International Journal of Research in Engineering, Science and Management
Volume-2, Issue-7, July-2019
www.ijresm.com | ISSN (Online): 2581-5792

Background Study of DSR in MANETs using NS2 Simulator

Himanshi Tyagi[1], Umer Bashir[2], Ranjan Kumar Singh[3]
[1,2]M.Tech. Student, Department of Electronics and Communication Engineering, Shri Ram College of Engineering & Management, Palwal, India
[3]Professor, Department of Electronics and Communication Engineering, Shri Ram College of Engineering & Management, Palwal, India

***Abstract*: This paper presents a study on DSR in MANETs using NS2 Simulator.**

***Keywords*: MANET**

1. Introduction of MANET

The advancement in establishing a network which is self-configuring and infrastructure less has led to development of MANET Mobile AD-Hoc Network. But limitation of range and unavailability of protocols for highly synchronized working also with limitation of power supply has always been a problem. The project aims to simulate a network for military operation in scenario where a team of members required communicating and topology changes are dynamic. The DSR Dynamic Source Routing protocol which is area of research is used, for its feature that is fast updating topologies. The MANET is one of the widely used networks for communication where there are the movement or mobile sources. The sources itself act as router independently and forms a connection to all available devices in its range. In case of natural calamity like earthquake, flood, cyclone infrastructure based communication is bound to suffer disturbance or lack of operability. Rescue or relief teams generally have to be well equipped with expensive equipments to enable communication between individuals. Introduction of ad-hoc network based voice communication can serve well in this purpose. Also people suffering such disaster can have a way to communicate with each other and the rescue team. Since this network doesn't require any predefined setup or any non-regular device, connectivity is instant and useful. Like any other infrastructure less network, ad-hoc network suffers from lot of complexities due to its dynamic nature and it fails to provide the reliability of a structured network.

There are lots of things to be done in this field. The primary goal of this research work is to go in deep of this field and to make ad-hoc voice communication as close as possible to voice communication systems available to general users in terms of both performance and usability. A wireless ad-hoc network is a collection of mobile/semi-mobile nodes with no pre-established infrastructure, forming a temporary network. Each of the nodes has a wireless interface and communicates with each other over either radio or infrared. Laptop computers and personal digital assistants that communicate directly with each other are some examples of nodes in an ad-hoc network. Nodes in the ad-hoc network are often mobile, but can also consist of stationary nodes, such as access points to the Internet. Semi mobile nodes can be used to deploy relay points in areas where relay points might be needed temporarily. The outermost nodes are not within transmitter range of each other. However, the middle node can be used to forward packets between the outermost nodes. The middle node is acting as a router and the three nodes have formed an ad-hoc network. An ad-hoc network uses no centralized administration. This is to be sure that the network won't collapse just because one of the mobile nodes moves out of transmitter range of the others. Nodes should be able to enter/leave the network as they wish. Because of the limited transmitter range of the nodes, multiple hops may be needed to reach other nodes. Every node wishing to participate in an adhoc network must be willing to forward packets for other nodes. Thus every node acts both as a host and as a router. A node can be viewed as an abstract entity consisting of a router and a set of affiliated mobile hosts.

A router is an entity, which, among other things runs a routing protocol. A mobile host is simply an IP-addressable host/entity in the traditional sense. Ad-hoc networks are also capable of handling topology changes and malfunctions in nodes. It is fixed through network reconfiguration. For instance, if a node leaves the network and causes link breakages, affected nodes can easily request new routes and the problem will be solved. This will slightly increase the delay, but the network will still be operational. Wireless ad-hoc networks take advantage of the nature of the wireless communication medium. In other words, in a wired network the physical cabling is done a priori restricting the connection topology of the nodes. This restriction is not present in the wireless domain and, provided that two nodes are within transmitter range of each other, an instantaneous link between them.

There is no clear picture of what these kinds of networks will be used for. The suggestions vary from document sharing at conferences to infrastructure enhancements and military applications. In areas where no infrastructure such as the

Internet is available an ad-hoc network could be used by a group of wireless mobile hosts. This can be the case in areas where a network infrastructure may be undesirable due to reasons such as cost or convenience. Examples of such situations include disaster recovery personnel or military troops in cases where the normal infrastructure is either unavailable or destroyed. Other examples include business associates wishing to share files in an airport terminal, or a class of students needing to interact during a lecture. If each mobile host wishing to communicate is equipped with a wireless local area network interface, the group of mobile hosts may form an ad-hoc network. Access to the 12 Internet and access to resources in networks such as printers are features that probably also will be supported. With more and more portable devices with Wi-Fi radio pre-built coming into market soon it might be get some real acceptance in the field of insecure but free voice communication over short rage in case of campus talk or disaster scenarios where no infrastructure might be up and running.

2. Characteristics

Ad-hoc networks are often characterized by a dynamic topology due to the fact that nodes change their physical location by moving around. This favours routing protocols that dynamically discover routes over conventional routing algorithms like distant vector and link state. Another characteristic is that a host/node has very limited CPU capacity, storage capacity, battery power and bandwidth, also referred to as a “thin client”. This means that the power usage must be limited thus leading to a limited transmitter range. The access media, the radio environment, also has special characteristics that must be considered when designing protocols for ad-hoc networks. One example of this may be unidirectional links. These links arise when for example two nodes have different strength on their transmitters, allowing only one of the hosts to hear the other, but can also arise from disturbances from the surroundings. Multi hop in a radio environment may result in an overall transmit capacity gain and power gain, due to the squared relation between coverage and required output power. By using multi hop, nodes can transmit the packets with a much lower output power.

3. Routing

Because of the fact that it may be necessary to hop several hops multi-hop before a packet reaches the destination, a routing protocol is needed. The routing protocol has two main functions, selection of routes for various source destination pairs and the delivery of messages to their correct destination. The second function is conceptually straightforward using a variety of protocols and data structures routing tables. Routing protocols use metrics to evaluate what path will be the best for a packet to travel. A metric is a standard of measurement; such as path bandwidth, reliability, delay, current load on that path etc; that is used by routing algorithms to determine the optimal path to a destination. To aid the process of path determination, routing algorithms initialize and maintain routing tables, which contain route information.

4. Conventional protocols

The main problem with link-state and distance vector is that they are designed for a static topology, which means that they would have problems to converge to a steady state in an ad-hoc network with a very frequently changing topology. Link state and distance vector would probably work very well in an ad-hoc network with low mobility, i.e. a network where the topology is not changing very often. The problem that still remains is that link-state and distance-vector are highly dependent on periodic control messages. As the number of network nodes can be large, the potential number of destinations is also large. This requires large and frequent exchange of data among the network nodes. This is in contradiction with the fact that all updates in a wireless interconnected ad hoc network are transmitted over the air and thus are costly in resources such as bandwidth, battery power and CPU. Because both link-state and distance vector tries to maintain routes to all reachable destinations, it is necessary to maintain these routes and this also wastes resources for the same reason as above.

Another characteristic for conventional protocols are that they assume bidirectional links, e.g. that the transmission between two hosts works equally well in both directions. In the wireless radio environment this is not always the case. Because many of the proposed ad-hoc routing protocols have a traditional routing protocol as underlying algorithm, it is necessary to understand the basic operation for conventional protocols like distance vector, link state and source routing.

A. *Link state*

In link-state routing, each node maintains a view of the complete topology with a cost for each link. To keep these costs consistent; each node periodically broadcasts the link costs of its outgoing links to all other nodes using flooding. As each node receives this information, it updates its view of the network and applies a shortest path algorithm to choose the next-hop for each destination. Some link costs in a node view can be incorrect because of long propagation delays, partitioned networks, etc. Such inconsistent network topology views can lead to formation of routing-loops. These loops are however short-lived, because they disappear in the time it takes a message to traverse the diameter of the network.

Link state routing protocols maintain complete road map of the network in each router running a link state routing protocol. Each router running a link state routing protocol originates information about the router, its directly connected links, and the state of those links. This information is sent to all the routers in the network as multicast messages. Link-state routing always tries to maintain full networks topology by updating itself incrementally whenever a change happen in network.

B. *Distance vector*

In distance vector each node only monitors the cost of its outgoing links, but instead of broadcasting this information to all nodes; it periodically broadcasts to each of its neighbours an estimate of the shortest distance to every other node in the network. The receiving nodes then use this information to recalculate the routing tables, by using a shortest path algorithm. Compared to link-state, distance vector is more computation efficient, easier to implement and requires much less storage space. However, it is well known that distance vector can cause the formation of both short-lived and long-lived routing loops. The primary cause for this is that the nodes choose their next-hops in a completely distributed manner based on information that can be stale.

C. *Source routing*

Source routing means that each packet must carry the complete path that the packet should take through the network. The routing decision is therefore made at the source. The advantage with this approach is that it is very easy to avoid routing loops. The disadvantage is that each packet requires a slight overhead.

D. *Flooding*

Many routing protocols uses broadcast to distribute control information, that is, send the control information from an origin node to all other nodes. A widely used form of broadcasting is flooding and operates as follows. The origin node sends its information to its neighbours in the wireless case; this means all nodes that are within transmitter range. The neighbours relay it to their neighbours and so on, until the packet has reached all nodes in the network. A node will only relay a packet once and to ensure this some sort of sequence number can be used. This sequence number is increased for each new packet a node sends.

5. Classification

Routing protocols can be classified into different categories depending on their properties.

- Centralized vs. Distributed
- Static vs. Adaptive
- Reactive vs. Proactive

One way to categorize the routing protocols is to divide them into centralized and distributed algorithms. In centralized algorithms, all route choices are made at a central node, while in distributed algorithms, the computation of routes is shared among the network nodes. Another classification of routing protocols relates to whether they change routes in response to the traffic input patterns. In static algorithms, the route used by source-destination pairs is fixed regardless of traffic conditions. It can only change in response to a node or link failure. This type of algorithm cannot achieve high throughput under a broad variety of traffic input patterns. Most major packet networks use some form of adaptive routing where the routes used to route between source-destination pairs may change in response to congestion A third classification that is more related to ad-hoc networks is to classify the routing algorithms as either proactive or reactive. Proactive protocols attempt to continuously evaluate the routes within the network, so that when a packet needs to be forwarded, the route is already known and can be immediately used. The family of Distance-Vector protocols is an example of a proactive scheme. Reactive protocols, on the other hand, invoke a route determination procedure on demand only. Thus, when a route is needed, some sort of global search procedure is employed. The family of classical flooding algorithms belongs to the reactive group. Proactive schemes have the advantage that when a route is needed, the delay before actual packets can be sent is very small. On the other side proactive schemes needs time to converge to a steady state. This can cause problems if the topology is changing frequently. The tools used for simulation provides all facilities for standard comparisons by visual as well as graph generation for simulation with event driven programming.

6. DSR protocol description

A. *Overview and important properties of the protocol*

The DSR protocol is composed of two mechanisms that work together to allow the discovery and maintenance of source routes in the ad hoc network: Route Discovery is the mechanism by which a node S wishing to send a packet to a destination node D obtains a source route to D. Route Discovery is used only when S attempts to send a packet to D and does not already know a route to D.

Route Maintenance is the mechanism by which node S is able to detect, while using a source route to D, if the network topology has changed such that it can no longer use its route to D because a link along the route no longer works. When Route Maintenance indicates a source route is broken, S can attempt to use any other route it happens to know to D, or can invoke Route Discovery again to find a new route. Route Maintenance is used only when S is actually sending packets to D. Route Discovery and Route Maintenance each operate entirely on demand. In particular, unlike other protocols, DSR requires no periodic packets of any kind at any level within the network. For example, DSR does not use any periodic routing advertisement, link status sensing, or neighbour detection packets, and does not rely on these functions from any underlying protocols in the network. This entirely on-demand behaviour and lack of periodic activity allows the number of overhead packets caused by DSR to scale all the way down to zero, when all nodes are approximately stationary with respect to each other and all routes needed for current communication have already been discovered. As nodes begin to move more or as communication patterns change, the routing packet overhead of DSR automatically scales to only that needed to track the routes currently in use. In response to a single Route Discovery (as well as through routing information from other packets overheard), a node may learn and cache multiple routes to any destination. This allows the reaction to routing changes to be much more rapid, since a node with multiple routes to a destination can try another cached route if the one it has been using should fail. This caching of multiple routes also avoids the overhead of needing to perform a new Route Discovery each time a route in use breaks. The operation of Route Discovery and Route Maintenance in DSR are designed to allow uni-directional links and asymmetric routes to be easily supported. In particular, as noted, it is possible that a link between two nodes may not work equally well in both directions, due to differing antenna or propagation patterns or sources of interference. DSR allows such uni-directional links to be used when necessary, improving overall performance and network connectivity in the system. DSR also supports internetworking between different types of wireless networks, allowing a source route to be composed of hops over a combination of any types of networks available. For example, some nodes in the ad hoc network may have only short-range radios, while other nodes have both short-range and long-range radios; the combination of these nodes together can be considered by DSR as a single ad hoc network. In addition, the routing of DSR has been integrated into standard Internet routing, where a “gateway” node connected to the Internet also participates in the ad hoc network routing protocols; and has been integrated into Mobile IP routing, where such a gateway node also serves the role of a Mobile IP foreign agent.

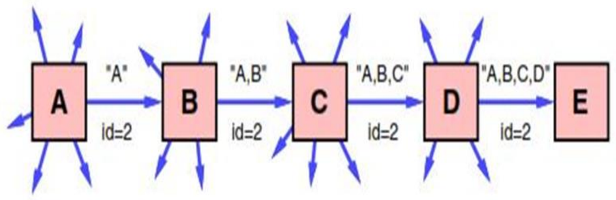

Fig. 1. Route discovery example: node A is the initiator, and node E is the target

B. *Basic DSR route discovery*

When some node S originates a new packet destined to some other node D, it places in the header of the packet a source route giving the sequence of hops that the packet should follow on its way to D. Normally, S will obtain a suitable source route by searching its Route Cache of routes previously learned, but if no route is found in its cache, it will initiate the Route Discovery protocol to dynamically find a new route to D. In this case, we call S the initiator and D the target of the Route Discovery. For example, Figure 1, illustrates an example Route Discovery, in which a node A is attempting to discover a route to node E. To initiate the Route Discovery, A transmits a ROUTE REQUEST message as a single local broadcast packet, which is received by (approximately) all nodes currently within wireless transmission range of A. Each ROUTE REQUEST message identifies the initiator and target of the Route Discovery, and also contains a unique request id, determined by the initiator of the REQUEST. Each ROUTE REQUEST also contains a record listing the address of each intermediate node through which this particular copy of the ROUTE REQUEST message has been forwarded. This route record is initialized to an empty list by the initiator of the Route Discovery.

When another node receives a ROUTE REQUEST, if it is the target of the Route Discovery, it returns a ROUTE REPLY message to the initiator of the Route Discovery, giving a copy of the accumulated route record from the ROUTE REQUEST; when the initiator receives this ROUTE REPLY, it caches this route in its Route Cache for use in sending subsequent packets to this destination. Otherwise, if this node receiving the ROUTE REQUEST has recently seen another ROUTE REQUEST message from this initiator bearing this same request id, or if it finds that its own address is already listed in the route record in the ROUTE REQUEST message, it discards the REQUEST. Otherwise, this node appends its own address to the route record in the

ROUTE REQUEST message and propagates it by transmitting it as a local broadcast packet (with the same request id). In returning the ROUTE REPLY to the initiator of the Route Discovery, such as node E replying back to A in Figure 1.1, node E will typically examine its own Route Cache for a route back to A, and if found, will use it for the source route for delivery of the packet containing the ROUTE REPLY. Otherwise, E may perform its own Route Discovery for target node A, but to avoid possible infinite recursion of Route Discoveries, it must piggyback this ROUTE REPLY on its own ROUTE REQUEST message for A. It is also possible to piggyback other small data packets, such as a TCP SYN packet, on a ROUTE REQUEST using this same mechanism. Node E could also simply reverse the sequence of hops in the route record that it trying to send in the ROUTE REPLY, and use this as the source route on the packet carrying the ROUTE REPLY itself. For MAC protocols such as IEEE 802.11 that require a bi-directional frame exchange as part of the MAC protocol, this route reversal is preferred as it avoids the overhead of a possible second Route Discovery, and it tests the discovered route to ensure it is bi-directional before the Route Discovery initiator begins using the route. However, this technique will prevent the discovery of routes using uni-directional links. In wireless environments where the use of uni-directional links is permitted, such routes may in some cases be more efficient than those with only bi-directional links, or they may be the only way to achieve connectivity to the target node. When initiating a Route Discovery, the sending node saves a copy of the original packet in a local buffer called the Send Buffer. The Send Buffer contains a copy of each packet that cannot be transmitted by this node because it does not yet have a source route to the packet’s destination. Each packet in the Send Buffer is stamped with the time that it was placed into the Buffer and is discarded after residing in the Send Buffer for some timeout period; if necessary for preventing the Send Buffer from overflowing, a FIFO or other replacement strategy can also be used to evict packets before they expire. While a packet remains in the Send Buffer, the node should occasionally initiate a new

Route Discovery for the packet's destination address. However, the node must limit the rate at which such new Route Discoveries for the same address are initiated, since it is possible that the destination node is not currently reachable. In particular, due to the limited wireless transmission range and the movement of the nodes in the network, the network may at times become partitioned, meaning that there is currently no sequence of nodes through which a packet could be forwarded to reach the destination. Depending on the movement pattern and the density of nodes in the network, such network partitions may be rare or may be common.

If a new Route Discovery was initiated for each packet sent by a node in such a situation, a large number of unproductive ROUTE REQUEST packets would be propagated throughout the subset of the ad hoc network reachable from this node. In order to reduce the overhead from such Route Discoveries, we use exponential back-off to limit the rate at which new Route Discoveries may be initiated by any node for the same target. If the node attempts to send additional data packets to this same node more frequently than this limit, the subsequent packets should be buffered in the Send Buffer until a ROUTE REPLY is received, but the node must not initiate a new Route Discovery until the minimum allowable interval between new Route Discoveries for this target has been reached. This limitation on the maximum rate of Route Discoveries for the same target is similar to the mechanism required by Internet nodes to limit the rate at which ARP REQUESTs are sent for any single target IP address.

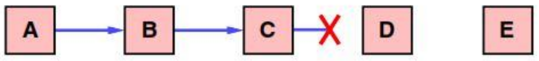

Fig. 2. Route Maintenance example: Node C is unable to forward a packet from A to E over its link to next hop D.

C. Basic DSR route maintenance

When originating or forwarding a packet using a source route, each node transmitting the packet is responsible for confirming that the packet has been received by the next hop along the source route; the packet is retransmitted (up to a maximum number of attempts) until this confirmation of receipt is received. For example, in the situation illustrated in Figure 2, node A has originated a packet for E using a source route through intermediate nodes B, C, and D. In this case, node A is responsible for receipt of the packet at B, node B is responsible for receipt at C, node C is responsible for receipt at D, and node D is responsible for receipt finally at the destination E. This confirmation of receipt in many cases may be provided at no cost to DSR, either as an existing standard part of the MAC protocol in use (such as the link-level acknowledgement frame defined by IEEE 802.11, or by a passive acknowledgement (in which, for example, B confirms receipt at C by overhearing C transmit the packet to forward it on to D). If neither of these confirmation mechanisms are available, the node transmitting the packet may set a bit in the packet's header to request a DSR-specific software acknowledgement be returned by the next hop; this software acknowledgement will normally be transmitted directly to the sending node, but if the link between these two nodes is uni-directional, this software acknowledgement may travel over a different, multi-hop path.

If the packet is retransmitted by some hop the maximum number of times and no receipt confirmation is received, this node returns a ROUTE ERROR message to the original sender of the packet, identifying the link over which the packet could not be forwarded. For example, in Figure 2, if C is unable to deliver the packet to the next hop D, then C returns a ROUTE ERROR to A, stating that the link from C to D is currently "broken." Node A then removes this broken link from its cache; any retransmission of the original packet is a function for upper layer protocols such as TCP. For sending such a retransmission or other packets to this same destination E, if A has in its Route Cache another route to E (for example, from additional ROUTE REPLYs from its earlier Route Discovery, or from having overheard sufficient routing information from other packets), it can send the packet using the new route immediately. Otherwise, it may perform a new Route Discovery for this target.

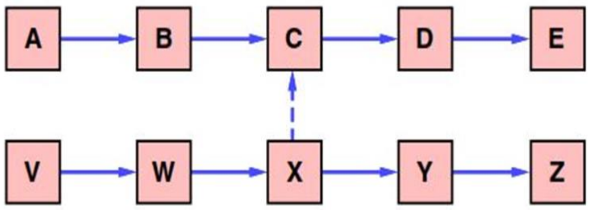

Fig. 3. Limitations on caching overheard routing information: Node C is forwarding packets to E and overhears packets from X

7. Additional route discovery features

A. Caching overheard routing information

A node forwarding or otherwise overhearing any packet may add the routing information from that packet to its own Route Cache. In particular, the source route used in a data packet, the accumulated route record in a ROUTE REQUEST, or the route being returned in a ROUTE REPLY may all be cache by any node. Routing information from any of these packets received may be cached, whether the packet was addressed to this node, sent to a broadcast (or multicast) MAC address, or received while the node's network interface is in promiscuous mode. One limitation, however, on caching of such overheard routing information is the possible presence of uni-directional links in the ad hoc network. For example, Figure 4 illustrates a situation in which node A is using a source route to communicate with node E. As node C forwards a data packet along the route from A to E, it can always add to its cache the presence of the "forward" direction links that it learns from the headers of these packets, from itself to D and from D to E. However, the "reverse" direction of the links identified in the packet headers, from itself back to B and from B to A, may not work for it since these links might be uni-directional. If C knows that the links are in fact bi-directional, for example due to the MAC protocol in use, it could cache them but otherwise should not. Likewise, node V in Figure 1.3 is using a different source route to communicate with node Z. If node C overhears node X transmitting a data packet to forward it to Y (from V), node C should consider whether the links involved can be known to be bi-directional or not before caching them. If the link from X to C (over which this data packet was received) can be known to be bi-directional, then C could cache the link from itself to X, the link from X to Y, and the link from Y to Z. If all links can be assumed to be bi-directional, C could also cache the links from X to W and from W to V. Similar considerations apply to the routing information that might be learned from forwarded or otherwise overheard ROUTE REQUEST or ROUTE REPLY packets.

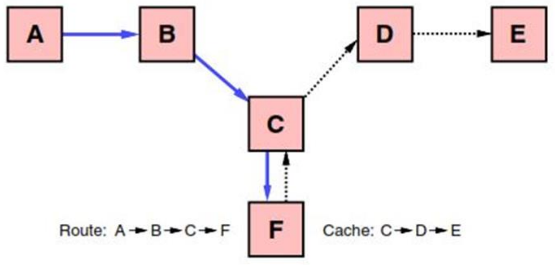

Fig. 4. A possible duplication of route hops avoided by the Route Discovery limitation on replying to ROUTE REQUESTs from the Route Cache

B. *Reply to ROUTE REQUESTs using Cached Routes*

A node receiving a ROUTE REQUEST for which it is not the target, searches its own Route Cache for a route to the target of the REQUEST. If found, the node generally returns a ROUTE REPLY to the initiator itself rather than forwarding the ROUTE REQUEST. In the ROUTE REPLY, it sets the route record to list the sequence of hops over which this copy of the ROUTE REQUEST was forwarded to it, concatenated with its own idea of the route from itself to the target from its Route Cache. However, before transmitting a ROUTE REPLY packet that was generated using information from its Route Cache in this way, a node must verify that the resulting route being returned in the ROUTE REPLY, after this concatenation, contains no duplicate nodes listed in the route record. For example, Figure 4 illustrates a case in which a ROUTE REQUEST for target E has been received by node F, and node F already has in its Route Cache a route from itself to E. The concatenation of the accumulated route from the ROUTE REQUEST and the cached route from F's Route Cache would include a duplicate node in passing from C to F and back to C. Node F in this case could attempt to edit the route to eliminate the duplication, resulting in a route from A to B to C to D and on to E, but in this case, node F would not be on the route that it returned in its own ROUTE REPLY. DSR Route Discovery prohibits node F from returning such a ROUTE REPLY from its cache for two reasons. First, this limitation increases the probability that the resulting route is valid, since F in this case should have received a ROUTE ERROR if the route had previously stopped working. Second, this limitation means that a ROUTE ERROR traversing the route is very likely to pass through any node that sent the ROUTE REPLY for the route (including F), which helps to ensure that stale data is removed from caches (such as at F) in a timely manner. Otherwise, the next Route Discovery initiated by A might also be contaminated by a ROUTE REPLY from F containing the same stale route. If the ROUTE REQUEST does not meet these restrictions, the node (node F in this example) discards the ROUTE REQUEST rather than replying to it or propagating it.

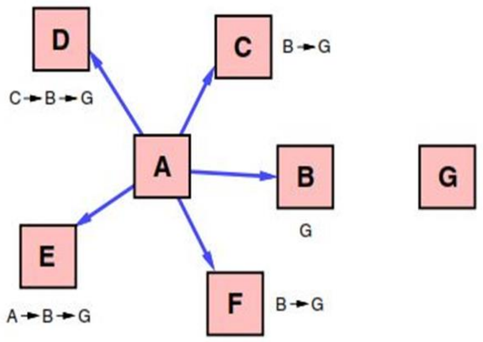

Fig. 5. A route reply storm could result if many nodes all reply to the same ROUTE REQUEST from their own Route Caches. The route listed next to each node shows the route to destination G currently listed in that node's Route Cache

C. *Preventing route reply storms*

The ability for nodes to reply to a ROUTE REQUEST based on information in their Route Caches, could result in a possible ROUTE REPLY "storm" in some cases. In particular, if a node broadcasts a ROUTE REQUEST for a target node for which the node's neighbours have a route in their Route Caches, each neighbour may attempt to send a ROUTE REPLY, thereby wasting bandwidth and possibly increasing the number of network collisions in the area. For example, in the situation shown in Figure 5, nodes B, C, D, E, and F all receive A's ROUTE REQUEST for target G, and each have the indicated route cached for this target. Normally, they would all attempt to reply from their own Route Caches, and would all send their REPLYs at about the same time since they all received the broadcast ROUTE REQUEST at about the same time. Such simultaneous replies from different nodes all receiving the ROUTE REQUEST may create packet collisions among some or all of these REPLIES and may cause local congestion in the wireless network. In addition, it will often be the case that the different replies will indicate routes of different lengths, as shown in this example. If a node can put its network interface into promiscuous receive mode, it should delay sending its own ROUTE REPLY for a short period, while listening to see if the initiating node begins using a shorter route first. That is, this node should delay sending its own ROUTE REPLY for a random period where h is the length in number of network hops for the route to be returned in this node's ROUTE REPLY, r is a random number between 0 and 1, and H is a small constant delay (at least twice the maximum wireless link propagation delay) to be introduced per hop. This delay effectively randomizes the time at which each. The route listed next to each node shows the route to destination G currently listed in that node's Route Cache node sends its ROUTE REPLY, with all nodes sending ROUTE REPLYs giving routes of length less than H sending their REPLYs before this node, and all nodes sending ROUTE REPLYs giving routes of length greater than h sending their REPLYs after this node. Within the delay period, this node promiscuously receives all packets, looking for data packets from the initiator of this Route Discovery destined for the target of the Discovery. If such a data packet received by this node during the delay period uses a source route of length less than or equal to h, this node may infer that the initiator of the Route Discovery has already received a ROUTE REPLY giving an equally good or better route. In this case, this node cancels its delay timer and does not send its ROUTE REPLY for this Route Discovery.

D. Route request hop limits

Each ROUTE REQUEST message contains a "hop limit" that may be used to limit the number of intermediate nodes allowed to forward that copy of the ROUTE REQUEST. As the REQUEST is forwarded, this limit is decremented, and the REQUEST packet is discarded if the limit reaches zero before finding the target. We currently use this mechanism to send a non-propagating ROUTE REQUEST (i.e., with hop limit 0) as an inexpensive method of determining if the target is currently a neighbour of the initiator or if a neighbour node has a route to the target cached (effectively using the neighbours' caches as an extension of the initiator's own cache). If no ROUTE REPLY is received after a short timeout, then a propagating ROUTE REQUEST (i.e., with no hop limit) is sent. We have also considered using this mechanism to implement an expanding ring search for the target [Johnson 1996a]. For example, a node could send an initial non-propagating ROUTE REQUEST as described above; if no ROUTE REPLY is received for it, the node could initiate another ROUTE REQUEST with a hop limit of 1. For each ROUTE REQUEST initiated, if no ROUTE REPLY is received for it, the node could double the hop limit used on the previous attempt, to progressively explore for the target node without allowing the ROUTE REQUEST to propagate over the entire network. However, this expanding ring search approach could have the effect of increasing the average latency of Route Discovery, since multiple Discovery attempts and timeouts may be needed before discovering a route to the target node.

8. Additional route maintenance features

1) Packet salvaging

After sending a ROUTE ERROR message as part of Route Maintenance as described, a node may attempt to salvage the data packet that caused the ROUTE ERROR rather than discarding it. To attempt to salvage a packet, the node sending a ROUTE ERROR searches its own Route Cache for a route from itself to the destination of the packet causing the ERROR. If such a route is found, the node may salvage the packet after returning the ROUTE ERROR by replacing the original source route on the packet with the route from its Route Cache. The node then forwards the packet to the next node indicated along this source route. For example, in Figure 6, if node C has another route cached to node E, it can salvage the packet by applying this route to the packet rather than discarding the packet. When salvaging a packet in this way, the packet is also marked as having been salvaged, to prevent a single packet being salvaged multiple times. Otherwise, it could be possible for the packet to enter a routing loop, as different nodes repeatedly salvage the packet and replace the source route on the packet with routes to each other. An alternative mechanism of salvaging that we have considered would be to replace only the unused suffix of the original route with the new route from this node's Route Cache, forming a new route whose prefix is the original route and whose suffix is the route from the Cache. In this case, the normal rules for avoiding duplicated nodes being listed in a source route are sufficient to avoid routing loops. However, this mechanism of salvaging would prevent the new route from "backtracking" from this node to an earlier node already traversed by this packet, to then be forwarded along a different remaining sequence of hops to the destination. Our current salvaging mechanism allows backtracking but prevents a packet from being salvaged more than once.

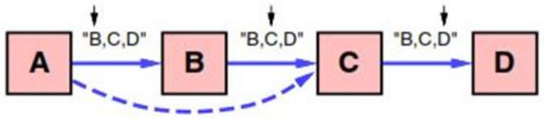

Fig. 6. Node C notices that the source route to D can be shortened, since it overheard a packet from A intended first for B

2) Automatic route shortening

Source routes in use may be automatically shortened if one or more intermediate hops in the route become no longer necessary. This mechanism of automatically shortening routes in use is somewhat similar to the use of passive acknowledgements. In particular, if a node is able to overhear a packet carrying a source route (e.g., by operating its network interface in promiscuous receive mode), then this node examines the unused portion of that source route. If this node is not the intended next hop for the packet 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 needed in the route. For example, Figure 1.6 illustrates an example in which node C has overheard a data packet being transmitted from A to B, for later forwarding to C; the arrow pointing to one node in the source route in each packet indicates the intended next receiver of the packet along the route.

In this case, this node (node C) returns a gratuitous ROUTE REPLY message to the original sender of the packet (node A). The ROUTE REPLY gives the shorter route as the concatenation of the portion of the original source route up through the node that transmitted the overheard packet, plus the suffix of the original source route beginning with the node returning the gratuitous ROUTE REPLY. In this example, the Route returned in10 the gratuitous ROUTE REPLY message sent from C to A.

3) Increased spreading of route error messages

When a source node receives a ROUTE ERROR for a data packet that it originated, this source node propagates this ROUTE ERROR to its neighbours by piggybacking it on its next ROUTE REQUEST. In this way, stale information in the caches of nodes around this source node will not generate ROUTE REPLYs that contain the same invalid link for which this source node received the ROUTE ERROR. For example, in the situation shown in Figure 1.5.b, node A learns from the ROUTE ERROR message from C, that the link from C to D is currently broken. It thus removes this link from its own Route Cache and initiates a new Route Discovery (if it doesn't have another route to E in its Route Cache). On the ROUTE REQUEST packet initiating this Route Discovery, node A piggybacks a copy of this ROUTE ERROR message, ensuring that the ROUTE ERROR message spreads well to other nodes, and guaranteeing that any ROUTE REPLY that it receives (including those from other node's Route Caches) in response to this ROUTE REQUEST does not contain a route that assumes the existence of this broken link. We have also considered, but not simulated, a further improvement to Route Maintenance in which a node, such as A in Figure 1.4, that receives a ROUTE ERROR will forward the ERROR along the same source route that resulted in the ERROR. This will almost guarantee that the ROUTE ERROR reaches the node that generated the ROUTE REPLY containing the broken link, which will prevent that node from contaminating a future Route Discovery with the same broken link.

In some cases, DSR could potentially benefit from nodes caching "negative" information in their Route Caches. For example, in Figure 7, if node A caches the fact that the link from C to D is currently broken (rather than simply removing this hop from its Route Cache), it can guarantee that no ROUTE REPLY that it receives in response to its new Route Discovery will be accepted that utilizes this broken link. A short expiration period must be placed on this negative cached information, since while this entry is in its Route Cache, A will otherwise refuse to allow this link in its cache, even if this link begins working again. Another case in which caching negative information in a node's Route Cache might be useful is the case in which a link is providing highly variable service, sometimes working correctly but often not working. This situation could occur, for example, in the case in which the link is near the limit of the sending node's wireless transmission range and there are significant sources of interference (e.g., multipath) near the receiving node on this link. In this case, by caching the negative information that this link is broken, a node could avoid adding this problematic link back to its Route Cache during the brief periods in which it is working correctly.

We have not currently included this caching of negative information in our simulations or implementation of DSR, although we have found situations in our DSR test bed implementation where it could potentially improve the performance of Route Discovery. A challenge in implementing the caching of negative information that we are currently researching is the difficulty of picking a suitable expiration period for such cache entries.

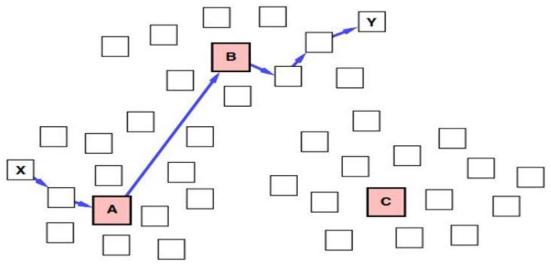

Fig. 7. An ad hoc network consisting of nodes communicating via short range radios, with nodes A, B, and C also having long-range radios. Communication between other nodes such as X and Y may involve multiple short-range hops, followed by a long-range hop, followed by additional short-range hops

4) Support for heterogeneous networks and mobile IP

In configuring and deploying an ad hoc network, in many cases, all nodes will be equipped with the same type of wireless network interfaces, allowing simple routing between nodes over arbitrary sequences of network hops. However, a more flexible configuration might be to also equip a subset of the nodes with a second and C also having long-range radios. Communication between other nodes such as X and Y may involve multiple short-range hops, followed by a long-range hop, followed by additional short-range hops network interface consisting of a longer-range (and thus generally lower speed) wireless network interface. For example, in a military setting, a group of soldiers might all use short-range radios to communicate among themselves, while relaying through truck-mounted higher power radios to communicate with other groups. This general type of network configuration is the ad hoc networking equivalent of wireless overlay networks. Due to the high degree of locality likely to be present among directly cooperating nodes communicating with each other, such a network configuration would allow high speed communication among such cooperating nodes, while at the same time allowing communication with other nodes further away without requiring very large numbers of network hops. The longer-range radios might also allow gaps between different groups of nodes to be spanned, reducing the probability of network partition. A simple example of such an ad hoc network configuration is shown in Figure 8. Nodes A, B, and C, here, each have both short-range and long-range radio interfaces, all other nodes in the ad hoc network have only short-range radio network interfaces. Node X is using a source route to node Y that uses a sequence of both short-range and long-range hops.

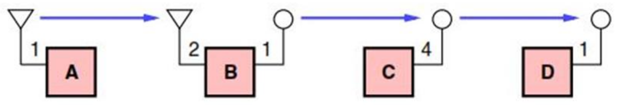

Fig. 8. An ad hoc network consisting of nodes with heterogeneous network interfaces

9. Use of interface indices in DSR

DSR supports automatic, seamless routing in these and other heterogeneous configurations, through its logical addressing model. Using conventional IP addressing, each ad hoc network node would configure a different IP address for each of its possibly many network interfaces, but as noted, each node using DSR chooses one of these as its home address to use for all communication while in the ad hoc network. This use of a single IP address per node gives DSR the ability to treat the overall network as single routing domain. To then distinguish between the different network interfaces on a node, each node independently assigns a locally unique interface index to each of its own network interfaces. The interface index for any network interface on a node is an opaque value assigned by the node itself. The particular value chosen must be unique among the network interfaces on that individual node but need have no other significance and need not be coordinated with any other nodes in choosing their own interface indices. On many operating systems, a unique value to identify each network interface is already available and can be used for this purpose; for example, the if index field in the if net structure for a network interface in BSD Unix-based networking stacks can be used directly by a node for the interface index for that network interface. For example, Figure 8 illustrates a simple ad hoc network of four nodes, in which node A is using one type of network interface (represented by the triangles), node C and node D are using an different type of physical network interface (represented by the circles), and node B is configured with both types of network interfaces and can forward packets between the two different types of radio technologies. The number labeling each network interface indicates the interface index chosen by the corresponding node for that interface. Since the interface indices are chosen independently by each node, it is possible, for example, that nodes B and D each chose index 1 for their circle network interfaces, but node C chooses index 4. The interface index is used as part of each hop in each source route discovered and used by DSR. Specifically, a path through the ad hoc network from a source node N0 to a destination node Nm is fully represented as a series of hops, where the notation Nk/ik is used to indicate that node Nk must transmit the packet using its network interface ik in order to deliver the packet over the next hop to node. In forwarding a ROUTE REQUEST, a node adds to the route record in the REQUEST, not only its own address (Section 3.2), but also the interface index of its own network interface on which it forwards the packet. To allow the reversing of a sequence of hops for a reverse route back to the originating node (when, for example, the existence of bi-directional links can by assumed based on the underlying MAC protocol), the node forwarding the ROUTE REQUEST may also add to the route record in the REQUEST, the interface index of its own network interface on which it received the ROUTE REQUEST packet. The interface indices to represent a route are carried in the ROUTE REQUEST, the ROUTE REPLY, and the source route in the header of data packets.

1) Internet interconnection and mobile IP

DSR supports the seamless interoperation between an ad hoc network and the Internet, allowing packets to transparently be routed from the ad hoc network to nodes in the Internet and from the Internet to nodes in the ad hoc network. To enable this interoperation, one (or more) nodes in the ad hoc network must be connected to the Internet, such that it participates in the ad hoc network through DSR, and also participates in the Internet through standard IP routing. We call such a node a gateway between the ad hoc network and the Internet. In this way, DSR allows the coverage range around a wireless Internet base station, for example, to be dynamically enlarged through multiple “hops” between nodes through the ad hoc network. It is also possible for such a gateway node to operate as a Mobile IP home agent or foreign agent, allowing nodes to visit the ad hoc network as a Mobile IP foreign network, and allowing nodes whose home network is the ad hoc network to visit other networks using Mobile IP.

This functionality of interconnection with the Internet is implemented through two special reserved interface index values, used by gateway nodes to identify their interconnection to the Internet. If the node has a separate physical network interface by which it connects to the Internet, other than the network interface(s) that it uses for participation in the ad hoc network, the reserved interface index is used to identify that interface. However, it is also possible for a node to use a single network interface both for participation in the ad hoc network and also for connection to the Internet through standard IP routing; in this case, the reserved interface index identifies the logically separate functionality of this interface for its Internet connection, and the node uses another (locally assigned) interface index value to identify this interface in its separate logical function of participation in the ad hoc network. If the gateway node is acting as a Mobile IP home agent or foreign agent (termed a mobility agent) on this network interface, it uses the reserved interface index value IF_INDEX_MA. Otherwise, the gateway node uses the reserved value IF_INDEX_ROUTER. The distinction between the reserved index values for mobility agents and for routers allows mobility agents to advertise their existence (as needed for Mobile IP) at no cost. A node in the ad hoc network that processes a routing header listing the interface index IF_INDEX_MA can then send a unicast Mobile IPAGENT to the corresponding address in the routing header to obtain complete information about the Mobile IP services being provided. In processing a received ROUTE REQUEST, a gateway node generates a ROUTE REPLY, giving its reserved interface index value, if it believes it may be able to reach the target node through its Internet connection. Thus, the originator of the Route Discovery may receive REPLYs both from the gateway and from the node itself, if the node is really present in the ad hoc network. When later sending packets to this destination, the sender should prefer cached routes that do not traverse a hop with an interface index of IF_INDEX_MA or IF_INDEX_ROUTER, since this will prefer Routes that lead directly to the destination node within the ad hoc network.

2) Multicast routing with DSR

DSR does not currently support true multicast routing, but does support an approximation of this that is sufficient in many network contexts. Through an extension of the Route Discovery mechanism, DSR supports the controlled flooding of a data packet to all nodes in the ad hoc network that are within some specified number of hops of the originator; these nodes may then apply destination address filtering (e.g., in software) to limit the packet to those nodes subscribed to the packet's indicated multicast destination address. While this mechanism does not support pruning of the broadcast tree to conserve network resources, it can be used to distribute information to all nodes in the ad hoc network subscribed to the destination multicast address. This mechanism may also be useful for sending application level packets to all nodes in a limited range around the sender.

To utilize this form of multicasting, when an application on a DSR node sends a packet to a multicast destination address, DSR piggybacks the data from the packet inside a ROUTE REQUEST targeted at the multicast address. The normal ROUTE REQUEST propagation scheme described in Section 3.2 will result in this packet being efficiently distributed to all nodes in the network within the specified hop count (TTL) of the originator. After forwarding the packet as defined for Route Discovery, each receiving node then individually examines the destination address of the packet and discards the packet if it is destined to a multicast address to which this node is not subscribed.

3) Location of DSR functions

When designing DSR, we had to determine at what layer within the protocol hierarchy to implement ad hoc network routing. We considered two different options: routing at the link layer and routing at the network layer. Originally, we opted to route at the link layer for several reasons:

Pragmatically, running the DSR protocol at the link layer maximizes the number of mobile nodes that can participate in ad hoc networks. For example, the protocol can route equally well between IPv4, IPv6 and IPX nodes.

Historically, as described more fully in Section 5, DSR grew from our contemplation of a multi-hop propagating version of the Internet's Address Resolution Protocol (ARP), as well as from the routing mechanism used in IEEE 802 source routing bridges. These are layer 2 protocols.

Technically, we designed DSR to be simple enough that that it could be implemented directly in the firmware inside wireless network interface cards, well below the layer 3 software within a mobile node. We see great potential in this for DSR running inside a cloud of mobile nodes around a fixed base station, where DSR would act to transparently extend the coverage range to these nodes. Mobile nodes that would otherwise be unable to communicate with the base station due to factors such as distance, fading, or local interference sources could then reach the base station through their peers.

4) NS2 (network simulator)

Network Simulator (Version 2), widely known as NS2, is simply an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviours. Due to its flexibility and modular nature, NS2 has gained constant popularity in the networking research community since its birth in 1989. Ever since, several revolutions and revisions have marked the growing maturity of the tool, thanks to substantial contributions from the players in the field. Among these are the University of California and Cornell University who developed the REAL network simulator,1 the foundation which NS is based on. Since 1995 the Defence Advanced Research Projects Agency (DARPA) supported development of NS through the Virtual Inter-Network Test-bed (VINT) project, Currently the National Science Foundation (NSF) has joined the ride in development. Last but not the least, the group of researchers and developers in the community are constantly working to keep NS2 strong and versatile. Again, the main objective of this book is to provide the readers with insights into the NS2 architecture. Researchers of IEEE Institute of Electrical and Electronics Engineering has described problem over simulation and no proper working for DSR has been implemented on NS2.

NS2 consists of two key languages: C++ and Object-oriented Tool Command Language. While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events. The C++ and the OTcl are linked together using TclCL. Mapped to a C++ object, variables in the OTcl domains are sometimes referred to as handles. Conceptually, a handle (e.g., n as a Node handle) is just a string in the OTcl domain, and does not contain any functionality. Instead, the functionality (e.g., receiving a packet) is defined in the mapped C++ object (e.g., of class Connector). In the OTcl domain, a handle acts as a frontend which interacts with users and other OTcl objects. It may define its own procedures and variables to facilitate the interaction. Note that the member procedures and variables in the OTcl domain are called instance procedures (instprocs) and instance variables (instvars), respectively. Before proceeding further, the readers are encouraged to learn C++ and OTcl languages. NS2 provides a large number of built-in C++ objects. It is advisable to use these

C++ objects to set up a simulation using a Tcl simulation script. However, advance users may find these objects insufficient. They need to develop their own C++ objects, and use a OTcl configuration interface to put together these objects. After simulation, NS2 outputs either text-based or animation-based simulation results. To interpret these results graphically and interactively, tools such as NAM and XGraph are used. To analyze a particular behaviour of the network, users can extract a relevant subset of text-based data and transform it to a more conceivable presentation.

The project stands for the simulating purpose with research purpose its only field of design. Many researchers have given DSR and MANET related projects but no such implementation which supports both consecutively is done with real case scenarios. Till now only Routing protocols are compared or implemented solemnly on pseudo networks. But this project deals with real cases and each constraint are taken in account to verify its correctness by graph and visualizing each packet of information. Some salient features of the project are:

- Real Time Scenario for implementation.
- Constraints are taken in best of knowledge of area.
- Graph and visual description of project.
- Bandwidth for Voice Communication is taken in consideration.

10. Conclusion

This implementation of DSR source routing protocol includes most of the basic facilities described in DSR–MANET IEEE draft, but some optimization measures like cached route request, and flow control are not implemented in this implementation. Packet salvaging, automatic route shortening (gratuitous route reply), caching negative information, caching overhead routing information and increased spreading of route error packets are the other options available in draft which can be implemented in this implementation the DSR protocol allows multiple routes to any destination and allows each sender to select and control the routes used in routing it's for example for use in load balancing or for increased robustness. Other advantages of the DSR protocol include easily guaranteed loop-free routing, support for use in networks containing unidirectional links, use of only "soft state" in routing, and very rapid recovery when routes in the network change. The DSR protocol is designed mainly for mobile ad hoc networks of up to about two hundred nodes, and is designed to work well with even very high rates of mobility.

References

[1] A Jean Marie and L Gun Parallel queues with resequencing Journal of ACM, Vol. 40, Nov. 1993, 1188-1208.

[2] M. Aissani, M. R. Senouci, W. Demigna and A. Mellouk, "Optimizations and Performance Study of the Dynamic Source Routing Protocol," *International Conference on Networking and Services (ICNS '07)*, Athens, 2007, pp. 107-107.

[3] Z. G. Al-Mekhlafi and R. Hassan, "Evaluation study on routing information protocol and dynamic source routing in Ad-Hoc network," *2011 7th International Conference on Information Technology in Asia*, Kuching, Sarawak, 2011, pp. 1-4.

[4] K. A. Anang, L. Bello, T. I. Eneh, P. Bakalis and P. B. Rpajic, "The performance of dynamic source routing protocol to path loss models at carrier frequencies above 2 GHz," *2011 IEEE 13th International Conference on Communication Technology*, Jinan, 2011, pp. 151-155.

[5] Ali Norouzi IJCSNS International Journal of Computer Science and Network Security, vol. 10, no. 6, June 2010

[6] Braden, R., "Requirements for Internet Hosts - Communication Layers" RFC Editor, United States, October 1989.

[7] Boon-Chong Seet, Bu-Sung Lee, Chiew-Tong Lau, Optimisation of Route Discovery for dynamic source routing in ad hoc networks, IET Journals & Magzines, October 2003.

[8] Borrong Chen and C. Hwa Chang, "Mobility Impact on Energy Conservation of Ad-hoc Routing Protocols", SSGRR 2003, Italy, July–August 2003.

[9] Baisakh International Journal of Computer Applications (0975 – 8887) Volume 68, No.20, April 2013

[10] Chris Karlof and David Wagner, Secure routing in wireless sensor networks: attacks and countermeasures, 2003

[11] C Hedrick Routing information protocol Internet Request for Comments RFC 1058, June 1998.

[12] C. Siva Ram Murthy, B. S. Manoj, "Ad Hoc Wireless Networks Architecture and Protocols", 2nd ed, Pearson Education, 2005.

[13] C. Yu, B. Lee, H. Youn, "Energy Efficient Routing Protocols for Mobile Ad Hoc Networks", Wireless Communication and Mobile Computing, Wireless Com. Mob. Computing (2003).