CN-DSR: THE CHEATING-NODE DYNAMIC SOURCE ROUTING PROTOCOL FOR MULTI-HOP WIRELESS AD HOC NETWORKS

Irshad Ahmad Khan
Srinagar, Kashmir
irshukhan@rediffmail.com

ABSTRACT

Dynamic Source Routing (DSR) protocol is one of the best routing protocols developed for wireless ad hoc networks. But at higher mobility rates of the network nodes, its performance degrades rapidly. Since wireless ad hoc network nodes always have a tendency to move, a routing protocol for such a mobile ad hoc network must have a robust mechanism to deal with rapid mobility of the network nodes so as to maintain its performance even at higher mobility rates of the network nodes. In this paper, I present a cheating-node dynamic source routing (CN-DSR) protocol for multi-hop wireless ad hoc networks that addresses the issue of rapid mobility of the network nodes. CN-DSR senses the signal strength of the network nodes to determine mobility of the nodes, and as such keeps alternate routes ready to replace nodes moving out of range. The signal sensing is done using a clever technique and does not increase any overhead. The end result is a refined protocol that is adaptive to node mobility.

INTRODUCTION

A wireless mobile ad hoc network is composed of independent mobile wireless nodes operating in the absence of network infrastructure and cooperating with each other to enable service delivery within the network. The ability of the network nodes to join, leave and move about at any point of time makes efficient routing within such a network a challenging task. A lot of research has already been done in this direction and many routing protocols have been developed. The Dynamic Source Routing (DSR) is an on-demand routing protocol for multi-hop wireless ad hoc networks (Johnson, Maltz, and Broch, 2001). DSR is an efficient and highly adaptive protocol but its performance degrades at higher rates of node mobility. The packet delivery ratio decreases and the end-to-end delay increases at higher rates of node mobility, resulting in an overall reduction in the efficiency of DSR.

Cheating-Node Dynamic Source Routing (CN-DSR) is mobility oriented version of the original DSR protocol and is designed to provide better performance and quicker route discovery than the DSR protocol in case of networks involving a lot of mobility. In the absence of any mobility within the network, there will be no difference between DSR and CN-DSR. CN-DSR employs the inherent route discovery and route maintenance mechanisms of the DSR protocol (Johnson, Maltz, and Broch, 2001), but it uses a totally different approach to handle node mobility. During an active transmission, the nodes keep sensing the signal strength of their neighbouring nodes on the active route while data packets are being

received. If the signal strength of a node, as received by its neighbouring node down the active route, is found to drop below a panic threshold level, the neighbouring node sends a request to the node to initiate the process of alternate route search. This process of initiating an alternate route search from an intermediate node to the destination node on an active route, while there still exists a valid and active route to the destination node, will be referred to as "Cheating", the node initiating such a request will be referred to as the "Cheating Node", the route thus obtained from the Cheating Node to the destination will be referred to as the "Cheat Route", and the node involved in the active route but moving away such that it would break the route will be referred to as the "moving node" in this paper. The minimum threshold level to "panic" is kept at a suitable level, say 35% of the signal strength, so that the cheat process is started while the active route is still in order. This means that when the signal strength sensed by a moving node reaching it from its immediate neighbouring node, up the route, drops below the panic threshold level, the moving node will request the neighbouring node to initiate the process of finding a Cheat Route from itself to the destination in anticipation of the probable route break due to the moving node, and thus the neighbouring node will become a Cheating Node. The Cheating Node will continue to forward the data packets of the active session to the moving node, since the moving node would be still in its transmission range, but will keep an alternate route ready. The Cheating Node will not switch to the alternate route before the signal strength reaches the "cheat" threshold level. The cheat threshold level would be kept at a suitable level, say 15% of the signal strength. As soon as the signal strength of the Cheating Node, as received by the moving node, drops below the "cheat" threshold level, the moving node sends a Request to Cheat (RTC) to the Cheating Node. The Cheating Node switches to the alternate route and sends a route update message to the original source node. In this way, CN-DSR provides an uninterrupted routing mechanism at higher rates of node mobility.

CN-DSR PROTOCOL DESCRIPTION

CN-DSR Fundamentals

CN-DSR is based on the DSR protocol. As such, CN-DSR uses the same route discovery and route maintenance mechanisms proposed for the DSR protocol (Johnson, Maltz, and Broch, 2001). However, CN-DSR has a signal strength sensing mechanism that is part of the route maintenance mechanism while a route is active. It also has another route discovery mechanism specifically used to discover a route from a Cheating Node to the destination node, referred to as the Cheat Route Discovery Process (CRDP) in this paper. In the absence of mobility of the nodes, CN-DSR behaves just like DSR.

Node Mobility Sensing Mechanism

While data packets are being forwarded from one node to another on an active route, each node checks the signal strength of either its immediate predecessor or immediate successor node depending on the technique implemented. Node mobility sensing can be achieved using two simple techniques that do not increase any overhead.

In the first technique, when an intermediate node receives a packet, it senses the signal strength of the transmitting node. If it finds the signal strength of the transmitting node below the panic threshold level, it sends a Request to Panic (RTP) to the transmitting node. For example, as shown in Figure 1, D is moving away from C and E. Since the route is active, D will sense its own mobility by sensing the signal strength received from C while it is receiving packets from it. No additional packets or beacons are required for sensing mobility. D does not take any action till the signal level received from C falls below the panic threshold level. When it falls below the panic threshold level, D sends a RTP to C.

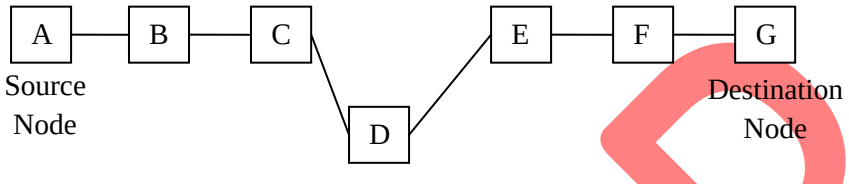


Figure 1: First technique of sensing node mobility. Node D is moving away from its neighbours C and E. Node D senses this situation by checking the signal strength of the node C while packets are being received from it. As D tends to move further away, the signal strength of C as sensed by D tends to keep dropping.

The second technique uses the promiscuous mode. C overhears the packets forwarded by D to E and thus senses the signal strength of D as received by it. In this case, if the signal level of D as received by C falls below the panic threshold level C raises a panic flag for itself, otherwise no action is taken.

In both the techniques a panic signal is generated. Either of the techniques can be used to sense node mobility within an active route. I will take the first technique into account for the rest of this paper.

Cheat Route Discovery and Selection

When a node receives a RTP, it starts the Cheat Route Discovery Process (CRDP). CRDP begins with the node sending a Cheat Route Request (CRREQ) to all its neighbours. The basic difference between a normal Route Request (RREQ) (Johnson, Maltz, and Broch, 2001) and a CRREQ is that while RREQ is flooded by all receiving nodes to all of their neighbours, CRREQ is ignored by the immediate neighbours of the Cheating Node that are on the active route as illustrated in Figure 2.

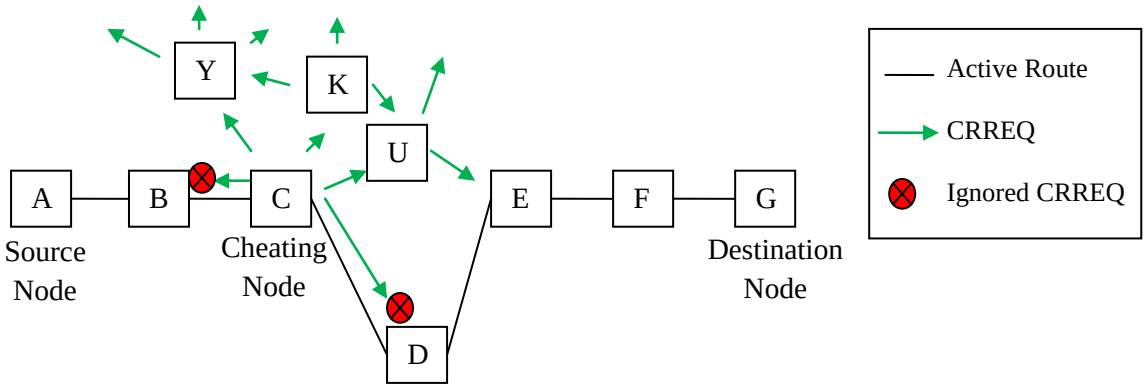


Figure 2: CRREQ is flooded by C to its neighbours. B and D simply ignore the CRREQ since they are the immediate neighbours of C and are on the active route.

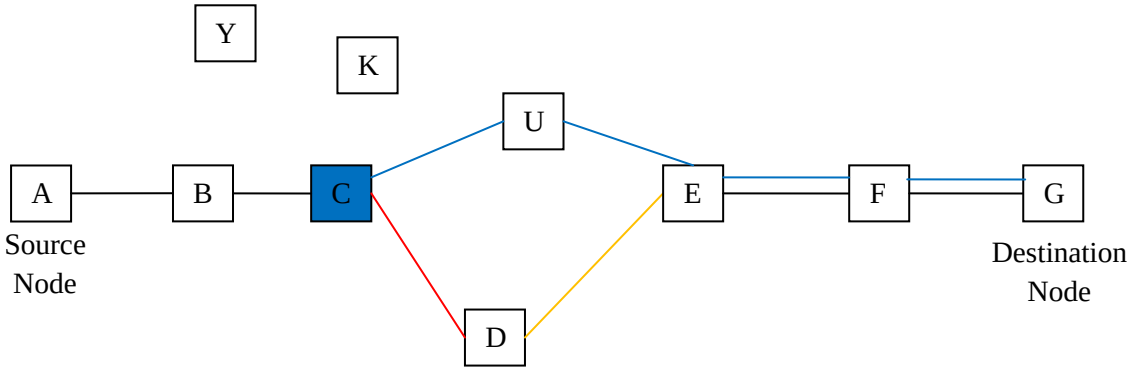
C floods the CRREQ to all its neighbouring nodes but since B and D are already a part of the active route, they will ignore this request. E is also on the active route but it is not an immediate neighbour of C. Therefore, E will process the CRREQ. This helps to ensure that a valid cheat route is generated.

Another difference between the RREQ and CRREQ is that when a node receives a CRREQ, it senses the signal strength of the transmitting node and adds it to the Total Signal Strength (TOSS) field of CRREQ. TOSS is initially set to zero by the cheating node. As the CRREQ traverses through the network each node on the respective link keeps on adding the signal strength of the previous node. The destination node sends the consolidated value of TOSS in the Cheat Route Reply (CRREPLY) along with the number of hops and complete route from the cheating node to the destination node. Since multiple routes may be present from the cheating node to the destination node, the cheating node may receive more than one CRREPLY. The cheating node calculates the Average Signal Strength (ASIST) for each CRREPLY by dividing TOSS by the number of hops from the cheating node to the destination node. The cheating node then selects a Cheat Route based on the number of hops and the ASIST. The route having maximum ASIST and minimum hops is selected.

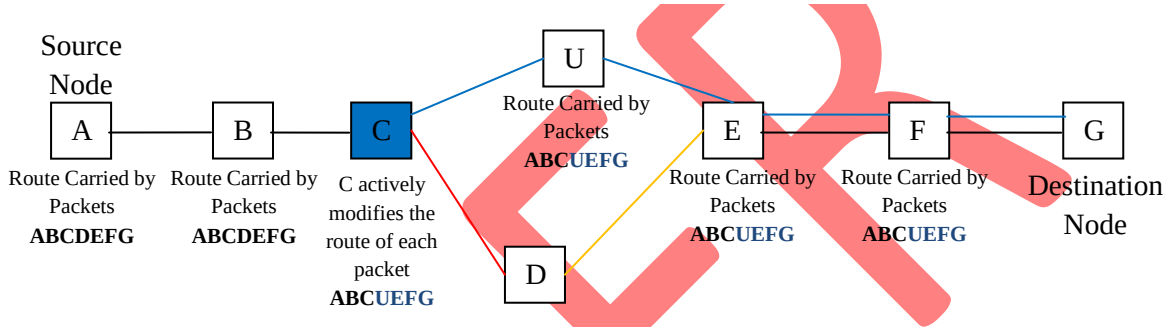
Cheating Process Initiation

Once a Cheat Route is selected, the Cheating Node is ready to switch over to the Cheat Route. But the switch over to the Cheat Route is not made by the Cheating Node till a Request to Cheat (RTC) is not received from the moving node. As soon as the RTC is received, the Cheating Node switches over to the Cheat Route, sets a Cheat Flag (CF) locally so that the node itself knows that it is cheating, and sends a Route Update Request (RUR) to the original source node. Since the RUR may take some time to reach the source node and the source node may already have forwarded many packets without route updates, the Cheating Node plays an active part till RUR reaches the source node and the source node carries out the route updation. The Cheating Node modifies the route carried by each data packet and partially replaces the old route with the Cheat Route as depicted in Figure 3. This process is continued till the CF is set and the route update has not been carried out by the source node.

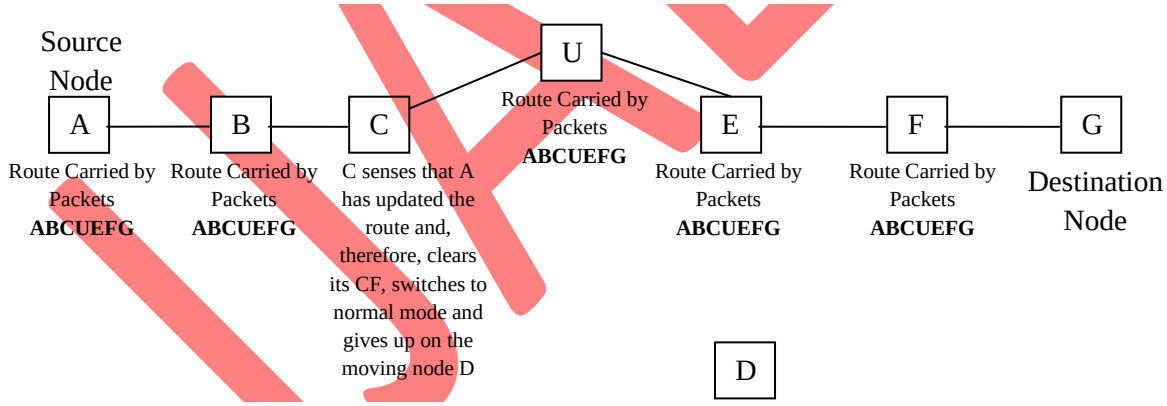
The Cheating Node detects the route updation by analysing the data packets being received from the source node. When the first packet with the updated route is received by the Cheating Node from the source node, it clears the CF and the Cheat Route becomes a part of the original route. The Cheating Node afterwards functions just like other nodes on the active route. In this way, the Cheating Node once again becomes a normal node and the Cheat Route becomes a part of the normal route.



(a): Cheat Route has been found from C to G (C→U→E→F→G). C has set the CF and should now act as a Cheating Node.



(b): Source node has not updated the route (A to G) yet. C is a Cheating Node and is actively modifying each data packet with A as source and G as destination.



(c): Source node has received the RUR and hence updated the route (A to G). C notices the update and stops cheating. C clears the CF and gives up on D.

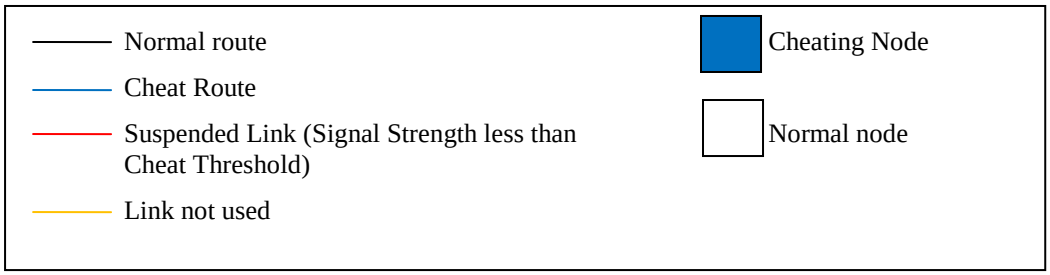


Figure 3: Cheat Route replacing the faulty route.

Issues and their Resolutions

1 Multiple RTPs

If we closely analyse Figure 1, we can realise that if D is moving away both from D and E, both D and E begin to sense this. As a result, D send a RTP to C and E sends a RTP to D. This will result in unnecessary multiple Cheating Nodes. To avoid this, a node takes action on receiving RTP if and only if it has not recently sent an RTP to another node up on the active route. So if D has sent RTP to C and D receives RTP from E, it will simply ignore the RTP from E. Similarly, if D has not yet sent a RTP to C but has received one from E, it will quit the Cheating process as soon as it sends a RTP to C.

2 Multiple RURs

Consider the illustration given in Figure 4. Since C and F are moving out of range, the situation gives rise to multiple Cheating Nodes, B and E. Consequently, both B and E may

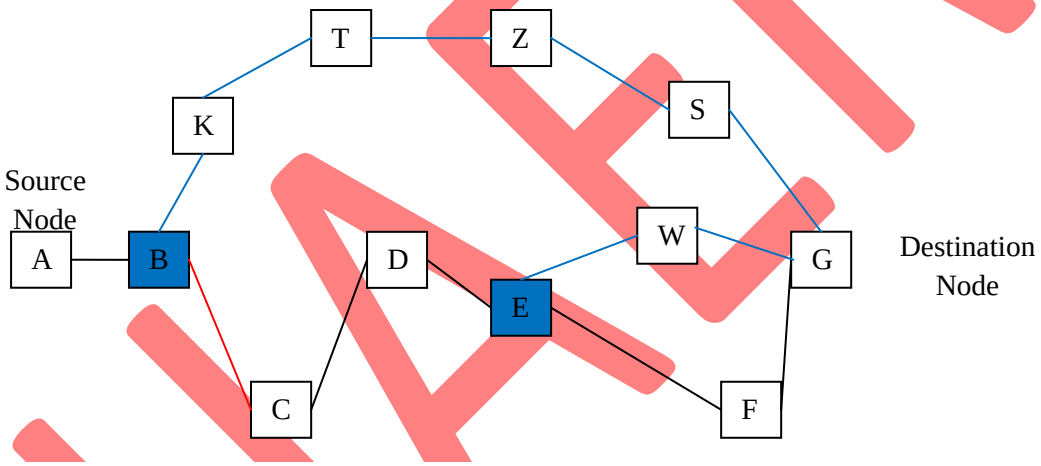


Figure 4: Multiple Route Update Requests (RURs) being generated, one by B and the other by E.

generate RURs. But CN-DSR automatically avoids this situation. Whenever the source node receives RUR from a Cheating Node, it is only processed if the sending Cheating Node is a part of the currently active route; otherwise the RUR is ignored by the source node. For example as in Figure 4, when A receives the RUR from B, it is processed as B is part of the active route (A→B→C→D→E→F→G). Consequently, A updates the route (A→B→K→T→Z→S→G). Now, if A receives the RUR from E, A will ignore it as E is not part of the currently active route (A→B→K→T→Z→S→G). As a result, no problems arise out of multiple Route Update Requests.

EVALUATION OF CN-DSR

Due to non availability of sufficient resources, I am unable to carry out simulation studies involving the implementation of CN-DSR and validate my claims. As such, I am leaving the evaluation of CN-DSR open to the community. I am always open to constructive criticism and suggestions for further improvement.

REFERENCES

Abolhasan M, Wysocki T, Dutkiewicz E. A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks* 2004; 21–22.

David B. Johnson, David A. Maltz, and Josh Broch. The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks. *Ad Hoc Networking* 2001, p 139–172.

Royer EM, Toh C-K. A review of current routing protocols for adhoc mobile wireless networks. *IEEE Personal Communications*, 1999; 4(2):46–55