International Journal of Information Technology and Knowledge Management
January-June 2011, Volume 4, No. 1, pp. 109-112

THR : A TWO-HOP LOOK AHEAD WITH PACKET BUFFERING PROTOCOL FOR MANET

Ela Kumar[1] & Ravinder Nath Rajotiya[2]

Wireless ad hoc network [8][9]consist of mobile nodes which rely on each other to transmit data over multi-hops by forwarding packets. In the process of data transmission route discovery and maintenance are the greatest challenges as the node may move unpredictably. A lot of work has been carried out and numbers of protocols have been designed to take care of the above problem. In this paper we propose a simple approach that by reducing the number of broadcasts and overhead guarantees the route availability to the destination. In our approach we use the technique in which the node keeps information about the two-hop distant nodes.
Keywords: Routing Protocol, Ad hoc Network, Data Buffer

1. Introduction

An ad hoc network consists of a group of wireless nodes, which cooperate with each other by forwarding packets to enable multi-hop communications. It requires no centralized or fixed network infrastructure. Ad hoc networks can be deployed in crisis applications such as in battlefield and also in civilian applications such as vehicular systems. Proactive, reactive and hybrid routing protocol have already been developed by the IETF. One or the other protocol does not suit to all situations. In this paper we try to suggest a modified design for manet routing protocol. Here the nodes in addition to maintaining the record of their immediate neighbour also record information about the neighbour of the neighbour. That means a node will keep the information about the two hop distant node. This strategy in addition to reducing the amount of memory required to store the information will ensure reduced traffic overhead and fast route discovery and maintenance and stability.

The remainder of this paper is organized as follows.

In Section 2 we investigate the previous work and the literature survey. Section 3 briefly describes the working of the two-hop (THR) routing protocol. This section gives description of the algorithm for route discovery and the route maintenance procedures. Section 4 concludes the paper. Finally the list of reference journal / conference proceedings are provided.

2. Background and Related Work

Ad hoc networks have several features, including possible frequent transmissions of control packets due to mobility, the multi-hop forwarding of packets, and the multiple roles of nodes as routers, sources, and sinks of data[1][2][3][4], that may produce unique queuing dynamics. We believe that the choice of scheduling algorithm to determine which queued packet to process next may have a significant effect on overall end-to end performance when traffic load is high. This belief motivated us to evaluate several applicable scheduling algorithms.

The study of the OMH protocol[5] has motivated us to come up with idea of the THR routing protocol, which not only will solve the problem of the selfish behaviour of the nodes but will also increase the throughput and performance in data transfer and route discovery and route maintenance. In addition the concept of data buffering further improve the performance of the routing protocol.

3. THR Protocol

3.1. Overview of THR

THR is a table driven routing protocol. Instead of maintaining the information about each and every node as in proactive routing protocol, THR only maintains the information about the immediate neighbor and his neighbor i.e information about the neighbor's neighbor. Thus this our approach will discover the destination route in less number of broadcasts and also consume less power and also the size of the table maintained at each node will also get reduced.

The basic principle of operation of the THR protocol is as follows:

When a node wish to transmit data packet to some destination node, it first checks its own routing table which contains route to two hop distant node if the route is found then the packet is transmitted to the destination node else a route discovery is initiated by the source node avoiding the

[1]Deptt. School of ICT, Goutam Budhha Univ. Greater Noida

[2]Advanced Institute of Technology & Management, Palwal, Haryana

Email: [2]ravinder.rajotiya@gmail.com

routing loops as discussed in many reactive routing protocol such as DSR, AODV etc.

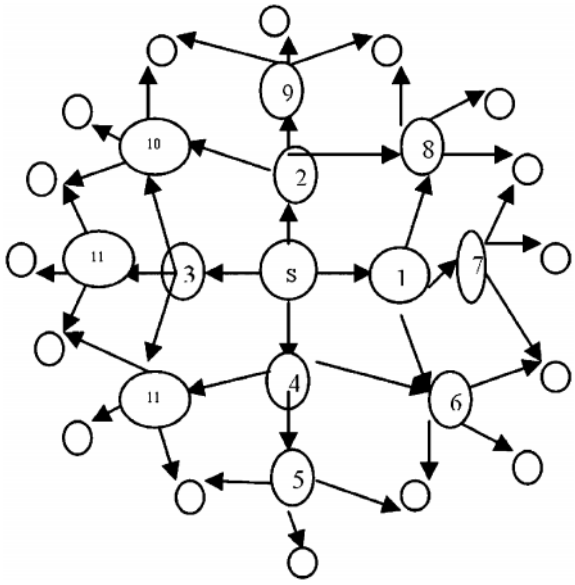

Fig. 3.1: A Typical MANET Topolgy

In this Section we introduce our assumption about the network model, then introduce the motivation of THR, and at last introduce the design of THR.

3.2. Route Discovery

In our proposed routing protocol, a routing path is constructed by the combined proactive and reactive process in that short beacons are transmitted by every node to know about their immediate and the next node. Unlike the pure proactive protocol, the n^{th} node in our approach gets info about $n+1^{st}$ and $n+2^{nd}$ in the visibility range. The information so obtained is stored by the nodes in their internal table known as LRT(local route table). When a source node (S) need to transmit data to some destination node (D) the node S will first check a route in its LRT.

If a route to destination exists, if so the data packet is transmitted to destination, on the other hand, a path (the main route) from source node S to destination node D need to be constructed before source node S can start the data transmission. The process of finding such a routing path is called the main route construction, which begins with the source node S sending a main route request (MRREQ) to all its neighbors. Every host that receives the MRREQ acts exactly the same as the source node does. MRREQ is thus flooded over the network, and would eventually arrive at node D - 2. Node D - 2 will send a route reply (RREP) to the source in h-2 hop counts, where h represents the hop count from source to the destination. Every node that receives the RREP will also keep a record of the main route to the destination node D, thus keep the details of the most recent and newer route that a node has seen.

The formation of the local routing table is built by using short interval Hello packets. The hello packet is processed as follows:

If (node_interface_addr = = main address)

Then

Discard the hello packet to avoid routing loop

Else

If (this node has already contains info about RREQ initiator in its List of recently seen RREQ)

Discard the packet

Else

Save the main address as the neighbor address

Reply this node address to source

Update its own local route table(LRT)

End-if

Source Node	Intermediate One-Hop Node	Two-Hop Node	Hop-Count
S	2	3	2
2	3	5	2
2	S	-	1
3	2	S	2
3	4	5	2
4	3	2	2
4	5	6	2
5	4	3	2
5	6	D	2
D	6	5	2

The route discovery process also keeps track of the sequence number and the packet ID to avoid any routing loops in the local as well as main route discovery process. The complete route discovery process is described as follows:

1. If the pair initiator address, request id for this route request is found in this host's list of recently seen requests, then discard the route request packet and do not process it further (This avoids routing loop).
2. Otherwise, if this host's address is already listed in the route record in the request, then discard the route request packet and do not process it further.
3. Otherwise, if the target of the request matches the host's address in the LRT(because the LRT contain information of two-hop reachable nodes), then the route record in the packet contains the route by which the request reached this host from the initiator of the route request. Return a copy of this route in a route reply packet to the initiator.

4. Otherwise, append this host's own address to the route record in the route request packet, and re-broadcast the request.

The above facts can be described by the algorithm given below:

broadcast(dest_addr.IP, RREQ)

Check if

This node already seen the RREQ_source_address && request_id

Then

Discard the packet

Else

RREQ_entry already contain DestHost address

Then

Discard the packet

This node LRT contain entry of DestHost addr.

Then

Generate RREP

Intimate to destination o RREQ by source

Else

Update the RREQ packet with this node addr.

Broadcast(dest_addr.IP, RREQ)

End

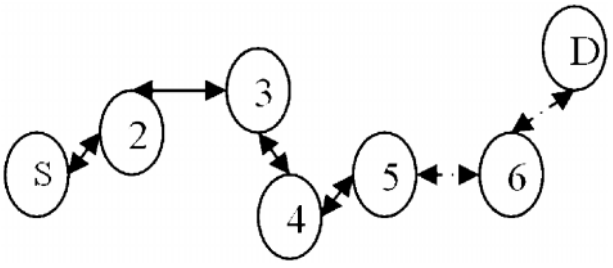

Fig. 3.2: Route-Discovery THR

The above algorithm can be explained as follows:

Suppose node 3 wish to transmit to node 'D', it first searches its LRT. Since route to node D is not available in the LRT, a RREQ is generated by the source node-3 and broadcasted to its one-hop neighbors.

Node-4 and node-2 hear the RREQ broadcast and searches their LRT, node-4 has an entry for node2 and node-6 as the reachable nodes; node- 2 on the otherhand has reachability to node-S.

So node-4 broadcast the RREQ and node-5 listens to this request.

Node-5 has a route to destination node-D and thus will send a RREP containing the list of intermediate node through which the data packet may be sent to destination node-D.

3.3. Route Maintenance

Instead, while a route is in use, the route maintenance procedure monitors the operation of the route and informs the sender of any routing errors. The route maintenance and thus the route error reporting can be performed at different levels:

1. Utilize a hop-by-hop acknowledgement at the data link level in order to provide early detection and retransmission of lost or corrupted packets.
2. If wireless network does not support a low-level acknowledgement, then node may use of the positive acknowledgement by overhearing the next node transmission.
3. Existing transport or application level replies or acknowledgements from the original destination could also be used as an acknowledgement that the route (or that hop of the route) is still working.
4. As a last resort, a bit in the packet header could be included to allow a host transmitting a packet to request an explicit acknowledgement from the next-hop receiver. If no other acknowledgement signal has been received in some time from the next hop on some route, the host could use this bit to inexpensively probe the status of this hop on the route.

In our approach, when a route error is detected, the node at that level will push the data packet in that node's buffer and will initiate a fresh route discovery. After a fresh route discovery, the buffered packet will be transmitted along the new path.

3.4. Buffer Design

In normal conditions a node should forward the packet towards the destination node. But, since the topology is dynamic, a node might have changed its position, and thus leading to a broken route to destination. Under this condition we propose to buffer the packet in the node. Now issue that needs to be resolved is the constraint of the buffer size. Looking at the advances in the technology, where the amount of storage per unit of are has increased to hundred times, the use of large storage capacity can definitely be explored.

Assuming the travel rate of a bit to destination at the speed of light i.e. 3x10^8 m/sec., and the distance to next node as 50 meters so a bit will take 1.66667E-07 seconds to reach the destination. Now if the route breaks and a fresh

route discovery is needed, then it take will take "N x 1.66667E-07 x M" seconds to discover the route to destination, here N is the number of hops to destination and M is the RREQ packet size. For a typical case if N=10, and RREQ = 1KB, then it takes 10 * 2 * 1.66667E-07 * 1024 = 0.003413 seconds or say 3.4 milli-seconds to discover the route.

3.4.1. Buffer Size and the Overflow Condition

Buffer overflows occurs because of an overflow condition i.e. full buffer and a new request. So we need to calculate the time when an overflow condition can occur. From the above equation, the time required to transfer one KB of data packet to a node(ignoring the time required in internal buffer state change) equals to 0.001365 seconds, so if we take the buffer size as 128 KB, the time required for buffer full condition will be 0.001365 * 128 ≡ 0.175 seconds. This is the upper limit if the route can not be established in this time or if the buffered packets can not be transferred to other nodes or places a buffer overflow condition will arise.

So there is a need to schedule the buffer, there have been many approaches to buffer scheduling[4], but we suggest a simple buffer scheduling the packets. For best performances we encourage to buffer to half its capacity, i.e to 128/2 = 64KB thus is takes only 0.087 seconds to full this size, any packet that now comes to this node will not be buffered here, but the previous hop will be intimated of the buffer full condition and thus asking the previous hop node to buffer the other incoming packets to the destination through this node. Likewise the previous hop node will ask the previous hop to buffer the packets. In mean time the route to destination will be discovered and the transmission resumed giving priority for the control packets over the data packets i.e. whenever a node see the incoming pckets as control packets it will first give transmission chance to it than the data packets.

4. CONCLUSION

The concept present in this paper will not only improve the route discovery, route maintenance process with minimum route breakages but also improve the performance by using the buffering technique. In addition the concept presented by [5] of improving the selfish behaviour[7] will be retained and improved using this technique.

REFERENCES

[1] David B. Johnson, David A. Maltz, Yih-Chun Hu, and Jorjeta G. Jetcheva. "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks", IETF Internet draft, Mobile Ad-hoc Network Working Group, IETF, February 2002.

[2] Charles E. Perkins, Elizabeth M. Royer, and Samir Das, "Ad Hoc on Demand Distance Vector (AODV) Routing", IETF Internet Draft, Mobile Ad-hoc Network Working Group, IETF, January 2002.

[3] J. Broch, D.A. Maltz, D.B. Johnson, Y.C. Hu, and J. Jetcheva, "A Performance Comparison of Multi-hop Wireless Ad hoc Network Routing Protocols", In Proceedings of ACM/IEEE MOBICOM, Dallas, TX, October 1998.

[4] Samir R. Das, Charles E. Perkins, and Elizabeth M. Royer, "Performance Comparison of Two on Demand Routing Protocols for Ad hoc Networks", In Proceedings of the IEEE INFOCOM, Tel-Aviv, Israel, March 2000.

[5] Byung-Gon Chun and Mary Baker, "Evaluation of Packet Scheduling Algorithms in Mobile Ad Hoc Networks", Mobile Computing and Communications Review, 6, No. 3.

[6] H.L. Chen, "Two Hops Backup Routing Protocol in Mobile Ad hoc Networks", Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on Issue Date: 20-22 July 2005 Volume: 2 On page(s): 600 - 604 , 2, ISSN: 1521-9097.

[7] Ravinder Nath Rajotiya, Sanjeev Gupta, "Security in Mobile Computing Environment", Presented & Published in Proc. of National Seminar Held at KIET, Ghaziabad on 2nd Feb 2008.

[8] Ravinder Nath Rajotiya, "AARAP: An Efficient Proactive Routing Protocol", Presented & Published in Proc. of National Conference held at MAIMT, Jagadhari, 16-18 Nov 2007.

[9] Ravinder Nath Rajotiya and Ela Kumar, "Network Aware Routing Protocol for MANET", IEEE International Advance Computing Conference 2009' to be Held on March 6 -7, 2009.

[10] Chengqi Song, Qian Zhang, "Suppressing Selfish Behavior in Ad hoc Networks with One More Hop", Published Online, Springer Science+Business Media, LLC 2009: 5 February 2009.