Dynamic Source Routing (DSR) Protocol Implementation in ns-3

Yufei Cheng, Egemen K. Çetinkaya, and James P.G. Sterbenz
Information and Telecommunication Technology Center
Department of Electrical Engineering and Computer Science
The University of Kansas, Lawrence, KS 66045, USA
{yfcheng, ekc, jpgs}@ittc.ku.edu

ABSTRACT

Routing protocols are essential to the performance of wireless networks especially in mobile ad-hoc scenarios. The development of new routing protocols requires comparing them against well-known protocols in various simulation environments. In this paper, we present an overview of the well-known MANET routing protocols and the implementation details of the DSR routing protocol in the ns-3 network simulator. We verify DSR routing performance under various scenarios and compare its performance against other protocols implemented in ns-3: AODV, DSDV, and OLSR. Our results show that the performance of DSR shares similar characteristics with AODV yet has slightly higher overall performance results in terms of the routing metrics we use.

Categories and Subject Descriptors

I.6 [**Simulation and Modeling**]: General, Model Development, Model Validation and Analysis; C.2.2 [**Computer-Communication Networks**]: Network Protocols — *routing protocols*

General Terms

Implementation, Analysis, Testing, Verification

Keywords

DSR implementation, MANET, mobile ad hoc routing protocol, ns-3, AODV, DSDV, OLSR

1. INTRODUCTION

Mobile ad hoc networks (MANETs) [5] are self-configuring networks with mobile nodes connected by wireless links to form an arbitrary topology without an infrastructure. In MANETs, nodes self-organise and act as both end and intermediate systems. The two major challenges for routing in MANETs are the dynamic topologies caused by mobility, and maintaining connectivity with wireless channels and nodes moving out of range from one another.

WNS3 2012, March 23, Desenzano del Garda, Italy

DOI 10.4108/icst.simutools.2012.247749

Many routing protocols for MANETs have been proposed with four of them arguably the most prominent ones in the research community: AODV [18, 16], DSDV [17], DSR [8, 9], and OLSR [4], due to their early emergence and varied characteristics. The operational characteristics and performance of these four protocols provide an important baseline with which new protocols should be compared.

Simulation has become the backbone for MANET research, since the simulation environment provides easily accessible resources to study new protocols or models [5, 10]. As an open-source simulator, ns-2 [13] has been widely employed in the academic research community. However, in response to a number of deficiencies, the ns-3 discrete event network simulator [14] is proposed as its successor and is still under development, providing greater flexibility, evolvability, modularity, and the support of heterogeneity including hybrid wired and wireless models.

Despite its advantages, ns-3 is relatively new with limited number of protocol models incorporated into its release distribution [19]; existing built-in MANET routing protocols are limited to the optimised link state routing (OLSR) and ad hoc on-demand distance vector (AODV) protocols. We have implemented destination-sequenced distance vector (DSDV) protocol and incorporated it in the distribution [11]. To have a fairly complete set of protocols for performance comparison, we have developed an ns-3 implementation of the dynamic source routing (DSR) protocol[1]. DSR is one of the major MANET routing protocols proposed [9], and it provides a baseline reactive source routing for performance comparison and for reproduction of previous results (e.g. [1]) in the ns-3 environment.

In this paper, we present our ns-3 implementation of the DSR routing protocol and compare its performance with other existing MANET routing protocol models in ns-3. The rest of the paper is organised as follows: Section 2 presents background and related work on MANET protocols. Section 3 presents the implementation details of DSR in ns-3. Performance evaluation and comparison of DSR against AODV, DSDV, and OLSR are presented in Section 4. Finally, Section 5 is our conclusions and future work.

2. BACKGROUND AND RELATED WORK

In this section, we present background information about MANETs, and briefly describe the four MANET routing protocols simulated in this paper: DSR, AODV, DSDV, and OLSR.

[1]Source code is at http://code.nsnam.org/tomh/ns-3-dsr-2

2.1 Routing Protocol Types

MANET routing protocols can be classified into two categories based on their update mechanisms: proactive and reactive. We briefly these types; a more detailed explanation is presented in our companion paper on DSDV [11].

2.1.1 *Proactive Table-Driven Routing Protocols*

Proactive routing protocols maintain updated routing information of all the nodes in the network by periodically distributing routing tables among each other. The advantage is that the routes to any destination are ready to use when needed. However, routing tables grow with the size and node density of the network, rather than the number of routes actually needed. The overhead of flooding route advertisements to maintain convergence is a major drawback of proactive protocols. Destination-sequenced distance vector (DSDV) [17] and optimised link state routing (OLSR) [4] are two major proactive routing protocols. DSDV maintains a routing table with entries for all the nodes. In order to reduce the amount of overhead, two types of update packets are used, with one type carrying all the available routing information while the other one carries only the information changed since the previous full update. In order to avoid routing loops, a *sequence number* is used. OLSR is a point-to-point protocol based on the traditional link-state algorithm. It uses HELLO messages at each node to discover neighbor information and select a set of multipoint relays (MPRs). Only MPRs are allowed to rebroadcast the received messages. OLSR floods topology data frequently enough over the network to make sure all nodes are synchronised with *link state* information.

2.1.2 *Reactive On-Demand Routing Protocols*

Unlike proactive routing protocols, reactive routing protocols construct routes only when needed for data transmission. When a route to a new destination is required, the node initiates a route request and must wait until the route is discovered. There is no need to distribute their routing information periodically or to maintain routes for all the nodes in the network. The disadvantage is the delay in finding routes to new destinations. Dynamic source routing (DSR) [9] specified in RFC 4728 [8] and ad hoc on-demand distance vector (AODV) [18, 16] are two well-studied reactive routing protocols. DSR is an on-demand routing protocol based on the source routing concept. It contains two major working phases: *route discovery* and *route maintenance*. When a node initiates packets to one destination, it first searches its route cache for saved route entries. If not found, it initialises the route request process. DSR carries the full routing information from the source to the destination in the packet header. Route maintenance is the process to validate the source route and is managed using either route error messages or acknowledgements. AODV is a distance vector routing protocol that operates only on demand. When a route does not exist to a give destination, a route request (RREQ) message is flooded by the source and by the intermediate nodes that do not have previous routes in their routing table. Once the RREQ message reaches the destination, the node responds by unicasting a route reply (RREP) message. This way nodes along this path set up forwarding entries in their routing tables and form the whole route. AODV uses sequence numbers created by the destination for each route entry to avoid routing loops.

2.2 Previous DSR Implementations

DSR has been implemented and analysed in a number of simulation tools. Most of the previous performance comparisons use the ns-2 network simulator [13]; including performance comparison with DSR, AODV, and DSDV [1, 7]. However, there are also studies that use different simulation tools. AODV and DSR performance comparison is performed using GloMoSim [12], QualNet [15], and ns-2 [2, 6]. Although being the predecessor of ns-3, the ns-2 implementation of DSR cannot be ported to the ns-3 environment due to the significantly different simulation architecture and code structure, however, the ns-2 implementation was used to provide insight and guide design decisions for our ns-3 implementation.

3. DSR MODULE FOR ns-3

This section describes our implementation of DSR. The two major components of the DSR operation are route discovery and route maintenance. All the major attributes used in this implementation are listed in Table 1. The relation among all the classes implemented in this module is shown in Figure 1.

We implemented the DSR routing protocol `ns3::dsr::DsrRouting` in ns-3 by extending from the abstract base class `ns3::Ipv4L4Protocol`. The `ns3::dsr::DsrFsHeader` and `ns3::dsr::DsrOptionHeader` is extended from `ns3::Header`, and they are essentially shim headers between the transport layer and network layer. We have also declared `ns3::dsr::RouteCache` to store all the routes that have been discovered in previous route discovery process. Similarly, we have declared the `ns3::dsr::SendBuffer` class to store all unsent data packets and `ns3::dsr::RreqTable` to avoid duplicate route requests as well as control the rate of consecutive route requests for one destination. The `ns3::dsr::MaintainBuff` is used to store the data packet when sent out from the send buffer and waiting for delivery of acknowledgment from the next hop node. An in-depth explanation of all these classes is presented in the following sections.

Most of the other routing protocol implementations in ns-3 are IP dependent, which render plugging-in DSR as a shim between IP and the transport layer protocol problematic. In the current implementation of DSR in ns-3, it acts as `ns3::Ipv4L4Protocol`, which uses the services of IP. Therefore, DSR should bypass IP's forwarding callback mechanism implemented in ns-3 and implement its own. To do this, the destination address in IP header is always set as the gateway address that is the next hop for the packet, and the real destination address will be shown in the DSR header. Figure 2 shows how DSR packets are encapsulated within IP in ns-3.

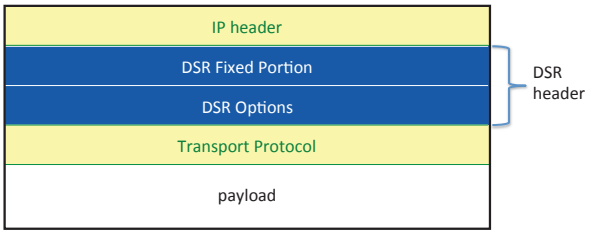

Figure 2: DSR header encapsulation within IP

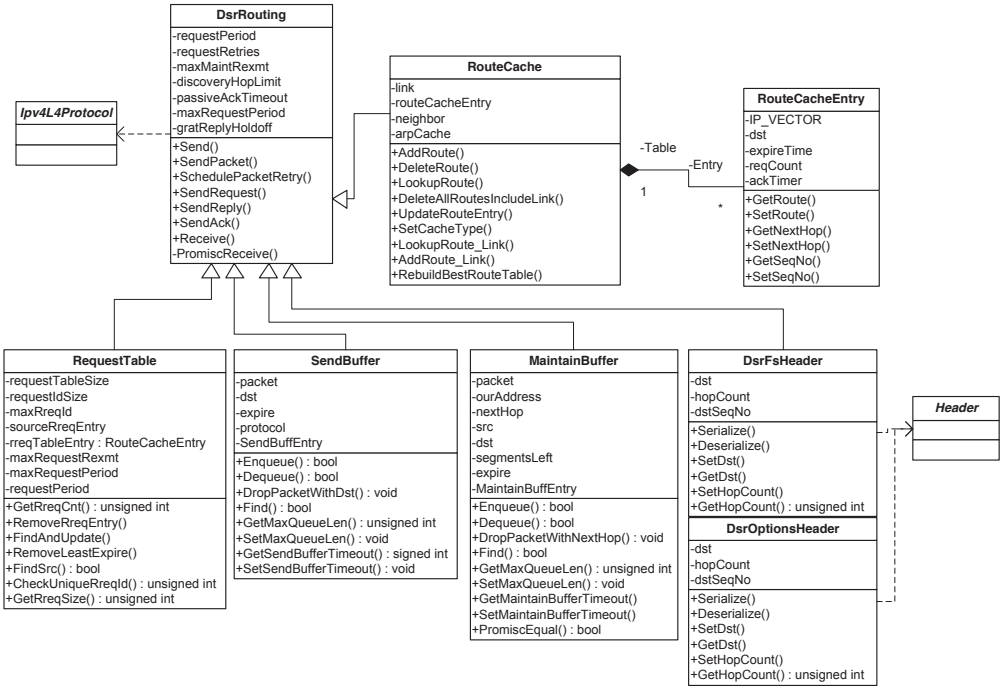

Figure 1: DSR class diagram

3.1 Routing Functions

The routing functions of DSR consist of two parts: route discovery and route maintenance. We illustrate these two mechanisms in detail as follows.

3.1.1 Route Discovery

When a node S has a packet to send to some destination node D but does not currently have a route to that node in its route cache, the node S saves the data packet in its send buffer and initiates route discovery process to find a route. To prevent from buffering the packets indefinitely, packets are dropped if they wait in the send buffer for more than `MaxSendBuffTime` (the default is 30 s). For route discovery, S transmits route request packets as local broadcast messages, specifying the target address and a unique request identifier. The node receiving the route request packet will check its identifier and target address in the request header; if the same one has been received before, the packet will be recognised as a duplicate and silently dropped. Otherwise, it appends its own node address to a list in the route request header and rebroadcasts it. When the route request packet reaches its destination, the target node sends a route reply packet back to the initiator of the request, including a copy of the accumulated list of node addresses in its reply header. When the route reply reaches the initiator of the request, it caches the new route in its route cache. Upon receiving the route reply message, node S will use the source route from route reply to send the data packet to D. Furthermore, all the intermediate nodes receiving route reply packet will cut the route from their own to the destination and save it in their own route cache.

Another feature of the route request packet is that when an intermediate node receives a packet, it searches the route cache for route to the destination address. If there is an existing route, the node attaches the route found to the route received from route request header, which forms a full route from the source to the destination. Then the node sends back a route reply packet to the source with the full route.

3.1.2 Route Maintenance

Route maintenance is the mechanism by which the source node or any intermediate node is able to detect link breakage when the network topology has changed such that the source route in the source route header no longer works. This is a hop-by-hop operation and there are three mechanisms to verify the delivery of data packets to the next hop as discussed in Section 3.2. If the data packet fails to deliver to the next hop, the sender will retransmit the data packet. After `MaintenanceRetries` times of retransmission fails, a route error packet will be sent out. The packet will not be dropped immediately; instead, a *salvage* mechanism will start by searching the route cache for alternative routes to the destination. If no alternative route is found, or the data packet has already been salvaged for MaxSalvageCount times, this data packet will be dropped.

DSR has a mechanism of removing stale route entries from the node's route cache. If a node does not use the route for a period of time, that route entry will be expired and removed from the route cache. In our implementation, DSR waits for RouteCacheTimeout before removing the entries, with default value as 300 s.

3.2 Acknowledgment Mechanisms

There are three types of acknowledgement mechanisms specified in the RFC 4728: link-layer, passive, and network-layer acknowledgement. If the media access control (MAC) protocol in use can provide feedback for the successful delivery of data packets, then it should be used to notify the

delivery of data packets and maintain the route validity. It requires the MAC protocol to be able to provide layer-2 feedback to the network layer about the data packet delivery. However, in ns-3, only `AdhocWifiMac` supports the layer-2 feedback mechanism. When link-layer acknowledgement is not available, but passive acknowledgement is, it should be used when the nodes can put themselves into promiscuous receive mode. In the absence of the above two acknowledgement mechanisms, the node should use network layer acknowledgements. When this mechanism is used, acknowledgement request header will be attached to the data packet before being transmitted. The acknowledgement will be sent back to the sender to notify the delivery of data packets.

3.3 DSR Optimisation Mechanisms

Several optimisations have been specified in RFC 4728: salvaging, gratuitous route reply, and increased spreading of route errors. The salvaging mechanism is triggered when the forward link is broken and there is an alternative route to the destination in the node's route cache. Instead of dropping the data packet, the node will try to retransmit it with the newly found route.

The gratuitous route reply is used when the node promiscuously received a data packet destined for other nodes but is named in the later unused portion of the packet's source route. Then it can infer that the intermediate nodes before itself in the source route are no longer necessary, and a gratuitous route reply packet will be sent back to the source with the shorter route.

The increased spreading of route errors means that when the node receives the route error for the data packet it originates, it is the source of the data packet and the node will piggy-back the route error packet in the next route request process. This way it ensures that the route error packet spreads to the neighboring nodes and gets the expired route entries deleted.

3.4 DSR Data Structures

There are several conceptual data structures that are important to support the DSR operation. Here we introduce the essential ones.

`RouteCache` contains all the routing information collected from route discovery process. The structure of the DSR `RouteCache` is implemented as follows. Each entry implemented by the `RouteCacheEntry` class corresponds to a node in the network and the entry is mapped to that node's IP address. Every entry stores the following attributes of a node: the IP addresses from the source to the destination saved in a vector, the destination address of the route, and the time-stamp of the route entry when it is initially saved. The `RouteCache` will save multiple route cache entries for a single destination since DSR accepts multiple route replies. All the route cache entries for a single destination are saved according to the hop-count and freshness of the route. The route with least hop count will be saved before others, and for those routes with same hop count, the route entry that is newly found will be saved before other routes. The `RouteCache` class has implemented functions to add, delete, update, look up, and print entries. Also, unlike other routing protocols, DSR saves the entire source route in the route cache, indexed only with the final destination. Therefore, when the direct route to a certain destination is not found, the look up route function will also search all the intermediate nodes in every single route entry for the expected destination. If the originating node finds any intermediate node that matches the expected destination, the newly found route will be removed from the original route and indexed with the new destination address for future use. All the route cache entries are governed by a global timeout value `RouteCacheTimeout`, with a default value of 300 s. The route cache will be purged periodically to get rid of outdated routes.

`PacketBuffer` is used to save data packet whenever there is no route or the node is waiting for next hop delivery notification. There are two types of packet buffers: `Send Buffer` and `Maintenance Buffer`. These both buffer data packets, the only difference is that the `Send Buffer` will save the packet when receiving it from transport layer and if a route is not found, while the packet will be saved in the `Maintenance Buffer` when sent out from the `Send Buffer` yet is waiting for the next hop acknowledgment. When DSR receives the data packet from the transport protocol, it first checks the `RouteCache` for previously found route entries. The data packet is saved to `Send Buffer` if no existing routes are found, and the node will initiate the route discovery process by broadcasting `Route Request` packets. When the packet in `Send Buffer` is sent out with `Source Route` header, the data packet will be saved in `Maintenance Buffer` waiting for next hop delivery notification from the acknowledgement mechanism used. After `MaxMaintenanceRetries` times of retransmission without receiving any delivery notification, the data packet will be removed from the `Maintenance Buffer`.

`RreqTable` is used to save route request information. It keeps track of two parts of route request operations: route request initiated by the node itself and request received by this node, which are implemented as `RreqRequestEntry` and `RreqRequestId`, respectively. `RreqTableEntry` saves the route requests have been initiated by the node itself. The fields of the route entries include the destination address that records the specific destination this node has initiated route requests to, the time-to-live (TTL) that allows the node to implement a series of mechanisms to limit the transmission hops of the route request packet, and the request number that is used to record the number of consecutive route requests sent for a certain destination. The other entry `RreqTable` record is `RreqTableId`. This entry keeps track of the route requests this node has received from other nodes, and drops duplicate route request packets. It is mapped to the source node that is originating this route request. The detail entries include the destination address that records the specific destination for this request, and the identification field that keeps track of the identification number of the route request packet. If the two fields in the `RreqTableId` are the same for two entries, the request packet will be recognised as a duplicate one and will be dropped silently.

`GratuitousRouteReply` class is used to shorten routes. Whenever a node overhears a transmission that is not destined for itself, but can infer a shorter route to a specific destination, it will send a `gratuitous route reply` to the source of the longer route indicating the shorter route. Its entry includes three fields: `ReplyTo` is the address to which the node originates a gratuitous route reply, `ReplyFrom` keeps track of the node from which this node overhears the packet that triggered the gratuitous route reply, and `GratuitousHoldoffTime` is the remaining time before the expiration of the entry.

Attribute	Defaults	Summary
MaxSendBuffLen	64	Maximum number of packets that can be stored in send buffer
MaxSendBuffTime	30 s	Maximum time packets can be queued in the send buffer
MaxMaintLen	50	Maximum number of packets that can be stored in maintenance buffer
MaxMaintTime	30 s	Maximum time packets can be queued in maintenance buffer
MaxCacheLen	64	Maximum number of route entries that can be stored in route cache
RouteCacheTimeout	300 s	Maximum time routes can be queued in route cache
SendBuffInterval	1 s	How often to check send buffer for pending data packets
NodeTraversalTime	100 ms	Time it takes to traverse the two neighboring nodes
MaxMaintRexmt	2	Maximum number of retransmissions for data packets
RreqRetries	16	Maximum number of retransmissions for route request discovery
MaintenanceRetries	2	Maximum number of retransmissions for data packets from maintenance buffer
RequestTableSize	64	Maximum number of request entries in the request table
RequestIdSize	16	Maximum number of request Ids in the request table
NonPropRequestTimeout	30 ms	Timeout value for non-propagation route requests
DiscoveryHopLimit	255	Maximum route discovery hop limit
MaxSalvageCount	16	Max salvage count for a single data packet
BlacklistTimeout	3 s	Time for a neighbor to stay in blacklist
GratuitousHoldoffTime	1 s	Time for gratuitous route reply entry to expire
BroadcastJitter	10 ms	Jitter time to avoid collision for broadcast packets
PassiveAckTimeout	100 ms	Time for a packet in maintenance buffer to wait for passive acknowledgment
RequestPeriod	500 ms	Base time interval between route requests
MaxRequestPeriod	10 s	Max time interval between route requests

Table 1: DSR attributes and their default values

3.5 DSR Header Format

The DSR header consists of two parts: DsrFsHeader and DsrOptionsHeader, as shown in Figure 3. The DsrFsHeader is fixed in size and used to carry information that must be present in all DSR packets, while the DsrOptionsHeader is used to carry information for specific DSR options.

3.5.1 DSR Fixed-size Header

The DsrFsHeader is a fixed size header which contains a next header field to indicate the immediate header following the DSR option header. The payload length field indicates the length of all the option headers following the DsrFsHeader. The message id, source id, and destination id are needed for the ns-3 implementation. These fields are added (in addition to the header fields from RFC 4728) to differentiate between data and control packets as well as identifying source and destination addresses for post-processing purposes. The modified header is shown in Figure 3; it is word aligned and adds four bytes extra overhead. The header fields are explained in detail below.

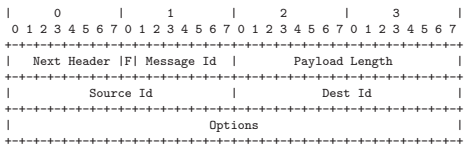

Figure 3: DsrFsHeader format

The next header is a 4-bit field that indicates the upper layer protocol id, indicating which transport protocol to pass to. The message id field is also 4 bits in length and indicates the type of message this DSR header is carrying, message id number 1 indicates a control packet, while id number 2 is a data packet. The source id and destination id are also added only for ns-3 implementation to indicate the initiator and the destination of the packet since the source and destination field in IP header is changed as the packet transmitted hop-by-hop. The payload length field indicates the payload length of all the DSR option headers combined.

3.5.2 DSR Options Header

The DsrOptionsHeader includes all the DSR options needed for protocol operation. The route request option, as shown in Figure 4, is attached to route discovery packet that will include all intermediate nodes' IP addresses in the header to form a full route to the destination. The route reply option is used to notify the source node with the whole route in its header with its header format as shown in Figure 5.

The source route option is attached to data packets and directs the packet from source to destination. It includes all intermediate nodes to form the full route, as shown in Figure 6. When a route error occurs, the route error option is used to indicate the link breakage and the node removes the erroneous routes from the route cache, as shown in Figure 7.

The acknowledgement request option is used to request next hop network acknowledgement as shown in Figure 8, while the acknowledgement option is used to indicate the successful delivery of data packet to the next hop, as shown in Figure 9.

4. DSR MODULE EVALUATION

To verify and validate our DSR routing protocol implementation, first we run nine unit test cases incorporated in the DSR module of ns-3 and verify its functionality. We investigate DSR performance with varying number of traffic sources and different pause times for mobility models and compare to the other existing MANET routing protocols in ns-3.12.1: DSDV, OLSR, and AODV. We provided preliminary results of DSR performance for different traffic types previously [3].

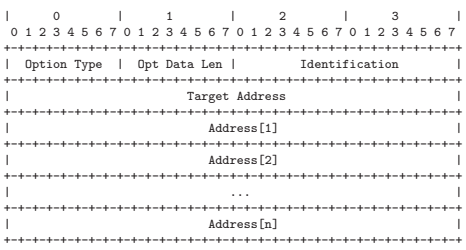

Figure 4: `RouteRequestHeader` **format**

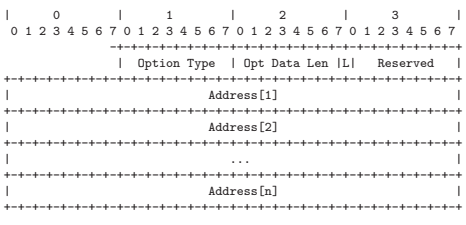

Figure 5: `RouteReplyHeader` **format**

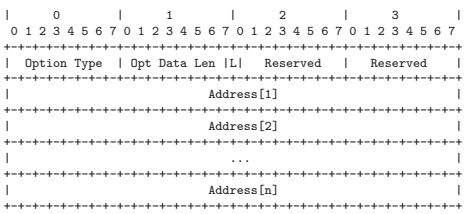

Figure 6: `SourceRouteHeader` **format**

4.1 Performance Metrics

The performance metrics for the evaluation of DSR routing protocol are packet delivery ratio (PDR) that is the number of packets received divided by the number of packets sent by the application, routing overhead that is the fraction of bytes used by the protocol to send the DSR control messages, and delay that is the difference in time between when the source transmits the packet at the MAC layer and when the MAC layer of the destination receives that data packet.

4.2 Simulation Setup

We configure simulation parameters as close as possible to previous studies [1] in order to have comparable results. We perform the simulations over an area of 1500 × 300 m^2. All the simulations are averaged over 10 runs with each simulation running for 1000 s; some are averaged over 20 runs to increase confidence. Simulations are performed with 50 nodes, and the source-sink traffic pairs are chosen to be 10, 20 and 30. We perform simulations with a packet size of

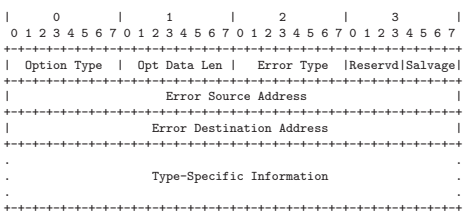

Figure 7: `RouteErrorHeader` **format**

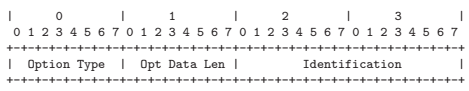

Figure 8: `AckRequestHeader` **format**

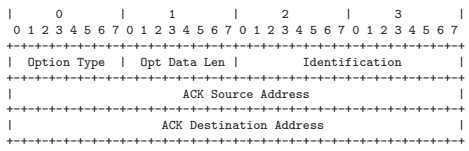

Figure 9: `AckHeader` **format**

64 bytes to exclude the potential network congestion caused by large packets. All the nodes are configured to send 4 packets/s. Using this lower packet rate, we can correctly evaluate the performance of the routing protocols. We use the ns-3 `On-Off` application to generate CBR (constant bit rate) traffic. The 802.11b MAC is the link layer over the Friis propagation loss model to limit the transmission ranges of nodes. Previous DSR performance study used a combination of free space propagation model and a two-ray ground reflection model [1]. The transmission range of the nodes is set at 250 m for evaluation. To achieve this transmission range, the transmit power is set to 8.9048 dBm. The mobility model used is steady-state random waypoint[2] with random velocities from 0.01 – 20.0 m/s and pause times of 0 – 900 s.

DSR has several parameters with some of them interconnected with each other and most of them are prone to change with different simulation scenarios like mobility model, node velocities, and node density. When the node velocity increases, the `RouteCacheTimeout` needs to decrease to get rid of invalid routes. Furthermore, the `NodeTraversalTime` is the time for a packet to traverse the transmission range, and requires careful consideration since it directly affects the time to detect link breakage: if set too small, there will be a possibility of false assumption of undelivered packets, while if too large it takes too long to respond to link breakages. Thus, proper choice of `RouteCacheTimeout` and `NodeTraversalTime` is important in different simulation scenarios. DSR protocol parameters for our simulations are shown in Table 2. The `NodeTraversalTime` is set as 2 μs to fit the 250 m transmission range in this case. RouteCacheTimeout is set as 300 s since this is a case with relatively low mobility nodes.

Parameter	Values
RouteCacheTimeout	300 s
NodeTraversalTime	2 μs
MaxSendBuffLen	64
MaxSendBuffTime	30 s
MaxMaintLen	50
MaxMaintTime	30 s
refer PassiveAckTimeout	4 μs

Table 2: DSR parameters

[2]This model eliminates the initial discrepancy of the RWP model and uses a stationary distribution.

4.3 Simulation Analysis

In the first scenario, we vary the pause time in steady state random waypoint mobility model so that we can analyse the performance of DSR in both mobile and static scenarios. Our simulation results show similar simulation results with previous work [1], from where we get our simulation configuration. Figure 10 shows the variation of PDR by varying the pause time. Note that the total number of nodes is 50 for all cases.

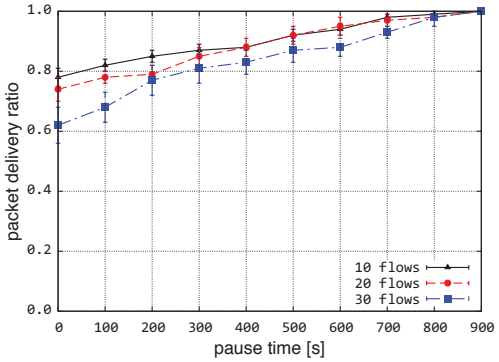

Figure 10: PDR with varying pause time

We can see that as pause time increases, the packet delivery ratio for all three traffic source cases also increases. When the pause time is 900, the PDR is 100% since the nodes are stable for the whole process of simulation. The PDR for 20 flows is greater than that for 30 flows for all pause times as shown in Figure 10. This is due to the fact that as the number of flows increase, the routing overhead also increases and this leads to more collisions in the network.

The routing overhead for different number of traffic flows with varying pause times is shown in Figure 11. This plot shows that overhead increases with the increasing number of flows. This is expected for DSR since as the flow number increases, more route discovery packets will be sent. The overhead for the 30 flows case has high overhead when the pause time is 0 s since as nodes constantly moving, route error packets will also be transmitted out more often.

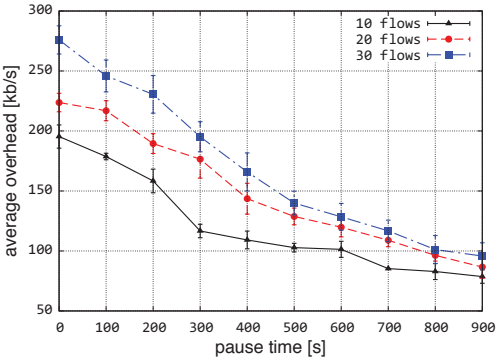

Figure 11: Overhead with varying pause time

Next, we analyse packet delay for the above scenario. Figure 12 shows the delay decrease as the pause time increases. This is because as the pause time increases the nodes will be immobile for longer durations and less route breakage will occur with fewer number of route discovery cycles needed. The delay increases slightly with an increase in the number of traffic flows, since more traffic flows cause more data to be queued for transmission.

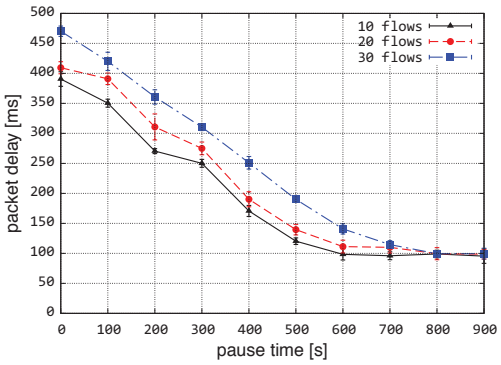

Figure 12: Packet delay with varying pause time

From Figure 13 to Figure 15 we only compare the 10 traffic flow cases with 50 nodes among the four routing protocols. In Figure 13, we compare the packet delivery ratio of existing MANET routing protocols implemented in ns-3 with DSR. From the plot we can see that OLSR outperforms DSR, DSDV, and AODV. The performance of DSDV follows the results from the previous work [1], while the performance of DSR and AODV has a little decrease compared to those results. This may be caused by the different MAC modules they have used.

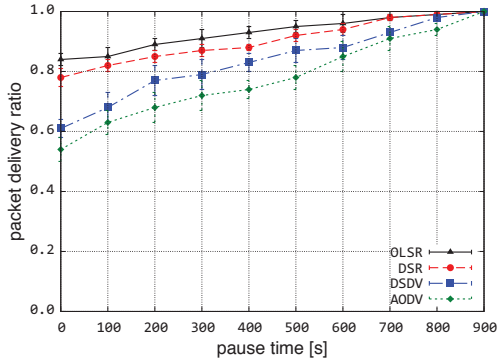

Figure 13: PDR with varying pause time

In our analysis, we also compare the routing overhead involved with all four protocols. The overhead for DSDV and OLSR stays around 200 kb/s and does not change significantly with the pause time increase as shown in Figure 14. As proactive routing protocols, both DSDV and OLSR need to initiate route discovery even when there is no need for the route. The overhead for AODV and DSR decreases as the pause time increase, because the route breakage decreases. The overhead for DSR is slightly less than that for AODV since DSR does not need periodic broadcast messages that AODV uses.

We analyse the packet delay for these protocols and the result is shown in Figure 15. The packet delay is greater for DSR and AODV compared to that for DSDV and OLSR. This is because as reactive routing protocols, both DSR and AODV need more time to react to link changes while nodes

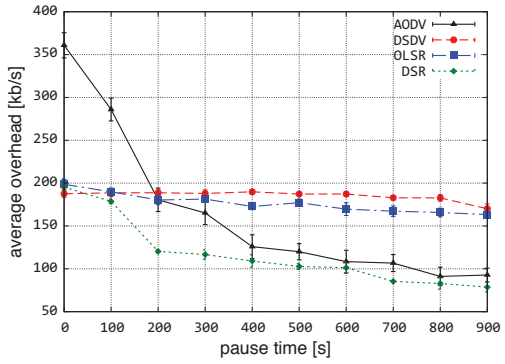

Figure 14: Overhead with varying pause time

are moving. Furthermore, we note that the variation of delay (shown as the error bar) for both DSR and AODV is comparably larger since they both rely on the buffering extensively in the route discovery cycle. As the pause time increases, the delay for all the protocols decreases since less link breakage occurs.

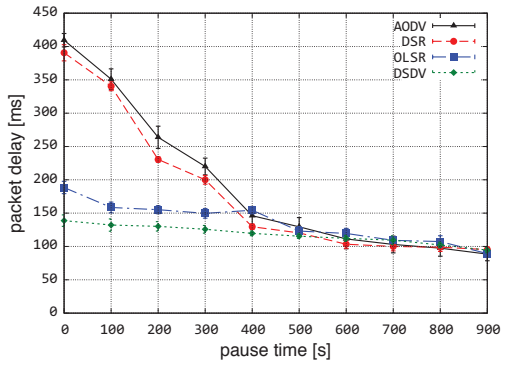

Figure 15: Packet delay with varying pause time

5. CONCLUSIONS

In this paper we presented the implementation of the DSR MANET routing protocol in ns-3. A detailed explanation of the components and how each class in the protocol interacts with one other is also provided along with attributes that can be modified in the protocol is presented. We analysed our DSR implementation with varying pause time for steady state random waypoint mobility model and compared its performance against DSDV, OLSR, and AODV. Our results indicate that DSR can achieve high packet delivery ratio with less routing overhead for low mobility scenarios. For part of the future work, we will test the performance of the MANET routing protocols in high mobility scenarios.

Acknowledgments

We would like to acknowledge the assistance of Justin P. Rohrer, and the members of the ResiliNets research group for their advice and suggestions which helped us with this implementation. We would also like to thank Tom Henderson and the ns-3 development team for their responsiveness to issues with ns-3 platform.

6. REFERENCES

[1] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-hop Wireless Ad hoc Network Routing Protocols. In *ACM/IEEE MobiCom*, pages 85–97, Oct. 1998.

[2] I. Broustis, G. Jakllari, T. Repantis, and M. Molle. A Comprehensive Comparison of Routing Protocols for Large-Scale Wireless MANETs. In *IEEE SECON*, volume 3, pages 951–956, Sep. 2006.

[3] Y. Cheng, E. K. Çetinkaya, and J. P. Sterbenz. Performance Comparison of Routing Protocols for Transactional Traffic over Aeronautical Networks. In *Intl. Telemetering Conf. (ITC)*, Oct. 2011.

[4] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626, Oct. 2003.

[5] M. Conti and S. Giordano. Multihop Ad Hoc Networking: The Theory. *IEEE Communications*, 45(4):78–86, Apr. 2007.

[6] H. Jiang and J. Garcia-Luna-Aceves. Performance Comparison of three Routing Protocols for Ad hoc Networks. In *IEEE ICCCN*, pages 547–554, Oct. 2001.

[7] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks. In *ACM/IEEE MobiCom*, pages 195–206, Aug. 1999.

[8] D. Johnson, Y. Hu, and D. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728, Feb. 2007.

[9] D. B. Johnson, D. A. Maltz, and J. Broch. DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks. In C. E. Perkins, editor, *Ad Hoc Networking*, chapter 5, pages 139–172. Addison-Wesley, Boston, MA, 2001.

[10] S. Kurkowski, T. Camp, and M. Colagrosso. MANET Simulation Studies: The Incredibles. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(4):50–61, Oct. 2005.

[11] H. Narra, Y. Cheng, E. K. Çetinkaya, J. P. Rohrer, and J. P. Sterbenz. Destination-Sequenced Distance Vector (DSDV) Routing Protocol Implementation in ns-3. In *ICST SIMUTools Workshop on ns-3 (WNS3)*, Barcelona, Mar. 2011.

[12] The GloMoSim. `http://pcl.cs.ucla.edu/projects/glomosim/`.

[13] The network simulator: ns-2. `http://www.isi.edu/nsnam/ns/`.

[14] The ns-3 network simulator. `http://www.nsnam.org`.

[15] The QualNet. `http://www.scalable-networks.com`.

[16] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561, July 2003.

[17] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *ACM SIGCOMM*, pages 234–244, Oct. 1994.

[18] C. E. Perkins and E. M. Royer. Ad-hoc On-Demand Distance Vector Routing. In *IEEE WMCSA*, pages 90–100, Feb. 1999.

[19] E. Weingartner, H. vom Lehn, and K. Wehrle. A Performance Comparison of Recent Network Simulators. In *IEEE ICC*, pages 1–5, Jun. 2009.