

CHAPTER 3

PROPOSED WORK

3.1 Introduction

This part is the core of our work. The mobility of nodes is the main factor which prompted us to propose a link failure prediction method and thinking to the need of improving the routing performance in MANET environments, by avoiding the frequent link breakage. In This chapter, section 3.2 will present the problem statement, followed by the proposed solution in section 3.3 which we suggested for getting better performance.

3.2 Problem identification

As mentioned in chapter 1, in the standard AODV protocol, when a link break occurs, the path has to be repaired (local recovery) or a new path has to be discovered. During alternate route discovery after link break, packets will be dropped. This leads to wastage of the scarce node resources and decreases the routing performance in terms of many metrics such as end to end delay and packet delivery ratio.

The next section (section 3.3) will give a solution for this problems, so, how can we improve the network performance?.

3.3 Proposed solution

In this section, we propose a link failure prediction algorithm AODV-LP (AODV with Link Prediction) Based on the Ad hoc On-demand Distance Vector AODV routing protocol, uses the received signal strength, RSS, and the received signal strength changing rate, to predict the time changing rate after which an active link will break. This is done by estimating the time at which received signal strength of the Hello packets will fall below a threshold power. The received power level below the threshold indicates that the two nodes are moving away from each other's radio transmission range. The nodes estimate the link breakage time and discover another route to the destination much earlier.

In this approach, two consecutive measurements of signal strength of packets received from the neighboring node are used to predict the link failure using the Newton divided difference method (The Newton interpolation polynomial) [48].

The received signal strengths of the tow latest Hello packets and their time of occurrence are maintained by each receiver for each transmitter from which it is receiving. Using tow received Hello packets signal power strengths as P_1 , P_2 and the time when Hello packets arrived as T_1 , T_2 instants respectively and $P_{\text{threshold}}$ as the threshold signal strength to be operative at the time $T_{\text{threshold}}$, one can predict $T_{\text{threshold}}$. We assume that at the predicted time $T_{\text{threshold}}$, when received power level reduces to threshold power, the link will break. The

threshold signal strength $P_{\text{threshold}}$, is the minimum power receivable by the device. This is the power at the maximum transmission range.

The expected $T_{\text{threshold}}$ for the threshold signal strength $P_{\text{threshold}}$ of the packets received can be computed as below:

Power (watts)	P_1	P_2	$P_{\text{threshold}}$
Time (s)	T_1	T_2	$T_{\text{threshold}} = ?$

Table 3.1 time threshold calculation

$$T_{\text{threshold}} = \frac{(T_2 - T_1) * (P_{\text{threshold}} - P_2)}{(P_2 - P_1)} + T_1 \quad (1)$$

Routing protocol needs a time to setup an alternate path, for this purpose a time changing rate is introduced and calculated like “ $T_{\text{threshold}} - T$ ”, this changing rate of time should be sufficient enough to send Request message to the neighboring nodes for finding another route to the destination much earlier, for initiate the process of discovering in advance, the time changing rate should be compared with a fixed time threshold, fixed_Th which we introduced, so the time changing rate should be smaller than fixed_Th to start the advance discovering route and broadcast a Request message. The Figure 3.1 shows our proposed method.

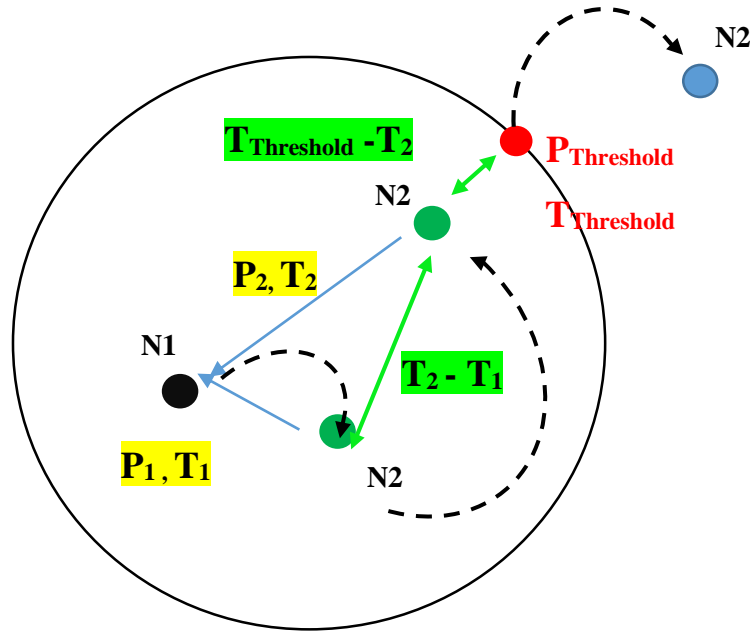


Figure 3.1 Proposed Method

3.3.1 Modified RREP packet format

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Type									R	A	Reserved									Prefix Sz				Hop Count							
Destination IP Address																															
Destination Sequence Number																															
Originator IP Address																															
Lifetime																															
MY RSSI (Received Signal Strength Indicator)																															
GetTime																															

Figure 3.2 Modified RREP packet format of AODV protocol

A node may offer connectivity information by broadcasting local Hello messages [1], it should only use hello messages if it is part of an active route. Every HELLO_INTERVAL milliseconds [6], the node checks whether it has sent a broadcast message within the last HELLO_INTERVAL. A node may determine connectivity by listening for packets from its set of neighbors. If does not receive any packets (Hello messages or otherwise), it should assume that the link to this neighbor is currently lost. When this happens, the node should sends a RERR message [6].

Hello packet format has the same format of the RREP message (Figure 3.1), so the same fields of this last, it is possible to modify the existing packet format by adding additional fields:

- The first additional field named “MY RSSI” (Received Signal Strength Indicator) added for getting the received power which taken from the wireless physique layer.
- The second field is “GetTime”, is the current time when the hello packet received, it’s also added from wireless physique layer.

Algorithm : Link Failure Prediction

```

▷ For each neighbor
▷ when Hello packet received
▷ Get P received signal strength "RSS" & T received time.
Input: list of neighbors & their received time & power = Null,  $P_{\text{threshold}} = 3.65 \times 10^{-10}$ ,  $\text{fixed\_T}_{\text{th}} = 0.4\text{s}$ 
Output: a Fresh path is found in advance, before link will break

1: procedure lk_fail_predict( $P_2$ ,  $T_2$ , Nb_addr) ▷ where  $P_2$  - power,  $T_2$  - time, Nb_addr neighbor@
2:  $n \leftarrow \text{try\_to\_find}(\text{Nb\_addr})$ 
3: if n is NULL then
4:   insert(Nb_addr,  $P_2$ ,  $T_2$ )
5: else
6:   if  $P_1 > P_2$  then
7:      $T_{\text{threshold}} = ((T_2 - T_1) * (P_{\text{threshold}} - P_2) / (P_2 - P_1)) + T_1$ 
8:     if  $T_{\text{threshold}} - T_2 < \text{fixed\_T}_{\text{th}}$ 
9:       deleteFromRt(Nb_addr)
10:    else
11:       $P_1 = P_2$ ,  $T_1 = T_2$ 
12:    else
13:       $P_1 = P_2$ ,  $T_1 = T_2$ 
14: end procedure

15: procedure deleteFromRt(address_BK)
16: Rt  $\leftarrow$  routing table head
17: while Rt  $\neq$  NULL do
18:   if nexthop@ = address_BK then
19:     ▷ Puts the route under repair
20:     SendRequest2(RtAddress_dst, address_BK) ▷ start discovering a route much earlier
21:     break
22:   else
23:     Rt  $\leftarrow$  the next
24: end while
25: end procedure

```

Continued
Algorithm : Link Failure Prediction
25: procedure SendRequest2(RtAddress_dst, address_BK) <ul style="list-style-type: none"> ▷ discovering another route to the destination address "RtAddress_dst" ▷ filling up the added field " Destination IP Address of NodeWillBk " by the address of the node which will break 26: end procedure
27: procedure try_to_find(Nb_addr) <ul style="list-style-type: none"> ▷ search in the "neighbors list & their received time & power" the Nb_addr ▷ if it exist, return it else return null 28: end procedure
29: procedure insert(Nb_addr, P ₂ , T ₂) <ul style="list-style-type: none"> ▷ Fill up the "neighbors list & their received time & power" by address of neighbor "Nb_addr", received power P₂, and received power instant T₂. 30: end procedure
<ul style="list-style-type: none"> ▷ For each neighbor ▷ when RREQ (Request) packet received
30: procedure recvRequest(Packet *p) <ul style="list-style-type: none"> ▷ Get address of neighbor which will break, and compare it with the index address "source address" ▷ If it's equal to the source address then packet drop. 31: end procedure

As shown in algorithm above, the implementation of our work includes six procedures:

1. lk_fail_predict (P₂, T₂, Nb_addr)
2. deleteFromRt (address_BK)
3. SendRequest2 (RtAddress_dst, address_BK)
4. recvRequest (Packet *p)
5. try_to_find (Nb_addr)
6. insert (Nb_addr, P₂, T₂)

First one, `lk_fail_predict (P2, T2, Nb_addr)` this is the main function which initiate the process of link failure prediction, it's based on three functions, so like we talked in algorithm above for each neighbor, when Hello packet received (`recvHello (Packet *p)`) from the Nb_addr, it got the received power value P₂, and the instant T₂ of received this last, and called the function `lk_fail_predict (P2, T2, Nb_addr)`, to start the prediction.

When the `lk_fail_predict (P2, T2, Nb_addr)` is called, the prediction process start from this function, the main operations made are:

1. Calling `try_to_find (Nb_addr)` function
2. Calling `insert (Nb_addr, P2, T2)` function if needed
3. Calculating T_{threshold}
4. Calculating time changing rate (T_{threshold} – T₂)
5. Calling `deleteFromRt (address_BK)` function

Second main function `deleteFromRt (address_BK)`, it's Responsible on routing management and routing table in our method. This function works by avoiding the address of the next hop from routing table, this address marked as an address of the node which will break, so the function compare the address of the node which will break with next hop address in routing table, if they are equals to each other, the function `SendRequest2 (RtAddress_dst, address_BK)` will call.

`SendRequest2 (RtAddress_dst, address_BK)` function, is the same function defined for standard AODV protocol, the only difference is that `SendRequest2` filling up the added field in RREQ packet format by the destination address of the node will break "address_BK", `SendRequest2` allows to discover another route to the destination RtAddress_dst such as the next hop of this last is predicted like a node will break. The aim of `SendRequest2` function is discover a route to the destination without the node which will break participating or make a part of an active route.

When a RREQ message is received, the receiving node gets the address of the node will break, from the field of the RREQ packet, then it comparing with his address, and if it has the same address, it will drop the received packet.

For `try_to_find (Nb_addr)`, function is just for search in neighboring list and return a pointer to this node if found or return Null. For `insert (Nb_addr, P2, T2)` if after the search of neighbor node by the previous function and the pointer pointes to null, a node will insert in neighboring list with its received power and received instant for this power.

3.4 Conclusion

The major issue created due to nodes mobility in the Ad hoc network is Link Failure, this causes less routing performance in several metrics, the standard Ad hoc on Demand Distance Vector (AODV) routing protocol doesn't provide a method for predicting a link breakage before a path will fail, it can only repair the route after it broken which decreases the routing performance. For this purpose, in this chapter we have proposed a link failure prediction algorithm added to AODV protocol, this method can increase the routing performance, so the next chapter proves that there is an improvement after the simulation results.