DSR: Dynamic Source Routing

Pekka Savola
CSC/FUNET
Pekka.Savola@funet.fi

Abstract

The Dynamic Source Routing Protocol (DSR) is an efficient routing protocol proposed and designed especially for multi-hop Wireless Mobile Ad-hoc Networks. The protocol automatically discovers ("Route Discovery") and maintains ("Route Maintenance") the routing in the network by storing source routes, discovered dynamically only when necessary, thus requiring no administration from the network operator. All nodes in the network participate in the packet forwarding process, acting as routers on an ad-hoc basis. DSR embeds control information to normal packet flow by adding "DSR header" to regular IP packets; the header is modified constantly while the packet is being forwarded. The use of source routing guarantees loop-freeness in the network, and does not require any kind of periodic routing protocol exchanges, thus minimizing the overhead. All nodes cache source routes either when overhearing them or when forwarding the packets, reducing the need for route discoveries. DSR is a reactive protocol and is highly adaptive in the event of movement and node number changes. DSR applies to MANET's with up to 100 nodes and can cope with a reasonably high rates of node mobility. There is still further study and clarifications to be done, especially in the areas of security and reliability.

1 Introduction

The Dynamic Source Routing (DSR) [1, 2] is a routing protocol proposed for multi-hop Wireless Mobile Ad-hoc Networks. The work is being done at IETF MANET working group [3].

The protocol automatically discovers and maintains the routing in the network by storing source routes, discovered dynamically only when necessary, thus requiring no administration by the network operator. All nodes in the network participate in the packet forwarding process, acting as routers on an ad-hoc basis. This is critical in the case that both nodes A and C can reach B, but A cannot reach C (for example, due to limits in wireless transmission range) as seen in fig. 1: B will automatically act as a router between two distinct network clouds, over multiple hops. When this is applied to a network of dozens or hundreds of moving nodes, the topology, relations, wireless transmission conditions, and such may change at a very rapid pace between the nodes; self-organizing and self-configuring routing protocols are necessary. The source routes in this network are discovered and maintained by the DSR protocol.

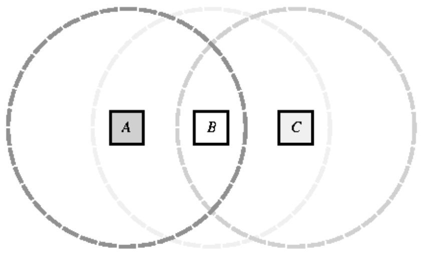

Figure 1: The simplest Ad-hoc scenario [4]

DSR embeds control information to normal packet flow by adding "DSR header" to regular IP packets, or by sending pure IP packets with only "DSR header"; the header is modified constantly while the packet is being forwarded. Source routes are dynamically discovered when needed. Packets, except Discovery packets, contain hop-by-hop source route list in the DSR header which will be used to determine which way the packets will be forwarded. In that fashion, loop-freeness will be guaranteed, as all the visited and to-be-visited nodes are listed in the source route list in each packet. Nodes forwarding or overhearing packets containing DSR header will cache the source route information, thus reducing the need for every individual node to perform route discovery itself.

The aim of DSR was to have a very low overhead which could still react quickly to changes. MANET protocols can be roughly divided in two categories, proactive and reactive. DSR is a reactive protocol, trying its best to ensure succesful data delivery in the event of changes in the network.

There are two main mechanims which form the core of DSR: Route Discovery and Route Maintenance. Route Discovery is used by a sending node S when it tries to send a packet to destination node D and a source route is not already known. Route Maintenance is used by any node in the path, while using a source route to D, to detect if the network topology has changed and the source route is no longer valid. When Route Maintenance detects an invalid route, other routes can be automatically tried, or the source can try to find a new route to the destination. Route Maintenance, like Route Discovery, is only used when packets are being sent.

These procedures operate on an on-demand basis: there are no periodic link status sensing packets or advertisements, like with traditional routing protocols. If the network topology stabilizes, the amount of control information will decrease dramatically; in a completely static network, there would be practically no DSR Route Maintenance or Route Discovery traffic. This "auto-adjusting" property may be important in some scenarios: potentially expensive bandwidth can be saved but on the other hand, the protocol still reacts quickly when the network is changing at a more rapid pace.

Nodes implement extensive caching based on overheard and forwarded source routed packets: usually a node has many cached paths to a destination, and if some paths fail, some others may still be perfectly usable without need for an instant route discovery.

DSR mechanisms work with both unidirectional and bidirectional links; however, as noted in the analysis section, there are certain problems in the protocol if all the links do not have the same directionality properties.

DSR is designed for simple IPv4 unicast traffic. More advanced features such as IPv6, multicast routing and QoS are out of the scope of the main proposal. Of these, previously Flow State, used to be able to omit DSR header from some packets, was also defined in the main proposal, but has been moved to a separate document [5].

DSR resembles older IEEE 802.5 (Token Ring) Source Route Bridging [6]. Route Requests and Replies were motivated by ARP protocol [7].

2 Problem, Motivation and Assumptions

2.1 Problem and Motivation

The applicability, usage scenarios, requirements and such are already discussed in other documents (e.g. [8]), and are as such, out of the scope; let it only be said that there are scenarios especially in the wireless networking field where MANET protocols are very useful.

The motivation of DSR is to provide a MANET routing protocol with very little overhead and very few unnecesessary data packets, but still one that can react quickly to changes in the network. Source routing provides loop-freeness but restricts the scope of the protocol to a hundred or so.

2.2 Assumptions

All nodes are assumed to participate in the packet forwarding: any node in the network can be a router.

Nodes may move without any notice; however, it is assumed that the speed is moderate with respect to the propagation delays and transmission range: if a node wishes to use some other node as a router for connectivity, it is natural to require that the first node will stay within the range at least for the duration of Route Discovery sequence.

The network is assumed not be too big, for example, the diameter could be 5-10 nodes. DSR is only applicable to a relatively small amount of nodes, e.g. less than 100 [9]; else managing the source routes to every node may become problematic.

If the nodes are able to enable "promiscuous" receive mode, also receiving packets not meant for themselves, several optimizations (such as more reliable route shortening) will be possible. However, this is not a requirement.

If no link-layer delivery acknowledgments, or in some cases promiscuous mode, are supported by the interface, Route Maintenance messages will need to use network-layer acknowledgments for reliable neighbour detection.

DSR also works in unidirectional links, but if all links are bidirectional, source route processing can be optimized by being able to reverse source routes.

DSR node, even if it has multiple interfaces, is expected to have one of its IP addresses acting as a node identifier.

As noted in the analysis section, it is also assumed that every node trusts every other node, link-layer medium has been secured against unwanted or untrusted users, that there is no packet filtering or firewalling except in the end-nodes unless DSR-specific support is implemented, that every potential router in the domain supports DSR, and other more or less implicit assumptions.

3 The DSR Protocol

First, conceptual data structures and DSR header are quickly introduced. Then basic Route Discovery and Route Maintenance procedures are discussed; after that, a few more advanced features and optimizations are noted.

3.1 Conceptual Data Structures

There are a few conceptual data structures which must be mentioned here for clarity and the sake of terminology; every node implements all of them. Detailed discussion is omitted.

Route Cache is the list of all the source routes known to the node. If the node learns of new links, it adds new data to the cache; upon the receipt or overhearing Route Errors, or after a timeout, entries are removed from the cache. There may be more than one route per destination; there is no load-sharing between the destinations, as is common with regular routing, but usually one best route is selected by some algorithm and the rest kept as backups.

Send Buffer is the queue of packets for which there is no source route yet, and are there waiting for Route Discovery to complete. Subject to rate-limiting and a back-off timer, Route Discoveries should be performed periodically for the packets still in the Send Buffer.

Route Request Table stores the list of Route Requests that have been recently forwarded or originated by this node. The table is mainly used to determine when a request could be retransmitted, or whether a request has already been here and should not be re-broadcast again.

Gratuitous Route Reply Table stores the list of Route Replies that have been sent by this node when performing automatic route shortening on behalf of some other node. As gratuitous route replies are rate-limited, the history has to be stored in a table.

Network Interface Queue and Maintenance Buffer; NIQ is the operating system's output queue for packets waiting to be over an interface, where packets are held as other packets are being sent on the interface. Maintenance Buffer is the queue of packets sent by this node, waiting for Route Maintenance procedure's next-hop neighbor reachability information. As Maintenance packets are retransmitted unless delivery is confirmed, there must be a buffer to keep the messages before the confirmation.

Blacklist keeps track of neighbors for which the bidirectionality is not yet certain. The blacklist must be kept for nodes the interfaces of which require physically bidirectional links.

3.2 DSR Header

DSR header has a fixed portion of 4 bytes ("Next header", "reserved" and "payload length") and any number of options encoded in Type-Length-Value (TLV) notation.

DSR header is an IP-level protocol just as TCP or UDP are; it is inserted directly after IP header. The protocol number in the IP header is changed to "DSR header" (to be assigned by IANA), and "next header" field in the

DSR header is updated to point to the original protocol number of the terminal header.

The following options have been defined: Route Request, Route Reply, Route Error, Acknowledgment Request, Acknowledgment, DSR Source Route, Pad1, and PadN. These are only quickly introduced here; all the gory details are omitted.

Route Request contains “Identification”, “Target Address” and “Address[1..n]” fields. Route requests will be sent to the limited broadcast address “255.255.255.255”, and the destination address where the source wants to find the route for is placed in “Target address” field. Identification is a sequence number, used to distinguish between already seen and old messages. “Address [1..n]” are used to store the addresses along the the path of Route Discovery.

Route Reply contains “L”-bit, “Reserved” and “Address[1..n]” fields. Last Hop External bit indicates that the last address represents the last node in the DSR network, and the actual destination is outside of MANET. “Address[1..n]” contains the source route gathered with in the route request. Route Replies are usually sent back to the originator of Route Request by either the node that was the Target of the original Route Request or by some intermediate node, as a Reply from Cache or as a gratuitous Reply.

Route Error contains “Error Type”, “Reserved”, “Salvage”, “Error Source Address”, “Error Destination Address”, and error type -specific information. Currently, only one error type, Node Unreachable, is specified. The contents of “Salvage” field are derived from the DSR Source Route option triggering the error. “Error Source Address” is the address of the node which encountered an error; “Error Destination Address” is the original source of the failed packet. In the case of Node Unreachable message, type-specific information contains the address of the unreachable node. Route Errors are sent to inform the source (and intermediate nodes) of failed source routes.

Acknowledgment Request contains “Identification” field. It is an unique value that will be used in Acknowledgment response to create a link between the two. Acknowledgments are used for ensuring the reliable delivery of Route Maintenance packets if no other form (e.g. link-layer acknowledgments, passive acknowledgments by promiscuous mode) is available.

Acknowledgment contains “Identification”, “ACK Source Address”, and “ACK Destination Address” fields. Identification is copied from the Request, ACK Source Address is the address of the node originating the acknowledgment, and ACK Destination Address is the address to which the acknowledgment will be delivered to.

DSR Source Route constains “F” and “L” -bits, “Reserved”, “Salvage”, “Segments Left”, and “Address[1..n]” fields. First and Last Hop External bits indicate whether the route leads to or from outside of the DSR network; such paths must not be returned from the cache of intermediate nodes. “Salvage” indicates the number of times the packet has been salvaged (see section 3.6.1), that is, rewritten by an intermediate node to ensure delivery. “Segments Left” indicates how many addresses in the option must still be traversed until reaching the final destination. “Address[1..n]” lists these intermediate nodes which the packet has been, and will be, through. DSR Source Route option is present in almost every packet.

Pad1 and PadN options include requested amount of padding, to ensure that the total DSR Header will be properly aligned to a multiple of 4 bytes.

3.3 Basic Route Discovery

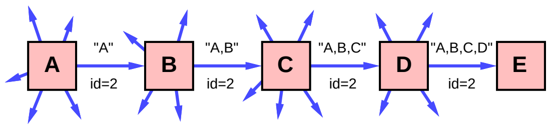

Figure 2: Route Discovery Example from Node A to Node E [1]

When a source node S is originating a packet to some destination node D, S inserts a DSR Header with DSR Source Route option to the packet, specifying which nodes the packet should be source routed through. Usually the source route is found by examing the node’s Route Cache: if valid information for destination D is not available, S will perform Route Discovery to find a new route.

Route Discovery is initiated by source S by transmitting a packet to local broadcast address 255.255.255.255 using a Route Request option. This is received by about all nodes close enough to S.

Figure 2 shows a simple example of Route Discovery. Node A wishes to obtain a route to E, and places its address to Target Address field. The Address[1..n] field lists only Node A. Nodes keep track of which sources have sent Requests (including Identification field) for which Target Addresses.

Every node processing Route Requests then performs roughly the following steps (all steps terminate the processing if they match):

1. If the node is the target of this Request, Reply back the accumulated source route that was in the Request.
2. If the node's address is listed in the Address fields of the Request, discard the whole packet for loop prevention.
3. If the Request arrived through an interface which requires bidirectional connectivity, and it was last forwarded by a node in the blacklist, do certain processing and discard the packet.
4. If the node has seen recently a request from the same source with the same Identification and Target Address pair, the packet has to be discarded as a flooding prevention feature.
5. If the node has a cached route for the destination, reply back under some circumtances as noted in section 3.5.1.
6. Else the node appends its own address to the list and re-broadcasts it with the same Identification.

When node E sends a Reply back to A it examines its Route Cache for A, and if found, sends the packet using that. If no such entry is found, node E should also perform Route Discovery back to A, but include the Reply in the DSR header to avoid possible recursion. In the case of bidirectional links, node E will simply reverse the source route to avoid unnecessary delays instead.

When node has to perform Route Discovery, the packets that trigger Discovery are placed in Send Buffer, which contains all the packets which can't yet be transmitted due to missing source route to the destination. The packets are kept there until a source route becomes available, a timeout expires or Send Buffer is about to overflow and some entries must be removed. While in the Send Buffer, new Route Discoveries should also be attempted periodically until the timeout to avoid temporary unreachabilities discarding packets. The period between subsequent Route Discoveries for the same Target uses an exponential back-off algorithm to keep the amount of request in reasonable numbers in the case of e.g. network partition.

3.4 Basic Route Maintenance

When a packet is forwarded or originated with a source route, each node in the source route is responsible for making sure that the packet has been received by the next node in the source route list.

This insurance is usually done with acknowledgments. In wireless networks, this is sometimes possible using link-layer mechanisms such as an option is MAC protocol, even possibly without a significant extra cost. In some cases, "passive acknowledgments" are also possible when the interface can be placed in promiscuous mode: node B can confirm that C has received a packet if it can overhear C trasmitting it to D.

If these are not available, DSR-specific acknowledgments can be used in IP-layer; these are referred to as "network-layer acknowledgments". Such Acknowledgment Request can be sent separately or e.g. included, that is, "piggybacked" on the original packet. If an acknowledgment is received, the node may omit sending acknowledgments for a short period of time, based on the assumption that it's unprobable that anything bad happens in e.g. 0.5 seconds, or if it does, the damage is limited.

If it seems that the neighbour has disappeared, whether e.g. by not getting link-layer acknowledgments or reaching the retransmission limit of network-level acknowledgments, the node should treat this next-hop as "broken". The link should be removed from Route Cache, and Route Error should be sent back to every sender which sent packets to that neighbor since the time acknowledgments were last received. The retransmission of the original data packet is handled by upper-level protocols such as TCP and UDP. Upon the receipt of a Route Error, the original sender will resent the data using other cached paths or perform a new route discovery.

3.5 Additional Route Discovery Features

3.5.1 Caching Overheard Routing Information

Nodes forwarding or overhearing (e.g. by being in promiscuous mode) packets add all information to its Route Cache; the usefulness of this depends on whether links can be assumed to be birectional or usually bidirectional.

When forwarding or overhearing packets, one can add the “forward” path of any source routes to the cache. In the case of generally bidirectional link-layer, the reverse paths can be calculated and also added to the cache.

3.5.2 Replying to Route Requests Using Cached Routes

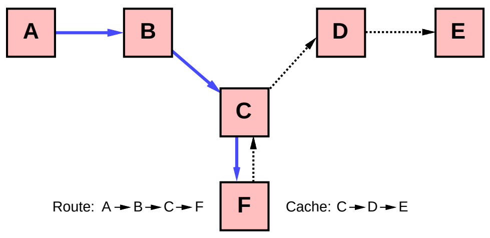

Figure 3: Noticing duplicates when an intermediate node is replying [1]

Nodes, even though they are not the target of a Route Request, examine their Route Cache before forwarding packets. If their own cache contains a valid path, there is no need to perform Route Discovery any further; a Reply is constructed using the accumulated path in the Request and Source Route stored in the Cache. Before sending the Route Reply, however, the intermediate node must examine whether there are any duplicates, as seen in fig. 3, and if so, not generate a reply: such a reply could easily lead into a routing loop.

3.5.3 Preventing Route Reply Storms

As cached route entries are returned, as discussed above, under some circumstances there are considitions where a large number of nodes might send Route Replies back to one source. This may happen for example when a packet is sent to the local broadcast address. To alleviate this, if a node sending a reply can put its interface into the promiscuous mode, it can delay replies by a small but random period. The promiscuity is required so that the node can hear if others have already sent a reply and the initiating node has begun to use it.

3.5.4 Route Request Hop Limits

Route Request messages can be limited in scope by setting Time-to-Live in IP header to a small value. This can be useful when one does not want to propagate the Route Requests sent to 255.255.255.255 to the whole DSR domain. This can be used to e.g. determine, without a large number of cached replies, whether a given node is a direct neighbor of the initiating node. This could also be to implement an “expanding ring” search for the target, similar to performing “traceroute”.

3.6 Additional Route Maintenance Features

3.6.1 Packet Salvaging

If a node in a source route becomes unreachable, detected by e.g. Route Maintenance procedures, an intermediate node that would have forwarded the packet to a lost neighbor should “salvage” the packet. Salvaging is possible if the node has a working source route to the destination, and can be done by simply replacing the source route in these incoming, nexthop-is-unreachable packets with a working source route.

To prevent routing loops and unexpected scenarios, packets will not be salvaged endlessly so a counter is kept in the source route field, and when the total salvage count exceed a defined value, the packet is no longer salvaged.

In this scenario, in addition to possibly salvaging a packet, a Route Error message should be sent back to the source, indicating that the original path had become unavailable. This way salvaging need not be done endlessly in one spot, but the nodes will notice the error and find source routes around it.

3.6.2 Queued Packets Destined over a Broken Link

Similar to packet salvaging, if an intermediate node's next-hop becomes unavailable, and there are packets in its queues destined for that next-hop, the packets should be removed from the queue, followed by a Route Error message. If the node has a new path towards removed packets' destination address, the packets can also be salvaged as above, else they'll just be discarded.

3.6.3 Automatic Route Shortening

In some cases, e.g. by overhearing transmissions in the promiscuous mode, an intermediate node may notice that a direct, shorter path is also available when forwarding packets to some destination, see fig 4.

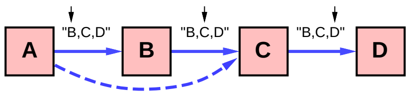

Figure 4: Route shortening [1]

In this scenario, some node, one or more hops after the current next-hop, is detected to be reachable directly. Then, a "gratuitous" Route Reply with the shortened path is returned to the sender of the packet, even though the source didn't send a Route Request. Gratuitous Route Replies are rate-limited, so that they aren't sent for the same route every time a packet to the destination comes by.

When performing automatic route shortening, additional information (e.g. signal-to-noise ratio) about the neighbor can also be considered.

3.6.4 Increased Spreading of Route Error Messages

When some node returns a Route Error message to the source, all nodes on the path (and possibly those overhearing the packet) will use information in that message to remove the broken link from their Route Caches.

In addition to that, the source, once it has received a Route Error message, can include the message in the next Route Request messages it sends. This ensures that the information spreads to off-path nodes too (those that didn't remove the link from the cache already), and that Route Replies to the Request will not contain the known-to-be-bad link.

This is very important as there can be very little verification on received route replies: one bad node with a stale cache could attract traffic it can't handle if the knowledge about broken links is not propagated quickly.

4 DSR Evaluation

4.1 Standardization, Simulations and Implementation Status

DSR was first specified in 1996 [4]. It has been an IETF Internet Draft for years, and has been adopted as a working group item in Mobile Ad-hoc Networks working group. It has been sent to IESG to be released as an Experimental RFC, but is still in the process and hasn't obtained any formal status yet. It can be expected that DSR will go Experimental reasonably soon, unless there are strong political objections to the design choices, such as source routing and adding routing information to payload packets.

DSR behaviour has been simulated extensively with e.g. ns2 [10, 11]. It can be seen that DSR performs very well in the simulated scenarios with relatively small amounts of DSR nodes (counted in dozens). Very high number (hundreds or thousands) of nodes have not been tested, and DSR is not really designed to scale to that; DSR is applicable to usually less than a hundred nodes in a network.

DSR was first implemented and made available for FreeBSD 2.2.7 platform. That version was used in a testbed experiment in Carnegie-Mellon University [12] which was used for about four months with five mobile nodes, as cars driving back and forth carrying laptops with WaveLAN network cards. Later, DSR was ported to FreeBSD 3.3. At least two Linux implementations have also been separately developed. DSR has also been implemented for Windows CE. Other implementations also exist, though less numerous than some other MANET protocols, like AODV.

4.2 Analysis of the Protocol Specification

The main feature of DSR is that it embeds control information in the data packets sent in the network; consequently it is able to boast of requiring no periodic link status sensing, routing advertisements and such.

This coupling of two traditionally entirely separate functionalities can be seen as a potentially problematic approach: when everything works as expected, it is probable that the protocol works very well and with very little overhead as specified. In the long term, though, problems arising due to implementation bugs, unforeseen configurations or such may be hard to pin down or debug.

More specifically, the protocol relies on every source knowing the best path to the destination. If, for some reason, this path is no longer valid, there may be circumstances where the other network elements already know that, but the source insists the path is correct. As the other network elements perform path caching for efficiency, the source may keep trying to update the path anyway, over and over again. This might lead to routing instability. This "one bad apple ruining the whole barrel" is a tradeoff between distributed and centralized route calculation.

The DSR protocol also seems to assume that link-layer medium is the same everywhere. More specifically, at least the directionality (links being either bi-, or unidirectional) must be the same in a MANET domain, or the DSR protocol must keep track of the directionality of each link. This seems to be a rather strict restriction at least in theory; one of the advantages of DSR was that it also supported unidirectional links. If all the links are required to be bidirectional, this problem goes away.

The header format is made such that DSR header is placed directly after the main IP header. Therefore, packet filters examining a packet with DSR header, looking to process UDP/TCP/ICMP headers will only see the protocol being DSR. Practically, packet filtering cannot be done in a DSR domain, or all packet filters must be upgraded to support DSR header: to be able to skip over it and look for the terminal header. As DSR header format was derived from IPv6 [13] header chaining, the problem (complete solutions still missing) exists there too.

Previous versions of DSR tried to make IP a reliable protocol by retransmissions and acknowledgments. This has now been removed, and such happens only with Route Maintenance procedure, but the text is a bit unclear even on that: if Route Maintenance packets are piggybacked on TCP/UDP data, which ends being retransmitted, the behaviour might not what the application expects.

DSR header cannot be secured with IPSEC [14]: as packets can be modified in transit by changing DSR header, the verification or encryption of DSR control information is impossible. If all nodes in a DSR domain shared a private key, using IPSEC would be theoretically possible. This seems to be an unacceptable operational security practise.

DSR relies on securing link-layer, and if necessary, using encryption there. The conflicts with Internet Protocol architecture: few link-layers provide this kind of service, and IP was designed to run uniformly on all the media. Further, for example IEEE 802.11 ("Wireless LAN") is seen as one of the main link layer mechanisms for DSR. The security of IEEE 802.11 has been broken, and for all practical purposes, none exists. Therefore the real applicability of DSR, especially in scenarios where security is important, is a very questionable.

DSR also requires that every router in the path examines every packet containing a DSR header to process certain fields, even if the destination is not the router itself. This is a rare requirement, and the scalability of it is questionable. Routers usually implement most of the forwarding logic in hardware, but this would require either hardware modifications or packets processed on CPU. In networks with low packet rates and throughput, this is not a big problem, but this might set a limit to extreme scaling up. Practically, however, DSR would only be used in relatively small stub networks, where this is unlikely to be highly problematic in most scenarios.

DSR implicitly requires that every router (behind which a DSR node would like to communicate) in the DSR domain supports DSR; again, in small stub networks, this may not be problematic, but in real networks, where incremental deployment would be a strict requirement, there may be deployment issues.

DSR, like some other MANET protocols (but not all), requires that every node in the network trusts each other completely [15], as each node must receive, cache and use source routes from any node. This becomes a very challenging problem even if so-called Byzantine failures [16] need not be dealt with. Also, one must be very careful to filter out all packets with DSR headers from outside of the DSR domain, so that malicious attackers cannot alter the routing from outside of the DSR domain by sending packets including forged DSR headers from e.g. Internet.

DSR mechanism of changing the packet en-route, in particular, the size being dynamically changed by constantly-modified DSR header, could cause problems with Path MTU Discovery [17]. As the source may stuff a packet full of payload and a slim DSR header, if some node in the network tries to add more DSR options to it, the total size might exceed link MTU. At the very least, packets might have to be re-sent and cached Path MTU's changed.

There may be a lot of "ideological resistance" to performing source routing in the Internet. Nevertheless, in a restricted scenarios, source routing seems to provide a large number of undeniable benefits.

5 Future Work

The scope of applicability of DSR needs to be more precisely defined. Interactions with IPSEC, and routing protocol security in general should be studied. Interactions with packet filters should be explored. The specification text should be clarified on some points, especially regarding retransmissions. Especially a mixed network of uni- and bidirectional links should be simulated. How destructive a DSR node (or a set of nodes) can be with old cache entries should be studied. Even though DSR is not meant for many hundreds of nodes, comparisons with other MANET protocols with these amounts of nodes might be interesting.

6 Conclusions

DSR applies rather well to a smallish, less than a hundred, network of nodes that trust each other. In such networks, DSR may be especially useful if it is important to be able to deal with both high and low node mobility with reasonably small overhead without any manual changes; DSR is highly adaptive.

There are some areas that need clarifications or more study; some of these are applicable to some other MANET protocols too, though:

- Study on the more generic problem of trusting routing updates in ad-hoc networking
- Clarifications and tests on cache management, especially ensuring cache freshness and how far wrong data could spread
- Being able to use IPSEC to verify either the payload or DSR header data, so that link-layer security would not be required
- Clarifications on retransmission mechanisms and how that applies to unreliable IP
- Clarifications on interactions with firewalls and packet filters
- Clarifications and tests with bi- and unidirectionality of links
- Some other clarifications on the specification text

References

[1] Johnson, D., Maltz, D., Broch, J., *DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks*, Ad Hoc Networking, edited by Charles Perkins, Addison-Wesley, 2001.

[2] Johnson, D., Maltz, D., Hu, Y., Jetcheva, J., *The Dynamic Source Routing Protocool for Mobine Ad Hoc Networks (DSR)* http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-07.txt, work-in-progress, February 2002.

[3] Mobile Ad-hoc Networks Charter, *IETF Working Group webpage*, http://www.ietf.org/html.charters/manet-charter.html

[4] Johnson, D., Maltz, D., *Dynamic Source Routing in Ad Hoc Wireless Networks*, Mobile Computing, edited by Tomasz Imielinski and Hank Korth, Kluwer Academic Publishers, 1996.

[5] Hu, Y., Johnson, D., Maltz, D., *Flow State in the Dynamic Source Routing Protocol for Mobile Ad Hoc Networks*, http://www.ietf.org/internet-drafts/draft-ietf-manet-dsrflow-00.txt, work-in-progress, February 2001.

[6] Baker, F., *An outsider's view of MANET*, http://www.ietf.org/internet-drafts/draft-baker-manet-review-01.txt, work-in-progress, March 2002.

[7] Plummer, D., *An Ethernet Address Resolution Protocol*, IETF Standards Track RFC 826, November 1982.

[8] Corson, S., Macker, J., *MANET: Routing Protocol Performance Issues and Evaluation Considerations*, IETF Informational RFC 2501, January 1999.

[9] Johnson, D., *DSR discussion on work to be done*, MANET IETF53 Meeting Minutes, March 2002.

[10] Broch, J., Maltz, D., Johnson, D., Hu, Y., Jetsceva, J., *A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols*, Proceedings of Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking, p. 85-97, October 1998.

[11] Maltz, D., Broch, J., Jetscheva, J., Johnson, D., *The Effects of On-Demand Behaviour in Routing Protocols for Multi-Hop Wireless Ad Hoc Networks*, IEEE Journal on Selected Areas of Communications, 17(8):1439-1453, August 1999.

[12] Maltz, D., Broch, J., Johnson, D., *Experiences Designing and Building a Multi-Hop Wireless Ad Hoc Network Testbed*, Technical Report CMU-CS-99-116, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, March 1999.

[13] Deering, S., Hinden, R., *Internet Protocol, Version 6 (IPv6) Specification*, IETF Standards Track RFC2460, December 1998.

[14] Kent, S., Atkinson, R., *Security Architecture for the Internet Protocol*, IETF Standards Track RFC2401, November 1998.

[15] Papadimitratos, P., Haas Z., *Secure Routing for Mobile Ad Hoc Networks*, SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), San Antonio, TX, January 2002.

[16] Perlman, R., *Network Layer Protocols with Byzantine Robustness*, Technical Report, MIT Laboratory for Computer Science #429, October 1988.

[17] Mogul, J., Deering, S., *Path MTU Discovery*, IETF Standards Track RFC1191, November 1990.