

Analysis of TCP-Reno and TCP-Vegas over AOMDV Routing Protocol for Mobile Ad Hoc Network

Mazliza Othman*, May Zin Oo**

Faculty of Computer Science & Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia

mazliza@um.edu.my, mayzinoo.mz@gmail.com

Abstract— A mobile ad hoc network eliminates the complexity of infrastructure configuration and allows wireless devices to communicate with each other on the fly. It does not rely on a base station to coordinate the flow of messages in the network. A primary challenge is to provide each device to maintain the information to properly route traffics using a routing protocol and conveying data packets efficiently using a transport protocol. Many routing protocols and transport protocols are currently being developed to handle the wireless environment efficiently. In this paper, we investigate the performance differences of TCP-Reno and TCP-Vegas utilizing an AOMDV routing protocol. Although TCP-Vegas provides a better throughput than TCP-Reno in a static environment, it suffers performance degradation in a mobile environment, where packet loss rate, throughput, normalized routing load and average delay are measured as performance metrics, as well as discussed by describing strengths and weaknesses of TCP-Reno and TCP-Vegas.

Keywords— Include at least 5 keywords or phrases

I. INTRODUCTION

A MANET is a temporary network with two or more hops without using any pre-existing and fixed infrastructure. A number of intermediate nodes take responsibility to relay information from one node to another by acting as a router to discover and maintain route information within the network. For MANETs, the issue of routing packets between any pair of nodes becomes a challenging task because the nodes can move randomly within the network. A route that is believed to be optimal at a given point in time might not work at all a few moments later.

Traditional routing protocols [1, 13, 2] are proactive and they maintain routes to all nodes. They react to any change in the topology, even if no traffic is affected by the change, and they require periodic control messages to maintain routes to every node in the network. As mobility increases, more of scarce resources, such as bandwidth and power will be used. Alternative reactive routing protocols [3, 4] determine the route when they explicitly need to route packets, thus avoiding nodes from updating every possible route in the

network. The behavior of routing protocols depends on the network size, link capacity, connectivity patterns and node mobility. Among routing protocols, on-demand routing protocols are most popular due to its reactive nature that removes unnecessary routing packets and reduces routing overhead. Among on-demand routing protocols, Ad hoc On demand Distance Vector (AODV) is the most popular protocol and its many variations are being enhanced.

AODV has evolved into a carefully designed ad hoc routing protocol in a wide variety of network topologies and environments since its first version. However, the prior version of AODV only provides a single path between a source and destination. All routes are discovered as needed, and are maintained only as long as they are needed. It provides a dynamic, self-configuring, multi-hop routing between mobile nodes to establish and maintain an ad hoc network. AODV was designed to avoid the following issues.

- Huge amount of control overheads
- Huge amount of processing overheads
- Formation of loops

AODV attempts to minimize the control overhead by utilizing only on demand messaging instead of sending route updates periodically. As a consequence, AODV messages are simple to compute and reduce processing overheads. By utilizing destination sequence number, AODV severely prevent the formation of routing loops. [10] extends the prominence of AODV, called Ad Hoc On-demand Multipath Distance Vector (AOMDV) to discover multi-path between the source and destination to provide efficient fault tolerance in the sense of faster and efficient recovery from route failures in dynamic network. By computing multiple paths in a single route discovery attempt, new route discovery is needed only when all paths fail. This reduces not only the route discovery latency but also the routing overheads.

In fact, Transmission Control Protocol (TCP) [6] was designed to provide a reliable end-to-end delivery of data packets in the wired networks. To facilitate TCP in a MANET environment, many problems such as high bit errors, node mobility, and packet losses due to congestion are encountered. TCP provides reliability, flow control, congestion avoidance,

fairness, and in-sequence delivery. Originally, the protocol did not have congestion avoidance, causing the networks to be overloaded. TCP-Tahoe [8] introduced congestion avoidance, where dropped packets are used as an indication of congestion, and slow start, where the initial window size keeps doubling until congestion is detected. The following TCP versions: Reno [11], NewReno [14], Vegas [9] and Westwood [12] perform accordingly based on their core mechanisms.

II. BRIEF DESCRIPTION OF AOMDV, TCP-RENO AND TCP-VEGAS

An enhanced version of AODV, called AOMDV, maintains routes for destinations in active communications. It is a timer-based protocol and provides a way for mobile nodes to respond to link breaks and topology changes. It uses sequence numbers to determine the freshness of routing information and to prevent routing loops.

In AODV, when a source node *A* has data packets for a destination, it first checks its routing table to ascertain whether it already has a route to the destination node *B*. If it does, it sends the packets by utilizing the existing route. If it doesn't, it initiates a route discovery procedure by broadcasting a Route REQuest (RREQ) packet to obtain a route to the intended destination *B*. Node *A* copies the destination sequence number in RREQ from the entry in its routing table. If the sequence number is unknown (e.g., an entry for node *B* does not exist in the routing table), the *U* flag (Unknown flag of RREQ) is set to 1. Node *A* increments its originator sequence number and inserts it in the RREQ. The hop count field of RREQ is set to 0. Before node *A* broadcasts the RREQ message, it buffers RREQ ID and originator IP address for a maximum period of RREQ timeout.

AOMDV is to compute multiple paths and observe each route advertisement to define an alternate path to the source or the destination during a route discovery procedure. RREQ packets arriving at the nodes are copied and sent back to the source nodes. This approach avoids the formation of loops due to accepting all copied routes. In order to eliminate any possibility of loops, AOMDV made some modifications based on AODV. To achieve this goal, it adds an *advertised hop count* field in the route tables. The *advertised hop count* of a node *S* for a destination *D* is set to the maximum hop count of the multiple paths for *D* at *S*. The *advertised hop count* is initialized each time the sequence number is updated. In this way, AOMDV only allows accepting alternative routes with lower hop counts.

Each RREQ conveys an additional *first hop* field to indicate the first neighbor of the source node. The intermediate nodes do not discard duplicate copies of RREQ immediately as long as each RREQ provides a new node-disjoint path to the source. If an intermediate offers a new path, a reverse path is set up. It sends back a Route REPLY (RREP) to the source. At the destination, reverse routes are established like in the situation of intermediate nodes. If a link break occurs between a source and destination, a node

that knows this link break needs to send Route ERRor (RERR) message back to the source node. For periodic route updates, HELLO messages are broadcast in a timely manner.

The packets that are ported from a routing layer protocol are needed to check at a source and destination nodes to ascertain whether the packets arrive at the destination correctly if file transfer applications of TCP is used as a traffic source. A conventional TCP uses a window-based protocol that controls the number of packets that can be sent and ensures the packets that are received by the destination successfully by utilizing acknowledgement (ACK) packets to inform a source that data packets are received correctly.

TCP-Reno possesses four congestion control algorithm: slow start, congestion avoidance, fast retransmit and fast recovery. The first two mechanisms are used by a source node to control the data packets that is being sent into the network. The slow start is used at the beginning of a data transfer and after repairing the packet losses. When a data packet has to be transmitted, TCP-Reno requires probing the network conditions such as bandwidth and capacity slowly. During the slow start, a TCP source starts increasing its congestion window (cwnd) by one and whenever it receives an ACK packet, the cwnd is incremented by an exponential growth and stops when the cwnd reaches slow start threshold. From now on, it deals with a congestion avoidance region, where cwnd is increased by one whenever an ACK is received until congestion is detected. When the congestion occurs, a TCP source reduces its transmission rate and invokes a slow start.

There are two indications of packet loss: a timeout occurs and the receipt of duplicate ACKs. When the packet loss occurs, a TCP receiver sends a duplicate ACK immediately while a TCP sender utilizes a fast retransmit mechanism, where the arrival of 3 duplicate ACKs indicates that a packet has been lost and recovers the missing packet without waiting for a retransmission timer to expire. A fast recovery algorithm, in turn, governs the transmission of new data packets until a non-duplicate ACK arrives [11].

TCP-Vegas introduces a proactive congestion avoidance technique which does not violate the congestion avoidance paradigm of TCP. Rather than increasing a sending rate until a packet loss occurs, it tries to prevent losses by decreasing the sending rate when it perceives incipient congestion even if there is no indication of packet loss. Packet delay indicates a congestion. In this situation, when a duplicate ACK is received, the timestamp for the ACK is compared to a timeout value. If the timestamp is greater than the timeout value, then it will retransmit rather than wait for three duplicate ACKs. It detects congestion at an initial stage based on increasing Round Trip Time (RTT) values of the packets, whereas TCP-Reno detects congestion only after it has actually happened via packet drops. It adopts a more sophisticated bandwidth estimation scheme. To estimate the available bandwidth, the differences between expected and actual flow rates are taken into account. When the network is congested, the actual flow rate is smaller than the expected flow rate. Otherwise, it will be close to the expected flow rate. This can be calculated the following equation.

$$Diff = (Expected - Actual)BaseRTT \quad (1)$$

where *Expected* is the expected rate, *Actual* is the actual rate, and *BaseRTT* is the minimum round trip time. Based on *Diff*, the source updates its window size (*CWND*) as follows.

$$CWND = \begin{cases} CWND + 1 & \text{if } Diff < \alpha \\ CWND - 1 & \text{if } Diff > \beta \\ CWND & \text{otherwise} \end{cases} \quad (2)$$

In this way, it is able to utilize extra bandwidth without the network congestion and oscillation in window size, whereas TCP Reno always updates its window size to guarantee full bandwidth utilization, which leads to packet losses constantly.

III. SIMULATION PARAMETERS

We use a Network Simulator NS-2 [17] that was extended by the CMU Monarch research group. The distributed coordination function (DCF) of IEEE 802.11 [5] for wireless LANs is used at the MAC layer. We use the random waypoint mobility model [15], where a node starts moving from a randomly chosen position and stay in one location for a certain period of time (i.e. a pause time). Once this time expires, the node chooses a destination and moving speed randomly. This speed is uniformly distributed between minimum and maximum speed. We vary not only the maximum node speed to change node speed, but also the offered traffic load to test the network congestion. We simulate 100 nodes in a 1200 x 600 simulation area during 600 seconds simulation time. Only 512 byte TCP or ftp (file transfer protocol) packets are transmitted and its window size is set to 32.

The following parameters are used as performance metrics:

- 1) Packet Loss Rates (PLR) — how many of packets loss in application layer while transferring data packets i.e.,

$$PLR = \frac{Dropped\ Packets}{Highest\ Packet\ ID + 1} * 100$$

- 2) Throughput — the rate of successfully delivered data per second to individual destinations during the network simulation.
- 3) Normalized Routing Load (NRL) — the number of routing packets transmitted per data packet delivered at the destination. This metric is also highly correlated with the number of route changes occurred in the simulation.
- 4) Average Delay (AD) — a transmission delay of data packets that are delivered to the intended destination successfully.

IV. SIMULATION RESULTS

In this section, we discuss the performance differences based on their core protocol mechanism.

A. Varying Node Speed

We show four performance metrics as a function of node speed. The maximum node speed is changed from 0 to 20 m/sec for varying node mobility. The 60 TCP traffic sources are simulated for file transfer applications and nodes move at

the maximum speed and pause 10 sec at the corner, then loop until simulation ends.

Fig 1(a) shows that TCP-Reno delivers packets with a lower loss rate than TCP-Vegas at every node mobility whereas the loss rate variation of TCP-Vegas is almost stable. As node speed increases, the packet loss occurs frequently — most likely due to route failures rather than network congestion. However, the congestion control algorithm of TCP assumes all packet losses as congestion and invokes the congestion control algorithm, resulting in more packet loss. Also, the throughput of TCP-Vegas is lower than that of TCP-Reno as shown in Fig 1(b). The node mobility is always directly proportional to the packet loss rate and inversely proportional to the throughput. Fig 1(c) shows that NRL of TCP-Reno is slightly lower than TCP-Vegas. For three performance metrics over varying node mobility, TCP-Reno achieves a better performance than TCP-Vegas. On the other hand, average delay of TCP-Vegas is significantly lower than TCP-Reno at all speed rates (Fig 1(d)).

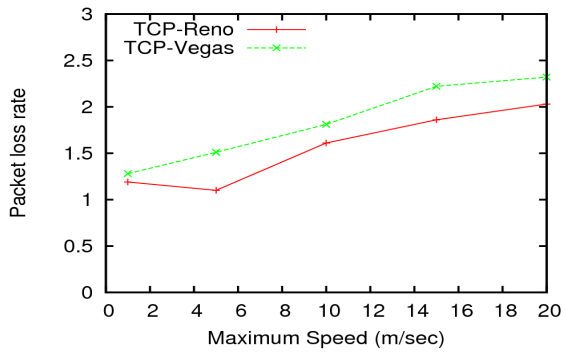
B. Varying traffic load

We measure the performance metrics by varying the offered traffic loads from 20 to 100 ftp connections. The 100 nodes move at 10 m/sec maximum speed and pause 10 sec at the border, then repeat action during the simulation time.

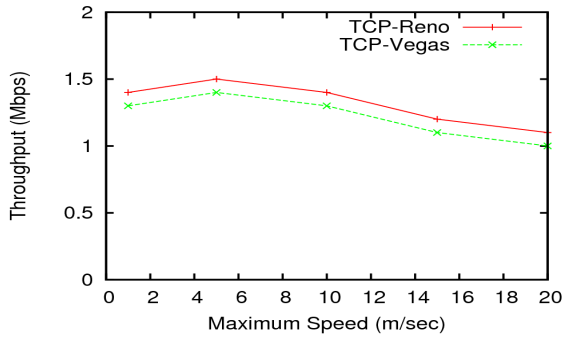
Even though the packet loss rates of TCP-Reno and TCP-Vegas are slightly different below 60 ftp connections, TCP-Reno has significantly lower packet losses than TCP-Vegas after 60 ftp connections (Fig 2(a)). Like in mobility variations, the data transfer rate of TCP-Reno is still higher than TCP-Vegas as the offered traffic loads are varied (Fig 2(b)). Also, TCP-Reno still succeeds with lower NRL as traffic load increases. No matter how the background traffic loads and speeds are varied, TCP-Vegas always has a considerable lower delay than TCP-Reno (Fig 2(d)).

Adjusting the sending rate before packet loss occurs results in a correct sending rate without relying on packet loss indication and this leads to a prominently lower delay. Moreover, as TCP-Vegas uses a delay estimation based on round trip time to adjust its window size, it is very important to obtain an accurate estimate in a mobile environment. Otherwise, its core mechanism results in worse performance. As the results show, TCP-Vegas suffers from packet losses, throughput and routing overhead worse than TCP-Reno in all background speed changes and traffic loads.

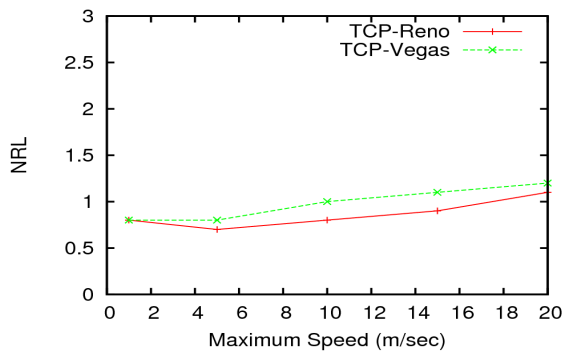
After a source node has sent data packets to a destination node, it needs to ascertain whether the sent packets reach the destination correctly and successfully. For this purposes, the destination node has to acknowledge all received packets. During this period, the route that was used for sending data packets may not be available when ACK packets are sent back via this route. In this situation, TCP-Vegas does not possess any mechanism to deal with the rerouting of a connection, which may change the delay of the connection and reduce the performance throughput. If a new route has a longer delay due to a longer path length, it assumes such delay as a sign of congestion and increases the window size by storing the same number of packets in the buffer. This approach reduces the performance of TCP-Vegas in MANET environments.



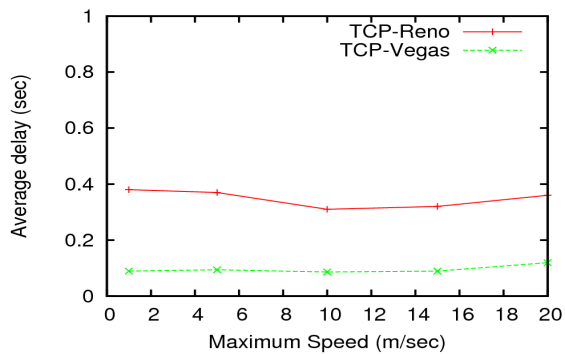
(a) Packet loss rate



(b) Throughput (Mbps)

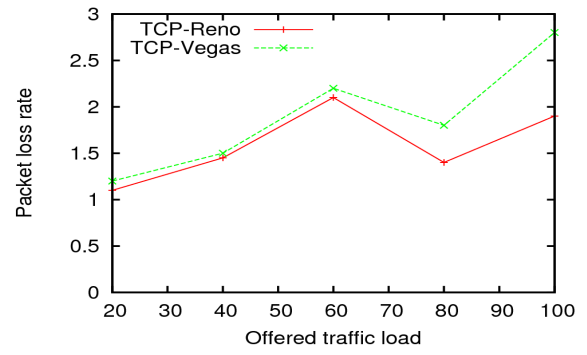


(c) Normalized routing load

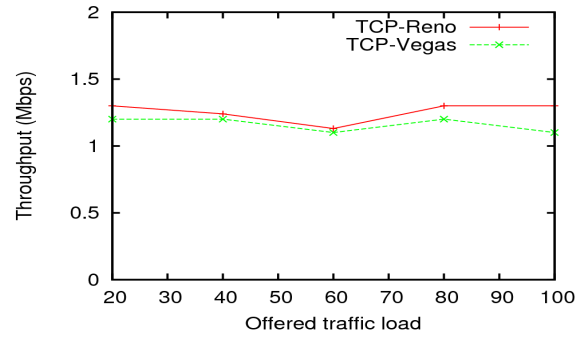


(d) Average delay (sec)

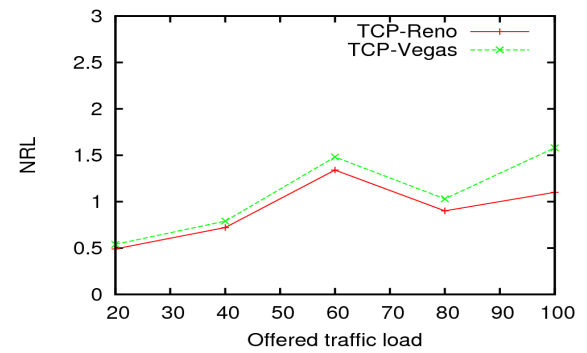
Figure 1: Performance analysis with varying node mobility



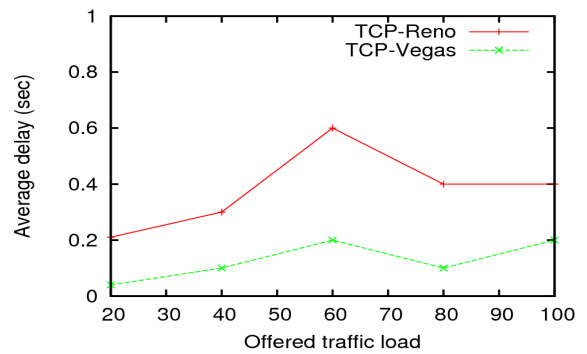
(a) Packet loss rate



(b) Throughput (Mbps)



(c) Normalized routing load



(d) Average delay (sec)

Figure 2: Performance analysis with varying traffic load

On the contrary, as AOMDV routing protocol maintains multiple next hops in the routing table, it improves throughput and reduces routing overhead by salvaging routes when a route break occurs. [10] has investigated AOMDV performance by using Constant Bit Rate (CBR) as a traffic source and their analysis results pointed that AOMDV achieves a higher performance if compared with AODV by maintaining one more routes to a destination. As an extension, [18] shows that AOMDV does not perform as well as AODV if it is heavily loaded with more than 100 nodes network size and very heavy traffic load. However, in this work, we make use of AOMDV in light load traffic and network size to test the performance of two TCP variants.

[7, 9] demonstrates that TCP-Vegas performs better than other implementations of TCP in many cases. However, in our analysis, TCP-Reno achieves fewer packet losses, higher throughput and lower NRL, except average delay than TCP-Vegas in a MANET environment because it encounters a difficult bandwidth estimate between a source and destination pair due to unexpected movement of mobile nodes.

V. CONCLUSIONS

In this paper, we examine the performance differences of TCP-Reno and TCP-Vegas when utilizing AOMDV as a routing protocol in a mobile environment. We vary the background node mobility and offered TCP traffic loads to measure the effectiveness of each transport protocol. Our simulation results show that TCP-Reno performs better than TCP-Vegas in the analysis of packet loss rate, throughput and NRL, except average delay. Even though TCP-Vegas controls network congestion with a delay estimate scheme, it is difficult to obtain a precise estimate in mobile environment, where packet losses are likely to be due to node movement and wireless channel errors. TCP interprets all packet losses as congestion, tending to performance degradation. TCP-Vegas suffers worse than TCP-Reno whereas its estimation scheme helps to reduce average delay. Even though TCP-Reno's packet losses detection mechanism gets worse average delay, it offers significant performance improvement in all background changes. On the other hand, AOMDV's core mechanism that salvages packet losses by maintaining the next hop information in the routing tables is mutually advantageous over itself and transport protocols when node mobility and traffic loads increases. All in all, TCP-Reno is the best protocol to apply in MANET with AOMDV routing protocol if compared with TCP-Vegas.

REFERENCES

- [1] C.E. Perkins, and T.J. Watson, "Highly dynamic destination sequenced distance vector routing (DSDV) for mobile computers," *Proc. ACM Communications, Architectures, Protocols and Applications*, vol. 24, pp. 234-244, 1994.
- [2] C.-C. Chiang, H.-K. Wu, W. Liu, and M. Gerla, "Routing in clustered multihop mobile wireless networks with fading channel," *Proc. IEEE SICON'97*, pp. 197-221, 1997.
- [3] C.E. Perkins, E. M. Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561, IETF, July 2003.
- [4] D. Johnson, Y. Hu, and D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," Internet experimental RFC 4728, February 2007.
- [5] IEEE 802.11: IEEE Standards for Information Technology, 1999.
- [6] J. Postel, "Transmission Control Protocol (TCP)," RFC 793, September 1981.
- [7] J.S. Ahn, P.B. Danzig, Z. Liu, and L. Yan, "Evaluation of TCP Vegas: emulation and experiment," *IEEE Transactions on Communications*, vol. 25, October 1995.
- [8] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno and SACK TCP," *Proc. ACM SIGCOMM Computer Communication Review*, vol. 26, pp. 5-21, 1996.
- [9] L.S. Brakmo and L.L. Peterson, "Tcp Vegas: end to end congestion avoidance on a global internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1465-1480, October 1995.
- [10] Marina, M.K. and Das, S.R., "On-demand Multipath Distance Vector Routing for Ad Hoc Networks," *Proc. Of 9th IEEE Int. Conf. On Network Protocols*, pp. 14-23, 2001.
- [11] M. Allman, V. Paxson, and W. Stevens, "RFC 2581-TCP Congestion Control," Internet Society, 1999.
- [12] M. Gerla, M.Y. Sanadidi, RenWang, A. Zanella, C. Casetti, and S. Mascolo, "TCP Westwood: congestion window control using bandwidth estimation", *IEEE Global Telecommunications Conference*, vol.3, pp. 1698-1702, Nov. 2001.
- [13] S. Sung, Y. Seo, and Y. Shin, "Hierarchical clustering algorithm based on mobility in mobile ad hoc networks," *Lecture Notes in Computer Science*, pp. 954-963.
- [14] S. Floyd, T. Henderson, and A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm", Internet Draft, <http://www.ietf.org/rfc/rfc3782.txt>, April 2004.
- [15] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communication and Mobile Computing Special Issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, pp. 483-502, 2002.
- [16] V.D. Park and M.S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," *Proc. sixteenth Ann. Joint conf. of the IEEE computer and Communication Societies*, vol. 3, pp. 1405-1413, 1997.
- [17] VINT Group, "Network Simulator NS-2," source code available: <http://www.isi.edu/nsnam/ns/>
- [18] M.Z. Oo and M. Othman, "Analysis of Single-path and Multi-path AODVs for Mobile Ad Hoc Network", (Submitted for review).