

# The Dynamic Source Routing protocol for Ad Hoc Networks

Christian Niggemeyer    nigchr@upb.de

## Abstract

The Dynamic source Route Protocol is a simple, self organizing and self-configuring routing protocol designed for use in wireless Multihop Ad Hoc Networks. This paper shows the routing mechanism of this protocol, which is parted in the Route Discovery and the Route Maintenance. At the end of this paper there is also small performance analysis showing how the Dynamic Source Routing protocol acts in different environments.

## Table of Contents

1 ASSUMPTIONS.....	2
2 DSR ROUTE Discovery.....	3
3 DSR ROUTE Maintenance.....	4
4 Additional Route Discovery Features.....	4
4.1 Preventing Route Reply Storms.....	5
4.2 Caching overheard Routing Information.....	5
4.3 Replying to Route Requests Using Cached Routes.....	6
5 Additional Route Maintenance Features.....	7
5.1 Packet Salvaging.....	7
5.2 Increase Spreading of Route Error Messages.....	7
6 Efficiency.....	8
6.1 Routing Overhead with respect to movement of nodes.....	8
6.2 Scalability .....	9
7 Conclusion.....	10

## 1 ASSUMPTIONS

For the DSR Protocol there are several conditions that have to be full filled in order to route packages to our network. The first condition is that every Node that is part of the DSR network is willing to participate fully in the network protocol. Another assumption is the diameter of the network. It's assumed that the diameter of the Network is small ( about 5 to 12 Hops ), but normally grater than one. For larger diameters the efficiency of the network is falling rapidity, as elucidated at the end of this paper. As we are talking of an wireless network, we also allow movement of the nodes, but with a movement speed that is low with respect to the packet transmission latency and the transmission rate of the underlying hardware. The last condition that is assumed is that all nodes are capable of the promiscuous mode, what means that every node is able to read every data package it receives and not only the data packages that are delivered to this node (indicated by the MAC address). The promiscuous mode is not needed for DSR to work, but with the promiscuous mode there are some performance improvements possible ( see Additional Route Discovery/Maintenance features).

## 2 DSR ROUTE Discovery

The DSR protocol works complete on demand, what means that there are no periodic route updates in the network. That means that if a node (S) wants to send a package to another node (D), it first has to discover a route to node D. The first step in Route Discovery is that the Source Node (S) searches his own Route Cache for a Route to the target Node. If there's a route to the target node, the source route uses this route to send it's data package. If there isn't a route to the source node sends a Route Request Package to it's neighbours using broadcasting. This Route Request Package identifies the initiator node and the target node of the Route Discovery. Also the Package contains a unique request ID, determined by the initiator of the Route Discovery. Within the Route Request there is also a list of nodes holding the nodes the package went through that is initially empty. When this Route Request is received by another node the node first checks if it has already seen the combination of the source Node and the request ID of the Route request Package. If so, it discards the package and not processes is further. Otherwise the node checks if his address is already in the node list of the Route Request Package. If so, the package is discarded too and the node does not process this package any further. Otherwise it is checked if this node is the target node the route is searched for. If so, the node returns a Route Reply Packet to the source node of the route Request containing the route from the source to the target node given by the node list within the Route Request Package. The Route Reply Packet is sent along the same list of nodes given by the internal list of the Route Request Package. This is done because in IEEE 802.11b there are only bidirectional links possible because of the hardware layer. So we can expect that the route is also usable from the target node to the source node. If none of the conditions before are full filled, the node adds his own address to the node list of the Route Request Package and broadcasts it to his neighbours.

### 3 DSR ROUTE Maintenance

In DSR every Node is responsible for confirming that the package has been received by the next hop along the route. In standard MAC protocols this feature is given for no costs because it is part of the standard MAC protocol ( In 802.11 for example given by the link level acknowledgement ). If any node detects that a package cannot be transmitted to the next hop, because this hop is for example no longer a neighbour of the current node, it sends a Route Error Package to the source node of the package that couldn't be delivered. This Route Error package identifies the link over which the package could not be forwarded. The route to the source node can be extracted out of the hop list from this packet. When the source node receives this a Route Error package it removes the route from it's cache, and the next time the source node sends a package to the target node it uses either another route from his cache or, if there is none, initiates a new Route Request for the target node. The retransmission of lost packages, like packages that are lost due to a Route Error is not a part of the DSR protocol, it's part of the overlaying protocols such as TCP.

## **4 Additional Route Discovery Features**

Based on the basic function of the Dynamic Source Routing protocol this protocol can be optimized in several ways. Some possibilities to optimize are specified below.

### ***4.1 Preventing Route Reply Storms***

If the DSR protocols replies to Route Requests using cached routes a Route Request can result in a Route Reply Storm. For Example if a any node broadcasts a Route Request to it's neighbours, it's possible that some of his neighbours have a route to the target node in their cache. Because they all received the Route Request at the same time, they will probably try to response to this request at the same time, what leads to network collusions, congestions and waste of network bandwidth by sending several Route Response packages back. To prevent this all nodes capable of the promiscuous mode will wait for a certain delay  $d$ , given by the formula  $d = H * (h - 1 + r)$ , before responding to the Route Request. While this delay the nodes will overhear the network traffic for other Route Response packages for this Route Request and, if there is a Route Response package, discard their own Route Response package. The formula  $d = H * (h - 1 + r)$ , with  $r$  a random number between 0 and 1,  $H$  as an small constant delay and  $h$  as the length in number of network hops of the route to be returned provides not only a mechanism that prevents a Route Storm, it also provides a simple way to return shorter routes sooner than longer routes.

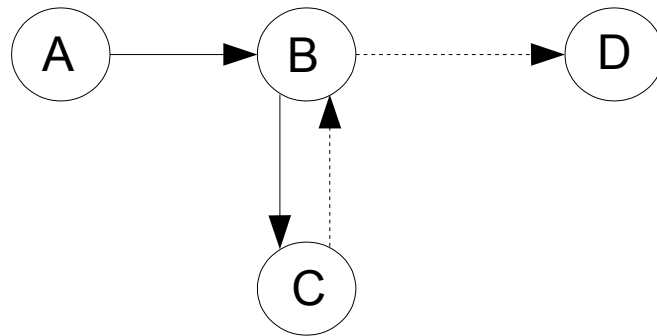
## ***4.2 Caching overheard Routing Information***

Instead of just routing packages that are not dedicated to the current node, a node can receive the routing information from these packages and update its own routing cache. This is done by crosschecking its cache with the routes given by the packages or the broken links information from Route Error packages. When updating the own route cache with routes from the forwarded packages only the route from the source to the current node is used, the route from the current node to the target node will not be used because it could possibly be a wrong route. By using this mechanism especially Route Error Message will spread a lot faster throughout the network helping the network keeping up-to-date with minimal costs. If the node is capable of the promiscuous mode the card can even overhear the network and use informations that are not going through a node.

## ***4.3 Replying to Route Requests Using Cached Routes***

A node receiving a Route Request package that is not the target node normally forwards the the route Request to its neighbours without looking in his own cache for a route to the target node. One optimization now is that every node receiving a Route Request package that is not the target node searches his own cache for a route to the destination node. If there is one, it concatenates the hop list of the Route Request package with his own cached route to the target node. The node then creates a new route Reply package with the new route. But before sending this package to the source node it has to be checked that the new route has no duplicated nodes listed in the route.

For example in the picture below node A sends a Route request for target node D.



For any reason B does not know that D is his neighbour and forwards the Route request to C. C knows a route to D and could now concatenate the Route of the Route Request ( $A \rightarrow B \rightarrow C$ ) with his cached route to D ( $C \rightarrow B \rightarrow D$ ) to the route  $A \rightarrow B \rightarrow C \rightarrow B \rightarrow D$ . C now could also delete the duplicate node B and create a route  $A \rightarrow B \rightarrow D$ , but it won't do this. C will forward the normal Route request instead of sending a new route. This is done because B could already know that there is no longer a link between B and D, while C doesn't know this. If C then would send the new route to the source node, the source would receive a invalid path and would receive another Route Error packet when using it resulting in more network load and higher delay.

## **5 Additional Route Maintenance Features**

### ***5.1 Packet Salvaging***

When a node detects a broken route during a packet forwarding, it sends a Route Error package to the source node of this package and discards the data package. Instead of discarding this package the node can try to salvage the data package and send it to the target node by another route than the route given by the data package. To do this, the node sending the Route Error searches its own cache for a route from itself to the target node of the data package. If there is one, the node forwards the data package using this route to the target node, but marks the data package as salvaged. This is needed because if a salvaged data package is not marked as salvaged other nodes might also detect a broken link and again try to salvage this package. If this happens, the data package could possibly be forwarded in a loop and never reach the target.

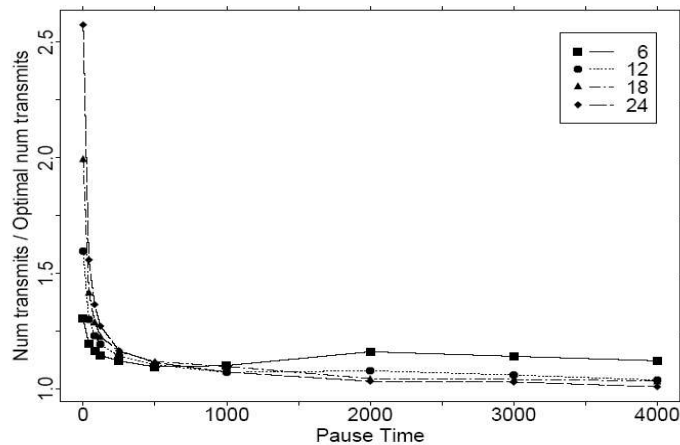
### ***5.2 Increase Spreading of Route Error Messages***

When a node receives a Route Error Package for a data package it originated, the node deletes the broken link from its cache and sends a Route Request for this route. Instead of just sending a Route Request this node could piggyback the Route Error package on the Route Request package to broadcast the broken links to the other nodes. By this the information about the broken links spreads faster through the network and saves network bandwidth by saving some Route Error messages for this particular broken link.

## 6 Efficiency

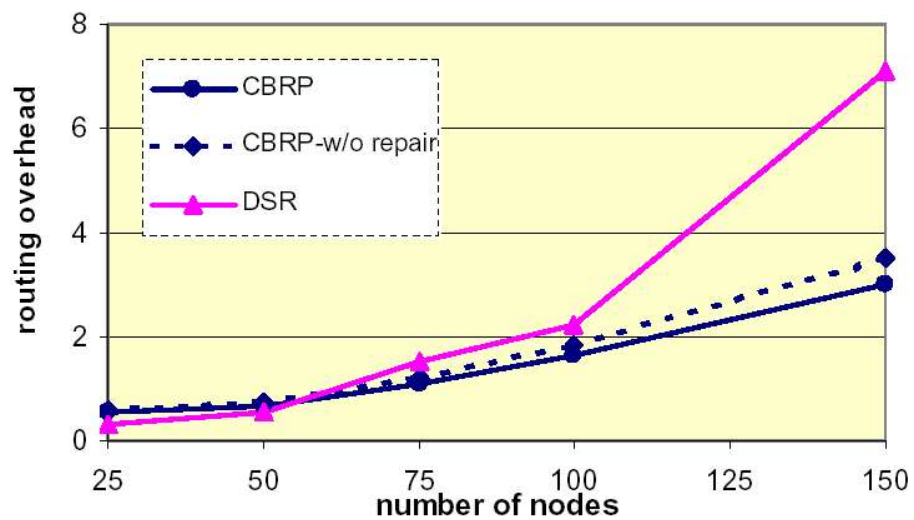
### 6.1 Routing Overhead with respect to movement of nodes

A simulation of a small DSR network by David B. Johnson and David A. Maltz shows that Dynamic Source Routing has an excellent performance for routing in small networks. In their simulations Johnson and Maltz simulated between 6 and 24 nodes over a time of 4000 seconds. In the diagram below you can see the results of this simulation, whereby pause time means the time the nodes are not moving within the network. So a pause time of 4000 means that the nodes are not moving at all, whereby a pause time of zero means that the nodes are moving constantly. The y-axis shows the factor of the number of transmits to the optimal number of transmits. For a network with all nodes moving DSR performs very bad, having an overhead of about 2.6 ! But this overhead is falling down very fast if the nodes are not moving constantly. At a pause time of about 250 (what means the nodes are moving 94 % of the simulation time) the overhead has fallen down to approximately 1.2, what means an overhead of 20%. Moving to higher pause times also reduces this overhead resulting in an even lower overhead of about 1 % at an pause time of 1000 s ( nodes are still moving 75 % of the simulation time )

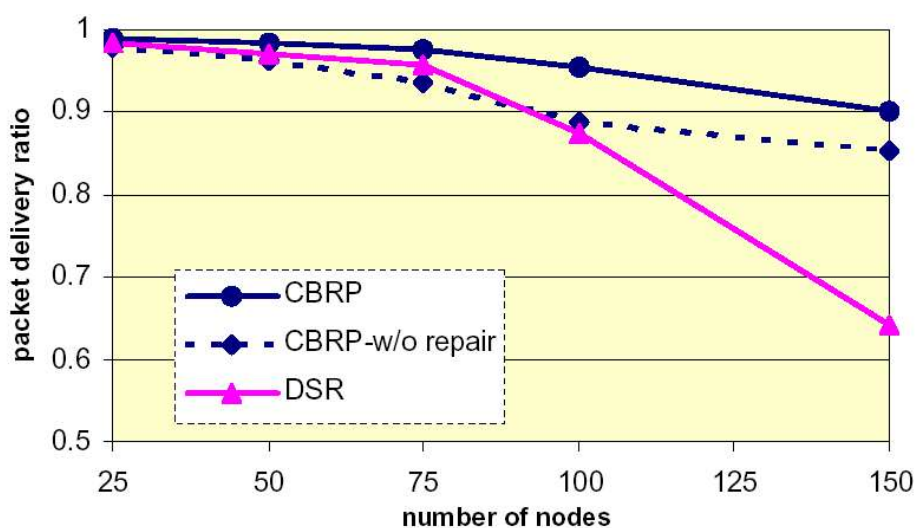


### 6.2 Scalability

As seen in the previous graph, the Dynamic Source Routing protocol works great in small networks, even when the nodes are moving about 75 % of the simulation time. But when moving to bigger networks with more than 75 nodes, the Dynamic Source Routing protocol falls back in its performance behind other protocols like the Cluster Based Routing protocol. As the graph below shows, in a network with about 125 nodes the routing overhead with the Dynamic Source Routing protocol is about twice big as the overhead of the Cluster Based Routing protocol, and it's getting even worse for bigger networks.



And it's not only the routing overhead that falls back in comparison to other routing protocols, also the package delivery ratio is falling back compared to other routing protocols. In the graph below we can again figure out that the Dynamic Source Routing protocol falls back in routing efficiency at a network size of 75 nodes compared to the Cluster Based Routing protocol.



## 7 Conclusion

Concluding it can be said that the Dynamic source Routing protocol is a very simple, reliable and efficient routing protocol for networks with a size of up to 75 – 100 nodes. For networks with more than 75 – 100 the Dynamic Source Routing protocol loses its efficiency rapidly, falling down to results that are not acceptable. For the application in PAMANET this means that the PAMANET this means that this protocol is only a good choice is the network has a size of up to 75 – 100 nodes. Alternatively we could think of an Network with several layers whereby in the lowest level there are several DSR networks where a specific node of every DSR network takes also part of an upper-level DSR network.

## **8 Literature**

### **Dynamic Source Routing in Ad Hoc Wireless Networks**

David B. Johnson David A. Maltz

Computer Science Department

Carnegie Mellon University

5000 Forbes Avenue

Pittsburgh, PA 15213-3891