PRACTICAL METHODOLOGY FOR MODELING WIRELESS ROUTING PROTOCOLS USING OPNET MODELER

Vasil Hnatyshin, Hristo Asenov, and John Robinson
Department of Computer Science
Rowan University
201 Mullica Hill Rd.
Glassboro, NJ 08062
USA
hnatyshin@rowan.edu, hristo.s.asenov@gmail.com, robinsonj@rowan.edu

ABSTRACT

OPNET Modeler is one of the most popular commercial products for simulating and modeling of computer networks and related technologies. While creating a new simulation study using standard models is a fairly straight-forward task, developing new models or modifying existing ones could become a challenging and often frustrating undertaking. This paper provides an overview of OPNET Modeler's software architecture for modeling wireless networks and MANET routing protocols. In particular, this paper concentrates on the modeling and simulation portion of the research project that studies improvement of Ad-Hoc On-Demand Distance Vector (AODV) routing protocol through the use of GPS coordinates. Using AODV modifications as an example, this paper introduces practical methodology for changing existing simulation models of MANET routing protocols and seamlessly integrating them within OPNET Modeler. In addition this paper introduces GeoAODV protocol which reduces the route discovery overhead through the use of GPS coordinates.

KEYWORDS

Modeling, simulation, MANET routing, AODV, OPNET Modeler

1. Introduction

Using modeling and simulation methodologies in the performance evaluation of computer and communication systems is one of the most active fields of research. While measurement- and formula-based approaches are still frequently used to study system performance, they have numerous disadvantages which often force researchers to choose the simulation approach. The measurement of system performance using hardware prototypes is a very accurate approach but it is also inflexible, expensive, time-consuming, and not appropriate for large systems. On the other hand, a formula-based approach, which is often based on simplified models, may provide insight into system operation, but it is not very accurate and is difficult to use for evaluation of the complex systems. A simulation-based approach allows for system modeling with some defined level of detail. With a simulation-based approach one can easily integrate mathematical and empirical models, using a formula-based approach for the portions of simulation where the accuracy is not of the highest importance, and incorporating measured device characteristics to provide precision to vital portions of the simulation model [1].

OPNET Modeler [2] is the leading commercial software for simulation and modeling of computer networks and related technologies. OPNET software contains a wide range of simulation models of various computer devices, communication mediums, and network protocols and technologies. OPNET Modeler relies on combination of the C programming language and state transition diagrams to implement simulation models and supporting technologies. While creating a new simulation study using standard models is a fairly straight-forward task, developing new models or modifying existing ones could become a challenging and often frustrating undertaking. OPNET products model performance of simulated systems with the high degree of accuracy, which results in huge amounts of code. Despite extensive documentation and good naming conventions, it is often difficult to identify the process models, external files, and portions of code which are responsible for simulating specific aspects of the system's performance. That is why modifying and extending OPNET's simulation models can become a very challenging and time consuming task.

Routing and route discovery in mobile ad hoc networks (MANETs) are among the most actively researched issues in the area of wireless communication. Our research efforts concentrate on improving efficiency of the route discovery in MANETs through the use of the Global Positioning System (GPS). We developed, implemented, and tested a new protocol called GeoAODV which uses GPS coordinates to improve the route discovery process of AODV protocol by reducing the amount of control packets traveling throughout the network. Other approaches that study influence of GPS on MANET routing have been examined in [3 - 6].

This paper, however, focuses on the portion of our research project that deals with modeling of the routing protocols using OPNET Modeler software package. Using

GeoAODV as an example, this paper describes in detail the key steps for successful integration of desired changes with existing MANET routing protocol implementations in OPNET Modeler. Specifically, the main contributions of this paper are detailed description of OPNET modeling framework for MANET routing protocols, a practical methodology for introducing changes to existing OPNET Modeler protocol implementations and creating new protocol models, as well as account of the author's endeavors to develop and implement GeoAODV MANET routing protocol and integrate it within OPNET network simulation software. This work could be of great value to other researchers who use OPNET Modeler in their studies of wireless mobile ad hoc networks.

The rest of the paper is organized as follows. Section 2 provides a brief overview of MANET route discovery process and introduces GeoAODV protocol. The OPNET Modeler architecture, an overview of OPNET's framework for modeling MANET routing protocols, and a detailed description of AODV simulation model are introduced in Section 3. Implementation details of GeoAODV protocol are presented in Section 4, followed by a generalized methodology for development and deployment of MANET routing protocols in OPNET Modeler in Section 5. Simulation study of GeoAODV protocol and a summary of collected results are presented in Section 6. The paper concludes in Section 7.

2. MANET routing and GeoAODV

Route discovery in MANET often relies on some form of network flooding where each node in the network forwards a route request message to all of its neighbours. This process is inefficient because it results in control messages visiting the network nodes even if the path to destination is located in a different portion of the network. To improve network utilization of the flooding-based route discovery protocols, it has been proposed to employ node's geographical position and limit route-discovery process only to the area that is likely to contain the path to destination.

GeoAODV is based on Ad-Hoc On-Demand Distance Vector (AODV) routing protocol [7-9] and it tries to take advantage of GPS coordinates to reduce the overall number of control messages traversing the network during route discovery. Unlike AODV, GeoAODV does not flood the whole network with control messages to discover the route to destination. Instead, GeoAODV uses the last known GPS coordinates of destination node to limit the route discovery flooding area to a region that is likely to contain a path to destination.

When the path to destination is unknown, the originating node (i.e. source) initiates the route discovery phase by broadcasting a route request (RREQ) message. In GeoAODV, a node forwards RREQ messages only if it belongs to the forwarding area, a region which is likely to contain the route to destination. Specifically, if an intermediate node that receives an RREQ message belongs to the forwarding area then it forwards the RREQ message further, otherwise the message is discarded, thus limiting the scope of the route discovery process.

GeoAODV defines the forwarding area as an area formed by two lines originating from the source node and extending towards the destination node at a certain angle. We refer to the angle between these lines as *flooding angle*. Notice that the line formed by source and destination nodes divides the flooding angle evenly. A flooding angle is computed based on the freshness of the last known location of destination node. If the destination coordinates are unknown then the flooding angle is set to a maximum value (i.e. 360 degrees), indicating that the search region is the whole network. In such a situation, GeoAODV operates the same way as AODV.

GeoAODV starts the route discovery process by sending RREQ message with the initial flooding angle value. If the route discovery process does not succeed then the originating node increases the value of the flooding angle and repeats the route discovery process. Eventually, the route to destination is found or the value of the flooding angle reaches its maximum of 360 degrees in which case GeoAODV operates the same way as AODV. GeoAODV gives up the search if the route to destination is not found with the flooding angle value of 360 degrees.

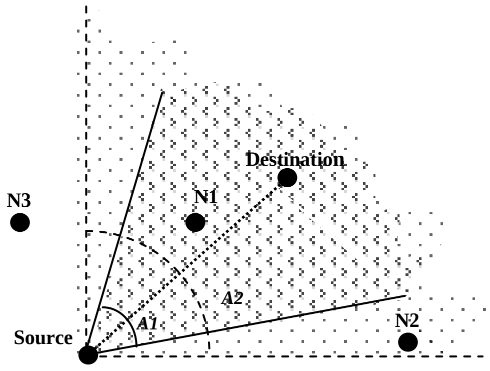

Figure 1. Example of GeoAODV operation

Figure 1 illustrates an example of GeoAODV route discovery process which starts when Source sends a RREQ message in an attempt to discover a path to Destination. A RREQ message carries the value of the flooding angle A1 which indicates that only node N1 will re-broadcast the initial route request. If the first route discovery attempt fails, then Source restarts the process again with a new larger value of flooding angle (i.e. A2). Since node N3 is outside of the forwarding area defined by flooding angle A2, only nodes N1 and N2 will re-broadcast RREQ message.

3. OPNET's modeling of MANET routing

3.1. Overview of OPNET Modeler architecture

Creating a simulation model of a communicating system could become a time-consuming and often error-prone task. OPNET Modeler provides a flexible and highly cohesive architecture which allows for reusability and extensibility of existing models. OPNET Modeler is structured in a hierarchical fashion and consists of three distinct layers: the network, the node, and the process domain levels.

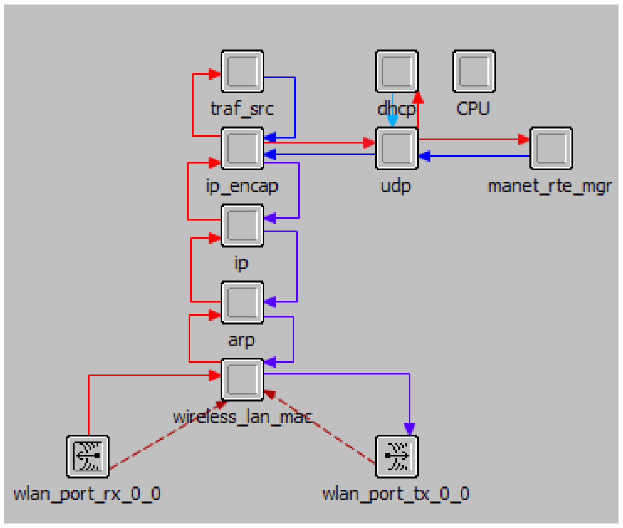

Figure 2. Node model of MANET station

The network topology of a modeled system together with the attribute values are specified at the network domain level which is the top-level view of the simulation study. The attribute values (i.e. protocol parameters, device configuration values, simulation statistics, etc.) that are specified at this level propagate down to the lower hierarchical levels (i.e. node and process). Various network devices, such as routers, servers, switches, etc are modeled at the node domain level. A network device model usually consists of one or more interconnected modules, each of which is defined either via one or more process models or a set of configuration values and associated external files. Figure 2 illustrates a node level model of a MANET station, where individual modules are depicted as gray squares and the arrows represent the flow of information between them.

A process domain level models operation of a particular networking process or technology, such as a routing protocol, an upper-layer application, load-balancing discipline, etc. Each process model consists of the finite state machine and a set of Proto-C instructions that specify conditions for transitioning from one state into another and a set of actions to be performed in each state. The process models often rely on external files which contain a set of supporting functions or data structures.

3.2. Modeling AODV in OPNET

The node model of MANET station contains two IP-related modules: **ip_encap** which models packet (de)encapsulation within the IP header and **ip** which simulates various IP operations. The latter module uses **ip_dispatch** as its root process model. One of the main responsibilities of **ip_dispatch** process model is to implement IP-level packet forwarding. During process initialization **ip_dispatch** identifies the routing protocol(s) employed at the interfaces of the current device and then creates and invokes the corresponding routing processes. Even though **ip_dispatch** is responsible for IP forwarding, it does not handle MANET routing protocols directly. If **ip_dispatch** discovers that an interface is configured to run MANET protocol then it invokes **manet_mgr** process model, which is responsible for identifying and then invoking a specific MANET routing protocol such as: AODV, DSR, GRP, or TORA[1].

OPNET implements AODV protocol [7-9] via **aodv_rte** process model and the following external 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.

As shown in Figure 3, the **aodv_rte** process mode consists of two states: ***init*** and ***wait***. The ***init*** state is responsible for initialization of various data structures including state variables, local statistics, buffers, protocol parameters, and others. Once initialization is complete, the process model moves into ***wait*** state which models operation of 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.

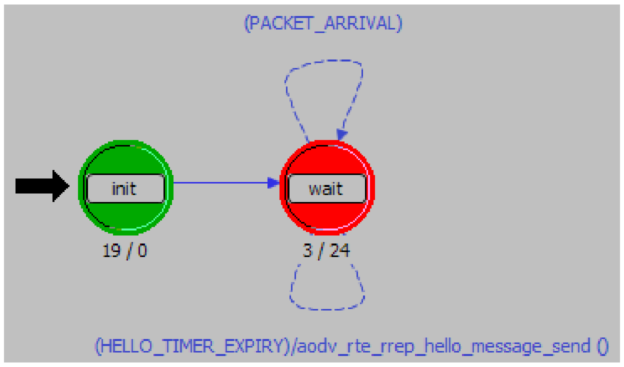

Figure 3. OPNET's **aodv_rte** process model

Function ***aodv_rte_pkt_arrival_handle***[2] is called upon a packet arrival into AODV process model. Based on the packet type, **aodv_rte** initiates the packet processing by

[1] Optimized Link State Routing (OLSR) protocol is handled via **manet_rte_mgr** process model.

[2] For readability, we omit **aodv_rte** prefix from function names defined in **aodv_rte** process model. We mark modified names by using *italics*.

calling their corresponding functions. Upon data packet arrival **aodv_rte** calls ***app_pkt_arrival_handle*** function which examines the packet's header and determines if the route to the packet's destination is known. If there is no route to destination, then the process model initiates route discovery by generating a new RREQ message. Function **aodv_pkt_support_rreq_option_create**, defined in **aodv_pkt_support** file, is responsible for creating RREQ packet header while function ***route_request_send*** in **aodv_rte** forwards created RREQ to IP layer.

Upon control packet arrival **aodv_rte** calls the function that handles the corresponding control packet. For example, function ***rreq_pkt_arrival_handle*** processes arrival of RREQ packets, while ***rrep_pkt_arrival_handle*** function deals with route reply (RREP) messages. Other key functions of **aodv_rte** process model include ***route_request_send*** and ***route_reply_send*** which are responsible for generating and forwarding RREQ and RREP messages.

OPNET provides set of standard functions which allow the processing of incoming packets. Specifically, **aodv_rte** extracts the options field from the incoming IP packet by calling function **op_pk_nfd_access**. This function call returns a pointer to **AodvT_Packet_Option** structure which consists of two fields: the packet type and a void pointer to the structure that contains AODV packet header. RREQ and RREP message headers are modeled via **AodvT_Rreq** and **AodvT_Rrep** C structures which are defined in an external header file **aodv_pkt_support.ex.h**.

External header file **aodv.ex.h** contains various supporting data types for maintaining such information as routing table, request table, connectivity table, AODV statistics, and others. Specifically, AODV routing table is defined via C structure called **AodvT_Route_Table**, while the routing table entry is defined via **AodvT_Route_Entry.** AODV routing table keeps track of valid routes to destination nodes. OPNET Modeler implements AODV routing table in a form of a hash table indexed by the destination's IP address. AODV routing table is populated and updated using route request and route reply messages. As expected, OPNET implements packet forwarding within the IP module which relies on the common IP routing table. The common IP routing table is updated by the routing protocols used in the simulation study. Thus, AODV and other routing protocols, in addition to maintaining their own internal routing tables, also update common IP routing table each time a new route is discovered.

The request table is implemented via C structure called **AodvT_Request_Table**, which keeps track of RREQ messages generated from and forwarded by this node. RREQs generated by this node (e.g. originating RREQs) are stored in a separate hash table indexed by the destination node's IP address. The entries of this hash table are defined in **AodvT_Orig_Request_Entry** C structure which stores such information as request id, insertion time, current TTL value, number of retries, etc. Originating RREQs allow the node to keep track of already initiated route discovery procedures. If a data packet arrives and the path to destination is unknown then the node consults the request table to determine if the route discovery procedure for packet's destination has been initiated already. Additionally, originating RREQs allow for implementation of AODV's expanding ring search technique [7-9].

Forwarded RREQs are defined as a separate hash table with the **AodvT_Request_Table** structure. This hash table is indexed by IP address of originating node and contains such information as RREQ request id, and insertion time. Forwarded RREQs helps the AODV process to identify and discard duplicate RREQ messages.

Connectivity table keeps track of the neighboring nodes, i.e. the nodes that are only one hop away from the current node. Connectivity table is populated with the help of AODV HELLO messages which are periodically exchanged among neighboring nodes. Connectivity table is implemented using **AodvT_Conn_Info** structure, which is also a hash table. Connectivity table is indexed by the IP address of a neighboring node and contains the last time HELLO message from the neighbor has been received.

4. GeoAODV implementation

Instead of creating a new process model which would result in significant amount of duplicate code, we implemented GeoAODV by extending **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

OPNET defines configuration parameters of MANET routing protocols as **Model Attributes** in **manet_mgr** process model. However, parameter parsing and processing is performed in the ***init*** state of the corresponding process model (i.e. **dsr_rte**, **aodv_rte**, etc). To implement GeoAODV protocol we added several model attributes including **protocol type**, an attribute that specifies version of AODV protocol configured on the node (i.e. AODV, GeoAODV, etc), and several GeoAODV configuration parameters (e.g. initial flooding angle, etc). All of the added protocol configuration parameters were specified as **Model Attributes** of **manet_mgr** process model. These attributes were parsed in the ***attributes_parse_buffers_create*** function of the **aodv_rte** process model.

Since GeoAODV control packets carry additional data, we modified **AodvT_Rreq** and **AodvT_Rrep** C structures, which represent RREQ and RREP packet headers respectively, to store such additional fields as flooding angle and coordinates of originating and destination nodes. To handle modified control packet headers we added two functions that create GeoAODV RREQ and

RREP messages. These functions were inserted into **aodv_pkt_support.ex.c**, an external C file that contains all functions related to AODV packet managing. In addition, we modified functions ***rreq_pkt_arrival_handle*** and ***rrep_pkt_arrival_handle***, in ***aodv_rte*** process model, to properly parse new packet headers.

GeoAODV protocol has been implemented by adding the following modifications:

(1) GeoAODV nodes maintain GPS coordinates of other nodes in the network in their internal tables
(2) GeoAODV nodes update their internal tables with location information carried in control messages
(3) Decision about re-broadcasting arriving RREQ messages is made based on GeoAODV algorithms

GeoAODV maintains a separate hash table which stores node locations and is indexed via the node's IP address. This table is called geo-table and is populated via RREQ and RREP messages which carry node coordinates. Upon RREQ arrival, in addition to regular AODV validation procedures, an intermediate node calls GeoAODV functions to determine if RREQ should be re-broadcast or not. Specifically, GeoAODV uses source 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.

To provide clear separation between the original implementation of AODV and introduced changes, 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 *Declare External Files...* drop-down option of the Process Model Editor.

Finally, we modified AODV's route request table to store the flooding angle value used in the last round of route discovery process. Using this information we modified the route discovery process in the originating node as follows. If the latest round of route discovery fails to find the route to destination then the value of the flooding angle is increased, expanding the route discovery search area, and the process is repeated again. This procedure continues until the route to destination is found or the route discovery process that uses a flooding angle value of 360 degrees (i.e. regular broadcast) fails to find the route to destination.

We validated correctness of our model by performing several simple simulation studies (i.e. small size networks) of GeoAODV protocol. In each study we carefully traced execution of GeoAODV model and verified correctness of intermediate and final results.

5. Generalized methodology

This section summarizes methodology for modifying existing implementation of MANET routing protocols or developing new models using OPNET Modeler network simulation tool.

Generally, to extend an existing OPNET model or to create a new one, the developer may need to perform the following steps:

- add configuration parameters, if needed,
- implement desired modification by changing the existing model or adding a new one,
- define and collect additional statistics as needed,
- declare all external files and child processes that have been added, and
- verify correctness of implementation.

In OPNET Modeler, MANET routing protocols are managed by **manet_rte_mgr** and **manet_mgr**. All introduced configuration parameters must be added as model attributes in one of the above process models. These model attributes are user configurable at the node and network domain levels. Any process can access model attributes by using standard OPNET function **op_ima_obj_attr_get**. Note that the developer needs to declare state variables (i.e. variables that are global to the whole process model) to store the values of model attributes. Each process model that implements MANET routing protocols (i.e. **olsr_rte, aodv_rte**, **dsr_rte**, **grp_rte**, **manet_tora**) contains code for parsing model attributes which could be used as an example.

Implementation of desired changes is problem specific and is different in each situation. However, developing a new MANET routing protocol requires creation of new process model(s). Integration of such new model(s) with the rest of MANET routing protocols requires the following changes in **manet_mgr**:

- modifying **manet_mgr_routing_protocol_determine** function to identify new routing protocol type
- modifying **manet_mgr_routing_process_create** function to create and invoke new process model
- declaring new process models as child process models of **manet_mgr**

Modification of existing routing protocols should require no changes to **manet_rte** or **manet_rte_mgr** process models besides declaration of additional model attributes. It should be noted, that the process model which uses added external files or process models must explicitly declare them via Process Editor's drop-down menu. Without such declarations modified process model will not compile.

Introduced modification often results in the need for collecting additional simulation statistics. Generally, adding new statistics is a four-step process. First, the developer needs to declare new statistics via drop-down menu in Process Editor. All declared statistics are configurable (i.e. can be selected for collection) at the network domain level. After that, the developer should define state variables for the declared statistics. These variables will accumulate simulation statistic values.

Next step is to register declared statistics, which is done by calling **op_stat_reg** function. Most of the process models combine all the statistic registration calls in a single function (i.e. **dsr_rte_stats_reg** in **dsr_rte** process model, **aodv_rte_local_stats_reg** in **aodv_rte**, etc). Finally, the statistic values should be periodically updated using **op_stat_write** function call, which stores the value and the time when the value has been recorded. Generally, simulation statistics are updated upon occurrence of certain events, such as packet arrival, packet departure, etc. The exact location where the call to **op_stat_write** function is placed depends on the nature of simulation statistics being collected.

When all desired changes have been implemented it is prudent to verify correctness of new simulation model. OPNET provides a powerful debugging tool which allows line-by-line examination of the code, breaking at any point of the simulation execution, examining in detail current state and call hierarchy of all active process models, tracing of packets passing through the network, etc. Once the model appears to work correctly it is a good idea to perform a small size simulation study and verify the validity of obtained results.

6. Simulation study and results

This section provides a summary of a simulation study that compares performance of route discovery processes of AODV and GeoAODV routing protocols.

6.1. Simulation Set-Up

MANET network, examined in our simulation study, consisted of 50 MANET nodes randomly placed within the 1000 x 1000 meters area. Our simulation study was divided into two scenario sets. In the first set of scenarios MANET network was populated with stationary nodes only. The second set of scenarios had all nodes moving according to the random waypoint model. The average node speed was uniformly distributed between 1 and 10 meters/second. The nodes did not pause between moves and continued their movement until the end of simulation.

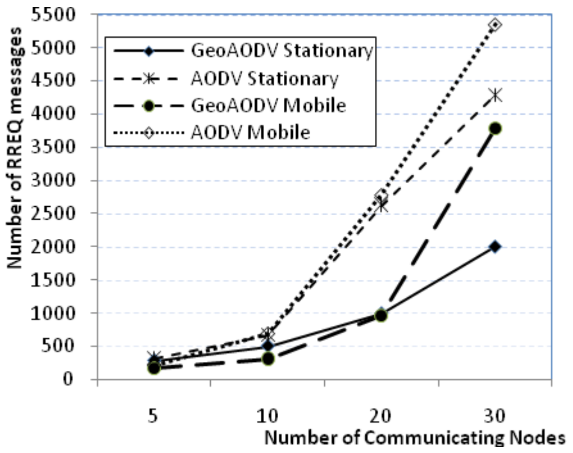

Figure 4. Summary of simulation results

Each scenario set was further divided into individual scenarios, each configured with a different number of communicating nodes. Each communicating node was selected randomly and was configured to transmit over 11 Mbps channel with transmit power of 0.005 Watts and received power threshold of -95dBm. Data transmission to a random destination node was initiated at time 100 seconds and continued until the end of simulation. The packet inter-arrival time was computed using exponential distribution with mean outcome of 1 second, while the packet size was computed using exponential distribution with mean outcome of 1024 bits. We ran each simulation scenario for 300 seconds.

6.2. Summary of collected results

Our simulation study confirmed that during the route discovery process GeoAODV generates fewer control messages than regular AODV. As expected, when the number of communicating nodes increased, the number of control messages generated by AODV and GeoAODV increased as well. However, even when there were 30 communicating nodes, GeoAODV outperformed AODV because it did not have to search the whole network. GeoAODV performed better than AODV in mobile scenarios as well. However the overall improvement was lower than in stationary node scenarios because node movement caused the destination coordinates stored throughout network to be less accurate which may have resulted in several rounds of GeoAODV route discovery. Refer to [6] for a detailed description of GeoAODV simulation study.

7. Conclusions

This paper introduced practical methodology for adding new process models and modifying existing implementations of MANET routing protocols using OPNET Modeler network simulation software package. Specifically, this paper describes AODV simulation model and provides detailed account of the author's endeavours to implement GeoAODV routing protocol by extending AODV process model. In addition, this paper provides an overview of GeoAODV protocol and summary of simulation results which show that with the help of GPS coordinates, GeoAODV routing protocol significantly reduces the control packet overhead during the route discovery process by limiting the size of the route discovery area. This paper could greatly benefit other researchers who use OPNET tools to study and evaluate performance of wireless mobile ad hoc networks.

References

[1] M. C. Jeruchim, P. Balaban, K. S. Shanmugan, *Simulation of communication systems: modeling, methodology, and techniques.* (Kluwer Academic Publishers, 2000)

[2] OPNET Modeler ver. 14.5. OPNET Technologies, Inc®, www.opnet.com last visited 2/09/10.

[3] D. Kadono, T. Izumi, F. Ooshita, An ant colony optimization routing based on robustness for ad hoc

networks with GPSs, *Ad Hoc Networks, 8*(1), January 2010, 63-76.

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

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

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

[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, February 1999.

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