http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 5 (2): 373-385, 2006
ISSN 1812-5638

Scalability of Zone Routing Protocol Extensions for Mobile Ad-hoc Networks

[1]Muhammad Bilal Nazir, [1]Malik Sikander Hayat Khiyal and [2]Tauseef Ur Rahman
[1]Department of Computer Science, [2]Department of Telecom and Computer Engineering,
International Islamic University, Islamabad, Pakistan

Abstract: This study is a modification of the research by Dr. Sinha, which explains the scalability of ZRP for unidirectional links. Ad-Hoc Networks are the mobile nodes that communicate with each other. An ad-hoc network neither not have any infrastructures nor it posses any fixed topology. The routing of data between such networks is quite a difficult task. Several routing protocols have been proposed for wireless ad-hoc networks. Most of the proposed are assumed to be of bi-directional links but in actual the ad-hoc networks have different transmission range and routing capabilities. The Zone routing Protocol is designed keeping the unidirectional behavior of nodes in the networks. The Zone Routing Protocol (ZRP) is a hybrid routing protocol that proactively maintains routes within a local region of the network (which we refer to as the routing zone). Knowledge of this routing zone topology is leveraged by the ZRP to improve the efficiency of a reactive route query/reply mechanism. The ZRP can be configured for a particular network through adjustment of a single parameter, the routing zone radius. In this study, we address the issue of configuring the ZRP to provide the best performance for a particular network at any time. Previous study has demonstrated that an optimally configured ZRP operates at least as efficiently as traditional reactive flood-search or proactive distance vector/link state routing protocols (and in many cases, much more efficiently)

Key words: ZRP, inter zone routing protocol, intra zone routing protocol, querry control mechanism, zone radius, bordercast

INTRODUCTION

Ad-hoc networks are mobile wireless networks that have no fixed infrastructure. There is no router attached to nodes rather each node itself works as the router and routes data to other nodes. The mobility of the nodes makes the topology of the network time-variant. The Change in the topology of the network depends upon the velocity of the mobility of the nodes. Furthermore, the wireless network is characterized by low bandwidth links that are subject to harsh conditions of fading and interference[1]. Thus, routing in such a network is difficult and challenging. Many routing protocols have being proposed which are classified on basis of reactive or proactive. When proactive routing protocols are employed a node would possess routing information to a destination before it would actually need to route data to that destination. For this purpose routing tables are maintained. Route updates are exchanged periodically to reflect the changes in topological information. Popular proactive routing protocols for ad hoc networks include the Destination Sequenced Distance Vector (DSDV) Protocol, the Wireless Routing Protocol and the Source Tree Adaptive Routing (STAR) Protocol. Numerous on-demand routing protocols have been proposed. Some of the on-demand routing protocols are the Adaptive On-demand Distance Vector (AODV) protocol, the Dynamic Source Routing (DSR) Protocol and the Temporally Ordered Routing Algorithm (TORA). The proactive routing protocols usually require the maintenance of routing tables and thus, in the dynamically changing mobile ad hoc network, nodes would need to exchange routing updates periodically. This exchange of route updates would consume bandwidth and if the network is large, these control messages could contribute to a significant amount of overhead. On the other hand if on-demand routing protocols are used, when data is to be routed to a destination, a source node might be required to initiate a search for the destination. If the network is large, significant latency may be incurred before the destination is found. Thus, the scalability of both the table-driven and the on-demand routing protocols is limited. The Zone Routing Protocol (ZRP) provides a hybrid proactive/reactive routing framework in an attempt to achieve scalability. Each node would maintain routing tables which would only offer routes to a destination if the destination were to be mechanism called

Corresponding Author: Dr. Malik Sikander Hayat Khiyal, Department of Computer Science, International Islamic University, Islamabad, Pakistan

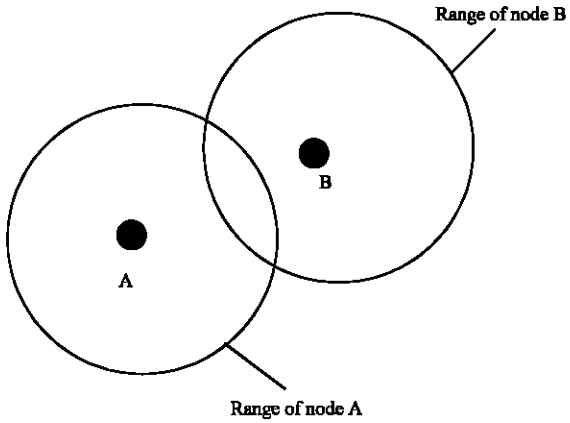

Fig. 1: Ad-hoc network between two nodes

bordercasting. Bordercasting provides an efficient means for searching for a destination by sequentially using the routing tables of the intermediate relay nodes. The routing protocols mentioned in the previous paragraph assume that the links in the network are bi-directional in nature. However, a wireless ad-hoc network could potentially consist of heterogeneous nodes with differing power capabilities. The transmission range of one node might be different from that of another. Thus, a node (node A) whose transmission range is larger than that of another node (node B) will be able to transmit information to node B, but will be unable to receive the destination. For this purpose routing tables are maintained. Route updates are exchanged periodically to reflect the changes in topological information. Each node would maintain routing tables which would only offer routes to a destination if the destination were to be transmissions of node B (Fig. 1). This would result in the creation of unidirectional links in the network. Routing protocols for ad-hoc networks with unidirectional links have been studied to a limited extent. They identify the fact that a reverse path is required from node B to node A (Fig. 1) in order for node A to realize that it has the link to node B. However, these studies do not address scalability issues. On the other hand, an on-demand flooding would ultimately result in the data being delivered to the destination if a path was to exist, but then, such a scheme is very inefficient. In this study we provide extensions to the zone routing protocol in order to provide a scalable framework for routing when unidirectional links are present. When the reverse path from the node at the tail of a unidirectional link to the node at the head of the link is long, we resort to an on-demand search mechanism. The on demand search mechanism recursively attempts to build a path to the destination by recognizing nodes that have a route to the destination[2].

OVERVIEW OF THE PROPOSED EXTENSIONS TO THE ZONE ROUTING PROTOCOL

Our protocol consists of the Intra Zone Routing Protocol (IARP), which is the proactive component and the Inter Zone Routing Protocol (IERP), which is the reactive component. The IARP is responsible for maintaining information about the neighbor's links and nodes. Every node transmits information about its inbound neighbors (besides other information to be described later) to nodes within a restricted neighborhood defined by the parameter called the Zone Radius. This information is used by each node (say node A) to compute its outbound tree, which is the shortest path tree rooted at node A to nodes from which the previously mentioned transmission restricted to Zone Radius hops has been heard. The nodes reachable by the computed outbound tree, define the node's zone and hence, unlike the usual notion of clusters, zones overlap heavily. The IERP is the component that enables route computation when the outbound tree maintained by the IARP of a node does not have a path to the destination. Bordercasting, which refers to sending the route query by using a tree (bordercast tree) to a set of nodes (the border nodes), preferably towards the periphery of the zone, is an important sub-component of IERP. The border nodes are nodes that are known to have links to other nodes that the current node cannot reach by means of its outbound tree. The computation of the bordercast tree is complicated by the possible existence of unidirectional links. The border nodes upon receiving a bordercast message, repeat the same algorithm (as executed at the source), which involves checking if a path to the destination exists within the node's local routing table and bordercasting again if a path to the destination is not known locally. The intermediate nodes that initiate a bordercast, include their unique identifiers in the route query packet before forwarding it. Once the query reaches a node that knows a path to the destination, it includes its identifier in the response packet and sends the response to the originator of the query. The list of nodes that stamped the packet while it traversed its forward path is used for identifying the reverse path via which a response is sent to the source of the query. Note that this list only consists of the border nodes of some intermediate zones and the computation of the bordercast tree would guarantee that each of these border nodes has a path to the next as well as previous border nodes. Bordercasting usually results in an increase the number of query threads. Without implementing mechanisms for controlling these query threads, deploying the protocol could result in flooding the network with query messages. This is highly inefficient in terms of the number of messages. Some query control mechanisms have been

adopted from the original ZRP proposal and have been modified to function in the presence of unidirectional links. The query enhancement mechanism, which is a part of the IERP algorithm, is useful for computing route that consists of unidirectional links with inclusive cycles larger than the zone size. In the event that a route to the destination is not discovered, this mechanism computes a set of alternative destinations that are known to have paths to the desired destination. The original sender then queries for this set of alternative destinations, by initiating a fresh query. Repeating the same mechanism can further enhance this enhanced query. But for practical implementations, the maximum number of times that a query may be enhanced may be limited to some predetermined value. An important point to note is that our protocol treats a bidirectional link as consisting of two separate unidirectional links and in the rest of the document, a 'link' denotes a unidirectional link. The rest of this document consists of a description of the IARP component, the IERP component and the various Query Control Mechanisms used for improving the performance of our protocol. The novel query enhancement mechanism is described as part of the IERP.

INTRA ZONE ROUTING PROTOCOL (IARP)

The proactive component: The goal of the IARP algorithm is to maintain an outbound tree to some nearby nodes. In case of networks with only bi-directional links, ZRP defines the zone as consisting of nodes which are within ZONE_RADIUS hops. In our protocol, zone membership of a node is not determined by the number of hops to the node, but rather by the number of hops from the node. Thus, for a node (say j) to be in the zone of a node (say I), node I must be reachable from node j in ZONE_RADIUS hops or less. For computing the outbound tree, every node uses the units obtained from some nearby nodes. Every node (say node x) formulates a unit that consists of the following information:

- IN (Inbound Neighbors): The set of neighbors which have a link to the node x.
- ON (Outbound Neighbors): The set of neighbors to which node x has a link.
- OT (Outbound Tree Nodes): The nodes on the outbound tree of node x (computed from the units obtained from other nodes).
- SN (Sequence Number)
- UF (Urgent Flag): If this flag is set, then the unit is to be forwarded as soon as possible. This is typically used when the unit is generated after the deletion of a link since mis-information about presence of a link should be removed from other nodes' routing tables as soon as possible, in order to avoid wrong route computations.
- TTL (Time to Live): Number of hops up to which the unit can be further forwarded. The TTL is initialized to the ZONE_RADIUS and is decremented as the unit traverses a path.

At startup, IN, ON and OT are each initialized to empty. A sequence number is assigned to the SN field and the UF is not set. The information in the IN field of a unit is used for computing the outbound tree. The sequence number, SN is used to identify the most up to date unit when more than one are received. The fields (ON and OT) are used for computing the bordercast tree and for the query control mechanisms used in the IERP protocol. The fields IN and ON have a space complexity bounded by the maximum degree of a node, i.e., the size of the IN field is O(D), where D is the maximum degree of a node. The field OT has a space complexity of the maximum number of nodes in a zone, i.e. the size of the OT field is O(N), where Nz is the maximum number of nodes in a zone. Hence, the IARP message size depends on the maximum degree of a node and the maximum number of nodes in a zone, which in turn depends on the ZONE_RADIUS. But the important thing to note is that the size of the unit is independent of the network size and hence is key to the scalability of our protocol. Periodically (with period BEACON_INTERVAL), the unit is formulated and a new sequence number is assigned. A packet which includes the node's unit and units from other nodes for which the TTL has not become zero, is then locally broadcast. Other nodes use sequence numbers to keep track of the latest unit initiated by a node. Each unit has a purge time (determined by the parameter UNIT_PURGE_TIME_INTERVAL) associated with it. If a link goes down, then the node, on which the link was incident, creates a new unit and sets the urgent flag for this unit. The urgent units are forwarded in a separate packet immediately rather than waiting for the next beacon to be generated. Using the information in the IN field of each received unit, the outbound tree is computed periodically. For computing the outbound tree, the link information from the IN fields of all received and stored live units (which have not expired) are used to construct a graph which represents a partial network. As an example, let node x be in the process of computing the outbound tree and let node x have a unit that originated at node I. If this unit includes nodes j, k and l in the IN field, then links j->I, k->I and l->I are added to the graph being constructed. A shortest path algorithm is then used to compute the shortest path tree from node x to other nodes. Thus the IARP protocol supports the maintenance of an outbound tree from every node.

Routing of data packets within a zone: Any routing protocol may be used for routing within a zone. A shortest path tree may be computed from node x to other nodes within its zone. However, it is to be noted that the entire route might have to be included in the packet. This is required since, for some unidirectional links present along the path to the destination, the presence of the link might not be known to the node at the head of the unidirectional link. Thus, the head node might not even be aware of the existence of the tail node of the unidirectional link and if only the address of the next border node is provided, the node might not be able to forward the packet appropriately. Providing a source route will enable the node to not only forward packets correctly but also to cache the information with regards to the unidirectional and use it for subsequent routing requests. The details of the methodology for caching information with regards to the unidirectional link are beyond the scope of this study. Note that the outbound tree information might also be cached or propagated to enable more efficient routing, but this would result in excessive overhead.

INTER ZONE ROUTING PROTOCOL (IERP)

The reactive component: The purpose of the IERP algorithm is to compute routes when the outbound tree computed by the IARP algorithm does not have a route to the destination. IERP mainly relies on a mechanism called bordercasting, which stands for forwarding the route request to a subset of nodes (border nodes) using a tree, called the bordercast tree. When the routing layer receives a route request, the outbound tree is first inspected to look for a route to the desired destination. If this lookup fails, the route request is forwarded by the node to its border nodes (the methodology for choosing border nodes is described in a subsequent subsection) by using its bordercast tree. If a border node knows a path to the destination, then the particular border node responds to the query; otherwise bordercasting is repeated at the border node. Every border node which receives the query and does not know a path to the destination, checks its inbound tree to see if it knows of alternate nodes which have a path to the desired destination. If such nodes exist, the border node then sends a query enhancement message to the sender informing it of these alternate destinations which might be queried for, in order to reach the desired destination. If the source of the query does not receive a route response message within a preset time interval (ENHACEMENT_INTERVAL), it then checks for any received query enhancement messages. If at least one query enhancement message has been received then the alternate destinations specified in the query enhancement messages are specified in a newly created enhanced query message. This enhanced query is then processed like a new query. However, the number of times that a query can be enhanced is usually limited to a preset number, since multiple queries for the same destination might lead large latencies in route computation. Note that the above mechanism assumes small inclusive cycles. To enhance the performance of our protocol for large inclusive cycles, we use a heuristic approach outlined below. If the source does not receive a response to a query (either a query response message or a query enhancement message within a preset time-out referred to as the ENHANCEMENT_INTERVAL, it issues a fresh query to enquire about nodes that know of partial paths to the destination. Thus the source learns of alternate destinations for which it can issue a fresh query. Since the bordercast nodes have already been enquired, the heuristic uses a tree different from the bordercast tree to possibly reach a different subset of border nodes. Furthermore, when the bordercast tree is empty such an enhancement request needs to be sent out. For this purpose, a two-way tree is used to initiate a modified bordercast. The two-way tree is a shortest path tree such that all nodes in the tree are two-way nodes. Two-way nodes are those nodes in a querying node's outbound tree that can reach the querying node by using their own outbound trees. The advantage of using such a tree is that all nodes on this tree are known to have a reverse route to the current node, unlike in a bordercast tree in which only the border nodes are required to be two-way nodes. This two way feature is used when responding to a route query. It is to be noted that the leaves of the two-way tree may be different from the leaves of the bordercast tree described in the previous paragraph. These concepts will be elucidated in the forthcoming subsections. Following is a list of the different kinds of IERP messages deployed.

Route Query Request (RQRQ): It is targeted for one destination and is generated by the source of the query. This message is bordercast using the bordercast tree.

Query Enhancement Request (QERQ): A message explicitly requesting other nodes to respond if they know of alternate nodes that have paths to the original destination, so that these nodes can then be queried for as alternate destinations. This message is bordercast using the Two-Way Tree (TWT bordercast) when either the bordercast tree is empty or when it has failed to reach nodes that know of alternate destinations.

Enhanced Route Request (ERRQ): This message is similar to the Route Query Request except for the fact that it

cannot be enhanced any further if it has already been enhanced MAX_NUMOF_ENHANCEMENTS times, the upper limit on the number of times that a query can be enhanced. Thus, a counter is incremented each time an ERRQ is transmitted to keep track of the number of times it has been enhanced.

Query Response (QR): QR is generated as a response to a RQRQ or QERQ, when a border node knows a path to the destination by means of its outbound tree. This response is sent back using the reverse path formulated by the recorded traversed path in the query. The path recorded consists of only the border-nodes processing the query and by virtue of the fact that the border nodes (in the bordercast tree) are two-way nodes, the reversed sequence of border nodes can be followed on the reverse path. The non-border nodes visited in the reverse path could be different from the ones traversed on the forward path.

Query Enhancement Response (QER): This message could be generated for an RQRQ, QERQ or an ERRQ. For an ERRQ, a QER can be sent back to the original source only if the query can be enhanced at least once more (based on the MAX_NUMOF_ENHANCEMENTS). It is generated if the border node does not have a path to any of the queried destinations but it knows of at least one node with a path to at least one of the queried destinations by means of its inbound tree. The set of nodes which are known to have a path to the subset of queried destinations are then reported back to the source node, which then adds these nodes to the set of alternate destinations to form a modified ERRQ, if this query does not result in the discovery of a path to any of the former destinations. As described above, the bordercast tree and the two-way tree are very crucial to the functioning of the IERP. Some mechanisms that are a part of the IERP, especially the query control mechanisms are based on the assumption that the packets are transmitted reliably and are not dropped by a lower layer. The following sub-sections describe these two trees in detail and the section concludes with a detailed description of the IERP[3].

BORDERCAST TREE

The bordercast tree is a tree used for sending a bordercast message to a set of nodes. When the destination is not reachable by using the outbound tree computed by IARP, this tree is used for forwarding the route query. As the bordercast tree, preferably, is a shortest path tree, it is a sub-graph of the outbound tree. Here are some other properties that the bordercast tree needs to satisfy:

- When a route has been discovered from a source node to a destination node, unlike in the case of bi-directional networks, it is possible that the query response may not be able to retrace the path traversed by the query in the reverse direction, as some of the links in the forward path may be unidirectional. It appears that another query for discovering a path from the destination to the source might be required. However, if each border node has the preceding querying border node (let us denote each border node involved in the query as a center node) in their outbound trees then the same center nodes can be used to tunnel the query response back to the source. A node in the outbound tree that has a path to the root node of the tree, is defined as a two-way node. The set of two-way nodes is determined using the list of nodes in the Outbound Tree (OT); note that the list is a field in the unit. Hence, it is essential that each border node be a two-way node.
- The border nodes are supposed to lead to destinations that are being searched for. Hence, they must have links incident to nodes outside the bordercast tree. The list of Outbound Neighbors (ON; also a field in the units), is used to identify the nodes from which such links are incident. Such nodes are candidates for being chosen as border nodes.
- The inner nodes of the bordercast tree should not be candidates for border nodes. Note that, the inner nodes of the bordercast tree do not have links incident to nodes that are not a part of the outbound tree of the querying node.

Based on the above criteria, the following algorithm is used for Constructing the bordercast tree:

- Identify the two-way nodes in the outbound tree. The nodes that are one hop away on the outbound tree, are always two way nodes. This is based on the following argument: Let node B be a node at a distance of one hop from node A in the outbound tree of node A. As node A is using the link from node A to node B, node A must have node B's unit, implying that the path from node B to node A must be less than the ZONE_RADIUS. The link from node A to node B and the reverse path from node B to node A together form a cycle of length at most ZONE_RADIUS+1. Hence node B would also know of a path to node A from its IARP.
- Mark nodes (may or may not be two-way nodes) in the outbound tree that have outbound neighbors not belonging to the outbound tree, as candidates for being border nodes.

- Unmark a marked node if there is another marked node on the path from that node to the root of the outbound tree (which is in fact the node initiating the bordercast). Since the leaves of the bordercast tree are going to be the final border nodes, two nodes along the same path on the tree, from the root, cannot be border nodes. If the candidate node closer to the root (say A) is not selected as a border node in the final bordercast tree, then some of the nodes that A can reach and which are outside the outbound tree might not be reachable at all by the route query.
- Initialize the bordercast tree to the smallest rooted sub-tree of the outbound tree, which has the border nodes (marked as in Steps 2 and 3) as its leaves.
- If there is a border node which is not a two-way node then delete the sub-tree rooted at its parent from the bordercast tree.
- Repeat 5 until all such nodes are pruned. Thus, this algorithm computes a bordercast tree such that the border nodes are two-way nodes. Figure 2 shows the formation of the outbound tree and the bordercast tree by means of an example. Figure 2a shows a network that has only one unidirectional link. The ZONE_RADIUS is assumed to be 3. Figure 2b shows the links that are known to node A. Note that nodes that have a path of ZONE_RADIUS hops or less have broadcast their units, which contains their inbound neighbors. For example, the link FE is known at A because node E has a path of ZONE_RADIUS hops to node A. But the shortest path from F to A is 4 (more than ZONE_RADIUS) hops and hence, the link EF incident on F is not known at A. Figure 2c shows the outbound tree at the node A, computed from the available link state information. It is essentially the shortest path tree computed from the link state shown in Fig. 2b. Using the outbound tree, the bordercast tree is computed and this tree has only one border node, namely E, since E knows of links to nodes (namely F and G corresponding to the links EF and EG) that do not belong to the outbound tree. This is made known to node A by the ON field of the unit that originated at node E.

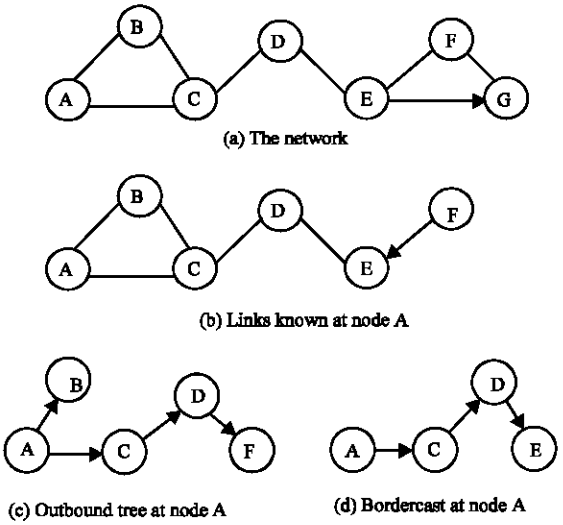

Fig. 2: Example showing the outbound of a node

TWO WAY TREE

The two-way tree is used to find alternate destinations. Alternate destinations are nodes that are known to have routes to the desired destination. The process of trying to compute the list of alternate destinations is termed as query enhancement. When the bordercast tree fails to compute a route to the destination either because the tree is empty or because the unidirectional links prevented any route computation or query enhancement, the two-way tree is used to possibly reach a different set of nodes in an attempt to enhance the query. In the query enhancement phase, the aim is to be able to reach some nodes that were not reached by the bordercast tree. So a tree is needed whose leaves are two-way nodes (so that the query response can be sent back through the same border nodes) and is different from the bordercast tree. So, we simply define the two-way tree as the largest sub-tree of the outbound tree that has all of its nodes as two-way nodes. Figure 3 shows the outbound tree, the two-way tree and the enhancement mechanism with a 8 node network. Figure 3a shows the network, which has two unidirectional links. The ZONE_RADIUS is assumed to be 3. The link state information available at node 6 is shown in Fig 3b. Based on the available link state, node 6 computes the outbound tree, which is shown in Fig 3c. Since none of the non-root nodes in the outbound tree, namely 4, 5, 7 and 8 have reported outbound links to nodes not in the outbound tree, none of the nodes are a part of the bordercast tree. So, the bordercast tree is empty in this case. However, all these nodes, 4, 5, 7 and 8 have reported to node 6 that node 6 exists in their outbound trees, using the OT field of the corresponding units. Thus the two-way tree is same as the outbound tree for this example.

If node 6 issues a query request with the destination node as node 1, then first, node 6 finds that the outbound tree does not have a route to node 1. The problem stems from the fact that the inclusive cycle for the unidirectional link from node 8 to node 1 is too large for a zone radius of 3 and so 8 is not aware of the link to node 1. Hence node 6 has to initiate a bordercast. But since it has an empty bordercast tree, it tries to enhance the query by asking other nodes if they know of nodes having paths to the

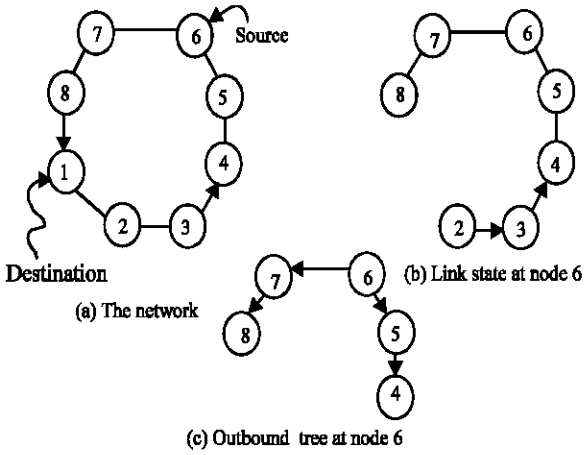

Fig. 3: Example showing the outbound tree and two way tree

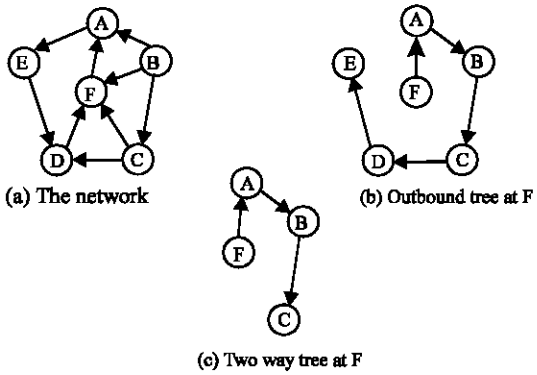

Fig. 4: Showing outbound and two way tree

destination. The two-way tree is then used to send the query enhancement request. When the request reaches node 4, it computes the inbound tree to the destination (node 1), using the link state it currently possesses. The idea of the inbound tree to the destination is to compute shortest paths to the destination from other nodes. Thus node 4 is able to compute that node 8 is an alternate destination for node 1. This inbound tree to node 1 (trivially the link 8 to 1 here) is reported back to the source (node 6) .After the ENHANCEMENT_INTERVAL, the source issues a new query request with a list of alternate destinations obtained from the query enhancement responses obtained. This new request only has one alternate destination (namely 8). Since the source has a path to the node 8 in its outbound tree, the route computation is completed, the discovered route being through the nodes 7 and 8 to the destination viz., node 1. Figure 4 shows outbound tree and two-way tree with another example. The ZONE_RADIUS is 3 for this example and all links are unidirectional in the network shown in Fig 4a. The distance from any node, to node F is within 3 hops, hence node F has the entire topology using which it computes the outbound tree shown in Fig. 4b. Note that outbound tree of node F has a path to node E, whereas the outbound tree of node E does not have a path to node F. In fact, the outbound tree of node E will be empty as the only outgoing link from node E is link EA and the shortest path from A to E is 4 hops, which is larger than the ZONE_RADIUS. Thus all outbound tree nodes need not be two-way. In the outbound tree of node F, only nodes A, B and C are two-way. And the largest rooted sub-tree of the outbound tree with all nodes being two-way nodes, i.e., the two-way tree is shown in Fig. 4c. Nodes A, B and C learn of the paths to F as they lie on the cycle FABC, whose length 4 is within 1 more than the ZONE_RADIUS. As opposed to the previous example, this example also shows that the two-way tree need not be same as the outbound tree. Note that if the outbound tree is empty (like in the case of node E), no IARP is possible and hence one will have to rely on an on-demand flood mechanism to discover a destination node.

THE IERP ALGORITHM

The functioning of the IERP algorithm at the source and at the border nodes are presented as two different flow charts in Fig. 4, as the processing at the border nodes is different from that at the source. The details of the flow chart at different states are described below. The functionality of the states in Fig. 5 and 6 is as follows:

State 1: A new query initiated by the node has one destination and an enhanced query will have a set of destinations. If there is a path to any of the destinations in the outbound tree computed by the IARP algorithm, the path found is the desired path.

State 2: The bordercast tree could be empty (For an example refer to Fig. 3). In such a case an attempt is made to use a different tree to do the bordercasting to possibly learn about alternate nodes that know of one or more paths to the original destination.

State 3: The bordercast tree is stored in the query packet and is forwarded along the same tree. The intermediate nodes of the bordercast tree (non-border nodes) forward the query packet until it reaches a border node. The processing at the border node is shown in Fig. 4b. After sending the bordercast, there is a pause for ENHANCEMENT_INTERVAL, during which the source waits either for a query response or enhancement messages.

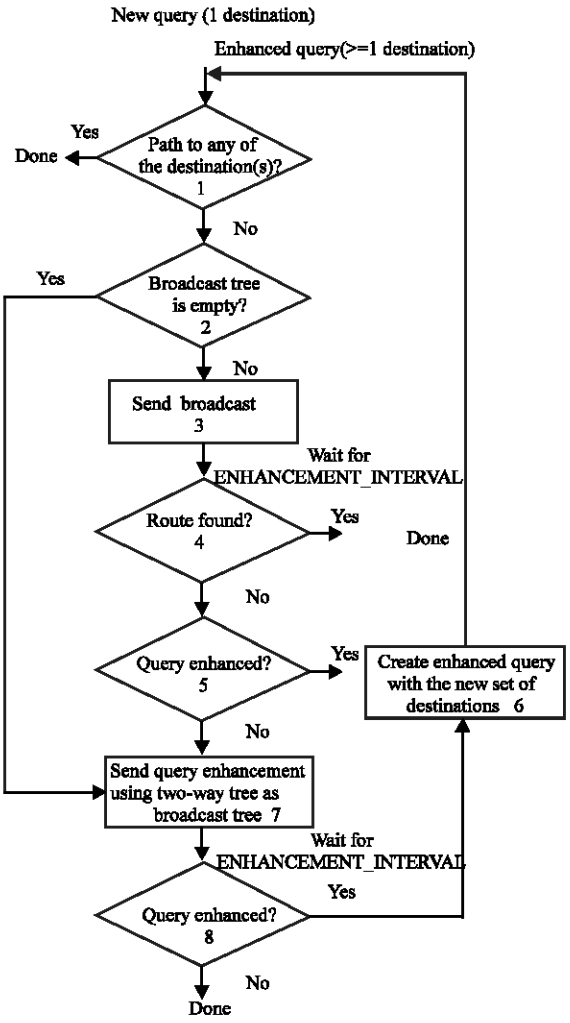

Fig. 5: IERP at route query source

State 4: If a response to the route query is received in the interim, then the query processing is termed complete and the computed route is returned to the higher layer.

State 5: Since a query response is not received, the source node checks for the reception of an enhancement message, the ENHANCEMENT_INTERVAL having passed since the initiation of the bordercast. If there were one or more query enhancement messages received in that interval, then the alternate destinations suggested in the query enhancement messages are queried for as they are supposed to have routes to the original destination.

State 6: A set of alternate destinations is formed from the query enhancement messages and is inserted into a new modified query, which is processed like the original query.

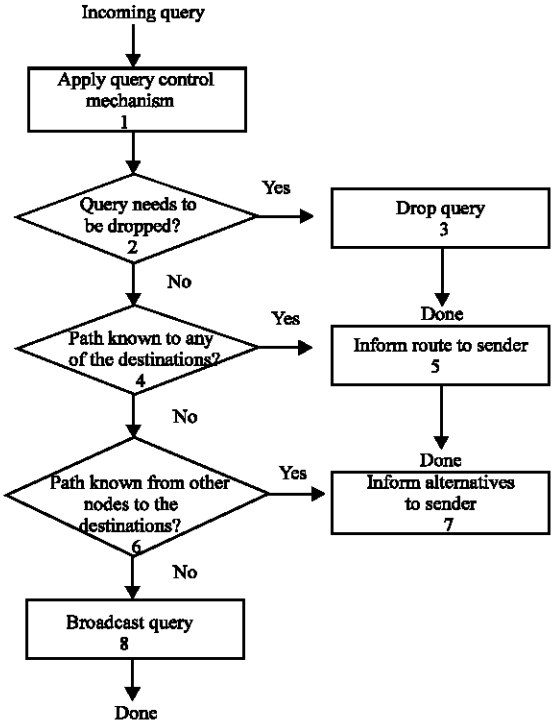

Fig. 6: IERP at border node

For practical implementations, the number of times that a query can be enhanced should be limited to reduce the amount of query traffic and the latency in finding routes, as every enhancement and repeated query for the same original destination increases the route discovery latency. For simplicity of presentation, the flow chart does not limit the number of times that the query can be enhanced.

State 7: Since the bordercasting did not result in any enhancement of the query, the bordercast tree is incapable of reaching nodes that can enhance the query (assuming no message losses). This state is also reached from the State 2, when the bordercast tree is empty. A different tree, namely the two-way tree, is then used for sending a request to enhance the query. The source and the border nodes forward this Query Enhancement Request (QERQ) just like they would forward a regular query, except that the two-way tree is used for bordercasting, instead of the bordercast tree. The key idea here is to try and discover nodes, which know of paths to the destination.

State 8: After waiting for ENHANCEMENT_INTERVAL, the source node checks to see if there were any responses to the request. Once again, as in State 5, if there were one or more Query Enhancement Responses (QERs) received

in that interval, the alternate destinations suggested in the QERs can be queried for as they are supposed to have routes to the desired destination. If no such enhancement message was received then the destination is assumed to be unreachable.

The various states of Fig. 6 are explained below. This flow chart shows the manner in which a query is processed at a border node.

State 1: Apply the query control mechanisms, namely Query Detection (QD) and Early Termination (ET). These mechanisms are described later in the study. This essentially involves extracting the query identifier and matching it with the recently cached query identifiers seen by the node. If the query identifier has been seen before, then the query can be dropped.

State 2: The node checks to identify if the query is to be dropped or sent out as another bordercast.

State 3: The query is dropped as the query control Mechanisms have identified this query thread to be unnecessary.

State 4: If a path is known to any of the destinations in the query, then the route discovery is complete.

State 5: A response to the query is initiated which contains the computed path (only the border nodes, also referred to as center nodes, traversed by the route query packet are recorded). The response is sent along a path that traverses the same center nodes. This is possible because each center node has a path to the previous center node. Thus, the response is forwarded from one center node to another center node until it reaches the source node, which initiated the query.

State 6: Inbound trees are computed for each of the destinations being queried. The links discovered by the IARP are used to compute these trees. If any such trees exist and can be computed, then the nodes (besides the destinations) in these inbound trees would denote the alternate destinations. These trees are computed using a mechanism similar to the one used for computing the outbound trees. The IN field of the live units available at the node are used to construct a graph. Then for each destination the shortest path algorithm is executed on the graph by considering each destination as a sink node. Any shortest path algorithm can be used to compute these inbound trees.

State 7: The inbound trees computed in State 3 are sent back to the sender using the same mechanism as in Step 2 above.

State 8: The node ID is stamped on the query packet and it is then sent out using the bordercast mechanism. If the query is Route Query Request (RQRQ) or Enhanced Query Request (EQRQ) then the bordercast tree is used for bordercasting. If instead, the request is a Query Enhancement Request (QERQ), the two-way tree is used for bordercasting.

QUERY CONTROL MECHANISMS

In the IERP algorithm, each bordercast usually results in increasing the number of query threads (unless there is only one border node). As a result of this, typically with every bordercast, the number of query threads keeps increasing. These query threads may result in the degeneration of the zone routing protocol to flooding. The Query Control Mechanisms are used to stop unnecessary route query threads, which are probing previously queried zones. These mechanisms were originally proposed for bi-directional networks for ZRP and have been modified to function with unidirectional networks in this work. This part of the work requires further investigation, but has provided encouraging experimental results.

Query Detection (QD): The goal of the QD mechanism is to identify nodes that do not need to initiate bordercast. Trivially, the nodes which have already initiated bordercast (e.g. the source node) or have been border nodes in some bordercast of the same query, need not perform subsequent bordercasts for the same query if there are no enhancements. To identify a query, the query identifier, which is a pair consisting of the source address and a unique query number assigned by the source, is used. Each border node keeps track of query identifiers seen in the recent past (based on the largest time taken by a query to transit from one node to another). After a border node receives a query, if the query identifier matches an identifier stored in the cache, then the node simply drops the query. Furthermore, if a node (say node x) has already been a non-border relay node for some query, it does not need to initiate a bordercast or be a border node for a subsequent query thread with the same query identifier. When an earlier query passed through node x, node x would have been selected as a border node if it were a candidate for being a border node. Hence, each non-border node also keeps track of query identifiers seen in the recent past. Thus QD helps in limiting the number of bordercasts that can take place, to the number of nodes in the network[4].

Early termination (ET): Although QD provides an upper bound on the number of bordercasts for a single query, it does not prevent previously traversed nodes (central and non-central) from being a non-central node in the future. ET states that if a query has visited a node (as a border node or otherwise), then it need not transport a thread for the same query to any other border node. For this we assume that if node A's outbound tree has a link from node B to node C, then node B also knows of the link from node B to node C. This condition might not hold in some cases. It appears that if the network has a large percentage of unidirectional links, then this condition might not hold. Figure 7 shows it with an example. The ZONE_RADIUS is assumed to be three. In both the networks (Fig. 7a and b), node A learns about the link from node B to node C. But in Fig. 7 a, node B knows of the link (node B to C), because it is bidirectional, whereas in Fig 7. b, B does not learn of the link because the path from node C to node B is more than 3 (ZONE_RADIUS) hops. With this assumption, let us consider the following two cases for node B[5].

- Node B has been a border node for a particular query. Subsequently, a bordercast message for the same query from node D is to use node B as a non-border node with node C as a node following B in the bordercast tree. Then by the assumption, node B knows about the existence of the link to node C and hence, node B would have already considered that link, when it received the bordercast message in its role as a border node. So, the bordercast message from node D can be dropped at node B[6].
- Node B has been a non-border node (an earlier bordercast from a different node, node M) and now a bordercast for the same query from node D is to use node B as a non-border node with node C as a node following node B in node D's bordercast tree. Then by the assumption, node B knows about the existence of the link to node C and hence, the existence of the link BC must have been propagated to node M and this information must have been taken into account by node M when this node M constructed its bordercast tree. So, the bordercast message from node D can be dropped at node B. The above two cases do not consider the latency incurred while updating link information and possible discrepancy of information between different nodes about the existence/non-existence of links. Hence QD and ET together imply that a node need not process a query thread if the node has been seen that query before. So, for implementing QD and ET, the query identifier is cached at every node processing the query and is kept for a short preset time interval. If any other query thread with the same identifiers is received in that interval, then it is dropped without further processing. Thus, the number of messages propagated for a single query is upper-bounded by the number of links in the network, which is same as the number of messages required for flooding the network. It is to be noted that this upper bound is applicable for a Route Query Request (RQRQ), or a Route Enhancement Request (RERQ), or a Enhanced Route Request (ERRQ). As shown in Fig. 3a, a route request may result in the source first sending out a RQRQ, then sending out a RERQ upon receiving no response and then sending out an enhanced query ERRQ if it gets a response to the enhancement request QER. Assuming that MAX_NUMOF_ENHANCEMENTS is set to one, these three messages, namely RQRQ, ERRQ and ERRQ together achieve a message complexity of thrice the number of links in the network.

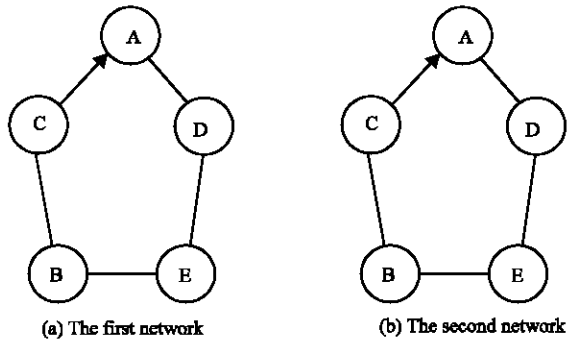

Fig. 7: Links between different nodes of an ad-hoc network

RESULTS AND DISCUSSION

We implemented the Zone Routing Protocol with our extensions using the ns-2.26 simulator. The extension, which uses the two-way tree for soliciting alternate destinations, has not been studied using our simulator. The results presented here are produced not having that extension. The study reported here is based on a network of 100 nodes moving in an area of 1750×1750 m following the random waypoint model. In this model every node picks random position and moves towards it with a speed uniformly chosen between 0 and a maximum speed (we have varied it between 0 and 50 m s^{-1}). Once it reaches that position, it pauses for some amount of time and then chooses another location and the process is repeated. For our simulations, we used a pause time of 10 to simulate continuous mobility.

To simulate unidirectional links, we used two different transmission ranges. Some nodes had a transmission range of 250 m and others had a range of 125 m. All the

data points reported are based on simulating the network for 2 h of real time. Since the network can get partitioned, the route queries used to test the protocol were generated in such a way that a query was generated only when the underlying network had a route to that particular destination. Queries were generated at the rate of 1 query/s, by picking random source and destination nodes that had a path between them in the underlying network at that time. We studied the performance with a varying percentage of nodes having lower transmission ranges, which in turn varies the number of unidirectional links in the underlying network. We tested the performance with 10, 20 or 40% of the nodes having lower transmission ranges. We present four kinds of results in this section and discuss them in the following paragraphs[7,8]. Figure 8 shows the percentage of queries that were resolved. Queries could be resolved if the IARP or IERP succeeds, or if the query is enhanced by some node that reports that there exist alternate destinations having paths to the original destination, resulting in a new query with an alternate set of destinations. The enhanced query however is not enhanced further, to avoid large querying latencies. We see that the initial query and the enhanced queries together are capable of resolving more than 90% of the queries even at high mobility, independent of the number of unidirectional links. Figure 8 shows the number of computed routes that physically exist after the route computation is over.

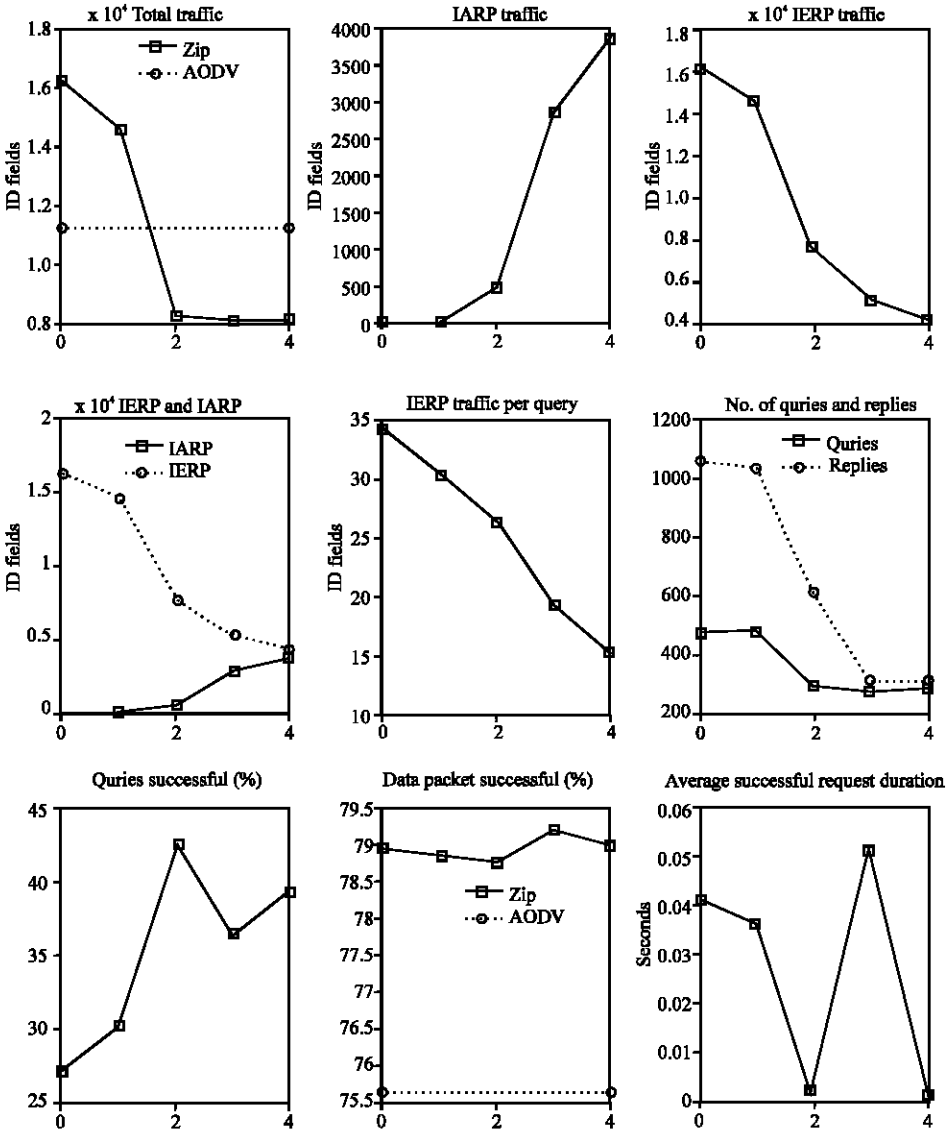

Fig. 8: Comparative of analysis of IARP Vs. IERP

We see that the accuracy of the computed routes decreases as the mobility increases; this is because the link-state information gets stale faster as the nodes move faster. Also, for a higher percentage of lower transmission range nodes, the computed route would typically consist of more number of hops and hence, has a higher chance of being invalid at the end of route computation. But even at 20 m s^{-1} in scenarios wherein 40% of the nodes have half the (125 m) transmission range of the other 60% of the nodes, the accuracy is close to 90%. Figure 8 shows that without the enhancement mechanism a lot of queries (up to 40%) would remain unresolved. For mobility from 20 to 30 m s^{-1}, close to 40% of the queries are resolved on account of the enhancement mechanism. When mobility is introduced we see a sudden increase in the number of enhanced queries. In such cases, the topology changes are not learnt quickly resulting in the failure of IARP/IERP mechanisms for a fresh query. However, the enhancement mechanisms are able to help and suggest alternate destinations, which could instead be queried for. This significant contribution from the enhancement mechanism is able to keep the total number of queries resolved (Fig. 8) above 90%, even at high mobility. Percentage of queries resolved by enhancement. Moreover we have evaluated performances of common ad-hoc network routing protocols: DSDV, DSR, AODV and TORA using a detailed packet-level simulator NS. We have experimented with the bi-directional traffic scheme, TCP traffic scheme, measured the power consumption and the network load distribution of these protocols. We simulated each protocol for ad-hoc networks with 50 nodes traveling within 1175x750 m_ geography. Results have shown that quick management of route maintenance is an important factor that effects all the performance measures, especially the successful delivery rate at high workloads and increased speeds. All evaluated protocols created bi-directional traffic on a single route, which caused a dramatic performance loss for all performance measures. None of the evaluated protocols were power aware, however AODV was power efficient for the bi-directional traffic scheme. In present simulations, DSDV had the least processing for route maintenance and the highest throughput and TORA had the lowest throughput. TORA generated a large amount of control messages to manage DAG's and its control messages encapsulated in IP were being dropped because of collisions, which led too much more decrease in performance. AODV used much less control messages, limited to the hosts involved in the routing process, therefore it has the highest standard deviation, which means that it distributes the load over the network in the least efficient way. Because of the same reason, AODV also had the least power consumption for the lowest workload. For very high speed networks, AODV performed to be the best in successful packet delivery and throughput. The successful packet delivery rate of DSR, which is a source routing protocol, is directly related to the generation of control messages, therefore to the frequency of data packet transmissions. TCP traffic led to multi path transmissions, which is caused by TCP retransmissions through multiple mobile hosts, therefore more throughput has been achieved and more reliable and better packet transmission has been unwittingly done. TCP causes more power consumption with increased load and network speed. For protocols having large update periods, we see that the power consumption is less evenly distributed, because of less frequent updates of routes. As a result, a new adaptive protocol for increased mobility and a special transmission control protocol is needed. AODV is the best protocol for high-speed networks and DSDV 90 performs to be the best for low speed networks. Protocols such as AODV and TORA, instead of trying to detect the link state and discovering neighbors at IP level, should use the link layer 802.11, because frequent control messages cause collisions and losses more with increased mobility or workload.

CONCLUSIONS

This study includes the comparative analysis of ZRP with the other routing protocols, which was not present in the analysis Given by Dr. Sinha, The comparative analysis shows that with the increase in the mobility the ZRP performs well as compared to the other. The intra zone and inter zone routing protocols have been modified to work for unidirectional links. For unidirectional links with large (larger than ZONE_RADIUS) inclusive cycles, a mechanism for recursive enhancement of the query has been proposed. The nodes that do not know of the destination but know of alternate nodes that have paths to the destination are reported back to the source. If the query is unresolved the source then issues an enhanced query that computes route for one of the alternate destinations. A heuristic has also been proposed to solicit enhancement messages from nodes when all the previous mechanisms fail to compute routes due to unidirectional links with large inclusive cycles. The protocol has been implemented and studied using the ns-2 simulator. The results show that even in the presence of a large number of unidirectional links and high mobility of 10 m s^{-1}, about 90% queries are resolved with a very high accuracy (correctness of route) of 90%. The proposed enhancement mechanism by itself is responsible for route computation in 40% of the cases and the routes computed due to this

enhancement are valid 80% of the time, in scenarios with mobility of 10 m s^{-1} and 40% of nodes having low transmission ranges. Thus, we have proposed and studied the performance of an extended Zone Routing Protocol for functioning in networks with unidirectional links.

REFERENCES

1. Sinha, P. and S.V. Krishnamurti, 1997. Scalability of unidirectional zone routing protocol extension of mobile ad-hoc networks. www.cs.ucr.edu/~krish/prasun_wcn.pdf, pp: 1-10.
2. Broch, J., D. Maltz, D. Johnson, Y. Hu and J. Jetcheva, 1998. A performance comparison of multihop wireless ad hoc network routing protocols. Proc. MOBICOM'98, Dallas, TX., pp: 12-15.
3. Park V.D. and S. Corson, 1998. A performance comparison of the temporally-ordered routing algorithm and ideal link-state routing. Proc. IEEE Symp. Comp. Commu., ISCC'98, Athens, Greece, pp: 3.
4. Perkins, C.E., E.M. Belding-Royer and S.R Das, 1997. Ad hoc on demand distance vector (AODV) routing. www.cs.edu/courses/cs295-1/9odv.pdf, pp: 1-5.
5. Johnson, D.B, D.A. Maltz and J.G. Jetcheva, 1999. The dynamic source routing protocol for mobile ad hoc networks. sherry.ifi.unizh.ch/context/345581/0, pp: 3-6.
6. Clausen, T., P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum and L. Viennot, 1998. Optimized Link State Routing Protocol. IETF Internet Draft, pp: 1.
7. Gerla, M., X. Hong and G. Pei, 1998. Landmark routing protocol (LANMAR) for large scale ad hoc networks, IETF internet draft, http://www.ietf.org/ internet-drafts/draft-ietf-manet-lanmar-05.txt, pp: 1.
8. Perkins, C.E. and T.J. Watson, 1997. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers ACM SIGCOMM, 1997-1998. www.cs.virginia.edu/~c17v/cs851-papers/dsdv-sigcomm94.pdf, pp: 2-6.