Abbreviated Dynamic Source Routing: Protocolo DSR abreviado para máquinas con pocos recursos

Miguel Angel Ortuño Pérez, Vicente Matellán Olivera,
Luis Rodero Merino, José Centeno González
{mortuno,vmo,lrodero,jcenteno}@gsyc.escet.urjc.es
Grupo de Sistemas y Comunicaciones
Departamento de Informática, Estadística y Telemática
Universidad Rey Juan Carlos

Abstract *DSR is one of the best known protocols for datagram routing in ad-hoc networks. It uses source routing, so each datagram must carry the addresses of all the machines in its path. Under some circumstances, specially with the use of IPv6, headers size can become very big. In this paper we propose ADSR protocol, a modification of DSR that drastically reduces its headers size by using abbreviated addresses for the routing. This mechanism can lead to two different nodes having the same address, fact that we call a* **collision** *and analyze in this work. We will also show some results about this protocol performance obtained by simulations implemented using the ns-2 network simulator.*

1. Redes Ad-Hoc

Las redes *ad-hoc* se definen como redes de comunicaciones que se forman cuando se necesitan, compuestas por las estaciones que están en determinado momento en determinado lugar y que no precisan de ninguna infraestructura externa. Los enlaces suelen ser inalámbricos: Esta tecnología junto con la alimentación mediante baterías, además de los algoritmos adecuados permite prescindir de cableado, puntos de acceso, *routers* pre-existentes o alimentación externa [7]. Estas redes están formadas por nodos similares entre sí que cooperan, normalmente no jerarquizados donde todos son al tiempo encaminadores y estaciones finales. Al menos idealmente, tampoco deben necesitar administración por parte de ningún usuario, ni usuario normal ni súper-usuario.

Si en el nivel de enlace todos los nodos fueran visibles entre sí, el problema del encaminamiento estaría resuelto. Pero esto es muy infrecuente y probablemente conlleve un derroche de energía (emisión a gran potencia) y reducción de ancho de banda (aumento de estaciones compitiendo por el medio). En general, cada nodo será capaz de acceder por sus propios medios sólo a los nodos más próximos, confiando en la colaboración de la red para llegar a todos los demás. Actuando de buena fe todos se comportarán como encaminadores además de como productores y consumidores de paquetes.

1.1. Protocolos de Encaminamiento para redes Ad-Hoc

Desde mediados de los años 90 se han desarrollado una serie de protocolos para redes Ad-Hoc.

Actualmente destacan de forma clara dos: DSR Dynamic Source Routing Protocol [4] y AODV (Ad-hoc on-demand distance vector routing) [6]. Ambos trabajan bajo demanda. DSR hace encaminamiento en origen (*source routing*), mientras que AODV emplea tablas de enrutamiento convencionales que actualiza frecuentemente y con números de secuencia para determinar su frescura.

Para nuestro trabajo partimos de DSR. Estudios recientes [1] [2] apuntan a que DSR sobrecarga menos la red con información de enrutado y ofrece mejores resultados en situaciones de moderada y media *tensión* en la red (carga, número de nodos y movilidad). Si bien a partir de ciertos niveles, AODV puede resultar más adecuado.

1.2. Descripción del protocolo DSR

DSR es un protocolo de encaminamiento en origen compuesto de dos mecanismos que trabajan coordinadamente: *Descubrimiento de ruta* y *mantenimiento de ruta*. Trabaja *bajo demanda*, no hay ninguna operación que se realice periódicamente: Cuando el nodo emisor se mueva o cuando cambie la topología de la red, el algoritmo percibe los cambios y se adapta a ello, pero

sólo para las rutas que estén en uso.

1.2.1. Descubrimiento de Ruta (*Route Discovery*)

Supongamos una red como la de la figura 2. El nodo A dispone de alguna tecnología inalámbrica que le permite alcanzar B pero no más allá. B llega hasta A, C y E, etc. Si A quiere enviar un paquete al nodo D y no sabe por dónde encaminarlo debe hacer un descubrimiento de ruta, que a grandes rasgos podemos describir así:

- A radia una *petición de ruta hasta D* a todos sus vecinos. Si quien lo recibe no es D, reenvía la petición. Esta es la conocida técnica de encaminamiento por inundación [8]. Para controlar la inundación, cada petición tiene un identificador único, de forma que cada nodo reenvía esta petición sólo una vez.

- En cada reenvio del datagrama con la petición, el nodo añade su propia dirección, de tal forma que queda registrada la ruta por la que ha ido pasando.

- Si la petición llega a D, éste extrae la ruta del datagrama (ABCD) y se la envía a A en un *Route Reply*. Usando encaminamiento en origen, este *Route Reply* volverá por el mismo camino *deshaciendo lo andado*.

- Una vez que A conoce la ruta para D, la incluirá en todos los paquetes que le envíe, empleando de nuevo encaminamiento en origen.

1.2.2. Mantenimiento de Ruta (*Route Maintenance*)

Cada nodo se hace responsable de que el paquete que recibe llegue al siguiente nodo en la ruta. En la red de la figura 2, cuando B recibe un paquete de A, se asegura de que llegue a C. (Protocolos de enlace como IEEE-802.11 ya ofrecen este servicio, con lo que no supone esfuerzo adicional).

Así, si mientras A envía sus paquetes a D, el nodo C se mueve fuera del alcance de B, este lo descubrirá y se lo comunicará a A con un un mensaje *Route Error* indicando que esta ruta no es válida.

- El protocolo es *best effort*, en caso de que una ruta se *caiga* no se reenvía el paquete, eso lo harán, si procede, niveles superiores.

- La siguiente vez que A deba enviar un datagrama a D, si conoce un camino alternativo, lo usará. En otro caso lanzará una nueva *Route Request*.

2. Máquinas con recursos limitados

A pesar de las continuas mejoras y abaratamientos del hardware, siempre habrá dispositivos capaces de comunicación sin cables pero con pocos recursos, probablemente por restricciones en su precio o tal vez por limitaciones de ocupación del espacio radioeléctrico o de consumo de la energía de sus baterías: PDAs, electrodomésticos, juguetes, equipos industriales, etc.

Tomamos como ejemplo los equipos con los que los estudiantes de la asignatura de Robótica en nuestra universidad hacen prácticas: Lego Mindstorm RCX. Cuenta con un procesador Hitachi H8/300, ROM de 16K y RAM de 32 k. Podemos comparar sus prestaciones con las de un micro-ordenador de los años 80, el Sinclair ZX-Spectrum. Puede además comunicarse mediante infrarojos con otros RCX o con un PC. Su protocolo de comunicaciones, LegOS, usa una trama de 256 bytes. Dispositivos similares tienen tramas de este orden o incluso menores.

Si intentamos llevar el protocolo DSR sobre IPv4 a una máquina de estas características o similares, una buena parte del datagrama estará ocupado por las cabeceras. La longitud de las cabeceras es variable: tomando como referencia la implementación de DSR disponible para el simulador de redes ns-2 [3] serían necesarios 88 bytes: Un tercio del total disponible.

Si DSR se usa 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 46 y 65 bytes respectivamente (figura 1).

3. Solución Propuesta

Proponemos el protocolo denominado *Abbreviated Dynamic Source Routing*, o ADSR. Es una modificación de DSR basada en construir las rutas usando direcciones abreviadas: Cada ruta no contiene la dirección de los nodos que la componen, sino un nuevo identificador que se construye a partir de la dirección original y que tendrá tamaño menor o igual. Esto supone romper la idea de una dirección que identifique de forma única a una estación: Podrá haber más de una máquina con la misma dirección abreviada, hecho al que

denominamos **colisión**. Modificaremos el protocolo DSR para que tolere estas colisiones. Esta modificación a DSR puede verse como la aplicación de técnicas de hashing 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 abreviarla con cualquier función $Abb()$ que satisfaga lo siguiente:

1. Dadas una ruta convencional cualquiera:

   $R1 = (D_1, D_2, ...D_n)$

   y su ruta abreviada

   $Abb(R1) = (d1, d2, ..., d_n)$

   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.

2. Dadas dos rutas convencionales cualquiera:

   $R1 = (D_1, D_2, ...D_n)$

   $R2 = (E_1, E_2, ...E_m)$

   Sean sus rutas abreviadas

   $Abb(R1) = (d1, d2, ...d_n)$

   $Abb(R2) = (e1, e2, ...e_m)$

   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 plabras, la última dirección de cada ruta se construye de forma que no se produzcan colisiones (O que la probabilidad de una colisión 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.

Tras estudiar otras alternativas [1], se propone una función Abb(R) muy sencilla:

- Para $1 \leq i \leq n-1$, d_i será el último byte de D_i
- Para $i = n$, $d_i = D_i$

A partir de estos principios en ADSR se modifica el protocolo DSR lo mínimo necesario para permitir su funcionamiento con este tipo de rutas. Enumeraremos estas modificaciones posteriormente.

4. Clasificación de las colisiones

El ahorro de espacio en las cabeceras que hemos descrito en el apartado 2 supone un único problema: Lo que hemos denominado *colisiones*, dos máquinas distintas cuya dirección abreviada coincide. A continuacion analizamos y clasificamos las colisiones, describiendo el comportamiento del protocolo ADSR atendiendo al tipo de colisión.

Para mayor claridad en las figuras, en este apartado no haremos referencia a una ruta genérica $r_1 = (d_1, d_2, ..., d_{n-1}, d_n)$ sino que tomaremos una ruta concreta (la misma en todos los casos): Un nodo a buscando una ruta hasta d que será $r = a, b, c, d$.

La notación a , a' indicará dos direcciones que colisionan ($d_i = e_j$ con $D_i \neq E_j$).

- Colisión indiferente

  Obviamente, si la colisión se produce en un nodo que no es visible por *a,b,c* ni *d*, esta no afecta de ninguna manera.

- Colisión en destinatario.

  Situaciones como la representadas en la figura 3 no se darán nunca, por la segunda condición de la función $Abb()$.

- Colisión distante

  La figura 4 representa una colisión que no plantea problemas. La petición de ruta que inunda la red, tras pasar *c'* será descartada, al ser imposible alcanzar *d* desde *c'* (a menos que la petición de ruta volviese por *a* , pero esto lo impide el control de la inundación).

  Cuando el paquete sea enviado por *b* a *c*, *c'* no lo recibe y por tanto no interfiere. Llamamos a este caso *colisión distante* porque los nodos cuya dirección colisiona están distanciados en la red.

[1] Buscar el que la probabilidad de colisión sea nula es tanto como decir que buscamos hashing perfecto. El hashing perfecto tiene un coste computacional elevadísimo [5] . Además exige conocer las claves sobre las que se aplica en el momento de definir la función hash, lo que es inviable: 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.

- Colisión adyacente

  El único caso de colisión conflictivo está representado en la figura 5. La petición de ruta que inunda la red, cuando pase por *c'* no generará respuesta alguna, cuando lo haga por *c* sí, con lo que *d* devolverá una respuesta con la ruta *a,b,c,d* como en todos los casos anteriores.

  Consideremos ahora el paquete con la ruta *a,b,c,d* tras atravesar *b*: tanto *c* como *c'* lo interpretarán como dirigido a ellos, esto genera dos copias del paquete: Una *legítima* que llegará correctamente a *d*. La que atraviesa *c'* no podrá alcanzar su destino y acabará generando un mensaje *Route Error*, diremos entonces que *c'* **hace sombra** a *d*. Este es el caso peor, el mensaje de error provocará que se deje de usar una ruta correcta, si bien la ruta que atraviesa la copia legítima del paquete seguirá en funcionamiento el tiempo que transcurra hasta que se genere el error, llegue a *a* y sea procesado. Algunos paquetes tendrán oportunidad de llegar a su destino. Entonces *a* hará una nueva petición, si los nodos no cambian su posición se generarán las mismas respuestas y todo el proceso se repetirá de nuevo.

5. Características de ADSR

ADSR es esencialmente igual a DSR, excepto en todo aquello que resulta incompatible con el uso de direcciones abreviadas.

A continuación se enumeran los aspectos que difieren en ambos protocolos.

Construcción de Rutas

- DSR: En la fase de *descubrimiento básico de ruta* el paquete de petición de ruta se distribuye por inundación y va almacenando la dirección de cada nodo por el que pasa.
- ADSR: El paquete de petición de ruta almacena la dirección abreviada de cada máquina que recorre, teniendo en cuenta que el nodo final de una ruta es un caso especial que exige la ausencia de colisiones.

Múltiples destinatarios en el nivel de enlace

- DSR: Para un paquete con la ruta $R1 = (D_1, D_2, ..., D_i, D_{i+1}, ...D_n)$ que llega a D_i, alcanzar D_{i+1} es inmediato: es un envio *unicast* a una dirección conocida. Bajo DSR habrá un nivel de enlace que probablemente emplee un esquema de direccionamiento distinto, pero entre la dirección de red y la de enlace habrá una correspondendia uno a uno que podrá resolverse con técnicas como ARP o similares.
- ADSR: Dada una ruta $r_1 = (d1, d2, ..., d_i, d_{i+1}, ..., d_n)$ el datagrama debe transmitirse desde d_i hasta d_{i+1}, pero d_{i+1} no identifica de forma única un nodo, por lo que este envío debe llegar a todas las máquinas cuya dirección abreviada coincida con d_{i+1}, con tal de que sean visibles desde d_i.

  Así, cada envio desde el punto de vista del nivel de enlace se convierte potencialmente en una transmisión *multicast*. Esto en general no estará previsto, con lo que inevitablemente habrá que convertir este multicast en una de estas dos opciones

  - Varios *unicast*, lo que exige conocer las direcciones completas de todas las estaciones que deban recibir el envio.
  - Un *broadcast*. Cada receptor, una vez que haya recibido el paquete lo descarta si su dirección abreviada no coincide con la de los destinatarios. Cabe destacar que una consecuencia importante del uso de broadcast es la imposibilidad de usar asentimientos:

Rutas Parciales

- DSR: una ruta $R_1 = (D_1, D_2, ...D_n)$ indica cómo alcanzar D_n, pero también puede usarse para encaminar paquetes hasta $D_2, ...D_{n-1}$.
- ADSR: Una ruta $r_1 =$

  $(d_1, d_2, ...d_{n-1}, d_n)$ es válida sólo para alcanzar d_n. Si se requiere enviar un datagrama a D_i (con $i < n$) es necesario solicitar una nueva ruta, puesto que d_i podría corresponderse con D_i o con cualquier otro nodo cuya dirección abreviada coincida con esta.

Inversión de Rutas

- DSR: Sea una petición de ruta que llega a su destino $R_1 = (D_1, D_2, ...D_n)$ Si el nivel de enlace es bidireccional, el nodo d_n puede construir una una ruta para d_1 directamente a partir de esta: Basta invertirla $(D_n, ...D_2, D_1)$ para almacenarla en caché.

- ADSR. Si la ruta $r_1 =$

  $(d1, d2, ...d_{n-1}, d_n)$ se invierte resulta $(d_n, ..., d_2, d_1)$ que no es una ruta válida para d_1 al no satisfacer la segunda condición de Abb(), con lo que será necesaria una nueva petición de ruta.

  Aunque no resultará extraño que ADSR está bajo un protocolo de red, tal vez IP. Si se tiene acceso a la cabecera del nivel de red, es posible conocer la dirección D_1 y generar de nuevo una dirección abreviada cuya probabilidad de colisión con la dirección de otro nodo sea despreciable. De esta forma se ahorra una petición de ruta.

Control de Inundación

- DSR: La petición de ruta se extiende por inundación. Cada petición tiene un identificador. Para evitar ciclos, cuando un nodo recibe una petición de ruta, si la ha procesado previamente la descarta, en otro caso la reenvía. Para saber si debe descartarse se hacen dos comprobaciones

  - Se mantiene una caché de identificadores de peticiones tratadas recientemente.
  - Además, el nodo comprueba si su dirección está incluida en la ruta seguida por el paquete.

- ADSR: Sólo podrá aplicarse la primera de las comprobaciones descritas en el párrafo anterior: La segunda no es válida, un nodo verá su dirección abreviada en una ruta si el paquete ha visitado otro nodo cuya dirección abreviada colisione con la suya.

  Como consecuencia, en casos extremos podrían producirse ciclos (aunque siempre finitos).

Simplificación de rutas

- DSR: Cualquier ruta

  $(D_1, D_2, ..., D_i, ..., D_j, D_{j+1}..., D_n)$ con $D_i = D_j$ indica un bucle, y por tanto puede simplificarse resultando $(D_1, D_2, ..., D_i, D_{j+1}, ..., D_n)$

- ADSR:

  Una ruta

  $r1 = (d_1, d_2, ..., d_i, ..., d_j, d_{j+1}..., d_n)$ con $d_i = d_j$ no puede simplificarse, pues direcciones abrevidas iguales pueden referirse a distintos nodos.

6. Experimentación

Para analizar el rendimiento de ADSR lo implementamos sobre el simulador de redes ns-2, una herramienta muy extendida, libre y que implementa una gran cantidad de protocolos, incluyendo DSR.

Ofrecemos a continuación unos resultados preliminares. Debemos indicar que partimos de la implementación de DSR existente en ns-2 a la que hacemos las modificaciones descritas en el apartado 5. Esta implementación tiene una serie de optimizaciones que sin duda habría que suprimir al llevar el protocolo a una máquina real de las características de las citada en el apartado 2, pero eso lo obviaremos ahora: queremos comparar una implementación de DSR con una de ADSR.

La configuración de la red, la carga de trabajo y los escenarios sobre los que se podría usar el protocolo ADSR pueden ser extremadamente diversos, es difícil hablar de una configuración *típica* o unos parámetros extraídos de la *realidad*. Además, los resultados dependen mucho de las condiciones iniciales, y son muy variables en función de estos. Así que hemos tomado como punto de partida los parámetros de entrada usados por Broch *et al* [1] en su comparativa del redimiento de varios protocolos ad-hoc:

Sobre un escenario de 1500x300 metros, durante un tiempo simulado de 900 segundos se situarán arbitrariamente 50 nodos, de los cuales 30 a la vez establecerán conexiones a una velocidad constante de 100 bytes/s. Cada nodo hará una pausa antes de moverse en dirección aleatoria con velocidad aleatoria, de media 10 m/s. Esta pausa variará entre 0 (movimiento continuo) y el tiempo total de la simulación (nodos estáticos). Cada simulación se repite 10 veces con la misma configuración y se toma la media aritmética del resultado.

Los resultados se representan en la figura 6 donde tomamos como métrica el tanto por uno de paquetes entregados a su destinatario. Como era de esperar, los mejores resultados corresponden a DSR (el ahorro de espacio visto en la figura 1 tiene un precio). En esta misma figura 6 se observan los resultados de ADSR cuando forzamos el porcentaje de colisiones a un 20 %, 50 % y 100 % (valores muy altos, extremadamente poco probables en situaciones reales).

Se observa que a pesar de someterse al protocolo a situaciones de elevada *tensión* en cuanto a número de colisiones, la pérdida de rendimiento no es excesivamente acusada, obteniendo a cambio un gran ahorro de espacio en el datagrama.

7. Conclusiones

Se han mostrado unas condiciones donde es difícil o imposible aplicar el protocolo DSR. Como respuesta se propone el protocolo ADSR, que basado en el anterior reduce drásticamente el tamaño de las cabeceras; con un gran ahorro en el tamaño de los paquetes, supone una moderada pérdida de rendimiento.

Referencias

[1] 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.

[2] Das, S. R., Perkins, C. E., and Royer, E. E. Performance comparison of two on-demand routing protocols for ad hoc networks. In *Proceedings of IEEE INFOCOM - The Conference on Computer Communications* (2000), pp. 3–12.

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

[4] 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.

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

[6] Perkins, C. Ad hoc on demand distance vector routing. citeseer.nj.nec.com/article/perkins99ad.html, 1997.

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

[8] Perkins, C. E., Belding-Royer, E. M., and Das, S. R. IP flooding in ad hoc mobile networks. www.ietf.org/proceedings/01dec/I-D/draft-ietf-manet-bcast-00.txt, 2001. IETF Internet Draft.

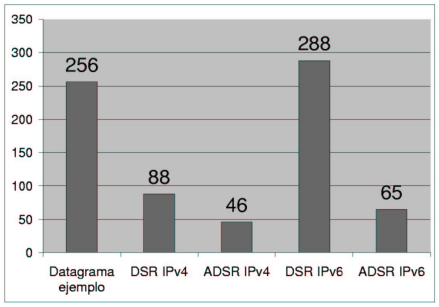

Figura 1: Comparación del tamaño del datagrama (bytes)

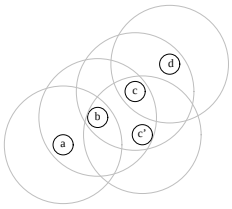

Figura 5: Colisión adyacente

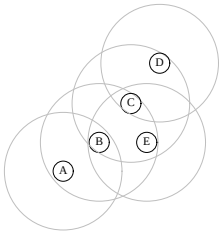

Figura 2: red ad-hoc

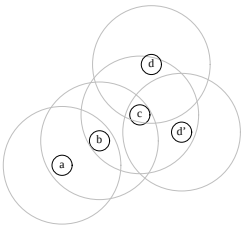

Figura 3: Colisión en destinatario

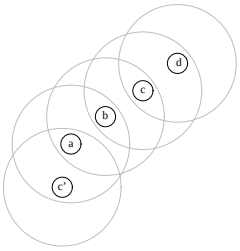

Figura 4: Colisión distante

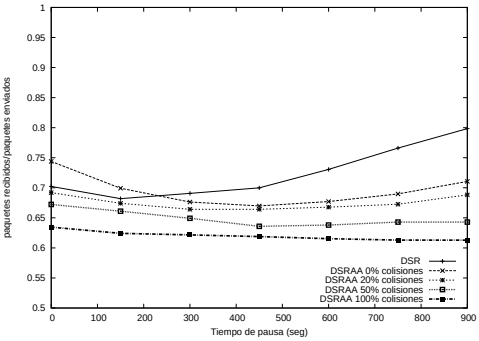

Figura 6: Paquetes Entregados: Media