Design and Implementation of an OPNET model for simulating GeoAODV MANET routing protocol

Vasil Hnatyshin*and Hristo Asenov+

*Department of Computer Science
Rowan University
Glassboro, NJ 08028
E-mail: hnatyshin@rowan.edu

+Department of Computer and Information Science
University of Delaware
Newark, DE 19716
E-mail: hasenov@udel.edu

Abstract

Ad-Hoc On-Demand Distance Vector (AODV) is a routing protocol for mobile ad hoc networks (MANET). AODV floods the network with control messages to discover a route to the destination, which often results in a large number of control packets traveling through the network. This paper introduces the design and implementation of GeoAODV routing protocol for OPNET Modeler. GeoAODV takes advantage of Global Positioning System (GPS) coordinates to only search the area where a path to the destination is likely to be located. This paper discusses the authors' endeavors to develop a model of GeoAODV protocol and seamlessly integrate it within OPNET Modeler. In addition the paper provides an overview of OPNET Modeler's software architecture for modeling wireless networks and MANET routing protocols.

1. Introduction

This paper describes the design and implementation of the Geographical AODV (GeoAODV) routing protocol using the OPNET Modeler network simulating software. GeoAODV is a new protocol for simple and efficient location-based routing in mobile ad hoc networks (MANET). GeoAODV uses Global Positioning System (GPS) coordinates to improve performance of the route discovery phase of the Ad-Hoc On-Demand Distance Vector (AODV) routing protocol [7, 8]. GeoAODV is a stateless reactive protocol, i.e. it does not maintain state information, it relies only on local timers, and it initiates the route discovery process only when a route to the destination is required. However, unlike AODV, GeoAODV does not flood the network with control messages to discover the route to the destination. Instead, GeoAODV uses the destination's last known GPS coordinates to estimate the probable location of the destination node and then limits the route discovery flooding area to a region that is likely to contain a path to the destination. The idea of using node coordinates is not new and has been examined in the literature before [2-4, 6, 9].

OPNET Modeler [5] is a popular network simulation software package that provides a flexible and accurate platform for evaluating the performance of a variety of networking technologies. OPNET Modeler gives access to a rich set of API calls and process models usually implemented in the programming language called Proto-C which is a combination of the C/C++ programming language and state transition diagrams. These features of OPNET software make almost any aspect of simulated networking technologies modifiable and allow for the development and study of new communication devices, protocols, and networking paradigms [5]. We used OPNET Modeler version 14.5 to study GeoAODV protocol and to compare its performance with that of a regular AODV routing protocol. In particular, we extended the existing implementation of the AODV protocol to accommodate new GeoAODV features and compared the performance of the route discovery phases of AODV and GeoAODV protocols. The goal of this paper is to introduce a new MANET routing protocol and provide the details of how such a protocol can be implemented using OPNET software.

The rest of the paper is organized as follows. Sections 2 and 3 provide a brief overview of the AODV routing protocol and introduce the idea of GeoAODV protocol, respectively. An overview of the OPNET architecture is given in Section 4, followed by an overview of the OPNET implementation of AODV protocol in Section 5. Design and implementation of GeoAODV protocol are presented in Section 6. Discussion and plans for future work are described in Section 7. The paper concludes in Section 8.

2. Overview of AODV protocol

In this section we give a brief overview of the AODV protocol [7-8]. As its name implies, the Ad-hoc On-demand Distance Vector (AODV) routing protocol operates in the on-demand fashion, that is, it attempts to discover a path to the destination only when the source has data to send but does not have a route to the destination. AODV consists of two primary phases: route discovery and route maintenance. In this paper we are primarily interested in the route discovery phase which relies on a flooding technique to locate a path to the destination. The route maintenance phase is responsible for removing outdated or broken path entries from the routing table and is of no interest to our study.

AODV initiates the route discovery phase by having the source or the originator node broadcast a Route Request (RREQ) message through the network. The RREQ message is re-broadcast by each intermediate node until it reaches either the destination node or a node with a fresh route to the destination. In such a case the node generates a Route Reply (RREP) message back to the originator. The route discovery phase terminates when an RREP message that contains a route to the destination arrives at the originator node. As the RREP traverses the network back to the originator node it retraces the path of the RREQ message, which was recorded as the RREQ message was traveling through the network. Similarly, intermediate nodes that receive an RREP message update their routing tables with the route to the destination node. Once the route discovery phase completes the originator node sends data to its destination over the newly discovered path.

AODV also employs an expanding ring search technique which works as follows: the originator node sets the TTL field in the IP header of the RREQ message to a certain initial value. If the

route discovery process fails to find a path to the destination then the originator node increments the value of the TTL field and repeats the process again. The process continues until either the originator node finds a path to the destination or the whole network has been searched and the path was not found; i.e. an RREQ message with IP TTL field set to **ttl_threshold** value was sent out but a route to the destination was not found. Such an expanding ring search technique prevents unnecessary network-wide dissemination of RREQs.

Despite the expanding ring search technique the route discovery process in AODV often results in an unnecessarily large number of control packets traveling through the network and consuming such already scarce network resources as bandwidth, processing power, and battery power. GeoAODV attempts to mitigate this problem by reducing the scope of the control message broadcast during the route discovery phase.

3. GeoAODV routing protocol

To reduce the search region during the route discovery phase GeoAODV keeps track of and distributes known node GPS coordinates in the network. Generally, a GPS position is defined by a vector [x, y, z, t], where x, y, and z represent the coordinates in three-dimensional space and t represents the time. For simplicity, we assume that the z coordinate is always 0 (i.e. all the nodes are located on the surface of the earth) while the time coordinate t is maintained separately. Thus, GeoAODV protocol keeps track of and distributes the last known node position in the form of x and y coordinates, only.

Similar to AODV, in GeoAODV the originator node initiates the route discovery phase when the path to destination is unknown. However, the GeoAODV RREQ message carries additional information which includes the originator and destination node coordinates and the flooding angle. The flooding angle specifies the search area within which the route discovery procedure will take place. Specifically, if an intermediate node that receives an RREQ message belongs to the search area specified by the flooding angle then it forwards the RREQ message further, otherwise the message is discarded, thus limiting the scope of the route discovery process.

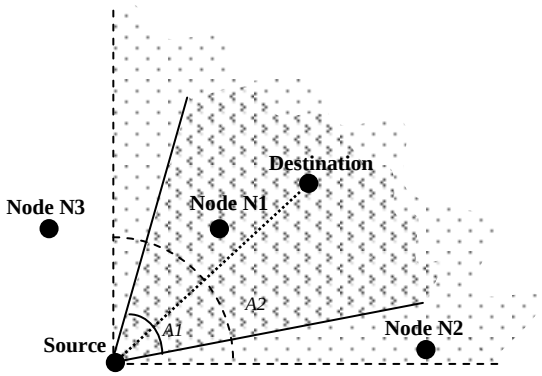

Figure 1. Example of GeoAODV operation

Generally, the flooding angle is computed based on the freshness of the last known destination position. If the destination coordinates are unknown then the flooding angle is set to a maximum value, indicating that the search region is the whole network. GeoAODV starts the route discovery process by sending an RREQ message with the initial flooding angle value. If the route discovery process does not succeed (i.e. no RREP arrives prior to route request timer expiration) then the originator node increases the value of the flooding angle and repeats the process again. Eventually, the node will either find a route to the destination or the value of the flooding angle will reach 360 degrees and GeoAODV will behave as AODV.

Figure 1 illustrates a basic idea of the route discovery process in GeoAODV. When *Source* generates an RREQ message with the flooding angle value *A1* in an attempt to discover a path to *Destination* only node *N2* will forward the RREQ because it is the only node within the search area defined by the flooding angle. If the first round of route discovery with the flooding angle *A1* fails to find a path to *Destination* then *Source* will generate another RREQ message with the flooding angle value *A2*, which is larger than angle *A1*. The RREQ message with flooding angle *A2* may arrive at nodes *N1*, *N2*, and *N3*, however only nodes *N1* and *N2* will forward the message because they are inside of the search region defined by angle *A2*. Node *N3* is outside the search regions defined by flooding angles *A1* and *A2* and thus it will discard RREQ messages during both rounds of route discovery.

4. Overview of OPNET architecture

The OPNET modeling architecture is structured in a layered fashion. The process domain is the lowest layer in the hierarchy and it contains the actual code of modeled protocols and technologies. Node domain is responsible for creating models of individual devices. It often relies on the process models to simulate the behavior of certain modules within modeled devices. Network domain is responsible for representing a complete system of interconnected devices as a simulation model.

Within the process modeling domain the developer implements the behavior of various processes, such as upper layer applications, routing protocols, IP interfaces, etc. In OPNET, the modular implementations of these processes are often referred to as process models. The complete specification of an OPNET process model consists of a finite state machine, action statements expressed in C/C++, and configurable parameters. In addition, process models often rely on external files that contain C/C++ implementations of various additional features.

Within the node modeling domain the developer implements the behavior of various network devices, such as end nodes, hubs, switches, routers, etc. Node models are usually defined via one or more functional elements called modules. The behavior of individual modules is specified either via a set of built-in parameters or through one or more process models.

Network system models that include individual nodes and interconnecting communication links are developed within the network domain. Such network system models also include the configuration of various additional features such as traffic generation sources, application user profiles, network protocols, etc. Generally, system models are configured via attribute values that specify either local characteristics, applicable to individual devices, or global characteristics, applicable to multiple devices

in the network. The attribute values specified in the network domain layer propagate all the way down to the process models.

5. OPNET Implementation of AODV

IP protocol's process model ***ip_dispatch*** is part of every layer-three networking device. One of many responsibilities of this process is to identify and invoke the routing protocol that was configured at the network domain layer. All MANET routing protocols, except Optimized Link State Routing (OLSR), are also indirectly invoked via the ***ip_dispatch*** process model. If a simulation is configured to run one of the MANET routing protocols then ***ip_dispatch*** creates and invokes the ***manet_mgr*** process model. Similarly, ***manet_mgr*** parses attribute values specified at the network domain layer to identify the MANET routing protocol configured in the simulated network and then creates and invokes the process model that implements the routing protocol of interest.

OPNET's implementation of the AODV protocol is defined via several external C code files and a process model called ***aodv_rte***. As shown in Figure 2, the ***aodv_rte*** process model consists of two states: ***init*** and ***wait***. The ***init*** state is responsible for initialization of various data structures including state variables, local statistics, buffers, and protocol parameters. Once initialization is complete, the process model moves into the ***wait*** state which models the operation of the AODV protocol. The ***aodv_rte*** process idles in the ***wait*** state until packet arrival or expiry of a timer occurs. Based on the event type the process performs the necessary actions by calling one or more of its functions.

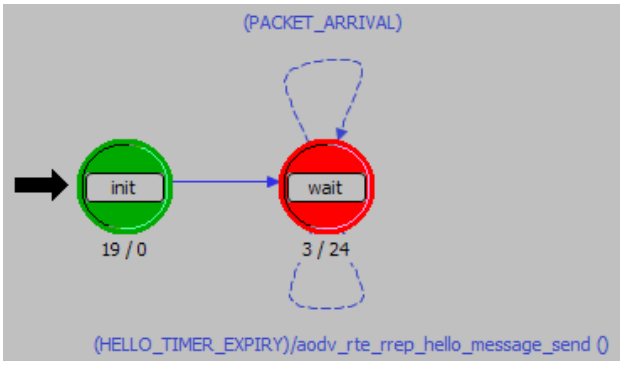

Figure 2. OPNET's aodv_rte process model

Packets that arrive to AODV process model are handled by the function ***aodv_rte_pkt_arrival_handle***. The packet type allows the process model to identify the function to be called next. If the packet came from the application layer, it is a data packet and it is processed by the ***aodv_rte_app_pkt_arrival_handle*** function. In this function the packet's header is examined against a routing table to determine whether there is a route to a destination. If there is no known route to destination then function ***aodv_rte_route_request_send*** is called to generated and send out a new RREQ message.

Control packets are handled separately from the data packets. To process an incoming RREQ packet the ***aodv_rte*** process model calls function ***aodv_rte_rreq_pkt_arrival_handle*** which is responsible for determining if the RREQ packet should be re-broadcast, discarded, or replied to with RREP. If the packet's source IP address matches the current node's IP address or if the packet have been seen before then the RREQ packet is discarded. Each AODV node maintains a request table which keeps track of all RREQ messages passed through this node and allows identifying RREQ duplicates. Therefore, if the RREQ packet is not discarded then its identity (i.e. source IP address and request number) is recorded in the request table and its hop count is incremented.

Next, the node attempts to update information about the route to the originator node. Notice that each RREQ packet carries IP addresses of the following nodes:

- Originator node IP address is stored in source address fields of AODV RREQ message header
- Destination node IP address is stored in destination address fields of AODV RREQ message header
- Previous node IP address is stored in source field of IP header. Destination field of IP header is set to broadcast address.

Routing table entry consists of the destination node IP address, destination node sequence number (i.e. this number is issued by the destination node and it determines the freshness of the route), and the next hop IP address. Therefore, if the routing table does not contain the route to originator node or if existing route is invalid then the node updates its routing table as follows:

- Destination node IP address is set to the originator IP address from RREQ header
- Destination node sequence number is set to originator sequence number from RREQ header
- Next hop address is set to source IP address from IP header (i.e. IP address of the previous node)

If the routing table already contains a entry for the originator node then the entry is updated only if RREQ message carries a better route (i.e. hop count is smaller) and if the route information is fresher (i.e. the originator sequence number value carried in RREQ message is greater than that of route entry). Otherwise, the routing entry remains unchanged. Notice, that the route to originator is needed when the RREP message that carries the path to destination travels back.

Once the routing table is updated the node checks if it has a valid and fresh (i.e. sequence number for destination node recorded in the routing entry is greater than the corresponding sequence number carried in RREQ) path to the destination node or if it is the destination node itself. If the node has a "valid" path to destination then it generates a RREP message and unicasts it back to the originator node using ***aodv_rte_route_reply_send*** function. Otherwise, the node sends RREQ message to IP layer for rebroadcast.

In addition, ***aodv_rte*** process model employs following external C code files:

- ***aodv_packet_queue*** – contains functions for managing hash table which buffers data packets.
- ***aodv_route_table*** – contains functions for managing AODV routing table.
- ***aodv_support*** – contains various supporting functions for updating collected statistic values and printing debugging information.

- ***aodv_pkt_support*** – contains function for creating AODV data and control packets and headers.
- ***aodv_request_table*** – contains functions for managing AODV's request table.
- ***manet_support*** – contains functions that provide an interface between MANET protocols and neighboring layers.

AODV control message headers are modeled as C structures defined in external header file called ***aodv_pkt_support.ex.h***. Specifically, the RREQ message header is defined in ***AodvT_Rreq*** structure, while RREP header is defined in ***AodvT_Rrep***. OPNET obtains the AODV message header from the arriving IP packet by extracting the option field structure of type ***AodvT_Packet_Option*** which consists of the following two values:

- ***type*** – an integer value that specifies message type
  - ***AODVC_ROUTE_REQUEST***
  - ***AODVC_ROUTE_REPLY***
  - ***AODVC_ROUTE_ERROR***
  - ***AODVC_RREP_ACK***
  - ***AODVC_HELLO***
- ***value_ptr*** – a pointer of type ***void**** that contains the location of the corresponding message structure.

Using the ***type*** field of structure ***AodvT_Packet_Option***, the ***aodv_rte*** process identifies the AODV message type, retrieves the data structure associated with the corresponding AODV message header, and finally calls the function to handle the corresponding AODV control message arrival.

The AODV process employs the routing table to keep track of valid routes to destination nodes. Data structures ***AodvT_Route_Table*** and ***AodvT_Route_Entry*** define the AODV routing table and routing table entry, respectively. AODV routing table is implemented as a hash table indexed by an IP address. AODV routing table is populated and updated via RREQ and RREP control messages. As expected, OPNET implements packet forwarding within the IP module which uses a common IP routing table. This routing table is updated and maintained by the routing protocol configured for the simulation study. Thus, AODV, and other routing protocols, in addition to maintaining their internal routing tables are also responsible for updating routing table at IP layer.

AODV process implements request table as a C structure called ***AodvT_Request_Table***. AODV's request table has a dual purpose:

(1) to keep track of RREQ messages generated from this node and
(2) record all RREQs forwarded by this node.

A list of the RREQ messages originated from the node allows implementing AODV's expending ring search technique in the route discovery process. This list is maintained in the form of a hash table, indexed by destination IP addresses. The entries into this hash table are defined by the ***AodvT_Orig_Request_Entry*** C structure which stores request id, insertion time, current TTL value, number of retries, and other information.

The record of RREQs forwarded by the node helps the AODV process to identify and discard duplicate RREQ messages. OPNET also maintains a list of RREQ messages forwarded by this node, but not generated by it, as a hash table. This hash table is indexed by the originator's IP address and contains such information as request id and insertion time.

Finally, the ***AodvT_Conn_Info*** structure implements the neighbor connectivity table which keeps track of the neighbor nodes, i.e. the nodes located one hop away from the current node. AODV relies on the periodic exchange of HELLO messages between neighboring nodes to maintain one hop routes. The neighbor connectivity table is also implemented as a hash table. It is indexed by the IP address of the neighboring node and stores the time when the last HELLO message was received. All the data structures that define AODV's routing, request, and neighbor connectivity tables are specified in the ***aodv.h*** header file.

6. Design and Implementation of GeoAODV

Instead of creating a new process model which would result in a significant amount of duplicate code, we implemented GeoAODV by extending the ***aodv_rte*** process model. Specifically, we introduced the following changes:

(1) Added new protocol configuration parameters
(2) Modified the AODV control packet headers
(3) Modified AODV control packet managing routines
(4) Implemented GeoAODV protocol functionality

Model attributes for MANET routing protocols are specified in the ***manet_mgr*** process model. However, the ***init*** state in ***aodv_rte*** and other MANET routing protocol process models are responsible for parsing and processing these attributes. For our GeoAODV implementation we added several model attributes including ***protocol type*** (i.e. AODV, GeoAODV, etc), and certain configuration parameters (e.g. initial flooding angle, etc). We defined new attributes in the ***manet_mgr*** process model and parsed their values in the ***aodv_rte_attributes_parse_buffers_create*** function of the ***aodv_rte*** process model.

We modified the ***AodvT_Rreq*** and ***AodvT_Rrep*** C structures defined in the ***aodv_pkt_support.h*** file, to specify additional data carried by GeoAODV control packets. ***AodvT_Rreq*** was modified to store originator and destination coordinates together with the flooding angle carried in RREQ message; while the ***AodvT_Rrep*** structure, which contains the format of AODV's RREP and HELLO messages, was changed to additionally store destination coordinates. In our implementation we define node coordinates as a pair of double precision floating point values and, for simplicity, we represent the flooding angle as an integer number, called ***request level***, that has the following meaning:

$$request\_level = \begin{cases} 0, & flooding\_angle = 90^{\circ} \\ 1, & flooding\_angle = 180^{\circ} \\ 2, & flooding\_angle = 270^{\circ} \\ 3, & flooding\_angle = 360^{\circ} \end{cases} \quad (1)$$

We added two new functions that create AODV control packets according to the modified definitions of the ***AodvT_Rreq*** and ***AodvT_Rrep*** C structures. We also updated the functions in the ***aodv_rte*** process model that handle arrival and parsing of the control packets.

We implemented GeoAODV protocol by adding the following functionality to the ***aodv_rte*** process model:

(1) Each node maintains an internal table, called ***geo-table***, that stores the coordinates of other nodes in the network
(2) The ***geo-table*** is updated with node coordinates carried in RREQ, RREP, and HELLO messages
(3) Decision about re-broadcasting arriving RREQ messages is made based on a new GeoAODV algorithm

The ***geo-table*** is maintained in a similar fashion to that of the AODV routing table. Each ***geo-table*** entry contains the following information about the destination of interest: IP address, sequence number, and location coordinates. The IP address uniquely identifies the destination node. The sequence number identifies the freshness of coordinates, the same way as in regular AODV. Location coordinates specify the last known location of the node. The ***geo-table*** is populated via RREQ, RREP, and HELLO messages which carry node coordinates. We added function calls for updating ***geo-table*** into the following functions:

- ***aodv_rte_rreq_pkt_arrival_handle***
- ***aodv_rte_rrep_pkt_arrival_handle***
- ***aodv_rte_rrep_hello_pkt_arrival_handle***

Implementation of GeoAODV functionality consists of two distinct parts: (a) initiating and managing the route discovery process at the originator node and (b) determining if the RREQ packet is to be rebroadcast based on the flooding angle value. The rest of GeoAODV protocol is very similar to AODV with a few minor exceptions that were described above (i.e. distributing node coordinates via GeoAODV control packets).

In GeoAODV, the originator node initially starts with a small flooding angle value which it increases if the previous round of route discovery failed to locate a route to the destination. Therefore, the originator node needs to keep track of the flooding angle used in the current round of route discovery. We modified C structure ***AodvT_Orig_Request_Entry*** by adding an integer field to record the value of the flooding angle used in the last RREQ message originated from this node.

1. If destination coordinates are unknown then return 3
2. Set the value of the request level to 0
3. Repeat while request level < 3
   a. If there is at least one neighboring node within the search area defined by the current value of request level then return request level
   b. Increment request level by 1
4. Return request level

Figure 3. Computing initial value of request level

Always starting the route discovery process with the smallest value of the flooding angle may lead to unnecessary delays if there are no neighbouring nodes within the area defined by the flooding angle. Therefore, the originator node computes the initial value of the flooding angle by selecting a value so that there will be at least one neighbouring node within the corresponding search area. Also recall that if the originator node does not have any coordinates for the destination node then it sets the flooding angle to 360° and operates as in regular AODV. Figure 3 describes the general algorithm used by the originator node to compute the initial value of the request level that represents the flooding angle.

Once the initial value of the flooding angle is computed the originator node initiates the route discovery process by generating an RREQ message and sending it to the IP layer for broadcast. We updated the function ***aodv_rte_route_request_send*** defined in the ***aodv_rte*** process model to send an RREQ message based on modified definitions of the ***AodvT_Rreq*** structure. Summary of the algorithm used at the GeoAODV originator node upon the start of the route discovery process is described in Figure 4.

1. Retrieve current node's location coordinates
2. If the destination's coordinates are available in geo-table then
   i. Retrieve destination coordinates from geo-table
   ii. Compute the initial value of the request level
3. Otherwise (destination coordinates are unknown)
   i. Set the request level to 3 (e.g. regular AODV)
   ii. Set destination coordinates to (-1, -1) (e.g. unavailable)
4. Generate RREQ message using obtained request level and source and destination coordinates
5. Send generated RREQ message
6. Update Request Table (i.e. record RREQ message information)

Figure 4. Start of GeoAODV Route Discovery Process

Upon RREQ arrival, in addition to regular AODV validation procedures, an intermediate node calls GeoAODV functions to determine if the RREQ should be re-broadcast or not; specifically, GeoAODV uses the originator and destination coordinates and the flooding angle carried in the RREQ message to determine if it is located within the route discovery search area. Only those RREQs that satisfy both validation conditions are re-broadcast farther.

Specifically, an intermediate node examines the angle θ formed between the originator-destination and originator-intermediate node vectors. If the angle θ is within half of the flooding angle then the intermediate node is located within the search area, otherwise it is not and the RREQ should be discarded. Angle θ is computed as follows:

$$\theta = \cos^{-1}\left(\frac{\overrightarrow{SD} \bullet \overrightarrow{SN}}{|SD| \times |SN|}\right) \qquad (2)$$

Where,

- $\overrightarrow{SD}$ is a vector between originator and destination,
- $\overrightarrow{SN}$ is a vector between originator and intermediate node *N*,
- $|SD|$ and $|SN|$ are absolute values of vectors $\overrightarrow{SD}$ and $\overrightarrow{SN}$, respectively.

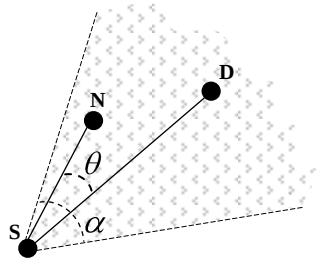

Figure 5. Example of GeoAODV Search Area

Figure 5 illustrates a situation where intermediate node *N* rebroadcasts the RREQ message because the angle θ formed by the originator-destination and originator-intermediate node vectors is within half of flooding angle α. Generally, the originator-destination vector always divides the flooding angle evenly. Thus, an intermediate node belongs to the search area if it is located on either side of the originator-destination vector and its angle θ is not larger than half of the flooding angle, or is located directly on the originator-destination vector.

In addition to determining if the RREQ message is to be re-broadcast, an intermediate node also updates its ***geo-table*** and Request Table and checks if it has a route to the destination. Summary of GeoAODV algorithm for processing RREQ message at an intermediate node is presented in Figure 6.

1. Update geo-table with originator node coordinates and destination node coordinates if fresher
2. If the node is destination or the node knows route to destination
   a. Generate RREP
   b. Return from function
3. If AODV determines that RREQ message is to be rebroadcast and if the node is within the search area then
   a. Rebroadcast RREQ
   b. Update Request Table
   c. Return from function
4. Drop RREQ message

Figure 6. Processing of RREQ at intermediate node

The RREP message is generated upon RREQ message arrival at the destination node or an intermediate node that knows a path to the destination. We modified the function **aodv_rte_route_reply_send** defined in the ***aodv_rte*** process model to send the RREP message based on the modified definitions of ***AodvT_Rrep*** structure as well. The processing of the RREP message at the intermediate nodes is almost identical to that in regular AODV, with the exception that an intermediate node also updates its ***geo-table*** with the most up-to-date value of destination coordinates.

Once the RREP message arrives at the originator node the route discovery process completes and the originator node transmits all accumulated application layer packets. However, if the RREP message never arrives at the originator node then the route request timer expires and the route discovery process is repeated with a larger value for the flooding angle. The process is repeated until either a route to the destination is found or the route discovery process uses a flooding angle value of 360 degrees (i.e. regular broadcast) and fails to find a route to the destination. Figure 7 shows the summary of the overall route discovery process.

Finally, GeoAODV also takes advantage of HELLO messages to distribute node coordinates in the network. We modified the function ***aodv_rte_route_table_entry_from_hello_update*** to update ***geo-table*** with the coordinates of the neighboring nodes carried in the HELLO message. Since HELLO messages use the same format as RREPs the code changes were minimal.

To provide clear separation between the original implementation of AODV and the changes we have introduced, all algorithms for managing ***geo-table*** and limiting the route discovery search area were placed into external files. Note that all external files used by the process model must be explicitly declared via the *Declare External Files...* drop-down option of the Process Model Editor.

1. Done = false
2. While (!Done)
   a. Compute new flooding angle
   b. Update Request Table
   c. Send RREQ
   d. Start Route Request Timer
   e. Wait until RREP arrival or Route Request Timer expiry
   f. If RREP arrives then Done = true
   g. If Route Request Timer expires
      a. If (flooding angle == 360) then Done = true
      b. Else Done = false
3. Send accumulated application packets

Figure 7. Summary of route Discovery Process

We validated the correctness of our model by performing unit testing of the newly-added functions and conducting several simple simulation studies (i.e. small size networks) of GeoAODV protocol. In each study we carefully traced the execution of the GeoAODV model using the OPNET debugger and DES logs and verified correctness of intermediate and final results.

7. Discussion and Future work

We performed a simple simulation study to verify the advantages of GeoAODV protocol. The primary goal of this study was to compare the number of control messages generated by the AODV and GeoAODV protocols during the route discovery process. Our study examined two sets of scenarios: one with stationary nodes and another with mobile nodes. In each set of scenarios we used a network topology that consists of 50 MANET nodes randomly placed within a 1000 meters by 1000 meters area.

In addition, each scenario used a different number of communicating nodes and the communicating node pairs were selected randomly. In mobile scenarios the nodes were moving according to the random waypoint model without pause between each step and the average node speed was uniformly distributed between 1 and 10 meters/second. We examined scenarios with 5, 10, 20, and 30 communicating nodes. Table 1 presents the summary of the main configuration parameter values. The attributes marked with an asterisk (*) had their value computed using a probability distribution function with the function name and input parameter values specified in the value column.

Configuration Parameter	Value
Channel Data Rate	11 Mbps
Transmit Power	0.005 Watts
Packet Reception Power Threshold	-95 dBm
Start of data transmission*	*normal* (100, 5) seconds
End of data transmission	End of simulation
Packet inter-arrival time*	*exponential* (1) second
Packet size*	*exponential* (1024) bytes
Duration of Simulation	300 seconds

Table 1. Summary of Route Discovery Process

Simulation results confirmed our hypothesis that GeoAODV improves the performance of the route discovery process of AODV protocol and reduces the number of control packets that traverse the network. Summary of simulation results is presented in Figure 8. Refer to [1] for a detailed description of GeoAODV simulation study.

Even though the simulation results are promising we believe that the GeoAODV protocol has room for improvement. Currently, we are investigating several variations to GeoAODV which will limit the search area during the route discovery process even further. Specifically we are considering the following optimizations of GeoAODV protocol:

(a) Modify the processing at the intermediate node so that it computes the search area formed in respect to previous–destination node line instead of originator–destination line. Such a modification is easy to implement and would re-orient the search area in the direction of the destination at each individual node. This modification might result in fewer control messages re-broadcast through the network.

(b) Add provisions to ensure that the route discovery does not extend too far beyond the location of the destination node. A possible way to implement such a provision is to check that the distance between the current node and destination is not larger than the distance between the previous node and the destination [4].

(c) Modify GeoAODV route discovery process so that the flooding angle dynamically increases at the intermediate nodes if there are no immediate neighbors within the search area defined by the flooding angle specified in the arriving RREQ message. Such an approach may reduce the number of unnecessary route discovery attempts with a small value of the flooding angle.

(d) Combine multiple ideas presented above to maximize the overall improvement. One such hybrid approach could be to use only two values of flooding angle, 180 and 360, and always compute the search area based on the location of the previous node. Such an approach will reduce the number of GeoAODV route discovery attempts to at most two which could be advantageous in sparsely populated MANETs.

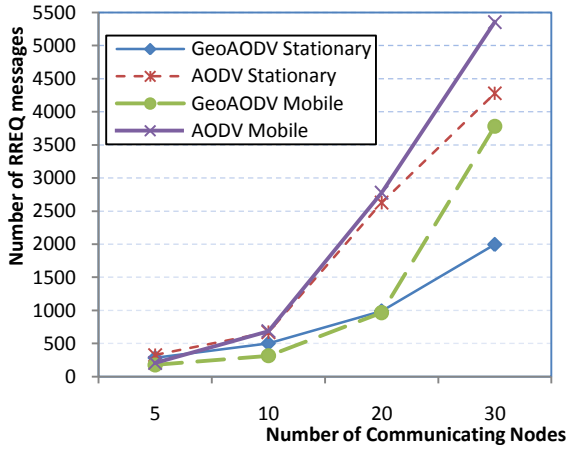

Figure 8. Summary of simulation results

In addition, we are currently in the process of implementing other approaches [2-4, 6] that employ GPS coordinates to improve the route discovery process of AODV protocol. We plan to conduct a comprehensive comparative study of such protocols. Finally, we plan to examine the effectiveness of GeoAODV protocol in distributing node location coordinates throughout the network, which is one of the side-effects of this protocol.

8. Conclusions

This paper presents the design and implementation of GeoAODV, a new protocol for improving the performance of AODV route discovery process. GeoAODV takes advantage of the GPS system and uses node coordinates to reduce the search area during the route discovery process. Specifically, this paper describes OPNET architecture for modeling MANET routing protocols and AODV protocol in particular as well as provides a detailed account of the authors' endeavours to implement GeoAODV routing protocol by extending AODV process model using OPNET Modeler version 14.5. In addition, this paper provides a summary of simulation results which show that with the help of GPS coordinates, GeoAODV routing protocol reduces the control packet overhead during the route discovery process. Currently, we are investigating new ideas for improving performance of GeoAODV protocol and we plan to implement other location-enabled routing schemes and compare their performance with that of GeoAODV. Finally, we believe that this paper could greatly benefit other researchers who use OPNET tools to study and evaluate the performance of wireless mobile ad hoc networks.

References

[1] H. Asenov and V. Hnatyshin, "GPS-Enhanced AODV routing", Proc. 2009 International Conference on Wireless Networks (ICWN'09), Las Vegas, NV, 2009.

[2] D. Espes, Z. Mammeri, "Adaptive expanding search methods to improve AODV Protocol," IST Mobile and Wireless Communications Summit, July 2005.

[3] Y. Ko and N. H. Vaidya, "Flooding-based geocasting protocols for mobile ad hoc networks," Mobile Networks and Applications, 7(6), Dec. 2002, pp. 471-480.

[4] Y. Ko and N. H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," Wireless Networks, 6(4), July 2000, pp. 307-321.

[5] OPNET Modeler ver. 14.5. OPNET Technologies, Inc®, www.opnet.com last visited 6/5/10.

[6] G. Pei, M. Gerla, and T.-W. Chen, "Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks" in Proc. of the IEEE International Conference on Communication, New Orleans, LA, June 2000.

[7] E. M. Royer and C. E. Perkins, "An Implementation Study of the AODV Routing Protocol," Proc. of the IEEE Wireless Communications and Networking Conference, Chicago, IL, September 2000.

[8] C. E. Perkins and E. M. Royer, "Ad hoc On-Demand Distance Vector Routing," Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, Feb. 1999, pp. 90-100.

[9] B. Zhou, Y. Lee, M. Gerla, and F. de Rango, "Geo-LANMAR: a scalable routing protocol for ad hoc networks with group motion: Research Articles", Wireless Communications & Mobile Computing, 6(7), Nov. 2006, pp. 989-1002.