al protocolo y que serán modelados. Después de esto, el modelo se puede validar según lo explicado en la Sección 2.4.4. Por ejemplo, en [62] se describe la especificación del servicio de RSVP y en [38] se detalla la especificación del servicio de la gestión de conexiones de la capa MAC del estándar IEEE 802.16.

Especificación del Protocolo

La especificación del protocolo incluye una descripción detallada de las características del protocolo, que permiten que este proporcione los servicios explícitos o implícitos. Dicha especificación consiste en un conjunto de reglas, de formatos, y de procedimientos para que dos o más entidades remotas se comuniquen a través de una red [4]. Algunos ejemplos de procedimientos son la iniciación y la terminación del intercambio de datos y la recuperación de los paquetes con error [23].

La *síntesis de protocolo* se puede utilizar para generar o para mejorar la especificación del protocolo, empleando la especificación del servicio y cierta información inicial sobre el mismo [3].

Después de determinar la relación entre los diversos mecanismos y protocolos implicados y de estudiar la descripción del protocolo, se puede saber con claridad las características del

[1] ver http://www.itu.int.

mismo que necesitan ser modeladas. Después de lo cual, se puede realizar la validación y la verificación del modelo (ver Sección 2.4.4).

2.4.2. Modelo

Se desarrolla un modelo basado en la especificación usando una técnica formal. La técnica formal usada en este libro para el modelado es las *Redes de Petri Coloreadas*, con la ayuda de la herramienta de software CPN Tools [50].

2.4.3. Propiedades Definidas

Una propiedad se refiere a una característica particular, que debe estar presente en un protocolo. Por ejemplo, Holzmann [23] define un conjunto de las características generales, que pueden aplicarse a cualquier protocolo, tales como: ausencia de abrazos mortales inesperados (*deadlocks*) y ausencia de *livelocks*. Algunas propiedades particulares, que se aplican al protocolo bajo estudio, también pueden ser definidas. Las mismas pueden ser generadas a partir de la especificación del protocolo.

2.4.4. Técnicas de Análisis del Modelo

La simulación se puede utilizar para eliminar errores iniciales del modelo. Por ejemplo, la simulación automática o interactiva proporcionada por CPN Tools (ver la Sección 3.5.2) puede ayudar a encontrar errores, tales como inscripciones erróneas en arcos y secuencias de eventos del protocolo erróneas. Mientras se encuentren errores en la simulación, se modifica el modelo y las actividades de simulación se repiten.

Una vez depurado el modelo se procede a su análisis. El análisis del modelo del protocolo se puede dividir en: *verificación de las propiedades* y *verificación contra la especificación del servicio*. La verificación de las propiedades requiere la prueba de las propiedades deseadas del protocolo. La verificación contra la especificación del servicio consiste en comprobar si el lenguaje del protocolo satisface el lenguaje del servicio. Varios métodos para el análisis formal de los protocolos de comunicación se han definido (por ejemplo, análisis del grafo de estado, invariantes del sistema, lógica temporal, chequeo de modelos). Holzmann [23] describe algunos de estos métodos. Además, múltiples ejemplos de técnicas formales de análisis y de sus aplicaciones se pueden encontrar en los artículos presentados en [49]. Ejemplos más específicos se presentan en [32], el cuál describe los métodos usados para analizar modelos CPN. En este libro se emplea el método del grafo de estado incluido en CPN Tools (también llamado grafo de ocurrencias, *occurrence graph* (OG) o grafo OCC). Las razones para utilizar el método del grafo de estado se explican en la Sección 3.3.9.

El modelo de la especificación del servicio del protocolo no necesita ser verificado, puesto que define qué servicios se requieren del protocolo. En este caso, el análisis tiene la finalidad de validar y eliminar errores del modelo.

Los resultados del análisis pueden arrojar algunos errores. Los errores necesitan ser analizados para determinar sus causas y pueden ser una consecuencia de, por ejemplo, un error en el modelo (como el caso de las inscripciones erróneas), una inexactitud de la especificación o de las asunciones hechas en el modelo. Así, el modelo puede o no ser modificado. Si el modelo requiere la modificación entonces las actividades de simulación y análisis se deben repetir.

2.4.5. Lenguaje

Un *alfabeto* se define como conjunto no vacío de símbolos y se denota como Σ [34]. Por ejemplo, en [62] se define el alfabeto del servicio del protocolo de comunicación RSVP. Otros ejemplos se pueden encontrar en [3] y en [38]. Una secuencia finita de símbolos sobre el alfabeto se llama una *palabra* o *secuencia* y es denotado por *W*. Cualquier colección de palabras se llama *lenguaje* sobre el alfabeto, Σ.

De acuerdo con las definiciones antedichas, un *lenguaje del servicio* se puede definir sobre el alfabeto que consiste en todos las primitivas de servicio del protocolo de comunicación (es decir el alfabeto del servicio). El lenguaje consiste de todas las secuencias de ocurrencias de las primitivas de servicio posibles, que pueden suceder entre los usuarios del protocolo, donde una secuencia de primitivas es una palabra en el alfabeto del servicio. El lenguaje del servicio se escribe como L_S.

El *lenguaje del protocolo* está definido sobre el mismo alfabeto (es decir alfabeto del servicio) y consiste en todas las secuencias de las ocurrencias de las primitivas de servicio posibles, que se pueden generar por las entidades del protocolo. El lenguaje del protocolo se escribe como L_P.

Los lenguajes del servicio y del protocolo se pueden obtener de los grafos de estado generados por los modelos. Una técnica de reducción se utiliza para reducir al mínimo el grafo de estado para solamente mostrar las secuencias de las primitivas de servicio (el lenguaje del servicio o del protocolo). Los pasos implicados en la generación del lenguaje dependen de la herramienta, que soportan las tareas de modelado y análisis. Una explicación detallada de estos pasos esta fuera del alcance de este libro y se detalla en [62].

2.4.6. Comparación del Lenguaje

El lenguaje (del servicio o del protocolo) resultante puede ser analizado. El análisis puede consistir en comprobar que todas las secuencias de las primitivas de servicio son las esperadas. Las tablas de las primitivas de servicio en la especificación se pueden utilizar para validar el lenguaje. Sin embargo, el análisis del lenguaje puede ser difícil si el número de secuencias es grande. También, la herramienta usada para la generación del lenguaje puede no proporcionar la ayuda apropiada para el análisis del mismo.

El análisis del lenguaje puede mostrar algunos errores. De una manera similar al análisis del grafo de estado, las causas de los errores deben ser determinadas. Esas causas se pueden relacionar con, por ejemplo, una inexactitud de la especificación, el modelado de las asunciones o un error en el modelo. Cuando sea posible, se modifica el modelo y la simulación, el análisis, y las actividades relacionadas a la generación y análisis del lenguaje se repiten.

2.4.7. Comparación del Lenguaje

Se espera que los lenguajes del servicio y del protocolo sean equivalentes, es decir todas las secuencias (es decir secuencias de las ocurrencias de las primitivas de servicio) en el lenguaje del protocolo, L_P, deben estar en el lenguaje del servicio, L_S, y viceversa. Esto es denotado como $L_P = L_S$.

Si $L_P \neq L_S$, las secuencias de las primitivas de servicio, que están en el lenguaje del protocolo, pero no en el lenguaje del servicio o viceversa, necesitan ser analizadas. Estas secuencias pueden ser un resultado de un error en el modelo, una inexactitud de la especificación o debido a las asunciones de modelado y alcance. El modelo puede necesitar ser modificado. Si es así, las actividades de simulación y de análisis deben ser repetidas.

3. Técnicas y Herramientas Formales

3.1. Introducción

Los métodos formales abarcan una variedad de técnicas de modelado basadas en matemáticas, que son aplicables a los sistemas informáticos. Ellas son útiles en la construcción y el mantenimiento de los protocolos de comunicación complejos y permiten que las especificaciones sean formalmente analizadas y verificadas. Una amplia gama de métodos formales se han desarrollado, sin embargo, este libro no tiene la finalidad de presentar un tratamiento detallado de los mismos, centrándose solamente en las *Redes de Petri Coloreadas* (*CPNs*) [31].

Este capítulo proporciona una introducción informal de las CPNs, de la técnica de análisis denominada lógica temporal y de la herramienta de software que soporta su uso práctico, llamada CPN Tools [50].

3.2. Redes de Petri

Las Redes de Petri [42][51][52] son una técnica gráfica con una fundación matemática sólida y se pueden utilizar para describir y estudiar varios sistemas informáticos caracterizados por ser concurrentes, asincrónicos, no deterministas, paralelos y distribuidos. Una explicación detallada de las Redes de Petri y de sus formalismos está fuera del alcance de este libro; sin embargo se puede encontrar en [42][51]. En su lugar, esta sección tiene la finalidad de dar una breve descripción de los conceptos y de las definiciones principales relacionadas con las Redes de Petri por medio de algunos ejemplos.

Las Redes de Petri ordinarias también se llaman sistemas de Plazas / Transiciones (sistemas PT). Los componentes principales de las Redes PT son *las plazas*, *transiciones* y *arcos* y se muestran en la Figura 3.1. Las plazas se dibujan como elipses o círculos mientras que las transiciones se dibujan como rectángulos. Los arcos se dirigen de una plaza a una transición o de una transición a una plaza y pueden tener un *peso* (número entero positivo) asociado a ellos (el peso por defecto de un arco es uno y no se muestra). Dependiendo del sistema que se modelará, las plazas y las transiciones pueden tener diversas interpretaciones [42]. Por ejemplo, una plaza puede representar una condición y una transición un evento. También, una plaza puede representar recursos y una transición una tarea o un trabajo, que requieren esos recursos.

Finalmente, una plaza puede representar el estado del sistema y una transición, una acción que pueda ser tomada, basada en ese estado.

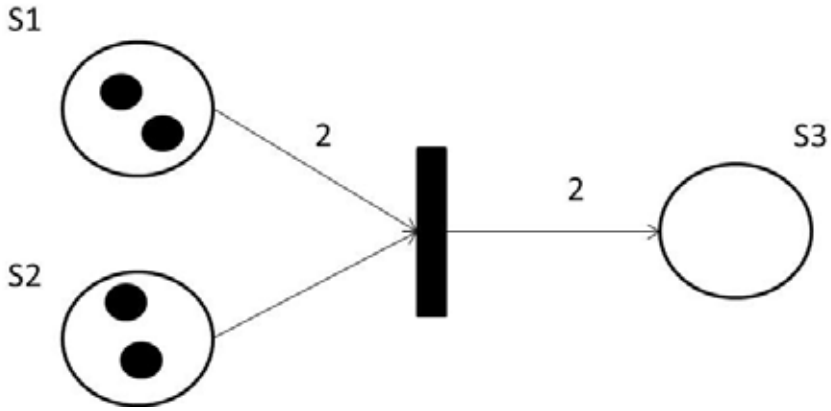


Figura 3.1: Ejemplo una Red de Petri.

Las plazas pueden contener una o más *marcas (tokens)*, las cuales se dibujan como puntos negros (ver la Figura 3.1). Una distribución de las marcas en las plazas se llama un *marcado* (es decir *marcado* de la red). El estado inicial del sistema se llama el *marcado inicial*. Un *marcado de la plaza* indica el número de marca*s* en la plaza particular.

El comportamiento dinámico de un sistema PT se puede considerar como el estado o el marcado de la red cambiando según las ocurrencias de las transiciones. Una transición puede tener *plazas de entrada* conectadas por los arcos entrantes y plazas de salida conectadas por los arcos salientes. Una transición está *habilitada* (es decir puede ocurrir) si el marcado de cada plaza de entrada contiene tantas marca*s* como lo indicado por el peso del arco de entrada, que conecta las plazas con la transición. La ocurrencia de una transición habilitada toma marca*s* de las plazas de entrada y agrega marca*s* a las plazas de salida. El número de marca*s* removidas de cada plaza de entrada corresponde al número de marca*s* indicadas por el peso del arco (de entrada), que conecta las plazas y la transición. El número de marca*s* agregadas a cada plaza de salida corresponde al número marca*s* indicadas por el peso del arco (de salida), que conecta la transición con las plazas. La Figura 3.2 presenta un ejemplo de la ocurrencia de la transición t1 en la Figura 3.1. Después que la transición t1 ocurre, dos marca*s* se remueven de la plaza S1 y uno de la plaza S2, y dos marca*s* se agregan a la plaza S3.

3.3. Redes de Petri Coloreadas

Mientras que los sistemas se tornan más complejos, los modelos basados en las Redes de Petri pueden llegar a ser muy grandes, complejos y probablemente ilegibles [31]. Este problema ha sido superado introduciendo una nueva clase de Redes de Petri, llamadas *Redes de Petri de Alto Nivel* [27]. Las *Redes de Petri Coloreadas* son redes de alto nivel que incorporan algunas definiciones, tales como tipos de datos y el procesamiento de valores de datos encontrados en los lenguajes de programación [31]. En esta sección, las CPNs se introducen con una versión

preliminar del modelo del establecimiento de la conexión Bandabase de Bluetooth presentado el Capítulo 4. Una explicación más detallada y las definiciones formales de las CPNs se pueden encontrar en [31].

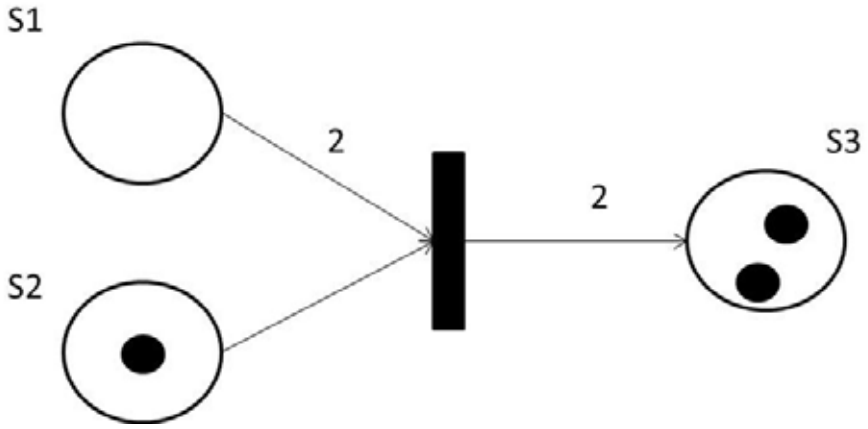


Figura 3.2: Una ilustración de la ocurrencia de la transición t1.

Una red Bluetooth está formada por un maestro y uno a más esclavos como se explica en el Capítulo 4 y en [8]. El establecimiento de la conexión Bandabase de Bluetooth se inicia cuando un dispositivo busca otros dispositivos que estén en su proximidad llamándose este procedimiento de búsqueda o *inquiry*. Un dispositivo por su parte debe estar en modo de "ser buscado" o *inquiry scan* para poder ser encontrado. La Figura 3.3 muestra una descripción del sistema que consiste dos dispositivos conectados a través de una red Bluetooth.

Figura 3.3: Topología de la red Bluetooth del ejemplo.

3.3.1. Modelo CPN

Similarmente a los modelos basados en las Redes de Petri, los modelos de CPN se crean como dibujos. La Figura 3.4 muestra el modelo del ejemplo mostrando los componentes básicos de las CPNs los cuales se describen a continuación.

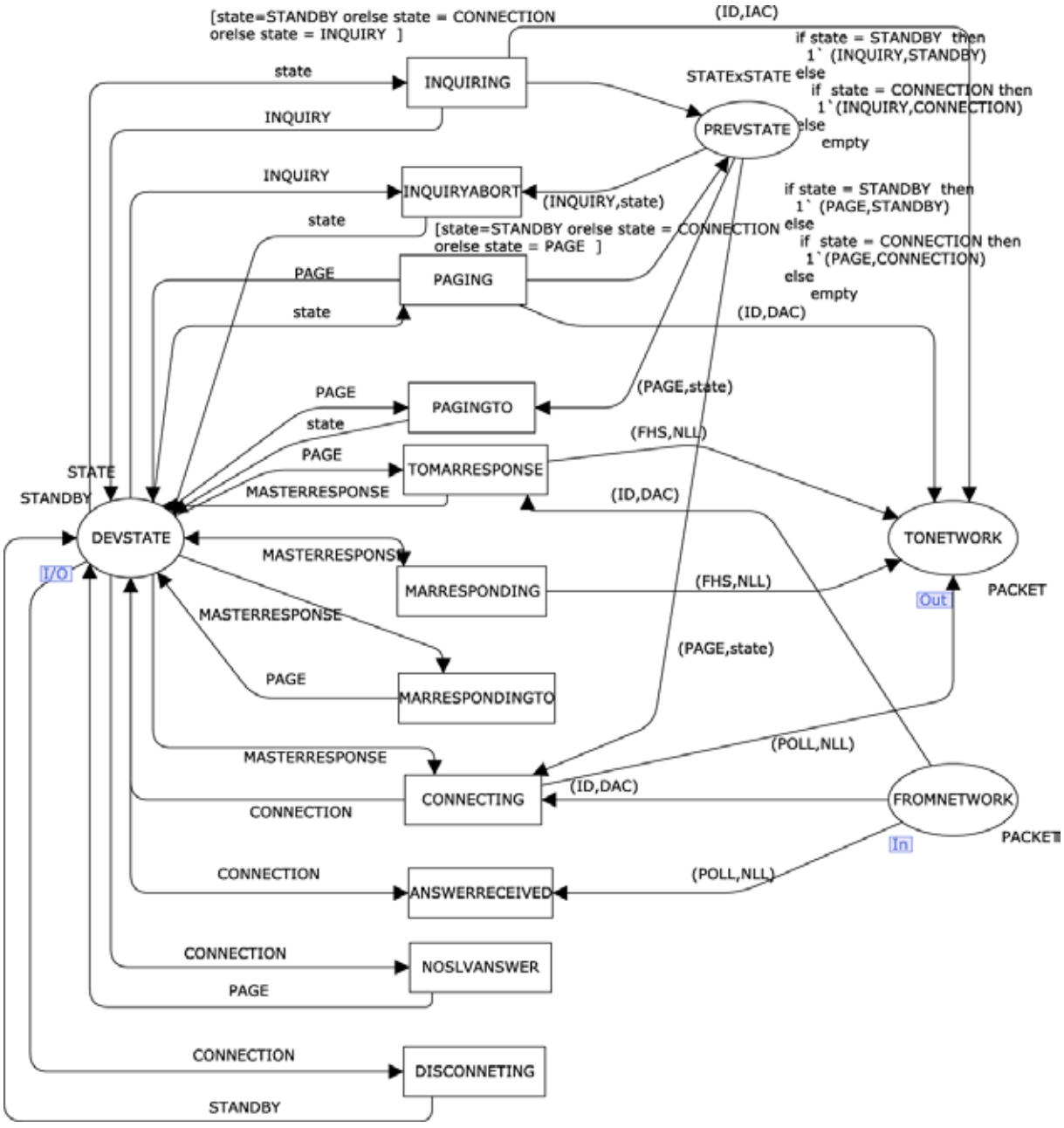


Figura 3.4: Módulo del establecimiento de una conexión Bluetooth en el maestro.

3.3.2. Tipos

Cada plaza tiene un *tipo* asociado o un *conjunto de colores* (*color set*), el cuál determina el tipo de datos que la plaza puede tener. Ellos se escriben en el modelo a un lado de la plaza. Las definiciones de tipo se muestran en la Figura 3.5 y son similares a los tipos en los lenguajes de programación.

```
(* Standard declarations *)
    colset STATE = with STANDBY|
                INQUIRY|INQUIRYSCAN|
                INQUIRYRESPONSE|
                PAGE|PAGESCAN|
                MASTERRESPONSE|
               SLAVERESPONSE|
               CONNECTION;
     colset TYPE = with ID|FHS|POLL;
     colset AC = with IAC|DAC;
     colset PACKET = product TYPE * AC;
     colset STATExSTATE = product STATE * STATE;
     colset IND =with  NONE1;
     var state: STATE;
     var prevstate: STATE;
     var anypacket: TYPE;
     var state2: STATE;
     var packettype: TYPE;
     var par: AC;
```

Figura 3.5: Definición de los conjuntos de colores.

El *color set* STATE es del tipo enumerado y representa los estados en los cuales un dispositivo intentando establecer una conexión Bandabase puede estar. El *color set* TYPE es un enumerado y representa los tipos de paquetes involucrados en el intercambio de mensajes entre el maestro y los esclavos que están intentando establecer una conexión. El *color set* AC es también un tipo enumerado y contiene el parámetro de control de acceso usado para identificar dispositivos durante el establecimiento de una conexión. El *color set* PACKET es el producto del tipo TYPE y AC y representa un paquete Bandabase. El *color set* STATExSTATE es el producto de STATE por STATE. El *color set* IND es un enumerado con un solo valor usado para controlar ciertas acciones en el modelo. Las demás declaraciones corresponden a variables (var) usadas en el modelo.

3.3.3. Plazas

En la figura hay cuatro plazas dibujadas como elipses. La elipse DEVSTATE es del tipo STATE y representa los estados del establecimiento de una conexión Bandabase en los cuales un dispositivo puede estar. Las plazas TONETWORK y FROMNETWORK son del tipo PACKET y representan los paquetes Bandabase que viajan hacia la red o vienen de la red, respectivamente. La plaza PREVSTATE es del tipo STATExSTATE y representa el estado inmediatamente anterior en que se encontraba un dispositivo.

3.3.4. Marcados

Las marcas se asocian a cada plaza. Una *marca* (*token*) es un valor (*color*), que pertenece al tipo de la plaza. El *marcado* de una plaza es el *multi-conjunto* (*multi-set*) de marcas presentes en la plaza. Es un *multi-conjunto*, puesto que puede contener varias marcas con el mismo valor. Por ejemplo, la plaza TONETWORK puede tener un marcado 2`(ID,IAC), lo cual significa que la plaza tiene dos marcas, cada una con el valor (ID,IAC). Significa que el maestro ha enviado dos paquetes de INQUIRY con el mismo código de acceso IAC.

Las CPNs incluyen el estado inicial del sistema, que se denomina *marcado inicial* y la cual se escribe al lado de la plaza. En el marcado inicial, la plaza DEVSTATE es inicializada con el estado de STANDBY y las plazas restantes no contienen ninguna marca.

3.3.5. Transiciones

Las transiciones representan las acciones del sistema. Se dibujan como rectángulos en la Figura 3.4. La misma incluye once (11) transiciones, las cuales representa las acciones para pasar de un estado a otro. La transición INQUIRING modela las acciones a través de las cuales el maestro colecta información de otros dispositivos Bluetooth cercanos, a través de la transmisión de paquetes ID enviados periódicamente. Un dispositivo sale del estado de INQUIRY, INQUIRYABORT, cuando el Manejador de Recursos de Bandabase decide que hay suficientes número de respuestas de los esclavos, cuando se ha alcanzado un *timeout* (*inquiryTO*) o cuando el procedimiento es cancelado por el *host*. En dicho caso el dispositivo debe retornar al estado desde el cual entró al estado de INQUIRY y el cual es almacenado en la plaza PREVSTATE.

La transición PAGING modela las acciones ejecutadas por el (potencial) maestro, destinadas a activar y conectarse a un esclavo. Un dispositivo sale del estado de PAGE, PAGINGTO, cuando un *timeout* (*pageTO*) es excedido. En cuyo caso debe retornar al estado en el cual se encontraba cuando inicio el procedimiento de PAGING y el cual está almacenado en la plaza PREVSTATE. Si estando en el estado de PAGE, el maestro recibe una respuesta del esclavo, se debe cambiar al estado de MASTERRESPONSE y enviar un paquete FHS con ciertos parámetros necesarios para el establecimiento de la conexión posteriormente; esto es modelado por la transición MARRESPONSE. En este modelo, estos parámetros han sido ignorados porque no son necesarios para modelar las transiciones entre estados. El maestro debe mantenerse enviado paquetes FHS, lo cual es modelado por la transición MASTERRESPONDING. El maestro ejecuta esta acción hasta que un *timeout* (*pagerespTO*) es excedido, modelado por MARRESPONDINGTO, o una segunda respuesta del esclavo, es decir, un paquete ID, con un código de acceso DAC, es recibido. En este último caso, el maestro ejecuta las acciones para finalmente establecer una conexión y las cuales son representadas por la transición CONNECTING. El maestro entonces se cambia al estado de CONNECTION. El maestro envía su primer paquete de tráfico denominado POLL al esclavo. Luego el maestro debe esperar una respuesta del esclavo, representada por un paquete de cualquier tipo. Con la finalidad de evitar

una explosión de estados innecesaria durante el análisis del modelo, se utiliza un paquete del tipo POLL. La transición ANSWERRECEIVED modela la recepción de este paquete por parte del maestro. Si el maestro no recibe este paquete en cierto tiempo (*newconnectionTO*), debe retornar al estado de PAGE. Esto es modelado por la transición NOSLVANSWER. Finalmente, el maestro puede desconectarse, DISCONNETING, e ir al estado de STANDBY en cualquier momento.

3.3.6. Arcos

Los arcos conectan transiciones y plazas. Una transición puede tener plazas de entrada conectadas por los arcos entrantes y plazas de salida conectadas por los arcos salientes. Los arcos tienen expresiones asociadas con ellos, las cuales están situadas al lado de los arcos y determinan cuales marcas son removidas o agregadas a las plazas según lo explicado en la sección siguiente.

3.3.7. Comportamiento Dinámico

El comportamiento dinámico del sistema CPN se puede describir como el cambio del marcado de la red según las ocurrencias de la transición, que dependen de las expresiones de los arcos circundantes.

Variables y Asociaciones (*Bindings*)

Una expresión del arco es evaluada asignando valores de los datos (*binding*) a las variables. El resultado de la evaluación de una expresión del arco es un multi-conjunto de marcas. La declaración de variables del ejemplo se muestra en la Figura 3.5.

Una asociación (*binding*) se representa de la forma $\{v_1 = d_1, v_2 = d_2 \ldots, v_n = d_n\}$ donde v_i, $i \in \{1,2..n\}$ es una variable y d_i es el valor de los datos asignados a v_i. Por ejemplo, en la Figura 3.6 los valores de las marcas se incluyen en cajas y el número de marcas en círculos pequeños. La plaza DEVSTATE tiene un marcado actual que consiste de una marca 1`PAGE, la plaza PREVSTATE tiene una marca 1`(PAGE,STANDBY) y la plaza TONETWORK tiene una marca 1`(ID,DAC). En este marcado del sistema, dos ejemplos de asociaciones son posibles:

$b_1 = \{state = PAGE\}$

$b_2 = \{state = STANDBY\}$

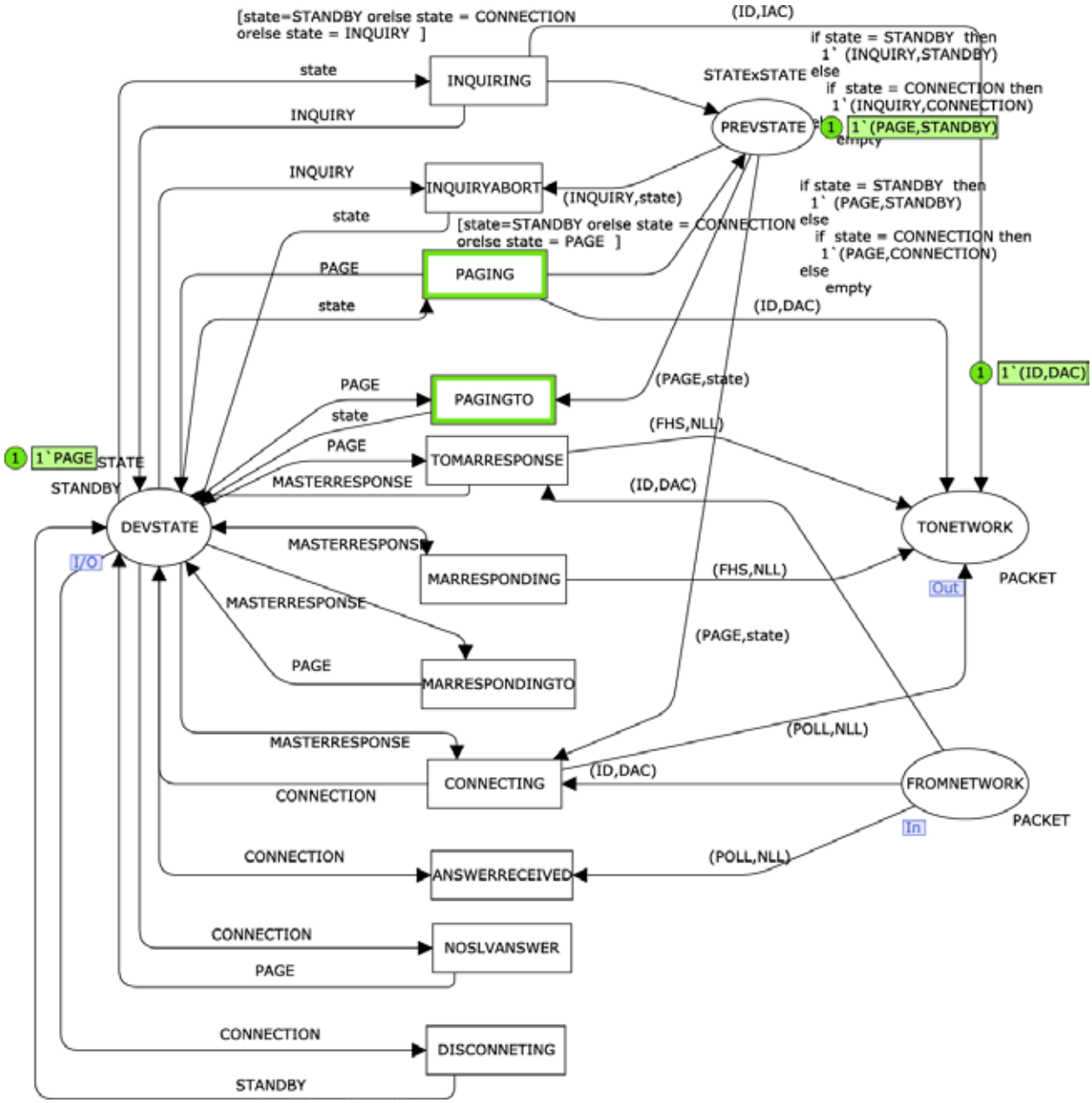


Figura 3.6: Un ejemplo del marcado del sistema.

Para la asociación b_1, la expresión del arco asociada con el arco entrante de la transición PAGING evalúa al multi-conjunto de marcas 1`PAGE. Para la asociación $b_2$, la expresión del arco asociada con el arco entrante de la transición PAGINGTO evalúa a 1`STANDBY.

Una transición puede tener una expresión booleana asociada a ella denominada *guarda* y se incluye entre corchetes. Similarmente a la expresión de un arco, una guarda puede tener variables. La guarda tiene que evaluar verdad para aceptar la asociación. En la Figura 3.4, la transición INQUIRING tiene una guarda asociada a él, la cual especifica que el estado del

maestro debe ser STANDBY, CONNECTION o INQUIRY para realizar un proceso de indagación (*inquiry*).

Asociaciones y Ocurrencia de las Transiciones

Una transición puede ocurrir si está *habilitada* (*enabled*). Para que una transición este habilitada en el marcado actual, debe ser posible asociar (asignar) valores de los datos a las variables que aparecen en las expresiones circundantes al arco y en la guarda y las siguientes condiciones debe ser satisfechas. En primer lugar, cada una de las expresiones de los arcos entrantes evalúa a las marcas que están presentes en las plazas de entrada correspondientes. En segundo lugar, si hay cualquier guarda, debe evaluar a verdad.

La ocurrencia de una transición remueve marcas de las plazas entrantes y agrega marcas a las plazas salientes. Las marcas removidas son el resultado de evaluar las expresiones en los arcos entrantes correspondientes, mientras que los valores de las marcas agregadas son el resultado de evaluar las expresiones del arco en los arcos salientes correspondientes. Por ejemplo, la transición PAGINGTO está habilitada en el marcado actual que se muestra en la Figura 3.6 para la asociación b_2 (introducido en la sección pasada). La Figura 3.7 ilustra el marcado resultante de la ocurrencia de la transición.

La ocurrencia de la transición PAGINGTO actualiza el marcado de la plaza DEVSTATE. Así, la marca que representa el estado del maestro muestra que su estado es de STANDBY. También remueve una marca de la plaza PREVSTATE, quien queda sin ninguna marca en este estado del sistema.

Puede haber más de una asociación posible para las expresiones de arco que rodean una transición. Estas asociaciones definen la manera que la transición puede ocurrir. Sin embargo, para un marcado dado, la transición estará habilitada para un subconjunto de ellos.

La ejecución de un modelo CPN se puede considerar como una *secuencia de ocurrencias* consistentes de marcados que se alcanzan y de *pasos*. Un paso consiste potencialmente de varios elementos de asociación (*binding*) permitidos que ocurren concurrentemente. Un *elemento de asociación* incluye una transición y una asociación de sus variables. Por ejemplo, en el marcado mostrado en la Figura 3.6, son dos los elementos de asociación permitidos y se muestran a continuación:

be_1 = (MASTERCONNECTIONSETUP'PAGING,{state = PAGE})

be_2 = (MASTERCONNECTIONSETUP'PAGINGTO,{state = STANDBY})

Los elementos de asociación be_1 y be_2 están en *conflicto*, puesto que no pueden conseguir la única marca (es decir PAGE) en la plaza DEVSTATE simultáneamente. Por lo tanto, las transiciones PAGING y PAGINGTO necesitan compartir la marca. Significa que el maestro puede realizar una operación de *paging* o el período de *paging* puede expirar.

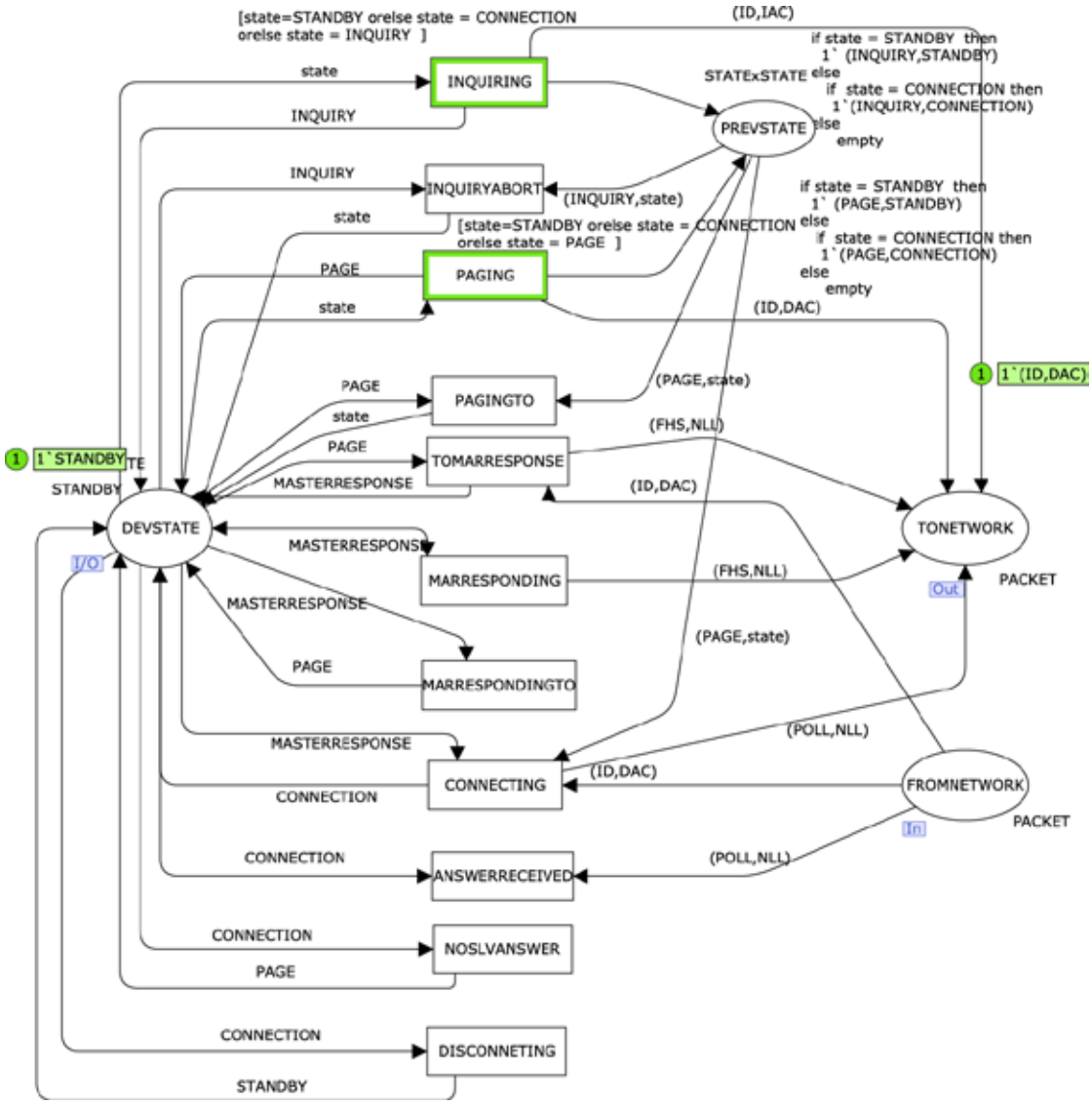


Figura 3.7: Marcado resultante de la ocurrencia de la transición PAGINGTO habilitada en la Figura 3.6.

Dos o más elementos de asociación pueden estar *concurrentemente habilitadas*, cuando el multi-conjunto de marcas que resulta de la evaluación de las expresiones de los arcos entrantes que están presentes en las plazas entrantes correspondientes y las transiciones utilizan conjuntos disjuntos de marcas. Así, estos elementos de asociación pueden ocurrir simultáneamente.

3.3.8. CPNs Jerárquicas

Las CPNs jerárquicas permiten que los modelos sean construidos en una manera modular y estructurada. Los diversos niveles jerárquicos muestran los diferentes niveles de detalle del modelo. Así, el ejemplo del establecimiento de la conexión Bluetooth se puede modelar a un alto nivel de abstracción según lo mostrado en la Figura 3.8. De esta manera, las CPNs jerárquicas permiten la construcción de modelos grandes y complejos de una manera más compresiva.

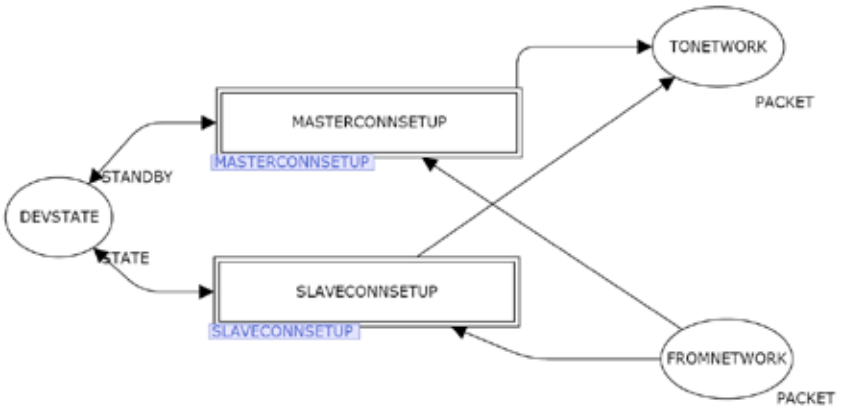


Figura 3.8: Vista jerárquica del ejemplo de Bluetooth.

Módulos

Un modelo CPN puede estar conformado por un número de módulos. Las CPNs jerárquicas proporcionan dos mecanismos para interconectar estos módulos: *transiciones de substitución* y *plazas fusionadas* [31].

Transiciones de Substitución

Las *transiciones de substitución* permiten que los modelos sean construidos en una manera *top-down*. Si una transición es una transición de substitución, tiene un sub módulo relacionado con ella, que incluye una descripción más detallada del modelo. Por ejemplo, en la Figura 3.8, la transición MASTERCONNSETUP es una transición de substitución, lo cual es indicado por el rectángulo sombreado con el nombre de dicha transición de sustitución y el doble recuadro en la transición. El sub módulo de esta transición se muestra en la Figura 3.4 y fue descrito detalladamente en las secciones anteriores.

Los módulos de una CPN se interconectan a través de plazas *puertos* y *zócalos*. Los módulos tienen plazas puertos, que les permiten recibir, entregar o recibir y entregar marcas de los módulos de un nivel más alto. Por ejemplo, en la Figura 3.4, las plazas DEVSTATE, TONETWORK y FROMNETWORK son los puertos entrantes y están marcados con las etiquetas, I/O, Out, In, respectivamente.

Las plazas conectadas con la transición de substitución MASTERCONNSETUP (Figura 3.8), tienen tres plazas entrantes, que se llaman *zócalos*. Los zócalos son relacionados con las plazas puertos en los correspondientes módulos proporcionando las *asignaciones de puertos*. Por ejemplo, la plaza puerto TONETWORK en la Figura 3.4 ha sido asignada a la plaza zócalo TONETWORK en la Figura 3.8, por lo cual son idénticas (es decir tienen el mismo marcado).

Plazas de Fusión

Las *plazas de fusión* incluyen un conjunto de plazas, que son funcionalmente idénticas, por lo cual tienen el mismo marcado. Un conjunto de plazas de fusión es un *conjunto de fusión*. Los miembros de un conjunto de fusión pueden pertenecer a un solo módulo o estar distribuidos a través de diversos módulos. Una plaza global de fusión tiene una etiqueta con el nombre del conjunto de fusión al lado.

3.3.9. Análisis de las CPNs

El modelo CPN puede ser simulado. Las simulaciones permiten que el usuario entienda y elimine los errores del modelo. Sin embargo, no es posible probar propiedades del modelo por medio de la simulación a menos que el sistema sea trivial. Así, se han creado varios métodos de análisis. En este libro, se utiliza el método del *grafo de estado*. Una descripción de otras técnicas se puede encontrar en [31][32].

El grafo de estado incluye todos los marcados posibles que se puedan alcanzar desde el marcado inicial y se representa como un grafo dirigido donde los nodos representan los marcados y los arcos los elementos de asociación que ocurren. El grafo de estado también se conoce como, *espacio de estado* (*state space*), *grafo de ocurrencia* o *grafo de accesibilidad*.

Comparado con otras técnicas de análisis, tales como el análisis de invariantes, los métodos de grafo de estado tienen varias ventajas [31]. En primer lugar, el grafo de estado puede ser construido automáticamente, lo que permite el análisis y verificación automatizada del comportamiento del sistema modelado. Además, la herramienta de grafo de estado se integra completamente en CPN Tools, que es el software usado en este libro. En segundo lugar, el grafo de estado incluye mucha información sobre el comportamiento del sistema, que puede contestar a un gran conjunto de preguntas de análisis y verificación. En tercer lugar, el método del grafo de estado se puede utilizar para eliminar errores y probar el sistema. Por ejemplo, en algunos casos, el modelo del sistema puede generar un número muy grande o inclusive infinito de marcados. En tales casos, el grafo de estado no puede ser generado, no obstante algunas propiedades se pueden probar o refutar basado en grafos de estado parciales incluyendo sub-grafos finitos del grafo de estado completo.

La principal desventaja del método de grafo de estado es el problema de la explosión de estados. El problema con grafos de estado muy grande es que no pueden ser generados con recursos de cómputo limitados (por ejemplo, memoria) y si se pueden generar, pueden ser difíciles de analizar. En las últimas décadas, los investigadores han estado trabajando para intentar encontrar maneras de solucionar o de aliviar el problema. Así, se han desarrollado varias técnicas de reducción del grafo de estado [31].

Las ventajas del método del grafo de estado enumeradas previamente lo hacen una buena opción para analizar y para verificar los protocolos de comunicación descritos en este libro.

Para ilustrar el concepto anterior se ha modificado el grafo de estado del ejemplo ilustrado en la Figura 3.4 para obtener un grafo cuyas dimensiones lo hagan fácilmente desplegable y entendible. Primero, las transiciones relacionadas al proceso de *paging* han sido eliminadas y solo se han dejado las transiciones INQUIRING e INQURINGABORT. Segundo, el modelo genera un OG infinito debido a los paquetes periódicos (de *inquiry*) que envía el maestro, y al hecho de que las plazas de comunicación (FROMNETWORK y TONETWORK) no están acotadas. Así, se ha modificado el modelo de forma tal que las plazas de comunicación tengan una capacidad finita. La Figura 3.9 muestra el modelo modificado del establecimiento de una conexión Bandabase en el maestro. El paquete tipo SLOT ha sido incluido en el *color set* del tipo TYPE y un nuevo código de acceso del tipo nulo, NLL, ha sido incluido al tipo AC (ver Figura 3.5). Cada vez que se desea enviar un paquete debe existir suficiente capacidad en la plaza de comunicación TONETWORK; dicha capacidad viene determinada por el número de marcas (SLOT,NLL). Adicionalmente, se ha incluido la transición SENDPKT la cual modela el envío de un paquete a la red. El marcado inicial de la plaza DEVSTAT es STANDBY y el de la plaza TONETWORK es (SLOT,NLL), las demás plazas no tienen ninguna marca en este estado inicial. El grafo de estado del modelo modificado es mostrado en la Figura 3.10.

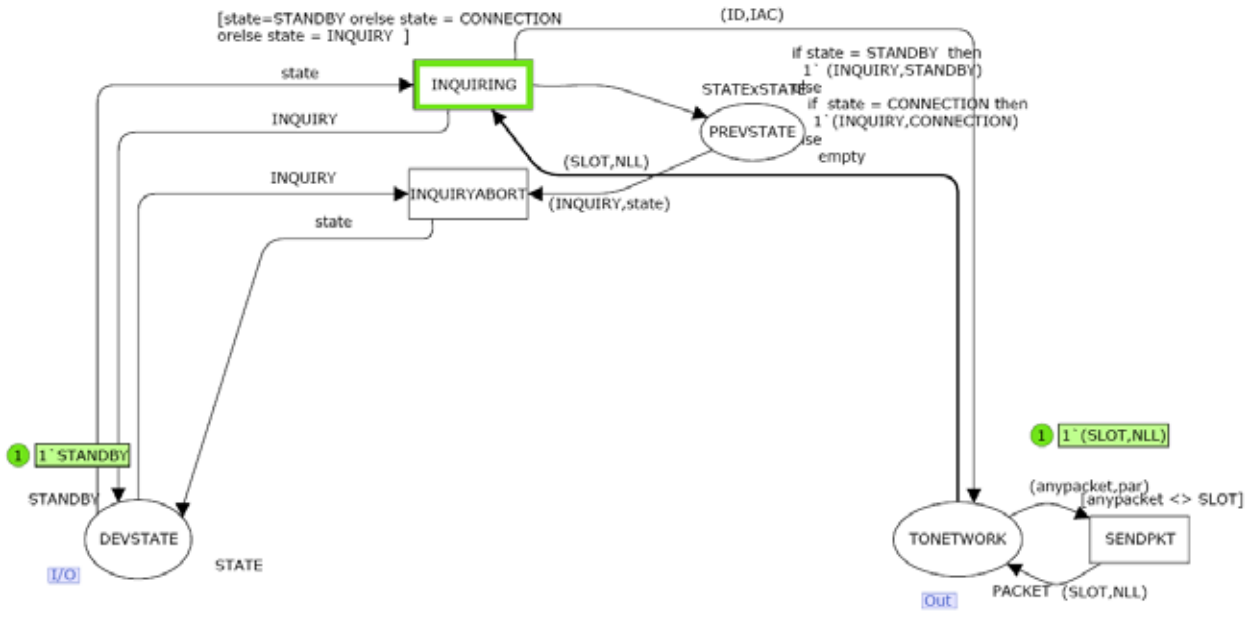


Figura 3.9: Módulo del establecimiento de una conexión en el maestro modificado.

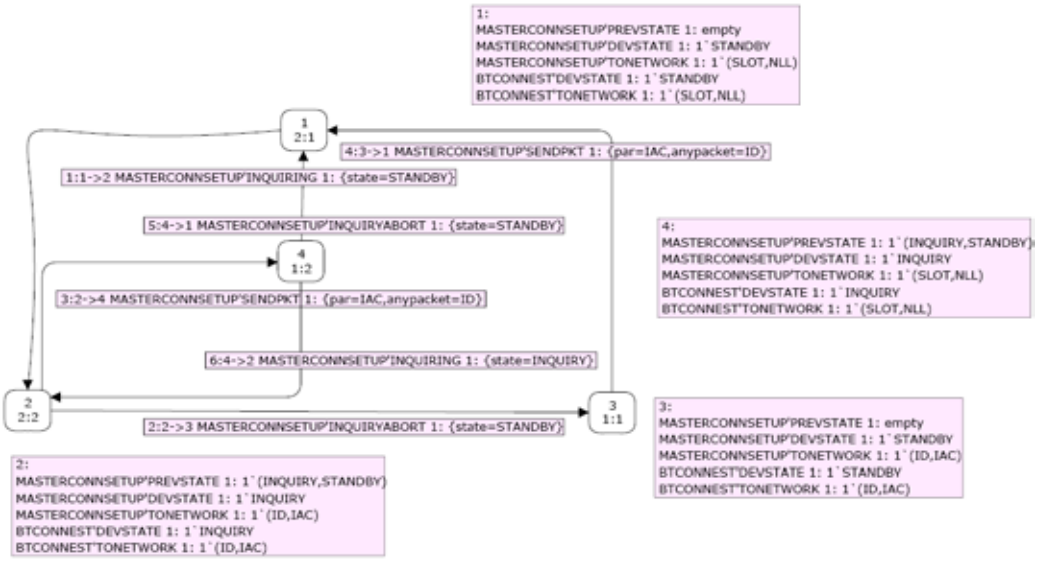


Figura 3.10: Grafo de estado del modelo CPN de la Figura 3.9.

Hay cuatro (4) marcados accesibles (o nodos) en la Figura 3.10, representados por las cajas redondeadas. Cada marcado tiene un número de identificación situado en el tope. También, en cada caja de marcado, hay dos números separados por dos puntos (":"), que representan el número de arcos de entrada y de salida, respectivamente. El nodo 1 es el marcado inicial. El marcado de cada plaza se muestra en cajas sombreadas localizadas próximas al nodo. Por ejemplo, en el marcado inicial (es decir nodo 1), la plaza DEVSTATE tiene una marca con el valor STANDBY, la plaza TONETWORK tiene una marca con el valor (SLOT,NLL) y la plaza PREVSTATE no tiene ninguna marca.

Un arco representa la ocurrencia de un elemento de asociación, el cual se muestra en la caja sólida próxima al arco. Esta también incluye el número de identificación del arco (situado en la esquina izquierda superior) y cierta información del marcado (encontrada al lado del número). La información del marcado, representada por el n_1->n_2, indica que el marcado n_2 se puede alcanzar desde el marcado n_1 cuando ocurre el elemento de asociación. Por ejemplo, el elemento de asociación *1* (representado por el arco 1) es el único permitido en el marcado inicial. Este incluye la transición INQUIRING y la asociación state = STANDBY (es decir el maestro ha comenzado a realizar indagaciones sobre los dispositivos Bluetooth próximos a él).

Componentes Fuertemente Conectados

Un *Componente Fuertemente Conectado (Strongly Connected Component, SCC)* del grafo de estado es un sub-grafo máximo, cuyos nodos son mutuamente accesibles entra cada uno de ellos [31]. Un grafo SCC tiene un nodo por cada SCC y arcos que conectan cada nodo del SCC con uno más nodos SCC. Un SCC sin arcos entrantes se llama *SCC inicial*, y un SCC sin arcos salientes se llama *SCC terminal*. Cada nodo en el grafo de estado pertenece solamente a un SCC, así que el grafo SCC será más pequeño o igual que el grafo de estado correspondiente. El

SCC del OG mostrado en la Figura 3.10 tiene un (1) nodo y ningún arco indicando que el sistema siempre retorna al estado inicial.

Propiedades de que Describen el Comportamiento de las CPNs

Las propiedades que describen el comportamiento previsto del modelo pueden ser definidas. En esta sección, las principales propiedades de comportamiento o dinámicas de las CPNs se describen de una manera informal. Más detalles sobre estas propiedades y sus definiciones formales se pueden encontrar en [31].

Accesibilidad (Reachability)

Por convención, Mn denota el marcado del nodo número *n*. Mn es *alcanzable* desde M_1 si hay una secuencia de ocurrencia desde de marcado M_1 al M_n. Por ejemplo, en la Figura 3.10, M_2 es directamente alcanzable M_1. Adicionalmente, todos los marcados son alcanzables por el resto de los marcados.

Acotamiento (Boundedness)

Las *cotas enteras superiores* e *inferiores* indican el número máximo y mínimo de marcas que se pueden colocar en cada plaza en los marcados alcanzables. En la Tabla 3.1 se muestran los valores de las cotas enteras del modelo modificado mostrado en la Figura 3.9. Cada una de las plazas excepto PREVSTATE puede contener una marca representando un estado o un mensaje. La plaza PREVSTATE puede tener un estado o ninguno.

Tabla 3.1: Cotas enteras de las plazas del modelo de la Figura 3.9

Plaza	**Mejores Cotas Enteras**	
	Superior	**Inferior**
DEVSTATE	1	1
TONETWORK	1	1
PREVSTATE	1	0

El otro concepto relacionado con las propiedades de acotamiento es el de *cotas de los multi-conjuntos,* las cuales proporcionan información sobre los valores de las marcas que las plazas pueden contener. La cota de multi-conjunto superior de una plaza se define como el más pequeño multi-conjunto que es más grande o igual que todos los marcados alcanzables de la plaza [31]. El límite de multi-conjunto inferior de una plaza se define como el más grande multi-conjunto que es más pequeño que o igual que todos los marcados accesibles de la plaza [31]. En la Tabla 3.2 se muestran los límites multi-conjuntos de todas las plazas del modelo modificado mostrado en la Figura 3.9. La plaza DEVSTATE puede tener una marca con un valor de STANDBY o una marca con un valor de INQUIRY. Mientras tanto la plaza TONETWORK puede tener una marca representando un paquete Bluetooth (ID,IAC) o uno representando un *slot*

vacío (SLOT,NLL). Finalmente la plaza PREVSTATE puede tener una marca con un valor indicado que el estado previo a INQUIRY fue STANDBY, es decir, (INQUIRY, STANDBY).

Tabla 3.2: Cotas multi-conjuntos de las plazas del modelo de la Figura 3.9

Plaza	**Mejores Cotas Enteras**	
	Superior	**Inferior**
DEVSTATE	1`STANDBY++1`INQUIRY	Vacía
TONETWORK	1`(ID,IAC)++1`(SLOT,NLL)	Vacía
PREVSTATE	1`(INQUIRY,STANDBY)	Vacía

Marcados Locales

Un *marcado local (home marking)* es un marcado que puede ser siempre alcanzado por el resto de los marcados alcanzables. En la Figura 3.10, todos los marcados son locales. Un *espacio local (home space)* es un conjunto de marcados tales que desde cada marcado alcanzable, es posible alcanzar por lo menos uno de estos marcados.

No Abrazos Mortales

Un *marcado muerto* (*dead marking*) es un marcado sin elementos de asociación habilitados. En la Figura 3.10, no hay marcados muertos. Un sistema se dice estar libre de abrazos mortales si ningún marcado muerto se puede alcanzar desde el marcado inicial.

Transiciones Muertas

Una *transición muerta (dead transition)* no está habilitada en ningún marcado alcanzable. En el modelo de la Figura 3.9, no hay transiciones muertas indicando que no hay código muerto en este sistema.

Transiciones Vivas

Una *transición viva* (*live transition*) es una transición la cual siempre puede ocurrir otra vez. Más exactamente, una transición *t* está viva si para cada marcado alcanzable, es posible alcanzar un marcado *t*. Una Red de Petri se dice estar viva si todas sus transiciones están vivas. La CPN de la Figura 3.9 está viva ya que todas sus transacciones están vivas.

3.4. Lógica Temporal

La lógica temporal es una rama de la lógica formal usada para la especificación y el análisis de programas concurrentes y en particular puede ser aplicada al análisis de sistemas distribuidos [21]. La lógica temporal permite la especificación de las propiedades de un sistema de forma precisa contrariamente a lo que se puede lograr con el lenguaje natural el cual es impreciso. Un tipo de lógica temporal es la *lógica temporal de tiempo enramada* (*braching time*

temporal logic), donde se asume que en cada instante de tiempo existen varias alternativas futuras, entonces se definen fórmulas que son interpretadas sobre árboles de estados. Un ejemplo de este tipo de lógica temporal es la lógica de árbol computacional (*computational tree logic, CTL*), la cual es considerada en este libro.

3.4.1. ASK-CTL

ASK-CTL [12] está basado en CTL [13], la cual ha sido adaptada para expresar las propiedades de los OGs de las CPNs. La sintaxis de ASK-CTL tiene dos categorías de fórmulas las cuales son mutuamente recursivas: de *estado* (*state)* y de *transiciones (transition)*, las cuales son expresadas sobre los dominios de los estados o de las transiciones de un OG.

Las fórmulas de estado son las siguientes:

$$A :: = tt \mid \alpha \mid \neg A \mid A_1 \wedge A_2 \mid <B> \mid EU(A_1,A_2) \mid AU(A_1,A_2) \mid EX(A) \mid AX(A) \mid Pos(A) \mid Inv(A) \mid Ev(A) \mid Along(A)$$

Las fórmulas de transición son las siguientes:

$$B :: = tt \mid \beta \mid \neg B \mid B_1 \wedge B_2 \mid <A> \mid EU(B_1, B_2) \mid AU(B_1, B_2) \mid EX(B) \mid AX(B) \mid Pos(B) \mid Inv(B) \mid Ev(B) \mid Along(B)$$

Ellas son definidas informalmente y formalmente en las siguientes secciones (ver también [12]). Se utiliza la siguiente notación en dichas definiciones: M denota los marcados de las CPNs, b denota los elementos de asociación y e denota los lados etiquetados del OG. Sea también (M,b,M') un lado del OG, donde el marcado M' es directamente alcanzable desde el marcado M por la ocurrencia del elemento de asociación b, i.e. $M \xrightarrow{b} M'$. Sea ρ_M el conjunto de caminos que inician en M, es decir, $\rho_M = \{M_0 b_1 M_1 b_2 \ldots \mid M_0 = M \text{ y } M \xrightarrow{b1} M_1 \xrightarrow{b2} M_2 \ldots\}$. Si un camino $\sigma = M_0 b_1 M_1 \ldots M_{n-1} b_n M_n \in \rho_M$ es finito, la longitud de σ es denotada como $|\sigma| = n$, de lo contrario $|\sigma| = \infty$.

Constantes Booleanas

tt representa el valor constante de verdad.

Operadores Constante Booleanos

¬ y ∧ son los operadores booleanos estándares.

Predicados Atómicos

α es una función que mapea marcados a booleanos, M → $\mathbb{B}$. β es una fórmula de transición que mapea elementos de asociación a booleanos, BE → $\mathbb{B}$.

Fórmulas de Estado

En esta sección se introduce la notación estándar para la fórmula de estado, $M \models_{ST} A$, que significa que la fórmula de estado A se cumple en el marcado M:

$$M \models_{ST} tt \text{ siempre se cumple}$$

$$M \models_{ST} \alpha \text{ si y sólo si } \alpha(M)$$

$$M \models_{ST} \neg A \text{ si y sólo si } \neg\, M \models_{ST} A$$

$$M \models_{ST} A1 \wedge A2 \text{ si y sólo si } M \models_{ST} A1 \wedge M \models_{ST} A2$$

Fórmula de Transición

En esta sección se introduce la notación estándar para la fórmula de transición, $M \models_{Tr} B$ que significa que la fórmula de transición B se cumple en la transición (M,b,M'):

$$M \models_{Tr} tt \text{ siempre se cumple}$$

$$M \models_{Tr} \beta \text{ si y sólo si } \beta(b)$$

$$M \models_{Tr} \neg B \text{ si y sólo si no } M \models_{Tr} B$$

$$M \models_{Tr} B_1 \wedge B_2 \text{ si y sólo si } M \models_{Tr} B_1 \wedge M \models_{Tr} B_2$$

Fórmula de Cuantificación de Camino

Las fórmulas de estado y transición son interpretadas sobre caminos. Un camino es una secuencia finita o infinita de ocurrencias de transiciones y estados. U (*until*) es el operador temporal y es combinado con los cuantificadores de camino E y A. El operador $EU(A_1,A_2)$ significa que existe un camino de un marcado dado tal que A_1 es verdad hasta que un marcado donde A_2 se cumple es verdad. El operador $AU(A_1,A_2)$ expresa que A_1 se cumple para todos los caminos hasta que un marcado donde A_2 se cumple es alcanzado. Las fórmulas de transición de cuantificación de camino pueden ser definidas de forma análoga. Las fórmulas se definen formalmente a continuación:

$$M \models_{ST} EU(A_1,A_2) \text{ iff } (\exists \sigma \in \rho_M, (\exists n \leq |\sigma| , (\forall\, 0 \leq i < n, M_i \models_{ST} A_1) \wedge M_n \models_{ST} A_2))$$

$$M \models_{ST} AU(A_1,A_2) \text{ iff } (\forall \sigma \in \rho_M, (\exists n \leq |\sigma| , (\forall\, 0 \leq i < n, M_i \models_{ST} A_1) \wedge M_n \models_{ST} A_2))$$

$$\alpha \models_{Tr} EU(B_1, B_2) \text{ iff } (\exists \sigma \in \rho_M, (\exists n \leq |\sigma| , (\forall 0 \leq i < n, (M_i, b_{i+1}, M_{i+1} \models_{Tr} B_1) \wedge (M_n, b_{n+1}, M_{n+1}) \models_{Tr} B_2))$$

$$\alpha \models_{Tr} AU(B_1, B_2) \text{ iff } (\forall \sigma \in \rho_M, (\exists n \leq |\sigma| , (\forall 0 \leq i < n, (M_i, b_{i+1}, M_{i+1} \models_{Tr} B_1) \wedge (M_n, b_{n+1}, M_{n+1}) \models_{Tr} B_2))$$

Fórmulas de Cuantificación de Caminos Derivadas

Algunos operadores han sido derivados de los operadores de cuantificación de camino descritos anteriormente. Ellos tienen la finalidad de incrementar la legibilidad de las fórmulas. *Pos(A*) expresa que es posible alcanzar un estado donde *A* se cumple. *Inv(A)* expresa que *A* es verdad para todos los estados alcanzables. *Ev(A)* significa que *A* se cumple dentro de un número finito de pasos para todos los caminos. Finalmente, *Along(A)* significa que existe un camino infinito o que finaliza en un marcado muerto para el cual *A* se cumple para cada estado. Los operados son formalmente definidos a continuación:

$$Pos(A) \underline{=} EU(tt,A)$$

$$Inv(A) \underline{=} \neg Pos \neg A$$

$$Ev(A) \underline{=} AU(tt,A)$$

$$Along \underline{=} \neg Ev \neg A$$

Abreviaciones y definiciones similares son usadas para las fórmulas de transición.

Fórmulas de Cambio de Dominios

Se define también el operador <*B*> el cual permite cambiarse de fórmulas de estado a fórmulas de transición. Similarmente, el operador <α> permite cambiarse de fórmulas de transición a fórmulas de estado.

$$M \models_{ST} <B> \text{ iff } (\exists b,\ M \xrightarrow{b} M' \wedge (M,b,M')\ M \models_{Tr} B)$$

$$\alpha \models_{Tr} <A> \text{ iff } M' \models_{ST} A$$

Fórmulas de Sucesor Inmediato

X es el operador *nexttime* y es combinado con los cuantificadores de camino *E* y *A*. *EX(A)* significa que *A* se cumple en alguno estado de sucesor inmediato. *AX(A)* significa que *A* se cumple en cada estado de sucesor inmediato. Antes de introducir la definición formal de estas fórmulas, se define la siguiente fórmula <*B*> *A,* la cual significa que existe un estado de sucesor inmediato, *M'*, en el cual *A* se cumple y *B* se cumple en la transición entre el estado en curso y *M'*.

$$<B> A \underline{=} <B \wedge <A>>$$

$$EX(A) \underline{=} <tt>A$$

$$AX(A) \underline{=} \neg EX(\neg A)$$

Las fórmulas de sucesor de transición inmediato son definidas de forma similar.

3.5. Herramienta de Software

CPN Tools es una herramienta de software que soporta la construcción, simulación y el análisis funcional y de rendimiento de los modelos CPN [31][50]. La herramienta fue originalmente desarrollada por el grupo de CPN de la Universidad de Aarhus y soportada por dicho grupo desde el año 2000 al 2010. Los principales diseñadores y guías del proyecto fueron Kurt Jensen, Søren Christensen, Lars M. Kristensen, y Michael Westergaard. Actualmente la herramienta es soportada por el grupo AIS de la Universidad Tecnológica de Eindhoven en Holanda [20].

En la actualidad la herramienta puede ser obtenida sin costo alguno y está disponible para los sistemas operativos Windows y Linux, sin embargo, las versiones más recientes solo están disponibles para el primero. La versión actual de CPN Tools es la 4.0.0 y fue liberada en septiembre de 2013 [15].

Una evaluación detallada de CPN Tools esta fuera del alcance de este libro, sin embargo, las principales características de la misma, las cuales se describen a continuación, la han hecho una de las mejores opciones para ser utilizada. La herramienta ha sido utilizada por varias organizaciones y compañías alrededor del mundo [15]. Se ha utilizado exitosamente para soportar modelos CPN de los protocolos de comunicación [15]. La herramienta soporta las tareas principales de modelado del sistema (es decir construcción, simulación y análisis del modelo CPN). CPN Tools está bien mantenida y documentada [15].

CPN Tools soporta la edición, simulación, análisis del grafo de estado (i.e. OG) y análisis del rendimiento de los modelos CPNs. El usuario interactúa directamente con la herramienta a través de la *interfaz gráfica de usuario* (*graphical user interface, GUI*). A continuación se describirán brevemente estos componentes.

3.5.1. Interfaz Gráfica de Usuario

La interfaz gráfica del usuario de CPN Tools no sigue un patrón convencional sino por el contrario está basada en el uso de paletas de herramientas y menús de marcado. Un ejemplo de la interfaz de la herramienta se muestra en la Figura 3.11, donde se despliega el modelo de Bluetooth presentado anteriormente. La interfaz incluye dos partes: el *index* (*índice*) que es el área rectangular ubicada en la parte izquierda y el *workspace* (*espacio de trabajo*) que es la parte restante. El índice incluye: *tool box*, *help* (*ayuda*), *options* (*opciones*) y *overview of a net*

(*esquema de la red*). Estos componentes se describen brevemente a continuación, sin embargo, más detalles de los mismos y como se pueden usar se encuentra en [15].

Tool box

El *tool box* incluye una lista de todas las paletas (*palettes*) disponibles (ver Figura 3.11) y las cuales se enumeran a continuación:

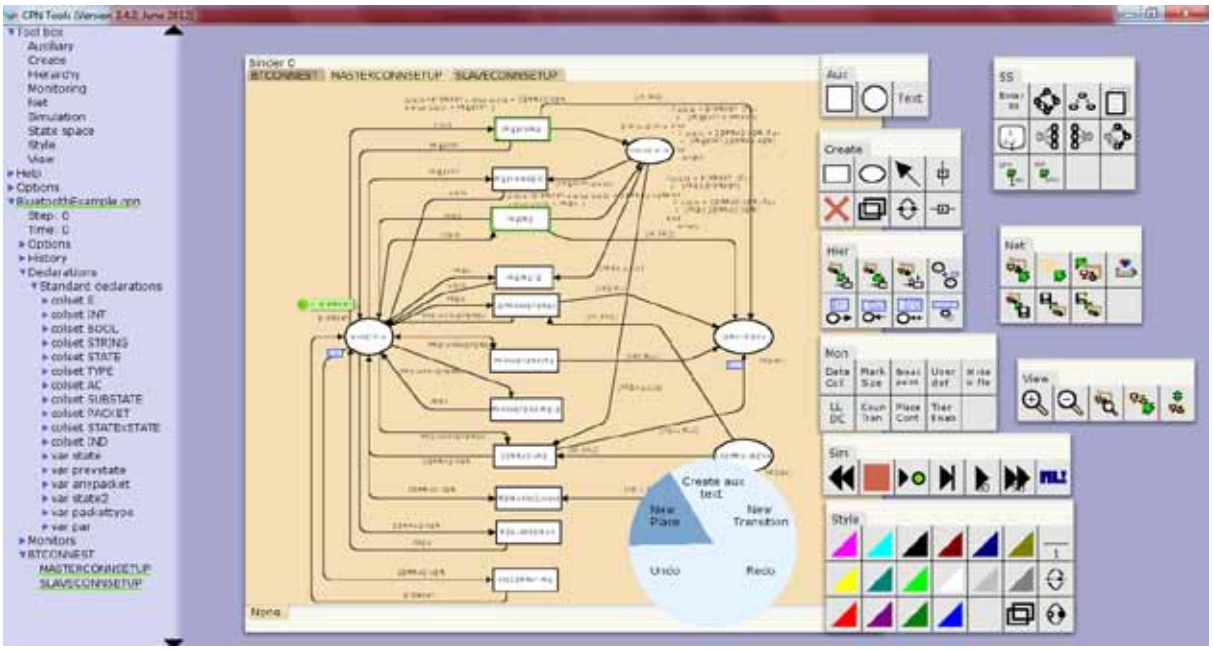


Figura 3.11: Pantalla de CPN Tools mostrando el modelo Bluetooth del ejemplo.

1) **Auxiliary tools (Aux):** son un conjunto de herramientas que permiten incrementar la legibilidad de la red pero no tienen ningún significado semántico.

2) **Create tools palette (Create):** permiten crear, editar y eliminar los componentes básicos de una CPN: arcos, plazas y transiciones.

3) **Hierarchy tools (Hier):** usada para manejar la estructuración de la red en niveles jerárquicos usando transiciones de sustitución o plazas fusionadas (ver Sección 3.3.8).

4) **Monitoring (Mon):** los monitores le permiten a un usuario indicar cuando y que datos deben ser recolectados durante una serie de pasos efectuados en una simulación automática. Esta paleta se usa para manejar los monitores.

5) **Net (Net):** usada para manejar las páginas (módulos) de una red y para cargar o salvar las CPNs.

6) **Simulation (Sim):** usada para efectuar simulaciones de la red.

7) **State Space (SS):** usada para la generación del OG y su correspondiente SCC y para crear lo reportes del OG. También permite realizar el cambio del simulador al OG.

8) **Style (Style):** usado para hacer el modelo más legible a través del cambio del grosor de las líneas y los colores. Estos cambios no tienen efecto en la semántica de la CPN.

9) **View (View):** usada para cambiar la vista de una página y sus elementos usando el *zoom* y el agrupamiento.

Help (Ayuda)

Esta sección incluye enlaces a diversas opciones de ayuda de la herramienta.

Options (Opciones)

El usuario puede cambiar diversas opciones de auto salvado de la red y si se deberían o no salvar los reportes de simulación y con qué tantos detalles.

Overview of a Net (Esquema de la Red)

Incluye detalles de la red abierta actualmente incluyendo: los pasos de la simulación ejecutados (*Step*), el tiempo del modelo en curso (*Time*), las opciones específicas de la red (*Options*), la lista de comandos ejecutados (*History*), la declaración de los conjunto de colores (*Declarations*) en lenguaje ML (ver Sección 3.3.2), los monitores definidos para la red (*Monitors*) y la lista de páginas (módulos) de la red.

Workspace

En el *Workspace* (espacio de trabajo) se pueden encontrar los siguientes elementos:

1) **Marking Menus:** son menús circulares cuyo contenido se despliega de acuerdo al contexto, es decir, de acuerdo al objeto sobre el cual este localizado el cursor. En la Figura 3.11 se observa un ejemplo del *Page Marking menu* desplegado cuando el cursor se encuentra en el módulo de MASTERCONNSETUP. Cabe aclarar que estos menús también se pueden desplegar cuando el cursor está en el área del *Index*. Por ejemplo, en la Figura 3.12 se muestra el *Net Marking menu* desplegado cuando el cursor está localizado sobre el nombre del modelo *BluetoothExample.cpn*. Otros *Marking menus* se describen en [15].

2) **Binders:** son ventanas rectangulares y se dividen en dos tipos: uno que contiene los elementos del modelo CPN (e.g declaraciones y módulos), tal como el *binder 0* en la Figura 3.11, y el otro tipo que contiene las herramientas que el usuario utiliza para manipular el modelo, tales como, los *Tools Palettes* explicados anteriormente. En la Figura 3.11 se muestran nueve (9) de estos *Tools Palettes*.

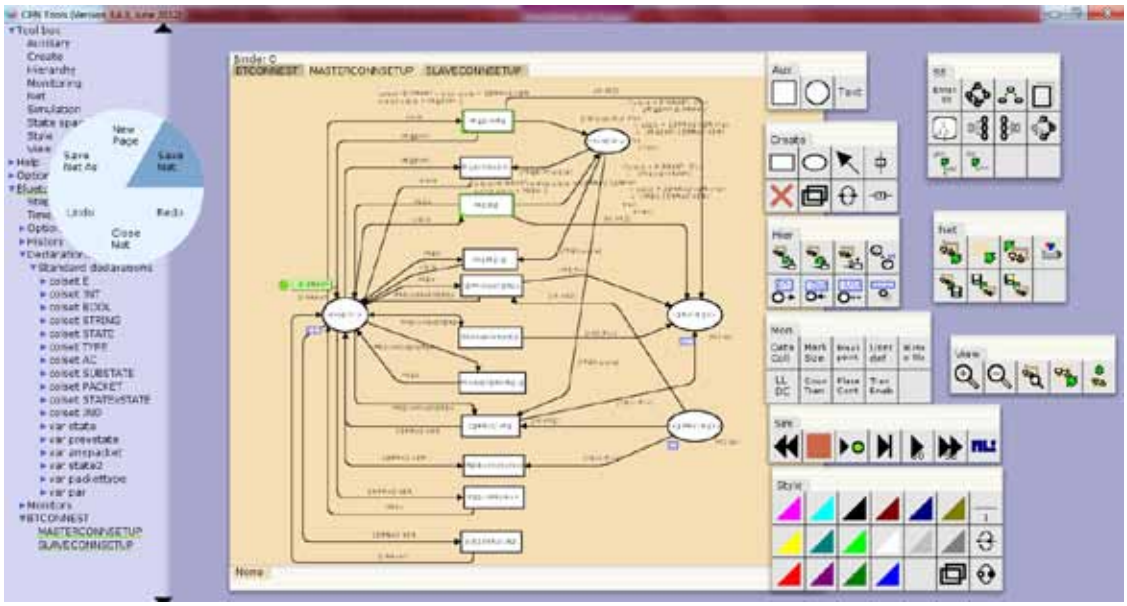


Figura 3.12: Despliegue del *Net Marking menu* en el ejemplo de modelo Bluetooth.

3.5.2. Simulador de CPN

CPN Tools proporciona *simulaciones interactivas* y *automáticas*, que tienen la finalidad de analizar y eliminar errores del modelo CPN. En el modo interactivo, el usuario puede elegir entre varios elementos de asociación, cambios de marcados, determinación de puntos de corte, entre otros. La simulación interactiva se utiliza para investigar el comportamiento de algunas partes del modelo. Los marcados que resultan de la simulación interactiva se exhiben en la pantalla según se muestra en la Figura 3.13. Contrariamente, la simulación automática tiene la finalidad de estudiar el comportamiento total del sistema. La simulación automática es más rápida que la simulación interactiva puesto que no implica la interacción humana y también no proporciona actualizaciones gráficas de los marcados. El usuario controla la simulación automática usando las opciones de parada, lo cual proporciona una cota al número de pasos de la simulación. Los resultados de la simulación automática pueden ser comprobados generando un informe de la simulación, que incluye todos los elementos de asociación que han ocurrido.

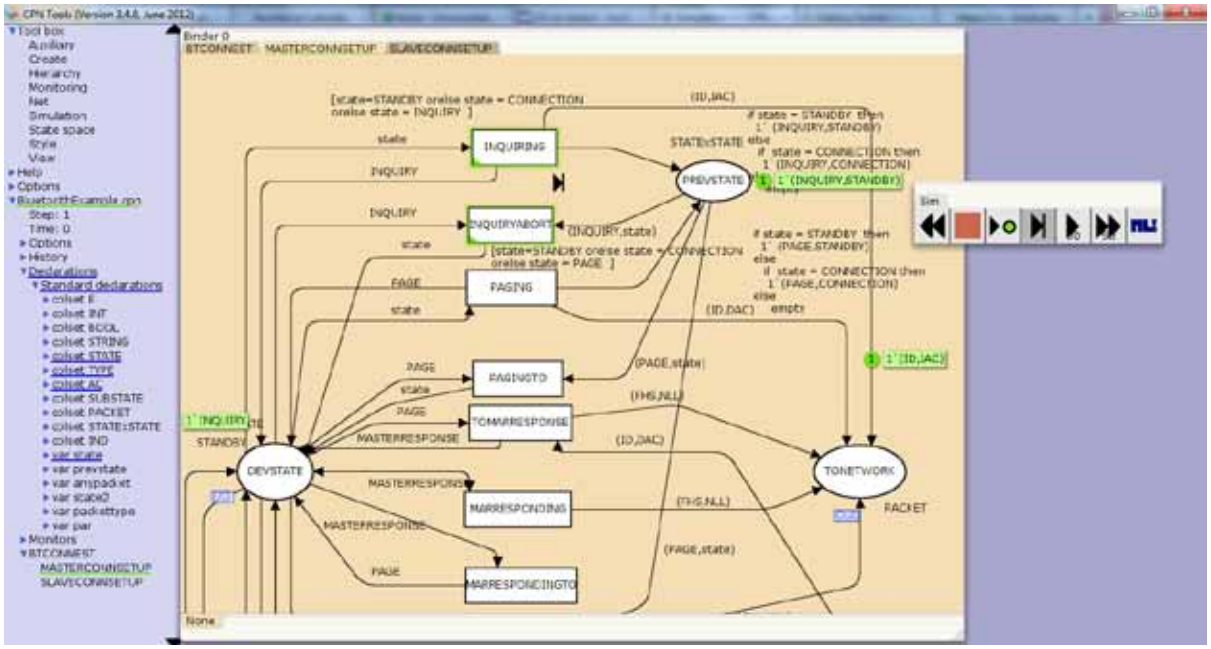


Figura 3.13: Ejemplo del simulación interactiva.

3.5.3. Grafo de Estado

La herramienta del grafo de estado genera el gráfico de ocurrencia (grafo OCC o OG) de un modelo CPN (ver Figura 3.14). El usuario puede controlar la generación automática del grafo de estado usando las opciones de parada y de ramificación. Las opciones de parada determinan las condiciones bajo las cuales la generación del grafo de estado no debe continuar. Esto se puede establecer colocando una cota en el número de nodos generados, arcos, y/o los segundos o usando una función de predicado que se evalúa después del cálculo de los sucesores. Si la misma evalúa a verdad la generación del OG se para.

Las opciones de ramificación permiten especificar bajo qué circunstancias no se generan todos los sucesores de un nodo. Esto puede conllevar a una generación parcial del OG. Tanto las opciones de parada como de ramificación se pueden cambiar desde la *Calculate State Space tool* [15] como se muestra en la Figura 3.14 o usando la funciones ML especificadas en [31][32].

La mayoría de las propiedades del comportamiento introducidas en la Sección 3.3.7 se pueden comprobar con el reporte del grafo de estado generado por CPN Tools usando la *State Space tool* (Figura 3.14). Este incluye la información estadística sobre el tamaño del OG y del SCC. También tiene información sobre las cotas enteras y de multi-conjunto superiores e inferiores de las plazas (propiedad de acotamiento). En adición, el reporte proporciona información sobre las propiedades locales y de vivacidad *(liveness)* tales como marcados locales y muertos, transiciones muertas y vivas.

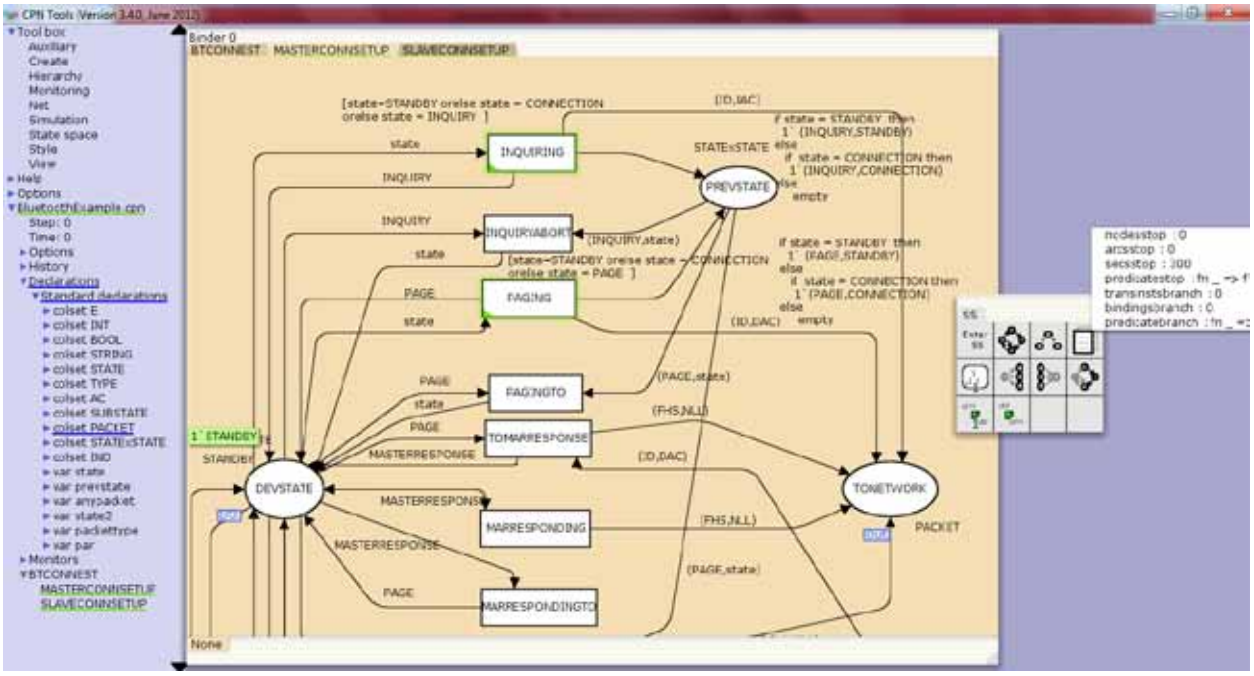


Figura 3.14: Herramienta de generación del OG (*state space*) de CPN Tools mostrando las opciones del *Calculate State Space tool*.

La herramienta del grafo de estado también proporciona una manera de hacer preguntas estándares y no estándares [31]. Las preguntas estándares se refieren a un conjunto predefinido de las funciones, que se pueden utilizar para examinar el grafo de estado. Las preguntas no estándares consisten en un conjunto de las primitivas implementadas sobre SML. Ambas propiedades son útiles para estudiar el comportamiento detallado del sistema modelado.

Finalmente, la herramienta del grafo de estado proporciona una facilidad para visualizar parte o todo el grafo de estado. El usuario puede dibujar el nodo del grafo de estado nodo por nodo o arco por arco o usar un número de funciones de dibujo integradas y que se pueden acceder desde la *State Space tool*. La herramienta de visualización es útil para encontrar errores en el sistema modelado. El OG mostrado en la Figura 3.10 fue dibujado usando esta facilidad.

3.5.4. Librería ASK-CTL

CPN Tools tiene una librería que implementa ASK-CTL (explicada en la Sección 3.4). La librería también contiene un *model checker*, el cual evalúa cada fórmula dada en el OG y retorna el resultado de la misma. Para verificar la fórmula eficientemente, el algoritmo toma en cuenta el SCC.

La sintaxis de ASK-CTL se da a continuación y está acorde a lo explicado en la Sección 3.4.

Fórmulas de Estado

A :: = TT| FF| NOT(A)| AND(A_1,A_2)| OR(A_1,A_2)| NF(<node expression>| EXIST_UNTIL(A_1,A_2)| FORALL_UNTIL(A_1,A_2)| POS(A)| INV(A)| EV(A)| ALONG(A)| MODAL(B) | EXIST_MODAL(A_1,B_2)| FORALL_MODAL(A_1,B_2)| EXIST_NEXT(A)| FORALL_NEXT(A)

Fórmulas de Transición

B :: = TT| FF| NOT(A)| AND(B_1,B_2)| OR(B_1,B_2)| NF(<node expression>| EXIST_UNTIL(B_1,B_2)| FORALL_UNTIL(B_1,B_2)| POS(B)| INV(B)| EV(B)| ALONG(B)| MODAL(A) | EXIST_MODAL(B_1,A_2)| FORALL_MODAL(B_1,A_2)| EXIST_NEXT(B)| FORALL_NEXT(B)

TT y FF son dos valores constantes de verdad y falso, respectivamente, mientras que los operadores NOT, AND, OR son equivalentes a las funciones booleanas ¬, ∧, ∨ respectivamente. Los operadores atómicos NF y AF implementan las fórmulas de transición α y β respectivamente.

Los operadores de cuantificación de camino EXIST_UNTIL(A_1,A_2), FORALL_UNTIL(A_1,A_2) EXIST_UNTIL(B_1,B_2) y FORALL_UNTIL(B_1,B_2) implementan las fórmulas ASK-CTL EU(A_1,A_2), AU(A_1,A_2), EU(B_1, B_2), AU(B_1, B_2), respectivamente.

Los operadores de cuantificación de camino derivados POS(A), INV(A), EV(A), ALONG(A), POS(B), INV(B), EV(B) y ALONG(B) implementan las fórmulas de estado y transición explicadas en la Sección 3.4.

Los operadores de cambio de dominio: MODAL(B) y MODAL(A) implementan las fórmulas <*B*> *y* <α>. Los operadores: EXIST_MODAL(A_1,B_2), FORALL_MODAL(A_1,B_2), EXIST_MODAL(B_1,A_2) y FORALL_MODAL(B_1,A_2) no están definidos en [12], pero si están incluidos en la librería ASK-CTL. EXIST_MODAL(A_1,B_2) (FORALL_MODAL(A_1,B_2)) significa que B_2 se cumple en algunos (en cada) estado de sucesor inmediato, M', y A_1 se cumple en la transición entre el estado en curso y M'. Las fórmulas de transición: EXIST_MODAL(B_1,A_2) y FORALL_MODAL(B_1,A_2) pueden ser definidas de forma análoga.

Los operadores de sucesor inmediato EXIST_NEXT(A), FORALL_NEXT(A), EXIST_NEXT(B) y FORALL_NEXT(B) implementan las fórmulas de estado y de transición EX(A), AX(A) EX(B) y AX(B), respectivamente.

4. Modelado y Análisis del Establecimiento de la Conexión Bandabase de Bluetooth Usando las Redes de Petri Coloreadas

4.1 Introducción

Bluetooth es un estándar para el soporte de la comunicación inalámbrica entre diversos tipos de dispositivos a distancias de aproximadamente 10 metros. Dada su corta cobertura, Bluetooth califica entre los estándares para las *Redes de Área Personal Inalámbricas (Wireless Personal Area Networks, WPAN)*, cuyo propósito es permitir la comunicación entre equipos en áreas de cobertura pequeñas que pueden abarcar, por ejemplo, un cuarto, una oficina o un automóvil. La especificación de Bluetooth, actualmente en la versión 4.0 [7], ha sido desarrollada por un grupo denominado Bluetooth *Special Interest Group (SIG)* que se forma a partir de fabricantes de dispositivos electrónicos de reconocida trayectoria como lo son Ericsson, IBM, Intel, Nokia y Toshiba [6]. Uno de los mayores inconvenientes de la especificación de Bluetooth es que hace poco uso de técnicas para la descripción de protocolos, tales como las tablas de estados, siendo la misma ampliamente narrativa. Como consecuencia, alguna de sus partes, tales como la descripción del establecimiento de una conexión Bandabase, son ambiguas y difíciles de entender. Este hecho puede claramente afectar las implementaciones de esta tecnología.

Por otra parte, ya se ha planteado el problema de la lentitud del establecimiento de una conexión Bluetooth y algunos mecanismos para aliviar este problema [53]. También se ha hablado de la incompatibilidad de ciertos equipos Bluetooth que fallan al momento del establecimiento de la conexión [16]. Dada la complejidad de la tecnología y del poco uso que se hace en la especificación de Bluetooth de herramientas formales para la descripción de protocolos, es posible que parte de los problemas antes mencionados sean debidos a la falta de una clara especificación de los procedimientos.

En este libro se usan las CPNs para modelar el establecimiento de una conexión a nivel del protocolo Bandabase de Bluetooth de forma funcional, para lograr una especificación clara de este procedimiento. El modelo es entonces validado usando la técnica de análisis del grafo de estado (ver Sección 3.3.9) en función de las propiedades de comportamiento de las CPNs. Adicionalmente se definen algunas propiedades de comportamiento del establecimiento de la

conexión Bandabase, las cuales son especificadas usando ambos ASK-CTL y la notación estándar de CPN. Dichas propiedades son verificadas usando las facilidades de *model checking* de CPN Tools y realizando ciertas consultas al OG utilizando funciones estándares provistas por la herramienta.

A parte de los trabajos publicados por el autor [63][64], se han encontrado pocos trabajos que incluyan la aplicación de técnicas formales a las actividades de ingeniería de protocolos asociadas a la tecnología Bluetooth. Uno de estos trabajos es el de Feldmann et al. [19], quienes modelan una *scatternet* de Bluetooth usando las CPNs, para estudiar el rendimiento de la red. En otro trabajo relacionado, Duflot M. et al. [6] presentan algunos resultados del rendimiento del proceso de indagación (*inquiry*) de Bluetooth usando la técnica de chequeo de modelos (*model checking*) probabilística con la ayuda de la herramienta PRISM. Por otra parte, en el presente libro se estudian la propiedades funcionales del protocolo Bandabase incluyendo cómo una *piconet* (ver Sección 4.2) es establecida y englobando tanto el proceso de *indagación* (*inquiry*) como el de *page*.

4.2 Descripción General de Bluetooth

Bluetooth [7][39] es una tecnología de radio frecuencia (*Radio Fraquency, RF*) que ofrece conectividad a corta distancia para equipos personales, portables, PDAs, entre otros. Bluetooth está orientado al reemplazo de interfaces tradicionales, tales como RS-232 y conectores propietarios, a proporcionar una interfaz uniforme para acceder servicios de voz y datos, a proporcionar acceso a una red de área amplia usando un *gateway* personal, tal como un teléfono celular, y a proporcionar una comunicación sin infraestructura, que se puede usar para el soporte a grupos colaborativos (reuniones, conferencias).

Los dispositivos Bluetooth trabajan en la frecuencia de 2.4 GHz (más específicamente la banda de frecuencia en la mayoría de países es de 2.4 – 2.4835 GHz) también conocida como la banda para uso *Industrial, Científico y Médico* (o banda *Industrial, Scientific and Medical*, *ISM*). La transmisión de la señal ocurre usando una técnica de saltos de frecuencia elegidos de forma aleatoria [7] entre 79 canales físicos de 1 MHz, en que se divide el ancho de banda usado por esta tecnología, siendo la tasa de transmisión de 1 Mbps.

4.2.1 Pila de Protocolos

La especificación de Bluetooth [7] incluye una descripción del núcleo que indica los detalles de los diversos protocolos que conforman la pila de protocolos; y una especificación de los perfiles que incluye los detalles del uso de la tecnología para soportar varias aplicaciones e indica cuales de los aspectos de la especificación del núcleo son obligatorios, opcionales y no aplicables. La Figura 4.1 muestra la pila de protocolos que conforman el estándar. La misma divide los protocolos en los siguientes niveles:

a) **Protocolos fundamentales de Bluetooth (protocolos del núcleo):** son específicos de Bluetooth y han sido desarrollados por el SIG de Bluetooth.

b) **Protocolos de sustitución de cable:** suministran señalización de control que emula el tipo de señalización que se asocia usualmente con los enlaces de cable.

c) **Protocolos de control de telefonía:** definen la señalización de control para establecer llamadas de voz y datos con dispositivos Bluetooth. También define un protocolo (Comandos AT) que especifica cómo puede controlarse un MODEM y un teléfono móvil.

d) **Protocolos adoptados:** son protocolos existentes que se utilizan para diversos fines en las capas superiores.

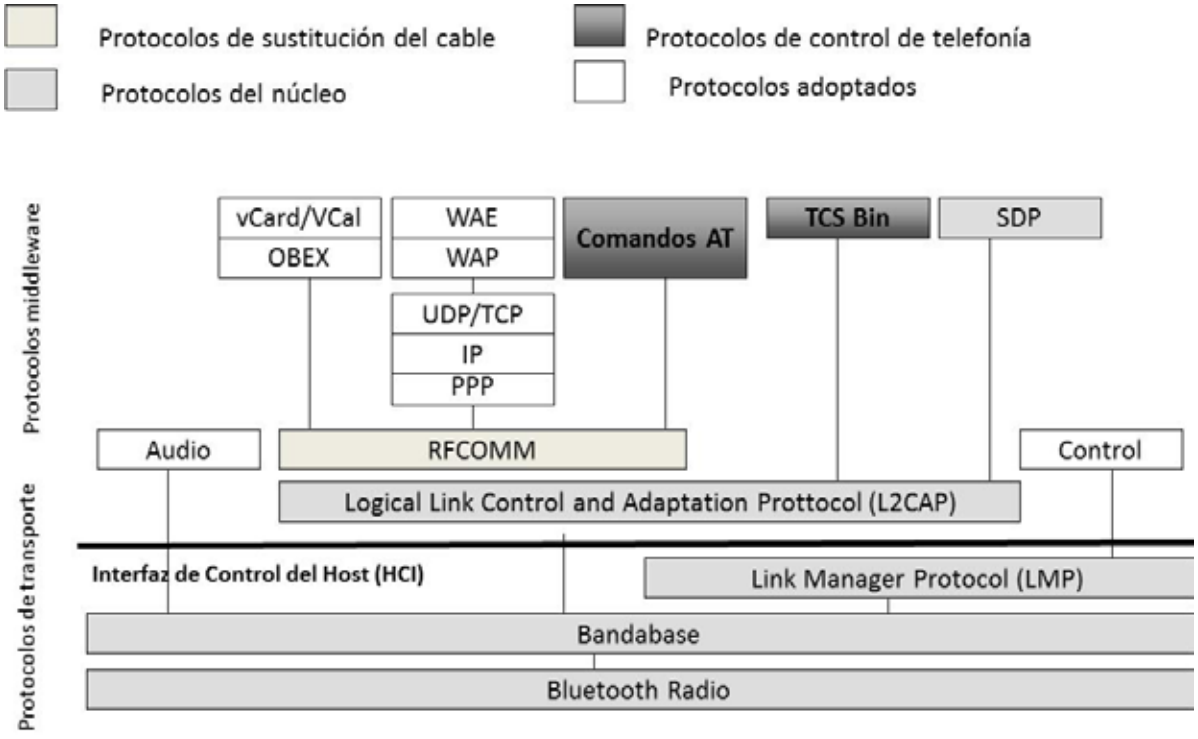


Figura 4.1: Pila de Protocolos de Bluetooth.

4.2.2 Núcleo de Bluetooth

El núcleo de Bluetooth incluye los protocolos asociados con las cuatro capas más bajas de la arquitectura y algunos protocolos que especifican ciertos servicios comunes a todas las aplicaciones soportadas por la tecnología. Los protocolos del sistema del núcleo son: el Protocolo de Radio (*Radio Protocol, RF*), el *Protocolo de Control de Enlace* (*Link Control Protocol, LCP*), *Link Manager Protocol* (*LMP*) y el *Protocolo de Adaptación y Control de Enlace Lógico* (*Logical Link Control and Adaptation Protocol, L2CAP*). Adicionalmente, el núcleo engloba un protocolo de la capa de servicio que es común para todos los demás servicios llamado el

Protocolo de Descubrimiento de Servicio (*Service Discovery Protocol, SDP*) y un conjunto de requerimientos de perfiles descritos en el *Perfil de Acceso Genérico* (*Generic Access Profile, GAP*). Un *perfil* define un conjunto de posibles casos de uso de un dispositivo e incluye algunas partes específicas de la pila de Bluetooth. Un perfil puede incluir algunos protocolos definidos en la especificación del núcleo y otros protocolos de las capa superiores, tal como el protocolo de *Intercambio de Objetos* (*Object Exchange, OBEX*) [9].

En la Figura 4.2 se muestra como las capas de la arquitectura de Bluetooth y los correspondientes protocolos pertenecientes al núcleo interactúan en un sistema. Las tres capas inferiores de Bluetooth se agrupan en lo que se conoce como un *controlador*, mientras que el L2CAP, las capas de servicio y los protocolos de las capas superiores se encuentran en el *host* Bluetooth. También se observa como las señales de control y de datos fluyen para soportar la funcionalidad de la tecnología. A continuación se explican brevemente los elementos de comunicación mostrados en la Figura 4.2, una explicación más detallada se encuentra en [8].

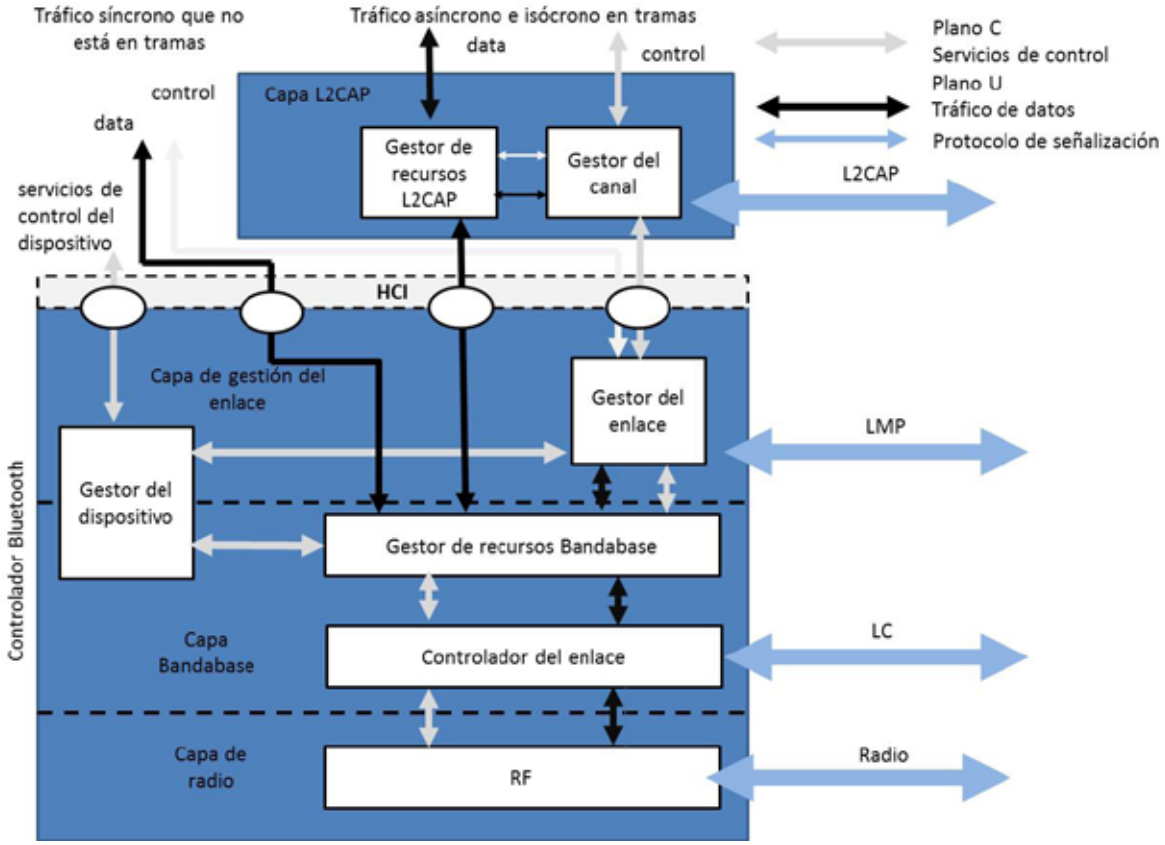


Figura 4.2: Arquitectura del núcleo de Bluetooth.

Radio (RF)

La especificación de Bluetooth establece el uso de la frecuencia ISM de 2.4 GHz, la cual en varios países corresponde al siguiente rango: 2.4 – 2.4835 GHz, definiéndose 79 canales

físicos de 1 MHz sobre esta banda. Adicionalmente, los radios Bluetooth vienen en tres (3) clases dependiendo de su potencia de transmisión **(**ver Tabla 4.1**),** siendo la técnica de modulación usada la *Gaussian Frecuency-Shift-Keying (GPSK)* [8] con una tasa de baudios de 1 Msps. Pero ya que el tiempo de bit es 1 ms, la tasa de transmisión es 1 Mbps.

Tabla 4.1: Potencias de transmisión de los equipos Bluetooth [7]

Clase	Máxima Potencia de Salida (Pmax)	Potencia de salida Nominal	Potencia de Salida Mínima (Pmin)	Control de Potencia
1	1000 mW (20 dBm)	N/A	1 mW (0 dBm)	Pmin <+4 dBm a Pmax Opcional: Pmin a Pmax
2	2,5 mW (4 dBm)	1 mW (0 dBm)	0,25 mW (-6 dBm)	Opcional: Pmin a Pmax
3	1 mW (0 dBm)	N/A	N/A	Opcional: Pamin a Pmax

Gestor del Enlace

El Gestor del Enlace es el encargado de gestionar diversos aspectos de los enlaces lógicos, tales como la creación, modificación y terminación. También se encarga de gestionar diversos aspectos de los enlaces físicos establecidos entre dispositivos. El Gestor de Enlace ejecuta sus funciones a través del LMP.

En Bluetooth se define una jerarquía para el transporte de los datos. En el nivel más inferior se encuentran los *canales físicos* caracterizados por una frecuencia RF junto con ciertos parámetros temporales. Estos canales están restringidos por consideraciones de distribución de las frecuencias. Sobre estos canales se definen los *enlaces físicos*, donde cada enlace se define como una conexión Bandabase (explicada más adelante en este capítulo) entre dos dispositivos activos y siempre está asociado con un canal físico. Sin embargo, un canal físico puede soportar más de un enlace físico. El enlace físico es un concepto virtual que no tiene una representación en la estructura de un paquete, sino por el contrario se identifica usando el código de acceso del paquete, información del reloj y la dirección del dispositivo maestro (explicado más adelante en este capítulo). El LM puede cambiar ciertas propiedades del enlace físico como la potencia de transmisión.

Por otro parte se tienen los *enlaces lógicos,* los cuales están ubicados por encima de los enlaces físicos y están disponibles para soportar el transporte de diversos tipos de tráfico acorde a los requerimientos de las aplicaciones. Cada enlace lógico es asociado con un *transporte lógico*, sin embargo, un transporte lógico puede transportar diferentes tipos de enlaces lógicos. Cada

transporte lógico incluye características como control de flujo, mecanismos de retransmisiones, secuenciamiento y mecanismos de planificación de paquetes. Los transportes lógicos son trasladados por enlaces físicos activos. Un tipo de transporte lógico es el asíncrono orientado a conexión usado para transportar información que requiere cierta confiabilidad y tiene algunas restricciones de tiempo, otros ejemplos de estos transportes se encuentran en [8].

Gestor del Dispositivo

El Gestor de Dispositivo soporta las funciones que no están directamente relacionadas con el transporte de los datos. Entre estas funciones se encuentra el buscar dispositivos cercanos y conectarse a otros dispositivos Bluetooth.

Gestor de Canales

El Gestor de Canales se encarga de la creación, modificación y terminación de canales L2CAP. Los canales L2CAP son establecidos entre dos entidades L2CAP para permitir que el tráfico proveniente de las capas superiores fluya y son multiplexados en enlaces lógicos. El Gestor de Canales debe interactuar con el Gestor del Enlace para solicitar los servicios de creación de enlaces lógicos, que soporten los canales L2CAP, cuando sea necesario y configurar estos enlaces para proporcionar la calidad de servicio acorde al tipo de aplicación cuyo tráfico es transportado. Las funcionalidades de este gestor son soportadas por el L2CAP.

Gestor de Recursos L2CAP

El Gestor de Recursos L2CAP se encarga de que los segmentos asociados a un PDU sean enviados en orden a la capa Bandabase. También es responsable de manejar la planificación entre canales L2CAP, de forma tal que el tráfico para el cual se tienen ciertos compromisos de QoS no se le impida su acceso a los canales físicos debido a que estos recursos, a nivel del controlador Bluetooth, se hayan acabado.

Interfaz entre el Controlador y el Host (HCI)

La Interfaz entre el Controlador y el Host (*Host Controller Interface, HCI*) permite una forma estandarizada de comunicación entre el *host* y las capas más bajas de la arquitectura de Bluetooth conocida como controlador. La misma permite que estos componentes funcionen aun no estando integrados en un solo dispositivo, tal como, cuando se tiene el controlador implementado en un *dongle* Bluetooth y las funcionalidades del *host* en el sistema operativo corriendo en el computador donde se instaló el *dongle*.

Protocolo de Descubrimiento de Servicio

El Protocolo de Descubrimiento de Servicio o SDP proporciona los mecanismos para descubrir los servicios y sus atributos (por ejemplo, clase del servicio y protocolos para usar el servicio). El SDP es necesario dado que el conjunto de servicios disponibles para una aplicación

cambian de acuerdo a la proximidad de los otros dispositivos en movimiento. El mismo trabaja usando una aproximación cliente-servidor, donde el cliente envía un requerimiento al servidor para obtener información de un servicio.

Perfil de Acceso Genérico

El Perfil de Acceso Genérico o GAP es un perfil base para otros perfiles y establece las operaciones necesarias para que un dispositivo establezca comunicación con otro. Estas operaciones incluyen el descubrimiento de dispositivos, el establecimiento y configuración del enlace y establecimiento de diferentes niveles de seguridad.

Capa Bandabase

La capa Bandabase es parte del núcleo de Bluetooth y sus funciones incluyen: sincronismo, establecimiento de la conexión, selección de la frecuencia de salto, procesamiento de los paquetes, control de potencia y manejo de seguridad a nivel Bandabase. Existen dos entidades básicas encargadas de soportar estas funcionalidades: el Gestor de Recursos Bandabase y el Controlador de Recursos (ver Figura 4.2). El Gestor de Recursos Bandabase es el encargado de manejar el acceso al medio tomando en cuenta los compromisos de QoS adquiridos para permitir que una aplicación tenga un rendimiento de a acuerdo a lo esperado. Una de sus funciones es planificar los tiempos de los canales físicos para todas las entidades acorde a su contrato de acceso. La otra función es negociar los contratos de acceso con estas entidades. Un *contrato de acceso* es un compromiso de proporcionar un nivel de QoS.

Las demás funciones de la capa Bandabase son ejecutadas por el Controlador de Enlace. Aquellas funciones que son relevantes para este libro se describen a continuación, las demás se describen en detalle en [8].

Piconets y Scatternets

Los dispositivos Bluetooth se conectan siguiendo una topología particular. Así, una *piconet* es una colección de dispositivos que pueden comunicarse, que se forma de una forma ad hoc y que contiene un dispositivo *maestro* y a lo sumo siete (7) dispositivos *esclavos* (ver Figura 4.3). Un dispositivo en una *piconet* puede ser parte de otra *piconet* (como maestro o esclavo), conociéndose esta especie de solapamiento como *scatternet* como se muestra en la Figura 4.3**.**

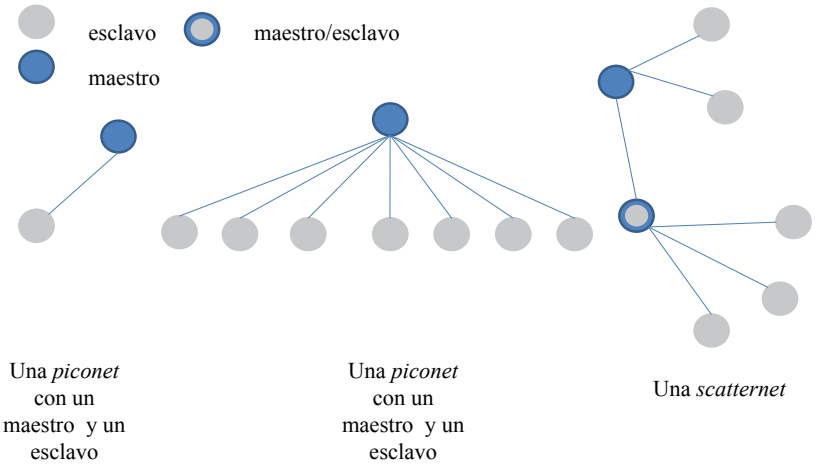


Figura 4.3: Ejemplo de *piconet* y *scatternet*.

Salto de Frecuencia

La forma en que se comparte el medio en Bluetooth es a través del uso de diversos mecanismos. Primero, el ancho de banda disponible es dividido en 79 canales físicos de 1 MHz (es decir, Acceso al Medio por División de Frecuencia o *Frequency Division Medium Access*, *FDMA*). Segundo, los dispositivos (radios) transmiten en una dirección a la vez y la transmisión se alterna entre las dos direcciones (es decir, Multiplicación por División de Tiempo o *Time Division Duplex, TDD)*. Los datos son enviados en distintas frecuencias que van cambiando de forma pseudo aleatoria a una tasa de tasa de salto de 1600 saltos/seg siendo cada canal ocupado por 0,625 ms. Este período es llamado *slot* y los *slots* se encuentran numerados secuencialmente. La frecuencia de salto es compartida entre todos los dispositivos de una *piconet* y es determinada por el maestro. Este último mecanismo es conocido como Espectro Disperso por Salto de Frecuencias (*Frequency Hop Spread Spectrum, FHSS*).

La transmisión de un paquete se inicia al comienzo de un *slot* y si un paquete requiere más de un *slot*, el radio permanece en la misma frecuencia, retornando después a la frecuencia requerida por la secuencia de salto. El maestro de una *piconet* transmite en *slots* de tiempos pares mientras que los esclavos transmiten en *slots* impares como se muestra en la Figura 4.4. La frecuencia de salto es determinada por el maestro y diferentes *piconets* en la misma área tendrán diferentes maestros. Así, la mayoría de la veces las transmisiones en diferentes *piconets* ocurrirá en diferentes frecuencias, sin embargo, ocasionalmente, pueden haber colisiones debido a dos o más transmisiones de dos o más *piconets* distintas (en la misma área) usando la misma frecuencia. Los errores en los paquetes, producidos como consecuencia de dichas colisiones, son tratados usando los mecanismos de control de errores provistos por la tecnología [8].

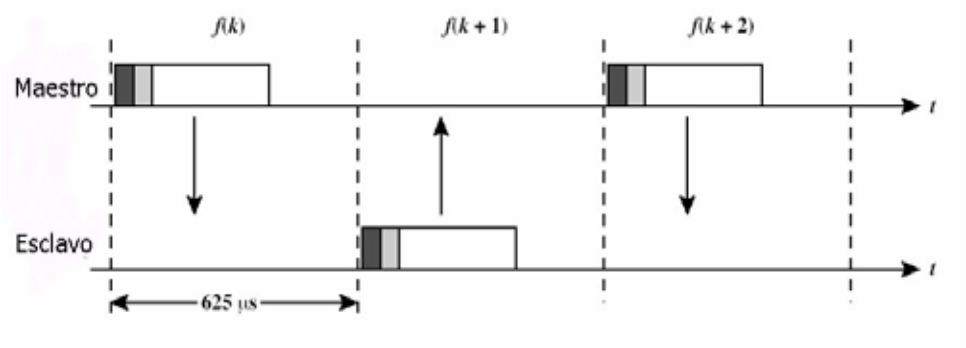


Figura 4.4: Salto de frecuencia en Bluetooth.

Paquetes

La Figura 4.5 muestra la estructura de un paquete Bandabase y a continuación se describe el mismo:

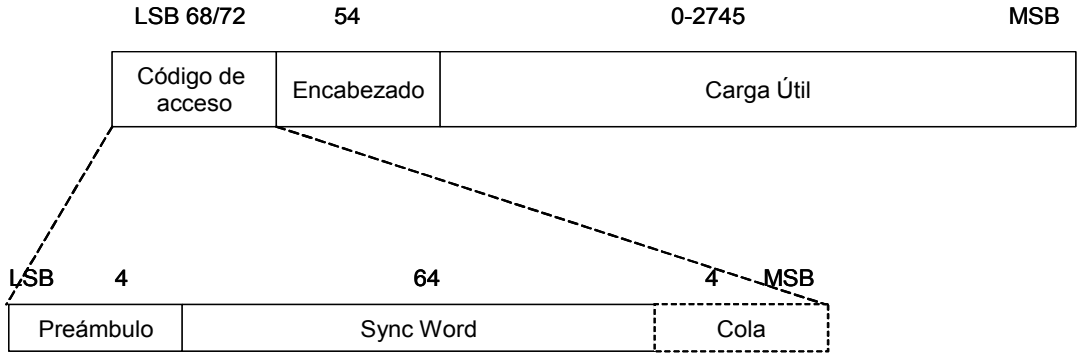


Figura 4.5: Formato del paquete Bandabase.

1) **Código de Acceso (*Access Code, AC*):** existe un código corto (68 bits) y otro largo (72 bits). El AC corto es usado para sincronismo, compensación del desplazamiento DC e identificación y distingue todos los paquetes en un canal físico. Los paquetes enviados por un mismo canal físico deben tener el mismo AC. Hay tres tipos de Código de Acceso (AC):

   - Código de Acceso del Canal (*Channel Access Code, CAC*): identifica una *piconet*. Todos los paquetes enviados en la misma *piconet* tienen el mismo CAC.
   - Código de Acceso del Dispositivo (*Device Access Code, DAC*): usado para búsquedas (*paging*) y subsecuentes respuestas.
   - Código de Acceso de Indagación (*Inquiry Access Code, IAC*): puede ser general (GIAC) o dedicado (DIAC). El primero se usa para descubrir dispositivos que estén en la proximidad, mientras que el segundo es usado para un grupo dedicado de dispositivos.

Adicionalmente, el AC incluye los siguientes campos:

- Preámbulo: tiene un patrón fijo de cuatro (4) símbolos y es usado para compensar el DC. Su valor depende del bit menos significativo (*least significant bit, LSB)* de la palabra de sincronización (es decir, *syncword*) ubicada a continuación.
- Cola: tiene un patrón fijo de cuatro (4) símbolos y es usado para extender compensación del DC junto con los tres bit más significativos (*most significant bit, MSB)* de la *syncword*.
- *Syncword*: cada dispositivo Bluetooth tiene una dirección de 48 bits. Los 24 bits menos significativos son denominados parte de la dirección menos significativa (*lower address part, LAP*). La LAP es usada para formar la *syncword*. Para el CAC, la dirección LAP del maestro es usada. Para el GIAC o el DIAC, las LAPs reservadas y dedicadas son usadas. Para el DAC, la dirección LAP del esclavo es usada.

La palabra *syncword* se forma de la siguiente forma:

1. Sumar bits a la LAP.
2. Generar la secuencia PN.
3. Hacer el XOR del resultado del paso 1 y parte de la secuencia PN.
4. Generar un código de error de 34 bits para el bloque de información anterior y colocarlo al comienzo.
5. Hacer XOR de la PN y la secuencia de 64 bits producida en el paso 4.

2) **Encabezado del paquete:** está formado por los siguientes campos (ver Figura 4.6):

- LT-ADDR: dirección de transporte lógico para el paquete. Es la dirección temporal asignada al esclavo en dicha *piconet*.
- Tipo: identifica el tipo de paquete.
- Flujo: proporciona un mecanismo para el control de flujo de un solo bit para el tráfico ACL.
- ARQN: proporciona un mecanismo de reconocimiento de un (1) bit para el tráfico ACL protegido por un CRC.

- SEQN: proporciona un esquema de numeración secuencial de un (1) bit. Los paquetes transmitidos son etiquetados con 1 o 0.
- HEC: es un código de detección de errores para proteger el encabezado.

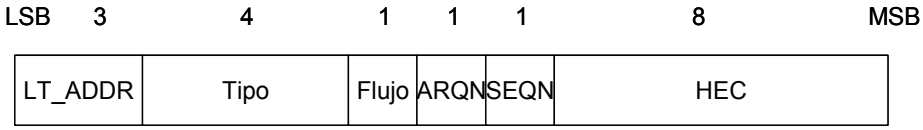


Figura 4.6: Formato del campo de encabezado del paquete.

3) **Carga Útil del Paquete:** transporta información proveniente de las capas superiores.

Establecimiento de una Conexión Bandabase

En esta sección se describe el procedimiento del establecimiento de una conexión Bandabase, cuyo modelo es presentado en detalle en este libro. El protocolo Bandabase de Bluetooth establece como se forma una *piconet* y como los dispositivos son incorporados a la red y removidos de la misma. La Figura 4.7 muestra un diagrama ilustrando los diferentes estados usados en la gestión de una *piconet*; el diagrama ha sido tomado de la especificación de Bluetooth [8]. Hay tres estados principales STANDBY, CONNECTION, y PARK y siete sub estados *page*, *page scan*, *inquiry*, *inquiry scan*, *master response*, *slave response* e *inquiry response*. En el estado inicial de STANDBY, un dispositivo no está conectado a ningún dispositivo y no es parte de ninguna *piconet*. Durante la operación de indagación (*inquiry*), un dispositivo colecta información, tal como la dirección del dispositivo y valores del reloj de los dispositivos en su proximidad. Hay tres sub estados de indagación, los cuales son descritos en el siguiente párrafo.

En el sub estado de *inquiry*, un dispositivo (potencial maestro[2]) transmite paquetes de *inquiry*, los cuales son recibidos por los dispositivos (potenciales esclavos[3]) en el sub estado *inquiry scan*. Después de recibir un paquete de *inquiry*, el esclavo entra en el sub estado de *inquiry response* y envía un mensaje de *inquiry response*. Un paquete de *inquiry* es denominado

[2] En el estado de *inquiry* los roles de maestro y esclavo no están definidos. Un dispositivo que inicia un procedimiento de indagación (*inquiry*) se dice que es un potencial maestro. De aquí en adelante para nombrar a este dispositivo solo se usará el término maestro.

[3] En el estado de *inquiry* los roles de maestro y esclavo no están definidos. Un dispositivo que está en el sub estado de *inquiry scan* se dice que es un potencial esclavo. De aquí en adelante para nombrar a este dispositivo solo se usará el término esclavo.

paquete ID y notifica al esclavo que un maestro está buscando a otro dispositivo Bluetooth en la cercanía. Un paquete de respuesta del *inquiry* es un paquete FHS. Tanto el paquete ID como el FHS contienen códigos de acceso (ACs) apropiados, los cuales son usados para diferentes propósitos incluyendo la identificación de las transmisiones en diferentes *piconets* como se explicó anteriormente.

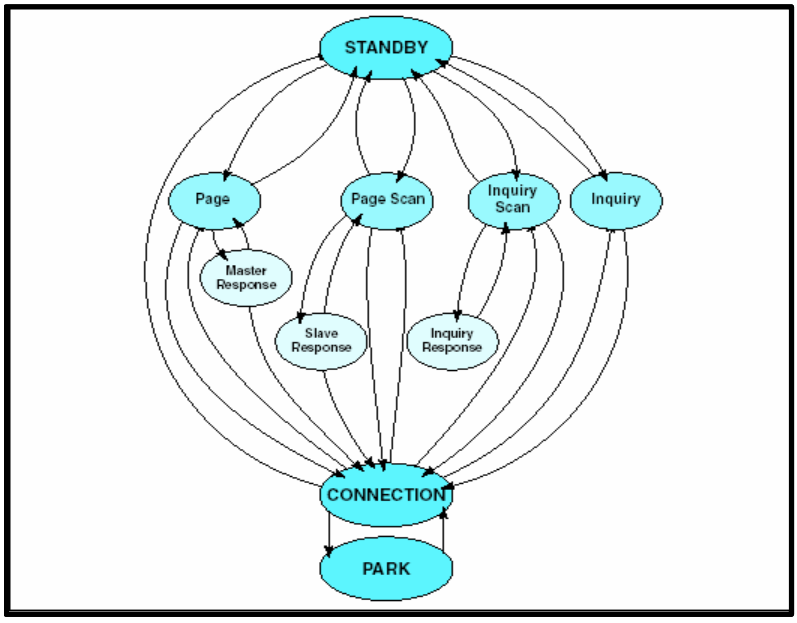


Figura 4.7: Diagrama de transición de estados del establecimiento de una conexión Bandabase.

Durante la operación de *page*, un dispositivo invita a otro a juntarse a una *piconet*. Así, en el sub estado de *page*, un dispositivo (potencial maestro) puede conectarse a cualquier dispositivo (potencial esclavo), el cual este en el sub estado *page scan*. En el sub estado de *page scan* un esclavo escucha por paquetes de *page* (paquetes IDs) provenientes de dispositivo realizando el *page*. Cuando el esclavo recibe un paquete de *page*, entra en el sub estado de *slave response*. En este sub estado, el esclavo espera recibir un paquete de response, es decir un paquete FHS, del dispositivo haciendo el *page*. Después de recibir el paquete FHS, el esclavo responde con otro paquete (paquete ID) y entra en el estado de CONNECTION (es decir, el esclavo está conectado al maestro). Por otra parte, el dispositivo haciendo el *page* entra en el sub estado de *master response* después de recibir un paquete de respuesta del *page* (el paquete ID) enviado por el esclavo. El maestro transmite un paquete FHS conteniendo información necesaria para que el dispositivo remoto entre en el estado de CONNECTION. Una vez que el maestro recibe un respuesta del esclavo (un paquete ID), entra en el estado de CONNECTION. El maestro envía su primer paquete de tráfico (un paquete de POLL) con sus parámetros de conexión. El esclavo, que ya se encuentra en el estado de CONNECTION, podrá responder con cualquier tipo de paquetes tal como un POLL. Los paquetes de ID y FHS del esclavo contiene códigos de acceso DAC del esclavo, mientras que el paquete de POLL enviado por el maestro contiene el CAC del maestro.

Una vez conectado, un dispositivo puede encontrarse en alguno de los siguientes estados: un estado *activo* donde participa en una *piconet*. En este estado escucha, transmite y recibe paquetes; un estado de *husmeo* (s*niff*) donde escucha en *slots* específicos; un estado de *sostenido* (*hold*) que es un estado de potencia reducida, donde aún puede participar en el intercambio de paquetes síncronos; y un estado de *estacionado* (PARK) donde no participa en la *piconet*, pero es retenido como parte de ella. Finalmente, el dispositivo puede desconectarse en cualquier momento retornando al estado por defecto de STANDBY. Estos estados están descritos en detalle en [8].

4.2.3 Protocolos de las Capas Superiores

La especificación de Bluetooth define aplicaciones de la tecnología a través de los perfiles. La descripción de un perfil debe incluir su relación con otros perfiles, los formatos de las interfaces con los usuarios y las partes específicas de la pila de protocolo. Se han definido una serie de perfiles, tales como, el Perfil del Puerto Serial que establece como deben configurarse los dispositivos Bluetooth para emular una conexión serial usando RFCOMM, y el Perfil de Transferencia de Archivos que ofrece la capacidad de transferir objetos de datos (por ejemplo, hoja de cálculo, presentaciones e imágenes) de un dispositivo. Una descripción más detallada de los perfiles de Bluetooth se puede encontrar en [9].

Por otra parte existen diversos protocolos propios de la tecnología y otros adoptados de otras tecnologías. Un protocolo propio de la tecnología es el *Bluetooth Network Encapsulation Protocol* (*BNEP*) usado para el transporte de protocolos de red comunes, tales como IPv4 e IPv6, sobre Bluetooth [9]. Mientras otros protocolos son adoptados de otra tecnología como es el caso del protocolo OBEX desarrollado por la *Asociación de Data Infrarroja* (*InfraRed Data Association, IrDA*) y que define objetos de datos y como los mismos pueden intercambiarse entre dos dispositivos. La descripción de otros protocolos pueden encontrase en [9].

4.3 Modelo CPN del Establecimiento de una Conexión Bandabase

En la especificación de Bluetooth no se describen claramente las transiciones entre los estados mostradas en la Figura 4.7. A continuación se presenta un modelo del establecimiento de una conexión Bandabase basado en dicha especificación [8] y en la descripción dada en [39]. El modelo es creado usando las CPNs con la ayuda de CPN Tools versión 2.2.0 (ver Sección 3.5).

4.3.1 Alcance

La especificación del protocolo Bandabase de Bluetooth es compleja [8], por lo cual se ha tomado una aproximación incremental para la realización del modelo del establecimiento de una conexión Bandabase. Así, en este libro se presenta una versión mejorada del modelo mostrado en [63] e incluye el procedimiento realizado para el establecimiento de una conexión entre dos dispositivos (ver Figura 3.3). En este modelo, solo el establecimiento de una conexión Bluetooth

entre un maestro y un esclavo es considerada. Vale destacar que esta es una limitación que se encuentra en varios de los dispositivos móviles con capacidades restringidas, tales como los teléfonos celulares. Adicionalmente, aunque las CPNs soportan el modelado de las restricciones temporales, en esta versión del modelo ellas han sido omitidas. Esto es porque inicialmente solo se está interesado en la especificación funcional de todas las transiciones mostradas en la Figura 4.7.

4.3.2 Modelo Jerárquico

Similarmente a otros modelos de protocolos de comunicación complejos (tales como el presentado en [62]), en este libro se utilizan los constructores jerárquicos de las CPNs (ver Sección 3.3.8), en particular, se usan las *transiciones de sustitución*. El modelo se inicia con un diagrama CPN en el nivel superior, el cual proporciona una visión general del sistema que está siendo modelado y su ambiente. El modelo del establecimiento de una conexión Bluetooth a un alto nivel de abstracción se muestra en la Figura 4.8. Este incluye una transición de sustitución para el procedimiento de establecimiento de la conexión que se realiza en el maestro (MASTER) y otra para el que se realiza en el esclavo (SLAVE), las cuales son definidas en sus propios módulos. El medio de comunicación, que en el caso de Bluetooth es compartido, es representado por la transición COMMUNICATION_CHANNEL.

En la Figura 4.8, las plazas MASTERTOSLAVE y SLAVETOMASTER son del tipo PACKET (el cual es descrito en la Sección 4.3.3) y representan los paquetes Bandabase que viajan hacia el esclavo o hacia el maestro, respectivamente. Estas plazas son zócalos y están conectadas con plazas que tienen el mismo nombre en los sub módulos MASTER, SLAVE Y COMMUNICATION_CHANNEL descritos posteriormente.

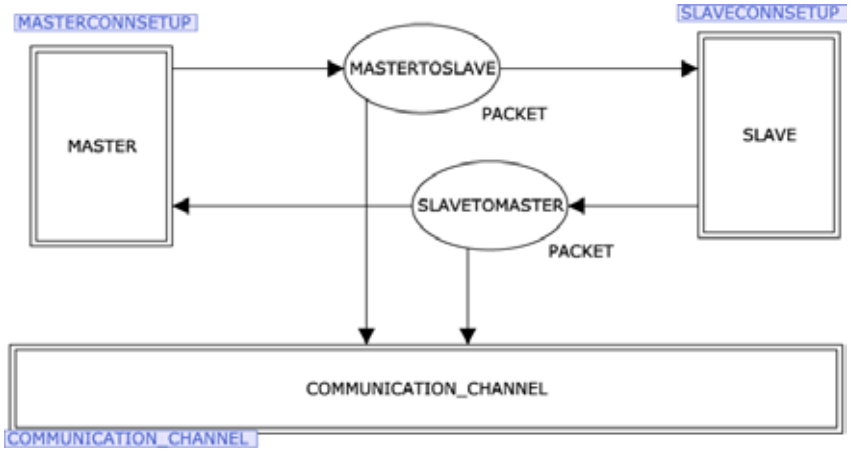


Figura 4.8: Vista jerárquica del establecimiento de una conexión Bandabase.

4.3.3 Declaración Global

La Figura 4.9 muestra los conjuntos de colores (tipos) y variables de la declaración global del modelo. Los colores BOOL, STRING e INT representan los tipos *boolean*, *string* y entero, respectivamente, presentes en otros lenguajes de programación. El color STATE es del tipo enumerado y representa los estados en que un dispositivo intentando establecer una conexión Bandabase puede estar. Estos estados fueron explicados en la Sección 4.2.2. El color TYPE es un enumerado y representa los tipos de paquetes involucrados en el intercambio de mensajes entre el maestro y el esclavo que están intentando establecer una conexión. El color AC es también un tipo enumerado y contiene el parámetro de control de acceso usado para identificar dispositivos durante el establecimiento de una conexión. El color PACKET es el producto del tipo TYPE y AC y representa un paquete Bandabase. El color TIND es un enumerado con un solo valor usado para controlar ciertas acciones en el modelo. Las demás declaraciones corresponden a variables (var) usadas en el modelo.

```
(* Standard declarations *)
     colset E = with e;
     colset INT = int;
     colset BOOL = bool;
     colset STRING = string;
     colset STATE = with STANDBY|
                INQUIRY|INQUIRYSCAN|
                INQUIRYRESPONSE|
                PAGE|PAGESCAN|
                MASTERRESPONSE|
               SLAVERESPONSE|
               CONNECTION;
     colset TYPE = with ID|FHS|POLL|SLOT;
     colset AC = with IAC|GIAC|DAC|NLL|CAC;
     colset PACKET = product TYPE * AC;
     colset TIND =with  s1|s2;
     var state: STATE;
     var prevstate: STATE;
     var anypacket: TYPE;
     var state2: STATE;
     var par: AC;
     var packet: PACKET;
```

Figura 4.9: Declaración global.

4.3.4 Módulo del Establecimiento de una Conexión en el Maestro

La jerarquía del modelo del establecimiento de una conexión Bandabase consiste de ocho (8) módulos incluyendo el mostrado en la Figura 4.8. La Figura 4.10 muestra el módulo de establecimiento de una conexión en el maestro. La misma incluye dos (2) transiciones de sustitución, INQUIRY y PAGE, las cuales modelan los dos grandes procedimiento envueltos en el establecimiento de una conexión, es decir *inquiry* y *page* (ver Sección 4.2.2), respectivamente.

Ellos se expanden en sus respectivos módulos. La plaza MASTERSTATE es del tipo STATE y representa los estados del establecimiento de una conexión Bandabase en los cuales un dispositivo maestro puede estar. Las otras plazas son plazas puertos de las plazas descritas en la Sección 4.3.2.

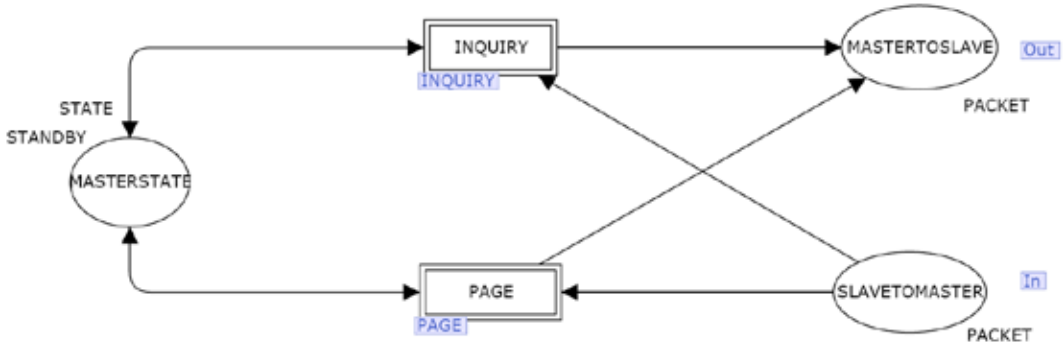


Figura 4.10: Módulo del establecimiento de una conexión en el maestro.

4.3.5 Módulo del Establecimiento de una Conexión en el Esclavo

La Figura 4.11 muestra el módulo de establecimiento de una conexión en el esclavo. La misma incluye dos (2) transiciones de sustitución, INQUIRYSCAN y PAGESCAN, las cuales modelan los dos grandes procedimiento envueltos en el establecimiento de una conexión, es decir *inquiryscan* y *pagescan* (ver Sección 4.2.2), respectivamente. Ellos se expanden en sus respectivos módulos. La plaza SLAVESTATE es del tipo STATE y representa los estados del establecimiento de una conexión Bandabase en los cuales un dispositivo esclavo puede estar. Las otras plazas son plazas puertos de las plazas descritas en la Sección 4.3.2.

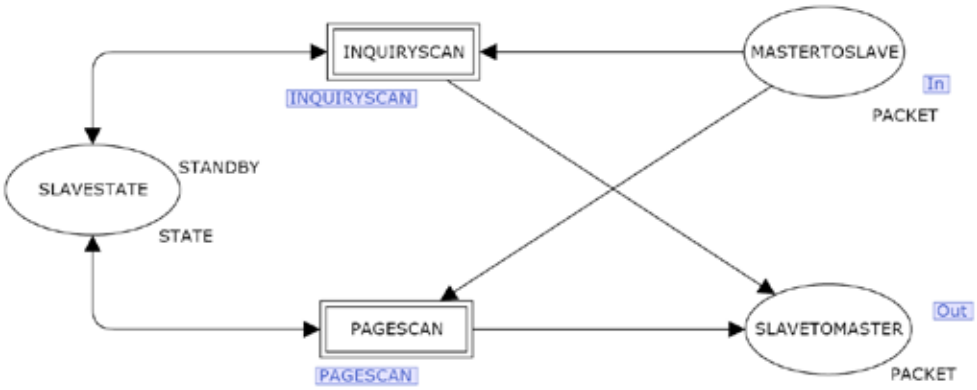


Figura 4.11: Módulo del establecimiento de una conexión en el esclavo.

4.3.6 Módulo del *Inquiry*

La Figura 4.12 muestra el módulo que modela el procedimiento de *inquiry* realizado por un dispositivo que pasa a ser un potencial maestro. La misma incluye siete (7) transiciones, las cuales modelan las acciones para pasar de un estado a otro. Un dispositivo puede entrar al modo

de *inquiry* automáticamente basado en un rango periódico especificado o manualmente cuando se invoca un comando de HCI Inquiry [8]. Las transiciones Inquiry_Period-Ends y HCI_Inquiry modelan estas dos acciones, respectivamente. Dichas transiciones hacen que un dispositivo salga del estado de STANDBY y entre en el estado INQUIRY y que se transmitan paquetes de indagación (ID). El dispositivo continua indagando a través del envío de paquetes ID, lo cual es modelado por la transición Inquiry-Continues. Entre las transmisiones de estos paquetes, el dispositivo puede escuchar por respuestas (es decir paquetes FHS) y esto es modelado por la transición ResponseReceived. El período de indagación termina cuando se genera un comando de cancelación de la indagación, cuando se ha alcanzado un *time out* (*inquiryTO*) o cuando el proceso es parado por el Manejador de Recursos Bandabase [8] porque se han recibido suficientes respuestas. Estas acciones son modeladas por las transiciones HCI_Inquiry_Cancel, InquiryTO y SufficientNumberResponses, respectivamente. Cuando alguna de estas transiciones ocurre el dispositivo regresa al estado de STANDBY.

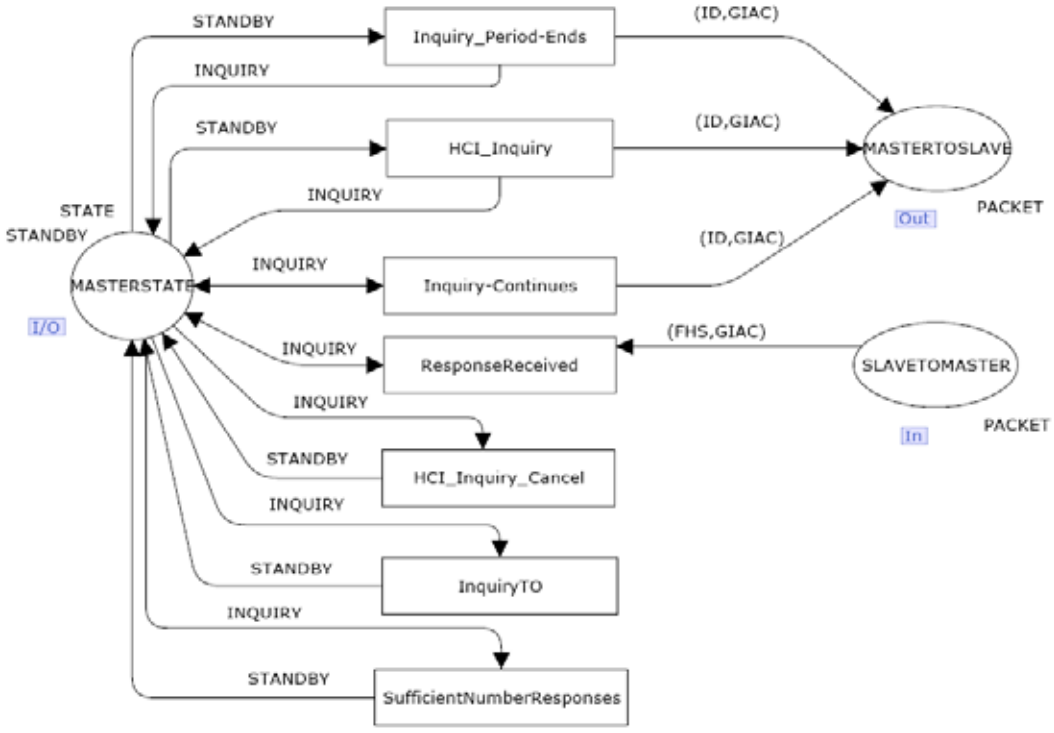


Figura 4.12: Módulo del *Inquiry*.

4.3.7 Módulo del *Page*

La Figura 4.13 muestra el módulo que modela el procedimiento de *page* realizado por el potencial maestro. La misma contiene diez (10) transiciones. La transición HCI_Create_Connection modela las acciones ejecutadas cuando se invoca un comando HCI_Create Connection [8], el cual causa que un dispositivo comience un proceso de *page* para establecer una conexión. Cuando esta transición ocurre el dispositivo pasa del estado de

STANDBY al estado de PAGE y comienza a transmitir paquetes ID con el código de acceso del dispositivo (DAC) esclavo. En el estado de PAGE el dispositivo continua transmitiendo paquetes de *page* (es decir paquetes ID) tal como es modelado por la transición Paging. Si un *time out* (*pageTO*) es excedido, el dispositivo debe retornar al estado de STANDBY. Esto es modelado por la transición pageTO. Por el contrario, la transición SlaveResponse modela la acciones ejecutadas por un dispositivo cuando una respuesta (ID, DAC) del potencial esclavo es recibida. En este caso el dispositivo entra en el estado de MASTERRESPONSE y envía un paquete FHS al esclavo. Después de que el maestro ha enviado este paquete, él debe esperar por una segunda respuesta del esclavo, reconociendo la recepción del paquete FHS. Esto es modelado por la transición ScndSlaveResponse. En este caso el maestro pasa al estado de conectado (CONNECTION) y transmite su primer paquete (un paquete de POLL).

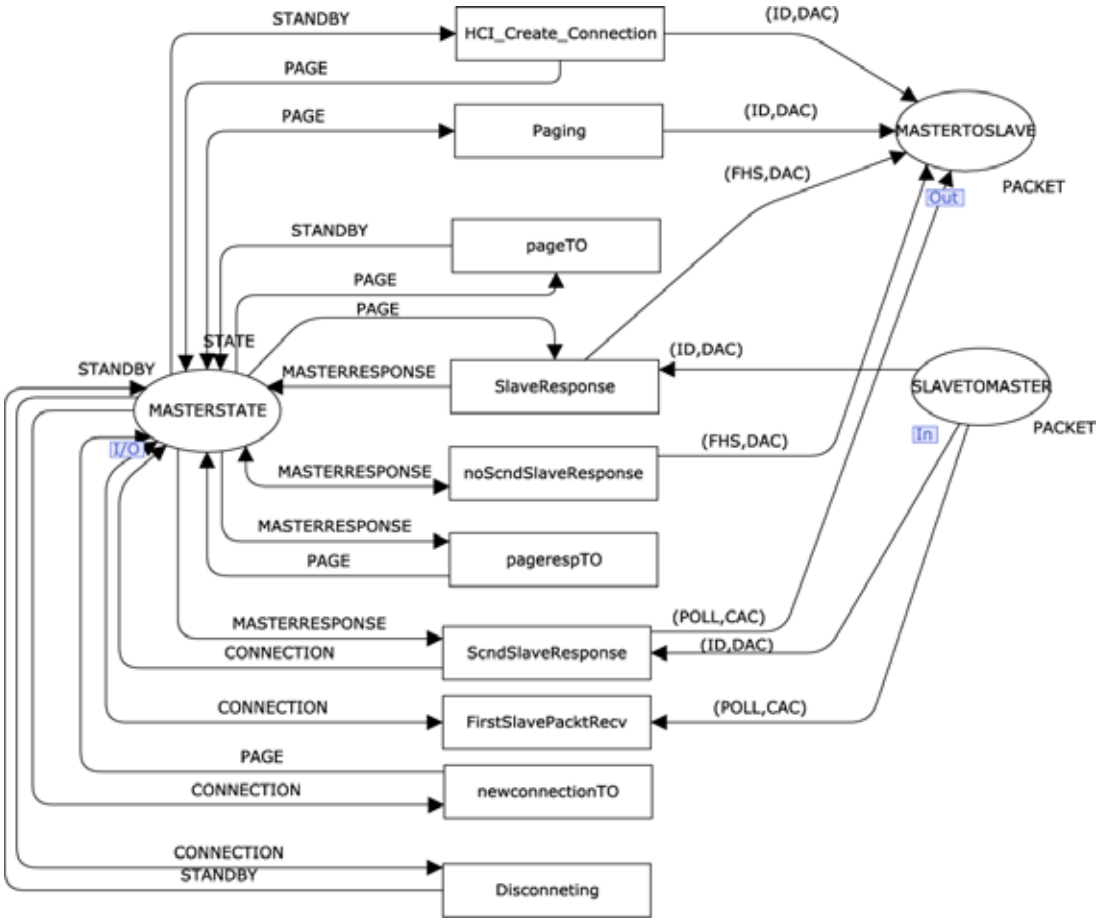


Figura 4.13: Módulo del *Page*.

En caso de que el maestro no reciba una segunda respuesta del esclavo, él debe retransmitir el paquete FHS. Esto es modelado por la transición ScndSlaveResponse. Este proceso debe continuar hasta que se recibe una respuesta o se excede un *time out* (*pageRespTO*), lo cual es modelado por la transición pageRespTO. En cuyo caso el dispositivo debe retornar al estado de PAGE.

El maestro debe esperar por una respuesta del esclavo al paquete de POLL, si el mismo no es recibido dentro de *newconnectionTO* número de *slots*, el maestro debe retornar al estado de PAGE. Esto es modelado por la transición newconnectionTO. Por el contrario, la transición FirstSlavePacktRecv modela las acciones ejecutadas cuando esta respuesta del esclavo ha sido recibida. Finalmente, un dispositivo puede desconectarse, Disconneting, y retornar al estado de STANDBY.

4.3.8 Módulo del *Inquiry Scan*

La Figura 4.14 muestra el módulo que modela el procedimiento de *inquiry scan* realizado por un dispositivo que pasa a ser un potencial esclavo. La misma incluye cinco (5) transiciones. La transición Inquiry_Scan_Interval-Ends modela las acciones ejecutadas por un esclavo cuando entra al período de búsqueda de paquetes de indagación. La ocurrencia de esta transición hace que el esclavo pase al sub estado de INQUIRYSCAN. La transición InquiryMsgReceived modela las acciones ejecutadas por un esclavo una vez que recibe un paquete de *inquiry*, paquete ID, con un código de acceso de indagación general (GIAC) del maestro. En este caso el maestro envía un paquete FHS con el GIAC correspondiente al maestro y se cambia al estado de INQUIRYRESPONSE. Un esclavo puede salir del estado de INQUIRYSCAN una vez finalizado el período de *scan*; esto es modelado por la transición Inquiry_Scan_Window-Ends. La ocurrencia de esta transición hace que el esclavo se mueva al estado STANDBY. Dado que un problema de contención puede ocurrir cuando varios dispositivos esclavos tratan de responder los paquetes de *inquiry* enviados por el maestro, la especificación establece un procedimiento para aliviar este problema [8]. Este procedimiento dispone que el esclavo deba retornar al estado de STANDBY o CONNECTON dependiendo del caso, pasando a través del estado de INQUIRY SCAN, e intentar el proceso de *inquiry scan* nuevamente después de esperar cierto tiempo calculado de forma aleatoria. En este punto la especificación establece que el dispositivo debe pasar a través del estado de PAGE SCAN (ver Sección 8.4.3 de [8]). Esto contradice el diagrama de la Figura 4.7, por lo cual el autor deduce que existe un error de trascripción en la especificación, siendo lo correcto que el dispositivo debe pasar a través del estado de INQUIRY SCAN, tal como se ha asumido en este libro. Esto es modelado por las transiciones Backing_Off e InTransitToStandby y el uso de la plaza InTransit.

4.3.9 Módulo del *Page Scan*

La Figura 4.15 muestra el módulo que modela el procedimiento de *page scan* realizado por un dispositivo esclavo. Hay ocho (8) transiciones. Un dispositivo esclavo en el estado de STANDBY puede cambiar al estado de PAGESCAN con la finalidad de escuchar por paquetes de *page scan* (es decir, paquetes ID) enviados por el maestro en el estado de PAGE y tratar así de establecer una conexión de forma exitosa. Esto es modelado por la transición ScanningDevices. Una vez recibido el paquete de *page* enviado por el maestro, es decir, el paquete del tipo ID, el esclavo entra el estado SLAVERESPONSE y le responde al maestro con un paquete ID con un DAC. Esto es modelado por la transición Slave_Responds. La transición

Page_Scan_Interval_Ends modela las acciones ejecutadas por un esclavo una vez que finaliza el tiempo de *page scan*. La ocurrencia de dicha transición hace que el dispositivo regrese al estado de STANDABY.

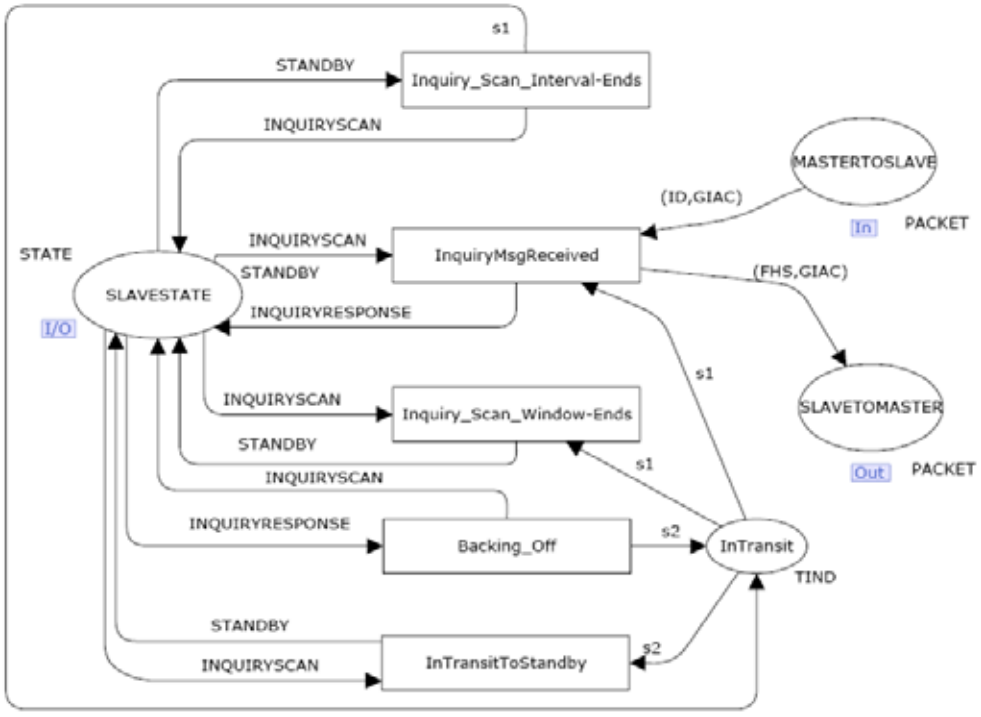


Figura 4.14: Módulo del *Inquiry Scan*.

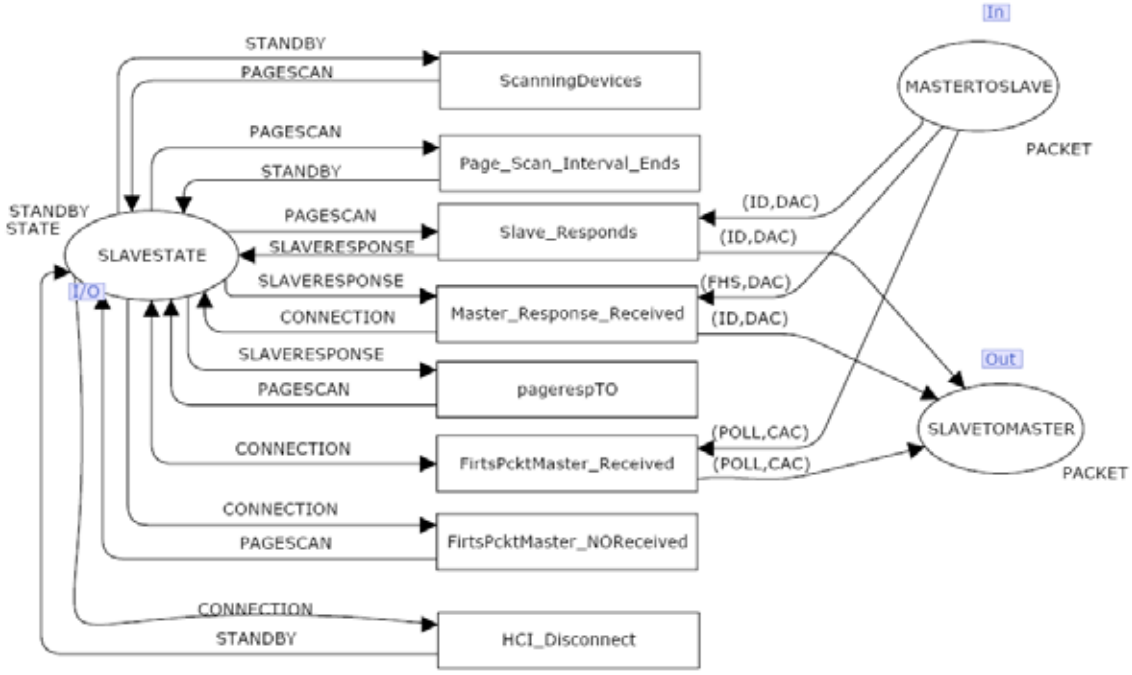


Figura 4.15: Módulo del *Page Scan*.

Si el esclavo recibe un paquete FHS del maestro, él entra en el estado de CONNECTION y realiza varias acciones modeladas por la transición Master_Response_Received, incluyendo el envío de una respuesta (i.e. un paquete del tipo ID) con un código DAC al maestro. Por el contrario, si el esclavo no recibe ningún paquete de respuesta del maestro por un tiempo llamado *pageTO* retorna al estado de PAGESCAN, lo cual es modelado por la transición pagerespTO.

Si la conexión es exitosa, es decir el esclavo está en el estado de CONNECTION, el mismo debe responder al mensaje de POLL enviado por el maestro. Como se dijo anteriormente, el esclavo puede contestar con cualquier paquete, pero en este libro, a fin de simplificar el análisis del modelo, se asume que el maestro responde con un paquete del tipo POLL. Esto se modela con la transición FirtsPcktMaster_Received. En caso contrario, el mensaje de POLL no es recibido o no es contestado, entonces el esclavo debe retornar al estado de PAGESCAN, tal como es modelado por la transición FirtsPcktMaster_NOReceived. Similarmente al maestro, un esclavo puede desconectarse en cualquier momento, retornando al estado de STANDBY; lo cual es modelado por la transición HCI_Disconnect.

4.3.10 Módulo del *Communication Channel*

La Figura 4.16 muestra el módulo que modela el canal de comunicación compartido entre los dispositivos Bluetooth. La misma tiene dos (2) transiciones. Los paquetes intercambiados entre dispositivos pueden perderse o dañarse en el camino, esto es modelado por las transiciones MASTERLOST y SLAVELOST.

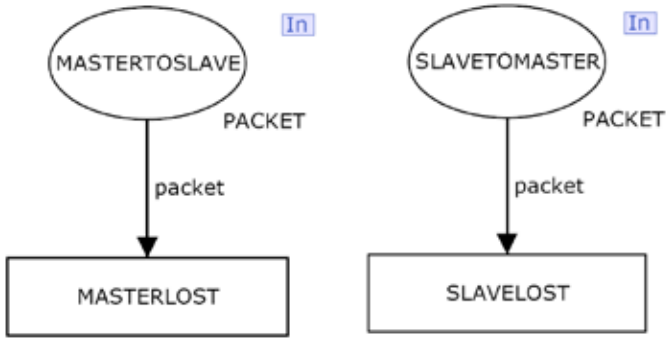


Figura 4.16: Módulo del *Communication Channel*.

4.4 Propiedades del Protocolo de Establecimiento de la Conexión Bandabase

Se desea verificar el protocolo del establecimiento de la conexión Bandabase de Bluetooth en función de un conjunto de propiedades. Estas incluyen propiedades específicas relacionadas al establecimiento de la conexión, las cuales son definidas en esta sección.

Inicialmente, se especifica la notación CPN [31] necesaria para definir las propiedades formalmente:

- M(p) denota el marcado de la plaza *p*.
- [M > denota el conjunto de marcados que son alcanzables desde el marcado M, con [M_0 > denotando el conjunto de marcados alcanzables desde el marcado inicial M_0.

Algunas propiedades se definen usando ASK-CTL cuya notación se explicó en la Sección 3.4.

5.1.1. Propiedad de Establecimiento de la Conexión

El propósito de un procedimiento de *page* es establecer una conexión entre un maestro y un esclavo. Así, la propiedad se define a continuación. Una vez que un maestro entra en el sub estado de *page*, es posible establecer una conexión con el dispositivo esclavo.

Sea $\mathbb{M}_{PAGEMASDEV}$ el conjunto de marcados donde el sub estado del maestro es *page*:

$$\mathbb{M}_{PAGEMASDEV} = \{M \in [M_0> \mid M(MASTERSTATE) = PAGE\}$$

Sea $\mathbb{M}_{CONNECTEDDEV}$ el conjunto de marcados donde el estado del maestro y el esclavo es CONNECTION:

$$\mathbb{M}_{CONNECTEDDEV} = \{M \in [M_0> \mid M(MASTERSTATE) = CONNECTION \wedge M(SLAVESTATE) = CONNECTION\}$$

Definición 1: El modelo CPN satisface la propiedad de establecimiento de la conexión si y sólo si:

$$\forall M \in \mathbb{M}_{PAGEMASDEV}, \exists M' \in \mathbb{M}_{CONNECTEDDEV}: M' \in [M>$$ □

5.1.2. Propiedad de *Inquiry*

El objetivo de procedimiento de indagación (*inquiry*) es descubrir los dispositivos cercanos. La propiedad es definida a continuación. Una vez que el dispositivo maestro entra en el al sub estado de *inquiry* es siempre posible que él descubra un dispositivo esclavo en su proximidad.

Sea $\mathbb{M}_{INQUIRYMASDEV}$ el conjunto de marcados donde el sub estado del dispositivo maestro es *inquiry*:

$$\mathbb{M}_{INQUIRYMASDEV} = \{M \in [M_0> \mid M(MASTERSTATE) = INQUIRY\}$$

Sea $\mathbb{M}_{INQUIRYRESPONSESLADEV}$ el conjunto de marcados donde el sub estado del dispositivo esclavo es *inquiry response*:

$$\mathbb{M}_{INQUIRYRESPONSESLADEV} = \{M \in [M_0> \mid M(SLAVESTATE) = INQUIRYRESPONSE\}$$

Definición 2: El modelo CPN satisface la propiedad de *Inquiry* si y sólo si:

$\forall M \in \mathbb{M}_{INQUIRYMASDEV}, \exists M' \in \mathbb{M}_{INQUIRYRESPONSESLADEV}: M' \in [M>$ □

4.4.1 Propiedad de Retardo en el Establecimiento de la Conexión

Los dispositivos Bluetooth podrían experimentar un largo retardo en el establecimiento de la conexión [53], así la propiedad se define a continuación. El dispositivo maestro puede permanecer en el sub estado *page* por un largo tiempo.

Sea IsPageNode(M´)= M´(MASTERSTATE) = PAGE.

Definición 3: El modelo CPN satisface la propiedad de retardo en el establecimiento de la conexión si y sólo si:

$\forall M \in \mathbb{M}_{PAGEMASDEV}, ALONG\ (IsPageNode)$ □

4.4.2 Propiedad de Retardo de Indagación

Los dispositivos Bluetooth podrían experimentar un largo retardo realizando la búsqueda de dispositivos cercanos (*inquiry*) [53], así la propiedad se define a continuación. El dispositivo maestro puede permanecer en el sub estado de *inquiry* por un largo tiempo.

Sea IsInquiryNode(M´)= M´(MASTERSTATE) = INQUIRY

Definición 4: El modelo CPN satisfice la propiedad de retardo de indagación si y sólo si:

$\forall M \in \mathbb{M}_{INQUIRYEMASDEV}, ALONG\ (IsInquiryNode)$ □

4.4.3 Propiedad de Desconexión

Un dispositivo en el estado de conectado puede permanecer en este estado aun cuando el otro dispositivo este desconectado. La propiedad se define a continuación. Una vez que el dispositivo maestro o el esclavo entra al estado de *standby* (i.e. el dispositivo está desconectado), el otro dispositivo puede permanecer en el estado de CONNECTION.

Definición 5: El modelo CPN satisfice la propiedad de desconexión si y sólo si:

$\exists M \in [M_0 >: (M\ (MASTERSTATE) = STANDBY \wedge M(SLAVESTATE) = CONNECTION) \wedge \exists M' \in [M_0 >: (M'(MASTERSTATE) = CONNECTION \wedge M'(SLAVESTATE) = STANDBY)$ □

4.5 Análisis y Verificación

CPN Tools [15] es usado para similar y analizar el modelo CPN. Este modelo es analizado generando el OG y su correspondiente SCC. Sin embargo, el modelo CPN descrito en

la Sección 4.3, genera un grafo de estado infinito debido a los paquetes periódicos (de *inquiry* y *page*) que envía el maestro, y al hecho de que las plazas de comunicación (MASTERTOSLAVE y SLAVETOMASTER) no están acotadas. Así, se ha modificado el modelo usando una aproximación estándar [33][64], de forma tal que las plazas de comunicación tengan una capacidad finita.

La Figura 4.17 muestra el modelo modificado para el módulo del *Inquiry* descrito en la Sección 4.3.6. El paquete tipo SLOT ha sido incluido en el *color set* del tipo TYPE (ver Sección 4.3.3). Cada vez que se desea enviar un paquete debe existir suficiente capacidad en la plaza de comunicación MASTERTOSLAVE; dicha capacidad viene determinada por el número de marcas (SLOT,NLL). Cada vez que se recibe un paquete de la plaza SLAVETOMASTER debe colocarse una marca (SLOT,NLL). Todos los módulos del modelo han sido modificadas de forma similar (ver Apéndice A).

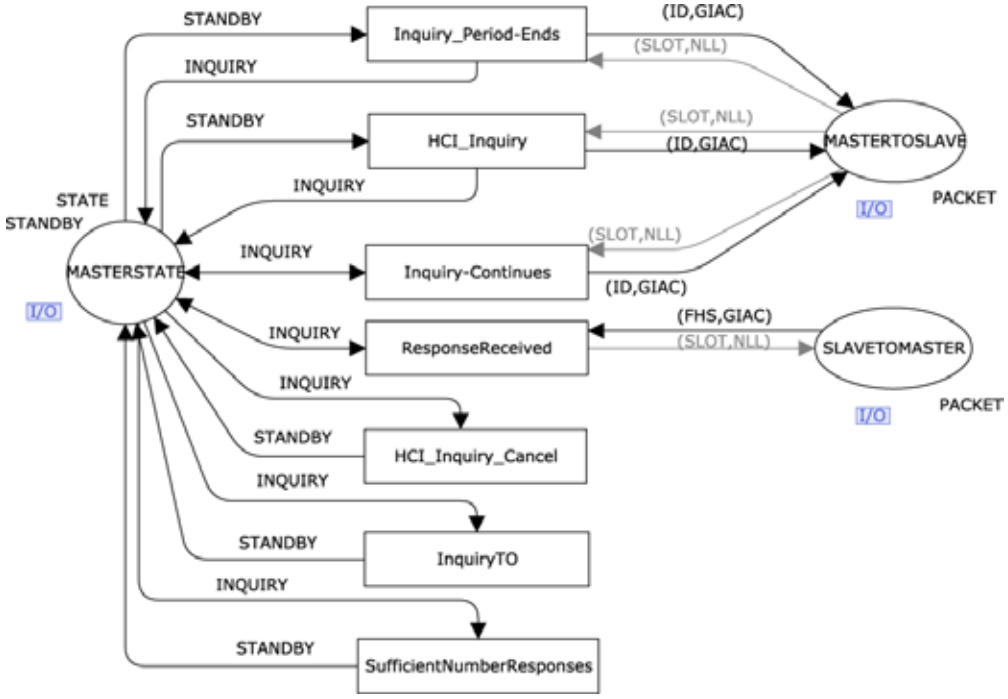


Figura 4.17: Módulo del *Inquiry* modificado.

Las propiedades son chequeadas implementando las definiciones descritas en la Sección 4.4 como funciones ML de indagación (*query*) del OG [32] y usando la librería ASK-CTL. Con la finalidad de ilustrar la aproximación usada para chequear las propiedades, en esta sección se describen las implementaciones de la propiedad de establecimiento de la conexión (ver Sección 5.1.1) y la de la propiedad de retardo de indagación (4.4.1). Todas las propiedades excepto la de retardo en el establecimiento de la conexión y la de retardo de indagación son verificadas examinando los nodos del OG. Las propiedades de retardo son chequeadas usando la librería de *model checking* embebida en CPN-Tools.

4.5.1 Marcado Inicial

A fin de analizar el modelo del establecimiento de la conexión Bandabase las plazas MASTERSTATE y SLAVESTATE son inicializadas con el estado de STANDBY. Las plazas MASTERTOSLAVE y SLAVETOMASTER son inicializadas para permitir un máximo de cinco (5) paquetes Bluetooth. Las demás plazas no tienen ninguna marca en el marcado inicial.

4.5.2 Estadística del Grafo de Ocurrencia

En el curso de las investigaciones [64] una serie de OGs fueron generados para diferentes marcados iniciales. Esto con la finalidad de ganar mayor confianza en el modelo en desarrollo. Por ejemplo, en el caso más simple, se considera un canal de capacidad dos. En este documento se presentan los resultados del modelo completo para el marcado inicial descrito anteriormente.

El tamaño del OG y del grafo SCC y los correspondientes tiempos de generación usando CPN Tools se muestran en la Tabla 4.2. CPN Tools corrió en una computadora con un procesador Intel Core 2 de 2.13 GHz PC con 3 GB RAM y corriendo el sistema operativo Windows. Ya que el tamaño del SCC es uno (1) el maestro o el esclavo pueden entrar al estado de STANDBY (es decir, el estado inicial) desde cualquier estado alcanzable. Esto está de acuerdo a lo mostrado en la Figura 4.7.

Tabla 4.2: Resultados del OG

Información estadística	**OG**	**Grafo SCC**
Número de nodos	200.364	1
Número de arcos	1.622.989	0
Tiempo de generación (hh:mm:ss)	(02:16:01)	(00:06:02)

4.5.3 Propiedades Generales

Las propiedades de acotamiento, local y vivacidad se chequearon para validar y depurar el modelo y para proporcionar una mejor comprensión del comportamiento del protocolo del establecimiento de la conexión Bandabase. Los resultados fueron obtenidos del reporte de OG estándar generado por CPN Tools y se muestran en la Tabla 4.3 y Tabla 4.4.

Acotamiento

Las cotas enteras y multi-conjuntos se analizaron para las plazas del modelo. Por ejemplo, la Tabla 4.3 lista las cotas enteras y multi-conjuntos para las plazas del modelo CPN. Las plazas MASTERSTATE y SLAVESTATE pueden tener un máximo de una marca. Esto es esperado ya que ellos indican el estado del maestro y del esclavo respectivamente. Los estados del maestro y

del esclavo varían entre todos los estados definidos en Figura 4.7. Cada una de las plazas de comunicación puede tener un máximo de cinco (5) marcas que corresponden con la máxima capacidad del canal de comunicación. Las cotas multi-conjuntos muestran que todos los paquetes Bluetooth requeridos para el establecimiento de la conexión son generados e intercambiados por el protocolo. Por ejemplo, la plaza MASTERTOSLAVE puede contener una marca representando un paquete ID con un código de acceso GIAC y es usado para colectar información acerca de otros dispositivos en la cercanía. Los resultados de la propiedad de acotamiento son esperados y confirman aún más que el modelo es válido.

Tabla 4.3: Cotas superiores para las plazas de comunicación

Plaza	**Cotas enteras**	**Cotas de multi conjuntos**
MASTERSTATE	1	1`STANDBY++1`INQUIRY++1`PAGE++ 1`MASTERRESPONSE++1`CONNECTION
SLAVESTATE	1	1`STANDBY++1`INQUIRYSCAN++ 1`INQUIRYRESPONSE++1`PAGESCAN++ 1`SLAVERESPONSE++1`CONNECTION
MASTERTOSLAVE	5	5`(ID,GIAC)++5`(ID,DAC)++ 5`(FHS,DAC)++5`(POLL,CAC)++ 5`(SLOT,NLL)
SLAVETOMASTER	5	5`(ID,DAC)++5`(FHS,GIAC)++ 5`(POLL,CAC)++5`(SLOT,NLL)

Marcados Muertos

No hay marcados muertos en el modelo CPN lo cual está acorde con el diagrama mostrado en la Figura 4.7.

Transiciones Vivas

Todas las transiciones del modelo están vivas y así el sistema puede siempre alcanzar el estado inicial donde ambos el maestro y el esclavo están en el estado STANDBY y las plazas de comunicación están vacías.

Marcados Locales

En el modelo todos los marcados son locales, así no importa que estado el sistema ha alcanzado siempre puede alcanzar otro estado incluyendo el estado inicial.

Marcados Muertos

El reporte estándar generado por CPN Tools muestra que no hay marcados muertos. Esto es esperado ya que no deberíamos tener "código muerto" en la especificación.

Tabla 4.4: Resultados de las propiedades de vivacidad y local

Propiedad	Resultado
Marcados muertos	Ninguna
Transiciones vivas	Todos
Marcados locales	Todos
Transiciones muertas	Ninguno

4.5.4 Código ML para el Chequeo de las Propiedades de Establecimiento de la Conexión y de Retardo de Indagación

Esta sección describe como dos de las propiedades, establecimiento de la conexión y retardo de indagación, son chequeadas usando funciones ML (*ML queries*). La implementación de cada una de las propiedades de establecimiento de la conexión y de retardo de indagación se simplifica porque en el modelo todos los marcados son locales como se muestran en la Tabla 4.4. Las implementaciones de las demás propiedades se encuentran en el Apéndice B. Así la propiedad de establecimiento de la conexión es chequeada usando la siguiente función ML:

```
1  fun ConnEstProp () = let
2  val pagemasdev = PredAllNodes (fn n  => (Mark.MASTERCONNSETUP'MASTERSTATE 1
   n = 1`PAGE));
3  val connecteddev = PredAllNodes (fn n  => (Mark.MASTERCONNSETUP'MASTERSTATE
   1 n = 1`CONNECTION andalso Mark.SLAVECONNSETUP'SLAVESTATE 1 n =
   1`CONNECTION));
4  in
5  (pagemasdev <> nil) andalso (connecteddev <> nil)
6  end;
```

La función **ConnEstProp** (línea1) chequea si una conexión entre el maestro, el cual está en el sub estado *page*, y un esclavo puede ser establecida. La función retorna verdad si ambas de las siguientes listas: la lista de todos los marcados donde el sub estado del maestro es igual a PAGE (línea 2) y la lista de todos los marcados donde el estado de ambos el dispositivo maestro y el esclavo es igual a CONNECTION (línea 3), no están vacías. La condición de marcado local no es chequeada en la función porque ha sido chequeada durante la generación del reporte estándar del OG. La propiedad de marcado local retornó que todos los marcados son marcados locales como se muestra en la Tabla 4.4.

La propiedad de retardo de indagación es chequeada usando la siguiente función ML:

```
1  fun IsInquiryNode n = Mark.MASTERCONNSETUP'MASTERSTATE 1 n = 1`INQUIRY;
```

```
2  fun InquiryLoop (n) = let
3    val ASKCTLformula = ALONG (NF("Is Inquiry Node",IsInquiryNode));
4  in
5     eval_node ASKCTLformula n
6  end;
7  fun InquiryDelayProp (a,b) = a andalso b;
8  fun InquiryDelayProperty () = SearchAllNodes (fn n  =>
   (Mark.MASTERCONNSETUP'MASTERSTATE 1 n = 1`INQUIRY), fn n => InquiryLoop
   n,true,InquiryDelayProp);
```

La función **InquiryDelayProperty** (línea 8) chequea si un dispositivo puede permanecer en el modo de indagación (INQUIRY) por un largo tiempo. La función retorna verdad si para todos los marcados donde el sub-estado del dispositivo maestro es igual a INQUIRY, la función **InquiryLoop** (líneas 2-6) retorna verdad. La evaluación de la función **InquiryLoop** (líneas 2-6) chequea si un maestro permanece en el estado de INQUIRY por un largo tiempo. La mimas invoca a la fórmula ASK-CTL, ALONG (línea 3) y retorna verdad si existe un camino el cual es infinito, a lo largo del cual el dispositivo maestro permanece en el sub estado INQUIRY para el marcado *n*. Finalmente, la función de combinación **InquiryDealyProp** (línea 7) combina el resultado en curso de la función **InquiryLoop** con el resultado previo usando el operador booleano *and*.

4.5.5 Propiedades del Establecimiento de la Conexión Bandabase

Las propiedades del protocolo Bandabase son verificadas ejecutando los *queries* correspondientes. La Tabla 4.5 lista los resultados, que indican que todas las propiedades se satisfacen (OK). Los resultados muestran que el establecimiento de la conexión Bandabase trabaja como se esperaba de acuerdo a la asunciones realizadas.

Tabla 4.5: Análisis de los resultados

Propiedad	Resultado
Establecimiento de la Conexión	OK
Indagación	OK
Retardo en el Establecimiento de la Conexión	OK
Retardo de Indagación	OK
Desconexión	OK

4.6 Resumen

Las CPNs se han utilizado para modelar el establecimiento de la conexión Bandabase de Bluetooth basado en una serie de asunciones. Se usa una topología de red simple pero representativa (una *piconet* con un dispositivo maestro y otro esclavo) para verificar que el protocolo funciona correctamente bajo estas condiciones. El principal problema encontrado

durante el modelado fue la falta de una especificación de Bluetooth bien definida [2], donde principalmente una descripción narrativa está presente. El modelo resultante proporciona una definición clara, inequívoca y precisa del protocolo de establecimiento de la conexión de banda base. El modelo fue desarrollado y comprobado de forma incremental en cada etapa para reducir la posibilidad de errores de modelado. El modelo es analizado en función de las propiedades generales del protocolo y un conjunto de cinco propiedades definidas y formalizadas en este documento. El análisis de tres de las propiedades se lleva a cabo mediante la consulta al OG mientras que el resto de las propiedades se analizaron usando la lógica temporal CTL, llamada ASK-CTL. El análisis del modelo muestra que el protocolo de Bandabase funciona como se esperaba bajo nuestros asunciones. Algunos trabajos de investigación, tales como el presentado en [53], han descrito el problema del retardo de establecimiento de la conexión de Bandabase. En este capítulo, el problema de la demora se demuestra mediante la verificación de las propiedades de retardo en el establecimiento de la conexión y retardo de indagación.

5. Construcción de MPDUs de WiMax Adaptativa y Óptima basada en Información de Retroalimentación

5.1. Introducción

WiMax es una tecnología de área metropolitana desarrollada por el Grupo de Trabajo de la IEEE 802.16 [25]. Esta tecnología proporciona una alternativa a las redes de acceso cableadas, tales como los enlaces de fibra óptica y las líneas de suscriptor digital (*digital subscriber line, DSL*). Comparado con las tecnologías de acceso cableadas, WiMax es más fácil de desplegar y es destinada a un acceso de banda ancha más ubicuo en el futuro. Los usuarios de WiMax denominados estaciones suscritoras (*subscriber stations, SS*) acceden a la red a través de redes externas comunicándose con estaciones centrales de radio llamadas estaciones bases (*base stations, BSs*). También WiMax es capaz de soportar tasas de datos mayores a los 350 Mbps usando canales múltiples para una transmisión simple [25]. Con la demanda creciente de servicios como la voz sobre IP (*voice over IP, VoIP*), el video bajo demanda y más recientemente la televisión sobre Internet (*Internet Protocol Television, IPTV*). WiMax se está volviendo más popular por su capacidad de proporcionar servicios adaptativos, tal como el video en tiempo real.

Aunque el estándar IEEE 802.16 ha definido las funcionalidades de la pila de protocolos, hay aún algunas características flexibles que permiten que WiMax se adapte a diferentes condiciones de los sistemas (tales como, tasas de error de bit variables, condiciones del canal cambiantes y alta movilidad del usuario), por ejemplo, el tamaño de un MPDU no es fijo y por el contrario puede ser cambiado antes de que la trama sea transmitida. Además, la capa física permite que cierta información de retroalimentación sea transmitida desde el receptor al transmisor, el cual puede tomar una decisión antes de transmitir las tramas subsecuentes, una característica altamente deseable para el envío de video. Más aún, el modelo de referencia del estándar 802.16 de la IEEE incluye el protocolo de la capa física que implementa la corrección de errores hacia adelante (*forward error correction*, *FEC*).

Las características antes mencionadas son adecuadas para el envío de caudales de información en tiempo real a través de enlaces inalámbricos caracterizados por su limitado ancho de banda, alta velocidad de transmisión de los datos y fluctuaciones periódicas de la red, así

como también para los altos requerimientos de calidad de servicio (*quality of service, QoS*) de las aplicaciones multimedia. Así, en este libro, se considera el tamaño óptimo del MPDU con diferentes valores de las tasas de error de bit (*bit error rate, BER*) y se propone un mecanismo para ajustar el tamaño óptimo del MPDU de acuerdo con las condiciones cambiantes del canal. Las condiciones del canal se estiman basándose en la información de retroalimentación definida en [11] y proporcionada por el receptor. En cuanto a la evaluación de la calidad de video se refiere, se usa el modelo de subjetividad de la ITU-T especificado en [30] en lugar de los modelos de QoS tradicionalmente utilizados. También se compara el esquema adaptativo propuesto basado en la eficiencia de la capa MAC.

5.2. Trabajos Relacionados

En lo que a mecanismos de retroalimentación se refiere, Chatterjee et al. [11] han propuesto un mecanismo de construcción de MPDUs adaptativa basada en retroalimentación para soportar la transmisión de datos vía *streaming* a través de canales inalámbricos. Un MPDU se construye basado en la información de retroalimentación recibida desde el receptor. El tipo de retroalimentación cambia de acuerdo con las condiciones del canal y por lo tanto el tamaño del MPDU es adaptado para que coincida con las condiciones del canal. Además, se clasifican los MPDUs de acuerdo con la importancia del nivel de la trama de video. En [11], el aumento / disminución del tamaño de la carga útil y el tamaño de la palabra de código FEC no está optimizado. Además, los parámetros se eligen de manera ad hoc. Así, en este libro, se propone un mecanismo de construcción de MPDUs adaptativa basada en retroalimentación con tamaños óptimos del MPDU y de la palabra código FEC. Se estima el tamaño óptimo del MPDU de una manera similar a como se estima en [24] y [37]. A diferencia de Hoymann [24] y Martikainen et al. [37], en esta investigación se estima no sólo el tamaño óptimo de la carga útil del MPDU sino también el número óptimo de bytes de redundancia en una palabra de código FEC, y se utiliza la información de retroalimentación recibida desde el receptor para adaptar los tamaños de los MPDUs a los valores óptimos de acuerdo con las condiciones cambiantes del canal.

En cuanto a la evaluación de diferentes enfoques de retroalimentación, la mayor parte de los trabajos de investigación se ha centrado en los parámetros de red o las métricas de video. Ninguno de estos captura la subjetividad que está asociada con la calidad del video. Por ejemplo, dos flujos de video expuestos a las mismas condiciones de la red pueden tener resultados finales muy diferentes. Recientemente, ha habido un énfasis por parte de los operadores de la red para capturar las expectativas de los usuarios finales, técnicamente conocidos como calidad de la experiencia (*quality of experience, QoE*) [17]. Varios modelos de opinión para el habla como el *modelo E* se han especificado y han sido ampliamente utilizados, sin embargo, se les ha dado una consideración modesta a los modelos de opinión para la estimación de la calidad del video. Uno de estos esfuerzos de investigación se muestra en [60], donde los autores trataron de encontrar

una definición de la Puntuación de Opinión Media (*Mean Opinion Score, MOS*) del video mediante la realización de varios experimentos en un *test bed* de simulación.

En este libro se utiliza el modelo subjetivo de la ITU-T [30] para estudiar la propuesta adaptativa descrita a continuación.

5.3. Introducción a la Capa MAC y la Capa Física

La Figura 5.1 muestra el modelo de referencia del IEEE 802.16 [25]. El mismo consta de dos componentes principales: el plano de datos / control y el plano de gestión. Las funciones del plano de datos se utilizan para manejar la información del usuario a ser transmitida por el transmisor, o para ser recibida por el receptor. El plano de datos incluye las capas física y MAC. Esta última se divide a su vez en tres sub-capas: subcapa de convergencia de servicio específico (*Service Specific Converge Sublayer,* CS), subcapa de parte común MAC (*MAC Common Part Sublayer,* MAC CPS), y subcapa de seguridad. Las funciones del plano de control están relacionadas con la gestión de la conexión y soportan varias configuraciones de las capas física y MAC. El plano de gestión se divide, también, en varias capas y sus funciones están relacionadas con la gestión de las tramas (por ejemplo clasificación de las tramas), de la seguridad, del QoS y con la configuración de la conexión. Una descripción detallada de estos planos, capas y subcapas están fuera del alcance de este documento, sin embargo, puede encontrarse en [25]. A continuación se van a describir algunas de las funciones de la MAC CPS y la capa física, que son relevantes para el alcance de este capítulo.

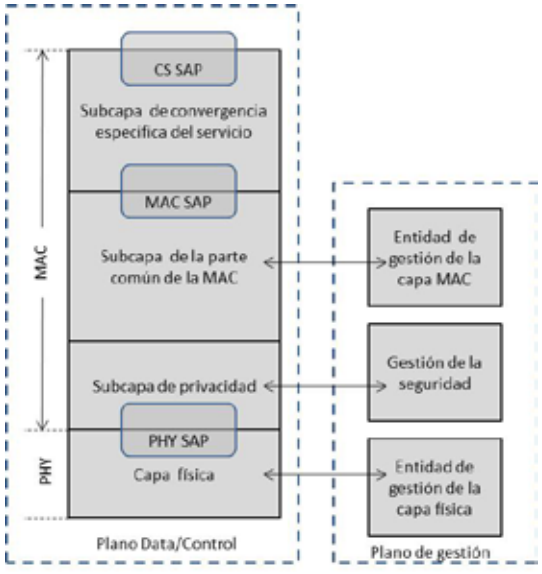


Figura 5.1: Modelo de referencia del IEEE 802.16.

5.4. Capa MAC

Al igual que con otras tecnologías que funcionan en un medio compartido, el IEEE 802.16 MAC CPS es responsable de controlar el acceso al medio. Las otras funciones básicas de la capa MAC son la encapsulación/des encapsulación, el empaquetamiento y la fragmentación de datos. También esta capa es responsable del control de errores de datos (detección de errores y los mecanismos de retransmisión) y proporciona mecanismos para el control del tráfico y el aprovisionamiento de QoS. La capa MAC es orientada a conexión, por lo que todos los servicios, incluidos los servicios sin conexión se asignan a una conexión.
Por otra parte, un MSDU está particionado lógicamente en bloques ARQ (*Automatic Repeat Request*) [18][25], que representan las unidades básicas de transmisión y retransmisión. Un bloque ARQ tiene una longitud fija, entre 1 y 2040 bytes, la cual es negociada durante el establecimiento de la conexión.

Dos características importantes de como puede construirse un MPDU se especifican en el estándar IEEE 802.16: empaquetamiento y fragmentación (ver la Figura 5.2). La capa MAC puede incluir más de un MSDU en un solo MPDU. De forma similar, un MSDU puede ser fragmentado en dos o más MPDUs. El uso simultáneo de la fragmentación y el empaquetamiento también es permitido. Por ejemplo, un MPDU puede tener capacidad para más de dos MSDUs completas pero no tres, por lo tanto, la tercera MSDU es fragmentada y parte de ella se acomoda con los dos MSDUs anteriores para llenar el campo de carga útil restante, para así evitar el desperdicio de recursos. La otra parte restante se incluirá en otro MPDU. La fragmentación sólo es permitida en los límites de los bloques ARQ.
Todas estas características son explotadas y se hace uso del empaquetamiento, la fragmentación y la retransmisión basada en bloques ARQ en los esquemas adaptativos para la construcción y transmisión de MPDUs descritos en este libro.

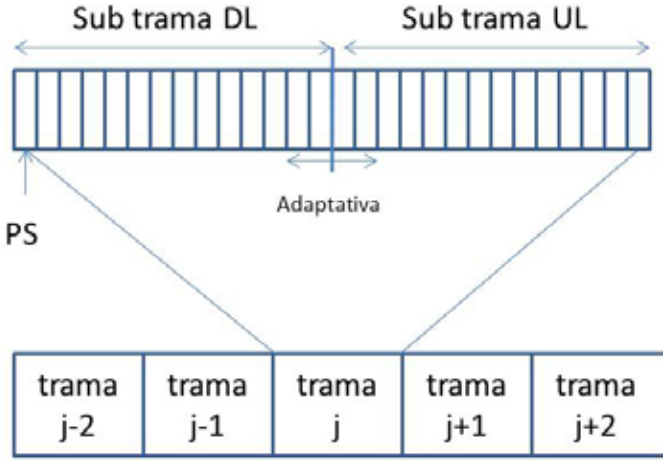


Figura 5.2: Estructura de una trama TDD.

5.5. Capa Física

La especificación de la capa física definida para trabajar en la banda de 10-66 GHz soporta la configuración de ráfaga adaptativa en los cuales algunos parámetros de transmisión, tal como la modulación y los esquemas de codificación, pueden ser cambiados por cada conexión o por cada suscriptor para adaptarse a las condiciones cambiantes del canal y para proporcionar niveles de servicio variantes. Las técnicas de duplexación por división de tiempo (*Time Division Duplexing*, *TDD*) y la duplexación por división de frecuencia (*Frequency Division Duplexing, FDD*) son soportadas. Los datos encapsulados en los MPDUs son transportadas en tramas físicas TDD o FDD, siendo la duración de una trama física fija para una configuración especifica del sistema y elegida entre tres posibles valores (0,5;1;2 ms) [25]. Cada trama incluye una trama de subida y una trama de bajada. Así, en la operación FDD, el canal de subida (*uplink*, UL) y el de bajada (*downlink*, DL) están ubicados en frecuencias separadas, mientras que en TDD, estos canales están separados como se muestra en la Figura 5.2. La duración de las subtramas puede ser cambiada de acuerdo a la cantidad de tráfico DL/UL y la calidad de servicio (QoS).

La técnica FEC utilizada es Reed-Solomon (RS), campo de Galois (Galois Field), GF (256), y, para algunas especificaciones físicas, se admiten tamaños de bloques y capacidades de control de errores variables. Así, una trama se divide en ranuras físicas (*physical slots*, PSs), que son las unidades básicas de asignación de ancho de banda. La definición de un PS depende de la alternativa de la capa física, por ejemplo, para las especificaciones de una sola portadora física un PS se define como 4 símbolos (*quadrature amplitude modulation, QAM*). El número de PSs por cada trama es una función de la velocidad de símbolos. Por ejemplo, con 20 Mbaudios de velocidad de símbolo, hay 5000 PSs en una trama de 1 ms.

Una subtrama TDD DL comienza con cierta información utilizada para propósitos de sincronización y ecualización, seguido por una sección de control que incluye un *dowlink-map* (*DL-MAP*) y un *uplink-map* (*UL-MAP*). El DL-MAP especifica cuando las transiciones de perfil de ráfaga ocurren dentro de una ráfaga DL, mientras que el UL-MAP describe asignaciones de ancho de banda, junto con el perfil de ráfaga, a los SSs específicos que transmiten datos en el canal UL. La siguiente sección de la subtrama DL lleva los datos organizados en orden decreciente de robustez.

Los datos son siempre codificados usando FEC y transmitidos en la porción TDM de la sub trama DL. Así un cierto número de PSs son asignados a una ráfaga en particular. Si la última parte de los datos no es suficiente para llenar un bloque FEC, el último bloque de la ráfaga se puede acortar.

Entre las capas MAC y PHY se encuentra la sub capa de Convergencia de Transmisión (*Transmission Convergence Sublayer, TCS*), la cual es responsable de la segmentación de los MPDUs en bloques de datos que se ajusten al tamaño de la palabra de código correspondiente después de que se ha añadido un puntero de un byte. El puntero identifica el número del byte en

el paquete en el cual comienza el primer MPDU o el principio de cualquier conjunto de bytes de relleno que preceden al siguiente MPDU. La Figura 5.3 muestra cómo las diferentes unidades de datos están encapsuladas en las diferentes capas definidas en el modelo de referencia. En este libro se asume el uso de ráfagas TDMA en la dirección *downlink* para las transmisiones de video.

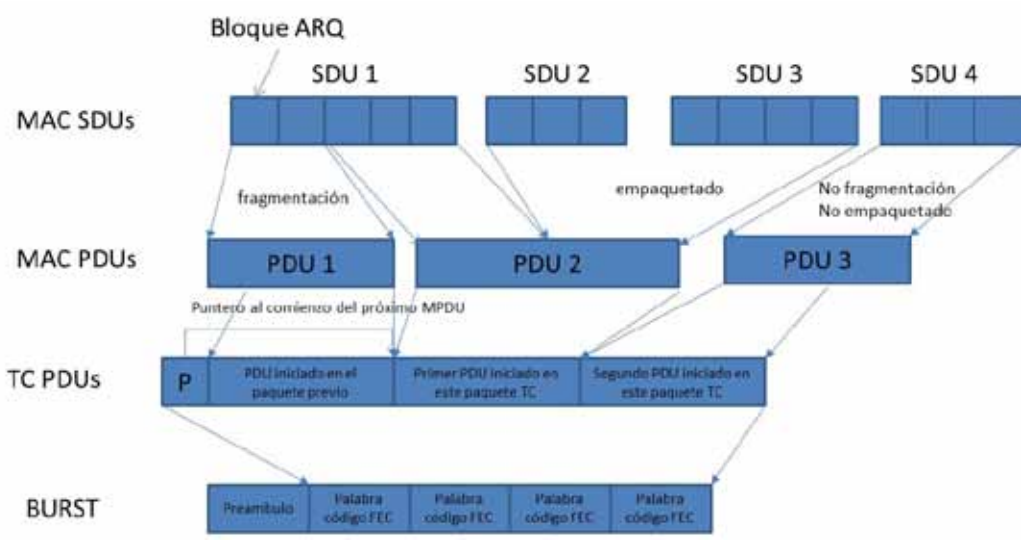


Figura 5.3: Transmisión de PDUs.

5.6. Construcción de MPDUs Adaptativa Basada en Retroalimentación

En esta sección se discuten dos aproximaciones adaptativas para la construcción de MPDUs en WiMax.

5.6.1. Aproximación Adaptativa con Incrementos/Decrementos Fijos (Ad Hoc)

El mecanismo de construcción de MPDU adaptativa basada en retroalimentación propuesto en [11] incluye información provista por el receptor (usualmente el SS) y un conjunto de acciones tomadas por el transmisor (usualmente la BS) basado en dicha información. La información de retroalimentación es provista por el receptor cuando recibe un MPDU basado en si la trama física ha sido recibida correctamente o con errores, en este último caso se tomará en cuenta si la trama se puede corregir. En este libro, se considera un subconjunto de los tipos de retroalimentación definidos en [11] y los cuales son mostrados en la Tabla 5.1.

Tabla 5.1: Diferentes posibilidades de retroalimentación

Tipo de retroalimentación	Clasificación de la retroalimentación
1	MPDU recibido correctamente
2	MPDU recibido con errores, pero corregible
3	MPDU recibido con errores, pero incorregible

Varias acciones pueden ser tomadas por la BS después de recibir la información de retroalimentación. Inicialmente, los MPDUs se clasifican de acuerdo a ciertos niveles de prioridad. Se utiliza un enfoque simple para clasificar las tramas de video, que consisten en categorizar un MPDU como *importante* y no *tan importante*. Los codificadores MPEG-2/4 clasifican las tramas de video en I, B y P. Las tramas I no requieren información de tramas anteriores o posteriores para la codificación o descodificación. Las tramas B y P son necesarias para la codificación y decodificación de otras tramas. Por lo tanto, las tramas I se consideran *importantes*, mientras que las tramas P y B *no son tan importantes*.

La Tabla 5.2 muestra las acciones que deben ser tomadas por la BS de acuerdo a la información de retroalimentación recibida. El tamaño de la carga útil y el número de bytes de redundancia en cada *palabra código FEC* (o FEC) aumentan / disminuyen de acuerdo con los valores de las variables *ni* (i = 1,2) y *mj* (j = 1 .. 4), respectivamente. Las variables se expresan en bytes y son dependientes de la implementación.

Tabla 5.2: Acciones tomada por la BS para los correspondientes tipos de retroalimentación

Tipo de retroalimentación	Incremento de la carga útil		Decremento de la carga útil		Incremento del FEC		Decremento del FEC	
	imp tramas	no imp tramas	imp tramas	no imp tramas	imp tramas	no imp tramas	imp tramas	no imp tramas
1	n1	n1	0	0	0	0	0	m1
2	0	0	0	0	m2	0	0	0
3	0	0	n2	n2	m3	m4	0	0

imp significa importante

5.6.2. Aproximación Adaptativa con Incremento/Decremento Óptimo

El principal problema con el enfoque descrito anteriormente es que los tamaños de la carga útil del MPDU y del código FEC se aumentan / disminuyen de manera ad hoc, es decir, no hay un criterio para la determinación de los valores de los parámetros que se muestran en la Tabla 5.2. Así, la utilización del mecanismo de construcción del MPDU bajo el enfoque anterior puede conducir a una utilización pobre del canal. Por lo cual se extiende la propuesta anterior y se propone una alternativa para definir los tamaños de la carga útil del MPDU y del código FEC. Con el nuevo enfoque, estos tamaños se cambian de acuerdo con las condiciones del canal. Los tamaños óptimos de la carga útil del MPDU y del código FEC para cada BER del canal se

calculan como se explica en la Sección 5.7. Por lo tanto, el MPDU se puede construir siguiendo cualquiera de las opciones siguientes. En primer lugar, si cierta información acerca de las tramas erradas no se puede obtener antes de la transmisión, la información de retroalimentación recientemente recibida desde el receptor (ver la Sección 5.6.1) y la información de retroalimentación anterior se pueden utilizar para cambiar los tamaños del MPDU y del código FEC de acuerdo con la Tabla 5.3. Para lograr esto, se podría estimar las variaciones de las condiciones del canal en términos del BER para establecer la relación entre la información de retroalimentación y el estado del canal. Por ejemplo, si la información de retroalimentación anterior es 1 y el valor de retroalimentación recibida es 3, los tamaños óptimos del MPDU y del código FEC pueden ser cambiados a los valores óptimos correspondientes a las malas condiciones del canal con BER de 10^{-2}. En segundo lugar, otra posibilidad es que el receptor no sólo envíe información de retroalimentación al emisor, sino también algo de información acerca de las condiciones de canal (por ejemplo, ver [37][57]). Basándose en la información recibida, el remitente puede adaptar los tamaños del MPDU y de la palabra de código FEC a los valores óptimos, dadas las condiciones existentes del canal. A continuación, y con el fin de demostrar la eficacia de esta propuesta, se asume que el MPDU es construido siguiendo la primera opción. Sin embargo, estos resultados también son válidos para la segunda opción.

Tabla 5.3: Acciones tomadas por la BS para cada uno de los tipos de retroalimentación recibida

Retroalimentación previa	**Retroalimentación recibida**	**Incremento de la longitud de la carga útil**		**Incremento la longitud de la palabra código FEC**	
		tramas imp	**tramas no imp**	**tramas imp**	**tramas no imp**
1	1	0	0	0	0
	2	-p1	-p2	+q1	+q1
	3	-p3	-p4	+q2	+q2
2	1	+p5	+p6	-q3	-q3
	2	0	0	0	0
	3	-p3	-p4	+q2	+q2
3	1	+p5	+p6	-q3	-q3
	2	+p1	+p2	-q1	-q1
	3	0	0	0	0

imp significa importante

5.7. Estimación de los Tamaños Óptimos de MPDU y de la Palabra Código FEC

En esta sección, se explica cómo se estiman los tamaños óptimos del MPDU y de la palabra código FEC.

5.7.1. Probabilidad de Recuperación de un Paquete

Siguiendo el estándar IEEE 802.16 [25], se usa RS (N, K) usando GF (2^m) donde (N-K) bytes de redundancia son adheridos a K bytes de información. Cada bloque puede ser correctamente recibido si no contiene más de T = (N-K)/2 bytes de error [55]. Cuando se usa el código RS, la probabilidad de que un bloque tenga un error, P_M, es:

$$P_M = \sum_{i=T+1}^{N} \binom{N}{i} p^i (1-p)^{N-i} \quad (1)$$

Donde *p* es la probabilidad de que un símbolo del canal sea recibido con error y se puede expresar como:

$$p = 1 - (1-b)^m \quad (2)$$

Donde *b* es el BER y los símbolos están compuestos de *m* bits cada uno. Así, la probabilidad de recuperación de una palabra código se puede expresar como:

$$PRP = (1 - P_M) \quad (3)$$

5.7.2. Estimación del Tamaño del MPDU Óptimo

Un MPDU incluye información de *overhead* como lo son el encabezado obligatorio y en algunos casos los encabezados opcionales [18]. Es deseable tener un tamaño de MPDU tal que el *overhead* sea mínimo. Sin embargo, la estimación del tamaño del MPDU depende de algunos factores. Por ejemplo, MPDUs más grandes significan menos *overhead*. Así, la tasa del *overhead* de la MAC con respecto a la carga útil puede ser expresada como:

$$O_{MAC} = \frac{header+subheader+CRC}{payload} \quad (4)$$

El tamaño del MPDU, *S*, puede ser expresado como:

$$S = user\_data + overhead \quad (5)$$

Mientras más grande el tamaño del MPDU mayor será la probabilidad de que un error ocurra durante la transmisión. Adicionalmente, tamaños más grandes de los MPDUs conllevan a que más datos sean retransmitidos cuando ocurre un error. Así el *overhead* de la capa MAC puede ser expresado como:

$$O_{retransmission} = \sum_{i=1}^{\infty} (1-(1-b)^L)^i \frac{L}{S} = \frac{(1-(1-b)^L)}{(1-b)^L} \frac{L}{S} \quad (6)$$

Donde *L* es el tamaño de la carga útil (datos del usuario). La suma de las ecuaciones (4) y (6) conlleva a obtener el *overhead* en presencia de errores de bit residuales lo cual puede ser expresado como:

$$O = \frac{1}{(1-b)^L}\frac{L}{S} \quad (7)$$

Y la eficiencia puede ser expresada como:

$$E = \frac{1}{O} = (1-b)^L\frac{S}{L} \quad (8)$$

Para cada valor del BER, hay un máximo valor de la eficiencia lo cual conlleva a un tamaño óptimo del MPDU. La Tabla 5.4 muestra los tamaños óptimos de MPDUs para diferentes valores del BER.

Tabla 5.4: Tamaños de MPDU óptimos

Tasa de error de bit (BER)	**Tamaño óptimo del MPDU (bytes)**	**Eficiencia**
10^{-6}	1231	0.9805
10^{-5}	393	0.9394
10^{-4}	129	0.8180
10^{-3}	45	0.5115
10^{-2}	20	0.0801

5.7.3. Estimación del Número de Bytes de Redundancia

El tamaño óptimo del MPDU puede ser afectado por el tamaño de la palabra código FEC como se indica en [37]. Así, la tasa de error del MPDU, P_{MPDU}, puede ser calculado usando la siguiente expresión:

$$P_{MPDU=}1-(1-P_M)^{\frac{L}{B}} \quad (9)$$

Donde P_M es la probabilidad de un error en un bloque estimada usando la ecuación (1) y B es el tamaño de la palabra código FEC. Ya que se está usando RS (N, K), B puede ser expresada como:

$$B = K + R \quad (10)$$

Donde R = N-K es el número de bytes de redundancia.

Usando la ecuación (9) y tomando en cuenta el *overhead* y la probabilidad de error en un bloque, se puede estimar la eficiencia como:

$$E_P = \frac{S}{L}\,(1 - P_{MPDU}) = \frac{S}{L}\,(1 - P_M)^{\frac{L}{(K+R)}} \quad (11)$$

Dado un tamaño óptimo de MPDU (mostrado en la Tabla 5.5) y la probabilidad de un error en un bloque, P_M, se puede obtener el número de bytes de redundancia, R, y el número de bytes de información, K, a partir de los cuales se puede obtener el mejor valor de la eficiencia, E_P. La Tabla 5.5 muestra el valor óptimo, R, para diversos valores de BER y los tamaños de MPDUs óptimos correspondientes. A coninuación se asume un valor fijo de K de 239 bytes.

Tabla 5.5: Tamaños óptimos de MPDU y número de bytes de redundancia

BER	**Tamaño Óptimo del MPDU (bytes)**	**R (bytes)**	**E**	**E_P**
10^{-6}	1231	10	0.9805	0.990252
10^{-5}	393	14	0.9394	0.969466
10^{-4}	129	16	0.8180	0.906977
10^{-3}	45	16	0.5115	0.733302
10^{-2}	20	16	0.0801	0.244356

5.8. Consideraciones de Implementación

En esta sección se detallan las consideraciones para el modelado del mecanismo propuesto.

5.8.1. Implementando las Acciones de la Estación Usando las Características de la Capa Física/MAC

En este apartado, se explica cómo las acciones descritas anteriormente podrían ser implementadas usando las funcionalidades existentes de las capas físicas y MAC. En las propuestas adaptativas, el tamaño de un MPDU podría ser incrementado empaquetando más MSDUs en un MPDU. De lo contrario, el tamaño de un MPDU podría disminuirse fragmentando un MSDU en múltiples MPDUs. El tamaño de la palabra código FEC se podría incrementar o disminuir sumando o sustrayendo, respectivamente, una cierta cantidad de bytes de redundancia a los bytes de la palabra código. En este libro, se asume que la capa física usada soporta tamaños de códigos variables (ver [25] para más detalles).

5.8.2. Implementación de la Información de Retroalimentación Usando la Capa Física

Una pregunta a contestar es cómo la MAC CPS distingue entre tramas de videos importantes y no tan importantes. Hay varias propuestas sobre la manera de manejar esta situación las cuales son descritas a continuación. Cada una requiere el uso de *cross-layer design* [66] para ser implementada, sin embargo los detalles de alguna implementación se encuentran fuera del alcance de este libro.

- Uso de las opciones de transporte de video: Han habido algunos trabajos que sugieren cómo transportar tramas MPEG sobre IP (por ejemplo, [35][45][59]). La opción de transporte puede incluir información sobre el tipo de trama de video que se transporta y la importancia de la trama. Por ejemplo, en el RFC 3640 [16], se define el formato RTP para el transporte de *streams* MPEG-4. Se establece el uso de un par de campos para distinguir entre flujos de video crucial y no crucial. La entidad CS se puede extender para que obtenga esta información de los paquetes RTP encapsulados en paquetes IP / UDP y se los comunique a la entidad MAC.

- El uso de un servidor: La estación base puede enviar todos los paquetes IP recibidos a un servidor. El servidor puede actuar como un usuario final donde los PDUs de video se mueven hasta la capa de aplicación que obtiene la información requerida. El resto los PDUs pueden desecharse ya que una copia se guarda en la estación base. La aplicación de video en el servidor procesa los PDUs con el fin de obtener la información y así clasificar las tramas de video. La información se encapsula y se envía de vuelta a la estación base.

- Uso de la aplicación de video en el lado del receptor: La aplicación de video (en el lado del usuario suscriptor) puede proporcionar información relacionada con el tipo de trama que se está recibiendo (por ejemplo, tramas I, B y P) a la entidad MAC subyacente. A continuación, la entidad MAC puede enviar información sobre las tramas de video que se están esperando a la estación base.

5.8.3. Modelo de Opinión para Estimar la Calidad del Video

Las aplicaciones multimedia generan no solo datos sino también imágenes, video y voz que demandan diferentes niveles de calidad de servicio. Por ejemplo, el retardo es bastante crucial para los servicios en tiempo real, sin embargo ellos son tolerantes a la pérdida. Estos parámetros de QoS son usualmente usados para los servicios garantizados. No obstante, ellos no capturan la perspectiva del usuario a nivel del servicio *fin a fin*. Por esto, los parámetros de calidad de la experiencia (QoE) tienen el objetivo de medir la calidad del servicio subjetiva asociada con la percepción humana.

Aunque se acepta que los modelos de QoE son los que mejor capturan la subjetividad asociada con la percepción y expectativas de los usuarios no existe un modelo de QoE universalmente aceptado. Un modelo de opinión tiene el propósito de estimar la calidad de la

comunicación completa como una función de algunos factores de calidad, tal como la calidad de la red, del terminal de entrada y de los datos. El E-model es un modelo de opinión aceptado ampliamente y es usado para estimar el QoE de los servicios del habla. De otra forma, pocos modelos han sido propuestos para la calidad del video. Uno de ellos es el E-model para aplicaciones de video telefonía propuestos por Yamagishi *et al.* [67]. El mismo ha sido estandarizado en la Recomendación G.1070 [30] y estima la calidad del video afectada por la distorsión de la codificación (tasa de codificación y tasa de la tramas) y la pérdida de paquetes.

El modelo propuesto en la Recomendación G.1070 [30] contiene tres funciones principales: una función de estimación de la calidad del video, una función de la calidad del habla y una función de integración de la calidad multimedia. Sus parámetros de entrada son la calidad del video y del habla y sus salidas la calidad multimedia (calidad del video solamente, calidad del habla solamente y calidad multimedia). En este libro, solo se usará la función de estimación de la calidad del video.

La calidad del video, *Vq* (*Mean Opinion Score, MOS video*) es estimada usando la siguiente ecuación:

$$Vq = 1 + Icoding\, e^{\left(-\frac{PplV}{DplV}\right)} \quad (12)$$

Esta representa la calidad del video afectada por la distorsión de la codificación y el factor de robustez de la pérdida de paquete, D_{PplV}, que representa la calidad del video debido a la pérdida de paquetes, P_{plv} (%). I_{coding} puede ser expresado como una función de varias variables, tales como, tasa de bit de video y la tasa de tramas de video y ciertos valores constantes elegidos de acuerdo a las características del video. Una descripción detallada de estos parámetros esta fuera del alcance de este libro y puede ser encontrada en [30].

5.9. Modelo de Simulación

Para validar la propuesta MAC de WiMax, se realizaron experimentos de simulación en el que se considera una red compuesta por varios suscriptores y una BS que ofrece los servicios de *video streaming*. Debido a que se está interesado en evaluar qué tan bien la propuesta de MAC trabaja en la interfaz de aire de WiMax, la tasa de pérdida de paquetes y el retardo en la parte cableada de la red se consideran insignificantes.

5.9.1. Modelo de Canal

El canal se modela según un modelo de Markov de tres estados. Cada estado se caracteriza por cierta probabilidad de error de bit (BER): un estado del canal malo tiene una BER de 0,01, un estado medio tiene una BER de 0,001, y un estado bueno tiene un BER de 0,0001. Se asume que el canal se mantiene en cada uno de los estados bueno, medio y malo por un período

medio de 50 ms, 100 ms, y 20 ms, respectivamente. Estableciendo las probabilidades de transición apropiadas entre los tres estados (ver Apéndice C), se logra modelar las condiciones del canal.

5.9.2. Parámetros de simulación

Los parámetros de simulación utilizados se pueden clasificar en: parámetros de red/MAC y parámetros relacionados con el video. Los primeros se han establecido de acuerdo con el estándar IEEE 802.16 [25] y se muestran en la Tabla 5.6. Para el enfoque ad hoc (ver Sección 5.6.1), se tomó un enfoque en el que los valores se aumentan o disminuyen de forma muy conservadora como en [11] para mostrar cómo los esquemas de adaptación basado en retroalimentación se comportan con respecto a las otras propuestas. La Tabla 5.7 muestra los parámetros utilizados para este esquema. La misma incluye la cantidad en que los tamaños de la carga útil y del código se incrementan o disminuyen de acuerdo a la explicación dada en la Sección 5.6.1.

Tabla 5.6: Parámetros de red

Parámetros de la simulación	**Valores**	**Parámetros de la simulación**	**Valores**
m (bits por símbolo)	8	Duración mínima de un mini slot (1 símbolo físico)	0.2 µs
Trama de 1 ms	5000 símbolos físicos	Duración máxima de un mini slot (128 símbolos físicos)	≈26 µs
Tasa del símbolo	20 Mbaudios	Ancho de banda	100 Mbps
Duración de la subtrama DL/UL	0.5 ms	RS (N,K) inicial	RS(255,239)

Tabla 5.7: Parámetros relacionados al mecanismo adaptativo ad hoc

Parámetro	**Valor (bytes)**
n1	1
n2	2
n3	50
m1	1
m2	2
m3	2
m4	1

Para el enfoque óptimo (Sección 5.6.2), los parámetros se han establecido de acuerdo a la explicación dada en la Sección 5.7 (ver Tabla 5.8). El tamaño de la carga útil de las tramas no importantes es 20% menor que el tamaño óptimo. Por simplicidad, se supone que las tramas de

video MPEG-4 se estructuran de la siguiente manera: IBBBPBBBPBBBPI (otras distribuciones de tramas de video IBP también son posibles y se pueden encontrar en [65]).

Tabla 5.8: Parámetros relacionados al mecanismo MAC adaptativo óptimo

Parámetro	Valor (bytes)
p1	45
p2	36
p3	20
p4	16
p5	129
p6	103
q1	16
q2	16
q3	16

En la simulación, se utilizan los coeficientes provisionales de la función de estimación de la calidad video especificada en la Recomendación G.1070 [30] para el códec MPEG-4 con un tipo de formato de video QVGA y un tamaño de pantalla de video de 4,2 pulgadas. La tasa de bits de video es de 1250 kbits / seg y la velocidad de tramas es de 30 fps (*frames per second*).

5.10. Resultados de la Simulación

Inicialmente, se asumen los siguientes valores: tamaños de la carga útil del MPDU óptimos de 20 y 129 bytes (ver la Tabla 5.5), tamaño de la carga útil del MPDU no óptimo de 100 bytes, encabezado del MPDU de 12 bytes, tamaño de bloque FEC de 239 bytes--establecido de acuerdo con el número máximo de bytes de información que se permiten en una palabra código FEC cuando se usa la especificación física de una sola portadora [25]--número de bytes de redundancia para el caso óptimo calculado según la ecuación (11), y número de bytes de redundancia para el caso no óptimo de 5 bytes. En las siguientes figuras, se utiliza la siguiente notación: tamaño de *carga útil :: tamaño del código*, para expresar el tamaño inicial de la carga útil del MPDU (en bytes) y el número inicial de bytes de redundancia en la palabra código FEC, respectivamente. De la Figura 5.4 a la Figura 5.6, se comparan el MOS para la calidad de video de las propuestas adaptativas ad hoc (Sección 5.6.1), óptima (Sección 5.6.2), y los sistemas no adaptativos. En el enfoque no adaptativo, ni la carga útil del MPDU ni el tamaño de la palabra de código FEC cambian durante la conexión. En las figuras se puede observar que cuando el MPDU inicial y los tamaños de las palabras código FEC se fijan a los valores óptimos 20::16 y 129::16, los valores de la calidad de video obtenidos utilizando cualquiera de las tres propuestas son similares. De lo contrario, cuando los valores iniciales de los tamaños se establecen en un valor no óptimo, 100::5, con el mecanismo adaptativo óptimo, hay una mejora con respecto a los otros

dos esquemas. Esto es esperado ya que el mecanismo adaptativo óptimo es capaz de ajustar el tamaño del MPDU a los valores óptimos según las condiciones cambiantes del canal.

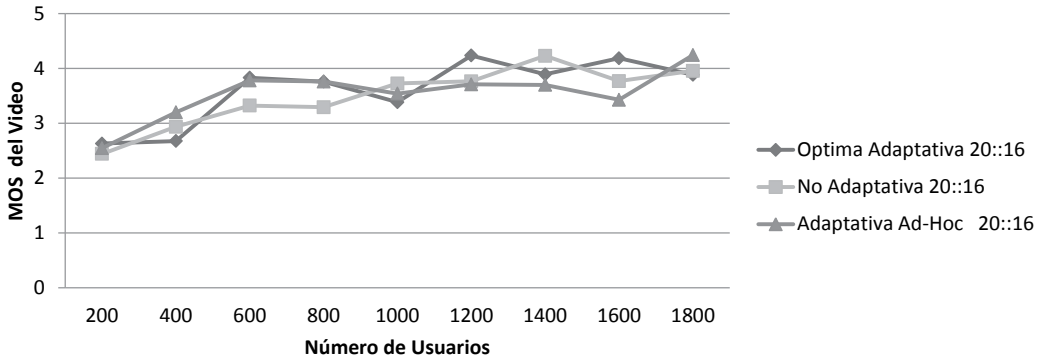


Figura 5.4: MOS del video versus el número de usuarios para tamaños iniciales 20::16.

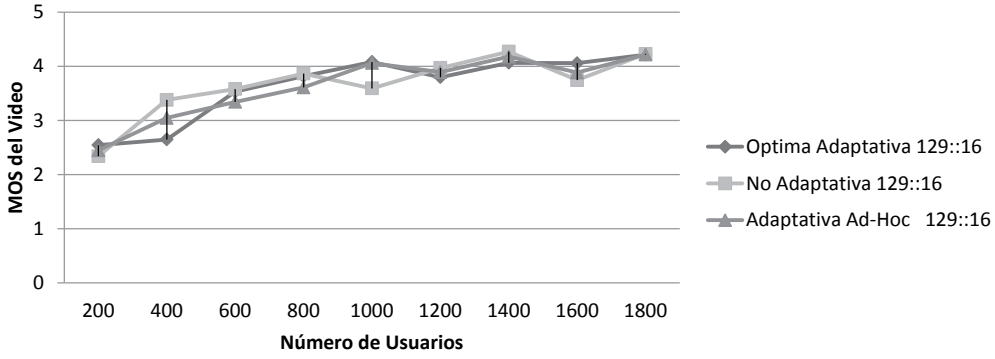


Figura 5.5: MOS del video versus el número de usuarios para tamaños iniciales de 129::16.

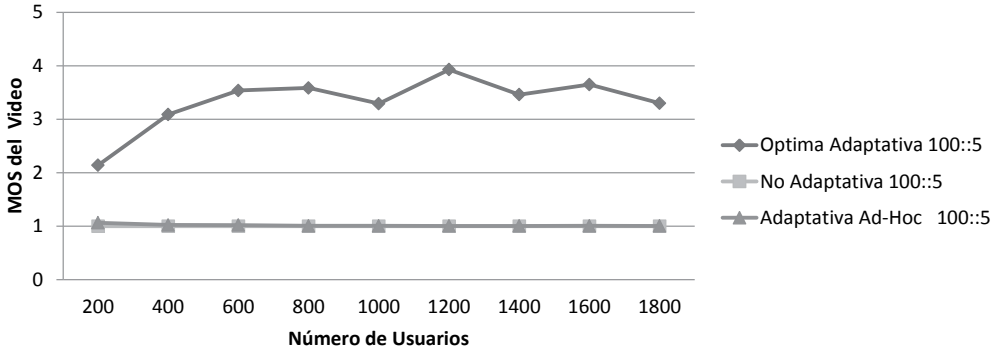


Figura 5.6: MOS del video versus el número de usuarios para tamaños iniciales de 100::5.

En la Figura 5.7 a la Figura 5.9, se compara la eficiencia promedio, la cual es el número de bytes de información (carga útil del MPDU) dividido entre el número total de bytes de transmisión. De estas figuras, se puede concluir que con el mecanismo adaptativo óptimo, se obtiene un rendimiento de más del 80% para todos los valores de los tamaños iniciales, mientras

que con los demás la eficiencia es menor al 80% para valores pequeños del tamaño inicial de la palabra código FEC y MPDU.

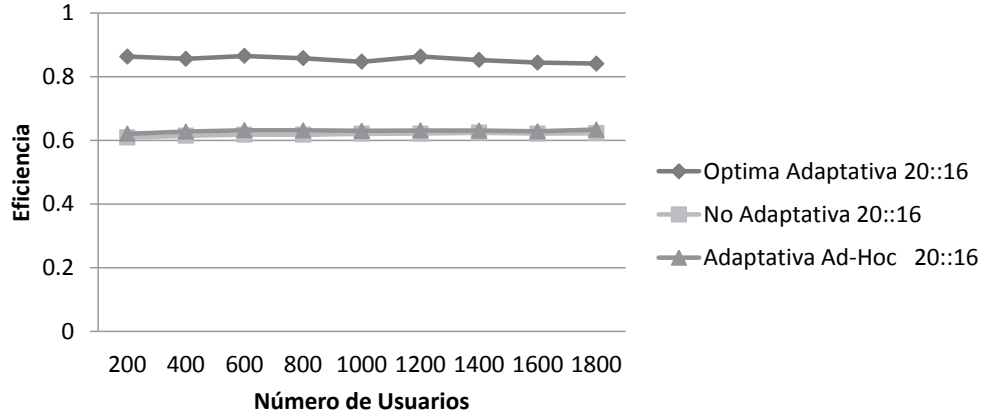


Figura 5.7: Eficiencia versus número de usuarios para tamaños iniciales de 20::16.

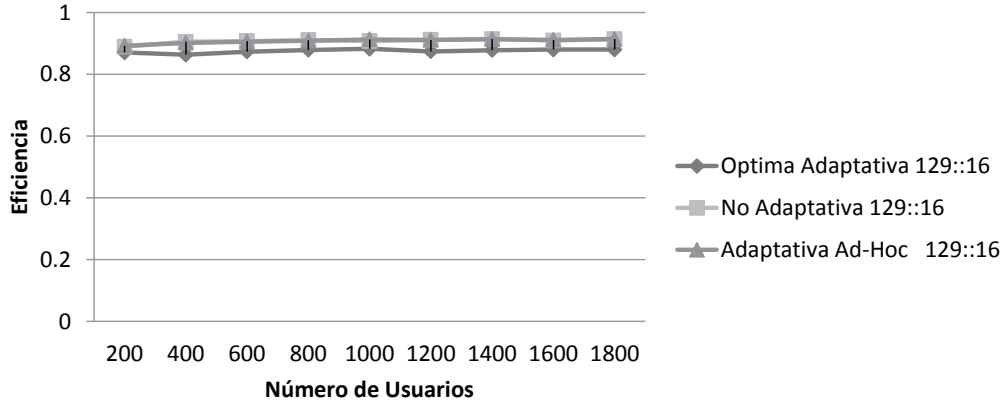


Figura 5.8: Eficiencia versus el número de usuarios para tamaños iniciales 129::16.

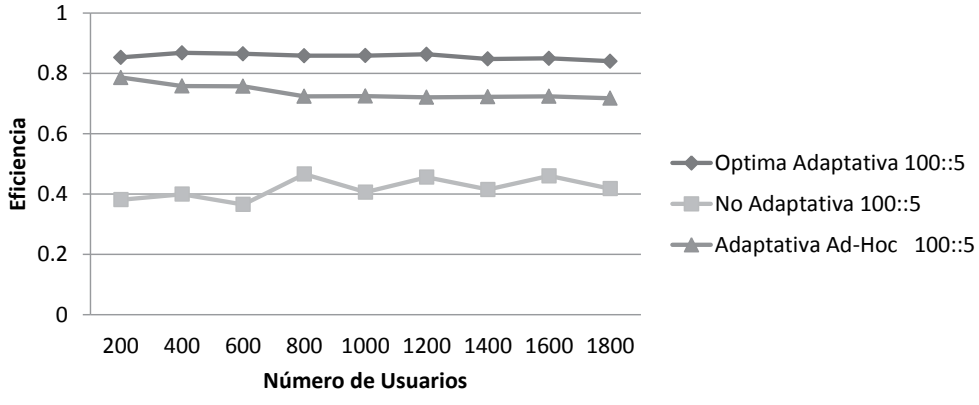


Figura 5.9: Eficiencia versus el número de usuarios para tamaños iniciales 100::5.

5.11. Resumen

WiMax incluye diversas características las cuales pueden ser explotadas para mejorar la transmisión de video. En esta investigación, se han estudiado dos mecanismos de construcción de MPDU adaptativas los cuales usan la técnica de FEC para una transmisión efectiva del video. En vez de usar las métricas tradicionales de calidad del video, se utiliza el modelo de la ITU-T que captura la subjetividad en la percepción del usuario. Los resultados muestran que con ambos mecanismos, adaptativos y no adaptativos, se obtienen buenos valores de la calidad del video mientras se maximiza la eficiencia cuando los valores iniciales de la carga útil del MPDU y del código FEC se colocan a los valores óptimos. Cuando se usan valores iniciales de los tamaños no óptimos, la aproximación óptima puede mejorar la calidad del video y maximizar la eficiencia promedio comparado con la otras aproximaciones, ya que es capaz de adaptar los tamaños de la carga útil del MPDU y código FEC a los valores óptimos de acuerdo a las condiciones cambiantes del canal.

6. Diseño y Análisis de una MAC de WiMax Adaptativa Basada en Retroalimentación Usando las Redes de Petri Coloreadas

6.1 Introducción

En el Capítulo 5 se introdujo una propuesta para construir de forma adaptativa un MPDU basándose en información de retroalimentación proveniente del receptor acerca de las condiciones del canal y la cual está basada en la propuesta de Chatterjee et al. [11]. Dichas propuestas presentan algunos inconvenientes. En WiMax, un MPDU es transportado en uno o más palabras códigos FEC. Algunas alternativas de la capa fisca no soportan tamaños de códigos variables o imponen algunas limitaciones en los cambios en los tamaños de los bloques. Aunque la tecnología pudiera soportar tamaños de bloques variables, un MPDU es transmitido en bloques los cuales se ajustan a los tamaños de las palabras códigos. Así, la información de retroalimentación descrita en [11] no puede ser obtenida analizando un MPDU sino chequeando cada uno de las palabras códigos que transportan un MPDU. En este capítulo se propone un mecanismo adaptativo basado en retroalimentación tal que los cambios el tamaño de los MPDU son ejecutados de acuerdo a un tamaño predeterminado de una unidad de bloque o bloque ARQ (ver Sección 5.4).

El diseño de los mecanismos de adaptación, tales como la construcción de MPDUs adaptativa basada en retroalimentación involucra varios niveles de ejecución del protocolo. Siendo un sistema complejo, la necesidad de una especificación clara y sin ambigüedades es muy importante. Por lo tanto, se utiliza una técnica formal, tal como las CPNs para diseñar el mecanismo propuesto. El modelo CPN se analiza para un conjunto de propiedades generales, tales como terminación correcta, y un conjunto de propiedades específicas del mecanismo que se definen en este documento. Algunas de las propiedades se comprueban mediante la consulta al grafo de estado, y las otras se verifican por medio del uso de lógica temporal.

Los mecanismos de construcción de MPDUs, como el propuesto en este libro, han sido ampliamente estudiados. La mayoría de los estudios utilizan la simulación para obtener resultados de rendimiento [11],[37],[54]. Los modelos de simulación desarrollados para esos fines, a veces no se validan para comprobar si cumplen con la especificación del sistema en

estudio. Por otra parte, las CPNs se pueden usar para validar y verificar la funcionalidad de un sistema nuevo, como el que se propone en este libro. Después de la verificación formal, el modelo puede ser usado para estudiar el rendimiento del sistema.

6.2 Construcción de MPDUs Adaptativa Basada en Retroalimentación

En la Sección 5.6 se describe el enfoque de construcción de MPDUs adaptativa basada en retroalimentación propuesto por [11]. En este enfoque no se toma en cuenta una de las características de la capa MAC de WiMax, como lo son las particiones lógicas de un MPDU en bloques ARQ, que además son la unidades de retransmisión de la información. Así en este capítulo se propone un nuevo enfoque adaptado a este funcionamiento de la capa MAC de WiMax. En este mecanismo el tamaño de la carga útil de un MPDU es aumentado o disminuido en unidades de *ni* bloques (ver Tabla 6.1). El tamaño del bloque ARQ [25], *l*, es fijo y puede ser inicialmente establecido al tamaño de K (número de bytes de información). El número de bytes en cada palabra código FEC se aumenta o disminuye de acuerdo a los valores de la variable *mj* (j=1..6).

Tabla 6.1: Acciones tomadas por la BS para cada uno de los tipos de retroalimentación recibida en el esquema basado en bloques ARQ

Tipo de retroalimentación	**Aumentar Carga Útil**		**Disminuir Carga Útil**		**Aumentar FEC**		**Disminuir FEC**	
	tramas imp	**tramas no imp**	**tramas imp**	**tramas no imp**	**tramas imp**	**tramas no imp**	**tramas imp**	**tramas no imp**
1	n1	n1	0	0	0	0	0	m1
2	0	0	0	0	m2	0	0	0
3	0	0	n2	n2	m3	m4	0	0

ni es expresado en número de bloques de *l* bytes, *l* es fijo; *mi* es expresado en bytes. *imp significa importante*

6.3 Modelo CPN

Se usan las CPNs para diseñar el mecanismo de construcción de MPDUs adaptativa basada en retroalimentación y el uso de los bloques ARQ utilizando CPN Tools versión 3.2.0 (ver Sección 3.5).

6.3.1 Asunciones y Alcance

El modelo de CPN se ha desarrollado de acuerdo a la descripción dada en las secciones 5.3, 5.6 y 6.2. Adicionalmente, aunque algunos MPDUs dañados pueden llegar al receptor, no se considera las retransmisiones en esta versión del modelo, ya que inicialmente se está interesado en estudiar el comportamiento del mecanismo basado en retroalimentación. Se asume que la conexión entre un BS y un SS ha sido establecida por los procedimientos provistos por el estándar [25], y cuyo modelado está fuera del alcance de este libro. La topología de red consta de

un SS y un BS, el cual ofrece servicios de video *streaming*. Esto permite considerar el comportamiento de los nodos (BS y SS) de la MAC de WiMax.

6.4 Jerarquía del Modelo

Dada la complejidad del sistema, se usan los constructores jerárquicos de la CPNs conocidos como transiciones de sustitución para hacer más entendible el modelo. El modelo se inicia con un diagrama que proporciona una visión general del sistema. El módulo del nivel superior se muestra en la Figura 6.1 y describe la topología de la red. Este módulo incluye una transición de sustitución para los protocolos de WiMax (IEEE 802.16) implementados en la BS y en el SS. Estos a su vez son definidos en más detalles en sus propios módulos, es decir el módulo BS y el módulo SS.

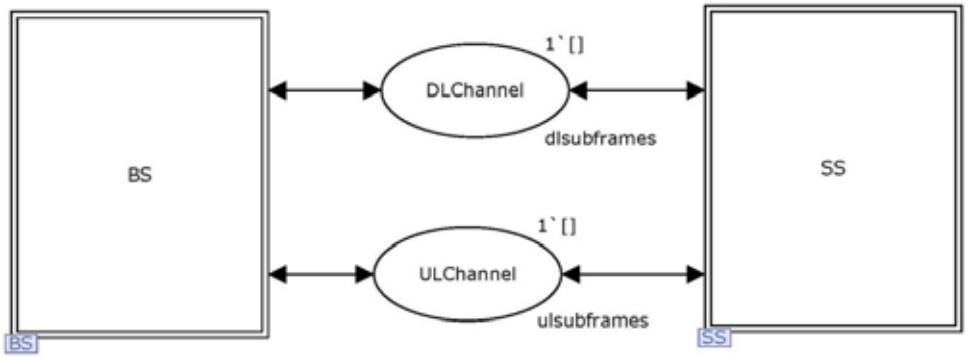


Figura 6.1: Módulo del nivel superior del modelo CPN.

En la Figura 6.1, las plazas DLChannel y ULChannel son del tipo *ulsubframes* y *dlsubframes*, respectivamente (ver la declaración global en la Sección 6.5) y representan la ráfaga física viajando hacia la BS (ULChannel) o desde la BS (DLChannel). Por otra parte se puede observar que existen dos transiciones de sustitución, BS y SS, en las cuales se modela con más detalle las funcionalidades asociadas a la construcción de MPDUs y la transmisión de los mismos en ráfagas físicas.

6.5 Declaración Global

La Figura 6.2 muestra los conjuntos de colores y variables en la declaración global. El tipo *sdu* es un producto de *cid*, *dtype* y *datapayload* y representa la información relacionada con el video que se desea transmitir. El tipo *cid* es un entero representando el identificador de la conexión, el tipo *dtype* es un enumerado representando el nivel de importancia de una trama de video (es decir, importante, I, o no importante, NI) y el tipo *datapayload* representa la longitud de la carga útil que contiene parte o todo el video. El tipo *sdus* representa una lista de *sdu*, es decir, una lista de tramas de video. El tipo *macpdu* es un producto de algunos campos de control pertenecientes a los encabezados genérico y de fragmentación de un MPDU (ver [25]), campos de control necesarios para construir el MPDU siguiendo el enfoque propuesto y la carga útil del

MPDU. Los campos de control incluyen el identificador de la conexión, *cid*, la longitud del MPDU, *mlen*, el tipo de fragmento (no fragmentado, último fragmento, primer fragmento, fragmento intermedio [25]), *fc*, el número de secuencia del fragmento, *fsn*, la importancia del video, *dtype*, la longitud del encabezado, *mheader* y longitud de la carga útil, *mpayload*. Los tipos son del conjunto de colores entero, INT. Una lista de *macpdu* es representada por el tipo *macpdus*. El tipo *macsdu* es el producto de un MPDU, *macpdu,* información necesaria intercambiada entre la capa CS y la MAC CPS, *macidu*. El tipo *macsdus* es una lista de *macsdu*.

Los conjuntos de colores que modelan la información de control o datos en la capa física se describen a continuación. Los tipos *tcsici*, *tcspayload*, and *tcssdu* representan la información de la interfaz entre la MAC CS y la sub capa física TCS, el PDU de la sub capa TCS y el SDU de la sub capa TCS, respectivamente. El conjunto de colores *ulsubframe* es del tipo producto del identificador de la conexión, c*i*d, y la información de retroalimentación, *fb*, que es enviada por el receptor después de analizar el MPDU. El conjunto de colores *tcssdus* es del tipo lista y representa los *tcssdu* que vienen de la sub capa TCS a la sub capa física. El conjunto de colores *dlsubframe* es del tipo producto del MPDU, *macpdu*, del tamaño de la carga útil que viaja en el PDU de la capa física, *fecpayload*, y del tamaño de la cantidad de bits de redundancia que viajan en el PDU de la capa física, *fec*. Los conjunto de colores *ulsubframes* y *dlsubframes* son del tipo lista y representan las ráfagas físicas que van del SS a la BS y viceversa, respectivamente.

Los siguientes conjuntos de colores representan la información que debe ser almacenada como parte del perfil del suscriptor para la conexión en curso y los cuales servirán para modelar la construcción del MPDU adaptativa. El color set *bppar* es del tipo producto del número de bytes de información, *K*, el número de bytes de redundancia, *R*, y el nivel de importancia del video (I, NI), *dtype*. El conjunto de colores *par* representa los parámetros adaptativos y es del tipo producto del identificador de la conexión, *cid*, el número de bloques ARQ, *nblocks*, las variables mostradas en la Tabla 6.1, el número de secuencia del MPDU, *fsn* y el nivel de importancia, *dtype*.

El conjunto de colores *feedback* representa la información de retroalimentación (es decir 1,2,3) retornada por el receptor para una conexión determinada en función del análisis del MPDU realizado como se explicó en la Sección 5.6. Este conjunto de colores es del tipo producto del identificador de la conexión, *cid* y un entero (INT) especificando la información de retroalimentación.

Los demás conjuntos de colores son usados en el proceso de fragmentación, empaquetamiento y posterior ensamblado de los MPDUs. Las variables, *var*, por otra parte son de los tipos definidos anteriormente. También se han definidos unos valores constantes, *val*.

```
colset UNIT = unit;
colset INT = int;
colset BOOL = bool;
colset STRING = string;
colset fecpayload = int;
colset fec = int;
colset fb = int;
colset mlen = int;
colset cid = int;
colset dtype = with I|NI;
colset datapayload = int;
colset sdu = product cid*dtype*datapayload;
colset sdus = list sdu;
colset mheader = int;
colset mpayload =  int;
colset fc = int with 0..3;
colset fsn = int with 0..100;
colset macidu = int;
colset macpdu = product
cid*mlen*fc*fsn*dtype*mheader*mpayload;
colset macsdu = product macidu*macpdu;
colset macpdus = list macpdu;
colset macsdus = list macsdu;
colset tcsici = int;
colset tcspayload = int;
colset tcssdu = product tcsici*macpdu*tcspayload;
colset tcssdus = list tcssdu;
colset ulsubframe = product cid*fb;
colset dlsubframe = product macpdu*fecpayload *
fec;
colset ulsubframes = list ulsubframe;
colset dlsubframes = list dlsubframe;
colset ety = with E|A|C;
colset temppayload = int;
colset temptcspdu = product
ety*mlen*temppayload;
colset K = int with 1..239;
colset R = int with 1..100;
colset bppar = product cid*K*R*dtype;
colset feedback = product cid*INT;
colset qsize = int;
colset n1 = int;
colset n2 = int;
colset nblocks =int;
colset m1= int;
colset m2 = int;
colset m3 = int;
colset m4 = int;
colset par = product
cid*nblocks*n1*n2*m1*m2*m3*m4*fsn*dtype;
colset seq = product cid*fsn*datapayload;
var p,p1,f,fbk,fbk2,cid,id2,nb: INT;
var ln,ln1,hd1,hd2,hd3,hd4,hd6,hd7,hdn: INT;
var r,r1: R;
var k,k1: K;
var hd5: dtype;
var e: ety;
var mpdu: macpdu;
var mpduq: macpdus;
var msdu: macsdu;
var msduq, msduq2: macsdus;
var sdu1: sdu;
var dlf: dlsubframe;
var sduq: sdus;
var tsdu: tcssdu;
var tsdus,tsduq:  tcssdus;
var ulf: ulsubframe;
var dlfq: dlsubframes;
var ulfq: ulsubframes;
var pr,pr1: par;
var feedbk: feedback;
var sn: fsn;
val MAXMQUEUE = 10;
val MAXR = 16;
val MINR = 10;
```

Figura 6.2: Declaración global del modelo CPN.

6.6 Módulo BS

La Figura 6.3 muestra el módulo BS. El mismo incluye tres (3) transiciones de sustitución, BSMACCPS, BSPHYTCS y BSPHYLAYER, las cuales modelan los procedimientos necesarios para implementar la construcción del MPDU basado en información de retroalimentación y la cual es ejecutada en las subcapas MAC CPS, TCS y física, respectivamente. Estas transiciones son expandidas en sus respectivos módulos. La plaza Videoframes representa los SDUs de video mientras que las plazas BSINMPDUS y BSOUTMPDUS representan los MSDUs intercambiados entre las entidades de las subcapas

MAC CPS y TCS. Las plazas BSINTCSPDUS y BSOUTTCSPDUS representan los PDUs intercambiados entre las entidades TCS de la BS y SS. Las otras plazas son puertos y fueron descritas en la Sección 6.4.

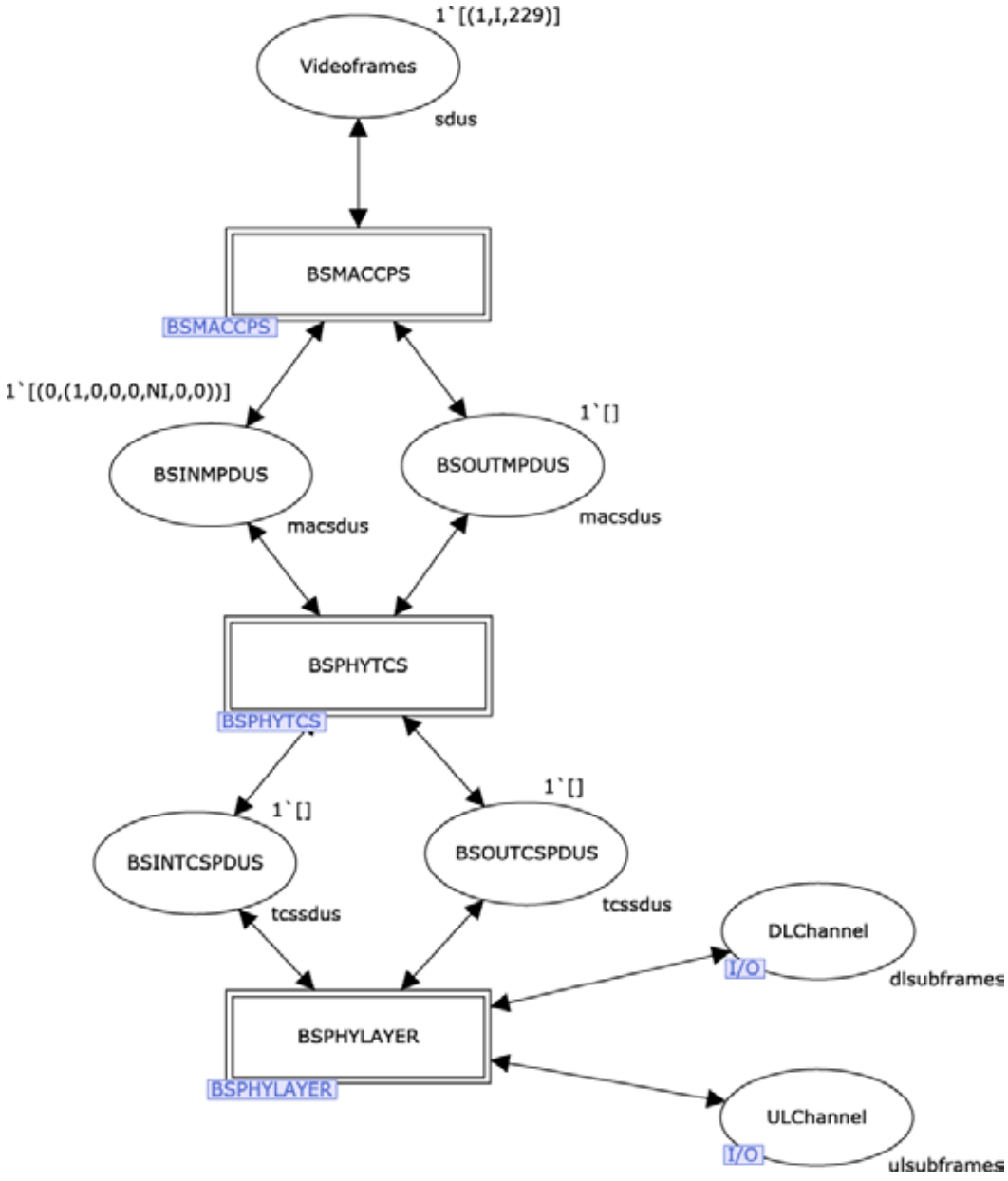


Figura 6.3: Módulo BS.

6.7 Módulo SS

La Figura 6.4 muestra el módulo SS, el cual incluye tres (3) transiciones de sustitución SSMACCPS, SSPHYTCS y SSPHYLAYER. Estas transiciones modelan el mecanismo de construcción del MPDU en el lado del suscriptor. Similarmente al módulo BS, las transiciones de sustitución se expanden en sus propios módulos, donde se modelan con más detalle las

funcionalidades del procedimiento propuesto. La plaza RcvVideoframes representa los SDUs de video que se reciben en el suscriptor, mientras que la plaza SSMPDUS representa los MSDUs intercambiados entre las entidades de las subcapas MAC CPS y TCS. Las plazas SSINTCSPDUS y SSOUTTCSPDUS representan los PDUs intercambiados entre las entidades TCS de la BS y del SS. Las otras plazas son puertos y fueron descritas en la Sección 6.4.

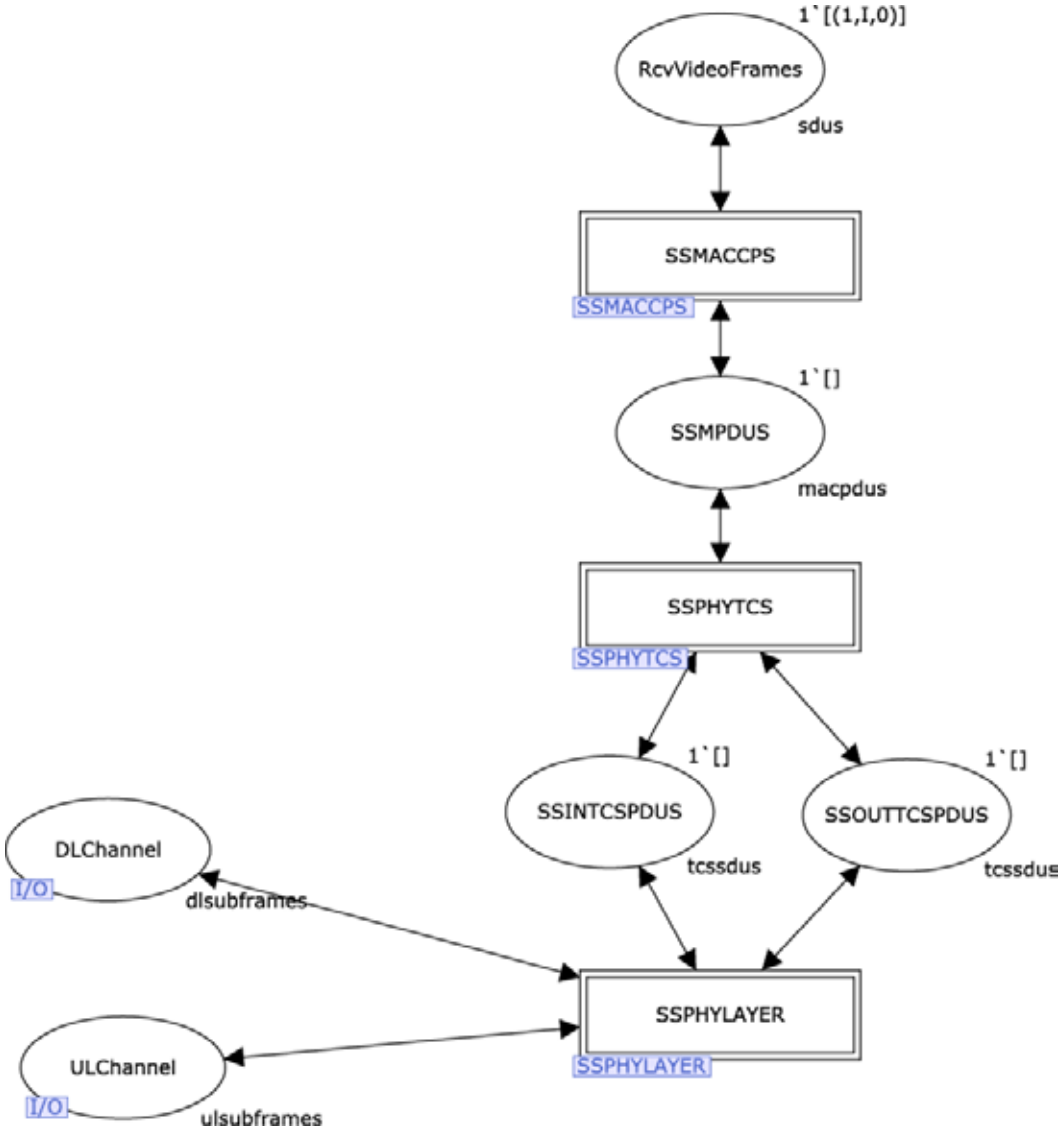


Figura 6.4: Módulo SS.

6.8 Módulo BSMACCPS

La Figura 6.5 muestra el módulo BSMACCPS que incluye cinco (5) transiciones. Las transiciones FEEDBACK1, FEEDBACK2 y FEEDBACK3 modelan las acciones tomadas por la BS basada en la información de retroalimentación enviada por el SS como se explicó en la Sección 6.2. INITIALFEEDBACK modela las acciones ejecutadas por el BS cuando ninguna información de retroalimentación del receptor ha llegado aún. Para facilitar el análisis del

modelo, cuando la cola de SDUs está vacía, la información de retroalimentación es removida de la cola representada por la plaza BSINMPDUS. Esta acción es modelada por la transición CLEANFEEDBACK. La plaza BurstProfile representa las características de la ráfaga física, mientras que la plaza AdaptivePar representa los parámetros adaptativos los cuales cambian de acuerdo a la información de retroalimentación enviada por el receptor. Las otras plazas ya han sido descritas.

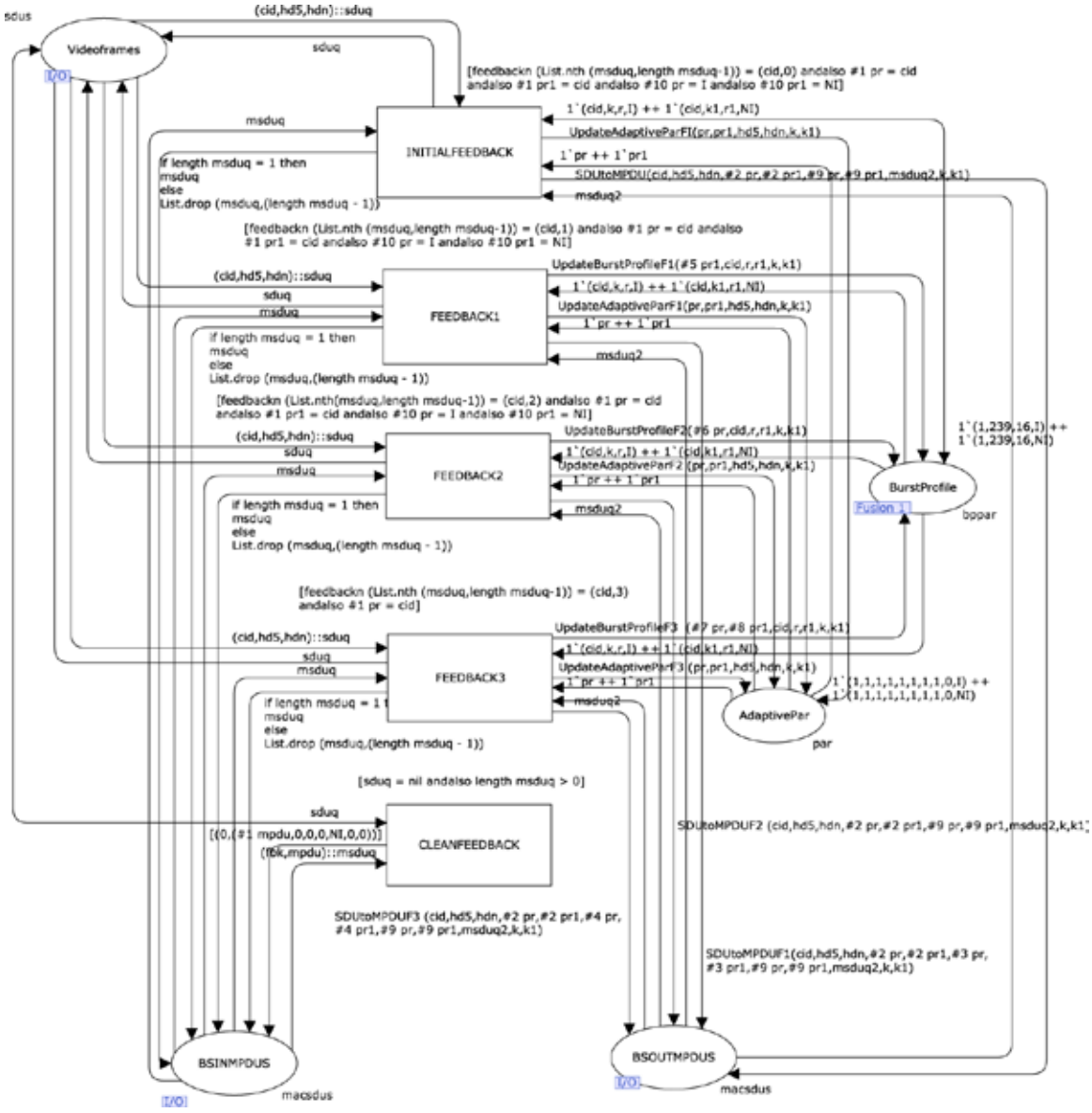


Figura 6.5: Módulo BSMACCPS.

Las funciones que aparecen en las inscripciones de los arcos o en las guardas asociadas a las transiciones se describen a continuación y se muestran en el Apéndice D:

1) SDUtoMPDU: encapsula un SDU de video en uno o más MPDUs siguiendo la aproximación propuesta descrita anteriormente para la construcción de los MPDUs adaptativa.

2) UpdateAdaptiveParF1, UpdateAdaptiveParF2, UpdateAdaptiveParF3: UpadateAdaptiveFi (i=1..3) actualiza el valor de los parámetros adaptativos mostrados en la Tabla 6.1 de acuerdo a la información de retroalimentación, *i*, recibida recientemente.

3) feedbackn: devuelve el identificador de la conexión, *cid*, y la información de feedback de un determinado SDU de la subcapa TCS.

4) UpdateBurstProfileF1, UpdateBurstProfileF2, UpdateBurstProfileF3: la información del perfil de la ráfaga es aquella que se usa cuando se construye la ráfaga física. En la propuesta de construcción de MPDUs los parámetros relacionados a la cantidad de bytes de información, K, y cantidad de bytes de redundancia que viajan en una palabra código (FEC) cambian de acuerdo a lo explicado en la Sección 5.6. Esta información es actualizada por la función UpdateBurstProfileFi (i= 1..3) de acuerdo a la información de retroalimentación, *i*, recibida.

Las demás funciones son propias del lenguaje ML y están descritas en [15].

6.9 Módulo BSPHYTCS

El módulo BSPHYTCS se muestra en la Figura 6.6. Este módulo modela las acciones tomadas por la subcapa TCS necesarias para la implementación de la propuesta descrita anteriormente. La transición MPDUTOPHY modela las acciones llevadas a cabo para encapsular un MPDU en un PDU de la TCS de acuerdo a lo descrito en la Sección 5.5, mientras que la transición PHYTOMPDU realiza las acciones para desencapsular los MPDUs entrantes. Adicionalmente, la función BuildTCSPDU (ver Apéndice D) devuelve los TCS PDUs en los cuales se encapsula el MPDU de la capa MAC CPS.

El módulo BSPHYLAYER, mostrado en la Figura 6.7, incluye dos (2) transiciones que modelan las acciones realizadas por la capa física para generar las palabras códigos y enviarlas en la ráfaga física. La transición FECENCODING y FECDECODING son complementarias, siendo la primera encargada de encapsular los TCS PDUs en las correspondientes palabras códigos que viajan corriente abajo (*downstream*) (ver Sección 5.5), mientras que la segunda modela la acciones para des encapsular los TCS PDUs entrantes (viajando en dirección corriente arriba o *upstream*).

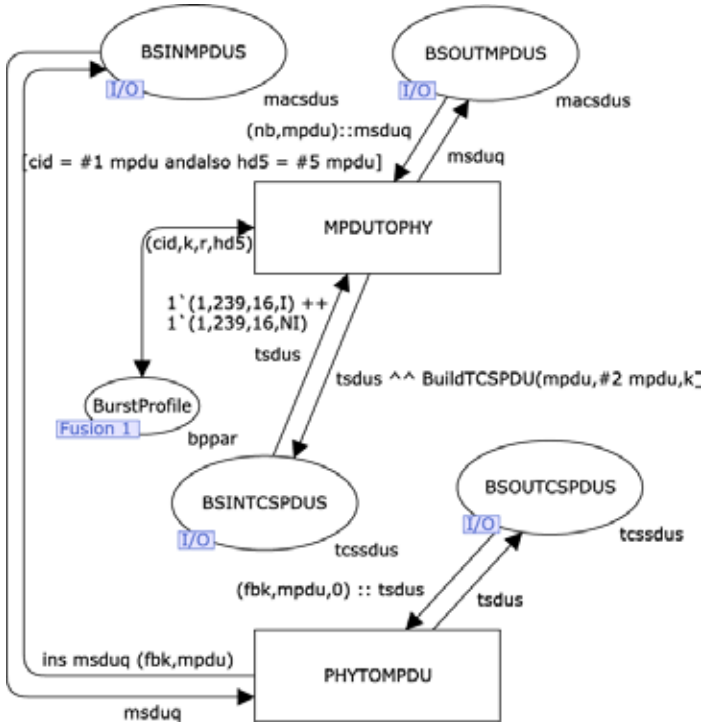


Figura 6.6: Módulo BSPHYTCS.

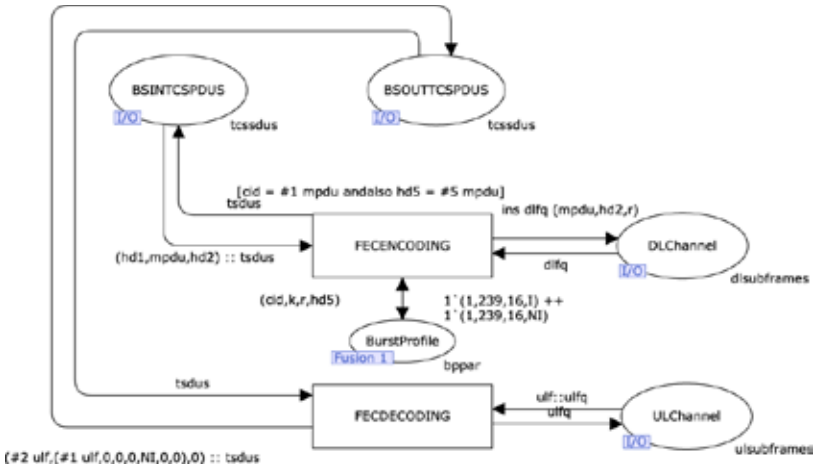


Figura 6.7: Módulo BSPHYLAYER.

6.10 Módulo SSMACCPS

Los procedimientos relacionados a la propuesta de construcción de MPDUs adaptativa realizadas en el receptor son más sencillos que los realizados en el transmisor. La Figura 6.8 muestra el módulo SSMACCPS que incluye una (1) transición que modela las acciones realizadas por la capa MAC CPS en el receptor. La transición RcvARQ modela las acciones de resemblado del video antes de enviarlo a la sub capa inmediatamente superior. La plaza nseq

mantiene información de secuenciamiento necesaria para poder realizar el procedimiento de resemblado del SDU.

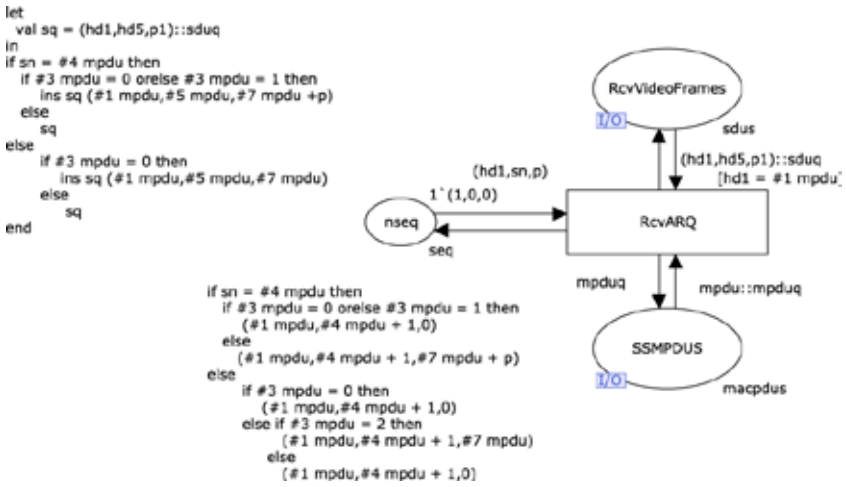


Figura 6.8: Módulo SSMACCPS.

6.11 Módulo SSPHYTCS

En la propuesta de construcción de MPDUs adaptativa, cada palabra código FEC es analizada independientemente. Desde que una palabra código puede contener parte de un MPDU, si una palabra tiene un error y el mismo no se puede corregir dicha palabra debe ser descartada, así como también todas las palabras que contenga información del MPDU transportado. De lo contrario, si ninguna de las palabra códigos que transportan un MPDU está dañada, o alguna palabra código está dañada pero el error no es detectado, o alguna palabra código está dañada pero puede ser corregida, el MPDU es aceptado y la información correspondiente es pasada a la capa superior para su procesamiento. En la Figura 6.9 se muestra el módulo SSPHYTCS que incluye cuatro (4) transiciones que modelan las acciones antes descritas. La transición blockdiscarded modela las acciones realizadas en el caso de que una de las palabras códigos que transportan un MPDU tenga un error no recuperable. Si esto sucede se deben descartar las restantes palabras códigos. Las demás transiciones modelan las acciones ejecutadas cuando todas las palabras códigos son recibidas correctamente o tienen errores que pueden ser recuperados.

6.12 Módulo SSPHY

En la Figura 6.10 se muestra el módulo SSPHYLAYER que modela las acciones realizadas por la capa física que son relevantes para este estudio. La misma incluye cuatro (4) transiciones. Las transiciones DataCorrect, DataCorrectable y DataUncorrectable modelan las siguientes condiciones en las cuales se pueden recibir los datos provenientes del BS: los datos pueden estar correctos, los datos pueden estar dañados pero ser corregibles o los datos pueden estar dañados y no pueden ser corregidos, respectivamente. La transición Encoding modela la acción ejecutada por la capa física para enviar una ráfaga física al BS.

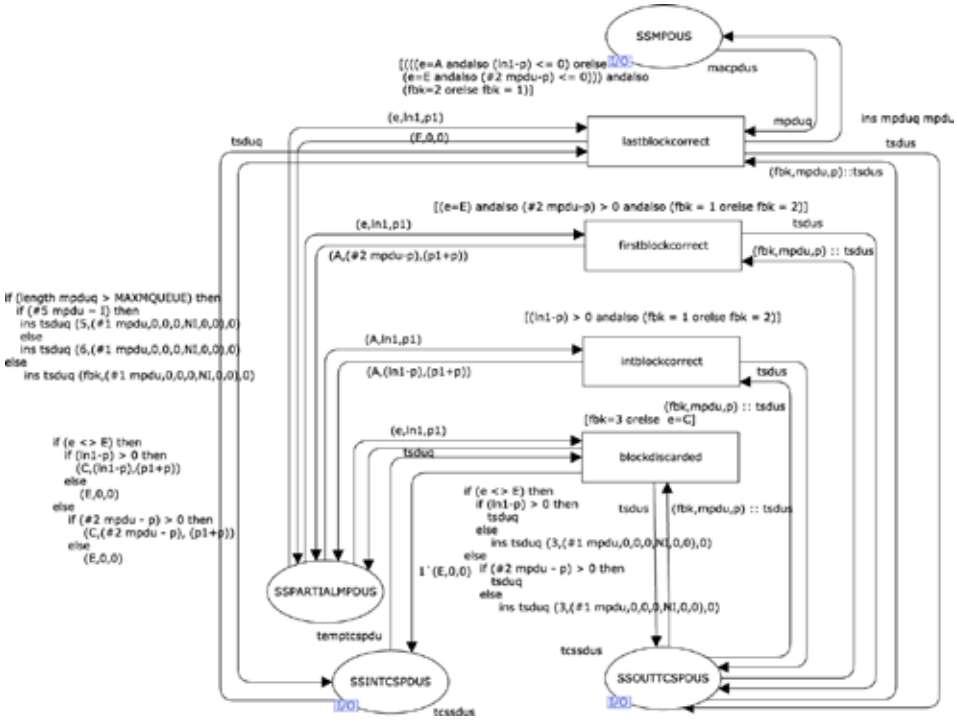


Figura 6.9: Módulo SSPHYTCS.

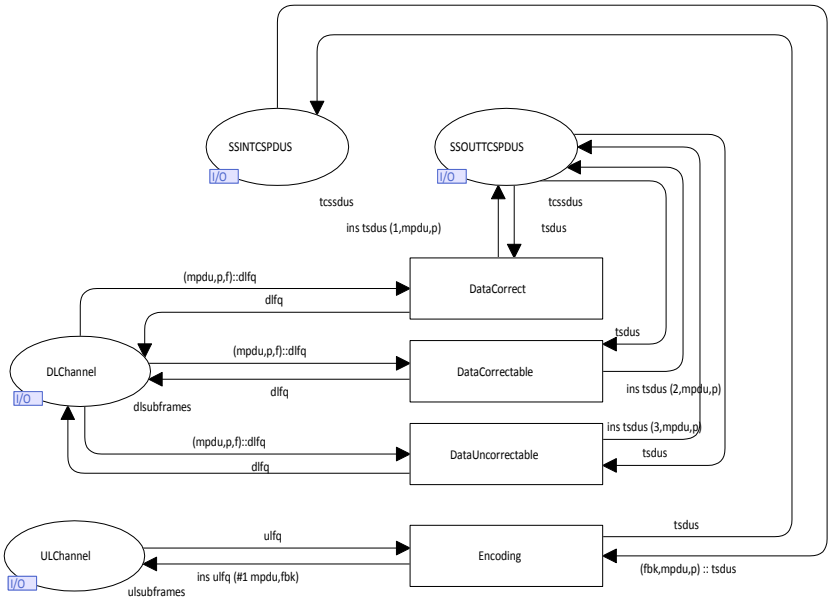


Figura 6.10: Módulo SSPHYLAYER.

6.13 Propiedades Deseadas de la MAC de WiMax Adaptativa Basada en Retroalimentación

Se desea verificar la MAC de WiMax Adaptativa basada en Retroalimentación contra un conjunto de propiedades deseables. Ellas son: no abrazos mortales y propiedades relacionadas el ensamblado de MPDUs y SDUs de acuerdo a la aproximación propuesta. Estas propiedades se formalizan en las siguientes secciones.

Inicialmente, se especifica la notación CPN [32] necesaria para definir las propiedades formalmente:

- B(t) denota el conjunto de asociaciones de las variables asociadas con la transición *t*.
- M(p) denota el marcado de la plaza *p*.
- DM denota el conjunto de todos los marcados muertos.
- $M \xrightarrow{(t,b)} M'$ denota que el marcado *M'* es directamente alcanzable desde el marcado *M* por la ocurrencia del elemento de asociación (t,b), $b \in B(t)$.
- SDULength (l) = sdul es la longitud del último SDU recibido por el receptor.
- SeqNum (s) = fsn es el número de secuencia del SDU o fragmento de un SDU.
- SLength (b) = p es el valor de la variable *hdn* en la asociación *hdn*, la cual representa la longitud del SDU.
- SeqN(mp) = fn es el número de secuencia asignado al primer segmento SDU.
- MPDUGeti(b) = m es el valor de la variable *mpdu* en la asociación *b*. *mpdu* representa un MPDU e i=1..3.

6.13.1 Propiedad de Terminación

La MAC de WiMax Adaptativa basada en Retroalimentación debe finalizar en un estado donde todas las colas de PDUs y de SDUs estén vacías.

Definición 6: El modelo CPN de la MAC de WiMax Adaptativa basada en Retroalimentación termina correctamente sii:

$\forall M \in DM$, M(BSINMPDUS)=M(BSINTCSPDUS)=M(BSOUTCSPDUS)= M(BSOUTMPDUS) = M(Videoframes) = M(SSINTCSPDUS) = M(SSMPDUS) = M(SSOUTTCSPDUS) = M(DLChannel) = M(ULChannel) = Ø) □

6.13.2 Propiedad de Re Ensamblado de un SDU

El mecanismo de la MAC de WiMax Adaptativa basada en Retroalimentación establece que un SDU de video podría ser fragmentado por la entidad MAC CPS para adaptarlo al tamaño del MPDU. Así, todos los SDUs de video deben ser re ensamblados correctamente por el receptor.

Sea $\mathbb{T}_{fb}$ = {INITIALFEEDBACK,FFEDBACK1,FEEDBACK2,FEEDBACK3} el conjunto de transiciones de retroalimentación y $t_{fb} \in \mathbb{T}_{fb}$, $b \in B(t_{fb})$ and $sdul = SLength\ (b)$. Sea $\mathbb{MPDU}$ = {mpdus| $M \xrightarrow{(tfb,b)} M'$, $mpdus = M'(BSOUTMPDUS)-M(BSOUTMPDUS)$} los MPDUs en los cuales un SDU es fragmentado y $fsn = SeqN\ (mpdus)$.

Definición 7: El modelo CPN de la MAC de WiMax Adaptativa basada en Retroalimentación satisface la propiedad de re ensamblado sii:

$\forall$ $t_{fb} \in \mathbb{T}_{fb}$, $\neg$ POS (SDUNeitherReassembledCorrectlyNorDiscarded) donde SDUNeitherReassembledCorrectlyNorDiscarded(t) = ((t=RcvARQ) $\wedge$ M $\xrightarrow{(RcvARQ,bra)}$ M' => SDULength (M'(RcvVideoFrames))= sdul $\wedge$ SeqNum(M'(nseq))=fsn)) $\vee$ ((t=blockdiscarded) $\wedge$ M'' $\xrightarrow{(blockdiscarded,bbd)}$ M''' => $\exists$ mpdux $\in$ $\mathbb{MPDU}$: MPDUGet1 (b_{bd}) = mpdux)) □

6.13.3 Propiedad de Reesamblado de un MPDU

La capa física de WiMax establece que un MPDU es fragmentado en bloques que se ajusten al tamaño de la palabra código. Así, todos los MPDUs deben ser reensamblados correctamente por el receptor.

Sea $\mathbb{T}_{m2p}$ = {MPDUTOPHY} el conjunto de transiciones de mpdu a físicas y $t_{m2p} \in \mathbb{T}_{m2p}$, $b \in B(t_{m2p})$ y $m = MPDUGet2\ (b)$.

Definición 8: El modelo CPN de la MAC de WiMax Adaptativa basada en Retroalimentación satisface la propiedad de Reensamblado de MPDU sii

$\forall$ $t_{m2p} \in \mathbb{T}_{m2p}$, $\neg$ POS (MPDUNeitherReassembledCorrectlyNorDiscarded) donde MPDUNeitherReassembledCorrectlyNorDiscarded (t) = ((t=lastblockcorrect) $\wedge$ M $\xrightarrow{(lastblockcorrect,blb)}$ M' => MPDUGet3 (b_{lb})= m)) $\vee$ ((t=blockdiscarded) $\wedge$ M'' $\xrightarrow{(blockdiscarded,bbd)}$ M''' => MPDUGet1 (b_{bd})= m)) □

6.14 Análisis y Verificación

CPN Tools se utiliza para simular y analizar el modelo CPN. El modelo es analizado generando el OG y su correspondiente SCC. Las propiedades son chequeadas implementando las

definiciones dadas en la Sección 6.13 usando funciones de *query* escritas en ML y usando la librería ASK-CTL (ver Sección 3.4.1). Para ilustrar la aproximación de verificación seguida en esta sección se describe la función de *query* escrita en ML para la implementación de la propiedad de Reensamblado de un MPDU. La propiedad de terminación es verificada examinando los nodos del OG. La propiedad de Reensamblado de un SDU es chequeada usando la librería de chequeo del modelo incluido en CPN Tools.

6.14.1 Inicialización

El modelo CPN es inicializado como se indica a continuación. La plaza Videoframes es inicializada para tener SDUs de diferentes tamaños (229 y 329 bytes) y niveles de importancia (I,NI) como se muestra en la Tabla 6.2. Se puede notar que un tamaño de MPDU de 229 bytes no requiere ser fragmentado en la capa física, ya que se ajusta al tamaño de una palabra código, es decir, 239 bytes. Mientras que un MPDU de 329 bytes necesita ser fragmentado en la capa física. La plaza AdaptivePar (ver Sección 6.5) ha sido inicializada para modelar el valor de los incrementos y decrementos en el tamaño de la carga útil y la palabra código de acuerdo a la explicación dada en la Sección 6.2 (ver Tabla 6.2). Inicialmente se toma una aproximación donde los valores de los incrementos/decrementos son asignados de una forma conservativa ya que solo se está interesado en conocer la forma en que se comportan el mecanismo adaptativo basado en retroalimentación. La plaza BurstProfile contiene dos marcas que representan el tamaño por defecto de una trama física (K=239 y (N-K) = 16), es decir, 1`(1,239,16,I) ++ 1`(1,239,16,NI). La plaza BSINMPDUS contiene una marca representando una lista de MSDUs la cual está vacía. El resto de las plazas no contiene ninguna marca.

6.15 Estadísticas del OG y SCC

Se generaron una serie de OGs para los diferentes marcados iniciales mostrados en la Tabla 6.2. El tamaño del OG y su SCC para cada marcado inicial se muestra en Tabla 6.3. CPN Tools se corrió en una PC con procesador Intel Core 2 de 2.13 GHz con 3 GB de RAM corriendo Windows 7. Como se puede observar en la tabla el tamaño del SCC es igual al tamaño del OG por lo cual no hay ciclos en el sistema modelado como es esperado.

6.16 Chequeo de la Propiedad de Reesamblado de MPDU

Esta sección describe como dos de las propiedades (Reesamblado de un MPDU y Terminación) son chequeadas usando funciones ML (*ML queries*) y la librería ASK-CTL. La implementación de la propiedad Re Ensamblado de un SDU se encuentra en el Apéndice D. La propiedad de Re Ensamblado de un MPDU es chequeada usando la siguiente función ML:

```
1.  fun WasMPDUReassembledCorrectly (a) = let
2.  fun takempdu (Bind.BSPHYTCS'MPDUTOPHY
3.  (1,{cid=_,hd5=_,k=_,mpdu=v1,msduq=_,nb=_,r=_,tsdus=_})) = v1;
4.  val mpdu1 = takempdu (ArcToBE a);
5.  fun NeitherReassembledCorrectlyNorDiscarded (a1) = let
```

```
6.  fun ItisTheMPDU (Bind.SSPHYTCS'lastblockcorrect (1,{e=_,fbk=_,ln1=_,mpdu=
  v1,mpduq=_ ,p=_,p1=_,tsduq=_,tsdus=_})) = v1 =mpdu1;
7.  fun ItisTheDiscardedMPDU (Bind.SSPHYTCS'blockdiscarded
  (1,{e=_,fbk=_,ln1=_,mpdu= v1,p=_,p1=_,tsduq=_,tsdus=_})) = v1 =mpdu1;
8.  val result = (if ArcToTI a1 = TI.SSPHYTCS'lastblockcorrect 1 then
9.  ItisTheMPDU (ArcToBE a1)
10. else
11. if ArcToTI a1 = TI.SSPHYTCS'blockdiscarded 1 then
12. ItisTheDiscardedMPDU (ArcToBE a1)
13. else
14. false);
15.       in
16.       not result
17.       end;
18.       val ASKCTLformula =
  POS(AF("MPDUReasembledCorrectly",NeitherReassembledCorrectlyNorDiscarded));
19.       in
20.       eval_arc ASKCTLformula a
21.       end;
22.       fun MPDUReassembly (a,b) = a andalso b;
23.       fun MPDUReassemblyProperty () = SearchAllArcs (fn a => ArcToTI a =
            TI.BSPHYTCS'MPDUTOPHY 1, fn a => WasMPDUReassembledCorrectly a,
            true,MPDUReassembly);
```

Tabla 6.2: Marcado inicial de la plazas Videoframes y AdaptivePar

# Marcado Inicial	Videoframes	AdaptivePar
1	1`[(1,I,229)]	1`(1,1,1,1,1,1,1,1,0,I) ++ 1`(1,1,1,1,1,1,1,1,0,NI)
2	1`[(1,NI,229)]	
3	1`[(1,NI,229),(1,NI,229)]	
4	1`[(1,I,229),(1,I,229)]	
5	1`[(1,I,229),(1,NI,229)]	
6	1`[(1,NI,229),(1,I,229)]	
7	1`[(1,I,329)]	
8	1`[(1,NI,329)]	
9	1`[(1,NI,329),(1,NI,329)]	
10	1`[(1,I,329),(1,I,329)]	
11	1`[(1,NI,329),(1,I,329)]	
12	1`[(1,I,329)]	1`(1,2,1,1,1,1,1,1,0,I)++ 1`(1,2,1,1,1,1,1,1,0,NI)++
13	1`[(1,I,329), (1,NI,329)]	
14	1`[(1,I,329),(1,I,329)]	
15	1`[(1,NI,329),(1,NI,329)]	
16	1`[(1,NI,329),(1,I,329)]	

Tabla 6.3: Tamaño del OG y SCC para los diferentes marcados iniciales mostrados en la Tabla 6.2

# Marcado inicial (ver Tabla 6.2)	# Nodos OG	# Arcos OG	# Nodos SCC	# Arcos SCC	Tiempo de generación OG (tiempo de generación SCC) en horas:minutos:segundos
1	30	38	30	38	00:00:00 (00:00:0)
2	30	38	30	38	00:00:00 (00:00:0)
3	609	1243	609	1243	00:00:00 (00:00:0)
4	609	1243	609	1243	00:00:00 (00:00:0)
5	609	1243	609	1243	00:00:00 (00:00:0)
6	609	1243	609	1243	00:00:00 (00:00:0)
7	461	997	461	997	00:00:00 (00:00:0)
8	461	997	461	997	00:00:00 (00:00:0)
9	68627	222915	68627	222915	00:16:49 (00:00:06)
10	68119	222881	68119	222881	00:17:09 (00:00:07)
11	68650	223985	68650	223985	00:22:48 (00:00:05)
12	53	80	53	80	00:00:00 (00:00:0)
13	1984	4802	1984	4802	00:00:00 (00:00:0)
14	1984	4802	1984	4802	00:00:01 (00:00:0)
15	1984	4802	1984	4802	00:00:01 (00:00:0)
16	1984	4802	1984	4802	00:00:01 (00:00:0)

La función **MPDUReassemblyProperty** (línea 23) retorna verdad si para todas las ocurrencias de las transiciones MPDUTOPHY, un MPDU es reensamblado correctamente o descartado. La evaluación de la función **WasMPDUReassembledCorrectly** (líneas 1-21) invoca a la fórmula ASK-CTL, POS (línea 18), y retorna verdad si no es posible, desde la transición MPDUTOPHY donde se está ahora, alcanzar un transición donde el MPDU es reensamblado (línea 8) o descartado (línea 11). Finalmente, la función **MPDUReassembly** (línea 22) combina el resultado actual retornado por la función **MPDUReassemblyProperty** con el resultado previo usando el operador AND.

La propiedad Terminación es chequeada usando la siguiente función ML:

```
1. fun DeadLocks() = PredNodes(ListDeadMarkings(), fn n => let
2. in
3. if (*Mark.BS'BSINMPDUS 1 n = 1`[]  andalso *)
4.    Mark.BS'BSINMPDUS 1 n = 1`[(0,(1,0,0,0,NI,0,0))] andalso
5.    Mark.BS'BSINTCSPDUS 1 n = 1`[]  andalso
6.    Mark.BS'BSOUTCSPDUS 1 n = 1`[]  andalso
7.    Mark.BS'BSOUTMPDUS 1 n = 1`[]  andalso
8.    Mark.BS'Videoframes 1 n = 1`[]  andalso
9.    Mark.SS'SSINTCSPDUS 1 n = 1`[]  andalso
10.           Mark.SS'SSMPDUS 1 n =1`[]  andalso
11.           Mark.SS'SSOUTTCSPDUS 1 n = 1`[]  andalso
12.           Mark.WimaxNetwork'DLChannel 1 n = 1`[]  andalso
13.           Mark.WimaxNetwork'ULChannel 1 n = 1`[]
14.        then false
15.        else true end,NoLimit);
```

La función **DeadLocks ()** chequea que en todos los marcados muertos (línea 1), todas las plazas representando el canal de comunicación de bajada y de subida y las plazas representando la colas de comunicación entre capas estén vacías (líneas 3 – 13). Cabe destacar que para fines del modelado, la plaza BSINMPDUS está vacía cuando tiene el marcado inicial 1`[(0,(1,0,0,0,NI,0,0))] (línea 4). La función **DeadLocks ()** retorna todos aquellos marcados muertos donde lo dicho anteriormente no se cumple.

6.17 Propiedades de la Construcción de MPDUs Adaptativa Basada en Retroalimentación

El modelo CPN es verificado en función de las propiedades de la Construcción de MPDUs Adaptativa basada en Retroalimentación corriendo los diferentes códigos ML (ver código para generar resultados de forma automática mostrados en el Apéndice D). Para cada uno de los marcados iniciales mostrados en la Tabla 6.2, se obtienen resultados como los mostrados en la Figura 6.11. Ellos indican que el modelo termina correctamente (no existen abrazos mortales), los MPDUs y SDUs son reensamblados correctamente, ya que las funciones retornan un valor de verdad (*True*). Los resultados muestran que el modelo CPN de la Construcción de MPDUs Adaptativa basada en Retroalimentación se comporta como se esperaba bajo las asunciones realizadas.

```
========== Start Umproper Termination ==========
DeadLocks= [ ]
========== End Umproper Termination ==========
========== Start MPDU REASSEMBLY ==========
MPDUReassembledCorrectly = [True]
========== End MPDU REASSEMBLY  ==========
========== Start SDU REASSEMBLY ==========
SDUReassembledCorrectly = [True]
```

Figura 6.11: Resultados del análisis.

6.18 Resumen

En este capítulo se ha propuesto un mecanismo para la construcción de MPDUs el cual, a diferencia de otros mecanismos similares, cumple con lo establecido en la especificación IEEE 802.16. Se han usado las CPNs para diseñar y verificar el esquema propuesto basado en un número de asunciones. Se utilizó una topología de red representativa (un BS y un SS) y diversos casos de estudio para verificar que la MAC adaptativa operará correctamente bajo ciertas condiciones. En vista de que el mecanismo propuesto hace uso de la fragmentación de MPDUs fue necesario verificar que los MPDUs y SDUs son reesamblados correctamente por el receptor. Así, el modelo es analizado basado en un conjunto de tres propiedades definidas y formalizadas por primera vez en este libro. El análisis del modelo muestra que el mecanismo de construcción de MPDUs adaptativa basada en retroalimentación trabaja como se espera bajo las asunciones realizadas.

El modelo de referencia contenido en el estándar IEEE 802.16 incluye un protocolo de la capa MAC que implementa Requerimiento de Repetición Automática (*Automatic Repeat Request*, ARQ) para tratar las pérdidas de MPDUs ocasionales. La retransmisión ARQ está basada en bloques ARQ, así el transmisor determina si algunos bloques ARQ necesitan ser retransmitidos. Se está interesado en investigar como el mecanismo propuesto en este capítulo se comporta cuando se usan los mecanismos de retransmisión basados en ARQ. Por lo cual trabajos futuros extenderán el modelo CPN para incluir retransmisiones ARQ. También, las CPNs han sido usadas para realizar análisis de rendimiento basadas en simulación, en un futuro se podría investigar el rendimiento de la red cuando se usa el mecanismo propuesto.

Referencias

[1] Billington J., Diaz M. and Rozenberg G. (eds), *Application of Petri Nets to Communication Networks*. Advances in Petri Nets, LNCS, Vol. 1605, 1999.

[2] Billington J., M.C. Wilbur-Ham and Bearman M.T. *Automated Protocol Verification. Protocol Specification, Testing, and Verification*. In M. Diaz (ed.) Protocol Specification, Testing, and Verification. Elsevier Science Publisher, 1986, pp 59-70.

[3] Billington J., Gallasch G. and Han B. A *Colored Petri Net Approach to Protocol Verification*. Lectures on Concurrency and Petri Nets. 2004, Volume 3098/2004, pp 49-70.

[4] Black U. *OSI: A Model for Computer Communications Standards*. Prentice-Hall, 1991.

[5] Blanco M and Villapol M.E. *Analyzing the Procedure for AMP Enabled Operation in Bluetooth 3.0 Using Colored Petri Nets*. Proceedings of the IEEE Global Information Infrastructure and Networking Symposium (GIIS), December 17-19, 2012, Choroní, Venezuela.

[6] Bluetooth SIG. *History of the Bluetooth Special Interest Group*. http://www.bluetooth.com/Pages/History-of-Bluetooth.aspx, consultado Agosto 2012.

[7] Bluetooth SIG. *Specification of the Bluetooth System*. Version 4.0, December 2009.

[8] Bluetooth SIG. *Specification of Bluetooth System*. Covered Core Packing version 2.1, July 2007.

[9] Bluetooth SIG. *Specification: Adopted Documents*. http://www.bluetooth.org/Technical/Specifications/adopted.htm, [Consulta: 2012, agosto 24].

[10] Braden R. et al. *Resource Reservation Protocol (RSVP) -- Version 1: Functional Specification*. RFC 2205, IETF, September, 1997.

[11] Chatterjee M., Sengupta S. and Ganguly S. *Feedback-based Real-Time Streaming over WiMax*, IEEE Wireless Communications Magazine, Vol. 14, No. 1, Feb. 2007, pp 64-71.

[12] Cheng A., Christensen S. and Mortensen K. H. *Model Checking Coloured Petri Nets Exploiting Strongly Connected Components*. Proc. International Workshop on Discrete Event Systems, Institution of Electrical Engineering, University of Edinburgh, UK, August 1996, pp 169-177.

[13] Clarke E.M., Emerson E.A., and Sistla A.P. *Automatic Verification of Finite State Concurrent System Using Temporal Logic*. ACM Transactions on Programming Language and Systems, 8(2), 1986, pp 244-263.

[14] Computer History Museum. *Internet History*. [Página Web en Línea]. Disponible: http://www.computerhistory.org/internet_history/index.html [Consulta: 2013, marzo 24].

[15] CPN Tools. [Página Web en Línea]. Disponible: http://cpntools.org/start [Consulta: 2014, noviembre 26].

[16] Cruz D. *Evaluación de los Mecanismos y Protocolos de Seguridad Desarrollados para Redes Bluetooth*. Trabajo Especial de Grado, Escuela de Computación, UCV, Octubre 2005.

[17] DSL Forum. *Triple-play Quality of Experience (QoE) Requirements*. Technical Report TR-126, December, 2006.

[18] Edklun C. et al. *WirelessMAN: Inside the IEEE 802.16 Standard for Wireless Metropolitan Networks*. IEEE press. 2006.

[19] Feldmann S, Hartmann T, Kyamakya K. *Modeling and Evaluation of Scatternets Performance by using Petri Nets*. In Proceedings of the International Conference on Wireless Networks, ICWN '03, June 23 - 26, 2003, Las Vegas, Nevada, USA. CSREA Press 2003.

[20] Grupo AIS de la Universidad de Eindhoven [Página Web en Línea]. Disponible: http://www.win.tue.nl/ais/doku.php?id=tools:start&purge=true [Consulta: 2013, julio 19].

[21] Gotzhein R. *Temporal logic and applications a tutorial*. Computer Networks and ISDN Systems, 24, 1992, pp 203-218.

[22] Han B. and Billington J. *Termination Properties of TCP's Connection Management Procedures*. 26th International Conference, ICATPN 2005, Miami, USA, June 20-25, 2005. Proceedings. LNCS 3536.

[23] Holzmann, Gerard. *Design and Validation of Computer Protocols*. Prentice Hall. 1991.

[24] Hoymann C. *Analysis and performance evaluation of the OFDM-based metropolitan area network IEEE 802.16*. Computer Networks 49 (2005), Elsevier, pp 341–363.

[25] IEEE Standard for Local and Metropolitan Area Networks, *Part 16: Air Interface for Fixed Broadband Wireless Access Systems*, IEEE Standard 802.16-2004, October 2004.

[26] International Organization for Standardization. [Página Web en Línea]. Disponible: http://www.iso.org/iso/home.html [Consulta: 2013, marzo 24].

[27] ISO/IEC 15909-1:2004 Software and system engineering -- High-level Petri nets -- Part 1: Concepts, definitions and graphical notation.

[28] ITU-T Recommendation X.200. *Information Technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*. 07/94.

[29] ITU-T Recommendation X.210. *Information Technology – Open Systems Interconnection – Conventions for Definitions of OSI Services*. 11/93.

[30] ITU-T. Recommendation G.1070. *Opinion model for video-telephony applications*, April 2007.

[31] Jensen K. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. Springer-Verlag, 2nd edition, Vol. 2, April, 1997.

[32] Jensen, K. and Kristensen, L. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. Springer-Verlag, Berlin, 2009.

[33] Jensen K., Kristensen L.M., and Wells L. *Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems*. International Journal on Software Tools for Technology Transfer, Springer, 2007, Vol. 9, Issue 3, May 2007, pp 213-254.

[34] Kelly D. *Automata and Formal Languages: An Introduction*. Prentice Hall, 1995.

[35] Kikuchi Y., Nomura T., Fukunaga S., Matsui Y., and Kimata H. *RTP Payload Format for MPEG-4 Audio/Visual Streams*, IETF RFC 3016, Nov. 2000.

[36] Kleinrock L. *A Brief History of the Internet*. October, 2004. [Página Web en Línea]. Disponible: http://www.oid.ucla.edu/Webcast/Inet35 [Consulta: 2013, marzo 24].

[37] Martikainen H., Sayenko A., Alanen O., Tykhomyrov V. *Optimal MAC PDU Size in IEEE 802.16*. IEEE IT-NEWS 2008, pp 66 – 71.

[38] Morales A. and Villapol M.E. *Towards Formal Specification of the Service in the IEEE 802.16 MAC Layer for Connection Management*. Proceedings of the 9th WSEAS International Conference on COMPUTATIONAL INTELLIGENCE, MAN-MACHINE SYSTEMS and CYBERNETICS (CIMMACS '10), December 14-16, 2010, Mérida, Venezuela.

[39] Millar B.A, Bisdikian C. *Bluetooth Revealed: The Insider's Guide to an Open Specification for Global Wireless Communications*. Prentice Hall, September 25, 2000.

[40] Miroslav Popović. *Communication protocol engineering*. Boca Raton, FL : CRC/Taylor & Francis, 2006.

[41] Moran J., Villapol M.E. y Perez D. *Efectos de interferencia sobre redes basadas en el estándar IEEE 802.11 en el espectro de 2.4 GHz*. Proceedings of Tenth LACCEI Latin American and Caribbean Conference (LACCEI'2012), July 23-27, 2012, Panama City, Panama.

[42] Murata T., *Petri Nets: Properties, Analysis and Applications*. Proceedings of The IEEE, Vol. 77, No. 4, April, 1989, pp 541-580.

[43] Network Simulator – NS2. [Página Web en Línea]. Disponible: http://www.isi.edu/nsnam/ns/ [Consulta: 2013, julio 19].

[44] OPNET. [Página Web en Línea]. Disponible: http://www.opnet.com/ [Consulta: 2013, julio 19].

[45] Pereira F. and Ebrahimi T., *The MPEG-4 Book*. Prentice Hall, 2002.

[46] Perahia E. *IEEE 802.11n Development: History, Process, and Technology. IEEE Communications Magazine*, July 2008, pp 48-55.

[47] Peterson J. *Petri Net Theory and Modeling of Systems*. Prentice-Hall, 1998.

[48] Petri Nets World. [Página Web en Línea]. Disponible: http://www.informatik.uni-hamburg.de/TGI/PetriNets/ [Consulta: 2013, julio 19].

[49] Proceedings of IFIP WG 6.1 International Conference on Formal Description Techniques for distributed Systems and Communications Protocols (FORTE), 1997-2012.

[50] Ratzer, A.V, Wells, L, at el. *CPN Tools for Editing, Simulating, and Analysing Coloured Petri Net*. Lecture Notes in Computer Science, Volume 2679 / 2003, pp. 450 – 462.

[51] Reisig W. *Petri Nets: An Introduction*. Springer-Verlag. 1985.

[52] Reisig W. and Rozenberg G. (Eds.). *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, Springer-Verlag, Vol.1, LNCS 1491, 1998.

[53] Salonidis T., Bhagwat P. and Tassiulas L. *Proximity awareness and fast connection establishment in Bluetooth*. 2000 First Annual Workshop on Mobile and Ad Hoc Networking and Computing, 2000. MobiHOC.

[54] Sengupta S., Chatterjee M., and Ganguly S. *Improving Quality of VoIP Streams over WiMax*. IEEE Transaction on Computers, Vol 57, No 2, IEEE Press, 2008, pp. 145-156.

[55] Sklar B., *Digital Communications*, 2nd edition, Prentice Hall.

[56] Socolofsky T. and Kale C. *A TCP/IP Tutorial*. Request for Comments: 1180, IETF, January 1991.

[57] So-In C., Jain R., and Tamimi A. *Scheduling in IEEE 802.16e Mobile WiMAX Networks: Key Issues and a Survey*. IEEE Journal on Selected Areas in Communications, vol. 27, no. 2, February 2009, pp 156-171.

[58] Steven G. and Billington J. *Analysing the WAP Class 2 Wireless Transaction Protocol Using Coloured Petri Nets*. Lecture Notes in Computer Science, Vol. 1825: 21st International Conference on Application and Theory of Petri Nets (ICATPN 2000), Aarhus, Denmark, June 2000, pages 207-226. Springer-Verlag, 2000.

[59] Van der Meer J., Mackie D., Swaminathan V., Singer D., and Gentric P., *RTP Payload Format for Transport of MPEG-4 Elementary Streams*, RFC 3640, IETF, November 2003.

[60] Venkataraman M., Sengupta S., Chatterjee M. and Neogi R., *Towards a Video QoE Definition in Converged Networks*. In Proceedings of ICDT'07, 2007.

[61] Villapol M.E. and Billington. *Analysing Properties of the Resource Reservation Protocol*. Proceedings of 24th International Conference on Application and Theory of Petri Nets, June 23-27, 2003, Eindhoven, The Netherlands, LNCS 2679, pp 377-396.

[62] Villapol M.E. *Modelling and Analysis of the Resource Reservation Protocol Using Coloured Petri Nets*, PhD Thesis, University of South Australia, December 2003.

[63] Villapol M.E. *Modelado y Análisis Inicial del Establecimiento de una Conexión Bluetooth Usando las Redes de Petri Coloreadas*. en Proceedings of the Thirty-Second Latin American Computing Conference, CLEI 2006, Santiago de Chile, Chile, 21-25 de Agosto 2006.

[64] Villapol M.E. *Modeling of the Connection Establishment between Two Bluetooth Devices Using Colored Petri Nets (In Spanish)*, Revista AVANCES EN SISTEMAS E INFORMÁTICA, Vol 5, Number 3, Diciembre del 2008.

[65] Wan P., Du Z., Wu W. *A Simple and Efficient MPEG-4 Video Traffic Model for Wireless Network Performance Evaluation*. IEEE Wireless Communications and Networking Conference, 2004, pp 1738-1742.

[66] Wang Q. *Cross-layer signalling for next-generation wireless systems*. Proceedings of Wireless Communications and Networking, 2003 (WCNC 2003), IEEE, vol.2, pp 1084 – 1089.

[67] Yamagishi K. and Hayashi T. *Opinion Model for Estimating Video Quality of Videophone Services*, IEEE GLOBECOM 2006, Nov. 2006.

APÉNDICE A: MODELO MODIFICADO DEL ESTABLECIMIENTO DE LA CONEXIÓN BANDABASE DE BLUETOOTH

Este apéndice muestra los módulos del modelo CPN usados en el análisis del establecimiento de la conexión Bandabase de Bluetooth.

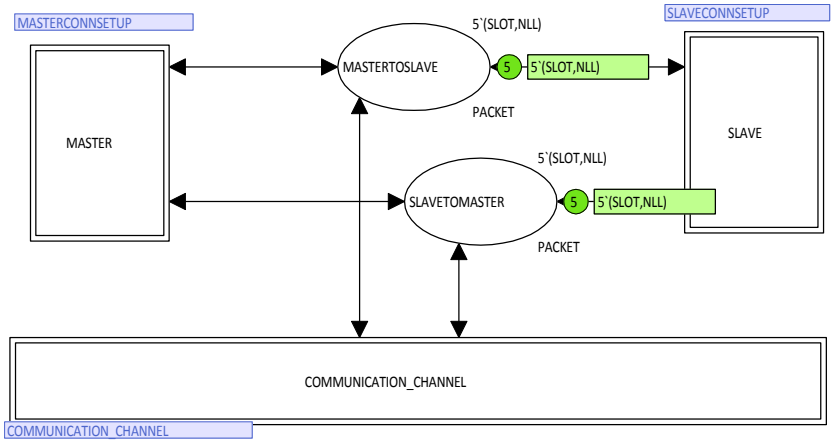


Figura A.1: Módulo superior de la jerarquía (BTNETWORK).

```
(* Standard declarations *)
  colset E = with e;
  colset INT = int;
  colset BOOL = bool;
  colset STRING = string;
  colset STATE = with STANDBY|
        INQUIRY|INQUIRYSCAN|
        INQUIRYRESPONSE|
        PAGE|PAGESCAN|
        MASTERRESPONSE|
       SLAVERESPONSE|
       CONNECTION;
  colset TYPE = with ID|FHS|POLL|SLOT;
  colset AC = with IAC|GIAC|DAC|NLL|CAC;
  colset PACKET = product TYPE * AC;
  colset TIND =with  s1|s2;
  var state: STATE;
  var prevstate: STATE;
  var anypacket: TYPE;
```

Figura A.2: Declaración estándar del modelo modificado.

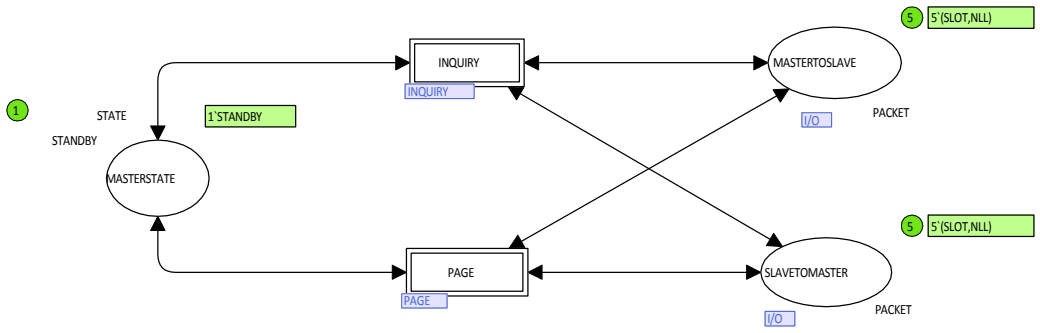


Figura A.3: Módulo del establecimiento de una conexión en el maestro (MASTER) modificado.

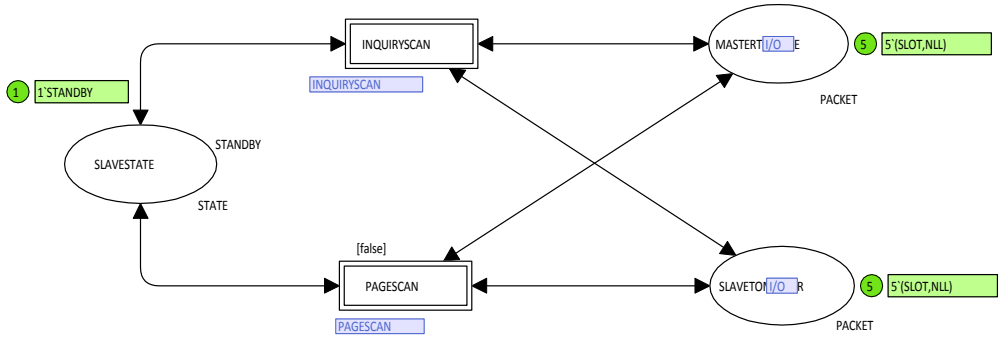


Figura A.4: Módulo del establecimiento de una conexión en el esclavo (SLAVE) modificado.

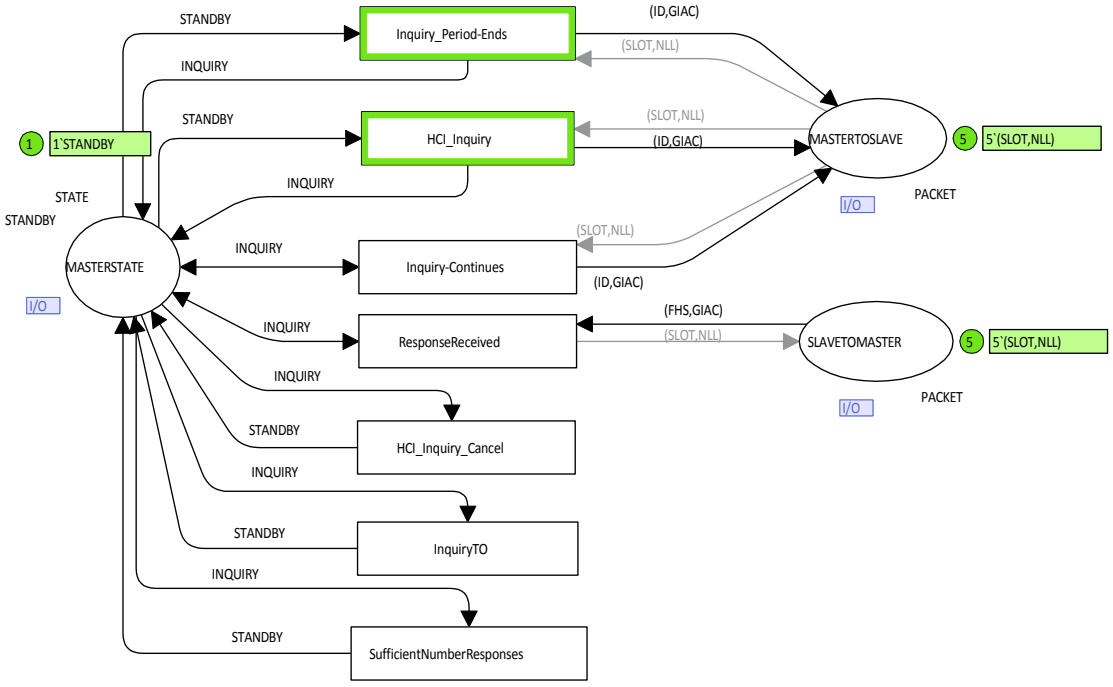


Figura A.5: Módulo del procedimiento de *Inquiry* (INQUIRY) modificado.

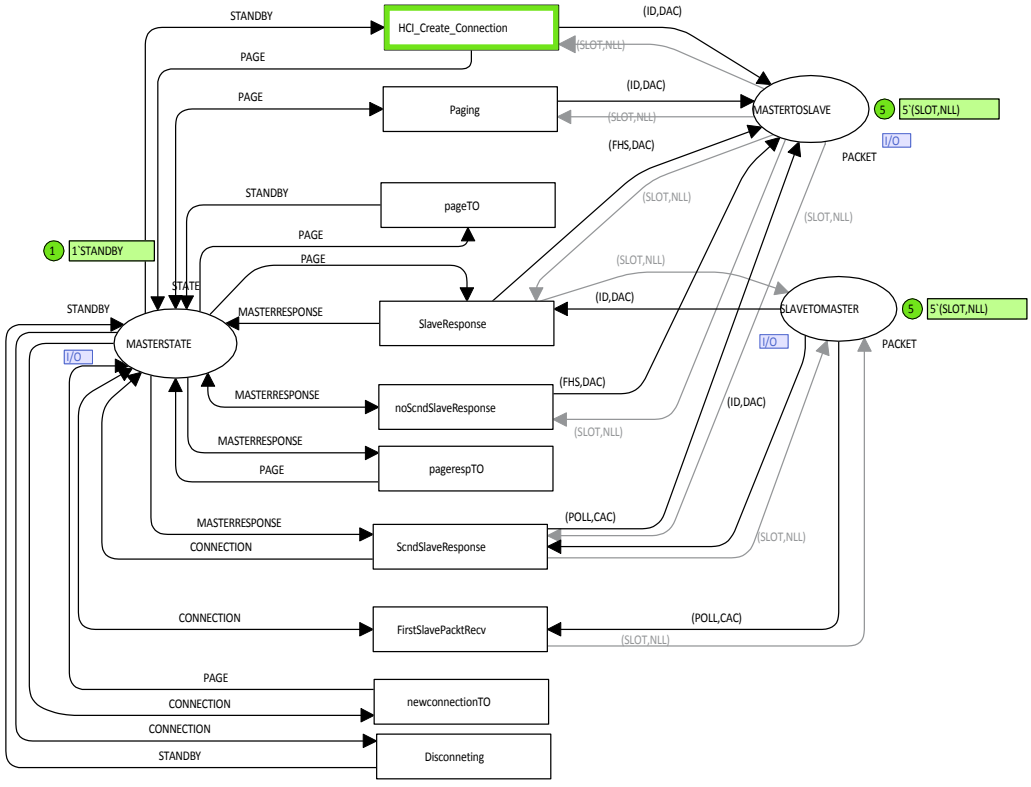


Figura A.6: Módulo del procedimiento de *Page* (PAGE) modificado.

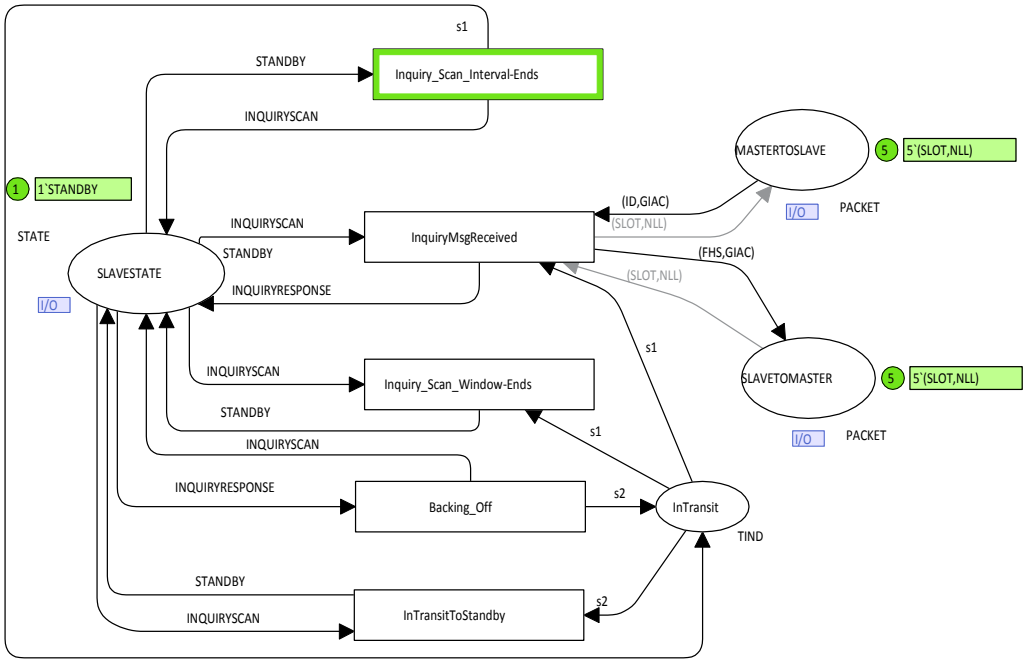


Figura A.7: Módulo del procedimiento *Inquiry Scan* (INQUIRY SCAN) modificado.

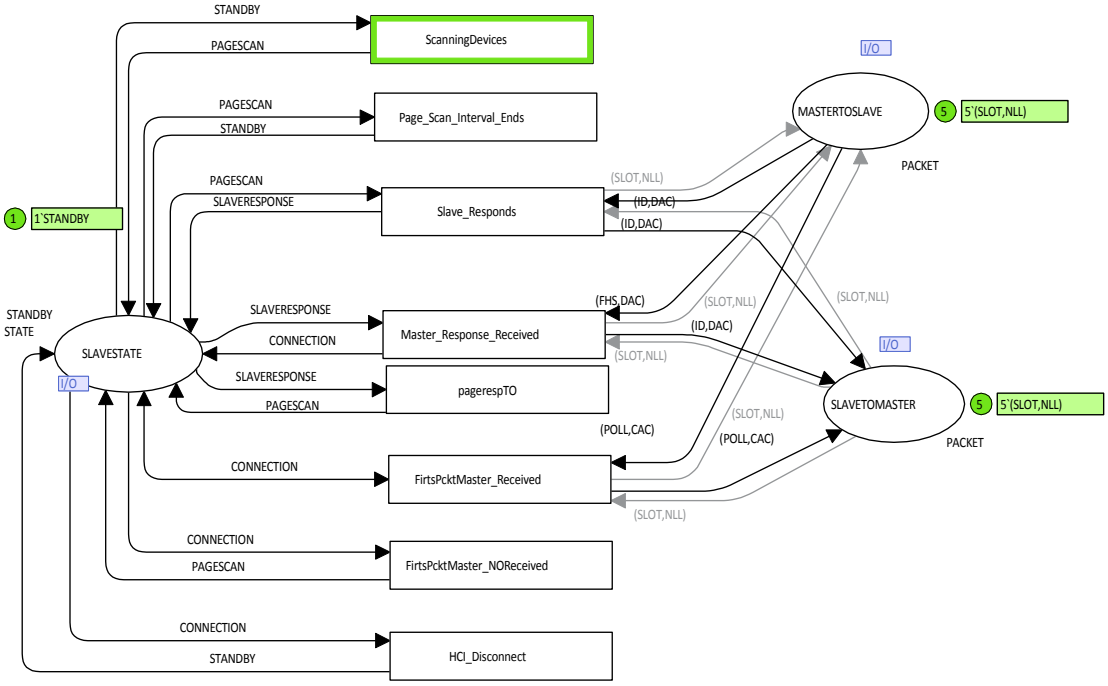


Figura A.8: Módulo del procedimiento de *Page Scan* (PAGE SCAN) modificado.

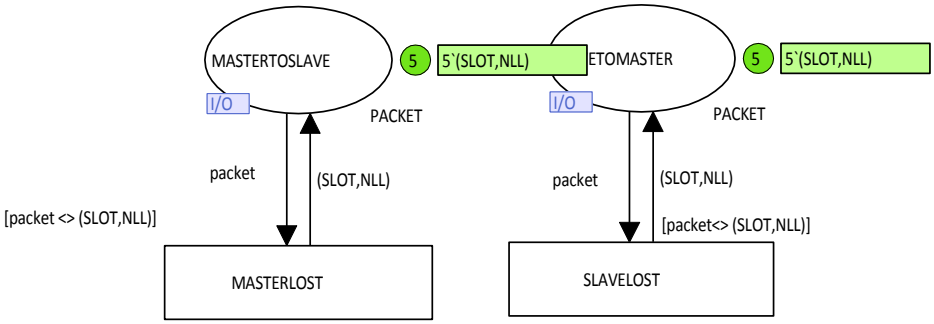


Figura A.9: Módulo del COMMUNICATION_CHANNEL.

APÉNDICE B: CÓDIGO ML PARA EL CHEQUEO DE LAS PROPIEDADES DEL MODELO DEL ESTABLECIMIENTO DE LA CONEXIÓN BANDABASE DE BLUETOOTH

A continuación se presenta el código ML para el chequeo de las propiedades de Establecimiento de la Conexión, Indagación, Retardo de Indagación, Retardo en el Establecimiento de la Conexión y Desconexión, respectivamente.

```
(******************** CONNECTION ESTABLISHMENT PROPERTY ******************)

fun ConnEstProp () = let
val pagemasdev = PredAllNodes (fn n  => (Mark.MASTERCONNSETUP'MASTERSTATE 1 n
   = 1`PAGE));
val connecteddev = PredAllNodes (fn n  => (Mark.MASTERCONNSETUP'MASTERSTATE 1
   n = 1`CONNECTION andalso Mark.SLAVECONNSETUP'SLAVESTATE 1 n =
   1`CONNECTION));
in
(pagemasdev <> nil) andalso (connecteddev <> nil)
end;
ConnEstProp ();

(******************** INQUIRY PROPERTY ******************)

fun InquiryProp () = let
val inquirymasdev = PredAllNodes (fn n  => (Mark.MASTERCONNSETUP'MASTERSTATE 1
   n = 1`INQUIRY));
val masterresponsesladev = PredAllNodes (fn n  =>
   (Mark.SLAVECONNSETUP'SLAVESTATE 1 n = 1`INQUIRYRESPONSE));
in
(inquirymasdev <> nil) andalso (masterresponsesladev <> nil)
end;
InquiryProp();

(******************** INQUIRY DELAY PROPERTY ******************)

fun IsInquiryNode n = Mark.MASTERCONNSETUP'MASTERSTATE 1 n = 1`INQUIRY;
```

```
fun InquiryLoop (n) = let
    val ASKCTLformula = ALONG (NF("Is Inquiry Node",IsInquiryNode));
in
  eval_node ASKCTLformula n
end;
fun InquiryDelayProp (a,b) = a andalso b;
fun InquiryDelayProperty () = SearchAllNodes (fn n  =>
   (Mark.MASTERCONNSETUP'MASTERSTATE 1 n = 1`INQUIRY), fn n => InquiryLoop
   n,true,InquiryDelayProp);
InquiryDelayProperty ();

(******   PAGE DELAY PROPERTY  *******)

fun IsPageNode n = Mark.MASTERCONNSETUP'MASTERSTATE 1 n = 1`PAGE;
fun PageLoop (n) = let
    val ASKCTLformula = ALONG (NF("Is Page Node",IsPageNode));
in
  eval_node ASKCTLformula n
end;
fun PageDelayProp (a,b) = a andalso b;
fun PageDelayProperty () = SearchAllNodes (fn n  =>
   (Mark.MASTERCONNSETUP'MASTERSTATE 1 n = 1`PAGE), fn n => PageLoop
   n,true,PageDelayProp);
PageDelayProperty ();

(******************** DESCONNECTION  PROPERTY ******************)

fun DesconnectionProp () = let
val slavedisconnected = PredAllNodes (fn n  =>
(Mark.MASTERCONNSETUP'MASTERSTATE 1 n = 1`CONNECTION andalso
Mark.SLAVECONNSETUP'SLAVESTATE 1 n = 1`STANDBY));
val masterdisconnected = PredAllNodes (fn n  =>
(Mark.MASTERCONNSETUP'MASTERSTATE 1 n = 1`STANDBY andalso
Mark.SLAVECONNSETUP'SLAVESTATE 1 n = 1`CONNECTION));
in
(slavedisconnected <> nil) andalso (masterdisconnected <> nil)
end;
DesconnectionProp ();
```

APÉNDICE C: MODELO DE MARKOV DEL CANAL DE COMUNICACION

El canal se modela según un modelo de Markov de tres estados (ver Figura C.1). Cada estado se caracteriza por cierta probabilidad de error de bit (BER): un estado del canal malo tiene una BER de 0,01, un estado medio tiene una BER de 0,001, y un estado bueno tiene un BER de 0,0001. Se asume que el canal se mantiene en cada uno de los estados bueno, medio y malo por un período medio de 50 ms, 100 ms, y 20 ms, respectivamente. Con estos parámetros se puede calcular las probabilidades de transición de estados, siendo p_{gg} la probabilidad de que el próximo estado sea bueno dado que el estado en curso es bueno, p_{bb} es la probabilidad que el próximo estado sea malo dado que el estado actual es malo y p_{mm} es la probabilidad de que el próximo estado sea medio dado que el estado actual es medio. Sea el tiempo promedio de duración en un estado Ti con i = g,b,m donde g = bueno, b = malo i, m = medio, las probabilidades anteriores se pueden obtener de la siguiente manera:

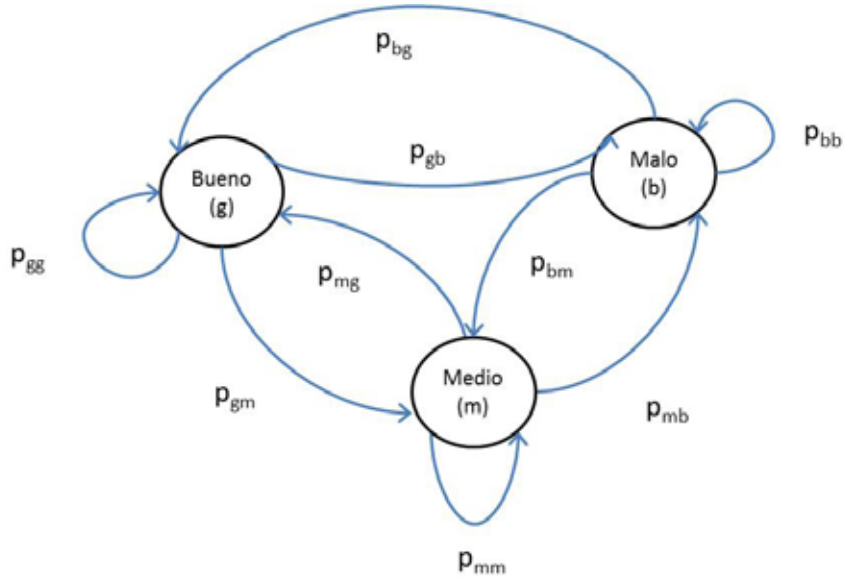


Figura C.1: Modelo de Markov de tres estados para el canal de comunicación usado en el Capítulo 5.

$$Tg = \frac{1}{1-p_{gg}} \quad (1)$$

$$Tb = \frac{1}{1-p_{bb}} (2)$$

$$Tm = \frac{1}{1-p_{mm}} (3)$$

Asumiendo que los cambios de estado ocurren en los límites de *slot* de longitud de 1 msec y colocando los valores iniciales dados anteriormente se tiene que p_{gg} = 0.98, p_{bb} = 0.95 y p_{mm} = 0.99. Los demás valores de las probabilidades de transición de estado se obtienen de la siguiente forma:

$$p_{gb} = \frac{1-p_{gg}}{2} (4)$$

$$p_{gm} = \frac{1-p_{gg}}{2} (5)$$

$$p_{bg} = \frac{1-p_{bb}}{2} (6)$$

$$p_{bm} = \frac{1-p_{bb}}{2} (7)$$

$$p_{mg} = \frac{1-p_{mm}}{2} (8)$$

$$p_{mb} = \frac{1-p_{mm}}{2} (9)$$

Colocando los valores de p_{gg} = 0.98, p_{bb} = 0.95 y p_{mm} = 0.99 en las ecuaciones (4) a la (9) se obtiene p_{gb}=0.01, p_{gb}=0.01, p_{bg}=0.025, p_{bm}=0.025, p_{mg}=0.005 y p_{mg}=0.005.

APÉNDICE D: FUNCIONES USADAS EN EL MODELADO Y ANALISIS DE LA MAC DE WIMAX ADAPTATIVA BASADA EN RETROALIMENTACIÓN

D.1 Funciones Usadas en el Modelado de la MAC de WiMax Adaptativa Basada en Retroalimentación

```
(* Model Functions
      Author: Maria Elena Villapol
      Ver: 1.0
*)

val header = 10;

fun BuildMPDU (cid,dtype,payload,k,numberofvblocks,fc,fsn) = if
(payload+header) <= (numberofvblocks*k) then (0,(cid,

(header+payload),fc,fsn,dtype,header,payload))
                                                          else
(0,(cid,(numberofvblocks*k),fc,fsn,dtype,header,

(numberofvblocks*k-header)));


(* This function builds one or more PDUs. The size of each MPDU is
"numberofvblocks" blocks of size k *)

fun BuildMPDUs (cid,dtype,payload,k,numberofvblocks,fsn,0) = if payload <= 0
then
                                                        []
                                                         else
                                                            if (payload +
header) <= (numberofvblocks*k) then
                                                  BuildMPDU
(cid,dtype,payload,k,numberofvblocks,0,fsn)::[]
                                                            else
                                                                BuildMPDU
```

```
(cid,dtype,payload,k,numberofvblocks,2,fsn)::BuildMPDUs (cid,dtype,(payload-
((numberofvblocks*k)-

header)),k,numberofvblocks,(fsn+1),3)|

    BuildMPDUs (cid,dtype,payload,k,numberofvblocks,fsn,fc) = if payload <= 0
then
                                                              []
                                                               else
                                                                  if (payload +
header) <= k then
                                                         BuildMPDU
(cid,dtype,payload,k,numberofvblocks,1,fsn)::[]
                                                                  else
                                                                      BuildMPDU

(cid,dtype,payload,k,numberofvblocks,3,fsn)::BuildMPDUs (cid,dtype,(payload-
((numberofvblocks*k)-

header)),k,numberofvblocks,(fsn+1),3);


(* cid,dtype,payload,k,int,fsn,action -> macsdus list *)

fun BuildTCSPDU (mpdu,0,k) = [] |
    BuildTCSPDU (mpdu,mpdulength,k) = if mpdulength >= k then (0,mpdu,k)::
BuildTCSPDU(mpdu,(mpdulength-k),k)
                                      else [(0,mpdu,mpdulength)];

(* Get cid and feedback information from a given msdu *)

fun feedbackn ((fbk,(cid,mlen,fc,fsn,dtype,mheader,mpayload))) = (cid,fbk);


(* Get number of MPDUs into which a SDU is segmented *)

fun NumberMPDUs (sdupayload:int,numberofvblocks:int,k:int) =  let
                                                 val msize = numberofvblocks*k
                                                 val r = sdupayload mod msize
                                                 fun nmpdus (x,y,z) = if  x = 0
then
                                                           y div z
                                                 else
                                                          (y div z) + 1;


                                                 val c = nmpdus
(r,sdupayload,msize)
                                                 val nheaders = (c*header)
                                                 val msize2 = nheaders +
sdupayload
                                                 val msize3 = msize2 mod msize

                                                       in
                                                       nmpdus
(msize3,msize2,msize)

                                                        end;
```

```
(* Updating Adaptive parameters *)

fun UpdateAdaptivePar (pr:par,nmpdu,n) = 1`(#1 pr,#2 pr + n,#3 pr,#4 pr,#5
pr,#6 pr,#7 pr,#8 pr, #9 pr + nmpdu,#10 pr);


fun UpdateAdaptiveParF3 (pr,pr1,hd5,hdn,k,k1) = let
                                          val nblocks = #2 pr - #4 pr
                                          val nblocks1 = #2 pr1 - #4 pr1
                                          val nmpdu = if hd5 = I then
NumberMPDUs (hdn,#2 pr,k) else NumberMPDUs(hdn,#2

pr1,k1)
                                        in
                                          UpdateAdaptivePar(pr,nmpdu,if
nblocks <=0 then (1 - #2 pr) else (0 - #4 pr)) ++
                                          UpdateAdaptivePar(pr1,nmpdu,if
nblocks1 <=0 then (1 - #2 pr1) else (0 - #4 pr1))
                                        end;

fun UpdateAdaptiveParF2 (pr,pr1,hd5,hdn,k,k1) = let
                                                   val nmpdu = if hd5 = I then
NumberMPDUs (hdn,#2 pr,k) else NumberMPDUs

(hdn,#2 pr1,k1)
                                                in

UpdateAdaptivePar(pr,nmpdu,0) ++ UpdateAdaptivePar(pr1,nmpdu,0)
                                                end;

fun UpdateAdaptiveParF1 (pr,pr1,hd5,hdn,k,k1) = let
                                      val nmpdu = if hd5 = I then
NumberMPDUs(hdn,#2 pr,k) else NumberMPDUs

(hdn,#2 pr1,k1)
                                    in
                                       UpdateAdaptivePar(pr,nmpdu,#3 pr) ++
UpdateAdaptivePar(pr1,nmpdu,#3 pr1)
                                    end;

fun UpdateAdaptiveParFI (pr,pr1,hd5,hdn,k,k1) = let
                                                  val nmpdu = if hd5 = I then
NumberMPDUs(hdn,#2 pr,k) else NumberMPDUs

(hdn,#2 pr1,k1)
                                                in

UpdateAdaptivePar(pr,nmpdu,0) ++ UpdateAdaptivePar(pr1,nmpdu,0)
                                                end;


(* Updating Burst Profile *)

fun UpdateBurstProfileF1 (m11,cid,r,r1,k,k1) = if (r1 - m11) < MINR then
1`(cid,k1,r1,NI) ++ 1`(cid,k,r,I)
```

```
                                                  else 1`(cid,k1,(r1 - m11),NI) ++
1`(cid,k,r,I)

fun UpdateBurstProfileF2 (m2,cid,r,r1,k,k1) = if (r + m2) <= MAXR then
1`(cid,k,(r + m2),I) ++ 1` (cid,k1,r1,NI)
                                                    else 1`(cid,k,r,I) ++ 1`
(cid,k1,r1,NI);

fun UpdateBurstProfileF3 (m3,m41,cid,r,r1,k,k1) = if (r -  m3) >= MINR then
                                                        if (r1 - m41) >= MINR
then
                                                              1`(cid,k,r -
m3,I) ++ 1`(cid,k1,r1 -  m41,NI)
                                                         else
                                                              1`(cid,k,r - m3,I)
++ 1`(cid,k1,r1,NI)
                                                   else
                                                         if (r1 - m41) >= MINR
then
                                                                1`(cid,k,r,I) ++
1`(cid,k1,r1 - m41,NI)
                                                         else
                                                                1`(cid,k,r,I) ++
1`(cid,k1,r1,NI);
 (* From SDU TO MPDU *)

fun SDUtoMPDU (cid,hd5,hdn,nb,nb1,fsn,fsn1,msduq2,k,k1) = if hd5 = I then
msduq2 ^^BuildMPDUs (cid,hd5,hdn,k,nb,fsn,0)
                                               else msduq2 ^^BuildMPDUs
(cid,hd5,hdn,k1,nb1,fsn1,0);

fun SDUtoMPDUF2 (cid,hd5,hdn,nb,nb1,fsn,fsn1,msduq2,k,k1) = if hd5 = I then
msduq2^^BuildMPDUs(cid,hd5,hdn,k,nb,fsn,0)
                                                  else msduq2^^BuildMPDUs
(cid,hd5,hdn,k1,nb1,fsn1,0);

fun SDUtoMPDUF1 (cid,hd5,hdn,nb,nb1,n1,n11,fsn,fsn1,msduq2,k,k1) = if hd5 = I
then msduq2^^BuildMPDUs(cid,hd5,hdn,k,(nb +

n1),fsn,0)
                                                                       else
msduq2^^BuildMPDUs (cid,hd5,hdn,k,(nb1 +

n11),fsn1,0);


fun SDUtoMPDUF3 (cid,hd5,hdn,nb,nb1,n2,n21,fsn,fsn1,msduq2,k,k1) = let
                                                      val nblocks = nb - n2
                                               val nblocks1 = nb1 - n21
                                               val x = if nblocks <=0 then 1
                                                         else nblocks
                                               val x1 = if nblocks1 <=0 then 1
                                                         else nblocks1
                                               in
                                                      if hd5 = I then msduq2 ^^
BuildMPDUs
```

```
(cid,hd5,hdn,k,x,fsn,0)
                                                    else msduq2 ^^
BuildMPDUs(cid,hd5,hdn,k1,x1,fsn1,0)
                                               end;
```

D.2 Funciones Usadas para Analizar las Propiedades de la MAC de WiMax Adaptativa Basada en Retroalimentación

```
(* Functions Used to Analyze the State Space
      Author: Maria Elena Villapol
      Ver: 2.0
*)

(*************** General Functions ******************)

fun ListLastMember (x::[]) = x |
    ListLastMember (x::y) = ListLastMember (y);

   (* Verify dead markings *)

(********** Member of a list ***********)
(* Check if an element is member of a list *)
(*****************************************************************)
fun ListMember (n, nil)=false|
ListMember (n, n1::nl) = if n=n1 then true else ListMember (n,nl);
(* "a * "a list -> bool *)

(********** Difference **********)
(* Returns the nodes in the nl1 list which are not in the nl2 list *)
(*******************************************************)
fun ListDiff ([],nl2,diffnodes) = diffnodes|
ListDiff (n1::nl1,nl2,diffnodes) = if ListMember (n1,nl2) then
ListDiff(nl1,nl2,diffnodes) else ListDiff(nl1,nl2,n1::diffnodes);
(* "a list * "a list -> list *)


fun DeadLocks() = PredNodes(ListDeadMarkings(), fn n => let

in

if (*Mark.BS'BSINMPDUS 1 n = 1`[]  andalso *)
   Mark.BS'BSINMPDUS 1 n = 1`[(0,(1,0,0,0,NI,0,0))] andalso
   Mark.BS'BSINTCSPDUS 1 n = 1`[]  andalso
   Mark.BS'BSOUTCSPDUS 1 n = 1`[]  andalso
   Mark.BS'BSOUTMPDUS 1 n = 1`[]  andalso
   Mark.BS'Videoframes 1 n = 1`[]  andalso
   Mark.SS'SSINTCSPDUS 1 n = 1`[]  andalso
   Mark.SS'SSMPDUS 1 n =1`[]  andalso
   Mark.SS'SSOUTTCSPDUS 1 n = 1`[]  andalso
   Mark.WimaxNetwork'DLChannel 1 n = 1`[]  andalso
   Mark.WimaxNetwork'ULChannel 1 n = 1`[]

then false
```

```
else true end,NoLimit);

(* unit -> Node list *)

(* Gives the size of the a ms element *)

fun size_mse ([]) = 0|
      size_mse (y::l2) = 1 + size_mse(l2);
(* Gives the size of a ms *)

fun size_ms ([]) =0|
      size_ms(x::l) = size_mse (x) + size_ms (l);

(* Gives the number of mse *)

fun number_mse ([]) = 0|
      number_mse (x::l) = 1 + number_mse(l);

(* Gives the maximum size of the size of a mse *)

fun max_size_ms ([]) =0|
      max_size_ms(x::[]) = size_mse(x)|
      max_size_ms(x::(y::l)) =  if size_mse(x) > size_mse(y) then max_size_ms
(x::l)
                                                  else max_size_ms(y::l);
(* Check if an element is member of a list *)

fun ListMember (n, nil)=false|
ListMember (n, n1::nl) = if n=n1 then true else ListMember (n,nl);

(* Returns the nodes in the nl1 list which are not in the nl2 list *)

fun ListDiff ([],nl2,diffnodes) = diffnodes|
     ListDiff (n1::nl1,nl2,diffnodes) = if ListMember (n1,nl2) then
     ListDiff(nl1,nl2,diffnodes) else ListDiff(nl1,nl2,n1::diffnodes);


fun max(x,y) = if x>y then x
                        else y;

fun maxr(x:real,y:real) = if x>y then x
                        else y;
fun sumr (x:real,y:real) = x+y;


fun NumberMPDUSperSDU (a)= let
               val dn= DestNode a;
               val sn = SourceNode a;
               val x1=Mark.BS'BSOUTMPDUS 1 dn;
               val x2 =Mark.BS'BSOUTMPDUS 1 sn;
             val y1 = ms_to_col (x1);
               val y2 = ms_to_col (x2);
               in
                (length y1 - length y2)
               end;
```

```
(* MAX NUMBER OF MPDUS THAT A SDU IS BROKEN *)

fun MaxNumberPDUsperSDU () = SearchAllArcs(fn a => (ArcToTI a =
TI.BSMACCPS'FEEDBACK3 1 orelse ArcToTI a = TI.BSMACCPS'FEEDBACK2 1
                              orelse ArcToTI a = TI.BSMACCPS'FEEDBACK1 1 orelse
ArcToTI a = TI.BSMACCPS'INITIALFEEDBACK 1), fn a => NumberMPDUSperSDU(a),0,
max);

(* MAX SIZE OF MDU *)


fun Max2((x1,(x2,x3,x4,n,x5,x6,x7))::[])=x3|
    Max2((x1,(x2,x3,x4,n,x5,x6,x7))::((y1,(y2,y3,y4,n1,y5,y6,y7))::l)) = max
(max(x3,y3),Max2((y1,(y2,y3,y4,n1,y5,y6,y7))::l));

fun Max1([])=0|
    Max1(msdu::l) = max(Max2(msdu::l),Max1(l));

fun MaxSizeMPDU([]) = 0|
    MaxSizeMPDU(t::l) = max(Max1(t),MaxSizeMPDU(l));

(* MAX NUMBER TCS PER MPDU *)

fun NumberTCSsperMPDU (a)= let
               val dn= DestNode a;
               val sn = SourceNode a;
               val x1=Mark.BS'BSINTCSPDUS 1 dn;
               val x2=Mark.BS'BSINTCSPDUS 1 sn;
             val y1 = ms_to_col (x1);
               val y2 = ms_to_col (x2);
               in
                (length y1 - length y2)
               end;

fun MaxNumberTCSsperMPDU () = SearchAllArcs(fn a => ArcToTI a =
TI.BSPHYTCS'MPDUTOPHY 1, fn a => NumberTCSsperMPDU(a),0, max);

(* MAX NUMBER OF BLOCKS *)

fun MaxNumberBlocks([]) = 0|
    MaxNumberBlocks((n1,n2,n3,n4,n5,n6,n7,n8,n9,n10) ::l) =
max(n2,MaxNumberBlocks(l));

(* MAX EFFICIENCY *)

fun gput (h:int,p:int,f:int) = Real./(Real.fromInt(p),Real.fromInt(p+h+f));

fun Efficiency2(((x1,x2,x3,x4,x5,x6,x7),x8,x9)::[])=gput(x6,x7,x9)|

Efficiency2(((x1,x2,x3,x4,x5,x6,x7),x8,x9)::((y1,y2,y3,y4,y5,y6,y7),y8,y9)::l)
= maxr
(maxr(gput(x6,x7,x9),gput(y6,y7,y9)),Efficiency2(((y1,y2,y3,y4,y5,y6,y7),y8,y9
)::l));

fun Efficiency1([])=0.0|
    Efficiency1(pframe::l) = maxr(Efficiency2(pframe::l),Efficiency1(l));
```

```
fun Efficiency([]) = 0.0|
    Efficiency(t::l) = maxr(Efficiency1(t),Efficiency(l));


(* AVERAGE EFFICIENCY    *)

fun NumPframe ([]) = 0|
    NumPframe (t::l) =  length t + NumPframe (l);

(*fun pf_to_val (((x1,x2,x3,x4,x5,x6,x7),x8,x9)) = (x6,x7,x9);*)

fun TotEfficiency2(((x1,x2,x3,x4,x5,x6,x7),x8,x9)::[])=gput(x6,x7,x9)|

TotEfficiency2(((x1,x2,x3,x4,x5,x6,x7),x8,x9)::((y1,y2,y3,y4,y5,y6,y7),y8,y9):
::[]) = sumr(gput(x6,x7,x9),gput(y6,y7,y9))|

TotEfficiency2(((x1,x2,x3,x4,x5,x6,x7),x8,x9)::((y1,y2,y3,y4,y5,y6,y7),y8,y9):
:l) = sumr(gput(x6,x7,x9),TotEfficiency2(((y1,y2,y3,y4,y5,y6,y7),y8,y9)::l));

fun TotEfficiency1([])=0.0|
    TotEfficiency1(pframe::l) = TotEfficiency2(pframe::l);


fun TotEfficiency([]) = 0.0|
    TotEfficiency(t::l) = sumr(TotEfficiency1(t),TotEfficiency(l));


fun AvEfficiency(l) = Real./(TotEfficiency (l),Real.fromInt(NumPframe(l)));

(* MAX OVERHEAD *)

fun ohead (h:int,p:int,f:int) = Real./(Real.fromInt(f),Real.fromInt(p+h));

fun Overhead2(((x1,x2,x3,x4,x5,x6,x7),x8,x9)::[])=ohead(x6,x7,x9)|

Overhead2(((x1,x2,x3,x4,x5,x6,x7),x8,x9)::((y1,y2,y3,y4,y5,y6,y7),y8,y9)::l) =
maxr
(maxr(ohead(x6,x7,x9),ohead(y6,y7,y9)),Overhead2(((y1,y2,y3,y4,y5,y6,y7),y8,y9
)::l));

fun Overhead1([])=0.0|
    Overhead1(pframe::l) = Overhead2(pframe::l);

fun MaxOverhead([]) = 0.0|
    MaxOverhead(t::l) = maxr(Overhead1(t),MaxOverhead(l));

(* AVERAGE OVERHEAD    *)


(*fun pf_to_val (((x1,x2,x3,x4,x5,x6,x7),x8,x9)) = (x6,x7,x9);*)

fun TotOverhead2(((x1,x2,x3,x4,x5,x6,x7),x8,x9)::[])=ohead(x6,x7,x9)|

TotOverhead2(((x1,x2,x3,x4,x5,x6,x7),x8,x9)::((y1,y2,y3,y4,y5,y6,y7),y8,y9)::[
]) = sumr(ohead(x6,x7,x9),ohead(y6,y7,y9))|
```

```
TotOverhead2(((x1,x2,x3,x4,x5,x6,x7),x8,x9)::((y1,y2,y3,y4,y5,y6,y7),y8,y9)::l
) = sumr(ohead(x6,x7,x9),TotOverhead2(((y1,y2,y3,y4,y5,y6,y7),y8,y9)::l));

fun TotOverhead1([])=0.0|
    TotOverhead1(pframe::l) = TotOverhead2(pframe::l);

fun TotOverhead([]) = 0.0|
    TotOverhead(t::l) = sumr(TotOverhead1(t),TotOverhead(l));

fun AvOverhead(l) = Real./(TotOverhead (l),Real.fromInt(NumPframe(l)));

(****************** ASKCTL Properties ******************)

use (ogpath^"/ASKCTL/ASKCTLloader.sml");

fun WasMPDUReassembledCorrectly (a) = let
   fun takempdu (Bind.BSPHYTCS'MPDUTOPHY
   (1,{cid=_,hd5=_,k=_,mpdu=v1,msduq=_,nb=_,r=_,tsdus=_})) = v1;
   val mpdu1 = takempdu (ArcToBE a);

   fun NeitherReassembledCorrectlyNorDiscarded (a1) = let

       fun ItisTheMPDU (Bind.SSPHYTCS'lastblockcorrect
(1,{e=_,fbk=_,ln1=_,mpdu= v1,mpduq=_ ,p=_,p1=_,tsduq=_,tsdus=_})) = v1 =mpdu1;


        fun ItisTheDiscardedMPDU (Bind.SSPHYTCS'blockdiscarded
(1,{e=_,fbk=_,ln1=_,mpdu= v1,p=_,p1=_,tsduq=_,tsdus=_})) = v1 =mpdu1;


      val result = (if ArcToTI a1 = TI.SSPHYTCS'lastblockcorrect 1 then
                  ItisTheMPDU (ArcToBE a1)
            else

                  if ArcToTI a1 = TI.SSPHYTCS'blockdiscarded 1 then
                        ItisTheDiscardedMPDU (ArcToBE a1)
            else
                  false);

      in

            not result
      end;

    val ASKCTLformula =
POS(AF("MPDUReasembledCorrectly",NeitherReassembledCorrectlyNorDiscarded));

    in

      eval_arc ASKCTLformula a

    end;

fun MPDUReassembly (a,b) = a andalso b;
```

```
fun MPDUReassemblyProperty () = SearchAllArcs (fn a => ArcToTI a =
TI.BSPHYTCS'MPDUTOPHY 1, fn a => WasMPDUReassembledCorrectly a,
true,MPDUReassembly);


(**************************** SDU REASSEMBLY ****************)


fun WasSDUReassembledCorrectly (a) = let
  fun sdulength (b) = case b of
Bind.BSMACCPS'INITIALFEEDBACK
   (1,{cid=_,hd5=_,hdn =
p,k=_,k1=_,msduq=_,msduq2=_,pr=_,pr1=_,r=_,r1=_,sduq=_}) => p|
Bind.BSMACCPS'FEEDBACK1
   (1,{cid=_,hd5=_,hdn =p,k=_,k1=_,msduq=
_,msduq2=_,pr=_,pr1=_,r=_,r1=_,sduq=_}) => p|
Bind.BSMACCPS'FEEDBACK2
   (1,{cid=_,hd5=_,hdn =p,k=_,k1=_,msduq=
_,msduq2=_,pr=_,pr1=_,r=_,r1=_,sduq=_}) => p|
Bind.BSMACCPS'FEEDBACK3
   (1,{cid=_,hd5=_,hdn =p,k=_,k1=_,msduq=
_,msduq2=_,pr=_,pr1=_,r=_,r1=_,sduq=_}) => p;

val l = sdulength (ArcToBE a);
val smpdums = Mark.BSMACCPS'BSOUTMPDUS 1 (SourceNode a);
val smpdu::x= smpdums;
val dmpdums = Mark.BSMACCPS'BSOUTMPDUS 1 (DestNode a);
val dmpdu::y= dmpdums;
val addedmpdus = ListDiff (dmpdu,smpdu,[]);
val (idu,(cid,mlen,fc,fsn,dtype,mheader,mpayload))::z= addedmpdus;

   fun SDUNeitherReassembledCorrectlyNorDiscarded (a1) = let

       fun ItisTheSDU (m1,m2) = let
          val m3::[] = m1;
          val m4::[] = m2;
          val (ct,t,p1) = ListLastMember (m3);
          val (x,s,y) = m4;
         in
            ((s-1) = fsn) andalso (l =p1)
         end;

        fun HaveSDUBeenDiscarded ((i,x)::[],Bind.SSPHYTCS'blockdiscarded
(1,{e=_,fbk=_,ln1=_,mpdu= v1,p=_,p1=_,tsduq=_,tsdus=_}),a2) = v1 = x |
           HaveSDUBeenDiscarded ((i,x)::y,Bind.SSPHYTCS'blockdiscarded
(1,{e=_,fbk=_,ln1=_,mpdu= v1,p=_,p1=_,tsduq=_,tsdus=_}),a2) = if v1 = x then
true else HaveSDUBeenDiscarded (y,ArcToBE a2,a2);


        val result =    (if ArcToTI a1 = TI.SSMACCPS'RcvARQ 1 then
                  ItisTheSDU (Mark.SSMACCPS'RcvVideoFrames 1 (DestNode
a1),Mark.SSMACCPS'nseq 1 (DestNode a1))
                      else
                         if ArcToTI a1 = TI.SSPHYTCS'blockdiscarded 1 then
                       HaveSDUBeenDiscarded (addedmpdus,ArcToBE a1,a1)
                  else
                      false);
```

```
      in
          not result
      end;

    val ASKCTLformula = POS(AF
("MPDUReasembledCorrectly",SDUNeitherReassembledCorrectlyNorDiscarded));

    in

      eval_arc ASKCTLformula a
    end;

fun SDUReassembly (a,b) = a andalso b;
fun SDUReassemblyProperty () = SearchAllArcs (fn a => ((ArcToTI a =
TI.BSMACCPS'INITIALFEEDBACK 1) orelse (ArcToTI a =  TI.BSMACCPS'FEEDBACK1
1)orelse (ArcToTI a = TI.BSMACCPS'FEEDBACK2 1)orelse (ArcToTI a
=TI.BSMACCPS'FEEDBACK3 1)), fn a => WasSDUReassembledCorrectly a,
true,SDUReassembly);
```

D.3 Funciones Usadas para Imprimir los Resultados

```
(********************** PRINT RESULTS ************************)

use "C:/ME Documents (Casa)/Trabajo/Investigacion/Proyectos/VideoOverWimax/CPN
Model/CPNModelAnalysisFunctionsV2.0.txt";

val filename = "C:/ME Documents
(Casa)/Trabajo/Investigacion/Proyectos/VideoOverWimax/CPN Model/Output.txt";

fun list2string (nil) = " "|
    list2string (s::sl) = Int.toString(s)^","^list2string(sl);

fun plist2string (nil) = " "|
    plist2string ((s1,s2)::sl) =
String.toString(s1)^","^Int.toString(s2)^","^plist2string(sl);

fun bool2string (b) = if b then "True" else "False";

fun writepropres (ofile,mssg) = let
val outfile = TextIO.openAppend(ofile);
val _ = TextIO.output (outfile,mssg);
in
TextIO.closeOut(outfile)
end;

(************************ DEADLOCK PROPERTY ****************)

fun DL () =
let
val _= writepropres(filename,"========== Start Umproper Termination ==========
\n");
val x= DeadLocks();
val _ = writepropres (filename,"DeadLocks= [" ^ list2string(x)^"] \n");
in
writepropres(filename,"========== End Umproper Termination ========== \n")
end;
```

```
(******************** MPDU REASSEMBLY PROPERTY ****************)


fun MPDUREASS () =
let
val _= writepropres(filename,"========== Start MPDU REASSEMBLY ==========
\n");
val x= MPDUReassemblyProperty ();
val _ = writepropres (filename,"MPDUReassembledCorrectly = [" ^
bool2string(x)^"] \n");
in
writepropres(filename,"========== End MPDU REASSEMBLY  ========== \n")
end;

(******************** SDU REASSEMBLY PROPERTY ****************)


fun SDUREASS () =
let
val _= writepropres(filename,"========== Start SDU REASSEMBLY ========== \n");
val x= SDUReassemblyProperty ();
val _ = writepropres (filename,"SDUReassembledCorrectly = [" ^
bool2string(x)^"] \n");
in
writepropres(filename,"========== End SDU REASSEMBLY  ========== \n")
end;

(******************** NETWORK PERFORMANCE **********************)

fun NetworkPerformance () = let
val _= writepropres(filename,"======== Start NETWORK PERFORMANCE ==========
\n");
val x = MaxNumberPDUsperSDU ();
val y =  MaxSizeMPDU(UpperMultiSet(Mark.BS'BSOUTMPDUS 1));
val z = MaxNumberTCSsperMPDU();
val x1= MaxNumberBlocks(UpperMultiSet(Mark.BSMACCPS'AdaptivePar 1));
val x2 = Efficiency(UpperMultiSet(Mark.WimaxNetwork'DLChannel 1));
val x3 = TotEfficiency(UpperMultiSet(Mark.WimaxNetwork'DLChannel 1));
val x4= NumPframe(UpperMultiSet(Mark.WimaxNetwork'DLChannel 1));
val x5 = MaxOverhead(UpperMultiSet(Mark.WimaxNetwork'DLChannel 1));
val x6 = AvOverhead(UpperMultiSet(Mark.WimaxNetwork'DLChannel 1));

val _ = writepropres (filename,"Max Number MPDUs per SDU = [" ^
Int.toString(x)^"] \n");
val _ = writepropres (filename,"Max Size a MPDU = [" ^ Int.toString(y)^"]
\n");
val _ = writepropres (filename,"Max Number TCSs per MPDU = [" ^
Int.toString(z)^"] \n");
val _ = writepropres (filename,"Max Number Blocks per MPDU = [" ^
Int.toString(x1)^"] \n");

val _ = writepropres (filename,"Efficiency = [" ^ Real.toString(x2)^"] \n");
val _ = writepropres (filename,"Total Efficiency = [" ^ Real.toString(x3)^"]
\n");
val _ = writepropres (filename,"Number PHY Frames = [" ^ Int.toString(x4)^"]
\n");
```

```
val _ = writepropres (filename,"Max Overhead = [" ^ Real.toString(x5)^"] \n");
val _ = writepropres (filename,"Average Overhead = [" ^ Real.toString(x6)^"]
\n");

in
writepropres(filename,"========== End NETWORK PERFORMANCE ========== \n")
end;


fun InitialMarking (l) =
let
val _= writepropres(filename,"========== Initial Video SDUs ========== \n");
val _ = writepropres (filename,"Initial SDUs = [" ^ plist2string(l)^"] \n");
in
writepropres(filename,"========== End Initial Video SDUs  ========== \n")
end;
```

Contenido

Lista de Figuras

Lista de Tablas

Siglas

ARPA	Advanced Research Projects Agency – Agencia de Proyectos de Investigación Avanzados
ARQ	Automatic Repeat Request- Requerimiento de Repetición Automático
BER	Bit Error Rate- Tasa de Error de Bit
BNEP	Bluetooth Network Encapsulation Protocol – Protocolo de Encapsulamiento de Red Bluetooth
BS	Base Station – Estación Base
CAC	Channel Access Code- Código de Acceso al Canal
CPN	Colored Petri Nets – Redes de Petri Coloreadas
CS	Service Specific Converge Sublayer – Subcapa de Convergencia Específica del Servicio
DAC	Device Access Code – Código de Acceso al Dispositivo
DL-MAP	Dowlink-Map – Mapa del Enlace Corriente Abajo
DoD	Department of Defense – Departamento de Defensa
FDD	Frequency Division Duplexing – Duplexación por División de Frecuencia
FEC	Forward Error Correction – Corrección de Errores Hacia Adelante
GAP	Generic Access Profile – Perfil de Acceso Genérico
GF	Galois Fields – Campos de Galois
GPSK	Gaussian Frecuency-Shift-Keying - Modulación por Desplazamiento de Frecuencia Gaussiana

GUI	Graphical User Interface – Interfaz de Usuario Gráfica
IAC	Inquiry Access Code – Código de Acceso de Indagación
IPTV	Internet Protocol Television – Televisión sobre el Protocolo IP
IrDA	InfraRed Data Association – Asociación de Datos Infrarrojos
ISM	Industrial, Scientific and Medical – Industrial, Científico y Médica
ISO	International Organization for Standardization – Organización Internacional para la Estandarización
L2CAP	Logical Link Control and Adaptation Protocol – Protocolo de Adaptación y Control de Enlace Lógico
LAP	Lower Address Part – La Parte de la Dirección más Baja
LCP	Link Control Protocol – Protocolo de Control de Enlace
LMP	Link Manager Protocol – Protocolo de Gestión del Enlace
LSB	Least Significant Bit – Bit Menos Significativo
MAC	Medium Access Control – Control de Acceso al Medio
MAC CPS	MAC Common Part Sublayer – Subcapa de Parte Común de la MAC
MOS	Mean Opinion Score – Puntuación de Opinión Media
MSB	Most Significant Bit – Bit más Significativo
OBEX	Object Exchange – Intercambio de Objeto
OG	Occurrence Graph – Grafo de Ocurrencia
OSI	Open Systems Interconnection – Interconexión de Sistemas Abiertos
PDU	Protocol Data Unit – Unidad de Datos de Protocolo
PS	Physical Slots – Espacio Físico
QAM	Quadrature Amplitude Modulation – Modulación por Amplitud y Cuadratura
QoE	Quality of Expierience – Calidad de la Experiencia

QoS	Quality of Service – Calidad de Servicio
RF	Radio Fraquency – Frecuencia de Radio
RS	Reed Solomon - Reed Solomon
RSVP	Resource Reservation Protocol – Protocolo de Reservación de Recursos
SAP	Service Access Point – Punto de Acceso al Servicio
SCC	Strongly Connected Component – Componente Fuertemente Conectado
SDP	Service Discovery Protocol – Protocolo de Descubrimiento del Servicio
SIG	Special Interest Group – Grupo de Interés Especial
SS	Subscriber Station – Estación Suscriptora
TC	Transmission Convergence – Convergencia de Transmisión
TDD	Time Division Duplexing – Duplexación por División de Tiempo
UL-MAP	Uplink-Map – Mapa del Enlace Corriente Arriba
VoIP	Voice over IP – Voz sobre IP
WPAN	Wireless Personal Area Netrwoks – Redes de Área Personal Inalámbricas
WWAN	Wireless Wide Area Networks – Redes de Área Amplia Inalámbricas

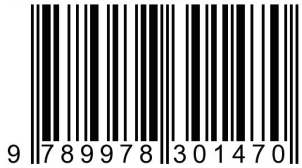
ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

LATACUNGA

ISBN: 978-9978-301-47-0



Publicaciones Científicas