Destination-Sequenced Distance Vector (DSDV) Routing Protocol Implementation in ns-3

Hemanth Narra, Yufei Cheng,
Egemen K. Çetinkaya, Justin P. Rohrer 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
{hemanth, yfcheng, ekc, rohrej, jpgs}@ittc.ku.edu
http://www.ittc.ku.edu/resilinets

ABSTRACT

Routing protocols are a critical aspect to performance in mobile wireless networks. The development of new protocols requires testing against well-known protocols in various simulation environments. In this paper we present an overview of several well-known MANET routing protocols and the implementation details of the DSDV routing protocol in the ns-3 network simulator. We analyse DSDV routing performance under various scenarios and compare its performance with the other protocols implemented in ns-3, AODV and OLSR. Our results verify the implementation of DSDV and show performance comparable to that of OLSR.

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

Algorithms, Design, Performance, Verification

Keywords

DSDV, ns-3, DSDV model implementation, MANET routing protocol, ns-3 simulation methodology, AODV, OLSR

1. INTRODUCTION

Wireless mobile ad hoc networks (MANETs) [3] that do not require infrastructure to operate have been the subject of significant research. In MANETs, nodes self-organise and act as both end systems and as intermediate systems. The two major challenges to routing in MANETs are the dynamic topologies that result from mobility, and maintaining connectivity in the face of wireless channels and nodes moving out of range from one another.

WNS-3 2011, March 21, Barcelona, Spain

DOI 10.4108/icst.simutools.2011.245588

Many routing protocols for MANETs have been proposed, however four are arguably the most prominent in the research community: AODV [11], DSDV [13], DSR [6], and OLSR [2], due to their early emergence and varied characteristics. The operation and performance of these four protocols provide an important baseline to which new protocols should be compared. In particular, DSDV provides the proactive distance vector case.

Simulation has been the backbone of MANET research [3, 7], since the simulation environment provides easily accessible resources to study new protocols and models. The ns-2 simulator [8] has been widely used due to its open-source model appropriate for the academic research community. In response to a number of deficiencies, the ns-3 discrete event network simulator [9] is under development, providing greater flexibility, modularity using C++, evolvability, and support for heterogeneity including hybrid wired and wireless models.

Despite its advantages, ns-3 is relatively new with few protocol models yet incorporated into its release distribution [17]; existing built-in MANET protocols are limited to the optimized link state routing (OLSR) and ad hoc on-demand distance vector (AODV) protocols. Thus we have developed an ns-3 implementation of the destination-sequenced distance vector (DSDV) protocol and are currently developing a DSR implementation. DSDV is one of the earliest MANET routing protocols proposed [13], and provides a baseline proactive distance vector algorithm for performance comparisons.

In this paper we describe our ns-3 implementation of the DSDV routing protocol and compare its performance against existing MANET routing protocols 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 details of the DSDV module implementation in ns-3. Baseline performance evaluation and comparison of DSDV against OLSR and AODV is presented in Section 4. Finally, Section 5 presents our conclusions and future work, including a DSR implementation in progress.

2. BACKGROUND AND RELATED WORK

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

2.1 Routing Protocol Types

MANET routing protocols can be classified into two categories based on their update mechanisms: proactive routing protocols and reactive routing protocols.

2.1.1 Proactive Table-Driven Routing Protocols

Proactive routing protocols maintain routing information of all the nodes in the network and add new routes or update existing routes by periodically distributing routing information among each other. One advantage of doing so is that routes to any destination are ready to use when needed. However, this is offset by the overhead of route updates in response to mobility, for which nodes may have to wait anyway. For a large dynamic network, convergence may not be possible. Routing tables grow with the size and 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.

2.1.2 Reactive On-Demand Routing Protocols

Unlike proactive routing protocols, reactive routing protocols construct routes only when they are required. Thus the nodes using reactive routing protocols do not need to update their routing tables as frequently and do not maintain routes for all nodes in the network. When a node using a reactive protocol requires a route to a new destination, it initiates a route request and must wait until the route is discovered. Reactive routing protocols have the disadvantage of delay in finding routes to new destinations traded against the savings of not needing to maintain tables for all possible routes.

2.2 MANET Routing Protocols

In this section we summarize the MANET routing protocols considered in this paper that have been implemented in ns-3 as of release 3.9.

2.2.1 DSDV

The DSDV (destination-sequenced distance vector) protocol [13] uses the Bellman-Ford algorithm to calculate paths. The cost metric used is the *hop count*, which is the number of hops it takes for the packet to reach its destination. DSDV is a table-driven proactive protocol, thus it maintains a routing table with entries for all the nodes in the network and not just the neighbors of a node. The changes are propagated through periodic and trigger update mechanisms used by DSDV. Due to these updates, there is a chance of having routing loops within the network. To eliminate routing loops, each update from the node is tagged with a *sequence number*. The sequence number from each node is independently chosen but it must be incremented each time a periodic update is made by a node. The sequence number of normal update must be an even number, since each time a periodic update is made the node increments its sequence number by 2 and adds its update to the routing message it transmits. The node cannot change the sequence number of other nodes. If a node wants to send an update for an expired route to its neighbors, only then it increments the sequence number of the disconnected node by 1. The nodes receiving this update will then look at the sequence number and if it is odd, will remove the corresponding entry from the routing table. Mobility of the nodes in MANETs causes route fluctuations, for which DSDV uses *settling time* to dampen.

2.2.2 AODV

AODV (ad hoc on-demand distance vector) [12, 11] is a distance vector routing protocol that operates reactively to reduce overhead finding routes only on demand. When a route does not exist to a given destination, a route request (RREQ) message is flooded by the source and by the intermediate nodes if they have no previous routes in their table. Upon receiving a RREQ message, the receiving node will record the route information in its own routing table. Once the RREQ message reaches the destination or an intermediate node, the node responds by unicasting a route reply (RREP) message back to the neighbor from which it first received the RREQ message. As the RREP message is forwarded back along the reverse path, nodes along this path set up forwarding entries in their routing tables, pointing to the node from which they received RREP message. AODV uses sequence numbers created by the destination for every route entry to avoid routing loops. Routes with the greatest sequence number are preferred in selecting routes from the source to the destination. AODV is contained in the standard ns-3 distribution, but there are problems with its implementation as of release 3.9, as will be shown in Section 4.

2.2.3 OLSR

OLSR (optimized link state routing) [2] is a proactive routing protocol, for which routes to all destinations within the network are discovered and maintained before a packet is sent from source to destination. OLSR uses HELLO and topology control (TC) messages to discover and broadcast *link state* information throughout the network regularly. Nodes receiving this topology information compute next hop destinations for all nodes in the network. HELLO messages at each node discover 2-hop neighbor information and select a set of multipoint relays (MPRs). MPRs are responsible for transmitting broadcast messages and constructing link state. OLSR floods topology data frequently enough over the network to make sure all nodes are synchronised with link state information. OLSR is also contained in the standard ns-3 distribution, and performs as expected for the analysis in this paper.

2.3 Previous DSDV Simulations

DSDV has been implemented and analysed in a number of simulation tools. A discrete event, packet-level routing simulator called MaRS (Maryland Routing Simulator) was used to evaluate DSDV performance under different network scenarios [4], assumed a channel bandwidth of 1.5 Mb/s with all data packets 512 B long, and interarrival times exponentially distributed with a mean of 300 ms. This analysis showed that for fraction of packets delivered, the proactive DSDV routing protocol outperforms on-demand protocols for both low and high mobility cases. Furthermore, the average end-to-end delays for DSDV show the minimum delay characteristics.

Most relevant to this paper is the ns-2 predecessor to ns-3. The performance of DSDV, AODV, and DSR have been compared using ns-2 [1, 5]. DSDV exhibits low delay because only packets belonging to valid routes at the sending instant get through compared to AODV and DSR. However, DSDV

Attribute	Defaults	Summary
PeriodicUpdateInterval	15 s	Time interval between exchange of full routing tables among nodes
EnableWST	true	Enables Weighted Settling Time for the updates before advertisement
SettlingTime	6 s	Minimum time duration an update is stored before transmission
WeightedFactor	0.875	Weighted factor for the settling time if EnableWST is true
EnableBuffering	true	Enables buffering of data packets if no route to destination is available
MaxQueueLen	100	Maximum number of packets that can be queued
MaxQueueTime	30 s	Maximum time duration for which packets can be queued
MaxQueuedPacketsPerDst	5	Maximum number of packets that can be buffered per destination
Holdtimes	3	Number of times PeriodicUpdateInterval to purge a route
EnableRouteAggregation	false	Enables aggregation of DSDV updates over a period of time
RouteAggregationTime	1 s	Time over which DSDV updates are aggregated

Table 1: DSDV attributes and default values

has the highest overhead of the three protocols because of the table updates flooded throughout the entire network. This analysis also shows DSDV's inability to converge when the mobility is high, especially at high loads.

The ns-2 implementation of DSDV 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. DSDV MODULE FOR ns-3

This section describes our implementation of DSDV, which has been included in ns-3.10 stable release [10]. The main components of the DSDV implementation are routing update mechanisms, route table creation, and route maintenance. DSDV maintains valid routes and flushes out invalid routes based on the periodic update interval. We implemented an optional packet buffering mechanism that was not part of the initial DSDV design [13]. This feature is implemented for testing the performance of the protocol with and without packet buffering and also to provide users with more options. All the attributes used in this implementation are listed in Table 1. The details of the classes implemented are explained next.

3.1 Class Interaction

The relation between all the classes implemented in this module are shown in Figure 1. We implemented the DSDV routing protocol `ns3::dsdv::RoutingProtocol` in ns-3 by extending from the abstract base class `ns3::Ipv4RoutingProtocol`. The `ns3::dsdv::DsdvHeader` is extended from `ns3::Header`. We have also declared `ns3::dsdv::RoutingTableEntry` to store the updates of a node and `ns3::dsdv::RoutingTable` to store all these entries in a table. Similarly we have declared the `ns3::dsdv::QueueEntry` class to store a packet and `ns3::dsdv::RequestQueue` to store all the queued entries. The main class that glues all these together is the `ns3::dsdv::RoutingProtocol` class. An in-depth explanation of all these classes is presented in the following sections.

3.2 Header

The DSDV message header (`DsdvHeader`) is 32 bits wide with the total header size of 12 bytes as shown in Figure 2. The fields in the DSDV header are the node's IP address, the number of hops required to reach that node, and its last known sequence number. The latter two are 32-bits long in our implementation to provide word alignment and allow simulation of very large networks, even though the ns-2 implementation used 16-bit fields. Note that unlike AODV and OLSR, there is no DSDV RFC to guide standards compliance.

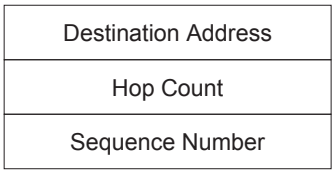

Figure 2: DSDV message header

DSDV is encapsulated in User Datagram Protocol (UDP) segments that are then encapsulated in IP packets, as shown in Figure 3.

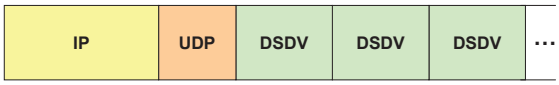

Figure 3: DSDV header encapsulation

3.3 Routing Table

The structure of the DSDV `RoutingTable` is implemented as follows. Each entry implemented by the `RoutingTableEntry` 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: its IP address, interface address, a pointer to its ns-3 net device, last known sequence number of the node, hop-count to reach the node, timestamp of the last update received for the node, and the settling time for that node. Also, we maintain a boolean value that specifies whether the entry for this node has changed since the last periodic update. This helps filter DSDV updates that are broadcasted through the trigger update mechanism. The `RoutingTable` class has methods to add, delete, update, look up, and print entries. It also defines the event functions explained in section 3.5.2.

DSDV maintains two routing tables: a permanent routing table and an advertising routing table. These tables store the permanent stable routes and the recently received routes respectively. The recently received routes might be unstable; therefore, when the node identifies a route to be stable, it

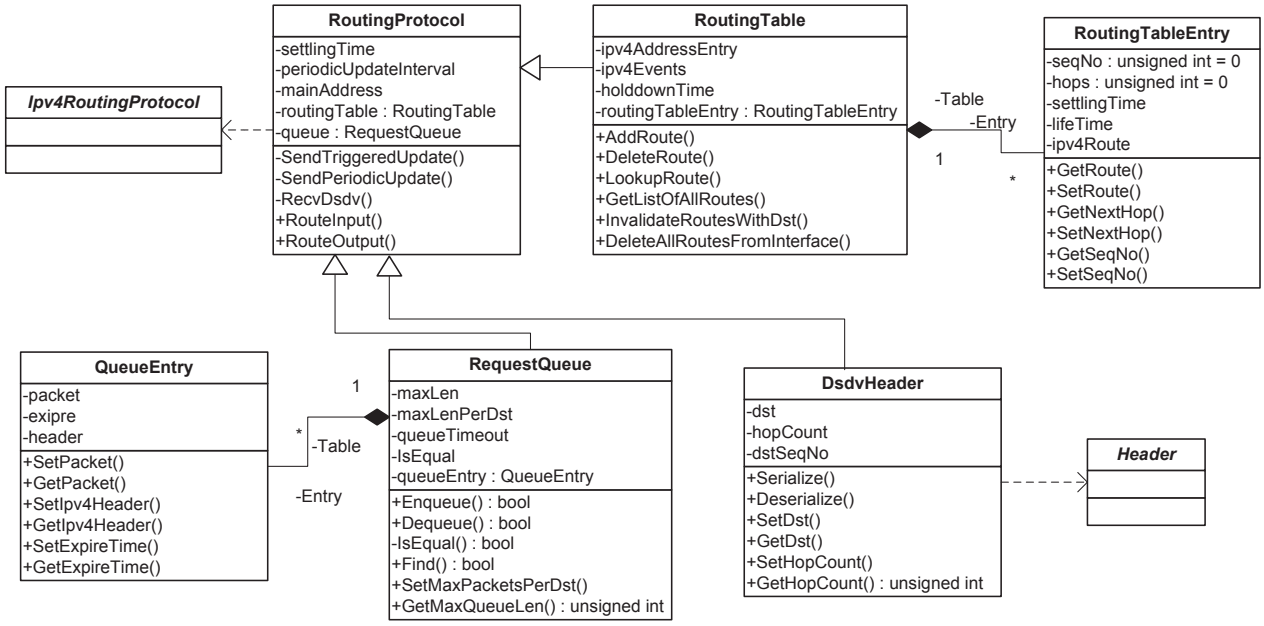

Figure 1: DSDV class diagram

advertises that route and moves it to the permanent routing table. This mechanism of identifying stable routes is done using *SettlingTime*, explained in detail in section 3.4. Furthermore, a node can identify the stability of a route based on the sequence number and hop count received through the update, explained in Section 3.5.

3.4 Routing Advertisements

A node combines all the DSDV messages that it has to transmit into a single packet over `RouteAggregationTime`, if `RouteAggregation` is enabled. However, to keep the packet size under the maximum transmission unit (MTU) in the implementation, we split the packets and send them separately if the packet size is longer than the MTU. As mentioned earlier, DSDV sends both periodic update messages and trigger messages. As soon as the routing protocol in the node is initialised, the node broadcasts its DSDV update message to the network to announce its presence. Each node will periodically broadcast its own routing table and all the nodes that are in range of this advertisement will use this information to update their routing tables. They may further trigger these updates to other nodes in their broadcast range. This mechanism is also used to keep the neighborhood relationship alive. One of the attributes that can be set for the routing protocol is the duration between these periodic updates, known as *periodic update interval*, using `PeriodicUpdateInterval`. It specifies the time duration for which a node has to wait before broadcasting its routing table. A node uses the trigger update mechanism when there are only a few updates to be transmitted. However, if the node identifies that the number of updates sent per trigger is comparable to that of a periodic update, then it sends a periodic update instead.

One more feature of DSDV routing protocol is the *settling time*, which is used to prevent the advertisement of an unstable route that arrives at the node before a stable route. Since DSDV uses broadcasting to propagate these changes, it would create unnecessary overhead in the network. Thus, a node waits for the period of `SettlingTime` before propagating any update. However to make sure updates for stable routes are not delayed, we use the attribute `WeightedFactor`. This is used to calculate the weighted average of the settling times for the updates received from a node. If the update is for an old and stable route, the settling time decreases. A node can not process multiple update messages simultaneously. If the nodes are highly mobile, the node might have to send many updates as there would be a lot of route changes. This will lead to more overhead in the network that may increase the number of collisions. To reduce overhead, we use `RouteAggregation`. This optional feature enables multiple update messages to be sent out as a single update message. The period over which routes are aggregated can be modified by `RouteAggregationTime` attribute.

3.5 Processing of Updates

As mentioned in Section 3.4, a node might receive many updates stacked within a single packet. Since the DSDV header size is fixed to 12 bytes, we iterate over the packet until it is empty to extract all the 12 B DSDV control messages. Each message is processed as it is extracted. We first verify the destination address in the extracted message. If it is same as the node's IP address, the message is discarded. If not, the protocol verifies whether the received update is for a new route with a valid sequence number. In this case the route is added to the permanent routing table and broadcast immediately. Otherwise, if the node already has an update for that IP address the protocol verifies the sequence number. If the sequence number is odd and if the node from which this update was received is the next hop neighbor in the table, then the route is deleted from the routing table and triggers an update of this broken route to other nodes immediately. However if the sequence number is valid we have three cases in which the sequence number can relate to the sequence number from the table:

- *Received* > *Local*: The protocol verifies the received hop-count with the local value of hop count. If they are not equal, the node updates its local entries in the advertising routing table and waits for settling time period if `SettlingTime` is enabled. This is implemented using events in ns-3. This mechanism is explained briefly in section 3.5.2. If the received hop-count is same as the local value, then the node does not wait for the settling time interval as this is an update for the stable route.

- *Received* = *Local*: If the received hop-count is less than the local value, then the local value is updated and the protocol waits for settling time to make sure that this update is not an unstable one. If it does not receive any further update for that destination address, the protocol updates the permanent table with this update and triggers this update back to all its neighbors. However if the received hop-count is greater than or equal to the local value, the message is discarded.

- *Received* < *Local*: The protocol discards this update message as it already has a most recent update from that destination.

After processing messages from the packet, the `SendTriggeredUpdate` method will be called. `SendTriggeredUpdate` iterates over the advertising routing table, computes all the needed updates, and creates a new packet with these updates and broadcasts.

3.5.1 Stale Entries

DSDV has a mechanism of removing stale entries from the node's routing table. If a node does not receive any updates for a destination over a period of time, it removes that entry from the routing table. In our implementation, DSDV waits for `Holdtimes` × `PeriodicUpdateInterval` interval. The default value of `Holdtimes` is set to 3, i.e. a node waits for 3 times the `PeriodicUpdateInterval` before deleting the route. Furthermore, the node must delete all the routes for which the deleted neighbor was the next hop.

3.5.2 Event Processing

In the implementation of DSDV we use EventId `ns3::EventId` to schedule events and keep track of them. These are declared in the `RoutingTable` class. The IP address of a node is mapped to the event id. We use these events to keep track of the updates in advertising table and broadcast them when their settling time is complete. When a node receives an update for a destination that is already waiting in the advertising table, the running event might be replaced by a new one depending on the new update received.

3.6 Packet Buffering

We have implemented a buffering mechanism for DSDV although it is not part of the DSDV as originally described [13], to allow fairer comparisons with *disruption-tolerant networks* (DTN) and *domain-specific* MANET routing protocols that do buffer packets that cannot be immediately sent [16, 14]. We implemented two classes in a manner similar to the routing table implementation. `QueueEntry` class is the entry that is stored in the queue, implemented from `RequestQueue` class. If the destination address for a packet is not present in the protocol's routing table, then the packet is buffered. As DSDV is a proactive protocol, it does not initiate any route discovery mechanism to identify the route to that destination. It has to only rely on the messages received from its neighbors through trigger and periodic updates. DSDV will periodically verify the buffer and look for packets with valid routes in the routing table and transmits them. By default, our DSDV implementation buffers up to 5 packets per destination. This can however be changed by modifying the `MaxQueuedPacketsPerDst` attribute. Furthermore, packets which are buffered for a long time will be dropped from the queue. The time interval for which a packet can be buffered is set using `MaxQueueTime`. By default packet buffering is enabled, but this can be disabled by setting `EnableBuffering` to `false`.

3.7 Parameter Tuning

An advantage of DSDV is that it is relatively simple compared to other MANET routing protocols. It is also similar to the conventional wired distance-vector routing protocols, with only minimum adaptations made. However, the drawback of DSDV is that its periodic overhead for broadcasting is unavoidable even if the network is static. If the node density increases in the network, the routing table will also become larger. This leads to more updates with larger packet sizes. With a highly dynamic network, the routing updates may take up the available bandwidth of channel. Furthermore, before the time of update, intermediate nodes may use stale information to forward packets. Thus proper choice of `PeriodicUpdateInterval` and `SettlingTime` is important in a highly mobile environment.

4. DSDV MODULE EVALUATION

To evaluate the performance of our DSDV routing protocol implementation, we performed simulations using the ns-3.9 version of the network simulator[1]. To verify its functionality, we investigate the DSDV performance with varying node densities as well as compared to the other existing MANET routing protocols in ns-3: OLSR and AODV. Note that a comparison to the DSR implementation currently in progress is future work.

4.1 Performance Metrics

The performance metrics for evaluation of the DSDV routing protocol are packet delivery ratio (PDR), routing overhead, and delay.

- **Packet Delivery Ratio PDR**: The number of packets received divided by the number of packets sent by the application.

- **Routing Overhead**: The fraction of bytes used by the protocol for DSDV control messages

- **Delay**: The time taken by the packet to reach the destination node's MAC protocol from the source node's MAC protocol.

4.2 Simulation Setup

We performed the simulations over an area of 1500 × 300 m^2. All the simulations were averaged over 10 runs with each simulation running for 1000 s. Simulations were performed with varying node densities: 10, 20 and 30 nodes.

[1]Before our DSDV was included in the ns-3 distribution in ns-3.10.

The communication model is peer-to-peer communication with as many flows as the number of nodes in the network. We initially performed some simulations with 1000 byte packets but observed that the PDR was low, therefore we used a packet size of 64 bytes based on previous study [1]. All the nodes are configured to send 4 packets/s. Using this lower packet size, we can correctly evaluate the performance of the protocol. We use the ns-3 `On-Off` application to generate CBR (constant-bit rate) traffic. The 802.11b MAC is used over Friis propagation loss model to limit the transmission ranges of nodes. The transmit power was set to 8.9048 dBm to achieve a 250 m transmission range. The mobility model used is random waypoint with random velocities from 0 – 20 m/s and pause times of 100 – 800 s. When comparing DSDV performance against AODV and OLSR, we use 0 s pause time. DSDV performance with optional buffer mode enabled was analysed. We used the default DSDV parameters values described above except for `PeriodicUpdateInterval` which was varied among {4, 5, 8, 12, 15, 30} s and `SettlingTime` which was varied among {0, 1, 2, 3, 4, 5, 6} s. Some of the simulation parameters were chosen based on the previous MANET comparison studies [1].

4.3 Simulation Analysis

In the first scenario, we vary the pause time in therandom waypoint mobility model so that we can analyse the performance of DSDV in both mobile and static scenarios. For this scenario, the `PeriodicUpdateInterval` is set to 15 s and `SettlingTime` is 6 s. Figure 4 shows the variation of PDR by varying the pause times.

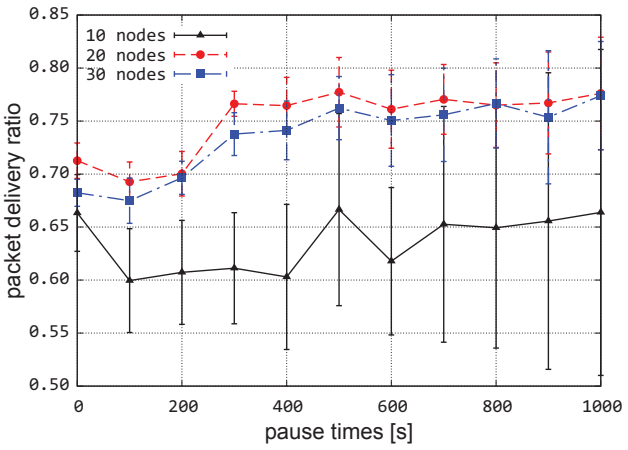

Figure 4: PDR with varying pause time

We can see that as the number of nodes is increased the packet delivery ratio also increased. This is due to the fact that when there are only 10 nodes, the chances of link breaks and network partitioning is more likely to happen than when there are more nodes making the network connected for most of the time. However, PDR for 20 nodes is greater than that for 30 nodes for all pause times. This might be because as the node density increases, the routing overhead also increases and this leads to more collisions in the network. Note the 95% confidence-interval error bars in Figure 4. As the pause time increases, so does the variation in packet delay (as depicted by error bars) for all the 3 curves for 10 nodes, 20 nodes and 30 nodes. This can be attributed to how the nodes were positioned in the network initially since very long pause times will reduce movement from the initial position.

The routing overhead for different node densities with varying pause times is shown in Figure 5. This plot shows that overhead increases with the number of nodes. This is expected for DSDV since it is a proactive protocol and every node keeps track of all the other nodes in the network; when a node sends out a periodic update, it is flooded to all other nodes. Depending on the changes based on an update received, a node may further trigger updates to other nodes.

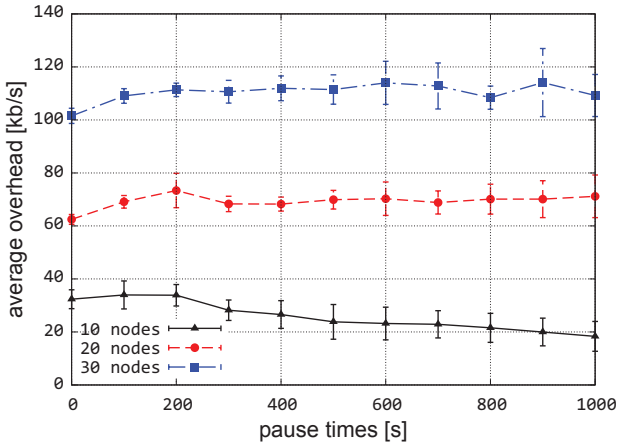

Figure 5: Overhead with varying pause time

The overhead as shown in Figure 5 slightly increased for all the 3 curves moving from a pause time of 0 s to 100 s. With zero pause time the nodes collect less information from the network because they are continuously moving. With the larger pause time of 100 s they collect more information from the network. This translates to more updates. Furthermore, as pause time is increased, the overhead is reduced.

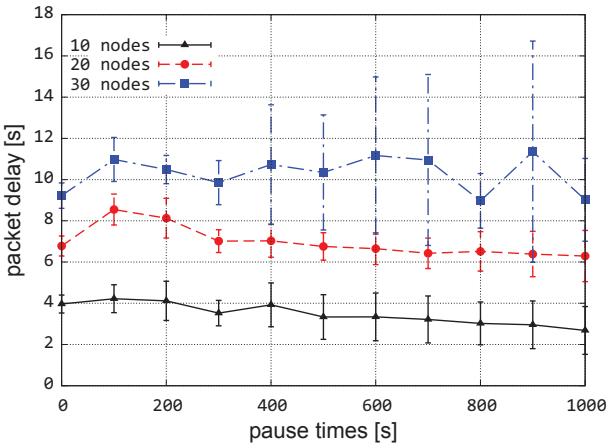

Figure 6: Packet delay with varying pause time

We also consider the packet delay for data packets between source and destination. Figure 6 shows the variations in packet delay (as depicted by error bars) increase as the pause time is increased for all the 3 curves for 10 nodes, 20 nodes

and 30 nodes. This is because as the pause time is increased the nodes are immobile for longer durations and thus the link connectivity depends on the position of the nodes, which directly affects the packet delay.

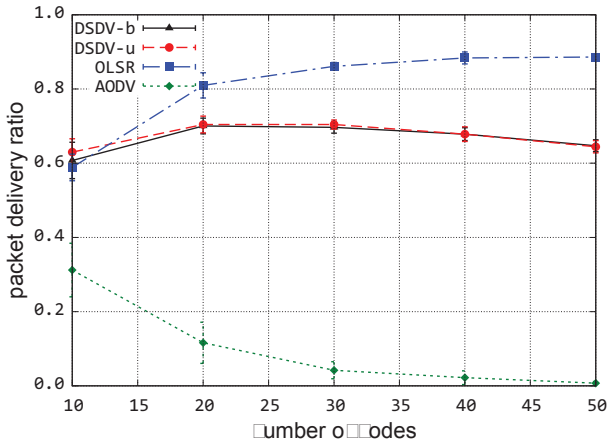

Figure 7: PDR with varying node density

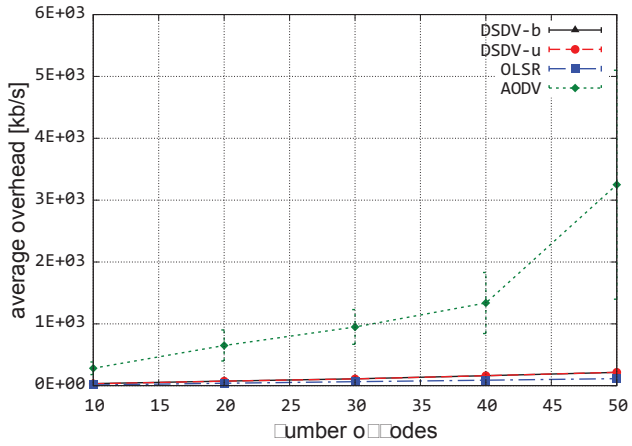

Figure 8: Overhead with varying node density

In Figure 7 we compare the packet delivery ratio of existing MANET routing protocols implemented in ns-3 with DSDV. From the plot we can clearly see that OLSR outperforms DSDV-buffer mode, DSDV-unbuffer mode, and AODV. This is expected as OLSR implementation in ns-3 exchanges `TC` messages every 5 s [2], thus the routing tables are computed/re-computed every 5 s. However DSDV uses a `PeriodicUpdateInterval` of 15 s making the convergence of nodes running OLSR quicker compared to those running DSDV. In DSDV the routes are not always accurate as it depends only on periodic and trigger messages to update the routes. AODV's performance was expected to be higher, however the current implementation of AODV has some bugs that need to be fixed[2].

[2]We have been working with ns-3 developers to report AODV performance issues, and the situation has been improving.

In our analysis, we also compare the routing overhead involved with all these protocols, AODV incurs significant overhead shown in Figure 8. DSDV and OLSR generates about 112 kb/s and 65 kb/s of routing overhead respectively for 30 nodes. However as the number of nodes increases, the overhead increases as well. For a 50 node simulation, DSDV incurred an overhead of 215 kb/s compared to 113 kb/s for OLSR.

We analyse the packet delay for these protocols. The packet delay is greater for DSDV when compared with OLSR as shown in Figure 9. For a 30 node simulation, packet delay for DSDV was 10 s where as it was 6 s for OLSR. Since these scenarios were generally connected, the results for DSDV-buffer and DSDV-unbuffer mode results were not significantly different. The performance of the ns-3 AODV model is considerably less than expected.

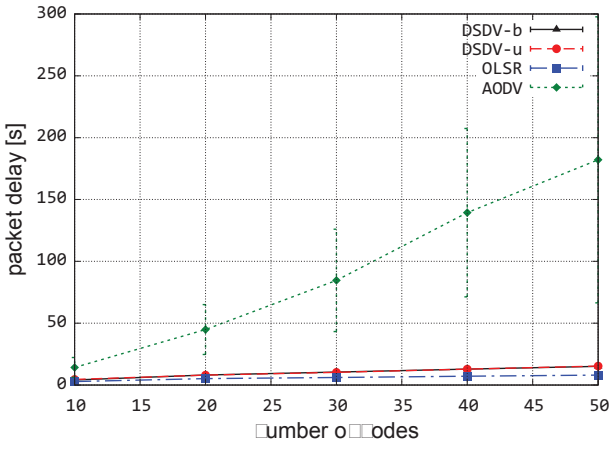

Figure 9: Packet delay with varying node density

5. CONCLUSIONS

In this paper we presented the implementation of the DSDV MANET routing protocol in ns-3. A detailed explanation of the components and how each class interacts with one other is also provided and the attributes that can be modified in the protocol are presented. We analysed our DSDV implementation in varying node densities and compared its performance against OLSR and AODV. Our results indicate that DSDV overhead increases as the node density increases. PDR performance of the DSDV is inversely affected as the overhead increases.

As part of the future work, we will explore the effects of buffering on sparser networks. We also plan to analyse the routing performance of the MANET protocols including DSR which we are implementing in ns-3 at the University of Kansas [15].

Acknowledgments

We would like to acknowledge the assistance of Abdul Jabbar and the members of the ResiliNets research group for their advice and suggestions that helped us with this implementation. We would also like to acknowledge Tom Henderson and the ns-3 development team for their responsiveness to issues with the relatively immature 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 *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 85–97, 1998.

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

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

[4] S. R. Das, R. Castañeda, and J. Yan. Simulation-based Performance Evaluation of Routing Protocols for Mobile Ad Hoc Networks. *Mob. Netw. Appl.*, 5(3):179–189, September 2000.

[5] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 195–206, 1999.

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

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

[8] The network simulator: ns-2. `http://www.isi.edu/nsnam/ns`, December 2007.

[9] The ns-3 network simulator. `http://www.nsnam.org`, July 2009.

[10] Dsdv code review. `http://codereview.appspot.com/1668042`, June 2010.

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

[12] C. Perkins and E. Royer. Ad-hoc On-demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 90–100, February 1999.

[13] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the ACM Conference on Communications Architectures, Protocols and Applications (SIGCOMM)*, pages 234–244, 1994.

[14] K. Peters, A. Jabbar, E. K. Çetinkaya, and J. P. Sterbenz. A Geographical Routing Protocol for Highly-Dynamic Aeronautical Networks. In *IEEE Wireless Communications and Networking Conference (WCNC)*, March 2011. to appear.

[15] J. P. Rohrer, E. K. Çetinkaya, A. Jabbar, D. Broyles, K. Peters, H. Narra, Y. Cheng, K. S. Pathapati, and S. Simpson. ResiliNets models and tools for ns-3 network simulator. `http://wiki.ittc.ku.edu/resilinets/Models_and_Tools_for_ns-3_Network_Simulator`, September 2010.

[16] T. Thedinger, A. Jabbar, and J. Sterbenz. Store and Haul with Repeated Controlled Flooding. In *Proceedings of the 2nd IEEE International Workshop on Mobile Computing and Networking Technologies (WMCNT)*, pages 728–733, October 2010.

[17] E. Weingartner, H. vom Lehn, and K. Wehrle. A Performance Comparison of Recent Network Simulators. In *IEEE International Conference on Communications (ICC)*, pages 1–5, June 2009.