Routing protocols in MANET confront challenges that are not usual in ordinary wired networks. Node movement, intermittent connections, and lack of infrastructure are some of these challenges. Several routing protocols have been proposed to provide better performance delivery and address these challenges. However, an environment is necessary to compare the efficiency of these protocols. In this paper, we present an overview of the SiFT routing protocol and its implementation in ns-3 [10], an open source network simulator. Performance of this protocol is compared with other implemented MANET protocols in ns-3.

Categories and Subject Descriptors
I.6 [Simulation and Modeling]: General, Model Development, Model Validation and Analysis; C.2.2 [Computer-Communication Networks]: Applications—SiFT routing protocol

General Terms
Implementation, Simulation, Analysis, Verification

Keywords
SiFT, MANET routing protocol, ns-3 simulator, AODV, DSDV, DSR, OLSR

1. INTRODUCTION

Mobile ad hoc networks (MANET) are a solution for rapid network development when network infrastructure is not available and nodes move in the environment. These two MANET characteristics impose other challenges such as weak or intermittent wireless links and dynamic network topology leading to challenges in routing protocols in such networks. On the other hand, the wireless broadcast medium makes flooding as a tempting solution in these networks. However, higher cost of delivery in flooding is a main concern in this method. Therefore, proposing routing algorithms with a higher delivery rate and lower delivery cost has been a main area of research in MANET.

Many routing protocols have been proposed for MANETs including AODV [13], DSDV [12], DSR [7] and OLSR [4]. Furthermore, other protocols with different characteristics considering trajectory or geolocation of nodes have been proposed. Since those first emergent have been implemented vastly in experimental and simulation environments, their performance are considered as a base line for comparing other protocols.

In many studies, ns-2, an open source discrete network simulator, had been used for the simulation. The successor ns-3 [10] currently supports AODV, DSDV, DSR and OLSR. However, implementing recently proposed protocols like SiFT and LAR that are based on node location makes ns-3 a more convenient simulator in other studies. As a contribution to this goal, we have implemented SiFT [1] in ns-3 to provide another base line for comparing protocols.

In this paper, we present our ns-3 implementation of the SiFT routing protocol and compare its performance with other available MANET protocols in the current version of ns-3.21. This paper is organized as follows: In section 2, related works in MANETs are reviewed. In section 3, we provide the detail design of SiFT implemented in ns-3. In section 4, we compare its performance with other MANET routing protocols in ns-3. Finally, in section 5 we conclude our study.

2. BACKGROUND AND RELATED WORK

The design of routing protocols for MANETs is different from conventional networks. For instance, conventional routing protocols generate a large amount of routing related traffic which is considered unacceptably high overhead in wireless networks. Furthermore, link state routing protocols need to know the topology of the network, while this is not easily achievable in MANETs. The network topology in MANETs, constantly changes, when nodes move with re-
spection to one another. This mobility leads to changing link conditions from totally disconnected to weak and intermittent connectivity that makes conventional routing protocols incapable of converging. On the other hand, these changes generate many control packets that waste precious wireless bandwidth. Therefore, MANET routing protocols have operations that are not common in conventional routing algorithms.

2.1 Routing Protocols in MANET

Various routing algorithms have been proposed to find a path from source to destination in MANETs. Location-based, trajectory-based, and social-aware are some of these techniques. However, MANET routing protocols can be categorized in two main groups as proactive and reactive, as described next.

2.1.1 Proactive Routing Protocols

Proactive protocols maintain path resulting from an individual source to all available destinations and update this information periodically by distributing their routing information among each other. Since any potential path in the network is identified, they are available for use instantaneously. However, high mobility is one of the main concerns with this technique. This may cause a discovered path to no longer be available at the time of forwarding. Therefore, high mobility prevents these protocols from converging or providing accurate information during forwarding. Another concern is the size of routing tables that increases rapidly in dense or large networks and contain all potential paths.

2.1.2 Reactive Routing Protocols

This group of routing protocols do not keep paths from an individual source to all possible destinations. In contrast, they discover an available path at the time of forwarding before a packet is sent. Therefore, reactive protocols do not need to update routing tables by propagating routing information to other nodes. Since routes are discovered when they are required, there is usually a delay between the request time to send the first packet and its actual transmission. When the path is set up in each node accordingly, it is used until the end of the transmission or the network topology changes. While this group of protocols do not suffer from the overhead of exchanging routing related information, the delay in the beginning of each transmission may be considerable.

2.2 MANET Routing Protocols

Many MANET routing protocols rely on some form of flooding. In pure flooding, each node sends a packet to all of its neighbors. If any of the neighbors receive the packet that it hasn’t already received, it adds its identification to the header and forwards to its neighbors, until the packet reaches the destination or it is assumed there is no path between the source and destination by expiration of the lifetime of the packet. In order to decrease the cost of delivery, controlled flooding keeps track of received packets so they aren’t re-forwarded. AODV and DSR, which are compared with SiFT in this paper, use some form of controlled flooding. SiFT and other common MANET protocols are described in the following sections.

2.2.1 SiFT

SiFT [1] is a simple reactive routing protocol. It uses geolocation of nodes to perform controlled flooding. Before sending any packet to the destination, SiFT acquires the geolocation of the destination. By knowing its own geolocation, the source node calculates a trajectory between itself and the destination and encodes this information to each packet. When a packet is forwarded to the neighbors, each neighbor calculates its distance from the trajectory and the last forwarding node; then it sets a timer proportional to the value of the calculated distance. If the timer times out and the node hasn’t received the same packet from other neighbors, it forwards the packets to its own neighbors, otherwise it drops the packet. Duplicate packets are recognized by assigning a sequence number to each packet. With this simple calculation, only those nodes which are closer to the trajectory and farther from previous sender forward the packet toward the destination. The rest of nodes which hear this transmission ignore forwarding. Since no specific information is kept except the location of the destination at the time of transmission, all mobility in the network is projected in the forwarding. Figure 1, (based on [1]), shows forwarding process in SiFT protocol.

Figure 1: Forwarding process in the SiFT protocol

2.2.2 AODV

AODV (Ad-hoc On-demand Distance Vector) [13][11] is a reactive routing protocol based on the distance vector algorithm. Nodes do not keep routes in their routing table permanently. If a route does not exist at the time of forwarding, a route request (RREQ) message is sent to the sender neighbors by flooding to discover the path between the source and destination. On receiving each message, each intermediate node records the path from the source to that node. When the RREQ message is received by the destination, it replies with unicast route response (RREP) message over the same path discovered by RREQ. When intermediate nodes receive RREP, they set up a forwarding path in their routing table to the destination from which they have received the RREP. The destination node also assigns a sequence number to the path to avoid routing loops. AODV is part of standard ns-3 distribution.
2.2.3 DSR

DSR (Dynamic Source Routing) [6][5] is an on-demand routing protocol that works reactively. It is very similar to AODV in terms of conserving wireless bandwidth by eliminating periodic table update messages. If a path is requested and it is not in the routing table of the source, it sends a route request message (RREQ) to all of its neighbors. The source also assigns a sequence number to each RREQ. The next hop will add its address to the source route in the header and flood the message if it does not have the same sequence number in its table. Similar to AODV, sequence numbers eliminate the formation of routing loops. The destination replies to the source over the reverse path recorded in the RREQ with a route reply (RREP) message. DSR also uses route cache to record all information extracted from source route in a data packets. DSR was developed by our ResiliNets research group [2], and is now in the standard ns-3 distribution.

2.2.4 DSDV

DSDV (Destination Sequenced Distance Vector) [12] is a table-driven proactive routing protocol. This protocol uses a distributed version of the Bellman-Ford algorithm. Each node keeps a table with the shortest distance and the first node on that path to every other node in the network. Since all paths to other nodes are maintained in each node’s table, paths are always ready to use. This reduces the initial delay to discover the path between the source and the destination in comparison to reactive routing protocol. However, routing information must be exchanged among all nodes, which produces high traffic overhead, particularly in a highly mobile environment. Small changes in the network topology are reported by incremental updates, while substantial changes are done by a full dump of the updated table. DSDV was also implemented by our ResiliNets research group [9] and is in standard ns-3 distribution.

2.2.5 OLSR

OLSR (Optimized Link State Routing) [3] is a proactive routing protocol that uses an efficient optimized link state packet forwarding mechanism. In order to optimize link state routing, OLSR keeps a subset of designated neighbors called multipoint relays, for each node. If a node receives a link state update from a neighbor which is not in its set, it processes the packet, but does not forward it to its other neighbors. Using this subset reduces the overhead of the protocol to propagate the link states updates. In addition, the link state update mechanism does not produce any control packets when a link is broken or a new link is added. OLSR achieves higher efficiency in a highly dense environment [8], and is in the standard ns-3 distribution.

3. SIFT MODULE FOR ns-3

In this section, we describe our implementation of SiFT in ns-3.21, using C++. The main component of SiFT is the routing component that calculates whether the node should forward a received packet or not. This component has a relation with geolocation component which returns the location of nodes.

SiFT routing protocol in ns-3 is implemented in ns3:sift::SiftRouting, that is the extension of the abstract base class ns3::Ipv4L4Protocol. The SiFT header defined in ns3:sift::SiftHeader, is also an extension of the abstract class ns3::Header. This is a shim header between the transport and network layer and adds information about the trajectory to each packet, encoded with a line from the source to destination, which makes calculation very simple. In addition, pairwise lines can be used to represent a trajectory [1], which provides more accuracy for designated paths such as service roads in vehicular networks. However, we tested SiFT with a single line that connects the source and destination. The SiftGeo class, implemented in ns3::sift::SiftGeo keeps and handles geographical location of each nodes. Since SiFT uses the broadcast service of the network layer and does not maintain any path information, no additional classes are required. However, a buffer is defined in each node that holds recently received packets. Figure 2 shows the interaction among classes for current implementation.

3.1 Routing Process

SiFT is categorized as a trajectory based forwarding (TBF) protocol suitable for ad hoc wireless networks. It is assumed nodes know their geographical position acquired by some mechanism such as GPS. Having this information, source nodes encode trajectory information into each packet. In contrast to source routing, these algorithms do not need to know the next node to forward the packet. Broadcasting is used to forward packets, while forwarding nodes are selected dynamically in each hop. Whenever a node receives a packet, its role changes from receiver to sender and decides whether to forward the packet or drop it. This decision is only based on the position of the node with respect to the trajectory. Lack of using control information and deciding about next forwarding node in each hop makes this protocol simple and appropriate for environment with high mobility.

The following formula calculates distance from the trajectory and last forwarding nodes reducing control overhead and energy consumption. Upon receiving a packet, each receiver sets a timer based on its position to the trajectory:

\[ T_{out} = \alpha D_t / D_f \]

where \( D_t \) is the distance of the node from trajectory, \( D_f \) the distance from the last forwarding node, and \( \alpha \) is a constant. If a copy of the same packet is received before timeout, the packet is removed from the forwarding queue. With this simple calculation, the node that has the closest distance to the trajectory and farthest distance from the previous forward node is likely to be successful to forward the packet along the trajectory.

In order to eliminate cycles, each node maintains a list of recently received packets that are removed when their lifetimes are expired. Furthermore, each node considers a threshold for the farthest distance from the trajectory. Since nodes may not be in the wireless range of each other, they may not get a copy of the new transmission of the packet and start sending another copy while other copies are on their route to the destination. This leads to receiving multiple copies at the destination. In order to eliminate some of these duplications, nodes that are farther than the threshold stop sending these packets. This threshold may adjust to the wireless range of a node.

3.2 Header

The SiFT protocol adds its own header for routing over IP, inserted between the network and transport layer headers, 320 bits long as illustrated in Figure 3.
This header carries information related to the SiFT routing protocol including header sequence number, source and destination IP address, and geolocation of the source, destination, and last sender, shown in Figure 4. Since broadcasting is used in the network layer, destination addresses are changed to the broadcast IP address. Therefore, the IP address of the source and destination must be kept in the SiFT header. In addition, keeping geolocations of the source, destination, and the last forwarding node are necessary to calculate distance from the trajectory. TTL is also kept in the SiFT header in order to eliminate loops, because TTL in the IP header is reset in each step due to broadcasting. Since it is assumed that nodes can get their geolocation by some mechanism such as GPS, the sender knows the location of the destination and adds it to the packet before transmission. This information is used to calculate the trajectory between the source and destination by other nodes on the path to the destination. The protocol number 47 is used in the header and passed to IP to identify the next header in the packet. The generic routing encapsulation (GRE) is used, because there is currently no standard protocol number assigned to SiFT.

Class SiftHeader, which defines SiFT header objects, includes other standard ns-3 methods for serialization and deserialization, header printing, and assigning values to header parameters. Since there is no standard for the SiFT protocol, we designed the header format for this implementation.

The only routing service that SiFT receives from the network layer is broadcasting. If a node receives a packet and it is the closest node to the trajectory, it updates the SiFT header based on its status to the trajectory and requests the network layer to broadcast the packet. Therefore, all other network layer routing protocols are disabled. The next forwarding node also decides forwarding the packet based on information in the SiFT header rather than IP header. This process continues until either the packet reaches its destination or its life time expires.
4. PERFORMANCE EVALUATION

We use the current version of the simulator ns-3.21, to evaluate the performance of our SiFT routing protocol implementation. We investigate the performance of SiFT in an environment with 30 nodes while the number of data flows and mobility are changed in each run.

4.1 Performance Metrics

Three metrics are used to evaluate SiFT routing protocol including Packet delivery ratio (PDR), delay and number of hops.

- Packet Delivery Ratio is the number of packets received divided by the number of packets sent by the source node. All duplicate packets in the destination are dropped and not considered in this calculation.

- Delay is the amount of time it takes for a packet to be received by the destination. We only considered the first packet arrival to its destination.

- Hop count is average number of hops a packet travels from the source to destination.

4.2 Simulation Setup

The simulations were performed over an area of 1500 m × 300 m. Each experiment was repeated 10 times and the results were aggregated over these 10 runs. Each run was performed for 1000s. In this scenario, four different flows of [5, 10, 20, 29] were run for different mobility speeds. We eliminated 30 flows, because at least one source should have been specified. The packet size was set to 64 bytes, and after each second one packet was generated from source nodes. The mobility model was set to the random waypoint with random node velocity from 0–20 m/s and pause times of 0–900 seconds. Nodes are completely mobile with the pause time of zero, to nearly stationary with the pause time of 900 seconds. The 802.11b MAC was installed in the link layer with transmission range of 250 m. Regarding the wireless link, the bandwidth was considered 11 Mb/s and delay was calculated based on this bandwidth. The only parameter adjustable in SiFT is $\alpha$. We did a very simple simulation with 10 nodes repeated for 10 times and changed the value of $\alpha$. It was observed that a smaller $\alpha$ produced better performance. Therefore $\alpha$ was set to 0.01 (as in [1]) as a trade off between simulation computation and performance. When $\alpha$ approaches to 1, degradation in results are obvious.

4.3 Simulation Analysis

In order to examine the effect of node mobility with different flows, we defined a scenario with variety of pause times. Changing pause time shows the effect of mobility and changing number of flows examines the effect of network traffic and different paths in the simulation environment. The only parameter in the SiFT algorithm is $\alpha$ which is fixed to 0.01 in all of these scenarios.

We can see that as the amount of pause time gets closer to the simulation time, PDR increases as illustrated in Figure 5. In other words, the best result is obtained when the nodes become immobile. When nodes are fixed, the PDR is nearly 100%. This is what we can expect from an algorithm that works based on control flooding. On the other hand, PDR is lower when nodes are highly mobile. We also observe that fewer number of flows produce better results in high mobility, while in low mobility, PDR of more flows are similar to the fewer flow result. This [1] is due to the high collisions in the MAC layer combined with the timer used in each node to forward a packet. The timer used in each node is in the network layer. When this timer times out, the network layer sends the packet to the MAC layer and does not have any control on it. If the MAC layer queues the packets due to a busy medium, other nodes’ timer for the same packet would timeout. Therefore, the same packet would be delivered many times and duplicate packets are observed in the destination. This effect is amplified in a busy medium, for example, when the number of flows is increased in the simulation. We also monitor the amount of traffic in the simulation. It is obvious that a higher number of flows generate more traffic, more packets are dropped in each node, and more duplicate packets reach the destination. Although the SiFT mechanism drops many packets in each node, duplicate packets are still received at the destination. A possible solution is to stop the timer when the MAC layer finds that the medium is busy [1]. However, this solution needs cross layered communication. In this version of SiFT implementation, we don’t use cross layering.

<table>
<thead>
<tr>
<th>Simulation Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation area</td>
<td>1500 m × 300 m</td>
</tr>
<tr>
<td>Number of runs</td>
<td>10</td>
</tr>
<tr>
<td>Warmup time</td>
<td>3 s</td>
</tr>
<tr>
<td>Total simulation time</td>
<td>1,000 s</td>
</tr>
<tr>
<td>Mobility model</td>
<td>Random waypoint</td>
</tr>
<tr>
<td>Node speed</td>
<td>0 – 20 m/s</td>
</tr>
<tr>
<td>Packet size</td>
<td>64 bytes</td>
</tr>
<tr>
<td>Number of packets</td>
<td>997 packets/simulation</td>
</tr>
<tr>
<td>Link layer</td>
<td>wifi-b-11Mb/s</td>
</tr>
</tbody>
</table>

Table 1: Simulation parameters

![Figure 5: Packet delivery ratio vs. pause time](image-url)

Since no path is established in advance, there aren’t any routing tables in any of the nodes. In addition, there is no delay for path establishment. Therefore, delay for each packet would be the summation of the bandwidth, propa-
gation and processing delay. Moreover, no control packet is necessary to keep the routing tables updated, which reduces the overhead of the protocol. Therefore, we were not able to compare the overhead of this protocol with other available MANET protocols in the next scenario. We conducted our simulation with OnOffApplication in ns-3, which uses UDP protocol. For any unsuccessful transmission, the transport protocol is responsible for any further retransmission.

Figure 6: Delay vs. pause time

Lack of control packets in SiFT leads to low delay in packet delivery. Figure 6 shows delay with a varying number of flows when pause time varies from zero to 900 seconds. Delay is reduced when mobility is reduced as well. We also observe that delay is less for the environment with fewer number of flows, which confirms the effect of collisions and timeouts as previously explained. Although higher delay is observed for more flows, error rates are smaller for a great number of flows. This is due to the fact that when the number of flows is small, the probability of selecting different destinations is greater than for a high number of flows. Therefore, the effect of node location in each run causes higher deviation from delay mean value. This effect is also obvious in PDR and hop count plots.

Figure 7: Number of hops vs. pause time

Figure 7 illustrates hop count versus pause time. The number hop-count number is reduced when nodes become immobile. We observe slightly shorter hop counts when the number of flows were small.

In the next scenario, we test density of nodes versus pause time in relatively low traffic. In this scenario, the number of nodes is varied from 10 to 50. For 10–30 nodes, we run each simulation 20 times, and 10 times for the rest. The reason is a reduction error rate, due to the sparse environment. Three pause times 200, 500 and 800 seconds are chosen as a representative of relatively low to high node mobility. We measure packet delivery ratio, delay, and average hop count for this scenario. As shown in Figure 8, packet delivery ratio is the largest for a pause time 800 seconds, when mobility is reduced in the environment. We observe that the packet delivery ratio is nearly 35% lower in high mobility, low node density environments. However, this value increases when the number of nodes increases. We concludes that, SiFT has good performance even in high mobility when the environment is dense. In low node density environments, the probability of being out of wireless range of other nodes is greater than in higher node density environments.

Figure 8: Packet delivery ratio vs. node density

Figure 9 shows delay for this scenario. We see higher delay with a pause time of 200 seconds. In other words, delay is relatively high in the high mobility environment. Although each node in SiFT considers distance from the last forwarding node, it doesn’t work as an opportunistic network. In other words, the farthest node from the last forwarding node is not always the one which forwards the packet. If it were, the number of hops should not be increased in Figure 10 when the environment is relatively dense and nodes cover the whole area. Yet, α in the SiFT algorithm has a great effect in this. As it was mentioned, when α is equal to 1, all performance metrics are worse than α equal to 0.01, including delay and average hop counts. However, delay decreases in all experiments, when the number of nodes increases. This is due to the fact that more paths can connect the source to destination.

As illustrated in Figure 10, the average number of hops is higher with a pause time of 200 seconds due to greater mobility. When mobility decreases, the average hop count decreases as well.
In the next scenario, we compare SiFT performance with other available MANET protocols in the current version of ns-3. The available implemented protocols are AODV, DSR [2], DSDV [9] and OLSR. Thirty nodes are used in this scenario for 1000 seconds with a variety of pause times from 0 to 900 seconds. Ten runs were performed for each pause time and the aggregate value of these runs were considered. In all of these simulations, 10 data flows with OnOffApplication were generated. Although some of these protocols have adjustable parameters, we used their default values set in ns-3.

Figure 11 shows PDR for all compared protocols. All protocols have high PDR when nodes are not mobile; however, in high mobility, the performance is lower for all protocols. This degradation is very low for SiFT. In contrast, PDR is significantly lower for DSDV and especially OLSR, since they are proactive protocols. High node mobility causes their routing information become invalid very quickly. Furthermore, this result confirms that SiFT has good performance in high mobility [1].

The average delay for the protocols is illustrated in Figure 12. As expected, the SiFT delay is much lower than other protocols. Generally delay decreases when node mobility decreases as well. Furthermore, OLSR has lower delay than other three protocols. Low delay in SiFT is the direct result of lack of control messages in this protocol, while in OLSR it is caused by the consequence of more stability in routing tables.

5. CONCLUSIONS

In this paper we presented the architecture and implementation of SiFT in ns-3. We also explained the main components of this protocol and measure its performance in different traffic and mobilities. We observed that SiFT had good PDR, low delay and hop count. This protocol does not have high overhead, since nodes don’t keep the information for the next hop. However, we observed high amount of packet dropped in heavier traffic and more duplicate packets in the destination. We also compared the performance of this protocol with other MANET protocol in ns-3.21. Low delay in SiFT is the direct result of lack of control messages in this protocol, while in OLSR it is caused by the consequence of more stability in routing tables.
6. ACKNOWLEDGMENTS

We would like to acknowledge the assistance of Mohammed Alenazi, Yufei Cheng, and the members of the ResiliNets research group for their advice and suggestions which helped us with this implementation.

7. REFERENCES


