

# A Pragmatic Approach towards the Improvement of Performance of Ad Hoc Routing Protocols

Rajesh Roy, Sudipto Das, Pradip K. Das

Department of Computer Science & Engineering,

Jadavpur University, Kolkata – 32

rajesh.roy@rediffmail.com, sudipto.das@rediffmail.com, pkdas@cse.jdvu.ac.in

## Abstract

*Mobile ad hoc networks are typically characterized by high mobility and frequent link failures that result in low throughput and high end-to-end delay. In order to facilitate communication within the network, a routing protocol is used to discover routes between nodes. The primary goal of such an ad hoc network routing protocol is correct and efficient route establishment between a pair of nodes so that messages may be delivered in a timely manner. Though a large number of routing protocols have been developed for this kind of networks, most of them suffer from implementation difficulties and some inherent incapacibilities to cope with highly dynamic scenarios as well as support for Quality of Services (QoS). In this paper we propose to introduce a new layer in between network layer and IEEE 802.11 MAC sub layer. This layer will interact with the underlying MAC layer and provide link connectivity information as well as various other valuable parameters (e.g. received signal strength, network resource usage etc.), which can be used for further enhancement of the routing algorithms.*

**Key Words:** Routing protocol implementations, Dynamic scenarios, IEEE 802.11 MAC Layer

## 1. Introduction

A mobile ad hoc network is an autonomous system of multi hop, wireless mobile nodes that does not require base stations or any fixed infrastructure. It is characterized by dynamic topologies, bandwidth-constrained, variable capacity links, energy constrained operation and frequent link failures. The lack of infrastructure, in combination with multi hop connections and constantly changing topology pose difficult challenges for the routing protocol; foremost among them is how to deliver data packets while incurring the least routing overhead possible.

Several ad hoc routing protocols like AODV [2], MP-AOMDV [3], SSA [4] or DSR [7] have been proposed in the last few years. These protocols have been subject to intensive evaluations through simulation, but far less effort have been documented on the evaluation of the corresponding protocol implementations. Creating working implementation of most of the routing algorithms is non-trivial and more difficult than developing simulations. In simulation, the developer controls the whole system, which is in effect only a single component. An implementation, on the other hand, needs to interoperate with a large, complex system. Some components of this system are the operating system, sockets, and network interfaces. Additional implementation problems surface because current operating systems are not built to support ad hoc routing protocols. Because these events encompass many system components, the components and their interactions must also be explored. For these reasons it takes significantly more effort to create a working routing protocol implementation as compared to a simulation.

Various routing protocols implementations face problems [17] due to the complexity of the underlying IEEE 802.11, which is difficult to handle by the routing protocol itself. And most of these algorithms assume that some of the key parameters are easily available from the underlying layer, which are critical for their operations.

In this paper we propose to introduce a new layer in between ad hoc routing entity and the underlying 802.11 MAC layer. This layer is designed in such a way that it will provide a generic solution towards the above-mentioned problems. This will also serve as a generic platform over which more efficient future ad hoc routing protocol can be developed without concerning about the intricate details of underlying layer.

The rest of this paper is organized as follows. Section 2 discusses the motivation for introduction of such a layer. In section 3, we present some overview of IEEE 802.11. Section 4 will present our layer design specification and a straightforward algorithm for detecting stable neighbor. In section 5 we present the proposed changes of AODV so that it can operate with the assistance of newly introduced

layer. Section 6 reports some implementation details and results obtained from actual experiments using the new layer. Finally, section 7 provides concludes the paper with a brief discussion of possible future enhancements.

## 2. Motivation

The last decade has seen a huge increase in wireless networks and gradually, the mobile ad hoc networks have become more popular. So, the performance evaluation of these networks is an active area of research. Various improvements in such networks have come forth in the last few years.

In [4] the Signal Stability based adaptive routing protocol (SSA) uses the signal strength and stability of individual hosts as a route selection criterion. Selecting the most stable link (i.e. those which exhibit strongest signals for the maximum amount of time) leads to longer-lived routes and less route maintenance. Both of these parameters can be determined only with the assistance of underlying MAC layer. Though the protocol itself introduces a extended device driver interface which communicates with DRP (Dynamic Routing Protocol) module as defined in [4] for its functioning, this extended interface is not well defined and does not have a generic solution towards other problems faced by other routing protocols.

In [3] the *Mobility Prediction Ad hoc On-Demand Multi path Distance Vector (MPAOMDV)* routing protocol propose to use of the signal strength metric over the usual hop count metric because the hop count of a route is not sufficient to determine the quality and stability of the path. A very weak link, even if on a low hop count route, could lead to a significant number of dropped packets. In contrast, since the signal strength metric is based on the signal strength of each individual link in the path, it provides information about both the quality and reliability of the path. This algorithm assumes the measurement of received signal strength is a trivial task even though it is not so. And this algorithm also demands a separate underlying layer, which will feed useful information to the routing module and keep track of the network connectivity.

The well-studied routing algorithm AODV [1,2] also faces critical performance degradation, which is shown in [5]. The problem of communication gray zones introduce increased packet loss and hence performance degradation. In such zones data messages cannot be exchanged although the HELLO messages indicate neighbor reachability. This leads to a systematic mismatch between the route state and the real world connectivity, resulting in disruptive behavior for data transfer over ad hoc routing protocols. The problem of gray zones invoke the issue that reachable neighbor list can not be determined only

with the exchange of HELLO messages but it requires separate algorithm to determine such information which is shown in [5]. This again leads to the introduction of a new layer in between ad hoc routing entity and underlying 802.11 MAC layer.

AODV faces another problem when operating in highly dynamic topologies. In [2] it is shown that a route breakage can be determined with the help of HELLO messages and can be notified to the concerned nodes with the help of Route Error Messages. But in a dynamic scenario when a new node arrives within the range of another node during an ongoing communication, the communicating node is incapable of exploring the possibility of a new route taking into account the newly arrived node. The problem exists because AODV itself has no mechanism to detect the presence of the new node in its neighborhood. The exchange of HELLO messages cannot solve this problem because according to [2] a node should only use hello messages if it is part of an active route. But it will not notify its neighbors about its emergence if the node itself is not a part of an active route.

The problems mentioned above are only a few faced by the various established ad hoc routing protocol implementations, which can be eliminated by the introduction of the proposed new layer thereby improving the performance of the network as well as relieving the additional overhead from the routing protocols.

## 3. Overview of the IEEE 802.11 Standard

The IEEE 802.11 Standard [8] is by far the most widely deployed wireless LAN protocol. This standard specifies the physical, MAC and link layer operation. Multiple physical layer encoding schemes are defined, each with a different data rate. Part of each transmission uses the lowest most reliable data rate, which is 1 Mbps. At the MAC layer IEEE 802.11 uses both carrier sensing and virtual carrier sensing prior to sending data to avoid collisions. Virtual carrier sensing is accomplished through the use of Request-To-Send (RTS) and Clear-To-Send (CTS) control packets. When a node has a unicast data packet to send to its neighbor, it broadcasts a short RTS control packet. If the neighbor receives this RTS packet, then it responds with a CTS packet. If the source node receives the CTS, it transmits the data packet. Other neighbors of the source and destination that receive the RTS or CTS packets defer packet transmissions to avoid collisions by updating their Network Allocation Vector (NAV). The NAV is used to perform virtual channel sensing by indicating that the channel is busy. After a destination properly receives a data packet, it sends an acknowledgment (ACK) to the source. This signifies that the packet was correctly received. This procedure (RTS-

CTS-Data-ACK) is called the Distributed Coordination Function (DCF). For small data packets the RTS and CTS packets may not be used. If an ACK (or CTS) is not received by the source within a short time limit after it sends a data packet (or RTS), the source will attempt to retransmit the packet up to seven times. If no ACK (or CTS) is received after multiple retries, an error is issued by the hardware indicating that a failure to send has occurred. Broadcast data packets are handled differently than unicast data packets. Broadcast packets are sent without the RTS, CTS or ACK control packets. These control messages are not needed because the data is simultaneously transmitted to all neighboring nodes. In ad hoc mode of operation one of the peer stations assumes the responsibility for sending the beacon. After receiving a beacon frame, each station waits for the beacon interval and then sends a beacon if no other station does so after a random time delay. This ensures that at least one station will send a beacon, and the random delay rotates the responsibility for sending beacons.

## 4. Design Specification

For the detection of potential neighbors by the proposed layer, it requires some sort of beacon messages, which will notify the presence of a neighbor.

There are various possible ways of solving the purpose of beaconing, viz. beacon messages inherent to the IEEE 802.11 MAC layer & protocol specific Hello messages. Both methods have certain accompanying problems. The 802.11 beaconing will not introduce any additional overhead, but obtaining feedback from the MAC layer poses a few problems [6] and is also not suitable for handling the gray – zone problem [5].

Numerous ad hoc routing protocols [1, 9, 10, 11, 12] make use of periodic broadcast messages to determine local connectivity. Also, because of the difficulty of obtaining IEEE 802.11 feedback about link connectivity in real networks, many current protocol implementations utilize hello messages [13, 14, 15, 16].

Our layer follows the latter one although the simplest version of hello messages will introduce new problems [5], which are described as follows:

a. *Different Transmission Rate*: In IEEE 802.11b, broadcasting is always done at a basic bit rate while data transmissions normally are sent at higher rates (up to 11 Mbit/s in IEEE 802.11b). Transmissions at lower bit rates are more reliable and can reach further than at higher rates. As HELLO messages are broadcasted, this is the main cause to why gray zones appear.

b. *Small Packet Size*: The size of a HELLO message is small compared to a data packet. Small packets are less prone to bit errors since there are less bits to transfer than in large packets. Also, they have a smaller probability of

colliding with the usually longer data packets. This makes it more likely for a HELLO message to reach a receiver than a data packet, especially over weak links.

c. *Fluctuating Links*: At the transmission borderline, communication tends to be unreliable due to fluctuating quality of links. This leads to spurious HELLO messages, which, once received, do not reflect correctly whether consistent communication between two nodes is possible, or not. As a consequence this means that shorter but unreliable ones can replace stable and longer routes.

In our layer design we have adopted a *N-consecutive periodic hello messages* scheme. In this scheme a node should receive N consecutive & periodic HELLO messages from the same source before accepting it as a neighbor. This will eliminate the *Fluctuating Neighbor problem* though will introduce some problem of delayed response. Again by setting some minimum signal strength threshold value for hello messages will eliminate the problem of “*communication gray zone*”. The subsequent sections give a brief explanation of the algorithm for detecting stable neighbor, used by us. In this section we will call this hello messages simply as beacons.

### 4.1. Overview

Any node that is a potential neighbor for the node may have one of the three states defined as follows:

❖ **Unstable Neighbor**: A Node B is termed as an *Unstable Neighbor* of the Host Node A if A has received some beacons from the Node B but the beacons received are not regular. Such a node cannot be considered as a route for data traffic.

❖ **Meta Stable Neighbor**: A Node B is termed as a *Meta Stable Neighbor* of the Host Node B if A has received a considerable number of beacons almost regularly. Such a neighbor may be considered as a route for data traffic under certain circumstances.

❖ **Stable Neighbors**: A Node B is termed as a *Stable Neighbor* of the Host Node A if A has received a significant number of beacons from the node B and is regularly receiving beacons from it. Such a neighbor is a valid route for data traffic.

As a node enters into the periphery (i.e. the range) of another node, it would receive the beacons from the other node. The State of such a node is gradually escalated from *Unstable* to *Stable*, through *Meta Stable* state, depending on the rate of beacon arrivals. Every State has a corresponding associated *Time Out*, and after each time out, the node status is somewhat degraded. Also, the *Unstable* and *Meta Stable* states has an associated counter which is after reaching a definite value, the corresponding node is promoted to the next higher state, and if it attains a non – positive value, the corresponding node is demoted to the next lower state.

## 4.2. Data Structures Used

This module uses a lookup table for storing the states of the neighboring nodes. We will call this as *neighbor\_table*. Each node will store the following parameters for their neighboring nodes: *destination\_address*, *beacon\_counter*, *time\_stamp* and *state*. As the name suggests, the function of each field should become quite evident. The *destination\_address* field is used to store the address of the Node corresponding to which the entry is made, the *beacon\_counter* field is used to count the number of beacons that have been received from that node, whereas the *time\_stamp* field is used to store a simple Time Stamp that saves the time when the entries were last modified. And the last field will denote the state of the neighbor i.e. whether the neighbor is *Unstable*, *Meta stable* or *Stable*.

## 4.3. Algorithm

The actual algorithm that is used to grade the nodes as *Unstable*, *Meta Stable* or *Stable* and to keep track of the node in the various states is explained as follows:

Whenever any Node B enters the range of a Host Node A, node A would receive beacons from node B and if A gets the beacon for the first time then there will be no corresponding entry in the *Neighbor\_Table*. As soon as the first beacon is received from B, it is tentatively marked as *Unstable* and a corresponding entry is made in the *Neighbor\_Table*, with the *beacon\_counter* field set at 1 and the *time\_stamp* field set according to indicate the present time. Now, for every new beacon received from this node B, the counter is incremented by 1 and the Time Stamp refreshed and if the value of the count exceeds a predetermined value  $\tau_1$  (typically 10), the node is promoted to *Meta Stable* state and the *beacon\_counter* field is set to 1. The *Unstable* state also has an associated *Time Out* interval  $\lambda_1 (\approx \gamma_1 \times \theta)$ ,  $\theta$  being the beaconing interval and  $\gamma_1$  being a predetermined constant whose typical value is 7). The *Neighbor\_Table* is periodically scanned and the entries whose time stamps are older than  $\lambda_1$  have their *beacon\_counter* fields decremented by 1. The entries, which have a non – positive value of the counter, are purged from the table. This purging is done to free up the resources used by a node that has left the radio range of the node considered and is no longer its neighbor.

Whenever a beacon received from a node which is marked as *Meta Stable* in the *Neighbor\_Table*, the counter field of the corresponding node is incremented by 1 and the Time Stamp is refreshed and if the value of the count exceeds a predetermined value  $\tau_2$  (typically 7), the node is promoted to *Stable* state and *beacon\_counter* is set to 1. The *Meta Stable* state also has an associated *Time*

*Out* interval  $\lambda_2 (\approx \gamma_2 \times \theta)$ ,  $\theta$  being the beaconing interval and  $\gamma_2$  being a predetermined constant whose typical value is 5). The *Neighbor\_Table* is periodically scanned and the entries whose time stamps are older than  $\lambda_2$  have their *beacon\_counter* fields decremented by 1. The entries which have a non – positive value of the counter are demoted to the *Unstable* state with the counter set at  $[\tau_1 * 3 / 4 + 0.5]$  (where  $[\ ]$  denotes the greatest integer function) and the *time\_stamp* field being refreshed to reflect the present time.

Whenever a beacon received from a node, which is marked as *Stable*, the *time\_stamp* is refreshed. The *Stable* state too has an associated *Time Out* interval  $\lambda_3 (\approx \gamma_3 \times \theta)$ ,  $\theta$  being the beaconing interval and  $\gamma_3$  being a predetermined constant whose typical value is 3). The *Neighbor\_Table* is periodically scanned and the entries whose time stamps are older than  $\lambda_3$  have the corresponding nodes demoted to the *Meta Stable* state with the counter set at  $[\tau_2 * 3 / 4 + 0.5]$  (where  $[\ ]$  denotes the greatest integer function) and the *time\_stamp* field being refreshed to reflect the present time.

The line of thought for such an algorithm is pretty straightforward, as the node is gradually promoted from the *Unstable* to the *Meta Stable* and finally to the *Stable* state, it is expected that beacons would be received with greater regularity and hence as the node move towards greater stability, the timeout intervals become more stringent, failing to abide by which, the node is demoted to the next lower state or a counter is decremented – whichever is relevant. Also the promotion from *Meta Stable* to *Stable* requires greater number of regularly arriving beacons as compared to that required for the promotion to the *Meta Stable* state from the *Unstable* state. The logic behind using an intermediate value of the *beacon\_counter* field while demoting a node to the immediately lesser stability state is that – since the node was previously occupying a more stable state, but due to some reason there has been a drop in the regularity of the beacons received, placing an intermediate value facilitates a quick return to the previous state subject to the constraint that the frequency of the arrival of beacons has increased. Such a measure places the demoted node at a better position as compared to a newly promoted node. But if there is no increase in the regularity, there is a gradual decrease in the counter and stability status of the node before it is finally purged from the table.

Through the lookup table maintained by this module, it provides the routing algorithm, operating in the network layer, with information about the stability of its neighbors, which the routing algorithm can use according to its requirements. For our implementation, we mark the control messages for the routing algorithm with the state of the node from which the message has been received. Now, as this message reaches the routing entity, it reads the stability information from the specified field and

interprets the message accordingly. In an attempt to refrain from spurious route updates, the node may neglect the route update messages if they are from an unstable neighbor.

We provide a pseudo code for the algorithm used to maintain the look-up table containing stability information of the neighbors of a node.

*Pseudo code for updating table (invoked whenever a beacon / data packet is received):*

```

if ((entry=get_entry(node_identification))=NULL){
    create entry in neighbor_table;
    beacon_counter := 1;
    entry.state := Unstable_State;
} else{
    if (entry.state == Unstable_State){
        increment beacon_counter;
        refresh timer;
        if (beacon_counter >  $\tau_1$ ){
            beacon_counter := 1;
            entry.state := Meta_Stable_State;
            refresh timer;
        }
    }
    if (entry.state = Meta_Stable_State){
        increment beacon_counter;
        refresh timer;
        if (beacon_counter >  $\tau_2$ ){
            beacon_counter := 1;
            entry.state := Meta_Stable_State;
            refresh timer;
        }
    }
    if (entry.state = Stable_State){
        refresh timer;
    }
}

```

*Pseudo code for refreshing neighbor\_table (invoked periodically):*

```

for( each entry in neighbor_table){
    entry=get_entry(node_identifier)
    if(entry.state = Unstable_State){
        if(Unstable_State timeout){
            decrement beacon_counter by 1;
            if (beacon_counter < 1) purge_entry;
            else refresh timer;
        }
    }
    else if (entry.state = Metastable_State) {
        if(Metastable_State timeout){
            decrement beacon_counter by 1;
        }
    }
}

```

```

if (beacon_counter < 1) {
    entry.state := Unstable_State;
    beacon_counter := floor ( $\tau_1 * 3 / 4 + 0.5$ );
}
refresh timer;
}
else if (entry.state = Stable_State) {
    if(Stable_State timeout){
        entry.state := Metastable_State;
        beacon_counter := floor ( $\tau_2 * 3 / 4 + 0.5$ );
    }
    refresh timer;
}
}
}

```

## 5. Changes over AODV

In this section we explore a potential use of this newly introduced layer concentrating mainly on Ad hoc On-demand Distance Vector routing (AODV). AODV is an on-demand routing strategy that creates and maintains the routes only when they are required. The best route is one which has the least hop-count. A limitation of this hop-count metric in selection of the best possible route is that it overlooks the stability of these routes. As a result of this, AODV performance is greatly affected due to fluctuating neighbors. If suppose a node B passing through the radio range of a node A provides a possible route to some destination C with lesser number of hops than the existing route to C, the AODV routing entity in A would promptly update its routing table to insert the newly found route with B as the next hop. In doing so, it does not consider the stability of the node from which this update was received. As a result, when node B soon exits A's radio range, this route becomes invalid. The route error followed by a new route discovery ensues and results in performance degradation in terms of decreased throughput and high end-to-end delay.

Our layer aims to overcome this problem of AODV by providing this much needed stability information. The only modification is that while making the route updates, it takes into consideration the stability of the nodes from which the messages come. If the messages are from an unstable neighbor, they are rejected. Only the messages from stable neighbors are used for updating the routes. As a result, the node refrains from accepting the spurious route update messages from passing nodes. The required changes to make AODV compatible with the newly introduced layer and benefit from the additional available information are given below:

## 5.1. Generating Route Requests

A node with this modified AODV generates a Route Request (RREQ) on-demand in the following situations:

i) Firstly the node disseminates RREQ when it determines that it needs a route to a destination and does not have one available. This can happen if the destination is previously unknown to the node or if a previously valid route to the destination expires or is marked as invalid.

ii) Secondly the node disseminates a RREQ when the underlying layer notifies AODV about the emergence of a newly arrived stable neighbor within its range. AODV will generate one RREQ for each valid route entry in its routing table. This is done in an attempt to explore the possibility of the existence of some routes, better than the previously existing routes based on some parameter like hop-count or stability, through the newly integrated node in the network.

## 5.2. Maintaining Routing table

When a node receives an AODV control packet from a neighbor, or creates or updates a route for a particular destination, it checks its route table for an entry for the destination. In the event that there is no corresponding entry for that destination, an entry is created. The sequence number is either determined from the information contained in the control packet, or else the valid sequence number field is set to false. The route is only updated if the new sequence number is either

i) Higher than the destination sequence number in the route table, or

ii) equal, but the newly discovered route is marked as stable but the existing route entry in the routing table is marked as unstable i.e. a stable route replaces an unstable route, or

iii) the sequence numbers are equal and the state of both the newly discovered route and the existing route entry in the routing table are same (i.e. either both are stable or unstable), but the hop count (of the new information) plus one, is smaller than the existing hop count in the routing table, or

iv) the sequence number is unknown.

## 5.3. Processing Route Request and Route reply

Whenever a node receives a route request or a route reply it reacts according to the AODV specification [2]. The only modification is that prior to processing the route request or route reply, it checks whether it has come from an unstable neighbor. If it comes from an unstable

neighbor, it is neglected. While forwarding a route request or reply of some other node, it sets the corresponding bits depending on the stability of the neighbor from it receives this message. In our implementation we have used a reserved bit field available in the RREQ and RREP headers as indicated in [2]. As a result of this modification, the messages from fluctuating neighbors can be segregated from the messages sent by stable neighbors thereby preventing spurious route updates.

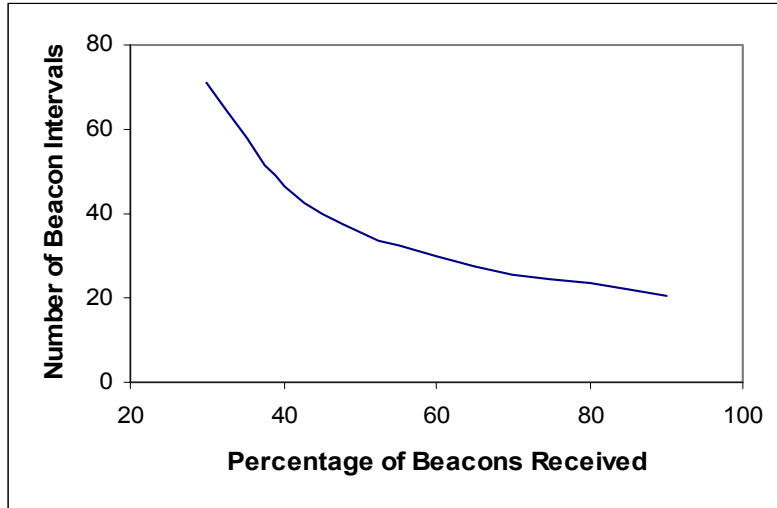
## 6. Implementation details and Experimental Results

We have developed a Linux Module that performs the tasks of the New Layer proposed by this paper. The module has been tested successfully for 2.4.22 kernel. The module augments an existing AODV implementation. We have used Kernel AODV (version 2.2.2) developed by NIST [14] which has been modified to suit our requirements. In our implementation, the maximum values of the counters for Unstable and Meta Stable states have been set to 10 and 7 respectively.

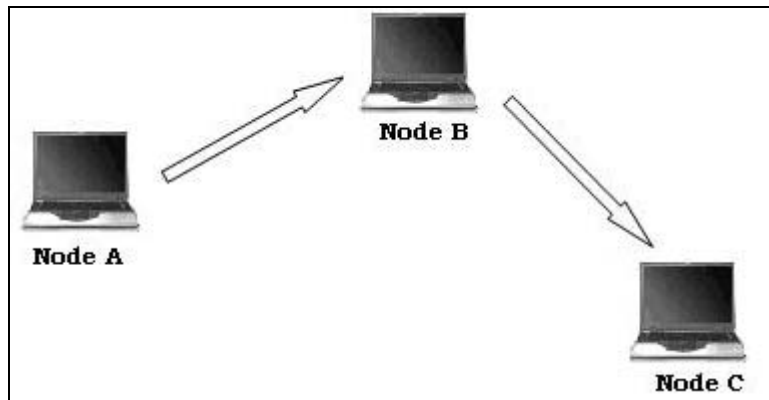
The following two subsections summarize the results obtained using this implementation.

### 6.1. Integrating a New Node into the network

An important value addition of this layer is that it can detect the presence of a new node in the network even if the new node is not participating in any kind of data transfers. This is achieved with the help of the Protocol Specific Beacons proposed by this layer. As a new node enters the radio range of another node, it starts receiving the beacons from this new node and an entry for this new node is made in the table maintained by this new layer. As more beacons are received from this node, the corresponding changes in state occur. As it reaches the stable state, it is integrated into the network and the possibility of a new route, better than the existing routes in terms of some parameters like hop-count, through this neighbor is explored. As a result, the proper detection of the neighbors becomes imperative. In this section, we performed experiments to determine how quickly a new node is integrated into the network. The stability of a neighbor is determined by what percentage of beacons it receives from its neighbor. The greater the percentage of beacons received, the more stable is the link between the two neighbors and lesser is the time required by it to become stable. Fig1 provides the number of beacon intervals required to integrate a new neighboring



**Fig.1 Average Number of Beacon Intervals vs. Percentage of Beacons received**



**Fig.2. Example topology**

node vs. varying percentage of beacons received. The experiment has been performed by varying the distance between the two nodes under consideration. Each reading was taken five times and the average values have been used in the plot.

The beacon intervals are a measure of the time required to integrate the new node into the network. From the implementation parameters set, a minimum of 18 beacons (without any loss or time-out) must be received from a node in order that it is marked as a stable neighbor by the recipient of the beacons.

As is evident from the Fig. 1, a more stable neighbor is escalated quicker as compared to a less stable neighbor. As the percentage of beacons received from the neighbor decreases, more time is required to integrate the new node into the network. A greater number of beacon intervals are required on account of some missed beacons and time-outs caused due to non-receipt of beacons. Hence the new layer is capable of differentiating between persistent neighbors and spurious passing nodes and refrains from accepting the spurious nodes as neighbors. A possible

new route through this newly integrated neighbor is explored only when it is considered to be stable, i.e. a fairly large percentage of beacons are received from it.

## 6.2. Performance Enhancements over AODV

Ad hoc routing protocol implementations suffer from a limitation that stems from Fluctuating Neighbors problem [5]. Let us consider a topology as in Fig 2 which we have used for our experiments. In this scenario, node A is sending data to node C using node B as an intermediate hop. In situations with high mobility, it might be possible that node C moves into the periphery of the radio range of node A. As a result, node A may receive some spurious route updates from C and hence would wrongly detect it as its neighbor. Node C being placed at the periphery of the radio range of Node A, very few data segments from node C arrive at node A. As a result, frequent route errors and consequently throughput degradation ensues. The New Layer prevents the acceptance of such spurious

messages and thereby prevents the throughput fluctuations due to the frequent route errors. Frequent route errors and consequent route discoveries, which involve significant route discovery latency, have adverse effects on system performance in addition to throughput degradation. Frequent route breaks cause the intermediate nodes to drop packets because no alternate path to the destination is available. This has adverse effects for applications such as VoIP where packet dropping may be detrimental to the overall system performance. Moreover the average end-to-end delay can be significantly high due to frequent route discoveries which can have adverse effect on traffic such as real-time streaming video which require paths with low jitter (i.e. low variation in delay of received packets).

In this sub section, we compare the performance of AODV [14] with the modified version of AODV based on the throughput metric. As is evident from Fig. 3, the modified AODV refrains from accepting the spurious route updates. As a result, it is able to eliminate the fluctuations in throughput as experienced by conventional AODV. In addition, as the modified AODV does not accept these spurious route updates, it can overcome other problems associated with frequent route discoveries.

## 7. Conclusion & Future Enhancements

The mobility scenario in an Ad hoc network has led to the degradation in the performance of the routing

algorithms. Due to the unavailability of various parameters, like received signal strength, network resource usage etc., to the routing protocols and the need for enhancement in their performance, the introduction of a new layer between the Network Layer and the MAC Layer becomes imperative. In this paper we have shown the claim is justified and we introduce such a layer and provide the design specifications of this new layer. Concentrating on AODV we have presented required modifications in the protocol specification so that it became compatible with the new layer and can take advantages of the layer. A comparison of performance with the existing implementations shows that this modified AODV, in addition to detecting new nodes within the network, is capable to segregate the reject spurious route updates.

We have developed a Linux implementation of the above-mentioned layer. We have used our prescribed algorithm for stable node detection and thus cope with *fluctuating neighbor* problem. We also used *minimum received signal strength threshold* to cope with “*communication gray zone*” problem. We hope to provide the performance evaluation of our implemented layer in an actual test bed in a future paper.

As this new layer has more accurate information about the network situation, it would be the perfect layer for introducing multi path routing, load balancing and QoS. We hope to explore these possibilities in future work.

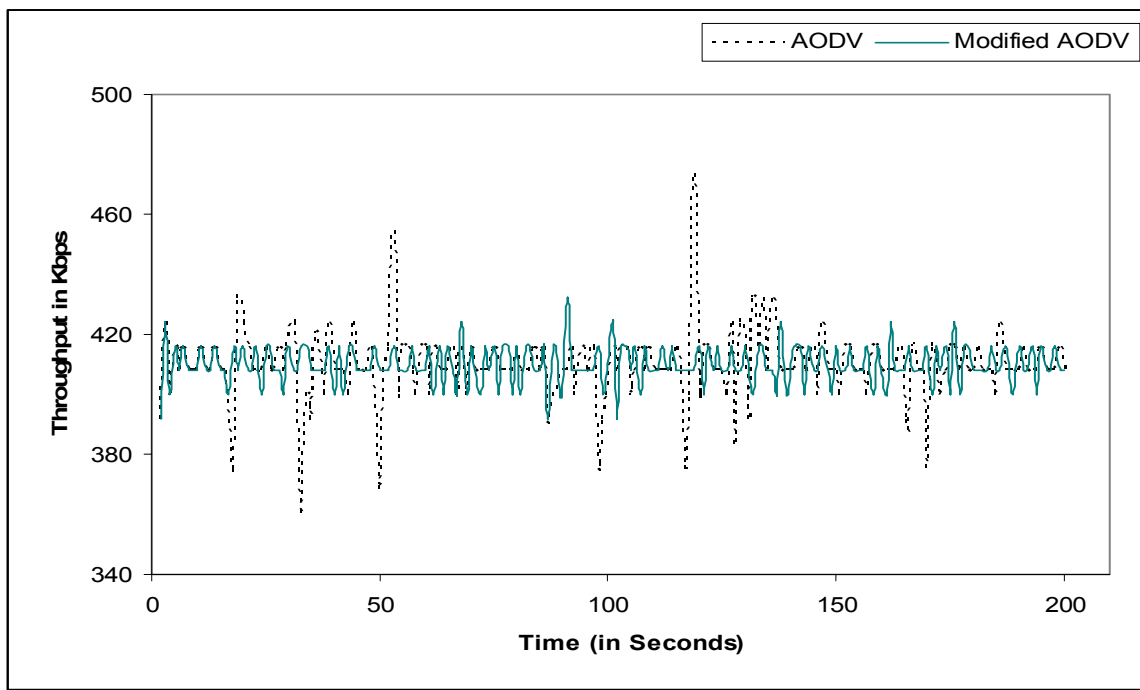


Fig. 3 Throughput (in kbps) comparison of AODV and modified AODV

## References

- [1] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das, “*Ad Hoc On-Demand Distance Vector (AODV) Routing.*” IETF Internet draft, draft-ietf-manet-aodv-13.txt, February 2003.
- [2] Charles E. Perkins and Elizabeth M. Royer, “*Ad hoc on-demand distance vector routing.*” in Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, February 1999, pp. 90–100.
- [3] Perumal Sambasivam, Ashwin Murthy, Elizabeth M. Belding-Royer, “*Dynamically Adaptive Multipath Routing based on AODV.*” Med-Hoc-Net 2004, The Third Annual Mediterranean Ad Hoc Networking Workshop June 27-30, 2004.
- [4] Rohit Dube, Cynthia D. Rais, Kuang-Yeh Wang, and Satish K. Tripathi, “*Signal stability based adaptive routing (SSA) for ad hoc mobile networks.*” in IEEE Personal Communication, February 1997, vol. 4.
- [5] Henrik Lundgren, Erik Nordstrom, and Christian Tschudin, “*Coping with communication gray zones in IEEE 802.11b based ad hoc networks.*” in Proceedings of the 5th ACM international Workshop on Wireless Mobile Multimedia, Atlanta, GA, September 2002, pp. 49–55.
- [6] Ian D. Chakeres and Elizabeth M. Belding-Royer, “*The utility of hello messages for determining link connectivity.*” in Proceedings of the 5th International Symposium on Wireless Personal Multimedia Communications (WPMC), Honolulu, Hawaii, October 2002.
- [7] D. Johnson, D. Maltz, J. Broch, “*DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks.*” in Ad Hoc Networking, edited by C. Perkins, pp. 139-172, Addison-Wesley, 2001.
- [8] IEEE Computer Society. IEEE 802.11 Standard, IEEE Standard For Information Technology, 1999.
- [9] Bhargav Bellur and Richard G. Ogier, “*A Reliable, Efficient Topology Broadcast Protocol for Dynamic Networks.*” *Proceedings of IEEE INFOCOM, pages 178–186, New York, NY, March 1999.*
- [10] Thomas Clausen, Philippe Jacquet, Anis Laouiti, Pascale Muhlethaler, Amir Qayyum, and Laurent Viennot, “*Optimized Link State Routing Protocol.*” Proceedings of the IEEE INMIC, Pakistan, 2001.
- [11] Mario Gerla, Xiaoyan Hong, and Guangyu Pei, “*Landmark Routing for Large Ad Hoc Wireless Networks.*” in Proceedings of IEEE GLOBECOM 2000
- [12] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris, “*A Scalable Location Service for Geographic Ad Hoc Routing.*” in Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom), pages 120–130, Boston, MA, August 2000.
- [13] Ian D. Chakeres. AODVUCSB Implementation from University of California Santa Barbara. <<http://moment.cs.ucsb.edu/AODV/aodv.html>>.
- [14] Luke KleinBerndt, Kernel AODV (v 2.2.2) from NIST. <[http://w3.antd.nist.gov/wctg/aodv kernel/](http://w3.antd.nist.gov/wctg/aodv%20kernel/)>.
- [15] Erik Nordstrom and Henrik Lundgren. AODVUU Implementation from Uppsala University. <[http://www.docs.uu.se/\\_henrikl/aodv/](http://www.docs.uu.se/_henrikl/aodv/)>.
- [16] Richard G. Ogier, Fred L. Templin, Bhargav Bellur, and Mark G. Lewis, “*Topology Broadcast Based on Reverse Path Forwarding.*” IETF Internet Draft, draft-ietf-manet-tbrpf-05.txt, March 2002. (Work in Progress).
- [17] Ian D. Chakeres, Elizabeth M. Belding-Royer “*AODV Routing Protocol Implementation Design.*”, icdcs, vol. 06, no. 6, pp. 698-703, 2004.