Escola Tècnica Superior
d'Enginyeria de Telecomunicació de Barcelona

Diseño e implementación de una calculadora de PSNR para evaluar las prestaciones de servicios de vídeo bajo demanda sobre redes MANET

Alumno: Miguel Toscano Rodero
Directora: Mónica Aguilar Igartua
Barcelona, Julio 2008

Departament d´Enginyeria Telemática de la Universitat Politécnica de Catalunya

UNIVERSITAT POLITÈCNICA DE CATALUNYA

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

GLOSARIO DE ACRÓNIMOS

ABR	Associativity-Based Routing
AC	Access Category
ADSL	Asymmetric Digital Subcriber Line
ADV	Adaptative Distance Vector
AF	Asured Forwarding
AIF	Arbitrary Interframe Space
a-MMDSR	Adaptative Multipath Multimedia Dynamic Source Routing
AODV	Ad Hoc On Demand Distance Vector Routing
ap-MMDSR	Adaptative-PEP-Multipath Multimedia Dynamic Source Routing
BER	Bit Error Ratio
BS	Base Station
CBR	Constant Bit Rate
CBRP	Cluster-Based Routing Protocol
CCITT	The United Nations Consultative Committee for International Telephony and Telegraphy
CEDAR	Core Extraction Distributed Ad-hoc Routing
CSMA	Carrier Sense Multiple Access
CW	Contention Window
d-MMDSR	Dynamic Contention Window Multipath Multimedia Dynamic Source Routing

DOOR	Out of Order and Response
DREAM	Distance Routing Effect Algorithm for Mobility
DSCP	Diffserv Code Point
DSDV	Destination-Sequenced Distance Vector
DSR	Dynamic Source Routing
DYMO	Dynamic MANET On demand
ECN	Explicit Congestion Notification
EF	Expedited Forwarding
ELFN	Explicit Link Failure Notification
FIFO	First In First Out
FQMM	Flexible QoS Model for Mobile Ad-hoc Networks
GoP	Group of Pictures
GNU	General Public License
g-MMDSR	Game Theory – Multipath Multimedia Dynamic Source Routing
HDTV	High Definitio TV
HM	Hello Message
HMR	Hello Message Reply
HSR	Hierarchical State Routing
ICMP	Internet Control Message Protocol
IDLE	Idenficador de Frame
IETF	Internet Engineering Task Force
INSIGNIA	IP-based QoS framework for Mobile Ad-hoc Networks
IP	Internet Protocol
ISO	International Organization of Standardization

LAR	Location-Aided Routing
LEP	Link Error Probability
MAC	Medium Access Control
MANET	Mobile Ad-hoc NETwork
MB	Mobility Metric
MIT	Massachusetts Institute of Technology
MMDSR	Multipath Multimedia Dynamic Source Routing
MMWM	Multimedia support in Mobile Wireless Networks
MN	Mobile Nodes
MPEG-2	Moving Picture Expert Group
MPLAYER	The Movie Player
MSE	Mean Square Error
NS	Network Simulator
OLSR	Optimised Link State Routing
PDA	Personal Digital Assistant
PEP	Probabilidad de Error de Camino
PHB	Per Hop Behaviour
PPM	Portable Pixel Map
PM	Probe Message
PMR	Probe Message Reply
PSNR	Peak Signal-to Noise Ratio
PQ	Priority Queue
QoS	Quality of Service

RDSI	Red Digital de Servicios Integrados
RFN	Route Failure Notification
RM	Reliability Metric
RRN	Re-establishmente Notification
RSVP	Resource Reservation Protocol
RTCP	Real Time Control Protocol
RTO	Recovery Time Objective
RTP	Real Time Procotol
RWMM	Random Waypoint Mobility Model
SINCR	Signal to interference plus noise ratio
SLURP	Scalable Location Update Routing Protocol
s-MMDSR	static Mulipath Multimedia Dynamic Source Routing
STAR	Source-Tree Adaptaive Routing
SWAN	Stateless Wireless Ad-hoc Networks
TBRPF	Topology Dissemination Based on Reverse-Path Forwarding
TCL	Tool Command Language
TCP	Transport Control Protocol
TORA	Temporally Ordered Routing Algorithm
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
VINT	Virtual Internetwork Testbed
ZHLS	Zone-based Hierarchical Link State
ZRP	Zone Routing Protocol

Introducción

Han pasado muchos años desde la aparición de Internet en nuestras vidas, desde los años 70. El escaso ancho de banda que proporcionaban los antiguos módems de 14.4 kbps apenas permitía la navegación web, que estaba basada principalmente en texto, acceso al IRC[1] (*Internet Relay Chat*), o la descarga de pequeñas aplicaciones cuyo tamaño no solía superar algunos cientos de kilobits.

La progresiva aparición de nuevas tecnologías en la red de acceso, como RDSI (Red Digital de Servicios Integrados), ADSL (Asymmetric Digital Subscriber Line), o UMTS (*Universal Mobile Telecommunications System*), y el avance tecnológico en el núcleo de las redes de comunicaciones (como la fibra óptica), ha permitido el desarrollo de aplicaciones con cada vez mayores requisitos en cuanto a ancho de banda o retardo se refiere. Servicios como la video-conferencia, la descarga de vídeo bajo demanda, o el acceso a base de datos con elevadas cargas de información, son en la actualidad posibles gracias a estas mejoras.

Sin embargo, desde el nacimiento de Internet poco hemos avanzado en relación a la calidad de servicio ofrecida al usuario. Es una realidad que disponemos de lo que es conocido como “acceso a banda ancha”, donde hay un elevado caudal para acceder a cualquier tipo de información. Pero muchas veces ese acceso no

[1] IRC fue creado por Jarkko Oikarinen en agosto de 1988, es un protocolo de comunicación muy sencillo que funciona en tiempo real basado en texto, que permite debates en grupo o entre dos personas.

asegura una adecuada disponibilidad de recursos en el transcurso de toda la comunicación.

Acompañado a esta explosión de ancho de banda disponible que han experimentado todos los usuarios, en los más recientes años hemos podido vivir la eclosión de las redes inalámbricas, que han llevado a la aparición masiva de puntos de acceso ubicuos. Estos nuevos puntos de conexión nos llevan a vivir un acceso a la información en constante movimiento, donde se valora de forma especial la posibilidad de acceder a nuestros tradicionales servicios sin ataduras del sistema de cableado.

Existen dos tipos de redes inalámbricas. Las primeras en aparecer fueron las llamadas con **infraestructura**, las cuales necesitan de un punto de acceso (*Access Point*) para establecer una comunicación. Pero se dan situaciones donde no se puede disponer de esta infraestructura, como en situaciones de emergencias, desastres naturales, vigilancia en fronteras, resolución de conflictos, turistas en museos o en áreas turísticas, estudiantes en campus universitarios, etc. Es entonces cuando aparecieron las redes llamadas **Ad-hoc**, que no necesitan de infraestructura para establecer una comunicación, en que cada estación móvil es capaz de llevar a cabo funciones de receptor, emisor o encaminador de paquetes para otras estaciones. Así, aparecen las redes MANET (*Mobile Ad Hoc Network*), objeto de estudio de este proyecto final de carrera. El crecimiento del número de equipos inalámbricos y portátiles (PDAs, laptops, móbiles…) y el éxito de la tecnología WIFI, basada en el estándar IEEE 802.11, han hecho que las MANET sea un área de investigación en auge durante los años 90. En los últimos años, muchos investigadores han centrado sus esfuerzos en desarrollar nuevas propuestas de arquitecturas de red para MANET capaces de proveeer QoS (*Quality of Service*) de forma eficiente sobre y así poder proveer servicios multimedia sobre dichas redes.

Las redes MANET todavía están en proceso de evolución y quedan todavía algunos aspectos a mejorar. Se está investigando en la reducción de consumo de batería de los terminales para conseguir más autonomía, en la seguridad de

un medio muy vulnerable a ataques fraudulentos, en la utilización de un protocolo de encaminamiento adecuado para cada medio y en la mejora de la provisión de calidad de servicio.

Es en este último punto donde esta centrado el presente proyecto fin de carrera. El objetivo que se persigue es diseñar una calculadora de PSNR (*Peak Signal-to Noise Ratio*) que permita hacer un estudio de calidad de servicio (QoS) a la hora de proveer vídeos bajo demanda dentro de una arquitectura de red MANET. Una vez diseñada la herramienta, ésta se ha aplicado a las distintas versiones del protocolo de encaminamiento para redes MANET denominado *Multipath Multimedia Dynamic Source Routing* (MMDSR) diseñadas por Víctor Carrascal Frías en su tesis de doctorado [CAR09]. Con los resultados obtenidos se han llevado a cabo una serie de conclusiones acerca de las prestaciones y la QoS que proporciona cada uno de los protocolos diseñados, que hasta ahora no había sido posible realizar tan fácilmente.

Este proyecto fin de carrera ha sido estructurado en siete capítulos. Los tres primeros capítulos están dedicados a explicar la base teórica sobre la que se sustenta el proyecto. En el capítulo 1 se hace un análisis de la situación actual de las redes MANET. Posteriormente, en el capítulo 2, nos centramos en los protocolos de enrutamiento MMDSR comentados anteriormente, detallando sus características y comparándolos con *Dynamic Source Routing* (DSR) [DSR07]. En el capítulo 3 vamos enfocandonos hacía nuestro objetivo final realizando un estudio sobre los modelos de calidad de servicio existentes para redes inalámbricas en general y concretamente para redes MANET.

El capítulo 4 es donde se explica el desarrollo de la calculadora. Es el capítulo fundamental del proyecto ya que se dan las pautas del diseño y la utilización de dicha herramienta. Después, en el capítulo 5 se detallan las herramientas utilizadas para la simulación, *Network Simulator* (ns-2) [NS2] y para la creación de escenarios, *Bonmotion* [BNM05]. Con estas herramientas, en el capítulo 6 se realizan las simulaciones donde se prueba el funcionamiento de la calculadora y se profundiza en el estudio de los protocolos MMDSR. Hemos diseñado tres

escenarios posibles:

- Escenario Pequeño: Es un escenario de 200x200m que se puede comparar con un parque, un campo de fútbol o una biblioteca.
- Escenario Grande: En este caso el escenario tienen unas dimensiones mayores 400x400m, como pude ser un barrio de una ciudad, un campus universitario, etc.
- Escenario *Game*: en este escenario las dimensiones no son tan relevantes como el hecho de que tenemos más de una fuente transmitiendo simultáneamente.

Tras las simulaciones, se establece un capítulo con las conclusiones finales del proyecto y posibles campos de investigación para el futuro.

Capítulo 1: Las redes MANET

Una red móvil Ad-Hoc (*Mobile Ad-Hoc NETwork*, MANET por sus siglas en ingles) [1] es una red inalámbrica que no requiere ningún tipo de infraestructura fija ni administración centralizada y donde las estaciones necesitan incorporar la funcionalidad de enrutamiento, retransmitiendo paquetes entre aquellas estaciones que no tienen conexión inalámbrica directa como se puede ver en la siguiente figura:

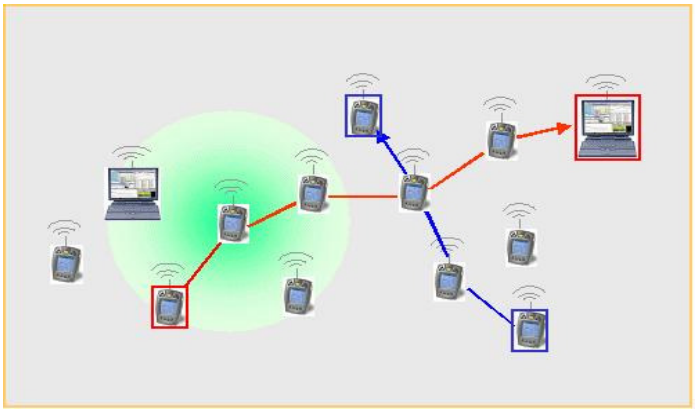

Figura 1.1 Red Ad-hoc

Tanto la estructura de la red como los nodos que la componen son altamente dinámicos y cambiantes, lo que se refleja en un enrutamiento extremadamente adaptativo. Factores a los que se debe adaptar la red son la posición de las estaciones, la potencia de la señal, el tráfico de la red, la distribución de la carga y el principal reto de las redes MANET, los cambios continuos en la topología.

Para superar estas limitaciones se hace necesario el diseño de nuevos algoritmos, protocolos y *middleware* que permitan establecer redes independientes y descentralizadas. Dichos protocolos deben cumplir el requisito de ser adaptativos, anticipando el comportamiento futuro de la red a partir de parámetros tales como el nivel de congestión, la tasa de errores, los cambios de rutas utilizadas... En posteriores secciones haremos un análisis de dichos protocolos.

Debido a que las redes MANET no requieren ningún nodo central controlando el tráfico de red, conforman un subgrupo muy significativo dentro de las redes inalámbricas. Las redes inalámbricas convencionales requieren una *Base Station* (BS), responsable de enrutar los mensajes de y para los *Mobile Nodes* (MN). Sin embargo, en una red Ad-Hoc, no se requiere ningún equipo extra a parte de dos

o más MN trabajando cooperativamente para la red. En lugar de confiar en una BS para la transmisión de mensajes, cada MN forma una red individual y reenvía mensajes entre los distintos MN. Este comportamiento adaptativo permite a la red formarse rápidamente incluso bajo condiciones adversas.

1.1. Características de las redes MANET

Algunas de las principales características de las redes MANET han sido ya mencionadas en la sección anterior y aquí se verán con más detalle:

- **Topología dinámica**: Los nodos son libres para moverse libremente a diferentes velocidades, por tanto, la red debe ser capaz de cambiar aleatoriamente y de manera impredecible.
- **Multi-hop**: El camino entre origen y destino atraviesa múltiples nodos.
- **Escalabilidad**: La red Ad-Hoc puede crecer y tener varios miles de nodos (sensores, despliegue del campo de batalla, muros de vehículos...).
- **Limitaciones energéticas**: La mayoría de los nodos dentro de la red funcionan con baterías que tienen una vida limitada y por tanto la optimización de protocolos para que hagan un uso eficiente de ella se antoja crucial dentro de las redes MANET.
- **Ancho de Banda limitado**: Los enlaces inalámbricos siguen teniendo una capacidad menor que aquellos con una estructura fija.
- **Seguridad**: La capacidad de procesamiento de los nodos y la seguridad física limitada hace que este tipo de redes sean más vulnerables a posibles ataques que las redes cableadas.

1.2 Aplicaciones de las redes MANET

Debido al dinamismo y la evolución que están sufriendo las redes MANET a lo largo de los últimos años, se han disparado el número de campos a los que se pueden aplicar:

- **Militares**: En este caso la configuración de red descentralizada supone una ventaja operativa e incluso una necesidad.
- **Sector Comercial**: Los equipos para informática inalámbrica no han estado a un precio atractivo para los grandes mercados. A medida que aumenta la capacidad de ordenadores móviles de manera uniforme, también se espera que crezca la necesidad de formación de redes ilimitadas. Ejemplos concretos serían:
    - Operaciones de rescate en zonas remotas.
    - Situaciones donde la cobertura local debe ser desplegada rápidamente en un sitio de construcción.
    - Acceso inalámbrico público en zonas urbanas.
- **Nivel local**: Las redes MANET que enlazan ordenadores portátiles y PDA (*Personal Digital Assistant*) podrían usarse para difundir información entre los participantes de una conferencia.
- **Redes Domésticas**: Los dispositivos pueden comunicarse para intercambiar información, tal como audio/vídeo, alarmas, y actualizaciones de configuración. Aplicaciones de mayor alcance en este ámbito podrían ser las redes más o menos autónomas de robots domésticas, interconectados que limpian, lavan los platos, cortan el césped, realizan vigilancia de seguridad, y otras labores parecidas.

- **Redes de Saltos Múltiples**: Este tipo de redes se llaman *censoras*. Un ejemplo sería la monitorización medioambiental, en este caso las redes se podrían utilizar para predecir la polución del agua o para alertar con tiempo de la aproximación de un tsunami.

1.3. Ventajas de las Redes MANET

La principal ventaja de las redes MANET es la no dependencia de routers: cada equipo es capaz de producir, consumir y encaminar paquetes de datos; cada nodo será capaz de acceder por sus propios medios solo a los nodos más próximos, confiando en la colaboración de sus iguales para llegar a todos los demás (como se ve en la *fig. 1.2*.). Es un tipo de red que se despliega inmediatamente donde no hay dependencia de un único punto de fallo. Además los nodos típicamente están alimentados por baterías y forman un grafo arbitrario cuya topología puede cambiar rápidamente.

Por otro lado, pueden constituir un sistema autónomo o, adicionalmente, uno o varios de sus nodos pueden estar conectados a otra red como Internet y hacer de pasarela (*gateway*) para los demás. Por tanto, ya no es necesaria la cobertura para todos los equipos.

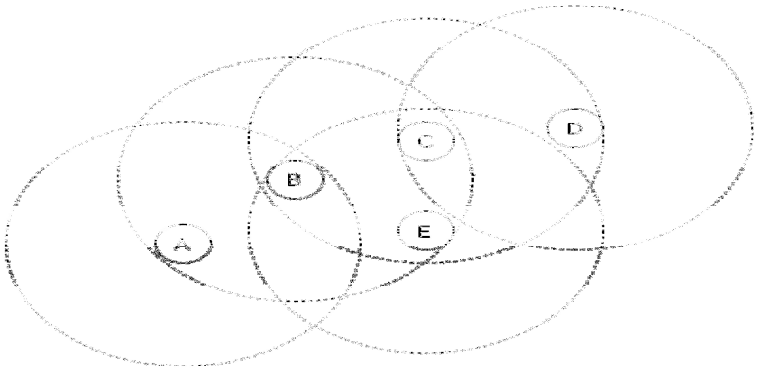

Figura 1.2 Cobertura Nodos en una red MANET

1.4. Retos de las Redes MANET

Este tipo de redes se encuentran en continuo cambio y extensión. Muchos son los retos que se afrontan a la hora de la consolidación de este tipo de redes tan dinámicas. En este proyecto nos ocupamos de resaltar los aspectos más significativos a los que investigadores de todo el mundo están dedicando sus esfuerzos:

- **Routing.** Es una cuestión muy comprometida siempre que hablamos de redes descentralizadas, con frecuentes cambios de topología, congestión de la red... Por ello, los esfuerzos se centran en encontrar protocolos que se adaptan a las situaciones "especiales" dentro de las que este tipo de redes se engloban. En este proyecto se verá en detalle el protocolo MMDSR, diseñado a partir del DSR en la tesis de Víctor Carrascal Frías [CAR09] dentro de la cual se encuadra el presente proyecto.

- **QoS.** Aunque inicialmente las redes MANET fueron diseñadas para entornos bélicos o de desastres naturales, la evolución de las tecnologías multimedia y el interés comercial que estas despiertan han relanzado los esfuerzos por establecer unos criterios de QoS sobre este tipo de redes. Más adelante haremos un estudio exhaustivo de los criterios utilizados para QoS en vídeos bajo demanda.

- **Escalabilidad.** En la mayoría de los casos, cuando se envía un paquete dentro de una red MANET, éste alcanza su destino usando enlaces *Multi-Hop.* Añadido al problema de *Multi-Hop*, surge un problema de direccionamiento al crecer el número de nodos.

- **Optimización del uso de baterías.** Para maximizar el tiempo de vida de una red Ad-Hoc es esencial prolongar la vida de cada nodo individual, minimizando el consumo de energía en cada solicitud de comunicación. Además un protocolo que emplee de manera eficiente la energía debe

tener una tasa de energía consumida en cada nodo equitativamente distribuida y al mismo tiempo una energía total para cada solicitud minimizada.

- **Seguridad.** La seguridad es una cuestión que puede condicionar el desarrollo y la implantación de redes MANET desde un punto de vista comercial e incluso militar. Los esfuerzos actuales están centrados en cuatro aspectos fundamentales:

    - Los enlaces inalámbricos sobre los que se sustentas las redes MANET son susceptibles a escuchas, mensajes repuesta o mensajes de distorsión que violan la confidencialidad.
    - Los nodos se pueden encontrar en un entorno hostil (por ejemplo, una batalla) con una relativa protección física. En estos casos habría que considerar no solo ataques desde fuera de la red sino también desde nodos en situación comprometida dentro de la red.
    - La red es muy dinámica, existen cambios continuos en los nodos que forman parte de ella por lo cual la confianza se ve comprometida. Por ejemplo, una solución pensada en términos de dominios estáticos no podría ser adoptada.
    - La escalabilidad de la red hace que el sistema de seguridad se deba adaptar a un crecimiento continuo de estas.

1.5 Protocolos de Encaminamiento

1.5.1. Encaminamiento Convencional vs Encaminamiento MANET

En general todo lo que conocemos sobre protocolos de redes convencionales se puede aplicar para redes MANET, lo que supone una gran ventaja. En situaciones que no son muy complejas, podríamos aplicar cualquiera de los protocolos convencionales de Internet, ya que son capaces de funcionar sin una configuración inicial, se adaptan a cambios en la topología y ofrecen múltiples caminos para un destino. Sin embargo, es necesaria una adaptación ya que el peor comportamiento de los protocolos convencionales se produce cuando los nodos se mueven con frecuencia cosa que es práctica habitual en redes MANET.

En Internet el encaminamiento se hace a partir de direcciones. Para las redes MANET inicialmente se intentó algo similar pero resultó inadecuado por ser muy costoso; en redes de este tipo no hay relación entre la dirección de red y la ubicación física, por tanto no sirve de nada agregar prefijos en las direcciones y el problema de la escalabilidad es un aspecto más a considerar, las tablas de enrutamiento pueden tener un tamaño inmanejable al exigir una entrada por dirección y no por grupo de direcciones.

Otra cuestión es el tráfico de red: en estas redes tenemos muchos nodos que se mueven, aparecen y desaparecen, pudiendo generar muchos mensajes de control. Hay que encontrar un equilibrio de tal manera que no se consuma todo el ancho de banda de la red con la señalización.

Además los mecanismos deben ser incrementales: no sería nada eficiente que un cambio en un nodo provocara que se tuvieran que recalcular los datos para toda la red.

Por todo ello, los protocolos de encaminamiento convencionales no son válidos para redes MANET y exigen una nueva clasificación:

- **Reactivos**: Las rutas se descubren en el momento que se necesitan.
- **Proactivos**: Cada nodo dispone de rutas hacia el resto de los nodos de la red de forma permanente.
- **Híbridos**: Mezclan ideas de los dos grupos anteriores.

1.5.2. Protocolos Reactivos

Son también conocidos como *bajo demanda* puesto que no intentan conocer todas las rutas; solo las buscan en el momento en que son necesarias lo que reduce los costes. Para su diseño se siguen las siguientes premisas:

- Es asumible una latencia alta para el primer paquete hasta que se halla la ruta.
- El número de rutas que puede interesar es relativamente bajo respecto al número de rutas posibles.
- El tráfico local es lo más importante, la mayoría de los caminos tienen pocos saltos.
- Las rutas cambian, pero una vez descubierta, puede ser usada varias veces antes de que pierda validez.

A este grupo pertenecen:

- *Ad Hoc On Demand Distance Vector Routing* (AODV), 2003 [AOD03]
- *Associativity-Based Routing* (ABR), 1996 [ABR96].
- *Cluster-Based Routing Protocol* (CBRP), 1999 [CBR99].
- *Dynamic MANET On demand* (DYMO), 2005 [DYM05].
- *Dynamic Source Routing* (DSR), 2004 [DSR07].

- *Location-Aided Routing* (LAR), 1998 [LAR98].
- *Temporally Ordered Routing Algorithm* (TORA), 2001 [TOR01].

A su vez, los protocolos reactivos se pueden dividir en dos grandes grupos: encaminamiento desde la fuente (*source routing*) y encaminamiento salto a salto (*hop-by-hop*) o punto a punto (*point-to-point*).

1.5.2.1. Encaminamiento desde la fuente

Cada paquete de datos transporta en su cabecera la ruta que debe seguir para llegar a su destino. En la cabecera del paquete encontramos una lista con todos los nodos por los que debe pasar para llegar a su destino. Los nodos intermedios solo tienen que mirar dicha cabecera para saber por dónde deben reenviar ese paquete.

Los nodos no necesitan mantener una tabla de rutas, ya que dicha ruta se encuentra en la cabecera de cada paquete. La parte negativa es que son protocolos difícilmente escalables, ya que a medida que aumenta el número de saltos que debe dar un paquete para llegar a su destino, también lo hace la cabecera de los paquetes. Además, la ruta viene establecida desde el origen, cosa que provoca una mala adaptación a los rápidos cambios en la topología de la red.

Entre los protocolos basados en fuente encontramos: DSR y CBRP.

1.5.2.2. Encaminamiento salto a salto

En este caso en la cabecera solo encontramos la dirección destino y el próximo salto a seguir, de forma que cada nodo intermedio deberá consultar en su tabla de encaminamiento por dónde debe enviar el paquete.

Una de las ventajas con respecto a los protocolos basados en fuente es la disminución de las cabeceras de los paquetes. También cabe destacar que cada nodo actualiza su tabla de encaminamiento de forma continua e independientemente de forma que el encaminamiento de los paquetes se realiza en cada nodo intermedio según el estado actualizado de la red y así las rutas pueden adaptarse mejor a la topología cambiante de estas redes. Los nodos no conocen las rutas hacia todos los destinos posibles de la red, pero si que conocen las rutas de las comunicaciones que están siendo utilizadas en cada momento.

En contra podemos destacar que cada nodo necesita actualizar su tabla de encaminamiento de manera constante mediante mensajes de señalización con sus nodos vecinos.

Pertenecen al grupo de los protocolos reactivos basados en encaminamiento salto a salto: AODV, ABR, DYMO, LAR y TORA.

Hay que destacar también que tanto DYMO como TORA pueden trabajar como protocolo reactivo como proactivo.

1.5.3. Protocolos Proactivos

Fueron los primeros en aparecer, se usan desde los 90. Guardan información en todo momento sobre las rutas hacia todos los nodos, lo que resulta muy costoso. Además consume mucho ancho de banda en la transmisión de las tablas de rutas que aparecen y desaparecen, cuando es posible que el nodo no las necesite. La principal ventaja es que cuando una aplicación necesita enviar

datos la latencia es baja porque la ruta ya está establecida. En general estos protocolos están indicados cuando:

- Cuando las rutas deben estar disponibles lo antes posibles.
- El número de rutas que se necesitan es relativamente alto respecto al número de rutas posibles.
- La longitud de las rutas es indeterminada.

A este grupo pertenece:

- *Adaptative Distance Vector* (ADV), 2000 [ADV00].
- *Destination-Sequenced Distance Vector* (DSDV), 1994 [DSD94].
- *Distance Routing Effect Algorithm for Mobility* (DREAM), 1998 [DRE98].
- *Hierarchical State Routing* (HSR), 2000 [HSR00].
- *Multimedia support in Mobile Wireless Networks* (MMWN), 1998 [MMW98].
- *Optimised Link State Routing* (OLSR), 2003 [OLS03].
- *Source-Tree Adaptaive Routing* (STAR), 1999 [STA99].
- *Topology Dissemination Based on Reverse-Path Forwarding* (TBRPF), 2004 [TBR04].

1.5.4. Protocolos híbridos

Este grupo de protocolos aprovecha las ventajas de los protocolos reactivos y proactivos. Normalmente, los nodos de la red se agrupan en grupos o zonas. Para descubrir las rutas hacia nodos cercanos se utiliza un encaminamiento proactivo y para las rutas hacia nodos lejanos se utiliza un encaminamiento reactivo.

Encontramos en este grupo:

- *Scalable Location Update Routing Protocol* (SLURP), 2004 [SLU04].
- *Zone-based Hierarchical Link State* (ZHLS), 1999 [ZHL99].
- Zone Routing Protocol (ZRP), 2002 [ZRP02].

1.6. Formato MPEG (*Moving Pictures Expert Group*)

Como el propio nombre del proyecto indica, estamos tratando el tema de los vídeos bajo demanda a través de redes MANET. Dentro de este contexto se hace indispensable hacer un recorrido por el formato de vídeo que se va a emplear durante todo el estudio.

El formato MPEG [MPG] es un método éstandard para transmitir vídeos digitales y audio en un formato comprimido usando menos ancho de banda que el tradicional método analógico. MPEG-2 está convirtiéndose en el éstandard de referencia en el mundo de la televisión digital. Este método arregla muchos de los problemas inherentes a MPEG-1, como la resolución, escalabilidad y manejo de vídeos. Todo ello permite mejores imágenes (calidad de estudio e incluso HDTV – *High Definition TV*) y permite también a múltiples canales con diferentes tasas de bits ser multiplexado en un solo flujo de datos. Fue oficialmente adoptado por ISO (*International Organization of Standardization*). Es menos potente que el MPEG-4, pero también requiere menos potencia de CPU.

MPEG-2 codifica vídeos formados por tres tipos de cuadros (*frames*):

- *Intra-Code Pictures* (I-Pictures)
- *Predictive-Code Pictures* (P-Pictures)

- *Bidirectionally-predictive-coded Pictures* (B-Pictures)

Estos tres grupos de *frames* se combinan para formar un GoP (*Group of Pictures*), normalmente con entre 4-20 *frames* cada uno, I, P y B *frames* siguen un único patrón de *frames* en un vídeo, el cuál es repetido cada GoP.

Las *frames* I tienen especial importancia. Poseen la capa base y determinan la calidad del vídeo. Además llevan la información más importante para llevar a cabo el proceso de decodificación en el lado del receptor, de hecho, son imprescindibles. El GoP puede ser codificado solo con este tipo de *frames*. Las *frames* P y B proporcionan mejoras de capas, para que la granulabilidad fina que se pueda alcanzar. Llevan diferente información de procesado y sobre las siguientes *frames*. Un GoP típico lleva 15 *frames*, donde P y B podrían seguir a I como sigue IBBPBBPBBPBB. Sin embargo, el Standard es bastante flexible y permite distintas estructuras para el GoP. En la *fig. 1.3*. podemos ver la estructura de un GoP así como la relación de dependencia entre *frames* al decodificar.

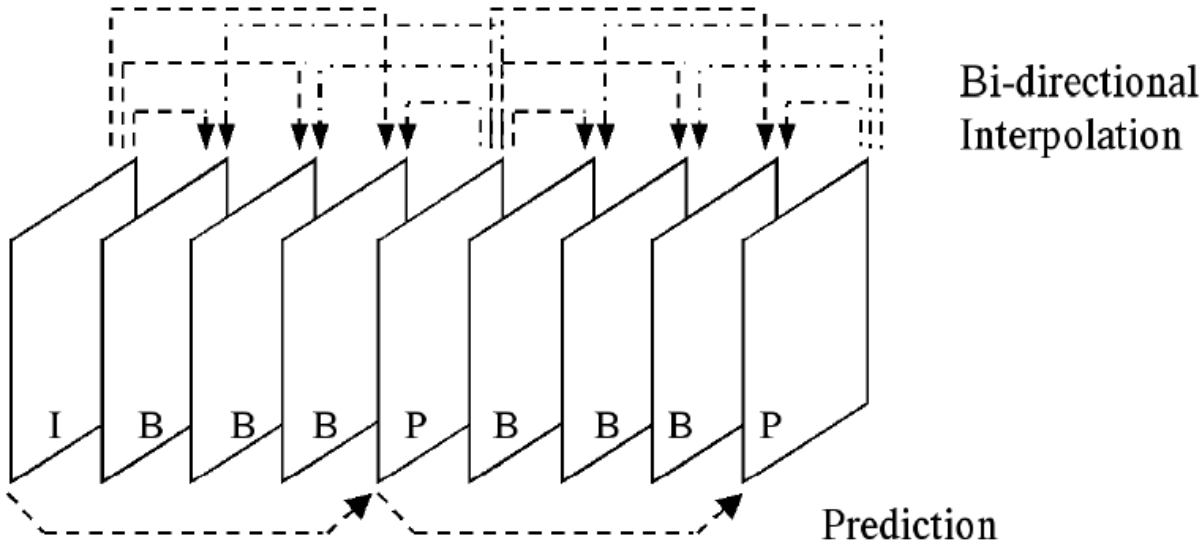

Figura 1.3 Estructura de un GoP codificado en MPEG-2

Las características relacionadas con cada una de las *frames* deben ser tenidas en cuenta cuando se haga un estudio de QoS en servicios de flujos de vídeo. Por ejemplo se pueden asignar diferentes prioridades a las *frames* de vídeo de acuerdo con su relevancia dentro del flujo de vídeo. De esta manera, las *frames* I obtendría la mayor prioridad mientras que las *frames* B se llevarían la peor.

Capítulo 2:
MMDSR
(*Multipath Multimedia Dynamic Source Routing*)

En este capítulo se describen las distintas versiones del protocolo utilizado en este proyecto. Para ello partiremos del análisis del protocolo *Dynamic Source Routing* (DSR) [DSR07], sobre el cual está basado el *Multipath Multimedia Dynamic Source Roturing* (MMDSR). Este protocolo ha sido desarrollado por Víctor Carrascal Frías en su tesis [CAR09] y su código se encuentra accesible. Acabaremos viendo la versión con más prestaciones que permite un comportamiento eficaz y seguro en la transmisión de vídeo.

El objetivo que perseguimos con este capítulo es tener una visión objetiva del funcionamiento y de las prestaciones de las distintas versiones del MMDSR. Posteriormente, cuando lleguemos a la fase de simulaciones, se probaran la utilidad de nuestra calculadora *Peak Signal-to Noise Ratio* (PSNR) bajo escenarios en los que se emplearán estas versiones del MMDSR y con ello podremos dar un paso más en estudio de este protocolo, desde el punto de vista de la distorsión entre señales de vídeo.

MMDSR es un protocolo basado en DSR (*Dynamic Source Routing*) con el cual es posible mejorar la comportamiento de los servicios de video-streaming aplicando algoritmos de capa cruzada y técnicas de enrutamiento multicamino. El desarrollo del *framework* está basado en un diseño de capa cruzada, por tanto, la colaboración entre las distintas capas del protocolo es imprescindible para alcanzar los objetivos de *Quality of Service* (QoS).

2.1. DSR (*Dynamic Source Routing*)

Es una de los protocolos bajo demanda (*Reactivos*) más puros. Las rutas se buscan solo cuando se necesitan y es el nodo origen quien las fija completamente con lo que se reduce considerablemente el *Overhead*. Así, cuando un paquete sale del nodo origen ya conoce la lista completa de los nodos por los que pasará para llegar a su destino. Por tanto, garantiza la ausencia de bucles. Usando DSR, la red es autoorganizable y autoconfigurable, sin necesidad de infraestructura ni administración añadida. Ha sido diseñado para trabajar con redes de hasta 200 nodos y para trabajar bien incluso con una alta movilidad de los nodos.

Es un protocolo situado en el nivel de red por ser el único sitio por donde de forma natural puede haber nodos con distintos interfaces y de distintos tipos. El protocolo está formado por dos mecanismos:

- Descubrimiento de rutas (*Route Discovery*): El nodo fuente quiere enviar un paquete al nodo destino y no conoce la ruta por tanto la busca.
- Mantenimiento de rutas (*Route Maintenance*): Cuando un nodo está intercambiando paquetes con un nodo destino y le llega un mensaje de error indicando que esa ruta está rota, si conoce una ruta alternativa la usa, si no, lanza un nuevo descubrimiento de ruta.

En los siguientes apartados veremos más detalladamente el funcionamiento de los mecanismos utilizados por el DSR.

2.1.1 Descubrimiento de ruta

La mayor parte de las veces, cuando un nodo deba enviar un paquete, tendrá la ruta en caché. Si no es así, la buscará. Para ello inundará la red con una petición de ruta (*Route Request*), que tendrá un identificador (*Request ID*).

Cuando un nodo reciba una serie de peticiones, si se refiere a él mismo, responderá al nodo que la originó con una paquete de asentimiento (*Route Reply*). En otro caso, volverá a transmitir la petición, a menos que sea la segunda vez que lo reciba en cuyo caso la descartará para evitar paquetes viajando por la red formando bucles infinitos. El *Route Request* contiene una lista con el historial de la ruta que ha recorrido, una información muy valiosa que será almacenada por los nodos.

Normalmente los enlaces son bidireccionales, con lo que la respuesta sigue exactamente el camino inverso a la petición.

Mientras un nodo busca rutas para los paquetes a enviar, los almacena con una marca de tiempo en una cola denominada *Send Buffer.* Si transcurrido cierto tiempo no llega una ruta, se envía una nueva solicitud de ruta. Tras varios intentos sin éxito, se descarta el paquete. También deben tenerse en cuenta que si hay una petición en curso y llega un nuevo paquete para el mismo destino, no debe generarse una nueva petición.

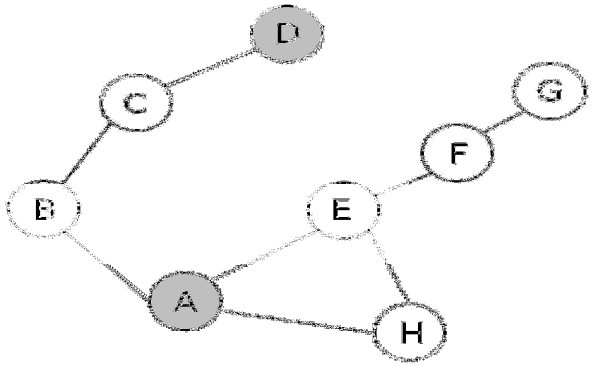

Figura 2.1 Grafo de una red Ad-Hoc

Supongamos una red como la de la *Fig 2.1*. El nodo A dispone de alguna tecnología inalámbrica que le permite alcanzar B, E, H pero no los nodos más lejanos. B llega hasta A y C, etc. Si A quiere enviar un paquete al nodo D y no sabe por donde encaminarlo, debe hacer un descubrimiento de ruta que podemos describir así:

- A transmite una petición de ruta para D a todos sus vecinos. Si quien lo recibe no es D, reenvía la petición. Esta es la técnica de *encaminamiento por inundación*.
- En cada reenvío del datagrama, cada nodo añade su propia dirección, de tal forma que cada registrada la ruta por donde ha ido pasando (*Fig 2.2)*. Para controlar la inundación, cada petición tiene un identificador único, de forma que cada nodo lo reenvía solo una vez.
- Si la petición llega a D, éste extrae la ruta del datagrama (ABCD) y se la envía a A usando un paquete *Route Reply.* Usando encaminamiento en

origen éste paquete llegará a A usando el mismo camino por el que llegó a D (*Fig 2.3*)

- Una vez que A conoce la ruta para D, la incluirá en todos los paquete que le envíe, determinando de esta forma el encaminamiento de origen.

2.1.2. Mantenimiento de la ruta

Cada nodo se hace responsable de que el paquete que recibe sea reenviado con éxito al siguiente nodo en la ruta. Así, en la *Figura 2.1*, cuando B recibe un paquete de A, se asegura que éste llegue a C, y a esta política se la conoce como *transmisión fiable*. Puede emplearse un asentimiento (*ACK*) pasivo: sí el nodo A envía algo a B y luego percibe que B lo reenvía a C puede tener la certeza de que no hubo problemas. Si este mecanismo no está disponible, se activa una *Flag* en la cabecera pidiendo asentimiento explícito; en caso de que el enlace sea bidireccional, la respuesta volverá por el mismo camino, si no lo es, será necesario encontrar una nueva ruta.

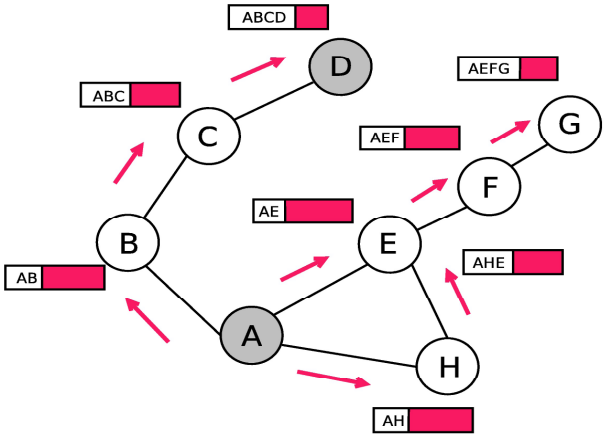

Figura 2.2 Inundación de los paquetes de petición de ruta

Si mientras A envía sus paquetes a D, C se mueve fuera del alcance de B, este

lo descubrirá y se lo comunicará a A con un mensaje de error (*Route Error)* indicando que esta ruta no es válida. El protocolo es *Best Effort* por lo que en este caso no se transmiten nuevas peticiones de ruta ni se reintenta el reenvío por caminos alternativos. Si procede, esta funcionalidad la ofrecerán protocolos de nivel superior.

La siguiente vez que A tenga que enviar un paquete a D, si conoce una ruta alternativa la usará. En otro caso lanzará otro *Route Request.*

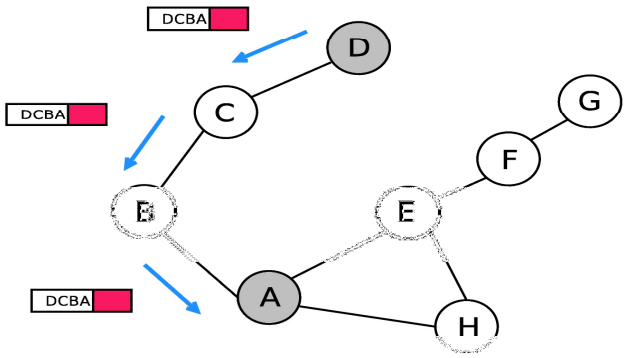

Figura 2.3 Vuelta al origen del paquete con respuesta de ruta

2.1.3. Ventajas e inconvenientes de DSR

Podemos destacar las siguientes características del DSR:

Ventajas:

- Simple y eficiente.
- Pocos mensajes de control.
- No requiere *broadcasts* periódicos de información de encaminamiento.
- *Overhead* reducido. Se gestionan sólo las rutas entre nodos que desean comunicarse.

Inconvenientes:

- Alta latencia en la búsqueda de nuevas rutas.
- Poco escalable. La cabecera aumenta mucho con el número de saltos.
- Alta probabilidad de colisión de los *Route Requests*. Un nodo intermedio puede corromper la información de encaminamiento si envía un *Route Reply* utilizando una caché obsoleta.

Podemos concluir entonces que el DSR funcionará bien en redes no muy cambiantes, lentas o no muy dinámicas y en redes no muy grandes. No funcionará bien en redes muy cambiantes ya que al tener cada paquete la ruta a seguir desde que es enviado por el nodo origen, no es capaz de adaptarse a los cambios de la red. Por otro lado, si la red es muy grande y el número de nodos que tiene que atravesar cada paquete para llegar a su destino es muy grande, las cabeceras de los paquetes aumentan demasiado y se reduce la eficiencia del protocolo.

2.2. MMDSR (Multipath Multimedia Dynamic Source Routing)

MMDSR es un protocolo de capa cruzada específico para servicios de *streaming* de vídeo con capacidad de encaminamiento multicamino diseñado por Víctor Carrascal Frías en su tesis realizada en el Departamento de Ingeniería Telemática de la Universidad Politécnica de Cataluña [CAR09].

Para la creación del MMDSR se ha partido del Standard DSR como enrutado básico. Todas las decisiones (selección de ruta) y operaciones (cálculo de parámetros) se realizan desde el origen y dependen del estado de la red. El vídeo es distribuido usando RTP/RTCP (*Real Time Protocol*/ *Real Time Control Protocol*) [RTC96] sobre UDP como protocolo de transporte.

La arquitectura sobre la que se va a realizar el análisis no mantiene más de 3

caminos simultáneos ya que gracias a la experimentación se ha comprobado que no se mejora aumentando el número, en técnicas multicamino.

En los siguientes puntos se va a proceder a explicar las principales características y el funcionamiento del protocolo MMDSR y sus dos versiones iniciales, *static*-MMDSR (*s-MMDSR*) y *adaptative*-MMDSR (*a-MMDSR*).

2.2.1. Principales características del protocolo MMDSR

2.2.1.1. Multicamino

En general, los algoritmos de encaminamiento para redes Ad-hoc tienen la habilidad de descubrir más de un camino, si es que existe. Esto se debe principalmente a la característica de amplia difusión, *broadcast*, de los interfaces inalámbricos y a los mecanismos de inundación usados para el descubrimiento de caminos. Los protocolos de encaminamiento monocamino de redes Ad-hoc activan y mantienen un único camino para la transmisión de datos entre emisor y receptor, usualmente el camino más corto o el que optimiza otra métrica o conjunto de métricas.

La diversidad de caminos en redes Ad-Hoc hace posible que se puedan ejecutar aplicaciones que requieren un mínimo de recursos de red, los cuales no puedan ser suministrados por un solo camino (tales como el ancho de banda, el porcentaje de pérdidas de paquetes o el tiempo de retardo extremo a extremo). Esto se logra transmitiendo simultáneamente los paquetes de datos a través de varios caminos. Es preferible que los caminos sean óptimamente disjuntos entre sí (que tengan el mínimo de nodos comunes) de manera que se reduzca la probabilidad de tener puntos críticos entre el nodo fuente y el nodo destino.

Además, el encaminamiento multicaminos puede potenciar el uso de mecanismos de balanceo de carga y de suministro de de QoS. Por otra parte, el

descubrimiento de múltiples caminos óptimamente disjuntos entre un nodo fuente y un nodo destino puede robustecer la red, facilitando así la recuperación de rutas en caso de fallos, así como el reforzamiento de la seguridad de las transmisiones.

Pero, a la vez, los protocolos multicamino tienen algunos inconvenientes: el mantenimiento de varias rutas cuesta más recursos de red y de nodo, lo que puede estar un problema importante en las redes Ad Hoc que tendrían que ahorrar estos recursos. Además, el uso de caminos totalmente diferentes puede ocasionar problemas a nivel de la interfaz física.

2.2.1.2. Orientado a aplicaciones de vídeo en tiempo real

Específicamente, los servicios de tiempo real necesitan especial atención debido a que la naturaleza dinámica de estas redes hace difícil aplicar una gestión tradicional de QoS.

Dado que la provisión de QoS no depende de una única capa de red sino de un esfuerzo coordinado desde todas las capas, se hace necesario desarrollar soluciones dinámicas basadas en una aproximación de capa cruzada "*cross-layer"* que tenga en cuenta las diferentes especificaciones de las redes Ad-hoc. Una arquitectura apropiada para ofrecer QoS en redes Ad-hoc debe asegurar la cooperación entre todos los componentes relacionados con la provisión de QoS (por ej.: señalización, encaminamiento y mecanismos de acceso al medio (MAC, *Medium Access Control*).

El algoritmo del protocolo MMDSR es un algoritmo capaz de proporcionar QoS a aplicaciones de *vídeo-streaming* en las redes Ad-Hoc.

2.2.2. Funcionamiento del protocolo MMDSR

Se parte de una transmisión de *streaming* de vídeo teniendo lugar entre dos nodos. MMDSR periódicamente descubre rutas disponibles entre la fuente y el destino usando DSR. Del mismo modo, MMDSR percibe el estado global de las rutas disponibles mandando periódicamente paquetes de monitorización.

Antes del comienzo de la transmisión del vídeo un mensaje sonda *Probe Message* (PM) es enviado a través de cada uno de los caminos descubiertos por el mensaje "*Descubriendo rutas*" de DSR. Entonces, en el destino, un *Probe Message Reply* (PMR) es generado para recoger la información con respecto a la calidad del enlace. Esta información será analizada en el origen. Un puntuación es asignada a cada uno de los caminos y estos serán clasificados de acuerdo a un patrón previamente establecido. Estos caminos se utilizarán hasta la próxima iteración del algoritmo. Después de este proceso, el origen selecciona tantas rutas como sean necesarias para su esquema multicamino.

Este algoritmo se ejecuta en iteraciones cada 10 segundos. En cada iteración se computan los valores de dos tipos de parámetros que permiten la clasificación de las rutas: parámetros que dependen únicamente de la red y parámetros que son los requisitos del cliente. Con estos valores se define primero la mejor ruta, después la segundo mejor y por la tercera mejor.

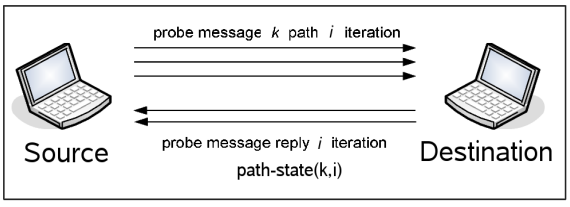

Figura 2.4 Paquetes PM y PMR

2.2.3 *Hello Messages*

Periódicos *Hello Messages* (*HM*) ayudan al cálculo de dos nuevos parámetros de QoS llamados *Reliability Metric (RM)* y *Mobility Metric (MB)* que serán analizados posteriormente. HM es enviado una vez por segundo por los nodos que se encuentran en el camino de transmisión del vídeo para monitorizar la fuerza de señal de sus nodos vecinos. Cuando un HM es recibido, el nodo receptor calcula la SINCR (*Signal-to-Interference plus Noise Ratio)* mirando el paquete recibido y la adjunta al paquete respuesta *Hello Message Reply (HMR).* Se muestra el funcionamiento en la *Fig 2.5.*

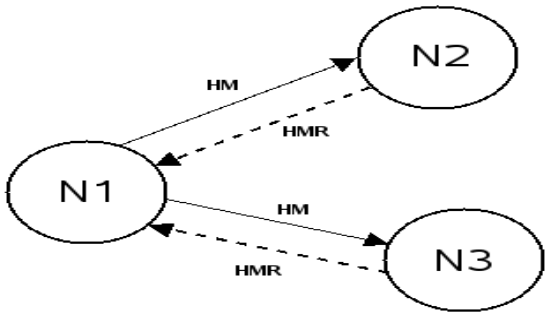

Figura 2.5 Paquetes HM y HMR entre el nodo 1 y sus vecinos

2.2.4. Parámetros QoS

Como ya se ha comentado previamente, MMDSR computa de manera dinámica una serie de parámetros QoS sobre todos los caminos posibles que unen la fuente y el destino, de manera que se clasifican en función de su estabilidad.

Estos parámetros son los siguientes:

- **Reliability Metric (RM):** Calcula la calidad general de un mismo camino realizando medidas de SINR entre nodos vecinos. El valor se obtiene como una media de las mediciones realizadas.

- **Mobility Metric (MN):** Cada nodo X detecta la potencia de su nodo vecino Y (el cúal pertenece al mismo camino entre fuente y destino) usando la transmisión de sucesivos paquetes (*Hello Messages*). Posteriormente, el nodo X calcula la movilidad relativa mirando a los nodos Y. Una vez el *Hello Message* ha vuelto a su nodo original, cada medida local es computada. Finalmente, el nodo destino hace una media aritmética de las mediciones individuales de cada nodo y obtiene un valor del grado de movilidad del camino.
- **Loss, Delay and Jitter:** Cuando el mensaje PRM vuelve a llegar al nodo fuente, éste obtiene una valor para cada uno de estos parámetros medido en cada camino.
- **Ancho de Banda:** El paquete PM también recoge el BW disponible en el enlace entre los nodos de un camino entre fuente y destino. Entonces, el destino computa el valor total para cada camino que se corresponde con el BW de un cuello de botella en el enlace. Esta información se manda de regreso usando el paquete PMR.

Una vez analizados estos parámetros tendremos que decidir cuales serán determinantes para elegir un camino. Se han elegido RM y MN porque para aplicaciones de *streaming* de vídeo mientras más eficiente y estable sea el camino mejores resultados se obtienen. Es decir, el protocolo MMDSR realiza la clasificación de los caminos en función de los valores de estos parámetros.

2.2.5. Clasificación de los caminos

Una vez determinados los parámetros anteriores para todos los caminos posibles entre un nodo fuente y un nodo destino, se hacen unos cálculos en el algoritmo. Se verifica primero que los caminos cumplen los requisitos del cliente. Todos los caminos que no cumplen estos requisitos son descartados inmediatamente. Después, según los resultados de unos cálculos, se ordenan las rutas y se seleccionan las mejores. En principio, se eligen tres como máximo.

Después, tenemos que asignar las rutas. Ya sabemos que trabajamos con vídeo. Este vídeo transmitido es MPEG-2 sin audio. El sistema envía flujos de vídeo que pueden tener diferentes calidades. Estos flujos MPEG-2 han sido codificados a partir de una secuencia de vídeo original de mayor calidad. El flujo de vídeo MPEG-2 está formado por GoPs (*Group of Pictures*) que mantienen una estructura de *frames* dada. Existen tres tipos de cuadros (*frames*):

- ***Frames* I**: el principal en la estructura MPEG-2. Son los encargados de llevar la información más relevante de las imágenes, codificando redundancia espacial. Sólo existe un cuadro I por cada GoP.

- ***Frames* P**: codifican redundancia temporal y sólo llevan información relativa a las diferencias respecto al último *frame* I. Su tamaño es un 20% respecto al tamaño medio de un *frame* I.

- ***Frames* B**: también codifican redundancia temporal y sólo llevan información relativa a las diferencias respecto al último *frame* P. Su tamaño es un 10% respecto al tamaño medio de los *frame* I. Son menos importantes que los *frame* P para el proceso de decodificación de la secuencia de vídeo.

Entonces, en función del número de camino que tengamos, asignamos los caminos de la siguiente manera:

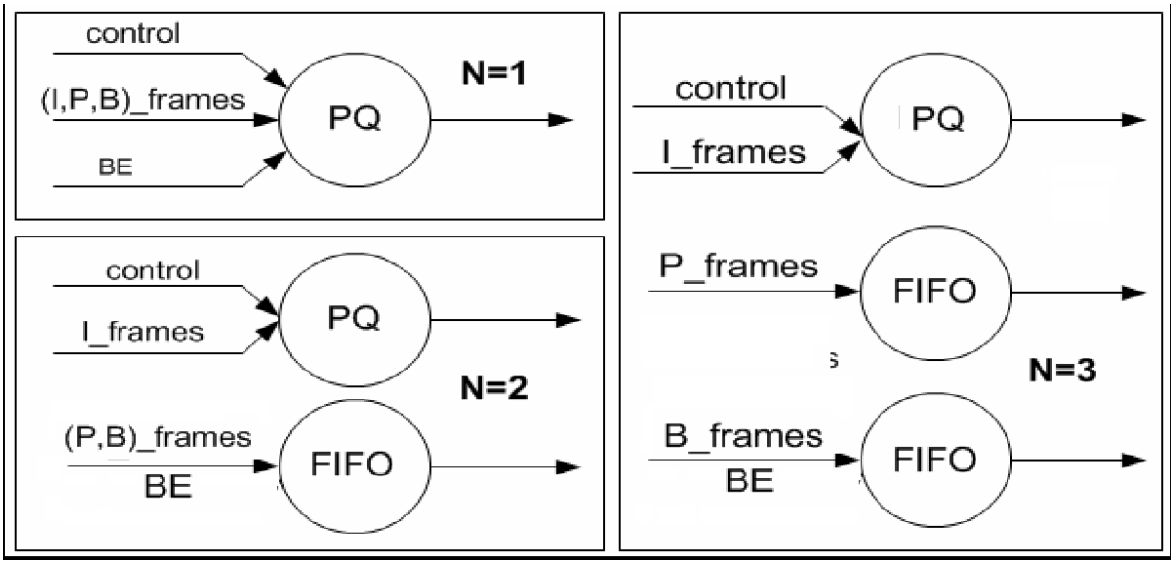

Figura 2.6. Asignación de los caminos

Las *frames* principales son del tipo I, que son la única clase de *frames* que pueden ser auto decodificados. Al ser las *frames* más importantes, las enviamos por el mejor camino obtenido por el algoritmo en la última iteración. El objetivo principal es el de proteger estas *frames* I otorgándoles alta prioridad, debido a su influencia directa en la calidad que finalmente recibirá el usuario. Las otras *frames* tipo P y B no pueden ser auto decodificadas por sí mismas ya que necesitan el soporte de las *frames* I. Las *frames* P y B mejoran la calidad del vídeo de la secuencia decodificada en el lado del receptor.

En la *Fig 2.6*, se observa qué tipo de *frames* se envían por cada camino. Estos tres esquemas están diseñados de tal manera que siempre se envían las *frames* más importantes a través de los mejores caminos. La figura muestra la estructura de colas usadas, tanto las que usan colas con prioridades (PQ, *Priority Queue*) como las FIFO (*First In First Out*) para N=1, 2, 3 caminos. Si sólo hay un camino disponible (N=1) el tráfico se organiza usando una PQ para transmitir el tráfico de control (ej. datos de encaminamiento), el flujo de vídeo y datos BE (*Best Effort*). Si seleccionamos dos caminos, el flujo I y el tráfico de control se envían a través del mejor camino, mientras que los flujos B, P y BE a través del peor camino. Si disponemos de más caminos, los siguientes flujos de

vídeo mejorados son enviados a través de los caminos que han sido ordenados por el algoritmo. Debido a la naturaleza dinámica de las redes Ad Hoc, es posible que en algunos casos no haya suficientes caminos diferentes para aplicar el esquema multicamino con el que estemos trabajando entre fuente y destino. En estos casos, deberemos aplicar el esquema multicamino equivalente al número de caminos que se hayan encontrado.

2.2.6. Estático vs Adaptativo

En esta sección vamos a detallar las dos primeras versiones del protocolo MMDSR: *static*-MMDSR (s-MMDSR*)* y *adaptative*-MMDSR *(*a-MMDSR*).* El comportamiento adaptativo se refleja en el dinamismo en el cálculo de parámetros de configuración así como en el período de iteración del algoritmo de enrutado MMDSR.

Las características principales del *static*-MMDSR (s-MMDSR) son:

- Funciona con un período de iteración fijo de 10 segundos. Por tanto, cada 10 segundos se producirá una nueva iteración del algoritmo, refrescando caminos, calidad y otros parámetros.
- Los parámetros de QoS son fijados cada 10 segundos lo que influye a la hora de clasificar los caminos.
- En situaciones de alto tráfico todos los caminos obtienen altas calificaciones, dificultando su distinción.
- Como aspecto positivo, al tener un período de algoritmo fijo, el tráfico de señalización es el mismo tanto para situaciones de poco tráfico como de alta carga.

Sin embargo, las características inherentes a las redes MANET, con una topología de red extremadamente dinámica, exigen un diseño que implique dinamismo también. Para alcanzar este dinamismo, el protocolo debe ser capaz de adaptarse a las diferentes situaciones de red que se pueda tener que enfrentar. Para ello la versión adaptativa del protocolo introduce una serie de novedades con respecto a la estática que se resumen básicamente en convertir en dinámico todo lo que era estático:

- Se aplica un factor de corrección dinámico para adaptar los rangos para asignar calificaciones. Este factor de corrección varía en función de los parámetros QoS recogidos en cada camino.
- El uso de este factor de corrección permite clasificar los posibles camino mas eficientemente, haciendo un uso mejor de los recursos.
- Dependiendo de la probabilidad de error de camino, el período de iteración del camino cambia dinámicamente. Para bajas probabilidades de error, tenemos altos períodos de iteración.
- Se produce menos tráfico de señalización en situaciones estables, mientras que en situaciones de alta movilidad se necesitan caminos nuevos más frecuentemente (la topología cambia más rápidamente).

Además, la versión adaptativa del protocolo introduce dos nuevos parámetros:

- ***Hop Metric* (h):** Este parámetro es una medida de la distancia y se obtiene mirando el salto más largo en la ruta y el salto más corto para cada iteración. Se asignan puntaciones más altas a las rutas más cortas. En general se prefieren los caminos cortos porque minimizan las pérdidas.
- ***Network State*:** Da información sobre el estado global de la red en cada iteración. Su valor se obtiene como la media de los valores del *Reliability Metric* en la red durante cada iteración.

2.3 Probabilidad de error de camino (PEP)

En esta sección vamos a describir la tercera versión del protocolo MMDSR que va a ser analizada en la parte de simulaciones, es llamada *adaptative*-PEP *Multipath Multimedia Dynamic Source Routing (*ap-MMDSR*).* Esta versión se basa en un modelo analítico para el cálculo de la *Probabilidad de error de camino* (PEP) y para ello es necesario modelar el movimiento de los nodos y posteriormente computar el *Link Error Probability (*LEP*)*.

El objetivo de esta mejora es hacer más preciso y simple el cálculo analítico que asiste al proceso de enrutado, permitiendo que nuevos caminos estén disponibles cuando se produzca alguna rotura en el actual. Además, el esquema de enrutada puede usar el PEP para calcular de manera proactiva el mejor camino para realizar la transmisión de vídeo.

Pasamos analizar los elementos esenciales para el funcionamiento de este protocolo.

2.3.1 Modelo para el movimiento de los nodos en redes MANET

Vamos asumir un escenario cuadrado de W metros (*Fig 2.7)* para cada lado. Los nodos se mueven basándose en un *Random Waypoint Mobility Model (*RWMM*)* y asumimos un entorno homogéneo, es decir, los nodos usan el mismo modelo de movilidad y con los mismos parámetros.

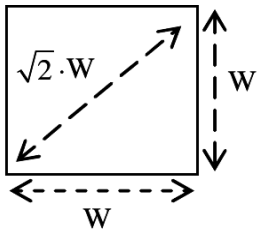

Figura 2.7 Area de la red Ad-Hoc

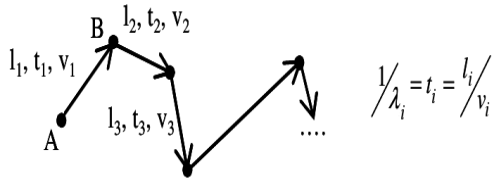

Figura 2.8 Secuencia de epochs

El movimiento de los nodos es modelado con una secuencia de intervalos (llamados *epochs*) de longitud aleatoria que se encuentran exponencialmente distribuidos con una media de $1/\lambda$. Durante cada intervalo el nodo se mueve con una velocidad y dirección constantes como se ve en la *Fig 2.8*. Al final de cada intervalo, el nodo para durante un tiempo. Posteriormente, el nodo reanuda el movimiento con otros valores de velocidad y dirección.

2.3.2 Link Error Probability (LEP)

La PEP puede ser obtenida fácilmente a partir del *Link Error Probability* (LEP). Una ruta esta compuesta por una serie de enlaces o saltos. El objetivo es encontrar una fórmula para estimar de forma analítica la PEP. De esta manera, los nodos fuentes pueden usar esta herramienta para predecir la PEP de los caminos disponibles y hacer una buena decisión de enrutado para enviar los paquetes. Además puede ayudar a los nodos fuentes a encontrar un camino adecuada antes de que se produzca una rotura en el actual. Por tanto, esta nueva propuesta trata de mejorar el comportamiento global, es decir, reducir la señalización, el retraso y el número de paquetes perdidos. Como ya sabemos, en las redes Ad-hoc las baterías son muy limitadas, por lo que esta herramienta debe ser simple y fácil de computar a costa de cierta pérdida de precisión.

El cálculo analítico de la expresión del LEP constituye un duro desarrollo numérico que no es objeto de este proyecto pero que fue realizado en la tesis de Víctor Carrascal Frías [CAR09].

2.3.3. Estimación de la Probabilidad de error de camino (PEP)

Partiendo del cálculo del LEP se puede obtener fácilmente la PEP. Como ya comentamos en el punto anterior, el desarrollo analítico se encuentra en la tesis de Víctor Carrascal Fías [CAR09] accesible para su revisión para aquellos interesados.

En el ámbito de este proyecto, preferimos centrarnos en la utilidad de estos cálculos, es decir, en como puede mejorar el proceso de enrutado el conocimiento del PEP. El procedimiento será el siguiente: cuando el PEP exceda un valor de *Threshold* determinado, el protocolo de enrutado conmutará a un camino alternativo. Este momento determinará el período de trabajo ($T_{routing}$) del protocolo de enrutado.

Para incluir estas modificaciones dentro del protocolo MMDSR tenemos que analizar como los nodos usarán esta propuesta en sus esquemas de enrutado multicamino. El proceso es como sigue:

- Tras la recepción de un mensaje *Probe Message Reply* (PMR), el nodo fuente procesa el valor de la Probabilidad de error para cada uno de los caminos disponibles, PEP_k.
- Posteriormente, cada PEP_k es comparado con un $PEP_{threshold}$ establecido.
- Solo aquellos caminos que cumplan $PEP_k < PEP_{threshold}$ son válidos para ser usados para enviar *frames* de video.

2.3.4. a-MMDSR vs. ap-MMDSR

En esta sección se discutirán las similitudes y diferencias entre los dos protocolos adaptativos descritos hasta este momento. Por un lado, recordamos que el a-MMDSR fue la primera versión que introdujo el comportamiento adaptativo en nuestro esquema, siendo capaz de configurarse por sí mismo en función del estado de la red. Especialmente el a-MMDSR configuraba dinámicamente el *Threshold* para calificar y clasificar los distintos caminos dentro del esquema de enrutado multicamino. En comparación con s-MMDSR e incluso con DSR, a-MMDSR implica una reducción en el número de paquetes perdidos y una disminución de la señalización.

Por otro lado, ap-MMDSR incluye un modelo analítico para evaluar la Probabilidad de error de camino. La PEP es usada para conocer el momento exacto en el que se debe conmutar a un nuevo camino, solo es necesario definir un *Threshold* (umbral) para el PEP y entonces aparecerá un instante de conmutación. El nodo fuente calcula este momento para cada camino cuando el esquema ya se ha establecido. Entonces, el nodo fuente selecciona aquel con el valor más bajo y este constituye el período de enrutamiento, $T_{routing}$. El algoritmo de enrutamiento es refrescado cada $T_{routing}$ *segundos.*

En la siguiente lista podemos ver las principales discrepancias entre ambas versiones adaptativas:

- ap-MMDSR calcula $T_{routing}$ usando un modelo analítico mientras que *a-MMDSR* emplea una expresión obtenida heurísticamente. Además, ap-MMDSR parece se más “listo” aunque a-MMDSR puede ser más realista.
- En segundo lugar, ap-MMDSR estima el valor $T_{routing}$ solo una vez en la fase de establecimiento del esquema multicamino, mientras que a-

MMDSR lo calcula en cada iteración del algoritmo.

- Por último, ap-MMDSR calcula $T_{routing}$ basándose en la evaluación de la PEP en cada camino mientras que a-MMDSR lo deriva del estado actual de la red. Además, a-MMDSR parece tener información más precisa aunque ap-MMDSR más simple.

En conclusión, ap-MMDSR es más convincente que a-MMDSR gracias a su elegante y simple modelo analítico. Además, ap-MMDSR actúa proactivamente, mientras a-MMDSR actúa reactivamente lo cual es interesante en término de un entorno dinámico. No obstante, a-MMDSR es más fiel que ap-MMDSR ya que usa información recogida dinámicamente para decidir el momento exacto de conmutar. Por todo ello, obtenemos una mejora en términos de señalización, a-MMDSR introduce mayor *overhead* que ap-MMDSR por lo que emplea el ancho de banda disponible de una manera menos eficiente.

2.4 Ventana Dinámica

En esta sección vamos a presentar una nueva propuesta para mejorar el comportamiento del protocolo MMDSR sobre redes Ad-Hoc. En este caso se va profundizar sobre el concepto de Ventana dinámica (CW) que ya se encuentra presentado en el estándard IEEE 802.11e [SA]. La ventana, en este contexto, es quien regula el acceso al medio compartido. El objetivo es mejorar el manejo de la ventana dinámica incluida en el estándard obteniendo variaciones más suaves de los valores del CW en cada *Access Category* (AC). Esta nueva versión del protocolo ha sido llamada d-MMDSR (*Dynamic Contention Window-MMDSR*).

2.4.1 Ventana de Contención Dinámica

Los primeros protocolos que se han propuesto para MANETs usan como criterios para la clasificación del mejor camino el número de saltos, calidad del enlace, ancho de banda y fuerza de la señal. Estimar el tiempo de vida de una ruta es útil también como métrica para los protocolos de enrutados así que los caminos con mayor tiempo de vida serán preferidos.

Recientemente muchos investigadores han centrado sus esfuerzos en algoritmos relacionas con los parámetros MAC, los cuales pueden evolucionar de manera dinámica en función de eventos ocurridos en la red Ad-hoc. En la nueva versión del protocolo (d-MMDSR), se han centrado los esfuerzos en introducir un cálculo dinámico de la CW, para cada una de las categorías de accesos, usando nuevas funciones para atenuar las transiciones suaves y evitar cambios de valores en los parámetros MAC. De esta manera, los nodos tienen altas probabilidades de tener diferentes CW y con ello decrementar el número de colisiones.

Para introducir el dinamismo en el cálculo de la ventana, se parte de los valores de ventanas que se asignan con el estándard IEEE 802.11e [SA] (*Tabla 2.1.*) y suprimimos la política de actuar en cada paso, por ejemplo reducir la CW a CW_{min} en cada transmisión exitosa.

Tabla 2.1. Límites categorías de acceso en el estándard 802.11e

	CWmin	CWmax
AC0	7	15
AC1	15	31
AC2	31	1023
AC3	31	1023

En su lugar, con la nueva propuesta se trata de suavizar la asignación de valores para la CW en cada AC de manera que sea más difícil obtener los mismos valores de CW en nodos vecinos. Después de algunas pruebas, se ha decidido que la mejor función para suavizar la asignación de CW es la parabólica, introduciendo el consiguiente cambio en el estándard (Esta adaptación fue realizada también en la tesis de Víctor Carrascal Frías [CAR09]). Gracias a esta decisión, los valores CW cambian suavemente y tienen un rango mayor de valores que en el estándard, de manera que los nodos vecinos tomarán diferentes CW con una alta probabilidad de que las oportunidades de colisión disminuyan.

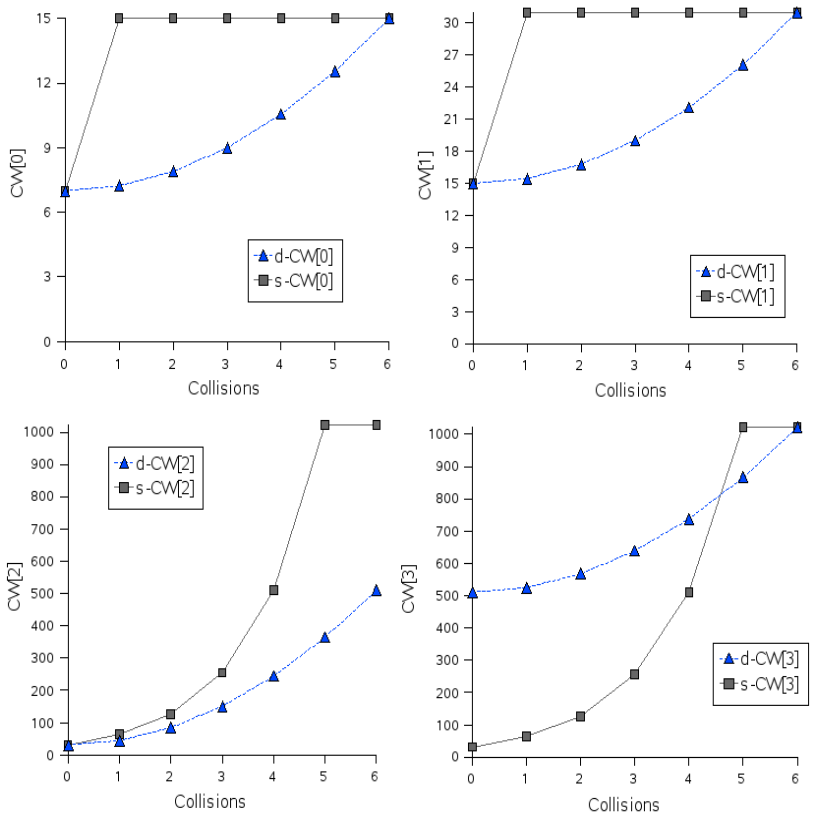

Figura 2.9 Asignación Dinámica / stática de CW vs Colisiones Consecutivas

En la figura superior, se puede apreciar como después de la primera colisión, los valores CW varían lentamente y solo cuando el número de colisiones consecutivas es muy alta (*Ejemplo inferior derecha*) los valores CW aumentan notablemente. Este comportamiento tiene como finalidad evitar incrementos innecesarios en los valores de CW cuando hay un número bajo de colisiones. En el *Ejemplo superior derecha,* después de una colisión, *d – CW [1] = 15* (d – CW indica asignación dinámica de los valores), después de dos colisiones *d – CW [1] = 16,* después de 5 colisiones consecutivas el valor es *d – CW [1] = 26.* Sin embargo, con la asignación estándard, la misma evolución nos da *d – CW [1] = 31* tras una o dos colisiones consecutivas. Este hecho indica que la falta de suave incremento puede producir largos *back-off* y retrasos altos en el acceso al medio.

Las aplicaciones multimedia son muy sensibles a los retrasos, por tanto bajos CW son preferibles para decrementar la duración de los *back-off.* Sin embargo, las oportunidades de colisión aumentan entre nodos vecinos con bajos CW. Por tanto, CW bajos son preferibles mientras el número de colisiones permanezca acotado.

2.4.2 Conclusiones para la Ventana Dinámica

La inclusión de la ventana dinámica en el esquema de enrutamiento tiene como objetivo evitar los cambios abruptos realizados por las funciones para la asignación de los valores CW, y el incremento de las colisiones cuando nodos cercanos tiene transmisiones satisfactorias de paquetes al mismo tiempo. Para ello se ha propuesto una asignación de valores CW menos abrupta para cada AC basada en una función parabólica. Además, si el número consecutivo de colisiones en un AC es mayor que 6, el CW_{max} del siguiente AC es recalculado usando dicha función parabólica de manera que la evolución de los valores del CW es más suave. Consecuentemente, para mantener las diferentes prioridades asociadas a cada AC, los valores CW_{max} y CW_{min} de los AC con más prioridad son recalculados usando sus funciones asociadas.

2.5. Teoría de Juegos

El objetivo de esta sección es explicar como aplicando la Teoría de juegos a nuestro esquema de enrutamiento podemos conseguir que los nodos comportan los recursos más eficientemente y los servicios mejoren reduciéndose el número de paquetes perdidos en entornos de alta congestión.

Teoría de Juegos es una rama de las Matemáticas Aplicadas que se usa en ciencias sociales, economía pero también en biología, ciencias políticas, informática y filosofía. La Teoría de juegos intenta capturar el comportamiento matemático en situaciones estratégicas, en las cuales el éxito a la hora de hacer elecciones depende de las elecciones de los demás. La Teoría de juegos estudia

la interacción estratégica entre individuos en situaciones llamadas *games*. Especialmente, la Teoría de juegos no cooperativa trata de analizar como cada individuo debe interaccionar con otros para conseguir sus propios objetivos. Además proporciona una serie de herramientas matemáticas para modelar la interacción entre jugadores que han de tomar decisiones en distintas situaciones.

En los siguientes puntos describiremos unos principios básicos de la teoría de juegos que se aplican a la nueva versión del protocolo MMDSR, (*g-MMDSR*) y que redefine la estrategia a seguir para transmitir los cuadros (*frames*) I, P y B.

2.5.1 Principios de la Teoría de Juegos

Un juego de estrategia *G* puede ser descrito con tres elementos principales:

- El conjunto de jugadores N
- El espacio de acción A
- El conjunto de funciones de utilidad individual

Tenemos una serie de estrategias que pueden seguir los jugadores en el desarrollo del *game*:

- **Estrategía pura**: proporciona una descripción completa de como el jugador va a jugar el *game*.
- **Estrategía mixta**: el jugador asigna una probabilidad para cada estrategia pura de manera que podrá seleccionar aleatoriamente entre la colección de estrategias puras.
- **Mejor respuesta**: produce el resultado más favorable para un jugador, tomando las estrategias de los otros jugadores como suyas.

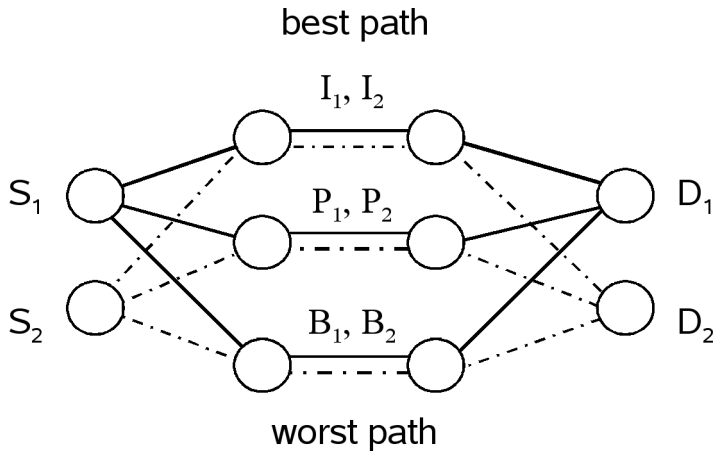

Figura 2.10 Estrategía fija usada hasta ahora para enviar recursos

En la *Fig 2.10* se puede observar como hasta ahora, se elegía en MMDSR siempre el camino mejor para las *frames* I, el siguiente mejor para las *frames* P y por último el camino peor para las *frames* B.

2.5.2. Enrutado con Teoría de Juegos

Con la Teoría de juegos se pretende ofrecer una alternativa a la política de enrutamiento comentada en el último párrafo de la sección anterior. En determinadas circunstancias, si la regla para la asignación de recursos es siempre que el mejor camino posible es para las *frames* I y todos los usuarios (fuentes de vídeo) lo usan, puede darse una congestión en ese camino, además, teniendo en cuenta que el tamaño de las *frames* I es considerablemente mayor que las P o B. Alternativamente, se puede emplear la Teoría de juegos sobre los 2 mejores caminos para transmitir *frames* I y P. Por simplicidad, las *frames* B se transmitirán siempre por el tercer camino que es el peor. Esta política será beneficiosa para los usuarios ya que en muchas ocasiones el segundo mejor camino será mejor que el primero debido a que tendrá una congestión menor.

El juego se realizará por rondas, en cada iteración de MMDSR, los usuarios tomarán una estrategía mixta. Hay cuatro posibles situaciones resultantes de aplicar esta estrategia mostrada en la *Fig 2.11*. Cada usuario elige el mejor camino para transmitir las *frames* I o P con una cierta probabilidad. Previo a jugar el juego, ambos jugadores estarían bajo las circunstancias a), en la cual las *frames* I y P serán siempre mandadas a través del mejor camino. Como alternativa, ellos pueden jugar este juego en el que surgen 3 nuevos escenarios.

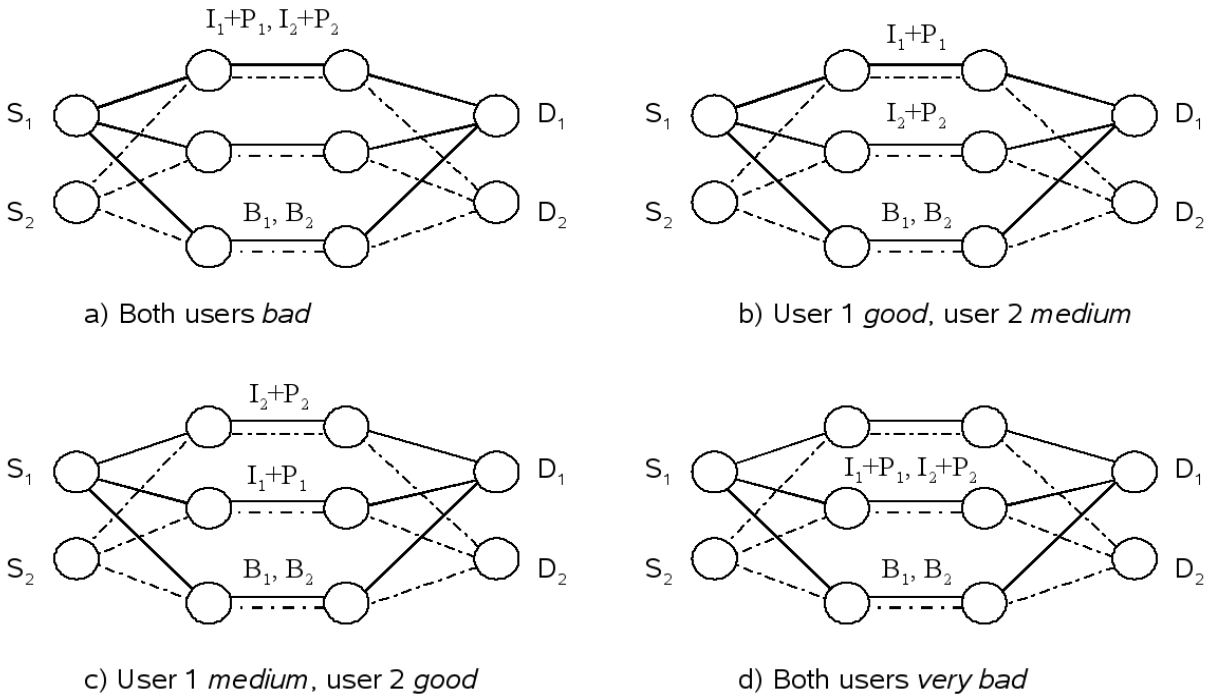

Figura 2.11 Cuatro posibles asignaciones cuando aplicamos estrategias mixtas

Bajo las situaciones b) y c), el usuario que manda las *frames* I y P a través del mejor camino nota una notable mejora en el comportamiento del vídeo, mientras que los otros usuarios notan simplemente un ligera mejora. Situaciones b) y c) en las cuales usuarios mandan sus *frames* separadamente mejoran la situación a). Situación d) es peor que la a) para ambos usuarios, ya que los usuarios mandan las *frames* I a través del peor camino.

2.5.3. Conclusiones Teoría de Juegos

En el capítulo 6, dedicado a simulaciones, tendremos la oportunidad de analizar y extraer conclusiones de escenarios reales. En este punto del proyecto, solo queremos concluir que al incluir la Teoría de juegos en el esquema de enrutamiento MMDSR permitimos a los jugadores tomar parte en la asignación de recursos. Esencialmente, en lugar de mandar siempre las *frames* I y P a través del mejor camino posible, los usuarios juegan de manera que se mandaran las *frames* I y P por uno de los dos mejores caminos dependiendo de una probabilidad.

Capítulo 3: QoS en redes MANET

El principal objetivo de este proyecto es el diseño e implentación de una herramienta calculadora de PSNR para medir la calidad objetiva del vídeo en transmisiones a través de redes MANET. La provisión de QoS es un aspecto fundamental cuando se hace un diseño de cualquier red, protocolo, servicio, ya que influye en la percepción que tiene el usuario de cómo el objeto en cuestión esta funcionando. Siempre que se lanza un nuevo producto al mercado, por ejemplo, éste ha tenido que pasar una serie de controles de QoS para garantizar un mínimo de prestaciones al usuario.

Dada la importancia de este aspecto en general y especialmente para este proyecto que nos ocupa, se ha diseñado un capítulo donde se analizaran conceptos generales de QoS, modelos, parámetros, mecanismos,... y se verán las particularidades que tiene este estudio en lo referente a redes MANET.

3.1. Conceptos generales sobre QoS

El término QoS [4] tiene una gran variedad de significados dependiendo de la perspectiva desde la que se trate. Por ejemplo, desde el punto de vista de una aplicación o de un usuario final, la QoS será lo que dicho usuario o aplicación aprecie en ese momento (ej: "*esta aplicación funciona muy bien*"), sin preocuparse de que mecanismos concretos han sido necesarios para obtener esa impresión. Sin embargo, desde el punto de vista de la red, la QoS puede ser una medida de la calida, en términos de parámetros o recursos óptimos, que la red ofrece al usuario o a la aplicación. Estas dos formas las podemos ver en la *Fig 3.1.*:

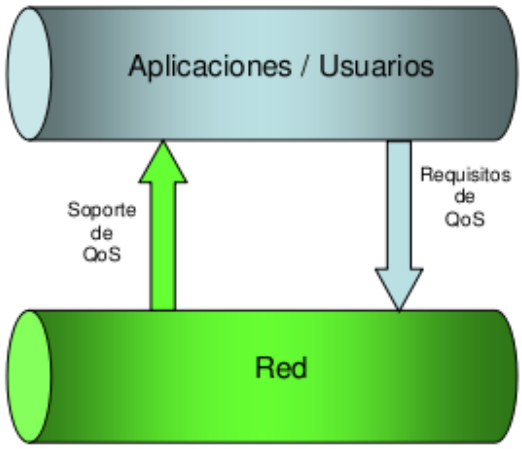

Figura 3.1 Perspectiva de la QoS

Desde el punto de vista de redes inalámbricas también existe una gran controversia. La mayoría de los vendedores implementan protocolos de QoS teniendo en mente un escenario específico, y considerando diferentes

parámetros, topologías de red y variables. *The United Nations Consultative Committee for International Telephony and Telegraphy* (CCITT) ha definido la QoS como, "*El efecto colectivo del comportamiento del servicio que determina el grado de satisfacción del usuario del servicio*". Esta definición es ampliamente aceptada y no hace referencia a características mínimas como ancho de banda, retrasos, o mecanismos como control de admisión, protocolos de señalización etc.

No hay una metodología concreta a seguir que nos proporcione una QoS óptima, para cada una de las aplicaciones. Lo que ocurre es que hay una serie de métodos que se han ido usando a lo largo del tiempo y que combinan diversas técnicas para conseguir una QoS adecuada. Algunas de estas técnicas las vemos a continuación:

- **Aprovisionamiento**: Proporciona suficiente capacidad de encaminamiento, almacenamiento en buffer y ancho de banda para que los flujos de paquetes se transmitan con facilidad. Es la solución más fácil para también la más cara, lo que hace que no sea factible para la mayoría de las aplicaciones.
- **Almacenamiento en buffer**: Los flujos de paquetes pueden almacenarse en un buffer receptor, antes de que se entreguen a las aplicaciones finales. Esto no afecta a la fiabilidad o al ancho de banda, puesto que los paquetes ya están en el receptor pero afecta al retardo. La ventaja de está técnica es que minimiza el *jitter* causado por congestiones de red, por pérdidas de sincronización o simplemente debido a las distintas rutas que hayan usado los paquetes para llegar a su destino. Este parámetro es clave para aplicaciones que requieran tiempo real, como video bajo demando que es lo que nos ocupa.
- **Modelado de Tráfico**: Permite ofrecer QoS en aplicaciones del tipo de videoconferencias donde es necesario que el servidor (transmisor) ofrezca una tasa de bits constante, es decir, que el tráfico se modere

desde el transmisor y no desde el receptor. Con esta técnica se consiguen dos aspectos:

- Se fuerza el tráfico entrante en la red a estructurarse según un patrón de flujo específico.
- Se supervisa dicho flujo de tráfico para saber si se ajusta en todo momento al patrón de tráfico fijado.

Esta técnica reduce la congestión de la red, parámetro crucial para las aplicaciones en tiempo real.

- **Reserva de recursos**: El uso de la técnica anterior implica de alguna forma obligar a que los paquetes pertenecientes a un flujo de datos sigan todos la misma ruta, ya que el envío a través de encaminadores aleatorios dificulta la garantía de QoS. Cuando un flujo sigue una ruta específica, es posible reservar recursos para él a lo largo de esa ruta y asegurarse así la capacidad necesaria requerida. Se pueden reservar tres tipos de recursos:
  - *Ancho de Banda*: Para no sobrecarga ninguna línea de salida del encaminador.
  - *Espacio de Buffer*: Para que el encaminador no descarte un paquete por no poder almacenarlo antes de retransmitirlo.
  - *Ciclos de CPU*: Suficientes para que el encaminador sea capaz de procesar el paquete en un tiempo determinado.
- **Control de Admisión**: Usando técnicas como las anteriores, logramos que un flujo de datos llegue a un encaminador. A partir de ese momento, el encaminador tiene que decidir si acepta (lo admite) o lo rechaza, en base a su capacidad y a las reservas ya realizadas por otros flujos. Tomar esta decisión es complicado, normalmente, la aplicación genera una especificación de flujo que contienen los parámetros de QoS que le gustaría utilizar. Conforme la especificación se propague por la ruta, los encaminadores intermedios irán modificando los valores de los

parámetros en base a su capacidad (siempre que sean valores que signifiquen una reducción del flujo), hasta que llegue al otro extremo, momento en el que se establecen los parámetros.

- **Encaminamiento proporcional**: Muchos de los algoritmos de encaminamiento tienen como objetivo encontrar la mejor ruta hacia un destino y enviar a través de ella todo el tráfico dirigido a él. Un sistema distinto de encaminamiento que se ha propuesto para obtener una QoS más alta es dividir el tráfico para un determinado destino a través de diversas rutas. Una forma simple de realizar esto, es dividir el tráfico en partes iguales o en proporción a la capacidad de los enlaces salientes.
- **Planificación de paquetes**: Un encaminador que maneje varios flujos simultáneamente no debería procesar los paquetes según el orden de llegada (encolamiento FIFO), ya que sí una fuente emite paquetes de formas masiva monopoliza los recursos del encaminador en detrimento de la QoS de los otros flujos. Para evitar esta situación, se han diseñado varios algoritmos de planificación de paquete:
    - Encolamiento imparcial (*fair queueing*): Usa varias colas separadas para cada línea de salida, una por flujo. Por el puerto de salida se enviará un paquete por cada flujo de forma circular (*Round Robin*). El problema de este algoritmo es, a parte de necesitar mucha potencia de procesador para clasificar los paquetes y gestionar las colas, que debido a la política de salida *Round Robin*, los flujos que tengan los paquetes de mayor tamaño obtendrán mayor ancho de banda que los que lo tengan de menor tamaño.
    - Encolamiento imparcial ponderado (*Weighted Fair Queuing*): Mejora el rendimiento de los flujos de paquetes pequeños (flujos de vídeo) frente a los flujos de paquetes grandes (transferencia de archivos) y evita la monopolización del ancho de banda por un solo flujo. Este algoritmo de encolamiento proporciona

garantías de latencia necesarias para el tiempo real y multimedia sin afectar demasiado la latencia de los paquetes grandes.

3.2. Métricas para medir QoS

En esta sección se van a describir los principales parámetros utilizados para medir el comportamiento de una red MANET cuando transporta tráfico.

3.2.1. Retardo

El retardo de red se define como el tiempo que emplea un paquete para viajar desde la fuente al destino. En general se encuentra compuesto por:

- Retardo de procesado: el tiempo que el nodo necesita para procesar las cabeceras.
- Retardo de cola: el tiempo que el paquete aguarda en la cola del *router*.
- Retardo de trasmisión: el tiempo que se necesita para enviar los bits que componen el paquete.
- Retardo de propagación: el tiempo que tarde la señal para propagarse por el medio.

El retardo en redes MANET suele ir desde milisegundos hasta varios cientos de segundos.

3.2.2. Variación del retardo o *jitter*

La variación del retardo o *jitter* es la variación en cuanto a la cantidad de latencia entre paquetes que se reciben. Es un parámetro muy relevante en estudios de QoS con servicios multimedia. En flujos multimedia, el *jitter* puede ser abordado dimensionando en el *buffer* de recepción de manera que se reduzcan los efectos negativos que puedan afectar a la calidad el vídeo final cuando un cuadro (*frame*) del vídeo final no está disponible para ser decodificado o reproducido en

el instante que se requiere. Sin embargo, este *buffer* puede causar un retraso extra antes de que el vídeo comience a reproducirse. Por tanto, su dimesionado correcto es fundamental.

3.2.3. Paquetes Perdidos

Hay muchas razones que pueden causar la pérdida de paquetes en entornos inalámbricos: enlaces de red saturados, colisiones, rotura de enlaces… Normalmente se presenta como el porcentaje de paquetes perdidos dividido por el número total de paquetes enviados por la fuente.

$$\%P_{loss} = \frac{P_{sent} - P_{Rx}}{P_{sent}}.100 \qquad (3.1)$$

3.2.4. *Throughput*

Se define como la cantidad de datos útiles del usuario recibidos en una comunicación dividida por la información transmitida. Para su cálculo hay muchos aspectos a tener en cuenta como: cabeceras de paquetes, paquetes de señalización, tiempos de espera, colisiones, retransmisiones de paquetes, etc. En (3.2) podemos ver la expresión para el *throughput* teórico:

$$Throughput_{teórico} = \frac{Data}{T_{ciclo}}\left[bits/\sec\right] \qquad (3.2)$$

En la ecuación (3.2) *Data* es la cantidad real de datos de un usuario llevados por una *frame* en T_{ciclo}, siendo T_{ciclo} la cantidad de tiempo que una *frame* necesita para ser enviada. Esta cantidad de tiempo se calcula añadiendo los tiempos de espera que componen un ciclo de transmisión.

Una vez tenemos la expresión teórica, ahora estamos interesados en compararlo con el *throughput* actual. La ecuación (3.3) muestra cómo obtenerlo, donde

Data$_{útil}$ se corresponde con los datos actuales del usuario y T_{total} con el tiempo total que el nodo fuente está transmitiendo.

$$Throughput_{real} = \frac{Data_{útil}}{T_{total}} \left[bits / \sec \right] \quad (3.3)$$

3.2.5. Número de saltos

Este parámetro es muy relevante en entornos de redes MANET, cuando hay un número muy alto de saltos en un camino desde la fuente al destino, el retraso extremo a extremo podría ser más alto de lo deseable. Al mismo tiempo la probabilidad de sufrir roturas de enlaces aumenta. Además, pérdidas debido a colisiones también aumentan en camino grandes. Por tanto, caminos cortos son preferibles a largos. El número de saltos de un camino coincide con el número de enlaces, así como el número de saltos de un camino es definido como el número de nodos envueltos en el camino menos uno.

$$N_{saltos} = N_{enlaces} = N_{nodos\_camino} - 1 \quad (3.4)$$

3.2.6. Ancho de banda disponible extremo a extremo

Es un parámetro muy común para medir QoS en redes. Mide el máximo ancho de banda disponible entre los dos equipos finales de una conexión. Suele ser aproximado por el cuello de botella, es decir, por el ancho de banda del enlace más restricitivo que compone el camino de fuente a destino.

3.3 Modelos existentes de QoS

En este apartado vamos hacer un repaso sobre los modelos de QoS [2] con mayor auge en Internet, para en posteriores secciones tener en cuenta las especiales características de las redes MANET y las restricciones que nos impondrá. En el mundo de Internet las propuesta de QoS con mayor impacto son las estudias por el IETF a través de dos grupos de trabajo: *Intserv* (servicios integrados) y *Diffserv* (servicios diferenciados) que pasamos a ver con más detalle.

3.3.1. Servicios Integrados (*Intserv*)

El modelo de servicios integrados propone una solución para el soporte de QoS extremo a extremo basado en la reserva de recursos en los diferentes equipos de conmutación que componen el trayecto que seguirá la información en la comunicación. Con este modelo se pretende ofrecer soporte para un funcionamiento adecuado de aplicaciones en tiempo real.

Ese soporte supone una mejora sobre el servicio tradicional de Internet, de forma que permite a las propias aplicaciones especificar los requisitos de QoS necesarios. Esta lista de requisitos debe difundirse entre los diferentes elementos de conmutación (*routers*) por los que se encaminaran los paquetes de determinada aplicación. Estos equipos deben proporcionar mecanismos para el control de QoS ofrecido a estos flujos de información, lo que se consigue mediante la reserva de recursos.

El modelo de servicios integrados se basa en la definición de dos elementos, una arquitectura donde los elementos de red permiten reservar recursos de conmutación, y un protocolo que permita a las aplicaciones transmitir sus requisitos a estos elementos de conmutación, el protocolo RSVP (*Resource Reservation Protocol*).

De esta forma cuando una aplicación desea comenzar una comunicación debe realizar una petición de recursos, esta petición atravesará todos los nodos que formen el trayecto para el flujo de información, y en función de los recursos disponibles será aceptada o rechazada. Este procedimiento se aprecia en la *Fig 3.2*.

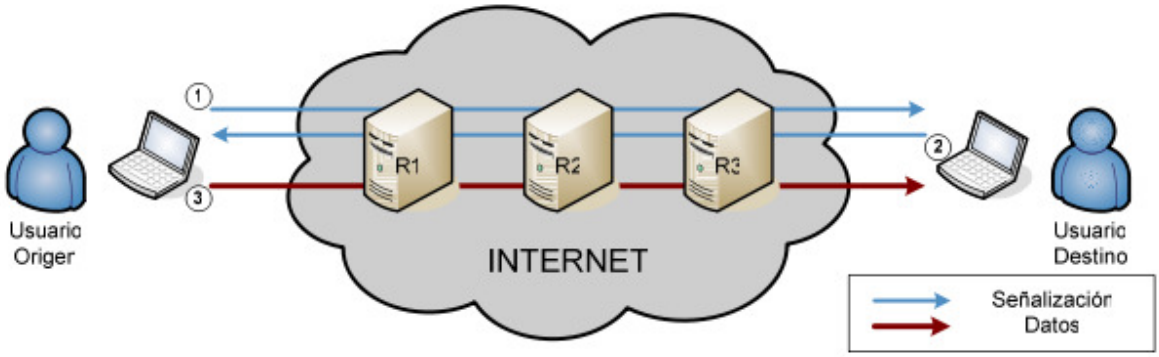

Figura 3.2. Esquema de funcionamiento del modelo Intserv

Modelo de Tráficos y modelo de Servicios

Para determinar los recursos que requiere cierta aplicación resulta necesario clasificar los posibles flujos de tráfico. La clasificación se realizará en función de los requisitos que plantea para su correcto funcionamiento: ancho de banda, retardo, variación de retardo. Con respecto a la dependencia del retardo se distingue entre:

- Tráfico Elástico: Son aquellas aplicaciones donde el retardo que sufren las diferentes tramas entre fuente y destino no afecta de forma substancial al servicio ofrecido al usuario. Ejemplos de este tráfico serían la navegación web, correo electrónico, descarga ftp…
- Tráfico Inelástico: Se corresponden con aquellas aplicaciones muy sensibles al retardo sufrido por las tramas. Además de requieren un ancho de banda mínimo para que el usuario reciba el servicio sin percibir

diferencias en el mismo. Algunos ejemplos podrían ser las videoconferencias, o vídeos bajo demanda, que nos ocupan en este proyecto.

Por otro lado es importante definir un conjunto de servicios dentro del modelo *Intserv* con diferentes características en cuanto a la calidad del servicio ofrecido, y que permita ofrecer diferentes niveles en función de las necesidades de las aplicaciones.

- Servicio Garantizado (*Guarenteed Service*): Permite reservar un caudal mínimo extremo a extremo así como limitar el retardo máximo que sufrirán las tramas en su trayecto. Para lograr este propósito es necesarios que los nodos de conmutación intermedios hagan una reserva de los recursos necesarios, de forma que las reservas individuales aseguren los requisitos extremo a extremo. Este servicio se consigue gracias al protocolo RSVP, que transmite los requisitos de la aplicación al modulo de control de admisión.
- Servicio de Carga Controlada (*Controlled Load*): Al igual que el servicio anterior, la aplicación realiza una petición al Control de Admisión indicando una estimación de los recursos necesarios. Este servicio ofrece una transmisión con una probabilidad de error controlada apoyándose en un reparto estadístico de los recursos de la red.
- Servicio *Best-Effort*: Utilizado cuando la petición de recursos ha sido rechazada por la falta de recursos disponible en la red. Si se desea utilizar directamente este servicio, no resulta necesaria una petición a diferencia de los servicios anteriores. Este servicio resulta adecuado para aplicaciones con tráfico elástico.

Arquitectura *Intserv*

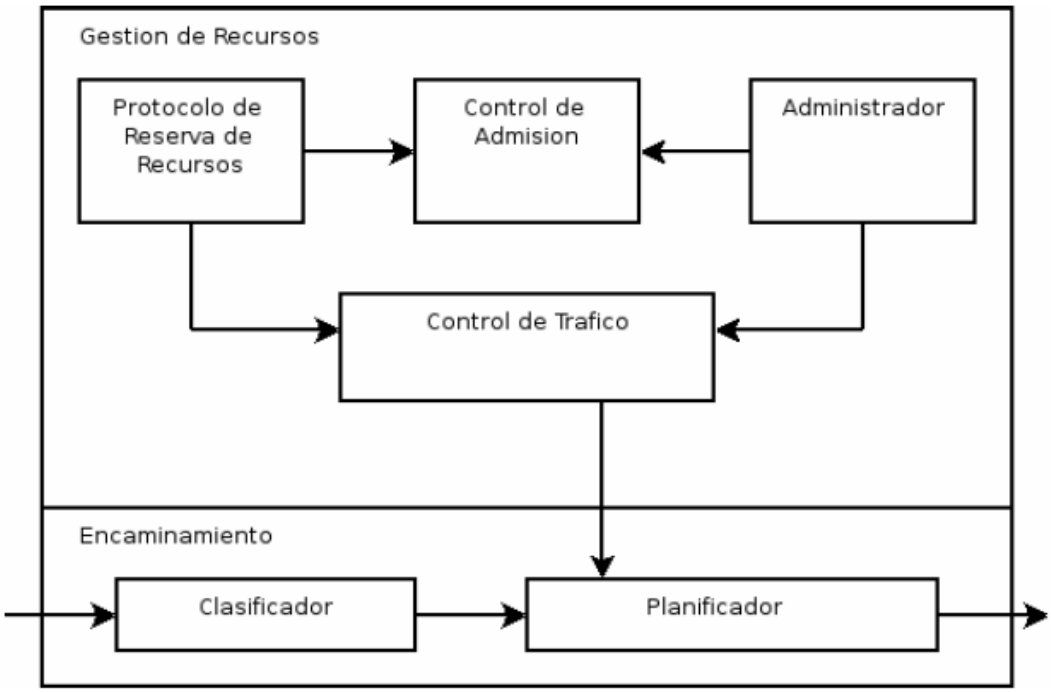

Figura 3.3 Esquema de funciones internas de Intserv

La arquitectura *Intserv* que podemos ver en la *Figura 3.3* engloba todos los aspectos vistos hasta ahora sobre los servicios integrados. El encaminamiento de las tramas se hace a través del clasificador y el planificador. La gestión de los recursos se lleva a cabo en el Control de Admisión, Protocolo de reservas y Administrador.

Protocolo de Reserva. RSVP

El modelo de servicios integrados define un protocolo específico para la gestión del QoS en la red, RSVP (*Resource Reservation Protocol*). Este protocolo permite a las aplicaciones el envío de peticiones de reserva de recursos a los diferentes nodos de conmutación de la red *Intserv*.

El protocolo RSVP especifica quien será el receptor responsable de efectuar la reserva en lugar del emisor. Este hecho provoca que el receptor necesite conocer previamente las características del tráfico para efectuar la reserva. Por otro lado esto permitirá que en el caso de las comunicaciones *multicast* los diferentes receptores puedan especificar diferentes parámetros de QoS, ofreciendo una mayor capacidad de configuración. Otra característica de gran importancia es el hecho de que las reservas realizadas por el protocolo RSVP son unidireccionales. De forma que si deseamos establecer una comunicación bidireccional será necesario que ambos receptores realicen su propia petición de recursos.

Inconvenientes de *Intserv*

Como ya se ha comentado, el modelo *Intserv* implica una reserva individual de recursos para cada flujo de información. La gran cantidad de usuarios que componen una red, así como el elevado número de flujos que puede generar cada usuario provoca que existan graves problemas de escalabilidad en el núcleo de la red.

Debemos tener en cuenta que cada nodo de conmutación tendrá que almacenar un listado de todos los flujos activos y los correspondientes recursos asignados. Por otro lado, estas reservas son temporales de manera que deben ser renovadas que cierto tiempo. Estos factores provocan que el modelo *IntServ* sea difícilmente implementable en una red de dimensiones considerables.

3.3.2. Servicios Diferenciados

El modelo de servicios diferenciados ofrece una solución para el soporte de QoS basado en la priorización de clases de tráfico. Al igual que en el modelo anterior, la provisión de QoS se realiza a través de una reserva de recursos en los nodos intermedios, pero en este caso las prerreservas se realizan por agregados de tráfico, en lugar de por flujos.

Esta prerreserva de recursos es una labor de administración de la red, es decir, las aplicaciones no realizan ninguna petición de recursos. Simplemente deberán marcar el tráfico que generan adecuadamente para que reciba un tratamiento específico en función de la clase a la que pertenezca.

Arquitectura del modelo de servicios diferenciados

El modelo de servicios diferenciados define un dominio *DiffServ* donde aparecen equipos de conmutación que se pueden dividir en:

- *Nodos Interiores* (CR,…*routers*): Son los nodos que forman el núcleo de la red. Sus funciones se limitan a formar un sistema de encolamiento que permita ofrecer diferentes tratamientos a los agregados de tráfico en función de sus requisitos preestablecidos.

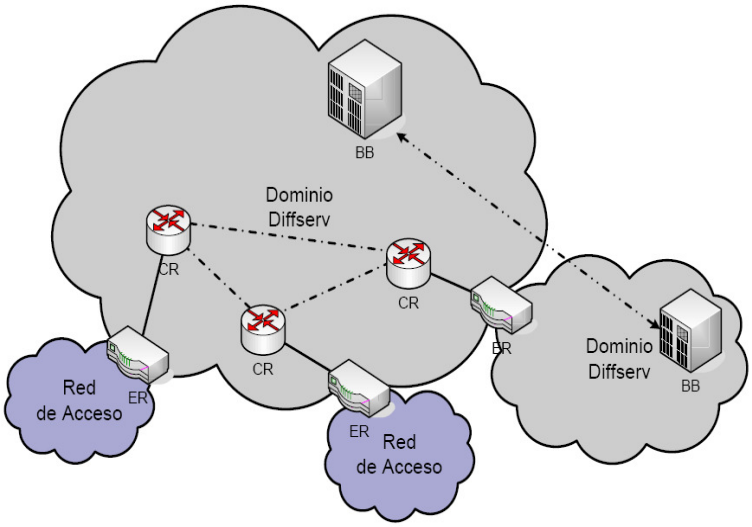

Figura 3.4 Elementos que forman un dominio Diffserv

- *Nodos Frontera* (ER, *Edge Routers*): Son aquellos que se encuentran en los límites del dominio y presentan algún interfaz con un nodo fuera del dominio Diffserv o con una red de acceso. Deben implantar las funciones discretas para los nodos interiores y además deben encargas de funciones de clasificación y acondicionamiento de tráfico, de forma que todo tráfico que entre en un dominio *Diffserv* cumpla una serie de requisitos.

Para la identificación de los diferentes agregados de tráfico se define un código llamado DSCP (*Diffserv Code Point*). Esta información va mapeada en un campo que tiene el aspecto de la *Fig 3.5*.

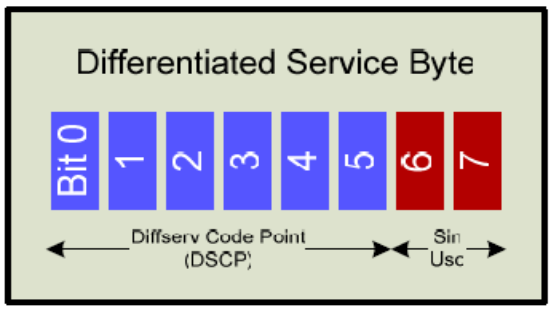

Figura 3.5 Campo de identifiación Diffserv

Podemos ver como los 6 bits se utilizan para identificar el agregado de tráfico al que pertenezca cada paquete, de forma que los *routers* que pertenecen al dominio *Diffserv* pueden aplicar el tratamiento correspondiente, es lo que se conoce como Comportamiento por salto (*PHB – Per Hop Behaviour*).

Los comportamientos por saltos o PHB definen un conjunto de condiciones para el tratamiento del tráfico conocidas como perfiles. Esto perfiles permiten que los diferentes agregados reciban más o menos recursos según como hayan sido etiquetados. Existen tres perfiles PHB definidos en el modelo *Diffserv*:

- *Expedited Forwarding* (EF): Se trata de aquellos flujos que requieran un caudal mínimo asegurado, así como un retardo limitado y una variación de retardo máximo determinada. Este perfil se ajusta a aplicaciones de tráfico en tiempo real, como puede ser audio/video conferencia, o descarga de video bajo demanda.
- Asured Forwarding (AF): Indicado para los flujos de tráfico con menores requisitos que los descritos en EF, ya que no es posible indicar recursos temporales para estos flujos (retardos/*jitter*). Este perfil define cuatro tipos de clase diferentes en función de los recursos reservados a las mismas. Además resulta muy adecuado para la implementación de recursos olímpicos, donde se puede asignar a cada agregado de tráfico la clasificación de oro, plata y bronce, de forma que reciba los recursos correspondientes a cada nodo que atraviese por el dominio *Diffserv*. Un nodo del dominio *Diffserv* que pertenezca a este dominio debe ser capaz de detectar situación de congestión en la red y aplicar descarte de tramas en función de la clase que pertenezca el mismo.
- *Best Effort* (BE): Aunque no pertenece exclusivamente al modelo *Diffserv*, este perfil se utiliza para el tráfico que no tiene requisitos de QoS (caudal garantizado o consideraciones temporales). Este perfil es adecuado para las aplicaciones que trabajan en *background* o que no requieren trabajo en tiempo real, por ejemplo, descarga de fichero de ftp, navegación web…

Arquitectura de un nodo *Diffserv* y sus funciones

En la *Fig 3.6* aparecen las diferentes funciones que deben implementar *routers* interiores y fronteras que forman el dominio *Diffserv*.

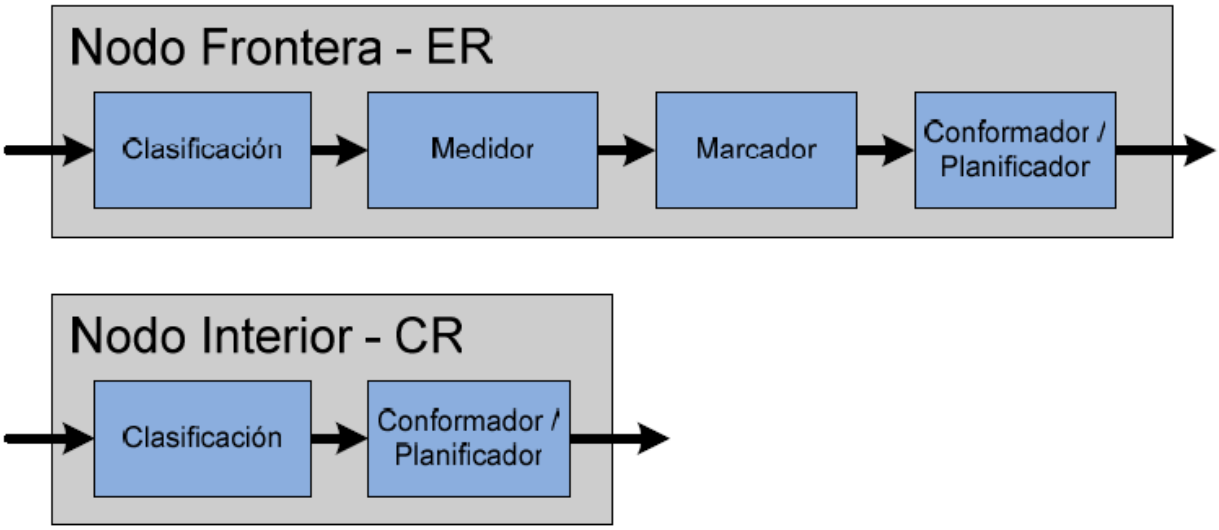

Figura 3.6. Funciones de los nodos en un dominio Diffserv

La función de clasificación multicampo, así como el acondicionamiento suele implementarse en los nodos frontera, de esta forma aseguramos que el tráfico existente dentro del dominio *Diffserv* se ajusta a un Control de Admisión. Gracias a este política los nodos interiores no necesitan implementar estas funciones que podrían ser bastante costosas en el núcleo de red debido a la elevada carga.

De igual forma, estas funciones podrían realizarse directamente en los nodos orígenes del tráfico, pero para ello tendríamos que tener una total confianza en los mismos, ya que si se saltasen ambas funciones estarían introduciendo tráfico en la red descontrolado, lo que podría provocar un funcionamiento incorrecto del modelo *DiffServ*.

Podemos comprobar que los problemas de escalabilidad que aparecían en el modelo de servicio integrados desaparecen, ya que la agregación del tráfico provoca que las funciones a implementar por los nodos *Diffserv* no incrementen de forma desorbitada según el número de usuarios y flujos.

3.4. QoS en redes MANET

En primer lugar vamos a reseñar las particularidades de las redes MANET que imposibilitan el uso de los modelos detallados en la anterior sección para acometer un estudio de QoS sobre las redes MANET [6].

En el capítulo 1, se comentaron ciertas características de las redes MANET que las diferencian con respecto a sus homólogas. En esta sección vamos a profundizar en éstas desde el punto de vista de la QoS, destacando la dificultad que suponen para alcanzar ciertos niveles de satisfacción:

- **Propiedades de enlaces impredecibles**: El medio inalámbrico es impredecible por naturaleza, y conlleva de manera intrínseca las colisiones ente paquetes. La propagación de la señal sufre interferencias, cancelaciones *multicamino*... lo que provoca que la medida del ancho de banda y de los retrasos del enlace sea una cuestión impredecible.
- **Movilidad de los nodos**: Provoca una topología de red dinámica. Los enlaces son formados de manera dinámica cuando dos nodos entran dentro de su dominio de acción y se rompe cuando uno de ellos sale.
- **Baterías limitadas**: Los dispositivos móviles dependen de baterías. La provisión de QoS debe considerar un consumo extra y este hecho debe ser tenido en cuenta a la hora de elaborar técnicas de QoS.
- **Problemas con terminales expuestos y escondidos**: En la capa MAC con el tradicional protocolo de acceso múltiple (CSMA), los paquetes

multisalto introducen los problemas “terminal expuesto” y “terminal escondido”. El problema del “terminal escondido” ocurre cuando dos nodos que están fuera del rango de transmisión uno de otro trata de transmitir a un nodo receptor común. El problema del “terminal expuesto” cuando, en la misma situación que en el caso anterior, un nodo quiere transmitir, pero otro ya lo esta haciendo, lo que provoca interferencias entre las transmisiones de uno y otro.

- **Mantenimiento de Rutas**: La naturaleza dinámica de la topología de red en MANET hace que los protocolos de enrutamiento tengan que trabajar con información imprecisa. Además, los nodos pueden dejar el dominio o entrar de nuevo por lo que caminos ya establecidos se romperían. Este hecho afecta a la filosofía de reservar recursos de algunos modelos de QoS, ya que, con el carácter impredecible de los caminos no es posible hacer una prerreserva.

- **Seguridad**: La seguridad puede considerarse un atributo de QoS. El medio físico de comunicaciones es inherentemente inseguro, luego se necesitarán algoritmo de enrutado con proporcionen cierta seguridad.

Las primeras consecuencias de estas particularidades afectan a los modelos de QoS existentes y anteriormente comentados.

- *Intserv*: No es aplicable en redes MANET debido a su limitación de recursos. Hay tres factores fundamentales que afectan:
  - Enorme almacenamiento y procesamiento de señalización para cada nodo móvil, debido a que tienen que establecer y mantener toda esta información. Además, la cantidad de información de estado aumenta con el número de flujos,
  - La reserva RSVP y el proceso de mantenimiento consume procesos de red. Por tanto, los paquetes RSVP y los datos tienen que compartir

los recursos de red, es decir, el ancho de banda. Este se produce fundamentalmente porque RSVP es un protocolo *out-of-band*.

- Para tratar de tener un mecanismo de QoS como el Control de Admisión, clasificación y planificación debe ser provisto. Estos mecanismos requieren gran cantidad de recursos de red que en redes MANET no están disponibles.

- *Diffserv*: Este modelo necesita menos recursos interiores de la red ya que en lugar de usar flujos individuales usa agregados de tráfico. Por tanto, este modelo podría ser un mecanismo potencial para redes MANET, sin embargo, no está clara la definición, de interior o frontera en una topología dinámica como es la MANET.

3.5 QoS desde una perspectiva de capas

En esta sección vamos hacer un análisis de QoS en redes MANET desde la perspectiva de capas, empezando por la capa física para acabar en la capa de aplicaciones.

3.5.1. Soporte de QoS en técnicas de canal

Los canales inalámbricos en rede MANET cambian con el tiempo, lo que significa que el modelo de canal también varía con el tiempo. Uno de los mayores retos en soporte de QoS sobre canales inalámbricos es la estimación del canal. Esto engloba una estimación precisa en el lado del receptor y el tratamiento eficiente de los antecedentes en el transmisor de manera que ambos se encuentren sincronizados.

La comunicación sobre canales inalámbricos esta sujeta a ruido y colisiones. Además, con la demanda de transmisiones de video en tiempo real hacen este problema más complicado. Por tanto, no solo se debe mejorar el soporte de QoS

a nivel de canal sino también la comunicación con las capas superiores, como algoritmo de compresión en la capa de aplicaciones. Básicamente, usando una tasa de codificación de fuente mayor (menos compresión de datos) podemos decrementar la distorsión extremo a extremo. Mientras tanto, usando mayor protección de canal (códigos de palabra mayores), podemos reducir posibles errores en el canal, lo que implica menos distorsión extremo a extremo. Debemos de llegar a un compromiso entre ambos.

3.5.2. Provisión de QoS en la capa MAC

Recientemente muchos esquemas han sido propuestos para garantizar la QoS de tráfico en tiempo real en redes inalámbricas. IEEE 802.11 [SA] puede ser considerada la versión inalámbrica de Ethernet soportando servicios *Best Effort* (sin garantías a nivel de servicios para usuarios o aplicaciones). En 2001, un grupo E fue creado para expandir la IEEE 802.11 para requerimientos de QoS en aplicaciones, y también mejorar la capacidad y eficiencia del protocolo. En Julio de 2005, los trabajos fueren concluidos y surgió la IEEE 802.11e [SA].

Esta nueva versión incluye mejoras en la capa MAC para aplicaciones sensibles a los retrasos como de audio o vídeo en tiempo real sobre entornos inalámbricos. Las mejores fundamentales son dos:

- A diferencia de cómo se hacía en el estándard IEEE 802.11, con esta nueva versión no se manda un asentimiento por cada *frame* recibida exitosamente con objeto de evitar aumentar el retraso. La nueva política, es que la siguiente *frame* a mandar debe estar preparado durante un tiempo determinado desde el final de la transmisión previa.
- Se crean cuatro categorías de acceso, las cuales permiten manejar colas con diferentes prioridades asignadas a cada una de ellas.

Cada categoría de acceso es configurada con sus propios valores de AIFS (*Arbitrary Interframe Space*), CWmin (*Contention Window Minimum*), CWmax (*Contention Window Maximum*). AIFS define un período de tiempo que debe ser esperado antes de que se pueda acceder al medio inalámbrico.

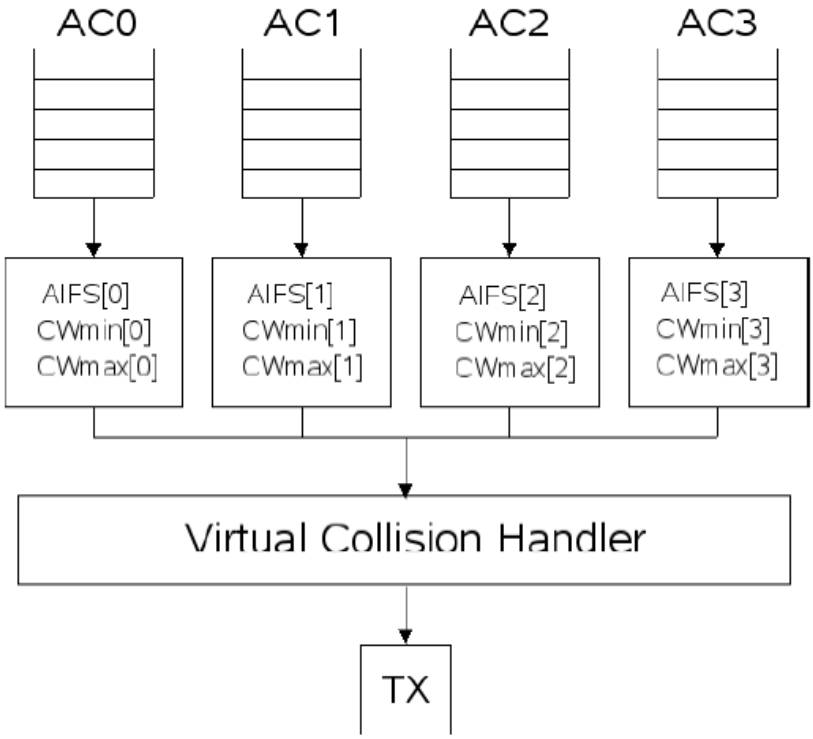

Figura 3.7. Esquema del MAC IEEE 802.11e

3.5.3. Qos en la capa de red

En el capítulo 2 han sido propuestos los protocolos de enrutados que proporciona el nivel más óptimo en cuento a QoS para aplicaciones de video/audio en tiempo real. Además en este mismo capítulo se han descrito los dos modelos existentes de QoS, *Intserv* y *Diffserv*, y se han argumentado las razones por las que estos modelos no son directamente aplicables a una red MANET.

3.5.4. Soporte de QoS en la capa de transporte

TCP diseñado para Internet asume que la principal causa por la que se pierden paquetes es la congestión de red. Sin embargo, esto no es cierto para entornos inalámbricos, donde las pérdidas suelen ser debidas a ruido de canal o cambios en las rutas.

Recientemente, muchos trabajos se han llevado a cabo para mejorar el comportamiento de TCP sobre enlaces inalámbricos en MANET. Algunos de estos nuevos protocolos pueden distinguir entre congestiones de red y pérdidas de manera que acciones específicas puedan llevarse a cabo, otros están basados en la implicación y la estimación desde la observación. Enumeramos los más importantes:

- **Chandran**: Propone una mejora del TCP basada en un estudio de los antecedentes. Cuando una rotura de caminos es detectada por un nodo intermedio, éste mandará un RFN (*Route Failure Notification*). Al recibirlo, el transmisor entre en un estado de "congelación" (detiene los contadores de retransmisión y para de mandar paquetes) hasta que un RRN (*Re-establishment Notification*) llega.
- **Holland**: Investigó el impacto de la rotura de caminos en el comportamiento de TCP y propuso usar ELFN (*Explicit Link Failure Notifiaction*) para mejorarlo. Cuando el transmisor recibe un ELFN, deshabilita sus contadores de retransmisión y entre en *stand-by.* En este estado, el transmisor manda periódicamente mensajes de prueba para saber si la ruta esta reestablecida o no.
- **Liu**: Relaciona los problemas de comportamiento del TCP con errores de camino debido a tasas altas de error (BER). Propone inserta una nueva capa ATCP entre TCP e IP, de manera que este nuevo esquema mantiene la compatibilidad con el estándard TCP/IP, ATCP escucha el estado de la

red gracias a los mensajes ECN (*Explicit Congestion Notification*) e ICMP (*destination unreachable*), y pone al transmisor en el estado apropiado de acuerdo con estos mensajes.

- **Dyer**: Propone la técnica llamada RTO (*Recovery Time Objective*) fijo. Está técnica no depende de ninguna algoritmo de información explicito. Cuando se producen varios errores de *timeout* consecutivos, el transmisor lo toma como una rotura de enlace. El paquete con asentimiento negativo es retransmitido otra vez sin doblar el valor RTO una segunda vez. El RTO permanece fijo hasta que se reestablece la ruta y el paquete retransmitido recibe un asentimiento positivo.
- **Wang**: Propone la detección de DOOR (*Out of Order and Response*), que no depende de ningún algoritmo de información tampoco. DOOR está basado en la capacidad de TCP para detectar eventos fuera de orden y cambios en las rutas provocados desde estos eventos.

3.5.5. Soporte de QoS en la capa Aplicaciones

Estrategias adaptativas juegan un papel muy importante en soporte de QoS en redes MANET. La capa de Aplicaciones incluye aspectos tan flexibles como interfaces de usuarios, rangos dinámicos de QoS, algoritmos de compresión adaptativos, codificación fuente de canal o esquemas de codificación de red.

Un rango dinámico de QoS en lugar de una fijo puede ser usado para la reserva de recursos para cumplir la naturaleza dinámica de las redes MANET. Esta estrategia dinámica tiene implicaciones para las aplicaciones. Primero, las aplicaciones deben tener algunas nociones del rango de QoS donde van a operar. Estos rangos pueden ser programados o configurados por el usuario de acuerdo con la finalidad de su aplicación. Segundo, durante la ejecución, la aplicación debe ser capaz de adaptar su comportamiento al estado de las capas inferiores.

3.6. Modelos de QoS para redes MANET

Tras haber analizado todos los detalles referentes a las redes MANET desde el punto de vista de QoS vamos a describir los modelos existentes para llevar a cabo esta tarea sobre redes MANET.

SWAN (*Stateless Wireless Ad-hoc Networks*)

SWAN asume una MAC *Best-Effort* y propone un simple y distribuido modelo de red que incluye mecanismo de control para el soporte de aplicaciones en tiempo real y servicios diferenciados en redes Ad-hoc. Un clasificador incluido en SWAN diferencia entre *Best-Effort* y tráfico de tiempo real. SWAN usa el control de tráfico para UDP (*User Datagram Protocol*) y tráfico TCP (*Transport Control Protocol*) *Best-Effort* para mantener el ancho de banda y los retrasos dentro los requerimientos exigidos por el tráfico en tiempo real. Además, usa notificaciones explícitas (ECN) de congestión de tráfico para regular dinámicamente el tráfico en tiempo real admitido y adaptarse así a las características dinámicas de la red. En la *Fig 3.8*. se muestra el modelo. Este modelo presente los típicos mecanismos de ayuda a la QoS como son clasificador, formador, control de admisión.

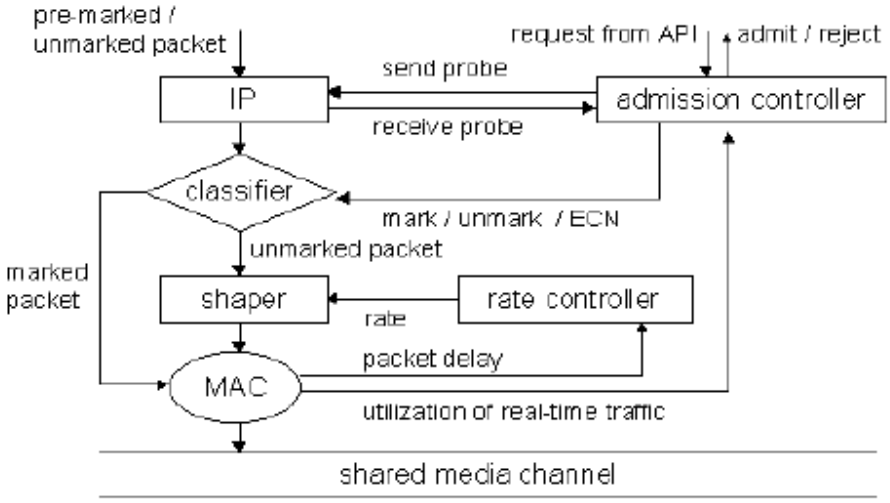

Figura 3.8 Arquitectura SWAN

FQMM (*Flexible QoS Model for Mobile Ad-hoc networks*)

FQMM [FQM00] es una combinación entre la clasificación por flujos de *Intserv* y la clasificación por agregados de tráfico de *Diffserv*. El tráfico más prioritario es enviado como en *Intserv*, por flujo, y el resto como agregados. De esta manera no se producen problemas de escalabilidad. Los perfiles de tráfico se usan para mantener las diferencias entre sesiones y para adaptarse al dinamismo de la red considerando la capacidad efectiva del enlace. El nodo fuente hace una clasificación de los paquetes. El papel de los nodos intermedios es proveer la QoS asegurando que el tráfico de cierta clase insertado en el camino no supera al porcentaje de ancho de banda concedido a ese perfil de tráfico. Mirando a los protocolos de encaminamiento esta política estaría en concordancia con los protocolos proactivos. Se muestra sus principales componentes en la *Fig 3.9*.

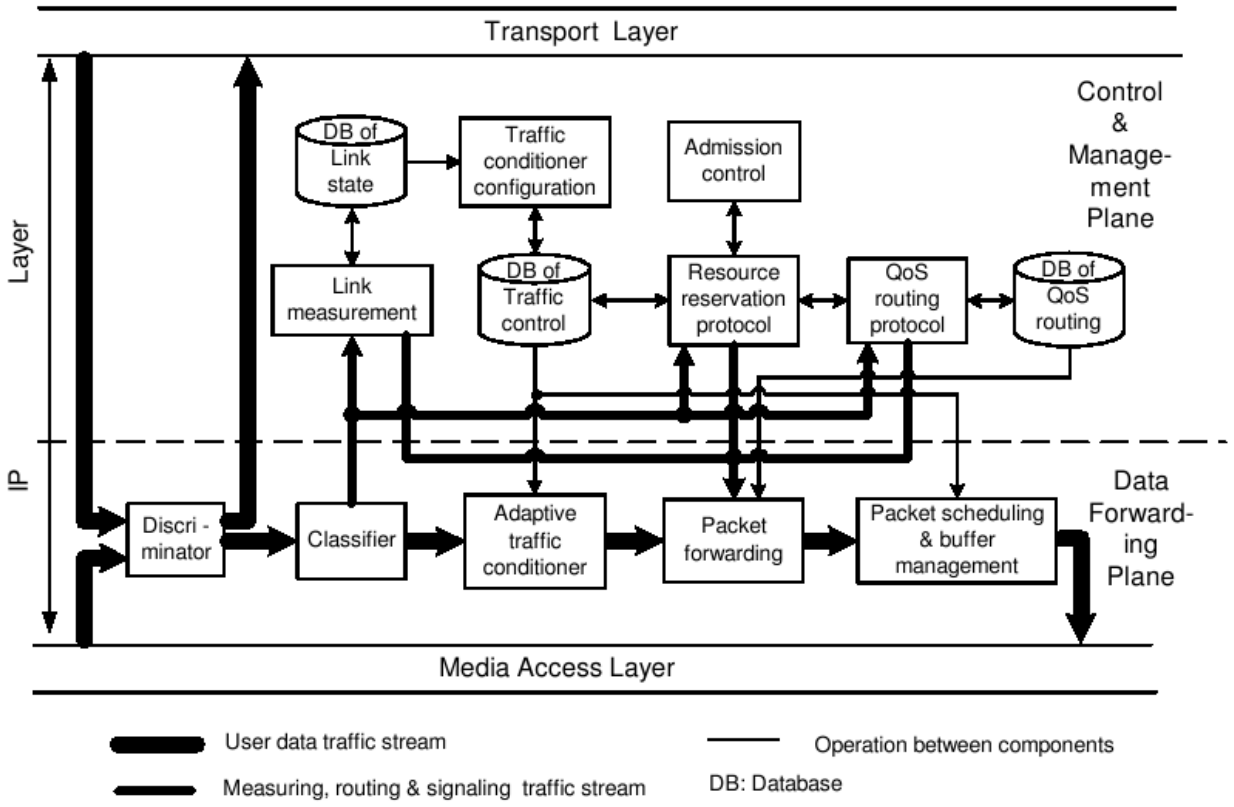

Figura 3.9 Arquitectura FQMM

INSIGNIA (*IP-based QoS framework for Mobile Ad-hoc Networks*)

INSIGNIA es un método de QoS basado en IP que soporta servicios adaptativos en redes Ad-hoc. Es el primer protocolo diseñado exclusivamente para redes MANET. Para poder alcanzar una señalización correcta tiene los siguientes requerimientos:

- La transferencia de señales entre *routers* debe ser efectiva.
- Debe existir una correcta interpretación y activación de mecanismos para manejar las señales.

Es decir, la señalización que se manda por los *routers* dentro de la red debe ser comprendida e interpretada por el resto de los nodos. La transferencia de señales entre *routers* con el protocolo ISIGNIA es *in-band-signalling*, es decir, toda información de control de red es encapsulada dentro de los paquetes de datos por lo que el envío de señalización es ligero y fácil.

INSIGNIA soporta reservas rápidas de flujo, restauración y algoritmos adaptativos que están especialmente diseñados para gestionar servicios en tiempo real sobre entornos MANET. Como se comentó anteriormente, la señalización se encuentra encapsulada en la opción IP de cada paquete IP (*Fig 3.10*), que pasa a llamarse opción INSIGNIA. Además, se mantiene la información del estado del flujo en tiempo real sacando la información extremo a extremo, informando a la fuente del estado de su flujo de datos. Todas estas configuraciones convierten a INSIGNIA en un protocolo adecuado para MANET.

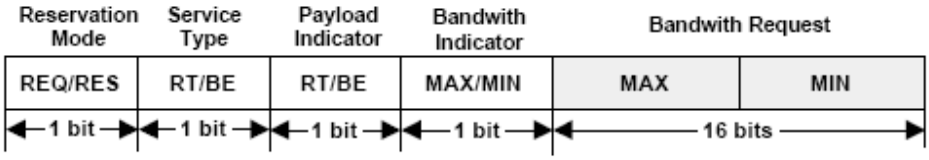

Figura 3.10 INSIGNIA campo opción IP

El módulo INSIGNIA esta envuelto cada vez en un paquete IP que es recibido. En coordinación con el módulo de Control de Admisión, se concede el ancho de banda requerido por el flujo si los requerimientos de recursos pueden ser satisfechos. Por otro lado si los recursos no pueden ser satisfechos el paquete se señala con *Best-Effort* y se reenvía si fuera necesario.

Es importante tener claro que INSIGNIA es un protocolo de señalización y no de enrutado. Es decir, seguimos teniendo la necesidad de usar alguno de los protocolos vistos en los capítulos 1 y 2 para llevar a cabo el enrutado de los paquetes. El mayor retroceso de INSIGNIA a pesar de que la información sobre el estado de los flujos de datos se mantiene en los nodos móviles, es debido a los problemas de escalabilidad que surgen si aumenta el número de estados de los flujos. Además, es un protocolo diseñado casi en exclusiva para servicios en tiempo real o *Best-Effort*.

CEDAR (*Core Extraction Distributed Ad Hoc Routing*)

Es un algoritmo de enrutado que proporciona QoS en redes Ad-hoc [CED99]. Sus tres aspectos fundamentales son:

- Establecimiento y mantenimiento de una autoorganizada de infraestructura de enrutado, llamada el núcleo, para llevar a cabo los cálculos de enrutado.
- La propagación del estado de enlaces de alto ancho de banda en el núcleo a través de incrementar y decrementar las ondas.
- Un algoritmo de QoS que se ejecuta en el núcleo usando solo información de estado local.

El hecho de elegir una serie de nodos como núcleo para evitar una señalización alta afecta a la red debido a que las continúas actualizaciones de la información

del estado de la red. Grupos de nodos son elegidos continuamente para gestionar las rutas y extender la información de los nodos de su dominio dentro de la red. Así que, cuando un nodo necesita una ruta hacia su destino se comunica con el nodo núcleo, éste calcula una ruta hacía el nodo núcleo del dominio del destino. Estos paquetes son difundidos al mínimo número de paquetes núcleo posible.

Hay al menos un nodo núcleo cada tres saltos y cada nodo núcleo conoce la existencia de los nodos núcleos vecinos. Cuando un nodo núcleo se desplaza debe encontrar de nuevo sus nodos núcleos vecinos. Este produce un control substancial de la señalización, especialmente en entornos de alta movilidad. Cada nodo conoce su topología local, usando mecanismos *Link-State*. La información de enlaces con alto ancho de banda y estabilidad se propaga por toda la red, sin embargo, la información de enlaces con ancho de banda bajos y poca estabilidad permanecen en el ámbito local. Este protocolo tiene sus propios mecanismos para el establecimiento y mantenimiento de rutas usando adecuadamente los nodos núcleos. Su mayor desventaja es la computación de rutas es llevada a cabo exclusivamente por los nodos núcleo, por lo que el movimiento de dichos nodos afecta al comportamiento del protocolo. Además, las actualizaciones de la información de los nodos núcleos pueden causar una alta señalización.

3.7. Conclusiones

QoS en redes MANET es un nuevo pero rápidamente creciente ámbito de interés. Este crecimiento y el interés desde el punto de vista del mercado se debe principalmente al incremento de su popularidad y necesidad de aplicaciones multimedia y al potencial uso de redes MANET. Por tanto, la provisión de QoS en redes MANET es una cuestión imprescindible.

Por ello, se ha comenzado con un análisis de la QoS, sus características, sus modelos para pasar a justificar porque no son directamente aplicables en entornos MANET, debido a sus restricciones de ancho de banda y su topología de red dinámica. Finalmente se ha ofrecido una visión de los modelos que actualmente trabajan para logra ciertos grados de QoS en este tipo de redes.

El establecimiento de videos bajo demanda sobre redes MANET aún presenta retos que deben ser resueltos para obtener un cierto grado de QoS en el usuario final. En este proyecto, se pretende poder medir un aspecto más de este de QoS sobre este tipo de redes y sobre tráfico de vídeo, la PSNR. Esta medida nos da una idea de la calidad subjetiva del vídeo que el usuario va a percibir, pero de una forma fácil de medir mediante una expresión matemática. En este trabajo hemos desarrollado una herramienta calculadora de la PSNR para medir la calidad del vídeo transmitido sobre redes MANET. Esta herramienta permite evaluar de forma fácil diferentes protocolos de encaminamiento que se diseñen.

Capítulo 4: Diseño de la Calculadora PSNR

En este capítulo se va a explicar detalladamente cómo se ha implementado la calculadora de PSNR que es el objeto principal de este proyecto. Para ello, se comenzará con un análisis de las razones que convierte a esta métrica, *Peak Signal-to-Noise Ratio* (PSNR), en un medidor muy convincente en situaciones de transmisión de vídeo a través de redes MANET. Posteriormente, se mostrarán las herramientas que se han utilizado para su implementación, donde podremos ver desde lenguajes de programación hasta herramientas de libre distribución. Luego, se detallará paso por paso, el diseño y la realización de la calculadora y por último se presentará conclusión a modo de resumen.

4.1. Objetivos

Haciendo una recapitulación de la situación en que nos encontramos, podremos darnos cuenta del papel que juega la calculadora que aquí se presenta dentro de este tipo de redes. Partimos de un red Ad-hoc con un número de nodos suficientemente grande para que se puedan transmitir paquetes entre ellos usando un esquema multicamino (mínimo 3 caminos posibles), con un tráfico CBR interferente, una velocidad de nodos y un radio de cobertura establecido. Dentro de este esquema tenemos una o varias fuentes que pretende transmitir un vídeo a uno o varios destinos a través de la red. A la hora de analizar la transmisión tenemos muchas métricas que se pueden contrastar, todas ellas tienen como objetivo caracterizar a la red. Sin embargo, a la hora de ver las prestaciones de la red a nivel de aplicaciones nos encontramos con un parámetro que destaca sobre el resto, la PSNR.

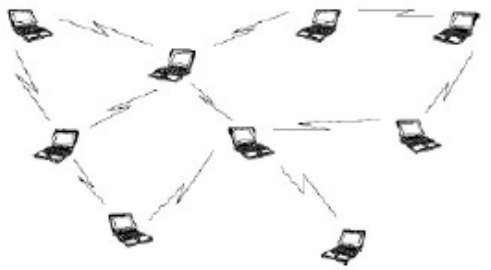

Figura 4.1 Nodos de una red Ad-hoc

La PSNR estima la calidad de una imagen reconstruida en comparación con la imagen original. El objetivo es encontrar un valor que refleje la calidad de la imagen reconstruida. El *Mean Square Error* (MSE) y la PSNR son dos métricas de error empleadas para evaluar la calidad de compresión de una *frame*. El MSE representa el error cuadrático acumulado entre la *frame* original y la comprimida mientras que la PSNR respresenta el error de pico. Para computar la PSNR primero calculamos la MSE entre la *frame* original, P_{org} y su correspondiente decodificada, P_{dec} ambas con el tamañao MxM *pixeles* siguiendo las siguientes ecuaciones:

$$MSE = \frac{1}{M \cdot N} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} [P_{org}(m,n) - P_{dec}(m,n)]^2 \quad (4.1.)$$

$$PSNR = 10 \cdot log \frac{R^2}{MSE} \quad (4.2.)$$

R es el máximo valor posible para el *pixel* de la imagen. Cuando los *pixeles* están representados usando 8 bits por muestra, el valor es 255.

En nuestro caso, tenemos una serie de cuadrados (*frames*) de vídeo que se transmiten a través de la red. Muchas de ellas llegarán deterioradas y nuestro estudio consiste en cuantificarlo para poder analizar así la eficiencia de los protocolos de enrutamiento. Es decir, la PSNR nos dará una medida de la calidad del enrutamiento que conseguimos con los distintos protocolos vistos en el capítulo 2. Así, analizaremos mediante el uso de la calculadora diseñada fácilmente que protocolo de encaminamiento tiene mejores prestaciones y ofrece una mayor PSNR al usuario final.

4.2. Herramientas de diseño

Para poder alcanzar el objetivo de computar el valor de la PSNR, se ha diseñado una calculadora adaptada a las particularidades del entorno con que se trabaja, las redes MANET. Como se explicará a lo largo de las siguientes secciones las herramientas que aquí se presentan, que van desde lenguajes para el manejo de ficheros hasta herramientas *open-source* de computación, son necesarias para alcanzar el objetivo final, la implementación de la calculadora PSNR.

4.2.1. Lenguaje AWK

El awk es un lenguaje de búsqueda y procesado de patrones. Es decir, awk es capaz de buscar un patrón dentro de un fichero y tratarlo según unas operaciones determinadas. La configuración del fichero log de salida del simulador (se verá posteriormente) hace ideal el uso de awk, ya que con él es muy sencillo trabajar con ficheros con sus variables ordenadas por columnas.

Para acceder a una determinada columna en un fichero awk basta con escribir el símbolo $ seguido del número de columna a la que queremos acceder. Así pues con $1 accederemos al valor de la primera columna, con $4 al valor de la cuarta columna y así sucesivamente.

El procedimiento es el siguiente: primero realizamos un fichero .awk donde incluiremos el tratamiento que queremos realizar en los datos. Este fichero es en realidad un filtro que extrae la información que nosotros queremos del fichero de trazas. Una vez hecho el filtro, sólo lo tenemos que aplicar al fichero de log que queramos a través de un shell de Linux. La sintaxis para su ejecución en la shell de Linux es la siguiente:

awk –f filtro.awk log

El fichero .awk debe tener siempre la misma estructura. En el siguiente código podemos ver cómo debe ser el fichero escrito en lenguaje awk.

BEGIN {acción}

{ acción }

END {acción}

La primera parte (BEGIN) se ejecuta una sola vez y antes de leer ninguna línea del fichero. Se suele utilizar para inicializar las variables que vayamos a utilizar y para abrir el fichero o ficheros donde queremos guardar los resultados.

La segunda parte, es el filtrado del fichero. Esta parte se ejecuta para cada línea del fichero. Aquí es dónde debemos indicar el procesado que queremos aplicar a cada una de las líneas presentes en el fichero de log resultado de la simulación.

Por último, tenemos la parte END que se ejecuta al acabar de leer todas las líneas del fichero tratado. Se suele utilizar para escribir en el fichero de salida los resultados del procesado que hemos llevado a cabo en la segunda parte.

4.2.2. Bash Shell Unix

El *Bash Shell* [9] es un intérprete de comandos que sirve de interfaz de comunicación entre el usuario y el sistema operativo. Es el encargado de traducir los comandos introducidos por el usuario en instrucciones que el sistema operativo puede entender y viceversa. *Bash* ha sido escrito por el Proyecto GNU y pertenece a la categoría de interfaz de usuarios en modo texto o carácter.

No es objeto de este proyecto profundizar en todas las variables, comandos y estructuras de control que acepta *Bash*, simplemente enumeraremos aquellas más importantes y que han sido usadas en este proyecto.

En cuanto a comandos:

- **cp**: Copia uno a más fichero en otra ubicación
- **cat**: Muestra por pantalla el contenido de un fichero que se pasa como argumento.
- **grep:** Buscan en un fichero coincidencias con una determinada cadena de caracteres que se le pasa como argumento.
- **mkdir.**: Crea una nueva carpeta.

- **read**: Lee una línea de la entrada estándard.

Estructuras de control:

- **if/else**: Ejecuta una serie de comandos dependiendo de si una cierta condición se cumple o no.
- **while**: Ejecuta una serie de comandos mientras una cierta condición se cumpla.

El motivo por el cual se ha elegido este intérprete de comandos es porque se encuentra integrado en la plataforma UNIX, donde se desarrollan las simulaciones, y además, porque gracias a las funciones y estructuras de control mencionadas anteriormente facilitan el alcanzar los objetivos buscados por la calculadora.

4.2.3. PSNRCORE

Es un comando integrado dentro del paquete *vidprofile*. Este paquete es una colección de herramientas para hacer un testeo completo del mpeg2enc. Los objetivos que persigue este paquete son:

- Encontrar la tasa de bits para la salida de video usando todas las opciones de MPEG-2.
- Encontrar el factor de tiempo cuando codificamos.
- Capturar y conservar una *frame* de cada codificador para su posterior medida.
- Encontrar la PSNR entre la señal de video original y la codificada.

Dentro de este contesto se ha desarrollado el comando *psnrcore* que facilita el último de los objetivos perseguidos por el grupo de desarrolladores que componen el *vidprofile*.

PSNRCORE calcula la PSNR entre las *frames* de dos vídeos. Las *frames* deben estar en formato *Portable Pixel Map* (PPM) y numeradas de manera secuencial (000001.ppm, 000002.ppm, etc). El primer archivo de vídeo es generalmente el original, y el segundo es una versión modificada del original. A menudo, el segundo vídeo se encuentra codificado con un *codec* diferente al primero (o el mismo, con diferentes opciones), o usa filtros para mejorar o cambiar el video.

PSNRCORE compara secuencialmente *frames* de ambos videos, calculando la PSNR para cada *frame* y finalmente la media de PSNR para ambos vídeos. *Frame* por *frame* datos son escritos en un archivo separado por comas (.CSV), mientras la PSNR final es devuelta por la salida estándard.

Los campos obligatorios que necesita el PSNRCORE para funcionar son:

- -o, -original: Especifica la ubicación del directorio de *frames* del video original.
- -c, -compare: Especifica la ubicación del directorio de *frames* del video a comparar.
- -l, -log: Especifíca el *path* (incluído el nombre) del archivo donde la PSNR de cada *frame* será escrita. Si no existe, lo creará. Los datos que se grabarán son:
    - Número de *frames*
    - Y (o luminosidad) PSNR (dB)
    - Cb (Croma Azul) PSNR (dB)
    - Cr (Croma Rojo) PSNR (dB)
    - PSNR frame (dB)
    - Error de PSNR (dB)

- Suma de Error

Un ejemplo de llamada para el comando podría ser:

psnrcore –o ./orig-frames –c ./comp.-frames –l$HOMR/psnr-log.csv

En principio puede parecer que podríamos usar este comando para calcular directamente la PSNR entre los vídeos que hemos transmitido a través de la red. Sin embargo, explicaremos más adelante el porqué de la imposibilidad de hacer esto.

4.2.4. *The Movie Player* (MPLAYER)

MPLAYER [MPL] es un reproductor de películas para Linux (también se ejecuta en muchas otras Unidades, y CPUs no-x86,...). Reproduce la mayoría de los archivos *MPEG, VOB, AVI OGG/OGM, VIVO, ASF/WMA/WMV, QT/MOV/MP4, FLI, RM, NuppelVideo, YUV4MPEG, FILM, RoQ, PVA,* soportados por algunos *codecs* nativos, Xanim, y DLL's Win32. Puede ver VideoCD, SVCD, DVD, 3ivx, DivX ¾/5 e incluso películas WMV también (sin la biblioteca avifile).

Sin embargo, *MPLAYER* es algo más que un reproductor de vídeo muy versátil, sirve también para otras muchas cosas, entre ellas, genera a partir de un archivo de vídeo codificado en MPEG-2, un salida formada por *frames* de dicho video que es configurable tanto en el número de *frames* que se quieren obtener como en los tests de filtros y otras muchas opciones. En nuestro caso, esta herramienta será esencial para la conversión de los archivos de vídeo en *frames*, una posible llamada al reproductor sería:

mplayer –vo pnm blade.m2v

De esta manear, el vídeo blade.m2v sería convertido a *frames* en el directorio actual

4.3. Implementación

A lo largo de esta sección vamos a explicar con detalle las causas que han llevado a la creación de esta calculadora, como se ha creado, cuáles son sus componentes y que resultados se obtienen. Para obtener una visión general de cómo se ha implementado, presentamos la *Fig 4.2*, donde se pueden ver con claridad los parámetros de entrada y de salida que se obtienen de la calculadora. El funcionamiento interno de ésta se verá con más adelante.

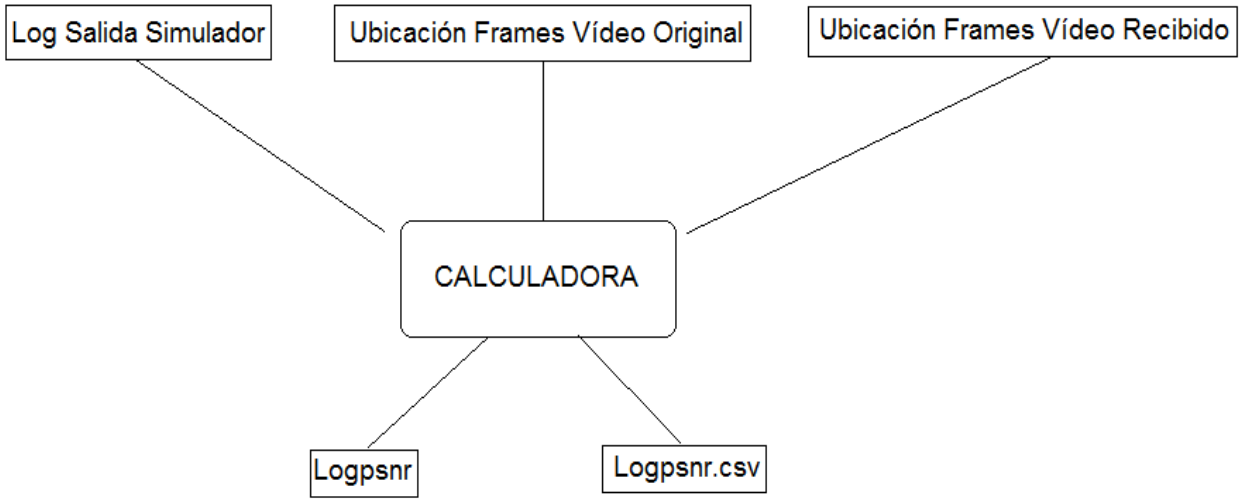

Figura 4.2 Parámetros de Entrada/Salida de la calculadora

4.3.1. Parámetros de Entrada

En el capítulo 5, que se verá posteriormente, se presenta el simulador NS-2 [NS2] y las variaciones necesarias para permitir el tráfico de vídeo, y se comentará que las salidas que se obtienen tras una simulación son, el log de salida y un archivo de vídeo codificado en MPEG-2 [MPG] y compuesto por las *frames* que han sido recibidas en el nodo destino. Si tenemos varias fuentes de vídeo, tendremos varios archivos de vídeo recibidos. Además de estos parámetros de entrada, tenemos el archivo de vídeo original, también codificado en MPEG-2, que se transmite desde el nodo fuente. Todos juntos constituyen los argumentos de entrada de la calculadora.

Analizamos separadamente los parámetros de entrada enumerados anteriormente:

1- Log Salida del Simulador

Es un fichero de texto con información generada por el simulador. Donde podemos distinguir varias partes y que nos serán de mucha utilidad.

```
psnr writing0 seqno= 2931 cseqno= 51 cipb_cseqno= 34 psnr_cseqno= 34 numframe= 2258 size= 1210 srcid= 0
psnr writing0 seqno= 2932 cseqno= 54 cipb_cseqno= 34 psnr_cseqno= 34 numframe= 2258 size= 722 srcid= 0
psnr writing0 seqno= 2932 cseqno= 54 cipb_cseqno= 35 psnr_cseqno= 35 numframe= 2258 size= 722 srcid= 0
psnr writing0 seqno= 2933 cseqno= 55 cipb_cseqno= 35 psnr_cseqno= 35 numframe= 2259 size= 435 srcid= 0
psnr writing0 seqno= 2933 cseqno= 55 cipb_cseqno= 36 psnr_cseqno= 36 numframe= 2259 size= 435 srcid= 0
psnr writing0 seqno= 2934 cseqno= 56 cipb_cseqno= 36 psnr_cseqno= 36 numframe= 2260 size= 314 srcid= 0
psnr writing0 seqno= 2934 cseqno= 56 cipb_cseqno= 37 psnr_cseqno= 37 numframe= 2260 size= 314 srcid= 0
psnr writing0 seqno= 2935 cseqno= 57 cipb_cseqno= 37 psnr_cseqno= 37 numframe= 2261 size= 373 srcid= 0
psnr writing0 seqno= 2935 cseqno= 57 cipb_cseqno= 38 psnr_cseqno= 38 numframe= 2261 size= 373 srcid= 0
psnr writing0 seqno= 2936 cseqno= 58 cipb_cseqno= 38 psnr_cseqno= 38 numframe= 2262 size= 220 srcid= 0
psnr writing0 seqno= 2936 cseqno= 58 cipb_cseqno= 39 psnr_cseqno= 39 numframe= 2262 size= 220 srcid= 0
psnr writing0 seqno= 2937 cseqno= 59 cipb_cseqno= 39 psnr_cseqno= 39 numframe= 2263 size= 236 srcid= 0
```

En esta primera sección, podemos ver que cada linea se corresponde con un paquete que se ha recibido en el nodo destino. Los campos más significativos son:

- ***numframe***: Es un identificador para cada *frame*. El archivo original se convierte en *frames* antes de ser transmitido desde el nodo fuente, y para cada *frame* hay un identificador, este identificador se corresponde con este campo del log. Luego veremos su importancia.
- ***size***: El tamaño de los paquetes es configurable en el fichero .tcl (archivo de configuración que se verá más adelante) del simulador.

Gracias a este campo podemos ver que hay paquetes que no van llenos y lo más probable es que hayan sufrido algún deterioro.

- ***srcid***: Es un identificador para el nodo fuente. Será muy útil a la hora de tener varias fuentes transmitiendo para poder distinguir que *frames* pertenecen a cada una.

La siguiente sección destacable del log de salida la tenemos a continuación.

```
INIT of ITERATION 2: 21.000  FLOW: 0 I_am_a_source= 1 net_id=0

GenerateProbeMessage() net_id=0 actual_dest_[0]=18 [1]=NONE Probe size_cache = 3

        Sending Probe!!! pm.dest = 18, pm.route = [(0) 6 7 18 ] i = 0 pm.route.backup = [(0) 6 7 18 ]

srh->probe_PEP()=0.498911 RxPr_info_[6].PEL=0.498911 node=0 vecino=6

PM cmnh =-48 size_=0 IP_HDR_LEN=20        Sending Probe!!! pm.dest = 18, pm.route = [(0) 17 18 ] i = 1 pm.route.backup = [(0) 17 18 ]

srh->probe_PEP()=0.405137 RxPr_info_[17].PEL=0.405137 node=0 vecino=17

PM cmnh =-48 size_=0 IP_HDR_LEN=20        Sending Probe!!! pm.dest = 18, pm.route = [(0) 14 18 ] i = 2 pm.route.backup = [(0) 14 18 ]

srh->probe_PEP()=0.451894 RxPr_info_[14].PEL=0.451894 node=0 vecino=14

PM cmnh =-48 size_=0 IP_HDR_LEN=20GenerateProbeMessage() net_id=0 actual_dest_[0]=18 [1]=NONE Probe size_cache = 0

Probes enviados size_cache=0!!!

srh->probe_PEP()=0.202305 RxPr_info_[7].PEL=0.405493 node=6 vecino=7

srh->probe_PEP()=0.106277 RxPr_info_[18].PEL=0.525328 node=7 vecino=18

PM ha llegado, viene del nodo 0

srh->probe_PEP_ult()=0.893723 PEP_threshold=0.950000 PMinfo_RX[PM_RX-1].PEP=1

        ---- Receiving Probe Message node=18 p.route=[0 6 7 (18) ] srh->probe_route_string()=[(0) 6 7 18 ]

srh->probe_PEP()=0.163830 RxPr_info_[18].PEL=0.404382 node=17 vecino=18

PM ha llegado, viene del nodo 0

srh->probe_PEP_ult()=0.836170 PEP_threshold=0.950000 PMinfo_RX[PM_RX-1].PEP=1

        ---- Receiving Probe Message node=18 p.route=[0 17 (18) ] srh->probe_route_string()=[(0) 17 18 ]

srh->probe_PEP()=0.207090 RxPr_info_[18].PEL=0.458272 node=14 vecino=18

PM ha llegado, viene del nodo 0

srh->probe_PEP_ult()=0.792910 PEP_threshold=0.950000 PMinfo_RX[PM_RX-1].PEP=1
```

```
---- Receiving Probe Message node=18 p.route=[0 14 (18) ] srh->probe_route_string()=[(0) 14 18 ]


All ProbeMessages have arrived! PM_RX=3
Generate PMR is going to be done!!


GeneratePMR INIT nodeid=18
Sending PMR pmr.dest = 0 pmr.route = [(18) 7 6 0 ] pmr_info_size()=3
Sending PMR pmr.dest = 0 pmr.route = [(18) 17 0 ] pmr_info_size()=3
Sending PMR pmr.dest = 0 pmr.route = [(18) 14 0 ] pmr_info_size()=3
GeneratePMR END nodeid=18
Generation of PMR is done!! NEXT PM_actual_iteration_RX = 2
********** Probe Message REPLY Received!!! srh->pmr_info_size()=3
To SelectBestPaths(); size_cache_PM=0                                    net_id=0
dos2
FINAL ASSIGNMENT: num_paths=3
ROUTES IN ORDER!
P E R I O D  MMDSRfreq=26.000000
```

Se corresponde con la iteración del algoritmo de enrutamiento multicamino del MMDSR. Algunos de sus elementos pueden diferir en función de las distintas versiones del protocolo MMDSR que se vio anteriormente. Destaca como se envían los mensajes *Probe Messages* a través de la red, como se descubren los caminos y como se clasifican finalmente. Este información es esencial para comprender como se esta comportando la red.

Por último, otra sección destacable dentro de este fichero log es la proporcionada por el protocolo RTP. A continuación vemos un ejemplo:

```
s_->Icounter()=118, s_->Pcounter()=162 s_->Bcounter()=296
s_->I_sent_counter()=544, s_->P_sent_counter()=756 s_->B_sent_counter=1783
srcid() = 0 %_I_LOST= 78.30882 %_P_LOST= 78.57143 %_B_LOST= 83.39877
```

rtcph2->ehsr() = 3455

rtcph2->srcid() = 0

rtcph2->flow_id() = 0

ehsr_prev_ = 3191

pack_intRR = 264

rtcph2->cnopl() = 2816

np = 639

time = 71.747660

rtcph2->flost() = 0.00000

rtcph2->ehsr()=3455 np=639 cnopl_prev_=2816 pack_intRR=264

session_->jitter() = 0.003256

RTCPRR flost()= 0.00000 cnopl()= 2816 jitter()= 3.256016e-03 s->ehsr()=3455 srcid=0

Dentro de esta sección se pueden ver los campos:

- ***%_I_LOST, %_P_LOST, %_B_LOST***: Nos proporciona el porcentaje de *frames* del tipo I, P y B que se han perdido al viajar a través de la red.
- ***time***: instante en el que se recoge la información.
- ***jitter***: Este valor se corresponde con el *delay jitter* en el instante dado por *time*.

2. Video Recibido

Es un archivo de video codificado en MPEG-2 que esta compuesto por las *frames* que han llego al nodo destino. Si durante la transmisión se han perdido *frames*, estas no estarán incluidas aquí, además si reproducimos este archivo se podrá ver su deterioro en caso de que lo hubiera. Tras usar la herramienta *MPLAYER* obtendremos las *frames* del tipo PPM, 00000001.ppm, 00000002.ppm, 00000003.ppm… Y es en este formato como se le pasa a la calculadora.

Figura 4.3. Frame Video Original

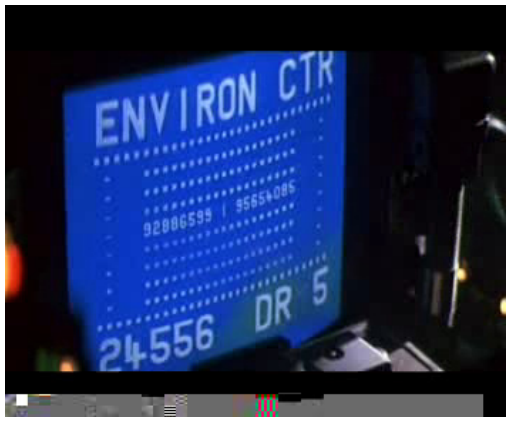

Figura 4.4 Frame Video Recibido

3. Video Original

Por la definición que hemos dado anteriormente de PSNR se puede deducir que para conocer el valor de este parámetro necesitamos tener la *frame* original para ver lo que difiere con la recibida. Por tanto, el video original se convierte en un parámetro imprescindible para poder computarlo. Igual que en el Video Recibido, lo que se le pasa a la calculadora es la ubicación de las *frames* del Video Recibido. En las *Fig 4.3* y *Fig 4.4* se pueden ver una *frame* Original y una Recibida respectivamente. Como podemos ver la Recibida tiene algo de distorsión y cuando tras realizar el cómputo de la PSNR se obtendría un valor bajo, próximo a 20 dB.

4.3.2. Planteamiento del Problema

Una vez vistos los parámetros de entrada estamos en condiciones de plantear el problema de la calculadora. Como se comentó en la sección anterior, para computar la PSNR se utiliza la herramienta *psnrcore*. El problema reside en que esta herramienta compara *frame* con *frame* y de manera secuencial. Es decir, se compara la *frame* 00000001.ppm del vídeo origen con la *frame* 00000001.ppm del vídeo recibido y así sucesivamente. Sin embargo, en una situación real, que se simula con el NS-2 se obtienen pérdidas por lo que no todas las *frames* que se envían llegarán a su destino. Esto se traduce en la imposibilidad de aplicar el *psnrcore* directamente ya que podríamos estar comparando *frames* equivocadas. Lo aclaramos con el siguiente ejemplo:

Supongamos un vídeo origen que tras aplicarle el *MPLAYER* conocemos que esta compuesto por 10 *frames*. Este vídeo es transmitido a través de una red MANET (simulada con el NS-2), dentro de esta red se producen pérdidas, y en el destino hay algunas de estas *frames* que se han perdido, por ejemplo, 2, como se ve en la *Fig 4.5*.

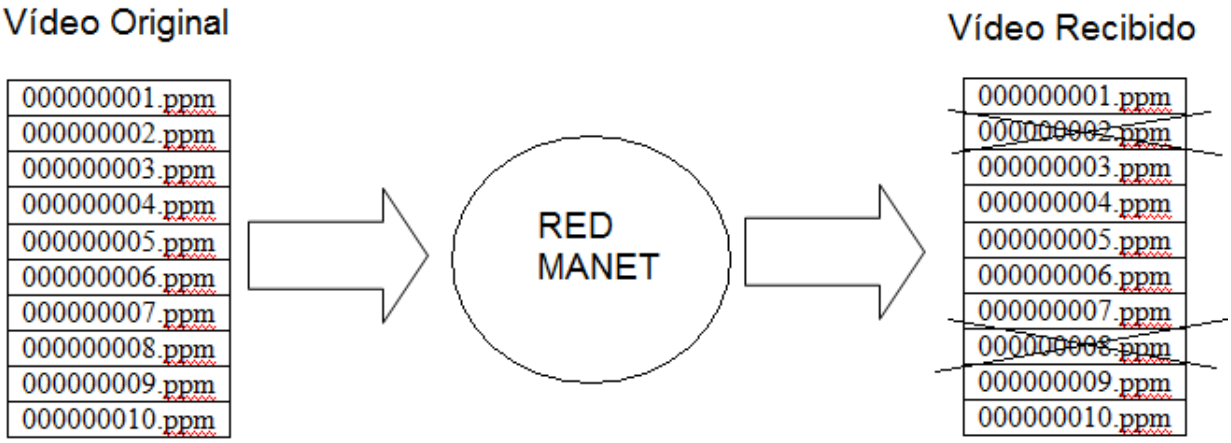

Figura 4.5 Esquema de transmisión de frames a través de una red MANET

Como se ha dicho anteriormente, el NS-2, nos devuelve un archivo de vídeo recibido, que tras aplicarle el *MPLAYER* nos muestra 8 *frames* (se habían perdido 2). Tanto el archivo original como el recibido tienen sus *frames* numeradas secuencialmente para el problema es que no se corresponden, y al aplicar el *psnrcore* sin ninguna adaptación obtenemos resultados completamente erróneos como se ve en la siguiente *Fig. 4.6*:

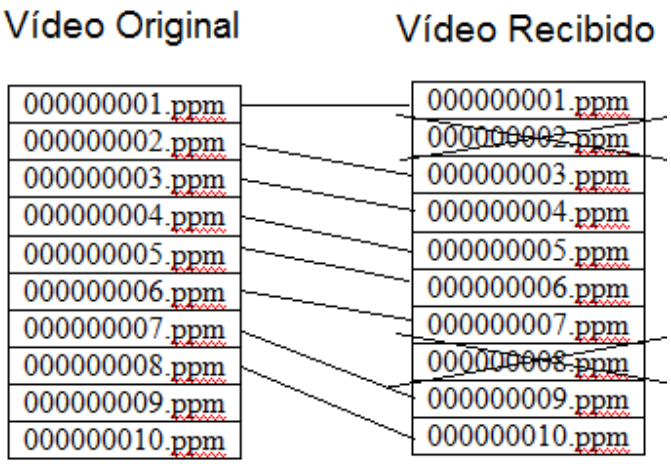

Figura 4.6 Comparación de frames usando psnrcore

Por tanto, el objetivo de la calculadora es tratar de resolver este problema, es decir, conseguir alinear las *frames* de manera que cuando apliquemos el *psnrcore*, se haga sobre las *frames* correctas y obtengamos valores reales de PSNR que nos permitan conocer el estado real de la red.

4.3.3. Solución: *PSNRCOMPUTATOR*

Para conseguir resolver el problema planteado en la sección anterior e implementar una calculadora con prestaciones suficientes como para ser aplicada a simulaciones de tráfico de vídeo en redes MANET hemos diseñado nuestro *pnsrcomputator*. Vemos paso por paso que hace esta calculadora de

PSNR para conseguir preparar los datos de manera que se pueda aplicar la herramienta *psnrcore*.

- Se filtra el fichero log de salida del simulador para que solo aparezca la información referente a las *frames* que se han recibido en el nodo destino.

```
psnr writing0 seqno= 104 cseqno= 3 cipb_cseqno= 0 psnr_cseqno= 0 numframe= 45 size= 476 srcid= 0
psnr writing0 seqno= 104 cseqno= 3 cipb_cseqno= 1 psnr_cseqno= 1 numframe= 45 size= 476 srcid= 0
psnr writing0 seqno= 108 cseqno= 4 cipb_cseqno= 1 psnr_cseqno= 1 numframe= 48 size= 289 srcid= 0
psnr writing0 seqno= 108 cseqno= 4 cipb_cseqno= 2 psnr_cseqno= 2 numframe= 48 size= 289 srcid= 0
psnr writing0 seqno= 109 cseqno= 6 cipb_cseqno= 3 psnr_cseqno= 3 numframe= 49 size= 317 srcid= 0
psnr writing0 seqno= 109 cseqno= 6 cipb_cseqno= 4 psnr_cseqno= 4 numframe= 49 size= 317 srcid= 0
psnr writing0 seqno= 113 cseqno= 5 cipb_cseqno= 2 psnr_cseqno= 2 numframe= 53 size= 601 srcid= 0
psnr writing0 seqno= 113 cseqno= 5 cipb_cseqno= 3 psnr_cseqno= 3 numframe= 53 size= 601 srcid= 0
psnr writing0 seqno= 116 cseqno= 7 cipb_cseqno= 4 psnr_cseqno= 4 numframe= 56 size= 1243 srcid= 0
psnr writing0 seqno= 116 cseqno= 7 cipb_cseqno= 5 psnr_cseqno= 5 numframe= 56 size= 1243 srcid= 0
psnr writing0 seqno= 117 cseqno= 8 cipb_cseqno= 5 psnr_cseqno= 5 numframe= 56 size= 1280 srcid= 0
psnr writing0 seqno= 117 cseqno= 8 cipb_cseqno= 6 psnr_cseqno= 6 numframe= 56 size= 1280 srcid= 0
```

- A partir de este fichero filtrado, se obtiene el identificador de las *frames* (idle) que se han recibido y el número de fuentes de vídeo que están transmitiendo.

```
45    48    49    52    53    55    56    59    62
```

- Con esta información, idle y el número de fuentes, se reconstruye el nombre de las *frames* que se deben mover del archivo de video original para poder compararlas con las *frames* del vídeo recibido.

```
00000045.ppm    00000048.ppm    00000049.ppm    00000052.ppm
00000053.ppm    00000055.ppm    00000056.ppm    00000058.ppm
00000059.ppm    00000062.ppm
```

- Como deben de estar ordenadas de manera secuencial, al realizar la

copia se renombrar las *frames* de manera que empiece en la 00000001.ppm y acabe en la última.

00000001.ppm	*00000002.ppm*	*00000003.ppm*	*00000004.ppm*
00000005.ppm	*00000006.ppm*	*00000007.ppm*	
00000008.ppm	*00000009.ppm*	*00000010.ppm*	

- Una vez construidos los nombres, se crea un bucle que copia las *frames* del fichero original en un directorio destino, esta *frame* se corresponden exactamente con aquellas que se han recibido.
- Se aplica la herramienta *psnrcore* utilizando como argumentos las *frames* que han sido copiadas en los pasos anteriores y las *frames* del fragmento de video recibido a la salida del simulador.

"Frame"," Y (dB)"," Cb (dB)"," Cr (dB)"," PSNR (dB)"," Error"," Error Sum"

1,"13,53","21,72","16,02","18,58","0,03","0,03"

2,"12,89","19,2","14,51","19,75","0,04","0,08"

3,"13,19","19,54","14,51","20,01","0,04","0,12"

4,"18,07","29,1","26,71","22,6","0,01","0,13"

5,"18,14","30,46","28,45","26,74","0,01","0,14"

6,"17,77","30,41","28,66","27,39","0,01","0,15"

7,"12,92","20,8","15,73","26,99","0,04","0,19"

8,"18,18","28,96","29,36","29,77","0,01","0,2"

9,"13,18","21,01","15,68","22,22","0,04","0,24"

10,"13,23","21,04","15,67","21,26","0,04","0,28"

4.3.4. Esquema de la solución

Para poder desarrollar los pasos mencionados anteriormente se han elaborado cuatro scripts:

- **psnrcalculator**: es el programa que se encarga de coordinar la llamada al resto. Además es donde se desarrolla el bucle principal de copia de *frames*.
- **select_idle**: se encarga de obtener el identificador de las *frames* recibidas (idle) y calcular el número de fuentes de video que están transmitiendo.
- **create_name**: a partir del idle y de número de fuentes se generan los nombres de las *frames* a mover.
- **create_seq:** crea una secuencia de nombres de *frames* tan larga como el número de *frames* distintas recibidas en el nodo destino.

Estos scripts se encuentran integrados dentro del siguiente esquema para que se comprenda mejor la secuencia de los acontecimientos.

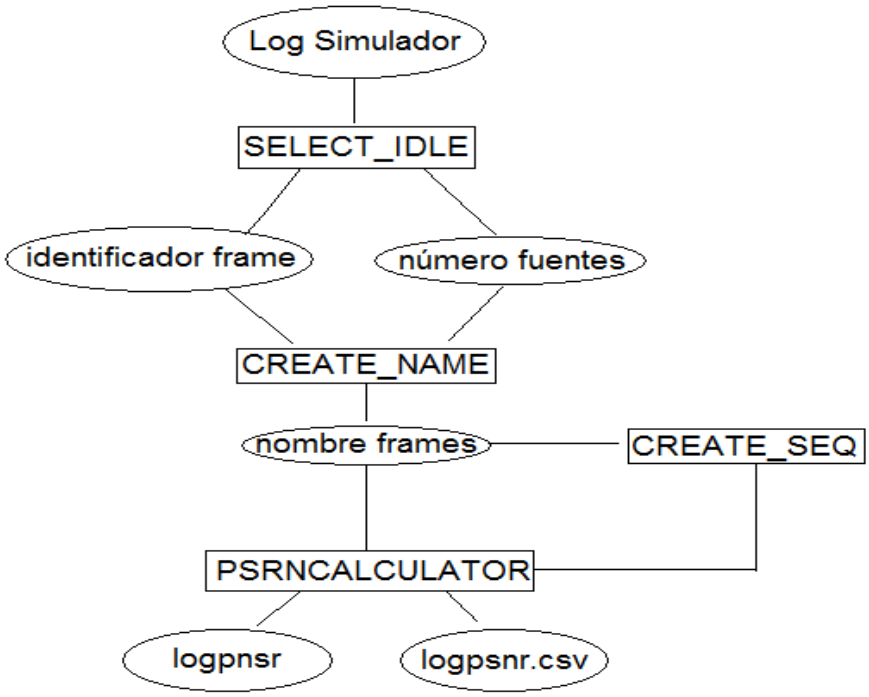

Figura 4.7 Esquema general de la calculadora

4.3.5. Composición de la calculadora

A lo largo de esta sección se va a mostrar el código de los distintos programas y subprogramas que componen la calculadora. Como ya se ha comentado, son cuatro scripts que son llamados desde *psnrcomputator* que actúa como *main*.

Primero vamos a presentar a ***psnrcomputator***. Este script esta programado en *bash shell*. La elección de este lenguaje es debida a que se tienen que hacer copia de ficheros y la opción más directa es con comandos del *shell*. Desde este mismo programa se hacen llamadas a subprogramas escritos en *AWK* como veremos a continuación. La salida son ficheros de la forma *logpsnr"i".csv* y *logpnsr"i"*, tantos como fuentes de vídeo estén transmitiendo en el escenario. En estos ficheros está contenido el valor de la PSNR, como veremos en la siguiente sección.

Ahora mostramos el código con comentarios intercalados para facilitar su compresión:

```
# PSNRCOMPUTATOR


linea="algo"
i=0
j=0


# Con este comando se filtra el fichero log de salida del simulador de manera que solo nos queden aquellas líneas que nos dan información de los paquetes
cat $1 | grep "numframe" > logX


# Se selecciona el número de fuentes de video y las idle de las frames que se han recibido
awk -f select_idle logX
```

sources=(`cat num_sources | tr '\n' ' '`)

 # Bucle principal. Para cada fuente de video se realizan las siguientes operaciones

while [$i -le $sources]; do

 # En la siguiente línea a partir de la idle se crea el nombre de la *frame* a mover

awk -f create_name idle"$i" > namefilesX"$i"

 # Para que las *frames* puedan ser comparadas usando el *psnrcore*, debemos ponerlas en orden (secuencia numérica). Esta línea crea la secuencia en orden

awk -f create_seq namefilesX"$i" > sequenceX"$i"

num=(`cat sequenceX"$i" | tr '\n' ' '`)

 # Subbucle que a partir del nombre de los ficheros a mover y la secuencia creada mueve de la carpeta donde están todas las *frames* del video original solo las *frames* que se han recibido para ser comparadas

mkdir OriginalX"$i"
while [! -z "$linea"]
do
read linea
if [! -z "$linea"]
then
 # cp /home/bass/Documentos/Proyecto/Calculator/Complete/$linea

```
./OriginalX"$i"/${num[$j]}
        cp /$2/$linea ./OriginalX"$i"/${num[$j]}

        let j=$j+1
        fi

        done < namefilesX"$i"

# Para cada fuente de video se calcula la PSNR
        if [ $i -eq 0 ];
        then
        psnrcore -o ./OriginalX"$i"/ -c ./$3/ -l ./logpsnr"$i".csv > logpsnr"$i"
        elif [ $i -eq 1 ];
        then
        psnrcore -o ./OriginalX"$i"/ -c ./$4/ -l ./logpsnr"$i".csv > logpsnr"$i"
        elif [ $i -eq 2 ];
        then
        psnrcore -o ./OriginalX"$i"/ -c ./$5/ -l ./logpsnr"$i".csv > logpsnr"$i"
        fi

        j=0
        linea="algo"
        let i=$i+1
done
```

A continuación, se muestra el código del siguiente subprograma que es llamado desde *psnrcomputator*, llamado **select_idle**. Este script esta implementado en lenguaje *AWK* ya que realiza operaciones sobre un fichero de texto, el log de salida del simulador. Se encarga, en primer lugar, de seleccionar el número de

fuentes de vídeo que están transmitiendo y posteriormente para cada fuente selecciona el identificador de *frame* que ha llegado. La salida se muestra en los ficheros *idleX"i*" y *numsources* respectivamente. Habrá tantos ficheros del primero como fuentes de video transmitiendo.

Como se comentó a la hora de explicar los pasos de funcionamiento de la calculadora, estos ficheros son parámetros de entrada para el siguiente script **create_name**. Vemos en detalle el código, comentado entre líneas:

```
# SELECT_IDLE


# Se declaran las variables que servirán de contadores y para almacenar información
BEGIN{
        aux2 = 0
        aux1 = 0
        num_sources = 0
        cont = 0
        j=0
}
{
# Con este primer bucle se obtiene el número de fuentes de video y este valor se almacena en la variable num_sources.
        while ( cont <= NR)
        {
                if($1 == "psnr")
                {
# $16 hace referencia al contenido del campo srcid del log de salida del simulador.
                        if ($16 != 0)
                        {
```

```
                        if ( $16 != aux)
                        {
                                if ( $16 > aux)
                                {
                                        num_sources = num_sources + 1
                                        aux = num_sources
                                }
                        }
                }
        }
                cont = cont + 1


}
```

 # En este bucle se selecciona el identificador de *frame* para cada *frame* recibida. Este identificador se manda al fichero *ilde"i"* que se corresponde con cada fuente de video

```
        if($1 != 0)
        {
                if($1 == "psnr")
                {
                        if($12 != aux2)
                        {
```

#Si el campo $12 (*numframe)* es igual a -1 indica que no es un paquete con datos sino que es de señalización

```
                                if($12 != -1)
                                {
                                        printf($12"\n") > "idle"$16
                                        aux2 = $12
                                }
```

```
                                    }
                                }
                            }
}
END{
        printf(num_sources) > "num_sources"
}
```

Tras la ejecución de este subprograma contamos con una valiosa información. Por un lado tenemos el número de fuentes de video que están transmitiendo en la red, por otro, disponemos de los identificadores de las *frames* recibidas para cada una de ellas. Es decir, sabemos que *frames* se deben comparar con que otras *frames* pero el problema reside en que estos idle no se encuentran en el formato adecuado. El formato del nombre de una *frame* es “00000001.ppm”, por ejemplo, y lo que tenemos ahora sería simplemente “1”. Para corregir este problema se ha diseñado el siguiente subprograma. Está escrito también en lenguaje *AWK* ya que su eficiencia en el manejo de cadenas de caracteres es asombrosa. El propósito de este programa es a partir de las idle crear los nombres de *frames* en el formato adecuado. Para ello nos basamos en que el formato del nombre de las *frames* debe ser siempre el mismo, PPM de 8 bits seguidos de la extensión pertinente.

La salida del subprograma es un fichero con el nombre de las *frames* que se han recibido. Vemos el código del programa a continuación:

```
# CREATE_NAME
{
# Se crea una tabla de cadena de caracteres con un tamaño igual al número de líneas del fichero que se le pasa como argumento, es decir, igual al número de frames que se han recibido y se almacena la ilde que contiene el argumento de entrada
 linea [NR] = $0
```

 # Esta bandera será explicada en su momento solo mencionar que en principio se encuentra activa

```
flag = 1
i = 1
j = 1
aux = 0
}
END {
```

 # Este bucle ordena las idles de menor a mayor por si se hubiera producido alguna desincronización en la transmisión

```
for (i=1; i<= NR ; i=i+1){
        while ( j < i )
        {
                if ( linea[j] > linea[i])
                {
                        aux = linea [j]
                        linea[j] = linea[i]
                        linea[i] = aux
                        j=j+1
                }
                else
                {
                        j=j+1
                }
        }
        linea[j] = linea[i]
        j=1
}
```

```
# Este bucle se ejecuta tantas veces como frames se hayan recibido
for (i=1; i <= NR ; i= i+1){
# Para cada idle se obtiene su tamaño ya que como conocemos que el formato del nombre de frames es de 8 bits
        tam = length (linea[i])
        if(tam == 1)
        {
# En función de su tamaño se crea una cadena de caracteres construyendo el nombre
          cadena = "0000000"linea[i]".ppm"
        }
        else
        {
                if(tam == 2)
                {
                cadena = "000000"linea[i]".ppm"
                }
                else
                {
                        if(tam == 3)
                        {
                        cadena = "00000"linea[i]".ppm"
                        }
                        else
                                {
                                if(tam == 4)
                                {
                                cadena = "0000"linea[i]".ppm"
                                }
```

```
else
{
    if(tam == 5)
    {
     cadena = "000"linea[i]".ppm"
    }
    else
    {
        if(tam == 6)
        {
         cadena = "00"linea[i]".ppm"
        }
        else
        {
            if(tam == 7)
            {
             cadena = "0"linea[i]".ppm"
            }
            else
            {
                if(tam == 8)
                {
                 cadena = linea[i]".ppm"
                }
                else
                {
```

 # La bandera es un mecanismo de seguridad. Si se produce algún error en el procesado, algún carácter de más que se introduce, este nombre saldrá invalido y desvirtúa el

computo global.

```
                                                    flag = 0
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                    if ( flag != 0)
                    {
```

 # Se imprime cada nombre en la salida Standard que esta redirigida a un fichero llamado *nameX"i"*

```
                    printf(cadena"\n")
                    }
                    else
                    {
                    flag = 1
                    }
}
}
```

El último subprograma que compone la calculadora es llamado **create_seq**. Como ya se comentó a la hora del planteamiento del problema, la herramienta *psnrcore* evalúa las *frames* de manera secuencial. Hasta ahora, nosotros hemos conseguido disponer de las *frames* que se han recibido, de sus nombres pero estas no se encuentran ordenadas de manera ascendente y secuencial. Para ello se ha elaborado este subprograma, también escrito en lenguaje *AWK*. Este subprograma recibe el número de *frames* que se han recibido y con ello elabora

una secuencia de nombres de *frames* para que la comparación sea perfecta. Su metodología es muy parecida a la de su predecesor. Por tanto, la salida es un fichero con una secuencia del tamaño exacto al número de *frames* que se han recibido. Vemos el código a continuación:

```
# CREATE_SEQ


# Se crea una tabla de cadena de caracteres del tamaño del número de frames recibidas
{
 linea [NR] = $0
}
END {
# Se elabora un bucle en el que se crean los nombres de manera secuencial hasta alcanzar el número de frames recibidas
 for (i=1; i <= NR ; i= i+1){
        if (i < 10)
        {
        cadena = "0000000"i".ppm"
        }
        else
        {
                if (i < 100)
                {
                        cadena = "000000"i".ppm"
                }
                else
                {
                        if (i < 1000)
                        {
```

```
                cadena = "00000"i".ppm"
        }
        else
        {
                if (i < 10000)
                {
                        cadena = "0000"i".ppm"
                }
                else
                {
                        if (i < 100000)
                        {
                                cadena = "000"i".ppm"
                        }
                        else
                        {
                                if (i < 1000000)
                                {
                                        cadena = "00"i".ppm"
                                }
                                else
                                {
                                        if (i < 10000000)
                                        {
                                                cadena = "0"i".ppm"
                                        }
                                        else
                                        {
```

```
                                                    cadena = i".ppm"
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
```

 # La salida se redirige a un fichero llamado *sequenceX"i"*

```
 printf(cadena"\n")
}
}
```

4.3.6. Salida de la calculadora

El objetivo fijado para la calculadora es el de obtener un cálculo de la PSNR entre unas *frames* que componen un video original y aquellas que se han recibido tras viajar a través de una red MANET. Por tanto, la salida de la calculadora no puede ser otra que la PSNR.

Sin embargo, la salida de la calculadora va un poco más allá. Tenemos dos ficheros de salida. El primero lo vemos a continuación y nos da el valor medio de la PSNR de todas las *frames* que se han recibido comparadas con sus homologas en el vídeo transmitido:

1. logpsnr

```
-e 21.704612084748345835 70
```

En segundo lugar, tenemos el fichero que contiene los valores individuales de la

PSNR para cada una de las *frames*, su valor medio y otra información adicional.

2. logpsnr.csv

"Found 2996 original frames and 2911 comparison frames!",,,,,,

"Using the first 2911 for PSNR calculation,",,,,,,

"Frame"," Y (dB)"," Cb (dB)"," Cr (dB)"," PSNR (dB)"," Error"," Error Sum"

1,"11,32","19,12","15,45","12,52","0,06","0,06"

2,"11,97","19,59","15,43","13,1","0,05","0,1"

3,"16,39","26,43","25,72","17,92","0,02","0,12"

4,"15,97","26,39","25,78","17,52","0,02","0,14"

5,"16,1","26,39","26,02","17,65","0,02","0,16"

6,"16,25","26,36","26,09","17,8","0,02","0,17"

7,"16,46","26,48","26,16",18,"0,02","0,19"

8,"16,59","26,51","26,09","18,12","0,02","0,2"

9,"13,72","20,96","15,7","14,67","0,03","0,24"

10,"20,95","28,31","27,52","22,29","0,01","0,24"

11,"14,46","20,85","15,91","15,3","0,03","0,27"

12,"14,41","20,87","15,89","15,26","0,03","0,3"

13,"14,38","20,83","15,91","15,23","0,03","0,33"

14,"14,39","20,88","15,87","15,24","0,03","0,36"

15,"14,26","20,88","15,84","15,13","0,03","0,39"

16,"14,34","20,86","15,82","15,19","0,03","0,42"

.....

.....

.....

2906,"39,45","47,09","46,26","40,82",0,"19,66"

2907,"39,61","47,08","46,27","40,96",0,"19,66"

2908,"38,78","47,09","46,19","40,2",0,"19,66"

2909,"38,32","47,02","46,72","39,79",0,"19,66"
2910,"36,54","46,88","46,73","38,1",0,"19,66"
2911,"35,26","46,57","46,66","36,86",0,"19,66"
"2911 frames"," AVG"," AVG"," AVG","21,7"," AVG","19,66"

4.4. Conclusiones

A lo largo de las secciones de este capítulo se han desglosado los aspectos más significativos referentes al diseño y a la implementación de la calculadora de PSNR que es el eje fundamental de este proyecto fin de carrera.

Podemos concluir que se ha diseñado una calculadora versátil, que realiza su tarea de manera eficiente, sin un código complicado y poco entendible. De todas manera, para terminar de clarificar se presenta un resumen a modo de *man*.

NOMBRE

psnrcomputator – calcula la *peak signal-to-noise* (PSNR) entre las *frames* de dos vídeos

SINOPSIS

psnrcomputator logSimulador FramesOriginal FramesRecibido0 FramesRecibido1 FramesRecibido2

DESCRIPCIÓN

Calcula la PSNR entre las *frames* de dos vídeos cuando el vídeo recibido no tiene el mismo tamaño que el original

OPCIONES REQUERIDAS

Las opciones mínimas que permite el funcionamiento de la calculadora son:

logSimulador: Fichero de texto de salida del simulador NS-2

FramesOriginal: *Path* de la ubicación donde se encuentran las *frames* del vídeo Original.

FramesRecibido: *Path* de la ubicación donde se encuentran las *frames* del vídeo Recibido

Capítulo 5: Herramientas de simulación

5.1 Simulador NS-2

Para llevar a cabo el estudio del comportamiento de las redes MANET con diferentes protocolos de encaminamiento y bajo diversos modelos de movilidad de los nodos, elegimos una herramienta de simulación que debía ser ampliamente utilizada, de reconocido prestigio, de libre acceso y que incluyera ya los protocolos propios de las redes MANET. El simulador escogido fue *Network Simulator* [NS2].

Network Simulator (NS) es un simulador de red dirigido por eventos, encapsulado dentro del lenguaje Tcl (*Tool Command Language*). El motor del simulador está implementado en C++ y está configurado y controlado mediante la interfaz Tcl. El simulador se invoca vía el intérprete de comandos del *NS*, que es una extensión del *shell otclsh.*

La simulación se define mediante un programa Tcl. Utilizando las instrucciones de *NS* definimos la topología de la red, configuramos las fuentes de tráfico, se recogen las estadísticas en un fichero de salida y se invoca el simulador. Las instrucciones de *NS* permiten invocar los procedimientos Tcl desde puntos arbitrarios de la simulación, ofreciendo un mecanismo flexible que permite modificar desde la topología de la simulación, hasta registrar acontecimientos que de forma estándar no se registran o se hacen en formatos diferentes.

Una topología de red se define mediante tres primitivas que construyen los bloques: nodos (*nodes*), enlaces (*links*) y agentes (*agents*).

Los nodos se crean mediante la instrucción "*ns node*" y se enlazan en una topología de red con la instrucción "*ns link*". Los nodos son objetos pasivos, que son dirigidos por los agentes, los cuales son los objetos que activamente conducen la simulación. Ejemplos de agentes son las fuentes de tráfico constante o módulos de encaminamiento dinámico, y se crean mediante la instrucción "*ns agent*".

5.1.1. El proyecto VINT (*Virtual Internetwork Testbed*)

El simulador NS forma parte del proyecto VINT. Este proyecto es una colaboración entre USC/ISI (*University of Southern California's Information Sciences Institute*), *Xerox* PARC (*Xerox Palo Alto Research Center*), LBNL (*Lawrence Berkeley National Laboratory*) y UCB (*University of California, Berkeley*), y tiene como objetivo crear un simulador de redes que permita estudiar el escalado y la interacción entre protocolos en la actualidad y en los futuros protocolos de red. Actualmente se encuentra en su versión 2 (NS-2), aunque globalmente sea una versión beta. La versión estable más reciente del

simulador de red NS es la ns-2.33 (del 31 de Marzo del 2008).

Los componentes necesarios son los siguientes (se indica también la última versión disponible en el momento de escribir este documento):

Componentes obligatorios:

a) Tcl/Tk (versiones 8.4.14)

b) Otcl (versión 1.13)

c) TclCL (versión 1.19)

d) ns-2 (versión 2.33)

Componentes opcionales:

a) nam-1 (versión 1.13)

b) xgraph (versión 12.1)

c) Cweb (versión 3.4g)

d) SGB (versión 1.0)

e) Zlib (versión 1.2.3)

5.1.2. Procedimiento de Instalación

Este *software* fue creado para su uso en sistemas operativos Linux, aunque existe también una versión para Windows, aunque no muy fiable. En la página web del simulador no se hacen responsables de los errores que pueden producirse al utilizar esta aplicación en esta última plataforma.

Como hemos mencionado en el párrafo anterior existe una pagina web (*www.isi.edu/nsnam/ns)* donde se puede descargar el simulador de forma gratuita, donde se incluye un manual de funcionamiento, preguntas y problemas

habituales en la instalación y funcionamiento de este software y las ayudas necesarias para el uso de la aplicación. A modo de resumen explicamos en este punto el procedimiento de instalación del NS-2 en una plataforma UNIX y detallaremos las particularidades necesarias para instalar las distintas versiones del protocolo MMDSR.

El primer paso consiste en **descargar una copia del programa** de instalación *ns-allinone-2.33.tar.gz*. En este archivo se agrupan todos los componentes antes mencionados, tanto obligatorios como opcionales. Si se quiere se puede ir descargando uno por uno, hasta instalarse los componentes necesarios que se crean necesarios para estudio que se pretenda hacer, pero en la primera vez que se quiera usar, recomendamos que se instale el conjunto de componentes *all-in-one* que se propone desde la página web.

En los siguientes puntos, detallaremos como adecuar el simulador ns-2 para la transmisión de *frames* de vídeo usando las distintas versiones del protocolo mencionadas en el capítulo 2.

5.1.2.1. Instalar RTP/RTCP

El primer paso para adecuar el simulador a la posibilidad de transmitir a través de esquemas multicamino es instalar la versión 2 del protocolo RTP/RTCP. Los pasos que debemos seguir son:

1- Descomprimir el *rtp_v2.tar* en la carpeta de instalación del simulador.

2- Añadir las siguientes lineas al fichero Makefile.in en la sección OBJ_CC:

rtp_v2/rtp_v2.o
rtp_v2/rtcp_v2.o

3- Añadir “-I./rtp_v2” en la sección “INCLUDES = “ del Makefile.in

4- Salvar los cambios y salir del Makefile.in

5- Copiar el fichero “session-rtp.cc” en la carpeta “ns-2.xx/common/”

6- Copiar los ficheros “rtp.cc” y “rtp.h” en la capeta “ns-2.xx/apps/”

7- Añadir las siguientes líneas al final del fichero “ns-2.xx/tcl/lib/ns-default.tcl”

```
Agent/RTP_v2 set seqno_ 0
Agent/RTP_v2 set interval_ 3.75ms
Agent/RTP_v2 set random_ 0
Agent/RTP_v2 set packetSize_ 210
Agent/RTP_v2 set maxpkts_ 0x10000000
Agent/RTP_v2 instproc done {} { }
Agent/RTP_v2 set rtcp_in_use_ 0
Agent/RTP_v2 set codification_ -1
Agent/RTP_v2 set lostcounter_ 0
Agent/RTP_v2 set losses_ 0
Agent/RTP_v2 set multipath_ 1
Agent/RTP_v2 set flow_id_ 0

Agent/RTCP_v2 set seqno_ 0
Agent/RTCP_v2 set interval_ 375ms
Agent/RTCP_v2 set random_ 0
Agent/RTCP_v2 set flow_id_ 0

Session/RTP set debug_ 0
```

8- Añadir el tipo “RTP_v2” en el fichero “/common/packet.h”. Para ello añadimos:

- En la estructura “enum_packet_t”, escribimos las siguientes líneas antes de la linea “PT_NTYPE”:

  PT_RTP_v2,
  PT_RTCP_v2,

- En la clase “p_info_public:p_info()” escribimos las siguientes líneas antes de “name_[PT_NTYPE]= undefined”:

name_[PT_RTP_v2]="rtp_v2";
name_[PT_RTCP_v2]="rtcp_v2";

9- Salvar cambios, salir y ejecutar “make”

5.1.2.2. Instalar MPEG2 Transmisor/Receptor

Para poder simular escenarios en los que se transmitan *frames* de vídeo usando ns-2 debemos incluir esta funcionalidad. Para ello seguimos los siguientes pasos:

1- Descomprimir *ns-mepg_v2.tar* en la carpeta de instalación del simulador.

2- Añadir las siguientes lineas al fichero “Makefile.in” en la sección OBJ_CC:

ns-mpeg_v2/mpegdata_v2.o
ns-mpeg_v2/mpegnulldata.o
ns-mpeg_v2/mpegdatafile.o
ns-mpeg_v2/mpeggen_vIPB.o
ns-mpeg_v2/mpeg2RX2_vIPB.o

3- Añadir “-I./ns-mpeg_v2” a la sección “INCLUDES=” del Makefile.in

4- Salvar y salir del Makefile.in

5- Copiar los archivos “ns-process.cc” y “ns-process.h” en la carpeta “./common/”

6- Ejecutar “make”

Tras adecuar el simulador para la transmisión de *frames* de vídeo pasamos a ver las particularidades de la instalación de las distintas versiones del protocolo MMDSR.

5.1.2.3. Instalar Protocolos MMDSR

Partimos de haber instalado el simulador en nuestro equipo usando el paquete *all-in-one* por lo que la versión del protocolo de enrutamiento que se encuentra instalada es la DSR. Además se han instalado el protocolo rtp_v2 y el ns-mpeg como se ha explicado en las dos secciones anteriores. En las secciones siguientes se va explicar los pasos necesarios para llegar acabo la instalación de las cuatro versiones del protocolo MMDSR que fueron desarrollados por Víctor Carrascal Frías en su tesis [CAR09]:

- Adaptativo (a-MMDSR): En este caso se espera a que se rompan los caminos para refrescarlo aunque existe un dinamismo en el período de refresco del algoritmo.
- Probabilidad de error de camino (ap-MMDSR): Esta versión no espera que se rompan los caminos para refrescarlos, lo hace evaluando un nuevo parámetro llamado Probabilidad de Error. En cuanto a la distribución de las *frames*, se mandan las I por el mejor camino y las P y B por el resto.
- Ventana Dinámica (d-MMDSR): En este caso las *frames* I también se manda por el mejor camino y las P y B por el resto. La particularidad de esta versión consiste en una mejora del Standard IEEE 802.11e [SA] y además introduce un refresco dinámico de las rutas.
- Teoría de Juegos (g-MMDSR): Establece un *game* con dos jugadores (fuentes y destinos), las *frames* B van por el peor camino y las I junto con las P van por uno de los de mejores. Con una probabilidad determinada se mandarán las *frames* I y P por el peor camino.

5.1.2.3.1. Adaptative

Para la instalación de esta versión del protocolo tenemos que realizar los siguientes pasos:

1- Descomprimir el fichero dsr.a.tar.gz en la carpeta de instalación del simulador.

2- Copiar el contenido de la capeta descargada en “ns-2.xx/dsr”

3- Modificar el Makefile.in original para que incluya la versión modificada del standard 802_11e

4- Copiar el fichero “ns-default.tcl” en la carpeta “ns-2.xx/tcl/lib/”

5- Ejecutar el “make”

5.2.1.3.2. Probabilidad de Error de Camino

Como se ha comentado anteriormente, se parte de un ns-2 básico, con un DSR instalado y preparado para transmitir frames de vídeo. Así, el proceso sería:

1- Descomprimir el fichero dsr.PEP.tar.gz en la carpeta de instalación del simulador.

2- Copiar el contenido de la capeta descargada en “ns-2.xx/dsr”

3- Modificar el Makefile.in original para que incluya la versión modificada del standard 802_11e

4- Copiar el fichero “ns-default.tcl” en la carpeta “ns-2.xx/tcl/lib/”

5- Ejecutar el “make”

5.2.1.3.3. Teoría de Juegos

En esta ocasión a parte del proceso habitual hay que incluir un versión modificada del protocolo rtp_v2:

1- Descomprimir el fichero dsr.GAME.tar.gz en la carpeta de instalación

del simulador.

2- Copiar el contenido de la capeta descargada en “ns-2.xx/dsr”

3- Descomprimir el fichero rtp_v2.GAME.tar.gz en la carpeta del simulador.

4- Modificar el fichero “Makefile.in” de manera que allí donde tengamos rtp_v2, sea rtp_v2_GAME.

5- Añadir las lineas para que incluya la versión modificada del standard 802_11e

6- Copiar el fichero “ns-default.tcl” en la carpeta “ns-2.xx/tcl/lib/”

7- Ejecutar el “make”

5.1.2.3.4. Ventana Dinámica

Como ya vimos en el capítulo dos, la ventana dinámica consiste en una nueva manera de regular el acceso al medio común. Por ello, para la instalación de esta versión necesitamos hacer una modificación sobre el estándard IEEE 802.11e que incluye las nuevas funcionalidades:

1- Descomprimir el fichero dsr.DVW.tar.gz en la carpeta de instalación del simulador.

2- Copiar el contenido de la capeta descargada en “ns-2.xx/dsr”

3- Copiar el fichero “mac-802_11e.dcw.cc” y “mac-802_11e.dcw.h” en la carpeta “/mac/802_11e”.

4- Modificar el Makefile.in original para que incluya la versión modificada del standard 802_11e modificado.

6- Copiar el fichero “ns-default.tcl” en la carpeta “ns-2.xx/tcl/lib/”

7- Ejecutar el “make”

5.1.3. Funcionamiento básico del NS-2

Tal y como se ha comentado en la introducción, NS es un simulador de red dirigido por acontecimientos. Pero en la versión 2 en lugar de encontrarse encapsulado en el lenguaje Tcl, se sustituye por el *Object Tool Command Language* de MIT (*Massachusetts Institute of Technology*), Otcl (una versión de Tcl orientada a objetos).

Para ejecutar el simulador, lo podemos hacer de dos maneras. Se ejecuta con la instrucción ns y añadiendo un argumento en la línea de comandos (en este caso se considera el nombre del fichero a ejecutar) o sin añadir ningún argumento:

- **Con un argumento**: Se abre el fichero (*script*) pasado como argumento, lo ejecuta y al acabar la simulación, retorna al intérprete de comandos. En el *script* deben aparecer todas las instrucciones para la generación de la simulación que se explican en apartados posteriores y tener la extensión *.tcl*. Un ejemplo de la estructura de la instrucción que se usa en este caso es el siguiente:

  bash$ ns simulación.tcl

  bash$

- **Sin argumentos**: En este caso el programa entra en modo interactivo, comportándose como cualquier intérprete de comandos tradicional, es decir, para cada línea introducida se analiza y si es correcta se ejecuta. El símbolo de petición de orden (*prompt*) es “%”. De esta manera se van introduciendo todas las instrucciones necesarias para la generación de la simulación. Para salir del modo interactivo debe ejecutarse la instrucción *exit*. En el siguiente ejemplo se crea una instancia del simulador y se sale de nuevo al *shell* de Linux:

  bash$ ns

  % set ns [new simulator]

% <demás instrucciones>

…

% exit

bash$

Debido a la complejidad de las topologías de las simulaciones usadas en este proyecto se ha optado por la utilización del modo “con argumentos” a la hora de realizar las simulaciones. Dentro de un *script* Tcl, también se pueden definir argumentos, que se le pasarán al NS-2 en el momento de ejecutar la simulación. Así, podemos repetir las simulaciones cambiando ciertos parámetros sin tener que cambiar el código *script*. Por ejemplo:

bash$ ns simulación.tcl ar1 arg2 arg3 [...]

Esta línea haría que NS ejecute el fichero *simulación.tcl*, con argumentos que le pasemos. Dentro de este fichero previamente habremos previsto el número de argumentos que puede aceptar y qué variable representarán cada uno de ellos.

Las salidas resultantes de la ejecución de las simulaciones se guardan en un archivo de trazas. Dependiendo de los resultados que se crea interesante obtener en las simulaciones existe la posibilidad de configurar el fichero de salida obteniendo la información deseada. En este proyecto haremos un estudio del fichero con las trazas de salida que saldrán por defecto. Para nuestro estudio, con la información que se desprende de ellas y algún que otro aspecto más, nos ha sido suficiente.

Estos ficheros de trazas, tanto por su tamaño como por su difícil formato visual, deben ser interpretados por terceros programas o herramientas de filtrado de datos. En nuestro caso, será con esta última, usando el lenguaje AWK que ya fue explicado en el capítulo 4.

En la elaboración de este proyecto, a parte del fichero de trazas se ha utilizado el log de salida del simulador. Éste es un fichero de texto donde se escribe información del tipo, identificador y tamaño de *frame*, porcentaje de *frames* enviadas de cada tipo, *delay jitter*... Estos datos y este fichero en concreto serán esenciales para las simulaciones que realizaremos con el objetivo de probar el buen funcionamiento de nuestra calculadora.

5.1.4. Generación de ficheros de simulación para redes Ad-hoc

Como se ha comentado anteriormente, el simulador ns-2 puede recibir los comandos usando un interfaz Tcl. Para ello se creo un script .tcl (*simulador.tcl)* que será pasado al simulador como argumento. En nuestro caso, este fichero tiene algunas particularidades, está adaptado a transmisiones de video y además acepta como protocolos la versiones MMDSR que se comentaron en el capitulo 2. La estructura de este tipo de fichero es la siguiente:

- Definición de variables globales
- Definición de ficheros de trazas
- Definición de la configuración de los nodos
- Creación de los nodos
- Definición de la posición inicial de los nodos y generación de sus movimientos
- Generación de agentes de tráfico de los nodos
- Definición de la planificación de sucesos
- Definición proceso de final de simulación
- Empezar la simulación

5.1.4.1. Definición de variables globales

El primer paso en la simulación es la definición de variables globales. Se debe adquirir como mínimo una instancia de la clase Simulator (Simulador). Para la definición de variables, se utiliza el comando **set** seguido del nombre que quiera darse y el valor que obtenga dicha variable. Las instancias de objetos en clases son creadas y distribuidas en Otcl utilizando el método **new**. Por ejemplo, una instancia del objeto Simulador se crea mediante el siguiente comando:

```
set ns_ [new Simulator]
```

A partir de entonces cualquier procedimiento de este objeto será llamado empezando con **$ns_**.

Además, en este primer paso se crean otras variables globales, como pueden ser variables creadas por nosotros para ser utilizadas por el programa, o variables globales existentes en las librerías que queramos redefinir para configurar la simulación. En el caso de las redes Ad-Hoc, por ejemplo incluiríamos en este capítulo las variables de configuración de nodos, la reconfiguración de los parámetros que definen el protocolo en la capa MAC, así como las variables donde guardaremos los valores que pasemos como argumentos en la línea de comando:

```
set opt(chan)   Channel/WirelessChannel ;   # tipo de canal
set opt(prop)   Propagation/TwoRayGround ;      # modelo de
                                                propagación
set opt(netif)  Phy/WirelessPhy ;           # tipo de capa física
set opt(mac)    Mac/802_11 ;                # tipo de capa MAC
set opt(ifq)    CMUPriQueue
set opt(ll)     LL ;                    # tipo de capa de enlace
set opt(ant)    Antenna/OmniAntenna ;           # tipo de antena
set opt(ifqlen)     -1                  # capacidad de la cola
set chan_1_ [new $opt(chan)]
```

```
#parámetros que pasemos al simulador
set NumofSources              1 ;          # número de fuentes
set opt(CBRpsize)        1500 ;            # tamaño de los
                                           paquetes CBR
set opt(CBRrate)         1000000 ;    # flujo de los paquetes CBR
set opt(num_nodes)       50 ;              # número de nodos
set opt(NumOfMN)              $num_nodes
set opt(RoutingProtocol)      DSR ;        # protocolo de
                                                encaminamiento

set opt(X).              500 ;             # X & Y dimensiones de
                                                la topolografía
set opt(Y)               500 ;
set opt(MNcoverage)      120
set opt(speed)           2 ;               #velocidad máxima
set opt(start)                0
set opt(stop)                 100 ;        #final de la simulación
```

Existen otras variables globales que nos vamos a escribir aquí para no sobrecargar la explicación.

También se debe adquirir una instancia de la clase *Topography* (Topografía), que nos defina el espacio de simulación. El procedimiento es similar a la instancia del simulador.

```
set topo [new Topography]
$topo load_flatgrid $opt(X) $opt(Y)
```

Igual que antes cualquier procedimiento de este objeto será llamado empezando con **$topo**, como es el caso de la segunda instrucción, que nos define el espacio de simulación en metros (ancho alto) usando los parámetros definidos un poco más alto.

5.1.4.2. Definición del fichero de trazas

El siguiente paso sería la definición de los archivos donde se guardan las trazas para ser evaluadas. Se puede ejecutar la simulación sin tener que definir estos ficheros. La definición de ficheros de salida se realiza siguiendo la siguiente estructura.

```
set trace [open "tr_multipath.tr" w]
$ns trace-all $trace
$ns use-newtrace
```

Estos comandos generaran el fichero: *tr_multipath.tr*. En él, quedarán anotadas todas las trazas resultantes de la simulación.

5.1.4.3. Definición de la configuración de los nodos

Para la realización de este procedimiento se debe crear primero un Director General de Operaciones (*god*) indicando el número de nodos que participarán en la simulación.

```
set god [create-god $opt(NumOfMN)]
```

donde NumOfMN representa el número de nodos como ya lo vimos antes.

Ahora es el momento de definir la configuración de los nodos. Estos datos se guardan utilizando el proceso **node-config**. Y se realiza de la siguiente manera:

```
$ns node-config \
   -adhocRouting $opt(RoutingProtocol) \
   -llType $opt(ll) \
   -macType $opt(mac) \
   -ifqType $opt(ifq) \
   -ifqLen $opt(ifqlen) \
   -antType $opt(ant) \
   -propType $opt(prop) \
   -phyType $opt(netif) \
```

-channel $chan_1_ \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace OFF \
-movementTrace OFF

Como podemos observar, en este punto se configuran los siguientes aspectos de los nodos:

- **Protocolo de encaminamiento** llamando a $opt(RoutingProtocol) definido antes en la parte de definición de las variables globales
- **Capa de enlace** llamando a $opt(ll)
- **Capa Mac** llamando a $opt(mac)
- **Tipo de cola** llamando a $opt(ifq)
- **Capacidad de la cola** llamando a $opt(ifqlen)
- **Tipo de antena** llamando a $opt(ant)
- **Tipo de propagación** llamando a $opt(prop)
- **Capa física** llamando a $opt(netif)
- **Tipo de canal** llamando a $chan_1_
- **Topología de la red** llamando a $topo
- **Trazas** : activación o desactivación de información acerca de la capa Mac, agentes, encaminamiento y movimientos en los ficheros de salida

Así vemos que al definición de las variables globales del principio es muy importante y sobre todo muy práctica: si queremos cambiar un parámetro entre dos simulaciones, solo tenemos que quedar al principio del fichero y cambiar directamente la variable que queremos. El cambio de la configuración de los nodos se hará después automáticamente.

Tenemos que decir a pesar de todo que no estamos obligados a definir todas estas variables globales al principio: si son fijas, les podemos pasar directamente en el fichero.

5.1.4.4. Creación de los nodos

Una vez definidos y configurados los nodos podemos pasar a su creación. Todos los nodos que generemos serán del tipo según se haya configurado en el apartado anterior. La instrucción para la creación de nodos es la siguiente:

```
set node_(0) [$ns_ node]
```

Que sigue el mismo modelo de definición de variables que en los casos anteriores. Mediante esta instrucción se creará un nodo al que hemos llamado **node_(0)**, con la configuración de la clase **node**.

Según sea la complejidad de la red o si queremos generar una simulación con un número de nodos variable según decidamos en el momento de hacer la simulación es útil utilizar la instrucción **for** para la creación de los nodos. La instrucción final usando este método quedaría de la siguiente manera:

```
for {set i 0} {$i < $opt(NumOfMN)} {incr i} {
        set node_($i) [$ns node]
}
```

Donde **opt(NumOfMN)** nos indica el número de nodos a generar, y **i** es un contador auxiliar para la generación de estos. **opt(NumOfMN)** forma parte de las variables globales definidas al principio del fichero.

5.1.4.5. Posición inicial de los nodos y generación de sus movimientos

En este apartado se define la posición inicial de los nodos, y si además queremos crear una simulación con movilidad, este sería también el momento de definir su movimiento. Este movimiento se puede definir manualmente, mediante el generador de NS (mediante la instrucción **setdest**) o mediante algún

generador de patrones de movimiento que seguirán los nodos durante las simulaciones. Dicho generador de patrones de movimiento debe ser compatible con instrucciones NS-2 (como es nuestro caso, *Bonnmotion*).

La definición de la posición inicial de los nodos debe hacerse siguiendo la siguiente estructura:

$node_(0) set X_ 5.0
$node_(0) set Y_ 2.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 400.0
$node_(1) set Y_ 2.0
$node_(1) set Z_0.0

Vemos que podemos asignar una posición de tres dimensiones, aunque normalmente la coordenada Z no suele usarse para este tipo de simulaciones.

La definición del movimiento de los nodos se expresa con una serie de comandos como los que siguen, dependiendo del movimiento que quiera darse a estos:

$ns_ at 50.000000000000 "$node_(1) setdest 369.463256473829 170.563728293847 2.000000000000"
$ns_ at 51.000000000000 "$node_(0) setdest 200.361782038371 100.098762524156 20.000000000000"
$ns_ at 52.000000000000 "$node_(1) setdest 350.765545434533 95.342543765778 3.000000000000"
$ns_ at 53.000000000000 "$node_(0) setdest 187.654654654656 50.765785544353 25.000000000000"
...

Donde cada una de estas instrucciones responden al siguiente esquema:

$ns_ at $time "$node setdest <x2> <y2> <speed>"

Es muy común utilizar generadores de patrones de movimiento externos. En esto casos, hay que estar alerta de que no se produzcan conflictos en la deficinición de las variables. Estos generadores definen la posición inicial de los nodos y su movimiento en el formato que hemos visto anteriormente y producen un fichero que debe ser incluido en el .tcl del simulador.

```
set opt(sc) "/home/mtoscano/ns-allinone-2.27/ns-2.27/scens/scenarioPEP.scen"
source $opt(sc)
```

5.1.4.6. Generación de agentes de tráfico de los nodos

Los agentes representan los puntos donde se generan o se reciben los paquetes que se envían durante la transmisión y son usados para la implementación de los protocolos. Los agentes de tráfico se tienen que definir a partir de la capa de transporte ya que hemos definido previamente la capa de enlace y la capa de red.

En este capitulo, tenemos que diferenciar los dos tipos de tráfico que usamos: tráfico vídeo o tráfico CBR. El primero es el que usaremos para transmitir las frames de vídeo y el CBR lo usaremos para congestionar la red y crear escenarios donde se produzcan pérdidas.

5.1.4.6.1. Tráfico vídeo

En el siguente esquema, vemos que la capa de aplicación recibe dos tipos de ficheros: un fichero de vídeo y un fichero de trazas. Este fichero de trazas sirve para simular el envío real del fichero de vídeo: el fichero de trazas dice cuantas tramas del fichero de vídeo la capa de aplicación debe enviar por segundo para simular la transmisión real del fichero vídeo.

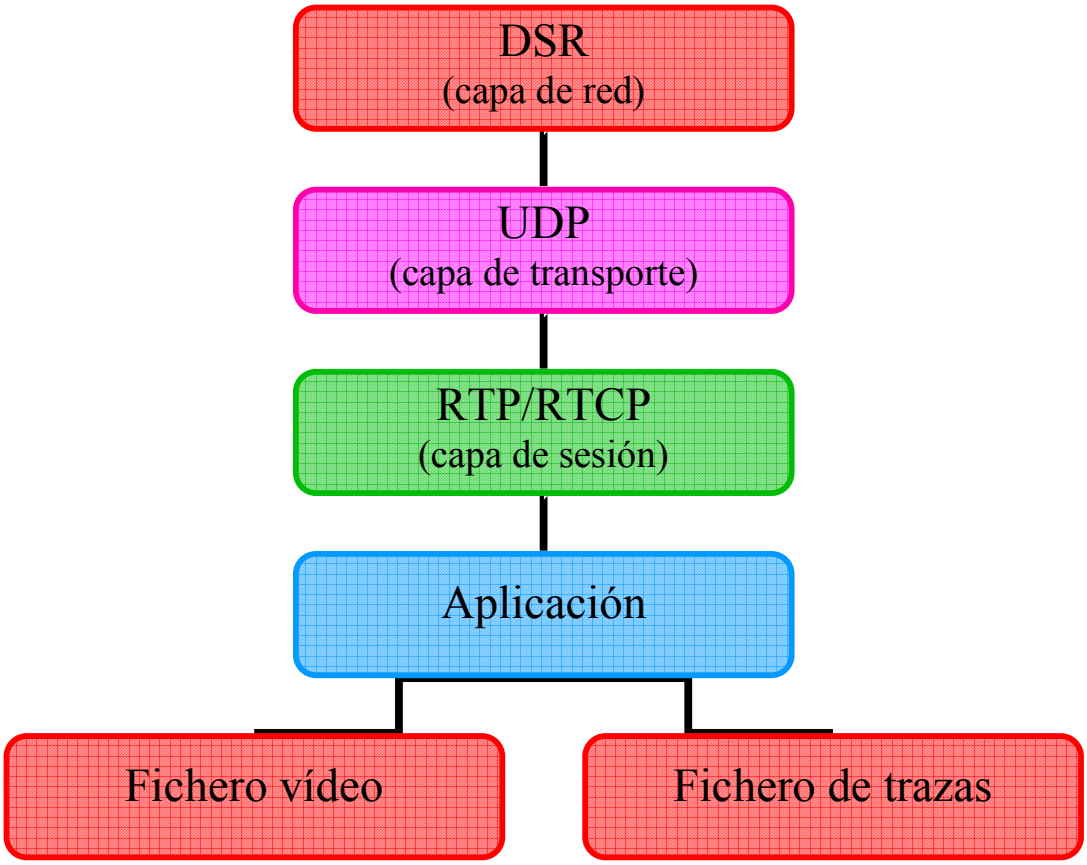

Figura 5.1. Arquitectura de protocolos utilizado para tráfico de vídeo

Este esquema es un esquema muy general: en realidad, aquí la capa de transporte esta incorporada en la capa de sesión. Este proviene de la naturaleza del simulador programado: todo lo concerniente la capa UDP ya esta incorporado en la capa RTP/RTCP.

Así, sólo tenemos que especificar la capa de sesión (RTP/RTCP) y la capa de aplicación durante la fase de generación de los agentes de tráfico.

Especificación de la capa de sesión

La especificación de la capa de sesión RTP/RTCP se hace mediante los comandos siguientes:

set rtp_($i) [new Agent/RTP_v2]
$ns attach-agent $node_($i) $rtp_($i)
$rtp_($i) set class_ 2
$rtp_($i) set seqno_ 0
$rtp_($i) set packetSize_ 1500
$rtp_($i) set multipath_ 3
$rtp_($i) set flow_id_ $i

Estos comandos crean un puntero **rtp** que apunta a un objeto de la clase **Agent/RTP_v2**. Este agente se asigna al nodo *i* emisor (nodos fuentes). Después, hay definiciones de propiedades de este agente.

Se hace la misma cosa para crear punteros **rtp** y **rtcp** para los nodos destinos (n).

set rtcp_($n) [new Agent/RTCP_v2]
$ns attach-agent $node_($n) $rtcp_($n)
$rtcp_($n) set class_ 3
$rtcp_($n) set interval_ 5000ms
$rtcp_($n) set seqno_ 0
$rtcp_($n) set flow_id_ $i
set rtp_($n) [new Agent/RTP_v2]
$ns attach-agent $node_($n) $rtp_($n)
$rtp_($n) set class_ 2
$rtp_($n) set seqno_ 0
$rtp_($n) set packetSize_ 1500

Después, definimos la conexión entre el nodo emisor y el nodo receptor:

$ns connect $rtp_($i) $rtcp_($n)

Creamos también un puntero **self** que apunta a una Session/**RTP**. Esta sesión se asigna al nodo emisor i. Asignamos después los agentes precedentes a esta sesión de la manera siguiente:

set self_($i) [new Session/RTP]

$rtcp_($n) session $self_($i)

$rtp_($i) session $self_($i)

$rtp_($n) session $self_($i)

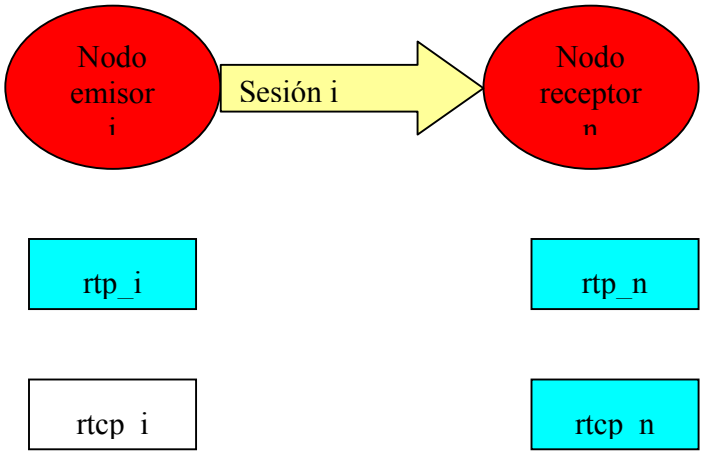

Figura 5.2. Sesión de tráfico de vídeo entre emisor y receptor

Vemos que asignamos **rtp_i**, **rtp_n** y **rtcp_n** a esta sesión entre el nodo i y el nodo n.

- rtp_i para la transmisión de los datos
- rtp_n para la recepción de los datos
- rtcp_n para el control de la recepción de los datos.

No tenemos que asignar **rtcp_i**: serviría para el control de los datos durante entre un nodo fuente n y un nodo receptor i.

Especificación de la capa de aplicación

Primero definimos el estilo de lo que la capa de aplicación recibe: es decir el fichero de vídeo y el fichero de trazas sirviendo para enviar de manera casi real el fichero de vídeo.

set trace_file_($i) [new MpegDataFile_v2]
$trace_file_($i) metafile
/home/mtoscano/ns-allione-2.27ns-2.27/blade2.mdipbf4444
$trace_file_($i) datafile /home/mtoscano/ns-allione-2.27ns-2.27/blade.m2v

Creamos luego un puntero **MPEG2** que asignamos al agente **rtp** del nodo emisor i que apunta a un objeto de la clase **Application/Traffic/Mpeg2Gen2.**

set MPEG2_($i) [new Application/Traffic/Mpeg2Gen2]
$MPEG2_($i) attach-mpegdatasrc $trace_file_($i)
$MPEG2_($i) attach-agent $rtp_($i)
$MPEG2_($i) set codification_ 4
$rtp_($i) set codification_ 4

Hacemos lo mismo para el nodo receptor n.

set sinkmpeg_($n) [new Application/Mpeg2RX2]
$sinkmpeg_($n) attach-agent $rtcp_($n)
$sinkmpeg_($n) attach-agent $rtp_($n)
$sinkmpeg_($n) dumpfile "($i)_multipath_vídeo.m2v"

La ultima línea indica en qué fichero de salida se pondrá el fichero de vídeo recibido en el nodo receptor.

5.1.4.6.2. Tráfico CBR

La arquitectura del tráfico CBR es la siguiente:

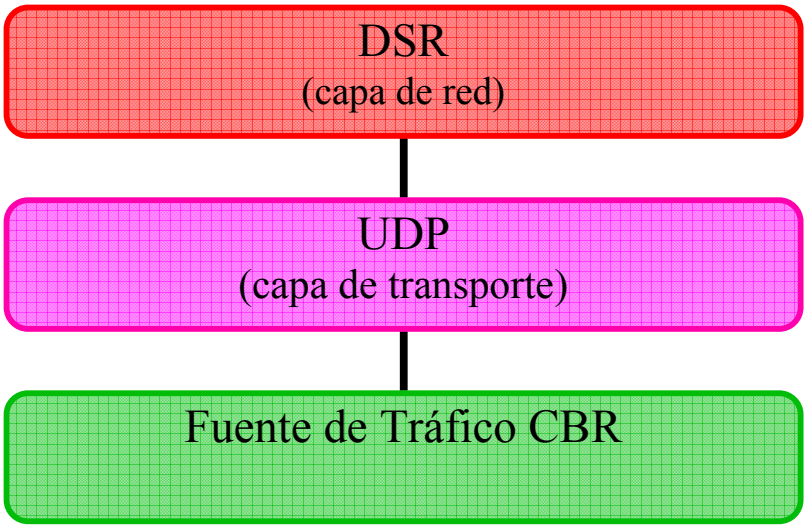

Figura 5.3. Arquitectura de protocolos para el tráfico CBR

Vemos con este esquema que solo se tiene que especificar la capa UDP, lo que hacemos de la manera siguiente:

```
set udp_($i) [new Agent/UDP]
$ns attach-agent $node_($i) $udp_($i)
set null_($n) [new Agent/LossMonitor]
$ns attach-agent $node_($n) $null_($n)
set cbr_($i) [new Application/Traffic/CBR]
$cbr_($i) set packetSize_ $opt(CBRpsize)
$cbr_($i) set interval_  $opt(CBRperiod)
$cbr_($i) set random_ 1
$cbr_($i) set CBRrate $opt(CBRrate)
$cbr_($i) attach-agent $udp_($i)
$ns connect $udp_($i) $null_($n)
$udp_($i) set fid_ 4
```

En este caso se crea un agente UDP como emisor y un agente *Null* como receptor, se conectan entre si y se les asigna un tráfico CBR.

5.1.4.7. Definición de la planificación de sucesos

Hasta este punto se han configurado todos los elementos que participarán en esta simulación. Ahora se necesita definir una planificación de sucesos, es decir especificar en cada momento los eventos que van a suceder.

Esta planificación empezara en el momento que empiece la simulación. La estructura para generar eventos es la siguiente:

$ns_ at <time> <event>

Mediante esta instrucción decidimos el inicio y fin de los tráficos entre nodos que hayamos definido en el punto anterior.

Un ejemplo de planificación de eventos es el que sigue. En este caso hemos considerado que en la simulación existen dos conexiones: una con tráfico CBR entre 1 y 100 segundos y a partir de 4 segundos tráfico vídeo:

```
$ns at 1 "$cbr_($i) start"
$ns at 100 "$cbr_($i) stop"
$ns at 4.0 "$MPEG2_($i) start"
$ns at 4.0 "$rtcp_($n) start"
```

Además de la planificación de eventos de la simulación en este apartado tenemos que ejecutar el proceso finish{} que se comenta a continuación:

```
$ns at $opt(stop)    {finish}
```

5.1.4.8. Definición del proceso de final de simulación

En toda simulación tienen que haber un procedimiento de finalización de la simulación. Este proceso nos cierra los ficheros que hemos ido abriendo durante

la simulación y nos ejecuta, si lo creemos necesario, alguna aplicación que queramos que se ejecute a continuación.

```
proc finish {} {
global ns node_ null_ nn2 opt trace namtrace namtrace_s
sinkmpeg
puts "Finishing ns at time [$ns now]."
$ns flush-trace
close $trace
exit 0
}
```

El comando **proc** declara un procedimiento, en este caso llamado **finish**, sin argumentos. En la definición de variables internas, la palabra **global** hace referencia a variables declaradas fuera del procedimiento.

La instrucción **$ns flush-trace** nos descarga las trazas en sus respectivos archivos. El comando Tcl **close** cierra los ficheros de trazas definidos anteriormente.

Y finalmente la instrucción **exit 0** acaba el procedimiento y nos devuelve un 0 como estado al sistema. El cero significa que la aplicación ha terminado con éxito. Al final de la planificación de sucesos se debe hacer una llamada a esta función para que se ejecuten todas las instrucciones de finalización, como se ha visto en el apartado anterior.

5.1.4.9. Empezar la simulación

Como último paso debemos iniciar la simulación. Es muy importante que esta instrucción este al final del *script*, ya que antes de ejecutarse deben estar todos los parámetros de la simulación. La simulación se inicializa usando el comando **run** siguiendo la siguiente estructura:

$ns run

Como resultado de la simulación, si hemos ido siguiendo todos estos pasos correctamente, obtendremos dos ficheros de trazas. Uno para ser interpretada con una aplicación de tratamiento de ficheros de trazas, como puede ser un *script awk*, y otro para ser interpretada por la herramienta NAM que en nuestro proyecto no será utilizada.

Ficheros de trazas

Mediante estos ficheros podemos obtener todos los parámetros para comprobar el *overhead*, el *throughput* y las pérdidas tanto perdidas totales como perdidas en tramas I, P o B de la simulación realizada. Estos ficheros nos han sido muy útiles para la obtención de las conclusiones de este proyecto final de carrera y por eso se hace un análisis en el capitulo de Simulaciones.

5.2 Bonnmotion

En elemento fundamental para el estudio de las redes Ad-hoc es el escenario. Este escenario define elementos como dimensiones, número de nodos, patrón de movimiento, velocidad de nodos, posiciones iniciales... Para su creación hemos utilizado en este proyecto la herramienta Bonnmotion [BNM05].

Bonnmotion es un software en Java que crea y analiza marcos de movilidad. Se desarrolla dentro del grupo de Sistemas de Comunicaciones en el Instituto de la Informática IV de la Universidad de Bonn, Alemania, donde sirve como una herramienta para la investigación acerca de características de redes Ad Hoc móviles.

Su funcionamiento es muy sencillo, se ejecuta en la línea de comandos siguiendo el siguiente esquema:

```
bm –f <fichero_escenario> <escenario> <parámetros>
```

Al aplicar este comando, obtenemos como salida 2 ficheros, uno con la extensión *.params* donde está el resumen de todos los parámetros de entrada que caracterizan nuestro escenario, y otro, comprimido en formato *.movements.gz*, con extensión *.movements*, que contiene todos los valores de movimiento para cada nodo durante todo el tiempo de simulación.

Para nuestro estudio, se ha utilizado como patrón de movimiento *RandomWaypoint* por ser el más general y utilizado.

5.2.1. Modelo *RandomWaypoint*

Este es el modelo utilizado tradicionalmente y más usual. Se trata del más aleatorio, aunque esto no significa que sea el que tenga un comportamiento más semejante a la realidad.

Este modelo incluye pausas entre cambio de direcciones y de velocidad de los nodos. Un nodo permanece en un posición hasta que le expira el tiempo asignado y calcula una velocidad distribuida uniformemente entre la velocidad mínima y máxima seleccionadas en la simulación. Entonces viaja hacia la nueva posición a la velocidad elegida. En ocasiones este modelo puede tener variaciones, como anular los tiempos de espera entre cambios. Es aconsejable que, para tener una buena inicialización en el nivel de aleatoriedad, dejar transcurrir la simulación durante unos 1000 segundos.

Ejemplos de este patrón de movimiento serían una conferencia, un museo donde la gente pasea sin rumbo fijo, etc...

5.2.2. Conceptos Generales

Existen dos posibilidades para definir los parámetros que utilizará el generador de escenarios *Bonnmotion*. La primera consiste en introducir los parámetros directamente en la línea de comandos, y la segunda es introduciendo en la línea de comandos un fichero que contenga los parámetros que caractericen los escenarios. Este último fichero debe tener el mismo formato que el fichero de salida *.params*. De manera que una vez realizado un escenario, podremos modificar este fichero y utilizarlo para hacer pequeñas variaciones en los escenarios ya creados.

Hay una serie de parámetros que por su importancia a la hora de definir un escenario, se usan de la misma manera en la caracterización de todos ellos. Estos se muestran en la **Error! Reference source not found.**:

Tabla 5.1 Comandos Generales Bonmotion

-n	Número de nodos en el escenario
-d	Duración del escenario en segundos
-i	Segundos de guarda antes de empezar a generar el escenario
-x	Ancho del escenario en metros
-y	Alto del escenario en metros
-R	Semilla aleatoria

Cabe destacar que el parámetro **–i** es importante que sea elevado, debido a que en ciertos escenario los nodos empiezan su movimiento a partir de las coordenadas (0,0). Para crear un escenario lo más parecido a la realidad, hay que dejar un buen tiempo de guarda antes de empezar a simular.

Otro parámetro a destacar es la semilla aleatoria (*Random Seed*). Este valor es el que se encarga de que dos escenarios, aunque tengan los mismos parámetros, sean diferentes, siempre y cuando la semilla sea también diferente. Si lo que quisiéramos conseguir fueran dos escenarios completamente iguales, deberíamos utilizar la misma semilla (aunque este no sea un procedimiento interesante para nuestro estudio, en el que queremos conservar la aleatoriedad del movimiento de los nodos dentro de las pautas marcadas por los patrones de movimiento escogidos).

A modo de ejemplo podemos ver la generación de un escenario basado en el modelo *Random Waypoint*:

```
bm –f scenario1 RandomWaypoint –n 100 –d 900 –i 3600 –x 20 –y 20
```

Este comando nos generaría un escenario de dimensiones 20 x 20 metros, con 100 nodos y una duración de 900 segundos. Antes de empezar a generar el escenario dejaría pasar 3600 segundos para que los nodos se movieran por todo el espacio de simulación.

De la ejecución del comando obtendríamos 2 ficheros, uno de parámetros y otro de movimientos, tal y como se ha comentado anteriormente. El siguiente paso sería adaptar estos dos ficheros a un lenguaje interpretable por el NS-2, ya que esta aplicación la que con los datos de cada escenario hará la simulación de una red Ad Hoc.

Bonnmotion trae consigo pequeñas aplicaciones para poder realizar la conversión de formato de sus ficheros de salida al formato de ficheros compatibles con otras aplicaciones o simuladores. La aplicación *NSFile*

transforma la salida de *Bonnmotion* obteniendo 2 ficheros nuevos. Uno con extensión *.ns_params* y otro *.ns_movements* para usarlos en el simulador NS-2.

También trae consigo *GlomoFile* que permite la conversión de los ficheros obtenidos, a ficheros interpretables por el simulador *Glomosim*. Existe también la opción de convertir los ficheros a formatos *Qualnet* y *Excel*.

Una vez obtenidos los ficheros de salida del *Bonnmotion* tendremos que convertirlos a formato NS-2 del siguiente modo:

```
bm NSFile –f scenario1
```

Y obtendremos los archivos:

- *scenario1.ns_params*
- *scenario1.ns_movements*

A continuación se mostrarán los comandos necesarios para obtener los escenarios con el patrón de moviemto RandomWayPoint. No obstante, la herramienta Bonnmotion esta preparada para simular otros modelos como Manhattan Grid, RPGM Reference Point Mobility Model o Gauss-Markov.

5.2.3. Comandos para RandomWaypoint

Para generar un escenario basado en el modelo *Random Waypoint* se debe seguir la siguiente estructura de comando:

```
bm –f RANDOM RandomWaypoint –n 100 –d 300 –i 3600 –x 500 –y 500 <parámetros opcionales del escenario>
```

Donde los parámetros opcionales de los escenarios pueden ser los comentados en la siguiente tabla:

Tabla 5.2 Comandos Random Waypoint

-a	Indica los puntos de atracción del escenario, separando por comas tantos puntos de atracción como hayan y sus propiedades (para cada punto se puede definir posición X, posición Y, intensidad y desviación estándar.
-c	Para activar el estilo circulo. Hace que el escenario sea un circulo intrínseco en el área de simulación definido por –x e –y.
-h	Velocidad máxima d los nodos en metros por segundo
-l	Velocidad mínima de los nodos en metros por segundo
-p	Tiempo máximo de parada de los nodos en segundos
-o	Restricciones del movimiento de los nodos: • **1**: movimiento permitido solo por el eje x • **2**: movimiento permitido o bien por el eje x o bien por el eje y • **3**: movimiento permitido tanto por el eje x como por el eje y

Capítulo 6: Simulaciones

Como parte de este proyecto fin de carrera se han llevado a cabo una serie de simulaciones cuyo objetivo es mostrar las prestaciones de la calculadora de *Peak Signal-to Noise Ratio* (PSNR) aquí desarrollada y al mismo tiempo, emplear hasta nueva herramienta para hacer un análisis de las distintas versiones del protocolo *Multimeda Multipath Dynamic Source Routing* (MMDSR), desarrollado en la tesis de Víctor Carrascal Frías [CAR09].

Los resultados de estas simulaciones van a ser presentados a lo largo de este capítulo enfatizando las consecuencias que tienen los distintos valores de PSNR sobre las transmisiones de vídeo a través de redes MANET.

6.1. Organización de las simulaciones

A lo largo del capítulo se va hacer un estudio de las distintas versiones del protocolo MMDSR, sometiendo a los nodos transmisores de vídeo en distintas situaciones de congestión que provocarán pérdidas, que implicarán a su vez la aparición de distorsión y con ella valores más bajos de PSNR.

Las simulaciones se encuentran divididas en tres bloques que coinciden con el número de escenarios que se han desarrollado donde se estudian tres versiones de protocolos de encaminamiento desarrolladas en la tesis de Víctor Carrascal Frías [CAR09].

- **Bloque 1**: El objetivo es evaluar las mejores introducidas por la versiones *adaptative Path Error Probability - Multipath Multimedia Dynamic Source Routing* (ap-MMDSR) y *Dynamic Contention Window - Multipath Multimedia Dynamic Source Routing* (d-MMDSR) con respecto a *Dynamic Source Routing* (DSR). Para ello se ha diseñado un escenario con nodos en movimiento, tráfico interferente, un nodo transmisor y dimensiones reducidas. Los detalles se verán más adelante.
- **Bloque 2**: En esta ocasión también se evaluarán las mismas prestaciones pero se someterá al nodo transmisor a unas características de red completamente distintas. En este bloque nos basaremos en un escenario grande de 400x400m con nodos en movimiento que permitirán la posibilidad de disponer de una topología de red altamente dinámica en que los caminos se crean y se sustituyen por otros.
- **Bloque 3**: Finalmente, en este bloque se analizarán las prestaciones de la versión *Game – Multipath Multimedia Dynamic Source Routing* (g-MMDSR) con respecto ap-MMDSR y DSR. Por la naturaleza particular de esta versión del protocolo MMDSR, se ha diseñado un escenario específico con dos nodos transmitiendo sobre los que se evalúa el algoritmo basado en la Teoría de Juegos.

Para cada bloque se va seguir la siguiente metodología. Se explicarán los objetivos perseguidos con las simulaciones realizadas, se describirá el escenario en cuestión sobre el que se han hecho las simulaciones, los parámetros concretos utilizados y posteriormente se mostrarán las gráficas con las conclusiones que se han alcanzado.

6.2. Aspectos previos a las simulaciones

Antes de pasar analizar las simulaciones en esta sección se van a detallar algunos aspectos que serán utilizados posteriormente.

En primer lugar, la métrica que se va a utilizar es la PSNR. Debido a que la herramienta que se ha desarrollado en este proyecto es una calculadora para dicho parámetro parece lógico que se haga un análisis de los valores de éste. Para consultar los valores de otras métricas de *Quality of Service* (QoS) como *Throughput, Overhead,* retardo, *jitter* o porcentaje de paquetes pérdidos sobre redes MANET se pueden consultar los proyectos “Evaluación de prestaciones del protocolo MMDSR (Multipath Multimedia Dynamic Source Rounting) sobre redes móviles Ad-hoc” [DOE08] y “Evaluación de prestaciones de varios protocolos de encaminamiento en diferentes escenarios de redes Ad-hoc mediante simulador ns-2” [LEFE08].

La PSNR, cuya expresión matemática se presentó en el capitulo 4, nos va a permitir conocer si los protocolos de enrutamiento están funcionando correctamente. Podremos comparar un cuadro (*frame*) original con una recibida y ver así cuánta distorsión ha sufrido el vídeo recibido.

En segundo lugar, es importante hablar de los aspectos que supuestamente deben producir pérdidas a la hora de simular. Por un lado, el tráfico interferente inyectado *Constant Bit Rate* (CBR) que es un componente de la capa de aplicaciones que genera un tráfico constante durante las simulaciones para

situar a la red en saturación. Sus efectos serán analizados posteriormente. Por otro, nos encontramos con el movimiento de los nodos que producirá la rotura de unos caminos y la creación de otros, con lo que las pérdidas de paquetes serán una realidad presente en las simulaciones obtenidas.

Por último, relacionando con los aspectos detallados en los párrafos anteriores es importante destacar un aspecto del envío de cuadros (*frames*) antes de analizar las simulaciones. Como ya se comentó en el capítulo 1, el archivo de vídeo se encuentra codificado usando MPEG-2 [MPG] antes de ser transmitido. Con esta codificación se empleaban tres tipos de cuadros I, P o B. Las primeras son esenciales para la decodificación del vídeo y donde se encuentra la mayor parte de la información y las dos últimas son prescindibles. A la hora de enviar las *frames*, estas se encapsulan en paquetes que son transmitidos a través de la red. Debido al tamaño de las *frames* I, estas pueden ir distribuidas en más de un paquete lo que aumenta la probabilidad de que se pierdan. Si por ejemplo una *frame* I se encuentra dividida en tres paquetes y uno se pierde, puede ser que la escena sea decodificada para tendrá más distorsión y la PSNR será más baja. Si lo que se perdieran fueran paquetes con *frames* P o B, afectaría a la distorsión y por tanto a la PSNR pero siempre se podría decodificar la escena.

6.3. Bloque 1: Escenario pequeño

En primer lugar se va a estudiar el escenario más pequeño con el objetivo de probar que las versiones ap-MMDSR y d-MMDSR mejoran al protocolo DSR en términos de la PSNR.

Se ha utilizado un tráfico interferente CBR de 5Mb, bastante alto, para provocar pérdidas. Este escenario es pequeño 200x200m, sus dimensiones pueden asemejarse a las de un parque, un campo de fútbol o una biblioteca. Con las simulaciones que hemos realizado se pretende hacer un paralelismo con la realidad y para ello se han escogido escenarios pensados al respecto. En esta

ocasión vamos a poder reflejar como se comportaría la red MANET en ambientes como los mencionados anteriormente.

Pasamos a definir las características concretas del escenario pequeño.

6.3.1. Características del escenario pequeño

En la siguiente tabla (*Tabla 6.1*), se ven las características del siguiente escenario:

Tabla 6.1. Características del Escenario Pequeño

Area	200x200
Número de Nodos	20
Vel. Máx. nodos	2 m/s
Cobertura	75 m
Patrón de movimiento	Random Waypoint
Especificación MAC	IEEE 802.11e
Ancho de Banda	11 Mbps
Tiempo de Simulación	100s
Codificación de vídeo	MPEG-2 VBR (Blade Runner)
Tasa del vídeo	150 kbps
Protocolo de transporte	RTP/RTCP/UDP
Tamaño máximo de paquete	1500 bytes
Tamaño de las colas	80 paquetes
Fuentes de vídeo	1
Ruido de canal	-92 dBm
Múltiples rutas	De 1 a 3

De los valores anteriormente detallados cabe destacar la velocidad de los nodos. Se ha utilizado un valor de 2 m/s como velocidad máxima, debido que es lo que más se asemeja al movimiento de un ser humano. También es importante el ancho de banda físico que se dispone, es de 11 Mbps y podemos constatar que

el tráfico CBR es importante. El número de rutas va de 1 a 3, para que los protocolos multicamino puedan funcionar con beneficios notables se necesitan 3 rutas disponibles. Por último, destacar que la especificación MAC que se ha utilizado es la IEEE 802.11e [SA] que permite distintas categoría de acceso y algunas opciones de QoS.

6.3.2. Valores de la PSNR. Escenario Pequeño

En la *Tabla 6.2* se pueden ver los valores de PSNR obtenidos para el escenario pequeño utilizando como protocolo de enrutamiento el DSR:

Tabla 6.2 Valores de PSNR para DSR en el escenario pequeño

	MEDIDAS PSNR (db) para DSR					
Tiempo (segundos)	**Peq1**	**Peq2**	**Peq3**	**Peq4**	**Peq5**	**Original**
10	20,62527262	23,5151546	21,242833	22,7630891	23,5151546	17,9354771
20	20,57103838	19,98333219	20,5212005	20,6022852	19,9833322	20,9505522
30	18,05802484	18,05802484	18,0580248	18,0017911	18,0580248	21,8244054
40	22,94907766	23,26726441	23,2672644	23,1354875	23,2672644	23,0150269
50	19,92443992	19,87294734	19,9345125	19,4678044	19,8729473	24,1340738
60	21,49470302	23,80856502	21,5695663	21,3835532	23,808565	22,3182862
70	20,97199152	21,39743144	21,8015724	21,1964814	21,9713079	21,5270312
80	16,97836062	19,33349226	19,1007519	20,0132715	20,3334923	20,1770107
90	20,55395332	20,50316356	20,8949649	19,9375103	20,6552602	19,0690714
100	19,80354086	19,16660091	19,7940447	20,0486621	20,4518689	20,0953182

Como se puede apreciar tenemos 6 medidas (Peq1, Peq2, Peq3, Peq4, Peq5 y Original) lo que nos permite aplicar intervalos de confianza para la representación de gráficas. Este concepto se encuentra explicado con detalle en el ANEXO I, pero básicamente se emplea para dar mayor valor rigor científico a las simulaciones.

En cuento a los valores, se encuentran próximos a 21dB que son valores bajos de PSNR. Sin embargo, como veremos en el resto de los bloques, para DSR será un valor habitual.

En la *Tabla 6.3* vemos los valores de PSNR para el protocolo ap-MMDSR:

Tabla 6.3 Valores de PSNR para ap-MMDSR en el escenario pequeño

	MEDIDAS PSNR (dB) para ap-MMDSR					
Tiempo (segundos)	**Peq1**	**Peq2**	**Peq3**	**Peq4**	**Peq5**	**Original**
10	23,5151546	23,5151546	20,0586336	21,242833	22,8459279	22,7144385
20	27,98333219	29,98333219	28,6434824	27,5212005	27,9021325	27,2914316
30	31,05802484	31,05802484	31,0559778	31,0303761	30,0675459	30,3409399
40	36,26726441	37,21550836	37,9182168	36,1760269	37,2904618	35,9335825
50	35,48729473	34,04547035	34,4118222	35,9345125	35,2869118	34,3908364
60	33,49470302	33,49049299	34,4070813	33,438616	33,9409188	32,9422224
70	28,1516048	29,99906743	31,5465997	28,1516048	29,3106247	32,3107195
80	41,18894036	41,34038208	40,0528913	40,5209469	37,6166643	40,2659969
90	30,82400417	31,09649889	33,5525597	30,020028	31,517005	30,2424179
100	30,58929643	31,3211433	33,9432675	30,3048512	31,8081741	29,9041239

En este caso, se ve que con el protocolo ap-MMDSR se ha producido un aumento de los valores de PSNR lo que nos da a entender que el enrutamiento se está haciendo de una manera más eficiente que en el caso de DSR.

Por último, en la *Tabla 6.4* podemos ver los valores de PSNR correspondientes al caso de d-MMDSR:

Tabla 6.4 Valores de PSNR para d-MMDSR en el escenario pequeño

	MEDIDAS PSNR para d-MMDSR					
Tiempo (segundos)	**Peq1**	**Peq2**	**Peq3**	**Peq4**	**Peq5**	**Original**
10	24,5551546	24,5151546	21,0546336	22,242833	22,8459279	22,7144385
20	29,98333219	30,17333219	27,9434824	27,4312005	28,9221325	28,3814316
30	31,19802484	32,27802484	33,6159778	31,5203761	30,9875459	30,3409399
40	37,03726441	39,81550836	39,1382168	37,1160269	37,3304618	36,0135825
50	38,56729473	34,66547035	34,6718222	36,2345125	36,7669118	34,3908364
60	34,77470302	32,49049299	34,8370813	32,798616	33,3209188	32,0422224
70	30,6916048	34,17906743	30,2765997	33,1816048	33,9531062	32,3107195
80	41,68894036	42,93038208	42,8432891	41,5609469	41,8766643	41,3659969
90	31,08400417	30,71649889	33,8325597	29,890028	31,787005	30,2424179
100	32,66929643	30,5511433	32,6132675	33,1648512	33,6481741	32,3741239

Podemos apreciar que no hay muchas diferencias entre los valores correspondientes a la *Tabla 6.3* y los presentados en la *Tabla 6.4*, sin embargo,

ésta última los mejora. Este mejor comportamiento es debido a las particularidades del protocolo d-MMDSR que relatamos a continuación.

6.3.3. Gráficas de la PSNR. Escenario Pequeño

En primer lugar presentamos la gráfica comparativa entre los protocolos DSR y ap-MMDSR basado en los valores anteriores. Todas las gráficas presentan intervalos de confianza del 95% obtenidos a partir de cinco simulaciones realizadas para cada escenario.

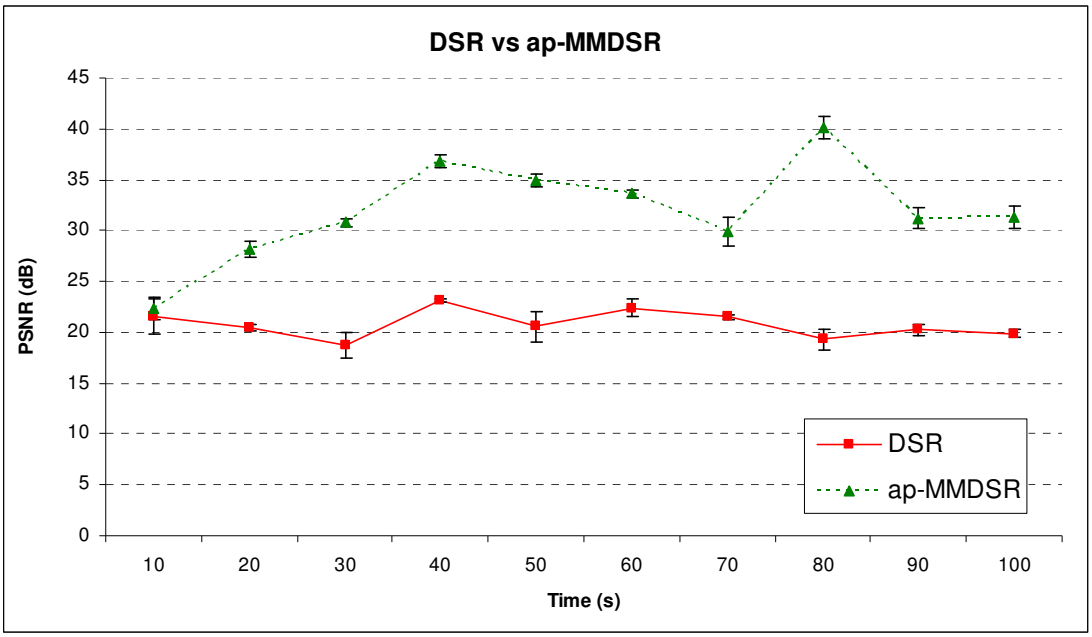

Figura 6.1. Escenario Pequeño. Protocolo DSR vs Protocolo ap-MMDSR

La primera conclusión que se estable observando la gráfica *Fig 6.1* es que el valor promedio de la PSNR para DSR es substancialmente inferior que para ap-MMDSR. La razón es que DSR elige el camino más corto entre el nodo fuente y el nodo destino pero no tiene en cuenta si este camino es el que posee mayor tráfico interferente. Sin embargo, ap-MMDSR desarrolla el algoritmo comentado en el capítulo 2, donde se descubren los posibles caminos y se clasifican dando mayor prioridad aquellos que tienen menores pérdidas. También se puede

observar como al comienzo las diferencias entre ambos protocolos no son grandes. Esto se explica ya que el algoritmo de enrutamiento ap-MMDSR necesita algun segundo para poder descubrir todos los caminos y clasificarlos.

Se puede ver como entorno a los 50 segundos de simulación, el protocolo ap-MMDSR sufre un poco de desvanecimiento ya que se manifiesta un descenso de los valores de PSNR. Esto puede ser debido a que los nodos se encuentran en continuo movimiento y algunos han podido perder el contacto con su vecino en el camino y por tanto el camino se rompería. En ese caso, el algoritmo debe buscar nuevas rutas.

Ahora incluimos en nuestro análisis al protocolo d-MMDSR, como se puede ver en la *Fig 6.2*:

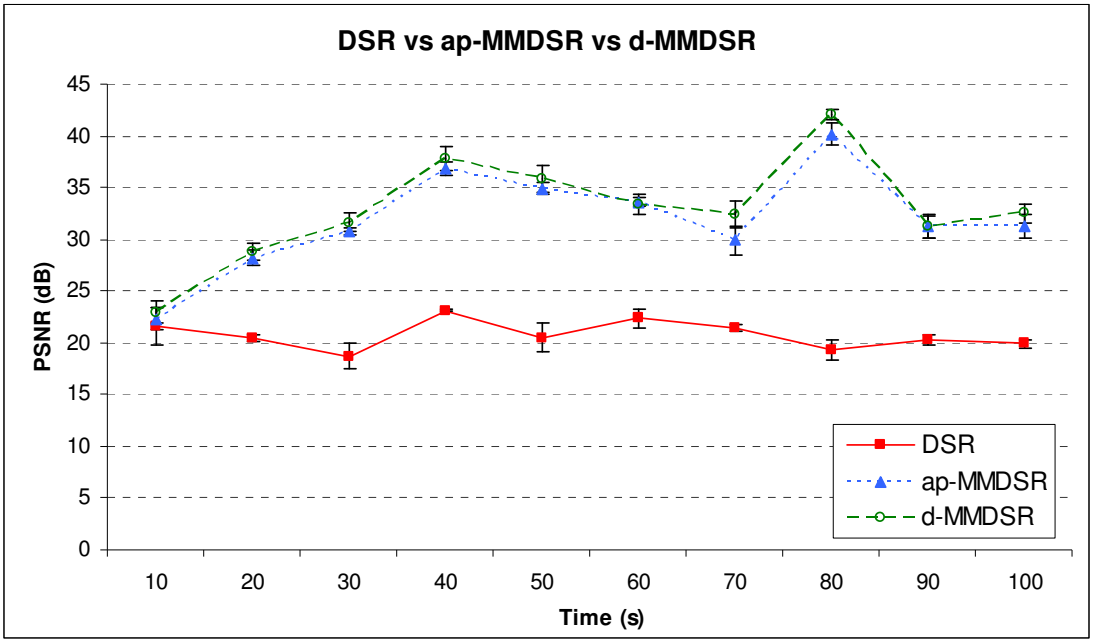

Figura 6.2. Escenario Pequeño. Protocolo DSR vs ap-MMDSR vs d-MMDSR

De esta gráfica se aprecia que el protocolo d-MMDSR realmente mejora al ap-MMDSR en aproximadamente 1dB en promedio. Esta mejora viene por como

este protocolo trabaja la asignación de los valores de ventana de contención para diferenciar a los nodos en las distintas categorías de acceso. De manera, que si dos nodos vecinos consiguen transmisiones satisfactorias se evitan colisiones. En el escenario pequeño, las dimensiones son reducidas y por tanto el peligro de colisiones aumenta.

Podemos concluir que en un escenario de dimensiones reducidas, con tráfico interferente y nodos en movimiento las PSNR se mantiene en valores altos en los protocolos ap-MMDSR y d-MMDSR. Además, este último mejora aún más al ap-MMDSR debido a que se comporta mejor en situaciones de colisiones.

6.4. Bloque 2: Escenario Grande

En este bloque se quiere analizar cómo se comportan los nodos en situaciones como un barrio dentro de una ciudad, un entorno de campus universitario o simplemente el metro de una gran ciudad. Las dimensiones hacen que los nodos estén más repartidos, sin embargo, su número es adecuado para la creación de caminos.

Bajo este contexto se va volver hacer un estudio de los protocolos DSR, ap-MMDSR y d-MMDSR para completar su análisis en otro tipo de escenario. En este caso deberíamos obtener unos resultados similares a los del escenario pequeño, es decir, el protocolo DSR debe dar unos valores de PSNR más bajos que los protocolos multicamino.

6.4.1. Características del escenario grande

Como se comentaba anteriormente, este escenario representa situaciones de entornos en zonas turíticas, zonas de rescates, parques naturales, estaciones de esquí, etc. Es decir, lugares que se caracterizan por su gran extensión. Además, el tráfico interferente CBR en esta ocasión es 3 Mb. Con esta tasa situamos a la

red en situración, por lo que tendremos pérdidas como se verá en las secciones posteriores.

Las características de este escenario se encuentran resumidas en la siguiente *Tabla 6.5.:*

Tabla 6.5. Características del Escenario Grande

Area	400x400
Número de Nodos	50
Vel. Máx. nodos	2 m/s
Cobertura	120 m
Patrón de movimiento	Random Waypoint
Especificación MAC	IEEE 802.11e
Ancho de Banda	11 Mbps
Tiempo de Simulación	100s
Codificación de vídeo	MPEG-2 VBR (Blade Runner)
Tasa del vídeo	150 kbps
Protocolo de transporte	RTP/RTCP/UDP
Tamaño máximo de paquete	1500 bytes
Tamaño de las colas	80 paquetes
Fuentes de vídeo	1
Ruido de canal	-92 dBm
Múltiples rutas	De 1 a 5

Podemos apreciar como el número de rutas ha aumentado, esto se consigue gracias a lo ya comentado y al haber aumentado el número de nodos a 50. El resto de parámetros permanecen similares al escenario pequeño ya que se busca poder hacer una comparativa fiable entre ambos. Igual que en el caso anterior, la velocidad máxima de los nodos permanece a 2 m/s porque suponemos el caso de personas caminando o corriendo.

6.4.2. Valores de la PSNR.

Para el escenario grande y el protocolo de enrutamiento DSR obtenemos la siguiente tabla de valores *Tabla 6.6.*:

Tabla 6.6. Valores de PSNR para DSR en el escenario grande

	MEDIDAS PSNR (dB) para DSR			
Tiempo (segundos)	**Gra1**	**Gra2**	**Gra3**	**Original**
10	23,2129064	23,21290643	23,2129064	23,4049957
20	22,1840876	22,18408759	22,1840876	22,2320042
30	23,2522777	23,2522777	23,2522777	22,5432659
40	20,5267326	20,52673265	20,5267326	20,4212993
50	20,9028753	20,90287529	20,9028753	20,6994185
60	20,1734736	20,17347359	20,1734736	20,5856318
70	18,4956735	18,00275931	18,4956735	19,7033325
80	21,3875297	21,40463169	23,3875297	21,6832252
90	20,5087489	20,42063106	20,5087489	19,3039809
100	19,7146824	18,92497427	19,7146824	21,7451852

En general se obtienen valores un poco mejores que para las mismas circunstancias en el escenario pequeño. La diferencia es muy pequeña y el protocolo DSR sigue mostrando sus deficiencias en este escenario si lo comparamos con los valores que vemos en la *Tabla 6.7.* correspondientes al caso de ap-MMDSR:

Tabla 6.7. Valores de PSNR para ap-MMDSR en el escenario grande

	MEDIDAS PSNR (dB) para ap-MMDSR			
Tiempo (segundos)	**Gra1**	**Gra2**	**Gra3**	**Original**
10	30,2129064	30,21290643	30,2129064	28,9039157
20	31,1840876	31,18408759	31,1840876	32,3723238
30	33,2522777	33,2522777	33,2522777	34,8443888
40	40,5267326	40,52673265	40,5267326	40,6828024
50	35,9028753	35,90287529	35,9028753	36,7269753
60	35,1734736	35,17347359	35,1734736	34,2402793
70	38,4956735	38,49567346	38,0527593	37,8081298
80	36,3875297	36,38752966	36,4042964	37,7021736
90	36,3373237	36,50874892	36,298303	34,3267179
100	33,6549439	33,7146824	33,8851178	34,9660208

Efectivamente los valores que se muestran correspondientes a la PSNR para el protocolo ap-MMDSR son mejores que para el DSR. Se aprecia como para 40 segundos existe un pico de valor de PSNR. Este valor implica que la comparativa entre las *frames* es casi perfecta y por tanto no existen a penas diferencias. Este aspecto desde el punto de vista del usuario final implica que el vídeo que esta viendo en su terminal ha sufrido escasas pérdidas con respecto al original en ese instante.

Por último mostramos también los valores correspondientes al d-MMDSR en la *Tabla 6.8.*:

Tabla 6.8. Valores PSNR para d-MMDSR en el escenario grande

	MEDIDAS PSNR (dB) para d-MMDSR			
Tiempo (segundos)	**Gra1**	**Gra2**	**Gra3**	**Original**
10	30,2129064	30,21290643	30,2129064	28,9039157
20	32,2860876	32,28608759	32,2860876	33,1623238
30	34,7922777	34,7922777	34,7922777	35,1547888
40	40,5867326	40,58673265	40,5867326	40,8828024
50	36,0128753	36,01287529	36,0128753	36,9069753
60	35,2834736	35,39347359	35,2834736	35,9102793
70	38,9156735	38,93567346	38,9327593	38,7981298
80	39,3875297	39,38752966	39,3142964	38,2121736
90	37,3173237	37,45874892	37,388303	36,3867179
100	34,1749439	33,9146824	33,7851178	35,1760208

Como era de esperar por lo visto en con el escenario pequeño, las diferencias no son significativas entre el protocolo ap-MMDSR y el d-MMDSR. Podemos observar como hasta el instante de 70 segundos, poseen una gran similitud pero en los instantes finales la PSNR mejora. La PSNR es un parámetro acumulativo por definición (es un sumatorio), por tanto el d-MMDSR ha estado haciendo un enrutamiento más eficiente que el ap-MMDSR durante toda la simulación pero realmente esta mejora se aprecia al final.

Los valores no difieren mucho porque el protocolo ap-MMDSR ya hace un enrutamiento de calidad por lo que el d-MMDSR mejora algo que ya funciona bien.

6.4.3. Gráficas de la PSNR. Escenario Grande

Al igual que para el bloque 1 empezamos mostrando el gráfico conjunta para DSR y ap-MMDSR y en este bloque de simulaciones se presentan también las gráficas con un intervalo de confianza del 95% obtenidas a partir de cuatro simulaciones realizadas sobre el escenario.

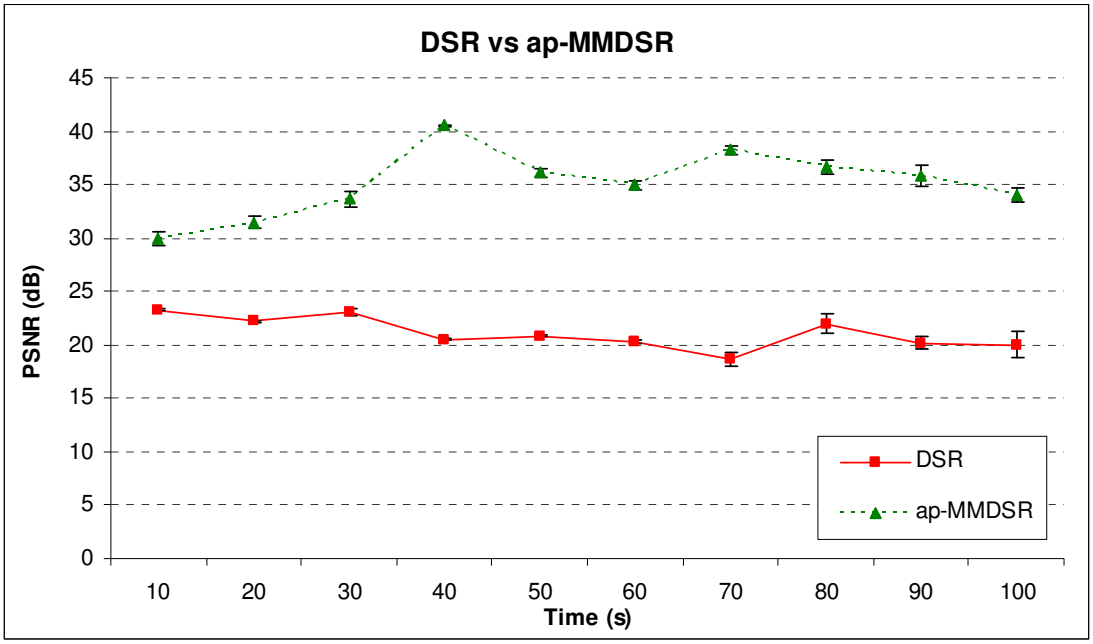

Figura 6.3. Escenario Grande. Protocolo DSR vs ap-MMDSR

Comprobamos gráficamente lo que hemos notado al ver los valores. En el caso de un escenario grande, el protocolo ap-MMDSR mantiene unos valores altos de PSNR durante toda la simulación mejorando el comportamiento del protocolo PSNR. Las diferencias se mantienen del orden de 10dB llegando a tener picos donde la diferencia entre ambos es de incluso hasta 20dB. Estas diferencias se justifican con el hecho de que DSR es un protocolo reactivo, que solo busca nuevas rutas cuando lo necesita. Esto es un punto grave en su contra a lo hora

de transmitir *streaming* de vídeo. Este comportamiento implica que existe un período de tiempo donde el protocolo todavía no es consciente de que se ha roto el camino, y además hay que sumarle el nuevo descubrimiento de los mismos. Ese hecho provoca pérdidas de paquetes con sus *frames* dentro y consequentemente un deterioro en la calidad de la imagen para el usuario final.

Sin embargo, el protocolo MMDSR y su versión ap-MMDSR no espera a que las rutas se rompan para refrescar el algoritmo de búsqueda. Se evalúa el parámetro de la Probabilidad de Error de Camino y con él se hace una estimación de cuándo puede ser que una ruta tanga una alta probabilidad de romperse, y entonces se busca una ruta alternativa antes de que se rompa de verdad. De esta forma, las posibilidades de perder paquetes descienden, como se refleja en la gráfica de la *Fig 6.3* en que la PSNR es mayor.

Para el bloque 2 tampoco hemos perdido la oportunidad de comparar también el protocolo d-MMDSR. Este se puede ver en la gráfica *Fig 6.4.*:

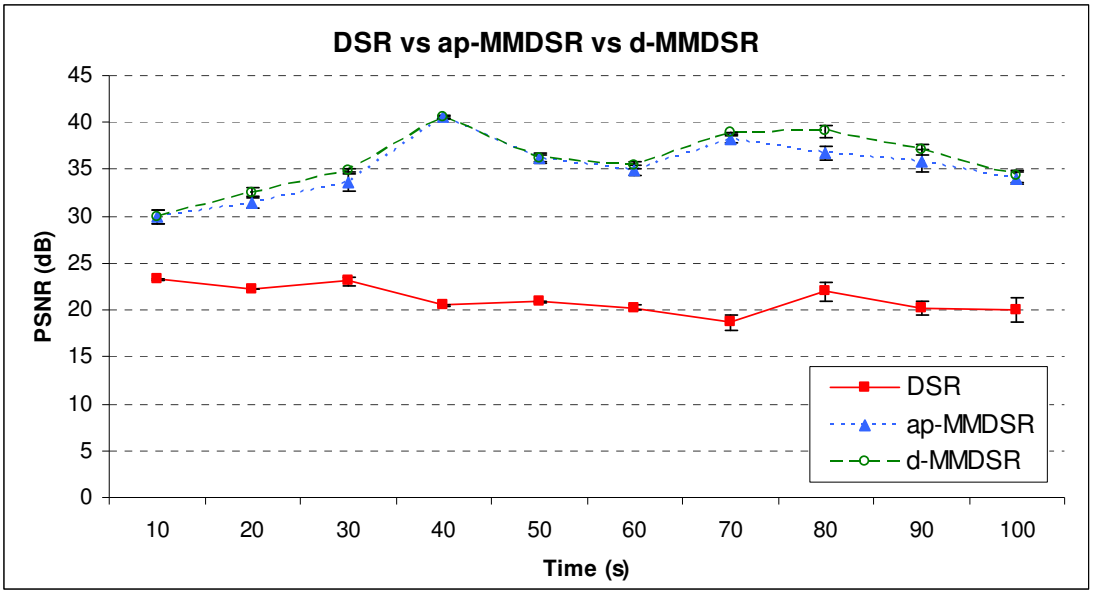

Figura 6.4. Escenario Grande. Protocolo DSR vs ap-MMDSR vs d-MMDSR

En esta ocasión el protocolo d-MMDSR también presenta un comportamiento mejor en términos de la PSNR. A lo largo de la simulación se aprecia que se mantiene por encima de los valores del protocolo ap-MMDSR pero muy próximo. De hecho si comparamos con la misma gráfica pero del escenario pequeño (*Fig 6.2.*) podremos ver que para los instantes comprendidos entre 40 segundos y 60, en esta ocasión existe menos diferencia. Esto se debe a que como ya explicamos en secciones anteriores, el protocolo d-MMDSR está centrado en regular el acceso al medio. Por tanto sus virtudes se reflejan mejor cuando estamos en entornos con posibilidad de colisiones. Para el escenario grande las colisiones se producen pero con menor probabilidad que en el caso del escenario pequeño.

Otro aspecto significativo de las gráficas que llevamos mostradas hasta ahora viene relacionado con el diseño de los protocolos DSR y las versiones ap-MMDSR y d-MMDSR. Hasta este punto hemos comprobado que las mejoras introducidas por MMDSR se reflejan en la PSNR. De hecho, otro aspecto que contribuye es el tratamiento de las *frames* en los diferentes protocolos. Las *frames* I son las más significativas e imprescindibles para la codificación. En el protocolo DSR, al no ser multicamino se envía por la misma ruta con B y P. Tanto en ap-MMDSR como en d-MMDSR se priorizan, y se envían por el mejor camino mientras que las P o B se hacen por los menos buenos. Esto provoca que en los casos multicamino aunque se produzcan pérdidas de *frames* P o B, el resultado final de la PSNR se mantiene con un valor alto gracias a que se salvaguardan las *frames* I.

Con los bloques 1 y 2 de simulaciones hemos pretendido cubrir las distintas posibilidades para los protocolos DSR, ap-MMDSR y d-MMDSR. Las conclusiones a las que hemos llegado son las siguientes:

- Tanto para el escenario pequeño como para el grande los protocolos multicamino (ap-MMDSR y d-MMDSR) obtienen mejores valores de PSNR que el DSR.
- El hecho de que los protocolo multicamino no esperen a que se rompan los caminos para refrescarlos contribuye a mejorar la PSNR.
- Las diferencias entre DSR y los protocolos multicamino al comienzo de la simulación son menores debido a que los algoritmos de búsqueda y clasificación de ap-MMDSR y d-MMDSR todavía no están funcionando.
- Cuanto mayor sea la densidad de nodos y por tanto la posibilidad de colisiones, mejor se reflejan las bondades del protocolo d-MMDSR con respecto a sus competidores.

6.5. Bloque 3: Escenario *game*

Este bloque ha sido diseñado para realizar un análisis del protocolo g-MMDSR con respecto a los protocolos DSR y ap-MMDSR. Como ya se explico en el capítulo 2, la Teoría de Juegos implica que haya más de un jugador. Por esta razón, se ha diseñado un escenario específico para este protocolo.

Además, se pretende constatar que el aplicar una teoría aparentemente estadística a un protocolo de enrutamiento podemos mejorar el funcionamiento de la red MANET en términos de QoS. Para ello, se realizará una evaluación de la PSNR, utilizando la calculadora diseñada en este proyecto, con lo que podremos terminar el estudio de las distintas versiones del protocolo MMDSR (diseñado por Víctor Carras Frías en su tesis [CAR09]).

En la *Fig 6.5.* se puede ver un esquema de lo que se pretende mostrar con este protocolo:

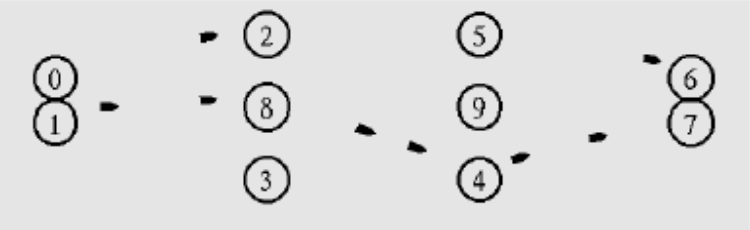

Figura 6.5. Esquema ejemplo para el protocolo g-MMDSR

En esta *Fig 6.5.,* a modo de ejemplo para el escenario *game,* se puede observar como tenemos dos nodos transmisores, 0 y 1, cuyos nodos destinos son 6 y 7 respectivamente. Existen un tráfico interferente entre origen y destino. El protocolo g-MMDSR funciona mandando con una cierta probabilidad (75% en este caso) las *frames* más relevantes, I, por el camino mejor y con otra probabilidad menor (25% en nuestro caso) las envía por un camino peor. Está política que en principio puede parecer errónea, probaremos a lo largo de las siguientes secciones que mejora la PSNR. La razón es que en ocasiones aquel que parece el peor camino puede estar menos congestionado y por tanto obtener una respuesta satisfactoria.

Una vez aclarado el objetivo que se busca para este bloque, pasamos a ver las características del escenario diseñado.

6.5.1. Características del escenario *game*

En la siguiente *Tabla 6.8.*, se encuentran las princípiales características de este escenario que ahora pasamos a detallar.

Tabla 6.9. Características del Escenario Game

Area	200x200
Número de Nodos	20
Vel. Máx. nodos	2 m/s
Cobertura	75 m
Patrón de movimiento	Random Waypoint
Especificación MAC	IEEE 802.11e
Ancho de Banda	11 Mbps
Tiempo de Simulación	100s
Codificación de vídeo	MPEG-2 VBR (Blade Runner)
Tasa del vídeo	150 kbps
Protocolo de transporte	RTP/RTCP/UDP
Tamaño máximo de paquete	1500 bytes
Tamaño de las colas	50 paquetes
Fuentes de vídeo	2
Tráfico CBR	1 Mbps
Múltiples rutas	De 1 a 3
Parámetro de Peso	1/7

Se ha diseñado un escenario con parámetros parecidos al escenario pequeño visto en el bloque 1. Sin embargo, se ha añadido una fuente más de vídeo para poder aplicar la Teoría de Juegos y que posteriormente veremos como influirá en el comportamiento general de los protocolos. Además, en cuento al tráfico interferente se ha reducido a 1 Mbps lo que como también veremos a continuación no implica una disminución de la distorsión.

La velocidad de movimiento de los nodos no se ha modificado por la misma razón que en los apartados anteriores, porque es similar a la de una persona en movimiento.

6.5.2. Valores de la PSNR. Escenario *Game*

Como en los casos anteriores, se empieza mostrando los valores obtenidos para el protocolo DSR. En la *Tabla 6.10.,* aparecen 4 escenarios generados aleatoriamente y para cada uno de ellos dos columnas, una para la fuente 0 de vídeo (*Source 0*) y otra para la fuente 1 (*Source 1*):

Tabla 6.10. Valores de PSNR para DSR en el escenario game

	MEDIDAS PSNR (dB) para DSR							
	Game1		Game2		Game3		Original	
Tiempo (segundos)	Source 0	Source 1	Source 0	Source 1	Source 0	Source 1	Source 0	Source 1
10	16,81384	17,95355	18,2364	17,73507	17,5015	18,2126	17,7865	16,027
20	19,17712	18,77182	18,5108	19,21581	19,2629	20,237	21,1582	19,8203
30	20,99921	19,55615	19,5562	18,67836	19,4396	19,5849	19,7474	19,6978
40	19,36709	17,58212	19,2414	19,6924	17,8466	17,1245	16,9327	16,16
50	18,64788	20,09398	18,2215	20,63925	20,1069	20,4706	19,7168	18,2819
60	17,11252	22,20065	18,2692	22,54423	17,8664	20,6178	18,8726	17,0459
70	20,57117	20,43999	19,7981	19,91566	17,1147	21,0312	17,305	19,2834
80	19,99649	22,69468	19,6799	19,85893	19,7133	21,1092	19,5668	19,0288
90	18,28454	20,20981	18,7736	19,18806	17,9737	20,4148	18,5076	19,9352
100	16,41621	17,66983	16,6103	18,803	18,1542	17,1999	16,9747	19,0204

Observando la tabla anterior podemos darnos cuenta que los valores de PSNR han disminuido en este escenario. Este descenso se aprecia tanto para la fuente 0 como para la fuente 1.

En la siguiente tabla se muestran los valores correspondientes al protocolo ap-MMDSR en el escenario *game*:

Tabla 6.11. Valores de PSRN para ap-MMDSR en el escenario game

	MEDIDAS PSNR (dB) para ap-MMDSR							
	Game1		Game2		Game3		Original	
Tiempo (segundos)	Source 0	Source 1	Source 0	Source 1	Source 0	Source 1	Source 0	Source 1
10	18,19783	19,81384	23,4491	17,52139	24,5673	19,1523	19,4759	17,6564
20	18,58119	16,1221	16,0407	17,29328	19,243	18,5941	21,2825	16,9762
30	21,08017	19,53631	21,0813	19,43478	18,2762	19,5751	20,5352	20,7532
40	20,2019	19,08179	19,0523	18,75693	20,1682	17,4093	18,1976	16,2948
50	20,85053	18,86937	19,504	20,71491	22,0658	21,7881	19,3904	18,4788
60	22,04608	22,18515	22,1688	22,04444	20,0573	20,9217	19,3023	27,1015
70	20,59495	20,41926	20,3881	20,68515	21,1538	21,0616	20,3804	19,2115
80	22,01697	22,31989	22,3456	21,79167	23,4789	21,1605	23,859	26,232
90	20,79167	23,68976	19,7778	23,47633	19,7629	20,2406	17,146	20,9308
100	20,1057	17,32888	17,3496	17,77986	17,7045	16,8932	20,2619	21,6889

Para el caso de ap-MMDSR también se ha puesto de manifiesto que tenemos valores más bajos de PSNR. Se explicará con más detalle cuando pasemos al análisis de las gráficas, pero las causas de este descenso se encuentran relacionadas con el hecho de que haya dos fuentes transmitiendo a la vez. Tenemos un número de nodos limitado, el mismo que en el caso del escenario pequeño, sin embargo, ahora se tienen que generar caminos para dos fuentes. Estos caminos pueden coincidir provocando conflictos y que se pierdan paquetes. La consecuencia es que desciende la calidad del vídeo recibido en las fuentes, como queda reflejado en la *Tabla 6.11*, con valores de incluso 16,1221 dB o 16,0407dB para instante de 20 segundos.

Por último, pasamos a ver los resultados obtenidos para el protocolo g-MMDSR que es para el que ha sido diseñado este escenario.

Tabla 6.12. Valores PSNR para g-MMDSR en el escenario game

	MEDIDAS PSNR (dB) para g-MMDSR							
	Game1		Game2		Game3		Original	
Tiempo (segundos)	Source 0	Source 1	Source 0	Source 1	Source 0	Source 1	Source 0	Source 1
10	18,19783	22,95707	23,5248	17,52139	24,5069	20,7211	19,4675	17,6564
20	18,65123	22,17712	19,9617	21,51006	19,1771	22,5474	24,4889	21,9762
30	20,09475	21,26079	21,2302	19,82743	18,7443	19,575	21,0975	20,7532
40	18,75693	21,23045	19,1556	22,78242	19,899	21,6965	21,3584	22,2948
50	20,88574	21,72786	20,1707	22,03647	22,1047	21,7858	21,626	23,4788
60	22,13186	22,20755	20,5245	23,84175	22,3292	20,7361	22,294	27,1015
70	19,83786	20,59138	20,9719	20,05285	21,1072	21,0314	22,9336	19,2115
80	21,986	22,62068	23,9203	20,64703	23,2168	21,7876	23,0077	26,232
90	21,1205	22,46281	19,7781	21,44685	22,2475	20,1407	21,7374	20,9308
100	18,80432	20,51309	19,1852	20,82404	17,2974	21,7732	22,891	21,6889

Para el protocolo g-MMDSR se mejora la PSNR con respecto a los anteriores. Esta mejora se encuentra relacionada con la aplicación de la Teoría de Juegos que constituye un aspecto positivo en este escenario. En la siguiente sección veremos con más detalle estos resultados sobre las gráficas.

6.5.3. Gráficas de la PSNR. Escenario *Game*

Viendo los valores obtenidos en la sección anterior ya podemos imaginarnos que las cosas para este escenario van a variar con respecto a lo visto hasta ahora. Como el objetivo fundamental del bloque 3 es hacer un análisis del protocolo g-MMDSR, la primera gráfica es una comparativa entre g-MMDSR y DSR. Useramos un intervalo de confianza del 95% para este bloque de gráficas obtenido a partir de cuatro simulaciones sobre este escenario.

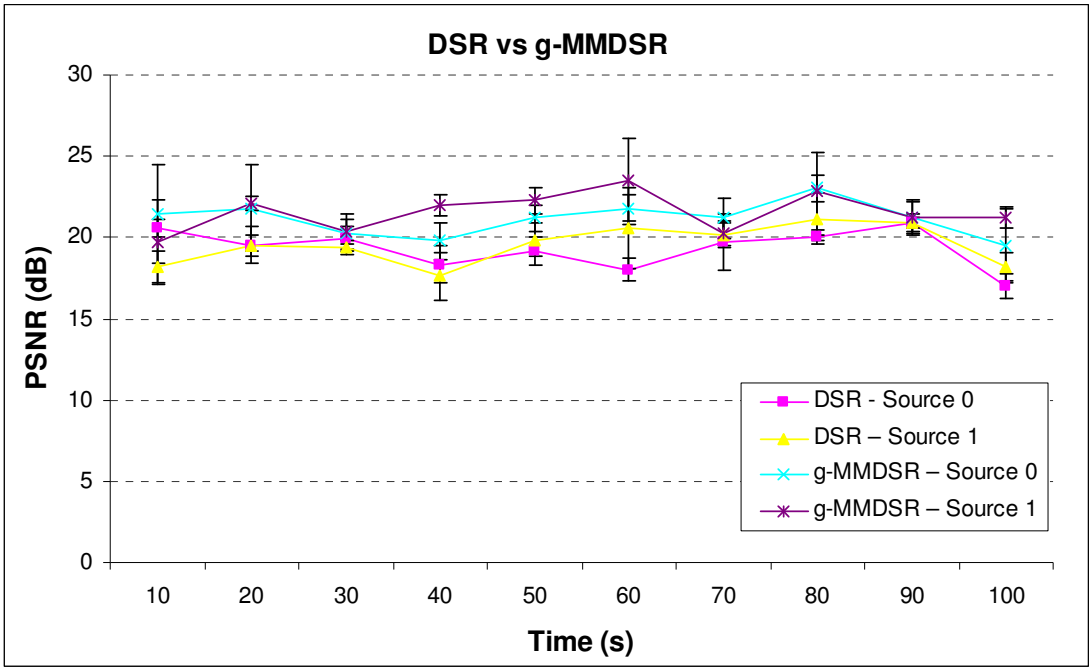

Figura 6.6. Escenario Game. Protocolo DSR vs g-MMDSR

Lo primero que constatamos es que para ninguno de los dos protocolos obtenemos ningún valor superior a 25dB. Esta diferencia con respecto a los bloques anteriores ya la habíamos empezado a notar en la sección anterior. La explicación es que ahora hay mucho más tráfico en la red, ésta se congestiona. Los nodos reciben tráfico de distintas fuentes pero si coinciden, descartan el paquete de una, esto puede ser interpretado por el protocolo de enrutado como la ruptura de un camino por lo que se debe de buscar otro. Estos factores provocan que en general la PSNR disminuya sus valores máximos.

Si pasamos analizar concretamente los resultados obtenidos en la *Fig 6.6.*, podemos ver como tanto para la fuente 0 como 1 los resultados de g-MMDSR son mejores que los de DSR. Sin embargo, en esta ocasión las diferencias no son tan altas como en el casos anteriores. Este hecho esta relacionado con lo comentado en el párrafo anterior, en entornos congestionados, con nodos sobrecargados, no es tan evidente que el refresco de rutas antes de que se

rompan sea tan beneficioso ya que cuando hay que descubrir nuevas puede ser que estén siendo empleando la otra fuente de vídeo.

En la *Fig 6.7.*, se ve la gráfica comparativa entre los protocolos g-MMDSR y ap-MMDSR:

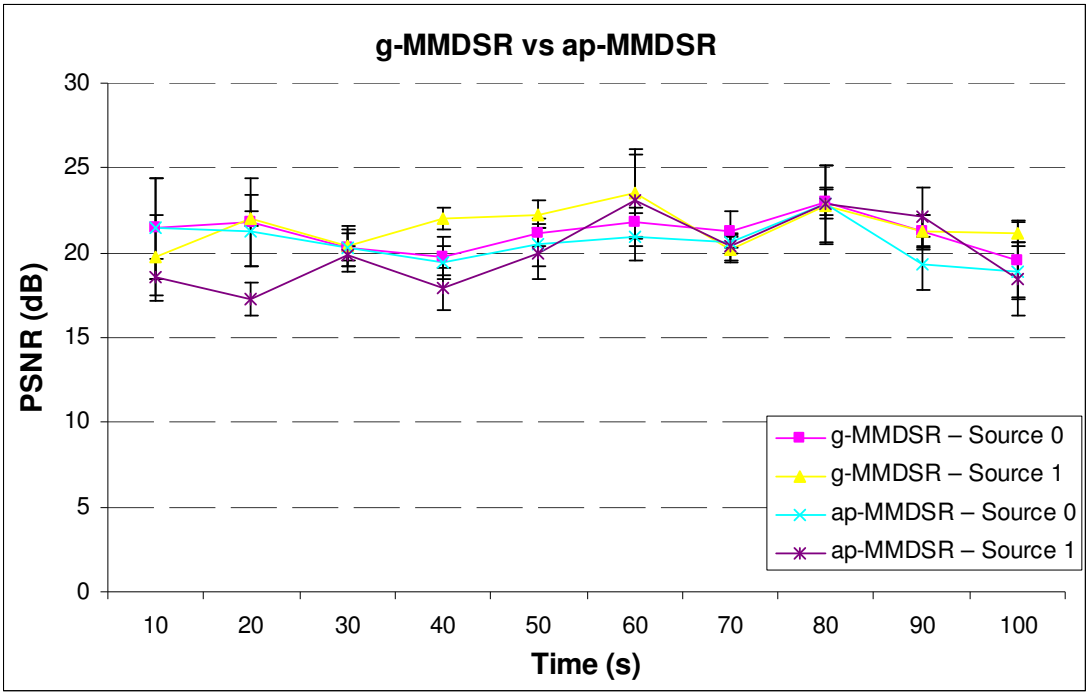

Figura 6.7. Escenario Game. Protocolo g-MMDSR vs ap-MMDSR

Si observamos con detenimiento la figura anterior podemos obtener una serie de resultados curiosos. En primer lugar, fijándonos en la fuente 1, hasta los 60 segundos de simulación vemos como claramente el protocolo g-MMDSR tienen un comportamiento mejor en términos de PSNR con respecto a ap-MMDSR. Sin embargo de ese tiempo al final, la situación se iguala y nos podríamos preguntar porqué. La explicación está relacionada con el propio diseño del protocolo g-MMDSR. En principio, con una probabilidad alta se envían los paquetes con *frames* I por el mejor camino pero como ya se comentó, existe una probabilidad de enviar las *frames* I pero el peor camino. Hasta los 60 segundos, esta política mejora la PSNR pero a partir de ahí, no funciona tan bien.

En segundo lugar, analizando la fuente 0, podemos ver que existe una mejora constante del protocolo g-MMDSR con respecto al protocolo ap-MMDSR. Esto constata que para casos con más de una fuente, las opciones de enrutamiento a utilizar serían, g-MMDSR, ap-MMDSR y DSR en ese orden.

Por último, se ha incluido una gráfica más en la que se hace una comparativa entre los tres protocolos para la fuente 1:

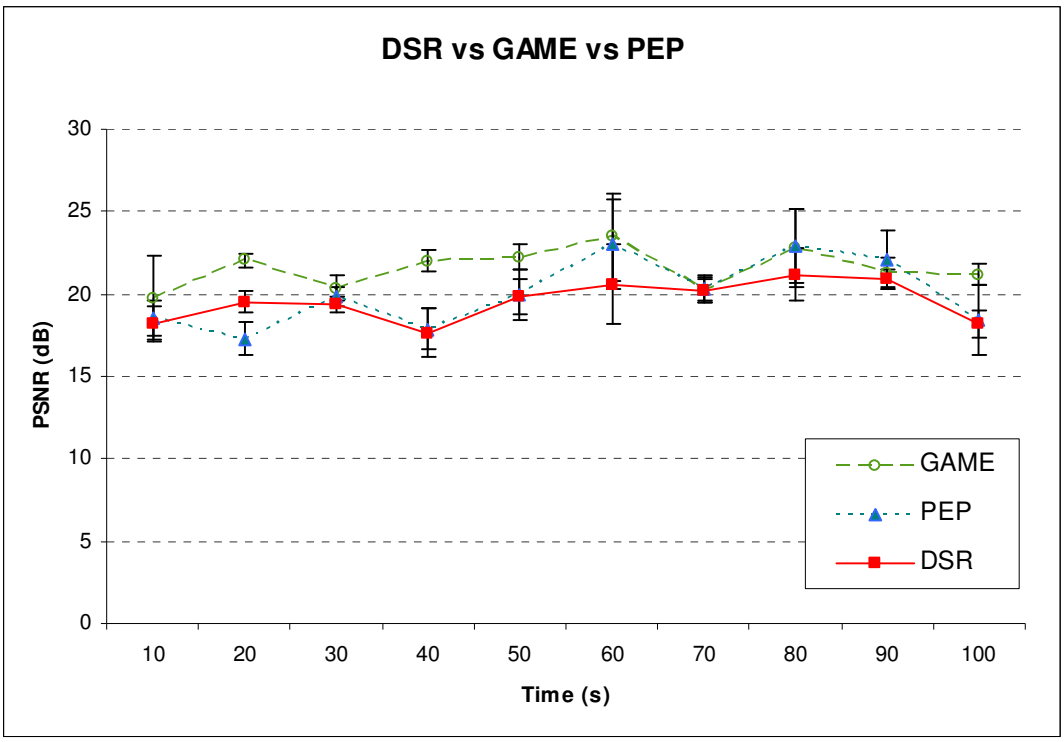

Figura 6.8. Escenario Game. DSR vs ap-MMDSR vs g-MMDSR fuente1

En esta gráfica podemos reafirmar las conclusiones que hemos obtenido con las gráficas anteriores. La versión bautizada como GAME (g-MMDSR), se comporta mejor que DSR y PEP (ap-MMDSR) en términos de PSNR. Sin embargo, en el caso de tener más de una fuente transmitiendo, con un número más o menos reducido de nodos y unos dimensiones pequeñas las diferencias entre los

protocolos disminuyen. Poniéndose de manifiesto en la *Fig 6.8.* que desde el instante 10 segundo hasta el instante 30 segundos el protocolo DSR tiene un comportamiento de aproximadamente 1dB mejor que el PEP.

Capítulo 7: Conclusiones y líneas futuras

Este proyecto fin de carrera se encuadra dentro del contexto de una nueva generación de redes cuya expansión, eficiencia y despliegue comienza a ser una realidad. Las redes MANET (*Mobile Ad-hoc Networks*) vienen a cubrir las deficiencias que otras como *Wireless* o *Bluetooth* no han podido satifacer, debido a su fuerte dependencia de un nodo central.

Es un dato objetivo que los servicios multimedia están sufriendo un increíble aumento en número de usuarios, soportes, protocolos o *software* especializado. Dada su relevancia, en este proyecto se ha hecho un análisis de prestaciones de servicios de vídeo bajo demanda en redes MANET desde el punto de vista de un estudio de *Quality of Service* (QoS). Con esta finalidad se ha desarrollado una calculadora de PSNR (*Peak Signal-to Noise Ratio*). Como se ha comentado a lo largo del proyecto, este parámetro a parte de dar una estimación de la calidad el vídeo transmitido es un indicador del comportamiento de los protocolos de

enrutamiento sobre redes MANET, por lo que sirve para comparar su prestaciones de manera fácil.

En principio se ha hecho un análisis teórico de los protocolos de enrutamiento desarrollados en la tesis de Víctor Carrascal Frías [CAR09], para una vez implementada la herramienta de cálculo de PSNR, poder aplicarla sobre ellos y conocer su funcionamiento en entornos de vídeo bajo demanda sobre redes MANET. Posteriormente se ha utilizado el simulador ns-2 [NS2] para crear escenarios parecidos a la realidad y hacer estudios comparativos del comportamiento de los protocolos MMDSR. Con los resultados obtenidos de las simulaciones se ha aplicado la calculadora, obteniendo así la PSNR en los distintos escenarios evaluados y realizando un análisis profundo de resultados.

Posteriormente pasaremos a detallar las conclusiones obtenidas tras la realización de este proyecto desde dos puntos de vista, por una lado, de la implementación de la calculadora y por otro, de la aplicación de esta misma para el análisis de protocolos de enrutamiento sobre redes MANET.

7.1. Desarrollo de la calculadora PSNR

El eje fundamental de este proyecto ha sido el desarrollo de una herramienta capaz de obtener el valor de la PSNR entre vídeos que han sido transmitidos a través de una red MANET. Las conclusiones que obtenemos de su implementación son:

- La calculadora se ha programado en AWK y *bash shell*, lo que le concede una gran portabilidad en aquellos entornos donde mayoritariamente se utiliza el simulador ns-2, es decir entornos UNIX y LINUX.
- Permite su utilización en escenarios con más de una fuente, computando los valores de PSNR individuales para cada uno de ellos.

- Genera como salida un fichero con el valor medio de PSNR entre ambos vídeos pero también con la comparación cuadro a cuadro (*frame a frame*).
- Posee un alto grado de precisión en los cálculos, ya que como se ha comprobado en el capítulo 6, os valores obtenidos se corresponden con aquellos que se preveían teóricamente.

En definitiva, la calculadora PSNR desarrollada en este proyecto supone una aportación importante como herramienta de evaluación de prestaciones de arquitectura de red para ofrecer servicios de vídeo bajo demanda sobre redes MANET. La métrica de QoS en que nos hemos basado es la PSNR, que permite evaluar de forma numérica la calidad subjetiva del vídeo que percibe el usuario.

7.2. Estudio de protocolos de enrutamiento

Gracias al desarrollo de la calculadora ha sido posible obtener valores de la PSNR de simulaciones realizadas con la herramienta ns-2. Este parámetro nos ha permitido establecer una serie de conclusiones sobre los protocolos MMDSR diseñados en [CAR09] que hasta ahora no había sido posible.

- En general, los protocolos multicamino desarrollados en la tesis de Víctor Carrascal Frías [CAR09] se comportan, mejor en términos de la PSNR, que el protocolo DSR. Es decir, se obtienen valores más altos de PSNR.
- El protocolo *adaptative - Multipath Multimedia Dynamic Source Routing* (ap-MMDSR) presenta valores de PSNR buenos, tanto escenarios pequeños como grandes.
- Sin embargo, en entornos con posibilidad de congestión, escenarios pequeños con alta densidad de nodos, el protocolo *Dynamic Contention Window – Multipath Multimedia Dynamic Source Routing* (d-MMDSR) mejora los valores de PSNR obtenidos por ap-MMDSR.
- En entornos con más de un nodo fuente transmitiendo, se ha probado que el protocolo que mejor se comporta es *Game - Multipath Multimedia*

Dynamic Source Routing (g-MMDSR) gracias a su posibilidad de enviar en determinadas ocasiones los cuadros I, que contienen la información más relevante para el proceso de decodificación de imagen, por el camino teóricamente peor pero más descongestionado.

- De las simulaciones obtenemos también que todos los protocolos proporcionan unos niveles altos de PSNR, pero que los multicamino no alcanza su mejor funcionamiento hasta pasados unos segundos de la simulación. Este es el tiempo que necesita para establecer el esquema multicamino.

7.3. Líneas futuras de trabajo

Como ya se ha hecho referencia anteriormente en este proyecto, las redes MANET se encuentra muy de actualidad y los grupos de investigadores que proponen mejoras están creciendo cada año. Por tanto, los campos en los que se puede trabajar son muy amplios y aquí mencionamos algunas líneas que se pueden continuar a partir de nuestro proyecto:

- La implantación de una interfaz gráfica para la calculadora que podría consistir en un modulo escrito en TCL (*Tool Command Language*) que se distribuiría con el simulador ns-2 en una futura entrega.
- La extensión del estudio de la PSNR a otros protocolos de enrutamiento para MANET mencionados en este proyecto.
- La creación de nuevos escenarios de simulación donde se modifican aspectos no tanto del enrutamiento sino de la codificación de vídeo y el tratamiento del vídeo, donde el conocer valores de PSNR puede ser una gran ayuda para su evaluación y diseño
- La extensión del estudio de la PSNR a otros escenarios de redes Ad-hoc, como las WSN (*Wireless Sensor Networks*) y las VANET (*Vehicular Ad-hoc Networks*).

ANEXO I: Cálculo de Intervalos de Confianza

En el contexto de estimar el valor de un parámetro poblacional, un **intervalo de confianza** es un rango de valores (calculado de una muestra) en el cual se encuentra el verdadero valor del parámetro, con una probabilidad determinada.

La probabilidad de que el verdadero valor se encuentre en el intervalo construído se denomina **nivel de confianza** y se denota 1-α. La probabilidad de equivocarnos se llama **nivel de significancia** y se simboliza con α. Generalmente se contruyen intervalos con confianza 1-α= 95% (o significancia α=5%). Menos frecuentes son los intervalos con α= 10%, (1-α = 90%) o α= 1%, (1-α = 99%).

Para construir un intervalo de confianza se puede comprobar que la distribución Normal Estándar cumple que:

$$P(-1.96 < Z < 1.96) = 0.95 \qquad \text{(A1.1)}$$

El valor 1,96 lo obtenemos de la tabla de distribución normal tipificada donde fijamos el valor α=95% y se observa se obtiene. Para otros intervalos de confianza como α=90% se corresponde con Z< 1,29, y para α=99% el valor es Z< 2,58.

Tabla A.1.1. Tabla distribución normal tipificada

Zp	0	1	2	3	4
0	0, 500 000	0, 841 344	0, 977 249	0, 998 650	0, 999 968
0,01	0, 503 989	0, 843 752	0, 977 784	0, 998 693	0, 999 969
0,02	0, 507 978	0, 846 135	0, 978 308	0, 998 736	0, 999 970
0,03	0, 511 966	0, 848 494	0, 978 821	0, 998 777	0, 999 972
0,04	0, 515 953	0, 850 830	0, 979 324	0, 998 817	0, 999 973
0,05	0, 519 938	0, 853 140	0, 979 817	0, 998 855	0, 999 974
0,06	0, 523 922	0, 855 427	0, 980 300	0, 998 893	0, 999 975
0,07	0, 527 903	0, 857 690	0, 980 773	0, 998 929	0, 999 976
0,08	0, 531 881	0, 859 928	0, 981 237	0, 998 964	0, 999 977
0,09	0, 535 856	0, 862 143	0, 981 691	0, 998 999	0, 999 978
0,1	0, 539 827	0, 864 333	0, 982 135	0, 999 032	0, 999 979
0,11	0, 543 795	0, 866 500	0, 982 570	0, 999 064	0, 999 980
0,12	0, 547 758	0, 868 643	0, 982 997	0, 999 095	0, 999 981
0,13	0, 551 716	0, 870 761	0, 983 414	0, 999 125	0, 999 981
0,14	0, 555 670	0, 872 856	0, 983 822	0, 999 155	0, 999 982
0,15	0, 559 617	0, 874 928	0, 984 222	0, 999 183	0, 999 983
0,16	0, 563 559	0, 876 975	0, 984 613	0, 999 211	0, 999 984
0,17	0, 567 494	0, 878 999	0, 984 996	0, 999 237	0, 999 984
0,18	0, 571 423	0, 880 999	0, 985 371	0, 999 263	0, 999 985
0,19	0, 575 345	0, 882 976	0, 985 737	0, 999 288	0, 999 986
0,2	0, 579 259	0, 884 930	0, 986 096	0, 999 312	0, 999 986
0,21	0, 583 166	0, 886 860	0, 986 447	0, 999 336	0, 999 987
0,22	0, 587 064	0, 888 767	0, 986 790	0, 999 358	0, 999 987
0,23	0, 590 954	0, 890 651	0, 987 126	0, 999 380	0, 999 988
0,24	0, 594 834	0, 892 512	0, 987 454	0, 999 402	0, 999 988
0,25	0, 598 706	0, 894 350	0, 987 775	0, 999 422	0, 999 989
0,26	0, 602 568	0, 896 165	0, 988 089	0, 999 442	0, 999 989
0,27	0, 606 419	0, 897 957	0, 988 396	0, 999 462	0, 999 990
0,28	0, 610 261	0, 899 727	0, 988 696	0, 999 480	0, 999 990
0,29	0, 614 091	0, 901 474	0, 988 989	0, 999 499	0, 999 991
0,3	0, 617 911	0, 903 199	0, 989 275	0, 999 516	0, 999 991
0,31	0, 621 719	0, 904 902	0, 989 555	0, 999 533	0, 999 991
0,32	0, 625 515	0, 906 582	0, 989 829	0, 999 549	0, 999 992
0,33	0, 629 299	0, 908 240	0, 990 096	0, 999 565	0, 999 992
0,34	0, 633 071	0, 909 877	0, 990 358	0, 999 581	0, 999 992
0,35	0, 636 830	0, 911 491	0, 990 613	0, 999 595	0, 999 993
0,36	0, 640 576	0, 913 084	0, 990 862	0, 999 610	0, 999 993
0,37	0, 644 308	0, 914 656	0, 991 105	0, 999 624	0, 999 993
0,38	0, 648 027	0, 916 206	0, 991 343	0, 999 637	0, 999 994
0,39	0, 651 731	0, 917 735	0, 991 575	0, 999 650	0, 999 994
0,4	0, 655 421	0, 919 243	0, 991 802	0, 999 663	0, 999 994

0,41	0, 659 096	0, 920 730	0, 992 023	0, 999 675	0, 999 994
0,42	0, 662 757	0, 922 196	0, 992 239	0, 999 686	0, 999 995
0,43	0, 666 402	0, 923 641	0, 992 450	0, 999 698	0, 999 995
0,44	0, 670 031	0, 925 066	0, 992 656	0, 999 709	0, 999 995
0,45	0, 673 644	0, 926 470	0, 992 857	0, 999 719	0, 999 995
0,46	0, 677 241	0, 927 854	0, 993 053	0, 999 729	0, 999 995
0,47	0, 680 822	0, 929 219	0, 993 244	0, 999 739	0, 999 996
0,48	0, 684 386	0, 930 563	0, 993 430	0, 999 749	0, 999 996
0,49	0, 687 933	0, 931 887	0, 993 612	0, 999 758	0, 999 996
0,5	0, 691 462	0, 933 192	0, 993 790	0, 999 767	0, 999 996
0,51	0, 694 974	0, 934 478	0, 993 963	0, 999 775	0, 999 996
0,52	0, 698 468	0, 935 744	0, 994 132	0, 999 784	0, 999 996
0,53	0, 701 944	0, 936 991	0, 994 296	0, 999 792	0, 999 997
0,54	0, 705 401	0, 938 219	0, 994 457	0, 999 799	0, 999 997
0,55	0, 708 840	0, 939 429	0, 994 613	0, 999 807	0, 999 997
0,56	0, 712 260	0, 940 620	0, 994 766	0, 999 814	0, 999 997
0,57	0, 715 661	0, 941 792	0, 994 915	0, 999 821	0, 999 997
0,58	0, 719 042	0, 942 946	0, 995 059	0, 999 828	0, 999 997
0,59	0, 722 404	0, 944 082	0, 995 201	0, 999 834	0, 999 997
0,6	0, 725 746	0, 945 200	0, 995 338	0, 999 840	0, 999 997
0,61	0, 729 069	0, 946 301	0, 995 472	0, 999 846	0, 999 997
0,62	0, 732 371	0, 947 383	0, 995 603	0, 999 852	0, 999 998
0,63	0, 735 652	0, 948 449	0, 995 730	0, 999 858	0, 999 998
0,64	0, 738 913	0, 949 497	0, 995 854	0, 999 863	0, 999 998
0,65	0, 742 153	0, 950 528	0, 995 975	0, 999 868	0, 999 998
0,66	0, 745 373	0, 951 542	0, 996 092	0, 999 873	0, 999 998
0,67	0, 748 571	0, 952 540	0, 996 207	0, 999 878	0, 999 998
0,68	0, 751 747	0, 953 521	0, 996 318	0, 999 883	0, 999 998
0,69	0, 754 902	0, 954 486	0, 996 427	0, 999 887	0, 999 998
0,7	0, 758 036	0, 955 434	0, 996 532	0, 999 892	0, 999 998
0,71	0, 761 148	0, 956 367	0, 996 635	0, 999 896	0, 999 998
0,72	0, 764 237	0, 957 283	0, 996 735	0, 999 900	0, 999 998
0,73	0, 767 304	0, 958 184	0, 996 833	0, 999 904	0, 999 998
0,74	0, 770 350	0, 959 070	0, 996 927	0, 999 907	0, 999 998
0,75	0, 773 372	0, 959 940	0, 997 020	0, 999 911	0, 999 998
0,76	0, 776 372	0, 960 796	0, 997 109	0, 999 915	0, 999 999
0,77	0, 779 350	0, 961 636	0, 997 197	0, 999 918	0, 999 999
0,78	0, 782 304	0, 962 462	0, 997 281	0, 999 921	0, 999 999
0,79	0, 785 236	0, 963 273	0, 997 364	0, 999 924	0, 999 999
0,8	0, 788 144	0, 964 069	0, 997 444	0, 999 927	0, 999 999
0,81	0, 791 029	0, 964 852	0, 997 522	0, 999 930	0, 999 999
0,82	0, 793 892	0, 965 620	0, 997 598	0, 999 933	0, 999 999
0,83	0, 796 730	0, 966 375	0, 997 672	0, 999 935	0, 999 999
0,84	0, 799 545	0, 967 115	0, 997 744	0, 999 938	0, 999 999
0,85	0, 802 337	0, 967 843	0, 997 813	0, 999 940	0, 999 999
0,86	0, 805 105	0, 968 557	0, 997 881	0, 999 943	0, 999 999
0,87	0, 807 849	0, 969 258	0, 997 947	0, 999 945	0, 999 999
0,88	0, 810 570	0, 969 946	0, 998 011	0, 999 947	0, 999 999
0,89	0, 813 267	0, 970 621	0, 998 073	0, 999 949	0, 999 999
0,9	0, 815 939	0, 971 283	0, 998 134	0, 999 951	0, 999 999
0,91	0, 818 588	0, 971 933	0, 998 192	0, 999 953	0, 999 999
0,92	0, 821 213	0, 972 571	0, 998 249	0, 999 955	0, 999 999
0,93	0, 823 814	0, 973 196	0, 998 305	0, 999 957	0, 999 999
0,94	0, 826 391	0, 973 810	0, 998 358	0, 999 959	0, 999 999
0,95	0, 828 943	0, 974 412	0, 998 411	0, 999 960	0, 999 999
0,96	0, 831 472	0, 975 002	0, 998 461	0, 999 962	0, 999 999
0,97	0, 833 976	0, 975 580	0, 998 510	0, 999 964	0, 999 999
0,98	0, 836 456	0, 976 148	0, 998 558	0, 999 965	0, 999 999
0,99	0, 838 912	0, 976 704	0, 998 605	0, 999 966	0, 999 999

La condición (A1.1) se puede comprobar con una tabla de probablidades o con un programa computacional que calcule probabilidades normales.

Luego, si una variable $\overline{X}$ tiene distribución normal, N de media y varianza (μ,σ^2), entonces el 95% de las veces se cumple:

$$-1.96 \leq \frac{\overline{X} - \mu}{\sigma}\sqrt{n} \leq 1.96 \qquad \text{(A1.2)}$$

Despejando μ en la ecuación (A1.2):

$$\overline{X} \text{ -1.96 x } \frac{\sigma}{\sqrt{n}} \leq \mu \leq \overline{X} + 1.96 \text{ x } \frac{\sigma}{\sqrt{n}} \qquad \text{(A1.3)}$$

El resultado es un intervalo de confianza que incluye el μ el 95% de las veces. Es decir, un intervalo de confianza al 95% para la media μ cuando la variable X es normal y σ^2 es conocido.

Intervalo de confianza para un promedio

Generalmente cuando se quiere construir un intervalo de confianza para la media poblacional μ, la varianza poblacional σ^2 es desconocida, por lo que el intervalo para μ construído en (A1.3) es muy poco práctico.

Si en el intervalo se reemplaza la desviación estándard poblacional σ, por la desviación Standard muestral S, el intervalo de confianza toma la forma:

$$\overline{X} \text{ -1.96 x } \frac{S}{\sqrt{n}} \leq \mu \leq \overline{X} + 1.96 \text{ x } \frac{S}{\sqrt{n}} \qquad \text{(A1.4)}$$

Lo cual es una buena aproximación para el intervalo de confianza del 95% para μ con σ^2 desconocido. Esta aproximación es mejor a medida que el tamaño muestra sea más grande.

Cuando el tamaño muestral es pequeño, el intervalo de confianza requiere utilizar la distribución t de Student (con n-1 grados de libertad, siendo n el tamaño de las muestras), en vez de la distribución normal (por ejemplo, para un intervalo de 95% de confianza, los límites del intervalo ya no serán construídos usando el valor 1,96).

Podemos ver la siguiente distribución para la t-Student:

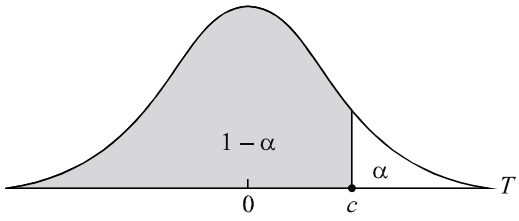

Figura A1.1. Función de distribución de probabilidad de t-Student

Donde la tabla de areas 1-α y valores $C = t_{1-\alpha,r}$, donde $P[T \leq c] = 1 - \alpha$ y donde T tiene distribución T-Student con *r* grados de libertad.

Tabla A1.2 Areas para la distribución T-Student

	1- α							
r	0.75	0.80	0.85	0.90	0.95	0.975	0.99	0.995
1	1.000	1.376	1.963	3.078	6.314	12.706	31.821	63.657
2	0.816	1.061	1.386	1.886	2.920	4.303	6.965	9.925
3	0.765	0.978	1.250	1.638	2.353	3.182	4.541	5.841
4	0.741	0.941	1.190	1.533	2.132	2.776	3.747	4.604
5	0.727	0.920	1.156	1.476	2.015	2.571	3.365	4.032
6	0.718	0.906	1.134	1.440	1.943	2.447	3.143	3.707
7	0.711	0.896	1.119	1.415	1.895	2.365	2.998	3.499
8	0.706	0.889	1.108	1.397	1.860	2.306	2.896	3.355
9	0.703	0.883	1.100	1.383	1.833	2.262	2.821	3.250
10	0.700	0.879	1.093	1.372	1.812	2.228	2.764	3.169
11	0.697	0.876	1.088	1.363	1.796	2.201	2.718	3.106
12	0.695	0.873	1.083	1.356	1.782	2.179	2.681	3.055
13	0.694	0.870	1.079	1.350	1.771	2.160	2.650	3.012
14	0.692	0.868	1.076	1.345	1.761	2.145	2.624	2.977
15	0.691	0.866	1.074	1.341	1.753	2.131	2.602	2.947
16	0.690	0.865	1.071	1.337	1.746	2.120	2.583	2.921
17	0.689	0.863	1.069	1.333	1.740	2.110	2.567	2.898
18	0.688	0.862	1.067	1.330	1.734	2.101	2.552	2.878
19	0.688	0.861	1.066	1.328	1.729	2.093	2.539	2.861
20	0.687	0.860	1.064	1.325	1.725	2.086	2.528	2.845
21	0.686	0.859	1.063	1.323	1.721	2.080	2.518	2.831
22	0.686	0.858	1.061	1.321	1.717	2.074	2.508	2.819
23	0.685	0.858	1.060	1.319	1.714	2.069	2.500	2.807
24	0.685	0.857	1.059	1.318	1.711	2.064	2.492	2.797
25	0.684	0.856	1.058	1.316	1.708	2.060	2.485	2.787
26	0.684	0.856	1.058	1.315	1.706	2.056	2.479	2.779

27	0.684	0.855	1.057	1.314	1.703	2.052	2.473	2.771
28	0.683	0.855	1.056	1.313	1.701	2.048	2.467	2.763
29	0.683	0.854	1.055	1.311	1.699	2.045	2.462	2.756
30	0.683	0.854	1.055	1.310	1.697	2.042	2.457	2.750
40	0.681	0.851	1.050	1.303	1.684	2.021	2.423	2.704
60	0.679	0.848	1.046	1.296	1.671	2.000	2.390	2.660
120	0.677	0.845	1.041	1.289	1.658	1.980	2.358	2.617
□	0.674	0.842	1.036	1.282	1.645	1.960	2.326	2.576

Para ilustrar el cálculo de intervalos de confianza que se ha empleado en todas las gráficas de PSNR mostradas en el capítulo 6, mostramos el siguiente ejemplo:

Tenemos un entorno del cual queremos conocer la tasa en bits por segundo que se están transmitiendo, para así caracterizar el medio y conocer el ancho de banda disponible. Para ello realizamos tres ejecuciones en el simulador para un mismo escenario variando un parámetro (en este caso hemos variado el tráfico interferente) y obtenemos 10 valores para cada uno de ellos. Podemos ver los resultados en la siguiente tabla donde además se encuentra incluído el promedio.

Tabla A1.3. Valores del caudal para 3 ejecuciones

Caudal Kbytes/s	Caudal en Kbytes/s	Caudal en Kbytes/s	**PROMEDIO**
362,000	341,76	343,952	349,237
369,024	341,424	340,076	350,175
366,120	341,712	346,856	351,563
365,412	344,232	345,78	351,808
366,628	341,136	337,268	348,344
368,552	340,944	339,98	349,825
365,648	339,876	340,076	348,533
364,196	336,528	338,912	346,545
370,748	336,588	334,748	347,361
370,276	341,76	336,104	349,380

Una vez obtenidos los valores de promedio para obtener el intervalo de confianza se computan los valores de la desviación estándard muestral, S, y con éste el valor $\frac{S}{\sqrt{n}}$. Lo podemos ver en la siguiente tabla:

Tabla A1.4 S y $\frac{S}{\sqrt{n}}$ para todos los caudales

Desviación estandar	**Desv/raiz n**
11,107	6,413
16,338	9,433
12,867	7,429
11,807	6,817
15,952	9,210
16,225	9,367
14,822	8,558
15,332	8,852
20,274	11,705
18,316	10,575

El siguiente paso sería conocer que intervalo de confianza deseamos, puede ser de 90%, 95% o 99%. En este ejemplo vamos a centrarnos en el valor de 99%. Para ello aplicamos la ecuación (A1.4) imponiendo los valores obtenidos en la tabla anterior y sustituyendo el valor 1,96 por 2,58 correspondiente al 99% y obtenido de la *Tabla A1.1*. Vemos estos valores en la siguiente tabla:

Tabla A1.5. Valores para el intervalo de confianza de 99%

0,5*duración intervalo 99%		
z alfa/2 99% * desv/raiz n	media-z alfa/2 99% * desv/raiz n	media+z alfa/2 99% * desv/raiz n
16,54458478	332,693	365,782
24,3363575	325,838	374,511
19,16577963	332,397	370,728
17,58699083	334,221	369,395
23,76163883	324,582	372,106
24,16806685	325,657	373,993
22,07842249	326,455	370,612
22,83848436	323,707	369,384
30,1999087	317,161	377,561
27,28300477	322,097	376,663

Ahora estamos en disposición de hacer una representación utilizando los intervalos de confianza lo que nos permite dar mayor rigor cientifico al promedio que estamos representando.

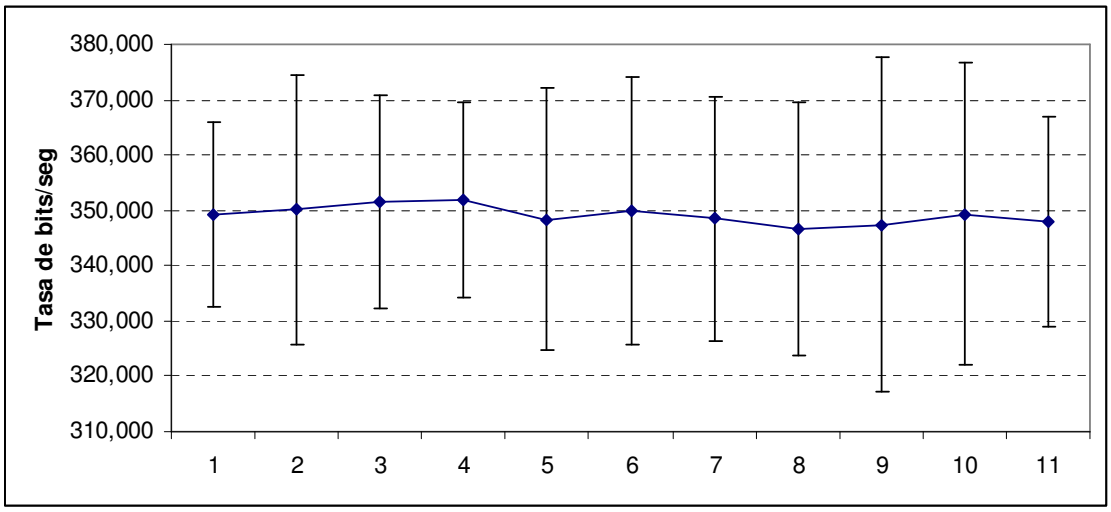

Figura A1.2. Gráfica de la tasa de bits/seg intervalo de confianza

En la *Fig. A1.2.* se puede ver el gráfico del promedio de las tasas de bits en las 3 ejecuciones. Para cada punto vemos el intervalo de confianza al 99%.

Como se comentó anteriormente, se ha aplicado este método en todas las gráficas que se muestran en el proyecto. Se ha fijado un intervalo de confianza de un 95% y el número de muestras ha varíado de 4 a 6 en función del conjunto de gráficas.

ANEXO II: Ficheros .tcl usados para las simulaciones con ns-2

Ficheros usados para las simulaciones usando el protocolo de enrutamiento DSR

```
# Simulaciones con protocolo de enrutamiento DSR
set opt(chan)           Channel/WirelessChannel    ;# channel type
set opt(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set opt(netif)          Phy/WirelessPhy            ;# network interface type
set opt(mac)            Mac/802_11                 ;# MAC type
set opt(ifq)            CMUPriQueue
set opt(ll)             LL                         ;# link layer type
set opt(ant)            Antenna/OmniAntenna        ;# antenna model
set opt(CBRpsize)       1500 ;# Packet size
set opt(CBRrate)        3000000 ;# CBR rate in bps.
set NumofSources        1 ;
```

```
set num_nodes               50 ;
set opt(NumOfMN)        $num_nodes ;# number of mobilenodes
set opt(RoutingProtocol) DSR ;# routing protocol
# DSR has 4 different physical queues: queue size set in dsr-priqueue.h

set opt(X)                  420  ;# X & Y dimension of the topography
set opt(Y)                  420
set opt(MNcoverage)     120
set opt(speed)          2
set opt(start)          0
set opt(stop)           100

Mac/802_11 set SlotTime_          0.000020        ;# 20us
Mac/802_11 set SIFS_              0.000010        ;# 10us
Mac/802_11 set PreambleLength_    144             ;# 144 bit (Long version) 72 (short version)
Mac/802_11 set PLCPHeaderLength_  48              ;# 48 bits 24 bits
Mac/802_11 set PLCPDataRate_      11.0e6          ;# (1Mbps)
Mac/802_11 set dataRate_          11.0e6          ;# 11Mbps
Mac/802_11 set basicRate_         11.0e6          ;# (1Mbps)

# Computes the necessary power fot a given coverage (the cell radius)
proc SetPt { coverage } {
  set Gt [Antenna/OmniAntenna set Gt_]
  set Gr [Antenna/OmniAntenna set Gr_]
  set ht [Antenna/OmniAntenna set Z_]
  set hr [Antenna/OmniAntenna set Z_]
  set RXThresh [Phy/WirelessPhy set RXThresh_]
  set d4 [expr pow($coverage,4)]
  set Pt [expr ($RXThresh*$d4)/($Gt*$Gr*$ht*$ht*$hr*$hr)]
  return $Pt
}
set ns [new Simulator]
set trace [open "tr_dsrgra4.tr" w]
$ns trace-all $trace
$ns use-newtrace
set namtrace [open animation.nam w]
$ns namtrace-all-wireless $namtrace $opt(X) $opt(Y)
```

```
set topo [new Topography]
$topo load_flatgrid $opt(X) $opt(Y)
set god [create-god $opt(NumOfMN)]

# unity gain, omni-directional antennas
# set up the antennas to be centered in the node and 1.5 meters above it
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1
Antenna/OmniAntenna set Gr_ 1

set ReceptionThreshold 1.77827941e-13
#                      3.87827941e-12


# ======================================================
# Initialize the SharedMedia interface with parameters to make
# it work like the 5004 MP Atheros 4G miniPCI card
# ======================================================
Phy/WirelessPhy set CPThresh_   10.0
Phy/WirelessPhy set CSThresh_   2.0e-14
#Phy/WirelessPhy set RXThresh_   1.77827941e-13                ;#-97.5 dBm
Phy/WirelessPhy set RXThresh_   $ReceptionThreshold            ;
Phy/WirelessPhy set freq_       2.472e9                        ;# Frequency of work
Phy/WirelessPhy set L_          1                              ;# System Loss Factor
Phy/WirelessPhy set bandwidth_ 11Mb
Phy/WirelessPhy set Pt_ [SetPt $opt(MNcoverage)]

puts "pt _ $[SetPt $opt(MNcoverage)]"
puts "$[SetPt $opt(MNcoverage)]"

set chan_1_ [new $opt(chan)]
$ns node-config \
   -mobileIP OFF \
   -adhocRouting $opt(RoutingProtocol) \
   -llType $opt(ll) \
```

```
    -macType $opt(mac) \
    -ifqType $opt(ifq) \
    -ifqLen $opt(ifqlen) \
    -antType $opt(ant) \
    -propType $opt(prop) \
    -phyType $opt(netif) \
    -channel $chan_1_ \
    -topoInstance $topo \
    -agentTrace ON \
    -routerTrace ON \
    -macTrace OFF \
    -movementTrace ON
# Connets upper and lower MNs:
Agent/Null set sport_               0
Agent/Null set dport_               0
Agent/CBR set sport_                0
Agent/CBR set dport_                0
set opt(CBRperiod)  [expr 8.0 * double($opt(CBRpsize)+20) / double($opt(CBRrate))]
$ns color 1 Blue
$ns color 2 Green
$ns color 3 Red
$ns color 4 Yellow
Agent/RTP_v2 set sport_                     0
Agent/RTP_v2 set dport_                     0
Agent/RTCP_v2 set sport_        0
Agent/RTCP_v2 set dport_        0
set rng [new RNG]
set opt(seed) 0.1
$rng seed $opt(seed)

for {set i 0} {$i < $opt(NumOfMN)} {incr i} {
            set node_($i) [$ns node]
}
puts "Loading scenario file..."
set opt(sc) "scens/scenarioGRA4.scen"
source $opt(sc)
puts "Load complete..."
```

```
for {set i 0} {$i < $opt(NumOfMN)} {incr i} {
    $node_($i) set RTPsource_ 0
    $node_($i) set RTPdestination_ 0
    $node_($i) set Pt_ [SetPt $opt(MNcoverage)]
    $node_($i) set MNcoverage_ $opt(MNcoverage)
}
set tmp 0.011
set start_tx_time 4
set selfs 0
for {set i 0} {$i < $NumofSources} {incr i} {
        set n [expr {$i + 24}]
        $node_($i) set RTPsource_ 1
        $node_($n) set RTPdestination_ 1
        $node_($i) set MMDSRfreq_ 15
        puts "RTPsource $i"
        puts "RTPdestination $n"

        set trace_file_($i) [new MpegDataFile_v2]
        $trace_file_($i) metafile blade2.mdipbf4444
        $trace_file_($i) datafile blade.m2v
        set rtp_($i) [new Agent/RTP_v2]
        $ns attach-agent $node_($i) $rtp_($i)
        $rtp_($i) set class_ 2
        $rtp_($i) set seqno_ 0
        $rtp_($i) set packetSize_ 1500
        $rtp_($i) set multipath_ 3
        $rtp_($i) set flow_id_ $i

        set rtcp_($n) [new Agent/RTCP_v2]
        $ns attach-agent $node_($n) $rtcp_($n)
        $rtcp_($n) set class_ 3
        $rtcp_($n) set interval_ 5000ms
        $rtcp_($n) set seqno_ 0
        $rtcp_($n) set flow_id_ $i

        set rtp_($n) [new Agent/RTP_v2]
```

```
$ns attach-agent $node_($n) $rtp_($n)
$rtp_($n) set class_ 2
$rtp_($n) set seqno_ 0
$rtp_($n) set packetSize_ 1500

# Variable for RTCP: 0-> not in use, 1-> in use
$rtp_($i) set rtcp_in_use_ 1

# Variable for RTP: 0: no losses scheme, 1: use the losses C++ code scheme
$rtp_($i) set losses_ 0

$ns connect $rtp_($i) $rtcp_($n)
$ns attach-agent $node_($n) $rtcp_($n)

set MPEG2_($i) [new Application/Traffic/Mpeg2Gen2]
$MPEG2_($i) attach-mpegdatasrc $trace_file_($i)
$MPEG2_($i) attach-agent $rtp_($i)

#0: without codification
#1: todo por el mismo camino
#2: I+B por camino 1, P por el 2
#3: I+P+B pares por camino 1, B impares por camino 2
#4: I / P / B, cada uno por un camino diferente (Multipath de 3) */
$MPEG2_($i) set codification_ 4
$rtp_($i) set codification_ 4

set sinkmpeg_($n) [new Application/Mpeg2RX2]
$sinkmpeg_($n) attach-agent $rtcp_($n)
$sinkmpeg_($n) attach-agent $rtp_($n)
$sinkmpeg_($n) dumpfile "$i.dsrgra4.m2v"

set self_($i) [new Session/RTP]
$rtcp_($n) session $self_($i)
$rtp_($i) session $self_($i)
$rtp_($n) session $self_($i)

set udp_($i) [new Agent/UDP]
```

```
        $ns attach-agent $node_($i) $udp_($i)
        set null_($n) [new Agent/LossMonitor]
        $ns attach-agent $node_($n) $null_($n)
        set cbr_($i) [new Application/Traffic/CBR]
        $cbr_($i) set packetSize_ $opt(CBRpsize)
        $cbr_($i) set interval_  $opt(CBRperiod)
        $cbr_($i) set random_ 1
        $cbr_($i) set CBRrate $opt(CBRrate)
        $cbr_($i) attach-agent $udp_($i)
        $ns connect $udp_($i) $null_($n)
        $udp_($i) set fid_ 4
        $ns at 1 "$cbr_($i) start"
        $ns at 100 "$cbr_($i) stop"

        $ns at 4.0 "$MPEG2_($i) start"
        $ns at 4.0 "$rtcp_($n) start"
        incr start_tx_time
        incr selfs
}
$ns at $opt(stop)    {finish}

for {set i 0} {$i < $opt(NumOfMN)} {incr i} {
   $ns initial_node_pos $node_($i) 15
}
proc finish {} {
   global ns node_ null_ nn2 opt trace namtrace namtrace_s sinkmpeg
#   puts "Finishing ns at time [$ns now]."
   $ns flush-trace
   close $trace
   exit 0
}
puts "Starting Simulation..."
puts "Número de nodos:$num_nodes"
$ns run
```

Fichero usado para simulaciones con protocolo de enrutamiento MMDSR

```
# Simulaciones de multicamino con MM-DSR
set opt(chan)           Channel/WirelessChannel    ;# channel type
set opt(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set opt(netif)          Phy/WirelessPhy            ;# network interface type
set opt(mac)            Mac/802_11                 ;# MAC type
set opt(ifq)            CMUPriQueue
set opt(ll)             LL                         ;# link layer type
set opt(ant)            Antenna/OmniAntenna        ;# antenna model
set opt(CBRpsize)       1500 ;# Packet size
set opt(CBRrate)        1000000 ;# CBR rate in bps.
set NumofSources        2 ;
set opt(ifqlen)         -1 ;# max packet in ifq, se corresponde con la queue size del MAC
#Queue set limit_  1000

set num_nodes             20 ;
set opt(NumOfMN)        $num_nodes ;# number of mobilenodes

set opt(RoutingProtocol) DSR ;# routing protocol
# DSR has 4 different physical queues: queue size set in dsr-priqueue.h

set opt(X)                220  ;# X & Y dimension of the topography
set opt(Y)                220
set opt(MNcoverage)     120
set opt(speed)          2
set opt(start)          0
set opt(stop)           95

Mac/802_11 set SlotTime_          0.000020         ;# 20us
Mac/802_11 set SIFS_              0.000010         ;# 10us
Mac/802_11 set PreambleLength_    144              ;# 144 bit (Long version) 72 (short version)
Mac/802_11 set PLCPHeaderLength_  48               ;# 48 bits 24 bits
Mac/802_11 set PLCPDataRate_      11.0e6           ;# (1Mbps)
Mac/802_11 set dataRate_          11.0e6           ;# 11Mbps
Mac/802_11 set basicRate_         11.0e6           ;# (1Mbps)



# Computes the necessary power fot a given coverage (the cell radius)
proc SetPt { coverage } {
  set Gt [Antenna/OmniAntenna set Gt_]
  set Gr [Antenna/OmniAntenna set Gr_]
  set ht [Antenna/OmniAntenna set Z_]
  set hr [Antenna/OmniAntenna set Z_]
  set RXThresh [Phy/WirelessPhy set RXThresh_]
  set d4 [expr pow($coverage,4)]
  set Pt [expr ($RXThresh*$d4)/($Gt*$Gr*$ht*$ht*$hr*$hr)]
  return $Pt
}

set ns [new Simulator]
set trace [open "tr_dsrgame3.tr" w]
$ns trace-all $trace
$ns use-newtrace
```

```
set namtrace [open animation.nam w]
$ns namtrace-all-wireless $namtrace $opt(X) $opt(Y)
set topo [new Topography]
$topo load_flatgrid $opt(X) $opt(Y)
set god [create-god $opt(NumOfMN)]

# unity gain, omni-directional antennas
# set up the antennas to be centered in the node and 1.5 meters above it
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1
Antenna/OmniAntenna set Gr_ 1

set ReceptionThreshold 1.77827941e-13
#                      3.87827941e-12


# ======================================================
# Initialize the SharedMedia interface with parameters to make
# it work like the 5004 MP Atheros 4G miniPCI card
# ======================================================
Phy/WirelessPhy set CPThresh_   10.0
Phy/WirelessPhy set CSThresh_   2.0e-14
#Phy/WirelessPhy set RXThresh_   1.77827941e-13                ;#-97.5 dBm
Phy/WirelessPhy set RXThresh_   $ReceptionThreshold            ;
Phy/WirelessPhy set freq_       2.472e9                        ;# Frequency of work
Phy/WirelessPhy set L_          1                              ;# System Loss Factor
Phy/WirelessPhy set bandwidth_ 11Mb
Phy/WirelessPhy set Pt_ [SetPt $opt(MNcoverage)]

puts "pt _ $[SetPt $opt(MNcoverage)]"
puts "$[SetPt $opt(MNcoverage)]"

set chan_1_ [new $opt(chan)]
$ns node-config \
   -mobileIP OFF \
   -adhocRouting $opt(RoutingProtocol) \
   -llType $opt(ll) \
   -macType $opt(mac) \
   -ifqType $opt(ifq) \
   -ifqLen $opt(ifqlen) \
   -antType $opt(ant) \
   -propType $opt(prop) \
   -phyType $opt(netif) \
   -channel $chan_1_ \
   -topoInstance $topo \
   -agentTrace ON \
   -routerTrace ON \
   -macTrace OFF \
   -movementTrace ON

# Connets upper and lower MNs:
Agent/Null set sport_               0
Agent/Null set dport_               0
Agent/CBR set sport_                0
```

```
Agent/CBR set dport_              0

set opt(CBRperiod)  [expr 8.0 * double($opt(CBRpsize)+20) / double($opt(CBRrate))]
$ns color 1 Blue
$ns color 2 Green
$ns color 3 Red
$ns color 4 Yellow

Agent/RTP_v2 set sport_                 0
Agent/RTP_v2 set dport_                 0
Agent/RTCP_v2 set sport_        0
Agent/RTCP_v2 set dport_        0

set rng [new RNG]
set opt(seed) 0.1
$rng seed $opt(seed)

for {set i 0} {$i < $opt(NumOfMN)} {incr i} {
        set node_($i) [$ns node]
}
puts "Loading scenario file..."
set opt(sc) "scens/scenarioPEQ3.scen"
source $opt(sc)
puts "Load complete..."


for {set i 0} {$i < $opt(NumOfMN)} {incr i} {
   $node_($i) set RTPsource_ 0
   $node_($i) set RTPdestination_ 0
   $node_($i) set Pt_ [SetPt $opt(MNcoverage)]
   $node_($i) set MNcoverage_ $opt(MNcoverage)
}
set tmp 0.011
set start_tx_time 4
set selfs 0

for {set i 0} {$i < $NumofSources} {incr i} {
        set n [expr {$i + 18}]
        $node_($i) set RTPsource_ 1
        $node_($n) set RTPdestination_ 1
        $node_($i) set MMDSRfreq_ 15
        puts "RTPsource $i"
        puts "RTPdestination $n"

        set trace_file_($i) [new MpegDataFile_v2]
        $trace_file_($i) metafile blade2.mdipbf4444
        $trace_file_($i) datafile blade.m2v


        set rtp_($i) [new Agent/RTP_v2]
        $ns attach-agent $node_($i) $rtp_($i)
        $rtp_($i) set class_ 2
        $rtp_($i) set seqno_ 0
        $rtp_($i) set packetSize_ 1500
        $rtp_($i) set multipath_ 3
        $rtp_($i) set flow_id_ $i
```

```
set rtcp_($n) [new Agent/RTCP_v2]
$ns attach-agent $node_($n) $rtcp_($n)
$rtcp_($n) set class_ 3
$rtcp_($n) set interval_ 5000ms
$rtcp_($n) set seqno_ 0
$rtcp_($n) set flow_id_ $i

set rtp_($n) [new Agent/RTP_v2]
$ns attach-agent $node_($n) $rtp_($n)
$rtp_($n) set class_ 2
$rtp_($n) set seqno_ 0
$rtp_($n) set packetSize_ 1500

# Variable for RTCP: 0-> not in use, 1-> in use
$rtp_($i) set rtcp_in_use_ 1

# Variable for RTP: 0: no losses scheme, 1: use the losses C++ code scheme
$rtp_($i) set losses_ 0

$ns connect $rtp_($i) $rtcp_($n)
$ns attach-agent $node_($n) $rtcp_($n)



set MPEG2_($i) [new Application/Traffic/Mpeg2Gen2]
$MPEG2_($i) attach-mpegdatasrc $trace_file_($i)
$MPEG2_($i) attach-agent $rtp_($i)

#0: without codification
#1: todo por el mismo camino
#2: I+B por camino 1, P por el 2
#3: I+P+B pares por camino 1, B impares por camino 2
#4: I / P / B, cada uno por un camino diferente (Multipath de 3) */
$MPEG2_($i) set codification_ 4
$rtp_($i) set codification_ 4

set sinkmpeg_($n) [new Application/Mpeg2RX2]
$sinkmpeg_($n) attach-agent $rtcp_($n)
$sinkmpeg_($n) attach-agent $rtp_($n)
$sinkmpeg_($n) dumpfile "$i.dsrgame3.m2v"

set self_($i) [new Session/RTP]
$rtcp_($n) session $self_($i)
$rtp_($i) session $self_($i)
$rtp_($n) session $self_($i)

set udp_($i) [new Agent/UDP]
$ns attach-agent $node_($i) $udp_($i)
set null_($n) [new Agent/LossMonitor]
$ns attach-agent $node_($n) $null_($n)
set cbr_($i) [new Application/Traffic/CBR]
$cbr_($i) set packetSize_ $opt(CBRpsize)
$cbr_($i) set interval_  $opt(CBRperiod)
$cbr_($i) set random_ 1
$cbr_($i) set CBRrate $opt(CBRrate)
```

```
        $cbr_($i) attach-agent $udp_($i)
        $ns connect $udp_($i) $null_($n)
        $udp_($i) set fid_ 4
        $ns at 1 "$cbr_($i) start"
        $ns at 100 "$cbr_($i) stop"

        $ns at 4.0 "$MPEG2_($i) start"
        $ns at 4.0 "$rtcp_($n) start"
        incr start_tx_time
        incr selfs
}

$ns at $opt(stop)    {finish}

for {set i 0} {$i < $opt(NumOfMN)} {incr i} {

   $ns initial_node_pos $node_($i) 15
}


proc finish {} {
   global ns node_ null_ nn2 opt trace namtrace namtrace_s sinkmpeg
#   puts "Finishing ns at time [$ns now]."
   $ns flush-trace
   close $trace
   exit 0
}

puts "Starting Simulation..."
puts "Número de nodos:$num_nodes"
$ns run
```

REFERENCIAS

[ABR96] ABR (Associativity-Based Routing protocol), Chai-Keong Toh: A Novel Distributed Routing Protocol To Support Ad hoc Mobile Computing, Proc. IEEE 15th Annual International Phoenix Conference on Computers and Communications, IEEE IPCCC 1996, March 27-29, Phoenix, AZ, USA, pp. 480-486.

[ADV00] ADV (*Adaptive Distance Vector routing*), R. V. Boppana, S. P. Konduru. An adaptive distance vector routing algorithm for mobile ad hoc networks, IEEE INFOCOM 2001 Nº 1, April 2001. pp. 1753-1762.

[AOD03] AODV (*Ad hoc On Demand Distance Vector routing protocol*), C. E. Perkins, E. M. Belding-Royer, S. Das, July 2003, Internet Draft: http://www.ietf.org/rfc3561.txt

[BNM05] Bonmotion v1.3.a, Julio 2005. Herramienta para la generación de escenarios con mobilidad y para su posterior análisis. http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/

[CAR09] Víctor Carrascal Frías, "Contribution to provide QoS Mobile Ad-hoc Networks for Video-Streaming Services based on Adaptative Cross-Layer Architecture", PhD Disertation, Advisor: Dra. Mónica Aguilar Igartua, 2nd March 2009, Departement of Telematic Engineering, Technical University of Calania, Barcelona, Spain. Avaible in http://sertel.upc.es/tesis.php.NS-2, contributed code avaible in http://globus.upc.es/vcarrascal/ns2/.

[CBR99] CBRP (Cluster Based Routing Protocol), M. JIANG, J. LI, Y. C. TAY, Cluster Based Routing Protocol (CBRP), Functional Specification Internet Draft, draft-ietf-manet-cbrp.txt, June 1999.

[CED99] CEDAR (*Core Extraction Distributed Ad hoc Routing*), Raghupathy Sivakumar, Prasun Sinha, Vaduvur Bharghavan, Core Extraction Distributed Ad hoc Routing (CEDAR) Specification, Internet Draft, draft-ietf-manet-cedar-spec-00.txt

[DIF98] DiffServ, RFC 2475 (1998)

[DOE08] Isabelle Doé de Maindreville, Proyecto fin de carrera (ETSETB-UPC), "Evaluación de prestaciones del protocolo MMDSR (Multipath Multimedia Dynamic Source Rounting) sobre redes móviles Ad-hoc", directora: Mónica Aguilar Igartua, Marzo 2008.

[DRE98] DREAM (*Distance Routing Effect Algorithm for Mobility*), S. Basagni, I. Chlamtac, V. R. Syrotiuk, B. A. Woodward, A Distance Routing Effect Algorithm for Mobility (DREAM), In Proc. ACM/IEEE Mobicom, pages 76-84, October 1998.

[DSD94] DSDV (*Destination-Sequenced Distance Vector*), C. E. Perkins, P. Bhagwat, Highly Dynamic Destination-Sequenced Distance Vector (DSDV) for Mobile Computers, Proc. Of the SIGCOMM 1994 Conference on Communications Architectures, Protocols and Applications, Aug 1994, pp. 234-244.

[DSR07] RFC-4728, The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4, February 2007, http://www.ietf.org/rfc/rfc4728.txt

[DYM05] DYMO (*Dynamic MANET On-demand routing protocol*), I. Chakeres, E. Belding-Royer, C. Perkin, Internet Draft: http://www.ietf.org/internet-drafts/draft-ietf-manet-dymo-06.txt, 2006.

[FQM00] FQMM (*Flexible QoS Model for MANET*), H. Xiao, Winston K. G. Seah, A. Lo, K. C. Chua, A Flexible Quality of Service Model for Mobile Ad Hoc Networks. In the proceedings of IEEE Vehicular Technology Conference, 15-18 May 2000, Tokyo, Japan, pp. 445-449. http://www1.i2r.a-star.edu.sg/~winston/papers/VTC2000Spring-FQMM.pdf

[HSR00] HSR (*Hierarchical State Routing protocol*), A. O'neill Hongy Li, Hierarchical State Routing protocol Internet Draft: http://www.ietf.org/internet-drafts/draft-oneill-li-hsr-00.txt, 2000.

[INT94] Integrated Services (IntServ), RFC 1633 (1994)

[LAR98] LAR (*Location-Aided Routing protocol*), Y. B. KO, V. N. H., Location-Aided Routing in mobile Ad hoc networks In Proc, ACM/IEEE Mobicom, pp. 66-75, October 1998.

[LEFE08] Antoni Leyva y Moris Ferrer, Proyecto final de carrera (ETSETB-UPC), "Evaluación de prestaciones de varios protocolos de encaminamiento en diferentes escenarios de redes Ad-hoc mediante simulador ns-2", directora: Mónica Aguilar Igartua, Abril 2008.

[MMW98] MMWN (*Mobile Multimedia Wireless Network*), M. Bergano, R.R. Hain, K. Kasera, D. Li, R. Ramanathan, M Steentrup, Technical report, DARPA project DAAB07-95-C-D156, October 1996.

[MPG] Moving Picture Experts Group. http://www.chiariglione.org/mpeg/

[MPL] Mplayer, the movie player for linux. GNU GPL2 *license*

[OLS03] OLSR (*Optimized Link State Routing Protocol*), P. Jacquet, P. Muhlethaler, A. Qayyum, A. Laouitti, L. Viennot, T. Clausen, Optimized Link State Routing Protocol, RFC 3626: http://www.ietf.org/rfc/rfc3626.txt , 2003.

[NS2] The Open Source Network Simulator, ns-2. http://nsnam.isi.edu/nsnam/index.php/Main_Page

[RSV97] RSVP, RFC 2205 (1997)

[RTP96] RTP, Real Time Protocol (1996), RFC 1889, http://www.ietf.org/rfc/rfc1889.txt

[SA] IEEE Standards Association. http://standards.ieee.org/

[SLU04] SLURP (*Scalable Location Update-based Routing Protocol*), Seung-chul M. Woo and Suresh Singh. ACM/KluwerWireless Networks (WINET) Journal, 7(5): 513-529, 2001.

[STA99] STAR (*Source Tree Adaptive routing protocol*), J. J. Garcia-Luna, M. Spohn, Source Tree Adaptive routing protocol, Internet Draft: http://www3.ietf.org/proceedings/99nov/I-D/draft-ietf-manet-star-00.txt , October 1999.

[TBR04] TBRPF (*Topology Broadcast based on Reverse-Path Forwarding routing* protocol), B. Bellur, R. G. OgieR, F. L. Templin, Topology Broadcast based on Reverse-Path Forwarding (TBRPF), RFC 3684: http://www.ietf.org/rfc/rfc3684.txt.

[RTC96] RTCP, Real Time Control Procotol (1996), RFC 1890, http://www.ietf.org/rfc/rfc1890.txt

[RTS96] RTSP, Real Time Streaming Protocol, http://www.cs.columbia.edu/~hgs/rtsp/

[TOR01] TORA (*Temporally-Ordered Routing Algorithm routing* protocol), V. Park, S. Corson, Temporally-Ordered Routing Algorithm (TORA) Version 1 Internet Draft: http://www3.ietf.org/proceedings/02mar/I-D/draft-ietf-manet-tora-spec-04.txt, June 2001.

[ZHL99] ZHLS (*Zone-Based Hierarchical Link State Routing*), Mario Joa NG, I-Tai Lu, Zone-Based Hierarchical Link State Routing (ZHLS). An abstract routing protocol and medium access protocol for mobile ad hoc networks, January 1999.

[ZRP02] ZRP (*Zone Routing Protocol*), Z. J. HAAS, M. R. Pearlman, P. Samar, The Zone Routing Protocol (ZRP) for Ad Hoc Networks, Internet Draft, http://www.ietf.org/proceedings/02nov/I-D/draft-ietf-manet-zone-zrp-04.txt, July 2002.

BIBLIOGRAFÍA RELACIONADA

[1] "Multicasting in MANET", Juan Vera del Campo, 30 de Junio 2006. http://globus.upc.es/~juanvvc/downloads/files/SIP-juanvi.pdf DEFINICIÓN

[2] "Caracterización de calidad de servicios en redes inalámbricas de sensores", Oscar Ortiz Ortiz. http://www.diatel.upm.es/oortiz/Doctorado/Trabajo%20de%20Investigaci%C3%B3n%20Oscar%20Ortiz.pdf

[3] "Quality of Service in Mobile Ad-hoc Networks – Myth or Reality?", Winston Seah. http://www1.i2r.a-star.edu.sg/~winston/papers/ATNAC2004%20Keynote.pdf

[4] "Propuesta de arquitectura QoS en entorno inalámbrico 802.11e basado en Diffserv con ajuste dinámico de parámetros", Carlos García García. http://www.it.uc3m.es/cgarcia/articulos/tesis-carlos-garcia-15jun.pdf

[5] "A Glance at Quality of Services in Mobile Ad-hoc Networks", Zeinalipour-Yazti Demetrios, Department of Computer Science, University of California. http://www.cs.ucr.edu/~csyiazti/courses/cs260/manetqos.pdf

[6] "Quality of Service Provision in Mobile Ad-hoc Networks (MANET)", Aura Ganz, Electrical and Computer Engineering Department, University of Massachusetts, http://www.engin.umd.umich.edu/vi/w1_workshops/manet_ganz_w1.pdf

[7] Estudio sobre Intervalos de confianza. Universidad Católica de Chile http://escuela.med.puc.cl/recursos/recepidem/EPIANAL9.HTM#1

[8] "Image Quality Computation", University of Berckley http://bmrc.berkeley.edu/courseware/cs294/fall97/assignment/psnr.html

[9] "Learning the bash Shell: Unix Shell Programming", Cameron Newham