Dirección discriminante para el encaminamiento: Un nuevo tipo de identificador para la computación ubicua *

Miguel A. Ortuño Pérez Vicente Matellán Olivera Carlos E. Agüero Durán
Gregorio Robles
Departamento de Ingeniería Telemática y Tecnología Electrónica
Universidad Rey Juan Carlos, Móstoles, Madrid
Email: {miguel.ortuno,vicente.matellan,carlos.aguero,gregorio.robles}@urjc.es

Resumen

Abstract *Any device we want to be able to connect to a global network should have an unique global identifier. The size of this identifier can be an unacceptable overhead for devices with limited resources (sensors, toys, disposable devices, micro-robots, etc.), because conventional protocols use full addresses to transmit, process and store the data required for routing.*

The usual solution for such devices is to limit the address space to one or two bytes, but this sacrifices the global unicity of the identifiers. Another more drastic measure is to do without any routing at all.

The proposal presented in this dissertation enables limited resources devices to retain addresses that globally identify hosts. We propose the use of selective addresses for routing *or* abbreviated addresses for routing. *We have developed a new protocol named ADSR. This protocol is a modified version of DSR based on the use of abbreviated addresses. The abbreviation procedure can lead to two different nodes having the same address, which we will term collision. ADSR allows rather than averts collisions.*

1. Introducción

Mark Weiser introduce en 1991 el concepto de *computación ubicua* en un artículo que ya es un clásico [21]. Define un escenario donde pequeños o grandes ordenadores se integran entre sí envolviendo por completo a las personas, hasta el punto de resultar prácticamente invisibles. Esta tecnología debe ser capaz de dimensionarse a cualquier escala, tener siempre en cuenta los elementos del entorno próximo (la mesa, la habitación) y enmascarar situaciones heterogéneas: distintos dispositivos, aplicaciones, fabricantes, estándares, etc. Algunos autores [19] establecen una clasificación donde el primer paso son los sistemas distribuidos, el segundo la computación móvil y el tercero la computación ominipresente (*pervasive computing*), que podemos considerar un sinónimo algo más moderno de *computación ubicua*.

De las muchas facetas que presenta la computación ubicua, en nuestro trabajo nos centraremos en las redes *Ad-Hoc*, también llamadas *mesh networks*. Se definen como redes de comunicaciones auto-organizadas, compuestas por las estaciones que están en determinado momento en determinado lugar y que no precisan de ninguna infraestructura además de las propias estaciones. Emplean tecnologías inalámbricas, que junto con la alimentación mediante baterías y los algoritmos adecuados permiten prescindir de cableado, puntos de acceso, *routers* pre-existentes o alimentación externa [18]. Estas redes están formadas normalmente por nodos similares, no jerarquizados y que cooperan entre sí, donde todos son al tiempo encaminadores y estaciones finales.

Los nodos de estas redes pueden ser, o bien ordenadores, o bien simples sensores con una pequeña capacidad de cálculo. Para el primer caso se puede usar el término *MANET* (*Mobile Ad-Hoc Networks*, redes Ad-Hoc móviles), para el segundo se habla de *redes de sensores* [1] [2].

Los protocolos para MANETs suponen que la red está formada por ordenadores móviles, posiblemente pequeños y alimentados por baterías, pero ordenadores convencionales a la postre. Y si se trata de dispositivos diferentes, se les dota de la capacidad de comunicación de uno de estos ordenadores.

Puede haber ejemplos de redes de ordenadores muy sencillos equipados con sensores que estén en la frontera entre las MANETs y las redes de sensores, siendo discutible su inclusión en uno u otro grupo. Pero en general ambos tipos de redes son distintos y su investigación se considera perteneciente a disciplinas relacionadas pero diferentes. En nuestro trabajo, cuando hablemos de redes Ad-Hoc o de redes Ad-Hoc de dispositivos de recursos limitados nos referiremos a MANETs de ordenadores *pequeños*, esto es, con prestaciones muy inferiores a las de un PC convencional, y no a redes de sensores.

2. Origen histórico

Los comienzos de este trabajo los situamos en el momento en el que intentamos portar DSR (*Dynamic Source Routing Protocol*) [12], uno de los protocolos más destacados para redes Ad-Hoc, a un ordenador *Lego Mindstorm RCX* [3], un juguete con las prestaciones de un micro-ordenador

*Este trabajo ha sido financiado parcialmente por el ministerio de Ciencia y Tecnología, proyecto ACRACE DPI2004-07993-C03-01

de los años 80. Su sistema operativo *brickOS* (`http://brickos.sourceforge.net`, anteriormente denominado *LegOS*) usa un protocolo de nivel de enlace llamado LNP (*Lego Network Protocol*). LNP está basado en una trama de 256 bytes, lo que hace inviable el uso de DSR o cualquier protocolo similar, como muestra la figura 1.

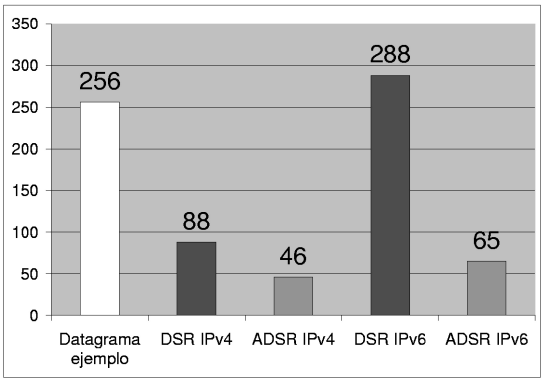

Figura 1: Comparación del tamaño del datagrama (bytes)

Este caso particular nos lleva hasta el problema general de las máquinas *pequeñas*, no en tamaño, sino en prestaciones. Arquitecturas más recientes (como p.e. CotsBots [4]) suelen estar basadas en IEEE 802.15.4/*ZigBee* [11] [23], donde encontramos limitaciones análogas: una trama de 127 octetos.

A consecuencia de esto, comenzamos a modificar el protocolo DSR para posibilitar su uso en dispositivos de recursos tan limitados como este, resultando el protocolo ADSR *Abbreviated Dynamic Source Routing* [14] y [15]. Aunque su origen está en el *Lego Mindstorms*, ADSR no está orientado a *brickOS*, de hecho no hemos trabajado sobre esta plataforma sino sobre el simulador de red de propósito general *ns-2* [9].

A partir de los algoritmos concretos que diseñamos para nuestro protocolo ADSR, hemos dado un paso más en el camino de lo particular a lo general desarrollando un concepto más amplio, el de la *dirección discriminante para el encaminamiento*, que presentamos en este trabajo.

3. Dirección discriminante para el encaminamiento

En los años noventa se estableció la dicotomía entre la dirección *identificador* (que distingue un equipo de otro) y la dirección *localizador* (que aporta la ubicación física de un nodo dentro de la red). Un nodo móvil mantendrá su identificador, pero obtendrá un localizador distinto en cada punto de la red en que se encuentre.

El propósito de las redes de comunicaciones es llevar paquetes de datos a la estación adecuada, y para lograrlo, reciben, procesan, almacenan y transmiten una enorme cantidad de localizadores e identificadores. Continuamente se multiplica el número de dispositivos capaces de conectarse a una red, apareciendo nuevas categorías y aplicaciones. Esto provoca una explosión en el número de dispositivos que potencialmente podrían comunicarse, lo que demanda un gran aumento del tamaño de identificadores y localizadores, que exige un incremento de los recursos que deben destinarse a manejar identificadores y localizadores, lo que podemos considerar *datos auxiliares*.

En los protocolos de encaminamiento en origen (*source routing*), la carga principal de estos datos auxiliares cae sobre los datagramas, mientras que en los protocolos *salto a salto* (*hop by hop routing*), son las estaciones las que soportan esta carga. El esfuerzo requerido para gestionar datos auxilares puede llegar a ahogar a los sistemas en el caso de que los datagramas o los nodos tengan capacidades modestas.

Para que esto no suceda, la solución típica es limitar el número de estaciones que forman parte de las redes de recursos limitados, de tal forma que constituyan diferentes sistemas aislados: menos estaciones potencialmente partícipes en la red permite menor tamaño de los identificadores y localizadores. Así, lo habitual es que las redes Ad-Hoc de dispositivos de recursos limitados empleen esquemas de direccionamiento particulares de uno o dos bytes, y no direcciones universales.

Las ventajas de un identificador universal son evidentes, mientras que un identificador de uno o dos bytes está lejos de serlo. Para la pregunta *¿Cuál puede ser un espacio de direccionamiento común y universal?* hay una respuesta muy clara: IPv6. Mediante direcciones IPv6 podríamos asignar un identificador único y universal a cualquier dispositivo que deseemos integrar en una red de comunicaciones; se puede comparar el número de direcciones disponibles en IPv6 con el número de moléculas de la superficie de la tierra [20].

Las direcciones de hasta dos bytes forman *islas* con direccionamientos particulares, tradicionalmente esto se considera una opción aceptable: en primer lugar porque no se conoce otra alternativa, y además, porque en general no es necesario ni deseable una red única, absolutamente global y *orwelliana* donde cualquier nodo se comunique continuamente con cualquier otro; por el contrario, el grueso del tráfico circulará entre dispositivos de un mismo sistema.

Con frecuencia estos sistemas de redes Ad-Hoc son completamente ajenos a Internet, aunque en algunas ocasiones están integrados en la red de redes. Naturalmente, la integración se produce cuando hay una pasarela disponible; el caso contrario no debe ser obstáculo para que la red Ad-Hoc siga funcionando.

Pero estas islas plantean serios inconvenientes si buscamos la *computación ubicua*, ya que entonces querremos que cierto dispositivo de cierto sistema interaccione con otro, de un sistema completamente diferente, pero con el que coincide en el mismo lugar durante cierto tiempo, siendo esta coincidencia difícil de prever. Para lograr este objetivo se pueden implementar diversas *pasarelas* (*gateways*) entre los

sistemas, cuyo funcionamiento no es ni mucho menos trivial.

Una pasarela deberá realizar la traducción entre direcciones de dos (o más) sistemas. Tal vez la pasarela deba también coordinarse con otras pasarelas para mantener la identidad de las estaciones con independencia del punto donde se conecten en cada momento. Debe haber también una entidad encargada de proporcionar las direcciones particulares, garantizando su unicidad, lo que supone la exigencia de una nueva infraestructura.

Este es el enfoque predominante, pero en nuestro trabajo desarrollamos una aproximación diferente, que nos parece ventajosa. Consiste en mantener un direccionamiento universal común, que evita o simplifica enormemente estos mecanismos añadidos y hace innecesario traducir direcciones: diferentes protocolos pueden usar una misma jerarquía de identificadores y localizadores.

¿Cómo conseguirlo sin que el tamaño de los identificadores genere unos datos auxiliares de dimensiones inabordables? Nuestra tesis es que además de direcciones *identificador* y direcciones *localizador*, para el uso interno de los algoritmos de encaminamiento podemos usar direcciones *discriminante para el encaminamiento* (*selective addresses for routing*). Las estaciones pueden mantener su identificador universal único (y muy pesado) si los datos auxiliares que manejan los algoritmos emplean un *discriminante para el encaminamiento* no universal, no único y por tanto mucho más ligero. En la sección 4 proporcionaremos un acercamiento intuitivo a esta idea.

Eso sí, los algoritmos deben rediseñarse o adaptarse para tolerar la no unicidad. Esto es obligado en los algoritmos del nivel de red y del nivel de enlace, en niveles superiores no es imprescindible pero puede ser conveniente considerarlo para mejorar el rendimiento.

Todo esto permite que recursos que se consumían tratando datos auxiliares pasen a emplearse en datos verdaderamente útiles.

En resumen, apreciamos una tendencia clara a conectar a Internet todo tipo de equipos, grandes o pequeños, y nos unimos a las voces que afirman que no tiene sentido integrar todos los dispositivos mundiales en una única *red de redes* mastodóntica; antes al contrario, se debe potenciar la autonomía de las redes.

Nuestra propuesta es consistente con esta idea. Nótese que hablamos estrictamente de las direcciones IPv6, y no del resto de elementos asociados a este protocolo, tales como algoritmos, infraestucturas, etc: usar *siempre* el direccionamiento IP facilita mucho el poder conectarse *eventualmente* a Internet.

De esta forma:

1. Se elimina la necesidad de traducir direcciones. O dicho de otro modo, para proporcionar conectividad con Internet bastará un *router*, no siendo preciso un *gateway*.
2. Cada dispositivo cuenta con un identificador universal único.
3. No será necesario contar con una entidad propia que asigne direcciones únicas, basta el organismo correspondiente de Internet.
4. Se facilita la reutilización del código existente basado en IP y se pueden desarrollar aplicaciones más portables para redes Ad-Hoc.

La contrapartida es que presenta dificultades que exigen desarrollar nuevas técnicas. Mantener el direccionamiento de IP, que no está pensado para redes Ad-Hoc, puede parecer un lastre, pero a nadie se le escapa que la compatibilidad con la tecnología precedente es un factor determinante, no solo en comunicaciones.

4. Aproximación intuitiva al discriminante para el encaminamiento

En el mundo físico el direccionamiento debe ser preciso y unívoco: si somos turistas en una ciudad desconocida, al llegar a un cruce debemos saber qué dirección tomar para llegar a nuestro destino. Esta dirección solo puede ser una, aún si no conocemos el camino exacto y estamos haciendo una búsqueda.

En transmisión de datos, el direccionamiento no siempre es preciso y unívoco. Un ejemplo son los algoritmos de *cotilleo aleatorio* (*Randomized Gossip Algorithms* [6]), otro, los algoritmos de inundación.

En los algoritmos de *cotilleo aleatorio*, como consecuencia de las limitaciones energéticas, de capacidad de proceso y de capacidad de comunicación, cada nodo se comunica (normalmente) con un único vecino elegido de forma aleatoria. El propósito no es alcanzar un punto concreto sino la *agregación*: proporcionar a los componentes de un sistema distribuido acceso a información global, como tamaño, carga media, localización de puntos de interés, etc. Volviendo al ejemplo anterior, sería como dejar un grupo de turistas en una ciudad desconocida para ellos y permitirles deambular al azar por sus calles. Al final del día les recogeríamos y recopilaríamos su información, experiencias, fotografías, etc. Algún turista podrá quedar perdido en la ciudad, pero esto no supone ningún inconveniente.

Los algoritmos de *inundación* llenan la red de datagramas enviados más o menos a ciegas en todas o ciertas direcciones, con el propósito de alcanzar el destino buscado. Manteniendo la metáfora anterior, el propósito de un algoritmo de encaminamiento por inundación es que un turista llegue a cierto punto en la ciudad, pero con la ventaja de tratarse de un *turista mágico*, capaz de *clonarse* en los cruces. La técnica básica consiste en que en cada esquina generemos una réplica del turista para cada calle, excepto la calle por la que llega. Existen muchas formas de optimizar y acotar la inundación, pero suele resultar muy costosa, con lo que normalmente solo se emplea como mecanismo de transición hasta que se dispone de un direccionamiento único y determinista.

Supongamos ahora que queremos orientar a un turista para que se desplace sobre el plano de una ciudad (fig. 2) de forma determinista. Una forma de hacerlo es indicarle, para cada esquina que va a encontrarse, por qué calle debe ir. Así, para que vaya desde la Plaza de Cascorro hasta la calle del Oso, le anotaríamos en un papel la ruta de la forma $R_1 =$ (*embajadores, embajadores, embajadores, embajadores, oso*). Esto es esencialmente lo que hace el encaminamiento en origen convencional [1].

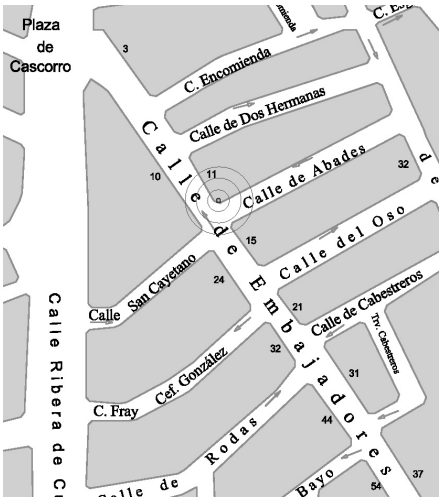

Figura 2: Ejemplo de encaminamiento sobre un plano

Introduzcamos ahora la premisa de que resulta imprescindible reducir el tamaño de lo anotado en el trozo de papel. Empleando identificadores discriminantes para el encaminamiento, podríamos emplear rutas abreviadas. Por ejemplo podríamos abreviarlo no apuntando el nombre completo de la calle, sino las dos últimas letras; excepto la calle final, que figura con todas sus letras. Abreviando de este modo la ruta anterior resultaría $Abb(R_1) =$ (*es, es, es, es, oso*).

Esto especifica una ruta sin ambigüedades en la inmensa mayoría de los casos. Para decidir en cada esquina qué calle tomar, no es necesario saber el nombre completo de la calle, basta con un discriminante que permita elegir una y descartar el resto. A menos que tengamos la mala suerte de que en una intersección coincidan dos calles con la misma dirección abreviada. Por ejemplo en Madrid esto es infrecuente, pero en la figura 2 hemos conseguido localizar un caso: la calle Embajadores y la calle Abades hacen esquina. Nótese que si dos calles tienen la misma dirección abreviada, pero no son contiguas, no supone ningún problema.

En el mundo físico y con rutas abreviadas el turista no tendría forma de saber si debe ir por la calle Abades o por la calle de Embajadores, puesto que ambas acaban en *es*. Pero en el *mundo telemático* bastaría con replicar el datagrama, o usando la metáfora anterior, el *turista mágico* se clonaría en la esquina, de tal forma que una *copia legítima* iría por la calle Embajadores y alcanzaría su destino, mientras que una *copia espuria* tomaría la calle Abades y se perdería, causando ciertas complicaciones: los mensajes de error emitidos por la *copia espuria* pueden provocar que rutas correctas se consideren inválidas, así que es necesario ignorarlos, dotando al protocolo de un mecanismo de *filtrado de falsos errores*.

El encaminamiento que proponemos no es preciso y unívoco, como tampoco lo son el encaminamiento por inundación y los algoritmos de tipo *gossip*. Al igual que en el caso de la inundación, el objetivo es alcanzar un punto concreto, no ser óptimo. Asímismo, los datagramas se duplicarán, esto es, el turista es capaz de clonarse en las esquinas. La ventaja frente a la inundación es que ya no se trata de que en todas las esquina se genere una instancia de turista mágico por cada calle diferente de la calle de origen. En nuestra aproximación, el turista solo deberá replicarse en algunas calles muy concretas, aquellas que siendo esquina compartan la dirección abreviada.

En cierta forma los algoritmos basados en *dirección discriminante para el encaminamiento* pueden verse como una mejora de la inundación ciega, aunque consideramos que está mucho más cerca de una *degradación* del encaminamiento unívoco convencional. Al ser una degradación, en los sistemas ordinarios no conseguirá mejora alguna, sino todo lo contrario.

Degradar es *reducir o desgastar cualidades*, y un objetivo importante será reducir el impacto negativo de esta degradación, tanto como nuestra habilidad en el diseño de nuevos algoritmos nos permita. Pero *degradar* también significa *transformar una sustancia compleja en otra de constitución más sencilla*. Esta constitución más sencilla es nuestra propuesta, que hará viable en máquinas sencillas comunicaciones de datos que de otro modo serían impensables.

5. ADSR

En este apartado describiremos el protocolo que hemos desarrollado, *Abbreviated Dynamic Source Routing* o ADSR, que emplea *dirección discriminante para el encaminamiento*, demostrando su validez. De este trabajo presentamos ya versiones preliminares en [14] y [15]. Hemos seguido trabajando en él desde entonces, desarrollando nuevos algoritmos especialmente en lo relativo a la fiabilidad en las capas inferiores. En la actualidad contamos con una versión mucho más madura, cuyas características fundamentales describimos aquí.

El principal reto de ADSR es tolerar que dos nodos diferentes compartan la misma dirección abreviada, lo que denominaremos *colisión de direcciones*.

Si intentamos usar el protocolo DSR con direcciones IPv4 en una arquitectura con un datagrama modesto, por ejemplo de 256 bytes como el de

[1] Hay algunas diferencias derivadas de que un callejero es un grafo donde los nombres se asignan a los arcos, mientras que en una red de comunicaciones se asignan a los nodos. Pero como aproximación intuitiva, este modelo es válido.

LNP, una buena parte de este estará ocupado por las cabeceras. La longitud de las cabeceras es variable: tomando como referencia la implementación de DSR bajo IPv4 disponible para el simulador de redes *ns-2* [9] serían necesarios 88 bytes, un tercio del total disponible.

Si DSR se usase bajo IPv6 se requerirían 288 bytes, lo que resultaría inviable con la arquitectura que proponemos como ejemplo. En estas misma condiciones, el protocolo que presentamos precisaría de 45 y 65 bytes respectivamente (figura 1).

El objetivo de este ahorro no es intentar apurar unos bits para mejorar cierta métrica unas décimas, se trata de permitir el uso de aproximaciones completamente diferentes, por ejemplo el direccionamiento IPv4 o IPv6 en máquinas que de otro modo emplearían un espacio de direcciones de uno o dos octetos.

ADSR surge como una modificación de DSR donde cada ruta no contiene la dirección de los nodos que la componen, sino que emplea *direcciones abreviadas*: una dirección construida a partir de la original, pero con un tamaño menor o igual.

Esto supone romper la idea de una dirección que necesariamente identifique de forma única a una estación, podrá haber dos o más máquinas diferentes con la misma *dirección abreviada* o *dirección discriminante para el encaminamiento*. A este hecho le denominamos *colisión de direcciones*, donde los *hosts* implicados son los *sinónimos*. ADSR no intenta evitar estas colisiones, sino que las tolera. Podemos considerar que supone la aplicación de técnicas de *hashing* [13] sobre las direcciones de los nodos, o también, en cierta forma, un algoritmo de *compresión con pérdida* sobre las rutas.

Si R es una ruta convencional como las que usa DSR, podremos reducir su tamaño con cualquier función de abreviación de rutas $Abb()$ que satisfaga lo siguiente:

1. Dadas una ruta convencional cualquiera:

   $$R_1 = (D_1, D_2, ...D_n)$$

   y su ruta abreviada

   $$Abb(R_1) = (d_1, d_2, ..., d_n)$$

   Debe cumplirse:

   $$\forall i, 1 \leq i \leq n$$
   $$size(d_i) \leq size(D_i)$$

   donde $size(d)$ es el tamaño en bytes de una dirección. Según esta definición, *abreviar* significa hacer que el tamaño de la ruta sea menor, o en algunos casos, igual. (Obsérvese que las letras mayúsculas denotan direcciones ordinarias, y las minúsculas, direcciones abreviadas).

2. Dadas dos rutas convencionales cualquiera:

   $$R_1 = (D_1, D_2, ...D_n)$$
   $$R_2 = (E_1, E_2, ...E_m)$$

   Sean sus rutas abreviadas

   $$Abb(R_1) = (d_1, d_2, ...d_n)$$
   $$Abb(R_2) = (e_1, e_2, ...e_m)$$

   Debe cumplirse:

   $$d_i = e_j \wedge D_i \neq E_j \Rightarrow$$
   $$i < n \wedge j < m$$

   Esto es, si dos direcciones colisionan, no son las últimas de una ruta. O en otras palabras, la última dirección de cada ruta se construye de forma que no se produzcan colisiones.

   También se puede aceptar la posibilidad de una colisión en el destino, con tal de que su probabilidad sea despreciable.

El propósito de esta segunda condición no es tanto evitar que un nodo reciba paquetes que no le corresponden, puesto que el nivel de red superior lo percibiría y podría eliminarlos, como impedir que un nodo crea disponer de una ruta para determinada máquina, cuando en realidad lleva a otra, cuya dirección colisiona con la deseada.

La función de abreviación de rutas $Abb(R)$ es el resultado de aplicar a cada dirección de la ruta la función de abreviación de direcciones $abb(D)$. Es decir:

$$R = (D_1, ...D_n)$$
$$Abb(R) = (abb(D_1), ...abb(d_n)) = (d_1, ...d_n)$$

$abb(D)$ la descomponemos en dos funciones de abreviación $f(D_i)$, $g(D_i)$, se aplicará una u otra según el valor de i

$$abb(D_i) = \begin{cases} f(D_i), & \text{para } i < n \\ g(D_i), & \text{para } i = n \end{cases}$$

Para $f(D_i)$ no intentamos buscar una función cuya probabilidad de colisión sea nula: sería tanto como decir que buscamos el *hasing perfecto* [13], que tiene un coste computacional elevadísimo. Además exigiría conocer las claves sobre las que se aplica en el momento de definir la función hash, lo que es inviable, esto implicaría conocer en todo momento las direcciones de todas las estaciones en la red. Y aún consiguiéndolo, con direcciones abreviadas de 1 byte estaríamos limitando el tamaño de la red a 255 nodos.

Para permitir que una máquina de recursos muy limitados pueda calcularla con poco esfuerzo, la función de abreviación de rutas $Abb()$ que proponemos es muy sencilla:

- $f(D_i)$ será el último byte de D_i
- $g(D_i) = D_i$

A partir de estos principios, ADSR modifica el protocolo DSR lo mínimo necesario para permitir su funcionamiento con este tipo de rutas, lo que incluye: almacenar direcciones abreviadas en la fase de construcción de rutas, manejar múltiples destinatarios en el nivel de enlace, no inferir rutas parciales a partir de rutas completas, no invertir ni simplificar rutas, modificar los controles de inundación e inhabilitar la recepción promiscua de mensajes de error.

Detallar estas modificaciones excede el alcance de este artículo, remitimos al lector a [14] y [15].

Asimismo hemos analizado los diferentes tipos de colisión, que quedan clasificados en: *colisión indiferente de tipo 1*, *colisión en destinatario*, *colisión distante*, *colisión indiferente de tipo 2* y *colisión adyacente*.

La tabla 1 resume los tipos de colisión y sus consecuencias. Las colisiones inofensivas no requieren ningún tratamiento especial, las colisiones perjudiciales necesitan de mecanismos como el del filtrado de falsos errores de ruta para eliminar su impacto negativo.

Nombre	Lugar de la colisión	Perjuicio
Indiferente tipo 1	un nodo ajeno a la ruta	ninguno
En destinatario	último nodo	ninguno
Distante	dos nodos no contiguos	ninguno
Indiferente tipo 2	dos nodos contiguos	ninguno
Adyacente	dos nodos contiguos	sí

Cuadro 1: Clasificación de las colisiones en ADSR

6. Experimentación

Para validar el protocolo ADSR propuesto, hemos desarrollado una implementación sobre el simulador de red *ns-2* [9]. El código fuente está disponible bajo licencia GPL [10] en `http://gsyc.es/~mortuno/adsr.tgz`. Esta versión se ha realizado a partir de las implementaciones de DSR e IEEE 802.11 con las que cuenta *ns-2*, que hemos modificado para que incorporen nuestros protocolos. A modo de resumen mostraremos aquí alguno de los resultados que consideremos más significativos.

El rendimiento de ADSR en una máquina de recursos limitados será sin duda menor que el de DSR sobre una arquitectura sin estas restricciones: uno de nuestros objetivos en la experimentación es determinar en qué medida. La configuración de la red, la carga de trabajo y los escenarios sobre los que cabría usar el protocolo ADSR pueden ser extremadamente diversos. Además, los resultados dependen mucho de las condiciones iniciales, y son muy variables en función de estos. Para nuestro trabajo hemos tomado la misma configuración y los mismos escenarios empleados por Broch *et al* [7] en su comparativa del redimiento de varios protocolos para redes Ad-Hoc.

El protocolo ADSR soporta colisiones en las direcciones de sus nodos por lo que la configuración de los experimentos añade un nuevo parámetro: el *patrón de direcciones*. Este patrón está constituido por la dirección de cada nodo y la función de abreviación de direcciones *abb*(). Su característica más importante es el valor NUA (*Not Unique Addresses*, direcciones no únicas), que es el porcentaje de nodos de un escenario cuya dirección abreviada no es única, esto es, el porcentaje de nodos sobre cuyas direcciones se producen colisiones.

Sobre un escenario de 1500x300 metros, durante un tiempo simulado de 900 segundos se situarán arbitrariamente 50 nodos, de los cuales 10, 20 o 30 simultáneamente establecerán conexiones a una velocidad de transmisión constante de 4 paquetes de 88 bytes por segundo. El alcance de la transmisión de cada uno de ellos es de 250 metros.

El patrón de movimiento usado es el modelo sintético *Random Waypoint* (o RWP), que es el más extendido en la simulaciones de redes Ad-Hoc. Es sin duda un clásico, aunque se le han encontrado algunos inconvenientes [22] y para el que hay alternativas recientes más avanzadas [8], [5]. Una máquina siguiendo el *Random Waypoint* se mueve en *zigzag* entre varios puntos de una superficie rectangular. En cada posición se detiene durante un intervalo de tiempo denominado *pause time*. Este valor es uno de los parámetros principales de los escenarios, puede tomar cualquier valor entre cero (el movimiento de los nodos es continuo) y la duración de la simulación (los nodos permanecen estáticos).

En nuestras representaciones gráficas llevaremos *pause time* al eje de abscisas, con valores de 0, 30, 60, 120, 300, 600 y 900 segundos. La velocidad se modela como una distribución aleatoria, con media de 1 m/s. La mayoría de las gráficas representan por tanto 210 simulaciones: combinan 3 valores posibles para el número de conexiones simultáneas, 7 valores de *pause time* y 10 escenarios diferentes con la misma configuración (la componente aleatoria es alta, por lo que cada punto es la media aritmética de 10 repeticiones).

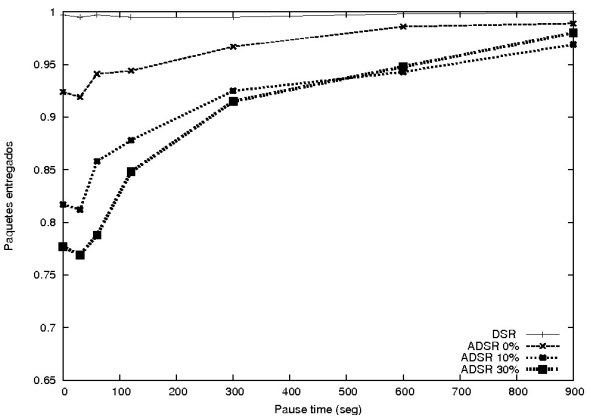

Figura 3: Ratio de paquetes entregados, 30 conexiones

En la figura 3 se representa el tanto por uno de paquetes entregados a su destinatario, tanto por DSR como por ADSR. Nuestros resultados referentes al protocolo DSR coinciden con los de los experimentos originales de Boch, lo que los valida parcialmente. Como era de esperar, los mejores resultados corresponden a DSR, que no tiene ninguna de las limitaciones impuestas a ADSR.

En la línea correspondiente a *ADSR 0 %* representamos el máximo rendimiento de la implementación actual de ADSR, con escenarios sin colisiones. En los escenarios de mayor movilidad el ratio de paquetes entregados es del 92 %, mejorando hasta el 98 % con el aumento en la estabilidad de la red. La aparición de colisiones vuelve a decrementar el ren-

dimiento, sin que haya una diferencia muy notable en los resultados con un 10 % y un 30 % de colisiones. Con movilidad alta el ratio de entrega ronda el 80 %, a partir de un *pause time* de 300 los valores superan el 90 %.

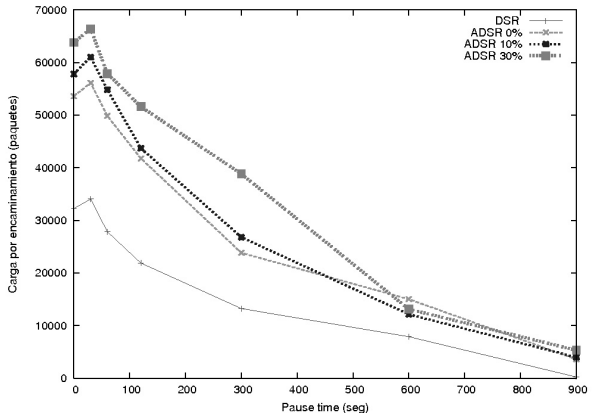

Figura 4: Carga provocada por el encaminamiento, 30 conexiones

La figura 4 muestra la evolución del parámetro *routing overhead*: el número de paquetes empleados en la simulación para el descubrimiento y mantenimiento de las rutas.

Las variaciones en el porcentaje de direcciones no únicas no provocan cambios demasiado importantes en el número de paquetes generados por el protocolo, lo que nos ratifica en la validez de nuestra propuesta. El mayor incremento de paquetes lo provoca el paso desde DSR hasta ADSR, aunque no haya colisiones. Recordemos que las características de ADSR permiten el uso de direcciones abreviadas a costa de dejar de aplicar muchas de las optimizaciones de DSR.

Como no podía ser de otro modo, los resultados que ofrecemos de ADSR son peores que los del protocolo DSR original: estas gráficas presentan el rendimiento de ADSR sobre una máquina limitada frente al de DSR sobre una máquina sin tales restricciones.

Deben tenerse en cuenta las siguientes consideraciones, que presentamos en orden de importancia creciente:

Primero: Para permitir la comparación, las simulaciones están hechas con la misma velocidad de transmisión del trabajo original de Broch. Para máquinas más sencillas sería razonable aplicar condiciones menos exigentes, lo que mejoraría el ratio de paquetes entregados.

Segundo: DSR fue evaluado sobre un protocolo de enlace muy estable y maduro como es 802.11. La implementación que ofrecemos de ADSR es una primera versión experimental, ejecutándose sobre un protocolo de enlace desarrollado por nosotros, LLRB [16], del que no tenemos espacio para hablar aquí. Aún así estos resultados son similares o algo mejores a los de los primeros protocolos de encaminamiento para redes Ad-Hoc, como TORA y DSDV [7].

Tercero: La idea más importante y aspecto clave para poder interpretar estos resultados, es que la razón de ser del protocolo ADSR es su uso en sistemas de recursos limitados, donde las técnicas de protocolos convencionales como DSR y 802.11 *no caben*. Si las máquinas son de menores prestaciones sin duda el rendimiento será inferior, pero teniendo en cuenta que estamos aplicando técnicas nuevas donde las técnicas clásicas son inviables, la mejora es sustancial. En cierta forma el incremento es $+\infty$, ya que sobre las máquinas que ejecutan ADSR, los resultados de DSR serían una lína plana de valor 0.

7. Conclusiones

Hemos introducido un nuevo tipo de encaminamiento basado en el concepto de *direcciones discriminante*. Esto permite emplear direcciones de gran tamaño, como IPv6, en arquitecturas de recursos limitados donde habitualmente se usan direciones de uno o dos octetos. La clave es que los nodos mantengan identificadores con direcciones *grandes*, pero que la *mecánica* del encaminamiento emplee identificadores *pequeños*. La reducción del tamaño de las direcciones genera duplicidad, pero esta será tolerada por los algoritmos adecuados. El encaminamiento resultante no es completamente determinista, al igual que tampoco lo son los algoritmos de tipo *Randomized Gossip* y los algoritmos de inundación.

Hemos desarrollado ADSR (*Abbreviated Dynamic Source Routing*), un protocolo de encaminamiento en origen para redes *Ad-Hoc* que emplea esta técnica con resultados satisfactorios, demostrando su viabilidad.

El empleo de *identificadores discriminante para el encaminamiento*, fuerza a identificar y descartar muchas optimizaciones comunes en los algoritmos tradicionales que dejan de ser aplicables en este caso. Además, es necesario detectar y eliminar los efectos provocados por la falta de precisión en el encaminamiento con direcciones discriminante. Para ADSR la duplicidad de una dirección abreviada solo es relevante en nodos adyacentes, en este caso la consecuencia es la duplicidad de tramas: unas serán *legítimas* y otras *espurias*, estas últimas podrán provocar problemas en forma de tramas de control erróneas, si bien es un fenómeno poco frecuente que podrá ser detectados y suprimido con los mecanismos adecuados.

La evaluación experimental del protocolo ADSR ofrece resultados satisfactorios, puesto que con una reducción importante de los requerimientos exigidos a los dispositivos de comunicaciones, la pérdida de rendimiento es asumible.

8. Trabajo futuro

Consideramos que las ideas aquí aportadas son innovadoras en muchos aspectos, por lo que se abren muchas posibles líneas de trabajo, como son:

- Considerar el uso de ADSR en escenarios de topologías más complejas, como escenarios con

mayor número de estaciones, mayor movilidad o nodos heterogéneos en capacidad de almacenamiento y alcance de su transmisión.

- Aplicar las direcciones *discriminante para el encaminamiento* a otros protocolos para MANETs además de ADSR. Sería especialmente interesante una variante de AODV [17] con direcciones discriminante.
- Implementar ADSR sobre arquitecturas reales, no solo sobre un simulador de red, así como adaptarlo para IEEE 802.15.4/(*ZigBee*) [11] [23].
- Evaluar el impacto de estas técnicas en el rendimiento del nivel de transporte y superiores.
- Aplicar esta aproximación al ámbito de las redes de sensores. Una característica deseable en este tipo de redes es el direccionamiento basado en atributos, que en la actualidad no suele aplicarse por generar direcciones de gran longitud. Este obstáculo podría obviarse mediante direcciones discriminante para el encaminamiento.

Referencias

[1] Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. A survey on sensor networks. *IEEE Communications Magazine 8*, 40 (2002), 102–114.

[2] Al-Karaki, J., and Kamal, A. Routing techniques in wireless sensor networks: a survey. *IEEE Wireles Communications 11* (Dec. 2004), 6–28.

[3] Barrera, P., Robles, G., Cañas, J. M., Martín, F., and Matellán, V. Impact of libre software tools and methods in the robotics field. *SIGSOFT Softw. Eng. Notes 30*, 4 (2005), 1–6.

[4] Bergbreiter, S., and Pister, K. Cotsbots: An off-the-shelf platform for distributed robotics. *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems* (October 2003).

[5] Boudec, J. L., and Vojnovi, M. Perfect simulation and stationarity of a class of mobility models. In *Proceedings of IEEE INFOCOM 2005* (2005).

[6] Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. Randomized gossip algorithms. *IEEE/ACM Trans. Netw. 14*, SI (2006), 2508–2530.

[7] Broch, J., Maltz, D. A., Johnson, D. B., Hu, Y.-C., and Jetcheva, J. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Mobile Computing and Networking* (1998), (ACM MOBICOM'98), pp. 85–97.

[8] Camp, T., Boleng, J., and Davies, V. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications 2*, 5 (2002), 483–502.

[9] Fall, K., and Varadhan, K. The ns manual. http://www.isi.edu/nsnam/ns/doc. UC Berkeley and Xerox PARC.

[10] Free Software Fundation. Gnu general public license. http://www.gnu.org/copyleft/gpl.html.

[11] IEEE Standars Association. 802.15.4 - 2003 IEEE Standard for Information Technology, 2003.

[12] Johnson, D., Maltz, D., and Broch, J. *DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks.* Addison-Wesley, 2001, ch. 5, pp. 139–172.

[13] Lewis, T. G., and Cook, C. R. Hashing for dynamic and static internal tables. *IEEE Computer 21* (1988), 45–56.

[14] Ortuño, M. A., Matellán, V., Rodero, L., and Centeno, J. Abbreviated dynamic source routing: Protocolo DSR abreviado para máquinas con pocos recursos. In *Actas de las IV Jornadas de Ingeniería Telemática* (2003), pp. 385–391.

[15] Ortuño, M. A., Matellán, V., Rodero, L., and Robles, G. Abbreviated Dynamic Source Routing: Source routing with non-unique network identifiers. In *Proceedings of WONS 2005. Second Annual Conference on Wireless On-demand Network Systems and Services. IEEE Computer Society* (2005), pp. 76–82.

[16] Ortuño-Pérez, M. A. *Protocolo de encaminamiento en origen con identificadores no únicos para redes Ad-Hoc de dispositivos con recursos limitados.* PhD thesis, Universidad Rey Juan Carlos, Madrid, 2006.

[17] Perkins, C. Ad hoc on demand distance vector routing. citeseer.nj.nec.com/article/perkins99ad.html, 1997. IETF Internet Draft, work in progress.

[18] Perkins, C. E. *Ad Hoc Networking.* Addison-Wesley, 2001.

[19] Satyanarayanan, M., and et al. Pervasive computing: Vision and challenges, 2001.

[20] Tanenbaum, A. S. *Computer Networks, Fourth Edition.* Prentice Hall, 2003.

[21] Weiser, M. The computer for the 21st century. *Scientific American 265*, 3 (1991), 94–104.

[22] Yoon, J., Liu, M., and Noble, B. Random waypoint considered harmful. In *Proceedings of IEEE INFOCOM 2003* (2003), pp. 1312–1321.

[23] ZigBee Alliance. ZigBee Home Page. http://www.zigbee.org/.