IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 10, Issue 3 (Mar. - Apr. 2013), PP 71-77
www.iosrjournals.org

Cross-layer based performance optimization for different mobility and traffic scenarios in adhoc networks

Pankaj Kumar[1], Ajeet Kumar[2] , Ashwini Singh[3], Raj Kumar[4],Dr. Ritesh Kumar Mishra[5]
[1]*(ECE, National Institute of Technology Patna, India)*
[2]*(ECE, National Institute of Technology Patna, India)*
[3]*(ECE, National Institute of Technology Patna, India)*
[4]*(ECE, National Institute of Technology Patna, India)*
[5]*(Asst. Prof. ECE, National Institute of Technology Patna, India)*

Abstract : *The self configuring dexterous autonomy of MANET imposes some network challenges constrained to traditional dynamic routing behaviour. So as working with different mobility and traffic patterns with normal management schemes may lead some minor pitfalls to some important network performance parameters and hence can degrade the whole network performance. Here, Our aim is to make some DSR and MAC based cross layer optimizations and testify it on different mobility and traffic scenarios so as to justify the robustness of our proposed improvement.*
Keywords *– cross- layer,MANET,MAC,optimization*

I. INTRODUCTION

In wired networks the infrastructure of the network is fixed hence the routes are being selected according to some predetermined tables within some fixed routing devices such as routers ,3-layer switches etc. But in case of wireless Ad-hoc networks, the nodes themselves comprises the network ,and they do not need any fixed infrastructure. Therefore each nodes executes routing functionalities, such as forwarding network traffic. So before we design, we should consider different features such as the use of MAC protocol, routing protocol, transport layer protocol, quality of service etc.For proper working the different protocols in wireless ad-hoc networks must handle different issues, such as noise of the network, routing information, transmission ranges, etc. Sometimes in one node, only part of the information collected by one protocol is delivered to Another protocol and a misinterpretation among these protocols may happen. To deal with this, we propose a modification in the MAC 802.11 protocol to avoid launching unnecessary operations in the DSR (Dynamic Source Routing) protocol., achieving better performance in the network, i.e. less routing overhead, less routing changes, less collision of packets, less route errors, less MAC errors, and more throughput. Concretely, DSR protocol launches route error when a neighbouring node is still near, because it understands the information received from the MAC layer as a broken link. Usually, the interferences among radio ranges of nodes could lead to this misunderstanding. The proposed approach tracks the signal strength of each node, informing the routing layer that the node has enough signal strength, skipping the route error launched by DSR..With these things in mind our goal is:

- To detect the presence of neighbour nodes in MAC 802.11 for each nodes within its reachable radio transmission. This can be achieved by tracking the signal strength of neighbouring nodes.
- To inform the higher layer, routing protocol DSR in this case, when a transmission was not successful, and if a neighbouring node is in the transmission range of each other.
- To adapt routing protocol DSR using this information, detecting error links and avoiding unnecessary route maintenance processes.
- To compare the achieved results with previous values, performing simulations in several (static and mobility) scenarios, and for different type of traffic.

II. RELATED WORKS

The Dynamic Source Routing protocol (DSR) is an efficient routing protocol designed for use in multi-hop wireless ad hoc networks of mobile nodes. DSR allows the network to be completely self-organizing and self-configuring, without the need for any existing network infrastructure or administration. DSR has been implemented by numerous groups, and deployed on several test beds. This protocol was used as base for other protocols as well. Some implementations of the DSR are:

- The Monarch Project implementation [1]: It is a set of kernel patches that supports FreeBSD 3.3 and 2.2.7. It is a pre-alpha release and is available because of educational purposes and for researchers.

- The Microsoft Research Mesh Connectivity Layer (MCL) MCL [2]: It implements a layer between link layer and network layer multi-hop routing protocol on Windows XP. This protocol is derived from the DSR protocol and is called Link Quality Source Routing (LQSR) protocol that means that DSR was widely modified.
- The Click DSR Router Project at the PecoLab at UC Boulder [3].. A user level and open source implementation of DSR implemented in accordance with the IETF draft specifications of DSR. It is ready on Linux and a 802.11g Wi-Fi card.
- The National Institute of Standards and Technology (NIST) [4] developed a simulation model for the DSR for MANETs based on the Internet Draft version 4 with the purpose of using it at OPNET in their communications system simulation software.
- Alex Tzu-Yu Song [5] has implemented DSR according to the fifth draft of The IETF DSR

III. DSR PROTOCOL

DSR is works on demand, without any periodic updates. It's characterized by the use of source routing. That is, the sender knows the complete hop-by-hop route to the destination. These routes are stored in a route cache.[6] The protocol is composed of route discovery and route Maintenance. The DSR agent checks every data packet for source-route information [7]. It forwards the
packet in accordance with the routing information. If the DSR agent does not find routing information in the packet, one of two options can be followed. If the route is known, it provides the source route. Otherwise, if the route is unknown, it caches the packet and sends out route queries. Route queries messages are broadcasted to all neighbours whenever there is no route to reach the destination. Then, route replies messages are sent back either by the intermediate nodes, if they have such path in their cache, or by the destination node. The source files for implementation of DSR protocol in ns-2 belongs to the ns-2/dsr directory. More details can be found in tcl/mobility/dsr.tcl

3.1 DSR FUNCTION BEHAVIOUR

The DSR strategy for ns-2 network simulator was ported from the CMU/Monarch's code and
it is as follows [8]:

- It is only worth discovering bidirectional routes, since all data paths must be bidirectional to work properly for 802.11 ACKs.
- Reply to all route requests in destination nodes, but reply to them by reversing the route and unicasting. Then, do not trigger a route request. By reversing the discovered route for the route reply, only routes that are bidirectional will make it back the original requestor.
- Once a packet goes into the sendbuffer, it cannot be piggybacked on a route request. The code assumes that the only reason that removes packets from the send buffer is the StickPktIn routine, or the route reply arrives routine.
-

IV. CHALLANGES AND CONCEPT

A bad interaction between DSR protocols and the MAC layer could decrease the throughput in wireless 802.11 networks [9]. Once the MAC layer has been improved with a table of received powers of neighbouring nodes with the purpose tracking their distances to the sender node, our goal in this thesis is to handle that information in the routing layer to avoid that DSR triggers a route error process or route maintenance if it is not required. DSR protocol triggers a route error when the receiver node of the communication did not reply after several attempts of RTS in the MAC layer. Consequently, DSR assumes 'link error', manages that link error as broken link and triggers the route maintenance process .However, as we know, unsuccessful communication among nodes may arise because of different reasons than broken links. In such case, the route maintenance process is not necessary when a neighbouring node is still reachable. A cross layer design should identify when a link error was due to broken link (node not reachable), triggering the route maintenance process only in such case, or other reasons such as increased contention

1.1 MAC 802.11 RELATED ISSUES

The main reasons for lost packets are high levels of congestion (high traffic), equipment failures (power problems) or errors due to noise or low signal (mobility). It would be beneficial for DSR the ability to distinguish these reasons, adjusting transmission rate in case of congestion, or deleting old routes in case of mobility.

Normal DSR interprets a link failure (in MAC layer)as a broken link, even when it was caused by congestion at receiver. The sender node should know why communication was not possible. here, we implemented an approach that tracks the received signal strength of each neighbouring node in order to know when a neighbouring node is near enough for a successful transmission. If lost packets are due to congestion and high traffic, normal DSR triggers route error but this is counterproductive because it adds more. If lost packets is due to low signal quality or misrouted packets, then route error is needed because receiver is not reachable. Then, the signal strength of neighbouring nodes can be used to detect the reason for lost packets, distinguishing between congestion and broken links due to mobility, because in broken links due to mobility, the receiver is not reachable and its signal strength is now available.

The implementation is divided into two parts; the first one keeps the last twenty received signals from a node in an array, and the second part decides the kind of message (link failure, either due to errors or due to congestion using signal strength of neighbouring nodes) to be sent to the upper layer, whenever the communication is not possible but the destination node is in the transmission range of the sender.

In the graphic below, the received power of neighbouring nodes is tracked with the purpose of using it later for distinguishing if neighbouring nodes are reachable or not.

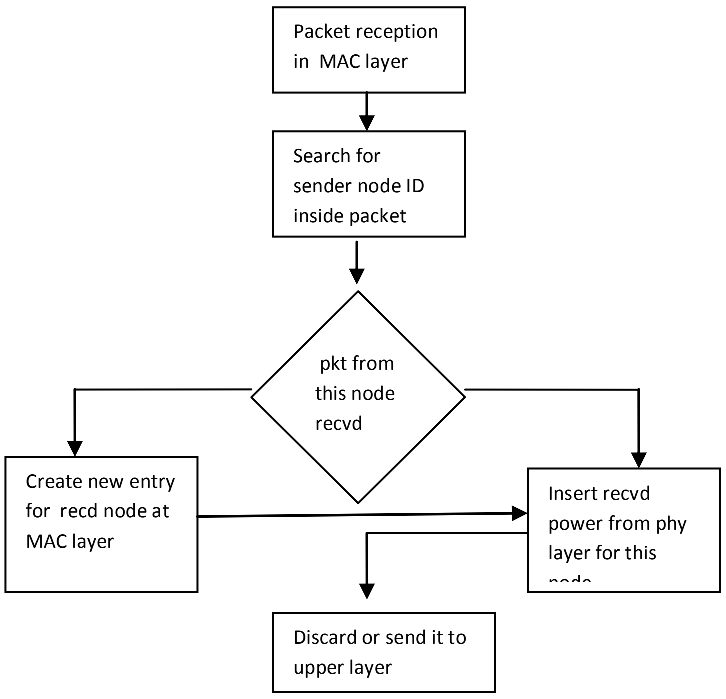
Packet reception in MAC layer
Search for sender node ID inside packet
pkt from this node recvd
Create new entry for recd node at MAC layer
Insert recvd power from phy layer for this node
Discard or send it to upper layer

Fig.1. MAC layer at receiver node. Packet is received even when the current node is not the destination node

These modifications are made at the sender as:

```
void
      Mac802_11::recv(Packet *p, Handler *h)
    {

       . . .
          if(tx_active_ && hdr->error() == 0) {
             hdr->error() = 1;
              }
              hdr_mac802_11 *mh = HDR_MAC802_11(p);
              u_int32_t idNode = ETHER_ADDR(mh->dh_ta);
              double power = p->txinfo_.RxPr;
              insertNode (idNode,power,nodesPower);

              }
         . . . .
              struct time_power {
              int pos;
              double power[20];
```

```
                                        u_int32_t idNode; }
                                    };
```

The proposed approach for future work that we designed adds new functionalities for every node in the network. Since each neighbouring node is tracked via its received signal strength, a sender node is able to discern if neighbouring nodes are moving away or not. In addition, we propose to calculate and use the average signals of nodes with the purpose of stop retransmitting packets when destination is not reachable because it moved away.

The reason of using an average value, instead of only the last received value from neighbouring nodes, is to adapt this approach to more realistic scenarios, where objects such as furniture, may interfere temporally in communication among mobile nodes. Modification suggested on MAC layer:

```
. . .
if (average_selected){
  u_int32_t idNode = ETHER_ADDR(rf->rf_ra);
  int pos =findNode (idNode, nodesPower);
  if (pos!=-1){
      float av=0; int movAw=0;
      average(nodesPower[pos],av,movAw);
      if ( (movAw >= 18) && (av < RxThreshold) ) {
          discard(pktRTS_,   DROP_MAC_RETRY_COUNT_EXCEEDED); pktRTS_ = 0;
           hdr_cmn *ch = HDR_CMN(pktTx_);
           if (ch->xmit_failure_) {
               ch->size() -= phymib_.getHdrLen11();
               ch->xmit_reason_ = XMIT_REASON_RTS;
                ch->xmit_failure_(pktTx_->copy(),ch-    >xmit_failure_data_); }
               discard(pktTx_,  DROP_MAC_RETRY_COUNT_EXCEEDED);
pktTx_ = 0; ssrc_ = 0;rst_cw();
               return;}
        }
}
. . .
```

1.2 DSR RELATED ISSUES

When a node tries to communicate to a neighbouring node and this communication is not successful (after several attempts, MAC layer sends an error to the routing layer), the normal behaviour of DSR is to interpret that the neighbouring node is not present anymore, that is, DSR assumes that the communication failure was due to mobility. Once the DSR protocol at sender interprets that link error as a broken link,which information is delivered for each intermediate node back to the sender node, which receives that information as well. A route error may trigger a route request process (processBrokenRouteError or handlePacketWithoutSR), if the sender does not have an alternative route towards the destination, or there is an alternative route towards the destination, which will be used. In both cases, the route containing the broken link is removed from the caches of the nodes involved in the route error**.**

In a scenario without mobility communication failures may arise, but DSR will interpret that it was due to mobility, when actually it was due to congestion. Therefore, the process of
route error should not be performed since it increases even more the congestion, decreasing
the overall performance in the network. The proposed modifications will make DSR able to distinguish between both situations, avoiding the route error process when the link error at MAC layer was due to congestion and not due to mobility of nodes causing broken links. In the proposed approach, when MAC layer is not able to communicate to a neighbouring node, MAC layer informs to the routing layer not only that there was a problem, it is also included if the neighbouring node is still reachable. Therefore, DSR at the sender realizes about the real situation and do not perform a route error if the error received from MAC layer informs that the node is still reachable.

The DSR modifications can be appended as:

```
void DSRAgent::xmitFailed(Packet *pkt, const char* reason)

{
(...)
if(cmh>xmit_reason_==XMIT_REASON_HIGH_POWER
&& !strcmp(reason, "DROP_RTR_MAC_CALLBACK"))      {
Packet::free(pkt);
pkt = 0;
```

```
    return;
    }
    (…)
    /* send out the Route Error message */
    sendOutPacketWithRoute(p, true);
}
```

V. SIMULATION AND RESULTS

Extensive simulations were carried out to compare the NEW-DSR routing protocol proposed in this paper with the conventional DSR. Network Simulator (NS-2) [10] is used to simulate these protocols

5.1 SIMULATION PERFORMANCE PARAMETER

a)Throughput: It can be calculated as
Throughput = Received _bytes/Time_of_simulation
b)Routing Overhead:It comes as
Routing_Overhead=number_routing_bytes/number_(routing+data)bytes
c)Lost Packets

5.2 TRAFFIC SOURCE

In order to perform our simulation, a scenario defining position and number of nodes, physical interfaces, routing protocol and a transport protocol (TCP/UDP) are needed. Besides this, it is also needed to attach some traffic to such transport protocol.

5.3 SIMULATION SCENARIOS

In our simulations, we utilized different scenarios, with the purpose of evaluating diverse behaviours of mobile ad hoc networks. First, we distinguish between static and mobile scenarios. We used static scenarios to analyse and to show communication in the network, avoiding changes of routes or disconnections due to mobility of nodes. Later, we use mobile scenarios to verify the adaptability of our approach in mobility and high mobility scenarios.

5.3.1 STATIC SCENARIO

Chain scenario:

The chain scenario defines 22 nodes without mobility forming a chain with a separation of 200 meters. The carrier sense threshold for each node is 550 meters and the transmission range is set to 250 meters. The simulation results were obtained from two, four and eight simultaneous flows for TCP . The number of hops to reach the destination varies among 4, 5, 6, 7, 8, 10, 12, 14, 16 and 20 hops. The source is always the same, where the destination depends on the number of hops to simulate. For instance, four TCP flows and 7 or 12 hops means that there were simulated four TCP flows from node 0 (source) to node 7 or node 12 respectively (see figures 2, 3, 4). This scenario was designed to demonstrate the improvements in TCP when the fractional window [11] is used. Simulation time for this was set to 120 seconds.

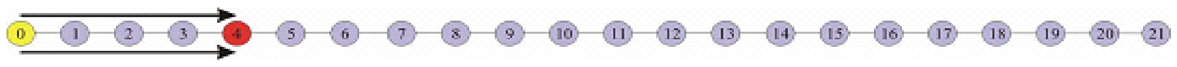

Figure.2. Two flows from node 0 to 4, example for 4 hops

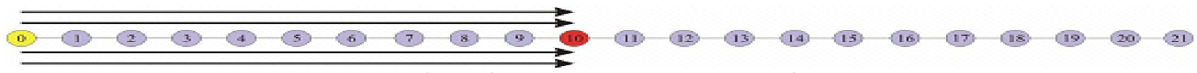

Figure.3.Four flows from node 0 to 10, example for 10 hops

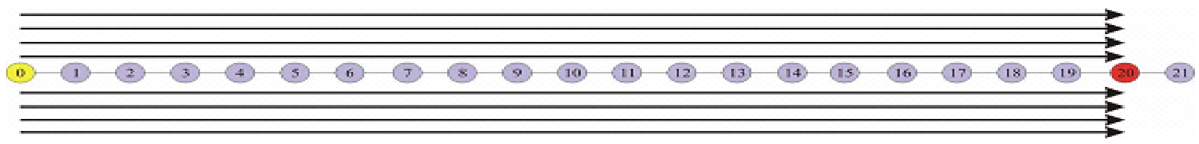

Figure.4. Eight flows from node 0 to 20 example for 20 hops

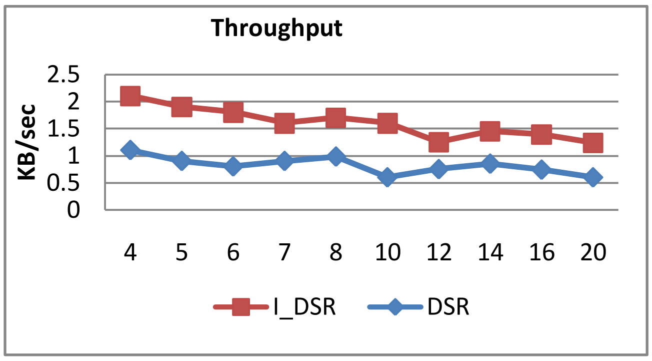

Fig.5. Throughput vs no of hops graph

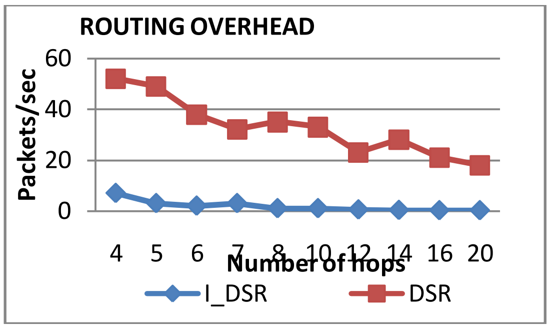

Fig.6. Routing Overhead vs no of hops graph

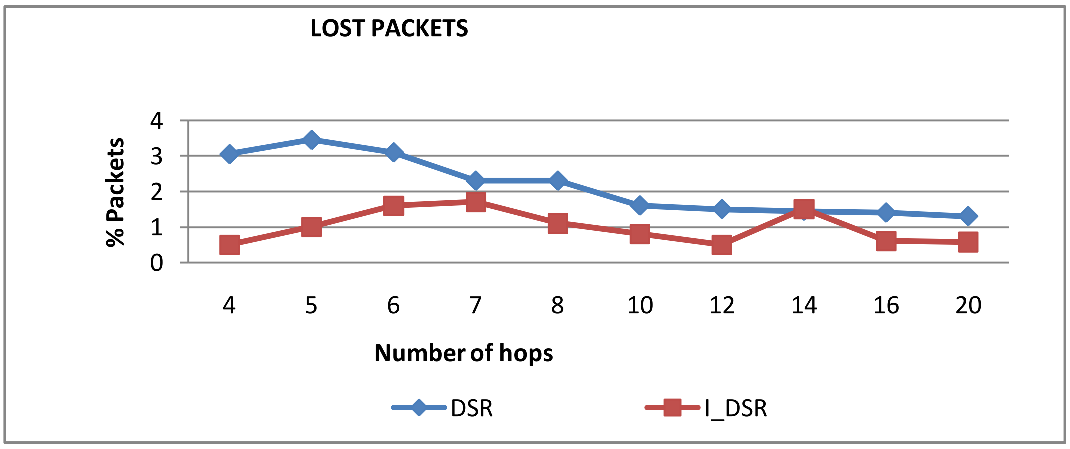

FIG.7. THROUGHPUT VS NO OF HOPS GRAPH

5.3.2 MOBILITY SCENARIO

Random Waypoint scenario:

Random way point scenario was created using the "scengen" tool provided in the ns-2.35 package. The scengen tool creates n nodes and places them randomly. The nodes are moving during the simulation with random speeds, from 1m/sto 10m/s. In mobility scenarios there is a special parameter called pause time, which means that a node stops for a while after a movement. In this scenario, the pause time value was set to five seconds, used mainly for simulation purposes. The size of this scenario is set to 2000x2000 meters, therefore it is quite probable the existence of, at least, one route from a sender to a receiver node. The carrier sense threshold is 550m and the transmission range is 250m for each node.Simulation time for this scenario was set to 120s.

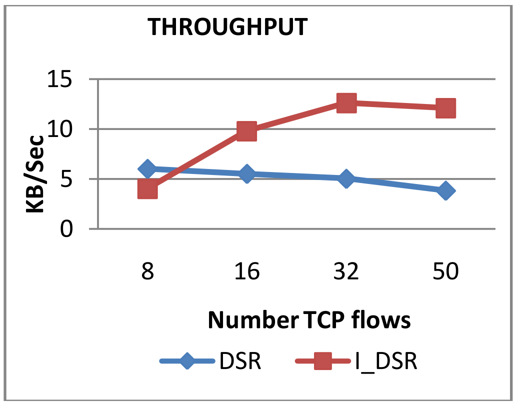

Fig.8. Throughput vs no. of TCP flows graph

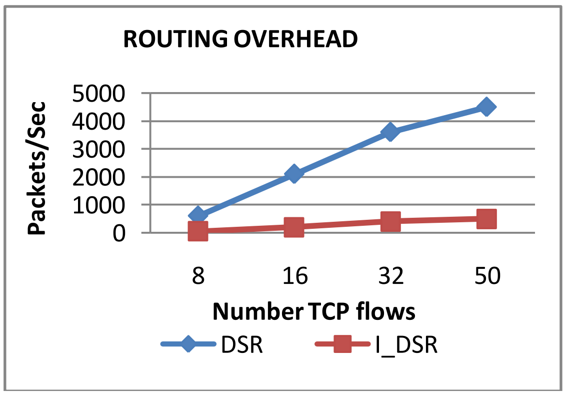

Fig.9. Routing overhead vs no. of TCP flows graph

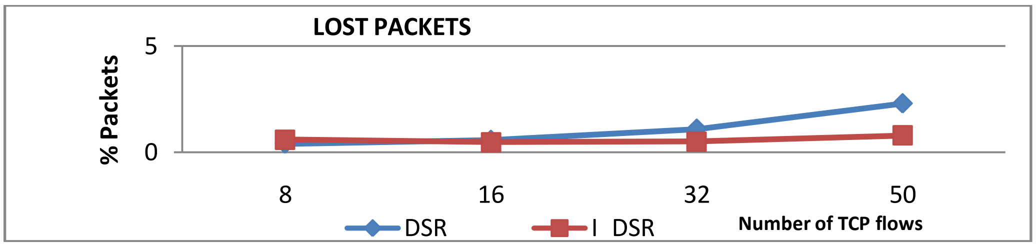

Figure.10. Lost packets vs no. of TCP flows graph

In TCP random waypoint scenario, MAC errors are increasing using normal DSR , because of the mobility of nodes (from a minimum of 0m/s to a maximum of 10m/s). This mobility, as well, causes a high number of route errors and therefore more routing overhead and packet loss. Moreover, the throughput decreases meanwhile the number of flows increases.

VI. CONCLUSION & FUTURE WORK

In the situation of congestion, the MAC 802.11 protocol does not perform well enough using the RTS/CTS/DATA/ACK dialog because RTS packets from a sender to a receiver may collide and communication fails, leading to bad interaction with the routing layer. The MAC layer could mistakenly inform the routing layer about broken links when the communication among nodes is still possible, when, however, communication failed due to the collisions of RTS packets caused by congestion. Thus, the routing protocol interprets broken links in the MAC layer as route errors, triggering the route maintenance process, therefore increasing the overhead in the network. Our main purpose is to avoid this misinterpretation by determining the cause of the broken links. To deal with this, the signal strength of each node is tracked. This information is used to notify the routing protocol if the node is still reachable but the communication was dropped. Then, at routing layer it is possible to distinguish if the different speeds of scenarios .Moreover, it is important to decide on the value of the average which determines if the protocol continues trying to retransmit or not, and on the number of times that this node was moving away. the route exists and do not trigger a route error process, avoiding routing packets which would increase more the congestion in the network.

we have developed a new feature with the objective of avoiding disconnections in scenarios with objects that interfere with the communication among nodes. The main idea makes sense in mobility scenarios, for example an office with mobile nodes. Here the connection can be interrupted easily due to furniture obstructing the communication. If the receiver node is not moving away from the transmitter node, it is possible to avoid disconnection by making an average of the received signals. This new approach makes a prediction based on the last twenty movements and interprets if the receiver node is not reachable anymore. We should be careful choosing what method of average (arithmetic mean, geometric mean).

ACKNOWLEDGMENT

This work is supported by National Institute of Technology, Patna as a part of partial fulfilment of post graduate degree in communication system.

REFERENCES

[1] The Monarch Project implementation. http://www.monarch.cs.rice.edu/dsr-impl.html
[2] The Microsoft Research Mesh Connectivity Layer. http://research.microsoft.com/mesh
[3] The Click DSR Router Project. http://pecolab.colorado.edu/DSR.html (2002) The IEEE website. /
[4] OPNET, http://w3.antd.nist.gov/wctg/prd_dsrfiles.html
[5] O Piconet II mobile router, implementing an ad hoc routing protocol. http://piconet.sourceforge.net/thesis/main.html
[6] Akintola, A.& Aderounmu, A. & Owojori, A. & Adigun,M.O.(2006) Performance modelling of UDP over IP-Based wireline and Wireless Networks.
[7] The network simulator. http://www.isi.edu/nsnam/ns/
[8] Bryan's ns2 DSR faq.http://www.geocities.com/b_j_hogan/
[9] Nahm, K, & Helmy, A. & Kuo, J. TCP over multihop 802.11 networks: Issues and performance enhancement.
[10] Dube, R. & Rasi, C & Wang, K. & Tripathi, K. (1997) Signal stability-based adaptive routing for ad hoc mobile networks.
[11] Nahm, K & Helmy, A. & Jay Kuo, C. Improving Stability and Performance of Multihop 802.11 Networks.