layer and the physical layer, and *PdP* denotes the processing delay at physical layer.

In our works we use delayed information from $ILCd_{NM}$, *OvMACLd*, and propagation for building routing metric. Therefore, node processing time able to consider as

$$NPT = ILCd_{NM} + OvMACLd + ILCd_{MP} + Pd_P \quad (2)$$

The MAC transmission delay depends on the backoff mechanism of the MAC protocol, and the average number of retransmissions required for a packet to be successfully transmitted. If i is the average number of retransmissions required for a successful packet transmission, then the amount of time spent by the MAC layer in transmitting this packet over the channel (*OvMACLd*) is given as [5], [6]

$$OvMACLd = \sum_{j=0}^{i-1}\left(DIFS + \frac{CW_j - 1}{2} * slot + RTS + \mu + SIFS\right) + DIFS + \frac{CW_j - 1}{2} * slot + RTS + CTS + DATA + ACK + 4 * \mu + 3 * SIFS \quad (3)$$

where *RTS* is the time spent in the RTS packet transmission, *CTS* is the time spent in the CTS packet transmission, DATA is the time spent in the data transmission, *ACK* is the time spent in the acknowledgement transmission, *SIFS* is the short inter-frame space, and μ is the propagation delay.

Assume that the source and destination are separated by an average of *n* hops we have

$$TotMACLd = n*OvMACLd \quad (4)$$

$$Tot\mu = n*\mu Avg \quad (5)$$

The average time spent for a packet transmission depends on the average number of retransmissions required for successfully transmitting a packet. Therefore, if i +1is the average number of attempts for the successful transmission of a packet, then the time spent by the MAC layer for transmitting a single RREQ packet (RREQ1*Pkt*) in the four-way handshake transmission is given by (3), where the DATA represents an RREQ packet. If the average number of route discovery processes required to be initiated to obtain a route to the destination is $n\text{RREQ}s_{avg}$, then the total time spent in the route discovery processes (RREQT_{tot}) is given by

$$\text{RREQT}_{tot} = (\text{RREQ1}Pkt + \text{RREP}_{out}) * n\text{RREQ}s_{avg} \quad (6)$$

where RREP_{out} is the average RREP time-out period at the network layer. We have an estimative end-to-end delay formula as

$$\text{EEdRDT} = \text{RREQT}_{tot} + \text{TotMACLd} + \text{Tot}\mu \quad (7)$$

The end-to-end delay information was assessed by formula (7). We use this information as route matric, and define set "*close neighbor*" of each node in the network. Node j is the neighbor of node i, and node j become *close neighbor* of node i if only if $\text{EEdRDT}_{ij} \leq \text{EEdRDT}i_{avr}$ where EEdRDTij is end-to-end delay time between node i and node j, $\text{EEdRDT}i_{avr}$ is the average end-to-end time delay between node i and it's neighbors. Therefore, we have set of *close neighbor* of node i (CNBN_i) as

$$\text{CNBN}_i = \{\text{node j} \mid (\text{node j is the neighbor of node i}) \,\&\, (\text{EEdRDT}_{ij} \leq \text{EEdRDT}i_{avr})\} \quad (8)$$

3.2 Route discovery strategy

Route discovery strategy is divided two phases. The first phase is the process to look for a destination node in the neighbor table. If Destination node is in neighbor table then the Source node sends RREQ to Destination node by using unicast, and Destination node replies RREP to Source node also by using unicast. After that the link between Source node and Destination node is setting up, the route discovery process is finished. If Destination node is not in neighbor table then the route discovery process changes to second phase. In the second phase of route discovery, the Source node sends address of Destination node and its *far* nodes to *close* nodes by using multicast. The Destination node address is for next node to continue discovery process to find Destination node. Its *far* node address is for restricting routing overhead in next steps. In next step of route discovery process, each node in the set of close neighbor node continues performing the route discovery process from the first phase. It finished when setting up route to the Destination node.

Parameters used in routing discovery process algorithm at one node are:

OVH_NBR: Number of overhead neighbor,

OVH_CLS_NBR: Number of overhead close neighbor,

RREQ_NUM: Number of allowable RREQ packets,

Node_i_address: IP address of node i.

```
IF node_i receives a requirement to connect to node_j
THEN {
FOR (i=0; OVH_NBR -1; i++)
{IF neighbor_address == node_j_address THEN {
Send (Addr, RREQ); exit}
ELSE {
```

```
FOR (j=0; OVH_CLS_NBR-1; j++)
{Send (Addr, RREQ); RREQ_Number++}
} } }
```

Algorithm.Route Discovery

where Send(Addr, RREQ) is a function to send RREQ message to next node, Addr is address of Destination nodes. If Addr is only one address, it uses for unicast. If Addr includes all of close neighbor, it uses for multicast. If Addr includes all of neighbor, it uses for broadcast. But in our proposed routing protocol not uses broadcast method. When Destination node received the RREQ, it can use Reply(RREP) function. Reply is a function to send RREP message back to prior node of current node. If prior node is intermediate node, it continues used Reply(RREP) to send RREP message to its prior node. This operation repeat until RREP reached Source node.

3.3 Protocol operation

For built the neighbor table, this protocol operation is the same as proactive routing protocol. One node uses "Hello" packet to maintain the relationship with neighbor nodes. If there is any a new node in it's the propagative range, this node is using "Hello" packet to update information of the new node. Therefore, neighbor table of each node in network always was updated instantaneously. The protocol operations were illustrated in figure 1.

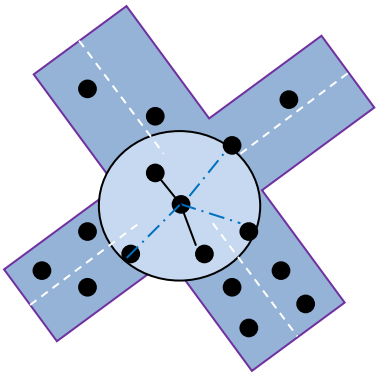

Figure 1: One node uses "hello" packet for updating and maintaining information of neighbor nodes.

The rest of the protocol is similar to reactive routing protocol, but there is one modification. When Source node receives a requirement to send message to Destination node, first steps it look for Destination node address in neighbor table. If Destination node is in neighbor table, it uses send (Addr, RREQ) send RREQ to Destination node, and Destination node uses reply (RREP) reply RREP to Source node and discovery process is finished. If Destination node is not in neighbor table, Source node sends RREQ to *close* neighbor node. This operation restricts the routing overhead the same as OLSR [7]. Furthermore, it is a condition will be sure the setting up route will be rapid convergence and reliable.

4 SIMULATION RESULTS

In this part we show preliminary results comparing between our proposed routing protocol EEDAHRP (end-to-en delay assessment and hybrid routing protocol) and one popular protocol AODV. We use the developed module in ns-2.34 [8] with support of Monarch [9] to simulate EEDAHRP, and AODV in the same condition of performance. The network was selected with the number of nodes varying from 50 to 200. Nodes are randomly distributed over 1000m x 1000m area. Send/receive data packets between Source node and Destination node use the IEEE 802.11g, uses IEEE 802.11 DCF for channel access, and data packet size is set to 512 bytes. Each routing protocol was deployed in the same scenario for run simulation.

Obviously, from figure 2 the time for route discovery is directly proportional to nodes. But our proposed routing protocol EEDAHRP was more slowly increase than AODV. The time for each step in route discovery can shrink by using end-to-end delay assessment for route matric and relaying destination node to 'close neighbors' mechanism. In figure 3, it is observed that end-to-end data transfer time inversely proportional to nodes. But we can see end-to-end data transfer time decreases very fast in EEDAHRP.

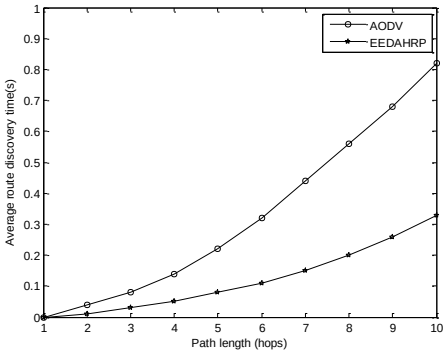

Figure 2: Average route discovery time.