UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Enginyeria Telemàtica

Contribución al diseño de protocolos de encaminamiento en redes móviles ad-hoc basados en reputación

Autor:

Lenin Guaya Delgado

Director:

Dr. Esteve Pallarès Segarra

Tesis presentada en cumplimiento parcial de los requisitos para el grado de Doctor en Filosofía en Ingeniería Telemática

en el

Departamento de Ingeniería Telemática

Barcelona, Junio 2019

A mis padres y hermanos por su ejemplo de fortaleza

“El hombre nace libre, pero en todos lados está encadenado”

— Jean-Jacques Rousseau

Agradecimientos

A todos los miembros de la familia Guaya y de la familia Delgado, por su valioso apoyo y motivación en esta importante etapa de mi vida.

En el Departamento de Ingeniería Telemática (ENTEL), a Esteve Pallarès Segarra, Jordi Forné Muñoz, Mónica Aguilar Igartua y Ahmad Mohammed Mezher por su aporte científico a la presente tesis.

Finalmente, mi eterno agradecimiento a la Secretaría Nacional de Educación Superior, Ciencia, Tecnología e Innovación (SENESCYT) de la República de Ecuador por seleccionarme para cumplir tan noble misión en Barcelona-España.

Abstract

The mobile ad-hoc networks are part of the next generation access networks (Next Generation Networking o NGN), "all-IP" networks facing the great scientific and technical challenge that is the ubiquitous computing

The mobile ad-hoc networks are useful in situations where rapid deployment is required in an area where there is no infrastructure. The nodes can act as sources, destinations or as routers to transmit data from one node to another node not accessible with a single jump

In this thesis we present two new algorithms of reputation-based routing protocols, REMODE_sw (Reputation for Mobile Devices Static Weight) and DrepSR (Dynamic reputation-based Source Routing), which use DSR (Dynamic Source Routing) as a routing protocol engine. The REMODE_sw protocol (Reputation for Mobile Devices Static Weight) is static weight and DrepSR (Dynamic reputation-based Source Routing) is dynamic weight, the difference is in the behavior of beta, which is the weight that reputation values (of each network node) new have in relation to historical values, REMODE_sw (Reputation for Mobile Devices Static Weight) maintains a constant beta value, while DrepSR (Dynamic reputation-based Source Routing) based on the behavior of the variance (of the reputation of each node in the network), independently updates the beta values of each node in the network, achieving that the reputation values(of each node in the network) approximate their average value with greater speed, in order that the reputation system has more accurate information on the behavior of each node

The two new algorithms estimate the RFVs (Reputation Forwarding Values) in an ad-hoc mobile network, the main thing is to be able to choose those paths that are formed by the nodes with greater reputation, thus avoiding the less collaborative nodes. Reputation estimation is done using only first-hand information available to the node based on its experience in the network.

For REMODE_sw (Reputation for Mobile Devices Static Weight) we have made the implementation of the algorithm on NS-2.35 (Network Simulator Version 2.35) using

scenarios created with Bonnmotion v3.0, while for DrepSR (Dynamic reputation-based Source Routing) we have developed a simulator of an ad-hoc mobile network using MATLAB R2017b, inside this simulator we have implemented the algorithm DrepSR (Dynamic reputation-based Source Routing), likewise this simulation scenarios are created using Bonnmotionv3.0.

As a general conclusion we have verified that the two new algorithms truly identify the most cooperative paths, which implies an improvement of the total packet loss fraction (average of all the paths of the network), since that allows to avoid the selfish nodes of the network.

Resumen

Las redes móviles ad-hoc son parte de las redes de acceso de próxima generación (Next Generation Networking o NGN), redes “all-IP” que hacen frente al gran desafío científico y técnico que es la computación ubicua

Las redes móviles ad-hoc son útiles en situaciones donde se requiere un rápido despliegue en una zona en la que no hay infraestructura. Los nodos pueden actuar como fuentes, destinos o como encaminadores para transmitir datos de un nodo a otro nodo no accesible con un único salto

En esta tesis presentamos dos nuevos algoritmos de protocolos de encaminamiento basados en reputación, REMODE_sw (Reputation for Mobile Devices Static Weight) y DrepSR (Dynamic reputation-based Source Routing), los cuales utilizan DSR (Dynamic Source Routing) como motor de protocolo de encaminamiento. El protocolo REMODE_sw (Reputation for Mobile Devices Static Weight) es de peso estático y DrepSR (Dynamic reputation-based Source Routing) es de peso dinámico, la diferencia se encuentra en el comportamiento de beta, el cual es el peso que los valores de reputación (de cada nodo de la red) nuevos tienen en relación a los valores históricos, REMODE_sw (Reputation for Mobile Devices Static Weight) mantiene un valor de beta constante, mientras que DrepSR (Dynamic reputation-based Source Routing) en base al comportamiento de la varianza (de la reputación de cada nodo de la red), actualiza de forma independiente los valores de beta de cada nodo de la red, consiguiendo que los valores de reputación (de cada nodo de la red) se aproximen a su valor medio con mayor rapidez, con la finalidad de que el sistema de reputación cuente con información más precisa sobre el comportamiento de cada nodo

Los dos nuevos algoritmos estiman las RFVs (Reputation Forwarding Values) en una red móvil ad-hoc, lo primordial es poder elegir aquellos caminos que son formados por los nodos con mayor reputación, evitando así los nodos menos colaborativos. La estimación de la reputación se realiza utilizando únicamente información de primera mano disponible para el nodo en función de su experiencia en la red

Para REMODE_sw (Reputation for Mobile Devices Static Weight) hemos realizado la implementación del algoritmo sobre NS-2.35 (Network Simulator Version 2.35) utilizando escenarios creados con Bonnmotion v3.0, mientras que para DrepSR (Dynamic reputation-based Source Routing) hemos desarrollado un simulador de una red móvil ad-hoc utilizando MATLAB R2017b, dentro de dicho simulador hemos implementado el algoritmo DrepSR (Dynamic reputation-based Source Routing), igualmente estos escenarios de simulación se crean utilizando Bonnmotion v3.0

Como conclusión general hemos podido comprobar que los dos nuevos algoritmos verdaderamente identifican los caminos más cooperativos, lo que supone una mejora de la fracción de pérdida de paquetes total (promedio de todos los caminos de la red), ya que permite evitar los nodos egoístas de la red

Resum

Les xarxes mòbils ad-hoc són part de les xarxes d'accés de pròxima generació (Next Generation Networking o NGN), xarxes "all-IP" que fan front al gran desafiament científic i tècnic que és la computació ubiqua

Les xarxes mòbils ad-hoc són útils en situacions on es requereix un ràpid desplegament en una zona en la qual no hi ha infraestructura. Els nodes poden actuar com a fonts, destinacions o com encaminaments per transmetre dades d'un node a un altre node no accessible amb un únic salt

En aquesta tesi presentem dos nous algoritmes de protocols d'encaminament basats en reputació, REMODE_sw (Reputation for Mobile Devices Static Weight) i DrepSR (Dynamic reputation-based Source Routing), els quals utilitzen DSR (Dynamic Source Routing) com a motor de protocol d'encaminament. El protocol REMODE_sw (Reputation for Mobile Devices Static Weight) és de pes estàtic i DrepSR (Dynamic reputation-based Source Routing) és de pes dinàmic, la diferència es troba en el comportament de beta, el qual és el pes que els valors de reputació (de cada node de la xarxa) nous tenen en relació als valors històrics, REMODE_sw (Reputation for Mobile Devices Static Weight) manté un valor de beta constant, mentre que DrepSR (Dynamic reputation-based Source Routing) sobre la base del comportament de la variància (de la reputació de cada node de la xarxa), s'actualitza de forma independent els valors de beta de cada node de la xarxa, aconseguint que els valors de reputació (de cada node de la xarxa) s'aproximin al seu valor mitjà amb major rapidesa, amb la finalitat que el sistema de reputació compti amb informació més precisa sobre el comportament de cada node

Els dos nous algoritmes estimen les RFVs (Reputation Forwarding Values) en una xarxa mòbil ad-hoc, el primordial és poder triar aquells camins que són formats pels nodes amb més reputació, evitant així els nodes menys col·laboratius. L'estimació de la reputació es realitza utilitzant únicament informació de primera mà disponible per al node en funció de la seva experiència a la xarxa

Per REMODE_sw (Reputation for Mobile Devices Static Weight) hem realitzat la implementació de l'algoritme sobre NS-2.35 (Network Simulator Version 2.35) utilitzant

escenaris creats amb Bonnmotion v3.0, mentre que per DrepSR (Dynamic reputation-based Source Routing) hem desenvolupat un simulador d'una xarxa mòbil ad-hoc utilitzant MATLAB R2017b, dins d'aquest simulador hem implementat l'algoritme DrepSR (Dynamic reputation-based Source Routing), igualment en aquest simulador s'han utilitzant escenaris creats en Bonnmotion v3.0

Com a conclusió general hem pogut comprovar que els dos nous algoritmes veritablement identifiquen els camins més cooperatius, fet que suposa una millora de la fracció de pèrdua de paquets total (mitjana de tots els camins de la xarxa), ja que permet evitar els nodes egoistes de la xarxa

Contenidos

Lista de figuras ..... **xv**
Lista de tablas ..... **xix**
Lista de algoritmos ..... **xxi**
Glosario de acrónimos ..... **xxii**

1 Introducción **1**
1.1 Motivación **2**
1.2 Objetivos de la tesis **6**
1.3 Organización de la tesis **7**

2 Redes móviles ad-hoc **9**
2.1 Introducción **10**
2.2 Tecnología empleada **12**
2.2.1 Redes IEEE 802.11 12
2.2.2 Problema de la terminal oculta y expuesta 17
2.2.3 Descripción técnica de IEEE 802.11g 22
2.3 Clasificación y aplicaciones de las redes ad-hoc inalámbricas **24**
2.3.1 Introducción 24
2.3.2 Historia y aplicaciones de las redes móviles ad-hoc 24
2.3.3 Clasificación de las redes ad-hoc 25
2.4 Protocolos de encaminamiento en redes móviles ad-hoc **32**
2.4.1 Conceptos generales sobre protocolos de encaminamiento en redes móviles ad-hoc 32
2.4.2 Protocolos de encaminamiento proactivos 34
2.4.3 Protocolos de encaminamiento reactivos 37
2.4.4 Protocolos de encaminamiento híbridos 39
2.5 Protocolo DSR (*Dynamic Source Routing*) para redes móviles ad-hoc **41**
2.5.1 Descubrimiento de caminos 41
2.5.2 Almacenamiento de caminos en memoria caché 44
2.5.3 Envío de paquetes de datos 45
2.5.4 Mantenimiento de caminos 45
2.5.5 Mecanismos adicionales 48
2.5.6 Implementación de DSR (*Dynamic Source Routing*) en NS-2.35 (*Network Simulator* Versión 2.35) 49
2.6 Sistemas de reputación en redes móviles ad-hoc **52**
2.6.1 Definición de comportamiento anómalo de un nodo 53
2.6.2 Protocolos de encaminamiento que utilizan sistemas de reputación 56
2.6.3 Principales componentes de los sistemas de reputación 57

2.6.4 Componentes presentes en REMODE_sw (Reputation for Mobile Devices Static Weight) y DrepSR (Dynamic Reputation Source Routing) 63
2.6.5 Comparativa de diversos protocolos de encaminamiento basados en sistemas de reputación 64
2.7 Propuesta de algoritmo de protocolo de encaminamiento REMODE_sw (*Reputation for Mobile Devices Static Weight*) basado en la reputación de los nodos 65
2.8 Protocolos utilizados para el diseño de cruce de capa 66
2.8.1 Definición de diseño de cruce de capa 66
2.8.2 RTP/RTCP (*Real-time Transport Protocol/Real-time Transport Control Protocol*) 66
2.8.3 TCP-Vegas (*Transmission Control Protocol* Versión Vegas) 66
2.8.4 TCP-Linux Vegas (TCP Vegas Versión LINUX) 68
2.9 Métricas de rendimiento de red utilizadas 68
2.9.1 FPL (*Fraction of Packet Losses*) 68
2.9.2 Retardo medio que sufren los paquetes 69
2.9.3 Número medio de saltos realizados por los paquetes 69
2.10 Herramientas de simulación utilizadas 70
2.10.1 NS-2.35 (*Network Simulator* Versión 2.35) 70
2.10.2 Bonnmotion v3.0.0 71
2.10.3 Matlab R2017b 72
2.11 Conclusiones 72

3 REMODE_sw (*Reputation for Mobile Devices Static Weight*) 76
3.1 Introducción 76
3.2 Evolución de la métrica de reputación de RDSR-V (*Reliable Dynamic Source Routing for video-streaming*) 76
3.3 Descripción matemática de REMODE_sw 77
3.4 Pseudocódigo para REMODE_sw 80
3.4.1 Recolección de información 80
3.4.2 Modelado de la información 81
3.4.3 Toma de decisiones 82
3.5 Resultados en NS-2.35 y análisis 84
3.5.1 Análisis de los valores de reputación al variar los valores de los pesos y en escenario estático 84
3.5.2 Análisis de reputación muestreada, reputación de nodos y calificación para caminos al incrementar las pérdidas de los nodos no cooperativos en escenario estático 88
3.5.3 Análisis del contenido de la memoria caché primaria del nodo fuente al existir múltiples caminos en escenario estático 90
3.5.4 Análisis de la reputación de los caminos activos, de la reputación de nodos y calificación de caminos en escenario móvil 92

3.5.5 Análisis de los valores de reputación de nodos cuando se tiene tráficos asimétricos ... 96
3.5.6 Análisis de la recuperación de la reputación de un nodo en un escenario móvil 97
3.6 Conclusiones ... 98

4 DrepSR (*Dynamic reputation-based Source Routing*) ... 102
4.1 Introducción ... 103
4.1.1 Trabajos relacionados ... 107
4.2 Protocolo de encaminamiento propuesto basado en reputación ... 110
4.2.1 Algoritmo para calcular la reputación de nodos y caminos ... 112
4.3 Algoritmo para promediar los valores de reputación de reenvío ... 116
4.3.1 Media móvil acumulada ... 117
4.3.2 Media móvil exponencial ... 119
4.3.3 Detección de ventanas de tiempo estacionaria ... 122
4.3.4 Algoritmo de promedio propuesto ... 125
4.4 Resultados Experimentales y Discusión ... 127
4.4.1 Validación del modelo teórico ... 127
4.4.2 Evaluación del desempeño del protocolo de encaminamiento propuesto ... 136
4.5 Conclusiones ... 138

5 Conclusiones y trabajo futuro ... 141
5.1 Conclusiones ... 141
5.2 Artículos de investigación publicados como resultado del trabajo de tesis 144
5.3 Trabajo futuro ... 144

Anexos ... 146
A Formato de cabecera de opciones DSR ... 146
A.1 Cabecera de opciones DSR (Dynamic Source Routing) ... 147
A.2 Opción de petición de camino ... 148
A.3 Opción de contestación de camino ... 150
A.4 Opción de camino fuente de DSR (*Dynamic Source Routing*) ... 152
A.5 Opción de error de camino ... 154
A.6 Opción de petición de acuse de recibo ... 156
A.7 Opción de acuse de recibo ... 157
B Manual de usuario ... 157
B.1 Introducción ... 158
B.2 Integrando REMODE_sw (Reputation for Mobile Devices Static Weight) a NS-2.35 (Network Simulator Versión 2.35) ... 158
B.3 Generando escenarios con Bonnmotionv3.0.0 ... 160
B.4 Creando nodos egoístas ... 163
B.5 Entendiendo los selectores de DSR que se utilizan en REMODE_sw. ... 166
B.6 Configuraciones de capa física. ... 171

B.7 Creación y ejecución de un script OTCL (*Object Oriented Tool Command Language*) con selectores adicionales para las configuraciones de REMODE_sw (*Reputation for Mobile Devices Static Weight*). 175
B.8 Entendiendo los archivos resultantes de la ejecución de la simulación 193
C Notación y símbolos 197
C.1 Notación y símbolos del Capítulo 4 197
D Código fuente para NS-2.35 198
D.1 Analizador de traza sobre cmu-trace.cc en NS-2.35 198
D.2 Más código fuente 215

Referencias 216

Lista de figuras

Figura 2.1: Redes móviles ad-hoc inalámbricas

Figura 2.2: Problema de la terminal oculta

Figura 2.3: (a) El problema del terminal oculto, y (b) el problema del terminal expuesto. Los círculos punteados representan el radio de cobertura de los transmisores

Figura 2.4: Mecanismo de acceso

Figura 2.5: Wi-Fi trabaja en modo de transmisión half-duplex

Figura 2.6: Red de sensores inalámbricos

Figura 2.7: Red de malla inalámbrica

Figura 2.8: Red móvil ad-hoc: (a) formada por un conjunto de laptops, (b) formada por laptops y teléfonos inteligentes, donde se compara el camino seleccionado por DSR, con el seleccionado por RDSR-V

Figura 2.9: Red vehicular ad-hoc

Figura 2.10: Arquitectura de referencia C2C-CC

Figura 2.11: Clasificación y ejemplos de protocolos de encaminamiento ad-hoc

Figura 2.12: Descubrimiento de camino (1-6) utilizando caminos desde memoria cache (4)

Figura 2.13: Ejemplo de mantenimiento de camino (el nodo C no está disponible para reenviar un paquete desde A hasta E sobre su enlace al siguiente salto, D)

Figura 2.14: Mecanismo de acortamiento automático de camino (ARS)

Figura 2.15: Mecanismo de Estado de Flujo (FSM)

Figura 2.16: Diagrama de herencia para un paquete DSR

Figura 2.17: Mal comportamiento del nodo

Figura 2.18: Sistemas de reputación en redes móviles ad-hoc.

Figura 2.19: Componentes de los sistemas de reputación

Figura 2.20: Vista simplificada de usuario NS2

Figura 3.1: Nodos que componen un camino

Figura 3.2: Diseño del escenario de simulación

Figura 3.3: $\overline{RFVp_\iota}$ variando α con β=0,05

Figura 3.4: $\overline{RFVn_6}$ variando β con α =0,25

Figura 3.5: $\overline{RFVn_6}$ variando β con α=0,05

Figura 3.6: Complemento de la FPL muestreada por cada camino

Figura 3.7: Calificaciones para caminos 5 y 5' calculadas con REMODE_sw

Figura 3.8: Escenario con patrón de movilidad *Random Waypoint* a 2 m/s

Figura 3.9: Reputación de los caminos activos

Figura 3.10: SCORE del camino activo observado por el nodo 0

Figura 3.11: SCORE para los caminos activos observados por el nodo 5

Figura 3.12: RFV_{n_7} recuperando su valor verdadero

Figura 4.1: Promedio de las muestras de reputación de un nodo si utilizamos una EWMA cuando el nodo cambia su comportamiento definiendo dos ventanas estacionarias.

Figura 4.2: Gráfica del factor que relaciona la varianza de las muestras σ_r^2 y la varianza de la segunda diferencia de la media móvil exponencial de las muestras σ_d^2, como una función de β para diferente número de muestras k (ver Ec. 4.22).

Figura 4.3: Topología de red utilizada en la simulación con un nodo fuente (S) y tres destinos (D_1, D_2 y D_3). En la simulación, todos los nodos reenvían el 100% de los paquetes, excepto el nodo 2 que reenvía solo un 70% de los paquetes durante la primera mitad de la simulación. Posteriormente, el nodo cambia su comportamiento y reenvía un 90% de los paquetes durante la última mitad de la simulación.

Figura 4.4: Valor promedio de las muestras (figuras en la parte superior) y detección de intervalos (figuras siguientes) cuando $\beta = 0.01$ y $F = 2$: (a) para el nodo 1 y (b) para el nodo 4.

Figura 4.5: Valor promedio de las muestras (figuras en la parte superior) y detección de intervalos (figuras siguientes) cuando $\beta = 0.1$ y $F = 2$: (a) para nodo 1 y (b) para nodo 4.

Figura 4.6: Valor promedio de las muestras (figuras en la parte superior) y detección de intervalos (figuras siguientes) cuando $\beta = 0.3$ y $F = 2$: (a) para el nodo 1 y (b) para el nodo 4.

Figura 4.7: Valor promedio de las muestras (figuras en la parte superior) y detección de intervalos (figuras siguientes) cuando $\beta = 0.7$ y $F = 2$: (a) para el nodo 1 y (b) para el nodo 4.

Figura 4.8: Valor promedio de las muestras (figuras en la parte superior) y detección de intervalos (figuras siguientes) cuando $\beta = 0.9$ y $F = 2$: (a) para nodo 1 y (b) para nodo 4.

Figura 4.9: Valor promedio de las muestras (figuras en la parte superior) y detección de intervalos (figuras siguientes) cuando $\beta = 0.1$ y $F = 1$: (a) para el nodo 1 y (b) para el nodo 4

Figura 4.10: Valor promedio de las muestras (figuras en la parte superior) y detección de intervalos (figuras siguientes) cuando $\beta = 0.1$ y $F = 3$: (a) para el nodo 1 y (b) para el nodo 4.

Figura 4.11: Comparación de los resultados simulados para el protocolo DSR y el protocolo de encaminamiento DrepSR propuesto. (a) Probabilidad de pérdida de

paquetes; (b) Número promedio de saltos. Las figuras muestran un intervalo de confianza del 95% utilizando 10 realizaciones independientes por punto.

Figura A.1: Parte fija del encabezado de opciones DSR

Figura A.2: Opción de petición de camino

Figura A.3: Opción de contestación de camino

Figura A.4: Opción de camino fuente de DSR

Figura A.5: Opción de error de camino

Figura A.6: Opción de petición de acuse de recibo

Figura A.7: Opción de acuse de recibo

Lista de tablas

Tabla 2.1: IEEE 802.11 estándares de la capa física

Tabla 2.2: Clases básicas en el paquete DSR

Tabla 2.3: Archivos que conforman DSR

Tabla 3.1: Parámetros de la simulación

Tabla 3.2: Reputación de nodos observada por el nodo origen cuando se incrementa la FPL

Tabla 3.3: Calificaciones de caminos observada por el nodo origen cuando se incrementa la FPL

Tabla 3.4: Caché primaria del nodo origen 8 cuando tenemos múltiples caminos

Tabla 3.5: Reputación observada por el nodo origen 0 cuando los nodos están en movimiento

Tabla 3.6: Reputación observada por el nodo origen 5 cuando los nodos están en movimiento

Tabla 3.7: Reputación de nodos observada por el nodo origen 5 con $\gamma_6 = \gamma_5$

Tabla 3.8: Reputación de nodos observada por el nodo origen 5 con $\gamma_6 = \frac{1}{2} * \gamma_5$

Tabla 3.9: Reputación de nodos observada por el nodo origen 5 con $\gamma_6 = 2 * \gamma_5$

Tabla 4.1: Ajustes de simulación

Tabla 4.2: Resultados de la simulación

Tabla C.1: Notación y símbolos del Capítulo 4

Lista de algoritmos

Algoritmo 3.1: Actualización de valores de reputación

Algoritmo 3.2: Calculando la calificación del camino

Algoritmo 3.3: Seleccionando camino de mejor calificación

Algoritmo 4.1: Actualiza la reputación estimada de los nodos

Algoritmo 4.2: Estimación de la reputación del camino

Algoritmo 4.3: Encaminamiento

Algoritmo 4.4: Promedio

Glosario de acrónimos

4G	Fourth Generation of Broadband Cellular Network Technology
5G	*5th Generation Mobile Networks or 5th Generation Wireless System*
ACK	*Acknowledge*
ADSL	*Asymmetric Digital Subscriber Line*
AODV	*Ad-Hoc On-Demand Distance-Vector Routing*
AU	*Application Unit*
ARS	*Automatic Route Shortening*
C2C-CC	*Car-to-Car Communication Consortium*
CBQ	*Class-Based Queueing*
CMA	*Cumulative Moving Average* (Media Móvil Acumulada)
CONFIDANT	*COoperation of Nodes-Fairness In Dynamic Ad-hoc NeT-works*
COOJA	*Contiki Network Simulator*
CORE	*COllaborative REputation mechanism*
CSMA/CA	*Carrier-Sense Multiple Access with Collision Avoidance*
CTS	*Clear To Send*
CWND	*Congestion Windows*
DARPA	*Defense Advanced Research Projects Agency*
DIFS	*Distributed IFS*
DLL	*Data Link Layer*

DoD	*United States Department of Defense*
DRBTS	*Distributed Reputation-Based beacon Trust System*
DrepSR	*Dynamic reputation Source Routing*
DSR	*Dynamic Source Routing*
DSSS	*Direct Sequence Spread Spectrum*
DSDV	*Destination-Sequenced Distance-Vector Routing*
D2D	*Device-to-Device Communication*
EBA	*Exponential Backoff Algorithm*
ECC	*Error-Correcting Code*
eNodeB	*enhanced Node B*
EWMA	*Exponentially Weighted Moving Average (Media Móvil Exponencial)*
FANETs	*Flying Ad-Hoc Networks*
FEC	*Forward Error Correction*
FHSS	*Frequency Hopping Spread Spectrum*
Flow_ID	*Flow Identifier*
FPL_total	*Fraction of Packet Losses_total*
FreeBSD	*Free Berkeley Software Distribution Operating System*
FSLS	*Fuzzy Sighted Link-State Algorithm*
FSM	*Flow State Mechanism*
FSR	*Fisheye State Routing*
FTP	*File Transfer Protocol*
GloMo	*Global Mobile Information System*
GloMoSim	*Global Mobile Information System Simulate*

GPRS	*General Packet Radio Service*
GRR	*Gratuitous Route Reply*
GSM	*Global System for Mobile Communications*
HFC	*Hybrid Fiber Coaxial*
IEEE 802.11g	*Institute of Electrical and Electronics Engineers 802.11g*
IETF	*Internet Engineering Task Force*
IFS	*Inter Frame Space*
IoT	*Internet of Thing*
IP	*Internet Protocol*
IR	*Infrared*
ISM	*Industrial Scientific Medical*
KeyNote	*The KeyNote Trust Management System Version 2*
LAN	*Local Area Network*
LANMAR	*Landmark Ad-Hoc Routing Protocol*
MAC	*Media Access Control*
MANETs	*Mobile Ad-hoc Networks* (Redes móviles ad-hoc)
MATLAB	*Matrix Laboratory*
MIMO	*Multiple-Input Multiple Output*
MIT	*Massachusetts Institute of Technology*
MiXiM	*Mixed Simulator*
MM-DSR	*Multipath Multimedia-Dynamic Source Routing*
MPR	*Multi Point Relays*
MU-MIMO	*Multi User-MIMO*

NAM	*Network AniMator*
NASA	*National Aeronautics and Space Administration*
NAV	*Network Allocation Vector*
NCS	*Neighborhood Comprensive Sensing*
NGN	*Next Generation Networking*
NTDR	*Near-Term Digital Radio*
NS-2	*Network Simulator* Version 2
OBU	*On-Board Unit*
OCEAN	*Observation-based Cooperation Enforcement in Ad-hoc Networks*
OFDM	*Orthogonal Frequency-Division Multiplexing*
OLSR	*Optimazed Link-State Routing Protocol*
ONE	*Opportunistic Network Environment simulator*
OTCL	*Object Oriented Tool Command Language*
PDA	*Personal Digital Assistant*
PhD	*Philosophiae Doctor*
PFM	*Positive Feedback Message*
PHY	*Physical Layer*
PIFS	*Point Coordination IFS*
PRNET	*Packet Radio Network*
PSC	*Physical Sense Carrier*
P2P	*Point-to-Point*
QoE	*Quality of Experience*
QoS	*Quality of Service*

QualNet	*Quality Network Simulator*
RDSR-V	*Reliable Dynamic Source Routing for Video-streaming over mobile ad-hoc networks*
RED	*Random Early-Detection*
REMODE	*Reputation for Mobile Devices*
REMODE_sw	*REMODE Static Weight*
RFVs	*Reputation Forwarding Values*
RFSN	*Reputation-based Framework for high integrity Sensor Networks*
RREP	*Route Reply*
RREQ	*Route Request*
RREQ_ID	*Route Request Identifier*
RR-RTCP	*Receive Report – Real-time Transport Control Protocol*
RSUs	*Road Side Units*
RTCP	*Real-time Transport Control Protocol*
RT	*Routing Table*
RT Framework	*Role-based Trust management Framework*
RTO	*Retransmission Time-Out*
RTP	*Real-time Transport Protocol*
RTS	*Request To Send*
RTT	*Round Trip Time*
SIFS	*Short Inter Frame Space*
SMA	*Simple Moving Average*
SURAN	*Survivable Adaptative Radio Network*

TICs	Tecnologías de la Información y las Comunicaciones
TCP	*Transmission Control Protocol*
TCP-Linux Vegas	*Transmission Control Protocol-Linux* Versión Vegas
TCP-Vegas	*Transmission Control Protocol* Versión Vegas
TTL	*Time-To-Live*
TV	*Trust Vector Model*
UC	*University of California*
UMTS	*Universal Mobile Telecommunications System*
UWB	*Ultra Wide Band*
VANETs	*Vehicular Ad-Hoc Networks*
VBR	*Variable Bit Rate*
VCS	*Virtual Carrier Sense*
VINT	*Virtual InterNetwork Testbed*
V2I	*Vehicle to Infrastructure*
V2V	*Vehicle to Vehicle*
Wi-Fi	*Wireless Fidelity*
WiMax	*Worldwide Interoperability for Microwave Access*
WLAN	*Wireless Local Area Network*
WMNs	*Wireless Mesh Networks*
WSAN	*Wireless Sensor and Actor Network*
WSNs	*Wireless Sensor Networks*
WUSB	*Wireless Universal Serial Bus*
ZRP	*Zone Routing Protocol*

Capítulo 1

Introducción

"La libertad no es poder elegir entre unas pocas opciones impuestas, sino tener el control de tu propia vida. La libertad no es elegir quién será tu amo, es no tener amo".

— Richard Matthew Stallman. Programador y activista de la libertad de software estadounidense.1953

"La tecnología es un siervo útil, pero un amo peligroso".

— Christian Lous Lange. Historiador noruego, profesor y politólogo. 1869-1938

1 Introducción

La tesis para el grado de PhD (del latín Philosophiae Doctor), la hemos centrado en la capa de red de las redes móviles ad-hoc, específicamente en la utilización de sistemas de reputación para mejorar el encaminamiento de las redes móviles ad-hoc. Reputación,

es una característica propia de seres humanos, pero en esta tesis la aplicamos a dispositivos móviles. Al interactuar entre un grupo de personas, cada una forma su propio criterio de cuan cooperativos son cada uno de los integrantes del grupo, y suele ser más probable pedir ayuda a integrantes del grupo que son más cooperativos. Igualmente, en las redes móviles ad-hoc (MANETs), mediante los sistemas de reputación, cada dispositivo móvil (nodo) que conforma la red puede tener su propio criterio, de cuan cooperativos son los demás nodos de la red, y suele ser más probable pedir ayuda en la retransmisión de paquetes a los dispositivos móviles más cooperativos, es decir, a los de mejor reputación.

La tesis se compone de cinco capítulos más cuatro anexos. Este capítulo es dedicado a la introducción de la tesis doctoral, en el cual presentamos una breve motivación a esta importante rama del conocimiento, los objetivos y finalmente una descripción del contenido de los capítulos.

1.1 Motivación

En el 2020 el 70% de la población mundial utilizará teléfonos inteligentes [**1**], los cuales tendrán capacidades de cómputo y comunicación demasiado altas en comparación al uso de sus propietarios, además la robustez, confiabilidad y cantidad de sensores y actuadores irá en aumento, por tanto quedarán subutilizadas las capacidades de dichos dispositivos, a menos que se desarrollen nuevos servicios telemáticos que hagan uso de estas capacidades [**2**] [**3**], con la finalidad de controlar y automatizar tareas, brindando una mejora significativa en la calidad de vida de las personas.

Mediante las TICs (Tecnologías de la Información y la Comunicación), la tendencia en esta segunda década del siglo XXI es que todo sea inteligente, que todo sea automático, que muchas tareas se realicen por si solas, de tal forma que cualquier objeto cotidiano pueda tener un determinado nivel de autonomía en base a su inteligencia pre programada. Un vehículo podría ser auto-conducido, una casa administrada por si sola e inclusive una ciudad podría llegar a operarse por su cuenta.

Para el desarrollo de servicios telemáticos que aportan inteligencia a nuestro mundo, entre otros elementos se requiere de un soporte de red, para lo cual existen múltiples opciones. La presente investigación se centra en un tipo de red ad-hoc inalámbrica

denominada red móvil ad-hoc (MANET). Las redes móviles ad-hoc están especializadas en el despliegue de dispositivos móviles sin la existencia de ninguna infraestructura, debido a ello son muy utilizadas en áreas de desastres, en operaciones militares, entre otras aplicaciones [**4**].

Como ejemplo relevante de aplicación que usa adecuadamente los recursos disponibles en los teléfonos inteligentes y que coopera con una ciudad inteligente, tenemos el envío de video mensajes de advertencia para reportar accidentes sobre redes móviles ad-hoc, no simplemente el clásico envío de mensajes de texto. Cuando un ciudadano observa un accidente en la ciudad, podría actuar como un sensor, capturando y enviando un video corto a través de la red móvil ad-hoc con la finalidad de que el nivel de gravedad del accidente sea evaluado correctamente por las autoridades (por ejemplo las unidades de salud, la policía y los conductores de ambulancia) y además otros ciudadanos puedan ser conscientes rápidamente del incidente [**5**].

Para proporcionar una plataforma conectada común para una variedad de aplicaciones, en la arquitectura general de red celular 5G, entre sus componentes tecnológicos estará presente el internet de las cosas (IoT) [**6**] y la comunicación dispositivo a dispositivo (D2D). El internet de las cosas permitirá la interconexión digital de objetos cotidianos con la internet, y la comunicación dispositivo a dispositivo permitirá la comunicación directa entre dispositivos para el intercambio local de tráfico de usuario sin pasar por una infraestructura de red [**7**], al aumentar la complejidad de la comunicación dispositivo a dispositivo se podrá establecer comunicación a través de múltiples saltos mediante el uso de protocolos para redes móviles ad-hoc, inclusive será posible formar una red híbrida al conectar la red móvil ad-hoc mediante un nodo puerta de enlace con algún punto de acceso conectado a su vez con infraestructura 5G.

La infraestructura de red 5G evolucionará incrementando su capacidad, mejorando las tasas de datos, decrementando los retardos, permitiendo una conectividad masiva de dispositivos, reduciendo en medida de lo posible costos, brindando una consistente calidad de experiencia (QoE) y además permitiendo la comunicación de dispositivos móviles que se encuentren a mayor velocidad, a pesar de todo ello aún se seguirá requiriendo la presencia de redes móviles ad-hoc. En estadios, hoteles, centros comerciales y otros lugares donde se congregan masas de personas se requiere una

mayor necesidad de densificar la red móvil a través del despliegue de Wi-Fi y de células pequeñas. Mientras que Wi-Fi ha demostrado un gran éxito para los lugares poblados, la industria de la radio con licencia en general ha sido lenta para adoptar y desplegar células pequeñas debido al costo que ello implica [**8**], por tal razón una alternativa muy viable son las redes móviles ad-hoc que no requieren ningún costo en la instalación de infraestructura y que se implementan mediante Wi-Fi en modo ad-hoc (el módulo Wi-Fi está presente prácticamente en todos los teléfonos inteligentes), entonces así evolucione la infraestructura de red de las operadoras se seguirá requiriendo redes sin infraestructura como las redes móviles ad-hoc.

Las redes móviles ad-hoc no intentan competir con las redes con infraestructura sino ser un complemento muy necesario, ya que cada tipo de red tiene su aplicación y es más adecuada para algún escenario en particular. Si consideramos el área de desastre que tenemos en un terremoto en la que todos los e-NodeB (enhanced Node B) no están operativos, y se requiere con urgencia y precisión coordinar el rescate de personas afectadas mediante mensajes de audio y video, entonces la red móvil ad-hoc estaría siendo utilizada como un complemento perfecto de respaldo en caso de que la infraestructura de red de las operadoras tengan problemas graves.

Un ejemplo avanzado del uso de las redes móviles ad-hoc, lo tenemos en un diseño de comunicaciones sobre la superficie lunar o planetaria. En el ejemplo puntual, la NASA (National Aeronautics and Space Administration) utiliza una implementación del protocolo DSR (Dynamic Source Routing) en FreeBSD 5.x y 6.0 para el diseño de la red móvil ad-hoc. En el diseño de la red se selecciona DSR debido a su habilidad para mantener una tabla de encaminamiento con múltiples caminos, esta redundancia es un requerimiento crítico en este tipo de aplicación. En misiones futuras de exploración, la NASA requerirá establecer una comunicación interactiva entre humanos y robots, que trabajarán juntos para lograr objetivos científicos y de ingeniería en superficies planetarias, de esta manera la comunicación con Tierra no será la única comunicación existente, sino que los elementos superficiales podrán interactuar entre ellos mediante una red móvil ad-hoc, por ejemplo un geólogo que recolecta muestras en la superficie lunar puede trabajar con un asistente robotizado para anotar o analizar estas muestras [**9**].

Para que los teléfonos inteligentes (nodos) que participan en las redes móviles ad-hoc tengan la capacidad de buscar caminos hacia determinados destinos, se requiere de la presencia de protocolos de encaminamiento, estos pueden ser proactivos, reactivos e híbridos. La presente investigación se enfoca en el uso de reputación local y técnicas de cruce de capa para desarrollar dos nuevos algoritmos de protocolos de encaminamiento reactivos que seleccionan caminos de mejor reputación.

En un sistema de reputación, cada nodo tiene una visión consciente de la reputación de los otros nodos de la red, es decir cada nodo mantiene una tabla que almacena todos los valores de reputación de los vecinos [**10**]. La tabla de reputación de un nodo la puede aprender el propio nodo fuente, y en tal caso sería información local de primera mano, o la puede obtener de información compartida por otros nodos y en tal caso sería información de segunda mano [**11**].

La presencia de sólo unos pocos nodos con mal comportamiento puede dañar de manera espectacular el rendimiento de todo el sistema, por ello se requiere la implementación de sistemas de reputación. Algunos sistemas de reputación no siempre son lo suficientemente eficientes o lo suficientemente efectivos, es decir consumen demasiados recursos, o carecen de la capacidad de reflejar con precisión el comportamiento de los nodos [**12**], esto debe ser considerado al momento de integrar un sistema de reputación dentro del protocolo de encaminamiento.

La investigación en redes móviles ad-hoc es muy amplia, existen una infinidad de servicios telemáticos por ser desarrollados y a la vez existen una infinidad de técnicas desarrolladas y por desarrollar para mejorar el encaminamiento o para la adaptación del protocolo de encaminamiento a determinado servicio. Las redes móviles ad-hoc como complemento a las redes 5G, permitirán densificar la red en áreas muy pobladas, servirán como respaldo en áreas de desastres y solucionarán problemas tan complejos como la intercomunicación sobre la superficie lunar o planetaria.

Considerando la importancia y magnitud del contexto previo, esta tesis doctoral contribuye a la mejora de los protocolos de encaminamiento en redes móviles ad-hoc utilizando sistemas de reputación.

1.2 Objetivos de la tesis

El objetivo general de la presente tesis es:

- Contribuir al diseño de protocolos de encaminamiento en redes móviles ad-hoc basados en reputación

Para conseguir este objetivo general, se plantean los siguientes objetivos específicos:

- Estudiar las generalidades de las redes móviles ad-hoc
- Proponer un algoritmo de protocolo de encaminamiento basado en la reputación de peso estático de los nodos de la red móvil ad-hoc, evaluar, analizar y comparar las mejoras propuestas
- Proponer un algoritmo de protocolo de encaminamiento basado en la reputación de peso dinámico de los nodos de la red móvil ad-hoc, evaluar, analizar y comparar las mejoras propuestas

Como punto de partida de la investigación hemos estudiado el estado del arte de los protocolos de encaminamiento para redes móviles ad-hoc, posteriormente hemos investigado la estructura de la memoria caché de encaminamiento del nodo que participa en una red móvil ad-hoc, hemos investigado el mecanismo de selección de camino que utilizan los protocolos en las redes móviles ad-hoc, hemos propuesto un algoritmo para la implementación de un sistema de reputación y confianza, hemos propuesto un protocolo de encaminamiento que utilice técnicas de reputación y confianza, hemos evaluado dichas propuestas utilizando para ello herramientas de simulación; y, finalmente hemos analizado y comparado las mejoras propuestas.

Los dos nuevos algoritmos de protocolos de encaminamiento, son una mejora de DSR y pueden ser utilizados en aplicaciones diversas que utilicen RTP/UDP (por ejemplo video streaming) como TCP (por ejemplo páginas HTML, y servicios varios que utilicen TCP). Los dos nuevos algoritmos de protocolos de encaminamiento basados en reputación para redes móviles ad-hoc los hemos denominado REMODE_sw (Reputation for Mobile Devices Static Weight) y DrepSR(Dynamic reputation Source Routing). Para realizar las simulaciones, hemos utilizado un entorno de trabajo sobre NS-2.35 para REMODE_sw y un entorno de trabajo en MATLAB R2017b para DrepSR.

Las redes móviles ad-hoc, permiten establecer un sistema de comunicación inalámbrico multi-salto entre personas mediante sus teléfonos inteligentes, portátiles o algún otro dispositivo móvil que se requiera desarrollar, incrementando significativamente la conectividad, misma que podría ser utilizada para brindar una infinidad de servicios, aportando múltiples mejoras en el día a día de las personas.

Con la presente tesis enfocada en mejorar el encaminamiento de las redes móviles ad-hoc, podemos entender que el camino más corto no siempre es el mejor, ya que pueden haber caminos cortos en los que participan nodos poco cooperativos que degradan significativamente el rendimiento de la red; y, por el contrario, existir caminos alternos un tanto más largos, cuyos nodos que lo conforman son cooperativos, y permiten obtener un muy buen rendimiento de la red.

1.3 Organización de la tesis

Hemos desarrollado sistemáticamente la tesis en cinco capítulos. En esta sección hemos presentado el contenido de cada capítulo, resaltando que el mayor aporte investigativo se encuentra en los capítulos 3 y 4, ya que es aquí donde se plantean los dos nuevos algoritmos de protocolos de encaminamiento para redes móviles ad-hoc.

El capítulo 2 estudia las generalidades de las redes móviles ad-hoc que se requieren tomar en cuenta para el diseño de protocolos basados en reputación. El capítulo comienza con una breve introducción a las redes móviles ad-hoc, para posteriormente estudiar la tecnología empleada, la clasificación y aplicaciones, los protocolos de encaminamiento haciendo énfasis en la clasificación de dichos protocolos, el protocolo DSR y los mecanismos que lo conforman, los sistemas de reputación, la propuesta de protocolo de encaminamiento REMODE_sw basado en reputación de los nodos, los protocolos utilizados para el diseño de cruce de capa, las métricas de rendimiento de red utilizadas, las herramientas de simulación utilizadas; y, finalmente las conclusiones del capítulo.

El capítulo 3 presenta a REMODE_sw. El capítulo inicia con una breve introducción, a continuación se presenta la evolución de la métrica de reputación de RDSR-V que da lugar a REMODE_sw (y posteriormente a DrepSR), luego se desarrolla la descripción matemática de REMODE_sw, posteriormente se presenta el pseudocódigo de

REMODE_sw, se continua el desarrollo del capítulo con los resultados en NS-2.35 y análisis, donde se realiza un análisis de los valores de reputación al variar los valores de los pesos y en escenario estático, un análisis de la reputación muestreada, reputación de nodos y calificación para caminos al incrementar las pérdidas de los nodos no cooperativos en escenario estático, un análisis del contenido de la memoria cache primaria del nodo fuente al existir múltiples caminos en escenario estático, un análisis de la reputación de los caminos activos, de la reputación de nodos y calificación de caminos en escenario móvil, un análisis de los valores de reputación de nodos cuando se tiene tráficos asimétricos, un análisis de la recuperación de la reputación de un nodo en escenario móvil; y, finalmente se presenta las conclusiones del capítulo 3.

El capítulo 4 presenta DrepSR, el capítulo inicia con la introducción, a continuación se presenta el protocolo de encaminamiento propuesto, se continúa el capítulo con el algoritmo para promediar los valores de reputación de reenvío, resultados experimentales y discusión; y, se cierra el capítulo con las respectivas conclusiones.

El capítulo 5 presenta las conclusiones de toda la tesis, además enumera los artículos de investigación que hemos publicado como resultado del trabajo investigativo; y, finaliza con lineamientos para trabajos futuros.

Capítulo 2

Redes móviles ad-hoc

"Las organizaciones gastan millones de dólares en firewalls y dispositivos de seguridad, pero tiran el dinero porque ninguna de estas medidas cubre el eslabón más débil de la cadena de seguridad: la gente que usa y administra los ordenadores"
—Kevin David Mitnick. Consultor de seguridad informática estadounidense. 1963
"La mejor forma de predecir el futuro es implementarlo"
—David Heinemeier Hansson. Programador danés y creador del popular framework de desarrollo web Ruby on Rails. 1979

2 Redes móviles ad-hoc

El presente capítulo lo dedicamos a las generalidades de las redes móviles ad-hoc, se explican los conceptos necesarios para poder comprender los capítulos posteriores. Este

estado del arte, es fundamental para que podamos explicar los aportes al conocimiento que realiza la tesis doctoral.

2.1 Introducción

Figura 2.1: Redes móviles ad-hoc inalámbricas [**13**]

Las redes móviles ad-hoc son un tipo de red móvil conformada por dispositivos inalámbricos y de cómputo con capacidades de auto-configuración dinámica, auto-optimización y auto-reparación, además no cuentan con una infraestructura fija. Todos estos dispositivos (clientes móviles como portátiles, teléfonos inteligentes, tabletas, etc.) pueden actuar como encaminadores (routers) y/o usuarios finales (terminals) y ofrecer caminos alternativos en una amplia gama de escenarios. Como mostramos en la Figura 2.1, las redes móviles ad-hoc se pueden utilizar en muchos escenarios, tales como redes de vuelo ad-hoc FANETs [**14**] (Flying Ad-hoc Networks), redes móviles ad-hoc metropolitanas, redes vehiculares ad-hoc VANETs [**15**] (Vehicular Ad-hoc Networks) y redes móviles ad-hoc MANETs (Mobile Ad-Hoc Networks) propiamente dichas.

El conjunto de terminales de datos digitales con transceptores inalámbricos sin infraestructura de red y sin control centralizado forman una red móvil ad-hoc. En una red móvil ad-hoc, cuando un nodo quiere transmitir un paquete a otro nodo que está localizado fuera del alcance, el paquete pasa a través de una secuencia de nodos intermedios utilizando un principio de transmisión de almacenamiento y reenvío "multisalto". Las redes móviles ad-hoc son utilizadas en redes de comunicaciones militares, en campos de batalla, operaciones de rescate de emergencia, operaciones

submarinas, monitoreo ambiental y exploración espacial, que son entornos en los cuales es necesario un despliegue de red "sobre la marcha" [**16**].

El diseño de las redes ad-hoc se enfrenta a desafíos únicos. La mayoría de ellos surgen debido a dos razones principales. La primera es que todos los nodos en una red ad-hoc, incluyendo los nodos de origen, los destinos correspondientes, así como los nodos de encaminamiento que reenvían tráfico entre ellos, pueden ser móviles. A medida que el alcance de transmisión inalámbrica es limitado, los enlaces inalámbricos entre un par de nodos vecinos se rompen tan pronto como se mueven fuera del rango. Por lo tanto, la topología de red que se define por el conjunto de enlaces de comunicación físicos en la red (enlaces inalámbricos entre todos los pares de nodos que pueden comunicarse directamente entre sí) puede cambiar con frecuencia y de forma impredecible. Esto implica que el camino multisalto para cualquier par dado de nodos de origen y destino también cambia con el tiempo. La movilidad también provoca imprevisibilidad en la calidad de un enlace inalámbrico existente entre vecinos. Una segunda razón que hace el diseño de redes ad-hoc complicado, es la ausencia de un control centralizado. Todas las funciones de red, tales como la determinación de la topología de red, accesos múltiples, y encaminamiento de datos sobre la mayoría de caminos multisalto apropiados deben ser realizados de forma distribuida. Estas tareas son particularmente difíciles debido al ancho de banda de comunicación limitado disponible en el canal inalámbrico.

Estos desafíos deben ser abordados en todos los niveles del diseño de la red. La capa física debe abordar la pérdida del camino, desvanecimiento, e interferencia multiusuario para mantener enlaces de comunicación estables entre pares. La capa de enlace de datos (DLL) debe tener un vínculo físico fiable y resolver las disputas entre los usuarios no sincronizados que están transmitiendo paquetes en un canal compartido, esta última tarea se realiza por la subcapa de control de acceso al medio (MAC) en la capa de enlace de datos. La capa de red debe seguir los cambios en la topología de la red y determinar adecuadamente el mejor camino a cualquier destino deseado. La capa de transporte debe compensar el retardo y las características de pérdida de paquetes específicas a una red inalámbrica tan dinámica. Incluso la capa de aplicación necesita manejar desconexiones frecuentes [**16**].

La convergencia de redes móviles ad-hoc y redes de sensores inalámbricas conduce al desarrollo de la nueva plataforma de comunicación IoT con un potencial de aplicaciones en diversos dominios. Como la movilidad de los nodos debe ser apoyada en ciudades inteligentes, las redes móviles ad-hoc se utilizan en ciudades inteligentes. En redes móviles ad-hoc, los caminos se configuran repetidamente y este trabajo lo hacen todos los nodos de la red y no tienen un controlador central. Las redes móviles ad-hoc no tienen soporte de infraestructura, los nodos están configurados automáticamente, son tolerantes a fallos y admiten dispositivos móviles. Todas estas características se pueden utilizar para construir una ciudad inteligente basada en redes móviles ad-hoc. En aplicaciones de atención de la salud donde la calidad de vida está involucrada, el retraso, la pérdida de paquetes y el ancho de banda deben ser reducidos. Se debe dar soporte de QoS adaptada al contexto en las ciudades inteligentes [**17**].

Aprovechando que las redes móviles ad-hoc pueden ser hibridas, se puede integrar con varias redes para ampliar la comunicación. En algunas redes con múltiples puertas de entrada (gateways), la selección de puerta de entrada adecuada garantiza la calidad de servicio deseable y la optimización de la utilización de recursos de red [**18**].

Al ser redes IP, las redes móviles ad-hoc, serán utilizadas como un tipo importante de red de acceso de nueva generación (NGN), por ello en este capítulo es fundamental realizar un estudio de las principales generalidades de estas redes.

2.2 Tecnología empleada

2.2.1 Redes IEEE 802.11

El estándar IEEE 802.11 [**19**] es un conjunto de especificaciones de control de acceso al medio (MAC) y capa física (PHY) para implementar la comunicación de red de área local inalámbrica (WLAN) de bajo costo. La familia 802.11 es una serie de técnicas de modulación por aire que proporcionan la base para productos de redes inalámbricas que utilizan la marca Wi-Fi. El segmento del espectro de radiofrecuencia utilizado por 802.11 varía de un país a otro [**20**].

Respecto a la capa física (PHY), el simulador NS-2.35 [**21**], para los radioenlaces terrestres de microondas (1 GHz y 300 GHz según IEEE 100), permite seleccionar entre tres opciones, FreeSpace, TwoRayGround y Shadowing, los experimentos de esta tesis

se los ha realizando utilizando TwoRayGround, ya que es el que más se adapta a los escenarios que se plantean en los capítulos siguientes.

Al utilizar el modelo de reflección en tierra (TwoRayGround), el simulador NS-2.35 utiliza la siguiente lógica, primeramente calcula la distancia de cruce que se muestra a continuación en la Ec. 2.1:

$$Crossover\_dist = \frac{4*\pi*ht*hr}{\lambda} \quad (2.1)$$

Donde:

Crossover_dist: Distancia de cruce en metros

ht: Altura del transmisor en metros

hr: Altura del receptor en metros

λ: Longitud de onda en metros

En la distancia de cruce, la potencia recibida predicha por el modelo de dos rayos, es igual a la predicha por la ecuación de Friis.

En caso de que d<Crossover_dist utiliza la ecuación del modelo de propagación de espacio libre o ecuación de Friis mostrada a continuación en la Ec. 2.2:

$$\Pr(d) = \frac{Pt*Gt*Gr*\lambda^2}{L*(4\Pi)^2*d^2} \quad (2.2)$$

Donde:

Pr(d): Potencia recibida, la cual es una función de la separación entre el transmisor y el receptor, esta variable está en vatios

Pt: Potencia transmitida en vatios

Gt: Ganancia de la antena transmisora es una cantidad adimensional

Gr: Ganancia de la antena receptora es una cantidad adimensional

λ: Longitud de onda en metros

L: Factor de pérdidas del sistema. Varias pérdidas, usualmente debidas a la atenuación en la línea de transmisión, perdidas en filtros y pérdidas en la antena. L=1 indica que no existen pérdidas en el hardware del sistema de comunicación

d: Es la distancia en metros, la separación entre el transmisor y el receptor

Caso contrario el simulador utiliza la ecuación del modelo de reflexión a tierra o modelo de dos rayos mostrada a continuación en la Ec. 2.3:

$$Pr = \frac{Pt*Gt*Gr*ht^2*hr^2}{d^4*L} \quad (2.3)$$

Donde las variables que participan en la ecuación, son las mismas que se enumeró en la ecuación anterior, con la diferencia de que aquí no se toma en cuenta la longitud de onda, pero sí la altura del transmisor y del receptor:

ht: Altura del transmisor en metros

hr: Altura del receptor en metros

Para ampliar más al respecto de estas últimas ecuaciones de propagación de ondas de radio con el modelo de dos rayos, se puede revisar la referencia [**22**] para recordar la teoría, y la [**23**] para una aplicación en el desarrollo de un protocolo para redes vehiculares ad-hoc sobre AODV; y, desde luego la implementación que viene por defecto en el código fuente de NS-2.35, que se encuentra instalada en : /usr/src/ns-allinone-2.35/indep-utils/propagation/threshold.cc

En el script OTCL primeramente se selecciona el modelo de propagación de ondas de radio:

set val(prop) Propagation/TwoRayGround; #Radio-propagation model

Y, luego este modelo se lo establece para cada nodo de la red móvil ad-hoc:

$ns_ node-config –propType $val(prop)

La versión original de IEEE 802.11 de 1997, es básicamente obsoleta hoy, en ella se especificaron velocidades de bits de 1 o 2 Mbps, se especificaba tres tecnologías de capa física alternativas, la infrarroja a 1Mbps, la de espectro de dispersión con salto de frecuencia (FHSS) operando a 1 o 2 Mbps, y la de espectro de dispersión de secuencia

directa (DSSS) operando a 1 o 2 Mbps. Las dos últimas tecnologías de radio utilizaron la transmisión por microondas a través de la banda de frecuencia Industrial Científica y Médica (ISM) a 2,4 GHz. Su velocidad de datos especificada debía transmitirse por medio de señales de infrarrojos (IR), o por señales de radio de salto de frecuencia o de secuencia directa de espectro ensanchado. Infrarrojos sigue siendo una parte de la norma, pero no tiene implementaciones reales.

Estándares de la capa física de IEEE 802.11						
Lanzamiento	**Estándar**	**Banda de Frecuencia (GHz)**	**Bandwidth (MHz)**	**Modulación**	**Tecnologías de Antena Avanzada**	**Máxima Tasa de Datos**
1997	802.11	2.4	20	DSSS, FHSS	-	2Mbps
1999	802.11b	2.4	20	DSSS	-	11Mbps
1999	802.11a	5	20	OFDM	-	54Mbps
2003	802.11g	2.4	20	DSSS,OFDM	-	54Mbps
2009	802.11n	2.4 y 5	20 y 40	OFDM	MIMO, hasta 4 secuencias espaciales	600Mbps
2013	802.11ac	5	40, 80 y 160	OFDM	MIMO, MU-MIMO, hasta 8 secuencias espaciales	6.93Gbps

Tabla 2.1: IEEE 802.11 estándares de la capa física [**20**]

Una debilidad de esta especificación original era que ofrecía tantas opciones que la interoperabilidad era a veces desafiante. En realidad, es más una "especificación beta" que una especificación pero con poca o ninguna operatividad entre vendedores.

La versión DSSS del tradicional 802.11 fue rápidamente completada (y popularizada) por la enmienda 802.11b en 1999, que aumentó la velocidad de bits a 11Mbps. La amplia difusión de la adopción de redes 802.11 sólo se produjo después del lanzamiento

de 802.11b. Como resultado, se implementaron pocas redes utilizando el estándar original 802.11 de 1997 [**20**].

La Tabla 2.1 resume de una manera sencilla la fecha de lanzamiento, la banda de frecuencia, el ancho de banda, la modulación, si usa o no alguna tecnología avanzada de antena y finalmente, la tasa de datos máxima que puede alcanzar cada estándar, es decir se presentan las características de capa física de cada estándar IEEE 802.11.

Una WLAN (Wireless LAN) utiliza un medio compartido donde un número indeterminado de host puede competir por el medio en cualquier momento. Las colisiones son un hecho constante en la WLAN porque está en modo "half dúplex" y siempre dentro de la misma frecuencia. Sólo una estación puede transmitir en un determinado momento de tiempo. Para lograr el modo "full dúplex" todas las estaciones que transmiten y las que reciben deberían hacerlo en frecuencias diferentes. Operación no permitida en IEEE 802.11 [**24**].

Para que una estación pueda enviar una trama Ethernet 802.3 hacia otra estación, primeramente pasará por la subcapa Logical Link Control (802.2), luego por la subcapa MAC, a continuación por la capa física (secuencia directa) y finalmente a la interfaz radio [**25**].

AirDrop-P basado en un PIC (18LF8621) y AirDrop-A en un AVR (ATMEGA128-L) son ejemplos prácticos de implementaciones de 802.11, donde se trata en profundidad el hardware y el software (en C) necesarios para realizar la implementación de 802.11 en un micro controlador [**26**].

Existen cuatro áreas de aplicación para las redes LAN inalámbricas: ampliación de redes LAN, interconexión de edificios, acceso nómada y redes ad-hoc. Como ejemplo de ampliación de redes LAN, tenemos edificios que poseen una gran superficie, como plantas de fabricación, plantas comerciales y almacenes, edificios históricos con insuficiente cantidad de par trenzado y en los que está prohibido hacer más agujeros para introducir nuevo cableado, y pequeñas oficinas donde la instalación y el mantenimiento de una LAN cableada no resulten rentables. Como ejemplo de interconexión de edificios tenemos la conexión de redes LAN situadas en edificios vecinos, sean LAN cableadas o inalámbricas. En este caso se utiliza un enlace punto a

punto inalámbrico entre los dos edificios, los dispositivos así conectados son puentes o dispositivos de encaminamiento. Como ejemplo de acceso nómada, tenemos el posibilitar a un empleado que vuelve de un viaje la transferencia de datos desde un computador personal portátil, a un servidor en la oficina, el acceso nómada resulta útil también en un entorno amplio, como un campus o un centro financiero situado lejos de un grupo de edificios. En ambos casos, los usuarios se pueden desplazar con sus computadores portátiles y pueden desear conectarse con los servidores de una LAN inalámbrica desde distintos lugares. Finalmente en una red ad-hoc, en una red entre iguales (sin servidor central) establecida temporalmente para satisfacer alguna necesidad inmediata. Por ejemplo, un grupo de empleados, cada uno con su computador, puede reunirse para una cita de negocios o para una conferencia conectando entre sí sus computadores en una red temporal sólo durante la reunión [**27**].

2.2.2 Problema de la terminal oculta y expuesta

En la subcapa de control de acceso al medio (MAC), tenemos que en una red cableada todas las estaciones se disponen en un cable y se limita la longitud del cable, permitiendo con ello que todas las estaciones siempre se puedan escuchar, y cualquier colisión sea detectada por todas las estaciones del cable. En contraparte en una red inalámbrica es improbable que todas las estaciones puedan recibir todas las transmisiones todo el tiempo, y a esto se conoce como el problema de la terminal oculta.

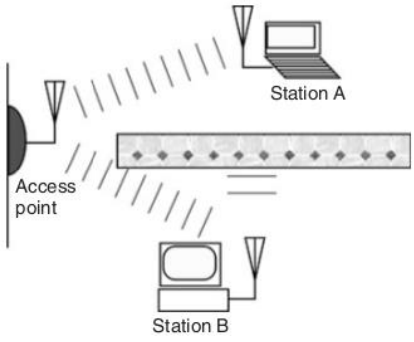


Figura 2.2: Problema de la terminal oculta [**16**]

En la Figura 2.2, el punto de acceso puede comunicarse con las estaciones A y B, pero A y B no pueden recibir las transmisiones del otro directamente, por ello la detección de colisión fallaría si A y B intentan transmitir al mismo tiempo.

Para superar esta dificultad, las estaciones 802.11 utilizan el acceso múltiple por detección de portadora evitando colisiones (CSMA/CA). Todas las estaciones escuchan el canal de radio actual antes de transmitir, si se detecta una señal, se considera que el medio está ocupado y la estación difiere su transmisión. Se proporciona un mecanismo de detección de portadora virtual, y el vector de asignación de red (NAV) reduce aún más la probabilidad de colisiones [**16**].

Una estación que desea transmitir un paquete primero transmite un paquete de control corto llamado petición para enviar (RTS), la estación destino responde (si el medio está libre) con un paquete de control de respuesta llamado listo para enviar (CTS).

Debido a que las colisiones no pueden ser detectadas, se toma el enfoque opuesto a las redes cableadas, 802.11 depende del reconocimiento positivo de una transmisión satisfactoria a través de la recepción de un acuse de recibo (ACK*)* por la estación transmisora. El no recibir un acuse de recibo, ya sea debido a la debilidad de la señal, colisiones con otras estaciones, o interferencia, se considera que es indicativo de una colisión, la estación transmisora esperará hasta que el medio vuelva a estar libre, elija una ranura de tiempo aleatoria dentro de una ventana más grande posible (el mecanismo de retroceso de Ethernet) e intente transmitir el paquete de nuevo [**16**].

La detección de portadora es dependiente de la ubicación

Puesto que la fuerza de la señal recibida depende de la distancia desde el transmisor, la misma señal no se escucha igualmente bien por todos los terminales. Por lo tanto, la detección de la portadora no es muy eficaz en medios inalámbricos. Los problemas típicos de utilizar el sensor de portadora para determinar la disponibilidad del canal inalámbrico son:

- El problema del terminal oculto: un nodo puede estar oculto o fuera del alcance de un remitente pero dentro del alcance de su receptor deseado. Por ejemplo, en la Figura 2.3a, el nodo C está fuera del rango de A, y por lo tanto, cualquier transmisión de C no puede ser escuchada por A. Así que mientras C está transmitiendo a B, el nodo A piensa que el canal está inactivo y transmite simultáneamente un paquete de datos al nodo B. Esto hace que ambos paquetes

se pierdan en B debido a la interferencia, y se considera que los paquetes han sufrido una "colisión". Una transmisión de A a B tendrá la misma consecuencia incluso si C está transmitiendo a algún otro nodo, como D.

- El problema del terminal expuesto: este es el problema inverso, donde un nodo transmisor o "expuesto" está dentro del alcance de un remitente pero está fuera del rango del destino. El problema se ilustra en la Figura 2.3b, donde el nodo B, que desea transmitir un paquete de datos a A, encuentra que el canal está ocupado debido a la transmisión de C a D. Por lo tanto, B podría esperar a que la transmisión de C se de antes de la transmisión a A, lo cual no es necesario ya que la transmisión desde B no interferiría en D.

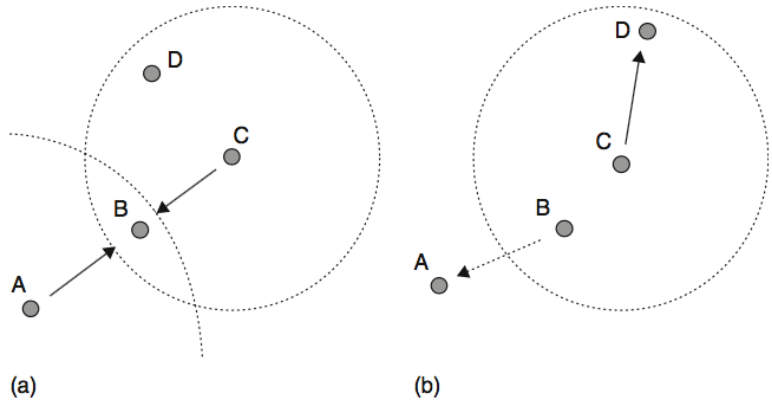


Figura 2.3: (a) El problema del terminal oculto, y (b) el problema del terminal expuesto. Los círculos punteados representan el radio de cobertura de los transmisores [**16**]

Tanto el problema del terminal oculto como el del terminal expuesto surgen debido al hecho de que la detección de portadora sólo se realiza en el transmisor, mientras que su efecto está determinado por la potencia de interferencia en el receptor, que son usualmente diferentes debido a las características de pérdida de trayecto de propagación.

La detección de colisiones no es posible

Un transceptor inalámbrico no puede transmitir y recibir al mismo tiempo, ya que la señal transmitida será siempre mucho más fuerte que cualquier señal recibida. Por lo tanto, un terminal inalámbrico no puede detectar si su transmisión ha sido exitosa. Para

informar al nodo transmisor acerca de una transmisión de paquetes exitosa, el receptor envía un paquete de acuse de recibo (ACK) de vuelta al transmisor después de recibir un paquete de datos. Si el transmisor no recibe un acuse de recibo dentro de un período de tiempo fijo, asume que el paquete transmitido se ha perdido. Sin embargo, esto se aprende sólo después de completar la transmisión del paquete de datos y esperar un período de tiempo de espera de acuse de recibo, no más [**16**].

Espacio Inter Trama IFS (Inter Frame Space)

La Norma define 4 tipos de espacios inter trama, que se utilizan para proporcionar diferentes prioridades:

- SIFS: *Short Inter Frame Space*, se utiliza para separar las transmisiones pertenecientes a un solo diálogo, y es el espacio inter trama mínimo. Siempre hay una sola estación para transmitir en cualquier momento dado, por lo tanto, se le da prioridad sobre todas las otras estaciones.

- PIFS: *Point Coordination IFS*, es utilizado por el punto de acceso (o coordinador de puntos, como se llama en este caso), para acceder al medio antes de cualquier otra estación. Este valor es SIFS más un tiempo de ranura.

- DIFS: *Distributed IFS*, es el espacio inter trama utilizado para una estación que está dispuesta a iniciar una nueva transmisión, que se calcula como *PIFS* más un intervalo de tiempo.

- EIFS: *Extended IFS*, que es un espacio inter trama más largo utilizado por una estación que ha recibido un paquete que no pudo entender. Este es necesario para evitar que la estación (que no puede entender la información de duración para el sensado de portadora virtual) colisione con un paquete futuro que pertenece al diálogo actual.

Algoritmo de retroceso exponencial (EBA)

Retroceso (*back-off*) es un método bien conocido utilizado para resolver conflictos (*contention*) entre diferentes estaciones que desean acceder al medio. El método requiere que cada estación elija un número aleatorio (n) entre 0 y un número dado, y

espere este número de ranuras antes de acceder al medio, siempre verificando si una estación diferente ha accedido al medio antes.

El tiempo de ranura se define de tal manera que una estación siempre será capaz de determinar si otra estación ha accedido al medio al principio de la ranura anterior. Esto reduce la probabilidad de colisión a la mitad.

Retroceso exponencial significa que cada vez que la estación elige una ranura y pasa a chocar, aumentará el número máximo para la selección aleatoria de forma exponencial.

El estándar 802.11 define un Algoritmo de Retroceso Exponencial (EBA), que debe ejecutarse en los siguientes casos:

- Cuando la estación sensa el medio antes de la primera transmisión de un paquete, y el medio está ocupado

- Después de cada retransmisión, y

- Después de una transmisión exitosa

- El único caso en que no se utiliza este mecanismo es cuando la estación decide transmitir un nuevo paquete y el medio ha estado libre durante más de DIFS.

La Figura 2.4 muestra un esquema del mecanismo de acceso [**28**].

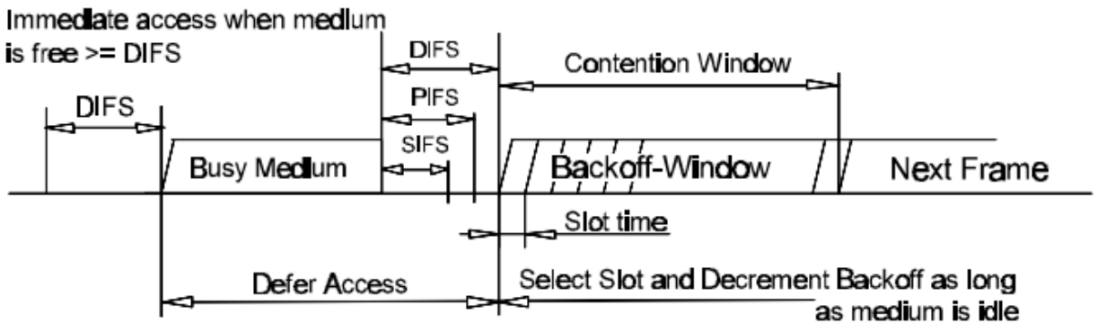


Figura 2.4: Mecanismo de Acceso [**28**]

2.2.3 Descripción técnica de IEEE 802.11g

802.11g apareció en el 2003 y funciona en la banda de 2,4 GHz al igual que 802.11b, pero utiliza el mismo esquema de transmisión basada en OFDM que 802.11a. Funciona a una velocidad de bits de capa física máxima de 54Mbps. El hardware 802.11g es totalmente compatible con el hardware 802.11b. Sin embargo, en una red 802.11g, la presencia de un dispositivo 802.11b reducirá considerablemente la velocidad de la red global 802.11g.

A pesar de su gran aceptación, 802.11g sufre la misma interferencia que 802.11b en la saturada banda de 2,4 GHz. El éxito del estándar ha causado problemas de densidad por el amontonamiento de dispositivos Wi-Fi en áreas urbanas. Para evitar la interferencia existen solamente 3 canales que no se solapan en los Estados Unidos y otros países con regulaciones similares (canales 1, 6, 11, con separación de 25 MHz), y cuatro en Europa (canales 1, 5, 9, 13, con separación de sólo 20 MHz). Incluso con dicha separación, existe cierta interferencia debida a los lóbulos laterales, aunque es considerablemente más débil [**20**].

¿Por qué no conseguimos los 54Mbps que indica el estándar IEEE802.11g

Algunas de las razones que no nos permiten conseguir la máxima velocidad de transmisión teórica con Wi-Fi, y en particular en 802.11g, son las que comentaremos a continuación:

- El Forward Error Correction (FEC) o canal de codificación, es una técnica utilizada para controlar los errores en transmision de datos sobre canales de comunicación no confiables o ruidosos. La idea central es que el transmisor codifique su mensaje en una forma redundante mediante el uso de un código corrector de errores (Error-Correcting Code ECC). La redundancia permite al receptor un número limitado de errores que pueda ocurrir en cualquier parte del mensaje y a menudo corregir estos errores sin retransmisión. FEC da al receptor la habilidad de corregir errores sin necesidad de un canal inverso para petición de retransmisión de datos, pero a costa de un precio fijo, un mayor ancho de banda del canal de ida [**20**]. Entonces la combinación de modulación y redundancia por el uso de códigos para corregir errores, hace perder eficiencia y

no poder conseguir exactamente los máximos valores que especifican los estándares.

- El estándar Wi-Fi utiliza un modo de transmisión llamado half-dúplex, que básicamente significa que una comunicación mediante Wi-Fi no puede enviar y recibir datos simultáneamente, esto es lo que observamos en la Figura 2.5. En la práctica significa que esos teóricos 54 Mbps se reparten entre la transmisión y la recepción de datos. En determinados momentos se transmiten datos y en otros momentos se reciben, pero no puede hacerse a la vez. Por comparar, las tecnologías cableadas de acceso como ADSL, HFC (utilizado por los operadores de cable) o de redes locales como Ethernet, tienen canales separados e independientes para transmitir y recibir datos. Los dispositivos Wi-Fi que cumplen el estándar IEEE 802.11g alcanzan una velocidad de transmisión media de 22 Mbps. En condiciones óptimas pueden llegar hasta 24 Mbps, pero nunca a los 54 Mbps oficiales.

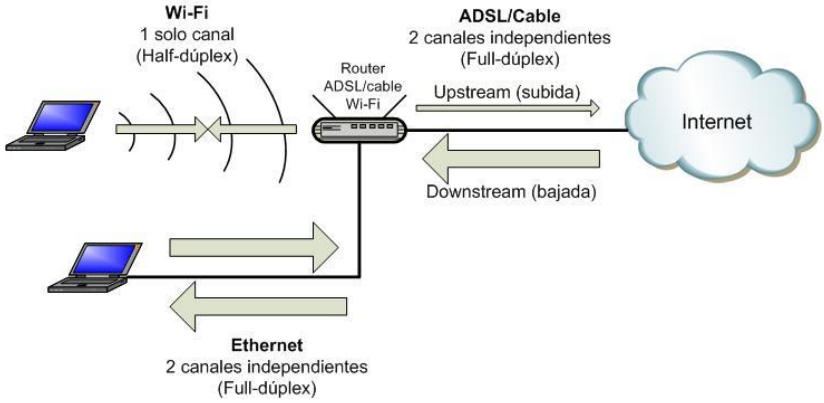


Figura 2.5: Wi-Fi trabaja en modo de transmisión half-duplex [**29**]

- La distancia entre el dispositivo móvil inalámbrico y el punto de acceso (o encaminador con capacidades inalámbricas) en el caso de modo infraestructura, o la distancia entre dispositivos móviles inalámbricos en el caso de modo ad-hoc, es un factor muy importante de atenuación de la señal Wi-Fi. Y esto tiene un efecto inmediato en la velocidad.

- Lo mismo ocurre con los obstáculos (principalmente paredes y techos). La frecuencia utilizada por las ondas electromagnéticas en Wi-Fi son capaces de

atravesar obstáculos, pero la señal queda atenuada y por tanto la velocidad se ve afectada.

- El estándar IEEE 802.11g utiliza para su transmisión una banda de frecuencias de libre uso, por lo que puede existir, en el área de cobertura de una red Wi-Fi, otras transmisiones que utilicen esa misma banda. Esto causa la aparición de las llamadas interferencias. El estándar está preparado para "esquivar" dichas interferencias pero tiene un coste en prestaciones. Los ejemplos más típicos de elementos que pueden producir interferencias son hornos microondas, teléfonos inalámbricos, intercomunicadores inalámbricos caseros y por supuesto, otras redes Wi-Fi.
- A diferencia de lo que ocurre en las redes cableadas en las que se establece un canal exclusivo para las comunicaciones entre el dispositivo de interconexión (switch o router) y los equipos conectados a la red, en Wi-Fi se utiliza un canal compartido por todos los dispositivos conectados a una red inalámbrica. Por tanto, la velocidad de 54 Mbps realmente se refiere a la velocidad del canal, compartido por todos los dispositivos inalámbricos de la red [**29**].

2.3 Clasificación y aplicaciones de las redes ad-hoc inalámbricas

2.3.1 Introducción

Dado que no se requieren estaciones base, las redes ad-hoc pueden ser desplegadas rápidamente, sin tener que realizar ninguna planificación anticipada o construcción de infraestructura de red costosa, esta facilidad da lugar a numerosas aplicaciones que se las enumerará a continuación.

2.3.2 Historia y aplicaciones de las redes móviles ad-hoc

A pesar del auge que han tenido estas redes en los últimos años, la idea de la creación de redes ad-hoc se inició en la década de 1970 cuando la Agencia de Investigación de Proyectos Avanzados para la Defensa de EEUU (DARPA), patrocinó el proyecto Red de Paquetes de Radio (PRNET) en 1972. Esto fue seguido por el proyecto Red de Radio de Supervivencia Adaptativa (SURAN) en la década de 1980. Estos proyectos apoyaron la investigación sobre el desarrollo del establecimiento de llamada automática y el mantenimiento de redes de radio por paquetes con una movilidad moderada. Sin

embargo, el interés en esta área creció rápidamente en los años 1990 debido a la popularidad de un gran número de dispositivos digitales portátiles tales como ordenadores portátiles y de bolsillo, y la disponibilidad común de dispositivos de comunicación inalámbricos. La creciente popularidad de Internet añade el interés de desarrollar protocolos de interconexión para redes móviles ad-hoc que operan en la banda de frecuencia de radio libre de licencia (tal como, la ISM en Estados Unidos). Con el fin de desarrollar protocolos basados en IP para redes ad-hoc, un grupo de trabajo para redes móviles ad-hoc (MANET) se formó dentro del Grupo de Trabajo de Ingeniería de Internet (IETF). El Departamento de Defensa (DoD) también renovó su apoyo a los objetivos de investigación similares al iniciar los proyectos Sistema Global de Información Móvil (GloMo) y el de Radio Digital a Corto Plazo (NTDR). Estimulado por el creciente interés en la creación de redes ad-hoc, una serie de normas comerciales se desarrollaron a finales de 1990. Estos incluyeron la capa física IEEE 802.11 y el protocolo MAC en 1995, que desde entonces está dentro de más versiones de actualización. Hoy en día, se puede crear una red ad-hoc con sólo conectar tarjetas PCMCIA 802.11 en ordenadores portátiles. Bluetooth e Hiperlan son algunos ejemplos de los productos existentes relacionados.

Las aplicaciones típicas de las redes móviles ad-hoc incluyen redes de comunicación militares en los campos de batalla, las operaciones de rescate de emergencia, operaciones submarinas, monitoreo ambiental y la exploración espacial. Debido a su calidad de implementación "on the fly" y relativo bajo costo de implementación, las redes ad-hoc (redes sin infraestructura) son también utilizadas en lugares donde son más baratas que sus homólogas infraestructuradas. Ejemplos de estas aplicaciones consisten en una red de ordenadores portátiles en una sala de conferencia, una red de equipos electrónicos digitales y aparatos (por ejemplo, aparatos de vídeo, ordenador, impresora, control remoto) para formar una red de área doméstica, redes de robots móviles y juguetes inalámbricos [**16**].

2.3.3 Clasificación de las redes ad-hoc

- Redes de Sensores Inalámbricos (WSNs)

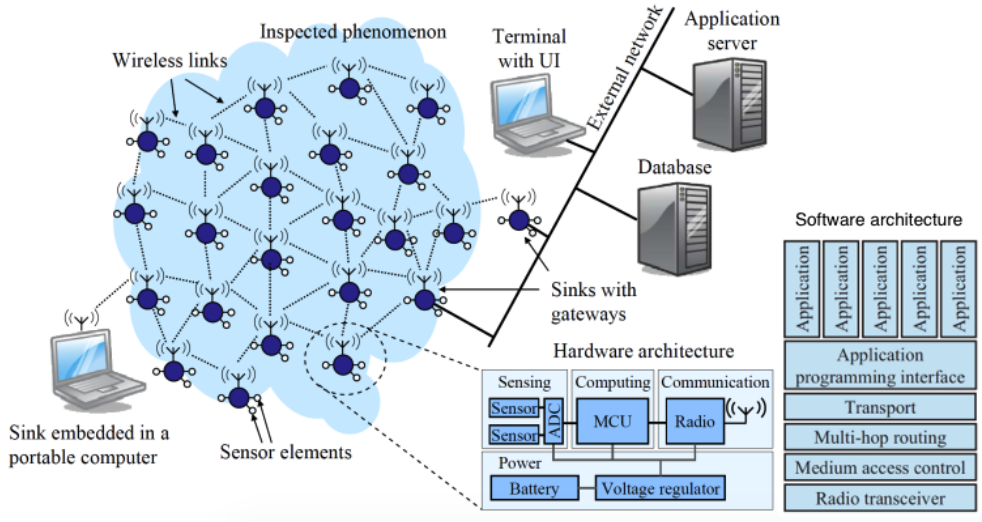


Figura 2.6: Red de sensores inalámbricos [**30**]

Una red de sensores inalámbricos (WSN) es un grupo de nodos (motas) de sensores pequeños, de bajo costo y de baja potencia que monitorean de forma cooperativa las cantidades físicas y actuadores de control. Una red puede estar formada por miles de nodos auto-configurados y desplegados aleatoriamente que funcionan de forma autónoma para formar una topología multi-salto.

Las redes de sensores inalámbricos son una nueva tecnología en investigación y presente en la industria, con un alto potencial para numerosas aplicaciones. Las redes de sensores inalámbricas se prevé que serán una tecnología que permite obtener redes ubicuas, donde la potencia de computo se encaja invisiblemente alrededor nuestro y se accede a través de interfaces inteligentes. La red de sensores inalámbricos se compone de numerosos nodos que funcionan de manera totalmente autónoma en la interacción con su entorno, y ejecutan tareas de detección, informática, actuación y comunicación. En la Figura 2.6 se observa con facilidad todos los elementos que pueden estar presentes en una red de sensores inalámbricos [**30**].

Durante la operación, los sensores serían difíciles o incluso imposibles de acceder y por lo tanto su red necesita funcionar de forma autónoma. Además, con el tiempo es posible que los sensores fallen (una de las razones es el agotamiento de la batería) y no pueden ser reemplazados. Por lo tanto, es esencial que los sensores aprendan unos sobre otros y se organicen por sí mismos en una red. Otro requisito crucial es que, puesto que los sensores pueden a menudo ser desplegados al azar (por ejemplo, simplemente

esparcidos desde un avión), para ser útiles, los dispositivos necesitan determinar sus ubicaciones. En ausencia de un control centralizado, todo este proceso de auto-organización debe llevarse a cabo de manera distribuida [**31**].

Una red de sensores inalámbricos (WSN), a veces denominada una red de sensores y actores inalámbricos (WSAN) son sensores autónomos distribuidos espacialmente para monitorear condiciones ambientales físicas, tales como temperatura, sonido, presión, etc. y para pasar cooperativamente sus datos a través de la red a una localización principal. El desarrollo de redes de sensores inalámbricos fue motivado por aplicaciones militares como la vigilancia en el campo de batalla, hoy en día se utilizan en muchas aplicaciones industriales y de consumo, tales como la industria, monitoreo y control de procesos, monitoreo del correcto funcionamiento de las máquinas [**32**].

- Redes de Malla Inalámbricas (WMNs)

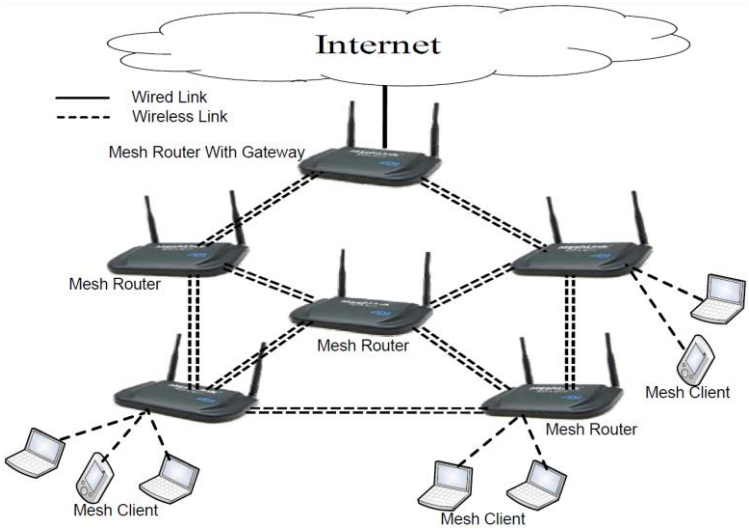


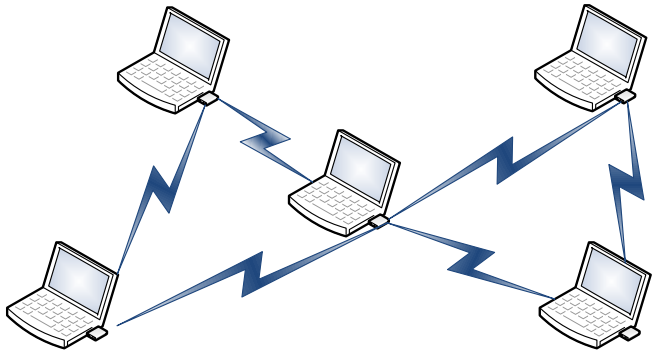
Figura 2.7: Red de malla inalámbrica [**33**]

Las redes de malla inalámbrica (WMN) consisten en encaminadores de malla que recopilan y reenvían el tráfico generado por los clientes de la malla. Los encaminadores de la malla suelen ser fijos y equipados con múltiples interfaces de radio. Los clientes de la malla son móviles y los datos son reenviados por encaminadores de malla al destino previsto. Uno o más encaminadores de la malla pueden tener funcionalidad de puerta de enlace y proporcionar conectividad a otras redes como acceso a Internet. En la Figura 2.7 se observa una sola puerta de enlace que permite la conexión a Internet de los 7 dispositivos móviles inalámbricos. En las redes de malla inalámbricas, la mayoría de

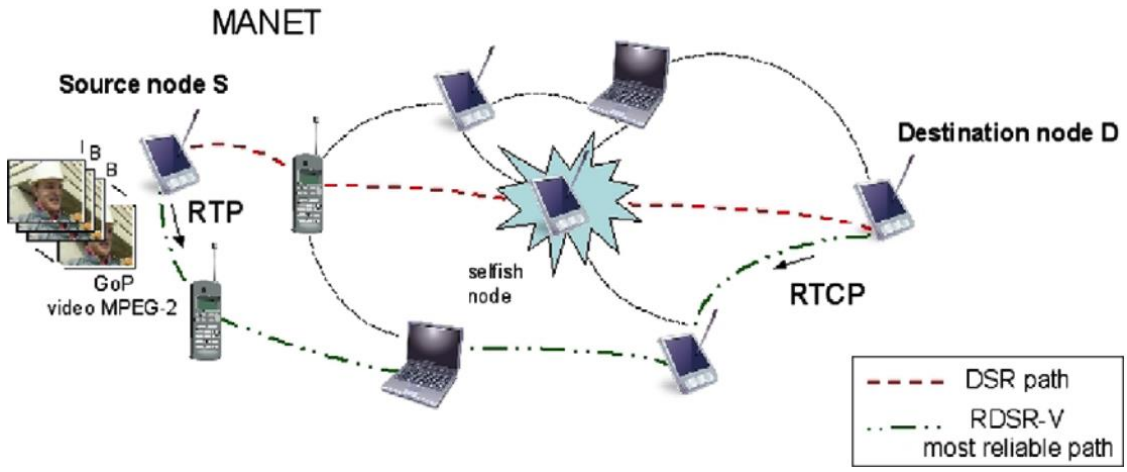
los flujos están entre el cliente de malla y la puerta de enlace, este tipo de tráfico se llama tráfico de Internet, que es el tráfico de red de malla inalámbrico común que los usuarios necesitan para acceder a los recursos cableados [**33**].

En redes de malla inalámbricas, los servicios de transporte de información se construyen sobre un conjunto de nodos ubicados arbitrariamente, que son posiblemente móviles. Cada nodo se comporta tanto como un host móvil como un encaminador inalámbrico. Hay muchas aplicaciones para tales redes, tales como proporcionar servicios de comunicación en situaciones de emergencia como en áreas afectadas por tormentas, inundaciones y terremotos. Una red de malla inalámbrica también puede proporcionar conectividad a las flotas de vehículos que operan en áreas sin infraestructura de red. Por supuesto, también hay muchas aplicaciones militares. En todas estas aplicaciones, podemos identificar un conjunto de flujos de paquetes punto a punto entre los nodos de la red con cada flujo de paquetes que tiene su propio requisito de calidad de servicio, por ejemplo, un requisito de rendimiento mínimo y, posiblemente, un promedio de requerimiento de retardo de paquete extremo a extremo [**31**].

- Redes Móviles Ad-Hoc (MANETs)



(a)



(b)

Figura 2.8: Red móvil ad-hoc: (a) formada por un conjunto de laptops [**16**], (b) formada por laptops y teléfonos inteligentes, donde se compara el camino seleccionado por DSR, con el seleccionado por RDSR-V [**11**]

Las redes ad-hoc móviles (MANETs) son redes dinámicas y sin infraestructura que consisten en una colección de nodos móviles inalámbricos que se comunican entre sí sin el uso de ninguna autoridad centralizada. Debido a sus características fundamentales, como el medio inalámbrico, la topología dinámica y la cooperación distribuida, las redes móviles ad-hoc son vulnerables a diversos tipos de ataques de seguridad como agujero de gusano, agujero negro, ataque apresurado, etc. [**34**]. En la Figura 2.8 (a) y (b) se observa una característica principal, y es que cada nodo es encaminador y terminal a la vez.

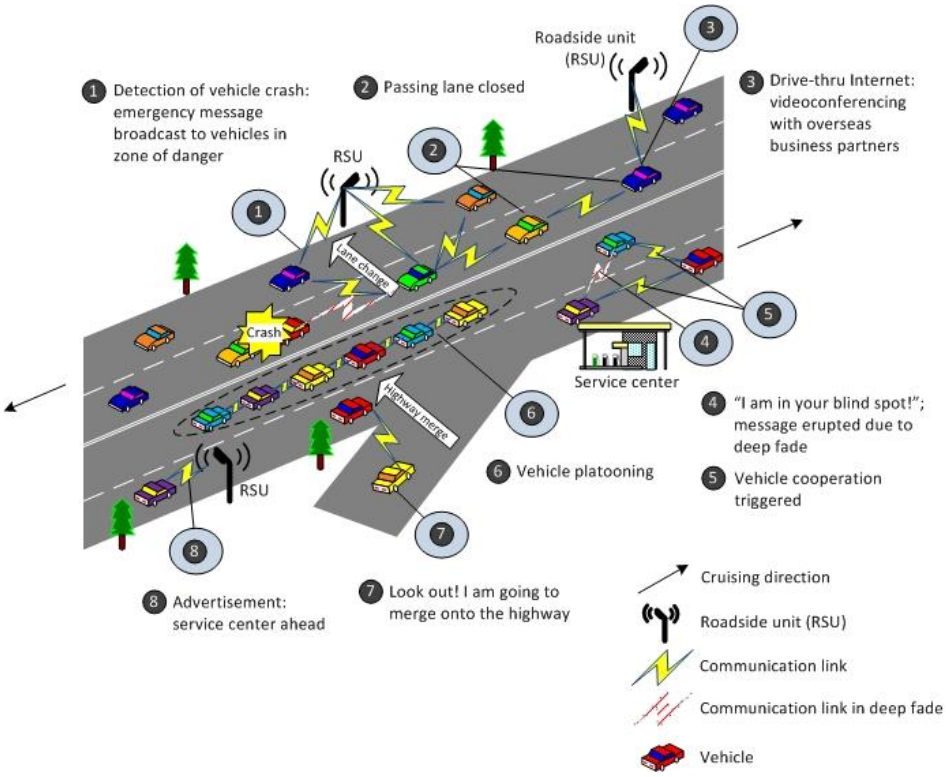


Figura 2.9: Red vehicular ad-hoc [**35**]

La Figura 2.9 muestra una red vehicular ad-hoc que es una variante de red móvil ad-hoc, y una tecnología emergente para futuras comunicaciones en carretera. Debido a la virtud de las comunicaciones vehículo a vehículo y vehículo a infraestructura, se espera que las redes vehiculares ad-hoc (VANET) permitan una gran cantidad de aplicaciones automovilísticas basadas en la comunicación, incluyendo diversas aplicaciones de infotainment en vehículos y servicios de seguridad vial [**35**].

Las redes vehiculares ad-hoc (VANETs) son una variante de red móvil ad-hoc que se crean a partir del concepto de establecer una red de vehículos para una necesidad o situación específica. Las redes móviles ad-hoc han sido establecidas como redes confiables que los vehículos utilizan para propósitos de comunicación en carreteras o entornos urbanos. Junto con los beneficios surgen un gran número de desafíos tales como aprovisionamiento de calidad de servicio, alta conectividad, ancho de banda, seguridad para vehículos y privacidad individual [**36**].

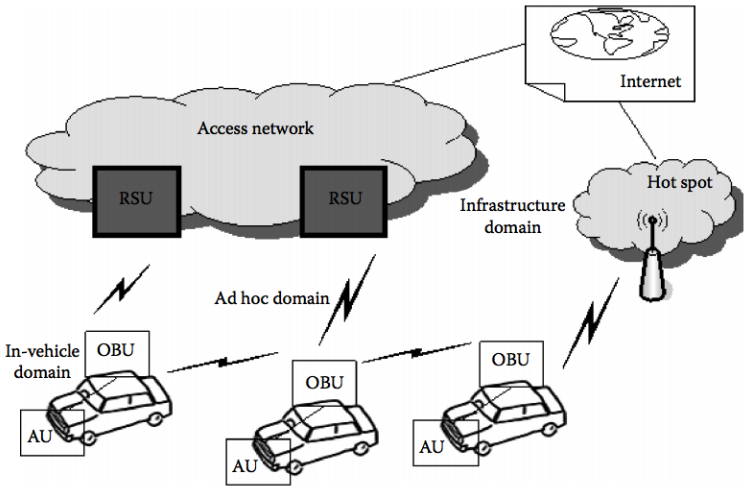


Figura 2.10: Arquitectura de referencia C2C-CC [**37**]

Se propone una arquitectura de referencia para redes vehiculares dentro del C2C-CC (Car-to-Car Communication Consortium), distinguiendo entre tres dominios: dentro del vehículo, ad-hoc y dominio de infraestructura [**38**]. La Figura 2.10 ilustra esta arquitectura de referencia. El dominio dentro del vehículo se refiere a una red local dentro de cada vehículo compuesta lógicamente de dos tipos de unidades: (i) una unidad a bordo (OBU) y (ii) una o más unidades de aplicación (AU). Una OBU es un dispositivo en el vehículo que tiene capacidades de comunicación (inalámbricas y / o cableadas), mientras que una AU es un dispositivo que ejecuta una única o un conjunto de aplicaciones mientras hace uso de las capacidades de comunicación del OBU. De hecho, una AU puede ser una parte integrada de un vehículo y estar permanentemente conectada a una OBU. También puede ser un dispositivo portátil, como una computadora portátil o PDA, que se puede conectar de forma dinámica (y desconectarse de) una OBU. La AU y la OBU generalmente están conectados con una conexión por cable, mientras que la conexión inalámbrica también es posible (utilizando, por ejemplo, Bluetooth, Wireless Universal Serial Bus o Ultra Wide Band). Esta distinción entre AU y OBU es lógica, y también pueden residir en una sola unidad física.

El dominio ad-hoc es una red compuesta por vehículos equipados con OBU y unidades a lado de la carretera (RSU) que están estacionarias a lo largo de la carretera. Las OBU de diferentes vehículos forman una red móvil ad-hoc (MANET), donde una OBU está equipada con dispositivos de comunicación, que incluyen al menos un dispositivo de comunicación inalámbrica de corto alcance dedicado a la seguridad vial. Los OBU y

RSU se pueden ver como nodos de una red ad-hoc, respectivamente, nodos móviles y estáticos. Una RSU se puede conectar a una red de infraestructura, que a su vez se puede conectar a Internet. Las RSU también pueden comunicarse entre sí directamente o mediante multihop, y su función principal es la mejora de la seguridad vial, ejecutando aplicaciones especiales y enviando, recibiendo o reenviando datos en el dominio ad-hoc.

Existen dos tipos de acceso a dominio de infraestructura: RSU y punto caliente (hot spot). Las RSU pueden permitir que las OBU accedan a la infraestructura y, en consecuencia, se conecten a Internet. Las OBU también pueden comunicarse con Internet a través de puntos calientes públicos, comerciales o privados (puntos de acceso Wi-Fi). En ausencia de RSU y puntos calientes, las OBU pueden utilizar las capacidades de comunicación de las redes de radio celular (GSM, GPRS, UMTS, WiMax y 4G) si están integradas en la OBU [**37**].

2.4 Protocolos de encaminamiento en redes móviles ad-hoc

En contraste con las redes basadas en infraestructura, en las redes móviles ad-hoc, todos los nodos son móviles y pueden ser enlazados dinámicamente de manera arbitraria. Todos los nodos de una red ad-hoc pueden actuar como encaminadores y pueden manejar el descubrimiento y el mantenimiento de caminos en la red. La topología de la red es en general dinámica, debido a que la conectividad entre los nodos puede variar en el tiempo debido al movimiento del nodo. Por lo tanto, hay una necesidad de protocolos de encaminamiento eficientes para permitir a los nodos comunicarse a través de caminos de múltiples saltos.

2.4.1 Conceptos generales sobre protocolos de encaminamiento en redes móviles ad-hoc

Un protocolo define el formato y el orden de los mensajes intercambiados entre dos o más entidades que se comunican, así como las acciones que se toman en la transmisión y/o recepción de un mensaje u otro evento. Llegar a dominar el campo de las redes es equivalente a comprender el qué, el por qué y el cómo de los protocolos de red [**39**].

Algunas de las cualidades deseables de los protocolos de encaminamiento dinámico para las redes ad-hoc son los siguientes:

- *Reducción de la sobrecarga de encaminamiento:* Los cambios de seguimiento de la topología de red requieren el intercambio de paquetes de control entre los nodos móviles. Estos paquetes de control deben llevar varios tipos de información, tales como identidades de nodo, lista de vecinos, métricas de distancia, y así sucesivamente, que consumen más ancho de banda para la transmisión. Es deseable que el protocolo de encaminamiento minimice el número de tamaño de los paquetes de control utilizados en el seguimiento de las variaciones de la red.

- *Oportuna adaptación:* Dado que las rupturas de enlace ocurren en momentos aleatorios, es difícil predecir cuándo expirará un camino existente. La oportuna adaptación de los protocolos de encaminamiento es crucial. Un camino roto provoca interrupción en una comunicación hasta que se establece un nuevo camino. Idealmente, un nuevo camino debe ser determinado antes de que se rompa el existente, lo cual puede no ser posible. Alternativamente, se debe establecer un nuevo camino con un mínimo de retraso.

- *Caminos óptimos:* Con restricción en la sobrecarga de encaminamiento, los protocolos de encaminamiento para redes móviles ad-hoc están más preocupados por evitar interrupciones de comunicaciones entre los nodos de origen y de destino que por la optimización de los caminos. Por lo tanto, para evitar el exceso de transmisión de paquetes de control, se puede permitir que la red opere con caminos subóptimos (que no son necesariamente los más cortos) hasta que se rompen. Sin embargo, un buen protocolo de encaminamiento debe minimizar la sobrecarga, así como las longitudes de camino. De lo contrario, dará lugar a retrasos de transmisión excesiva y desperdicio de energía.

- *Libre de bucle:* Dado que los caminos se mantienen en una modalidad distribuida, la posibilidad de bucles dentro de un camino es una preocupación seria. El protocolo de encaminamiento debe incorporar características especiales para que los caminos permanezcan libres de bucles.

- *Baja complejidad de almacenamiento:* Otro problema de las arquitecturas de encaminamiento distribuido es la cantidad de espacios de almacenamiento utilizados para el encaminamiento. Las redes ad-hoc pueden aplicarse a dispositivos portátiles pequeños, como sensores, que tienen limitaciones severas en memoria y hardware. Por

lo tanto, es deseable que el protocolo de encaminamiento se diseñe para requerir una complejidad de almacenamiento baja.

- *Escalabilidad:* El protocolo de encaminamiento debe ser capaz de funcionar eficientemente incluso si el tamaño de la red se hace grande. Esto no es muy fácil de conseguir, ya que la determinación de un camino desconocido entre un par de nodos móviles se hace más costosa en términos del tiempo requerido, el número de operaciones y el ancho de banda gastado cuando aumenta el número de nodos [**16**].

Uno de los aspectos más interesantes del encaminamiento en las redes móviles ad-hoc, se refiere a si los nodos deben mantener un seguimiento de los caminos a todos los destinos posibles, o simplemente realizar un seguimiento de los destinos de interés inmediato [**40**]. Así, en la Figura 2.11 que se presenta a continuación, podemos distinguir los siguientes tipos de protocolos:

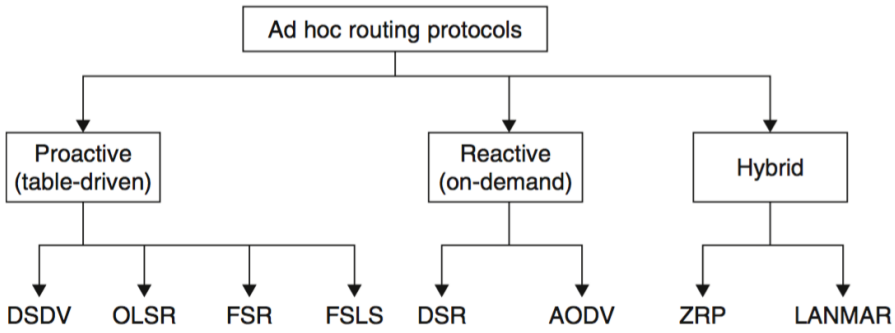


Figura 2.11: Clasificación y ejemplos de protocolos de encaminamiento ad-hoc [**16**]

Entre los mecanismos (o técnicas) para encontrar camino de un nodo fuente a un nodo destino que utilizan los protocolos de encaminamiento, tenemos: vector distancia, encaminamiento desde la fuente, estado de enlace, y encaminamiento salto a salto.

2.4.2 Protocolos de encaminamiento proactivos

Los protocolos de encaminamiento proactivo, también llamados controlados por tabla, utilizan transmisiones periódicas para hacer el seguimiento del conjunto de los vecinos. Cada vez que un nodo desea enviar paquetes a otros nodos, la red conoce el camino, debido a que la información del camino se almacena incluso antes de que se necesite. Sin embargo, los protocolos proactivos sufren de la desventaja del tráfico de control adicional que se necesita para actualizar continuamente las entradas de caminos dañados

[**40**]. Por otra parte, contienen numerosos nodos móviles, por lo que el riesgo de caminos rotos es frecuente. La reparación de caminos, incluso si no se utilizan, puede ser un esfuerzo inútil. A continuación se presenta algunos ejemplos de protocolos de encaminamiento ad-hoc proactivos.

Destination-Sequenced Distance-Vector (DSDV)

DSDV [**16**] se basa en el algoritmo clásico de Bellman-Ford con las adaptaciones que son apuntadas específicamente para las redes móviles [**41**]. El algoritmo de Bellman-Ford utiliza el enfoque de vector de distancia, donde cada nodo mantiene una tabla de encaminamiento que registra el "salto siguiente" para cada destino alcanzable a lo largo del camino más corto y la distancia mínima (número de saltos). Siempre que haya algún cambio en esta distancia mínima, la información es reportada a los nodos vecinos y las tablas se actualizan según sea necesario.

Para hacer este algoritmo adecuado para las redes móviles ad-hoc, DSDV agregó un número de secuencia con cada entrada de distancia para indicar cuan nueva es la entrada. Un número de secuencia se origina en el nodo de destino y es incrementado por cada nodo que envía una actualización a sus vecinos. Por lo tanto, una actualización de tabla de encaminamiento más reciente para el mismo destino tendrá un número de secuencia más alto. Las actualizaciones de la tabla de encaminamiento se transmiten periódicamente a través de la red, con cada nodo actualizando sus entradas de tablas de encaminamiento basadas en el número de secuencia más reciente correspondiente a esa entrada. Si dos actualizaciones para el mismo destino tienen números de secuencia idénticos pero distancias diferentes, entonces se registra la distancia más corta. La adición del número de secuencia elimina la posibilidad de bucles de larga duración y también el problema de "contar hasta el infinito", donde se necesita un gran número de mensajes de actualización para asegurarse de que un nodo no es accesible.

Optimized Link-State Routing Protocol (OLSR)

OLSR [**16**] es un protocolo de encaminamiento proactivo relativamente nuevo [**42**]. Es una adaptación del encaminamiento de estado de enlace convencional en el que cada nodo intenta mantener información sobre la topología de red. Cada nodo determina el coste del enlace para cada uno de sus vecinos difundiendo mensajes “hola” periódicamente. Cada vez que hay un cambio en los costes del enlace, el nodo transmite

esta información a todos los otros nodos. En algoritmos clásicos de estado de enlace, esto se hace por cada nodo que inunda toda la red con paquetes de actualización que contienen los costos actualizados del enlace. Los nodos utilizan esta información para aplicar un algoritmo de trayectoria más corta (como el algoritmo de trayecto más corto de Dijkstra [**43**]) para determinar el mejor camino a un destino específico.

OLSR optimiza el protocolo de estado de enlace de dos maneras. En primer lugar, reduce el tamaño de los paquetes de actualización enviados durante la transmisión, incluyendo sólo un subconjunto de enlaces a sus vecinos. Estos son los enlaces a un conjunto selecto de vecinos conocidos como los relés multipunto (MPR). El conjunto de MPRs de un nodo consiste en el conjunto mínimo de vecinos de un salto de ese nodo para que el nodo pueda alcanzar todos sus vecinos de dos saltos utilizando estos nodos como puntos de retransmisión. Cada nodo calcula su conjunto MPR a partir del intercambio de información de vecindad con todos sus vecinos.

En segundo lugar, en vez de que cada vecino difunda los paquetes de actualización enviados por un nodo, sólo los nodos MPR participan en la difusión de estos paquetes en OLSR. Esto minimiza el tráfico de paquetes de control durante las inundaciones. Sin embargo, el ahorro de ancho de banda logrado mediante estas dos técnicas tiene un costo de propagación de información de topología incompleta en la red. Las actualizaciones incluyen sólo conjuntos MPR y no los conjuntos de todos los vecinos de los nodos de difusión.

Por lo tanto, un algoritmo de trayecto más corto basado en esta información de topología parcial generará caminos que contienen los nodos MPR solamente. Cuando la red es densa, es decir, cuando cada nodo tiene muchos vecinos, OLSR funcionará para ser eficiente debido a la reducción del tráfico de control para actualizaciones en la red.

Fisheye State Routing (FSR)

Se han propuesto varios enfoques nuevos para que los protocolos proactivos sean más escalables [**16**]. Un ejemplo es Fisheye State Routing [**44**], que es también una adaptación de encaminamiento de estado de enlace a redes ad-hoc. FSR intenta limitar la carga de encaminamiento evitando inundar la red con información de encaminamiento. Toda la información del estado del enlace sólo se transmite a los vecinos del primer salto. Además, utiliza tasas de actualización más bajas para los

nodos situados más lejos. Por lo tanto, FSR mantiene la información exacta del camino en los nodos que están cerca, pero la exactitud se degrada con el aumento de la distancia del destino desde la fuente. En general, esta técnica ahorra el volumen y el tamaño del tráfico de encaminamiento.

Fuzzy Sighted Link-State (FSLS)

Un enfoque similar al anterior se adopta en el algoritmo FSLS [**45**]. Se considera una clase de enfoque que intentan escalar el encaminamiento de estado de enlace limitando el alcance de la difusión de actualización en el espacio y en el tiempo. Esta clase abre un nuevo espacio de diseño, ya que no es global ni local; representando una nueva forma de pensar donde cada nodo puede tener una visión diferente de la red. Utiliza una perspectiva novedosa de la "sobrecarga" de un protocolo que incluye no sólo la sobrecarga debido a los mensajes de control, sino también debido a lo subóptimo del camino.

2.4.3 Protocolos de encaminamiento reactivos

Los protocolos de encaminamiento reactivos, también llamados en demanda, han sido diseñados para que la información de encaminamiento sea adquirida sólo cuando es necesaria. A menudo utilizan menos ancho de banda para el mantenimiento de las tablas de caminos en cada nodo [**40**]. Y un nodo intermedio que ha intercambiado datos con el destino poco antes del descubrimiento de camino puede proveer información de la petición, es decir, el camino hacia el destino. A continuación se presenta algunos ejemplos de protocolos de encaminamiento ad-hoc reactivos.

Dynamic Source Routing (DSR)

DSR [**16**] [**46**] es un protocolo de encaminamiento reactivo que utiliza un concepto llamado encaminamiento de origen. Cada nodo mantiene una cache de caminos donde lista los caminos completos a todos los destinos para los cuales se conocen los caminos. Un nodo de origen incluye el camino a seguir por un paquete de datos en su encabezado. Los caminos son descubiertos a petición por un proceso conocido como descubrimiento de camino. Cuando un nodo no tiene una entrada de caché de camino para el destino al que necesita enviar un paquete de datos, inicia un descubrimiento de camino difundiendo una petición de camino (RREQ) o consulta buscando un camino

hacia el destino. El paquete de petición de camino contiene las identidades de la fuente y el destino deseado. Cualquier nodo que reciba un paquete de petición de camino comprueba primero su caché de caminos para buscar una entrada existente al destino deseado. Si no tiene tal entrada, el nodo añade su identidad al encabezado del paquete de petición de camino y lo transmite. Eventualmente, el paquete de petición de camino inundará toda la red atravesando todos los nodos rastreando todos los caminos posibles. Cuando un paquete de petición de camino llega al destino, o un nodo tiene un camino conocido al destino, se envía una contestación de camino (RREP) a la fuente siguiendo el mismo camino que fue atravesado por ese paquete de petición de camino en la dirección inversa. Esto se hace simplemente copiando el secuenciador de las identidades de nodo obtenidas de la cabecera del paquete de petición de camino. El paquete de contestación de camino contiene todo el camino hacia el destino, que se registra en la caché de caminos del nodo fuente.

Cuando un camino existente se rompe, se detecta por el fallo del reenvío de paquetes de datos en el camino. Este fallo se observa por la ausencia de los acuses de recibo de la capa de enlace esperados por el nodo donde se ha producido la falla del enlace. Al detectar el fallo del enlace, los nodos devuelven un paquete de error de camino (RERR) al origen. Todos los nodos que reciben el paquete de error de camino, incluida la fuente, eliminan todos los caminos existentes de sus caches de caminos que contienen el enlace especificado. Si todavía se necesita un camino, un nuevo descubrimiento de camino es iniciado.

Ad-hoc On-demand Distance Vector (AODV)

AODV [**16**] [**47**] puede describirse como una extensión bajo demanda del protocolo de encaminamiento DSDV. Al igual que DSDV, cada camino mantiene tablas de encaminamiento que contienen el siguiente salto y números de secuencia correspondientes a cada destino. Sin embargo, los caminos se crean a petición, es decir, sólo cuando se necesita un camino para el que no hay un registro "reciente" en la tabla de encaminamiento. Con el fin de facilitar la determinación de lo reciente que es la información de encaminamiento, AODV mantiene el tiempo desde que se utilizó por última vez una entrada. Una entrada en la tabla de encaminamiento está "expirada" después de un determinado umbral predeterminado de tiempo.

El mecanismo para crear caminos en AODV es algo diferente de aquel utilizado en DSR. Aquí, cuando un nodo necesita un camino a algún destino, emite un paquete de petición de camino (RREQ) en el que incluye el último número de secuencia conocido para ese destino. El paquete de petición de camino es reenviado por todos los nodos que no tienen un camino reciente (determinada por los números de secuencia) al destino especificado. Mientras reenvía el paquete de petición de camino, cada nodo registra el salto anterior tomado por el paquete de petición de camino en su entrada de tabla de encaminamiento para el origen (originador del descubrimiento de camino). Por lo tanto, una propagación de los paquetes de petición de camino crea caminos inversos a la fuente en las tablas de encaminamiento de todos los nodos de reenvío. Cuando el paquete de petición de camino alcanza el destino deseado o un nodo que conoce un camino más reciente, genera un paquete de contestación de camino que se devuelve a lo largo del mismo camino que fue tomado por el paquete de petición de camino correspondiente. El paquete de contestación de camino contiene el número de saltos al destino, así como el número de secuencia más reciente. Cada nodo que envía el paquete de contestación de camino introduce la información de encaminamiento para el nodo de destino en su tabla de encaminamiento, creando así el camino hacia el destino.

Las entradas de la tabla de encaminamiento se eliminan cuando se recibe un paquete de error de camino (RERR) desde uno de los nodos intermedios del camino que envía un paquete de datos al destino. Cuando tal paquete de error de camino llega a la fuente, puede iniciar un descubrimiento de camino nuevo para determinar un camino nuevo hacia el destino.

2.4.4 Protocolos de encaminamiento híbridos

El uso de encaminamiento híbrido es un enfoque que se utiliza a menudo para obtener un mejor equilibrio entre la adaptabilidad a diferentes condiciones de red y la sobrecarga de encaminamiento. Estos protocolos utilizan una combinación de principios reactivos y proactivos, cada uno aplicado bajo diferentes condiciones, lugares o regiones. Por ejemplo, un protocolo de encaminamiento híbrido puede beneficiarse dividiendo la red en clústeres y aplicando actualizaciones de caminos proactivos dentro de cada clúster y encaminamiento reactivo a través de diferentes clústeres. También se han considerado los esquemas de encaminamiento que emplean el mantenimiento

proactivo de los caminos por encima de los descubrimientos de caminos reactivos [**16**]. A continuación se presenta algunos ejemplos de protocolos de encaminamiento ad-hoc híbridos.

Zone Routing Protocol (ZRP)

ZRP [**16**] [**48**] divide la red en zonas o grupos de nodos. Los nodos dentro de cada zona mantienen la información de encaminamiento entre sí utilizando un algoritmo proactivo tal como un vector de distancia o un protocolo de estado de enlace. Por lo tanto, todos los nodos mantienen tablas de encaminamiento actualizadas que consisten en caminos a todos los otros nodos dentro de la misma zona (conocidos como encaminamiento intrazona).

Cada zona también identifica un conjunto de nodos periféricos que están situados en los bordes de la zona para la comunicación con otras zonas. Cuando se envía un paquete a un nodo para el cual la fuente no tiene una entrada en su tabla de encaminamiento, se supone que el destino se encuentra en otra zona. En ese caso, el nodo solicita a los nodos periféricos que envíen un paquete de petición de camino a todas las demás zonas en las redes. Esto se conoce como encaminamiento inter-zona, que utiliza un proceso que es similar a DSR excepto que los paquetes de petición de camino sólo son manejados por los nodos periféricos de la red. Cuando el paquete de petición de camino llega a un nodo periférico de la zona que contiene el destino, se devuelve una respuesta al origen. La sobrecarga de inundación en tal descubrimiento de camino es limitada debido a la participación de nodos periféricos solamente. El protocolo proactivo en este marco híbrido limita la propagación de paquetes de actualización periódica dentro de cada zona. ZRP es especialmente adecuado para grandes redes; sin embargo, la inundación de los paquetes de solicitudes durante los descubrimientos de caminos inter-zonas puede ser motivo de preocupación.

Landmark Ad-hoc Routing Protocol (LANMAR)

LANMAR [**16**] está diseñado para redes ad-hoc que tienen las características de movilidad de grupo, como un grupo de soldados que se mueven juntos en un campo de batalla. Cada grupo identifica dinámicamente un nodo específico dentro del grupo para ser un nodo de referencia. Un protocolo de encaminamiento de estado de enlace

proactivo se utiliza para mantener la información de encaminamiento dentro del grupo y un algoritmo de vector de distancia se utiliza para hacer lo mismo entre todos los nodos de referencia. Por lo tanto, cada nodo tiene información de topología detallada para todos los nodos dentro del grupo, distancia e información vectorial de encaminamiento a todos los puntos de referencia. No se mantiene ninguna información de encaminamiento para los nodos que no son puntos de referencia pertenecientes a otros grupos. Los paquetes que se envíen a dicho destino se envían hacia el punto de referencia correspondiente. Cuando el paquete llega a los nodos dentro del grupo que contiene el destino, se envía al destino, posiblemente sin pasar por su punto de referencia. Este esquema reduce el tamaño de las tablas de encaminamiento, así como la sobrecarga del tráfico de encaminamiento que forma una jerarquía de encaminamiento de dos niveles. Por lo tanto, se espera que sea más escalable que los llamados protocolos de encaminamiento planos.

2.5 Protocolo DSR (*Dynamic Source Routing*) para redes móviles ad-hoc

Dynamic Source Routing es un protocolo simple y eficiente, diseñado específicamente para una utilización en redes móviles ad-hoc inalámbricas multisalto. Se basa en el concepto de encaminamiento de origen: el emisor de un paquete determina la secuencia completa de los nodos a través de los cuales se reenvía el paquete. Este protocolo también pertenece a los protocolos de encaminamiento en demanda, lo que significa que los caminos se crean sólo cuando sea necesario (en contraparte a los protocolos de encaminamiento controlados por tablas, donde todas las actualizaciones de caminos son mantenidas en cada nodo). El protocolo se compone de dos mecanismos, el de descubrimiento de camino (Route Discovery) y mantenimiento de camino (Route Maintenance), que trabajan juntos para permitir que los nodos descubran y mantengan caminos en la fuente, a los destinos en la red.

2.5.1 Descubrimiento de caminos

Descubrimiento de camino, es un mecanismo dinámico iniciado cuando un nodo, dispuesto a transmitir un paquete, no encontró ningún camino hacia el destino deseado en su memoria caché de caminos. Permite a un nodo descubrir dinámicamente un

camino a cualquier otro nodo en las redes ad-hoc a través de múltiples saltos intermedios.

Paquetes de petición de camino (RREQ)

Cuando se inicia un descubrimiento de camino, el nodo remitente transmite el paquete de datos con ayuda de una petición de camino (RREQ), que es recibida por los hosts dentro del rango de transmisión. Un paquete de petición de camino lleva la siguiente información:

- Dirección del iniciador del descubrimiento de camino (nodo origen).
- Dirección del destino del descubrimiento de camino (nodo destino).
- ID de petición, único y fijado por el iniciador.
- Registro de camino, listando las direcciones de cada nodo intermedio a través del cual la petición de camino ha sido reenviada.

Cuando la petición de camino es recibida por algún nodo de la red, los siguientes pasos son procesados por este nodo:

- Primero de todo, evitar la duplicación de peticiones de camino. El par (IP_Source, Request_ID) es chequeado con la lista de pares vistos recientemente para determinar si esta petición fue recientemente vista, de ser así, el paquete de petición de camino es descartado.

- Entonces, para evitar los bucles en los caminos, el nodo intermedio chequea si su propia dirección está ya listada en el registro de camino de los paquetes de petición de camino, si es así el paquete es descartado.

- Entonces, si la dirección del host es el destino, o si los nodos intermedios conocen un camino no expirado hacia el host destino, entonces un paquete de contestación de camino (RREP) es generado. El camino al destino es establecido con la secuencia de saltos.

- De otra manera, se añade su propia dirección para el registro de camino.

Este proceso permite al paquete de petición de camino propagarse a través de la red ad-hoc hasta el destino, o a un nodo que conoce un camino al destino en su memoria cache de caminos. Este proceso previene los bucles y la duplicación de las peticiones.

Paquetes de contestación de camino (RREP)

Una contestación de camino es generada para ser enviada de regreso al nodo fuente. Luego de alcanzar el destino, el registro de camino de la petición de camino se encuentra en la contestación de camino, de otra forma, en el caso del nodo intermedio, la memoria cache de caminos del nodo y el registro de camino de la petición de camino son concatenados y localizados en la contestación de camino. La contestación de camino tiene que retornar a la fuente para entregar el descubrimiento del camino. Entonces la contestación de caminos puede estar retornando de varias formas:

- El nodo destino puede realizar su propio descubrimiento de camino para la fuente, pero para evitar posibles recursiones infinitas de descubrimientos de caminos, este debe acarrear al final de la contestación de camino, su propio mensaje la petición de camino hacia la fuente.

- El nodo destino puede también simplemente invertir la secuencia de saltos en el registro de caminos que este está intentando enviar, y utilizar esto como el camino fuente. Esta forma de responder puede ser utilizada sólo si todos los enlaces son bidireccionales.

Características adicionales del descubrimiento de caminos

Respondiendo a peticiones de camino utilizando caminos de memoria caché

Un nodo recibe una petición de camino, el cuál no es el objetivo que se busca, si el nodo encuentra un camino en su cache, este genera una contestación de camino, y en la contestación de camino, el nodo agrega su secuencia conocida de saltos necesaria para alcanzar el objetivo, esto lo podemos apreciar en la Figura 2.12.

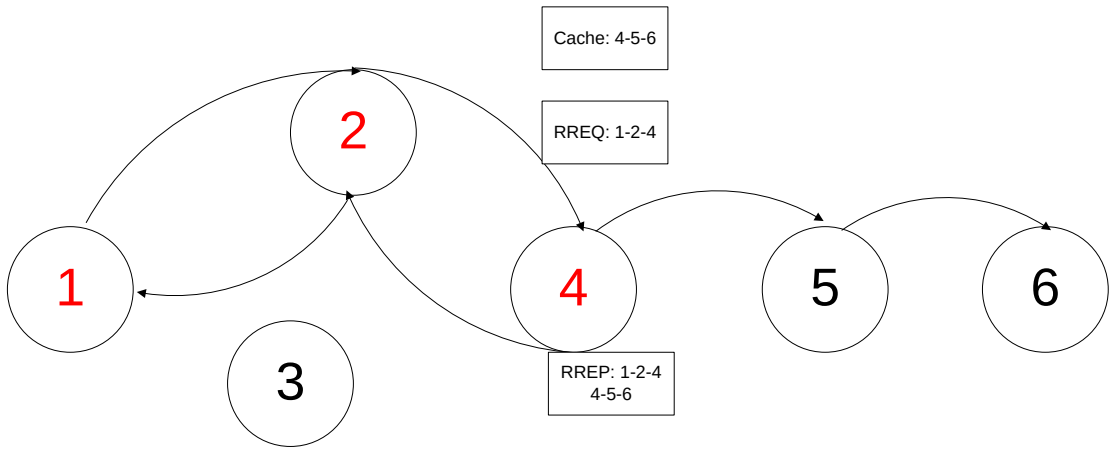


Figura 2.12: Descubrimiento de camino (1-6) utilizando caminos de memoria cache (4)

Información de encaminamiento, sobrecarga de almacenamiento en cache

Un nodo escuchando algún paquete puede agregar la información de encaminamiento a su propia memoria cache. Particularmente, el camino fuente utilizado en un paquete de datos, el registro de camino acumulado en una petición de camino (RREQ), o el camino que está siendo retornado en una contestación de camino (RREP), pueden todas ser almacenadas en cache por algún nodo [**40**].

2.5.2 Almacenamiento de caminos en memoria caché

La capacidad de memoria caché es también importante para la elección en el diseño de una estrategia de caché para el protocolo DSR. Para la memoria caché de enlaces, puede almacenar todos los enlaces que se descubren en la caché, ya que hay enlaces NxN máximos fijos que pueden existir en una red ad-hoc con N nodos, estos caminos almacenados requieren menos espacio de almacenamiento. Mientras que en la memoria caché de caminos, el espacio máximo de almacenamiento se requiere que sea mucho más grande, ya que un camino se almacena por separado. La capacidad de memoria caché se puede dividir en dos mitades: la primera mitad llamada "memoria caché primaria", que representa caminos que han sido utilizados por el nodo actual, la segunda mitad llamada "memoria caché secundaria", que representa caminos que aún no han sido utilizados. Cuando el nodo de origen intenta agregar un nuevo camino de acceso que ha aprendido y aún no se ha utilizado en la memoria caché, los caminos antiguos de la caché secundaria se eliminan debido a la limitación de la capacidad de memoria

caché en la red. Mientras que con la memoria caché primaria, los caminos antiguos se eliminan más activamente debido a la operación de mantenimiento de camino [**49**].

2.5.3 Envío de paquetes de datos

En esta investigación hemos utilizado CBR (Constant Bit Rate) como modelo de tráfico para la generación de los paquetes de datos. La codificación de velocidad de bits constante significa que la velocidad a la que los datos de salida de un códec deben consumirse es constante. Desde capa de red únicamente se distinguen direcciones IP, pero vale recalcar que únicamente para la transmisión de los paquetes de datos se abre un puerto específico tanto en el transmisor como en el receptor, paquetes de petición de camino, de contestación de camino o paquetes de error no abren ningún puerto, únicamente trabajan en capa de red.

2.5.4 Mantenimiento de caminos

Cuando se genera un paquete utilizando un camino fuente, cada nodo que está transmitiendo es responsable por confirmar si los datos pueden fluir sobre el enlace desde ese nodo al siguiente salto. El mantenimiento de camino es un procedimiento, el cual es responsable de informar al transmisor de algún error de encaminamiento.

Esto es realizado a través del uso de paquetes de error de camino y acuses de recibo [**46**]. A continuación, la Figura 2.13 ilustra un ejemplo del mecanismo de mantenimiento de camino.

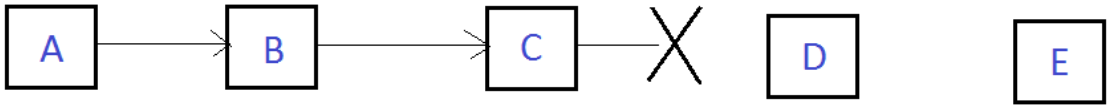


Figura 2.13: Ejemplo de mantenimiento de camino (el nodo C no está disponible para reenviar un paquete desde A hasta E sobre su enlace al siguiente salto, D) [**40**]

Acuses de recibo y paquetes de error

La detección del error de camino es basada en acuses de recibo salto a salto en la capa de enlace de datos para proveer una detección temprana y retransmisión de paquetes perdidos o dañados (corruptos). Los acuses de recibo son utilizados para chequear la correcta operación de los enlaces del camino. Si algún acuse de recibo no ha sido

recibido, el transmisor considera que el enlace al siguiente salto destino está “roto”, y retorna un error de camino al transmisor original del paquete. El error de camino está compuesto por la dirección del host que detecta el error y del host que debió recibir el paquete. Cuando un nodo recibe tal paquete de error de camino, este nodo actualiza su memoria caché de caminos truncando ese salto en todos sus caminos.

Características adicionales del mantenimiento de camino

Acortamiento automático de camino (ARS)

Una característica adicional del mantenimiento de camino es el proceso de acortamiento automático de camino (ARS), el cual permite un acortamiento automático de camino en uso cuando uno o más nodos intermedios en el camino no son necesarios [**46**]. Básicamente, esto permite a un nodo transmisor detectar que un camino corto existe mediante una contestación de camino gratuita (GRR). Para el ejemplo de la Figura 2.14 sucede lo siguiente:

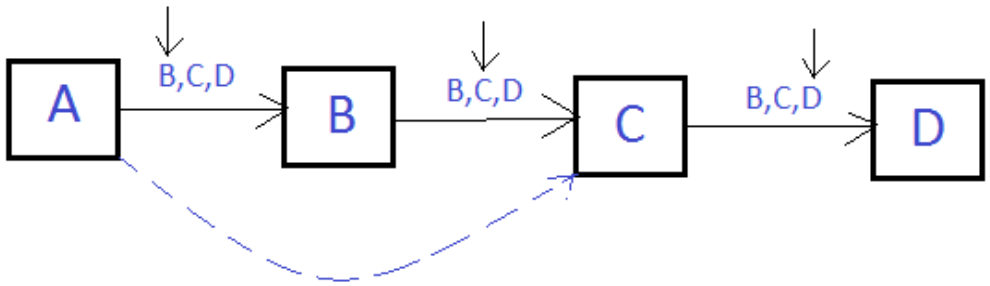


Figura 2.14: Mecanismo de acortamiento automático de camino (*ARS*)

- Se supone que el nodo A está transmitiendo al siguiente salto, el nodo B, posteriormente al nodo C y finalmente al nodo D.
- Se supone que el nodo C aprende un camino más directo.
- El nodo C envía una contestación de camino gratuita hacia A, y el nuevo camino tomado por el transmisor para el envío de paquetes de datos es A->C->D.

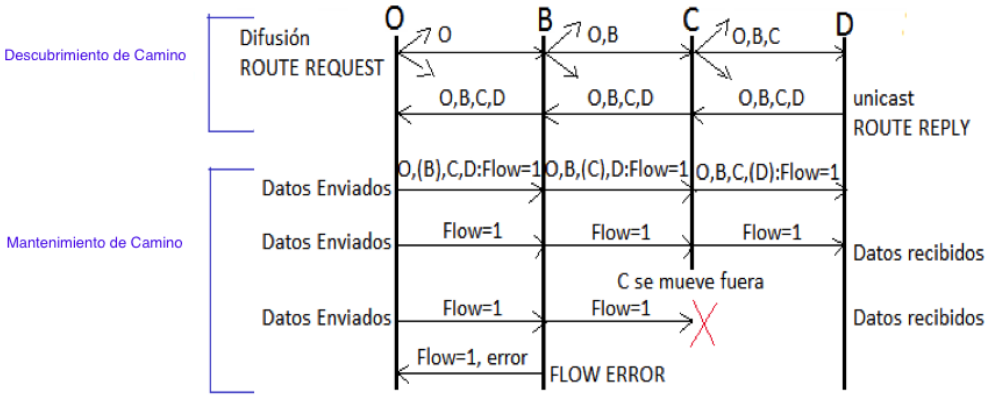


Figura 2.15: Mecanismo de Estado de Flujo [**50**]

En la Figura 2.15, se tiene una visión completa de los mecanismos de descubrimiento de camino y mantenimiento de camino, cuando se utiliza el mecanismo de estado de flujo (FSM). Es importante mencionar que es opcional el mecanismo de estado de flujo, al no utilizarlo se trabaja con caminos explícitos que aumentan la sobrecarga de los paquetes. En la Figura 2.15 se observa el descubrimiento de camino y mantenimiento de camino que son dos mecanismos principales de DSR, primeramente se difunde una petición de camino desde el nodo fuente, al dar tres saltos se llega al nodo destino donde se consigue la contestación de camino que va de regreso al nodo fuente. En los nodos que conforman el camino se instala el flujo con identificativo igual a 1 y se envían los paquetes de datos. Cuando el nodo intermedio B detecta que su enlace al nodo C está roto se elimina ese enlace de la caché de caminos. Luego se envía un paquete de error de camino desde el nodo B al nodo origen para que el origen pueda actualizar ese enlace y buscar un nuevo camino.

Paquete de salvamento

Se produce cuando un nodo intermedio en la transmisión de un paquete detecta a través del mantenimiento de camino, que el siguiente salto en el camino para ese paquete está roto. Si el nodo no tiene otro camino para el destino del paquete en su caché, el nodo debe salvar el paquete en lugar de descartarlo. Este proceso es interesante, ya que un paquete de datos es llevado con una petición de camino. Un paquete salvado, es un paquete donde el nodo sustituye al camino fuente original en el paquete con el camino desde su memoria cache. El nodo reenvía el paquete al siguiente nodo indicado a lo largo de este camino fuente. Con el fin de evitar problemas de salvamento infinito, un

paquete recuperado contiene un contador que limita el número de salvamentos. El proceso de salvamento debe ser procesado después de detectar un error de camino.

2.5.5 Mecanismos adicionales

La extensión de estado de flujo

El estado de flujo es una extensión (opcional) del protocolo DSR. Este permite el encaminamiento de la mayoría de los paquetes sin un camino fuente explícito en la cabecera del paquete. [**46**]. Esto se puede lograr cuando el iniciador de la transmisión de datos genera localmente un número de identificación único para el flujo. Entonces un flujo puede identificarse en la red gracias a la siguiente tripleta: (IP_source, IP_dest, Flow_ID*)*. Antes de utilizar un flujo para enviar los paquetes, el transmisor tiene que enviar un paquete con el encabezado de camino fuente y el encabezado de estado de flujo (Flow ID), con el fin de informar a todos los nodos de la correspondencia entre un flujo determinado y el camino fuente asociado al mismo. El estado de flujo DSR reduce la sobrecarga del protocolo DSR, pero conserva sus propiedades fundamentales.

Buffer de envío

El buffer de envío de un nodo, es una cola de paquetes que no pueden ser enviados por este nodo porque este no tiene aún algún camino fuente para alcanzar el destino del paquete.

Tabla de peticiones de camino

La tabla de peticiones de camino de un nodo implementa información acerca de las peticiones de camino que han sido recientemente originadas o reenviadas por este nodo. La tabla es indexada por direcciones IP. Esta contiene información tal como el número de descubrimientos de camino ya originado, el último tiempo de procesado del descubrimiento de camino y el identificador de la petición de camino (RREQ_ID) al igual que la dirección IP del iniciador (utilizada para enviar duplicación de peticiones de camino).

Tabla de flujo

La tabla de flujo registra información sobre los flujos desde los cuales los paquetes recientemente han sido enviados o reenviados por este nodo. La tabla es indexada por

una tripleta (IP Source Address, IP Destination Address, Flow ID), donde *Flow ID* es un token de 16 bits. El principal rol de esta tabla es configurar la correspondencia entre un *Flow ID* y su camino fuente asociado.

2.5.6 Implementación de DSR (*Dynamic Source Routing*) en NS-2.35 (*Network Simulator* Versión 2.35)

Paquete DSR

En *Network Simulator*, cuando hacemos una simulación con encaminamiento DSR, para cada nodo, es asignado un objeto responsable del comportamiento concerniente a los procesos de encaminamiento. Este objeto es actualmente una instancia de la clase *DSRAgent*, la cual puede ser vista consecuentemente como el gestor DSR en un nodo.

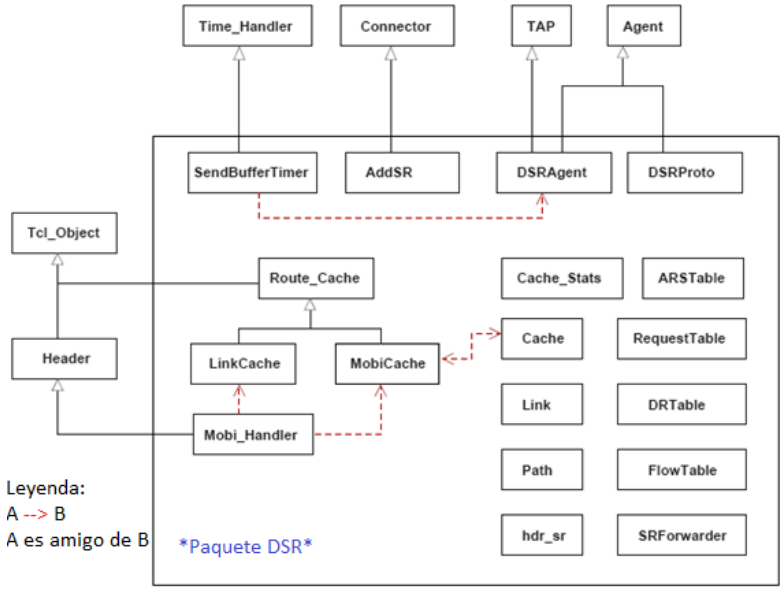


Figura 2.16: Diagrama de herencia para un paquete DSR [**51**]

Para tener una visión simplificada de todas las clases en el paquete, es posible referir el diagrama de herencia de la Figura 2.16.

En cuanto a A es amigo de B, quiere decir que las clases A y B, son amigas, y los amigos pueden tener acceso a todos los datos miembro y a todas las funciones miembro de una clase.

Para el propósito de esta tesis, no se necesita examinar el código del paquete entero. Es bien conocido que el punto de entrada real del paquete es la función DSRAgent::recv(Packet*, Handler*) el cual es llamado siempre que el nodo mueve un paquete arriba/debajo de la subcapa de encaminamiento (parte de la tercera capa en la pila del protocolo).

La Tabla 2.2, lista las clases y las estructuras que han sido estudiadas, y una explicación rápida para cada una de sus principales funcionalidades. La Tabla 2.3 muestra los archivos de la implementación de DSR en NS-2.

Clases	**Funcionalidades**
Punto de entrada del paquete DSR	
Agente DSR	Gestor de los paquetes recibidos. Llama el método correcto para la manipulación (handling) del paquete dependiendo de sus características
Herramientas de clase para gestión de caminos	
Struct ID	Caracteriza un nodo dado (dirección IP) y provee algunas funciones para la gestión de nodos en el paquete DSR
Path	Caracteriza un camino. Este puede ser visto como una lista de objetos "*struct ID*", con algunos métodos adicionales para la gestión de los caminos
Gestión de la cache	
RouteCache	Clase abstracta la cual declara el método virtual y el mecanismo básico requerido por todos los tipos de cache. En NS-2, esto es posible para seleccionar una de las dos siguientes caches: cache de enlace (Link Cache) o cache de caminos (Path Cache)
MobiCache	Implementación de cache de caminos, una cache de nodo donde los caminos completos conocidos desde el nodo a un destino dado son almacenados
LinkCache	Implementación de un link cache: una cache de nodo donde todos los

	enlaces conocidos existentes entre dos nodos dados son almacenados
Link	Caracteriza las entradas en la cache de enlace
Gestión del paquete	
hdr_sr	Esta es la cabecera DSR. En el código, esta clase es considerada como una estructura. Esto significa que es imposible definir métodos virtuales en algunas clases derivadas de esta
Struct SRPacket	Esta estructura tiene cuatro campos: un apuntador al paquete, el cual es actualmente manipulado, y otros tres campos concernientes al paquete: la dirección IP del destino y el camino fuente. Básicamente, esta clase está aquí para simplificar el tratamiento de los paquetes en el paquete DSR
Control de la propagación de peticiones de camino (RREQ)	
RequestTable	En esta tabla son almacenadas el par de identificación de las peticiones de camino (IP_source, Request_ID) que se tiene ya vistas

Tabla 2.2: Clases básicas en el paquete DSR [**51**]

add_sr.cc	-	Definición de la clase *addSR* + métodos asociados
dsr_proto.cc	dsr_proto.h	Definición de la clase *addSR* + métodos asociados
simplecache.cc	-	Definición de la clase *DSRProto* + métodos asociados
mobicache.cc	-	Una versión más completa del archivo previo
linkcache.cc	linkcache.h	Definición de la clase *Link* + *LinkCache* + métodos asociados
requesttable.cc	requesttable.h	Definición de la clase *RequestTable* + métodos asociados
flowstruct.cc	flowstruct.h	Definición de la clase *ARSTable* + *DRTable* +

		FlowTable + métodos asociados
sr_forwarded.cc	sr_forwarded.h	Definición de la clase *SRForwarded* + métodos asociados.
dsragent.cc	dsragent.h	Definición de la clase *SendBufferTimer* + *DSRAgent* + métodos asociados
routecache.cc	routecache.h	Definición de la clase *RouteCache* + métodos asociados
hdr_sr.cc	hdr_sr.h	Definición de la clase *hdr_sr* + métodos asociados
path.cc	path.h	Definición de la clase *Path* + métodos asociados
-	cache_stats.h	Definición de la clase *cache_stats* (los métodos asociados están en línea)
-	constants.h	Archivo general
-	srpacket.h	Archivo general

Tabla 2.3: Archivos que conforman DSR [**51**]

2.6 Sistemas de reputación en redes móviles ad-hoc

En [**52**] tenemos una amplia explicación sobre los sistemas de reputación, de la cual podemos rescatar algunas valiosas conceptualizaciones que se las expone a continuación.

La reputación es la opinión de una entidad sobre otra. En un contexto absoluto, es la fiabilidad de una entidad [**53**]. Confianza, por otra parte, es la expectativa de una entidad sobre las acciones de otra [**54**]. La tendencia es creer en personas que tienen una buena reputación, la tendencia es a interactuar con dichas personas en la vida real. Entendemos como reputación, al historial del comportamiento de una persona. Tomando en cuenta la reputación, se construye la confianza. La expectativa de que una persona manifieste determinado comportamiento es la confianza [**52**].

Aunque ambos términos son muy parecidos, la diferencia radica en que confianza se refiere a una experiencia personal, es decir que es lo que yo pienso de alguien. En tanto

que reputación es que piensan los demás de mí, es el concepto o experiencia que los demás tienen de mí. El concepto de reputación de la vida real aplicado a personas, en esta investigación se lo traslada a redes móviles ad-hoc, con la finalidad de detectar los nodos más cooperativos de la red y poder encargarles la retransmisión de paquetes de datos a lo largo de un camino.

2.6.1 Definición de comportamiento anómalo de un nodo

El comportamiento no cooperativo intencional de un nodo, como se lo identifica en la referencia [**55**], es principalmente causado por dos tipos de malos comportamientos: comportamiento egoísta, por ejemplo, nodos que quieren ahorrar potencia, ciclos de CPU, y memoria. Y comportamiento malicioso, el cual no se preocupa principalmente de la potencia o algún otro ahorro, pero sí interesados en atacar y dañar la red. Cuando el mal comportamiento de un nodo se manifiesta como egoísmo, el sistema puede aún arreglárselas con esto porque este mal comportamiento puede ser previsto. Cuando el mal comportamiento se manifiesta como malicioso, es difícil para el sistema arreglárselas con esto, un nodo malicioso siempre intentará maximizar el daño causado al sistema, incluso a costa de su propio beneficio.

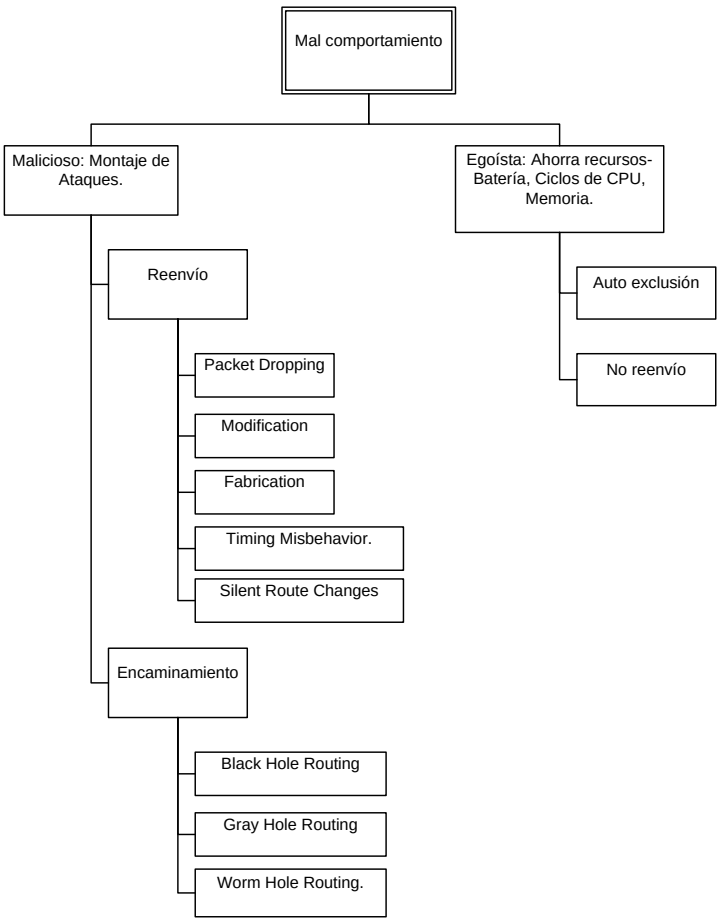


Figura 2.17: Mal comportamiento del nodo

Mal comportamiento malicioso en reenvío de paquetes, es dividido generalmente en dos tipos de mal comportamiento: reenvío y encaminamiento. Algunos malos comportamientos de reenvío comunes son: caída de paquetes (packet dropping), modificación (modification), fabricación (fabrication), ataques de tiempo (timing misbehavior) y cambios de camino en silencio (silent route changes).

Caída de paquetes (packet dropping), modificación (modification) y fabricación (fabrication) son auto explicados. Ataque de tiempo (timing misbehavior), es un ataque en el cual un nodo malicioso retarda el reenvío de paquetes para asegurar que los paquetes expiran su tiempo de vida (TTL), así que esto no es inmediatamente obvio, que esto está causando el problema. Cambios de camino en silencio (silent route changes) es un ataque en el cual un nodo malicioso reenvía un paquete a través de un camino diferente del que estaba planificado.

Un nodo malicioso con ataques de mal comportamiento en el encaminamiento durante la fase de descubrimiento de camino puede tener tres ataques comunes: agujero negro (black hole), agujero gris (gray hole), y agujero de gusano (worm hole). Un ataque agujero negro (black hole) es uno en el cual un nodo malicioso reclama tener el camino más corto y entonces cuando piden reenviar, perderá los paquetes recibidos. El ataque agujero gris (gray hole) es una variación de ataque agujero negro, el nodo malicioso selectivamente pierde algunos paquetes. El ataque agujero de gusano (worm hole, también conocido como tunnelling) es un ataque en el cual el nodo malicioso envía paquetes desde una parte de la red a otra parte de la red, donde estos son repetidos. El comportamiento egoísta de un nodo, como lo muestra la Figura 2.17, puede ser generalmente clasificado como auto-exclusión y no reenvío. El mal comportamiento egoísta de auto-exclusión es aquel en el cual un nodo egoísta no participa cuando el descubrimiento de camino es ejecutado. Esto asegura que el nodo es excluido desde la lista de encaminamiento de otros nodos. Esto beneficia a un nodo egoísta ya que ahorra su potencia, porque esta no es requerida para reenviar paquetes a otros nodos. Un modelo basado en reputación es una forma efectiva para prevenir las intensiones de tales nodos egoístas. Desde que un nodo no reenvía paquetes para otros nodos en la red, es denegada alguna cooperación por otros nodos. Por otra parte en el mal comportamiento egoísta de no reenvío de un nodo, el nodo participa completamente en la fase de descubrimiento de camino, pero rechaza reenviar paquetes para otros nodos en un tiempo posterior.

Desde que los sistemas basados en reputación pueden arreglárselas con algún tipo de mal comportamiento observable, son útiles en la protección de un sistema. Los investigadores han estado constantemente haciendo esfuerzos para modelos exitosos de redes de sensores inalámbricas (WSN) y redes móviles ad-hoc (MANETs) como también para sistemas basados en reputación y confianza. La Figura 2.18, presenta una breve introducción a los sistemas de reputación en redes móviles ad-hoc.

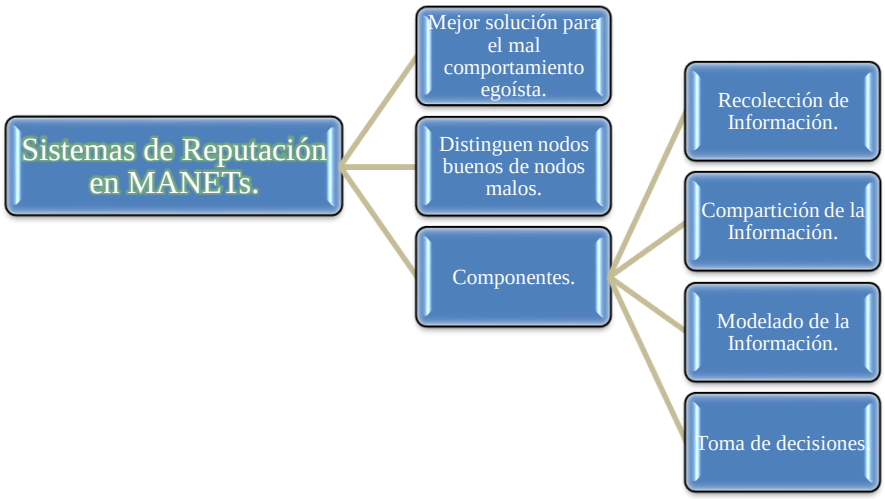


Figura 2.18: Sistemas de reputación en redes móviles ad-hoc

Se han propuesto técnicas para combatir el egoísmo de los nodos y tratar de incentivarlos a cooperar en la retransmisión de los paquetes. Tradicionalmente se han dividido en tres conjuntos principales: técnicas de reputación, técnicas de crédito y técnicas basadas en teoría de juegos. En las técnicas de reputación, cada nodo observa el comportamiento de sus vecinos en cuanto a si realizan o no las tareas que le son encomendadas, y utilizan esta información para distinguir si son nodos fiables y predecir su comportamiento futuro. Por otro lado los esquemas basados en crédito emplean una moneda virtual que puede corresponderse o no con un valor monetario real, para compensar a aquellos nodos que accedan a realizar la retransmisión de paquetes. Además, los nodos necesitan también este crédito para mandar sus propios paquetes. Finalmente, la teoría de juegos modela el proceso de retransmisión como un juego donde cada uno de los nodos, supuestamente racionales, modifica su estrategia para optimizar su propio beneficio, que es la transmisión de sus propios paquetes a costa del menor gasto de energía [**56**].

2.6.2 Protocolos de encaminamiento que utilizan sistemas de reputación

Adaptando sistemas basados en reputación y confianza a redes de sensores inalámbricas, se presentan grandes cambios con respecto a redes móviles ad-hoc y sistemas punto a punto (P2P) debido a sus limitaciones de energía. CORE [**57**], CONFIDANT [**58**], RFSN [**54**], DRBTS [**53**], KeyNote [**59**], and RT Framework [**60**] son unos pocos intentos exitosos. Sin embargo, RFSN y DRBTS sólo trabajan

enfocados en redes de sensores inalámbricos. Los otros se concentran en redes móviles ad-hoc y redes punto a punto.

2.6.3 Principales componentes de los sistemas de reputación

Hemos identificado cuatro componentes en los sistemas basados en reputación y confianza: recolección de información, compartición de información, modelado de información y toma de decisiones. Se analizará cada componente, presentando ejemplos de sistemas del mundo real cuando sea apropiado. En la Figura 2.19, se muestra estos cuatro componentes en resumen

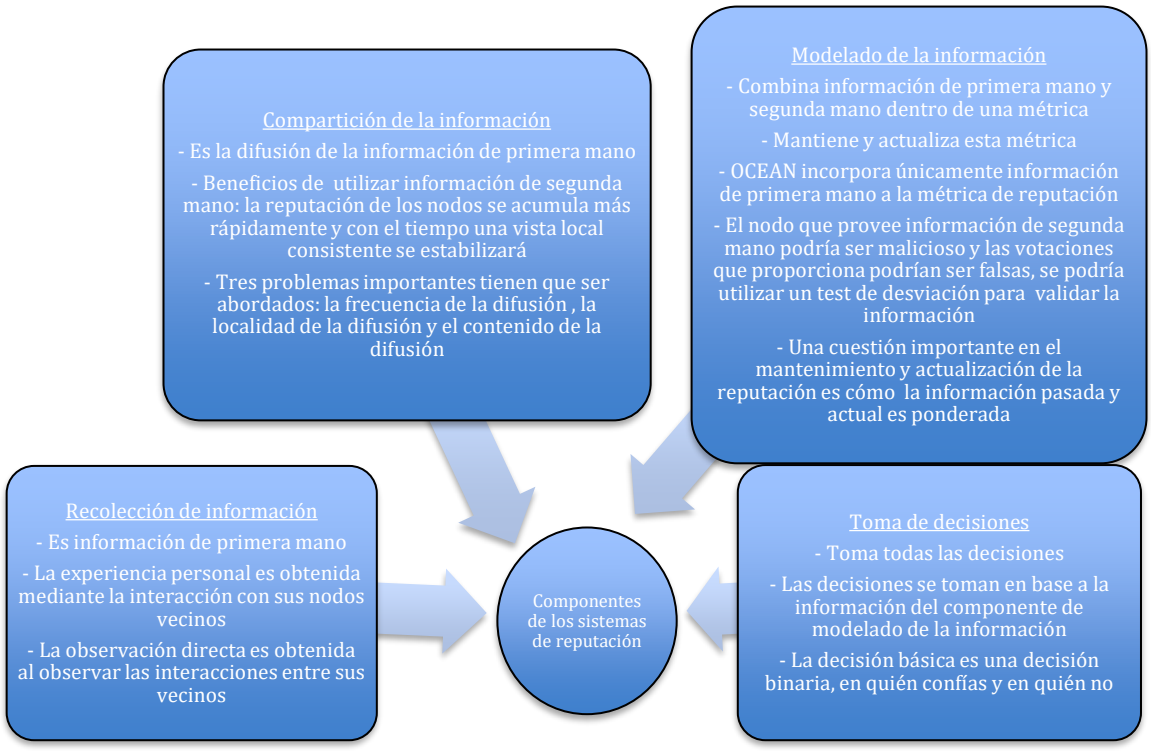


Figura 2.19: Componentes de los sistemas de reputación

Recolección de información

Recolección de información, es el proceso por el cual un nodo recolecta información acerca de los nodos que a este le interesan. Este componente de los sistemas basados en reputación y confianza se preocupa solamente de la información de primera mano. La información de primera mano es la información reunida por un nodo puramente basada en su observación y experiencia. Sin embargo acorde a CONFIDANT [**58**], la información de primera mano puede ser clasificada más a detalle en: experiencia

personal y observación directa. Experiencia personal del nodo se refiere a información que se reúne a través de cada interacción con sus nodos vecinos. Por otra parte observación directa se refiere a la información que un nodo reúne al observar las interacciones entre sus vecinos. CONFIDANT es actualmente el único sistema que hace esta distinción.

La mayoría de los sistemas basados en reputación y confianza hacen uso de un componente llamado *watchdog* [**61**] para monitorizar su vecindario y reunir información basada en observaciones múltiples. Así, información de primera mano es limitada al rango de recepción inalámbrico de un nodo. Sin embargo, el sistema *wathdog* no es muy efectivo en el caso de antenas direccionales, limitando su amplitud de aplicación, especialmente en situaciones de alta seguridad. Esta limitación ha recibido muy poco estudio actualmente.

Compartición de información

Este componente de los sistemas basados en reputación y confianza tiene que ver con la difusión de la información de primera mano recogida por los nodos. Hay una desventaja inherente entre la eficiencia en el uso de información de segunda mano y robustez frente a falsas acusaciones. Utilizando información de segunda mano se tiene muchos beneficios. La reputación de los nodos se acumula más rápidamente, debido a la capacidad de los nodos para aprender de los errores de los demás. No existe información en el sistema que no es utilizada. Utilizando información de segunda mano se tiene un beneficio adicional de que, con el tiempo, una vista local consistente se estabilizará.

Sin embargo, el intercambio de información tiene un precio. Este hace que el sistema sea vulnerable a ataques de informes falsos. Esta vulnerabilidad puede ser mitigada mediante la adopción de una estrategia de intercambio de información limitada, es decir, el intercambio de información sólo sea información positiva o información negativa.

La dificultad con la solución es que mientras se comparte sólo información positiva, limita la vulnerabilidad del sistema a los ataques de falsos elogios, tiene sus propios inconvenientes. Cuando se comparte sólo información positiva no se utiliza toda la

información en el sistema, ya que los nodos no pueden compartir sus malas experiencias. Esto es particularmente perjudicial ya que aprender de la propia experiencia en este escenario tiene un precio muy alto. Además, la complicidad de los nodos maliciosos puede prolongar el tiempo de los demás a través de los informes de falsos elogios. CORE [**55**] sufre de este ataque.

Del mismo modo, compartiendo sólo información negativa se impide el ataque de falsos elogios que hemos mencionado con anterioridad, pero tiene sus propios inconvenientes. No toda la información en el sistema se utiliza, ya que los nodos no pueden compartir sus nuevas experiencias. Más importante aún, los nodos pueden lanzar un fuerte ataque de montaje en los nodos benignos, ya sea individualmente, o en complicidad con otros nodos maliciosos. CONFIDANT [**58**] sufre de este ataque.

Otra forma de evitar las consecuencias negativas de compartir información, es no compartir información en absoluto. OCEAN [**62**] es un modelo que construye su reputación puramente basada en su propia observación. Estos sistemas, aunque son completamente robustos frente a la difusión de rumores, tienen algunos inconvenientes graves. El tiempo necesario para que el sistema construya una reputación es incrementado dramáticamente; y éste tarda más para una reputación a la baja, lo que permite que los nodos maliciosos permanezcan en el sistema por más tiempo.

Sistemas como DRBTS [**53**] y RFSN [**54**] comparten tanto información positiva como negativa. Los efectos negativos de compartir información, como se discutió anteriormente, pueden ser mitigados por una apropiada incorporación de información de primera mano y segunda mano dentro de la métrica de reputación. El uso de diferentes funciones de ponderación para información diferente es una técnica eficaz.

Con respecto a compartir información, se han identificado tres formas en las que la información puede ser compartida entre nodos: lista de amigos, lista negra, y una tabla de reputación. Una lista de amigos comparte sólo información positiva, una lista negra comparte sólo información negativa, mientras que la tabla de reputación comparte tanto información positiva como información negativa.

Para compartir información, tres problemas importantes tienen que ser abordados: la frecuencia de difusión, la localidad de la difusión, y el contenido de la difusión. La frecuencia de difusión puede ser clasificada en dos tipos:

- Difusión proactiva. En difusión proactiva, los nodos publican información durante cada intervalo de difusión. Incluso si no ha habido ningún cambio en los valores de reputación que almacena el nodo, este aún publica los valores de reputación. Este método es adecuado para redes densas para evitar la congestión, ya que tienen que esperar hasta el intervalo de difusión para publicar.

- Difusión reactiva. En la difusión reactiva, los nodos publican sólo cuando hay una cantidad predefinida de cambio de los valores de reputación que almacenan o cuando se produce un evento de interés. Este método reduce la sobrecarga de comunicación en situaciones donde no hay actualizaciones frecuentes en los valores de reputación. Sin embargo, la difusión reactiva puede causar congestión en redes densas con una elevada actividad de la red, o provocar que el sistema entre en pérdida si la actividad de red es especialmente baja.

En la difusión reactiva es muy común el término *piggybacking*, que es una técnica de transmisión de datos bidireccional en la capa de enlace de datos (Modelo OSI). Con esta técnica, en vez de enviar un acuse de recibo en un paquete individual, éste es incluido dentro del próximo paquete a enviar.

En ambos tipos de difusión de la información anteriormente revisados, la sobrecarga de comunicación puede ser reducida en gran medida por el *piggybacking* de la información con otro tráfico de red. Por ejemplo, en CORE, la información de reputación es cargada en el mensaje de respuesta, y en DRBTS es cargada en la información de ubicación de despacho. La difusión de la localización, puede ser clasificada en dos tipos:

- Local: En una diseminación local, la información es publicada con el vecindario. Podría ser a través tanto de una difusión local, como de una multidifusión, o una unidifusión. En DRBTS [**53**] , la información es publicada en el vecindario a través de una difusión local. Esto permite a todos los nodos baliza actualizar su tabla de reputación en consecuencia. Otros modelos pueden escoger unidifusión o multidifusión, dependiendo del dominio de la aplicación y los requerimientos de seguridad.

- Global: En la difusión global, la información es enviada a nodos fuera del rango del nodo de publicación. Como la diseminación local, la difusión global podría utilizar una de las tres técnicas de publicación: difusión, multidifusión o unidifusión. Para redes con movilidad de nodos, la difusión global es preferida, porque esta da a los nodos un entendimiento razonable de la nueva localización a la que se están moviendo. El contenido de la difusión de la información puede ser clasificada en dos tipos:

  - Raw. La información publicada por un nodo es sólo su información de primera mano. No refleja el valor total de la reputación, el cual incluye información de segunda mano de otros nodos en el vecindario. La difusión *raw* de información, previene información desde el bucle de regreso al nodo origen.

  - Procesado. La información publicada por un nodo es su opinión general de los nodos en sus tablas de reputación. Esto incluye la información proporcionada al nodo por otros en el vecindario.

Modelado de la información

Este componente de un sistema basado en reputación y confianza negocia con una combinación de información de primera mano e información de segunda mano, significativamente dentro de una métrica. También se ocupa de mantener y actualizar esta métrica. Algunos modelos optan por utilizar una única métrica, reputación, como CORE y DRBTS [**53**], [**55**], mientras que un modelo diferente, como RFSN puede optar por utilizar dos métricas separadas, la reputación y la confianza, para modelar la información [**54**]. Aunque la mayoría de los modelos hacen uso de información, tanto de primera mano como de segunda mano en actualización de reputación y/o confianza, algunos modelos como OCEAN [**62**] puede utilizar sólo información de primera mano. La información de primera mano puede ser directamente incorporada dentro de la métrica de reputación sin mucho procesamiento. Sin embargo, este no es el caso con información de segunda mano. El nodo que provee información de segunda mano podría ser malicioso, y las votaciones que proporciona podrían ser falsas. Por lo tanto, es necesario utilizar algunos medios de validación de la credibilidad de los informes del nodo. Un método consiste en utilizar un test de desviación como el de las referencias [**63**] y [**64**]. Si el informe del nodo califica la

prueba de desviación, entonces es considerado como digno de confianza y su información es incorporada para reflejar la reputación del nodo reportado. Sin embargo, los diferentes modelos pueden optar por utilizar una estrategia diferente para aceptar la información de segunda mano dependiendo del dominio de la aplicación y de los requerimientos de seguridad. Por ejemplo, el modelo en referencia utiliza la teoría de Dempster-Shafer [**64**] y el principio [**65**] para incorporar la información de segunda mano. Sin embargo, la distribución beta ha sido la más popular entre los investigadores en sistemas basados en reputación y confianza. Fue utilizado por primera vez por Josang e Ismail [**66**]. Desde entonces, muchos investigadores han utilizado la distribución beta incluyendo Ganeriwal y Srivastava [**54**] y Buchegger y Boudec [**67**]. La distribución beta es la más simple entre varios modelos de distribución, (como de Poisson, Binomial o Gaussiana) que pueden ser utilizadas para representación de la reputación. Esto es principalmente debido al hecho de que está indexado por sólo dos parámetros.

Una cuestión importante en el mantenimiento y actualización de la reputación es cómo la información pasada y actual es ponderada. Diferentes modelos tienden a ponderar de manera diferente, cada uno con una lógica diferente. Modelos como CORE tienden a dar más peso a las observaciones pasadas, con el argumento de que un reciente mal comportamiento esporádico debe tener una mínima influencia en la reputación de un nodo que se ha construido durante un largo período de tiempo. Esto ayuda a nodos benignos que son egoístas debido a la veracidad de condiciones de batería crítica.

También, ocasionalmente, los nodos pueden comportarse mal temporalmente debido a problemas técnicos tales como fallo de enlace, en cuyo caso se justifica para no castigarlos severamente.

Por otro lado, los modelos como RFSN tienden a dar más peso a las observaciones recientes que a las pasadas, con el argumento de que un nodo tiene que contribuir y cooperar de forma continua para sobrevivir. Esto se conoce como envejecimiento, donde las observaciones antiguas tienen poca influencia sobre el valor actual de la reputación de un nodo. Esto obliga a los nodos a cooperar en todo momento. De lo contrario, un nodo malicioso construirá su reputación durante un largo período y luego empezará a comportarse erráticamente, aprovechando de su bondad acumulada. Cuanto mayor sea la reputación que construye un nodo malicioso, más tiempo puede portarse

mal antes de que pueda ser detectado y excluido. Sin embargo, hay un inconveniente en la adopción de la técnica de envejecimiento. En períodos de baja actividad de red, un nodo benigno queda penalizado. Esto puede ser resuelto generando tráfico de red en regiones y períodos de actividad de red bajo, utilizando nodos móviles. DRBTS resuelve este problema con los nodos baliza siendo capaz de generar tráfico de red en función de las necesidades.

Toma de decisiones

Este componente de un sistema basado en reputación y confianza, es responsable de tomar todas las decisiones. Las decisiones hechas por este componente son basadas en la información proporcionada por el componente de modelado de la información. La decisión básica es una decisión binaria, en quién confías y en quién no. La decisión real podría traducirse a ser cooperar/no cooperar, retransmito/no retransmito, y así sucesivamente, basado en la función que está siendo monitoreada por el sistema.

La decisión de este componente varía junto con los valores de reputación y confianza en el componente de modelado de la información. La decisión puede variar desde confiable a no confiable, en la que en un nodo que se confió hasta el momento ya no se puede confiar después de que sus valores de reputación y confianza decaen por debajo de un umbral predeterminado. Del mismo modo, esto puede variar desde no confiable a confiable, en el que un nodo que inicialmente no fue confiable, será confiable poco después de que sus valores de reputación y confianza superen un determinado umbral.

2.6.4 Componentes presentes en REMODE_sw (Reputation for Mobile Devices Static Weight) y DrepSR (Dynamic Reputation Source Routing)

REMODE_sw (Reputation for Mobile Devices Static Weight) y DrepSR (Dynamic Reputation Source Routing) son algoritmos de protocolos de encaminamiento basados en reputación, y son el objeto de estudio de esta investigación. En ambos algoritmos tenemos la presencia de tres componentes bien diferenciados que son:

- Recolección de información: La recolección de información se la realiza desde capa de transporte mediante técnicas de cruce de capa, y se obtiene en capa de red la fracción de pérdida de paquetes de cada camino, que es la variable principal necesaria para calcular la reputación de los nodos.

- Modelado de la información: El modelado principalmente se lo realiza calculando los valores de reputación de los nodos mediante dos ecuaciones de media móvil exponencial.
- Toma de decisiones: Permite seleccionar los caminos de mejor calificación, es decir los caminos con nodos más cooperativos. A partir del producto de la reputación de los nodos intermedios del camino, se obtiene una calificación que identifica con facilidad a los caminos más cooperativos.

2.6.5 Comparativa de diversos protocolos de encaminamiento basados en sistemas de reputación

RDSR-V [**11**], OCEAN [**68**], CONFIDANT [**58**] y CORE [**69**] son implementados sobre DSR, y cada entidad de red mantiene un histórico de lo colaborativas que son otras entidades mediante una técnica de reputación que mejora la cooperación de los nodos y por tanto el rendimiento de la red. Estos cuatro algoritmos son descritos a continuación:

El algoritmo que forma la métrica de reputación de RDSR-V utiliza dos medias móviles para calcular valores probabilísticos de reputación. En RDSR-V el peso dado a alfa para nuevos valores es dividido para el número de nodos del camino, y la función de muestreo para el cálculo de la reputación muestreada es una exponencial $e^{-14\text{xFPLpi}}$.

El algoritmo que forma la métrica de reputación de OCEAN, utiliza información de primera mano, es decir observaciones directas, por tanto es menos vulnerable a falsas acusaciones. En este algoritmo, se nota la presencia del componente *route ranker*, el cuál con eventos positivos suma uno, con eventos negativos resta dos, y cuando alcanza un umbral de -40 el nodo es registrado en una lista de fallo.

El algoritmo que forma la métrica de reputación de CONFIDANT se compone de tres métricas, la primera es el *goodput* G que es un valor porcentual de 0 a 100 obtenido de la relación entre paquetes recibidos y paquetes originados, la segunda métrica es el *overhead* O el cual expresa la sobrecarga causada por mensajes extra; y, la tercera métrica es *utility* Ui la cual indica el costo-beneficio de un nodo.

Finalmente, se tiene que, el algoritmo que forma la métrica de reputación de CORE se compone de tres métricas, la reputación subjetiva que es calculada a partir de información de primera mano, la segunda métrica es la reputación indirecta que es actualizada desde información de segunda mano; y, la tercera métrica es la reputación

funcional que es la combinación de reputación subjetiva e indirecta para diferentes funciones. Además, los valores de reputación de CORE varían entre -1 y +1.

2.7 Propuesta de algoritmo de protocolo de encaminamiento REMODE_sw (*Reputation for Mobile Devices Static Weight*) basado en la reputación de los nodos

El objetivo del algoritmo, es elegir de entre los caminos disponibles, aquellos que tengan una mayor reputación, o lo que es lo mismo, los caminos que tengan menos pérdidas de paquetes. Para ello, cada nodo origen estima la reputación de sus nodos vecinos a partir de las pérdidas que se producen en los caminos que utiliza. La estimación de la reputación de los nodos se realiza repartiendo las pérdidas de cada camino por igual entre los nodos intermedios de dicho camino. Ello, aunque pueda parecer injusto para los nodos, no lo es, ya que el objetivo del algoritmo no es estimar dicha reputación con exactitud, si no conocer los caminos con menos pérdidas. Cuando un nodo origen dispone de más de un camino para alcanzar un destino, calcula la reputación de dichos caminos a partir de la estimación de la reputación de los nodos, eligiendo aquel camino que presente menores pérdidas, es decir, mayor reputación.

Las innovaciones de REMODE_sw respecto al protocolo RDSR-V [**11**] son: (i) la reputación se asocia directamente a la probabilidad de no encaminamiento por parte del nodo o camino, (ii) el cálculo de la reputación muestreada es mejor, $1 - \mathrm{FPL_{pi}}$ modela de mejor manera que al utilizar $e^{-14.\mathrm{FPLpi}}$, (iii) la información de realimentación (feedback) no sólo se la puede conseguir desde RTP, sino también desde TCP constituyéndose en un protocolo genérico que puede ser utilizado en distintos escenarios y aplicaciones, (iv) el protocolo propuesto es liviano (pocos métodos), eminentemente reactivo, no requiere el uso de paquetes de prueba o paquetes hola, únicamente trabaja con información de primera mano, hasta el punto de que los errores de camino se los atiende con una retransmisión desde la fuente, no desde los nodos intermedios, consiguiendo que siempre se utilicen caminos de mejor reputación; y, finalmente (v) a partir de REMODE_sw se ha conseguido DrepSR, que permite adaptar el protocolo a escenarios con comportamientos de nodos muy cambiantes.

2.8 Protocolos utilizados para el diseño de cruce de capa

2.8.1 Definición de diseño de cruce de capa

Identificando los componentes principales en REMODE_sw, tenemos que la recolección de información de pérdidas la realiza el nodo fuente mediante el feedback de los paquetes RR-RTCP o directamente desde TCP en el nodo fuente. Cuando se utiliza RTP/RTCP la fracción de pérdida de paquetes es calculada mediante el cociente entre paquetes perdidos y paquetes enviados, en tanto que cuando se utiliza TCP, la fracción de pérdida de paquetes se calcula mediante el cociente entre paquetes retransmitidos y paquetes enviados. Entonces desde capa de transporte se consigue una FPLpi por cada camino existente y esta información pasa a capa de red para el cálculo de la reputación de los nodos.

2.8.2 RTP/RTCP (*Real-time Transport Protocol/Real-time Transport Control Protocol*)

RTP proporciona funciones de transporte de red de extremo a extremo para aplicaciones que transmiten datos en tiempo real, como audio y vídeo, a través de servicios de red multidifusión (punto a multipunto) o unidifusión (punto a punto), se establece el tipo de carga útil de audio o video del paquete en base al campo de la cabecera denominado "tipo de carga". En RTCP, existen cinco tipos de paquetes bien conocidos que son: informe del emisor, informe del receptor (RR-RTCP), descripción de la fuente, adiós y aplicación definida, de los cuales informe del receptor (RR-RTCP) mediante su campo fracción de pérdida de paquetes es capaz de indicar a la fuente el valor del número de paquetes perdidos dividido para el número de paquetes esperados. RTP transporta los datos que tienen propiedades de tiempo real y RTCP monitorea la calidad del servicio. RTP no aborda la reserva de recursos y no garantiza la calidad de servicio. RTP y RTCP están diseñados para ser independientes de las capas subyacentes de transporte y red [**70**].

2.8.3 TCP-Vegas (*Transmission Control Protocol* Versión Vegas)

A diferencia de TCP-Reno que aumenta su ventana de congestión (CWND) hasta que se detecta una pérdida de paquetes, TCP-Vegas utiliza el mecanismo evitar congestión para evitar la pérdida de paquetes al disminuir su ventana de congestión tan pronto como detecta una congestión que empieza. En TCP-Vegas, la ventana de congestión se

determina por la diferencia entre el rendimiento esperado y el rendimiento real de la siguiente manera. El rendimiento esperado es el cociente entre el tamaño de la ventana de congestión (CWND) y *BaseRTT*. El rendimiento actual es el cociente entre el tamaño de la ventana de congestión y *currentRTT*. La variable *Diff* en TCP-Vegas es la diferencia entre rendimiento esperado y rendimiento actual multiplicado por *BaseRTT*. Donde *BaseRTT* es el menor de todas las medidas de RTTs y *currentRTT* es el RTT actual. TCP-Vegas define dos umbrales, nombrados α y β. Si *Diff* < α, se considera la ausencia de congestión e incrementa su ventana de congestión en 1. Si Diff > β, se espera una congestión inicial y decrementa su ventana de congestión en 1. De lo contrario, mantiene la ventana de congestión actual. Con el fin de retransmitir paquetes perdidos, TCP-Vegas mantiene un valor de tiempo de espera de retransmisión (RTO) preciso para cada paquete transmitido, que se utiliza para determinar la ocurrencia de un evento de tiempo agotado (timeout) cuando un acuse de recibo (ACK) duplicado para un paquete correspondiente es recibido. Si se produce el evento de tiempo agotado, el remitente TCP piensa que el paquete se pierde y retransmite sin esperar acuses de recibo duplicados adicionales [**71**]. Es necesario aclarar que estos valores de α y β que utiliza TCP-Vegas corresponden a capa de transporte y no se refieren en absoluto a los valores probabilísticos α y β de capa de red utilizados para pesos de las ecuaciones de media móvil para el cálculo de reputación de caminos y nodos respectivamente. En las ecuaciones Ec. 2.4, Ec. 2.5 y Ec. 2.6 que se presentan a continuación, se muestra de forma resumida lo que se acaba de explicar:

$$Expected = \frac{WindowSize(CWND)}{BaseRTT} \tag{2.4}$$

$$Actual = \frac{WindowsSize(CWND)}{currentRTT} \tag{2.5}$$

$$Diff = (Expected - Actual) * BaseRTT \tag{2.6}$$

En TCP intervienen cuatro algoritmos que son: inicio lento, evitar congestión, retransmisión rápida y recuperación rápida. En TCP, el algoritmo que trata con las pérdidas de paquetes se llama evitar congestión por tanto es oportuno profundizar un poco más en dicho algoritmo. En evitar congestión, hay dos indicios de pérdida de paquetes: se produce un tiempo agotado y la recepción de acuses de recibo (ACKs)

duplicados. Evitar congestión e inicio lento son algoritmos independientes con objetivos diferentes. Pero cuando ocurre la congestión TCP debe retrasar su velocidad de transmisión de paquetes en la red, y luego invocar el algoritmo de inicio lento para conseguir crecer nuevamente [**72**].

2.8.4 TCP-Linux Vegas (TCP Vegas Versión LINUX)

Desde la introducción de TCP en los años 70, se han propuesto muchas variantes para hacer frente a las diferentes condiciones de red que podemos tener (por ejemplo, redes cableadas, redes inalámbricas, enlaces por satélite). En el kernel de *Linux OS* (versión 2.6.x) se incluyen trece diferentes versiones, partiendo desde la estándar TCP (TCP Reno) y su versión mejorada (TCP Vegas), a las variantes para redes inalámbricas (TCP Veno and TCP Westwood), redes de alta velocidad (TCP BIC, TCP CUBIC, HSTCP, H-TCP, TCP Illinois, Scalable TCP, y TCP YeAH), y redes satelitales (TCP Hybla), y también una versión de baja prioridad (TCP-LP) [**73**].
NS-2 TCP incorpora el código fuente de los módulos de control de congestión TCP de los *kernels* de Linux. En comparación con las implementaciones NS-2 TCP existentes, NS-2 TCP-Linux tiene tres mejoras: 1) una interfaz estándar para algoritmos de control de congestión similar a la de Linux 2.6, que garantiza una mejor extensibilidad para los algoritmos de control de congestión que aparecen; 2) un módulo de detección de pérdidas rediseñado que es más preciso; y 3) un nuevo planificador de colas de eventos que aumenta la velocidad de simulación. Como resultado, NS-2 TCP-Linux es más extensible, se ejecuta más rápido y produce resultados de simulación que están mucho más cerca del comportamiento TCP real de Linux. Además de ayudar a la comunidad de investigación en redes, NS-2 TCP-Linux también ayudará a la comunidad del kernel de Linux a depurar y probar sus nuevos algoritmos de control de congestión [**74**].

2.9 Métricas de rendimiento de red utilizadas

2.9.1 FPL (*Fraction of Packet Losses*)

Con RTP/RTCP la fracción de pérdida de paquetes es el cociente entre los paquetes perdidos y los paquetes enviados. Con TCP la fracción de pérdida de paquetes es el cociente entre paquetes retransmitidos y paquetes enviados, ya que se considera que por cada perdido habrá un retransmitido. Cuando el valor de FPL es cercano a cero quiere decir que no existen pérdidas en el camino, mientras que valores cercanos a 1 indican

un camino con pérdidas elevadas, es decir un camino no cooperativo. En redes inalámbricas, diferentes razones podrían conducir a la pérdida de paquetes, tales como: enlaces de red saturados, colisiones, roturas de enlaces, poca potencia de señal, entre otros.

2.9.2 Retardo medio que sufren los paquetes

El retardo de nodo está dado por la suma de los retardos más importantes que son: retardo de procesamiento, retardo de cola, retardo de transmisión y retardo de propagación. El retardo de procesamiento es el tiempo requerido para examinar la cabecera del paquete y determinar hacia dónde debe dirigirse. El retardo de cola es el tiempo que espera el paquete para ser transmitido en el enlace, el retardo de cola de un paquete específico, dependerá del número de paquetes que hayan llegado anteriormente y que están encolados y esperando para la transmisión sobre el enlace. El retardo de transmisión del nodo A al nodo B (también llamado retardo de almacenar y enviar) es L/R, donde L es la longitud del paquete en bits, y R es la tasa de transmisión del enlace al siguiente nodo B. Finalmente, una vez que un bit es empujado al enlace, necesita propagarse al encaminador B. El tiempo necesario para propagarse desde el comienzo del enlace al encaminador B es el retardo de propagación.

Luego de entender el retardo de nodo, definimos el retardo extremo a extremo (retardo desde la fuente hasta el destino) como N multiplicado por la suma de tres retardos: retardo de procesamiento, retardo de transmisión y retardo de propagación. En este cálculo se considera que hay N-1 encaminadores entre la máquina origen y la máquina destino, y que la red no está congestionada (es decir, que los retardos de cola son despreciables). Esta fórmula se la puede generalizar para el caso de retardos heterogéneos en los nodos y en presencia de un retardo promedio de cola en cada nodo [**39**].

El retardo terminal a terminal promedio (de paquetes) es lo que se evalúa como métrica de calidad de servicio en simulaciones posteriores con la finalidad de realizar las comparaciones entre protocolos.

2.9.3 Número medio de saltos realizados por los paquetes

El número de saltos indica cuántos saltos da el paquete para llegar al destino, por lo general es deseable valores bajos, pero en la presente investigación se considera que los

caminos más cortos no siempre son los mejores, ya que los caminos más cortos pueden tener nodos no cooperativos que aumentan demasiado las pérdidas del camino. El número de nodos que conforman el camino menos uno nos permite obtener el número de saltos del paquete, en tanto que el número de nodos que forma el camino menos dos nos da el número de nodos intermedios del camino.

2.10 Herramientas de simulación utilizadas

Las herramientas software utilizadas sobre Linux son NS-2.35 y Bonnmotion v3.0.0, mientras que sobre Windows la herramienta software utilizada fue Matlab R2107b. Linux es un conocido sistema operativo de código abierto, el competidor de Microsoft Windows y Macintosh de Apple. Hay dos formas de trabajar con un sistema Linux: mediante una interfaz gráfica de usuario con ventanas, íconos y controlada con el ratón, o mediante una interfaz de línea de comandos, denominada shell, para introducir y ejecutar comandos. En Linux la línea de comandos es fundamental [**75**]. La interfaz de línea de comandos de Linux (Ubuntu 14.04) fue muy utilizada para la obtención de los resultados del Capítulo 3, mientras que para la obtención de los resultados del Capítulo 4 fue muy útil Windows (Windows 7).

2.10.1 NS-2.35 (*Network Simulator* Versión 2.35)

NS-2 es un simulador de redes orientado a eventos desarrollado en la UC *Berkeley* que simula una variedad de redes IP. Pone en práctica protocolos de red como TCP y UPD, el comportamiento del tráfico fuente tal como FTP, Telnet, Web, CBR y VBR, el mecanismo de gestión de cola del encaminador, tal como Drop Tail, RED y CBQ, algoritmos de encaminamiento como Dijkstra, y mucho más. NS también implementa la multidifusión y algunos de los protocolos de capa de control de acceso al medio (MAC) para las simulaciones de redes de área local (LAN). El proyecto NS ahora es una parte del proyecto VINT que desarrolla herramientas para la simulación, para visualizar los resultados de la simulación, análisis y convertidores que transforman topologías de red hechas por generadores bien conocidos a los formatos de NS. Actualmente, NS (versión 2) escrito en C++ y OTcl (Tcl lenguaje de script con extensiones orientadas a objetos desarrollado en el MIT).

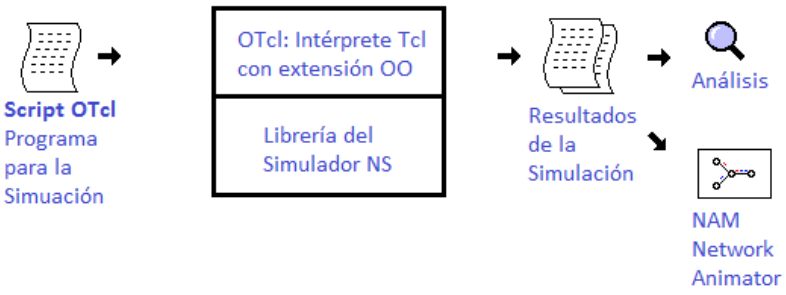


Figura 2.20: Vista simplificada de usuario NS2 [**76**]

Como se muestra en la Figura 2.20, en una vista de usuario simplificada, NS es un intérprete de scripts Tcl orientado a objetos (OTcl), que tienen un planificador de eventos de simulación, librerías de componentes de objetos de red, y finalmente, la configuración de la red. En otras palabras, para utilizar NS, se programa en lenguaje de *script OTcl*. Para configurar y ejecutar una simulación de red, el usuario debe escribir un *script OTcl* que inicia un planificador de eventos, configura la topología de la red mediante los objetos de red y las funciones de reparación de caminos, e indica a las fuentes de tráfico el inicio y fin de la transmisión de paquetes a través del planificador de eventos [**76**]. A partir de ello tenemos los resultados de la simulación (un archivo .tr), que permitirá obtener estadísticos para el análisis, mediante el uso de filtros, o simplemente se puede visualizar la simulación con la ayuda del NAM (Network Animator).

2.10.2 Bonnmotion v3.0.0

Bonnmotion es un software Java que crea y analiza escenarios de movilidad y es más comúnmente utilizado como una herramienta para la investigación de las características móviles ad-hoc de la red. Los escenarios también se pueden exportar para varios simuladores de red, como NS-2, NS-3, GloMoSim / QualNet, COOJA, MiXiM y ONE. *Bonnmotion* está siendo desarrollado conjuntamente por el Grupo de Sistemas de Comunicación de la Universidad de Bonn, Alemania, el Grupo Toilers de la Escuela de Minas de Colorado Golden, en Colorado EE.UU., y el Grupo de Sistemas Distribuidos de la Universidad de Osnabruck, Alemania. Algunos de los modelos que es posible obtener son los siguientes:

- Random Waypoint model

- Random Walk model
- Gauss-Markov model
- Manhattan Grid model
- Reference Point Group Mobility model
- Disaster Area model
- Random Street model

El detalle de los parámetros de entrada de los modelos que se acaba de enumerar, así como de algunos otros más que no son tan conocidos, lo encontramos en [**77**].

Para las simulaciones de esta tesis, el escenario de movilidad obtenido con Bonnmotion v3.0.0 es referenciado en el "script OTCL", con la finalidad de especificar el patrón de movilidad utilizado para determinada simulación. El patrón de movimiento que más se ha utilizado en esta tesis es el Random Waypoint, ya que es el que más se adapta a los escenarios propuestos.

2.10.3 Matlab R2017b

MATLAB [**78**] es un entorno informático técnico que proporciona análisis avanzados, visualización de datos y el marco para desarrollar algoritmos a medida. El lenguaje abierto de MATLAB permite compartir ideas y desarrollar soluciones. Se proporcionan interfaces a otros lenguajes de programación, incluyendo Fortran y C

Con más de 600 funciones matemáticas, estadísticas y de ingeniería, MATLAB brinda acceso inmediato a computación numérica de alto rendimiento. Esta funcionalidad se amplía con capacidades gráficas interactivas para crear gráficos, imágenes, superficies y representaciones volumétricas.

La funcionalidad se amplía aún más con las cajas de herramientas adicionales; las cajas de herramientas NAG, Maple, Simbólica y Procesamiento de Señales se encuentran entre algunas de las cajas de herramientas disponibles en este servicio. Simulink, que proporciona un entorno de diseño gráfico para modelar y simular control complejo, DSP y sistemas lógicos de supervisión, también está disponible en este servicio.

2.11 Conclusiones

El objetivo del capítulo fue estudiar las generalidades de las redes móviles ad-hoc, y luego de desarrollado el capítulo, llegamos a las siguientes conclusiones:

- La tecnología utilizada para los diseños de red de esta investigación es 802.11g, la cual permite formar varios caminos simultáneos en la red móvil ad-hoc, entre más caminos se puedan formar la actualización de los valores de reputación de los nodos será más precisa, considerando que tampoco es recomendable demasiados caminos, ya que es bien conocido que el ancho de banda es limitado.
- Las redes ad-hoc se clasifican en redes de sensores inalámbricas (WSN), redes de malla inalámbricas (WMN) y redes móviles ad-hoc (MANETs). Como variante de las redes móviles ad-hoc tenemos unas redes que actualmente están teniendo mucha atención por parte de los laboratorios de telemática que son las redes vehiculares ad-hoc (VANETs). Las redes ad-hoc tienen múltiples aplicaciones debido a su coste, disponibilidad y flexibilidad.
- Los protocolos de encaminamiento se clasifican en proactivos, reactivos e híbridos. Las dos propuestas de algoritmos de protocolo de encaminamiento que son el objeto de estudio de esta tesis, entran en la categoría de protocolos reactivos, ya que actualizan su información de encaminamiento únicamente cuando se requiere realizar una transmisión.
- El protocolo DSR (Dynamic Source Routing) es un protocolo eficiente para encaminamiento distribuido, diseñado específicamente para ser utilizado en redes inalámbricas ad-hoc multi-salto con nodos móviles. DSR está conformado por dos mecanismos principales que son el descubrimiento y el mantenimiento de caminos que trabajan juntos para descubrir y mantener caminos a destinos arbitrarios en la red ad-hoc. DSR es un protocolo que opcionalmente puede trabajar con un mecanismo adicional denominado “extensión de estado de flujo” que le permite trabajar con caminos implícitos.
- Los sistemas de reputación permiten a un nodo tener un histórico de cooperatividad de cada uno de los nodos que conforman la red móvil ad-hoc, a partir de dicha información se puede preferir los más cooperativos y evitar los menos cooperativos mejorando de esta manera el desempeño de la red.
- En este capítulo de tesis, presentamos la propuesta de un nuevo protocolo de encaminamiento denominado REMODE_sw (Reputation for Mobile Devices Static Weight), el cual a partir de información obtenida de cruce de capa calcula la reputación del camino, luego la reputación de los nodos de dicho camino; y

posteriormente obtiene la calificación de los caminos, a partir de la cual selecciona los caminos más cooperativos.

- Como métricas que nos permiten evaluar el rendimiento de REMODE_sw (Reputation for Mobile Devices Static Weight) y DrepSR (Dynamic Reputation Source Routing) tenemos, en primer lugar el promedio de la fracción de pérdida de paquetes por cada enlace de la red móvil ad-hoc, luego el promedio del retardo extremo a extremo de cada uno de los caminos y finalmente el promedio del número de saltos que los paquetes dan hasta llegar a su destino.
- En la presente investigación para comparativas de resultados entre DSR y REMODE_sw, hemos utilizado *Bonnmotion v3.0.0* para crear el patrón de movilidad, hemos utilizado NS-2.35 para crear y ejecutar escenarios de simulación; y, hemos utilizado *Network ANimator* (NAM) para analizar visualmente lo que sucede en la simulación. Adicionalmente, hemos desarrollado *Trace Analyzer v3.0*, un programa en C++ dentro de NS-2.35 que permite obtener las métricas de calidad de servicio en tiempo de ejecución, realizando el muestreo de los datos desde trazas, para posteriormente poder obtener las gráficas de métricas de calidad de servicio. Para probar toda la fundamentación matemática de DrepSR, y realizar las comparativas de resultados entre DSR y DrepSR, hemos utilizado *Matlab R2017b*.
- Luego de que hemos analizado el código fuente de la memoria caché de DSR en NS-2.35, y luego de que hemos realizado múltiples simulaciones en distintos escenarios, determinamos que efectivamente, DSR mantiene en cada nodo fuente una tabla de encaminamiento con múltiples caminos, de esta manera en los capítulos 3 y 4, es posible seleccionar los caminos cuyos nodos tienen la reputación más alta

Capítulo 3

REMODE_sw (Reputation for Mobile Devices Static Weight)

"El software no-libre trae con él un sistema antisocial que prohíbe la cooperación y la comunidad. No puedes ver el código fuente; no puedes decir qué trucos feos, ni que errores tontos, podría contener. Si no te gusta, no puedes cambiarlo. Y lo peor de todo, está prohibido compartirlo con alguien más. Prohibir que se comparta el software es cortar los lazos de la sociedad".

— Richard Matthew Stallman. Programador y activista de la libertad de software estadounidense.1953

"Libre no significa gratis".

— Richard Matthew Stallman. Programador y activista de la libertad de software estadounidense.1953

3 REMODE_sw (*Reputation for Mobile Devices Static Weight*).

En el presente capítulo se explica detalladamente la teoría y los experimentos realizados con REMODE_sw en el simulador de redes NS-2.35

3.1 Introducción

El objetivo del algoritmo REMODE_sw [**79**] es elegir, de entre los caminos disponibles, aquellos que tengan una mayor reputación, o lo que es lo mismo, los caminos que tengan menos pérdidas de paquetes. Para ello, cada nodo origen estima la reputación de sus nodos vecinos a partir de las pérdidas que se producen en los caminos que utiliza. La estimación de la reputación de los nodos se realiza repartiendo las pérdidas de cada camino por igual entre los nodos intermedios de dicho camino. Ello, aunque pueda parecer injusto para los nodos, no lo es, ya que el objetivo del algoritmo no es estimar dicha reputación con exactitud, si no conocer los caminos con menos pérdidas. Cuando un nodo origen dispone de más de un camino para alcanzar un destino, calcula la reputación de dichos caminos a partir de la estimación de la reputación de los nodos, eligiendo aquel camino que presente menores pérdidas, es decir, mayor reputación.

3.2 Evolución de la métrica de reputación de RDSR-V (*Reliable Dynamic Source Routing for video-streaming*)

La evolución de la métrica de reputación de RDSR-V [**11**] da como resultado el algoritmo REMODE_sw. La similitud está en que ambos algoritmos utilizan dos medias móviles exponenciales, pero REMODE_sw utiliza un valor de alfa para el cálculo de la reputación de los caminos y un valor de beta para la reputación de los nodos, en cambio RDSR-V tanto para caminos como para nodos únicamente utiliza una sola variable llamada alfa que indica el peso que tienen los valores nuevos respecto a los antiguos.

Las diferencias principales que se aprecian son dos, la primera es que en RDSR-V para calcular la reputación del nodo, el peso dado a alfa para nuevos valores es dividido para el número de nodos del camino, esta división no se la realiza en REMODE_sw con lo cual se tiene un valor más fácil de entender y asociar directamente con un valor

probabilístico de reputación del nodo; y, como segunda diferencia es que la función de muestreo para el cálculo de la reputación muestreada en RDSR-V es una función exponencial ($e^{-14.FPLpi}$), mientras que en REMODE_sw es simplemente una función lineal (1-FPLpi).

3.3 Descripción matemática de REMODE_sw

La Figura 3.1 representa los nodos que pertenecen a un camino. Si alguno de estos nodos tiene un comportamiento egoísta (no cooperativo), no reenviará una fracción de los paquetes recibidos. Definimos la fracción de pérdida de paquetes por el nodo *j* como la relación entre el número de paquetes no reenviados y la cantidad total de paquetes recibidos ($FPLn_j$ –fraction of packet losses – [**11**]). Para estimar el valor de reputación de reenvío de un nodo *j* ($RFVn_j$ – reputation forwarding value – [**11**]), se supone que el principal motivo por el cual se pierden los paquetes es el comportamiento egoísta de determinados nodos de la red. Definimos la reputación de reenvío para un nodo *j* como la probabilidad complementaria de la fracción de pérdida de paquetes (Ec. 3.1).

$$RFVn_j = 1 - FPLn_j \tag{3.1}$$

En esta propuesta asumimos que el nodo origen de la Figura 3.1, recibe información periódica sobre la fracción de pérdida de paquetes para cada camino *i* ($FPLp_i$) que está utilizando. Dicha periodicidad puede depender de la cantidad de tráfico cursado en dicho camino, o bien puede ser cada cierto intervalo de tiempo fijo. El valor complementario de la $FPLp_i$ es la reputación de reenvío asociada a dicho camino ($RFVp_i$). El emisor promedia dicho valor mediante una media móvil (Ec. 3.2)

$$\overline{RFVp_i}|_{\text{updated}} = (1-\alpha)\,\overline{RFVp_i}|_{\text{old}} + \alpha\,(1 - FPLp_i) \tag{3.2}$$

Por otro lado, considerando que el comportamiento de cada nodo a la hora de realizar los reenvíos es independiente del resto, se puede calcular analíticamente la reputación de un camino *i* ($RFVp_i$) como un producto de probabilidades (Ec. 3.3). En la siguiente sección a esta, a la reputación se la denomina SCORE, ya que nos indica la calificación del camino.

$$RFVp_i = \prod_{j \in path\ i} (RFVn_j) \tag{3.3}$$

Dado que a priori el nodo origen desconoce el comportamiento de los nodos que forman un camino, no es posible utilizar la Ec. 3.3 para estimar la reputación de reenvío de dichos nodos. El algoritmo propuesto para realizar dicha estimación, se basa en repartir las pérdidas por igual entre los nodos que forman el camino, excepto el nodo origen que se supone que no genera pérdidas. En este caso, se puede reescribir la Ec. 3.3 de la siguiente forma.

$$RFVp_i = (RFVn_j)^{n_i} \tag{3.4}$$

Siendo n_i el número de nodos intermedios en el camino *i*.

Cada vez que el nodo emisor actualiza la fracción de pérdida de paquetes de un camino *i*, estimamos el valor de la reputación de reenvío de cada uno de los nodos que pertenecen a dicho camino ($\widehat{RFVn_j}$). Para ello se utiliza el valor de la reputación del camino *i* calculado en Ec. 3.2 sobre la Ec. 3.4.

$$\widehat{RFVn_j} = (\overline{RFVp_i})^{\frac{1}{n_i}} \tag{3.5}$$

El hecho de repartir las pérdidas por igual entre los nodos que componen el camino perjudica a la reputación de los nodos que tienen un comportamiento colaborativo y beneficia a la reputación de los nodos con un comportamiento más egoísta. Es por ello que la reputación de los nodos debe promediarse teniendo en cuenta las estimaciones hechas en todos los caminos en los que el nodo participa. Para este caso también se realiza una media móvil (Ec. 3.6).

$$\overline{RFVn_j}|_{\text{updated}} = (1-\beta)\overline{RFVn_j}|_{\text{old}} + \beta\,\widehat{RFVn_j} \tag{3.6}$$

El objetivo final es que cada nodo disponga de una tabla con las reputaciones de los otros nodos de la red, que le permita evaluar la reputación de los distintos caminos que ha descubierto y elegir aquel de mayor reputación. La reputación de los caminos se obtiene utilizando de nuevo la Ec. 3.3.

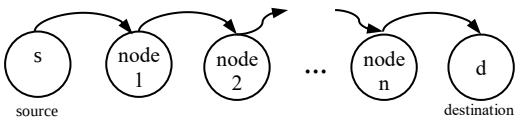


Figura 3.1: Nodos que componen un camino.

Es posible realizar un análisis teórico para obtener el valor medio de la reputación que un nodo origen *s*, asignará a un nodo intermedio *j* mediante el algoritmo propuesto. Para el caso que el nodo origen reciba información sobre las pérdidas de los caminos cada vez que manda un determinado volumen de tráfico el cálculo dependerá del número de caminos establecidos y el tráfico cursado en cada uno de ellos, tal como se muestra en la Ec. 3.7.

$$\overline{RFVn_{j,s}} = \sum_{i \supset \{n_s, n_j\}} \left(\frac{\gamma_i}{\gamma_{n_{j,s}}} (RFVp_i)^{\frac{1}{n_i}} \right) \tag{3.7}$$

Donde:

- i: Caminos con origen en el nodo *s*, a los cuáles pertenece el nodo *j*.
- n_i: Número de nodos intermedios del camino *i*.
- γ_i: Tráfico cursado en el camino *i*.
- $\gamma_{n_{j,s}}$: Tráfico de todos los caminos con origen en el nodo *s*, a los cuáles pertenece el nodo *j* (Ec. 3.8).

$$\gamma_{n_{j,s}} = \sum_{i \supset \{n_s, n_j\}} (\gamma_i) \tag{3.8}$$

Para el caso que el nodo origen reciba información sobre las pérdidas cada cierto tiempo fijo, todos los caminos tendrán el mismo peso independientemente del tráfico cursado. Ello equivale a utilizar la Ec. 3.7 con todos los tráficos iguales.

En la Ec. 3.7 la $RFVp_i$ de los caminos se obtiene a partir de la Ec. 3.3, tomando como dato la fracción de pérdida de paquetes en cada nodo (Ec. 3.2).

Así por ejemplo, en la Figura 3.2 representamos una topología con distintos caminos, siendo los nodos 7 y 9 los únicos que no retransmiten todos los paquetes que reciben (con pérdidas). El nodo 5 es el nodo origen para los caminos 5, 6 y 7. Tomando el mismo tráfico en todos los caminos se obtiene que la reputación que el nodo 5 le asignará al nodo 6 es 0,9426 (Ec. 3.9), cuando en realidad le correspondería una reputación de 1 ya que reenvía todos los paquetes.

$$\overline{RFVn_{6,5}} = \frac{1}{3}1 + \frac{1}{3}(0{,}75 \cdot 0{,}85)^{\frac{1}{5}} + \frac{1}{3}(0{,}75 \cdot 0{,}85)^{\frac{1}{5}} = 0{,}9426 \qquad (3.9)$$

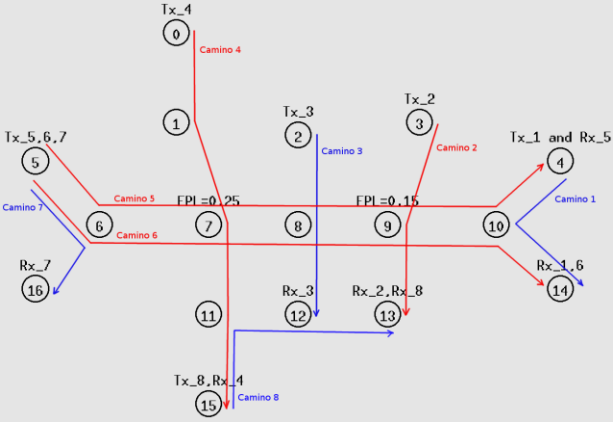


Figura 3.2: Diseño del escenario de simulación

3.4 Pseudocódigo para REMODE_sw

A continuación en el Algoritmo 3.1, Algoritmo 3.2 y Algoritmo 3.3 presentamos el pseudocódigo del algoritmo REMODE_sw utilizando nomenclatura similar a la utilizada en los algoritmos del artículo del protocolo RDSR-V [**11**]. Experimentalmente se ha determinado que el valor óptimo de α=1, el de β=0,6 y el intervalo de muestreo puede ser por tiempo o por número de paquetes. Los experimentos iniciales son por tiempo, en tanto que los experimentos de las secciones 3.5.5 y 3.5.6 y los del siguiente capítulo son por número de paquetes.

3.4.1 Recolección de información

En el Algoritmo 3.1, el primer evento que gestionamos es la recepción de un paquete de retroalimentación, asumimos que un nodo fuente S está enviando paquetes de datos a un destino D utilizando un determinado camino P. Asumimos también que el camino P está compuesto de K nodos. Cuando un paquete de retroalimentación RR-RTCP es recibido, el parámetro fracción de pérdida de paquetes (FPL) es extraído y la función ProcessFeedback() del Algoritmo 3.1 es ejecutada. RFV_{sample} se la obtiene a partir de la fracción de pérdida de paquetes (FPL) del camino incluida en el paquete RR-RTCP. En general, en sistemas de reputación la calificación de reputación es expresada con un valor numérico que va de cero a uno, donde cero es la peor calidad de servicio (QoS) y uno es la mejor calidad de servicio (QoS) con nodos completamente cooperativos.

1. *ProcessFeedback(P, FPL){*
2. $RFV_{sample} = 1 - FPL;$
3. *UpdateHistory(P,* RFV_{sample}*);*
4. *}*

Algoritmo 3.1: Pseudocódigo para procesar paquetes RR-RTCP

3.4.2 Modelado de la información

Como se muestra en Algoritmo 3.2, en UpdateHistory() utilizamos una media móvil exponencial tanto para el cálculo de la reputación histórica del camino $RFV_{history} \leftarrow RFV_{history}(1-\propto) + RFV_{sample}\alpha$, como para la reputación histórica del nodo $RFV_{history} \leftarrow RFV_{history}(1-\beta) + (RFVp)^{\frac{1}{n}}\beta$, filtros que toman en cuenta la reputación histórica acumulada y la muestra actual de reputación utilizando factores de envejecimiento (pesos) alfa y beta altos.

Los parámetros $\propto$ y β se pueden sintonizar para detectar lentamente los nodos egoístas (menos cooperativos), o para tener una medida precisa de la reputación de los nodos. Con un valor bajo (cercano a 0) puede tomar un largo período de tiempo para detectar nodos egoístas, pero la detección es más confiable. Con un valor alto (cercano a 1), se da un mayor peso a las calificaciones actuales antes que a las anteriores y así REMODE_sw es capaz de detectar nodos egoístas más rápido. Sin embargo, el sistema se vuelve muy sensible a la mala conducta instantánea que puede conducir a detecciones erróneas del egoísmo.

En la función UpdateHistory(), necesitamos definir lo que sucede cuando no tenemos historial anterior para un camino o un nodo. Para gestionar nodos desconocidos asignamos un valor de uno a las RFVs, y en pocos muestreos consigue el verdadero valor de reputación (como se lo demuestra más adelante en la sección 3.5.6)

Por otro lado, los caminos DSR se almacenan en una tabla de encaminamiento (RT). Esencialmente, los caminos DSR están formados por una lista de direcciones IP y un tiempo de caducidad que normalmente se establece en 300 segundos (DEFAULT_DSR_ROUTE_TIMER). Manejamos dos tablas desde cada nodo fuente, la tabla de reputaciones de caminos, y la tabla de reputación de nodos, donde todas las reputaciones inicialmente comienzan con un valor de uno.

1. *UpdateHistory(P, RFV_{sample}){*
2. *$RFV_P = Get(P, RFV)$;*
3. *if RFV_P==null then RFV_P=1;*
4. *$RFV_P \leftarrow RFV_P(1-\propto) + RFV_{sample}\alpha$ //Actualiza la reputación del camino P*
5. *Set(P,RFV, RFV_P); //Modifica la tabla de reputación de caminos*
6. *Foreach node K ∈ P do //Actualizar reputación de K nodos intermedios del camino P*
7. *$RFV_K = Get(IP_K, RFV)$; //Conseguir la RFV acumulada de IP_K (direcciones IP de nodos)*
8. *if RFV_K==null then RFV_K=1;*
9. *$RFV_K \leftarrow RFV_K(1-\beta) + (RFVp)^{\frac{1}{n}}\beta$; //n nodos intermedios del camino P*
10. *Set(IP,RFV, RFV_K); //Modifica la tabla de reputación de nodos*
11. *end*
12. *}*

Algoritmo 3.2: Pseudocódigo para actualizar reputaciones de caminos

3.4.3 Toma de decisiones

Este módulo toma decisiones, y realiza acciones considerando la reputación de los nodos. Incluye acciones para calcular la calificación del camino y seleccionar el de valor más alto, en caso de dos caminos de igual calificación (SCORE), selecciona el de menor longitud.

En DSR estándar, la selección de camino siempre elige el camino más corto (número mínimo de saltos) entre los caminos posibles disponibles para un destino. En REMODE_sw, la selección de camino toma en cuenta la reputación observada para seleccionar el mejor camino disponible, el de nodos más cooperativos que proporciona la mejor QoS. En REMODE_sw, la selección de camino funciona como sigue; supongamos que tenemos P posibles caminos de acceso a un nodo de destino D. Entonces, la función SelectedPath() del Algoritmo 3.3 se utiliza para seleccionar el mejor camino de reenvío.

Como una guía para entender la función SelectPath(), vale la pena remarcar que utilizamos la calificación de camino como el criterio de selección primario si este

parámetro es conocido. Si no se conoce la calificación del camino de acceso, por ejemplo, porque el camino todavía no se ha utilizado para la transmisión de datos, utilizamos la distancia al destino como segundo criterio de selección. Esto se debe a que consideramos que el camino más corto es usualmente el mejor camino cuando aún no se conoce el verdadero valor de reputación del nodo.

Resumiendo:

- Si conocemos las reputaciones de los nodos intermedios de los caminos, y por tanto las calificaciones (SCORE) de todos los caminos a un destino, el camino con mayor calificación será seleccionado como el mejor.

- Si no conocemos las reputaciones de los nodos intermedios de los caminos (esto se da en el inicio de la simulación) tendrán un valor de 1, y por tanto no tenemos valores reales de calificaciones (SCORE) de todos los caminos a un destino, en tal caso las calificaciones de todos los caminos disponibles serán iguales a 1, por ello seleccionamos el camino más corto como mejor camino, ya que el camino más corto será probablemente el mejor camino.

- En caso de que tengamos varios caminos más cortos (de la misma longitud), utilizamos la reputación de sus nodos intermedios, y seleccionamos el de mayor calificación (SCORE).

1. *SelectedPath(D){*
2. *route←null;*
3. *foreach path P to destination D of the P available ones do*
4. *SCORE=1; max_SCORE=0;*
5. *Foreach node K ∈ P do//K nodos intermedios del camino P*
6. $RFV_K = Get(IP_K, RFV)$; *//Conseguir la RFV acumulada de* IP_K *(direcciones IP de nodos)*
7. *SCORE=SCORE**RFV_K*;*
8. *end*
9. *length=Get(P, Distance);*
10. *if SCORE>max_SCORE //Si calificación de nuevo camino es mayor a calificación de camino analizado previamente*

```
11.      max_SCORE=SCORE;
12.      min_length = length;
13.      route=P;
14.   else if SCORE==max_SCORE
15.      if length<min_length
16.         max_SCORE=SCORE;
17.         min_length = len;
18.         route=P;
19.      end if
20.   end if
21. end foreach
22. return route;
23. }
```

Algoritmo 3.3: Pseudocódigo para seleccionar un camino al destino D sobre P disponibles

3.5 Resultados en NS-2.35 y análisis

3.5.1 Análisis de los valores de reputación al variar los valores de los pesos y en escenario estático

Hemos implementado el algoritmo en el simulador NS-2 [**76**], y hemos realizado distintas simulaciones tanto en escenarios fijos como móviles.

El primer escenario planteado es el de la Figura 3.2 con los parámetros de la Tabla 3.1, consta de 17 nodos dispuestos de tal forma que nos dan un total de 8 caminos, en color rojo se muestran los caminos no cooperativos (2, 4, 5 y 6) y en azul el resto (1, 3, 7 y 8). Dado que el protocolo DSR debe realizar un descubrimiento de camino para encontrar los posibles caminos hacia los distintos destinos, se han ubicado los nodos estratégicamente para forzar los caminos que aparecen en la Figura 3.2. Entonces al utilizar la Figura 3.2 tanto DSR como REMODE_sw seleccionarán los mismos caminos.

Parámetro	Valor
Simulador	NS-2 v2.35
Área	410 m x290 m
Número de nodos	17
Número de conexiones CBR	8
Rango de transmisión	95 m
Especificación MAC	IEEE 802.11g
Patrón de movimiento	Estático
Ancho de banda nominal	54 Mbps
Tiempo de simulación	700 s
Protocolo de encaminamiento	DSR con nodos egoístas
Protocolo de transporte	UDP
Tipo de tráfico	CBR sobre UDP
Máximo tamaño de paquete	1500 Bytes
Velocidad de tráfico CBR	1Mbps
Número de fuentes	6
Nodos fuente	Nodos 4,3,2,0,5,15
Número de destinos	6
Nodos destino	Nodos 14, 13, 12,15, 4 y 16
Tamaño de cola	50 paquetes
Número de nodos egoístas	2
Nodos que generan las pérdidas	Nodo 7 y nodo 9
Intervalo de muestreo	Cada 5 s
Intervalo de tiempo de transmisión	0 s– 700 s

Tabla 3.1: Parámetros de la simulación

Lo primero que se analizará es lo que sucede cuando se varían los valores de los pesos α y β (Ec. 3.2 y Ec. 3.6)

Inicialmente se varía el valor de α. En la Figura 3.3 se observa las reputaciones de 5 caminos, 4 que incluyen nodos no cooperativos (caminos 2, 4, 5 y 6) y uno con nodos cooperativos (camino 3), tanto para α=0,25 (línea con signos +) como para α=0,05 (línea continua). Observamos claramente que con α=0,05 los valores de $\overline{RFVp_i}$ tardan más en actualizarse, es decir presentan un transitorio mayor. Además observamos que el valor de los distintos transitorios no depende significativamente del valor final de la reputación del camino, sino que depende únicamente del valor de α. Para el caso del camino 5 con α=0,25, dicho transitorio se inicia más tarde debido a que el nodo origen ha necesitado más tiempo para realizar el descubrimiento de camino. Transcurrido el transitorio los valores de las reputaciones se estabilizan alrededor del valor que se obtiene a partir de la Ec. 3.3, el cual coincide para los caminos 5 y 6 como se deduce de la Figura 3.2. Cuanto mayor es α, mayor es la variación alrededor de dicho valor, de manera que existe un compromiso entre la duración del transitorio y la varianza alrededor de un valor estable de reputación.

Los valores de $\overline{RFVp_i}$ de los caminos 1, 7 y 8 siempre son iguales a 1, por ello no constan, en su lugar sólo se grafica la reputación del $RFVp_3$, que también es igual a 1.

En un segundo análisis se observa lo que sucede cuando se varía β, que es un peso que afecta directamente a los nodos (Ec. 3.6).

En las gráficas Figura 3.4 y Figura 3.5 se muestra la reputación del nodo 6 ($\overline{RFVn_6}$) calculada por el nodo 5, ya que el nodo 5 es el nodo origen de los caminos en los que participa el nodo 6 (caminos 5, 6 y 7). La reputación que finalmente el nodo 5 asigna al nodo 6 es inferior a su reputación real (que vale 1 al ser el nodo 6 un nodo cooperativo) debido a que en los caminos en que participa el nodo 6 existen nodos con un comportamiento egoísta.

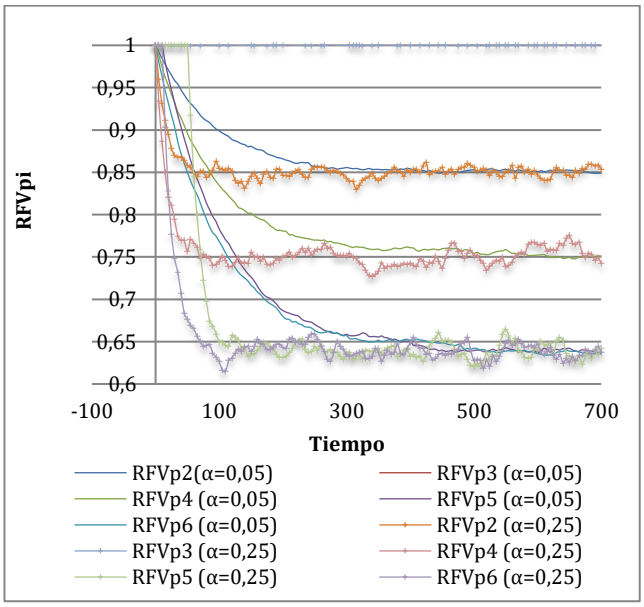


Figura 3.3: $\overline{RFVp_i}$ variando α con β=0,05

En la Figura 3.4 se muestran los valores de $\overline{RFVn_6}$ para β=0,05 y β=0,25 con α=0,25 constante. Al igual que ocurre con la reputación de los caminos, se nota mayor varianza al aumentar β, así para β=0,05 la varianza es de $5{,}260 \times 10^{-7}$, mientras que para β=0,25 la varianza es de $6{,}612 \times 10^{-6}$. También se muestra que para un mayor valor de β el transitorio es menor.

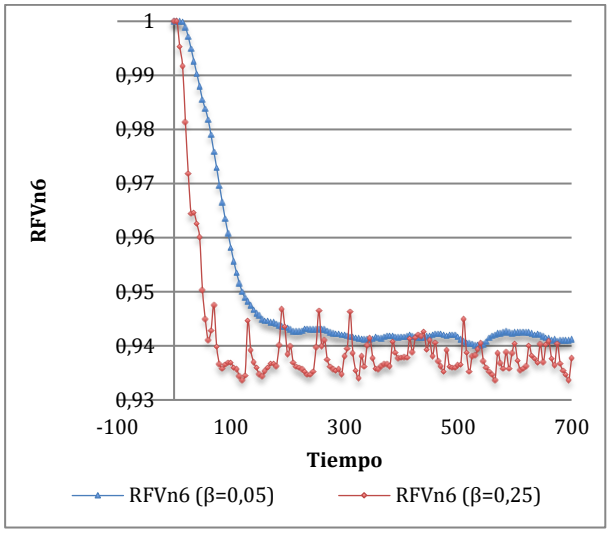


Figura 3.4: $\overline{RFVn_6}$ variando β con α =0,25

En la Figura 3.5 se muestran los mismos resultados para un valor de α =0,05. Para β=0,05 la varianza es de 1,0025x10^{-5}, y para β=0,25 la varianza es de 1,6483x10^{-5}. En este caso un mayor valor de β también implica un aumento de la varianza, pero no una mejora en el transitorio. Ello es debido a que para un valor de α tan pequeño la actualización de los caminos es muy lenta y enmascara el efecto que pueda tener β sobre el transitorio de la reputación de los nodos.

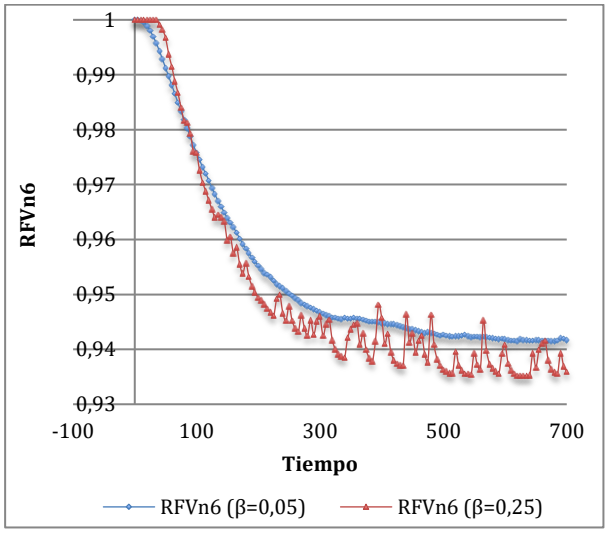


Figura 3.5: $\overline{RFVn_6}$ variando β con α=0,05

3.5.2 Análisis de reputación muestreada, reputación de nodos y calificación para caminos al incrementar las pérdidas de los nodos no cooperativos en escenario estático

Ahora planteamos un segundo escenario con la misma topología de la Figura 3.2 y los parámetros de la Tabla 3.1, pero aumentando las pérdidas de los nodos no colaborativos ($FPLn_7$=0,4 y $FPLn_9$=0,3) y utilizando α=β=0,25.

La Figura 3.6 muestra la reputación muestreada de los caminos, esta es la información que utilizan los nodos origen de cada camino para calcular la $\overline{RFVp_\iota}$, luego la $\overline{RFVn_J}$ y finalmente obtienen las calificaciones de cada camino.

La reputación de los caminos 1, 7, 8 al igual que el camino 3 son iguales a 1, por ello no se los ha representado, facilitando así, la lectura de la gráfica.

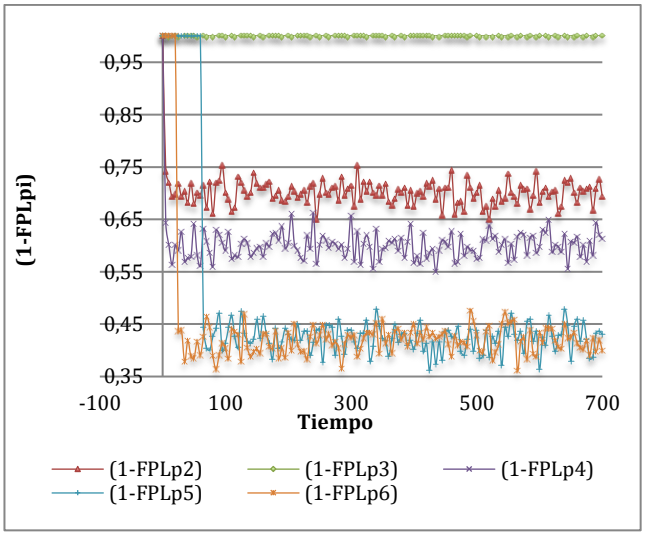


Figura 3.6: Complemento de la FPL muestreada por cada camino

En la Tabla 3.2 se muestra cuál es el cálculo de los valores de reputación de los nodos vistos desde cada nodo origen. El nodo origen 4 únicamente muestrea información del camino 1, el nodo 3 del camino 2, el nodo 2 del camino 3, el nodo 0 del camino 4, el nodo 5 de los caminos 5, 6 y 7, y finalmente para el nodo 15 sólo está a su alcance muestrear el camino 8. Por tanto el nodo origen 4 únicamente calculará la $\overline{RFVn_J}$ del nodo intermedio 10, el nodo origen 3 del nodo intermedio 9, el nodo origen 2 de nodo intermedio 8, el nodo origen 0 de los nodos intermedios 1, 7 y 11, el nodo origen 5 de

los nodos intermedios 6, 7, 8, 9 y 10, el nodo origen 15 de los nodos intermedios 11 y 12.

Además, la Tabla 3.2 muestra una comparativa entre los valores reales, teóricos y los calculados a partir de la simulación, en donde se nota que los valores teóricos y simulados son prácticamente iguales, verificando la utilidad de la Ec. 3.7. Los valores simulados que hemos tabulado corresponden a los valores de reputación calculados en el último instante de la simulación. La diferencia entre el valor teórico y el final simulado es debido a la varianza que presenta el método propuesto, y que el resultado de la Ec. 3.7 es un valor medio. Adicionalmente se muestran los intervalos de confianza de los valores simulados promedio (de 210 a 700 s.).

Nodo	Val. Real	Val. Teór.	Val. Sim. (promedio)	Int. Conf. 95% de Val. Sim. (promedio)
N_0				
N_1	1	0,8434	0,8435	(0,8434; 0,8445)
N_7	0,6	0,8434	0,8435	(0,8434; 0,8445)
N_{11}	1	0,8434	0,8435	(0,8434; 0,8445)
N_2				
N_8	1	1	1	(1,0000; 1,0000)
N_3				
N_9	0,7	0,7	0,6996	(0,6996; 0,7017)
N_4				
N_{10}	1	1	1	(1,0000; 1,0000)
N_5				
N_6	1	0,8938	0,8817	(0,8807; 0,8887)
N_7	0,6	0,8407	0,8414	(0,8390; 0,8481)
N_8	1	0,8407	0,8414	(0,8390; 0,8481)
N_9	0,7	0,8407	0,8414	(0,8390; 0,8481)
N_{10}	1	0,8407	0,8414	(0,8390; 0,8481)
N_{15}				
N_{11}	1	1	1	(1,0000; 1,0000)
N_{12}	1	1	1	(1,0000; 1,0000)

Tabla 3.2: Reputación de nodos observada por el nodo origen cuando se incrementa la FPL

En la Tabla 3.2 se observa que los nodos 2 y 3 obtienen la reputación real de los nodos 8 y 9 respectivamente, ya que éstos son los únicos nodos intermedios para los caminos 3 y 2 respectivamente. Por el contrario el nodo 5 asigna la misma reputación a ambos nodos, lo cual beneficia a la reputación real del nodo 9 (el egoísta) en detrimento de la reputación de nodo 8 (colaborativo). El nodo 5 asigna la misma reputación a todos los nodos excepto al nodo 6, debido a que éste participa en el camino 7. En este caso el

nodo 6 no está afectado por el comportamiento malicioso de los nodos 7 y 9, siendo su reputación más alta que la del resto de nodos.

Nodo	Cal. Real	Cal. Sim (700 s.).	Cal. Real-Cal. Sim.
N_0			
$C_{1\text{-}3,\ 5\text{-}8}$	1	1	0
C_4	0,6	0,6008	-0,0008
N_2			
$C_{1\text{-}8}$	1	1	0
N_3			
$C_{1,\ 3\text{-}8}$	1	1	0
C_2	0,7	0,7026	-0,0026
N_4			
$C_{1\text{-}8}$	1	1	0
N_5			
$C_{1\text{-}4,\ 8}$	1	1	0
C_5	0,42	0,4369	-0,0169
C_6	0,42	0,4415	-0,0215
C_7	1	0,9108	0,0892
N_{15}			
$C_{1\text{-}8}$	1	1	0

Tabla 3.3: Calificaciones de caminos observada por el nodo origen cuando se incrementa la FPL

Cuando un nodo tiene paquetes para transmitir primeramente busca en su caché si tiene algún camino al destino solicitado, en caso de no tenerlo inicia un descubrimiento de camino para obtener el camino. Ahora partiendo de que se tiene uno o más caminos en caché, con el algoritmo aquí presentado se pueden obtener las calificaciones para cada camino, y en base a ellas seleccionar el camino de mayor reputación al destino solicitado. En nuestro caso, para la topología de la Figura 3.2, las calificaciones para cada camino, obtenidas a partir de las reputaciones de los nodos calculadas por cada nodo origen, son las que se muestran en la Tabla 3.3. Se observa que calificación real y simulada tienen una buena aproximación, a pesar que el objetivo del algoritmo no es obtener el valor exacto, si no discriminar cual es el camino con mayor reputación.

3.5.3 Análisis del contenido de la memoria caché primaria del nodo fuente al existir múltiples caminos en escenario estático

En los escenarios anteriores cada nodo origen disponía de un único camino activo, debido a que las distancias entre nodos fueron dispuestas de tal forma que se formen un único camino para cada destino. En este nuevo experimento se plantea la posibilidad que un nodo origen disponga de más de un camino hacia el mismo destino. Para ello

hemos utilizado la topología de la Figura 3.2, tomando el nodo 8 como único origen, y los nodos 0, 3, 4, 14, 13, 15, 16 y 5 sus destinos, formando los caminos 1, 2, 3, 4, 5, 6, 7 y 8 respectivamente, tal como se muestra en Tabla 3.4. Para la simulación hemos utilizado los parámetros de la Tabla 3.1 considerando α=β=0,25, y pérdidas $FPLn_7$=0,25 y $FPLn_9$=0,15.

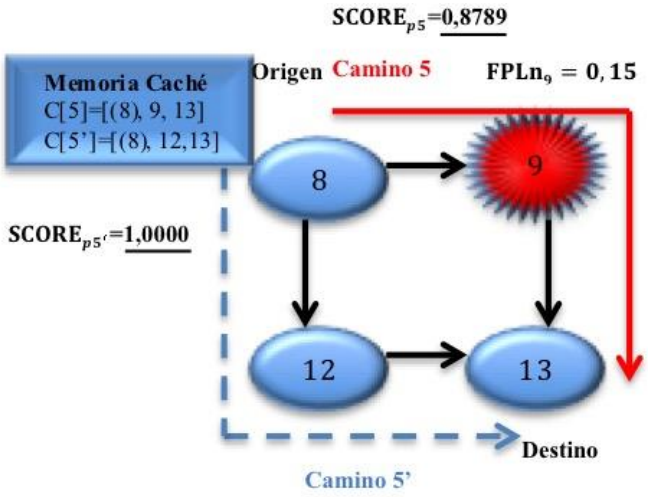


Figura 3.7: Calificaciones para caminos 5 y 5' calculadas con REMODE_sw

En este escenario el nodo 8, para transmitir datos al 13, tiene dos caminos disponibles tal como lo muestra la Figura 3.7. Ambos caminos tienen dos saltos para llegar al nodo 13. El protocolo DSR elige el primero de los caminos que descubre, en cambio el protocolo REMODE_sw compararía ambos caminos una vez descubiertos y compararía sus reputaciones para tomar la decisión. En la simulación la calificación para el camino 5 es 0,8789, mientras que para el camino 5' es de 1,0000, siendo este último la mejor opción.

Posición	Contenido de caché primaria	Camino	Calificación. Sim.
0	[(8) 9 3]	C_2	0,8724
1	[(8) 9 13]	C_5	0,8789
2	[(8) 12 13]	$C_{5'}$	1,0000
3	[(8) 7 6 5]	C_8	0,7439
4	[(8) 7 6 16]	C_7	0,7463
5	[(8) 12 11 15]	C_6	1,0000
6	[(8) 7 1 0]	C_1	0,7476
7	[(8) 9 10 4]	C_3	0,8124
8	[(8) 9 10 14]	C_4	0,8156

Tabla 3.4: Cache primaria del nodo origen 8 cuando tenemos múltiples caminos

3.5.4 Análisis de la reputación de los caminos activos, de la reputación de nodos y calificación de caminos en escenario móvil

Se presenta un cuarto escenario, utilizando los nodos y caminos definidos en la Figura 3.2, añadiendo a los nodos un patrón de movimiento *Random Waypoint* [**77**] con una rapidez de los nodos de 2m/s. Los parámetros de la simulación son los definidos en la Tabla 3.1, con α=β=0,25, así como $FPLn_7$=0,25 y $FPLn_9$=0,15. La posición inicial de los nodos es aleatoria y es la que se muestra en la Figura 3.8.

La Figura 3.9 muestra la reputación cambiante que presentan los caminos. Durante la simulación cada camino va cambiando de nodos intermedios, manteniendo su origen y destino. Cuando un camino cambia para incluir nodos no cooperativos, es notorio el decremento de su reputación.

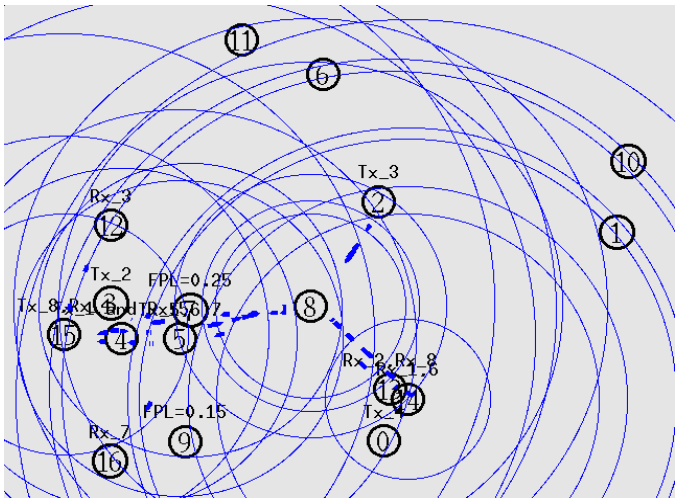


Figura 3.8: Escenario con patrón de movilidad *Random Waypoint* a 2 m/s

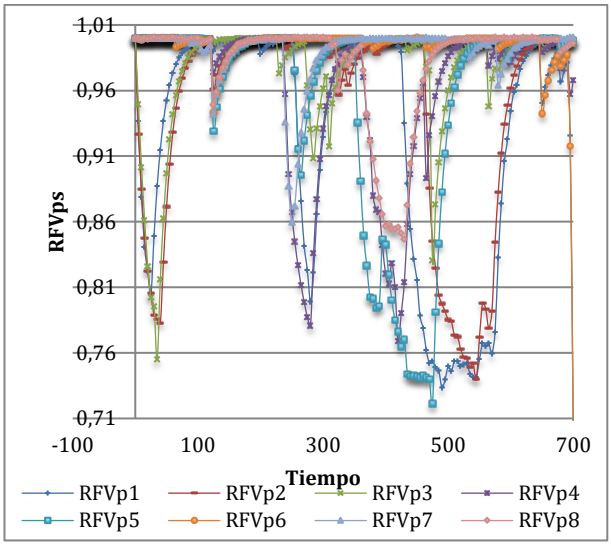


Figura 3.9: Reputación de los caminos activos

La Tabla 3.5 muestra los valores simulados promedio de $\overline{RFVn_{j0}}$ (reputaciones de los nodos vistas por el nodo origen 0). Este nodo se comunica con un único destino, el nodo 15, formando el camino 4 (Figura 3.2). En la tabla se observa que la mayoría de los nodos han participado en dicho camino de manera que su reputación ha cambiado (disminuido para los nodos colaborativos o aumentado para los egoístas). A pesar de ello los nodos con menor reputación son los que presentan un comportamiento egoísta, siendo el nodo 7 el de peor reputación.

Nodo	Val. Real.	Val. Sim.	Int. Conf. 95% de Val. Sim.
N_0			
N_0	1	1	(1,0000; 1,0000)
N_1	1	0,9712	(0,9754; 0,9939)
N_2	1	0,9990	(0,9976; 0,9992)
N_3	1	0,9729	(0,9710; 0,9762)
N_4	1	0,9993	(0,9965; 1,0007)
N_5	1	0,9924	(0,9903; 0,9929)
N_6	1	0,9802	(0,9252; 0,9566)
N_7	0,75	0,8837	(0,8899; 0,8986)
N_8	1	0,9993	(0,9931; 1,0002)
N_9	0,85	0,9610	(0,9448; 0,9549)
N_{10}	1	1	(0,9982; 1,0007)
N_{11}	1	0,9681	(0,9501; 0,9620)
N_{12}	1	0,9996	(0,9996; 1,0000)
N_{13}	1	0,9939	(0,9779; 0,9934)
N_{14}	1	0,9791	(0,9721; 0,9840)
N_{15}	1	1	(1,0000; 1,0000)
N_{16}	1	0,9995	(0,9945; 1,0007)

Tabla 3.5: Reputación observada por el nodo origen 0 cuando los nodos están en movimiento

La Figura 3.10 muestra la calificación que va obteniendo el camino 4 a lo largo de la simulación. Dicha calificación se inicializa a 1, presentando los picos más bajos cuando participa uno o dos nodos egoístas en el camino. La estabilización a un valor constante diferente de 1 se da cuando se produce una ruptura del enlace entre el nodo 0 y el 15.

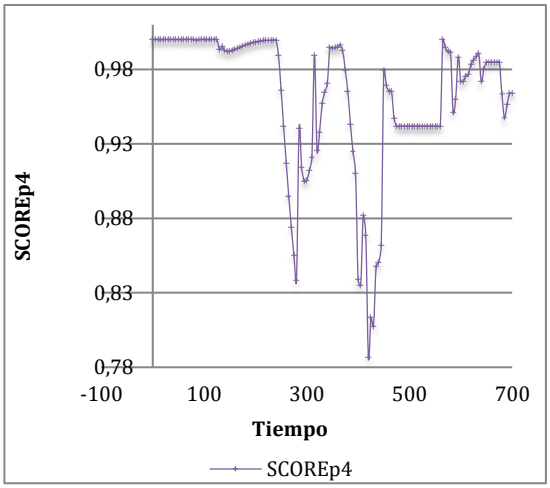


Figura 3.10: SCORE del camino activo observado por el nodo 0

La Tabla 3.6 muestra los valores simulados promedio de $\overline{RFVn_{j5}}$ (reputaciones de los nodos vistas por el nodo origen 5). En este caso la reputación del nodo 9 se mantiene en 1, debido a que el nodo 9 no participa aún como nodo intermedio en ninguna de las comunicaciones en las que el nodo 5 es el transmisor (caminos 5, 6 y 7). Como el nodo origen 5 es el transmisor en tres caminos distintos, entonces tendrá las calificaciones de los tres caminos activos como lo muestra la Figura 3.11.

De esta manera cada nodo origen tendrá las reputaciones de los nodos así como las calificaciones de los caminos, por tanto al igual que se han presentado los datos para el nodo transmisor 0 y 5, los nodos 4, 3, 2 y 15 también calcularán estos valores.

Nodo	Val. Real.	Val. Sim.	Int. Conf. 95% de Val. Sim.
N_5			
N_0	1	0,9867	(0,9820; 0,9883)
N_1	1	0,9541	(0,9516; 0,9759)
N_2	1	0,9987	(0,9950; 0,9978)
N_3	1	0,9998	(0,9973; 0,9998)
N_4	1	0,9979	(0,9942; 0,9980)
N_5	1	1	(1,0000; 1,0000)
N_6	1	0,9999	(0,9736; 0,9922)
N_7	0,75	0,9131	(0,9195; 0,9311)
N_8	1	0,9960	(0,9948; 0,9973)
N_9	0,85	1	(1,0000; 1,0000)
N_{10}	1	1	(0,9937; 1,0017)
N_{11}	1	1	(1,0000; 1,0000)
N_{12}	1	0,9998	(0,9909; 1,0005)
N_{13}	1	0,9742	(0,9793; 0,9876)
N_{14}	1	0,9989	(0,9770; 0,9921)
N_{15}	1	0,9996	(0,9972; 0,9999)
N_{16}	1	0,9998	(0,9969; 0,9985)

Tabla 3.6: Reputación observada por el nodo origen 5 cuando los nodos están en movimiento

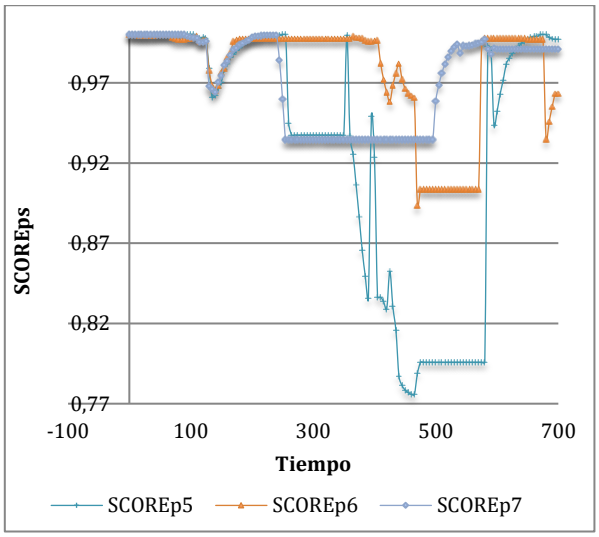


Figura 3.11: SCORE para los caminos activos observados por el nodo 5

La principal desventaja del algoritmo está en el hecho que un nodo con baja reputación participará en menos caminos que uno de mayor reputación, lo cual dificulta que su reputación mejore en caso que el nodo mejore su comportamiento en la red. Es por ello, que inicialmente la reputación de todos los nodos es la unidad. Una posible solución a este hecho es que los nodos olviden periódicamente las estimaciones de reputación que han hecho y las reinicialicen.

3.5.5 Análisis de los valores de reputación de nodos cuando se tiene tráficos asimétricos

Para este quinto experimento en escenario estático, se utiliza la Fig. 3.2 con los parámetros de la Tabla 3.1 considerando que en lugar de utilizar DSR con nodos egoístas, se utiliza REMODE_sw con nodos egoístas; considerando además que en lugar de un tiempo de muestreo "cada 5 segundos" se requiere de un intervalo de muestreo "cada 500 paquetes". Debido a que el escenario es estático no se requiere intervalos de muestreo más cortos. Se configura un valor de α=0,25 y de β=0,15. Al experimento se lo realiza con la finalidad de demostrar que la Ec. 3.7 se verifica tanto para tráficos simétricos como asimétricos.

En la Tabla 3.7 nos referimos a un tráfico simétrico $\gamma_6 = \gamma_5$, cuando el nodo fuente 5 por los camino 5, 6 y 7 transmite tráfico CBR a una velocidad de 1Mbps. En la Tabla 3.8, nos referimos a un tráfico asimétrico $\gamma_6 = \frac{1}{2}\gamma_5$ cuando el nodo fuente 5 por los caminos 5 y 7 transmite a 1Mbps y por el camino 6 a 0.5Mbps. Finalmente, en la Tabla 3.9, nos referimos a un tráfico asimétrico $\gamma_6 = 2\gamma_5$ cuando el nodo fuente 5 por los caminos 5 y 7 transmite a 1Mbps y por el camino 6 a 2Mbps.

Nodo	Val. Real	Val. Teór.	Val. Sim. (promedio)	Int. Conf. 95% de Val. Sim. (promedio)
N_5				
N_6	1	0,9426	0,9455	(0,9436; 0,9473)
N_7	0,75	0,9139	0,9140	(0,9136; 0,9144)
N_8	1	0,9139	0,9140	(0,9136; 0,9144)
N_9	0,85	0,9139	0,9140	(0,9136; 0,9144)
N_{10}	1	0,9139	0,9140	(0,9136; 0,9144)

Tabla 3.7: Reputación de nodos observada por el nodo origen 5 con $\gamma_6 = \gamma_5$

Nodo	Val. Real	Val. Teór.	Val. Sim. (promedio)	Int. Conf. 95% de Val. Sim. (promedio)
N_5				
N_6	1	0,9483	0,9524	(0,9509; 0,9538)
N_7	0,75	0,9139	0,9139	(0,9137; 0,9142)
N_8	1	0,9139	0,9139	(0,9137; 0,9142)
N_9	0,85	0,9139	0,9139	(0,9137; 0,9142)
N_{10}	1	0,9139	0,9139	(0,9137; 0,9142)

Tabla 3.8: Reputación de nodos observada por el nodo origen 5 con $\gamma_6 = \frac{1}{2} * \gamma_5$

Nodo	Val. Real	Val. Teór.	Val. Sim. (promedio)	Int. Conf. 95% de Val. Sim. (promedio)
N_5				
N_6	1	0,9354	0,9369	(0,9357; 0,9382)
N_7	0,75	0,9139	0,9139	(0,9136; 0,9141)
N_8	1	0,9139	0,9139	(0,9136; 0,9141)
N_9	0,85	0,9139	0,9139	(0,9136; 0,9141)
N_{10}	1	0,9139	0,9139	(0,9136; 0,9141)

Tabla 3.9: Reputación de nodos observada por el nodo origen 5 con $\gamma_6 = 2 * \gamma_5$

Cuando $\gamma_6 = \gamma_5$ la FPL total (de caminos de todo el escenario) es 0.1401, cuando $\gamma_6 = \frac{1}{2} * \gamma_5$ la FPL total es 0.1256, y cuando $\gamma_6 = 2 * \gamma_5$ la FPL total es 0.1643, lo cual indica que cuando el tráfico del camino 6 no cooperativo baja a la mitad la RFV_{n6} aumenta y la FPL total disminuye; y, por el contrario cuando el tráfico del camino 6 no cooperativo se duplica la RFV_{n6} disminuye y la FPL total aumenta, debido a que opaca la influencia del camino 7 que es cooperativo.

En los experimentos anteriores únicamente se calcula los valores de reputación, en este y próximos experimentos estos valores son utilizados para seleccionar el camino de mayor reputación, con lo cual se da lugar al encaminamiento mediante el protocolo REMODE_sw.

3.5.6 Análisis de la recuperación de la reputación de un nodo en un escenario móvil

Para este sexto experimento, igualmente se utiliza las conexiones de la Fig. 3.2 con los parámetros de la Tabla 3.1 considerando que en lugar de utilizar un tiempo de muestreo "cada 5 segundos" se requiere de un intervalo de muestreo "cada 10 paquetes". En este escenario móvil con patrón de movimiento *Random Waypoint* con rapidez de nodos de 2m/s, se configura un valor de α=1 y de β=0,6. Al experimento se lo realiza con la finalidad de demostrar que sin importar el valor de inicialización de la reputación de un nodo, al ser un escenario móvil, el nodo recupera su valor verdadero de reputación en muy poco tiempo.

De las 8 conexiones de la red móvil ad-hoc, analizamos la tercera que va del nodo 2 al nodo12, entonces analizaremos la reputación simulada que el nodo fuente 2 tiene del nodo intermedio 7. La Figura 3.12 muestra que primeramente la reputación del nodo intermedio 7 en el nodo fuente 2 ha sido inicializada en 1, luego en 0,5 y finalmente en 0; y, a los 18 segundos de simulación la RFV_{n_7} observada por el nodo fuente 2, tiene un valor de 0,9764 para los tres casos, independientemente del valor que se inicialice, con lo cual para este escenario móvil, sin importar a que valor se inicialice la reputación del nodo 7, el nodo 2 a los 18 segundos habrá recuperado su valor de reputación y podrá seguir calculando sus nuevos valores de reputación en base a las distintas topologías que se vayan formando en el escenario, y es así que a los 700 segundos la RFV_{n_7} observada por el nodo fuente 2 para los tres casos es 0,9419. Entonces en la Figura 3.12 se observa que desde los 18 a los 700 segundos los valores de reputación para el nodo 7 son los mismos en los tres casos.

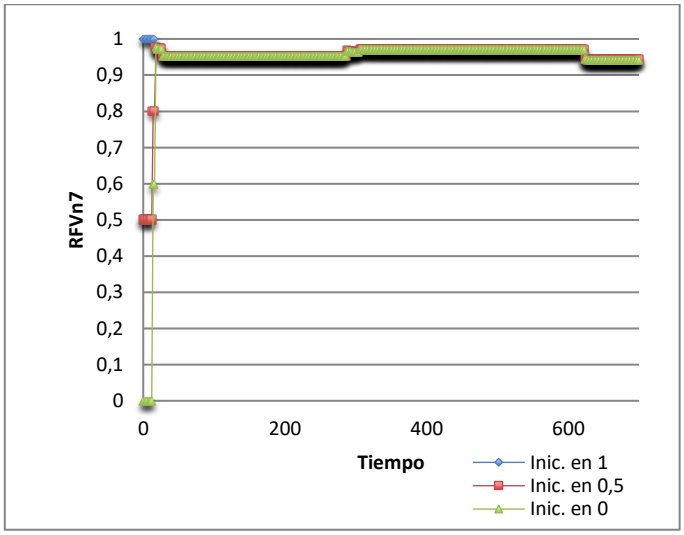


Figura 3.12: RFV_{n_7} recuperando su valor verdadero

3.6 Conclusiones

El objetivo del capítulo fue proponer un algoritmo de protocolo de encaminamiento basado en la reputación de peso estático de los nodos de la red móvil ad-hoc, evaluar, analizar y comparar las mejoras propuestas; y, luego de desarrollado el capítulo las conclusiones son las siguientes:

- En este capítulo hemos propuesto un algoritmo de encaminamiento que permite encontrar los caminos que incluyen a los nodos más cooperativos de una red móvil ad-hoc. Dicho algoritmo utiliza exclusivamente información de primera mano a partir de la experiencia de cada nodo en la red, lo cual simplifica su uso. Cada nodo de la red estima la reputación de reenvío del resto de nodos. Para ello es necesario el conocimiento de las pérdidas en cada camino que utiliza el nodo que hace la estimación, repartiéndolas por igual entre los nodos que forman cada camino. Ello conlleva que en un mismo camino, los nodos que tienen un comportamiento más colaborativo ven penalizada su reputación mientras que los que tienen un comportamiento más egoísta la ven recompensada. Para paliar este efecto, el cálculo se realiza promediando, mediante una media móvil, la reputación estimada, para cada uno de los caminos conocidos en los que el nodo participa. Así pues, la reputación de reenvío de un nodo, no sólo depende de su comportamiento con los reenvíos, sino que además está afectada por el comportamiento de sus nodos vecinos. Este efecto, que a priori puede parecer injusto, facilita el encaminamiento por caminos que incluyan a nodos más colaborativos.
- La principal desventaja del algoritmo está en el hecho que un nodo con baja reputación participará en menos caminos que uno de mayor reputación, lo cual dificulta que su reputación mejore en caso que el nodo mejore su comportamiento en la red. Es por ello, que inicialmente la reputación de todos los nodos es la unidad. Una posible solución a este hecho es que los nodos olviden periódicamente las estimaciones de reputación que han hecho y las reinicialicen.
- En este trabajo hemos presentado una expresión analítica para calcular la estimación de reputación que realizará cada nodo. Dicha estimación depende de la topología de la red y de los tráficos cursados por los distintos caminos.
- Además hemos implementado el algoritmo en el simulador NS-2 para comparar los resultados obtenidos con los resultados teóricos. Hemos modificado el comportamiento de algunos de los nodos de la red (generando pérdidas aleatorias) para que no reenvíen todos los paquetes que reciben.

- En el entorno de simulación hemos analizado la influencia de los parámetros α y β en las medias móviles con ponderación exponencial (EWMA). Valores pequeños de estos parámetros se traducen en una adaptación lenta en el caso que hayan variaciones significativas en el comportamiento de los nodos. Ello también afecta al transitorio inicial de manera que éste es mayor y se tarda más en tener un valor fiable de la reputación de los nodos. Dicho transitorio no parece verse afectado por el valor asintótico al que se tiende, sino únicamente a los valores de α y β. Debido al hecho que la estimación de la reputación de los nodos se hace a partir de la reputación de los caminos, si dicha reputación se actualiza lentamente (valor de α pequeño), la reputación de los nodos también se actualiza lentamente independientemente del valor de β. Por otro lado, valores mayores de estos parámetros dan como resultado valores menos estables en la estimación de la reputación presentando ésta una mayor varianza.
- También hemos simulado escenarios con múltiples caminos a un mismo destino, los cuales son descubiertos por el protocolo DSR y guardados en la memoria caché del nodo origen. Hemos estimado la reputación de dichos caminos observando que aunque los valores estimados de reputación para cada camino no se corresponden con la reputación real, los caminos con mayor reputación son los que incluyen el menor número de nodos con comportamiento egoísta. Es decir, a pesar que los valores de reputación no son exactos, el camino elegido sí es el que presenta mayor reputación real, lo cual valida el planteamiento del algoritmo propuesto.
- Además, hemos simulado el modelo en un escenario con movilidad, en donde hemos verificado que los nodos egoístas presentan una reputación más baja que el resto, y si un nodo no participa en ningún camino mantiene una reputación de 1.
- Hemos realizado simulaciones haciendo que REMODE_sw encamine y hemos verificado en escenario estático que con tráficos asimétricos se verifica el modelo planteado.
- Finalmente hemos realizado experimentos con REMODE_sw en escenario móvil que indican claramente que en escenario móvil sin importar el valor de inicialización de la reputación de un nodo, se consigue recuperar el valor

verdadero de reputación que un nodo fuente tiene de determinado nodo intermedio.

Capítulo 4

DrepSR (Dynamic reputation-based Source Routing)

"La inteligencia consiste no solo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica".
—Aristóteles. Filósofo y científico griego. 384-322 a.c.

"Experimenta, falla, aprende y repite".
—Steven Paul Jobs. Cofundador y presidente ejecutivo de Apple Inc. 1955-2011

4 DrepSR (*Dynamic reputation-based Source Routing*)

En el presente capítulo se muestra la teoría y experimentos realizados con esta nueva propuesta denominada DrepSR(Dynamic reputation-based Source Routing).

El encaminamiento en redes móviles ad-hoc (MANETs) se basa en la cooperación de los nodos de la red. La presencia de nodos egoístas que no cooperan en esta tarea reduce drásticamente el número de paquetes entregados. Para encontrar los mejores caminos que incluyan nodos dispuestos a cooperar, proponemos un nuevo algoritmo de encaminamiento basado en la reputación de los nodos. En nuestra propuesta, cada nodo asigna localmente un valor de reputación al resto de los nodos en la red y luego utiliza los valores de reputación asignados para descubrir los mejores caminos de encaminamiento, con el fin de minimizar la fracción total de pérdida de paquetes. Suponemos que los nodos tienen un comportamiento de encaminamiento estacionario, pero también incluimos un mecanismo para detectar cambios en su comportamiento. Nuestro enfoque ha sido evaluado en presencia de nodos egoístas, con el fin de compararlo con el algoritmo de encaminamiento de origen dinámico, obteniendo una reducción en la fracción de pérdida de paquetes a expensas de un pequeño aumento en el número de saltos tomados por los paquetes para llegar a sus destinos

4.1 Introducción

Una red móvil ad-hoc (MANET) es un grupo de nodos móviles (MN) inalámbricos autoorganizados capaces de comunicarse entre sí sin la necesidad de ninguna infraestructura de red fija ni soporte administrativo centralizado. Además, el alcance de transmisión en tales dispositivos móviles es limitado; por lo tanto, un paquete se reenvía en un camino de multiples saltos confiando en los nodos en el camino de encaminamiento. Debido a que, las redes móviles ad-hoc necesitan la cooperación de cada nodo en el camino para lograr una entrega de paquetes exitosa. Sin embargo, dependiendo de la aplicación de la red móvil ad-hoc, los nodos están dispuestos a cooperar entre sí (por ejemplo, servicios de rescate) ya que están controlados por una autoridad, o podrían ser reacios a cooperar (por ejemplo, compartir datos [**80**], monitoreo de tráfico [**81**], servicios de asistencia de emergencia [[**82**], [**5**]] y transmisión de datos multimedia [**83**]) tratando de ahorrar sus propios recursos. Dado que los nodos de la red móvil ad-hoc suelen tener una potencia limitada (es decir, batería) y escasos recursos computacionales (CPUs), los nodos podrían negarse a cooperar para ahorrar sus recursos limitados. Este mal comportamiento de los nodos afectaría drásticamente la operación del protocolo de encaminamiento.

Vale la pena mencionar que la presencia de unos pocos nodos egoístas podría degradar gravemente el rendimiento de la red móvil ad-hoc [**84**]. Por lo tanto, la detección de nodos egoístas es crucial para garantizar una operación de encaminamiento de la red móvil ad-hoc efectiva y eficiente.

En este capítulo, nuestro objetivo es mejorar el rendimiento general de las redes móviles ad-hoc en presencia de nodos egoístas mediante el diseño de un algoritmo eficiente y dinámico basado en la reputación que se utilizará en cualquier protocolo de encaminamiento ad-hoc. Nos gustaría resaltar el hecho de que nuestro algoritmo basado en la reputación se puede aplicar a cualquier protocolo de encaminamiento para redes ad-hoc que establezcan caminos de reenvío de extremo a extremo. Utilizando nuestro enfoque, el protocolo de encaminamiento establecerá el camino de reenvío con la mejor reputación de los nodos. Introducimos una métrica simple para estimar la reputación de los nodos de acuerdo con su comportamiento de reenvío de paquetes.

Vale la pena señalar que en este trabajo nos referimos a la reputación del nodo como su disponibilidad para reenviar paquetes. Debido a la naturaleza de su operación sin infraestructura, la funcionalidad de las redes ad-hoc depende en gran medida de la colaboración entre los nodos. Sin embargo, los nodos egoístas intentan ahorrar batería al no reenviar paquetes de otros. Por lo tanto, evitar los nodos egoístas que no pretenden colaborar conduce a asegurar un rendimiento de red adecuado. Recordemos que un nodo colaborativo que temporalmente causa pérdidas de paquetes debido a, por ejemplo, congestión de la red o colisión en el medio de transmisión, pronto recuperará su reputación cuando se mueva a otra área mejor. Por lo tanto, nuestro objetivo es detectar nodos con un comportamiento egoísta persistente. Queda fuera del alcance de la propuesta definir cualquier mecanismo para recompensar o penalizar a los nodos de la red con el fin de forzar su cooperación. En realidad, nuestro enfoque consiste en una estrategia de reenvío basada en la reputación para los protocolos de encaminamiento ad-hoc que evita la utilización de caminos de reenvío que incluyen nodos egoístas que no reenviarían paquetes correctamente. Así, la calidad del servicio ofrecido al usuario mejora.

A continuación, destacamos las novedosas aportaciones de nuestro trabajo:

- Proponemos un nuevo método para calificar el comportamiento de reenvío de los nodos de la red móvil ad-hoc basándose únicamente en la información de primera mano de cada nodo fuente. Esta característica es congruente con el hecho de que en las redes sin infraestructura, como las redes móviles ad-hoc, los nodos solo recopilan información local.
- Proponemos un método novedoso basado en la conocida media móvil exponencial (EWMA) [**85**] para detectar cualquier cambio en el comportamiento de reenvío de los nodos. La relación señal a ruido pico de la segunda diferencia de la media móvil exponencial se calcula para establecer los parámetros de la media móvil exponencial que optimice aquella relación señal a ruido pico. Esto nos permite detectar los límites de la ventana de tiempo estacionario en la cual la reputación de un nodo permanece estática. De esta manera podemos actualizar adecuadamente la medida del comportamiento de reenvío de un nodo promediando las muestras de reputación en esas ventanas de tiempo.
- Además, proponemos una estrategia de reputación basada en estimar la reputación de los caminos con el objetivo de encontrar el mejor camino de reenvío que utilizará el nodo emisor. Nuestra novedosa estrategia de reenvío basada en la reputación para los protocolos de encaminamiento ad-hoc evita la utilización de caminos de reenvío que incluyen nodos egoístas que no reenvíen paquetes correctamente, por lo que el porcentaje de pérdida de paquetes disminuye. La mayoría de los protocolos de encaminamiento ya propuestos en la literatura utilizan una métrica basada en la distancia mínima. Alternativamente, nuestra propuesta utiliza una métrica basada en la reputación. Esta reputación se estima utilizando solo información local, evitando así tener que depender de las opiniones de terceros.

Además:

- Analizamos diferentes técnicas de media móvil para reducir el ruido de las calificaciones muestreadas de reputación y poder asignar un valor de reenvío de reputación real a los nodos de la red.

- Llevamos a cabo varias simulaciones para comprobar la exactitud de los resultados teóricos.

Finalmente, probamos la estrategia de reenvío propuesta que opera a través del conocido protocolo de encaminamiento de fuente dinámica (DSR) [**86**]. Es importante tener en cuenta que la estrategia propuesta se puede aplicar fácilmente a cualquier protocolo de encaminamiento de extremo-a-extremo. Los resultados muestran una reducción significativa de las pérdidas de paquetes en la red a expensas de un pequeño aumento en el número de saltos tomados por los paquetes para llegar a sus destinos.

El resto del capítulo está estructurado de la siguiente manera: En primer lugar, la sección 4.1 incluye algunos trabajos relevantes relacionados con los sistemas de reputación en redes móviles ad-hoc. Luego, las sección 4.2 brinda una breve explicación del protocolo de encaminamiento propuesto basado en la reputación. A continuación, las sección 4.3 describe el análisis matemático de las diferentes propuestas para asignar un valor de reputación a los nodos de la red móvil ad-hoc. Después de eso, los resultados experimentales y la discusión se muestran en la sección 4.4. Finalmente, las conclusiones y trabajo futuro se dan en la sección 4.5

En cuanto a métodos y experimentos tenemos: en las secciones 4.2 presentamos nuestra propuesta, mientras que en la sección 4.4 validamos el modelo y mostramos los resultados de la simulación. En la sección 4.2 presentamos nuestro enfoque de un protocolo de encaminamiento basado en la reputación para redes móviles ad-hoc, denominado *dynamic reputation-based source routing* (DrepSR). DrepSR incluye algoritmos para calcular la reputación de nodos y caminos. La Sección 4.3 incluye nuestra metodología para promediar los valores de reputación de reenvío de los nodos, utilizando una media móvil acumulada (CMA) para promediar las muestras de reputación, y una media móvil exponencial (EWMA) para detectar cambios de comportamiento en los nodos. Las metodologías utilizadas para validar nuestra propuesta son:

- Hemos desarrollado un cálculo matemático para conseguir la relación señal a ruido pico de la segunda diferencia de una media móvil exponencial. Utilizamos este cálculo para definir el parámetro de la media móvil exponencial

que maximiza la relación señal a ruido pico y optimiza los cambios de comportamiento en los nodos de una red móvil ad-hoc (ver subsección 4.3.3).

- Realizamos simulaciones en una topología estática para validar el modelo matemático desarrollado (ver subsección 4.4.1).

Una vez que nuestro modelo teórico es validado correctamente en la subsección 4.4.1, mostrando un buen nivel de precisión, llevamos a cabo una evaluación del desempeño en la subsección 4.4.2. Para ello, presentamos:
Un conjunto representativo de simulaciones de una red móvil ad-hoc para diferentes escenarios con nodos móviles, en el que nuestra propuesta DrepSR se compara con el protocolo DSR. Los detalles completos de esta evaluación de desempeño se describen en la subsección 4.4.2, incluidos los ajustes de simulación (consulte la Tabla 4.1) y los resultados de la simulación (consulte la Tabla 4.2).

4.1.1 Trabajos relacionados

La cuestión cooperación de los nodos en la red móvil ad-hoc ha recibido mucha atención por parte de la comunidad de investigación en los últimos años. En general, los sistemas de reputación se pueden clasificar en dos categorías: *esquemas basados en precios* y *esquemas basados en reputación*. A continuación presentamos algunos trabajos representativos relacionados con cada categoría.

Los esquemas basados en precios tratan los servicios de reenvío de paquetes como transacciones por las que se puede pagar, e introducen beneficios virtuales como una forma de recompensa a los nodos que han estado participando en las actividades de reenvío de paquetes. Esto significa que los nodos obtienen créditos virtuales a cambio de ofrecer reenvío de paquetes. El artículo [**87**] propone un mecanismo de estímulo basado en un contador en cada nodo de la red móvil ad-hoc. El contador aumenta cuando el nodo reenvía paquetes para otros y disminuye, siguiendo una función que depende del conteo estimado de saltos al destino, cuando el nodo envía sus propios paquetes. Un esquema de micropago para redes celulares de múltiples saltos se presenta en [**88**] para fomentar la colaboración en el reenvío de paquetes al permitir que los usuarios se beneficien al transmitir los paquetes de otros. Además, proponen mecanismos para detectar y recompensar la colaboración, al tiempo que detectan y castigan el engaño. El trabajo [**89**] presenta un mecanismo de incentivo equitativo y

eficiente para estimular la cooperación de nodos en las redes móviles ad-hoc mediante el cobro de los nodos de origen y destino cuando ambos se benefician de la comunicación. Para implementar esta política de tarificación eficiente, operaciones hash se utilizan en los paquetes de acuse de recibo (ACK) para reducir el número de operaciones. También se ha señalado que los nodos de la red móvil ad-hoc deben cooperar y reenviar paquetes para otros; de lo contrario, sería imposible lograr una red ad-hoc pura [**90**].

Los esquemas basados en la reputación se centran básicamente en evaluar el comportamiento de cada nodo y en detectar los nodos que se comportan mal de acuerdo con sus valores de reputación. En [**84**], los autores proponen un método de vigilancia para identificar los nodos que se comportan mal y una técnica de clasificación de caminos que ayuda a los protocolos de encaminamiento para evitar elegir esos nodos para reenviar cualquier paquete. Los resultados de la simulación demuestran los beneficios de utilizar estas dos técnicas al aumentar el rendimiento al menos al 17%. Los autores en [**58**] proponen un protocolo llamado CONFIDANT (cooperación de nodos: equidad en redes dinámicas ad-hoc) capaz de detectar y aislar nodos que se comportan mal. Muestran que al utilizar CONFIDANT, una red puede funcionar bien incluso con un alto porcentaje de nodos maliciosos. En [**69**], los autores presentan un mecanismo genérico basado en la reputación para imponer la cooperación entre los nodos de la red móvil ad-hoc para evitar comportamientos egoístas. Los autores en [**68**] proponen la aplicación de la cooperación basada en la observación en redes ad-hoc (OCEAN) para detectar y mitigar el comportamiento de encaminamiento engañoso en redes ad-hoc. Los resultados de la simulación muestran que OCEAN funciona muy bien en términos de rendimiento. Sin embargo, OCEAN no es completamente capaz de penalizar el mal comportamiento de los nodos a pesar de que se desempeña al menos igual de bien en comparación con otros esquemas más complejos. La propuesta [**91**] emplea un sistema basado en la confianza para contrarrestar los comportamientos de los nodos maliciosos que consiste en un sistema de reputación y una técnica de vigilancia. Su técnica de vigilancia utiliza un mensaje de retroalimentación positiva (positive feedback message-PFM) como evidencia del comportamiento del nodo de reenvío. Con respecto a [**92**], los autores propusieron una gestión de confianza basada en la reputación para detectar y prevenir las vulnerabilidades de la red móvil ad-hoc. Los

resultados de su evaluación de desempeño muestran escalabilidad y robustez utilizando el esquema propuesto.

En [**93**], los autores proponen una nueva estrategia de encaminamiento que funciona con nodos que se consideraron confiables, pero debido a la pérdida de potencia, se reconsideran como egoístas. Su estrategia considera que los nodos egoístas no son maliciosos por defecto, y se ven obligados a proporcionar su nivel de energía actual. Esto ayuda a la estrategia de encaminamiento para evitar nodos con una potencia muy baja basándose en el supuesto de que un nodo con poca energía podría dejar caer paquetes para ahorrar su consumo de energía. La propuesta [**94**] propone un mecanismo de confianza basado en un modelo de vector de confianza (trust vector model-TV). El objetivo principal de este mecanismo es detectar nodos maliciosos para luego tomar medidas sobre ellos. Se implementa en dos protocolos de encaminamiento conocidos: encaminamiento de origen dinámico (DSR) [**86**] y vector de distancia bajo demanda ad-hoc (AODV) [**95**], produciendo dos propuestas llamadas TV-DSR y TV-AODV, respectivamente. Los resultados de la simulación en ambos protocolos de encaminamiento modificados muestran que pueden detectar nodos maliciosos de manera efectiva y mitigar sus ataques. Los autores en [**96**] proponen un nuevo mecanismo para detectar nodos egoístas. Consideran como una buena estrategia que los nodos egoístas eliminen los paquetes de control para evitar que se les pida que envíen datos y de esta manera ahorran recursos para su propio uso. Se espera que los nodos contribuyan a la red dentro de un marco de tiempo. Aquellos nodos que no lo hagan, se someterán a una prueba por su comportamiento sospechoso. Los resultados de simulación apoyan su esquema al obtener buenos resultados.

La teoría de juegos y el diseño de mecanismos se han aplicado ampliamente para encaminamiento egoísta. Los autores en [**97**] proponen un marco de reciprocidad indirecta basado en una solución de teoría de juegos para reforzar la cooperación entre nodos. Sobre la base del modelo propuesto, obtienen el umbral de la relación costo-beneficio para garantizar la convergencia de la cooperación. Los resultados de la simulación demuestran que su solución de teoría de juego refuerza la cooperación entre los nodos cuando la relación costo-beneficio del no egoísta supera la condición crítica. La propuesta [**98**] aborda la reducción del consumo de energía de los nodos inalámbricos con limitaciones de energía a través de un esquema de retransmisión

cooperativa consciente del uso de la energía con una distribución justa de la recompensa entre los jugadores, para mantenerlos satisfechos y desalentarlos a abandonar la alianza. Esta solución también propone un sistema basado en el crédito para recompensar a los jugadores cooperativos, por lo que excluye a los usuarios egoístas de las alianzas cooperativas.

El enfoque presentado en [**99**] consiste en un sistema de gestión de reputación dinámico para detectar y aislar nodos que se comportan mal en redes móviles ad-hoc. Introducen una novedosa técnica de monitoreo directo que evalúa la reputación de los nodos en la red. Esta técnica garantiza que los nodos que se comportan mal se detecten y aíslen de la red, mientras que el resto de los nodos que gastan su energía en el reenvío de datos podrán realizar sus actividades de red. Los resultados de la simulación muestran la efectividad de su modelo para restringir y mitigar los efectos de los nodos que se comportan mal en las redes móviles ad-hoc. Los autores en [**100**] muestran cómo el uso de su propuesta llamada detección de compresión en el vecindario (neighborhood compressive sensing-NCS) ayuda a reducir el consumo de recursos y al mismo tiempo protege la red de ataques y mal comportamiento. El modelo NCS comprime datos dispersos como actualizaciones de tablas de encaminamiento. Además, como los nodos individuales solo aceptan actualizaciones de las tablas de encaminamiento si provienen de la información procesada del nodo líder, NCS también evita que la red sufra ataques y mal comportamiento. Los resultados de la simulación muestran cómo el modelo NCS supera a DSR en términos de consumo de energía, vida útil y fracción de pérdida de paquetes.

Hasta donde sabemos, el enfoque de considerar la reputación de los nodos de la red móvil ad-hoc candidatos para conformar el esquema de reenvío, utilizando solo información de primera mano de los nodos para calificar su reputación y utilizar un modelo de media móvil exponencial para detectar cualquier cambio en la reputación de los nodos, es novedoso. En la siguiente sección, presentamos nuestra propuesta de un protocolo de encaminamiento basado en reputación para redes móviles ad-hoc.

4.2 Protocolo de encaminamiento propuesto basado en reputación

Nos gustaría resaltar el hecho de que nuestro esquema de reenvío basado en la reputación podría implementarse en cualquier protocolo de encaminamiento ad-hoc que

realice un esquema de reenvío de extremo a extremo. En aras de la simplicidad, en este trabajo hemos seleccionado el conocido protocolo DSR, aunque se podrían obtener beneficios similares sobre otros protocolos de encaminamiento ad-hoc. Brevemente, DSR [**86**] busca caminos formados por nodos intermedios cuando una fuente necesita enviar un paquete a un destino. Para hacerlo, DSR envía periódicamente paquetes de descubrimiento de camino para buscar caminos a ese destino. Los paquetes de contestación de camino contienen el camino completo para ese paquete. Para completar el encaminamiento fuente, los paquetes encaminados contienen la dirección de cada nodo que atravesará el paquete. En el caso de roturas de enlace, se generan paquetes de error de camino y se encontrarán caminos alternativos buscando caminos almacenados en la caché de caminos o buscando nuevos caminos.

Nuestra propuesta, denominada dynamic reputation-based source routing (DrepSR), utiliza DSR como motor de protocolo de encaminamiento para descubrir los caminos disponibles hacia el destino. Además, DrepSR estima la reputación de los caminos descubiertos. Luego, cuando se debe elegir un camino almacenado en caché, DrepSR selecciona el más confiable en lugar del más corto. El objetivo es elegir, entre el conjunto de caminos disponibles, aquellos caminos que producen la menor cantidad de pérdidas de paquetes. Suponemos que la causa principal de la pérdida de paquetes es el comportamiento de no reenvío de los nodos egoístas presentes en la red. A medida que aumentan los servicios móviles (por ejemplo, video bajo demanda, juegos, redes sociales ...), los usuarios están preocupados por la duración de la batería de sus teléfonos inteligentes y tratan de ahorrar batería. La retransmisión de paquetes para otros consume batería, por lo que los usuarios pueden sentirse reacios a cooperar en las tareas de reenvío debido a la preocupación por ofrecer su batería de energía limitada. De todos modos, aunque hubo otros factores para la pérdida de paquetes (por ejemplo, congestión, colisiones), también sería beneficioso utilizar nuestro algoritmo propuesto. En DrepSR, cada nodo asigna localmente un valor de reputación a los otros nodos con los que interactúa, utilizando solo información de primera mano basada en su propia experiencia. Después de eso, las reputaciones de los caminos disponibles en caché se calculan utilizando la reputación estimada de los nodos individuales que forman esos caminos. La reputación de un camino se calcula fácilmente como el producto de la reputación de esos nodos intermedios que forman ese camino. De esta manera, el

algoritmo asigna una calificación de reputación a los caminos descubiertos por el mecanismo DSR. Finalmente, cada vez que un nodo requiera enviar información a un destino específico y descubra más de un camino disponible, el nodo elegirá el camino con la mejor reputación.

Para estimar la reputación de un nodo, es necesario tener una retroalimentación del porcentaje de pérdidas de paquetes en el camino al que pertenece el nodo. De hecho, si bien un nodo de origen no tiene experiencia previa en la red, no puede estimar la reputación de los otros nodos. Inicialmente, el nodo fuente estima la fracción de pérdidas de paquetes (L_p) para cada camino p a través del cual se transmite. Para obtener el porcentaje de extremo a extremo de las pérdidas de paquetes, se utiliza la información de realimentación de la capa de transporte. En el caso de comunicaciones no orientadas a la conexión, se asume el uso de RTP/RTCP (Real Time Protocol/Real Time Control Protocol) sobre UDP (User Datagram Protocol). RTCP proporciona el porcentaje de pérdidas de paquetes a través de cada camino disponible. Para el caso de las comunicaciones orientadas a la conexión, el protocolo más utilizado es TCP (Transmission Control Protocol). En este caso, las pérdidas se estiman a partir del número de retransmisiones realizadas por el nodo fuente con respecto a un mismo mensaje. La razón principal para hacer esta elección es su simplicidad, y aunque puede parecer una estimación aproximada, es lo suficientemente precisa para seleccionar el mejor camino en la red. Con esta información de retroalimentación sobre las pérdidas de paquetes, el nodo calcula L_p como la relación entre la cantidad de paquetes perdidos y la cantidad de paquetes enviados a través del camino p. De esta manera, nuestra propuesta de capa cruzada utiliza información de transporte para tomar decisiones de reenvío en la capa de red. La L_p es calculada periódicamente, actualizando entonces la reputación de los nodos que pertenecen a ese camino.

4.2.1 Algoritmo para calcular la reputación de nodos y caminos

Una vez que se ha realizado la estimación de la fracción de las pérdidas de paquetes para un camino, la reputación de los nodos intermedios se estima compartiendo las pérdidas de manera equitativa entre los nodos. Es decir, el algoritmo considera que todos los nodos de un camino pierden la misma cantidad de paquetes, lo que no siempre es cierto. Por lo tanto, si un nodo de buen comportamiento pertenece a un camino en el

que hay varios nodos egoístas, su reputación será penalizada. Aunque puede parecer injusto desde el punto de vista de los nodos, no lo es, ya que el objetivo del algoritmo no es estimar un valor de reputación exacto, sino elegir el camino con menos pérdidas. En consecuencia, el diseño de esquemas para recompensar o penalizar nodos de modo que podamos forzar su cooperación en las tareas de reenvío no es nuestro objetivo en este documento. De todos modos, es importante tener en cuenta que debido a la movilidad inherente de los nodos de la red móvil ad-hoc, un nodo cooperativo penalizado erróneamente por haber estado en el misma camino de reenvío que un nodo egoísta pronto recuperará su reputación cuando se mueva a otra área.

El valor de reputación de reenvío (r_n) asignado a un nodo n representa la probabilidad de reenvío para el nodo n, es decir, la relación entre el número de paquetes reenviados por el nodo y el número total de paquetes que ha recibido. De la misma manera, definimos el valor de reputación (R_p) asignado al camino p como la relación entre los paquetes recibidos por el nodo de destino y los paquetes enviados por el nodo de origen. Entonces, la reputación de un camino se obtiene de su fracción de pérdida de paquetes dada por Ec. 4.1

$$R_p = 1 - L_p \tag{4.1}$$

Un paquete llega a su destino con éxito si todos los nodos del camino lo reenviaron con éxito. Por lo tanto, suponiendo que el comportamiento de reenvío de cada nodo sea independiente de los otros, la reputación (R_p) del camino p se puede calcular como el producto de las reputaciones (r_n) de los nodos n que forman ese camino p, ver Ec. 4.2.

$$R_p = \prod_{n \in path\ p} (r_n) \tag{4.2}$$

Como se dijo antes, el algoritmo propuesto estima la reputación de los nodos ($\widehat{r_n}$) compartiendo las pérdidas de manera equitativa entre los nodos que forman el camino, excepto el nodo de origen que no se supone que genere pérdidas. En este caso, Ec. 4.2 puede reescribirse como sigue, siendo $\widehat{r_n}$ la reputación estimada del nodo n, y N_p el número de nodos intermedios en el camino p.

$$R_p = (\widehat{r_n})^{N_p} \tag{4.3}$$

Cada vez que el nodo fuente actualiza la fracción de pérdidas de paquetes para el camino p, se estima el valor de reputación de reenvío de cada nodo que pertenece a ese camino. Para hacer eso, el valor de reputación del camino calculado en Ec. 4.1 es utilizado en Ec. 4.3 para obtener Ec. 4.4.

$$\widehat{r_n} = (1 - L_p)^{\frac{1}{N_p}} \tag{4.4}$$

Como se mencionó anteriormente, el hecho de dividir las pérdidas equitativamente entre los nodos que componen el camino podría afectar negativamente la reputación de los nodos que tienen un comportamiento colaborativo. Además, esto también puede ocultar un comportamiento egoísta (es decir, el nodo obtiene una reputación más alta de la que merece). Por este motivo, la reputación de los nodos debe promediarse teniendo en cuenta las estimaciones realizadas en todos los caminos en los que el nodo ha participado anteriormente. En la siguiente sección, proponemos una metodología para promediar las muestras de reputación de los nodos.

El objetivo principal es que cada nodo tenga una tabla con las reputaciones estimadas de los otros nodos. De esta manera, los nodos pueden evaluar la reputación de los diferentes caminos descubiertos y elegir el más confiable. La reputación de los caminos se calcula utilizando las reputaciones estimadas de los nodos en la Ec. 4.2

Un inconveniente del algoritmo propuesto es que los nodos con una mala reputación pueden experimentar dificultades para actualizar su reputación en caso de cambiar su comportamiento y cooperar de aquí en adelante. Esto sucede porque esos nodos no han sido seleccionados como nodos de reenvío durante un tiempo (ya que los nodos fuente intentan evitarlos debido a su comportamiento egoísta), por lo que no pueden proporcionar información actualizada de su comportamiento actual. Para abordar este problema, hacemos que los nodos olviden periódicamente todo lo que han aprendido sobre el resto de los nodos, como un tipo de reinicio en el sistema.

A continuación, presentamos los algoritmos propuestos para seleccionar el mejor camino de reenvío. El Algoritmo 4.3 es el algoritmo principal en nuestra metodología. Este algoritmo verifica, entre el conjunto de caminos activos, si tiene nueva información

sobre las pérdidas de paquetes experimentadas en el camino. Si es positivo, actualiza la reputación de los nodos que pertenecen a ese camino utilizando el Algoritmo 1 (*UPDATENODESREPUTATION*). En el caso de que sea necesario encontrar un camino a un destino, se realiza un descubrimiento de camino DSR para encontrar los caminos disponibles a ese destino. A continuación, la reputación de cada camino descubierto se calcula utilizando el Algoritmo 4.2 (*GETPATHREPUTATION*), y se selecciona el camino con la calificación de reputación más alta. Además, el Algoritmo 1 utiliza la función *AVERAGE* (descrita en el Algoritmo 4.4) para estimar la reputación de los nodos antes mencionados que se explicará en la siguiente sección.

Require: *p, L* *//path ID* and a new *L* sample of that path
Ensure: Updates the reputation of the nodes
1: **procedure** UPDATENODESREPUTATION (*p, L*)
2: *NodeReputationSample*←(1-*L*)^(*1/Path*(*p*).*NumberOfIntemediateNodes*)
3: **for** *n* in *Path*(*p*).*SetOfIntermediateNodes* **do**
4: *Node*(*n*).*Reputation*←AVERAGE(*n*, *NodeReputationSample*)
5: **end for**
6: **end procedure**

Algoritmo 4.1: Actualiza la reputación estimada de los nodos

Require: *p //path ID*
Ensure: Computes a path reputation
1: **function** GETPATHREPUTATION (*p*)
2: *PathReputation*←1
3: **for** *n* in *Path*(*p*).*SetOfIntermediateNodes* **do**
4: *PathReputation*←*PathReputation* **Node*(*n*).*Reputation*
5: **end for**
6: **return** *PathReputation*
7: **end function**

Algoritmo 4.2: Estimación de la reputación del camino

Require: DSR route discovery
Ensure: Selection of the most reputable route

```
1: repeat forever
2:     for p in ActivePaths do
3:         if NewLosesInformation(p) then
4:             UPDATENODESREPUTATION (p, L)
5:         end if
6:     end for
7:     if NewRouteNeeded.Destination then
8:         Routes ← GETDSRROUTES (Destination)
9:         MaxPathReputation← 0
10:        for p in Routes do
11:            Path(p).Reputation← GETPATHREPUTATION (p)
12:            if Path(p).Reputation > MaxPathReputation then
13:                MaxPathReputation← Path(p).Reputation
14:                Destination.Path← Path(p)
15:            end if
16:        end for
17:    end if
18: end repeat
```

Algoritmo 4.3: Encaminamiento

4.3 Algoritmo para promediar los valores de reputación de reenvío

Suponemos que los nodos tienen un comportamiento de reenvío estable durante un tiempo indeterminado. Esto implica que el valor de reputación de reenvío promedio de los nodos se mantiene estacionario durante un intervalo arbitrario. Sin embargo, también consideramos la posibilidad de que el comportamiento de reenvío de los nodos cambie a lo largo del tiempo, comenzando un nuevo intervalo estacionario. Esto puede deberse al hecho de que el nodo cambia su comportamiento o su vecindario, lo que afecta su reputación.

Como se mencionó anteriormente, cada vez que un nodo obtiene la reputación de un camino, se actualiza una estimación de la reputación de los nodos que pertenecen a ese camino. Para mayor comodidad en la nomenclatura, nombramos el conjunto de k muestras consecutivas del comportamiento estimado para un nodo n, como $r_1, r_2, \dots, r_k$

en lugar de $\widehat{r_{n_1}}, \widehat{r_{n_2}}, \ldots, \widehat{r_{n_k}}$. En aras de la simplicidad y las explicaciones claras, ignoramos el subíndice *n* para referirse al nodo *n* bajo evaluación. Para asignar una reputación a ese nodo utilizamos una media móvil. Los promedios móviles que hemos considerado son la media móvil acumulada (CMA) y la media móvil exponencial (EWMA) [**85**].

4.3.1 Media móvil acumulada

Si utilizamos una CMA [**85**], calculamos el promedio utilizando todo el historial de reputación de un nodo.

$$c_k = \frac{1}{k} \cdot \sum_{i=1}^{k} r_i \tag{4.5}$$

En Ec. 4.5, c_k representa la CMA del nodo *n* después que la k-th muestra de reputación ha sido estimada. Dado que las muestras anteriores tienen el mismo peso que las actuales, este promedio no refleja el comportamiento actual actualizado del nodo en caso de que cambie. Si el comportamiento del nodo fuera siempre el mismo, este sería un promedio correcto a implementar.

Suponemos que el nodo tiene un comportamiento estacionario dentro de una ventana de tiempo. De lo contrario, no tendría sentido promediar las muestras de reputación. Bajo este supuesto, podemos recalcular la CMA cada vez que una nueva ventana estacionaria comienza. Ahora, el problema es obtener el tamaño de esa ventana; es decir, debemos detectar cuándo se produce un cambio en el comportamiento del nodo, de modo que el promedio solo incluya muestras que tengan el mismo comportamiento estacionario. Por medio de un mecanismo que nos permite detectar esos cambios de comportamiento, podríamos definir la ventana en la que promediaríamos las muestras de reputación. Cada vez que comienza una nueva ventana, nos olvidamos de las muestras antiguas; es decir, restablecemos el contador *i* en Ec. 4.5.

El conjunto de muestras de reputación r_k de un nodo n que tiene un comportamiento estable se puede expresar como un valor constante $\bar{r}$ a lo que una señal ruidosa Δr_k se superpone.

$$r_k = \bar{r} + \Delta r_k \tag{4.6}$$

La señal con ruido es un proceso estocástico de media cero ($E[\Delta r_k\,] = 0$). Por lo tanto, $\bar{r}$ es el valor esperado de las muestras $E[r_k]$, que permanece constante para todas las k muestras debido al comportamiento estacionario de los nodos. El segundo momento de Δr_k es la varianza de las muestras, que asumimos como constante para la ventana temporal de comportamiento estable.

$$\sigma_r^2 = E[(\Delta r_k)^2] \tag{4.7}$$

De la misma manera podemos expresar $c_k = \bar{c}_k + \Delta c_k$. Entonces, utilizando Ec. 4.6 en Ec. 4.5 obtenemos

$$\bar{c}_k = E[c_k] = \bar{r} \tag{4.8}$$

$$\Delta c_k = \frac{1}{k} \cdot \sum_{i=1}^{k} \Delta r_i \tag{4.9}$$

Desde que $\Delta r_{k,j}$ es un proceso de media cero, Δc_k es un proceso de media cero también. Para evaluar la desviación del valor esperado, calculamos la varianza de las muestras CMA.

$$\sigma_c^2(k) = E[(\Delta c_k)^2] = \frac{1}{k^2} \cdot E\left[\left(\sum_{i=1}^{k} \Delta r_i\right) \cdot \left(\sum_{j=1}^{k} \Delta r_j\right)\right] = \frac{1}{k^2} \cdot \sum_{i=1}^{k} \sum_{j=1}^{k} E\left[\Delta r_i \Delta r_j\right] \tag{4.10}$$

Cada acción de reenvío de un nodo es independiente de otra acción de reenvío de ese mismo nodo. Por lo tanto, tenemos que los procesos Δr_i y Δr_j son independientes para $i \neq j$.

$$\sigma_c^2(k) = \frac{1}{k^2} \sum_{i=1}^{k} \sum_{\substack{j=1 \\ j \neq i}}^{k} E[\Delta r_i] \cdot E\left[\Delta r_j\right] + \frac{1}{k^2} \sum_{i=1}^{k} E[(\Delta r_i)^2] \tag{4.11}$$

Porque también son procesos de media cero y hemos supuesto que la varianza de las muestras es constante para la ventana temporal de comportamiento estable, ver Ec. 4.7, obtenemos la Ec. 4.12:

El valor esperado de las muestras promediadas utilizando CMA no depende de k y permanece constante para la ventana de tiempo con un comportamiento estable. Ya que Δr_k es un proceso de media cero, Δc_k es un proceso de media cero también. Para

evaluar la desviación del valor esperado, calculamos la varianza de las muestras. Suponemos que el reenvío es un proceso sin memoria; es decir, un nodo que reenvía solo el 50% de los paquetes lanza una moneda antes de cada reenvío independientemente de los otros. Bajo este supuesto, tenemos que los procesos Δr_i y Δr_j son independientes para $i \neq j$ y la varianza es dada por Ec. 4.12.

$$\sigma_c^2(k) = \frac{1}{k^2} \cdot \sum_{i=1}^{k} E[(\Delta r_i)^2] = \frac{\sigma_r^2}{k} \tag{4.12}$$

Concluyendo, para la CMA bajo el supuesto de un comportamiento estable en una ventana temporal bien definida, el valor esperado de las muestras de reputación sigue siendo el mismo. Además, la varianza de las muestras de reputación se reduce inversamente proporcional al número de muestras.

4.3.2 Media móvil exponencial

Otra alternativa ampliamente utilizada para las muestras promedio es EWMA [**85**]. Mientras se promedia, se coloca más énfasis en las muestras recientes, lo que reduce exponencialmente el peso de las muestras anteriores. Denotamos w_k como el valor de la EWMA después de procesar la muestra r_k .

$$w_k = (1 - \beta) \cdot w_{k-1} + \beta \cdot r_k \tag{4.13}$$

En Ec. 4.13, β es un factor entre 0 y 1 que pesa la muestra actual r_k. Cuanto mayor sea el β, se da mayor importancia al presente. En general, EWMA tiene menos retraso que CMA y por lo tanto es más sensible a los cambios.

Para calcular la EWMA de un nodo, es necesario tomar un valor de reputación inicial. A priori consideramos que todos los nodos de la red son colaborativos. Por lo tanto, la reputación inicial de todos los nodos debe ser uno. Sin embargo, esta propuesta se define de manera genérica para cualquier otro valor inicial. Utilizando Ec. 4.13 recursivamente, obtenemos la media móvil en función de todas las muestras de reputación ponderadas exponencialmente, donde w_0 es el valor inicial de la media móvil:

$$w_k = (1 - \beta)^k \cdot w_0 + \beta \cdot \sum_{i=0}^{k-1} (1 - \beta)^i \cdot r_{k-i} \tag{4.14}$$

De la misma manera que hemos hecho con r_k en Ec. 4.6, podemos expresar $w_k = \bar{w}_k + \Delta w_k$, siendo $\bar{w}_k$ el comportamiento asintótico de la EWMA, y Δw_k un proceso estocástico que se superpone a él. Utilizando Ec. 4.6 en Ec. 4.14 relacionamos $\bar{w}_k$ con $\bar{r}$ y Δw_k con Δr_k.

$$\bar{w}_k = E[w_k] = (1-\beta)^k \cdot (w_0 - \bar{r}) + \bar{r} \tag{4.15}$$

$$\Delta w_k = \beta \cdot \sum_{i=0}^{k-1} (1-\beta)^i \cdot \Delta r_{k-i} \tag{4.16}$$

Ya que Δr_k es un proceso de media cero, Δw_k es un proceso de media cero también:

$$E[\Delta w_k] = \beta \cdot \sum_{i=0}^{k-1} (1-\beta)^i \cdot E[\Delta r_{k-i}] = 0. \tag{4.17}$$

Tenga en cuenta que de acuerdo a nuestra definición de r_k (ver Ec. 4.6), tenemos que $\Delta r_k = 0, \ \forall\, k < 1$.

Para evaluar la desviación de la EWMA de las muestras de reputación, del comportamiento asintótico, calculamos la varianza de la EWMA de las muestras de reputación.

$$\sigma_w^2(k) = E[(\Delta w_k)^2] = \beta^2 \cdot \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} (1-\beta)^i \cdot (1-\beta)^j \cdot E\left[\Delta r_{k-i} \Delta r_{k-j}\right] \tag{4.18}$$

Utilizando los mismos supuestos que en CMA, tenemos que los procesos Δr_{k-i} y Δr_{k-j} son independientes para $i \neq j$. Por lo tanto,

$$\sigma_w^2(k) = \beta^2 \sum_{i=0}^{k-1} \sum_{\substack{j=0 \\ j \neq i}}^{k-1} (1-\beta)^i \cdot (1-\beta)^j \cdot E[\Delta r_{k-i}] \cdot E\left[\Delta r_{k-j}\right]$$

$$+ \beta^2 \sum_{i=0}^{k-1} (1-\beta)^{2i} \cdot E[(\Delta r_{k-i})^2] \tag{4.19}$$

Porque también son procesos de media cero y hemos supuesto que la varianza de las muestras es constante para la ventana temporal de comportamiento estable (Ec. 4.7), tenemos que

$$\sigma_w^2(k) = \beta^2 \cdot \sum_{i=0}^{k-1} (1-\beta)^{2i} \cdot E[(\Delta r_{k-i})^2] = \beta^2 \cdot \sum_{i=0}^{k-1} (1-\beta)^{2i} \cdot \sigma_r^2 \tag{4.20}$$

Finalmente, alcanzamos la desviación estándar de la EWMA en función de la desviación estándar de las muestras de reputación.

Podemos ver que después de k muestras, La diferencia entre el valor esperado de la muestra y el comportamiento asintótico de la media móvil, esto es $\overline{w}_k - \bar{r}$, se ha reducido por un factor $(1-\beta)^k$. Ya que $(1-\beta)^k < 1$, cuando el número de muestras k crece, esta diferencia tiende a cero y $\overline{w}_k$ aproxima asintóticamente a $\bar{r}$. Por lo tanto, cuanto mayor sea el factor β, más rápida será la aproximación asintótica al valor deseado $\overline{r}$.

Ya que Δr_k es un proceso de media cero, Δw_k es un proceso de media cero también. Tenga en cuenta que de acuerdo con nuestra definición de r_k (ver Ec. 4.6), tenemos que $\Delta r_k = 0,\ \forall\ k < 1$.

Para evaluar la desviación de la EWMA de las muestras de reputación, del comportamiento asintótico, calculamos la varianza de la EWMA de las muestras de reputación. Utilizando los mismos supuestos que en CMA alcanzamos la varianza de la EWMA como una función de la varianza de las muestras de reputación (ver [**85**] para mas detalles):

$$\sigma_w^2(k) = \frac{\beta}{2-\beta} \cdot [1-(1-\beta)^{2k}] \cdot \sigma_r^2 \qquad (4.21)$$

Nótese que aunque suponemos que σ_r^2 es constante en la ventana estacionaria, σ_w^2 depende de k. Desde siempre $\beta < 1$, tenemos que $\sigma_w^2(k) < \sigma_r^2$. Esto significa que la EWMA reduce la desviación de las muestras de su valor medio. Cuanto menor es el factor β, menor es la varianza $\sigma_w^2(k)$. Si comparamos las varianzas de EWMA y CMA, podemos notar que para CMA esta varianza tiende a cero cuando aumenta el número de muestras (Ec. 4.12). Por el contrario, para EWMA cuando el número de muestras crece, la varianza tiende a un valor que depende de β y que se muestra en Ec. 4.22

$$\sigma_w^2 \approx \frac{\beta}{2-\beta} \cdot \sigma_r^2 \qquad (4.22)$$

Además, para EWMA, existe un compromiso entre la velocidad de aproximación asintótica y la varianza. Altos valores para β conlleva un acercamiento rápido al valor estacionario promedio, pero con una reducción de varianza bajo. Por el contrario,

valores bajos para β reducen considerablemente la varianza, aunque con un enfoque lento.

4.3.3 Detección de ventanas de tiempo estacionaria

CMA es la mejor opción para promediar las muestras de reputación, en caso de que el comportamiento de los nodos cambie bruscamente y podamos detectar los momentos en que comienzan y terminan las ventanas de tiempo estacionarias. Para ello, proponemos un método para detectar cambios en los comportamientos de los nodos y delimitar estas ventanas de tiempo estacionarias. Nuestra propuesta se basa en el uso combinado de una EWMA para detectar el cambio en el comportamiento de los nodos y una CMA para promediar sus valores de reputación.

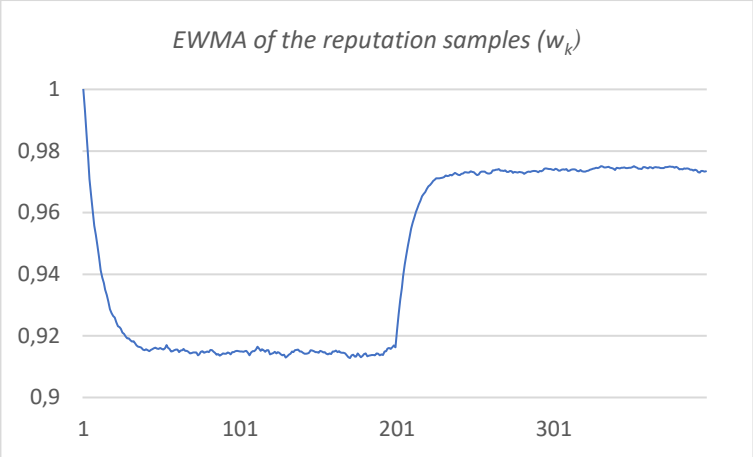


Figura 4.1: Promedio de las muestras de reputación de un nodo si utilizamos una EWMA cuando el nodo cambia su comportamiento definiendo dos ventanas estacionarias.

La Figura 4.1 muestra la EWMA para la reputación de un nodo que cambia, definiendo dos ventanas de tiempo estacionarias. Si el comportamiento asintótico de la EWMA fuera una función continua, tendríamos una discontinuidad en la derivada cada vez que cambia el comportamiento estacionario del nodo y habría una función delta de Dirac en la segunda derivada. Como tenemos una función de tiempo discreto, utilizamos la segunda diferencia de la EWMA en lugar de la segunda derivada para detectar el final de una ventana de tiempo estacionaria. Definimos la primera diferencia de la EWMA como:

$$f_k = w_k - w_{k-1} = \beta \cdot (r_k - w_{k-1}) \tag{4.23}$$

Y la segunda diferencia:

$$d_k = f_k - f_{k-1} = \beta \cdot (r_k - r_{k-1}) - \beta^2 \cdot (r_{k-1} - w_{k-2}) \tag{4.24}$$

Con esta definición, si un pico para d_k ocurre en k_0, significa que la muestra r_{k_0} pertenece a una nueva ventana de tiempo estacionaria. La expresión anterior no tiene sentido para $k < 3$, ya que al menos tres muestras son necesarias para calcular la segunda diferencia.

Con el fin de encontrar los valores de β que optimicen la detección de picos, relacionamos la segunda diferencia d_k con las muestras r_k. Utilizando la Ec. 4.14 en la Ec. 4.24, tenemos:

$$d_k = \beta \cdot r_k - \beta \cdot (1+\beta) \cdot r_{k-1} + \left(\frac{\beta}{1-\beta}\right)^2 \cdot \left[\beta \cdot \sum_{i=2}^{k-1} (1-\beta)^i \cdot r_{k-i} + (1-\beta)^k \cdot w_0\right] \tag{4.25}$$

Bajo el supuesto de que en la ventana estacionaria $r_k = \bar{r} + \Delta r_k$, podemos escribir $d_k = \overline{d_k} + \Delta d_k$, siendo

$$\overline{d_k} = \beta^2 \cdot (1-\beta)^{k-2} \cdot (w_0 - \bar{r}) \tag{4.26}$$

$$\Delta d_k = \beta \cdot \Delta r_k - \beta \cdot (1+\beta) \cdot \Delta r_{k-1} + \frac{\beta^3}{(1-\beta)^2} \cdot \left[\sum_{i=2}^{k-1} (1-\beta)^i \cdot \Delta r_{k-i}\right] \tag{4.27}$$

Para relacionar el ruido de la segunda diferencia de la EWMA con el ruido de las muestras, calculamos la desviación estándar de d_k. Como hemos dicho anteriormente, asumimos que $E(\Delta r_i \cdot \Delta r_j) = 0, \forall i \neq j$ y $E(\Delta r_i^2) = \sigma_r^2, \forall i > 0$ en la ventana estacionaria. En este caso,

$$\sigma_d^2 = E(\Delta d_k^2) = \beta^2 \cdot E(\Delta r_k^2) + \beta^2 \cdot (1+\beta)^2 \cdot E(\Delta r_{k-1}^2) + \frac{\beta^6}{(1-\beta)^4} \cdot \left[\sum_{i=2}^{k-1} (1-\beta)^{2i} \cdot E(\Delta r_{k-i}^2)\right]$$

$$\sigma_d^2 = \beta^2 \cdot \left[1 + (1+\beta)^2 + \beta^3 \frac{1-(1-\beta)^{2(k-2)}}{2-\beta}\right] \cdot \sigma_r^2 \qquad (4.28)$$

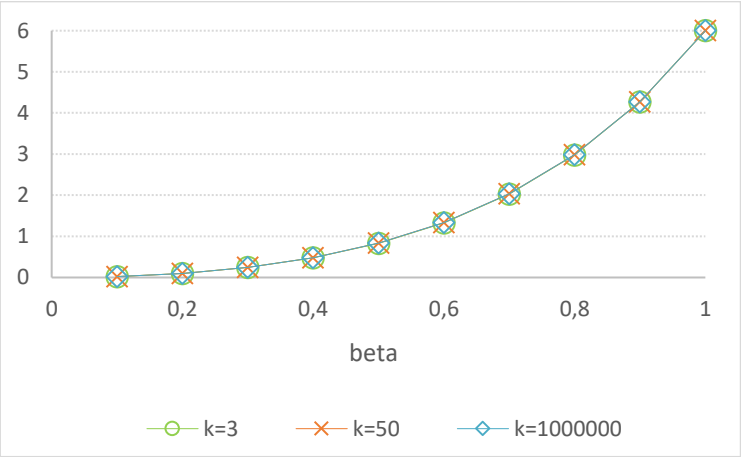


Figura 4.2: Gráfica del factor que relaciona la varianza de las muestras $\boldsymbol{\sigma_r^2}$ y la varianza de la segunda diferencia de la media móvil exponencial de las muestras $\boldsymbol{\sigma_d^2}$, como una función de $\boldsymbol{\beta}$ para diferente número de muestras $\boldsymbol{k}$ (ver Ec. 4.28).

Graficando el factor que relaciona ambas varianzas en función de β, utilizando un número diferente de muestras (ver la Figura 4.2), conseguimos prácticamente la misma gráfica siendo imposible distinguirlas, entonces podemos aproximar la expresión anterior para el caso $k \to \infty$ (en este caso aproximamos $(1-\beta)^{2(k-2)} \approx 0$, ya que $\beta < 1$).

$$\sigma_d^2 \approx 2\beta^2 \cdot \frac{2+\beta}{2-\beta} \cdot \sigma_r^2 \qquad (4.29)$$

Para valores grandes de β el ruido σ_d^2 incrementa, siendo mayor que el de las muestras σ_r^2. Por lo tanto tenemos que elegir un valor pequeño para β si queremos reducir el ruido y optimizar la detección.

Por otra parte, el pico de la segunda diferencia será mayor cuanto mayor sea el valor de β, siendo más fácil de detectar. Debemos comparar el valor del pico con el ruido y encontrar el valor de β que optimiza la relación señal-a-ruido pico. Para calcular el valor pico, asumimos que la reputación del nodo cambia de un valor $\bar{r}_1$ a un valor $\bar{r}_2$, y este cambio ocurre entre la muestra $k_0 - 1$ y k_0 así que $r_{k_0-1} = \bar{r}_1 + \Delta r_{k_0-1}$ y $r_{k_0} = \bar{r}_2 + \Delta r_{k_0}$. Utilizando Ec. 4.19 obtenemos:

$$\overline{d_{k_0}} = \beta \cdot (\overline{r_2} - \overline{r_1}) + \beta^2 \cdot (1-\beta)^{k_0-2} \cdot (w_0 - \overline{r_1}) \quad (4.30)$$

Comparando este resultado con Ec. 4.26 observamos que el pico que aparece en la segunda diferencia causada por el cambio de comportamiento es:

$$P = \beta \cdot (\overline{r_2} - \overline{r_1}) \quad (4.31)$$

Utilizando las Ecs. 4.29 y 4.31 podemos definir la relación señal-a-ruido pico en función de β:

$$\frac{P^2}{\sigma_d^2} = \frac{2-\beta}{2 \cdot (2+\beta)} \cdot \frac{(\overline{r_2} - \overline{r_1})^2}{\sigma_r^2} \quad (4.32)$$

Ya que $(2-\beta)/2(2+\beta)$ es una función decreciente de β, podemos concluir que cuanto menor sea el valor β, mejor es la detección de una nueva ventana de tiempo estacionaria. Obviamente, no podemos utilizar $\beta = 0$ porque en este caso no estábamos promediando muestras.

4.3.4 Algoritmo de promedio propuesto

Nuestra propuesta es promediar las muestras de reputación de un nodo utilizando una CMA en un período estacionario de comportamiento estable para cada nodo. Utilizamos este promedio como la reputación de los nodos para encontrar los mejores caminos de reenvío en la red móvil ad-hoc. Para delimitar este período estacionario, en paralelo calculamos la EWMA de las muestras utilizando un valor bajo para β, cuyo valor se discute en la sección de resultados de simulación. Además, calculamos la segunda diferencia de ese promedio EWMA obteniendo los valores d_k. Suponemos que las muestras d_k representan un proceso ergódico y por lo tanto su valor medio $\overline{d_k}$ y su desviación estándar σ_{d_k} se calculan de la siguiente manera:

$$\overline{d_k} = \frac{1}{k} \cdot \sum_{i=1}^{k} d_i \quad \text{y} \quad \sigma_{d_k} = \sqrt{\overline{d_k^2} - \overline{d_k}^2} \quad \text{siendo} \quad \overline{d_k^2} = \frac{1}{k} \cdot \sum_{i=1}^{k} d_i^2 \quad (4.33)$$

En Ec. 4.33, k representa el número de muestras que tenemos en la ventana de tiempo estacionario. Cuantas más muestras, más preciso será el resultado que logremos.

Cuando el valor absoluto de una nueva muestra d_{k+1} es mayor que un cierto número de desviaciones estándar agregadas al valor medio, ocurre un evento de detección de pico. En este caso, comenzamos una nueva CMA con las nuevas muestras que pertenecen a

una nueva ventana de tiempo estacionaria. La condición de detección de pico se muestra en Ec. 4.28.

$$|d_{k+1}| > |\overline{d_k}| + F \cdot \sigma_{d_k} \tag{4.34}$$

Como no sabemos si la reputación del nodo aumentará o disminuirá, no sabemos si el pico de la segunda diferencia será positivo o negativo. Por lo tanto, utilizamos el valor absoluto de d_{k+1} y $\overline{d_k}$. El factor F determina cuanto más alto debe ser el pico con respecto al ruido. Un pequeño valor F puede ser la causa de falsos positivos, por lo que el ruido en las muestras puede parecer un pico. Por otro lado, un gran valor F puede ser la razón por la que no se detectan pequeños cambios en el comportamiento de un nodo. Como se vio en Ec. 4.31, la amplitud del pico depende de la diferencia de ambas reputaciones, la antigua y la nueva.

En resumen, el algoritmo propuesto utiliza una CMA para promediar las muestras de reputación y la segunda diferencia de la EWMA con un pequeño β para detectar los límites de las ventanas de tiempo estacionario.

Require: *n, ReputationSample* // *Node ID* and a reputation sample for that node

Ensure: Average the reputation of the nodes

1: **function** AVERAGE(*n, ReputationSample*)

2: *Node*(*n*).*EWMA_Update* // update w_k using Ec. 4.13 and store the two last EWMA values

3: *Node*(*n*).*SecondDifference_Update* // update d_k using Ec. 4.24; $d_k = w_k - 2 \cdot w_{k-1}^2 + w_{k-2}$

4: **if** *PeakCondition* **then** //PeakCondition in Ec. 4.34

5: *Node*(*n*).*k* ← *1* // start a new stationary time window (only one sample)

6: *Node*(*n*).*CMA* ← *ReputationSample* //Reset c_k

7: *Node*(*n*).*SecondDifferenceMean* ← *Node*(*n*).*SecondDifference* //Reset $\overline{d_k}$

8: *Node*(*n*).*SecondDifferenceStd* ← *0* //Reset σ_d

9: **else**

10: $Node(n).k$ ++// increment the number of samples in the steady time window

11: $Node(n).CMA\_Update$ //update c_k using Ec. 4.5

12: $Node(n).SecondDifferenceMean\_Update$ // update $\overline{d_k}$ using Ec. 4.33

13: $Node(n).SecondDifferenceStd\_Update$ // update σ_d using Ec. 4.33

14: **end if**

15: **return** $Node(n).CMA$

16:**end function**

Algoritmo 4.4: Promedio

4.4Resultados Experimentales y Discusión

En esta sección primero validamos nuestra propuesta en la Sección 4.4.1, y luego realizamos una evaluación de desempeño en la sección 4.4.2

4.4.1 Validación del modelo teórico

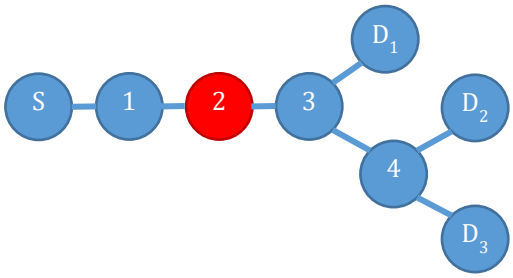


Figura 4.3: Topología de red utilizada en la simulación con un nodo fuente (S) y tres destinos (D_1, D_2 y D_3). En la simulación, todos los nodos reenvían el 100% de los paquetes, excepto el nodo 2 que reenvía solo un 70% de los paquetes durante la primera mitad de la simulación. Posteriormente, el nodo cambia su comportamiento y reenvía un 90% de los paquetes durante la última mitad de la simulación.

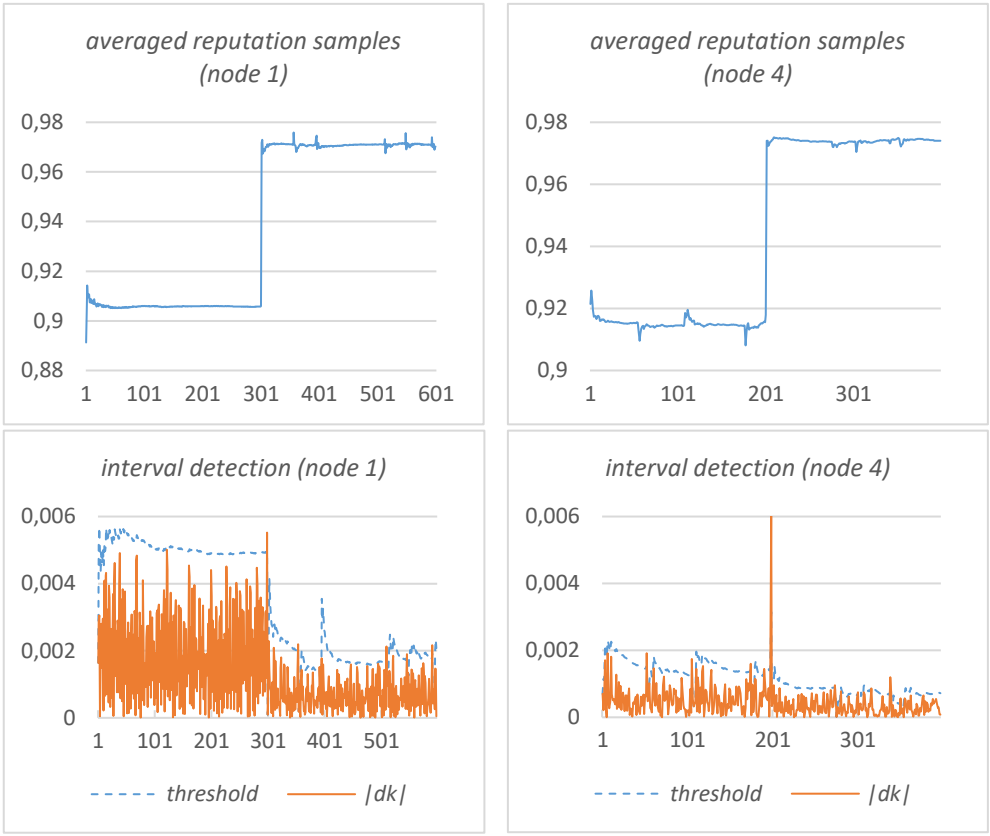
Para validar los resultados teóricos en la detección de cambios de comportamiento en los nodos de la red móvil ad-hoc y poder definir las ventanas temporales en las que los nodos tienen un comportamiento estacionario, varias simulaciones con Matlab R2017b [**101**] se han realizado con la topología mostrada en la Figura 4.3. Representa a un nodo de origen (S) que transmite a tres destinos diferentes (D_i, i=1, 2, 3). Cada 1000 paquetes

transmitidos a través de un camino, se calcula la probabilidad de pérdida para ese camino y se obtiene una muestra de reputación para cada uno de los nodos intermedios que forman ese camino. En total, 100,000 paquetes se han transmitido a cada destino. Todos los nodos reenvían todos los paquetes que recibieron, excepto el nodo 2, que no reenvía un 30% de los paquetes durante la primera mitad de la simulación y un 10% durante la segunda mitad. Es decir, su reputación real va de 0.7 a 0.9. El nodo 4 solo participa en los caminos S-D_2 y S-D_3 mientras los nodos 1, 2 y 3 participan en todos los caminos. Por lo tanto, el nodo S debe asignar la misma reputación a los nodos 1, 2 y 3.

Inicialmente, verificamos si el cambio de comportamiento del nodo 2 se detecta claramente y cuál es la reputación que el nodo S asigna a los nodos 1 y 4 para diferentes valores de β. El umbral de detección se ha establecido con $F = 2$, ya que este valor mostró una muy buena tasa de detección después de realizar muchas simulaciones. De esta manera se produce una detección si $|d_{k+1}| > |\overline{d_k}| + 2 \cdot \sigma_{d_k}$. En las figuras 4.4 a 4.10, el eje horizontal indica el número de muestras. Cada vez un pico de $|d_k|$ se coloca por encima del umbral, el promedio de las muestras de reputación se restablece.

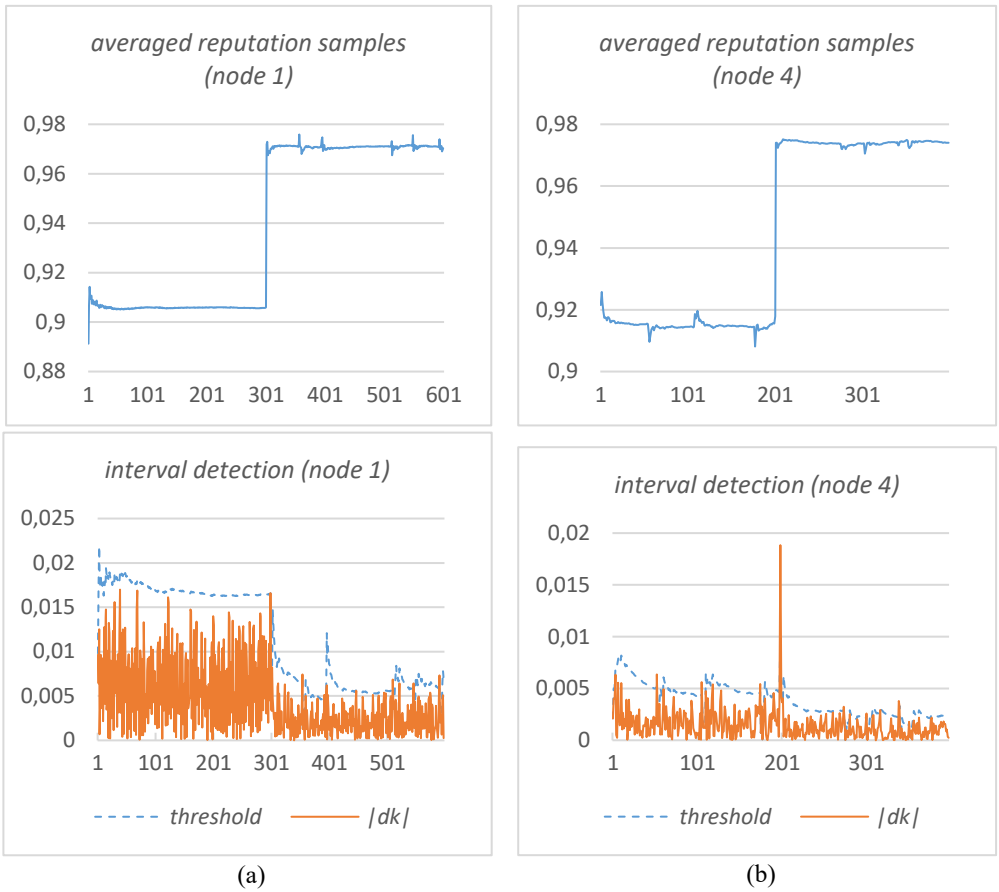
(a) (b)

Figura 4.4: Valor promedio de las muestras (figuras en la parte superior) y detección de intervalos (figuras siguientes) cuando $\beta = 0.01$ y $F = 2$: (a) para el nodo 1 y (b) para el nodo 4.



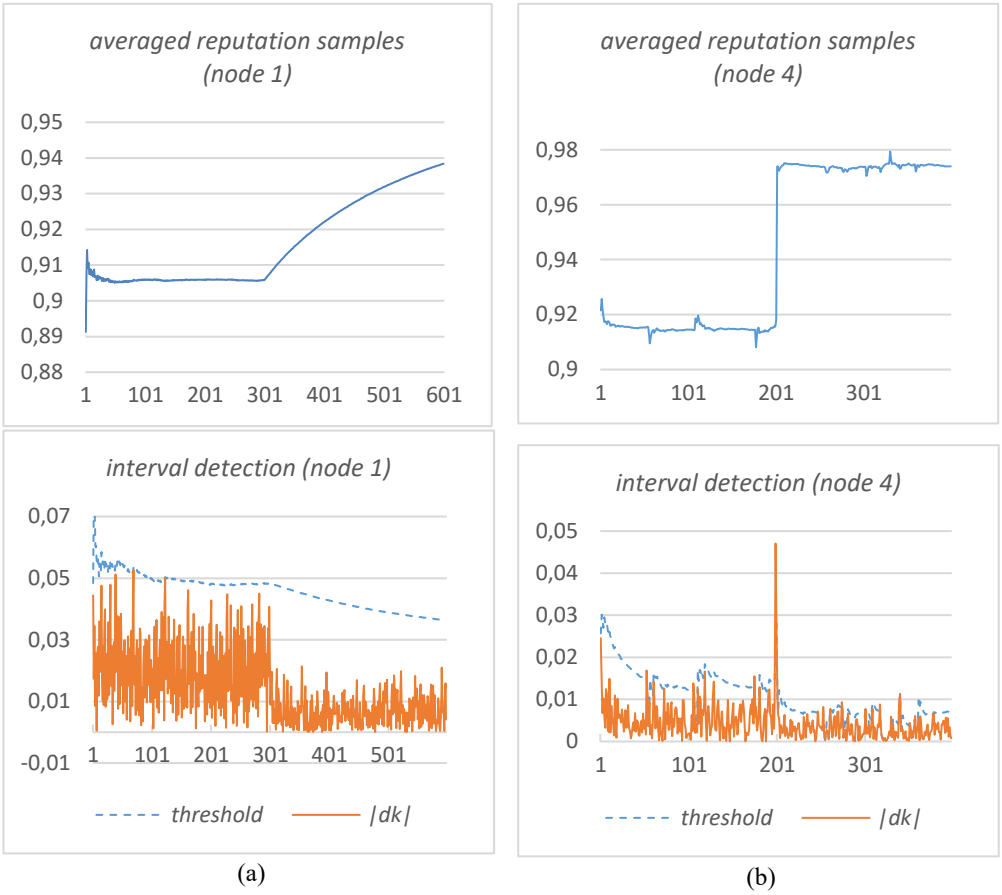
(a) (b)

Figura 4.5: Valor promedio de las muestras (figuras en la parte superior) y detección de intervalos (figuras siguientes) cuando $\beta = 0.1$ y $F = 2$: (a) para nodo 1 y (b) para nodo 4



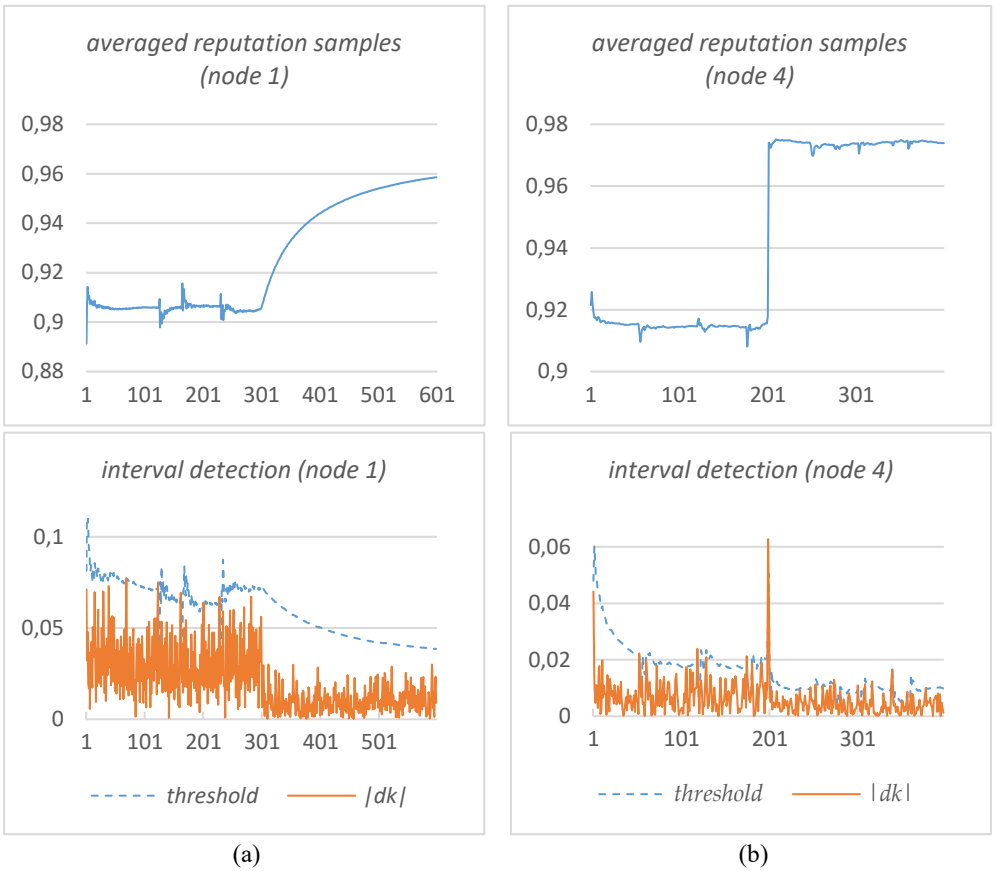
(a) (b)

Figura 4.6: Valor promedio de las muestras (figuras en la parte superior) y detección de intervalos (figuras siguientes) cuando $\beta = 0.3$ y $F = 2$: (a) para el nodo 1 y (b) para el nodo 4



(a) (b)

Figura 4.7: Valor promedio de las muestras (figuras en la parte superior) y detección de intervalos (figuras siguientes) cuando $\beta = 0.7$ y $F = 2$: (a) para el nodo 1 y (b) para el nodo 4

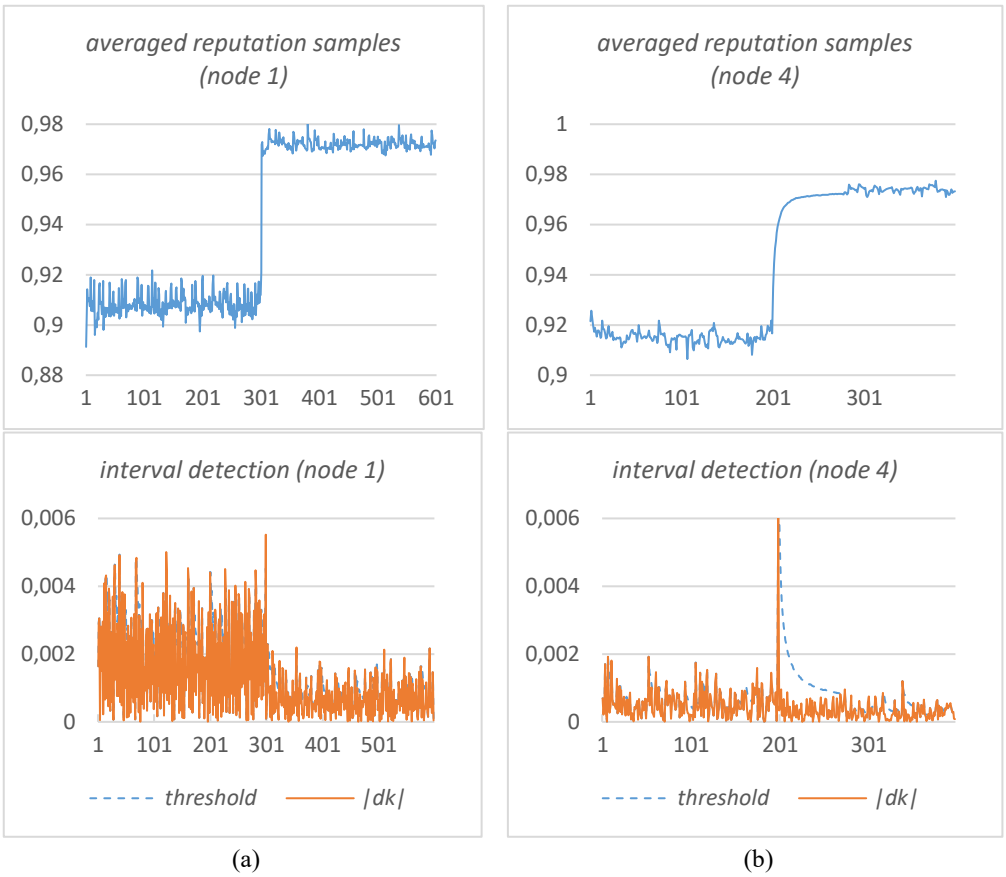


(a) (b)

Figura 4.8: Valor promedio de las muestras (figuras en la parte superior) y detección de intervalos (figuras siguientes) cuando $\beta = 0.9$ y $F = 2$: (a) para nodo 1 y (b) para nodo 4

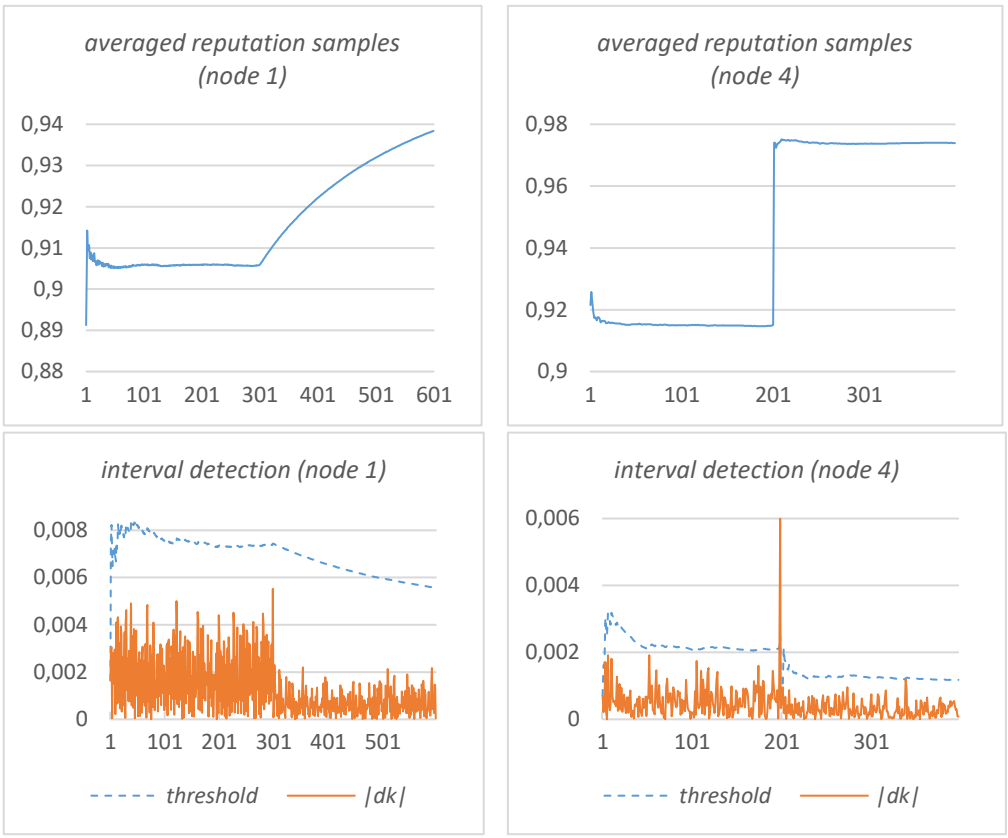
Como el nodo 1 participa en caminos diferentes, sus muestras son más ruidosas que las del nodo 4. El ruido dificulta la detección del cambio en el comportamiento del nodo 2 si el umbral de detección es demasiado alto. En la Figura 4.4 se observa que para un valor de $\beta = 0.01$ la detección se puede hacer perfectamente, aunque se obtienen falsos positivos. Un falso positivo ocurre cuando $|d_k|$ supera el umbral pero en realidad no hubo un cambio en el comportamiento del nodo. Uno de los falsos positivos se ha marcado con un círculo rojo en la Figura 4.4 en la muestra número 125. Como puede verse, un falso positivo produce una oscilación en el valor promedio de la reputación debido al hecho de que la CMA se reinicia (vea la gráfica en la parte superior izquierda en la Figura 4.4). Dado que la varianza de la CMA disminuye inversamente proporcional al número de muestras (ver Ec. 4.12), cuando se reinicia la CMA, su varianza es la misma que la varianza de las muestras de reputación, y disminuye a medida que promediamos más muestras de reputación. En la Figura 4.5 obtenemos

resultados similares utilizando un valor de $\beta = 0.1$. Incrementando el valor de β a 0.3 (ver Figura 4.6) estamos en el límite de detección. Sin embargo, para $\beta = 0.7$ (Figura 4.7) y $\beta = 0.9$ (Figura 4.8) la relación pico-a-ruido es más baja, de modo que no se produce ningún cambio en el comportamiento del nodo 1 y el promedio es más lento. Como ya se mencionó en la sección anterior, valor alto de β empeora la relación pico-a-ruido, lo que dificulta la detección de un cambio en el comportamiento del nodo.



(a) (b)

Figura 4.9: Valor promedio de las muestras (figuras en la parte superior) y detección de intervalos (figuras siguientes) cuando $\beta = 0.1$ y $F = 1$: (a) para el nodo 1 y (b) para el nodo 4



(a) (b)

Figura 4.10: Valor promedio de las muestras (figuras en la parte superior) y detección de intervalos (figuras siguientes) cuando $\beta = 0.1$ y $F = 3$: (a) para el nodo 1 y (b) para el nodo 4

El valor del umbral F debe establecerse experimentalmente. Problemas similares se pueden encontrar en la literatura (por ejemplo, gráficos de control [20]) y los investigadores ajustan este parámetro basado en experimentos. Para hacerlo, varias simulaciones se han realizado con $\beta = 0.1$, cambiando el valor del umbral de detección F en cada simulación: $F = 1$ en la Figura 4.9, $F = 2$ en la Figura 4.5 y $F = 3$ en la Figura 4.10. Podemos ver que umbrales bajos de F produce falsos positivos que generan pequeñas oscilaciones en el promedio de la reputación de los nodos (ver los gráficos superiores en la Figura 4.9). A pesar de esto, es preferible detectar falsos positivos que no poder detectar el cambio de comportamiento de los nodos. En caso de que siempre ocurran falsos positivos, las muestras de reputación no se promedian, ya que la CMA se reinicia constantemente, pero los cambios de comportamiento siempre se detectan. Por lo tanto, es más conveniente utilizar valor bajo F. Por otro lado, en la Figura 4.10,

podemos ver que valores más altos para F no permiten detectar el cambio del comportamiento del nodo con muestras de reputación más ruidosas, y su reputación no se actualiza correctamente. Como se mencionó anteriormente, como el nodo 4 solo participa en un camino, sus muestras de reputación son menos ruidosas que las de los otros nodos, siendo posible en este caso utilizar umbrales de F más altos para evitar el riesgo de cometer falsos positivos (lo que ocurre para valores bajos F). Después de muchas simulaciones representativas que hemos realizado, hemos obtenido un valor adecuado para el umbral $F = 2$.

4.4.2 Evaluación del desempeño del protocolo de encaminamiento propuesto

Para verificar el funcionamiento del algoritmo de encaminamiento propuesto, se ha desarrollado un simulador de una red móvil ad-hoc utilizando Matlab R2017b [**101**]. El simulador implementa DSR y también nuestro algoritmo de reenvío DrepSR propuesto. De esta manera, podemos comparar el rendimiento de DSR en ambos casos, ya sea utilizando el número mínimo de saltos (DSR básico) o, a la inversa, utilizando nuestro algoritmo para seleccionar el camino con la mejor reputación (DrepSR). Para realizar las simulaciones, se han generado 10 escenarios independientes de 200m x 200m con 16 nodos siguiendo un patrón de movimiento de punto de ruta aleatorio. La mitad de los nodos se han configurado para que se comporten de forma egoísta al dejar caer los paquetes que deben reenviar. Aquí mostramos resultados con nodos egoístas que solo reenvían el 50% de los paquetes que deberían reenviar. Los ajustes de simulación se muestran en la Tabla 4.1.

Parameter	Value
Simulation time	1000 s
Number of nodes	16
Movement pattern	Random waypoint
Area	200m x 200m
Minimum speed	0.5 m / s
Maximum speed	1.5 m / s
Maximum pause time	60 s
Transmission range	120 m
Traffic type	CBR at 20000 bps
β	0.1
F	2

Number of selfish nodes (50% losses)	8
Source nodes (destinations)	1 (2, 3, 4, 5, 6, 11)
	2 (3, 7, 12)
	3 (4, 8,12)
	4 (5, 9, 14)
	5 (1, 10, 15)

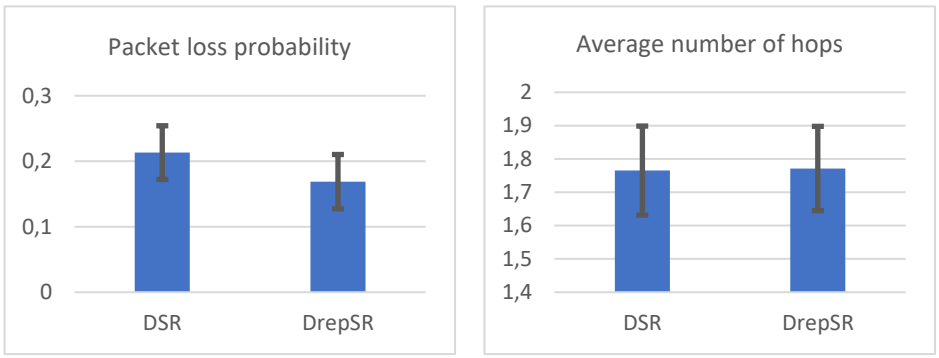
Tabla 4.1: Ajustes de simulación

Para cada simulación, hemos obtenido la probabilidad de pérdida de paquetes en la red móvil ad-hoc y el número promedio de saltos, tanto para los protocolos de encaminamiento DSR como DrepSR. Estos resultados se muestran en Tabla 4.2.

	DSR		DrepSR	
Scenario	**Packet loss probability**	**Mean number of hops**	**Packet loss probability**	**Mean number of hops**
1	0,1729	1,6431	0,13	1,6632
2	0,2514	1,9605	0,2224	1,9624
3	0,2034	1,7846	0,1578	1,8177
4	0,2562	1,8872	0,2023	1,904
5	0,186	1,7013	0,1521	1,6515
6	0,2137	1,6815	0,1642	1,6936
7	0,1424	1,5611	0,1158	1,6012
8	0,1693	1,6354	0,1241	1,6498
9	0,3696	2,248	0,3243	2,2141
10	0,1671	1,5476	0,0952	1,5557

Tabla 4.2: Resultados de la simulación.

Al promediar las 10 simulaciones, obtenemos que el DrepSR propuesto reduce las pérdidas en un 20% (DSR muestra una probabilidad de pérdida de paquetes del 21% y DrepSR del 16%) con respecto a DSR. Además, podemos ver que el número de saltos apenas aumenta y puede considerarse insignificante. El promedio de los resultados y el intervalo de confianza del 95% se muestran en la Figura 4.11.



(a) (b)

Figura 4.11. Comparación de los resultados simulados para el protocolo DSR y el protocolo de encaminamiento DrepSR propuesto. (a) Probabilidad de pérdida de paquetes; (b) Número promedio de saltos. Las figuras muestran un intervalo de confianza del 95% utilizando 10 realizaciones independientes por punto.

Opcionalmente, los investigadores pueden utilizar conjuntos de datos de código abierto para redes móviles ad-hoc para llevar a cabo la evaluación del desempeño de las propuestas, como la que se presenta en [**102**]. Este tipo de trazas generadas a partir de experimentos reales pueden ser muy útiles para evaluar protocolos y algoritmos con precisión.

4.5 Conclusiones

El objetivo del capítulo fue proponer un algoritmo de protocolo de encaminamiento basado en la reputación de peso dinámico de los nodos de la red móvil ad-hoc, evaluar, analizar y comparar las mejoras propuestas; y, luego de desarrollado el capítulo las conclusiones son las siguientes:

- En este documento presentamos una nueva estrategia de reenvío para protocolos de encaminamiento ad-hoc denominada DrepSR. DrepSR es una modificación del DSR básico que incluye nuestro nuevo algoritmo de reenvío basado en reputación. Nuestro enfoque se basa en la reputación de los nodos de la red móvil ad-hoc, donde consideramos la presencia de nodos egoístas que no cooperan en las tareas de encaminamiento. Cada nodo fuente utiliza el conocido protocolo DSR como motor de encaminamiento para descubrir los caminos disponibles hacia el destino. Luego, el nodo fuente calcula la reputación de cada

camino seleccionando el más confiable utilizando sólo información de primera mano.

- La ventaja de esta metodología es su simplicidad, y aunque no puede calcular la reputación real de los nodos individuales, puede encontrar esos caminos con menos pérdidas, que era nuestro objetivo inicial. Nuestro objetivo principal, es minimizar la fracción de pérdida de paquetes al enviar paquetes a través de diferentes caminos. Por lo tanto, es natural utilizar la fracción de pérdida de paquetes observada de cada nodo como una métrica de su reputación. Por supuesto, las pérdidas de paquetes no siempre pueden ser causadas por el comportamiento egoísta de los nodos, sino también por la congestión del tráfico o la naturaleza aleatoria de las comunicaciones inalámbricas, o incluso por las capacidades limitadas de los nodos. Sin embargo, la causa real de una pérdida de paquetes no es capturada por nuestro enfoque y suponemos que siempre está relacionado con un comportamiento egoísta, que afectan directamente a la reputación de los nodos implicados. Nos gustaría resaltar que aunque nuestra métrica de reputación no es una medida precisa del comportamiento egoísta para cualquier nodo individual, es una buena estimación que ayuda a elegir el mejor camino de encaminamiento disponible.
- Hemos validado nuestros resultados teóricos a través de simulaciones exhaustivas utilizando Matlab R2017b [**101**]. Después de la validación, la nueva estrategia fue implementada sobre DSR utilizando Matlab R2017b y muchas simulaciones se llevaron a cabo para diferentes escenarios con nodos móviles. Comparamos la probabilidad de pérdida de paquetes y el número promedio de saltos para el protocolo DSR (encaminamiento utilizando el camino más corto) y el protocolo DrepSR (encaminamiento utilizando el camino más confiable). Los resultados muestran que DrepSR reduce la probabilidad de pérdida de paquetes en un 20%, mientras que aumenta ligeramente el número promedio de saltos en un 8%.

Capítulo 5

Conclusiones y trabajo futuro

"Todas las ciencias matemáticas se basan en las relaciones entre las leyes físicas y las leyes de los números, por lo que el objetivo de la ciencia exacta es reducir los problemas de la naturaleza a la determinación de las cantidades mediante operaciones con números"

—James Clerk Maxwell. Científico escocés en el campo de la física matemática. 1831-1879

"Una obra de arte nunca se termina, sólo se abandona"

—Leonardo da Vinci. Erudito del Renacimiento Italiano. 1452-1519

5 Conclusiones y trabajo futuro

En esta sección se presenta las conclusiones de toda la tesis, el link de descarga del código fuente desarrollado para NS-2.35, los artículos publicados; y, las líneas de trabajo futuro.

5.1 Conclusiones

Esta tesis doctoral, se constituye en una contribución al diseño de protocolos de encaminamiento en redes móviles ad-hoc basados en reputación, por cuanto ha dado lugar a dos nuevos algoritmos de protocolos de encaminamiento para redes móviles ad-hoc llamados, REMODE_sw (Reputation for Mobile Devices Static Weight) y DrepSR(Dynamic Reputation Source Routing). Luego de finalizada la investigación, llegamos a las siguientes conclusiones:

- En esta tesis estudiamos el encaminamiento de las redes móviles ad-hoc, y luego nos centramos en diseñar e implementar dos nuevos algoritmos de protocolos de encaminamiento de capa cruzada sobre DSR, para dar lugar a REMODE_sw y DrepSR los cuales permiten encontrar los caminos que incluyen a los nodos más cooperativos de una red móvil ad-hoc.
- En esta tesis aplicamos el concepto de reputación que suele ser utilizado para personas, ya que cada nodo de la red estima la reputación de reenvío del resto de nodos; y, experimentalmente verificamos que la reputación de reenvío de un nodo no sólo depende de su comportamiento con los reenvíos, sino que además está afectada por el comportamiento de sus vecinos.
- Las dos propuestas de algoritmos de protocolos de encaminamiento utilizan información de retroalimentación desde capa de transporte, la cual le permite al nodo fuente conocer la fracción de paquetes perdidos de determinado camino. Dicha fracción de paquetes perdidos es la variable principal con la que cuenta el nodo fuente, para obtener la reputación de los nodos intermedios.
- Luego de que hemos analizado el código fuente de la memoria caché de DSR en NS-2.35, y luego de que hemos realizado múltiples simulaciones en distintos escenarios, determinamos que efectivamente, DSR mantiene en cada nodo fuente una tabla de encaminamiento con múltiples caminos, de esta manera en

los capítulos 3 y 4, es posible seleccionar los caminos cuyos nodos tienen la reputación más alta.

- En esta investigación hemos presentado una expresión analítica para calcular la estimación de reputación que realizará cada nodo. Dicha estimación depende de la topología de la red, del comportamiento de los nodos, y de los tráficos cursados por los distintos caminos.
- En las implementaciones, hemos modificado el comportamiento de algunos de los nodos de la red (generando pérdidas aleatorias) para que no reenvíen todos los paquetes que reciben.
- Para REMODE_sw, en el entorno de simulación desarrollado sobre NS-2.35, hemos analizado la influencia de los parámetros α y β en las medias móviles exponenciales (EWMA). Valores pequeños de estos parámetros se traducen en una adaptación lenta en el caso que hayan variaciones significativas en el comportamiento de los nodos. Ello también afecta al transitorio inicial de manera que éste es mayor y se tarda más en tener un valor fiable de la reputación de los nodos. Dicho transitorio no parece verse afectado por el valor asintótico al que se tiende, sino únicamente a los valores de α y β.
- En REMODE_sw, debido al hecho que la estimación de la reputación de los nodos se hace a partir de la reputación de los caminos, si dicha reputación se actualiza lentamente (valor de α pequeño), la reputación de los nodos también se actualiza lentamente independientemente del valor de β. Por otro lado, valores mayores de estos parámetros dan como resultado valores menos estables en la estimación de la reputación presentando ésta una mayor varianza.
- Hemos realizado simulaciones haciendo que REMODE_sw encamine y hemos verificado en escenario estático que con tráficos asimétricos se verifica el modelo planteado.
- Hemos realizado experimentos con REMODE_sw en escenario móvil que indican claramente que en escenario móvil sin importar el valor de inicialización de la reputación de un nodo, se consigue recuperar el valor verdadero de reputación que un nodo fuente tiene de determinado nodo intermedio
- La ventaja de los dos algoritmos de protocolos de encaminamiento planteados en esta tesis, es su simplicidad, y aunque no pueden calcular la reputación real de

los nodos individuales, pueden encontrar esos caminos con menos pérdidas, que era nuestro objetivo inicial.

- Nuestro objetivo principal, es minimizar la fracción de pérdida de paquetes al enviar paquetes a través de diferentes caminos. Por lo tanto, es natural utilizar la fracción de pérdida de paquetes observada de cada nodo como una métrica de su reputación.
- Las pérdidas de paquetes no siempre pueden ser causadas por el comportamiento egoísta de los nodos, sino también por la congestión del tráfico o la naturaleza aleatoria de las comunicaciones inalámbricas, o incluso por las capacidades limitadas de los nodos. Sin embargo, la causa real de una pérdida de paquetes no es capturada por nuestro enfoque y suponemos que siempre está relacionado con un comportamiento egoísta, que afectan directamente a la reputación de los nodos implicados.
- Nos gustaría resaltar que aunque nuestra métrica de reputación no es una medida precisa del comportamiento egoísta para cualquier nodo individual, es una buena estimación que ayuda a elegir el mejor camino de encaminamiento disponible.
- En DrepSR utilizamos una media móvil acumulada (CMA) para promediar las muestras de reputación, y una media móvil exponencial (EWMA) para detectar cambios de comportamiento en los nodos.
- En DrepSR proponemos un método novedoso basado en la conocida media móvil exponencial (EWMA) para detectar cualquier cambio en el comportamiento de reenvío de los nodos. La relación señal a ruido pico de la segunda diferencia de la media móvil exponencial se calcula para establecer los parámetros de la media móvil exponencial que optimice aquella relación señal a ruido pico. Esto nos permite detectar los límites de la ventana de tiempo estacionario en la cual la reputación de un nodo permanece estática. De esta manera podemos actualizar adecuadamente la medida del comportamiento de reenvío de un nodo promediando las muestras de reputación en esas ventanas de tiempo.
- Al compar la probabilidad de pérdida de paquetes y el número promedio de saltos para el protocolo DSR (encaminamiento con el número mínimo de saltos) y el protocolo DrepSR (encaminamiento que utiliza los caminos más

confiables). El protocolo DrepSR reduce la probabilidad de pérdidas en un 20%, mientras que aumenta el número promedio de saltos en un 8%.

- Finalmente, es importante mencionar que la mayoría de los protocolos de encaminamiento ya propuestos en la literatura utilizan una métrica basada en la distancia mínima. Alternativamente, nuestra propuesta utiliza una métrica basada en la reputación. Esta reputación se estima utilizando solo información local, evitando así tener que depender de las opiniones de terceros.

5.2 Artículos de investigación publicados como resultado del trabajo de tesis

En esta sección, listamos las publicaciones que hemos generado como resultado de esta tesis. Así mismo, el código desarrollado para NS-2.35 está disponible en:

https://sites.google.com/view/manets

- Lenin Guaya Delgado, Esteve Pallarés Segarra, Ahmad Mohammed Mezher and Jordi Forné. "A Novel Dynamic Reputation-based Routing Protocol for Mobile Ad Hoc Networks". *EURASIP Journal on Wireless Communications and Networking*. (2019). DOI : 10.1186/s13638-019-1375-7
- Lenin Guaya Delgado, Esteve Pallarès Segarra, Jordi Forné Muñoz. "REMODE: Un algoritmo para el cálculo de la reputación de nodos y caminos en redes ad-hoc cooperativas". Libro de Ponencias de las XII Jornadas de Ingeniería Telemática (JITEL 2015). pp. 277-284 (2015). ISBN: 978-84-606-8609-5.

5.3 Trabajo futuro

Durante el desarrollo de la tesis, algunas cuestiones nos llamaron la atención, y ahora se constituyen en las líneas de investigación futuras que se enumeran a continuación:

- Planeamos utilizar el modelo de Maximización de la Expectativa para estimar con precisión la reputación de los caminos y compararla con la estrategia propuesta desarrollada en este trabajo
- Estamos considerando la utilización de conjuntos de datos de código abierto para las redes móviles ad-hoc para llevar a cabo la evaluación del rendimiento de nuestro protocolo y algoritmos

- Evaluar los dos algoritmos propuestos utilizando tráfico de video en un simulador de código abierto. El tráfico de video por sus características particulares, siempre pone a prueba las redes, por lo cual sería interesante estudiar el rendimiento de REMODE_sw y DrepSR cuando trabajan con tráfico de video.

Anexos

Anexos

En el Anexo A, se muestran las cabeceras para las distintas opciones de DSR. En el Anexo B se muestra un manual de usuario para poder entender la integración y uso de las nuevas funcionalidades en el simulador de eventos discretos NS-2.35.

A Formato de cabecera de opciones DSR

La información del presente anexo es tomada del RFC4728 [**103**], con la finalidad de mostrar la cabecera adicional que se añade al paquete, cuando se trata de paquetes para redes ad-hoc (sin infraestructura). Además se debe tomar en cuenta que las dos propuestas de algoritmos de protocolos de encaminamiento son una mejora de DSR, y por ello utilizan las mismas cabeceras que DSR (sin estado de flujo).

El protocolo DSR hace uso de un encabezado especial que lleva información de control que se puede incluir en cualquier paquete IP existente. Este encabezado de opciones DSR de un paquete contiene una pequeña porción de 4 octetos de tamaño fijo, seguida

de una secuencia de cero o más opciones DSR con información opcional. El final de la secuencia de opciones DSR en el encabezado de opciones DSR está implícita en la longitud total del encabezado de opciones DSR.

Para agregar un encabezado de opciones DSR a un paquete, el encabezado de opciones DSR se inserta siguiendo el encabezado IP del paquete, antes de cualquier encabezado siguiente como un encabezado de capa de transporte tradicional (por ejemplo, TCP o UDP). Específicamente, el campo Protocolo en el encabezado IP se usa para indicar que un encabezado de opciones DSR sigue el encabezado IP, y el campo Next Header en el encabezado de opciones DSR se usa para indicar el tipo de encabezado del protocolo (como un encabezado de capa de transporte) siguiendo el encabezado de opciones DSR.

Si alguno de los encabezados sigue el encabezado de opciones DSR en un paquete, la longitud total del encabezado de opciones DSR (y por lo tanto la longitud combinada total de todas las opciones DSR presentes) debe ser un múltiplo de 4 octetos. Este requisito preserva la alineación de estos siguientes encabezados en el paquete.

A.1 Cabecera de opciones DSR (Dynamic Source Routing)

La parte fija del encabezado de opciones DSR se utiliza para transportar información que debe estar presente en cualquier encabezado de opciones DSR. Esta parte fija del encabezado de opciones DSR tiene el siguiente formato:

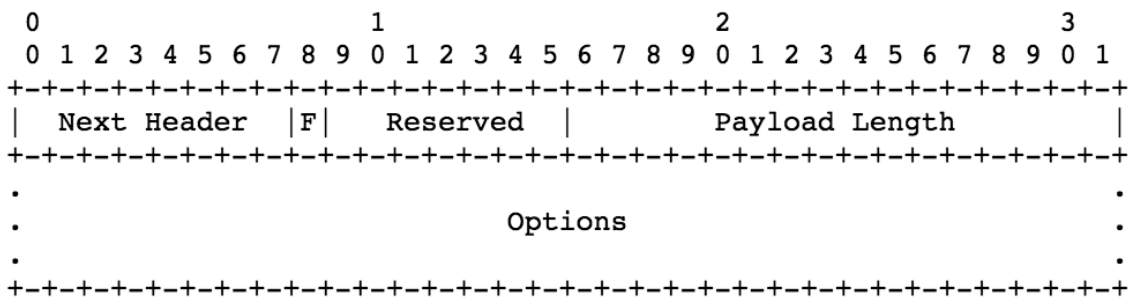


Figura A.1: Parte fija del encabezado de opciones DSR

Next Header (Encabezado Siguiente)

Selector de 8 bits que identifica el tipo de encabezado que sigue inmediatamente al encabezado de DSR. Si no sigue ningún encabezado, el encabezado siguiente debe tener el valor de 59, es decir "No Next Header".

F (Encabezado de Estado de Flujo)

Bit bandera, debe establecerse en 0. Este bit se establece en un Encabezado de Estado de Flujo DSR, y se borra en un encabezado DSR.

Reserved (Reservado)

Debe enviarse como 0 e ignorarse en la recepción.

Payload Length (Longitud de carga útil)

La longitud del encabezado de opciones DSR, excluyendo la porción fija de 4 octetos. El valor del campo Payload Length define la longitud total de todas las opciones llevadas en el encabezado de opciones DSR.

Options (Opciones)

Campo de longitud variable; la longitud del campo opciones se especifica mediante el campo Payload Length en este encabezado de opciones DSR. Contiene una o más piezas de información opcional (opciones DSR).

A.2 Opción de petición de camino

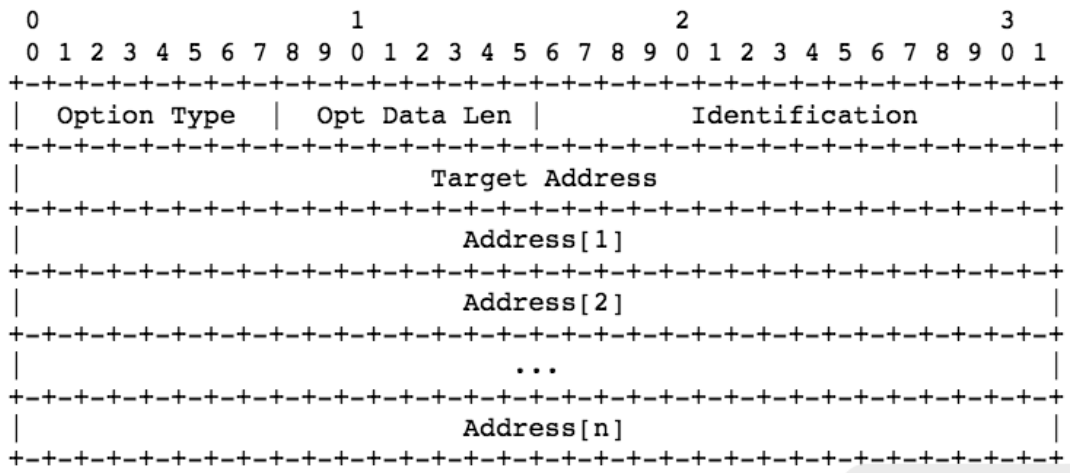


Figura A.2: Opción de petición de camino

Campos de IP:

Dirección de la Fuente (Source Address)

Debe establecerse la dirección del nodo que origina este paquete. Los nodos intermedios que retransmiten el paquete para propagar la petición de camino no deben cambiar este campo.

Destination Address (Dirección de Destino)

Debe establecerse en la dirección de difusión limitada de IP (255.255.255.255).

TTL (Límite de salto)

Puede variarse de 1 a 255, por ejemplo, para implementar peticiones de camino sin propagación y búsquedas de anillo expansible de petición de camino.

Campos de petición de camino:

Option Type (Tipo de Opción)

1. Los nodos que no entienden esta opción ignorarán esta opción.

Opt Data Len (Longitud de la Opción de Datos)

Entero sin signo de 8 bits. Longitud de la opción, en octetos, excluyendo los campos Option Type y Opt Data Len. Debe establecerse igual a (4 * n) + 6, donde n es el número de direcciones en la opción de petición de camino.

Identification (Identificación)

Un valor único generado por el iniciador (remitente original) de la petición de camino. Los nodos que inician una petición de camino generan un nuevo valor de identificación para cada petición de camino, por ejemplo, basado en un contador de número de secuencia de todas las peticiones de camino iniciadas por el nodo.

Este valor permite que un nodo receptor determine si ha visto recientemente una copia de esta petición de camino. Si este nodo de recepción encuentra este valor de identificación (para esta dirección de origen de IP y dirección de destino) en su tabla de petición de camino (en la caché de valores de identificación en la entrada para este nodo de inicio), este nodo de recepción debe descartar la petición de camino. Cuando se propaga una petición de camino, este campo debe copiarse de la copia recibida de la petición de camino que se está propagando.

Target Address (Dirección de Destino)

La dirección del nodo que es el objetivo de la petición de camino.

Address[1..n] (Dirección [1..n])

La Address[i], es la dirección IPv4 del i-ésimo nodo registrado en la opción de petición de camino. La dirección dada en el campo Source Address en el encabezado IP es la dirección del iniciador del descubrimiento de camino y no debe aparecer en la lista en los campos Address[i]; la dirección dada en Address[1] es, por lo tanto, la dirección IPv4 del primer nodo en el camino después del iniciador. El número de direcciones presentes en este campo se indica mediante el campo Opt Data Len en la opción (n = (Opt Data Len - 6) / 4). Cada nodo que propaga la petición de camino agrega su propia dirección a esta lista, aumentando el valor Opt Data Len en 4 octetos.

La opción de petición de camino no debe aparecer más de una vez dentro de un encabezado de opciones de DSR.

A.3 Opción de contestación de camino

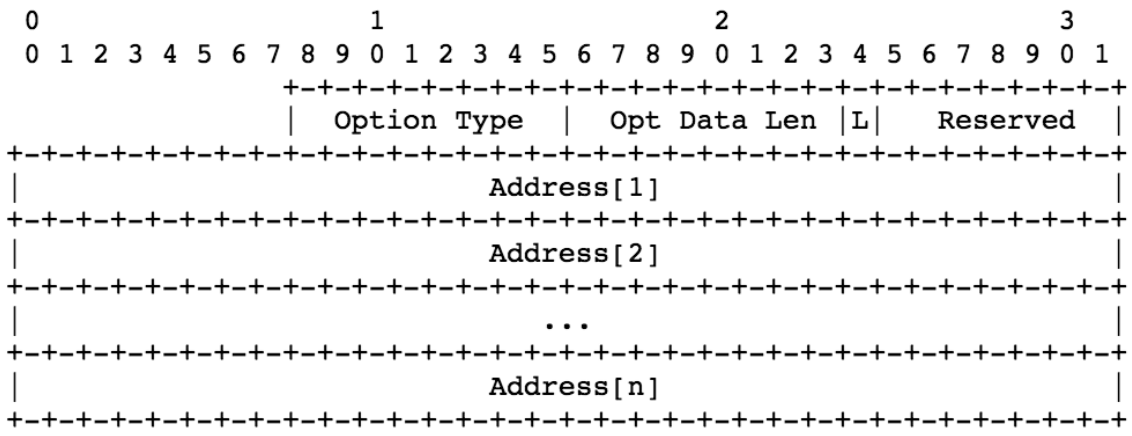


Figura A.3: Opción de contestación de camino

Campos de IP:

Source Address (Dirección de la Fuente)

Dirección del nodo que envía la respuesta de camino. En el caso de que un nodo envíe una respuesta desde su caché de camino o envíe una contestación de camino gratuita (GRR), esta dirección puede diferir de la dirección que fue el objetivo del descubrimiento de camino.

Destination Address (Dirección de Destino)

Debe establecerse en la dirección del nodo fuente del camino que se devuelve. Copiado del campo dirección de origen de la petición de camino que genera la contestación de camino, o en el caso de una contestación de camino gratuita (GRR), copiado del campo Source Address del paquete de datos que desencadena la respuesta gratuita.

Campos de contestación de camino:

Option Type (Tipo de Opción)

2. Los nodos que no entienden esta opción ignorarán esta opción.

Opt Data Len (Longitud de la Opción de Datos)

Entero sin signo de 8 bits. Longitud de la opción, en octetos, excluyendo los campos Option Type y Opt Data Len. Debe establecerse igual a (4 * n) + 1, donde n es el número de direcciones en la opción de contestación de camino.

L (Último Salto Externo)

Establecer para indicar que el último salto dado por la contestación de camino (el enlace de la Address[n-1] a la Address[n]) es en realidad un camino arbitrario en una red externa a la red DSR; el camino exacto fuera de la red DSR no está representado en la contestación de camino.

Los nodos que almacenan en caché este salto en su caché de caminos deben marcar el salto en caché con el indicador externo. Dichos saltos no deben devolverse en una contestación de camino desde caché de caminos, generada desde esta entrada de caché de caminos, y la selección de caminos desde la caché de caminos para encaminar un paquete que se envía debería preferir caminos que no contengan saltos marcados como externos.

Reserved (Reservado)

Debe enviarse como 0 e ignorado en la recepción.

Dirección [1..n] (Address[1..n])

El camino de origen devuelto por la contestación de camino. El camino indica una secuencia de saltos originada en el nodo de origen especificado en el campo Destination Address del encabezado IP del paquete que transmite la contestación de camino, a través de cada uno de los nodos de Address[i] en el orden indicado en la contestación de camino, finalizando en el nodo indicado por Address[n]. El número de direcciones presentes en el campo Address[1..n] se indica mediante el campo Opt Data Len en la opción (n = (Opt Data Len - 1) / 4).

Una opción de contestación de camino puede aparecer una o más veces dentro de un encabezado de opciones DSR.

A.4 Opción de camino fuente de DSR (*Dynamic Source Routing*)

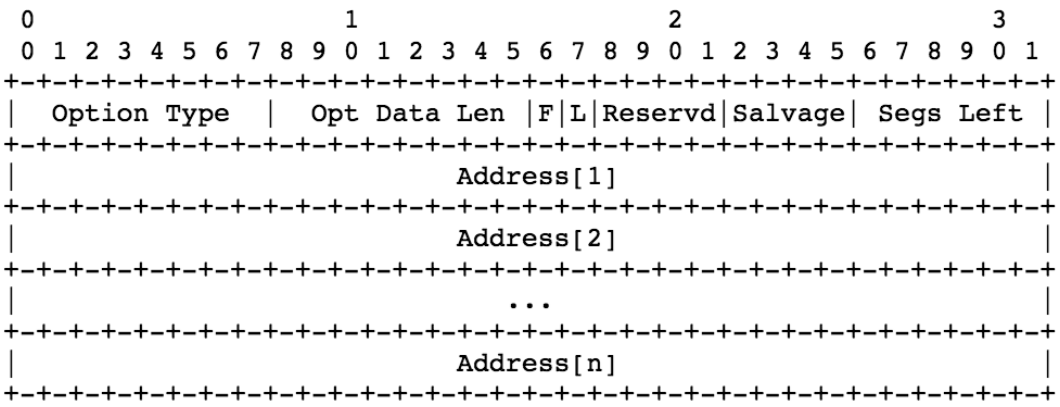


Figura A.4: Opción de camino fuente de DSR

Option Type (Tipo de Opción)

96. Los nodos que no entienden esta opción dejarán caer el paquete.

Opt Data Len (Longitud de la Opción de Datos)

Entero sin signo de 8 bits. Longitud de la opción, en octetos, excluyendo los campos Option Type y Opt Data Len. Para el formato de la opción camino fuente de DSR definida aquí, este campo debe establecerse en el valor (n * 4) + 2, donde n es el número de direcciones presentes en los campos Address[i].

F (Primer Salto Externo)

Establecer para indicar que el primer salto indicado por la opción de camino fuente de DSR es en realidad un camino arbitrario en una red externa a la red DSR; el camino

exacto fuera de la red DSR no está representado en la opción de camino fuente de DSR. Los nodos que almacenan este salto en caché de caminos deben marcar el salto en caché con el indicador externo. Dichos saltos no se deben devolver en una contestación de camino generada desde esta entrada de caché de caminos, y la selección de caminos desde la cache de caminos para encaminar un paquete que se está enviando debe preferir los caminos que no contienen saltos marcados como externos.

L (Último Salto Externo)

Establecer para indicar que el último salto indicado por la opción de camino fuente de DSR es en realidad un camino arbitrario en una red externa a la red DSR; el camino exacto fuera de la red DSR no está representado en la opción de camino fuente de DSR. Los nodos que almacenan en caché de caminos este salto deben marcar el salto en caché con el indicador externo. Dichos saltos no se deben devolver en una contestación de camino generada desde esta entrada de caché de caminos, y la selección de caminos desde la caché de caminos, para encaminar un paquete que se está enviando debe preferir los caminos que no contienen saltos marcados como externos.

Reservd (Reservado)

Debe enviarse como 0 e ignorado en la recepción.

Salvage (Salvar)

Un entero sin signo de 4 bits. Número de veces que este paquete se ha recuperado como parte del encaminamiento DSR.

Segs left (Segmentos a la izquierda)

Número de segmentos de camino restantes, es decir, número de nodos intermedios explícitamente listados que aún deben visitarse antes de llegar al destino final.

Address[1..n] (Dirección [1..n])

La secuencia de direcciones del camino desde el origen (o camino fuente). El número de direcciones presentes en el campo Address[1..n] se indica mediante el campo Opt Data Len en la opción (n = (Opt Data Len - 2) / 4).

Al reenviar un paquete a lo largo de un camino fuente DSR utilizando una opción camino fuente de DSR en el encabezado de opciones DSR del paquete, el campo Destination Address en el encabezado IP del paquete siempre se establece en la dirección del destino final del paquete. Un nodo que recibe un paquete que contiene un encabezado de opciones DSR con una opción de camino fuente de DSR debe examinar el camino fuente indicado para determinar si es el nodo del siguiente salto es el destinado para el paquete y cómo reenviar el paquete.

A.5 Opción de error de camino

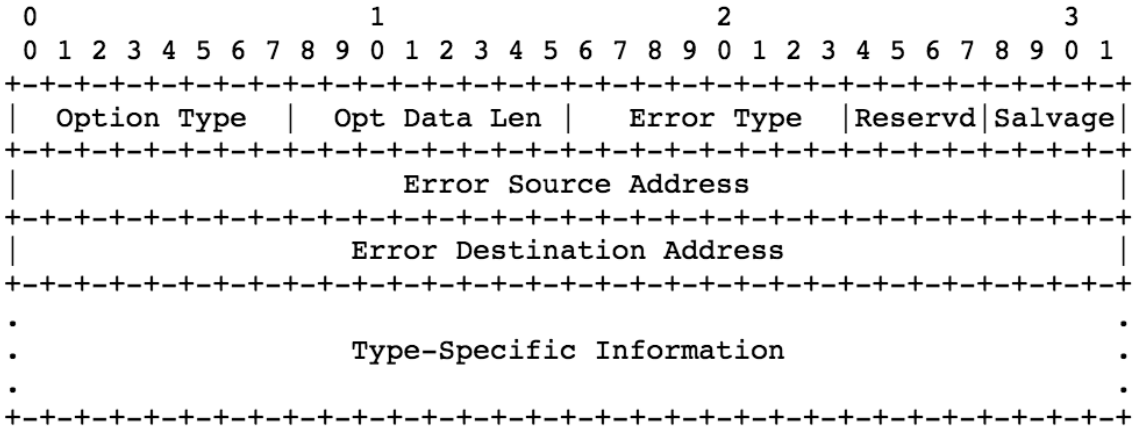


Figura A.5: Opción de error de camino

Option Type (Tipo de Opción)

3. Los nodos que no entienden esta opción ignorarán esta opción.

Opt Data Len (Longitud de la Opción de Datos)

Entero sin signo de 8 bits. Longitud de la opción, en octetos, excluyendo los campos Option Type y Opt Data Len.

Para la definición actual de la opción error de camino, este campo debe establecerse en 10, más el tamaño de cualquier Type-Specific Information presente en el error de camino. Se pueden incluir extensiones adicionales al formato de opción error de camino después de la parte Type-Specific Information de la opción de error de camino especificada anteriormente. La presencia de dichas extensiones se indicará mediante el campo Opt Data Len.

Cuando el Opt Data Len es mayor que el requerido para la parte fija del error de camino más la Type-Specific Information necesaria según lo indicado por el valor del Option Type en la opción, los octetos restantes se interpretan como extensiones. Actualmente, tales extensiones adicionales no se han definido.

Error Type (Tipo de error)

El tipo de error encontrado. Actualmente, se definen los siguientes valores de tipo:

1 = NODE_UNREACHABLE

2 = FLOW_STATE_NOT_SUPPORTED

3 = OPTION_NOT_SUPPORTED

Otros valores del campo Error Type están reservados para uso futuro.

Reservd (Reservado)

Reservado. Debe enviarse como 0 e ignorarse en la recepción.

Salvage (Salvar)

Un entero sin signo de 4 bits. Copiado desde el campo Salvage en la opción de camino fuente de DSR del paquete que desencadena el error de camino.

El "contador de salvamento total" de la opción de error de camino se deriva del valor en el campo Salvage de esta opción de error de camino y todas las opciones anteriores de error de camino en el paquete de la siguiente manera: el contador de salvamento total es la suma de (por cada una de tales opciones de error de camino) uno más el valor en el campo Salvage de esa opción de error de camino.

Error Source Address (Dirección de la Fuente del Error)

La dirección del nodo que origina el error de camino (por ejemplo, el nodo que intentó reenviar un paquete y descubrió la falla del enlace).

Error Destination Address (Dirección del Destino para el Error)

La dirección del nodo al que se debe entregar el error de camino. Por ejemplo, cuando el campo Error Type se establece en NODE_UNREACHABLE, este campo se

establecerá en la dirección del nodo que generó la información de encaminamiento alegando que el salto desde la dirección de origen del error a la dirección del nodo inalcanzable (especificada en la información específica del tipo) fue un salto válido.

Type-Specific Information (Información Específica de Tipo)

Información específica del tipo de error de este mensaje de error de camino.

Una opción de error de camino puede aparecer una o más veces dentro de un encabezado de opciones de DSR.

A.6 Opción de petición de acuse de recibo

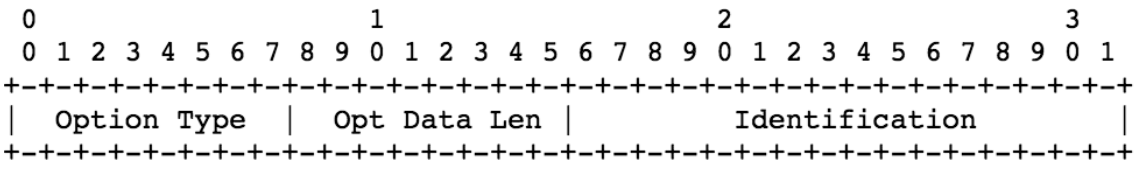


Figura A.6: Opción de petición de acuse de recibo

Option Type (Tipo de opción)

160. Los nodos que no entienden esta opción eliminarán la opción y devolverán un Error de camino.

Opt Data Len (Longitud de la Opción de Datos)

Entero sin signo de 8 bits. Longitud de la opción, en octetos, excluyendo los campos Option Type y Opt Data Len.

Identification (Identificación)

El campo Identification se establece en un valor único y se copia en el campo Identification de la opción de acuse de recibo cuando el nodo que recibe el paquete lo devuelve a través de este salto.

Una opción de petición de acuse de recibo no debe aparecer más de una vez dentro de un encabezado de opciones DSR.

A.7 Opción de acuse de recibo

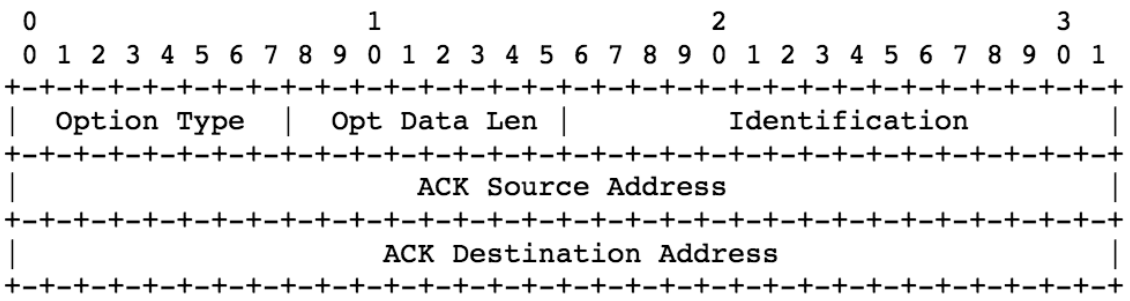


Figura A.7: Opción de acuse de recibo

Option Type (Tipo de Opción)

32. Los nodos que no entienden esta opción eliminarán la opción.

Opt Data Len (Longitud de la Opción de Datos)

Entero sin signo de 8 bits. Longitud de la opción, en octetos, excluyendo los campos Option Type y Opt Data Len.

Identification (Identificación)

Copiado desde el campo Identification de la opción de solicitud de acuse de recibo del paquete.

ACK Source Address (Dirección de fuente ACK)

La dirección del nodo que origina el acuse de recibo.

ACK Destination Address (Dirección de destino ACK)

La dirección del nodo al que se debe entregar el acuse de recibo.

Una opción de acuse de recibo puede aparecer una o más veces dentro de un encabezado de opciones DSR.

B Manual de usuario

En este Anexo B, mostramos mediante ejemplos prácticos, la forma en la que se utilizan las herramientas software de esta investigación.

B.1 Introducción

Las simulaciones para REMODE_sw las hemos realizado en NS-2.35 utilizando escenarios de movilidad generados con Bonnmotionv3.0.0, y el análisis se lo realizó con "Trace Analazer v3.0". La implementación de la variante estática de REMODE se las realizó principalmente, sobre los archivos de programa del protocolo DSR; y, la implementación del analizador de trazas "Trace Analizer v3.0", se la realizó sobre el archivo principal encargado de la impresión de trazas que es el cmu-trace.cc. Bonnmotionv3.0.0, nos permite obtener escenarios con determinada dimensión, patrón de movimiento, cantidad de nodos y velocidad, es decir nos permite obtener patrones de movilidad de características muy variadas, y adaptadas a cada una de nuestras necesidades de simulación. En cuanto a la experimentación con DrepSR, se la realizó en MATLAB.

Un uso avanzado de REMODE_sw en NS2.35 consiste en la modificación de los selectores que se encuentran al inicio del archivo dsragent.cc, lo que permite habilitar o deshabilitar distintos módulos con los que cuenta DSR; y, al configurar estos selectores propios de DSR, también se modifica el funcionamiento de REMODE_sw.

Los scripts OTCL utilizados en esta tesis se diferencian de los existentes por un conjunto de selectores colocados al inicio del script. Más adelante se explicará cada uno de los selectores presentes en el script OTCL.

B.2 Integrando REMODE_sw (Reputation for Mobile Devices Static Weight) a NS-2.35 (Network Simulator Versión 2.35)

Con la implementación del entorno de trabajo (cuyo link de descarga se presentó en el capítulo de conclusiones), y con la finalidad de realizar comparaciones de métricas de QoS, es posible seleccionar 2 variantes de DSR y tres variantes de REMODE como lo indican las siguientes líneas del script OTCL de simulación:

```
# 0: DSR(complete), 1:REMODE_static_weight, 2:REMODE_dynamic_weight_1, 3:REMODE_dynamic_weight_2, 4:DSR(REMODE imitates DSR)

Agent/DSRAgent set remode_selector 1
```

La variable remode_selector tiene un valor de 1, por tanto en este ejemplo activamos REMODE_sw en la simulación. Se realizaron miles de experimentos utilizando una estación de trabajo (Workstation), variando no sólo remode_selector, sino variando cada uno de los parámetros de simulación del script OTCL, en busca de ampliar conocimientos y de probar varios modelos matemáticos nuevos basados en reputación del nodo que pudieran mejorar el rendimiento de la red. Se determinó que la variante REMODE_sw presentaba una mejora significativa en comparación a DSR, por ello en el capítulo 3 se presentan los resultados de dichos experimentos.

Para que podamos agregar la implementación de las variantes de REMODE y del Trace Analizer v3.0, primeramente debemos tener correctamente instalado el simulador NS-2.35, luego reemplazamos algunos archivos de programa del NS-2.35, ya que tanto las variantes de REMODE, como Trace Analizer v3.0 están implementados en archivos de programa que se encuentran por defecto en NS-2.35.

Las ubicaciones de los archivos a modificar para integrar las variantes de REMODE son las siguientes:

/usr/src/ns-allinone-2.35/ns-2.35/dsr/dsragent.cc

/usr/src/ns-allinone-2.35/ns-2.35/dsr/dsragent.h

/usr/src/ns-allinone-2.35/ns-2.35/dsr/mobicache.cc

/usr/src/ns-allinone-2.35/ns-2.35/dsr/routecache.h

/usr/src/ns-allinone-2.35/ns-2.35/apps/rtp.cc

/usr/src/ns-allinone-2.35/ns-2.35/apps/rtp.h

/usr/src/ns-allinone-2.35/ns-2.35/tcp/rtcp.cc

/usr/src/ns-allinone-2.35/ns-2.35/tcp/tcp-linux.cc

/usr/src/ns-allinone-2.35/ns-2.35/tcp/tcp-vegas.cc

/usr/src/ns-allinone-2.35/ns-2.35/tcp/tcp.cc

/usr/src/ns-allinone-2.35/ns-2.35/tcp/tcp.h

La ubicación del archivo a modificar para integrar el analizador de traza "Trace Analizer v3.0" es la siguiente:

/usr/src/ns-allinone-2.35/ns-2.35/trace/cmu-trace.cc

Luego de que reemplazamos los archivos, realizamos una nueva compilación, mediante los comandos make clean y make desde la siguiente ubicación:

root@lenin-reputation:/usr/src/ns-allinone-2.35/ns-2.35#

Luego de la compilación ya tenemos integrado REMODE_sw en NS-2.35, esta variante de REMODE está lista para ser utilizada desde algún script OTCL.

B.3 Generando escenarios con Bonnmotionv3.0.0

A continuación explicamos la utilización del generador de escenarios de movilidad mediante dos ejemplos. Primeramente Bonnmotionv3.0.0 debe estar correctamente instalado, luego nos ubicamos en:

root@lenin-reputation:/usr/src/bonnmotion-3.0.0/bin#

Para el primer ejemplo, ejecutamos las tres líneas que vienen a continuación para obtener un escenario llamado "100mn_REMODE_2mps_10_ready.ns_movements", el cual es un escenario con una cantidad de 100 nodos, duración de 200 segundos, dimensión de 900mx900m, patrón de movimiento aleatorio (RandomWaypoint), y velocidad máxima de los nodos de 2m/s.

bm -f 100mn_REMODE_2mps_10 RandomWaypoint -n 100 -d 200 -x 900 -y 900
bm -f 100mn_REMODE_2mps_10_ready -I 100mn_REMODE_2mps_10.params RandomWaypoint -h 2.0
bm NSFile -f 100mn_REMODE_2mps_10_ready

Entonces, uno de los archivos resultantes, y que es de nuestro interés, es el archivo de texto llamado "100mn_REMODE_2mps_10_ready.ns_movements", el cual está listo para ser referenciado desde el script OTCL.

Algo importante a considerar en la generación de los escenario de movilidad, es la densidad de los nodos, para este ejemplo que acabamos de exponer, la densidad es de 123,46 nodos/Km^2. Para evitar colisiones excesivas, recomendamos utilizar densidades de 50, 100, y como mucho de 150 nodos/Km^2

A continuación se muestra un segundo ejemplo. Se requiere un escenario con patrón de movimiento RandomWaypoint, dimensión de 300mx300m, duración de 400 segundos, 10 nodos, y cuya velocidad máxima sea de 1.5m/s. Además, luego de creado el escenario móvil, se requiere que los nodos 0, 1, 2, 3, 8 y 9 se mantengan en una determinada posición fija.

Para que podamos crear el escenario dentro de Bonnmotionv3.0.0, ejecutamos las tres líneas siguientes:

bm -f 10mn_1p5mps_400s_r1 RandomWaypoint -n 10 -d 400 -x 300 -y 300
bm -f 10mn_1p5mps_400s_r1_ready -I 10mn_1p5mps_400s_r1.params RandomWaypoint -h 1.5
bm NSFile -f 10mn_1p5mps_400s_r1_ready

Luego de modificado (color azul) el archivo "10mn_1p5mps_400s_r1_ready.ns_movements", para que los nodos 0, 1, 2, 3, 8 y 9 se mantengan en una determinada posición fija, obtenemos lo siguiente:

$node_(0) set X_ 0.0
$node_(0) set Y_ 0.0
$node_(1) set X_ 75.0
$node_(1) set Y_ 75.0
$node_(2) set X_ 150.0
$node_(2) set Y_ 150.0
$node_(3) set X_ 225.0
$node_(3) set Y_ 225.0
$node_(4) set X_ 85.13334494708587
$node_(4) set Y_ 78.2158747241754
$ns_ at 0.0 "$node_(4) setdest 221.89435083706337 78.2158747241754 0.6440818578684852"
$ns_ at 212.33482082940873 "$node_(4) setdest 221.89435083706337 78.2158747241754 0.0"
$ns_ at 230.6826893674729 "$node_(4) setdest 221.89435083706337 89.16384388915056 0.7056423756462347"

$ns_ at 246.19758691731022 "$node_(4) setdest 221.89435083706337 89.16384388915056 0.0"

$ns_ at 278.16400035958213 "$node_(4) setdest 221.89435083706337 89.16384388915056 0.0"

$ns_ at 310.48606088737324 "$node_(4) setdest 221.89435083706337 89.16384388915056 0.0"

$node_(5) set X_ 102.10811258139994

$node_(5) set Y_ 309.3369617619447

$ns_ at 0.0 "$node_(5) setdest 102.10811258139994 309.3369617619447 0.0"

$ns_ at 6.033126409914075 "$node_(5) setdest 102.10811258139994 132.31766584393105 0.7135029563239788"

$ns_ at 254.1320226359794 "$node_(5) setdest 102.10811258139994 132.31766584393105 0.0"

$ns_ at 265.6161628889963 "$node_(5) setdest 102.10811258139994 202.58965981023331 0.6445853623113912"

$node_(6) set X_ 181.77128250191132

$node_(6) set Y_ 84.64330916387676

$ns_ at 0.0 "$node_(6) setdest 103.11314256368544 84.64330916387676 0.9772915806826153"

$ns_ at 80.4858462847751 "$node_(6) setdest 103.11314256368544 84.64330916387676 0.0"

$ns_ at 121.28289378233785 "$node_(6) setdest 308.0068144934127 84.64330916387676 1.094412333456449"

$ns_ at 308.50087887449945 "$node_(6) setdest 308.0068144934127 84.64330916387676 0.0"

$ns_ at 325.5243075051544 "$node_(6) setdest 308.0068144934127 51.613269791518725 1.4134447635620384"

$node_(7) set X_ 192.46574225543316

$node_(7) set Y_ 168.47454068728177

$ns_ at 0.0 "$node_(7) setdest 10.05095123437343 168.47454068728177 1.370585338213216"

$ns_ at 133.09261812100476 "$node_(7) setdest 10.05095123437343 168.47454068728177 0.0"

$ns_ at 158.52019076150373 "$node_(7) setdest 89.26181763511781 168.47454068728177 0.9795122145870083"

$ns_ at 239.38785654568392 "$node_(7) setdest 89.26181763511781 168.47454068728177 0.0"

$ns_ at 267.01864177122343 "$node_(7) setdest 89.26181763511781 112.27592671308443 0.7761615380243253"

$ns_ at 339.42446352483375 "$node_(7) setdest 89.26181763511781 112.27592671308443 0.0"

$ns_ at 384.68311119047485 "$node_(7) setdest 89.26181763511781 94.60387376491163 1.153762566793789"

$node_(8) set X_ 300.0

$node_(8) set Y_ 150.0

$node_(9) set X_ 150.0

$node_(9) set Y_ 300.0

Entonces, en el escenario que acabamos de crear, de los 10 nodos creados, 4 son móviles y 6 permanecerán fijos. Este escenario está listo para ser referenciado desde un script OTCL.

B.4 Creando nodos egoístas

La forma en que hemos creado las pérdidas en los nodos egoístas, es haciendo que se pierdan paquetes de datos (no de control) con una determinada probabilidad p, similar al comportamiento malicioso de un ataque “agujero gris”, en el cual el nodo participa en el descubrimiento de camino, pero luego en la retransmisión pierde paquetes de datos con determinada probabilidad p. En esta investigación simplemente se necesita simular un comportamiento egoísta mediante la pérdida de paquetes, por ello no se profundizará en establecer diferencias entre comportamientos maliciosos y egoístas.

A continuación, mostramos un ejemplo de la creación de nodos egoístas, insertando algunas líneas de código (en color azul) dentro de “dsragent.cc”. El objetivo de este ejemplo es conseguir que el nodo 7, tenga una FPLn7 de 0.25 (pérdidas del 25%), el nodo 9 tenga una FPLn9 de 0.15 (pérdidas del 15%), y cualquier otro nodo del

escenario que sea completamente cooperativo, es decir que tenga una FPLnj de 0 (sin pérdidas):

//Generating losses for nodes 7 and 9 from /usr/src/ns-allinone-2.35/ns-2.35/dsr/dsragent.cc

…

```
 void
DSRAgent::recv(Packet* packet, Handler*){
…

     if (srh->route_request())
          { // propagate a route_request that's not for us
            handleRouteRequest(p);
          }
     else
          { // we're not the intended final recpt, but we're a hop

//Seventh and nineth with losses.
if((strncmp(net_id.dump(),"7",1)==0))//7
{//Seventh with losses.
counter_fw_1=rand()%100;
  if (((counter_fw_1>=0)&&(counter_fw_1<75))||(iph->sport()==255))//75
  {
  trace ("X %.5f_%s_%s", Scheduler::instance().clock(), net_id.dump(), "Node 7 forwarding (node 7 with FPL=0.25)");
  handleForwarding(p);  //Forwarding.
  }
  else
  {
  trace ("X %.5f_%s_%s", Scheduler::instance().clock(), net_id.dump(), "Node 7 does not forwarding (node 7 with FPL=0.25)");
  }
```

```
}
else if((strncmp(net_id.dump(),"9",1)==0))//9
{//Nineth with losses.
counter_fw_1=rand()%100;
 if (((counter_fw_1>=0)&&(counter_fw_1<85))||(iph->sport()==255))//85
 {
 trace ("X %.5f_%s_%s", Scheduler::instance().clock(), net_id.dump(), "Node 9 forwarding (node 9 with FPL=0.15)");
                handleForwarding(p);  //Forwarding.
 }
 else
 {
 trace ("X %.5f_%s_%s", Scheduler::instance().clock(), net_id.dump(), "Node 9 does not forwarding (node 9 with FPL=0.15)");
 }
}
else //1. Comment when the losses are eliminated
{//2. Comment when the losses are eliminated
 trace ("X %.5f_%s_%s", Scheduler::instance().clock(), net_id.dump(), "Forwarding");
 handleForwarding(p);  //Forwarding.
}//3. Comment when the losses are eliminated
…
}
…
```

Entonces, desde el método de recepción "DSRAgent::recv()" que existe dentro de dsragent.cc, seleccionamos un determinado nodo (7 o 9 por ejemplo), luego generamos un aleatorio (en counter_fw_1) del 0 al 99. Si dicho aleatorio se encuentra entre determinado rango de reenvío, o si el paquete es de control, entonces lo reenviamos.
Analizando las 3 líneas principales que generan una probabilidad de pérdidas de 0.25 para el nodo 7, tenemos lo siguiente:

```
counter_fw_1=rand()%100;
if (((counter_fw_1>=0)&&(counter_fw_1<75))||(iph->sport()==255))//75
```

{handleForwarding(p); //Forwarding.}

Y, nos damos cuenta que sería lo mismo utilizar:

counter_fw_1=rand()%100;
if (((counter_fw_1>=0)&&(counter_fw_1<75))||(cmh->size()<PACKET_SIZE))//75
{handleForwarding(p); //Forwarding.}

Debido a que en esta tesis se utilizó paquetes de datos superiores o iguales a 512 Bytes, y escenarios en los cuales el número medio de saltos máximo era de seis saltos, por lo cual las cabeceras eran muy pequeñas. Entonces, es lo mismo referirse a un paquete de control como un paquete que tiene marcado su puerto fuente (y también su puerto destino) como 255, o como un paquete cuyo tamaño es menor al tamaño del paquete de datos. Simplemente, lo que estamos evitando con el código en color rojo, es perder paquetes de control.

B.5 Entendiendo los selectores de DSR que se utilizan en REMODE_sw.

Cuando hablamos de selectores, nos referimos a dos tipos de selectores, los primeros que se encuentran dentro de "dsragent.cc", que es el archivo principal de la implementación de DSR que viene por defecto en NS-2.35; y, los segundos que se encuentran dentro del script OTCL de cada simulación. En esta sección enumeramos y explicamos los selectores que se encuentran en el código fuente de DSR en el simulador NS2.35, dentro de dsragent.cc.

Para implementar un protocolo tenemos dos opciones, la primera es implementar todo el protocolo desde cero, lo cual llevaría demasiado tiempo, ya que gran parte del código fuente existente no se reutilizaría. La segunda opción, es realizar las mejoras en determinados métodos, e inclusive crear algunos métodos y atributos nuevos sobre un protocolo existente. En esta tesis hemos utilizado la segunda opción, es decir sobre el código fuente de DSR insertamos el sistema de reputación y damos lugar a REMODE_sw.

A continuación, detallamos la utilidad de los selectores disponibles en DSR, dentro de NS-2.35, con la finalidad de entender cuáles métodos de DSR reutiliza REMODE_sw.

De 12 selectores principales que dispone DSR, para REMODE_sw, únicamente

habilitamos 4 selectores, los cuales son mostrados a continuación en color azul:

// Selectors at the beginning of /usr/src/ns-allinone-2.35/ns-2.35/dsr/dsragent.cc

…

dsragent_enable_flowstate = false;//1.
//1. Caminos fuente implícitos, es decir estado de flujo DSR
dsragent_prefer_default_flow = false;//2.
//2. Los bytes de cabecera adicional en el paquete pueden ser completamente eliminados por el uso de flujos por defecto. En estado de flujo DSR
dsragent_prefer_shorter_over_default = false;//3.
//3. Preferir camino más corto sobre camino por defecto.
//En DSR con estado de flujo, las configuraciones 1, 2 y 3 debemos habilitarlas, con las variantes de REMODE o con DSR sin estado de flujo en cambio debemos deshabilitadas. Estas tres primeras configuraciones corresponden al estado de flujo DSR

dsragent_snoop_forwarded_errors = true;//4.
//4. ¿Debemos poner los errores que reenviamos a nuestra caché de caminos?. Debemos comprobar para ver si es un paquete de error de camino el que estamos manejando, y si es así llamamos al método processBrokenRouteError
dsragent_snoop_source_routes = false;//5.
// 5. ¿Deberíamos husmear en cualquier camino fuente que vemos?
dsragent_propagate_last_error = true;//6.
// 6. ¿Debemos tomar los datos del último mensaje de error de camino que recibimos y propagarlo en la próxima propagación de petición de camino que hacemos?. Esto es propagación de error de camino gratuito
dsragent_send_grat_replies = false;//7.
// 7. ¿Deberíamos enviar respuestas gratuitas para acortar el camino? En camino fuente enviar al origen una contestación de camino gratuita para indicar la existencia de un camino más corto.
dsragent_salvage_with_cache = false;//8.
// 8. Si recibimos un error de camino (falla xmit), salvaremos el paquete utilizando memoria cache

dsragent_use_tap = true;//9.
// 9. Deberíamos escuchar para capturar todo el tráfico que circula por la red?
dsragent_reply_from_cache_on_propagating = false;//10.
// 10. ¿Deberíamos consultar la caché de caminos antes de propagar peticiones de camino y responder si es posible?
dsragent_ring_zero_search = false;//11.
// 11. No propagar peticiones de camino como primera acción en cada descubrimiento de camino.
//NOTA: para desactivar completamente la respuesta desde caché, debemos establecer los selectores dsragent_ring_zero_search y dsragent_reply_from_cache_on_propagating en false
dsragent_dont_salvage_bad_replies = true;//12.
// 12. Si tenemos un error de camino (fallo xmit) en un paquete, y el paquete contiene un camino de respuesta, ¿deberíamos escanear la respuesta para ver si contiene el enlace muerto?

…

La versión por defecto de DSR, que es DSR con estado de flujo, requiere que estos 12 selectores que acabamos de presentar estén habilitados para su normal funcionamiento. Al habilitar alguno de estos selectores, habilitamos uno o más métodos.

Las variantes de REMODE para su normal funcionamiento sólo requieren tener habilitados cuatro selectores, y como experimento hemos habilitado cuatro selectores adicionales (cuatro funcionalidades) que las variantes de REMODE no necesitan, y se ha verificado el incremento del tiempo de ejecución (para un escenario determinado) de 4 minutos y 53 segundos a 30 minutos, es decir entre más selectores se habilitan, el protocolo usa más métodos, y la CPU del computador que ejecuta la simulación debe ejecutar mayor cantidad de instrucciones, ello explica el hecho de que los dispositivos con DSR (DSR con estado de flujo, con los 12 selectores habilitados) consuman más energía que los dispositivos con REMODE_sw. Luego de tomado en cuenta lo antes mencionado, se determinó que para que sea posible comparar DSR vs. REMODE_sw en igualdad de condiciones, es decir con la misma cantidad de módulos, y cuya diferencia principal sea que no escoga el camino más corto, sino el de mejor reputación,

a la implementación de REMODE_sw utilizando únicamente los cuatro selectores que se los mostró en color azul anteriormente, y trabajando únicamente con caché primaria, hicimos que imite el comportamiento de DSR. Esto se lo consiguió haciendo que entre dos caminos de igual reputación se seleccione el más corto, para ello previamente habíamos colocando las reputaciones de todos los nodos a 1, con lo cual no tomaba en cuenta la reputación sino simplemente buscaba el camino más corto directamente.

Para comprender porqué habilitamos estos cuatro selectores con las variantes de REMODE, es conveniente que recordemos el funcionamiento básico de una manera resumida. Cuando un nodo fuente requiere enviar un paquete la primera acción que realiza, es buscar si existe algún camino disponible en su cache, si es así va directamente al buffer de envío e inmediatamente envía el paquete, caso contrario se debe ejecutar el famoso descubrimiento de camino. En el descubrimiento de camino con ayuda de paquetes de petición de camino y paquetes de contestación de camino, se consigue tener disponible en la caché del nodo fuente todos los caminos disponibles de determinado origen a determinado destino. Una vez que tenemos los caminos disponibles en cache, en lugar de seleccionar el camino más corto como lo haría DSR, seleccionamos el de mejor calificación (mejor reputación), y entre dos caminos de igual calificación se selecciona el más corto. Calificación en REMODE_sw hace referencia al producto de las reputaciones de los nodos intermedios que forman el camino. Entonces, los selectores 4 , 6 y 12 que se habilitan en REMODE_sw, son para que se hagan cargo del mantenimiento de camino, y el selector 9 permite capturar el tráfico de la red, con lo cual el nodo estará receptivo a los paquetes de contestación de camino.

En el código fuente de DSR, es posible habilitar o deshabilitar las líneas de código que realizan la búsqueda en memoria cache secundaria. REMODE_sw, trabajan únicamente con caché primaria, aunque se realizó algunos experimentos en escenario estático (como si se tratara de una red mesh), y determinamos, que en escenarios estáticos era factible habilitar la caché secundaria en REMODE_sw.

En DSR, un nodo que reenvía paquetes captura los caminos de dichos paquetes, y esa información va a caché secundaria, en cambio en caché primaria están los caminos activos que son caminos que se seleccionan desde el nodo fuente. En escenarios móviles, que son los escenarios propios de las redes móviles ad-hoc, al trabajar

únicamente con caché primaria, se debería hacer demasiados descubrimientos de camino, porque el protocolo olvidaría mucha información valiosa. En cambio, al habilitar la caché secundaria, se corre el riesgo de que la mayoría de los caminos disponibles sean caminos más cortos, y no de mejor calificación (mejor reputación), lo cual interferiría con el funcionamiento normal de los algoritmos de las variantes de REMODE.

¿Por qué DSR con estado de flujo, requiere que estos doce selectores estén habilitados?

Los selectores 1, 2 y 3 se habilitan para trabajar con caminos implícitos, es decir en lugar de tener encaminamiento desde la fuente, se tiene encaminamiento hop by hop similar a AODV, lo que lo convierte a DSR (con estado de flujo) en un protocolo complejo, pero más tolerante a fallos. Cuando se tiene caminos implícitos cada paquete de datos solamente lleva dirección origen y dirección destino, y como previamente el camino ha sido instalado, los nodos intermedios utilizan su tabla de flujo, y realizan los reenvíos respectivos. Este mecanismo llamado "extensión de estado de flujo DSR", es una mejora de DSR que viene por defecto en NS-2.35, cuando deshabilitamos el selector 1, los selectores 2 y 3 se deshabilitan automáticamente. Propiamente, el selector 1 habilita el estado de flujo, el selector 2 el flujo por defecto en caminos implícitos, el selector 3 la preferencia de caminos más cortos sobre caminos implícitos por defecto. En los caminos implícitos en la cabecera del paquete de datos DSR únicamente viaja dirección origen y destino, en caminos explícitos se tiene las direcciones de todos los nodos que forman el camino. Cuando utilizamos caminos implícitos desde el nodo fuente, el nodo selecciona el camino, instala el camino en los nodos intermedios, y hasta se crean caminos implícitos por defecto, lo cual de cierta forma le hace perder su característica de encaminamiento desde la fuente (ya que sólo realiza un encaminamiento desde la fuente, mientras instala el camino en los nodos intermedios) y pasa a ser un encaminamiento salto a salto el cual reduce el encabezado de los paquetes de datos.

Los selectores 4, 6, 8 y 12 se encargan del mantenimiento de camino en DSR, gestionan errores y salvan paquetes.

Los selectores 5 y 9 alimentan la caché secundaria de los nodos de la red, es decir se

encargan de aprender caminos de los nodos vecinos.

El selector 7 habilita el acortamiento de caminos. Este selector que se habilita en DSR, es algo no deseable en variantes de REMODE, porque interferiría con el funcionamiento de REMODE, ya que los caminos de mejor calificación, no siempre son los más cortos.

Los selectores 10 y 11 habilitan la respuesta desde caché. El selector 10 permite contestar una petición de camino desde memoria caché de algún nodo intermedio, ya que existen nodos intermedios que pueden conocer cómo llegar a determinado nodo destino. El selector 11 hace que no se propaguen peticiones de camino como primera acción, al buscar un camino para un paquete.

Adicional a los doce selectores existen dos más, que no se los ha considerado porque siempre deberían quedarse con su configuración por defecto en los experimentos de esta tesis. El primer selector dice: "contestar_sólo_primera_petición_de_camino=falso" esta es su configuración por defecto, porque en realidad múltiples nodos en la red móvil ad-hoc, requieren que determinado nodo, conteste sus peticiones. El segundo selector dice "caminos_bi_direccionales=verdadero" ya que tanto REMODE_sw como DSR requieren intercambio bidireccional de paquetes de peticiones para envío (RTS) y paquetes borrar para enviar (CTS), así como también el acuse de recibo (ACK) de capa de enlace en IEEE 802.11.

En REMODE_sw, experimentalmente hemos verificado en NS-2.35 que únicamente habilitando los cuatro selectores mostrados en color azul al inicio de la sección, y únicamente utilizando cache primaria, se consigue un buen rendimiento en escenarios móviles. Al habilitar algún otro selector el rendimiento de las variantes de REMODE baja, no solo en cuanto a consumo de energía, sino que también aumenta la cantidad de paquetes perdidos.

B.6 Configuraciones de capa física.

A continuación mostramos una función llamada Cal_Pt { d } en OTCL, la cual fue desarrollada en esta tesis con la finalidad de facilitar el cálculo de la potencia de transmisión en función de la distancia, posteriormente verificamos el correcto

funcionamiento de la función mediante cálculos matemáticos.

```
# Calculation of transmission power as a function of distance
proc Cal_Pt { d } {
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 L [Phy/WirelessPhy set L_]
set RXThresh [Phy/WirelessPhy set RXThresh_]
set Frecuency [Phy/WirelessPhy set freq_ ]
set pi [expr (3.14159265359)]
set Lambda [expr (3e+8)/$Frecuency]
set Cross_dist [expr (4*$pi*$ht*$hr)/$Lambda]
 if {$d < $Cross_dist} {
                set n [expr $RXThresh*$L*pow(4*$pi*$d,2) ]
                set d [expr $Gt*$Gr*$Lambda*$Lambda]
 set Pt [expr ($n)/($d)]
 } else {
                set n [expr ($RXThresh*pow($d,4)*$L)]
                set d [expr ($Gt*$Gr*$ht*$ht*$hr*$hr)]
                set Pt [expr ($n)/($d)]
 }
puts " n= val($n)"
puts " d= val($d)"
puts " Pt= val($Pt)"
puts " Cross_dist= val($Cross_dist)"
puts " Lambda= val($Lambda)"
return $Pt
}
```

La tarjeta de red inalámbrica que hemos seleccionado para verificar el correcto funcionamiento de la función es el Adaptador Mini-PCI Atheros 400mW 2.4 GHz (Ubiquiti Networks SR2) cuya sensibilidad del receptor es de -74dBm para poder

transmitir a 54Mbps (teóricos), entonces convirtiendo este valor a vatios tendríamos un rxThresh_ de 3.9810717055349695e-11. Ahora podemos utilizar el simulador y conocer cuánto es la Pt_ para una distancia de 35 metros entre tarjetas de red, considerando además una frecuencia de $2{,}472x10^{9}$Hz, una Gt=Gr=1, y una ht=hr=1,5m.

La potencia calculada experimentalmente por la función Cal_Pt { d } que se ha mostrado previamente es de $5{,}2289\ x10^{-4}$W .

Ahora realizamos los cálculos matemáticos para verificar el resultado del experimento:

$$Cross\_dist = \frac{4 * \pi * ht * hr}{\lambda}$$

$$Cross\_dist = \frac{4 * \pi * (1{,}5) * (1{,}5)}{\frac{3x10^{8}}{2{,}472x10^{9}}}$$

$$Cross\_dist = 232{,}9805$$

El Cross_dist es igual a 232,9805m, entonces comparamos si d<Cross_dist, es decir si 35m<232,9805, como la respuesta es afirmativa tomamos la ecuación de Friis que toma en cuenta la frecuencia y calculamos la potencia de transmisión.

$$Pr = \frac{Pt * Gt * Gr * \lambda^2}{L * (4\Pi)^2 * d^2}$$

Despejando Pt, tenemos:

$$Pt = \frac{Pr * L * (4 * \pi * d)^2}{Gt * Gr * \lambda^2}$$

$$Pt = \frac{(3{,}9810717055349695x10^{-11}) * 1 * (4 * \pi * 35)^2}{1 * 1 * \left(\frac{3 * 10^{8}}{2{,}472x10^{9}}\right)^2}$$

$$Pt = \frac{7{,}70115415395x10^{-6}}{1{,}47280610802x10^{-2}}$$

$$Pt = 5{,}2289x10^{-4}\ W$$

Vale indicar que la potencia de transmisión del Adaptador Mini-PCI Atheros 400mW 2,4 GHz es de 0,1259 W para conseguir una tasa de transmisión de 54Mbps teóricos, por tanto garantizamos que la potencia mínima requerida para conseguir alcanzar la distancia "d" se cumple. El objetivo de restringir la potencia, es porque se puede conseguir con precisión caminos deseados en el diseño de la red móvil ad-hoc, y principalmente porque se garantiza un correcto radioenlace entre transmisor y receptor.

Ahora presentaremos un nuevo experimento para verificar que funciona correctamente la selección automática de la ecuación del modelo de dos rayos con el objetivo de calcular la Pt. Conservamos los mismos datos del experimento anterior, esta vez sólo cambiaremos la distancia (entre transmisor y receptor) a 233m. La potencia calculada experimentalmente por la función Cal_Pt { d } es de 0.0232W, a continuación verificamos el resultado mediante cálculos matemáticos.

Como d $\prec$ Cross_dist, es decir como 233m $\prec$ 232,9805m, entonces utilizamos la ecuación del modelo de dos rayos en la cual no se toma en cuenta la frecuencia.

$$Pr = \frac{Pt * Gt * Gr * ht^2 * hr^2}{d^4 * L}$$

$$Pt = \frac{Pr * d^4 * L}{Gt * Gr * ht^2 * hr^2}$$

$$Pt = \frac{(3{,}9810717055349695\text{x}10^{-11}) * 233^4 * 1}{1 * 1 * (1{,}5^2 * 1{,}5^2)}$$

$$Pt = \frac{0{,}117333948065}{5{,}0625}$$

$$Pt = 0{,}0232\ W$$

Las configuraciones de cápa física que hemos establecido en el script OTCL son las siguientes:

Assigning basic parameters of antennas for nodes.
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

Initialization of the parameters for the nodes' radio interface

Module used for design: Ubiquiti Adaptador miniPCI Atheros 400mW 2.4 GHz

Phy/WirelessPhy set RXThresh_ 3.9810717055349695e-11

Phy/WirelessPhy set bandwidth_ 20e6

Phy/WirelessPhy set freq_ 2.472e+9

Phy/WirelessPhy set L_ 1.0

Phy/WirelessPhy set Pt_ [Cal_Pt $val(Coverage)]

Lo que indica que la altura, ganancias y pérdidas tienen un valor de 1, la sensibilidad de recepción (RXThresh_) corresponde a la tarjeta de red inalámbrica real antes mencionada, las frecuencia corresponde al canal 13 de wifi, un canal usado en España, valor real de ancho de banda para 802.11g es de 20MHz y finalmente Pt_ se configura previo cálculo mediante la función Cal_Pt { d } programada en OTCL.

Luego de la explicación realizada, nos damos cuenta que la función Cal_Pt { d } facilita colocar las configuraciones de capa física en la realización de nuevos diseños de red considerando tanto la ecuación de Friis como el modelo de dos rayos.

B.7 Creación y ejecución de un script OTCL (*Object Oriented Tool Command Language*) con selectores adicionales para las configuraciones de REMODE_sw (*Reputation for Mobile Devices Static Weight*).

Los selectores adicionales permiten variar con precisión los parámetros con los que funciona REMODE_sw. A continuación se presenta un ejemplo de un script OTCL estableciendo los selectores (en color azul) adecuados para una simulación en particular. En el script OTCL, se referencia el escenario de movilidad 10mn_1p5mps_400s_r1_ready.ns_movements (creado en el Anexo B.3) con la siguiente línea de código:

set val(sc) "/usr/src/bonnmotion-3.0.0/bin/10mn_1p5mps_400s_r1_ready.ns_movements"

Luego de esta breve explicación, se presenta el ejemplo de uno de los tantos scripts OTCL que fueron creados para desarrollar esta tesis. Al escoger el protocolo de encaminamiento mediante “remode_selector”, aunque existen múltiples opciones en

realidad las opciones que se implementaron exitosamente son los selectores 1 y 4 que permiten hacer la comparación entre REMODE_sw y DSR

```
# mobile_reputation_tcp_vegas_1p5_mps_300x300_exp2.tcl

# My Simulation Using NS-2.35
#=============================================================
=========
# Lenin Guaya Delgado student at the Technical University of Catalonia.
# REMODE: Reputation for Mobile Devices.
# Network layer protocol that selects the route most reputable.
# FPLpi: For each path, one sample every n packets.
# RFVpi=(1-alfa)*RFVpi+alfa*(1-FPLpi).
# RFVnj=(1-beta)*RFVnj+beta*pow(RFVpi,(1/n))
# SCOREpi= Pi j e path i(RFVnj)
# ENTEL-BCN-Spain->2017.
#
=============================================================
=======
set ini [clock seconds]
#
=============================================================
=======
# REMODE Selectors
#
=============================================================
=======
# 0: DSR(complete), 1:REMODE_static_weight, 2:REMODE_dynamic_weight_1,
3:REMODE_dynamic_weight_2, 4:DSR(REMODE imitates DSR)
Agent/DSRAgent set remode_selector 1
# FPL from: 0:Trace, 1:RTCP, 2:TCP-Linux, 3:TCP_Vegas
Agent/DSRAgent set transport_selector 3
# 30: Normal primary size
Agent/DSRAgent set primary_size 30
```

```
# 64: Normal secondary size
Agent/DSRAgent set secondary_size 64
# 0...1: Weight for new reputations of paths
Agent/DSRAgent set alpha 1
# 0...1: Calculating beta
Agent/DSRAgent set beta 0.6
# 1...n (packets): Sampling interval
Agent/DSRAgent set sampling_interval 10
# 1...n (paths): Routes between nodes.
Agent/DSRAgent set paths 2
# 1...n (nn nodes): scenario's nodes.
Agent/DSRAgent set nodes 10
# Analizing source node
Agent/DSRAgent set source_node 0
# Analizing path of the source node
Agent/DSRAgent set path_s_n 1
# Dynamic reputation: analyzing node m
Agent/DSRAgent set m_r 0
# Dynamic reputation: analyzing node n
Agent/DSRAgent set n_c 7
# Packet size
Agent/DSRAgent set packet_size 512
# Seed: 7, 14, 21, 28, 35, 42, 49, 56, 63, 70,->77, 84, 91, 98, 105, 112, 119, 126, 133,
140
#Selected seeds: 7(r1),14(r2)
Agent/DSRAgent set seed 7

# Additional information for verifying results with trace information
# FPL from: 0:Trace, 1:RTCP, 2:TCP-Linux, 3:TCP_Vegas
set env(transport_selector_2) 3
# 1...n (paths): Routes between nodes.
set env(paths_2) 2
```

```
# 1...n (nn nodes): Scenario's nodes.
set env(nodes_2) 10
# 1...n (packets): Sampling interval
set env(sampling_interval_2) 10
# 0...1: Weight for new reputations of paths
set env(alpha_2) 1
# 0...1: Weight for new reputations of nodes
set env(beta_2) 0.6
# Analizing source node
set env(source_node_2) 0
# Analizing path of the source node
set env(path_s_n_2) 1
# Packet size (bytes)
set env(pck_size_2) 512
# Ignore initial time, integer variable (n seconds)
set env(ignore_initial_time_2) 100
# Initial energy for each node, float variable (Joules)
# Samsung Galaxy S6 Edge. Accumulated Energy=3.8Ahx3600sx3.85Voltios
set env(initial_energy_2) 52668
# Sampling interval results (n seconds)
set env(sampling_interval_results_2) 2

#
=============================================================
=======
# Definition of the node's configuration.
#
=============================================================
=======
set val(chan)       Channel/WirelessChannel  ;# Channel type
set val(prop)       Propagation/TwoRayGround ;# Radio-propagation model
set val(netif)      Phy/WirelessPhy          ;# Network interface
```

```
set val(mac)        Mac/802_11             ;# MAC type
set val(ifq)        CMUPriQueue               ;# Interface queue type
set val(ll)         LL                        ;# Link layer type
set val(ant)        Antenna/OmniAntenna       ;# Antena model
set val(ifqlen)         50  ;# Maximum packets in ifq (e.g. 50)
set val(nn)             10  ;# Number of mobile nodes
set       val(sc)                                                    "/usr/src/bonnmotion-
3.0.0/bin/10mn_1p5mps_400s_r1_ready.ns_movements"
set val(adhocRouting)   DSR ;# Routing protocol
set val(tcp_timestamp)  true;# Activated timestamp
set val(tcp_window)     20;# TCP window size
set val(cbr_interval)   0.008192;# CBR interval
set val(cbr_rate)       500Kb;# CBR rate
set val(pck)            512     ;# Packet size
set val(x)              310     ;# X dimension of the topography
set val(y)              310     ;# Y dimension of the topography
set val(stop)           400     ;# Simulation time (seconds)
set val(Coverage)       220.0   ;# Each node coverage.

# Assigning basic parameters of antennas for nodes.
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

# Calculation of transmission power as a function of distance
proc Cal_Pt { d } {
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 L [Phy/WirelessPhy set L_]
set RXThresh [Phy/WirelessPhy set RXThresh_]
set Frecuency [Phy/WirelessPhy set freq_ ]
set pi [expr (3.14159265359)]
set Lambda [expr (3e+8)/$Frecuency]
set Cross_dist [expr (4*$pi*$ht*$hr)/$Lambda]
  if {$d < $Cross_dist} {
                    set n [expr $RXThresh*$L*pow(4*$pi*$d,2) ]
                    set d [expr $Gt*$Gr*$Lambda*$Lambda]
  set Pt [expr ($n)/($d)]
  } else {
                    set n [expr ($RXThresh*pow($d,4)*$L)]
                    set d [expr ($Gt*$Gr*$ht*$ht*$hr*$hr)]
                    set Pt [expr ($n)/($d)]
  }
puts " n= val($n)"
puts " d= val($d)"
puts " Pt= val($Pt)"
puts " Cross_dist= val($Cross_dist)"
puts " Lambda= val($Lambda)"
return $Pt
}

# Initialization of the parameters for the nodes' radio interface
# Module used for design: Ubiquiti Adaptador miniPCI Atheros 400mW 2.4 GHz
Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
Phy/WirelessPhy set RXThresh_ 3.9810717055349695e-11
Phy/WirelessPhy set bandwidth_ 20e6
Phy/WirelessPhy set freq_ 2.472e+9
Phy/WirelessPhy set L_ 1.0
Phy/WirelessPhy set Pt_ [Cal_Pt $val(Coverage)]
```

```
# Assign the transmitted power from the previously defined function
Mac/802_11 set CWMin_ 31
Mac/802_11 set CWMax_ 1023
Mac/802_11 set SlotTime_ 0.000020;# 20us
Mac/802_11 set SIFS_ 0.000010;# 10us
Mac/802_11 set dataRate_ 54.0e6 ;# Setting MAC IEEE 802.11g.
Mac/802_11 set basicRate_ 6.0e6
Mac/802_11 set PLCPDataRate_ 6.0e6
Mac/802_11 set RTSThreshold_ 3000
Mac/802_11 set PreambleLength_ 144
Mac/802_11 set PLCPHeaderLength_ 48
Agent/TCP set packetSize_ 512;# TCP packet size

#
=============================================================
=======
# Main program
#
=============================================================
=======
# Initialization of global variables
# Initialize simulator
set ns_         [new Simulator]
$ns_ use-newtrace ;# New trace format is used

# Initialize trace file
set tracefd     [open mobile_reputation_tcp_vegas_1p5_mps-out.tr w]
$ns_ trace-all $tracefd

# Initialize Network Animator
set namtrace    [open mobile_reputation_tcp_vegas_1p5_mps.nam w]
```

```
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

# Set up topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)

# Create General Operations Director (GOD) object. It is used to store global information about the state of the environment, network, or nodes
# that an
# omniscent observer would have, but that should not be made know to any
# participant in the simulation
set god_ [create-god $val(nn)]

# Creating and configuring the nodes with the parameters previously determined
$ns_ node-config -adhocRouting $val(adhocRouting) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -channelType $val(chan) \

                                  -topoInstance $topo \

                                  -agentTrace ON \
                 -routerTrace ON \
                 -macTrace ON \

                                  -movementTrace OFF \
```

```
                                -energyModel "EnergyModel" \

                                -rxPower 1.181 \

                                -txPower 1.258 \

                                        -idlePower 0.884 \
                                        -sleepPower 0.042 \
                                        -transitionTime 0.005 \

                                -initialEnergy 52668

$ns_ set WirelessNewTrace_ ON

#  Create the specific number of nodes
for {set i 0} {$i < $val(nn) } {incr i} {
        set node_($i) [$ns_ node]
        $node_($i) random-motion 0  ;# disable random movement
}

# Define color index
$ns_ color 0 blue
$ns_ color 1 red

# Movement pattern
puts "Loading scenario file..."
source $val(sc)

# Selfish nodes labeling
$ns_ at  0.1 "$node_(1) label \"FPLn1=0.6\""
$node_(1) color red
```

```
$ns_ at  0.1 "$node_(1) color red"
$ns_ at  0.1 "$node_(2) label \"FPLn2=0.6\""
$node_(2) color red
$ns_ at  0.1 "$node_(2) color red"
$ns_ at  0.1 "$node_(3) label \"FPLn3=0.6\""
$node_(3) color red
$ns_ at  0.1 "$node_(3) color red"
# Source nodes labeling
$ns_ at  0.1 "$node_(0) label \"Tx_1,2\""
$ns_ at  0.1 "$node_(9) label \"Rx_1\""
$ns_ at  0.1 "$node_(8) label \"Rx_2\""

# Defines the initial position of the node in NAM
for {set i 0} {$i < $val(nn)} {incr i} {
   # 20 defines the size of the node in nam, you must adjust this according to your scenario
   # The function should be called after that the mobility model is defined
   $ns_ initial_node_pos $node_($i) 10
}

#
=============================================================
=======
# Setup traffic flow between nodes
#
=============================================================
=======
# Create 1 connections: 0->9
# TCP conection  between node_(0) and node_(9)
# Create a TCP agent and attach this to the node_(0)
set tcp1 [new Agent/TCP/Vegas]
$tcp1 set timestamps_ $val(tcp_timestamp)
```

```
$tcp1 set window_ $val(tcp_window)
$ns_ attach-agent $node_(0) $tcp1
# Create a cbr1 traffic source and attach this to tcp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ $val(pck)
$cbr1 set interval_ $val(cbr_interval)
$cbr1 set rate_ $val(cbr_rate)
$cbr1 attach-agent $tcp1
# Create a null1 agent (a destination) and attach this to the node_(9)
set null1 [new Agent/TCPSink]
$ns_ attach-agent $node_(9) $null1
# Connect the source agent with destination agent
$ns_ connect $tcp1 $null1

# Create 1 connections: 0->8
# TCP conection  between node_(0) and node_(8)
# Create a TCP agent and attach this to the node_(0)
set tcp2 [new Agent/TCP/Vegas]
$tcp2 set timestamps_ $val(tcp_timestamp)
$tcp2 set window_ $val(tcp_window)
$ns_ attach-agent $node_(0) $tcp2
# Create a cbr2 traffic source and attach this to tcp2
set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ $val(pck)
$cbr2 set interval_ $val(cbr_interval)
$cbr2 set rate_ $val(cbr_rate)
$cbr2 attach-agent $tcp2
# Create a null2 agent (a destination) and attach this to the node_(8)
set null2 [new Agent/TCPSink]
$ns_ attach-agent $node_(8) $null2
# Connect the source agent with destination agent
$ns_ connect $tcp2 $null2
```

```
#
# PLANNING EVENTS
#
# The traffic source are started
$ns_ at  0.0 "$cbr1 start"
$ns_ at  0.0 "$cbr2 start"

# The traffic source are finalized
$ns_ at  400.0 "$cbr1 stop"
$ns_ at  400.0 "$cbr2 stop"

# Indicating to the nodes the simulation finalization
for {set i 0} {$i < $val(nn) } {incr i} {
$ns_ at $val(stop) "$node_($i) reset";
}

# Ending the simulation
$ns_ at $val(stop) "stop"
$ns_ at $val(stop).01 "puts \"Ending the simulation\" ; $ns_ halt"

#
# Obtain CWND from TCP agent
#
```

```
proc plotWindow {tcpSource outfile1} {
  global ns_

  set now [$ns_ now]
  set cwnd [$tcpSource set cwnd_]

# Print TIME CWND   for  gnuplot to plot progressing on CWND
  puts  $outfile1  "$now $cwnd"

  $ns_ at [expr $now+0.1] "plotWindow $tcpSource  $outfile1"
}
proc plotWindow {tcpSource outfile2} {
  global ns_

  set now [$ns_ now]
  set cwnd [$tcpSource set cwnd_]

# Print TIME CWND   for  gnuplot to plot progressing on CWND
  puts  $outfile2  "$now $cwnd"

  $ns_ at [expr $now+0.1] "plotWindow $tcpSource  $outfile2"
}

set outfile1 [open  "cwnd_p1"  w]
set outfile2 [open  "cwnd_p2"  w]

$ns_ at  0.0 "plotWindow $tcp1  $outfile1"
$ns_ at  0.0 "plotWindow $tcp2  $outfile2"

proc stop {} {
global ns_ tracefd namtrace
```

```
$ns_ flush-trace
close $tracefd
close $namtrace
exec chmod 777 cwnd_p1 cwnd_p2 &
exec xgraph cwnd_p1 cwnd_p2 -geometry 800x400 &
exec nam mobile_reputation_tcp_vegas_1p5_mps.nam &
exit 0
}

puts "Simulation with REMODE protocol"
puts "nn $val(nn) x $val(x) y $val(y) rp with losses $val(adhocRouting)"
puts "ifq $val(ifqlen) coverage $val(Coverage)"
puts "prop $val(prop) ant $val(ant)"
$ns_ run
puts "   "
puts "   "
puts "   "
set fini [clock seconds]
set tiempo [expr $fini - $ini]
puts "tiempo: $tiempo segundos"
```

Para conseguir las pérdidas de los nodos 2 y 3 (FPLn2=FPLn3=0.6), lo hacemos como se explicó en el Anexo B.4. Luego de presentado el ejemplo procedemos a analizar los selectores del script OTCL. Primeramente nos fijamos en que, en el script OTCL existen dos secciones de selectores, los primeros que van bajo "#REMODE Selectors" y que corresponden a configuraciones internas a cada nodo, y los segundos que van bajo "# Additional information for verifying results with trace information", y que corresponden a configuraciones requeridas por Trace Analizer v3.0 para analizar resultados. A continuación se explicará un poco más acerca de cada selector que se requiere configurar para que funcionen adecuadamente las simulaciones.

==========

```
# REMODE Selectors
#
==========================================================
=======
# 0: DSR(complete), 1:REMODE_static_weight, 2:REMODE_dynamic_weight_1,
3:REMODE_dynamic_weight_2, 4:DSR(REMODE imitates DSR)
Agent/DSRAgent set remode_selector 1
```

Al configurar "remode_selector 1", indicamos al simulador que elegimos utilizar la variante estática de REMODE. Si elegimos 0, utilizaremos la versión de DSR con estado de flujo, si elegimos 2 utilizaremos la variante dinámica algoritmo 1 de REMODE, si elegimos 3, utilizaremos la variante dinámica algoritmo 2 de REMODE, y finalmente si elegimos 4, utilizaremos DSR sin estado de flujo.

Únicamente si elegimos 0, trabajamos con caché secundaria, y con los 12 selectores de dsragent.cc habilitados. Al elegir 1, 2, 3 4 trabajamos únicamente con cache primaria y con 4 selectores de dsragent.cc habilitados, como se lo explicó en el Anexo B.5.

```
# FPL from: 0:Trace, 1:RTCP, 2:TCP-Linux, 3:TCP_Vegas
Agent/DSRAgent set transport_selector 3
```

Al configurar "transport_selector 3", utilizamos TCP-Vegas como protocolo de capa de transporte, es decir el cruce de capa lo realizaremos desde TCP-Vegas. Si elegimos 0, el cruce de capa lo realizaremos desde el archivo de trazas, si elegimos 1, el cruce de capa lo realizaremos desde RTP / RTCP, si elegimos 2, el cruce de capa lo realizaremos desde TCP-Linux. Si utilizamos TCP-Linux, debemos especificar dentro del desarrollo del script OTCL cuál de las catorce versiones disponibles [**104**] requerimos utilizar, además en este entorno de desarrollo (TCP-Linux), es posible crear nuevas versiones de TCP modificando el control de congestión.

```
# 30: Normal primary size
Agent/DSRAgent set primary_size 30
```

Al configurar "primary_size 30", establecemos que la caché primaria es de 30 caminos.

```
# 64: Normal secondary size
Agent/DSRAgent set secondary_size 64
```

Al configurar "secondary_size 64", establecemos que la caché secundaria es de 64 caminos, esto al utilizar DSR, ya que DSR sí utiliza caché secundaria. Si se configura

algún tamaño de caché secundaria al utilizar alguna variante de REMODE, simplemente no se tomará en cuenta este parámetro.

0...1: Weight for new reputations of paths

Agent/DSRAgent set alpha 1

Al configurar “alpha 1”, utilizaremos únicamente los valores nuevos de reputación del camino.

0...1: Calculating beta

Agent/DSRAgent set beta 0.6

Al configurar “beta 0.6”, el valor de 0.6 lo estableceremos como peso para los valores nuevos de reputación del nodo.

1...n (packets): Sampling interval

Agent/DSRAgent set sampling_interval 10

Al configurar “sampling_interval 10”, el muestro para las variantes de REMODE lo realizaremos cada 10 paquetes.

1...n (paths): Routes between nodes.

Agent/DSRAgent set paths 2

Al configurar “paths 2” , especificamos que la simulación tiene 2 caminos.

1...n (nn nodes): scenario's nodes.

Agent/DSRAgent set nodes 10

Al configurar “nodes 10”, especificamos que la simulación tiene 10 nodos.

Analizing source node

Agent/DSRAgent set source_node 0

Al configurar “source_node 0”, especificamos que queremos analizar el nodo fuente 0.

Analizing path of the source node

Agent/DSRAgent set path_s_n 1

Al configurar “path_s_n 1”, especificamos que de entre los caminos que ha establecido el nodo fuente nombrado en la configuración anterior, analizaremos el camino 1.

Dynamic reputation: analyzing node m

Agent/DSRAgent set m_r 0

Dynamic reputation: analyzing node n

Agent/DSRAgent set n_c 7

Al configurar “m_r 0” y “n_c 7”, especificamos la posición m, n de una matríz de dos dimensiones. Dicho de otra manera para el ejemplo puntual, estamos observando la información que el nodo 0 posee del nodo 7, es decir observamos la fila 0, columna 7. La información de esta matríz se utiliza únicamente cuando elegimos trabajar con variantes dinámicas de REMODE en donde el valor de beta es variable.

```
# Packet size
Agent/DSRAgent set packet_size 512
```

Al configurar “packet_size 512”, especificamos que el tamaño del paquete de datos es 512 Bytes. Este valor, lo utilizamos para diferenciar paquetes de datos de paquetes de control.

```
# Seed: 7, 14, 21, 28, 35, 42, 49, 56, 63, 70,->77, 84, 91, 98, 105, 112, 119, 126, 133, 140
#Selected seeds: 7(r1),14(r2)
Agent/DSRAgent set seed 7
```

Al configurar “seed 7”, especificamos que 7 es la semilla que utilizamos en el programa de generación de pérdidas aleatorias, debido a que para obtener los intervalos de confianza, se requiere repetir el experimento varias veces cambiando la semilla.

```
# Additional information for verifying results with trace information
```

La mayoría de la información de los selectores que viene a continuación se repite, esto es debido a que el programa analizador de trazas Trace Analizer v3.0, requiere conocer muchos parámetros para hacer adecuadamente el filtrado, esto es debido a la infinidad de formas en las que podemos filtrar los datos.

```
# FPL from: 0:Trace, 1:RTCP, 2:TCP-Linux, 3:TCP_Vegas
set env(transport_selector_2) 3
```

Al configurar “env(transport_selector_2) 3”, especificamos que utilizamos TCP-Vegas.

```
# 1...n (paths): Routes between nodes.
set env(paths_2) 2
```

Al configurar “env(paths_2) 2”, especificamos que utilizamos 2 caminos en la simulación.

```
# 1...n (nn nodes): Scenario's nodes.
set env(nodes_2) 10
```

Al configurar “env(nodes_2) 10”, especificamos que utilizamos 10 nodos en la simulación.

1...n (packets): Sampling interval

set env(sampling_interval_2) 10

Al configurar “env(sampling_interval_2) 10”, especificamos que el muestreo lo realizaremos cada 10 paquetes.

0...1: Weight for new reputations of paths

set env(alpha_2) 1

Al configurar “env(alpha_2) 1”, especificamos que el peso para los valores nuevos de reputación de camino es de 1.

0...1: Weight for new reputations of nodes

set env(beta_2) 0.6

Al configurar “env(beta_2) 0.6”, especificamos que el peso para los valores nuevos de reputación de nodo es de 0.6.

Analizing source node

set env(source_node_2) 0

Al configurar “env(source_node_2) 0”, especificamos que analizamos el nodo 0.

Analizing path of the source node

set env(path_s_n_2) 1

Al configurar “env(path_s_n_2) 1”, especificamos que de entre los caminos establecidos por el nodo fuente nombrado en la configuración anterior, analizamos el camino 1.

Packet size (bytes)

set env(pck_size_2) 512

Al configurar “env(pck_size_2) 512”, especificamos que el tamaño de paquete de datos es de 512 Bytes.

Ignore initial time, integer variable (n seconds)

set env(ignore_initial_time_2) 100

Al configurar “env(ignore_initial_time_2) 100”, especificamos que para la toma de datos para resultados (gráficas), ignoramos los 100 segundos iniciales.

Initial energy for each node, float variable (Joules)

Samsung Galaxy S6 Edge. Accumulated Energy=3.8Ahx3600sx3.85Voltios

set env(initial_energy_2) 52668

Al configurar “env(initial_energy_2) 52668”, especificamos que la energía inicial de cada nodo del escenario es de 52668 Julios. Este parámetro no es objeto de estudio de esta tesis, pero como trabajo futuro resulta interesante optimizar el consumo de energía de los nodos de la red móvil ad-hoc, porque la cantidad de energía en el nodo es un recurso limitado. La clave estaría en reducir el número de paquetes de control que se transmiten, ya que la acción del nodo que más consume energía es la transmisión de paquetes.

Sampling interval results (n seconds)

set env(sampling_interval_results_2) 2

Al configurar “env(sampling_interval_results_2) 2”, especificamos que el intervalo de muestreo para impresión de resultados (datos para gráficas de resultados) es cada 2 segundos.

La información de los selectores en su mayoría se repite dos veces, esto es debido a que son dos implementaciones las que se han agregado al simulador NS2.35, por un lado la implementación de las variantes de REMODE con su respectivo cruce de capa; y, por otro lado la implementación de Trace Analizer v3.0, el cual filtra la información que se va imprimiendo en traza, en tiempo de ejecución.

B.8 Entendiendo los archivos resultantes de la ejecución de la simulación

Normalmente el archivo de trazas es el único resultado de la simulación, pero es difícil entenderlo por la cantidad de datos que tiene, en la presente tesis imprimimos en archivos de texto datos relevantes para el análisis del rendimiento de REMODE_sw. Además al integrar Trace Analyzer v3.0 a NS-2.35, podemos obtener información adicional del funcionamiento del protocolo, y sobre todo, las métricas de calidad de servicio (QoS). Entonces los archivos se imprimen desde dos lugares, por un lado desde dentro del protocolo (inside), que son cálculos internos; y, por otro imprimimos archivos de texto desde el archivo de impresión de trazas, que son los resultados propiamente dichos.

A continuación se presenta un ejemplo de simulación de fácil comprensión, se ejecuta el script OTCL mediante el siguiente comando:

ns mobile_reputation_tcp_vegas_1p5_mps_300x300_exp2.tcl

A continuación se explica el contenido de los archivos de texto que se imprimen tanto desde las variantes de REMODE (REMODE_v2.1_...), como desde el analizador de trazas (Trace_Analyzer_v3.0_...), que son desarrollos de software propios de esta tesis. La palabra "inside" en el nombre del archivo indica que es información procedente de los archivos de programa de las variantes de REMODE, la palabra "TCP" indica que es información procedente de los archivos de programa de TCP-Vegas; y, finalmente la palabra "TCP-Linux" indica que es información procedente de los archivos de programa de TCP-Linux. Los archivos de esta tesis, adicionales al archivo de traza (.tr) son los siguientes:

REMODE_v2.1_beta_comparison(inside).txt

Para un nodo en particular previamente indicado, en este archivo se imprimen cada uno de los valores de las variables que participan en las variantes dinámicas de REMODE.

REMODE_v2.1_cache(inside).txt

Aquí se imprime las reputaciones de los caminos que participan en la red móvil ad-hoc.

REMODE_v2.1_local_reputation(inside).txt

Para un nodo fuente y camino previamente indicados, imprime la reputación muestreada, reputaciones de caminos y nodos, así como la calificación y los nodos del camino que está siendo utilizado.

REMODE_v2.1_n_rows_cache_vegas(inside).txt

Aquí se imprime los sockets de los nodos origen y destino desde capa de red, que han sido abiertos para conformar cada uno de los caminos de la red móvil ad-hoc.

REMODE_v2.1_n_sockets_vegas.txt

Aquí se imprime los sockets de los nodos origen y destino desde capa de transporte, que han sido abiertos para conformar cada uno de los caminos de la red móvil ad-hoc.

REMODE_v2.1_selected_paths(inside).txt

Para un nodo fuente y camino previamente indicado, en este archivo se imprimen cada uno de los nodos que forman el camino, además la longitud y calificación del camino.

REMODE_v2.1_send_buffer(inside).txt

Para un nodo fuente previamente indicado, en este archivo se imprimen los caminos de los paquetes que esperan en el send buffer a ser enviados.

REMODE_v2.1_(TCP)_FPRtx_from_tcp_vegas_accumulated.txt

En este archivo se imprime la fracción de paquetes retransmitidos por cada camino, y el promedio de la fracción de paquetes retransmitidos de los caminos presentes en la red móvil ad-hoc. El número de paquetes retransmitidos y enviados que se considera, es un valor acumulado, es decir que no se reinicia cada cierto tiempo.

REMODE_v2.1_(TCP)_FPRtx_from_tcp_vegas_restarted.txt

En este archivo se imprime la fracción de paquetes retransmitidos por cada camino, y el promedio de la fracción de paquetes retransmitidos de los caminos presentes en la red móvil ad-hoc. El número de paquetes retransmitidos y enviados que se considera, es un valor que se reinicia cada 5 segundos.

REMODE_v2.1_(TCP)_information_from_tcp_vegas.txt

Por cada camino existente en la red móvil ad-hoc, en este archivo se imprime el número de eventos timeouts, acuses de recibo duplicados (dupacks), acuses de recibo (acks), retransmisiones, envíos por cada camino; y, el promedio de cada uno de los caminos que forman la red móvil ad-hoc. Además se imprime la fracción de paquetes retransmitidos total (acumulada), que es tomada como la fracción de pérdida de paquetes total.

REMODE_v2.1_(TCP-Linux)_information_from_tcp.txt

En este archivo se imprime el número de paquetes enviados por cada camino en cada segundo de la simulación.

Trace_Analyzer_v3.0_centralized_reputation.txt

Desde una visión centralizada (no distribuida), se imprime la siguiente información: reputación muestreada (que es 1-FPLpi) de cada camino, reputación (RFVpi) de cada

camino, reputación real de cada camino, calificación de cada camino, reputación estimada de cada nodo, reputación de cada nodo (RFVnj), reputación teórica de cada nodo, el primer nodo intermedio de cada camino, el número de nodos intermedios en cada camino, el tráfico total de un determinado nodo, el número de paquetes recibidos por cada camino, el número de paquetes enviados por cada camino, la fracción de pérdida de paquetes por cada camino, el número de paquetes recibidos total, el número de paquetes enviados total; y, la fracción de pérdida de paquetes total. Aunque todos los cálculos de reputación que utilizan las variantes de REMODE son de forma distribuída, este archivo sirve para entender de una forma sencilla todo lo que está sucediendo en el escenario.

Trace_Analyzer_v3.0_delay_and_hops_path.txt

En este archivo se imprime el retardo de cada camino, el número de saltos de cada camino, el promedio del retardo de todos los caminos, y el promedio del número de saltos de todos los caminos.

Trace_Analyzer_v3.0_FPLs_paths.txt

En este archivo se imprimen las fracciones de pérdida de paquetes de cada camino, y el promedio de las fracciones de pérdida de paquetes de cada camino que conforma la red móvil ad-hoc.

Trace_Analyzer_v3.0_local_reputation.txt

Para un nodo fuente y camino previamente indicados, en este archivo de una manera distribuída (no centralizada) se imprime la siguiente información: reputación del camino , reputaciones de los nodos, reputaciones teóricas de los nodos, calificación del camino, el número de paquetes perdidos, recibidos y transmitidos del camino; y, finalmente la fracción de pérdida de paquetes del camino.

Trace_Analyzer_v3.0_n_sockets_trace.txt

En este archivo se imprimen los sockets origen y destino de cada uno de los caminos que conforma la red móvil ad-hoc.

Trace_Analyzer_v3.0_path_each_node.txt

Para un nodo fuente y camino previamente indicados, en este archivo se imprime cada uno de los nodos que conforma el camino.

Trace_Analizer_v3.0_metrics.txt

Este es uno de los archivos principales que imprime Trace Analyzer v3.0, aquí se imprime las métricas de calidad de servicio (QoS) de toda la red móvil ad-hoc. La información que imprime es la siguiente: recibidos, transmitidos, fracción de pérdida de paquetes, retardo, número de saltos, energía, rendimiento y overhead.

C Notación y símbolos

En este Anexo C, mostramos la notación y símbología utilizada en el Capítulo 4 de la tesis.

C.1 Notación y símbolos del Capítulo 4

r_n	Reputación de nodo para el nodo n.
R_p	Reputación de camino para el camino p.
L_p	Probabilidad de pérdida de paquetes para el camino p.
$\widehat{r_n}$	Muestra de reputación estimada para el nodo n
N_p	Número de nodos intermedios en el camino p
r_k	k-ésima muestra de una reputación de nodo
$\bar{r}$	Valor medio de r_k
Δr_k	Desviación del valor medio de la muestra r_k
σ_r	Desviación estándar de r_k
c_k	Media móvil acumulada (CMA) de las muestras k de r_k
$\overline{c_k}$	Valor medio de c_k
Δc_k	Desviación del valor medio de c_k
σ_c	Desviación estándar de c_k
w_k	Media móvil exponencial (EWMA) de las muestras k de r_k
β	Peso para la siguiente muestra a promediar en la EWMA
w_0	Valor inicial para la EWMA
$\overline{w_k}$	Valor medio de w_k
Δw_k	Desviación desde el valor medio de w_k
σ_w	Desviación estándar de w_k
f_k	Primera diferencia de los valores w_k
d_k	Segunda diferencia de los valores w_k

Tabla C.1: Notación y símbolos del Capítulo 4

D Código fuente para NS-2.35

Al trabajar con NS-2.35, se obtiene los resultados de la simulación en trazas demasiado extensas, en archivos de varios gigabytes que se los analiza posterior a la simulación. El código que se presenta a continuación (al ser integrado en NS-2.35, específicamente sobre el archivo de impresión de trazas cmu-trace.cc), permite analizar las trazas en tiempo de ejecución de la simulación, y permite ir guardando y/o calculando sólo los datos que nos interesa analizar. En la sección D.1 se presenta el código de "Trace_Analizer_v3.0", el cual fue utilizado en múltiples simulaciones de esta tesis, ya que permite obtener las métricas de QoS de cada simulación, para posteriormente analizar esta información mediante un procesador de hojas de cálculo.

D.1 Analizador de traza sobre cmu-trace.cc en NS-2.35

```
--->
Trace_Analizer_v3.0
By: Lenin Guaya Delgado
Departament d'Enginyeria Telem√†tica-ENTEL-UPC-Barcelona Tech
Parallel to REMODE_v2.1 protocol runs another program called
"Trace_Analizer_v3.0" which is used to verify the correct
implementation of the protocol from the data that they are being
printed in the "trace file" (.tr) at runtime. "Trace_Analizer_v3.0"
allows to calculate theoretical and experimental values from a
local and centralized view with the aim of analyze the performance
of REMODE_v2.1 protocol. Furthermore, is possible to calculate QoS
metrics for a very detailed comparative analysis.
To include "Trace_Analizer_v3.0" in ns-2.35 replace the "cmu-
trace.cc" file. After compilation is possible to work with static
and mobile scenarios.
--->
*/
#include <stdio.h>//
#include <stdlib.h>//
#include <string.h>//
.
.
.
#include <dsr/hdr_sr.h>      // DSR
#include <dsr/dsragent.h>    // REMODE
.
.
.
   //Entering data
const int nodes=100; //0-99:100 nodes
const int paths_input=21;//0-21: 20 paths
const int paths=paths_input+1;//+1 because it is not initialized at
the zero position
int TRANSPORT_SELECTOR_OTCL=1;
double ALPHA_OTCL=1;
double BETA_OTCL=1;
int SOURCE_NODE_OTCL=0;
int PATH_S_N_OTCL=1;
int PCK_SIZE_OTCL=0;
int IGNORE_INITIAL_TIME_OTCL=0;
```

```
double INITIAL_ENERGY_OTCL=52668;//Joules
double RR_N[nodes];//Real (or assigned) reputation for nodes on
theoretical calculations
float gamma_path[paths];//Communication speed on every path
int socket_path[paths][4];//For each row except the first, we have
origin socket and destination socket.
//Additional variables that must be entered are losses for nodes 7
and 9 in dsragent.cc. Additionally the gamma value in
static_reputation_udp.tcl must be the same as gamma_path[paths]
//int sampling_time=5;//Sampling time for centralized reputation
   //Entering data
// Start: First variables declared for experimental calculations
bool tx_path[paths]={false};//Event:Transmitting
bool rx_path[paths]={false};//Event:Receiving
bool fw_path[paths]={false};//Event:Forwarding
float sent[paths];//Number of packets sent by each path
float received[paths];//Number of packets received by each path
float int_nodes[paths];//Intermediate nodes by each path
bool next_hop_is_destination=0;//1 if next hop is destination
int cont_header=1;//Print only in first iteration
int cont_print=0;//Print counter for centralized reputation
int index_buffer[paths];//Buffer path indexes for each path//...
int buffer_path[paths][nodes];//Buffer for paths
int ban1=0;//Printing headers
float RFV_P[paths];//Average reputation for paths in experimental
calculations.
float score_centralized_path[paths];//This array will receive the
product of the RFVnj of intermediate nodes that form a path
float ER_N[nodes];//Nth root of the RFVpi.
float RFV_N[nodes];//Results to be used as a metric
int source=1;//It contains the source node
int destination=1;//It contains the destination node
int actual_node;//It contains the actual node from among the
multiple intermediate nodes to reach the destination
int next_hop_2=0;//The next hop
//   Fin.      Primeras variables declaradas para c√°lculos
experimentales.
//Inicio: Nuevas variables para c√°lculo te√≥rico
float  RR_P[paths];//Reputaci√≥n real del camino en c√°lculos
te√≥ricos.
float   TR_N[nodes];//Inicializaci√≥n de reputaci√≥n te√≥rica
centralizada del nodo.
float centralized_gamma_paths=0,temp_centralized=0;
float local_gamma_paths=0,temp_local=0;
float index_stored_buffer[paths];
float c_N_b_P[paths];//centralized_node_belong_path
float l_N_b_P[paths];//local_node_belong_path
float SR_P[paths];
// End: First variables declared for experimental calculations
//         Start:Implementing the source node. Descentralized
information to source node
float  t_source_node[nodes][nodes];//Each row is information that
possesses a source node. Theoretical calculations
float  e_source_node[nodes][nodes];//Each row is information that
possesses a source node. Experimental calculations
```

```
float score_local_path[nodes][paths];//Paths' estimated reputation
seen by the source
float RFV_S_P[nodes][paths];//Paths' reputation observed by each
source node
int id_paquete=0;//Packet identifier. Used to identify broken
paths.
int change_packet[paths];//1: Follow another package to identify
the new routes
int id_in_source[paths];//Save the package ID when a package is
created
//         End:Implementing the source node. Descentralized
information to source node
//Gamma correction
int intermediate[paths];//Flags array that enables calculation
reputation when the packet reaches the destination
//Gamma correction
//Variables for FPL
int header_comparative_transport=1;
int reset_fpls_transport=0;
float rx_cmu=1;//Summation of all packets received
float tx_cmu=1;//Summation of all packets transmitted
int index_matrix_trace=1;//Count all sockets that are enabled in
the simulation
int index_matrix_trace_2=0;
int index_nones_trace=1;//Counter for sockets
int PATHS_OTCL=1;//Store variable that comes from OTCL script
int NODES_OTCL=1;//Store variable that comes from OTCL script
bool flag_sockets=0;//flag for sockets
//Variables for FPL
int SAMPLING_INTERVAL_OTCL=50; //packets, REMODE_interval
int SAMPLING_INTERVAL_RESULTS_OTCL=2; //packets, results_interval
//int packets_trace[paths];//Packet counter in every path
int new_sample_trace[paths];//(packets)Indicates when to take a
sample in every path
int first_ti=1;

int new_sample_trace_c=50;

double start_time, final_time, diff_time;

//int all_sent=1;

float                                          time_pck_rx[paths],
time_pck_tx[paths],delay_pck[paths];//Calculating delay
float
counter_pck[paths],accumulated_delay_pck[paths],avg_delay_pck[paths
];//Calculating average delay
int flag_delay=0,flag_delay_2=0;//Printing headers
//int increment_sample=1;
float acc_avg_delay=0,avg_avg_delay=0;
//int destination_received=0;

float hops[paths],accumulated_hops[paths],avg_hops[paths];
float acc_avg_hops=0,avg_avg_hops=0;

float consumed_energy[nodes];
```

```
float
acc_energy_consumption=0,initial_energy[nodes],remaining_energy[nod
es];//Joules
float avg_energy_consumption=0;
int pkt_size=0;
int hdr_size=0;
int pkt_size_2=0;
int recvd_hdr_size[paths];
int recvd_data_size[paths];
float acc_recvd_hdr_size=0, acc_recvd_data_size=0;
float avg_recvd_hdr_size=0, avg_recvd_data_size=0;

int flag_fpl=0;
int flag_energy=1;

//int counter_sample[paths];
int cont_header_cmu=1;
int cont_header_cmu_2=1;

int counter_trace_analyzer=0;
//End: Inicialization of variables.
int multiple_trace_1=1;
int multiple_trace_2=1;
int multiple_trace=1;

float acc_fpls;
.
.
.
//Beginning modification for verification of reputation equations

source=Address::instance().get_nodeaddr(ih->saddr());//Source node
destination=Address::instance().get_nodeaddr(ih-
>daddr());//Destination node
actual_node=src_;//Actual node
next_hop_2=ch->next_hop_;//Next node on the path
next_hop_is_destination=(next_hop_2==destination);  //True  when  a
packet to be sent reaches the destination
id_paquete=ch->uid();// Unique id of the packet
pkt_size=ch->size();
//Inicializing variables
if(first_ti==1){
char *transport_selector_otcl=getenv("transport_selector_2");
TRANSPORT_SELECTOR_OTCL=atof(transport_selector_otcl);
char *paths_otcl=getenv("paths_2");
PATHS_OTCL=atof(paths_otcl);
char *nodes_otcl=getenv("nodes_2");
NODES_OTCL=atof(nodes_otcl);
char *alpha_otcl=getenv("alpha_2");
ALPHA_OTCL=atof(alpha_otcl);
char *beta_otcl=getenv("beta_2");
BETA_OTCL=atof(beta_otcl);
char *source_node_n=getenv("source_node_2");
SOURCE_NODE_OTCL=atof(source_node_n);
char *path_s_n_n=getenv("path_s_n_2");
PATH_S_N_OTCL=atof(path_s_n_n);
```

```
char *pck_size_otcl=getenv("pck_size_2");
PCK_SIZE_OTCL=atof(pck_size_otcl);
char *ignore_initial_time_otcl=getenv("ignore_initial_time_2");
IGNORE_INITIAL_TIME_OTCL=atof(ignore_initial_time_otcl);
char *initial_energy_otcl=getenv("initial_energy_2");
INITIAL_ENERGY_OTCL=atof(initial_energy_otcl);
char *sampling_interval_otcl=getenv("sampling_interval_2");
SAMPLING_INTERVAL_OTCL=atof(sampling_interval_otcl);
char
*sampling_interval_results_otcl=getenv("sampling_interval_results_2
");
SAMPLING_INTERVAL_RESULTS_OTCL=atof(sampling_interval_results_otcl)
;
for(int ini_p=0;ini_p<=PATHS_OTCL;ini_p++)
{
new_sample_trace[ini_p]=SAMPLING_INTERVAL_OTCL;
gamma_path[ini_p]=1;//simetric traffics
sent[ini_p]=1;
received[ini_p]=1;
int_nodes[ini_p]=1;
RFV_P[ini_p]=1;
score_centralized_path[ini_p]=1;
RR_P[ini_p]=1;
index_buffer[ini_p]=1;
index_stored_buffer[ini_p]=1;
c_N_b_P[ini_p]=0;
l_N_b_P[ini_p]=0;
SR_P[ini_p]=1;
change_packet[ini_p]=1;
id_in_source[ini_p]=0;
intermediate[ini_p]=0;
for(int n_b=0;n_b<100;n_b++){// Numbers of hops. Can be modified
buffer_path[ini_p][n_b]=1;
}
time_pck_rx[ini_p]=0;
time_pck_tx[ini_p]=0;
delay_pck[ini_p]=0;
counter_pck[paths]=0;
accumulated_delay_pck[paths]=0;
avg_delay_pck[paths]=0;
hops[ini_p]=0;
accumulated_hops[ini_p]=0;
avg_hops[ini_p]=0;
recvd_hdr_size[ini_p]=0;
recvd_data_size[ini_p]=0;
counter_trace_analyzer=IGNORE_INITIAL_TIME_OTCL;
//counter_sample[ini_p]=IGNORE_INITIAL_TIME_OTCL;
}
//Assimetric traffics, e.g path 6 with double traffic on the path
//gamma_path[6]=2;
for(int ini_n=0;ini_n<NODES_OTCL;ini_n++){
consumed_energy[ini_n]=0;
initial_energy[ini_n]=INITIAL_ENERGY_OTCL;
remaining_energy[ini_n]=INITIAL_ENERGY_OTCL;
ER_N[ini_n]=1;
RFV_N[ini_n]=1;
```

```
TR_N[ini_n]=1;
for(int ini_n_2=0;ini_n_2<NODES_OTCL;ini_n_2++){
t_source_node[ini_n][ini_n_2]=1;
e_source_node[ini_n][ini_n_2]=1;
}
for(int pa=0;pa<=PATHS_OTCL;pa++){
score_local_path[ini_n][pa]=1;
RFV_S_P[ini_n][pa]=1;
}
RR_N[ini_n]=1;//Assigned reputation
}
first_ti=0;
/*RR_N[7]=0.75;//1. JITEL-2015 scenario
RR_N[9]=0.85;*/
/*RR_N[7]=0.70;//2. PFC scenario
RR_N[9]=0.85;
RR_N[4]=0.60;*/
/*RR_N[4]=0.40; //3. New short scenario
RR_N[5]=0.40;
RR_N[6]=0.40;*/
/*RR_N[4]=0.30;
RR_N[6]=0.60;*/
/*for(int lo=21;lo<=30;lo++) //4. Article for Sensor 10, 20, 30,
40, 50 and 60 selfish nodes
{
RR_N[lo]=0.4;//Varyning between 0.7,0.5,0.3,0.5,0.7 then average
0.54
}*/
RR_N[1]=0.40;
RR_N[2]=0.40;
RR_N[3]=0.40;
//start_time=(unsigned) time(&tt)/3600;
//printf("-->Simulation                                    time:%.4f,Real
Time:%.4f\n",Scheduler::instance().clock(),start_time);//start time
//printf("-->Simulation                                        time:%.4f
s.\n",Scheduler::instance().clock());//start time
printf("\n");
printf(">TRACE_ANALYZER_v3.0: QoS Metrics and more on files\n");
printf(">t:Time(milliseconds)                    R:Received(packets)
T:Transmitted(packets)\n");
printf(">F:FPL(Fraction of Packet Losses, probabilistic value)\n");
printf(">D:Delay(milliseconds) H:Hops(number) E:Energy(Joules)\n");
printf(">Th:Throughput(Kbps) O:Overhead(KB)\n");
printf(">By: Lenin Guaya Delgado.\n");
printf(">SERTEL-ENTEL-UPC-Barcelona Tech-2017\n");
}
if(TRANSPORT_SELECTOR_OTCL==2){
//-New sockets******************
//-Getting new sockets******************
flag_sockets=(index_matrix_trace<=PATHS_OTCL);//Flag     indicating
when to capture a socket
if (flag_sockets==1){
if(index_matrix_trace_2==0){
  socket_path[index_matrix_trace][0]=source;
  socket_path[index_matrix_trace][1]=ih->sport();
  socket_path[index_matrix_trace][2]=destination;
```

```
  socket_path[index_matrix_trace][3]=ih->dport();
  fp=fopen("Trace_Analizer_v3.0_n_sockets_trace.txt","a");
  fprintf(fp,"Time:%.4f,                    index_matrix_trace:%d,
PATHS_OTCL_:%d\n",Scheduler::instance().clock(),index_matrix_trace,
PATHS_OTCL);
  for(int s=1; s<=PATHS_OTCL; s++){
    for(int t=0; t<=3; t++){
    fprintf(fp,"%d\t", socket_path[s][t]);
    }
    fprintf(fp,"\n");
  }
  fclose(fp); //Close file
  index_matrix_trace++;
  index_matrix_trace_2=1;
}
else {
index_matrix_trace_2++;
if(index_matrix_trace_2==4){index_matrix_trace_2=0;}
     }
}//if (flag_s...
flag_sockets=0;
//-New sockets*****************
}//if(TRANSPORT_SELEC...
else{
//-Getting new sockets*****************
flag_sockets=(index_nones_trace<=PATHS_OTCL);//Flag indicating when
to capture a socket
if(flag_sockets==1)
{
if(index_matrix_trace%2==0)//Pairs sockets are not stored because
they are repeated
{index_matrix_trace++;
}
else{//Getting new socket
socket_path[index_nones_trace][0]=source;
socket_path[index_nones_trace][1]=ih->sport();
socket_path[index_nones_trace][2]=destination;
socket_path[index_nones_trace][3]=ih->dport();
fp=fopen("Trace_Analizer_v3.0_n_sockets_trace.txt","a");
fprintf(fp,"Time:%.4f,                    index_matrix_trace:%d,
PATHS_OTCL:%d\n",Scheduler::instance().clock(),index_matrix_trace,P
ATHS_OTCL);
for(int s=1; s<=PATHS_OTCL; s++)
{
  for(int t=0; t<4; t++){fprintf(fp,"%d\t", socket_path[s][t]);}
  fprintf(fp,"\n");
}
fclose(fp); //Close file
index_matrix_trace++;
index_nones_trace++;
}//End:if(index_matr...

}//End:if(flag_s...
flag_sockets=0;
//-Getting new sockets*****************
}//End:if(TRANSPORT_SELEC...
```

```
remaining_energy[actual_node]=energy;
consumed_energy[actual_node]=initial_energy[actual_node]-
remaining_energy[actual_node];//Joules

if((int)Scheduler::instance().clock()>=IGNORE_INITIAL_TIME_OTCL){//
If Time>=100s
if (flag_energy==1){
  for(int n=0;n<NODES_OTCL;n++){
  initial_energy[n]=remaining_energy[n];
  consumed_energy[n]=0;
  }
  flag_energy=0;
}
if(op == DROP){//If a path is broken, follow the path of another
packet (of another packet id) to obtanin intermediate nodes.
  for (int i=1;i<=PATHS_OTCL;i++){
    if((source==socket_path[i][0])&&(ih-
>sport()==socket_path[i][1])&&(destination==socket_path[i][2])&&(ih
->dport()==socket_path[i][3])){ //If a path has been broken
    change_packet[i]=1;// Order the restart index_buffer[i]=1 from
transmission
    }
  }
}

//Beginning for
//Modification for verification equations reputation with the zz
path
for (int zz=1;zz<=PATHS_OTCL;zz++)
{
if((source==socket_path[zz][0])&&(ih-
>sport()==socket_path[zz][1])&&(destination==socket_path[zz][2])&&(
ih->dport()==socket_path[zz][3])&&(strncmp(tracename,"RTR",3)    ==
0)){
  dsr_h->udp_fplp()=1-SR_P[zz]; //(1) FPLs calculated on the trace
file. Precise FPLs.
  tx_path[zz]=((op==SEND)&&(actual_node==socket_path[zz][0])); //If
the event of the trace indicates transmission of a packet
  if (tx_path[zz]==1){
  //tx_cmu++;
  sent[zz]++;//Increase sent counter
  //all_sent++;
  buffer_path[zz][0]=socket_path[zz][0];//Buffers for paths
    if(change_packet[zz]==1){// The order to follow another packet
appears when a path is broken, or when a determined packet is at
its final forwarding to reach the destination.
    id_in_source[zz]=id_paquete;//Following a new packet
    index_buffer[zz]=1;//Reference to index 1 of the new paths
    change_packet[zz]=0;//To disable change packet
    time_pck_tx[zz]=Scheduler::instance().clock();
    }
  }
  rx_path[zz]=((op==RECV)&&(actual_node==socket_path[zz][2])); //If
the event of the trace is receiving a packet
  if (rx_path[zz]==1){
```

```
  //rx_cmu++;
  received[zz]++;//Increase received packet counter
  SR_P[zz]=(received[zz])/(sent[zz]);//Calculating          sampled
reputation
    if (pkt_size>=PCK_SIZE_OTCL){
    //Rip off the header
    hdr_size = pkt_size%PCK_SIZE_OTCL;
    pkt_size_2= pkt_size;//
    pkt_size_2-= hdr_size;//
    //Store received packet's size
    recvd_hdr_size[zz]+=hdr_size;
    recvd_data_size[zz]+= pkt_size_2;
    }
  }

fw_path[zz]=((op==FWRD)&&(id_in_source[zz]==id_paquete)&&(change_pa
cket[zz]==0)); //If the event of the trace is forwarding a packet
  if (fw_path[zz]==1){
  buffer_path[zz][index_buffer[zz]]=actual_node;  //If  a  node  that
forwards puts it in the buffer//7??
    //printf("a_n=%d ",actual_node);
    if(next_hop_is_destination){//If the packet is arriving at the
destination to do the following, if not increase index_buffer
                                //and store the next intermediate
node
    int_nodes[zz]=ch->num_forwards();//Number of intermediate nodes
    //printf("next_hop_is_destination_int_node[%d]=%d
",zz,int_nodes[zz]);
      if(intermediate[zz]==1){//If  the  path  has  been  sampled  is
possible  to  calculate  the  reputation  when  packet  arrives  at  the
destination
      //printf("Intermediate_int_nodes[%d]=%d",zz,int_nodes[zz]);
      score_centralized_path[zz]=1;//When   packet   reaches   the
destination restart Scores.
      score_local_path[socket_path[zz][0]][zz]=1;//Local score from
the source node.
        for(int  b=1;b<=index_buffer[zz];b++){//Calculating  RFVnjs.
First centralized after decentralized (local)
      //Centralized reputation
      //Experimental values from a general view
      ER_N[buffer_path[zz][b]]=pow(RFV_P[zz],1/int_nodes[zz]);//Cal
culating estimated reputation for each node
      RFV_N[buffer_path[zz][b]]=(1-
BETA_OTCL)*RFV_N[buffer_path[zz][b]]+BETA_OTCL*ER_N[buffer_path[zz]
[b]];//Calculating reputation forwarding value for each node
      score_centralized_path[zz]=score_centralized_path[zz]*RFV_N[b
uffer_path[zz][b]];    //Calculating score for each path
      //Local reputation
      //Experimental values from source node
        e_source_node[socket_path[zz][0]][buffer_path[zz][b]]=(1-
BETA_OTCL)*e_source_node[socket_path[zz][0]][buffer_path[zz][b]]+BE
TA_OTCL*pow(RFV_S_P[socket_path[zz][0]][zz],1/int_nodes[zz]);//RFVn
js for each node. Take into count only the reputations of the paths
seen by each source node
      score_local_path[socket_path[zz][0]][zz]=score_local_path[soc
ket_path[zz][0]][zz]*e_source_node[socket_path[zz][0]][buffer_path[
```

```
zz][b]];//Multiplication of local average reputations of each
intermediate node
      }//End:for(int b=1...
      intermediate[zz]=0;//Sampling completed
      }//End:if(intermedi...
      //Beginning 1:New code for theoretical calculations
      //Calculating the reputation of the path from the reputation
assigned to each node (RR_N[# nodes])
      RR_P[zz]=1;//Real reputation of the path
      for(int r_p=1;r_p<=index_buffer[zz];r_p++){
      RR_P[zz]=RR_P[zz]*RR_N[buffer_path[zz][r_p]];//Calculating
the theoretical reputation of the path
      }
      //Calculating the theoretical reputation of each node
      for(int tv=1;tv<=index_buffer[zz];tv++){
      //Start: Search in all n paths, and look if n belongs to that
path
      //Obtainment of paths that belongs to a node
        for (int p=1;p<=PATHS_OTCL;p++){//Initializing variables
        c_N_b_P[p]=0;
        l_N_b_P[p]=0;
        }
        for(int p_s=1;p_s<=PATHS_OTCL;p_s++){
          for(int
i_n=1;i_n<=index_stored_buffer[p_s];i_n++){//Looking node on the
path
            if     (buffer_path[zz][tv]==buffer_path[p_s][i_n])//If
intermediate node belongs to a certain path. Global Vision
            {c_N_b_P[p_s]=1;}
            if
((buffer_path[zz][tv]==buffer_path[p_s][i_n])&&(buffer_path[p_s][0]
==socket_path[zz][0]))//(If intermediate node belongs to a certain
path)&&(And the zero position of the path is the source node)-
>Local Vision.
            {l_N_b_P[p_s]=1; }
          }
        }
        //End: Search in all n paths, and look if n belongs to that
path
        //Start:   Number   of   intermediate   nodes   in   the
buffer_camino..ss
        index_stored_buffer[zz]=index_buffer[zz];
        //End:    Number    of    intermediate    nodes    in    the
buffer_camino..ss
        //Beginning: Calculating the value of paths range
        centralized_gamma_paths=0;  //Total  traffic  that  passes
through a node from a global perspective (centralized)
        local_gamma_paths=0; //Total traffic that passes through a
node from a local perspective (decentralized)
        temp_centralized=0;
        temp_local=0;
        for (int g=1; g<=PATHS_OTCL;g++){
        temp_centralized=c_N_b_P[g]*gamma_path[g];          //Common
variable for global and local gamma_path[g]
       if(temp_centralized!=0){
```

```
centralized_gamma_paths=centralized_gamma_paths+temp_centralized;
        }
        temp_local=l_N_b_P[g]*gamma_path[g];
        if(temp_local != 0){
          local_gamma_paths=local_gamma_paths+temp_local;
        }
        }
        if (centralized_gamma_paths!=0){
        TR_N[buffer_path[zz][tv]]=0;
          for(int p=1;p<=PATHS_OTCL;p++){

TR_N[buffer_path[zz][tv]]=TR_N[buffer_path[zz][tv]]+c_N_b_P[p]*(gam
ma_path[p]/centralized_gamma_paths)*pow(RR_P[p],1/index_stored_buff
er[p]);
        }
        }
        if (local_gamma_paths!=0){
        t_source_node[socket_path[zz][0]][buffer_path[zz][tv]]=0;

          for(int p=1;p<=PATHS_OTCL;p++){

t_source_node[socket_path[zz][0]][buffer_path[zz][tv]]=t_source_nod
e[socket_path[zz][0]][buffer_path[zz][tv]]+l_N_b_P[p]*(gamma_path[p
]/local_gamma_paths)*pow(RR_P[p],1/index_stored_buffer[p]);
        }
        }
      }//End of for used in "search intermediate node on the path":
for(int tv=1;tv...

//End 1:New code for theoretical calculations
    change_packet[zz]=1;//Flag indicating that the packet which was
following has reached the destination, so from here the order is
given to follow another packet
    //id_in_source[zz]=0;   //Is restarted the packet identifier
which was following (analysing) hop by hop to set the number of
intermediate nodes on the path
    if(change_packet[zz]==1){//&&id=iD
    //discriminar por id de paquete
    time_pck_rx[zz]=Scheduler::instance().clock();
    delay_pck[zz]=time_pck_rx[zz]-time_pck_tx[zz];
    hops[zz]=ch->num_forwards()+1;//hops
    //Imprimir el promedio por camino
    counter_pck[zz]++;
    //destination_received++;
    accumulated_delay_pck[zz]+=delay_pck[zz];
    accumulated_hops[zz]+=hops[zz];//hops
    avg_delay_pck[zz]=accumulated_delay_pck[zz]/counter_pck[zz];
    avg_hops[zz]=accumulated_hops[zz]/counter_pck[zz];//n_nodes
    //printf("time_pck_rx[%d]:%.7f,           time_pck_tx[%d]:%.7f,
delay_pck[%d]:%.7f,
avg_delay_pck[%d]:%.7f\n",zz,time_pck_rx[zz],zz,time_pck_tx[zz],zz,
delay_pck[zz],zz,avg_delay_pck[zz]);
    if(zz==PATH_S_N_OTCL){
```

```
fp=fopen("Trace_Analizer_v3.0_delay_and_hops_path.txt","a");//Reput
ation trace
    //Printing headers.
    if(flag_delay_2==0){//Printing
    fprintf(fp,"Delay_path[%d]\n",PATH_S_N_OTCL);
    fprintf(fp,"Time\t");
    fprintf(fp,"time_pck_rx[%d]\t",PATH_S_N_OTCL);
    fprintf(fp,"time_pck_tx[%d]\t",PATH_S_N_OTCL);
    fprintf(fp,"delay_pck[%d]\t",PATH_S_N_OTCL);
    fprintf(fp,"avg_delay_pck[%d]\t",PATH_S_N_OTCL);
    fprintf(fp,"Time\t");
    fprintf(fp,"hops_p[%d]\t",PATH_S_N_OTCL);
    fprintf(fp,"avg_hops[%d]",PATH_S_N_OTCL);
    fprintf(fp,"\n");
    flag_delay_2=1;
    }
    //Printing headers.

fprintf(fp,"%.7f\t%.7f\t%.7f\t%.7f\t%.7f\t%.7f\t%.7f\t%.7f\n",Sched
uler::instance().clock(),time_pck_rx[zz],time_pck_tx[zz],delay_pck[
zz],avg_delay_pck[zz],Scheduler::instance().clock(),hops[zz],avg_ho
ps[zz]);
    fclose(fp); //Close file
    }

    }//End if(change_...
    }//End of next_hop_is_destination
    index_buffer[zz]++;//If there are more intermediate nodes, if
it is not the last hop the counter increase
  }//End of forwarding
  if(tx_path[zz]==1)//It is only comparative when transmitted by
certain path
  {//If Start:if (tx_path[z
//Sampling of the path every n packets
    if((int)sent[zz]==new_sample_trace[zz]){
    new_sample_trace[zz]+=SAMPLING_INTERVAL_OTCL;//next sample
    RFV_P[zz]=(1-
ALPHA_OTCL)*RFV_P[zz]+ALPHA_OTCL*SR_P[zz];//centralized  reputation
for paths
    RFV_S_P[socket_path[zz][0]][zz]=(1-
ALPHA_OTCL)*RFV_S_P[socket_path[zz][0]][zz]+ALPHA_OTCL*SR_P[zz];//l
ocal reputation for paths
    intermediate[zz]=1;
    }//End if((int)sent[zz...

    /*
    if(all_sent==new_sample_trace_c){
    new_sample_trace_c+=SAMPLING_INTERVAL_RESULTS_OTCL;
    }//End: if(all_sent==...*/
  }//If Stop: if (tx_path[z...
//For a certain path of transmition

}//End of path: if((source...
}//End for: for (int zz=1...
```

```
/*
//Sampling of the path every 5 seconds
if((int)Scheduler::instance().clock()==counter_sample[zz])
{
      RFV_P[zz]=(1-ALPHA_OTCL)*RFV_P[zz]+ALPHA_OTCL*SR_P[zz];
      RFV_S_P[socket_path[zz][0]][zz]=(1-
ALPHA_OTCL)*RFV_S_P[socket_path[zz][0]][zz]+ALPHA_OTCL*SR_P[zz];
      counter_sample[zz]+=sampling_time;
      intermediate[zz]=1;
}*/

float                                              residue_trace=(int)
Scheduler::instance().clock()%SAMPLING_INTERVAL_RESULTS_OTCL;
if(residue_trace==0){
if(multiple_trace==1){
//if((int)Scheduler::instance().clock()==counter_trace_analyzer){
  //Beginning: Printing trace reputation that provides centralized
information a global view of all reputations that are calculated
  rx_cmu=0;
  tx_cmu=0;
  for(int r_t=1;r_t<=PATHS_OTCL;r_t++){
  rx_cmu+=received[r_t];
  tx_cmu+=sent[r_t];
  }
  /*final_time=(unsigned) time(&tt)/3600;
  diff_time = final_time-start_time;
  diff_time=(final_time - start_time)/CLK_TCK;
  printf("-->Simulation   time:%.4f,   Real   Time:%.4f,   Difference
time(real):%.4f\r",Scheduler::instance().clock(),final_time,diff_ti
me);*/

fp=fopen("Trace_Analizer_v3.0_centralized_reputation.txt","a");//Re
putation trace
  if(cont_header==1)//Only in the first interaction printing header
  {
  fprintf(fp,"Trace_Analizer_v3.0_centralized_reputation:\n");
  fprintf(fp,"Time\t");
  for(int p=1; p<=PATHS_OTCL; p++){fprintf(fp,"(1-FPL_p%d)\t",p);}
  fprintf(fp,"Time\t");
  for(int p=1; p<=PATHS_OTCL; p++){fprintf(fp,"RFV_p%d\t",p);}
  fprintf(fp,"Time\t");
  for(int p=1; p<=PATHS_OTCL; p++){fprintf(fp,"RR_p%d\t",p);}
  fprintf(fp,"Time\t");
  for(int p=1; p<=PATHS_OTCL; p++){fprintf(fp,"SCORE_p%d\t",p);}
  fprintf(fp,"Time\t");
  for(int n=0; n<NODES_OTCL; n++){fprintf(fp,"ER_n%d\t",n);}
  fprintf(fp,"Time\t");
  for(int n=0; n<NODES_OTCL; n++){fprintf(fp,"RFV_n%d\t",n);}
  fprintf(fp,"Time\t");
  for(int n=0; n<NODES_OTCL; n++)
  {fprintf(fp,"TR_n%d\t",n);}
  fprintf(fp,"Time\t");
  for(int                    p=1;                    p<=PATHS_OTCL;
p++){fprintf(fp,"Interm_p%d[1]\t",p);}
  fprintf(fp,"Time\t");
  for(int p=1; p<=PATHS_OTCL; p++){fprintf(fp,"I_N_p%d\t",p);}
```

```
  fprintf(fp,"Time\tGamma_paths\t");
  fprintf(fp,"Time\t");
  for(int                    p=1;                    p<=PATHS_OTCL;
p++){fprintf(fp,"Received_p%d\tSent_p%d\t(FPL_p%d)\t",p,p,p);}
  fprintf(fp,"Received_total\tSent_total\tFPL_total\n");
  cont_header=0;//Only to print the first time
  }
  //Every "n seconds" to print all reputations
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int p=1; p<=PATHS_OTCL; p++){fprintf(fp,"%.4f\t",SR_P[p]);}
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int p=1; p<=PATHS_OTCL; p++){fprintf(fp,"%.4f\t",RFV_P[p]);}
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int p=1; p<=PATHS_OTCL; p++){fprintf(fp,"%.4f\t",RR_P[p]);}
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int                    p=1;                    p<=PATHS_OTCL;
p++){fprintf(fp,"%.4f\t",score_centralized_path[p]);}
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int n=0; n<NODES_OTCL; n++){fprintf(fp,"%.4f\t",ER_N[n]);}
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int n=0; n<NODES_OTCL; n++){fprintf(fp,"%.4f\t",RFV_N[n]);}
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int n=0; n<NODES_OTCL; n++){fprintf(fp,"%.4f\t",TR_N[n]);}
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int                    p=1;                    p<=PATHS_OTCL;
p++){fprintf(fp,"%d\t",buffer_path[p][1]);}
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int                    p=1;                    p<=PATHS_OTCL;
p++){fprintf(fp,"%.4f\t",index_stored_buffer[p]);}

fprintf(fp,"%.4f\t%.4f\t",Scheduler::instance().clock(),centralized
_gamma_paths);
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int                    p=1;                    p<=PATHS_OTCL;
p++){fprintf(fp,"%.4f\t%.4f\t%.4f\t",received[p],sent[p],1-
received[p]/sent[p]);}
  fprintf(fp,"%.4f\t%.4f\t%.4f\n",rx_cmu,tx_cmu,1-
(rx_cmu/tx_cmu)>=0?1-(rx_cmu/tx_cmu):0);
  fclose(fp); //Close file
  //End:  Printing  trace  reputation  that  provides  centralized
information a global view of all reputations that are calculated

  //Local reputation from trace
  //Printing headers.
  if(ban1==0){//Printing headers.

fp=fopen("Trace_Analizer_v3.0_local_reputation.txt","a");//Reputati
on trace
  fprintf(fp,"Source                                          node=%d,
analyzing_path=%d\n",SOURCE_NODE_OTCL,PATH_S_N_OTCL);
  fprintf(fp,"Time\t");
  for(int i=1;i<=PATHS_OTCL;i++){fprintf(fp,"RFV_p%d\t",i);}
  fprintf(fp,"Time\t");
  for(int i=0;i<NODES_OTCL;i++){fprintf(fp,"RFV_n%d\t",i);}
  fprintf(fp,"Time\t");
  for(int i=0;i<NODES_OTCL;i++){fprintf(fp,"TR_n%d\t",i);}
```

```
  fprintf(fp,"Time\t");
  for(int i=1;i<=PATHS_OTCL;i++){fprintf(fp,"SCORE_p%d\t",i);}
  fprintf(fp,"Time\t");
  for(int i=1;i<=PATHS_OTCL;i++){fprintf(fp,"Lost_p%d\t",i);}
  fprintf(fp,"Time\t");
  for(int i=1;i<=PATHS_OTCL;i++){fprintf(fp,"Received_p%d\t",i);}
  fprintf(fp,"Time\t");
  for(int
i=1;i<=PATHS_OTCL;i++){fprintf(fp,"Transmitted_p%d\t",i);}
  fprintf(fp,"Time\t");
  for(int i=1;i<=PATHS_OTCL;i++){fprintf(fp,"FPL_p%d_trace\t",i);}
  fprintf(fp,"\n");
  ban1=1;
  fclose(fp); //Close file
  }
  //Printing headers.
  //Printing(source node, path)

fp=fopen("Trace_Analizer_v3.0_local_reputation.txt","a");//Reputati
on trace
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int                    p=1;                    p<=PATHS_OTCL;
p++){fprintf(fp,"%.4f\t",RFV_S_P[SOURCE_NODE_OTCL][p]);}
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int                     p=0;                     p<NODES_OTCL;
p++){fprintf(fp,"%.4f\t",e_source_node[SOURCE_NODE_OTCL][p]);}
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int                     p=0;                     p<NODES_OTCL;
p++){fprintf(fp,"%.4f\t",t_source_node[SOURCE_NODE_OTCL][p]);}
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int                    p=1;                    p<=PATHS_OTCL;
p++){fprintf(fp,"%.4f\t",score_local_path[SOURCE_NODE_OTCL][p]);}
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int   p=1;   p<=PATHS_OTCL;   p++){fprintf(fp,"%.4f\t",sent[p]-
received[p]);}
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int                    p=1;                    p<=PATHS_OTCL;
p++){fprintf(fp,"%.4f\t",received[p]);}
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int p=1; p<=PATHS_OTCL; p++){fprintf(fp,"%.4f\t",sent[p]);}
  fprintf(fp,"%.4f\t",Scheduler::instance().clock());
  for(int    p=1;    p<=PATHS_OTCL;    p++){fprintf(fp,"%.4f\t",1-
received[p]/sent[p]);}
  fprintf(fp,"\n");
  fclose(fp); //Close file
  //Printing(source node, path)
  //Local reputation from trace

  //QoS metrics
  acc_avg_delay=0;
  for(int                    p=1;                    p<=PATHS_OTCL;
p++){acc_avg_delay+=avg_delay_pck[p];}
  acc_avg_hops=0;
  for(int                    p=1;                    p<=PATHS_OTCL;
p++){acc_avg_hops+=avg_hops[p];}//n_nodes
  avg_avg_delay=acc_avg_delay/PATHS_OTCL;
```

```
  avg_avg_hops=acc_avg_hops/PATHS_OTCL;//n_nodes
  acc_energy_consumption=0;
  for(int                    n=1;                    n<=NODES_OTCL;
n++){acc_energy_consumption+=consumed_energy[n];}//n_nodes
  avg_energy_consumption=acc_energy_consumption/NODES_OTCL;
  acc_recvd_hdr_size=0;
  acc_recvd_data_size=0;
  for(int p=1; p<=PATHS_OTCL; p++){
  acc_recvd_hdr_size+=recvd_hdr_size[p];
  acc_recvd_data_size+=recvd_data_size[p];
  }
  avg_recvd_hdr_size=acc_recvd_hdr_size/PATHS_OTCL;
  avg_recvd_data_size=acc_recvd_data_size/PATHS_OTCL;

  //Printing headers.
  if(flag_delay==0){//Printing
  fp=fopen("Trace_Analizer_v3.0_QoS_metrics.txt","a");//Reputation
trace
  fprintf(fp,"QoS_metrics\n");
  fprintf(fp,"Time\t");
  fprintf(fp,"Received\t");
  fprintf(fp,"Transmitted\t");
  fprintf(fp,"FPL_total\t");
  fprintf(fp,"Delay_total\t");
  fprintf(fp,"Hops_total\t");
  fprintf(fp,"Energy_total\t");
  fprintf(fp,"Throughput_total\t");
  fprintf(fp,"Overhead_total");
  fprintf(fp,"\n");
  flag_delay=1;
  fclose(fp); //Close file
  }
  //Printing headers.
  acc_fpls=0;
  for(int p=1; p<=PATHS_OTCL; p++){
  acc_fpls+=(1-received[p]/sent[p]);
  }
  fp=fopen("Trace_Analizer_v3.0_QoS_metrics.txt","a");//Reputation
trace
  fprintf(fp,"%.7f\t",Scheduler::instance().clock());
  fprintf(fp,"%d\t",(int)rx_cmu);
  fprintf(fp,"%d\t",(int)tx_cmu);
  fprintf(fp,"%.7f\t",1-(rx_cmu/tx_cmu)>=0?1-(rx_cmu/tx_cmu):0);
  fprintf(fp,"%.7f\t",avg_avg_delay*1000);
  fprintf(fp,"%.7f\t",avg_avg_hops);
  fprintf(fp,"%.7f\t",avg_energy_consumption);

fprintf(fp,"%.7f\t",Scheduler::instance().clock()==0?0:(avg_recvd_d
ata_size/Scheduler::instance().clock())*(8.0/1000.0));//Throughput
in Kbps
  fprintf(fp,"%.7f",avg_recvd_hdr_size/1000.0);//Header in KBytes
  fprintf(fp,"\n");
  fclose(fp); //Close file
  printf("t:%.2f  R:%d  T:%d  F:%.4f  D:%.4f  H:%.1f  E:%.0f  Th:%.0f
O:%.0f\r",Scheduler::instance().clock(),(int)rx_cmu,(int)tx_cmu,1-
(rx_cmu/tx_cmu)>=0?1-
```

```
(rx_cmu/tx_cmu):0,avg_avg_delay*1000,avg_avg_hops,avg_energy_consum
ption,Scheduler::instance().clock()==0?0:(avg_recvd_data_size/Sched
uler::instance().clock())*(8.0/1000.0),avg_recvd_hdr_size/1000.0);

  fp=fopen("Trace_Analizer_v3.0_FPLs_paths.txt","a");
  if(flag_fpl==0){//Printing, e.g. 20 FPL_pis
  fprintf(fp,"Time\t");
  for(int p=1; p<=PATHS_OTCL; p++){fprintf(fp,"FPLp%d\t",p);}
  fprintf(fp,"FPL_total\n");
  flag_fpl=1;
  }
  fprintf(fp,"%.7f\t",Scheduler::instance().clock());
  for(int    p=1;    p<=PATHS_OTCL;    p++){fprintf(fp,"%.7f\t",1-
(received[p]/sent[p])>=0?1-(received[p]/sent[p]):0);}
  fprintf(fp,"%.7f\n",acc_fpls/PATHS_OTCL);
  fclose(fp); //Close file
  //QoS metrics
  //counter_trace_analyzer+=SAMPLING_INTERVAL_RESULTS_OTCL;

//
    //Each node
  /*if(zz==PATH_S_N_OTCL){
   float                                    residue_cmu_trace=(int)
Scheduler::instance().clock()%SAMPLING_INTERVAL_RESULTS_OTCL;
    if(residue_cmu_trace==0){
      if(multiple_trace_1==1){  */

fp=fopen("Trace_Analizer_v3.0_used_path_each_node.txt","a");//Open
file
        if(cont_header_cmu==1){//Only  in  the  first  iteration  is
printed the header
        fprintf(fp,"Source            node=%d,            analyzing
path=%d\n",SOURCE_NODE_OTCL,PATH_S_N_OTCL);
        fprintf(fp,"Time\tEach_node\n");
        cont_header_cmu=0;
        }

fprintf(fp,"%.4f\t[(%d)",Scheduler::instance().clock(),socket_path[
PATH_S_N_OTCL][0]);
        for(int     h=1;     h<=index_stored_buffer[PATH_S_N_OTCL];
h++)//h, number of intermediate nodes for each path
        {fprintf(fp," %d",buffer_path[PATH_S_N_OTCL][h]);}
        fprintf(fp," %d ]\n",socket_path[PATH_S_N_OTCL][2]);
        fclose(fp); //Close file
/*
      multiple_trace_1=0;
      }
    }
    else{multiple_trace_1=1;}
  }*/
  /*if(zz==20){
    float                                  residue_cmu_trace_2=(int)
Scheduler::instance().clock()%SAMPLING_INTERVAL_RESULTS_OTCL;
    if(residue_cmu_trace_2==0){
      if(multiple_trace_2==1){
```

```
fp=fopen("Trace_Analizer_v3.0_path_each_node_p2.txt","a");//Open
file
        if(cont_header_cmu_2==1){//Only in the first iteration is
printed the header
        fprintf(fp,"Source node=%d, analyzing path=%d\n",20,20);
        fprintf(fp,"Time\tEach_node\n");
        cont_header_cmu_2=0;
        }

fprintf(fp,"%.4f\t[(%d)",Scheduler::instance().clock(),socket_path[
20][0]);
        for(int h=1; h<=index_stored_buffer[20]; h++)//h, number of
intermediate nodes for each path
        {fprintf(fp," %d",buffer_path[20][h]);}
        fprintf(fp," %d ]\n",socket_path[20][2]);
        fclose(fp); //Close file
        multiple_trace_2=0;
      }
    }
    else{multiple_trace_2=1;}
  //}*/
    //Each node

//


  multiple_trace=0;
  }
}
else{multiple_trace=1;}
  for (int bo=1;bo<=PATHS_OTCL;bo++)//Reset the variables for use
in the next interaction
  {tx_path[bo]=0; rx_path[bo]=0; fw_path[bo]=0;}
  next_hop_is_destination=0; //Is =1 when a nodo forwarding reach
the destination

}//End                                                          If
Time>=100s:if((int)Scheduler::instance().clock()>=IGNORE_INITIAL_TI
ME_OTCL...

//Finalizing modification for verification of reputation equations
```

D.2 Más código fuente

El código fuente que se integró al simulador NS-2.35, está disponible en:

https://sites.google.com/view/manets

Para poder utilizar el código fuente, se requieren conocimientos previos en manejo del simulador NS-2.35. Se requiere tener instalado NS-2.35, reemplazar los archivos nuevos que se presenta en el link por lo anteriores. Luego de ello se recompila, y listo.

Referencias

[1] ERICSSON, "Ericsson Mobility Report: 70 percent of world's population using smartphones by 2020," Press release. June 03, 2015.

[2] L. Guaya, *Tesis de Maestría: "Diseño de un sistema de telemedición y telecontrol mediante el uso de los estándares inalámbricos GPRS y Bluetooth"*, Maestría en Telemática. Facultad de Ingeniería, Ed. Cuenca, Cuenca, Ecuador: Universidad de Cuenca, 2011.

[3] L. Guaya, *Tesis de Ingeniería: "Diseño e implementación de un sistema de control mediante el servicio SMS de telefonía celular"*, Escuela de Electrónica y Telecomunicaciones, Ed. Loja, Loja, Ecuador: Universidad Técnica Particular de Loja, 2008.

[4] S. Harishankar, I. Woungang, S. Kumar, "E-MAntNet: An ACO-Based Energy Efficient Routing Protocol for Mobile Ad Hoc Networks," *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, p. 8.

[5] A. Mezher, M. Aguilar, L. de la Cruz, E. Pallarès, C. Tripp, L. Urquiza, J. Forné, E. Sanvicente, "A Multi-User Game-Theoretical Multipath Routing Protocol to Send Video- Warning Messages over Mobile Ad Hoc Networks," *Sensors*, p. 39, 2015.

[6] M. Thebiga, R. Pramila, "An Analysis of routing protocols in MANETs and Internet of Things," *2017 International Conference on IoT and Application (ICIOT)*, May 2017.

[7] A. Gupta, R. Kumar, "A Survey of 5G Networks: Architecture and Emerging Technologies," July 2015, Special Section on Recent Advances in Software Defined Networking for 5G Networks.

[8] CISCO, "Cisco 5G Vision Series: Laying the Foundation for New Technologies, Use Cases, and Business Models," White paper. 2016.

[9] J. Toung, R. Gilstrap, K. Freeman, "A Split Implementation of the Dynamic Source Routing Protocol for Lunar/Planetary Surface Communications," p. 8, January 2006, NASA Ames Research Center.

[10] C. Wu, M. Gerla, M. Schaar, "Social Norm Incentives for Network Coding in

MANETs," *IEEE/ACM Transactions on Networking*, p. 14, 2017.

[11] J. Muñoz, O. Esparza, M. Aguilar, V. Carrascal, J. Forne, "RDSR-V. Reliable Dynamics Source Routing for video-streaming over mobile ad hoc networks," *Computer Networks*, vol. 54, p. 18, 2010.

[12] H. Shen, Z. Li, "A Hierarchical Account-Aided Reputation Management System for MANETs," *IEEE/ACM Transactions on Networking*, vol. 23, p. 15, 2015.

[13] T. Costa, J. Ferreira, R. Francês, "Reliable Energy-Efficient Multilayer Mechanism with Realistic Battery Model and QoE Support in Wireless MANETs," *Journal of Microwaves, Optoelectronics and Electromagnetic Applications*, vol. 15, no. 1, March 2016.

[14] İ. Bekmezcia, O. Koray, S. Temel, "Flying Ad-Hoc Networks (FANETs): A survey," *Ad Hoc Networks*, vol. 11, no. 3, May 2013.

[15] F. Domingos, L. Villas, A. Boukerche, "Data Communication in VANETs: Survey, Applications and Challenges," *Ad Hoc Networks*, September 2016.

[16] P. Chandra, D. Dobkin, A. Bensky, R. Olexa, D. Lide, F. Dowla, *Wireless Networking know it all*, Elsevier, Ed., 2008.

[17] P. Lobo, S. Acharya, R. Obed, "Quality of Service for MANET based smart cities," *International Journal of Advanced Computational Engineering and Networking*, vol. 5, no. 2, February 2017.

[18] ZTE Communications, "Smart City: Key Technologies and Practices," *An International ICT R&D Journal Sponsored by ZTE Corporation*, vol. 13, no. 4, December 2015.

[19] IEEE, "Part 11:Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," in *Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements —.*, 2012, p. 2798.

[20] Tektronix, "Wi-Fi: Overview of the 802.11 Physical Layer and Transmitter Measurements," 2013.

[21] S. Mohammed, S. Sadkhan, "Design of Wireless Network Based on NS2," *Journal of Global Research in Computer Science*, vol. 3, December 2012.

[22] T. Rappaport, *Wireless Communications. Principles and Practice*, 2nd ed., Inc. Prentice Hall, Ed. Upper Saddle River, NJ, United States of America, 2002.

[23] S. Yuzhuo, K. Hao, W. Beibei, "A New Vehicle Network Routing Technology Based on Power Control," *2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*, September

2010.

[24] E. Ariganello, E. Barrientos, *Redes CISCO CCNP a Fondo. Guía de estudio para profesionales*, RA-MA S. A., Ed. Madrid, España, 2010.

[25] D. Dobkin, *RF Engineering for Wireless Networks. Hardware, Antenas, and Propagation*, Elsevier, Ed., 2005.

[26] F. Eady, *Implementing 802.11 with Microcontrollers. Wireless Networking for Embedded Systems Designers*, Elsevier, Ed. USA, 2005.

[27] W. Stallings, *Comunicaciones y Redes de Computadores*, Séptima ed., Pearson Educación, Ed. Madrid, España, 2004.

[28] Breeze Wireless Communications, "IEEE 802.11 Technical Tutorial".

[29] ¿Cuál es la velocidad real de las conexiones Wi-Fi (IEEE 802.11g)? [Online]. http://redestelematicas.com/cual-es-la-velocidad-real-de-las-conexiones-wi-fi-ieee-802-11g/

[30] M. Hännikäinen. (2017, July) Wireless Sensor Networks. [Online]. http://www.tkt.cs.tut.fi/kurssit/2456/wsnmotivation.html

[31] A. Kumar, D. Manjunath, J. Kuri, *Wireless Networking*. Burlington, USA, 2008.

[32] M. Shah, Tamanna, "A Review on Wireless Sensor Networks (WSN)," *Journal of Network Communications and Emerging Technologies (JNCET)*, vol. 3, no. 2, August 2015.

[33] H. Mogaibel, M. Othman, S. Subramaniam, "High Throughput Path Establishment for Common Traffic in Wireless Mesh Networks," in *Wireless Mesh Networks - Efficient Link Scheduling, Channel Assignment and Network Planning Strategies.*: INTECH, 2012.

[34] Aarti, S. Tyagi, "Study of MANET: Characteristics, Challenges, Application and Security Attacks," *International Journal of Advanced Research in Computer Science and Software Engineering*.

[35] H. Cheng, H. Shan, W. Zhuang, "Infotainment and road safety service support in vehicular networking: From a communication perspective," *Journal of Mechanical Systems and Signal Processing (MSSP, Elsevier)*, vol. 25, no. 6, August 2011.

[36] S. Rehman, M. Arif, T. Zia, L. Zheng, "Vehicular Ad-Hoc Networks (VANETs) - An Overview and Challenges," *Journal of Wireless Networking and Communications 2013*.

[37] H. Moustafa, S. Senouci, M. Jerbi, "Introduction to Vehicular Networks," in

Vehicular Networks: Techniques, Standards, and Applications., 2009.

[38] " Car2Car Communication Consortium Manifesto, work in progress, May 2007,".

[39] K. Ross J. Kurose, *Redes de computadores. Un enfoque descendente basado en Internet.*, Segunda ed., Pearson Educación, Ed. Madrid, 2004.

[40] C. Perkins, *Ad Hoc Networking*.: Addison-Wesley, 2001.

[41] C. Perkins, P. Bhagwat, ""Highly Dynamic Destination-Sequenced Distance Vector Routing (DSDV)" for Mobile Computers," *Proceedings of the ACM SIGCOMM 1994 Conference* , August 1994.

[42] P. Jacquet, P. Muhlethaler, A. Qayyum, ""Optimized Link State Routing Protocol"," *draft-ietf-manet-olsr-05.txt*, 2000.

[43] E. Dijkstra, ""A Note on Two Problems in connection with graphs"," *Numerical Mathematics 1*, October 1959.

[44] G. Pei, M. Gerla, T. Chen, ""Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks"," *Proceedings of the IEEE ICC 1*, June 2000.

[45] C. Santivanez, R. Ramanathan, I. Stavrakakis, ""Making Link-State Routing Scale for Ad Hoc Networks"," *Proceeding of 2001 ACM*, October 2001.

[46] D. Jonhson, D. Maltz, "The Dynamic Source Routing Protocol for Mobile Ad hoc Networks (DSR). ," *Internet Draft*, April 2003.

[47] C. Perkins, E. Royer. (1998, November) "Ad Hoc on-Demand Distance-Vector (AODV) Routing". [Online]. http://www.ietf.org/internet-drafts/draft-ietf-manetaodv-02.txt

[48] Z. Haas, M. Pearlman, ""The Perfomance of Query Control Schemes for the Zone Routing Protocol"," *ACM/IEEE Trans. Net. 9* , August 2001.

[49] S. Bhagchandani, D. Adane, "Route Cache Optimizations of DSR Protocol for VANET," *International Journal of Computer Science And Technology*, vol. 4, no. 1, March 2013.

[50] E. Royer, C. Toh, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Network.," April 1999.

[51] L. Bennani, M. Lorleac'h, P. Sailly, *Multi-parametric routing in Mobile Ad-hoc Networks*. Aalborg, 2004.

[52] A. Srinivasan, J. Teitelbaum, "Reputation and Trust Based Systems for Ad Hoc Networks.," in *Algorithms and Protocols for Wireless and Mobile Ad Hoc*

Networks. Otawa, Canadá: John Wiley & Sons, 2009, pp. 375-403.

[53] A. Srinivasan, J. Teitelbaum, J. Wu., "DRBTS: Distributed Reputation-Based beacon Trust System.," *In the 2nd IEEE International Symposium on Dependable, Automatic and Secure Computing (DASC'06), Indianapolis, USA.*, 2006.

[54] S. Ganeriwal, M. Srivastava., "Reputation-based Framework for high integrity Sensor Networks.," *In Proceeding of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'04)*, pp. 66-77, October 2004.

[55] P. Michiardi, R. Molva, "Simulation-based analysis of security exposures in mobile ad hoc networks.," *European Wireless Conference.*, 2002.

[56] F. Rodriguez, "Técnicas de Reputación para Redes de Comunicaciones Inalámbricas Multi-Salto," Departamento de Ciencias de Materiales, Óptica y Tecnología Electrónica, Universidad Miguel Hernández de Elche, Tesis para el grado de doctor 2013.

[57] P. Michiardi, R.Molva, "CORE: A COllaborative REputation mechanism to enforce node cooperation in mobile ad hoc networks.," *Communication and Multimedia Security*, September, 2002.

[58] S. Buchegger, J. Le Boudec, "Perfomance Analysis of the CONFIDANT Protocol (COoperation of Nodes- Fairness In Dynamic Ad hoc NeT-works).," *Proceeding of MobiHoc 2002, Lausanne, CH, June 2002*.

[59] M. Blaze, J. Feigenbaum, J. Ioannidis, A. Keromytis, *RFC2704. The KeyNote Trust Management System Version 2.*, 1999.

[60] N. Li, J. Mitchell, W. Winsborough., "Design of a Role-based Trust management Framework.," *In Proceedings of the IEEE Symposium on Security and Privacy, Oakland.*, 2002.

[61] S. Martí, T.J. Giuli, K. Lai, M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks.," *In Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom 2000).*

[62] S. Bansal, M. Baker, "Observation-based cooperation enforcement in ad hoc networks. http://arxiv.org/pdf/cs.NI/0307012," 2003.

[63] R. Jurca, B. Faltings, "An incentive compatible reputation mechanisms. In Proceedings of the IEEE Conference on E-Commerce," 2003.

[64] G. Shafer, "A Mathematical Theory of Evidence, Princeton University Press.," 1976.

[65] A. Jsang, "A logic for uncertain probabilities. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 9(3): 279-311, June," 2001.

[66] A. Josang, R. Ismail, "The beta reputation system. In Proceedings of the 15th Bled Electronic Commerce Conference," 2002.

[67] S. Buchegger, J. Le Boudec, "A robust reputation system for per-to peer and mobile ad hoc networks. In Proceedings of P2PEcon 2004, Harvard University," 2004.

[68] S. Bansal, M. Baker, "Observation-based Cooperation Enforcement in ad hoc Nerworks," *Stanford, C A, USA*, 2003.

[69] P. Michiardi, R. Molva, "CORE: A Collaborative Reputation Mechanism to enforce node cooperation in Mobile Ad hoc Networks," *Sophia Antipolis, France*.

[70] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. (2003, July) RFC: 3550. RTP: A Transport Protocol for Real-Time Applications. [Online]. http://www.rfc-editor.org/rfc/rfc3550.txt

[71] D. Kim, H. Bae, J. Song, "Analysis of the Interaction between TCP Variants and Routing Protocols in MANETs," in *Proceedings of the 2005 International Conference on Parallel Processing Workshops (ICPPW'05)*, p. 7.

[72] P. Meher, P. Kulkarni, "Analysis and Comparison of Performance of TCP-Vegas in MANET," in *2011 International Conference on Communication Systems and Network Technologies*, p. 4.

[73] D. Adami, C. Callegari, S. Giordano, M. Pagano, T. Pepe, "A Behavioral Study of TCP Linux Variants over Satellite Networks.," *2011 IEEE/IPSJ International Symposium on Applications and the Internet. University of Pisa. Italy*, p. 6, 2011.

[74] D. Wei, P. Cao, "NS-2 TCP-Linux: An NS-2 TCP Implementation with Congestion Control Algorithms from Linux," *WNS2'06*, p. 10, October 2006.

[75] D. Barrett, *Guía de bolsillo LINUX*. Madrid, España, 2012.

[76] C. Jae. NS by Example. [Online]. http://nile.wpi.edu/NS/

[77] University of Osnabrück, "BonnMotion: A Mobility Scenario Generation and Analysis Tool," February 2016.

[78] University of Birmingham. [Online]. https://intranet.birmingham.ac.uk/it/teams/infrastructure/research/bear/bluebear/

apps/MATLAB/matlab-r2017b.aspx

[79] L. Guaya, E. Pallarès, J. Forné, "REMODE: Un algoritmo para el cálculo de la reputación de nodos y caminos en redes ad hoc cooperativas," in *Libro de Ponencias de las XII Jornadas de Ingeniería Telemática (JITEL 2015)*. Palma de Mallorca, Islas Baleares, España: Universidad de las Islas Baleares, Octubre 2015, pp. 277-284.

[80] A. Duran, C. Shen, "Mobile Ad Hoc P2P File Sharing," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC)*, 2004.

[81] S. Goel, T. Imielinski, K. Ozbay, "Ascertaining Viability of WiFi Based Vehicle-to-Vehicle Network for Traffic Information Dissemination," *Proc. IEEE Int'l Conf. Intelligent Transportation Systems (ITS)*, 2004.

[82] H. Wu, R. Fujimoto, R. Guensler, M. Hunter, "MDDV: Mobility-Centric Data Dissemination Algorithm for Vehicular Networks," *Proc. ACM Int'l Workshop Vehicular Ad Hoc Networks (VANET)*, 2004.

[83] H. Gharavi, "Multichannel Mobile Ad Hoc Links for Multimedia Communications," *Proc. IEEE, vol. 96, no. 1*, 2008.

[84] S. Marti, T.J. Giuli, K. Lai, M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," *Proc. ACM MobiCom*, 2000.

[85] D. Montgomery, *"Introduction to Statistical Quality Control"*, 6th ed., John Wiley & Sons, Ed. New York, USA, 2009.

[86] D. Johnson, D. Maltz, J. Broch, "DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks. In Ad Hoc Networking," *Addison-Wesley Professional*, 2001.

[87] L. Buttyán, J. Hubaux, "Stimulating cooperation in self-organizing mobile ad hoc networks," *Mob. Netw. Appl.*, 2003.

[88] M. Jakobsson, J. Hubaux, L. Buttyán, "A micro-payment scheme encouraging collaboration in multi-hop cellular networks," *In Financial Cryptography*, 2003.

[89] M. Mahmoud, X. Shen, "FESCIM: Fair, Efficient, and Secure Cooperation Incentive Mechanism for Multihop Cellular Networks," *IEEE Transactions on Mobile Computing*, vol. 11, no. 5, May 2012.

[90] E. Chiejina, H. Xiao, B. Christianson, "A Candour-based Trust and Reputation Management System for Mobile Ad Hoc Networks," *In Proceedings of the 6th York Doctoral Symposium on Computer Science & Electronics*, 2013.

[91] N. Li, S. Das, "A trust-based framework for data forwarding in opportunistic

networks," *Ad Hoc Netw*, 2013.

[92] A. Banerjee, S. Neogy, C. Chowdhury, "Reputation based trust management system for MANET," *In Proceedings of the 2012 Third International Conference on Emerging Applications of Information Technology (EAIT)*, 2012.

[93] S. Kumar, "A Novel Routing Strategy for Ad Hoc Networks with Selfish Nodes," *J. Telecommun*, 2010.

[94] W. Gong, Z. You, D. Chen, X. Zhao, M. Gu, K. Lam, "Trust based routing for misbehavior detection in ad hoc networks," *J. Netw.*, 2010.

[95] C. Perkins, E. Royer, "Ad-hoc On-Demand Distance Vector Routing," *In Proceedings of the 2nd IEEE workshop on Mobile Computing Systems and Applications*, 1999.

[96] K. Bakar, J. Irvine, "A Scheme for Detecting Selfish Nodes in MANETs using OMNET++," *In Proceedings of the 2010 6th International Conference on Wireless and Mobile Communications (ICWMC)*, 2010.

[97] C. Tang, A. Li, X. Li, "When Reputation Enforces Evolutionary Cooperation in Unreliable MANETs," *In IEEE Transactions on Cybernetics*, vol. 45, no. 10, October 2015, doi: 10.1109/TCYB.2014.2366971.

[98] F. Saghezchi, A. Radwan, J. Rodríguez, "Energy-aware relay selection in cooperative wireless network: An assigment game approach," *Ad Hoc Networks*, vol. 56, March 2017.

[99] E. Chiejina, H. Xiao, B. Christianson, "A Dynamic Reputation Management System for Mobile Ad Hoc Networks," *Computers*, 2015.

[100] M. Khusru, G. Sahoo, "Enhancing cooperation in MANET using neighborhood compressive sensing model," *Egyptian Informatics Journal*, June 2016, doi: http://dx.doi.org/10.1016/j.eij.2016.06.007.

[101] MathWorks. (2018, July) [Online]. https://www.mathworks.com/

[102] A. Karygiannis, k. Robotis, E. Antonakakis, "Creating Offline MANET IDS Network Traces," *IEEE international symposiumon a world of wireless, mobile and multimedia networks (WoWMoM)*, July 2007.

[103] D. Jhonson, Y. Hu, D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," February 2007.

[104] D. Wei, P. Cao. (2007, September) A Linux TCP implementation for NS2 (Part of the NS-2 enhancement project). [Online]. http://netlab.caltech.edu/projects/ns2tcplinux/ns2linux/

[105] L. Buttyán, J. Hubaux, "Enforcing service availability in mobile ad-hoc WANs," *In Proceedings of the 1st ACM International Symposium on Mobile Ad Hoc Networking & Computing*, 2000.

[106] S. Zhong, J. Chen, Y. Yang, "Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks," *In Proceedings of the IEEE Societies Twenty-Second Annual Joint Conference of the IEEE Computer and Communications (INFOCOM 2003)*, vol. 3, 2003.

[107] M. Félegyházi, L. Buttyán, J. Hubaux, "Equilibrium analysis of packet forwarding strategies in wireless ad hoc networks–the static case," *In Personal Wireless Communications*, 2003.

[108] J. Cai, U. Pooch, "Allocate fair payoff for cooperation in wireless ad hoc networks using shapley value," *In Proceedings of the 18th International Parallel and Distributed Processing Symposium*, 2004.